Gjorgji Jovançevski
Silvija Nikollovska

BAZAT

E

PROGRAMIMIT

TEKST PËR VITIN II
DREJTIMI ELEKTROTEKNIK

Shkup, 2023

Gjorgji Jovançevski
Silvija Nikollovska

BAZAT E PROGRAMIMIT

TEKST PËR VITIN II

DREJTIMI ELEKTROTEKNIK

Shkup, 2023

BAZAT E PROGRAMIMIT

TEKST PËR VITIN II
DREJTIMI ELEKTROTEKNIK

Autorë:
Gjorgji Jovançevski
Silvija Nikollovska

Recensentë:
Dejan Spasov
Biljana Pejovska
Viktorija Dokaza

Titulli i origjinal:
ОСНОВИ НА ПРОГРАМИРАЊЕ
УЧЕБНИК ЗА II ГОДИНА ЕЛЕКТРОТЕХНИЧКА СТРУКА
Ѓорѓи Јованчевски
Силвија Николовска

Përkthyer nga gjuha maqedonase: Muzafer Beqiri

Redaktor profesional i botimit në gjuhën shqipe: Xhevair Beqiri

Lektor: Refail Sulejmani

REDAKTOR:
Refail Sulejmani

REDAKTIMI GRAFIK DHE TEKNIK:
Magdalena Avramovska, Evgenija Pavllova - ARS STUDIO

Botues: MINISTRIA E ARSIMIT DHE SHKENCËS EREPUBLIKËS SË MAQEDONISË SË VERIUT

Rr. "Shën Kirili dhe Metodi" nr. 54, 1000 Shkup

Vendi dhe viti i botimit: Shkup, 2023

Me vendimin nr 26-548/1 prej dates 19.08.2022 të komisionit nacional për tekste, lejohet përdorimi i këtij

CIP:

PËRMAJTJA

Hyrje

Ky tekst paraqet kurs fillestar për mësimin e programimit. Ai është konceptuar t'i përfshin bazat e të shprehurit algoritmik të problemeve, nëpërmjet pseudo-gjuhës në maqedonisht, si edhe kodimin e algoritmeve në gjuhën programore C++.

Teksti është përafruar me Programin arsimor për lëndën „Bazat e programimit", për vitin II drejtimi elektroteknik. Ai është shkruar sipas programit modular për këtë lëndë, qëpërbëhet prej tre njësive modulare.

Në njësinë e parë modulare „Hyrje në gjuhën programore C++", nxënësi njihet me algoritmet dhe të shprehurit e tyre tekstuale dhe grafike, me gjuhët programore dhe rrethinat për programim. Më tutje, njihet me elementet themelore të gjuhës C++, sikurse: ndryshore, lloj i të dhënave, urdhëra për lexim dhe shtypje dhe struktura e programit te gjuha C++.

Te njësia modulare e dytë „Kontrolli i rrjedhjes dhe funksioneve në C++", janë sqaruar strukturat kontrolluese algoritmike për kontrollin e rrjedhjes së realizimit të algoritmeve: strukturat kontrolluese për zgjedhje dhe strukturat kontrolluese për përsëritje. Ato janë të mjaftueshme për të shprehur çdo algoritëm dhe të kontrollohet rrjedhja e realizimit të tij. Implementimi i tyre është sqaruar dhe ilustruar nëpërmjet urdhërave kontrolluese përkatëse në gjuhën programore C++. Gjithashtu, në këtë njësi modulare, janë përpunuar nënalgoritmet, si edhe implementimi përkatës me funksione te gjuha C++. Janë përpunuar funksionet e bibliotekës në C++, por vëmendja më e madhe është përkushtimi të funksioneve përdoruese të përkufizuara.

Te njësia e tretë modulare „Llojet e përbëra të të dhënave në C++", nxënësi futet në shfrytëzimin e vargjeve numrike, të treguesve dhe të stringeve gjatë programimit në C++.

Mendojmë se materijali i parashtruar është i mjaftueshëm për çd nxënës fillestar në programim, por përvetësimi i tij nuk është i vështirë pasi tentuam ta parashtrojmë në mënyrë të lehtë dhe të kapshëm, me shumë shembuj, ushtrime dhe detyra të zgjidhura. Pas çdo njësia mësimre, janë parashtruar pyetje për kontroll të njohurive dhe detyra për zgjidhje të pavaur.

Autorët

Vërejtje: *Në bashkëpunim me lektorin, autorët u përpoqën ta shprehim në gjuhë maqedone disa terma informatike, por edhe terma prej gjuhës programore C++. Ato, ndoshta, do të tingëllojnë jo të zakonshme për disa lexues, por mendojmë se është koha e fundit të punohet në problemin e përdorimit të termave informatike në gjuhën maqedone.*

Konkretisht, për shkak të specifikave të kësaj gjuhe programore, por edhe të gjuhës që përdoret në informatikë te ne, në përgjithësi, terme të caktuara vetëm janë transkriptuar në gjuhën maqedone. Te teksti respektohet mendimi i autorëve se disa koncepte, për shkak të shumë domethënies së tyre, në këtë gjuhë programore duhet të shkruhen në formë e cila nuk korrespondon me rregullat për përdorimin e tyre në gjuhën maqedonase.

MODULI

HYRJE NË GJUHËN PROGRAMORE C++

1

Në këtë kapitull do të njiheni me këtë:

- Programi, programimi dhe gjuha programore.
- Softueri dhe ndarja e tij.
- Algoritmi, hapat algoritmike, algoritme të përgjithshme dhe detale.
- Paraqitja e algoritmeve.
- Programimi strukturor.
- Pseudo-gjuhë, bllok-diagrame, program burimor dhe programi i ekzekutushëm.
- Rrethina integruese zhvillimore për C++.
- Historia e C++.
- Elementet e gjuhës C++.
- Madhësitë, të dhënat, konstantat, ndryshoret dhe lloj i ndryshores.
- Lloj i të dhënës.
- Formatet për paraqitjen e numrave dhjetorë.
- Riemërtimi i llojit, caktimi automatik i llojit.
- Leximi dhe shtypja e të dhënave.

Fjalët kyçe

bool	Gjykimi
char	Fjalë kyçe
cin	Kodimi
Code::Blocks	Koment
cout	Programi përdorues
double	Literal
endl	Shprehje logjike
enum	Gjuhë logjike
enum	Operator logjik
float	Lloj logjik i të dhënës
int	Gjuha makinerike
iomanip	Gjuhë makinerike e pavarur
iostream	Programimi modular
istream	Lloj i numërueshëm i të dhënës
long	Dhëmbëzimi
long double	Pika e palëvizshme
long long	Lloj i pastrukturuar i të dhënës
Microsoft Visual Studio	Gjuhë objekti i orientuar
namespace	Operatori për dekreminim −−
ostream	Operatori për inkrementim ++
return	Operatori me bit
short	Operatori për futje (insertim)
string	Operatori për ndarje (ekstrahim)
Unified Modeling Languge – UML	Algoritmi i përgjithshëm
unsigned char	Lloj themelor i të dhënës
unsigned int	Lidhës
unsigned long	E dhëna
unsigned long long	Pika lëvizëse
unsigned short	Përkthyes (kompajler)
Caktimi automatik i llojit	Riemërtimi i llojit të të dhënës
Lloj i adresës të dhënës	Mbushje
Algoritmi	Gjuha e orientuar problemore
Struktura kontrolluese algoritmike	Programimi prej lart poshtë
Hapi algoritmik	Gjuha programore

Programi aplikativ	Ndryshorja
Aplikimi	Lloj i thjeshtë i të dhënës
Asembler	Gjuhë procedurale programore
Bllok-diagram	Pseudo-gjuhë
Funksionet e Bulit	Lloj real i të dhënës
Lloj i ndërtuar i të dhënës	Struktura kontrolluese rendore (sekuenca)
Madhësia	Fjala e rezervuar
Gjuha programore e lartë	Operatori i relacionit
Përrua hyrëse e të dhënave	Biblioteka standard e C++
Funksioni kryesor	Semantika e gjuhës programore
Gramatika e gjuhës programore	Gjuha simbolike
Gjuhë programore deklarative	Sintaksa e gjuhës programore
Ndryshorja deklarative	Programe sistemore
Algoritëm detal	Forma e shkurtër e operatorit (operatori i përbërë)
Përkufizimimi i ndryshores	Gjuhë skripte
Diabager	Lloj i përbërë i të dhënës
Konversioni i llojit eksplicit (hedhja e llojit)	Softuer
Shenja për vijë të re	Specifikator i llojit
Lloj i shenjës të të dhënës	Lidhja e stringeve
Identifikator	String
Programi burimor	Lloj i strukturuar i të dhënës
Programi i ekzekutueshëm	Programimi i strukturuar
Përrua dalëse e të dhënave	E dhëna tekstuale
Sekuenca dalëse (eskejp)	Redaktor tekstual editor
Konstanta e emërtuar	Lloj
Gjuha programore imperative	Funksioni
Konversioni implicit (lloj i detyrimit)	Gjuha funksionale
Inicializimi i ndryshores	Heder - datoteka
Integrimi i rrethinës zhvillimore	Lloj numër i plotë i të dhënës
Interpretaor	

1.1 Programet

Kompjuteri nuk është i aftë për asnjë punë të pavarur. Që të mund kompjuteri të realizojë çfarëdo pune, është e nevojshme t'i jepen urdhëresa përkatëse me të cilat aktivizohet dhe realizohet ndonjë **program** (angl. program). Nëse programi është drejt i shkruar, kompjuteri do ta realizojë dhe do të jep rezultatet përkatëse.

Puna e përgjithshme e kompjuterit realizohet nën kontrollin e veçantë të ashtuquajtur **programe sistemore** (angl. sistem programs), të cilët mund të ndahen në tre grupe:

- **Sisteme operative** (angl. operating sistems).
- **Programe kontrolluese** (angl. control programs).
- **Programe shërbyese** (angl. utility programs).

Programet të cilat realizojnë punë të caktuara për shfrytëzuesit quhen **programe aplikative** (angl. application programs) ose **programe përdoruese** (angl. user programs).

Të gjitha programet aplikative të cilat i kemi në kompjuterin tone zgjidhin detyra të caktuara, përkatësisht me ato realizojmë punë të caktuar. Për shembull, programe aplikative të njohura prej pakos Microsoft Office: MS Word, MS Excel, MS PowerPoint etj.

Të gjitha programet të cilët mund t'i realizojë kompjuteri, me një emër quhen **softuer** (angl. software). ***Te figura 1.1.1*** është dhënë një ndarje e mundshme e softuerit.

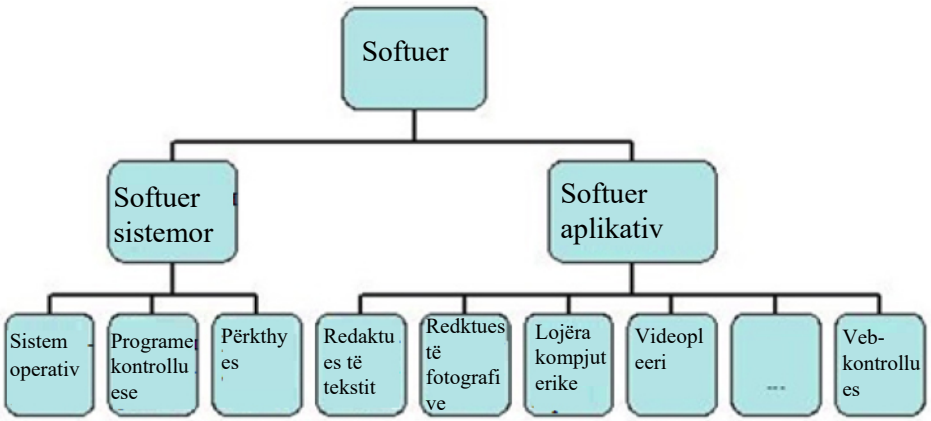


Figura 1.1.1

Programi[1] paraqet tekst të cilin kompjuteri mund ta kupton dhe realizon. Programet shkruhen në ndonjë **gjuhë programore** (angl. programming language), që shërben komunikimi ndërmjet njeriut dhe kompjuterit. Programet shkruhen me urdhëra që kompjuteri t'i „kuptojë" dhe, më tutje, t'i realizojë veprimet të cilat janë „treguar" me ato urdhëra.

Programet mund të redaktohet (shkruhen dhe ndryshojnë) me ndonjë redaktues të tekstit, të mbahen mend si datoteka dhe të realizohen.

Të shkruarit e programit është proces të quajtur **programim** (angl. programming), por njerëzit që i shkruajnë programet (programojnë) quhen **programues** (angl. programuess).

1.2 Algoritmet

Çdo njeri e di saktë çka duhet të bëjë kur duhet të ziejë kafe, të përgatitë ushqim të caktuar, të vozitë automobil, të instalojë program në kompjuter, ta përgatitë çantën për ditën e ardhshme etj.

Aktivitetet njerëzore, në pjesën më të madhe, mund të realizohen si renditje të numrit të caktuar të aktiviteteve të vogla (elementare), d.m.th., aktivitetet për të cilat nuk është i nevojshëm sqarim për të qenë të realizuara.

Për shembull, aktivitet elementare gjatë zierjes së kafes mund të jenë:

- Vendosja e ujit në xhezve.
- Futja kafe në xhezve.
- Vendosja e xhezves te shporeti.
- Kyçja e shporetit ku është venduar xhezvja.
- Pritja që kafja të vlon.
- Hekja e xhezves prej shpretit.
- Çkyçja e shpretit.

Që një realizues (sikurse ai që përgatit kafe) të mund të kryejë aktivitet të dhënë (zierja e kafes), ai duhet të dijë aktivitete elementare dhe mënyra (renditja) e realizimeve të tyre. Vargu i atillë prej aktiviteteve elementare quhet mënyra ose **algoritëm** (angl. algorithm).

Mund të themi se:

algoritmi është mënyrë prej numrit të fundshëm në mënyrë rigoroze të përkufizuar të aktiviteteve dhe saktë të dhënë renditja e realizimeve të tyre.

Konceptin algoritëm do ta sqarojmë më detalisht te shembulli për gjetjen e numrit më të madh prej tre numrave, me këtë mënyrë:

- Krahasimi të çfarëdo qoftë dy numra dhe caktimi i numrit më të madh të tyre.
- Krahasimi i numrit më të madh prej krahasimit të parë me numrin e tretë dhe caktimi i më të madhit prej tyre.

[1] Një program i shkruar në gjuhën programore C++ është dhënë te ***figura 1.3.4.***

Numri më i madh prej krahasimit të dytë është numri më i madh prej tre numrave.

Është e qartë se mënyra përbëhet prej vetëm dy aktivitete saktë të përkufizuara.

Për t'u zbatuar kjo mënyrë mbi çfarëdo tre numra, është e nevojshme ato të jenë të dhënë paraprakisht. Rezultati prej mënyrës është numri më i madh. Kjo do të thotë se zgjidhja e ndonjë detyre qëndron në caktimin e *rezultateve dalëse* me zbatimin e mënyrës së caktuar – algoritmi në bazë të *të dhënave hyrëse.*

Prandaj, mund të thuhet se:

algoritëm ***është mënyrë prej numrit të fundshëm në mënyrë rigoroze veprime të përkufizuara aktivitete, të zbatuara mbi të dhënat hyrëse sipas renditje rigoroze të shkruar, me të cilat arrihen rezultatet dalëse.***

Fjala *algoritëm* është marrë prej gjuhës latine dhe paraqet përkthim latin të mbiemrit Al–Khwarizmi të matematikanit arab të shekullit IX Abu Ja'far Muhammad ibn Musa al–Khwarizmi, i cili i pari i ka formuluar rregullat për realizimin e katër operacioneve me shifra arabe.

Hapat algoritmike

Veprimet prej të cilave përbëhet një algoritëm quhen **hapa algoritmike** (angl. algorithms steps). Varësisht prej asaj se hapat a janë më të përgjithshme ose më detale, algoritmi mund të jetë **i përgjithshëm** ose **detal**.

Për shembull. Algoritmi i përgjithshëm për detyrën prej pikës paraprake (caktimi i numrit më të madh prej tre numrave të dhënë), mund ta shkruajmë si te ***figura 1.2.1.***

hapi 1: Dhënia e tre numrave.
hapi 2: Krahasimi i çfarëdo dy numrave dhe gjetja e më të madhit prej tyre.
hapi 3: Krahasimi i numrit më të madh i gjetur në hapin 2 me numrin e tretë dhe gjetja e më të madhit prej tyre.
hapi 4: Shtypja e rezultatit.

Figura 1.2.1

Nëse i shënojmë numrat me a, b dhe c, më i madhi prej a dhe b me p, por më i madhi prej p dhe c me n, atëherë mund të shkruajmë algoritëm më detal, si te ***figura 1.2.2.***

hapi 1: Dhënia e numrave a, b dhe c.
hapi 2: Nëse a është më i madh se b, atëherë p = a, përndryshe p = b.
hapi 3: Nëse p është më i madh se c, atëherë n = p, përndryshe n = c.
hapi 4: Shtypja e n.

Figura 1.2.2.

Treguam se në çdo algoritëm ka të dhëna hyrëse dhe rezultate dalëse. Rezultatet dalëse fitohen me „transformimin" e të dhënave hyrëse gjatë realizimit të algoritmit hap pas hapi.

Poashtu, ai transformim nuk kryhet gjithmonë direkt, por me ndihmën e **ndërmjet rezultateve**. Kështu, te algoritmi i sipërm ndërmjet rezultat është p, të dhëna hyrëse janë a, b dhe c, por rezultat dalës është n.

Për ta kontrolluar korrektësinë e algoritmit, kryhet testimi mbi çfarëdo të dhëna hyrëse. Për shembull, ta testojmë algoritmin për numrat: a = 37, b = 12, c = 44.

hapi 1: a = 37, b = 12, c = 44.
hapi 2: a (= 37) është më i madh se b (= 12), atëherë p = a (= 37).
hapi 3: p (= 37) nuk është më i madh se c (= 44), pasi n = c (= 44).
hapi 4: shtyp n (= 44).

Detyra për ushtrime

1. Shkruani algoritmin për të bërë torte.
2. Shkruani algoritmin për përgatitje të mëngjesit të dilni prje shtëpie.
3. Shkruani algoritmin për kopjim të datotekave prej disku në uesbe.
4. Cilat janë hapat e mundshëm për caktimin e vlerës mesatare të tre numrave?
5. Përpiquni të shkruani algoritëm (me hapa algoritmike) për ndarjen e shifrës së mesme prej numrit treshifror të dhënë.
6. Shkruani algoritëm për të konstatuar ndonjë numër natyror n a është çift ose tek. (Nëse numri është plotpjesëtueshëm me 2, atëherë është çift, ndryshe është numër tek.
7. Shkruani algoritëm me të cilin do të gjeni cilën ditë me radhë në vit është sot. Për shembull, nëse sot është viti 01.09.2020, sot është dita 245 e vitit.
8. Shkruani algoritmin e përgjithshëm të pjesëtuesit më të madh të përbashkët (PMP) për dy numra natyrorë.
9. Shkruani algoritmin e përgjithshëm për gjetjen e shumëfishit më të vogël të përbashkët (SHVP) për dy numra natyrorë.
10. Shkruani algoritmin me të cilin do të njehsoni sa vjet, muaj dhe ditë keni sot.

1.3 Paraqitja e algoritmeve

Algoritmet mund të paraqiten në tri mënyra

- Paraqitja tekstuale.
- Paraqitja grafike.
- Me gjuhë programore

Paraqitja tekstuale e algoritmeve kryhet me hapa, sikure që është treguar te nënpika **1.2 Algoritme**. Poashtu mund të shfrytëzohet edhe e ashtuquajtura **pseudo-gjuhë** për përshkrimin e algoritmeve.

Ne do të shfrytëzojmë pseudo-gjuhë (me fjalë të reja të gjuhës shqipe), që përbëhet prej fjalëve: **algoritëm, nënalgoritëm, fillimi, fundi, nëse, atëherë, përndryshe, rast, ndërsa, realizo, përsërit, deri, për, hap, zmadho, zvogëlo, lexo, shtyp, vazhdo, ndërprerje, dalje, kërcim, dalje.**

Këto fjalë do t'i shkruajmë të zeza (të trasha) që të potencojmë se ato janë fjalë të rezervuara prej pseudo-gjuhës.

Shembulli për paraqitje tekstuale të algoritmit me pseudo-gjuhë të këtillë është dhënë te ***figura 1.3.1***, ku është paraqitur algoritmi për gjetjen e më të madhit prej tre numrave prej ***figura 1.2.2***. Në atë, me shigjetë shprehet veprimi me të cilën ndryshores p i ***shoqërohet*** vlera e ndryshores a.

```
algoritëm Më i madh
fillimi
        lexo a, b, c;
        nëse a > b
                atëherë
                        p ← a
                ndryshe
                        p ← b
      fundi_nëse {a > b}
        nëse p > c
                atëherë
                        n ← p
                ndryshe
                        n ← c
        fundi_nëse {p > c}
        shtyp n;
fundi {Më i madh}
```

Figura 1.3.1

Paraqitja grafike e algoritmeve kryhet me të ashtuquajturën **bllok-diagrame** (angl. flowchart). Te bllok-diagrame shfrytëzohen simbole grafike të veçanta (blloqe) për veprime të caktuara (operacione) të dhëna te ***figura 1.3.2***. Paraqitja e këtillë quhet **paraqitja standarde grafike**.

Algoritmi për detyrën për caktimin e më të madhit prej tre numrave të dhënë të paraqitur grafikisht me bllok-diagramin prej ***figura 1.3.1*** është dhënë te ***figura 1.3.3.***

Paraqitja grafike e algoritmeve ka përparësitë e tija dhe mungesa të tija. Përparësia është në dukshmërinë më të madhe të rrjedhjes së aktiviteteve te algoritmi pasi njeriu më lehtë e percepton figurën prej tekstit. Nga ana tjetër, bllok-diagrami për algoritëm më të madh mund të zen më shumë faqe dhe algoritmi të jetë i pakontrolluar.

Paraqitja e algoritmit mund të jetë edhe direkt nëpërmjet gjuhës programore. Poashtu, algoritmin e „shprehim", d.m.th., **kodojmë** (angl. code) me urdhrin prej ndonjë gjuhe programore. Kështu algoritmi i koduar me urdhër prej ndonjë gjuhe

programore quhet **program burimor** (angl. source program). Që të mund të realizohet program burimor, ai patjetër duhet të përkthehet në **program realizues** (angl. executive program).

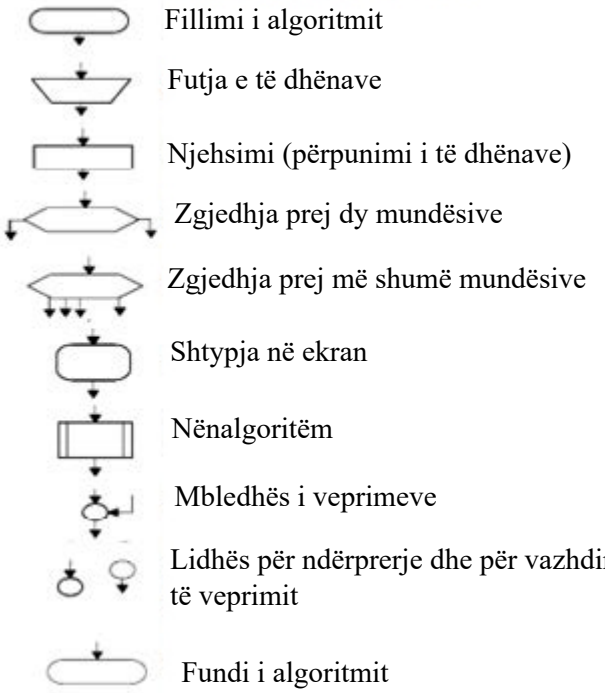


Figura 1.3.2

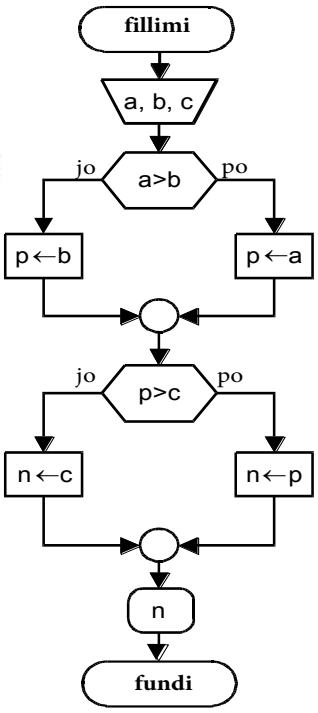


Figura 1.3.3

Për shembull, algoritmi i shqyrtuar për gjetjen e më të madhit prej 3 numrave të dhënë, të koduar në gjuhën programore C++ është dhënë te ***figura 1.3.4.***

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {    // Më i madhi prej tre numrave.
5
6       double a, b, c, p, n;
7       cout <<"futni numrin e parë:";
8       cin >> a;
9       cout <<"futni numrin e dytë:";
10      cin >> b;
11      cout << "futni numrin e tretë:"
12      cin >> c;
13      if (a > b)
14          p = a;
15      else
```

Figura 1.3.4

```
16          p = b;
17      if (p > c)
18          n = p;
19      else
20          n = c;
21      cout <<"Më i madhi prej numrave"<< a << ", "
22          << b << " i " << c << " e " << n << endl;
23
24      cout << endl;
25      system("Color 17");
26      system("pause");
27      return 0;
28  }
```

Figura 1.3.4 (vazhdim)

Pas realizimit të programit, dalja është:

```
futni numrin e parë : 123
futni numrin e dytë : 321
futni numrin e tretë :234
Më i madhi prej numrave  123, 321, 234 është  321
```

Algoritmet e strukturuara

Algoritmet për detyrat më të ndërlikuara mund të jenë shumë të gjata dhe jo të dukshme. Prandaj edhe programet që do të shkruhen sipas atyre algoritmeve mund të jenë të vështira të kuptueshme. Kjo është në veçanti e rëndësishme kur do të paraqitet nevoja prej ndonjë ndryshimi ose plotësimi te ato. Për gjetjen më të lehtë te programet e mëdha, sot ato shkruhen me teknikë për programin të njohur me emrin **programimi i strukturuar** (angl. struktured programming).

Emri programimi i strukturuar është fituar sipas shfrytëzimit të ashtuquajturave **strukturat algoritmike të kontrolluara** (angl. algorithm control strukturës), nëpërmjet të cilave menaxhohet (kontrollohet) gjatë realizimit të algoritmeve.

Te programimi i strukturuar shfrytëzohen dy teknika për programim:

- **Programimi prej lart** (angl. top-down programming).
- **Programimi modular** (angl. modular programming).

Programimi prej lart poshtë kryhet me ndarjen (zbërthimin) e detyrës në detyra më të vogla dhe më të thjeshta të ashtuquajtura **nëndetyra**. Nëse është e nevojshme, edhe ato nëndetyra më tej ndahen akoma në nëndetyra më të thjeshta deri sa nuk fitohen detyra që lehtë programohen.

Çdo nëndetyrë prej detyrës së zbërthyer mund të shqyrtohet si tërësi e veçantë, e pavarur prej të tjerave. Për çdonjërën në veçanti mund të shkruhet algoritëm, por pastaj dhe të programohet si pjesë e veçantë programore. Pjesët (tërësitë) e atilla të programit quhen **module** (angl. modules).

Çdo modul ka vetëm një pikë hyrëse (fillim) dhe vetëm një pikë dalëse (fund), ***figura 1.3.5.*** Të gjitha modulet te programi janë renditur në mënyrë sekuenciale (njëri pas tjetrit) dhe asnjëri prej tyre nuk mund të kapërcejë. Funksionin që e realizon një modul nuk guxon të përsëritet në tjetër modul, përkatësisht nuk guxon të ketë puthitje të moduleve. Gjithashtu, një modul mund të përbëhet prej më shumë moduleve të tjera, por mund të jetë edhe pjesë prej modulit më të madh.

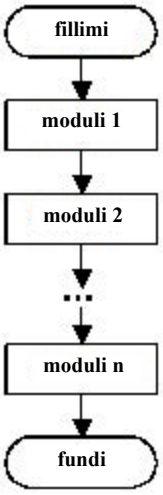


Figura 1.3.5

Programimi modular është në veçanti i dobishëm gjatë ndryshimit të programeve ose gjatë përmirësimit të gabimeve. Nëse duhet të ndryshojë programi ose të përmirësohet ndonjë gabim te ai, atëherë ndryshon vetëm moduli te i cili është paraqitur gabimi, por nuk ndryshojnë modulet e të tjera.

Paraqitja grafike e algoritmeve gjatë programimit të strukturuar, dallohet prej paraqitjes gjatë programimit të pa strukturuar, e cila quhet **paraqitje standarde**.

Algoritmet e strukturuara paraqiten me blloqe drejtkëndore. Çdo drejtkëndësh shpreh tërësi (modul) të veçantë me fillimin dhe mbarimin e vet. Nëse ndonjë modul ndahet në pjesë më të thjeshta, ai paraqitet me drejtkëndësha më të vegjël te më i madhi. Për shembull, algoritmi për gjetjen e më të madhit prej tre numrave të dhënë, i paraqitur në mënyrë standarde te ***figura 1.3.3***, në mënyrë të strukturuar është paraqitur te ***figura 1.3.6***.

Pyetje për kontroll të njohurive

1. Çka është programi, kurse çka është programimi?
2. Si quhen gjuhët te të cilat programohet?
3. Cilat janë programet sistemore, kurse cilat janë programet aplikative?
4. Si grupohen programet sistemore?
5. Cila është ndarja kryesore e softuerit?
6. Microsoft Windows a është program aplikativ?
7. Në cilin grup programet aplikative do ta vendojë programin Microsoft Excel?
8. Përmend tre programe aplikative të cilat punon kompjuteri Juaj. Përpiquni ta vendojë në kategorinë përkatëse prej nivelit të dytë te ***figura 1.1.1.***
9. Përpiquni të jepni shembull për program për çdonjërën prej grupeve softuer prej nivelit të tretë te ***figura 1.1.1.***

```
lexo a, b, c
nëse a > b
        atëherë
                p ← a
        ndryshe
                p ← b
nëse p > c
        atëherë
                n ← p
        ndryshe
                n ← c
shtyp n
```
Figura 1.3.6

10. Çka është algoritëm? Përmend një përkufizim.
11. Çka është hap algoritmik?
12. Si mund të jetë algoritmi sipas detalizimit?
13. Si mund të paraqitet një algoritëm?
14. Me cilat diagrame grafike janë paraqitur algoritmet?
15. Përmend simbolet (blloqet) grafike për paraqitje standarde grafike të algoritmeve.
16. Çka është programi burimor?
17. Kompjuteri a mund të realizojë direkt program burimor?
18. Çka është programim i strukturuar?
19. Cilat teknika shfrytëzohen gjatë programimit të strukturuar?
20. Si paraqiten grafikisht algoritmet e strukturuara?
21. Çka paraqet moduli programor?
22. Pse është e dobishme teknika modulare për programim?
23. Sqaroni (ose kërkoni t'u sqarohen) teknikat për programim prej lart poshtë.

1.4 Programimi

Që të mund ndonjë të shkruajë program për zgjidhjen e detyrës së caktuar me ndihmën e kompjuterit, është e nevojshme të dijë të programojë.

Programimi ***është proces i të shkruarit e programit.***
Njerëzit që programojnë quhen programuesë.

Fazat e programimit

Që të zgjidhet ndonjë detyrë me ndihmën e kompjuterit, nuk është e mjaftueshme vetëm të dimë të programojmë dhe të shkruajmë program. Programimi është një pjesë e atij procesi, që përbëhet prej disa fazave:

- Parashtrimi i detyrës.
- Përkufizimi i algoritmit (mënyra) për zgjidhjen e detyrës.
- Të shkruarit e programit.
- Testimi i programit.

Faza I: Vendosja e detyrës

Gjatë parashtrimit të detyrës, saktë duhet të përkufizohen dhe të precizohen kushtet nën të cilat ai do të zgjidhet. Prandaj, së pari duhet të kuptohet drejt detyra. Ajo bëhet me analize të natyrës së saj, por nëse është e nevojshme edhe me hyrje te fusha profesionale e të cilës i takon.

Për ta sqaruar atë që u përmend lart, do të parashtrojmë disa detyra dhe do t'i analizojmë.

1. Të gjendet shuma e 10 numrave natyrorë.
2. Të zgjidhet sistemi prej dy barazimeve lineare me dy të panjohura.
3. Të bëhet lista e artikujve në një shitore sipas sasisë dhe sipas vlerës dhe të njehsohet vlera e përgjithshme e mallit.
4. Prej 16 fijeve të shkrepëses dy lojtarë A dhe B marrin në mënyrë alternative nga 1, 2, 3 ose 4 fije. Fiton ai i cili i fundit do të marrë fije. Njëri prej tyre a mund të fitojë?

Për detyrën e parë çdonjëri do të thotë se nuk është e vështirë – pasi di të mbledhë di edhe të shkruajë program.

Për detyrën e dytë duhet të dihet ndonjë metodë për zgjidhjen e sistemit prej dy barazimeve lineare me dy të panjohura, sikurse janë: metoda e zëvendësimit ose metoda e koeficientëve të kundërt.

Për detyrën e tretë duhet të dihet:

- Emri i çdo artikulli.
- Sasia e çdo artikulli.
- Çmimi për njësi, për litër ose sipas paketimit.

Për detyrën e katërt nuk do të vendojmë fije në kompjuter, por prej numrit 16 në mënyrë alternative do të zbresim nga aq fije sa do të tregojë çdo lojtar - 1, 2, 3 ose 4. Thelbi i detyrës është të mendohet nga sa fije të shkrepëses duhet të merren, duke ditur sa kanë marrë kundërshtarët edhe sa kanë ngelur për të marrë të fundit dhe të fitojë.

Faza II: Përkufizimi i algoritmit për zgjidhjen e detyrës

Pas kryerjes së analizës së detyrës është e nevojshme të përkufizohet (gjendet/ mendon) ose të zgjedhë (nëse di më shumë) algoritme për zgjidhjen e saj. Më së shpeshti mënyra del prej analizës së bërë të detyrës dhe prej literaturës profesionale të konsultuar te e cila përfshihen relacione, formula dhe metoda të caktuara për zgjidhjen e saj. Po ashtu, duhet pasur llogari mënyra të jetë e zbatueshme për kryerje në kompjuter. Tek ai qartë duhet të përmenden të gjitha operacionet që duhet t'i kryejë kompjuteri për të dhënë rezultate të sakta. Çdo operacion duhet të jetë njëvlerësisht i përkufizuar, por edhe renditja e operacioneve duhet të jetë saktë e dhënë. Kuptohet, gjithë mënyra duhet të jetë e fundshme, përkatësisht të mbarojë pas numrit të fundshëm të operacioneve d.m.th., pas kohës së fundshme të realizimit. Mënyra e këtillë e përkufizuar për zgjidhjen e ndonjë detyre, sikurse që ishte edhe paraprakisht e përmendur, quhet **algoritëm.**

Të përmendim se për zgjidhjen e detyrës së njëjtë mund të shkruhen më shumë algoritme të ndryshme.

Faza III: Të shkruarit e programit

Të shkruarit e programit paraqet të shkruarit (të shprehurit, kodimit) të algoritmit me elementet e ndonjë gjuhe programore. (Për gjuhët programore do të flasim në nënpikën vijuese).

Faza IV: Testimi i programit

Te faza e katër kryhet testimi i programit për të kontrolluar a jep rezultate të sakta. Testimi kryhet më së shpeshti me vlerat e zgjidhjeve për të cilat i dimë ose mundemi (në ndonjë mënyrë) t'i kontrollojmë.

Gjuhët programore

Njerëzit ndërmjet veti kuptohen dhe komunikojnë me ndihmën e gjuhës. Kështu, pa dallim se a është paraqitur me shenja, me simbole ose duke folur, gjuha paraqet mjet themelor për komunikim.

Gjuhët mund të jenë:

- **Natyrore.**
- **Artificiale.**

Për komunikimin ndërmjet veti njerëzit i shfrytëzojnë gjuhët natyrore, si për shembull, shqipen, maqedonishten, anglishten, rusishten etj. Për komunikim ndërmjet njerëzve dhe makinave (ose ndërmjet dy makinave), shfrytëzohen gjuhët artificiale.

Lloj i veçantë i gjuhëve artificiale janë gjuhët programore, të cilat janë të zhvilluara për komunikim ndërmjet njeriut dhe kompjuterit. Me gjuhët programore shprehet mënyra (algoritmi) për zgjidhjen e ndonjë detyre të shkruar në formë tekstuale të quajtur **program**. Programet shkruhen me urdhëra të cilat kompjuteri i „kupton" dhe i realizon ato veprime të cilat janë „shprehur" me ato.

Prej paraqitjes së kompjuterit deri më sot janë zhvilluar një numër i madh i gjuhëve programore. Ato ndahen në tre grupe:

- **Gjuha makinerike.**
- **Gjuha simbolike.**
- **Gjuhë e programimit të lartë.**

Gjuhë makinerike

Gjuha makinerike (angl. machine language) është gjuha e kompjuterit. Ajo është gjuhë e zakonshme që çdo kompjuter e kupton dhe te e cila kompjuteri punon menjëherë. Përbëhet prej numrit të caktuar **instruksionesh makinerike** (angl. machine

[2] Mendohet se deri më tani janë zhvilluar mbi 3 000 gjuhë programore.

instructions), të cilat shprehen vetëm me zero dhe njëshe, d.m.th., vetëm me shifra binare[3], ***figura 1.4.1.***

Megjithatë, shënimi me numra binarë është shumë i madh dhe i pa shqyrtuar. Prandaj, programet makinerike (në letër më së shpeshti shkruhen në *sistemin numerik heksadhjetor*[4], ***figura 1.4.1.***

Instruksione makinerike

Shënimi binar	Shënimi heksadhjetor
10111000110101100001011O	B8D616
1000111011011000	8ED8
10100000000000000000000	A00000
00000010000001100000000100000000	02060100
10111011000100001000000O	BB1080
1000100000000111	8807
1011010001001100	B44C
1100110100100001	CD21

Figura 1.4.1

Gjuhë simbolike

Gjuha simbolike (angl. assembly language) është në nivel më të lartë prej gjuhës makinerike. Te kompjuterët e parë, programet janë shkruar vetëm me gjuhë makinerike. Me kohë, për më lehtë dhe për më shpejt paraqitja e zgjidhjes së detyrës së parashtruar, programuesit kanë filluar të shërbehen me fjalë të shkurtëra të thjeshta të quajtura **simbole** ose **mnemonikë**[5] për të shënuar instruksionet makinerike. Kështu kanë ndërtuar gjuhët simbolike. Emrin e kanë marrë pikërisht për shkak të asaj që me ato instruksione makinerike dhe të dhënat shprehen në mënyrë simbolike.

Për shembull, programi makinerik prej ***figurës 1.4.1*** ka shënim simbolik sikurse te ***figura 1.4.2***[6].

[3] Shifrat binare janë 0 dhe 1. Ato janë shifra të ashtuquajtura *sistemi numerik binar*. Ne e kemi të njohur *sistemin numerik dekad* te i cili ka 10 shifra – 0, 1, 2, 3, 4, 5, 6, 7, 8 dhe 9.

[4] *Sistemi numerik heksadhjetor* ka 16 shifra, edhe atë: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E dhe F.

[5] Mnemonika është teknikë për mbajtje mend.

[6] Në fillim të çdo urdhri te ***figura 1.4.2*** është treguar adresa te e cila gjendet në memorie.

Gjuhët simbolike kanë shkallë më të madhe të kuptueshmërisë prej gjuhës makinerike. Për shembull, prej ***figura 1.4.2*** shikohet se MOV është shkurtesë për zhvendosje (angl. move), ADD do të thotë zhvendosje, d.m.th., mbledhje (angl. add) etj.

Por gjuhët simbolike janë të pakuptueshme për kompjuterin, që do të thotë se program i shkruar me gjuhën simbolike nuk mund të realizohet menjëherë në kompjuter. Prandaj, janë përpunuar programe sistemore të veçanta të quajtura **përkthyes** (angl. compilers), të cilat programin prej gjuhës simbolike e përkthen në program të gjuhës makinerike.

```
16D7 : 0000   MOV    AX, 16D6
16D7 : 0003   MOV    DS, AX
16D7 : 0005   MOV    AL, [0000]
16D7 : 0008   ADD    AL, [0001]
16D7 : 000C   MOV    BX, 8010
16D7 : 000F   MOV    [BX], AL
16D7 : 0011   MOV    AH,4C
16D7 : 0013   INT    21
```

Figura 1.4.2

Gjuhët simbolike janë gjuhë **makinerike të orientuara** pasi çdo kompjuter ose familje kompjuterësh ka gjuhë simbolike personale të vet, që varet prej procesorit të kompjuterit.

Gjuhët simbolike më së shpeshti janë të njohura me emrin gjuhë asemble (angl. assembly), por përkthyesit e tyre quhen asemblerë (angl. assemblers).

Gjuhët programore të larta

Gjuhët programore të larta (angl. high-level languages), janë gjuhë te të cilat mendohet kur flasim për programimin. Zhvillimi i tyre fillon në vitet 50 të shekullit XX kur kompjuterët me universalitetin e tyre kanë filluar të hyjnë në shumë fusha të punës njerëzore. Te prodhuesit e kompjuterëve është imponuar pyetja: Se si kompjuterët të afrohen deri te njeriu që të mund t'i shfrytëzojnë edhe njerëzit të cilët nuk e dinë teknikën për programimin makinerik dhe simbolik?

Ideja themelore në zgjidhjen e këtij problemi ka qenë programet të shkruhen në gjuhë të ngjashme me gjuhën natyrore të njeriut. Për një kohë shumë të shkurtër, gjuhët programore të larta kanë qenë përgjithësisht të pranuara prej një numri të madh të shfrytëzuesve.

Pranimi i shpejtë dhe shfrytëzimi masiv i këtyre gjuhëve i përket këtyre fakteve:

- *Programimi dukshëm është lehtësuar për shkak të afërsisë së këtyre gjuhëve me mënyrën natyrore e shkruar të të shprehurit të njeriut edhe me terminologjinë nga fusha të cilës i takon detyra.*
- *Mundësojnë shpejt dhe në mënyrë efikase të shkruarit e algoritmeve.*
- *Nuk kërkojnë njohuri të vetive teknike të kompjuterit te e cila kryhet program, përkatësisht program i njëjtë mund të kryhet në kompjuter të ndryshëm.*
- *Shpejt dhe lehtë mësohen.*
- *Mundësojnë shkëmbim të programeve dhe të përvojave ndërmjet shfrytëzuesve.*

Të gjitha faktet e përmendura bazohen në strukturat e gjuhëve programore të larta, që në mënyrë të dukshme dallohet prej strukturës së gjuhës makinerike dhe simbolike.

Edhe se gjuhët programore të larta janë gjuhë artificiale, ato janë ndërtuar në mënyrë të ngjashme sikurse edhe natyrore. Përkatësisht, çdonjëra prej atyre gjuhëve ka **alfabetin** e vet i përbërë prej shkronjave, shifrave dhe shenjave speciale: shenjat e pikësimit, shenja për operacionet aritmetikore, për krahasim etj[7].

Me kombinimin e shenjave prej alfabetit, formohen konstruksionet elementare të gjuhës – **fjalët**, të cilat e përbëjnë **fjalorin e gjuhës**. Për shembull: read, do, while, new, OPEN, MULTIPLY, STOP, if etj.

Fjalët kanë domethënie të caktuar, por te programet nuk mund të figurojnë sikurse elemente të pavarura direkt do të ndikojnë mbi punën e kompjuterit, por kombinohen me konstruksionet gjuhësore të quajtura **fjali** ose **urdhëresa** (angl. statements). Urdhëresat kanë saktë domethënie të përkufizuar dhe mund te programi të qëndrojnë si tërësi të pavarura, d.m.th., të shkaktojnë aksion të caktuar te kompjuteri.

Qe disa shembuj për urdhëresa:

```
if (ti > unë) System.out.printën("Urime. Ti fitove. ");
throw new UserDefinedException("Është krijuar UserDefinedException! ");
MULTIPLY BROJ1 BY BROJ2 GIVING PRODHIM.
```

Është e rëndësishme të potencojmë (sikurse te gjuhët natyrore) se çdo varg (kombinim) shenjash prej alfabetit nuk paraqet fjalë prej gjuhës, përkatësisht se çdo konstruksion i fjalëve nuk paraqet urdhër. Prandaj, edhe gjuhët programore të larta, si edhe natyrore, kanë **gramatikën** e tyre e cila i përmban rregullat për ndërtimin e fjalëve dhe urdhërave.

Çdo gjuhë përmban bashkësi të caktuar **fjalë të rezervuara** (angl. reserved words). Programuesit mund të shfrytëzojnë edhe fjalë të tjera, të ndryshme prej të rezervuarave, të cilat janë formuar sipas rregullave të caktuara. Ato quhen **identifikator** (angl. identifiers).

Ekzistojnë rregulla rigoroze sipas të cilat konstruktohen urdhërat, të cilat quhen **rregulla sintetike** ose vetëm **sintaksa** (angl. syntax) e gjuhës. Me ato rregulla kontrollohet korrektësia e çdo urdhri te programi.

Çdo rregull te programi patjetër të ketë saktë domethënie të përkufizuar dhe të shkakton saktë veprime të përkufizuara. Domethënia (kuptimi) i urdhërave quhet **semantika** (angl. semantiks) e gjuhës.

Programet e shkruara në gjuhët programore të larta nuk mund direkt të kryhen te kompjuteri. Për këtë qëllim, janë të nevojshme programe sistemore që do të kryejnë përkthimin e tyre në gjuhën makinerike të quajtur **përkthyes**. Çdo gjuhë programore ka përkthyesin e vet.

[7]Te gjuhët programore më së shpeshti shfrytëzohet alfabeti anglez.

Programi i shkruar në gjuhën e lartë programore quhet **programi burimor** (angl. source program), por program i përkthyer makinerik quhet **program i realizuar** (angl. executive program).

Përkthimi në mënyrë skematike është paraqitur në këtë figurë.

Figura 1.4.3 a

Programi i ekzekutueshëm zakonisht mbahet mend si datotekë të ndonjë memorie të jashtme, që prej atje të thirret dhe të realzon sipas nevojës. Por ndonjëherë ajo vetëm memorohet në memorien e brendshme dhe realizohet.

Shpesh shfrytëzohet dhe direkt realizimi i programeve burimore. Ajo bëhet me programe të tjera sistemore të quajtura **interpretatore** (angl. interpreters.

Interpretuesit nuk bëjnë datoteka në formë makine te programi burimor, por në mënyrë sekuenciale i interpretojnë dhe i realizojnë urdhëresat e saja.

Skema e interpretimit të programit është dhënë te figura 1.4.3.b.

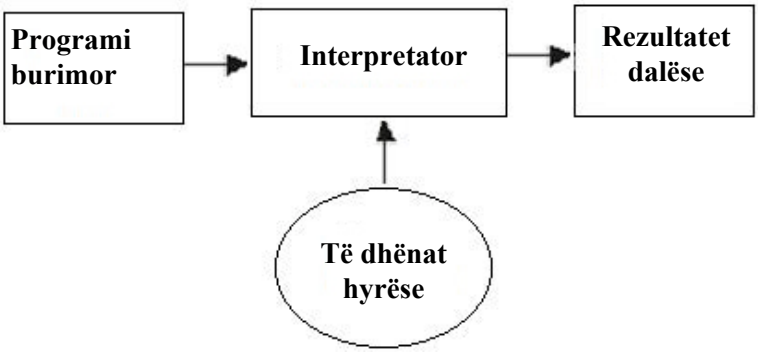
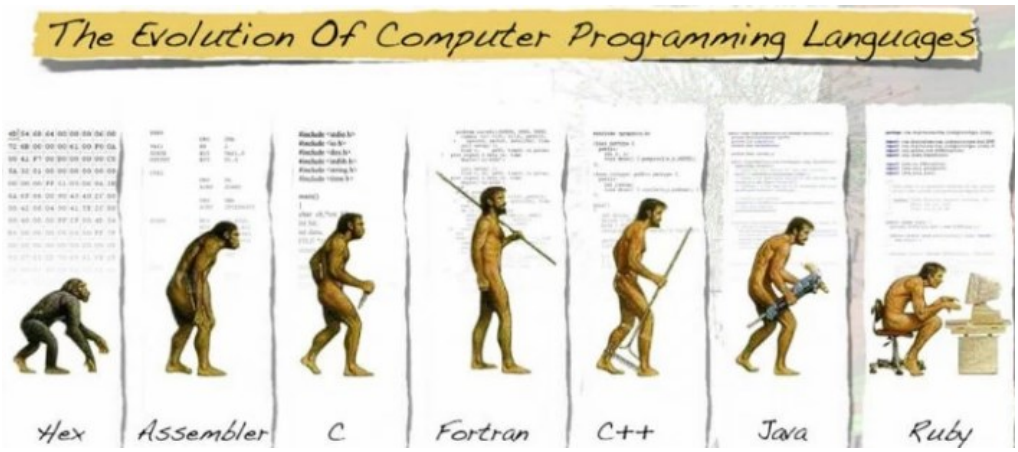


Figura 1.4.3 b

Gjuhët programore të larta nuk varen prej makinës te e cila realizohet ndonjë program i shkruar me atë dhe prandaj quhen **gjuhë makinerike të pavarura**. Prandaj, gjatë ndërtimit të gjuhëve programore të larta, nuk mbahet llogari për llojin e kompjuterit te i cili do të realizohen programet të shkruara me ato, por mbahet llogari për llojin e problemit që zgjidhet. Themi se ato janë orientuar kah zgjidhja e llojit të caktuar të problemeve (ekonomike, teknike, statistike etj.) dhe prandaj, quhen **gjuhë problemore të orientuara**.

Zhvillimi i gjuhëve programore të larta ka filluar në vitet 50-ta të shekullit të kaluar. Zhvillimi i tyre në mënyrë figurative është paraqitur me këtë figure, e marrë prej literaturës.



Te ***tabela 1.4.1*** është dhënë paraqitja kronologjike e disa gjuhëve programore më të rëndësishme.

Viti	Gjuhë		
1945	Gjuhë makinerike		
1950	Gjuhë simbolike		
1955	Fortran		
1960	Algol	Cobol	Lisp
1965	Basic	Simula	PL/I
1970	Pascal	Prolog	Logo
1975	Cobol	Scheme	ML
1980	C	Ada	Eiffel
1985	Objective-C	SQL	Perl
1990	C++	Visual Basic	Python
1995	Delphi	HTML	Ruby
	Java	JavaScript	PHP
2000	C#	Scala	R
2007	Go	Clojure	
2010	Swift	Google Dart	Julia
	Hack	Rust	Pixie
2012	TypeScript		
2016	Kotlin		

Tabela 1.4.1

Sot, gjuhët programore bashkëkohore të zhvilluara pas viti 2010 janë të dedikuara për programimin e veb-aplikacioneve të cilat punojnë në internet, por të cilat i integrojnë karakteristikat e mira të disa gjuhëve. Ato janë të ashtuquajturat **gjuhët e shkrimit** (angl. scripting languages), me të cilat janë punuar programe të veçanta – skripte (angl. scripts) të cilat realizohen në veb-faqet, por gjatë realizimit të programeve të tjera. Skriptet nuk përkthehen, por interpretohen. Gjuhët e shkrimit janë dizajnuar, për integrim dhe komunikim me gjuhët programore të tjera, sikurse: HTML, Java, C++ etj.

Gjuhë të shkrimit më të njohura janë: JavaScript, PHP, Python, VBScript etj.

Gjuhët programore më të shfrytëzuara për programin janë dhënë te ***figura 1.4.4***.

Figura 1.4.4

Te ***figura 1.5.5 a, b, c, ç, d*** dhe janë paraqitur programet e thjeshta për zgjidhjen e detyrës te e cila duhet të gjendet shuma e 10 numrave natyrorë të parë të shkruar Ce++, Java, Pascal, Fortran, FORTRAN), Rubby dhe Python.

```
#include <iostream>
using namespace std;

void main()
{
        int n,suma;
        suma = 0;
        n = 1;
        while(n <= 10) {
                suma = suma + n;
                n = n + 1;
        }
        cout << suma << endl;
}
```

a

Programi në C++

```
public class JavaPrimer {
   public static void main(String[] args){
        int n,suma;
        suma = 0;
        n = 1;
        while (n <= 10) {
            suma = suma + n;
            n = n + 1;
        }
        System.out.println(suma);
   }
}
```

b

Programi në Java

```
PROGRAM Suma;
VAR  N,Suma:integer;
BEGIN
     Suma := 0;
     N := 1;
     WHILE N <= 10 DO
        BEGIN
                Suma := Suma + N;
                N := N + 1;
        END;
     WriteLn(Suma);
END.
```

c

Programi në PASCAL

```
      SUMA = 0
      N = 1
      WHILE N <= 10
              SUMA = SUMA + N
              N = N + 1
      ENDWHILE
      WRITE(*, 30)SUMA
 30   FORMAT(I10)
      STOP
```

ç

Programi në FORTRAN

```
suma = 0
n = 1
while n <= 10
  suma = suma + n
  n = n + 1
end
puts suma
```

d

Programi në Ruby

```
suma = 0
n = 1
while n <= 10:
    suma = suma + n
    n = n + 1
print suma
```

dh

Programi në Python

Figura 1.4.5

Të shkruarit e programeve kryhet në ndonjë redaktues të tekstit, por pastaj ato shkruhen (ruhen) në ndonjë memorie të jashtme si datotekë nën ndonjë emër. Programi i atillë nuk mund të kryhet pasi është e nevojshme paraprakisht të përkthehet.

Programet burimore shpeshherë kanë zgjerim pas emrit që është shkurtesa prej gjuhës programore. Zgjerimi është ndarë prej emrit me pika:.pas,.bas,.for,.cpp,.java etj. Për shembull: shuma.pas, e parë.bas, Programime.cpp, rendit.java etj.

Programet e realizuara të cilat fitohen pas përkthimit të programeve burimore ka zgjerim:.exe,.com,.bat,.bin etj. Për shembull: shumë.exe, e para.com etj.

Gjenerata të gjuhëve programore

Sipas zhvillimit historic, gjuhët programore ndahen në pesë gjenerata (angl. Generation Languages – GL), edhe atë:

- Gjenerata e parë (1 GL): gjuhë makinerike.
- Gjenerata e dytë (2 GL): gjuhë simbolike.
- Gjenerata e tretë (3 GL): gjuhë programore të larta (C++, Java, Fortran etj.).
- Gjenerata e katërt (4 GL): shumë gjuhë programore të larta (angl. very high-level languages) (Perl, PHP, Python, SQL etj.).
- Gjenerata e pestë (5 GL): gjuhë programore të larta8 (Mercury, Prolog, OPS5 etj.).

Ndarja e gjuhëve programore

Ndarja e gjuhëve programore mund të bëhet dhe sipas mënyrës në të cilë janë përpunuar të dhënat:

- **Imperative**.
- **Deklarative**.

[8]Disa autor i quajnë gjuhë për inteligjencën artificiale (angl. artificial intelligence languages).

Mënyra imperative e mënyrës së urdhrit, ku programet shkruhen me urdhër për përpunimin e të dhënave.

Mënyra deklarative është mënyrë përshkruese që përshkruan se si arrihet deri te rezultati.

Gjuhët imperative ndahen në:

- **procedurale** (Pascal, C, Fortran, Basic, Ada, Modula dhe të tjera),
- **objektive të orientuara** (C++, Java, C#, Delphi, Smalltalk dhe të tjera).

Karakteristika kryesore e këtyre gjuhëve është shfrytëzimi i ndryshoreve, urdhëra dhe procedura[9]. Në gjuhët procedurale, procedurat me të cilat përpunohen të dhënat në mënyrë të ndarë, por në gjuhët objekte të orientuara procedurat (të quajtura funksione ose metoda) janë fshehur te vet objektet.

Gjuhët deklarative ndahen në:

- **funksionale** (Lisp, Scheme, Miranda, Haskell, ML dhe të tjera)
- **logjike** (Prolog dhe të tjera).

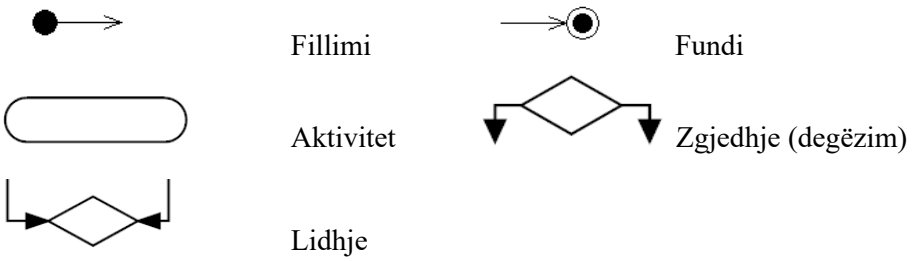
Karakteritika kryesore e gjuhëve funksionale është shfrytëzimi i shprehjeve dhe funksioneve. Programi paraqet funksion (se grup funksionesh) të përbëra prej funksioneve më të thjeshta.

Te gjuhët logjike, baza e njësisë për njehsim është relacioni, që është më e përgjithshme prej pasqyrimit. Programi përbëhet prej: bashkësisë së fakteve, bashkësia e përfundimeve të lejuara (rregulla për vërtetim), metoda për përfundime dhe gjithë vërtetimi.

Diagrami i aktiviteteve

Për paraqitje grafike të punës së programeve, shfrytëzohen diagrame të ndryshme. Për unifikimin e paraqitjes së diagrameve, kah fundi i viteve 90-ta të shekullit të kaluar është rijuar gjuha grafike e quajtur Unified Modeling Languga – UML.

UML ka numër të madh të simboleve grafike. Ne do të shfrytëzojmë këto:



Diagrami prej ***figura 1.3.3,*** i shprehur me gjuhën-UML, do të jetë si te ***figura 1.4.6***.

[9] Për ndryshoren dhe procedurat d të flasim më vonë.

Pyetje për kontrollin e njohurive

1. Çka është programimi?
2. Si quhen njerëzit që programojnë?
3. Sqaroni fazat për zgjidhjen e ndonjë detyre me ndihmën e kompjuterit.
4. Çka duhet të dimë për të mundur të shkruajmë program?
5. Për cilat gjuhë programore keni dëgjuar?
6. Me cilat shifra shkruhen programet në gjuhën makinerike? Pse nuk prograAmohet në gjuhën makinerike?
7. Në cilin sistem numerik shkruhen programet makinerike në letër?
8. A është më e thjeshtë gjuha makinerike ose gjuhët simbolike?
9. Si quhen programet sistemore që kanë detyrë t'i përkthejnë programet prej gjuhëve simbolike në gjuhë makinerike?
10. Në cilat fakte detyrohet shfrytëzimi masiv i gjuhëve të larta programore?
11. A ka dallim ndërmjet alfabetit natyror dhe të gjuhës programore?
12. Fjalët në gjuhët e larta programore a kanë domethënie?
13. Prej çka përbëhet fjalori i një gjuhe të lartë programore?
14. A mund një urdhër programor të përbëhet vetëm prej një fjale?
15. Si quhen fjalët që i formojnë programuesit te programet?

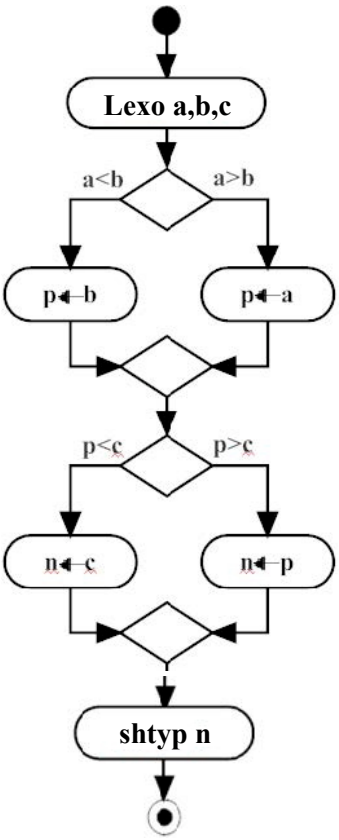


Figura 1.4.6

16. Si quhen rregullat sipas të cilës janë konstruktuar urdhërat te gjuhët programore?
17. Çka është sementika e gjuhës?
18. Si quhet program i shkruar në gjuhën e lartë programore, por si program i përkthyer në gjuhë makinerike? Cilat prapashtesa i kanë datotekat përkatëse?
19. Cili është dallimi ndërmjet përkthyesve dhe interpretuesve?
20. Pse themi se gjuhët e larta programore janë gjuhë të pavarura makinerike?
21. Përmend disa gjuhë të larta programore.
22. Përmend gjeneratat e gjuhëve të larta programore.

23. Përmend një ndarje të gjuhëve të larta programore.
24. Cila është karakteristika kryesore e gjuhëve procedural, por cila e gjuhëve objekte të orientuara?
25. Me cilën gjuhë grafike sot janë paraqitur diagramet? Vizato simbolet e tij themelore grafike.

1.5 Rrethina integruese zhvillimore

Sipas asaj që treguam më lart në nënpikat paraprake, që të mundemi të fillojmë me programim, patjetër të njihemi me ndonjë gjuhë të lartë programore. Pastj, e shkruajmë programin te ndonjë redaktues të tekstit dhe e rajmë si datotekë tekstuale

Që të mund të shikojmë programin që e kemi shkruar a e bën atë që pritet, ne atë duhet ta përkthejmë në gjuhë makinerike dhe ta realizojmë. Theksuam se gjatë përkthimit në gjuhën makinerike, shfrytëzohet ndonjë përkthyes dhe fitohet program/ datotekë realizuese. Programin realizues të fituar duhet ta kryejmë dhe madje atëherë të shikojmë çka bën program ynë. Nëse shfrytëzojmë interpretator, atëherë direkt përkthehet dhe realizohet, d.m.th. integrohet.

Për t'i lehtësuar punën programuesve, të gjitha programet të cilat na mundësojnë funksionet e lartpërmendura funksionet vendohen (integrohen) në një rrethinë, të quajtur ***rrethina e integruar për zhvillim*** RrIP (angl. Integrated Development Environment – IDE). Për programimin në C++, më shumë shfrytëzohen rrethinat integruese zhvillimore, sikurse: Microsoft Visual Studio, Code::Blocks, Netbeans, Eclipse, Visual Studio Code, CLion, CodeLite dhe të tjera.

Çdo rrethinë integruese zhvillimore përbëhet prej bshkësie të aplikimeve të cilat shfrytëzohen gjatë programimit. Ajo bashkësi e aplikimeve varet prej gjuhës programore, por aplikime të detyrueshme janë: **redaktues tekstual** (angl. text editor), **përkthyes** (angl. compiler), **dibager** (angl. debugger) dhe **lidhës** (angl. linker).

Redaktuesi tekstual

Ai është aplikacioni që mundëson futje të tekstit prej tastaturës direkt në programin burimor, sikurse edhe për realizimin e operacioneve themelore për përpunimin e teksteve. Kjo jep mundësi për ruajtjen e programeve në disk dhe hapja e programeve të ruajtura për ripërpunim, d.m.th., për ndryshim.

Redaktori i tekstit i ka të gjitha veglat standarde për futje dhe përpërpunim të tekstit. Gjatë redaktimit të tekstit, mundemi t'i shfrytëzojmë pothuajse urdhërat e njëjta të cilat i kemi në programet për redaktim të tekstit për punë në zyrë, sikurse: Microsoft Office Word, OpenOffice Writer etj. Gjithashtu, vlejnë të gjitha urdhërat për lëvizje të treguesit prej miut.

Urdhërat për selektim, kopjim, bartja dhe fshirja e tekstit janë të njëjta sikurse te redaktuesit e tjerë të tekstit. Poashtu, shfrytëzohen urdhërat prej menysë për redaktim, më shpesh të quajtur Edit.

Në shumicën e rasteve, mund të ketë nevojë prej kërkesës së tekstit të caktuar nëpër program, por pas gjetjes, dhe prej zëvendësimit të tij me tjetër tekst. Kjo realizohet me urdhërat në meny, më shpesh të quajtur Search.

Përkthyes/Interpretator

Përkthyesi është aplikacion special që përkthen program burimor (të shkruar në ndonjë gjuhë programore) te program burimor i gjuhës makinerike që të munduar kompjuteri ta realizojë.

Gjatë përkthimit, përkthyesi e përkthen gjithë programin burimor në programin realizues, që mund të realizohet menjëherë (nëse është në memories punuese) ose të ruhet te datoteka e ndonjë memorie të jashtme dhe më vonë (ose menjëherë) të realizohet, pa u përkthyer përsëri.

Interpretari e përkthen dhe realizon programin pjesë për pjesë (urdhër pas urdhri), d.m.th., pas disa urdhërave menjëherë. Gjatë realizimit përsëri të programit, interpretatori patjetër të përkthen dhe realizon çdo urdhër (ose grup të urdhërave).

Edhe përkthyesi edhe interpretuesit kanë përparësi dhe mangësi. Sot, disa gjuhë programore, sikurse Java dhe Visual Basic, shfrytëzojë kombinim prej përkthimit dhe interpretuesit. Për shembull, përkthyesi java e përkthen programin burimor në të ashtuquajturën ndërmjet kd, të njohur si **bajt-kod** (angl. bytecode). Programi në bajt-kod mund të realizohet me interpretuesin në çfarëdo kompjuter.

Ngjashëm bëhet edhe gjatë përkthimit të programit në C++, vetëm që përkthimi i programit burimor në ndërmjet kod trajtohet si faza e procesimit. Pastaj, forma e fituar e programit përkthehet në gjuhën makinerike, ***figura 1.5.1.***

Përkthyesit dhe interpretuesit janë dizajnuar të kontrollojnë gabimisht në program. Më të shpeshta janë gabime gjuhësore (sintaktike) kur urdhërat janë shkruar gabimisht. Për shembull, nëse në vend close, kemi shkruar cloce. Praqiten edhe gabime programore (semantike) gjatë shfrytëzimit jo të drejtë të urdhërave te gjuha. Për shembull, nëse në vend do {... }while(), emi shkruar do {... }for() dhe të ngjashme.

Dibager

Fatkeqësisht, përkthyesit mund të zbulojnë gabime te programi, për shkak të shfrytëzimit të gabuar të gjuhës programore. Lloj tjetër i gabimeve të cilat shpesh ndodhin, në veçanti prej programuesve fillestar, janë gabime logjike, d.m.th., programin nuk e bën ajo për të cilën është dedikuar edhe pse punon në mënyrë korrekte. Ato gabime zbulohen shumë vështirë dhe prandaj duhet të kontrollohet mënyra në të cilën realizohen programet paraprakisht me të dhënat hyrëse, për të cilat rezultati është i njohur. Të dhënat hyrëse të atilla quhen **test-shembuj**. Dibageri (angl. Debugger).

Dibageri është që ndihmon gjatë kërkimit të gabimeve logjike. Ai mundëson të përcjellim realizimin e programit të test-shembujve hap pas hapi dhe, t'i zbulojmë gabimet logjike. Gjatë kohës së atij procesi, dibageri mund të tregojë edhe disa ndërmjet rezultateve të cilat ndihojnë të caktohet vendi dhe lloj i gabimit.

Lidhësi

Ndonjëherë një program është shumë i madh për t'u shkruar si një datotekë. Përveç kësaj pjesët e ndryshme mund të shkruhen prej programuesve të ndryshëm. Ndonjëherë pjesët prej programit të dhënë mund të jenë të shfrytëzuara edhe në tjetër program. Në rastin e këtillë këto pjesë duhet të jenë formuar si datoteka të pavarura. Nëse ndonjë program duhet të përbëhet prej më shumë programeve të pavarura, ato duhet të bashkohen me lidhjen në datotekën e vetme, d.m.th. në program realizues të vetëm. Kjo bëhet me aplikacion të veçantë të quajtur **lidhës**.

Rol tjetër të lidhësit është t'i „lidh"programet e bibliotekave me programin që e shkruajmë. Çdo gjuhë programore ka bibliotekë (të quajtur edhe module) me programe të gatshme, të cilat mund të shfrytëzohen te çfarëdo program që e shkruajmë. Për shembull, program i bibliotekës për njehsimin e rrënjë katrore prej numrit real x, në shumë gjuhë programore quhet sqrt(x). Nëse te program jonë ka urdhër y=sqrt(x), atëherë lidhësi e lidh programin tone me bibliotekën te e cila gjendet program sqrt(x). (Emri sqrt(x) është shkurtesa prej square root).

Skema më reale e përkthimit të programit burimor, prej asaj të dhënës te ***figura 1.4.3*** a dhe b, është skema prej ***figura 1.5.1***.

Rrethina e integruar për programimin zakonisht duket si bashkësi prej më shumë aplikacioneve me të cilat për çdo ditë punojmë. Ai ka dritare themelore me *fushë punuese*, shirit për meny dhe elemente të tjera të zakonshme: buton për mbylloje dhe ndryshimi madhësisë së dritares, buton për minimizim etj. Në *fushën e punës* hapen dritaret e programeve të veçanta. Te shiriti për meny gjenden (lista prej urdhërave), me të cilën thirren funksionet e rrethinës: menyja për redaktim, menyja për përkthim, lidhja dhe realizimi i programeve, menyja për testim etj.

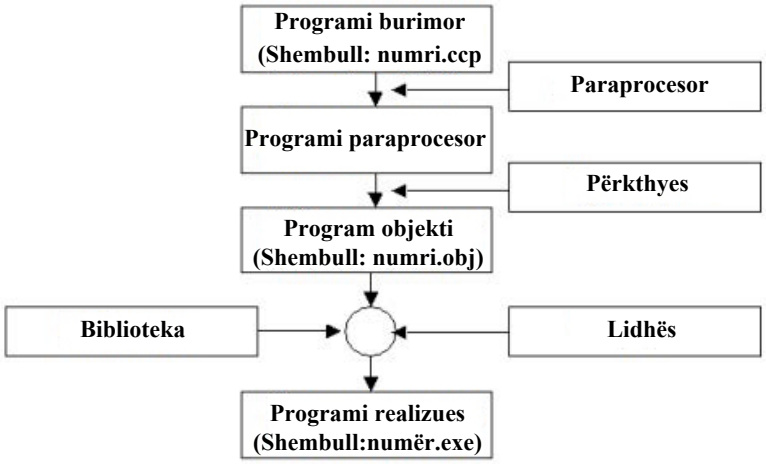


Figura 1.5.1

Në shumicën e rrethinave të integruara për programim zakonisht ka meny për ndihmë që jep mundësi për kontrolle të lidhura me rregulla të gjuhës programore dhe puna me vet rrethinën për programim.

Në vazhdim, do të sqarojmë rrethina zhvillimore të integruara Micro-soft Visual Studio 2019 dhe Code::Bloks.

Rrethina e integruar për programimin Microsoft Visual Studio 2019

Microsoft Visual Studio 2019 (MVS 2019) mund të merret prej shumë faqeve të internetit.

Pas instalimit dhe startimit, do të fitohet figura e paraqitur e MVS 2019.

Pastaj, paraqitet menyja për zgjedhje të mënyrës së startimit të MVS 2019, ***figura 1.5.3.***

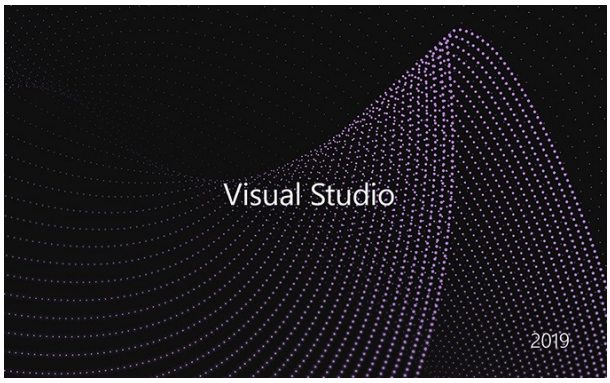


Figura 1.5.2

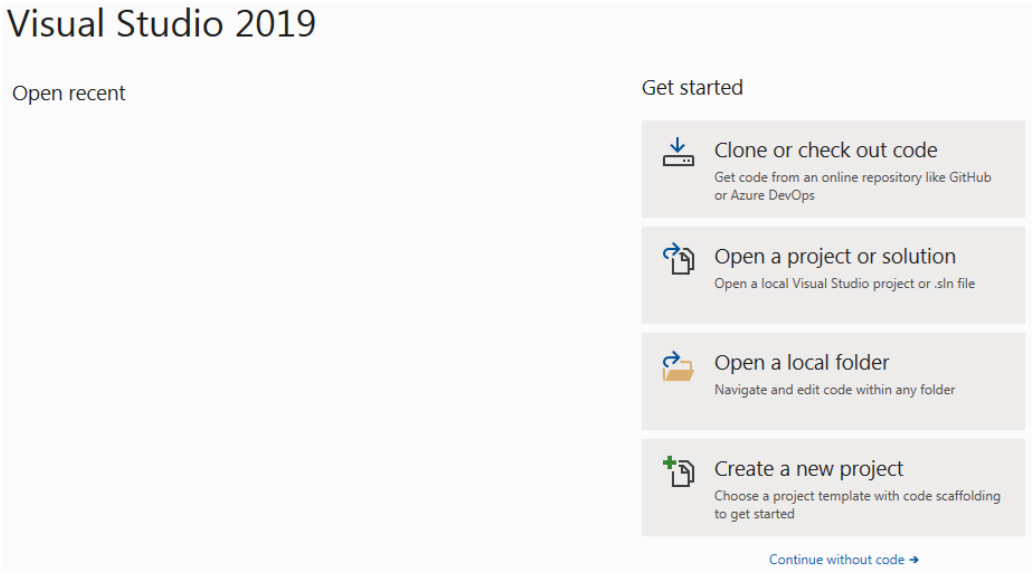


Figura 1.5.3

Fillestarët shpesh i zgjedhin mënyrën e dytë dhe të katërt të startimit.

Me zgjedhjen e mënyrës së katër, hapet menyja për krijimin e projektit të ri, ***figura 1.5.4***. Për fillestarët më e mire është projekt i zbrazët (Empty Projekt), ose aplikacioni komzol (Console App), ku janë paraqitur edhe shembull të programit në C++. Pastaj, klikohet në butonin Next.

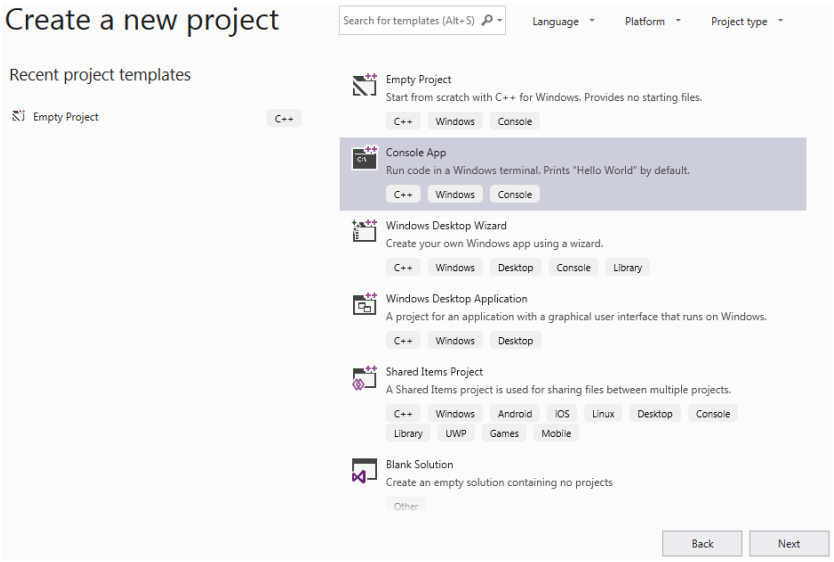


Figura 1.5.4

Hapet dritarja për konfigurim të projektit të ri, ***figura 1.5.5.***

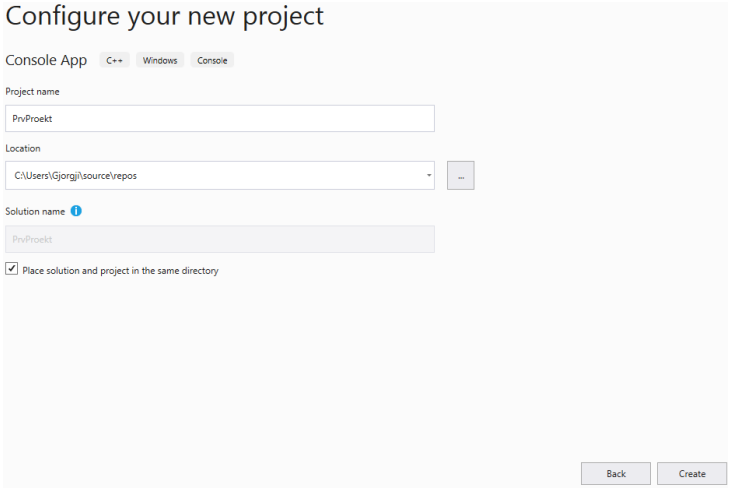


Figura 1.5.5

Te ai e futim emrin e projektit të ri, për shembull ProjetNjë dhe do ta shënojmë (me klikim te ai) katrori para Place solution and projekt in the same direktory.

Pas klikimit të butonit Create, fillon krijimi i projektit dhe hapet dritarja prej rrethinë zhvillimore të integruar të MVS 2019, ***figura 1.5.6***. Në atë tregohen edhe program në C++ nën emrin sikurse edhe projekti, ProjektiNjë.cpp. Mbi dhe nën programin ka më shumë sqarime, të cilat këtu nuk do t'i shqyrtojmë.

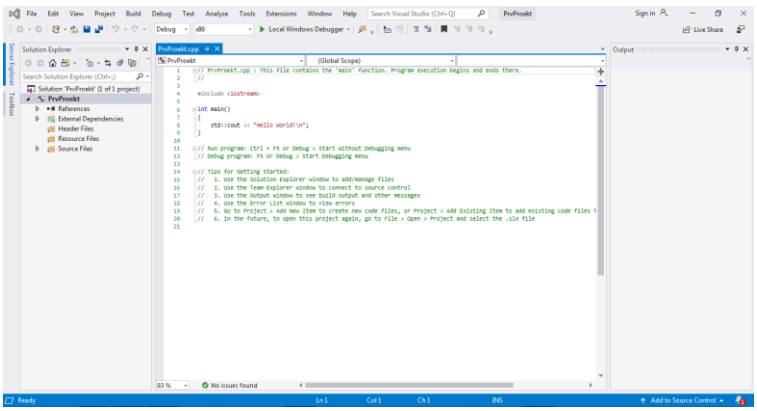


Figura 1.5.6

Nëse nuk duam të paraqitet program, atëherë prej menysë të ***figura 1.5.4*** duhet të zgjedhim Empty Projekt.

Programi ProjektiNjë.cpp mbyllet me klikimin mbi shenjën x sipas emrit.

Te dritarja Solution Explorer paraqiten projektet të krijuara në PIRr të MVS 2019. Kështu, te ***figura 1.5.6*** paraqitet direktoriumi i projektit ProjektiNjë me nëndirektoriumet e tij.

Me klikun e djathtë ose mbi nëndirektoriumin Source Files prej dritares Solution Explorer dhe zgjedhjes së komandës New Item… prej nënmenysë Add (***figura 1.5.7***), hapet dritarja Add New Item – ProjektiNjë, ***figura 1.5.8***.

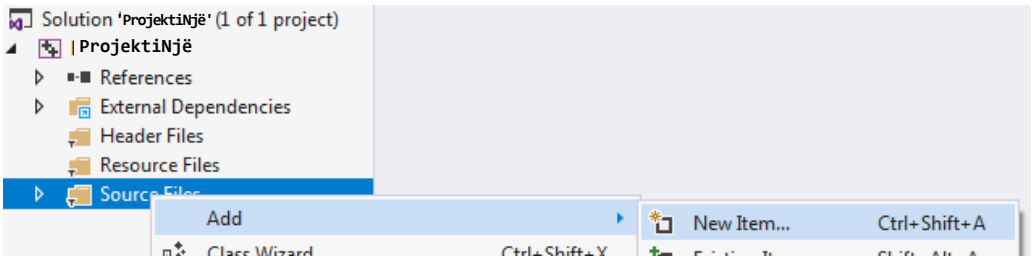


Figura 1.5.77

Te korniza e djathtë klikohet te C++ File (.cpp), por pastaj te fusha Name (te pjesa e poshtme) futet emri i programit, për shembull, Përshëndetje.cpp.

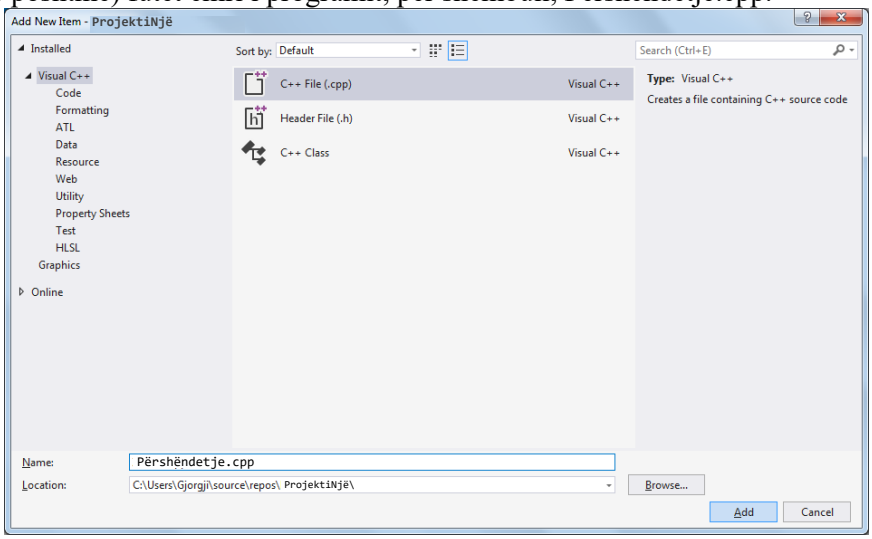


Figura 1.5.8

Me klikimin e butonit Add, hapet dritarja për të shkruar dhe redaktuar të programit Përshëndetje.cpp, ***figura 1.5.9***.

Dritarja kryesore

Te ***figura 1.5.9*** është paraqitur dritarja kryesore d.m.th., rrethina themelore për programim në MVS. Ajo i përmban këto dritare:

- Dritare me projekt (Solution Explorer).
- Dritare për të shkruar dhe redaktuar te programet – redaktues.
- Dritarja për paraqitje të rezultateve – dritare dalëse (Output).

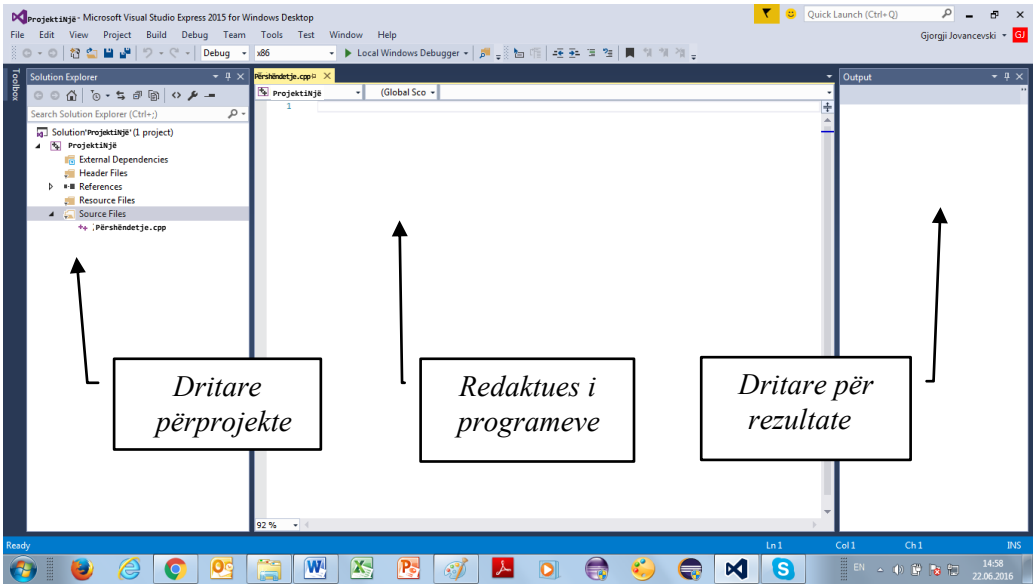
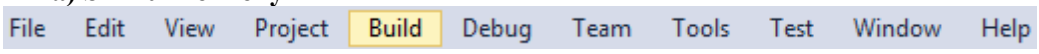


Figura 1.5.9

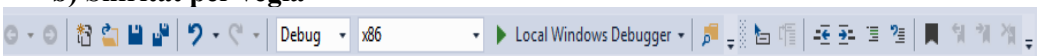
Elementet themelore të dritares kryesore janë:

a) Shiriti me meny



Shiriti me meny rënëse gjendet te pjesa më e lartë e dritares, menjëherë nën titullin e saj. Në atë gjenden menyja File, Edit, View, Projekt, Bu-ild, Debug, Tools, etj. Çdo meny përmban urdhëra për rrethinën dhe/ose meny të tjera.

b) Shiritat për vegla



Shiritat me vegla (buton për startimin më së shpeshti të komandet e shfrytëzuara të rrethinës) gjenden menjëherë nën shiritin me meny rënëse. Veglat te rrethina për programim janë aktive ose jo aktive (të zbehta). Veglat nëpërmjet të cilave jepen shumë komanda dthtas deri tea to kanë shigjetë poshtë ose me klikim në atë, ndarë në grupe, varësisht prej doethënies së tyre.

hapet menyja rënëse. Nëse klikohet në vet veglën, atëherë hapet dritarja nëpërmjet të cilës jepen komandat e njëjta.

Programi i parë C++

Te redaktimi prej ***figura 1.5.9*** mundemi ta shkruajmë edhe redaktojmë programin Përshëndetje.cpp, ***figura 1.5.10***.

Me komandën Compile prej nënmenysë Build (ose me Ctrl + F7), përkthehet program dhe te dalja e dritares (Output) fitojmë porosinën (***figura 1.5.11***) programi a është përkthyer me sukses.

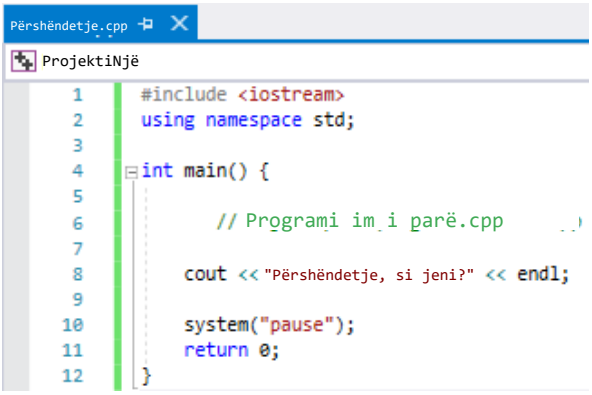


Figura 1.5.10

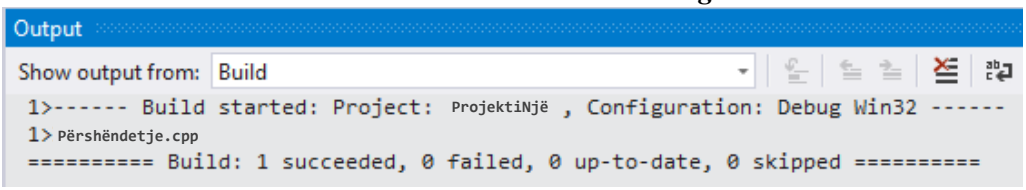


Figura 1.5.11

Me komandën Start Without Debugging prej nënmenysë Debug (ose me Ctrl + F5), realizohet program dhe rezultati tregohet te dritarja prej ***figura 1.5.12.***

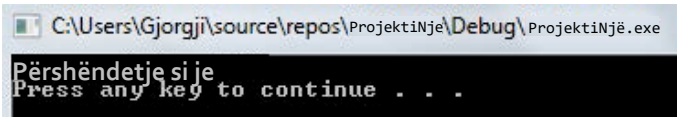


Figura 1.5.12

Për mbyllojen e dritare me rezultate, shtypet çfarëdo buton prej tastaturës.

Për ta ruajtur programin në disk, e japim komandën Save prej nënmenysë File (se Ctrl + S). Për ta ruajtur programin në disk me tjetër emër, e japim komandën Save As prej nënmenysë File.

Në një projekt mund të ketë më shumë programe të shkruara si datoteka. E dimë se çdo datotekë ka emrin dhe prapashtesën. Programet burimore të shkruara me gjuhën C++ kanë prapashtesë.cpp. Domethënë, emrin e çdo programi në C++ mund të jetë: P1.cpp, prog.cpp, ProjektiNjë.cpp, ProgramiIm. cpp etj.

Ushtrime

Ushtrimi 1.5.1

a) Programi Përshëndetje.cpp plotësoni të shikohet si te ***figura 1.5.15.***

Gjatë startimit të MVS 2019 pas herës së dytë (dhe çdo here vijuese), paraqitet faqeja MVS 2019, ***figura 1.5.3***. Nëse duam të hapim ndonjë projekt ekzistues, duhet të klikojmë te linku Open por projekt or solution. Do të hapet dritarja prej ***figura 1.5.13*** për zgjedhje të projektit.

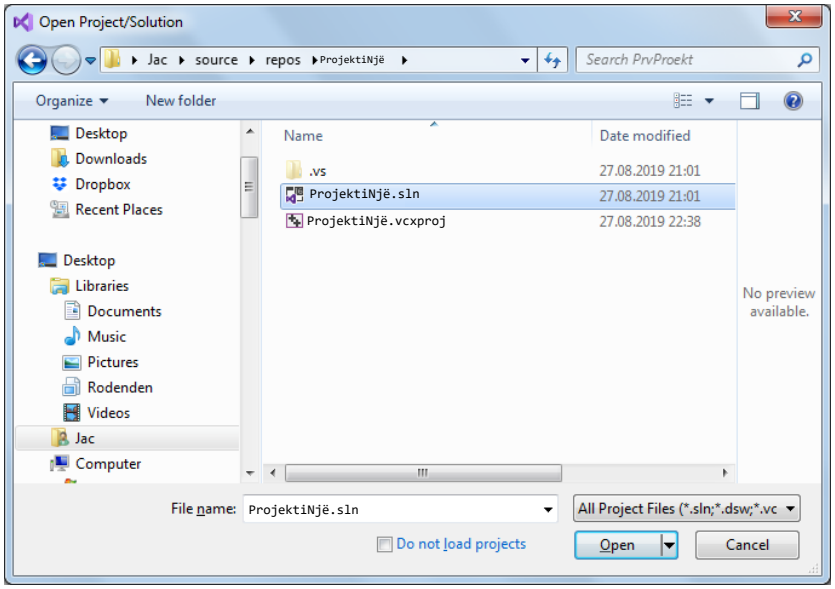


Figura 1.5.13

Me klikimin e ndonjërit projekt, për shembull ProjektiNjë.sln, ai hapet në dritaren Solution Explorer dhe oven të gjitha programet në atë, ***figura 1.5.14.***

Me klikimin e dyfishtë mbi emrin e ndonjë program (për shembull, të Përshëndetje.cpp), ai do të hapet te redaktuesi. Atë e plotësojmë programin Përshëndetje.cpp, sikurse është treguar te ***figura 1.5.15.***

b) Ruaje programin me komandën Save (ose Ctrl + S).

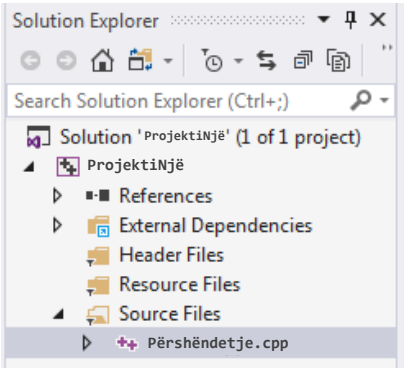
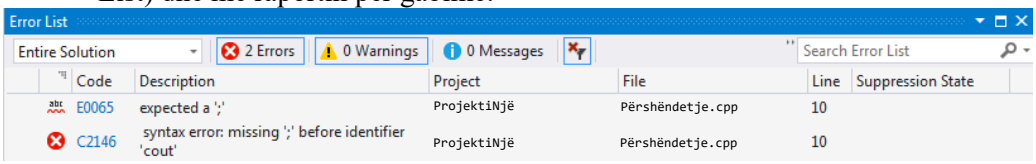


Figura 1.5.14

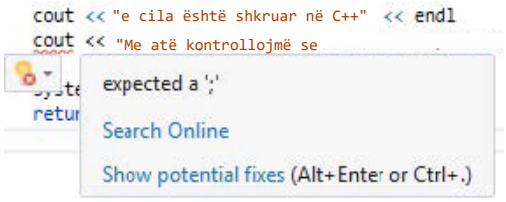
```cpp
#include <iostream>
using namespace std;

int main() {

    // Programi im i parë Përshëndetje.cpp

    cout << '"Përshëndetje, si je?" << endl;
    cout << "Ky është shembull për program" << endl;
    cout << "që është shkruar në C++" << endl;
    cout << "Me atë kontrollojmë MVS 2019 a punon mirë" << endl;

    system( "pause" );
    return 0;
}
```

Figura 1.5.15

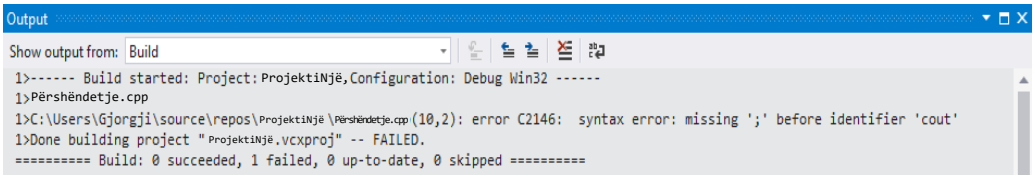
c) Nëse gjatë shkruarjes të programit keni gabuar ndonjë urdhër, gjatë përkthimit të tij, do të paraqitet porosia për gabime te dritarja dalëse. Për shembull, nëse kemi harruar të vendojmë pikëpresje në fund prej urdhrit të fundit, gjatë përkthimit të programit, do të hapet dritarja me listë me gabime (angl. Error List) dhe me raportin për gabime.



Vërejmë se urdhri pas asaj te e cila ka ndodhur gabimi është nënvizuar me vijë valore. Nëse sjellim treguesin mbi vijën, do të tregohet korniza me porosi për gabim.



Te dritarja dalëse (Output) paraqitet urdhri për llojin dhe vendin e gabimit. Pas vendosjes së pikës, programi do të punojë në mënyrë korrekte.



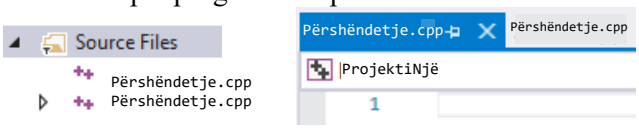
ç) Vijat te program të cilat fillojnë me dy vija të pjerrëta //, shërbejnë për futjen e komenteve te program dhe ato nuk realizohen. Mirë është çdo program në fillim të

ketë komentim për atë që punon. Këtë do ta praktikojmë edhe ne.

Ushtrimi 1.5.2

a) Krijoni edhe një program në projektin ProjektiNjë, për shembull, me emrin Përshëndetje.cpp.

Programi është krijuar me mënrën e njëjtë sikurse edhe program Përshëndetje.cpp, d.m.th., me klikune djathtë mbi Source Files dhe zgjedhja e Add ▶ New Item… Te dritarja që do të hapet (Add New Item – ProjektiNjë) klikohet te C++ File (.cpp) dhe në fushën Name (te pjesa e poshtme prej dritares) futet emri Përshëndetje.cpp. Në fund, klikohet te butoni Add. Te dritarja Solution Explorer, te nëndirektoriumin Source Files do të tregohet emri i programit të ri. Gjithashtu, te redaktuesi do të hapet programi i ri për të shkruar dhe redaktuar.

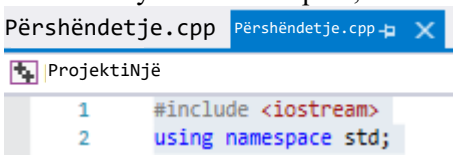


Vërejmë se edhe program Përshëndetje.cpp është hapur te redaktuesi.

Kalimi prej një program në tjetër te redaktuesi kryhet me klikimin mbi emrin e programit.

Për të mbyllur ndonjë program të hapur te redaktuesi, jepet komanda Close prej nënmenysë File ose klikohet mbi shenjën x djathtas prej emrit të programit.

b) Klikoni mbi titullin prej programit Përshëndetje.cpp dhe ai do të hapet. Selektoni dy komandat e para,



kopjoni me Ctrl + C, klikoni mbi emrin e programit Përshëndetje.cpp dhe vendosni edhe me Ctrl + V.

Vendoni një vjë të zbrazët me Enter dhe shkruani funksionin kryesor me komandat për komentim dhe për shtypjen e porosisë, ***figura 1.5.16.***

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // Programi im i dytë
6
7       cout << "Emri im është..." << endl;
8       cout << "Mbiemri im është..."  << endl;
9       cout <<"Datëlindja jeme është më..."<< endl;
10
11      system("pause");
12      return 0;
13  }
```

Figura 1.5.16

Ruani programin me Ctrl + S, përktheni me Ctrl + F7 dhe realizoni me Ctrl + F5.

Do të vëreni se gjatë realizimit, paraqitet porosia për gabim te dritarja vijuese.

[10]

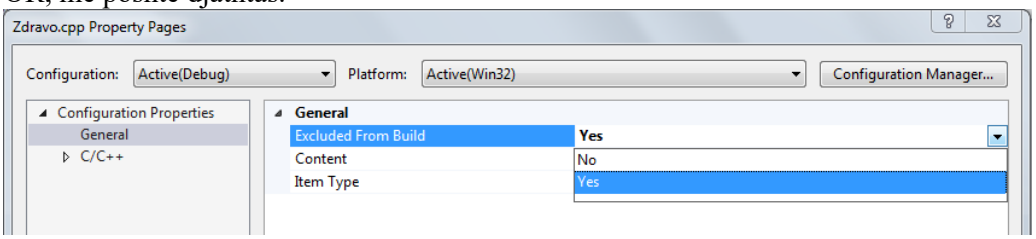
Porosia është për atë se gabimi është paraqitur gjatë ndërtimit të projektit. Në një projekt mund të ketë më shumë programe të cilat lidhen në një program realizues (themi se „ndërtohet projekti“), por patjetër të ketë vetëm një program me të cilin fillon realizimit të projektit. Ky është program që e përmban funksionin main() dhe quhet **aplikacioni** (angl. application). Pasi te projekti jonë ProjektiNjë ka dy programe të cilët e përmbajnë funksion main(), paraqitet gabimi i përmendur.

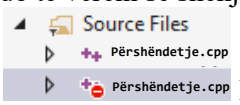
Kjo zgjidhet në mënyra të ndryshme, por njëra është i lehtë të përjashtohen prej projektit të gjitha programet të cilat e përmbajnë funksionin main(), përveç atij program që duam të jetë aplikimi dhe prej të cilit do të fillon realizimi i projektit.

Për ta përjashtuar programin Përshëndetje.cpp, do të klikojmë mbi atë me klikun e djathtë dhe prej dritares do të hapet (Open) e jep komandën:

[10] Nësese klikohet te Yes, do të kryhet program i fundit i realizuar me sukses. Nëse klikohet te No, do të realizohet program i fundit pa sukses.

Do të hapet dritarja Përshëndetje.cpp Property Pages edhe te menyja Excluded From Build prej kornizës General zgjedhet Yes. Pastaj, klikohet te butonët Apply dhe OK, më poshtë djathtas.



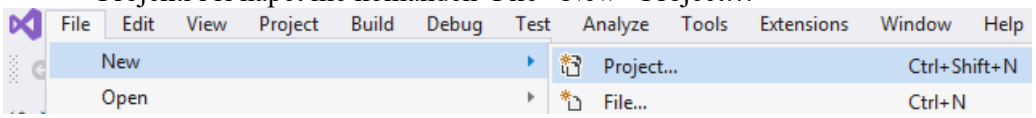
Te dritarja Solution Explorer do të vëreni se shenjat ++ para emrit Përshëndetje.cpp janë ndërruar te+–. 

Kjo do të thotë se programi Përshëndetje.cpp është përjashtuar prej projektit dhe se vetëm program Përshëndetje.cpp është aplikimi dhe mund të realizohet.

Ushtrimi 1.5.3

a) Krijoni projekt të ri (te shembulli, ProjektiDy) dhe te ai një program.

Projekti i ri hapet me komandën File ▸ New ▸ Project…



Do të hapet dritarja prej ***figura 1.5.4*** dhe linkohet te linku Empty Projekt ose Console App, sikurse është sqaruar te ***figura 1.5.4*** dhe ***figura 1.5.5***. Te fusha Name prej figura 1.5.5 (te faqes së poshtme) shkruhet emri i projektit, për shembull, ProjektiDy. Në fund, klikhet te butoni Create.

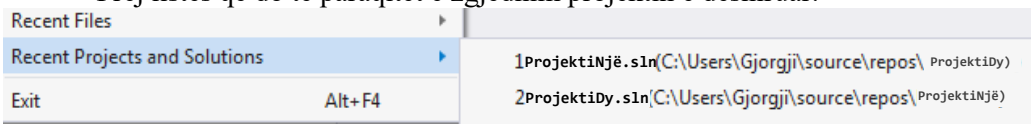
Te dritarja Solution Explorer do të hapet projekti dytë, por i pari do të mbyllet.

Programi te projekti hapet, sipas mënyrë së përmendur te ***Ushtrimi 1.5.2 a.***

Nëse duam të kalojmë te projekti parë (ose te ndonjë tjetër i hapur), atë e zgjedhim prej listës me projekti të krijuara me komandat:

File ▶ Recent Projects and Solutions

Prej listës që do të paraqitet e zgjedhim projektin e dëshiruar.



Rrethina zhvillimore e integruar Code::Blocks

Rrethina e integruar për programimin Code::Blocks mund të gjendet në adresën e internetit: http://www.codeblocks.org/downloads.

Në pjesën qëndrore të faqes ka tre linke: **Download the binary release, Download the source code** dhe **Retrieve source code from SVN**. Për instalimin e thjeshtë, preferohet të zgjedhet linku i parë **Download the binary release**. Pas klikimit të këtij linku,hapet faqeja e re te e cila ka ofruar marrjen e Code::Blocks për sistemin operativ nën të cilin punon kompjuteri Juaj.Për fillestarët preferohet ta shkarkojnë versionin që kyç MinGW setup. Në kohën e të shkruarit e këtij teksti, ai është linku codeblocks-20.03mingw-setup.exe i cili është dedikuar për shfrytëzuesit e të gjithë sistemeve operative Windows. Pas klikimit te lokacioni prej ku duani ta merrni Code::Blocks (për shembull **Sourceforge.net**), hapet faqeja e re, e cila pas kalimit të 5 sekondave vet ofron opsion të ruhet datoteka në lokalitetin të zgjedur prej shfryëzuesit. Pas ruajtjes së datotekës, duhet të përcillen instruksionet për instalim.

Startimi

Gjatë instalimit të Code::Blocks te menyja Programs prej menysë themelore Start, krijohet grupi i programeve me emrin CodeBlocks. Për të startuar rrethina e integruar për programim, duhet të zgjedhet CodeBlocks prej këtij grupi.

Dritarja kryesore

Te dritarja kryesore (***Figura 1.5.17***) e CodeBlocks ka vegla për ruajtje, kërkim, përkthim dhe realizim të programeve, vegla për dibagim etj. Dritarja punuese do t'i sqarojmë në vazhdim.

a) Shiriti me meny

Shiriti me meny gjendet te pjesa më e madhe e dritares, menjëherë nën titullin e tij. Në atë gjenden menyja File, Edit, View, Search, Projekt, Bu-ild, Debug, Fortran, wxSmith, Tools, Tools+, Plugging, Settings, DoxyBlocks dhe Help. Çdo meny përmban urdhëra për rrethinën dhe/ose meny të tjera. Domethënia e shumicës së tyre do të shqyrtohet më tutje.

b) Shiritat për vegla

Shiritat me vegla (buton për startimin më së shpeshti të komandave të shfrytëzuara të rrethinës) gjenden menjëherë nën shiritin me meny rënëse. Veglat te rrethina për programim Code::Blocks janë ndarë në grupe, varësisht prej domethënies

domethënies së tyre.Të treguarit dhe fshehja e grupeve të veglave të ndryshme janë realizuar në mënyrë të ngjashme sikurse te programet e Microsoft Office, me zgjedhjen e opsioneve View ▸ Toolbars. Çdo grup ka emrin e vet dhe nëse para emrit e ka shenjën për shënim, grupi shikohet në ekran, por nëse nuk e ka atë shenjë, ajo nuk shikohet. Çdo vegël korrespondon në disa prej komandave të menys.

Te dritarja kryesore gjendet dritaret e aplikacioneve të rrethinës. Më të rëndësishmet prej tyre janë:

- Te dritarja për redaktim të tekstit – redaktues. Te ai mund të ketë disa programe, varësisht prej asaj në sa programe punon programuesi në momentin e dhënë. Emri i programit është treguar te pjesa e sipërme e dritares për redaktim. Emri i programit që është aktiv në moment është shënuar ndryshe prej emrave të programeve të tjera.
- Dritarja për rezultate – dritare dalëse. Te ai programet japin informata për punën e vet gjatë përkthimit, lidhja, startimi etj. Nëse gjatë përkthimit të jenë zbuluar gabime, atëherë ato tregohen te korniza Build messages të kësaj dritare.
- Dritarja për organizim të punës së programit.
  Dritaret mund të mbyllen, të zmadhohen ose zvogëlohen etj.

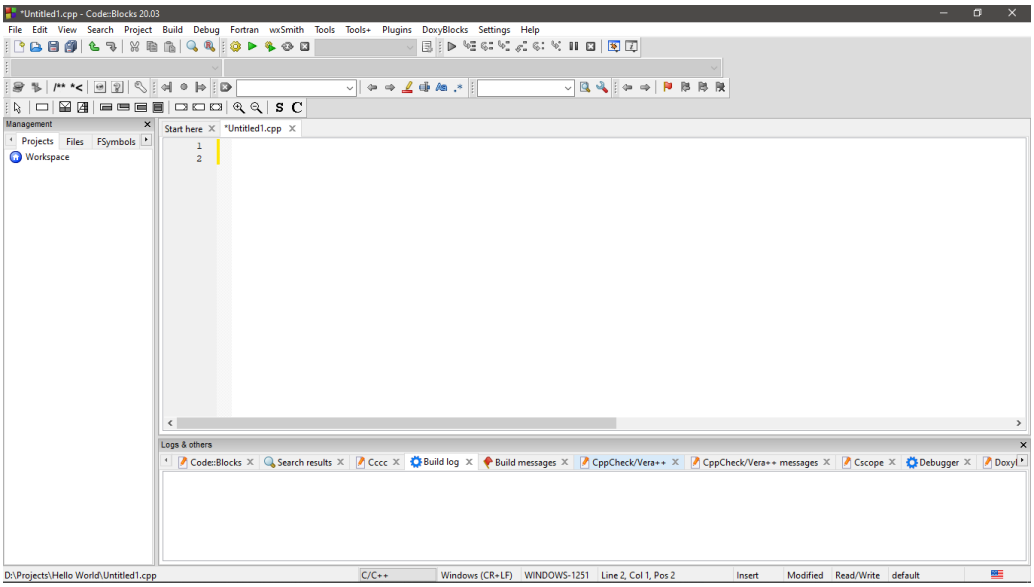


Figura 1.5.17

Redaktuesi i tekstit

a) Futja dhe përpunimi i tekstit

Redaktuesi i tekstit i ka të gjitha veglat standarde për futje dhe redaktim të tekstit. Gjatë redaktimit të tekstit, mund t'i shfrytëzojmë pothuajse të njëjtat urdhëra të cilat i kanë redaktuesit e tekstit te programet për punë në zyrë, (sikurse Microsoft Office Word, OpenOffice Writer dhe të tjera). Gjithashtu, vlejnë urdhërat e njëjta për lëvizje të treguesit prej miut.

b) Selektimi, kopjimi dhe zhvendosja e tekstit

Këto operacone janë dhënë nëpërmjet komandës menysë Edit.

c) Kërkimi dhe zëvendësimi

Në shumicën e rasteve mund të ketë nevojë prej kërkesës së tekstit të caktuar nëpër programin, por pas gjetjes, dhe zëvendësimi i tij me tjetër tekst. Ajo realizohet me ndihmën e urdhërave te menyja Search, Find, Replace etj.

Përkthimi dhe realizimi i programeve

Menyja Build përmban urdhëra për përkthim dhe realizim të programeve. Programi përkthehet me komandën Build ▸ Build (ose Ctrl + F9). Si rezultat te dosja te e cila gjendet datoteka me program burimor, krijohet program përkatës. Programi i realizuar starton Run (ose F9).

Për ta realizuar programin, sistemi operativ hap dritare tekstuale, te e cila futen të dhënat hyrëse dhe tregohen rezultatet prej realizimit të programit.

Datotekat të krijuara prej rrethinës

- Datotekat me prapashtesë.cpp i përmbajnë programet burimore dhe fitojnë pas realizimin e komandës File ▸ Save ose Ctrl + S.
- Datotekat me prapashtesë.exe përmbajnë program realizuese ekuivalent të burimores. Ato krijohen gjatë përkthimit të programit burimor.

Vijimi i realizimit të programit

Vijimi i realizimit të programit është mënyrë shumë e rëndësishme për kërksë për gabime logjike te programi. Rrethina Code::Blocks jep mundësi për vijim të atillë. Për këtë qëllim, mund të tregojë rreshta në tekst (të ashtuquajtura pika kontrolluese), te të cilat duam të ndalon realizimi i programit, si edhe ndryshoret përmbajtjen e të cilës duam ta vështrojmë. Kur fillojmë me realizimin e programit në regjim të vijimit, ai punon normalisht ndërsa nuk arrin deri te pika kontrolluese, por pastaj ndalon.

Në atë moment, programuesit mund ta shqyrtojnë përmbajtjen e ndryshoreve të zgjedhura për vështrim. Pastaj, vazhdon realizimi i programit, dhe programuesi mund t'i përcjell vlerat e ndryshoreve të vështruara dhe të kontrollojë se a u përgjigjen vlerave të pritura. Gjithashtu, programuesi mund të len programin të realizojë në mënyrë rregullore deri te pika kontrolli vijuese. Të gjitha komandat e lidhura me vijimin e programit gjatë realizimit janë te menyja Debug.

Vendosja e pikës së kontrollit bëhet kur do të vendoset tregusi i miut te vija e zgjedhur dhe kur do të jepet urdhri Debug ▸ Toggle Breakpoint (ose F5). Do të vëreni se majtas prej vijës vendoset pika e kuqe. Me klikimim e miut te pika e kuqe, ajo largohet. Vijimi i programit fillon me dhënien e urdhrin Debug -> Start. Programi realizohet drejtë deri sa pika kontrolluese ku do të ndalohet, duke vendosur një shigjetë të verdhë te vija. Prej atje vijimi i programit vazhdon nëpërmjet urdhërave të tjera prej menysë, edhe atë:

Debug – Next Step (realizohet vetëm urdhri, pasi që shigjeta e verdhë kalon në vijën vijuese).

Continue (realizohet program deri te pika kontrolluese vijuese ose deri në fund nëse nuk ka pikë kontrolluese).

Step into (shfrytëzohet vetëm për urdhëra te të cilët thirret funksioni dhe është e nevojshme vijimi i funksionit të thirrur).

Vijimi ndërpritet me urdhrin Debug ▸ Stop Debugger.

Gjatë vijimit të realizimit të programir, kjo vijë nuk do të jetë pika kontrolluese, përveç nëse nuk e vendosim treguesin e miut përsëri te ajo dhe nuk e japim komandën Debug ▸ Run to Cursor. Më tej para se të jepet komanda, treguesi duhet të vendoset në fillim të vijës prej të cilës duam të vazhdon vijimi i realizimit të programit.

Ushtrimi

Ushtrimi 1.5.4

Startoni CodeBlocks. Prej menysë zgjedhni krijimin e projektit të ri: File ▸ New ▸ Project ▸ Empty Project ▸ Go, ***figura 1.5.18***.

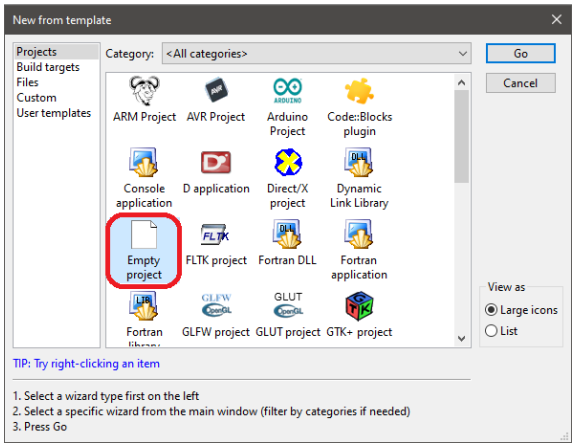


Figura 1.5.18

Te dritarja vijuese (***figura 1.5.19***) futni „Projekt Titëe", „Folder", „Projekt filename" dhe „Resulting file Name". Për shembull, për titull të projektit futni Projektiparë, por të tjerët fusha leni sikurse që janë.

Pas klikimit Next, te dritarja që hapet (***figura 1.5.20***) zgjidhni GNU GCC Compiler dhe shëno (me klikim

te katrorët (✓)) këto dy opsione për të krijuar konfiguracione „debug“ dhe „release“. Klikoni te butoni Finish.

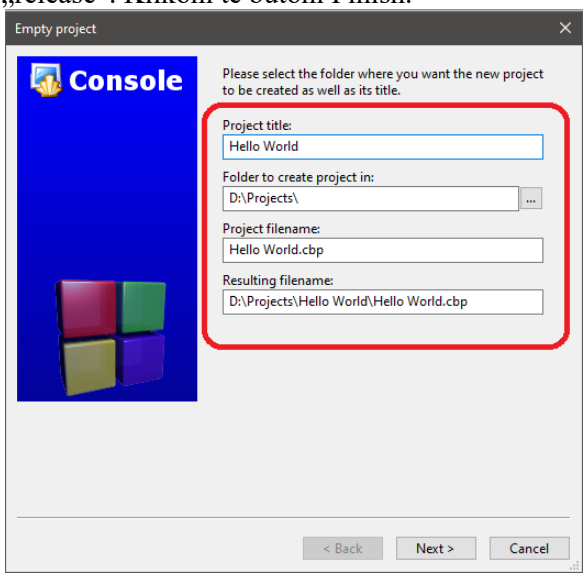


Figura 1.5.19

Shtoni datotekë burimore te projekti: File ▸ New ▸ File ▸ C/ C++ Source. Pastaj futni emrin e datotekës me shtegun e plotë (për shembull Ushtrimi1) te dosja e veçantë ku është projekti dhe mos harroni ta kyçni „Add file to aktive projekt“.

Për çdo projekt, duhet të vendoset n këto opsione: „Project ▸ Build Options.. ▸ Compiler Flags“.

Me këtë krijuat një projekt që në veti (për tani) përmban një datotekë të zbrazët, e cila është e gatshme në atë ta shkruani Programi juaj i parë.

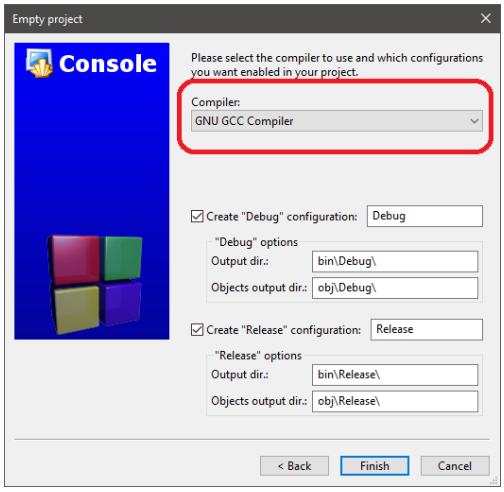


Figura 1.5.20

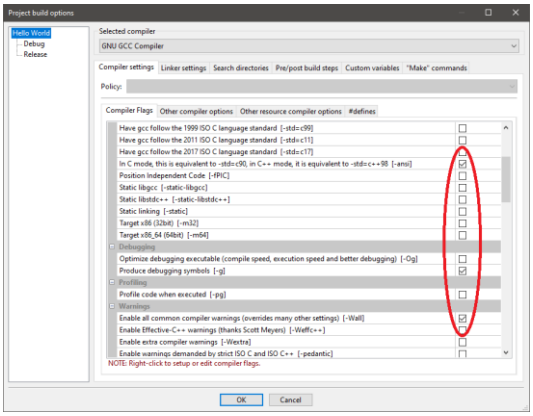


Figura 1.5.21

Ushtrimi 1.5.5

a) Te datoteka e hapur prej **Ushtrimit 1.5.4.** futni tekstin e këtij programi:

```
#include <iostream>
using namespace std;

int main()
{
	cout << "Ky është shembull për program";
	cout << "e cila është shkruar në C++.";
	cout << "Me atë e kontrolloj CodeBlocks a punon!\n";
	return 0;
}
```

Figura 1.5.22

b) Ruaje programin me urdhrin File ▸ Save, për shembull, me emrin Ushtrimi1.cpp.

c) Përktheni programin Ushtrimi1.cpp me kontrollin e butonit Ctrl + F9.

ç)) Nëse gjatë futjes së programit Ushtrimi1.cpp, keni lëshuar diçka në lidhje me tekstin e dhënë a), largoni gabimet dhe ruani ndryshimet me shtypjen e Ctrl + S. Pastaj realizoni programin me shtypjen e butonit F9.

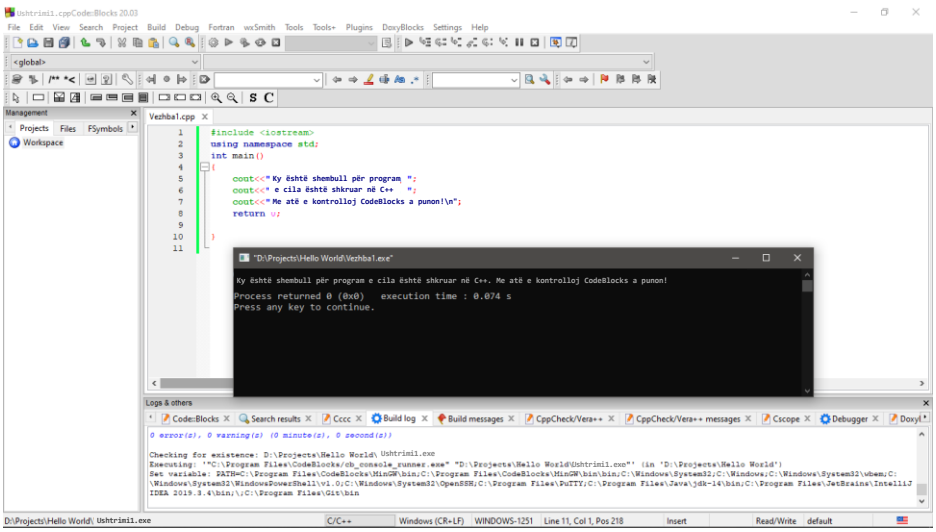


Figura 1.5.23

Ushtrimi 1.5.6

a) Krijoni projekt të ri dhe përsëri vendosni programin paraprak. Realizoni.
b) Me qëllim bëni ndonjë gabim te programi. Për shembull, në vend cout, shkruani caut në njërën prej urdhërave për shtypje. Përpiquni tani ta programin. Çka ndodh? Përmirësoni gabimin dhe përsëri realizoni programin.

c) Kopjoje vijën e katërt prej programit dhe vendosni sikurse te vija e pestë. Realizoni programin. Cili është rezultati prej realizimit?

Ushtrimi 1.5.7

Krijoni projekt të ri, shkruani këtë program (***figura 1.5.24***) dhe realizoni:

```
#include <iostream>
using namespace std;

int main()
{
	// Programi im i dytë në C++

	cout << "Emri im është... " << endl;
	cout << "Mbiemri është... " << endl;

	cout << "Ditëlindja ime është... " << endl;

}	return 0;
```

Figura 1.5.24

1.6 Hyrje në C++

Historia e shkurtër e C++

Në vitin 1972 laboratorët „AT&T Bell Labs“ është dizajnuar gjuha C nga ana e Dennis Ritchie (Denis Riçi). Te ai janë inkorporuar veti të cilat nuk i kanë pasur gjuhët programore të atëhershme:

- Lejon qasje deri te resurset në nivel shumë të ulët.
- Është i përshtatshëm për programim sistematik[11].
- Mund të realizohet në platformë të ndryshme kompjuterike.
- Punon nën sistemeve të ndryshme operative.

Në vitin 1980 te gjuha C janë mundësi të shtuara me të ashtuquajturën **klasa** dhe ai version i gjuhës C quhet „C me klasa“.

Zhvillimi i mëtejshëm i këtij versioni, përkatësisht përmirësimi i tyre, ka sjellë deri te gjuha e re në vitin 1983 që quhet C++, pasi paraqet përmirësim të C. C++ paraqet gjuhë për të ashtuquajturën **programimi objekt i orientuar** (angl. objekt-oriented programming).

C++ është dizajnuar prej Bjarne Stroustrup (Bjarne Stroustrup), por qëllimi është:

- të jetë gjuhë për qëllim të përgjithshëm prej gjuhës C,
- të mbështet **të dhëna të llojeve abstrakte**[12] (angl. abstract data types),
- të mbështet programimi objekt i orientuar.

Sot, C++ bën pjesë ndërmjet gjuhëve më të shfrytëzuara për përpunimin e programeve të llojeve të ndryshme, të cilët mund të ndryshojnë prej aplikimeve të dobishme për qëllime të ndryshme, pra deri te veglat sistemore.

Elementet e gjuhës C++

Gjuha C++ ka alfabet, i cili është bashkësia e simboleve të lejuara dhe atë:

- Shkronjat e vogla prej alfabetit anglez: a – z.
- Shkronjat e mëdha prej alfabetit anglez: A – Z.
- Shifrat: 0 – 9.
- Shenjat speciale: ~ ! @ # $ % ^ & * ( ) – + = { } [ ] : ; '...' "..." < > ? /.

[11] Programim në nivel të hardverit.

[12] Këto janë struktura me të cilat mund të pasqyrohet bota reale në program. Për shembull, në një program nund të përkufizon nxënësi, qarqe, drejtkëndësh etj., dhe me ato të realizohen operacione përkatëse, sikurse: njehsim të suksesit të nxënësit, syprina e drejtkëndëshit etj. Me lloje abstrakte të të dhënave punohet në programim objekt të orientuar.

Prej alfabetit të C++ formohen fjalët të cilat mund të jenë:

- Fjalë kyçe (angl. keywords)
- Numerike, simbolike dhe tekstuale konstante,
- Emra (identifikator),
- Operator.

Lloj i veçantë i simboleve janë ndarësit (separatorët) me të cilët ndahen fjalët njëra prej tjetrës. Ato janë:

- Vend i zbrazët (blanko).
- Rresht i ri.
- Tabulator.

Sikurse edhe të gjitha gjuhët e larta programore, ashtu edhe C++ e ka **gramatikën** e vet e cila i përmbana rregullat për ndërtimin e fjalëve dhe urdhërave. C++ Ka bashkësi të caktuar **fjalë të rezervuara**. Disa prej tyre kanë domethënie të veçantë për C++ dhe ato quhen **fjalë kyçe**. Programuesit mund të formjnë fjalë sipas rregullave të caktuara të quajtura **identifikator**. Fjalët e rezervuara nuk mund të jenë identifikator.

Me kmbinime saktë të përkufizuara të fjalëve dhe simboleve, konstruktohen urdhëra të gjuhës. Poashtu, shfrytëzohen rregulla rigoroze të quajtura rregulla sintaksore ose vetëm **sintaksa e** gjuhës. Poashtu përkthimi i programit, me rregullat sintaksore kontrollohet korrektësia e çdo rregulle.

Çdo rregull te program patjetër të ketë saktë domethënie të përkufizuar dhe të shkakton saktë veprime të përkufizuara. Domethënia (kuptimi) i urdhërave quhet **semantika**) e gjuhës.

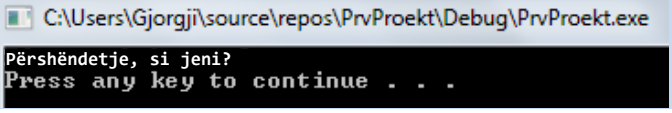
Për shembull, rregulla sintaksore në C++ për funksionin kryesor (main()) është:

```
int main() {
        urdhër;
        ...
        urdhër;
}
```

Programi në C++

Te ***figura 1.6.1*** është dhënë programi Përshëndetje.cpp që e krijum te pika **1.5 Rrethina zhvillimore e rientuar**, ***figura 1.5.10***.

Programi shtyp në ekran:



Do ta sqarojmë çdo vijë prej programit.

Vija 1: Dirktiva

Direktiva me të cilën kyçet biblioteka **iostream**, e cila përmban funksione për operacione hyrëse dhe për dalëse te programi.

Vija 2: Direktiva

Direktiva për formimin e hapësirës emërore (varg) të emrave.

Vija 3, 7 dhe 9: Vija e zbrazët.

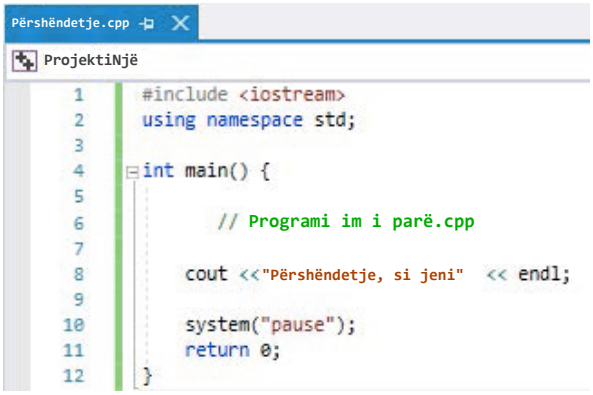


Figura 1.6.1

Vijat e zbrazëta shërbejnë për pamje më të madhe të programit, por përkthyesi i injoron.

Vija 4: Funksioni kryesor.

Programet në C++ përbëhen prej **funksioneve**[13]. Çdo program në C++ patjetër të ketë vetëm një funksion kryesor të quajtur main() dhe patjetër të jetë deklaruar, sikurse që është te shembulli. Fjala int do të thotë se funksioni main() kthen rezultatin numër të plotë. Më vonë do të shikojmë se mund të deklarohen edhe funksionet të cilat nuk kthejnë rezultat.

Vija 4: Pjesa hyrëse e kllapave të mëdha – fillimi i funksionit.

Pjesa hyrëse e kllapave të mëdha sipas funksionit main(), paraqet fillim zë **trupit të funksionit**. Trupi i çdo funksioni mbaron me pjesën dalëse prej kllapave të mëdha, në shembullin tonë, ai është pjesa dalëse prej kllapave te *Vija 12*. Vijat e trupit të funksionit (*Vija 6 deri te Vija 11*) futen djathtas dhe atë e quajmë **dhëmbëzim** (angl. indentation).

Pjesa hyrëse prej kllapave të mëdha me të cili fillon funksioni mund të vendohet edhe te kjo *Vijë 5*.

Vija 6: Komentim

```
// Programi im i parë Përshëndetje.cpp
```

Praktikë e mire programore është programet të komentohen. Prandaj, është e preferueshme në fillim të çdo programi të vendohet **komenti** për atë që punon program. Komentet janë të dobishme edhe në fillim të çdo funksioni.

Te program i dhënë, *Vija 6* nuk i tregon asgjë përkthyesit dhe ai nuk e përkthen. Prandaj, ajo vijë quhet komentim. Komentimi është tekst i cili është shfrytëzuar për atë i cili e lexon programin dhe është dedikuar për pamje më të madhe dhe të kuptuarit.

[13] **Funksioni** te gjuha programore është program i cili pas realizimit jep rezultate. Për shembull, funksioni për shumëzim të dy numrave, për pjesëtim të dy numrave, për gjetje të rrënjës katrore prej numrit etj.

Gjithashtu, komentimet shfrytëzohen edhe për programues të tjerë të cilët duhet ta përmirësojnë, ndryshojnë ose përmirësojnë programin. Shpesh ndodh programuesi të fut koment te program me qëllim të mund më lehtë ta kupton kur më vonë vet ai do ta shqyrton.

Vëreni se çdo koment fillon me dy viza të pjerrta //. Domethënë, çdo gjë që do të shkruhet pas // deri në fund të vijës paraqet koment.

Gjatë përkthimit të programit, përkthyesi i injron (nuk i përkthen) komentet.

Vija 8: Urdhri për shtyp.

```
cout << "Përshëndetje, si jeni? " << endl;
```

`cout`	– urdhri për shtypje
`<<`	– operatori për shtypje
`"Përshëndetje, si jeni?`	– tekst i cili shtypet
`" endl`	– urdhër për fund të vijës aktuale
`;`	– shenja për fund të urdhrit, përkatësisht ndarës të urdhërave.

Vërejmë se atë që e dum të shtypet e vendojmë në thonjëza. Shpesh, në kllapa vendohet ndonjë tekst, që paraqet **varg prej shenjave** dhe quhet **string**[14] (angl. string).

Vija 10: Urdhri për pause.

Urdhri për pause para së gjithash të mbyllet dritarja me rezultate. Ky urdhër nuk është i detyrueshëm, por është i dobishëm nëse duam t'i shikojmë rezultatet e deritanishme.

Vija 11: Urdhri për kthimin e vlerës

Nëse vlera që e kthen është 0 (sit e shembulli yn), domethënë se program në mënyrë korrekte realizohet. Nëse kthehet çfarëdo tjetër vlerë (1, 2...), domethënë se diçka ka qenë e gabueshme gjatë realizimit të programit. Ne do ta shfrytëzojmë urdhrin **return 0**.

Vija 12: Pjesa dalëse prej kllapave të mëdha për fund të funksionit main().

Vijat 1 dhe 2 patjetër të kyçen në çdo program C++.

Funksioni main() ka frmë standarde:

```
int main() {
        Urdhri;
        ...
        Urdhri;
        return 0;
}
```

[14] Më vonë d të sqarojmë çka janë stringe dhe si shfrytëzohen.

Programet në C++ mund të shkruhen me çfarëdo redaktues të tekstit. Pastaj, përkthehen dhe realizohen, varësisht prej rrethinë zhvillimore të integruar.

Urdhri për shtypje

Theksuam se urdhri për shtypje të stringeve ka formë:

```
cout << "Përshëndetje, si jeni?" << endl;
```

Domethënë, nëse duam të shkruajmë diçka në ekran, thjeshtë prej *Vijës 8* te program prej ***figura 1.6.1*** do të shtojmë edhe urdhëra, sikurse që është bërë te ***figura 1.6.2***, njëjtë me ***figura 1.5.15***.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5
6       //  Programi im i parë Përshëndetje.cpp
7
8       cout <<"Përshëndetje, si jeni"_<< endl;
9       cout <<"Ky është shembull për program"  << endl;
10      cout << "që është shkruar në C++"     << endl;
11      cout <<"Me këtë kontrollojmë se MVS 2019 a punon mirë"          << endl;
12
13      system( "pause" );
14      return 0;
15  }
```

Figura 1.6.2

Pas realizimit të programit, do ta fitojmë këtë dalje:

```
Përshëndetje, si jeni?
Ky është shembull për program
që është shkruar në C++
Me këtë kontrollojmë se MVS 2019 a punon mirë
Press any key to continue..
```

Vërejmë se çdo string te urdhërat shtypet në rresht të ri. Kjo arrihet në atë mënyrë që në fund urdhri cout vendohet në të ashtuquajturën manipulator endl[15].

Kjo do të thotë se urdhri vijues për shtypje e cila në fund e ka manipulatorin endl, do të shtype në vijën e re (rreshti i ri).

Për shtypjen në vazhdim në vijën e njëjtë, urdhri paraprak nuk duhet të ketë në fund endl.

Për shembull, nëse urdhri te Vija 9 prej ***figura 1.6.2*** në fund nuk ka endl, atëherë do të shtypet:

[15] Në C++ ekzistojnë më shumë manipulator (funksione ndihmëse) për kontroll të futjes dhe të shtypjes të të dhënave.

```
Përshëndetje, si jeni?
Ky është shembull për program që është shkruar në C++
Me këtë kontrollojmë se MVS 2019 a punon mirë
Press any key to continue . . .
```

Sikurse vërejmë, fjala e fundit prej urdhrit të dytë dhe fjala e parë prej urdhrit të tretë janë bashkuar. Për t'u ndarë duhet të lihet vend i zbrazët në fund të stringut prej urdhrit të dytë ose fillimit të stringut prej urdhrit të trtë.

Urdhërat te Vijat 8, 9, 10 dhe 11 ndonjëherë quhen edhe **gjykime**. Mund të vëreni se çdo gjykim mbaron me pikëpresje (;).

Vërejmë se urdhri për shtypje fillon me fjalën cout (lexohet „si aut"), pas të cilit vijjnë operatori për shtypje <<, të quajtur **operator për futje** (angl. insertion operator) dhe shprehja që shtyp. Në rastin tone, shprehja paraqet varg prej shenjave (string), që shtypet në pajisjen dalëse standard te kompjuteri, zakonisht në ekran.

Forma e përgjithshme për shtypje është:

```
cout << shprehje ;
```

Operatori << është operator binary operandi i majtë i të cilit është fjala cout, por operandi i djathtë është shprehja që duhet të shtypet.

Operatori << mund të shfrytëzon shumë herë në urdhrin e njëjtë. Për shembull:

```
cout << "Përshëndetje, si jeni? " << "Ky është shembull për program. " << endl;
```

Poashtu, të dy stringet do të shtypen në vijën e njëjtë:

```
Përshëndetje, si jeni? Ky është shembull për program
```

Në një vijë mund të shkruhen shumë urdhëra për shtypje, ku do të jenë ndarë me shenjën pikëpresje. Për shembull, nëse urdhërat te Vijat 9 dhe 10 prej programit te ***figura 1.6.2*** i shkruajmë në një vijë:

```
cout << "Ky është shembull për program. "; cout << "që është shkruar në  C++."
<< endl;
```

dalja do të jetë krejtësisht i njëjtë.

Me një urdhër për shtypje, mundemi të shtypim më shumë stringe, sikurse është treguar te ***figura 1.6.3***. Poashtu, nëse urdhri është i gjatë, mundemi ta hedhim te rreshti vijues me shtypjen e butonit Enter para ose pas operatorit <<.

Rezultati i realizimit të programit do të jetë:

```
Autor i librit Kalesh Angja është Stale Popov
Ose, me 2 urdhëra
Autor i librit Kalesh Angja është Stale Popov
ose, me më shumë urdhëra
Autor i librit Kalesh Angja është Stale Popov
Press any key to continue . . .
```

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {    // Shtypja e stringeve
5
6       cout << "Autor i librit"   << " " <<  "Kalesh Angja"
7           << "është" <<"Stale Popov" << endl;
8       cout <<"ose, me 2 urdhëra" << endl;
9       cout <<"Autor i librit"   << " " <<  "Kalesh Angja"
10      cout <<"është"<< "Stale Popov" << endl;
11      cout <<"ose, me më shumë urdhëra"<< endl;
12      cout <<"Autor i librit";
13      cout << "   ";
14      cout <<"Kalesh Angja";
15      cout << "është";
16      cout <<"Stale Popov" << endl;
17
18      system( "pause" );
19      return 0;
20  }
```

Figura 1.6.3

Pasi titujt e librave shkruhen në thonjëza, për t'u shtypur thonjëzat „...“, para tyre vendohet vija e pjerrët e kundërt, \ d.m.th., \"...". Stringu "Kalesh Angja" duhet të shkruhet "\"Kalesh Angja\"".

Gjithashtu, për shtypjen te vija e zbrazët jepet komanda: :

```
cout << endl;
```

Me kto ndryshime te program, dalja do të jetë:

```
Autor i librit    Kalesh Angja   është  Stale Popov
ose, me 2 urdhëra
Autor i librit    Kalesh Angja   është  Stale Popov
ose, me më shumë urdhëra
Autor i librit  Kalesh Angja   është  Stale Popov
Press any key to continue..
```

Shpesh, ndonjë string më i gjatë ta shtypim në më shumë vija. Për këtë qëllim, shfrytëzohet sekuenca speciale \n, e cila mundëson pas asaj të shtypet në vijën vijuese. Për shembull, nëse duam shtpim:

```
Autor i librit
"Kalesh Angja"
është
Stale Popov
```

kjo mund të bëhet me një urdhër:

```
cout << "Autor i librit \n\"Kalesh Angja\" \jo\nStale Popov" << endl;
```

Te urdhri vijues shfrytëzohen sekuencat \" për shtypje në thonjëza dhe \n për kalimin dhe shtypjen në vijë të re.

Në C++ ekzistojnë më shumë të ashtuquajtura **eskejp sekuenca** (angl. escape sequences).

Do të përmendim disa:

\n Kalimi në vijë të re.
\" Shtypja në thonjëza "...".
\' Shtypja në gjysmëthonjëza '...'
\t Kalimi në pozitën vijuese të tabulatorit.
\\ Shtypja në vijën e pjerrtë të kundërt \.

Urdhërat për shtypje në programin prej ***figura 1.6.3*** janë renditur njëra pas tjetrës. Ato saktë ashtu edhe do të jenë realizuar prej kompjuterit, së pari e para, pastaj e dyta, pastaj e treta etj., d.m.th., urdhërat do të jenë të realizuara sipas renditjes sikurse janë renditur. Struktura e këtillë prej urdhërave të njëpasnjëshme quhet **strukturë e renditur** ose **sekuenca** (angl. sequence).

Për shembull, me këtë program

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       //  Shembulli strukturë rendor-sekuenca
6
7       cout << "    *" << endl;
8       cout << "   * *" << endl;
9       cout << "  * * *" << endl;
10      cout << " * * * *" << endl;
11      cout << "* * * * *" << endl;
12      cout << " * * * *" << endl;
13      cout << "  * * *" << endl;
14      cout << "   * *" << endl;
15      cout << "    *" << endl;
16
17      cout << endl;
18      system("Color 17");
19      system("pause");
20      return 0;
21  }
```

Figura 1.6.4

me strukturën rendore të urdhërave, do të shtyp figurën:

```
    *
   * *
  * * *
 * * * *
* * * * *
 * * * *
  * * *
   * *
    *
Press any key to continue . . .
```

Te program e shtojmë urdhrin sistem („Color 17“); me të cilin vendoset ngjyra e kaltër prapa (numër 1) dhe ngjyra e bardhë te teksti (numri 7) për pamje më të këndshme të paraqitjes së rezultateve. (Lexuesi mund të eksperimenton me këtë urdhër

ku numri i parë është ngjyra e pjesës së prapme, por numri i dytë është ngjyra e tekstit. Numrat mund të jenë cilido kombinim i 16 numrave heksadhjetore të para: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F).

Te shembujt tonë do t'i shfrytëzojmë 4 urdhërat e fundit.

Detyra për ushtrime

1. Te program i ***figura 1.6.4***, te urdhri:
```
system("Color 17");
```
ndryshoni numrat e ngjyrës të pjesës së prapme (tani është 1) dhe numrat e ngjyrës së tekstit (tani është 7), të clët mund të jenë 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Zgjedhni kombinim që më së shumti Ju pëlqen.
2. Shkruani program që në ekran do ta japë këtë dalje:
```
     *
    * *
   *   *
  *     *
 *       *
* * * * *
```
3. Shkruani program që në ekran do ta shkruan emrin Tuaj me shenjën * se me tjetër shenjë.
4. Të shkruhet program që do t'i shtypë emrin Tuaj, por te rreshti i dytë do t'i shtyp emrin dhe mbiemrin Tuaj, por te rreshti i dytë do t'i shtyp numrin e viteve Tuaja.
5. Përmirësoni programin paraprak të shtyp sa muaj jeni rritur në vend sa vjet jeni rritur. Për shembull, nëse keni lindur në maj të vitit 2003, por tani jemi tror i vitit 2020, atëherë jen rritur (17 vjet) x (12 muaj) + (10 (tetor) – 5 (maj)) muaj.
6. Të shkruhet programi që do ta shtyp rezultatin prej shumëzimit të 1 234 me 5 678. Kontrolloni a keni fituar rezultat të saktë me ndihmën e kalkulatorit.
7. Të shkruhet programi që do ta shtyp rezultatin prej shumëzimit të 123 456 me 7 891 011. Çka keni fituar si rezultat? Ky rezultati a është i saktë?

Madhësitë dhe të dhënat

Theksuam se algoritmet janë mënyra për përpunimin e të dhënave, përkatësisht me ato përpunohen të dhënat hyrëse sipas renditjes rigoroze të shkruar dhe fitohen rezultatet dalëse. Me imponimin e pyetjes çfarë rezultate mund të përpunohen me algoritmet. Prandaj, të vërejmë çka janë të dhëna.

Madhësia është masa për ndonjë karakteristikë të lëndës ose dukurisë së dhënë. Për shembull, këto karakteristika kanë madhësi të vet: gjatësia, vëllimi, masa, ngjyra, mosha, shpejtësia etj. Madhësitë kanë vlera të cilat shprehen në masa ose vlera prej ndonjë bashkësie. Për shembull: cm është masa për gjatësi, m^3 është masa për

vëllim, kg është masa për masën, viti është masa për moshën, m/sek është masa për shpejtësinë etj.

Vlerat me të cilat shprehen madhësitë quhen **të dhëna**. Domethënë, të dhënat kanë vlerë. Për shembull, gjatësia e hapit të njeriut është 70 cm, vëllimi i kubit me brinjë 2 cm është 8 cm^3, mosha e profesorit të informatikës është 47 vjet etj. Të dhënat përpunohen me algoritme, d.m.th., programe.

Konstanta dhe ndryshoret

Në shumicën e rasteve e dhëna, d.m.th., vlera e ndonjë madhësie, nuk ndryshon. Për shembull, vlera e numrit $\pi = 3.14$ është fiks dhe prandaj themi se ai është **konstanta** (angl. konstant).Vlerat fikse quhen **literale** (angl. literals)[16]. Në çdo gjuhë programore ka literale të ndërtuara të cilat mund të shfrytëzohen si konstante te programet.

Për shembull, konstanta π në gjuhën programore C++ mund të shkruhet me PI dhe mund të shfrytëzohet te frmula[17], si:

```
perimetri  = 2 * r * PI;     syprina  = r * r * PI;    // * është shenjë për shumëzim
```

Por programuesit te programet mund të japin (emërtojnë) konstantet e veta. Për shembull, nëse te program shfrytëzohet emri i shtetit tonë, atëherë ajo do të jepet si konstante me emrin shteti dhe vlera `"Maqedonia"`[18]. Ose, mund të shfrytëzohet konstanta muaj, të cilat mund të pranon vlerë 12 ose ndonjë vlerë tjetër. Konstantat e këtilla, të cilat i përkufizojnë dhe emërtojnë programuesit dhe te të cilat mund t'u shoqërohet vlerë, quhen **konstante të emërtuara** (angl. named konstants).

Për dallim prej konstantëve, vlera e të dhënave për disa madhësi ndryshon. Për shembull, mosha e ndonjë njeriu, koha gjatë ditës, shpejtësia e ndonjë automobili etj., mund të fitojnë vlera të ndryshme dhe prandaj, themi se të dhënat e atyre madhësive janë ndryshore. Themi se vitet e vëllait tim janë 10, vitet e mija 17, vitet e qenit janë 5 etj. Domethënë, 10, 17 dhe 5 janë vlerat e të dhënës së moshës. Domethënë, mosha është e dhënë që mund të ketë vlera të ndryshueshme. Të dhënat e këtilla, të cilat mund të kenë vlera të ndryshme, quhen **ndryshore** (angl. variables).

Gjatë përpunimit të të dhënave me ndihmën e kompjuterit, konstantat dhe ndryshoret kanë emrat e tyre, Vlerat e tyre duhet të futen dhe të mahen mend

[16] Te programet mund të shfrytëzojmë literale të ndryshme, d.m.th., vlera fikse të të dhënave. Për shembull: Nxitimi i tokës, numri amë i qytetarit, emri i ndonjë qyteti etj.

[17] Konstantat në C++ shkruhen me shkronja të mëdha (të gjitha shkronjat janë të mëdha).

[18] Nëse literali është tekst, atëherë vendohet vetëm në thonjëza.

ndërsa përpunohen. Për shembull: numriPi, numriE, mosha, shpejtësi, ngjyra etj.

Nëse e dhëna është konstante, atëherë te lokacioni i memories nën emrin e dhënë (për shembull, numriPi) vlera nuk ndryshon për kohën e realizimit të programit.

Nëse, tani, e dhëna është ndryshore, atëherë vlera e lokacionit të memories te e cila është shkruar ndryshorja nën ndonjë emër (për shembull, mosha) ndryshon. Për shembull, mosha = 17, mosha = 20 etj. Themi se vlera e të dhënës është ndrshore dhe prandaj, e dhëna mosha quhet ndryshore.

Formulat matematikore shpesh kyçin në vete ndryshore. Themi se te funksioni y = f(x), x dhe y janë ndryshore pasi për dallim vlera e x fitohet vlerë e ndryshme e y. Ose, te formula për njehsimin e syprinës së drejtkëndëshit me brinjë a dhe b (S = a * b), brinjët a dhe b janë ndryshore. Ndonjëherë, te formulat janë kyçur edhe konstante. Për shembull, te formula për syprinën e rrethit është S = r * r * PI. Te formula mund të paraqitet edhe numër si konstante. Për shembull, V = 4 * r * r * r * PI / 3.

Me gjuhët programore mund të përpunohen të dhëna të ndryshme, vlerat e të cilave janë shprehur me:

- numra të plotë (… –2, –1, 0, 1, 2, 3…),
- numra real (… –2.34… –1.89… 0… 0.573… 123.45…),
- shenja[19] ('?', '*', '@', '7', '+'…),
- tekste[20] (`"Macedonia"`, `"Sot është shekulli 21"`, `"Informatika"`…),
- igura, zëri etj.

Themi se ato janë të dhëna prej **llojit** të ndryshëm (angl. type). Përveç llojit të dhënat kanë **emër** dhe **vlerë**. Prandaj, themi se të dhënat i kanë këto karakteristika:

- Emër.
- Lloj.
- Vlerë.

Sipas vlerës që u shoqërohet ndryshoreve, themi se ato janë të llojit. Për shembull, ndryshorja mosha (nëse shprehet në vite) është prej **lloj numër i plotë** pasi mund të marrë vlera vetëm numra të plotë: mosha = 18, mosha = 10, mosha = 123 etj. Ngjashëm, ndryshorja syprina (syprina e rrethit) është prej llojit real pasi merr vlera të numrave dhjetorë, syprina = = ($3^2 \cdot PI = 9 \cdot 3.14 = 28.2735$). Ndryshorja prej **llojit të shenjave** (mund të fitojë vlerë çfarëdo shenje e venduar në gjysmë thonjëza) është `shenjë = '+'`, ndryshore prej **llojit tekstual** (mund të marrë vlerë çfarëdo tekst në thonjëza) është `shtet = "Maqedonia është e përjetshme"` etj.

[19] Shenjat te gjuhët programore shënohen me gjysmëthonjëza.
[20] Tekstet te gjuhët programore shënohen e thonjëza.

Emrat e të dhënave

Emrat (identifikatorët[21]) e të dhënave në C++ mund të jenë:

- Fjalë kyçe ose të rezervuara[22].
- Emra e dobishëm të përkufizuar.

Fjalët e rezervuara kanë domethënie specifike dhe zbatim. Të atillë janë: const, double, float, int, strukt, unsigned, break, continue, else, for dhe të tjerë.

Emrat e dobishëm të përkufizuar janë formuar prej shfrytëzuesve (programuesve) me respektimin e këtyre rregullave:

- Emri fillon me shkronjën (A – ZH, a – zh) ose me vizë të tërhequr (_).
- Gjatësia e emrit është i pakufizuar.
- Shkronjat e vogla dhe të mëdha dallohen. Për shembull, unë dhe Unë janë emra të ndryshëm.
- Emri mund të përmbajë edhe shkronja të vogla dhe të mëdha prej alfabetit, shifra 0 – 9 dhe vizë e tërhequr.
- Emri nuk guxon të përmbajë vend të zbrazët.
- Emri nuk guxon të përmbajë shenjë special, sikurse: !, @, #, $, ^ etj.
- Emri nuk guxon të jetë fjalë kyçe ose e rezervuar.

Për shembull, emrat e drejtë të të dhënave janë:

```
a, A1, iTi, i2j3,, _emri, rrezja, R, numër_120, _char, TiMëso.
```

Emra jo të drejtë të të dhënave janë:

```
5_shi       - Emri nuk mund të fillon me shifër.
do$lar      - Emri nuk guxon të përmbajë shenjë speciale.
i 2         - Nuk është lejuar vend i zbrazët te emri.
char        - Nuk mund fjalë e rezervuar të jetë identifikator.
```

Për emrat e të dhënave nuk preferohen fjalë të rezervuara ose fjalë të ngjashme të rezervuara se fjalë të cilat përmbajnë fjalë të rezervuara.

Për shembull:

```
CHAR        – E ngjashme me fjalën e rezervuar char.
_float      – Përmban fjalë të rezervuar float.
__          – Dy viza të tërhequra
```

[21] Identifikatori (angl. identifier) ven prej fjalës identifikim, që d të thotë konstatimi i identitetit. Emrat e të dhënave shërbejnë për konstatim që është e dhënë dhe prandaj, quhen identifikator.

[22] Fjalët kyçe kanë domethënie special te gjuha. Fjalët e rezervuara për gjuhën dhe nuk mund të shfrytëzohen sikurse identfikator.

Llojet e të dhënave

Lloj i të dhënës paraqet bashkësi të fundshme mbi ndonjë bashkësi të përkufizuar të fundshme të operacioneve[23]. Vlerat e çdo lloj të të dhënave shkruhen në lokacione aq të memories sa është e caktuar me gjuhën për atë lloj. Prej këtu, shpesh thuhet se lloj i të dhënës tregon se si të interpretohet përmbajta e një ose më shumë lokacione të memories së njëpasnjëshme te e cila janë vendosur vlera e ndonjë të dhëne.

Në C++ ekzistojnë tre lloje të të dhënave:

- Të thjeshta (të pa strukturuara).
- Tëpërbëra (të strukturuara).
- Të adresave.

Llojet e të dhënave		
Të dhënat e thjeshta (të pa strukturuara)		
	Integrale[24]	
	numra të plotë	short, int, long, long long
	me shenjë	char
	logjike	bool
	real	float, double, long double
	i numërueshëm	enum
Lloje të përbëra (të strukturuara)		
	vargu	array
	struktura	struct
	unioni	union
	klas	class
Të adresave		
	tregues	pointer
	referencë	reference

Llojet e thjeshta ose **të pastrukturuara të të dhënave** janë ato të cilat nuk mund të ndahen në pjesë. Për shembull, 123, –456, 1234567890, 'W', 123.45 etj. janë të thjeshtë (të pastrukturuara, të pandara) të të dhënave, Ndërsa, "Ti"është e dhënë e strukturuar të llojit vargu tekstual[25], i cili përbëhet prej dy shenjave (të thjeshtë, të pastrukturuar) të të dhënave 'T' dhe 'i'. Prandaj, themi se **llojet e përëbërë (të strukturuarat) të të dhënave** përbëhen prej më shumë llojeve të thjeshta.

[23] Do të sqarojmë në vazhdim.
[24] Lloje integrale janë ato të cilat paraqesin një tërësi, d.m.th., nuk ndahen.
[25] Do të sqarojmë më vonë.

Të dhënat e llojit të adresave janë ato vlera e të cilëve është adresa e ndonjë lkacioni të memories[26].

Lloj i numrit të plotë të të dhënave dhe lloj i shenjës mund të jenë edhe të pashënuar (angl. unsigned):

```
unsigned[27] short
unsigned int
unsigned long
unsigned long long
unsigned char
```

Fjalët short, int, long, char, bool, float, double, enum, signed dhe unsigned janë të ashtuquajtura **lloje të specifikatorëve** (angl. type specifiers).

Lloj me numër të plotë, real, me shenja dhe logjike quhen edhe **lloje primitive të të dhënave** (angl. primitive data types) ose lloje themelore të të dhënave (angl. fundamental data types). Ato janë **ndërtuar në lloje të të dhënave** (angl. built-in data types).

Në vazhdim, shkurt do t'i sqarojmë llojet numra të plotë, real, me shenja dhe tekstuale të të dhënave.

Lloj numër të plotë të të dhënave janë vlera të numrave të plotë pozitiv dhe negativ {... –2, –1, 0, 1, 2... } prej saktë nënbashkësisë së caktuar për çdo nënlloj (short, int, long, long long). (Shiko ***tabelën 1.7.1, 1.7.2, 1.7.3***).

Lloj real i të dhënave janë vlerat e numrave real pozitiv dhe negative, d.m.th., numrat dhjetor {... –2.0..., –2.001..., –1.0..., 0.0..., 1.0..., 2.0... } prej saktë nënbashkësi të caktuar për çdo nënlloj (float, double, long double).

Lloj e shenjë të të dhënave janë vlera prej bashkësisë së shenjave, edhe atë: shkronja, shifra dhe shenja speciale. Gjatë shfrytëzimit të tyre si vlerë, vendohen te gjysmëthonjëzat për t'u dalluar prej shenjave të tjera të njëjta. Për shembull, numri i plotë 5 dallohet prej shenjës '5'. Të dhënat karakteristike të shenjës është një vend i zbrazët ' '.

Bashkësia e të dhënave të shenjave është saktë e renditur në të ashtuquajturën tabela ASCII (American Standard Code for Information Interchange), te e cila çdo shenjë ka paraardhësin e vet (përveç të parit) dhe pasardhësin e tij (përveç të fundit), (Shiko **Shtesa B: Shenja ASCII**).

Me lidhjen e të dhënave me shenjë, formohen varg (sekuenca) prej shenjave, të cilat trajtohen si lloj i veçantë i të dhënave të quajtura stringe.

Për t'u dalluar prej konstantave dhe prej ndryshoreve te program, stringet shkruhen në thonjëza.

[26] Do të sqarojmë më vonë.

[27] Unsigned do të thotë se të dhënat janë vetëm numra pozitiv dhe numri 0, d.m.th., nuk mund të kenë vlera negative.

Për shembull:

```
"C++ Bazat e programimit "
"Maqedonia"
"2020"
"E hënë, viti 21.06.2045 "
" "       // Ky është string me një vend të zbrazët – string blank (angl. blank).
""        // Ky është string i zbrazët (angl. null string) dhe dallohet
          // prej stringut blank.
"Stringu patjetër të shkruhet në një vijë dhe të mbarojë me thonjëza, nuk guxon
të kalon në rresht të rë, pasi ky është string"
```

Vëreni se i fundit prej shembujve nuk është string. Nëse stringu patjetër të jetë më i gjatë prej një vije, ai ndahet në më shumë stringje, të cilët lehtë bashkohen me shenjën +.

Për shembull, me shprehjen:

```
"Maqedonia" + " " + "2020"
```

do të fitohet stringu "Maqedonia 2021".

Ky operacion me stringje quhet **bashkim** (angl. concatenation).

Stringjet paraqein lloje të përbëra (të strukturuara) të të dhënave. Lloj tyre në C++ shënohet me **string**.

Stringjet nuk janë lloj i përkufizuar i të dhënave te gjuha C++, por janë të kyçur te **biblioteka standard e C**++ (angl. C++ Standard library).

Deklarimi i konstantave dhe ndryshoreve të të dhënave

Për t'u përpunuar të dhënat me ndonjë program, ato duhet të gjenden në memorien punuese të kompjuterit. Programi patjetër të dijë në cilën adresë te memoria gjendet çdo e dhënë dhe sa lokacione të memories zen. Thamë se numri i lokacioneve të memorisë që zënë ndonjë të dhënë varet prej llojit të tij, (shiko t***abela 1.7.1, 1.7.2, 1.7.3***). Prandaj, për të ditur përkthyesin cila e dhënë prej cilit lloj është, përveç të dhënës, patjetër të shkruhet edhe lloj i tij. Kjo mënyrë quhet **deklrimi i të dhënave** (angl. data dec-laration). Gjatë deklarimit, për çdo të dhënë jepet emri dhe lloj i tij. Nëse gjatë deklariit jepet edhe vlera fillestare e të dhënës, ajo quhet deklarimi dhe **inicializimi** (angl. deklaration and initialization).

Para se të shfrytëzohet ndonjë konstante ose ndryshore te programi (në operacione të ndryshme, sikurse: njehsim, kopjim, shtypje etj.), ai patjetër të jetë deklaruar. Gjatë përkthimit, përkthyesi rezervon memorie përkatëse për çdo konstante dhe ndryshore të deklaruar (sipas llojit) ose rezervon dhe vendos vlerë te lokacioni nëse kryhet deklarimi dhe inicializimi. Poashtu, programuesit dinë te cili lokacion i memories gjenden të dhënat, por përkthyesi e lidh lokacionin me emrin e konstantes ose ndryshorja (identifikatori) i të dhënës.

Thamë se të dhënat mund të jenë konstanta dhe ndryshore.

Konstante janë ato identifikator vlera e të cilit (përmbajtja e lokacionit te memoria) nuk ndryshon për kohën e realizimit të programit. (Nëse përpiqemi ta ndryshojmë vlerën e ndonjë konstante, do të paraqitet gabimi).

Konstantat deklarohen me fjalën const sipas të cilit përmendet lloj i emrit, por pastaj patjetër të shoqërohet vlera me operatorin për shoqërim =:

const *lloj emri* = *vlera*; [28]

Për t'u dalluar konstantat prej ndryshoreve, shfrytëzohet praktika ato të shkruhen me shkronja të mëdha, por fjalët të ndahen me vizë poshtë.

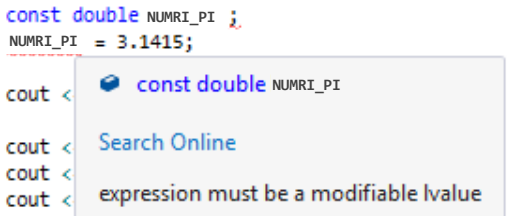
Shembuj :

```
const int M = 12;                          // Muaj në vit
const float NXITIMI_TOKËS  = 9.81;
const double NUMRI_PI = 3.1415;
const char DA = 'D';
const string DATA  = "E hënë, 21.06.2036";
```

Konstanta M është i llojit int dhe te lkacioni i tij mund të vendohet vetëm vlera numër i plotë. Konstanta DA është i llojit char, që do të thotë se vlera që mund t'i shoqërohet patjetër të jetë shenjë. Me të tjera fjalë, lloj konstanta cakton çfarë vlere mund të marrë konstanta.

Gjithashtu, është shumë e dobishme pas konstantes të shkruhet koment, nëse ajo është e nevojshme pas konstntes të shkruhet koment, nëse ai është i nevojshëm, që të dihet çka është ajo konstante.

Ë vërejmë edhe se gjatë deklarimit të konstantes, ajo patjetër të inicializohet. Nëse inicializohet pas deklarimit, do të paraditet gabimi.



Ndryshoret janë ato identifikator vlera e të cilit (përmbajtja e lokacionit te memoria me të cilën është lidhur emri i tyre), mund të ndryshojë.

Ndryshoret deklarohen me:

lloj emri;

ose:

lloj emri = *vlera*;

ose:

lloj emri = *shprehje*;

[28] Te përkufizimet e përgjithshme do të shfrytëzojmë shkronja të pjerrëta. Shprehjet prej një fjale do tÇi shkruajmë me fjalë të bashkuara, por shprehjet prej më shumë fjalëve (si lista) do t'i shkruajmë me fjalë të bashkuar me vijë të poshtme.

Mënyra e fundit paraqet njëkohësisht deklarim de inicializim (shoqërim të vlerës fillestare) të ndryshore. Vlera e ndryshores së inicializuar mund të (më vonë) të ndryshojë kudo në program, me urdhrin për shoqërim të vlerës së ndryshores.

Që të mund të shfrytëzojmë te program, ndryshorja patjetër të jetë **përkufizuar**. Ajo përkufizohet me shoqërim të vlerës gjatë deklarimit (me inicializim) ose me urdhër për shoqërim.

Peratori standard për shoqërim të vlerës së ndryshores është e =. Ana e djathtë gjatë shoqërimit mund të jetë vlerë ose shprehje.

Urdhri për shoqërim vlere të ndryshores është:

emri = *vlerë*;

ose:

emri = *vlerë*;

Deklarimi, deklarimi, dhe inicializimi dhe përkufizimi i ndryshoreve mund të bëhet kudo te programi.

Shembuj:

```
short i;                    // Deklarimi i ndryshores numër të plotë i.
int m = 10, n = 17;         // Deklarimi dhe inicializimi i
                            // ndryshoreve numra të plotë m dhe n.
float pesha  = 56.75f;      // Deklarimi dhe inicializimi i ndryshores
                            // reale të peshës.
double shuma;               // Deklarimi i ndryshores reale shuma.
char shenja  = '+';         // Deklarimi dhe inicializimi i
                            // ndryshores së shenjës.
string shteti;              // Deklarimi i stringut të shtetit.
i = 5;                      // Shoqërimi i vlerës së ndryshores i.
shuma  = 123.45;            // Shoqërimimi vlerës së ndryshores shumë.
shteti  = "MAQEDONIA";      // Shoqërimi i vlerës së ndryshores
                            // shteti.
double numriParë = (1 + 2 * 7) / (7 - 5);   // Deklarimi dhe inicializimi
```

Me urdhrin e parë, përkthyesi rezervon memorie prej 2 bajt për ndryshoren i (pasi është i llojit short), d.m.th., ai lokacion e lidh me emrin i. Nëse përpiqemi ta shfrytëzojmë ndryshoren i që është vetëm deklarim, por nuk është dhe të inicializuar, do të paraqitet ky gabim.

```
C4700    uninitialized local variable 'i' used
```

Me urdhrin:

```
i = 5;
```

të ndryshores i i shoqërohet vlerë 5, d.m.th., kjo vlerë vendohet në shoqërim të lokacionit për i në memorie. Pas këtij urdhri, ndryshorja i mund të shfrytëzohet kudo qoftë te programi.

Te vija e dytë prej **Shembujve** janë deklaruar dy ndryshore. Kjo është e mundur vetëm nëse ndryshoret janë të llojit të njëjtë.

Te shembulli i fundit, te ana e djathtë prej urdhrit gjendet shprehja aritmetike, që e para njehsohet dhe pastaj vlera e saj shoqërohet ndryshores numriDytë.

C++ e lejon edhe këtë mënyrë të shoqërimit të vlerave:

```
x = (y = 10) * (z = 5);     // Të x i shoqëron vlerë 50.
x = y = z = 20;             // të x, y dhe z u shoqëron vlerë 20.
```

Nuk ka rregull për të shkruar emra të ndryshoreve te C++, përveç kufizimeve të përmendura te titulli **Emrat e të dhënave** prej nënpikës **1.6 Hyrje në C++**. Çdo programues formon stilin e vet, por shpesh shfrytëzohet stili i shkronjës së parë të jetë e vogël, por çdo tjetër fjalë te emri të fillon me shkronjë të madhe. Për shembull: numriPI, çmimiNjësi, emriDheMbiemri, syprinaEKatrorit etj. Ky stil shfrytëzohet edhe te gjuha programore C++, por do ta shfrytëzojmë edhe ne.

Shembulli 1.6.1

Te shembulli (***figura 1.6.5***) është konstanta e deklaruar NUMRI_PI dhe i është shoqëruar vlera 3.1415. Pastaj, është deklaruar ndryshorja short ÇmimiNjësi, të cilës i shqërohet vlerë me urdhrin për shoqërim, por jo gjatë deklarimit.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // Deklarimi, inicializimi dhe shtypja e konstantave dhe ndryshoreve
6
7       const double NUMRI_PI   = 3.1415;  //Deklarimi dhe inicializimi i
8                                          //konstantes
9       short çmimiNjësi;                  // Deklarimi i ndryshores numër i plotë
10      int copaAkullore      = 100;       //Deklarimi dhe inicializimi
11      çmimiNjësi    = 50;                //Shoqërimi i vlerës ndryshores
12      cout << Konstanta PI     =" << NUMRI_PI  << endl;
13      cout << Vlera e        " <<  copaAkullore     << "akullore nga   "
14          << çmimiNjësi
15      cout << " denar dhe është  " << copaAkullore       * çmimiIVetëm
16          << " denar   " << endl;
17
18      cout << endl;
19      system( "Color 17" );
20      system( "pause" );
21      return 0;
22  }
```

Figura 1.6.5

Rezultati rej realizimit të programit është:

```
Konstanta PI=3.1425
Vlera e 100 akulloreve nga 50 denar është 5000 denarë
Press any key to continue...
```

Detyra për ushtrime

1. Shkruani 3 programe te të cilat do tÇi kyçim urdhërat për deklarim dhe inicializim. Pastaj, programet plotësoni me urdhëra për shtypje në ekran me të cilat do tÇi shtypni vlerat e të gjitha deklaratave të ndryshoreve. Çka ndodh kur shtypni vlerë të ndryshores të painicializuar ose të papërkufizuar ndryshorja?
2. Të shkruhet program te e cila do të deklaroni dhe inicializoni dy ndryshore me urdhrin int a = 12, b = 7;. Pastaj, shtypni shumën, ndryshimi dhe prodhimin e tyre në tre vija të njëpasnjëshme në ekran.
3. Le të jetë deklaruar/përkufizuar:
```
int x;
float y;
char z;
x = 2; y = 5.8; z = 'P';
```
Të shkruhet program me të cilin d të shtypen vlerat e ndryshoreve x, y dhe z në një vijë, ashtu që të jenë të ndara me nga një vend të zbrazët, por pastaj të kalohet te vija e re.
4. Le të jetë deklaruar/përkufizuar:
```
int dita  = 21;
string muaj  = "Qershor";
int viti  = 2020;
string sot  = "E hënë";
```
Të shkruhet program te i cili do t'i shfrytëzojmë ndryshoret e deklaruara dhe

inicializuara dhe do të shtypen:

Sot është e Hënë, 21 Qershor viti 2021.

Pyetje për kontroll të njohurive

1. Kush e ka dizajnuar gjuhën C++?
2. Prej çka përbëhet bashkësia e shenjave në C++?
3. Numëroni disa prej shenjave speciale që shfrytëzohen në C++.
4. Çka janë identifikatorët?
5. Çka është sintaksa, kurse çka është semantika e gjuhës?
6. Si ndahen identifikatorët?
7. Shkruani formën e funksionit main().
8. Me cilën shenjë ndahen urdhërat në C++?
9. Pse kryhet dhëmbëzimi i programit?
10. Si shënohen komentimet te programi?
11. Cili është urdhri për shtypje në C++?
12. Cili është operatori për shtypje?
13. Me cilin eskejp sekuencë kalohet për shtypje në rreshtin e ri?
14. Çka është struktura rendore prej urdhërave?
15. Çka është madhësia, kurse çka është e dhëna?

16. Çka është konstanta, kurse çka është ndryshorja?
17. Cilat vlera i quajmë literale?
18. Sa mund të jetë i gjatë emri i ndryshores në C++?
19. Cilat janë karakteristikat e të dhënës?
20. Si mund të jenë emrat e të dhënave në C++?
21. Përmend rregullat për përkufizim të emrave të përkufizuar të emrave të dobishëm (identifikatorëve).
22. Cilët prej këtyre emrave të të dhënave janë shkruar drejtë, kurse cilat nuk janë dhe pse?

a) _54abv	b)unë_5T	c) prinf	ç) aj De
d) emriIm	dh) for	e) a.7_b9	f) b9_c10

23. Sqaro çka është lloji i të dhënës?
24. Cilat lloje të të dhënave i dini në C++?
25. Sqaroni çka janë lloje të thjeshta, kurse çka janë lloje të përbëra të të dhënave.
26. Si mund të jetë përmbajtja e llojeve të adresave të të dhënave?
27. Cilat specifikator të llojeve i dini?
28. Çka paraqet fjala unsigned nëse përmendet para llojit të të dhënës?
29. Cilat vlera mund t'i kenë të dhënat prej llojit numër të plotë, të llojit real, prej llojit të shenjës dhe prej llojit tekstual. Me cilët specifikator shënohen këto lloje?
30. Për cilët identifikator themi se është konstant, por për cilin se është ndryshore?
31. Sa here dhe në cilat vende të programit mund të shfrytëzohet ndryshorja e njëjtë?
32. Çka do të ndodh nëse programin e njëjtë shfrytëzoni edhe konstante edhe ndryshore me emrin e njëjtë?
33. Emrin e programit a mund ta shfrytëzoni si ndryshore?
34. Çka është deklaratë, kurse çka është përkufizimi i ndryshores?
35. Çka do të thotë inicializimi i ndryshores?
36. Shkruani formën e përgjithshme të urdhrit për deklarim dhe inicializim.
37. Përmend disa shembuj për deklarim dhe disa për deklarim dhe inicializim të ndryshorve.
38. Me cilin operator shoqërohet vlera e ndryshores ose të konstantes?
39. Kur mund të shfrytëzohet ndonjë ndryshore që është vetëm e deklaruar?
40. Kjo deklaratë a është e drejtë: int m = 10, n = m;.

1.7 Llojet e të dhënave

Lloj numër i plotë i të dhënave int

Lloj numër i plotë i të dhënave mund të ketë vlerë të çfarëdo numri të plotë prej bashkësisë së numrave të plozë $Z = \{... -2, -1, 0, 1, 2...\}$.

Përmendëm se ka katër lloje të të dhënave numër të plotë: short, int, long dhe long long. Të varur prej llojit, madhësia e bashkësisë së vlerave të ndryshme.

Te ***tabela 1.7.1*** 1 janë dhënë lloje numra të plotë të të dhënave, vargjet e vlerave të tyre (angl. data range) dhe sasitë të memoria[29] që e zënë.

Lloj	Vlera	Memoria
short	–32768... 32767	16 bit
int	–2147483648... 2147483647	32 bit
long	–2147483648... 2147483647	32 bit
long long	–9223372036854775808... 9223372036854775807	64 bit
unsigned short	0... 65535	16 bit
unsigned int	0... 4294967295	32 bit
unsigned long	0... 4294967295	32 bit
unsigned long long	0... 18446744073709551615	64 bit

Tabela 1.7.1

Vargu i vlerave të çdo lloj varet prej kompjuterit te i cili kryhet programi. Vlerat minimale dhe maksimale përkufizohen si konstante:

```
short               SHRT_MIN = -32768
                    SHRT_MAX = 32767
unsigned short      USHRT_MAX = 65535
int                 INT_MIN = -2147483648
                    INT_MAX = 2147483647
unsigned int        UINT_MAX = 4294967295
long                LONG_MIN = -2147483648
                    LONG_MAX = 2147483647
unsigned long       ULONG_MAX = 4294967295
long long           _I64_MIN = -9223372036854775808
                    _I64_MAX = 9223372036854775807
unsigned long long  _UI64_MAX = 18446744073709551615
```

Për operacionet me lloje numër të plotë të të dhënave, shfrytëzohen këta operatorë aritmetikor:

+	Mbledhja
–	Zbritja dhe negacioni[30]

[29] Sasia e memories që zënë lloje të të dhënave mund të ndryshojë prej një platform të kompjuterit në tjetrin.

[30] Shenja minus (–) para një operandi quhet operator unar, d.m.th., minus unar.

*	Shumëzimi
/	Pjesëtimi numra të plotë – pjesa e plotë prej herësit të y numrave të plotë
%	Pjesëtim me modul-mbetje prej pjesëtimit të dy numrave të plotë

Rezultati prej pjesëtimit të dy numrave të plotë me operatorin / është pjesa e prej pjesëtimit. Për shembull:

```
10 / 4 = 2,  -15 / 6 = -2,    20 / (-6) = -3,    (-14) / (-3) = 4
5 / 0  do të paraqesë gabim gjatë përkthimit.
```

Rezultati prej zbatimit të operatorit % mbi dy numra t plotë është mbetja prej pjesëtimit. Për shembull[31]:

```
10 % 4 = 2,  -15 % 6 = -3,   20 % (-6) = 2,    (-14) % (-3) = -2
5 % 0  do të paraqesë gabim gjatë përkthimit.
```

Prioritet të operatorit për numra të plotë është:

1	*	/	%	prioritet i njëjtë
2	+	–		prioritet i njëjtë

Gjatë njehsimit të shprehjeve aritmetike, operacionet realizohen prej të majtës nga e djathta. Së pari realizohen operacionet me prioritet 1 (cili do prej tre operacioneve), por pastaj operacionet me prioritet 2 (të çfarëdo qoftë prej dy operacioneve).

Për shembull:

```
1 + 2 * 3 – 4 % 5 * 6 / 7 = 1 + 6 – 3 = 4
```

Me përdorimin e kllapave, ndryshon renditja e realizimit të operacioneve të caktuara me prioritetet e operatorëve.

Për shembull:

```
(1 + 2 * (3 – 4)) % (5 * 6 / 7) = (1 + 2 * (-1)) % 4 = (1 – 2) % 4 =
-1
```

Shembulli:

```
7 / 3 = 2
205 / 20 = 10
205 % 20 = 5
-25 % 3 = -1
7.0 / 4 = 1.7500
(1 + 2 * (3 + 4)) % 5 = 0
(1 + 2 * 3 + 4) % 5 = 1
1 + 2 * 3 + 4 % 5 = 7
-15 % (-4) = -3
15 % (-4) = 3
2147483647 + 1 = -2147483648      // mbushja [32]
```

[31] Nëse një operator është negative, shenja për mbetje varet prej përkthyesit.
[32] Për përgjysmimin do të sqarojmë më vonë.

Operator standard për shoqërim të vlerës së ndryshoreve te urdhërat për shoqërim është = . Gjatë shoqërimit të konstantes të pashënuar, duhet të vendohet prapashtesa (sufiksin) u ose U. Në të kundërtën, do trajtohet sikurse int.

Edhe për këto lloje të konstantave vendohet prapashtesa:

lloj	i prapashtesës
long	l ose L
long long	ll ose LL
unsigned long	ul ose UL
unsigned long long	ull ose ULL.

Shembuj:

```
int numriIParë, numriIDytë;
short shifra  = 8;
long shifraArabe = 1234567890;
numriIParë = 1 + 2 * 3 + 45 % 6;          (= 10)
numriIParë = numriIParë + shifra;         (= 18)
numriIDytë  = numriIParë / shifra;        (= 2)
numriIDytë  = numriIDytë  % shifra;       (= 0)
const long PREJ_DIELLI_DERI_SATURN  = 1424600000L;
long long njësiaAstronomike = 149597870700LL;
unsigned int tëgjithaShifrat = 1234567890u;
unsigned long maxLong = 4294967295ul;
unsigned long long maxLL = 18446744073709551615ull;
x = (y = 10) * (z = 5);
x = y = z = 20;
```

Te shembulli vijues do të paraqesim pjesëtim numër të plotë, pjesëtim sipas modulit, d.m.th., mbetja prej pjesëtimit numër të plotë dhe prioritet të operacioneve për numrat e plotë.

Shembulli 1.7.1

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {    // Operacione me numra të plotë
5
6       // Pjesëtimi me numra të plotë-pjesa e plotë prej herësit të dy numrave të plotë
7       cout << "10 / 4 = " << 10 / 4 << endl;
8       cout << "-15 / 6 = " << -15 / 6 << endl;
9       cout << "20 / (-6) = " << 20 / ( -6 ) << endl;
10      cout << "-14 / (-3) = " << ( -14 ) / ( -3 ) << endl;
11      // Pjesëtimi sipas modulit-mbetja prej pjesëtimit të dy numrave të plotë
12      cout << "10 % 4 = " << 10 % 4 << endl;
13      cout << "-15 % 6 = " << -15 % 6 << endl;
14      cout << "20 % (-6) = " << 20 % ( -6 ) << endl;
```

Figura 1.7.1

```
15      cout << "-14 % (-3) = " << ( -14 ) % ( -3 ) << endl;
16      //   Prioritet i operatorëve për numrat e plotë
17      cout << "1 + 2 * 3 - 4 * 5 % 6 / 7 = "
18          << 1 + 2 * 3 - 4 % 5 * 6 / 7 << endl;
19      cout << "(1 + 2 * (3 - 4)) * (5 % 6 / 7) = "
20          << ( 1 + 2 * ( 3 - 4 ) ) % ( 5 * 6 / 7 ) << endl;
21
22      cout << endl;
23      system( "Color 17" );
24      system( "pause" );
25      return 0;
26  }
```

Figura 1.7.1 (vazhdim)

Një dalje prej realizimit të programit prej ***figura 1.7.1*** është:

```
10 / 4 = 2
-15 / 6 = -2
20 / (-6) = -3
-14 / (-3) = 4
10 % 4 = 2
-15 % 6 = -3
20 % (-6) = 2
-14 % (-3) = -2
1 + 2 * 3 - 4 * 5 % 6 / 7 = 4
(1 + 2 * (3 - 4)) * (5 % 6 / 7) = -1

Press any key to continue . . .
```

Forma e shkurtuar e operatorëve

Operatorët në C++ mund të shfrytëzohen në të ashtuquajturën **forma e shkurtuar** e cila shprehet me operatorin aritmetik dhe operatorin për shoqërim =. Ato quhen **operator të përbërë për shoqërim** (angl. compound assignment operators).

Forma e përgjithshme e operatorëve për shoqërim është:

operatoriaritmetik =

Format e shkurtuara të operatorëve për shoqërim për të dhënat e numrave të plotë janë:

+= **-=** ***=** **/=** **%=**

Shembuj:

`numriIParë += shifra;` është njëjtë me `numriIParë = numriIParë + shifra;`
`numriIDytë %= shifra;` është njëjtë me `numriIDytë = numriIDytë % shifra;`

Te shembulli vijues është ilustruar shfrytëzimi i forms së shkurtuar të operatorëve.

Shembulli 1.7.2

Te ***figura 1.7.2*** është ilustruar shfrytëzimi i formës së shkurtur të operatorëve.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {    // Forma e shkurtur e operatorëve
5
6       int cifra = 8;
7       cout << "shifra   " << shifra << endl;
8
9       int numriIParë,numriIDytë,numriITretë;
10       numriIParë = numriIDytë = 100    // Ova e dozvoleno vo C++
11      cout <<"Numri i parë " <<numriIParë<< endl;
12      cout << "Numri i dytë " <<numriIDytë << endl;
13
14     numriIParë +=shifra                // forma e shkurtuar e operatorit +
15     numriIDytë *= shifra               // forma e shkurtuar e operatorit *
16      cout << "numriIParë= cc      <<numriIParë<< endl;
17      cout << "numriIDytë*= cifra =  " <<numriIDytë<< endl;
18
19      //  Shoqërimi i vlerave ndryshoreve te shprehja
20      //  edhe kjo është lejuar në C++
21    numriITretë = (numriIParë= 123 ) + ( numriIDytë= 321 );
22      cout << "numriITretë= " <<numriIParë<< " + " <<numriIDytë<< " = "
23          <<numriITretë<< endl;
24
25      cout << endl;
26      system( "Color 17" );
27      system( "pause" );
28      return 0;
29  }
```

Figura 1.7.2

Një dalje prej realizimit të programit është:

```
Shifra = 8
Numri i parë = 100
Numri i dytë = 100
numriIParë += shifra = 108
numriIDytë  *= shifra = 800
Numri i tretë = 123 + 321 = 444
Press any key to continue . .
```

Detyra për ushtrime

1. Shkruani program te i cili për numrat x = 123 dhe y = 52 do t'i njehsoni dhe shtypni: shumën, ndryshimin, prodhimin, pjesën e plotë prej herësit të tyre. Poashtu, të shfrytëzohet vetëm një ndryshore z.
2. Të shkruhet program me të cilin do t'i njehsoni dhe shtypni katrorin dhe kubin e numrit të plotë numriIPlotëX = 123, pa i shfrytëzuar ndryshoret e të tjera.

3. Të shkruhet program me të cilin do t'i shtypni shifrën e mesme të numrit treshifror numriX = 274.
4. Treni prej Manastiri për në Shkup niset në ora 5 dhe 20 minuta. Në Shkup arrin në ora 8 dhe 12 minuta. Të shkruhet program me të cilin do të njehsoni sa kohë treni ka udhëtuar prej Manastiri deri në Shkup.
5. Të shkruhet program me të cilin do të njehsoni shuma dhe ndryshimi i këndeve

   $\alpha = 100° 47 ′38″$ dhe $= \beta 35 °53 ′49″$.

Lloj real i të dhënave **float, double, long double**

Lloj real i të dhënave janë ato të cilat fitojnë vlera prej bashkësisë së numrave real. Me llojreal shpesh mendohet në numrat dhjetor.

Në C++ janë ndërtuar te lloje të të dhënave reale, edhe atë: **float, double** dhe **long double**.

Të dhënat prej llojit real mund të jenë konstanta dhe ndryshore.

Inicializimi kryhet sikurse edhe te llojet e numrave të plotë të të dhënave.

Gjatë deklarimit të ndryshoreve prej llojit float, duhet të vendohet prapasjtesa (sufiksi) f ose F. Në të kundërtën, do të trajtohet si lloj double.

Gjithashtu, llojit long double vendohet prapashtesa l ose L.

Shembuj:

```
float x;
float peshEPërgjithshme;
double shumaEPërgjithshme  = 12345.67;
long double PI = 3.14159265358979323846;
const float NXITIMI_TOKËS  = 9.81f;
float numri_është  = 2.7182818284590452354F;
long double rrezjaEElektronit = 0.00000000000000282L;  //2.82x10-15
```

Preciziteti i të dhënave prej llojit **float, double** dhe **long double** është i ndryshëm, edhe atë:

- Të dhënat **float** zënë 32 bit dhe kanë 7 shifra të rëndësishme.
- Të dhënat **double** zënë 64 bit dhe kanë 15 shifra të rëndësishme.
- Të dhënat **long double** zënë 80 bit dhe kanë 18 shifra të rëndësishme.

Vargu i vlerave të llojit real të të dhënave është dhënë te ***tabela 1.7.2.*** Vargu është njehsuar sipas precizitetit të paraqitjes së numrave me 32, 64 dhe 80 bit. Ekzistojnë formate për paraqijen e të plotave dhe të numrave dhjetorë[33].

[33] Ekziston IEEE standard (Institute of Electrical and Electronics Engineers) me të cilin janë përshkruar formatet për paraqitjen e të plotave dhe numrave dhjetorë në kompjuter, sipas precizitetit, d.m.th., sipas numrit të bitëve.

Lloj	Vlera	Memoria
float	$\pm 3.4028234 \bullet 10^{-38} ... \pm 3.4028234 \bullet 10^{38}$	32 bit
double	$\pm 1.7976931348623157 \bullet 10^{-308} ...$ $\pm 1.7976931348623157 \bullet 10^{308}$	64 bit
long double[34]	$\pm 3.4 \bullet 10^{-4932} ... \pm 1.1 \bullet 10^{4932}$	80 bit

Tabela 1.7.2

Vlerat minimale dhe maksimale të numrave dhjetorë në C++ varen prej përkthyesit dhe janë dhënë si konstante[35]. Për shembull, te kompjuteri im ato janë:

```
float                FLT_MIN  = 1.17549e-38
                     FLT_MAX  = 3.40282e+38
double               DBL_MIN  = 2.22507e-308
                     DBL_MAX  = 1.79769e+308
long double          LDBL_MIN = 2.22507e-308
                     LDBL_MAX = 1.79769e+308
```

Operatorët aritmetik për të dhënat prej llojit real janë:

+	mbledhja
–	zbritja ose negacioni
*	shumëzimi
/	pjesëtimi real (nëse të paktën njëri prej të dhënave është real)

Nëse gjatë pjesëtimit me 0.0 del jashtë prej vargut të numrave, nuk paraqitet gabim, por për rezultat fitohet inf ose –inf, (e pafundshme, angl. infinity).

Prioriteti i operatorëve për numrat real në shprehjet aritmetike shqyrtohet prej të majtës në të djathtë, edhe atë:

1	*	/	prioritet i njëjtë (operatorët multiplikativ)
2	+	–	prioritet i njëjtë (operatorët aditiv)

Edhe këta operatorë mund të shfrytëzohen në formën standard dhe të shkurtuar:

+=	–=	*=	/=

[34] Ky lloj është shfrytëzuar në versionet e mëhershme në C++. Në standardet e sotshme për C++, ky lloj është ekuivalent me double.

[35] Vlerat më të vogla pozitive për llojet reale të të dhënave janë përkufizuar si konstante. Te kompjuteri im ato janë:

```
float                FLT_EPSILON  = 1.19209e-07
double               DBL_EPSILON  = 2.22045e-16
long double          LDBL_EPSILON = 2.22045e-16
```

Shembulli 1.7.3

Me këtë shembull (***figura 1.7.3***) ovet ndryshimi në precizitetin prej llojit float (të paraqitur me 32 bit) dhe prej llojit double (të paritur me 64 bit).

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {    // Operacione me numra real
5
6       //Për tip float
7       float
8      numriIParë= 1.000000000f;
9      numriIDytë = 0.999898989f;
10
11      cout <<                                          << endl;
12      cout <<numriIParë<< " - " <<numriIDytë << " = " << rezultati << endl;
13
14      //Për tip double
15      double numriIParë, numriIDytë, rezultati
16      numriIParë  = 1.000000000;
17      numriIDytë   = 0.999898989;
18
19      cout << "Ndryshimi i numrave dhjetorë të llojit        " << endl;
20      cout << numriIParë << " - " << numriIDytë  << " = "
21          << rezultati << endl;
22
23      cout << endl;
24      system( "Color 17" );
25      system( "pause" );
26      return 0;
27  }
```

Figura 1.7.3

Shembull prej realizimit të programit është:

```
Ndryshimi i numrave dhjetorë të llojit float:
1 - 0.999899 = 0.00010103
Ndryshimi i numrave dhjetorë të llojit float:
1 - 0.999899 = 0.000101011
Press any key to continue . . .
```

Prej daljes vërehet se edhe pse vlera e të dhënave është e njëjtë, rezultati është i ndryshëm.

Formati për paraqitjen e të dhënave dhjetore

Të dhënat dhjetore te gjuhët programore paraqiten në dy mënyra:

- Te formati dhjetor (**f**- formati, **F**- formati), mënyra e njohur si **pika e palëvizshme** (angl. fixed point).
- Te formati eksponencial (**e**- formati, **E**- formati), mënyra e njohur si **pika e lëvizshme** (angl. floating point).
- Shkronja **f** vjen prej fjalës „float“, por shkronja e vjen prej fjalës „exponent“. Numrat dhjetor te f- formati (ose F- formati) shkruhen me pika dhjetore në

vendin fiks, e cila i ndan pjesën e plotë dhe dhjetore të numrit. Prandaj, ajo mënyrë e paraqitjes së numrave dhjetorë quhet paraqitja me pikë të palëvizshme. Për shembull: 1234.56, –0.25, 0.5432, –123.45 etj.

Te e- formati (ose E- formati) pika e lëvizshme, ku shfrytëzohet eksponent. Për shembull, numri 123456789.12 mund të shkruhet me zhvendosjen e pikës dhjetore kudo qoftë majtas ose djathtas:

$1234567891.2x10^{-1}$, $12345.678912x10^{4}$, $0.0012345678912x10^{11}$

Numrat e njëjtë në e- ose E-format shkruhen në këtë mënyrë:

1.234568e+08, 1.234568E+08.

Kur numrat dhjetor shtypen në e-format (ose E-format), shkruhen *vetëm një shifër* majtas prej pikës dhjetore, ndërsa shkronja e ose E shkruhet para eksponentit. Eksponenti shkruhet me shenjë – ose +.

Numrat dhjetor të shtypur në format f, F, e ose E kanë precizitet prej 6 shifrave të sakta, por e shtata rrumbullakohet.

Për shembull, nëse janë përkufizuar (të deklaruara dhe të inicializuara) ndryshoret vijuese në f-format dhe te e-format:

```
float a = 1234.56f, a1 = -25000.f, a2 = 0.5432f;
double c = 76.543e3, c1 = -25.e-1, c2 = 2e3;
```

gjatë shtypjes së tyre në f-format, fitohet:

```
1234.560059     -25000.000000   0.543200
76543.000000    -2.500000       2000.000000
```

por gjatë shtypjes në e-format, fitohet:

```
1.234560e+03    -2.500000e+04   5.432000e-01
7.654300e+04    -2.500000e+00   2.000000e+03
```

Ngjashëm, nëse te program prej ***figura 1.7.3*** pak ndryshon vlera e ndryshores numriIDytë (vendi 0.999898989, jepet 0.999998989), vlera e ndryshores rezultati do të shtypet në **e**-format:

```
Ndryshimi i numrave dhjetorë të llojit float:
1 - 0.999999 = 1.01328e - 06
Ndryshimi i numrave dhjetorë të llojit float:
1 - 0.999999 = 1.011e-06
Press any key to continue . . .
```

Shprehjet aritmetike dhe konversioni i llojit të të dhënave

Gjatë njehsimeve, te programet paraqiten shprehje të thjeshta dhe të përbëra aritmetike. Ato shprehen me konstanta, ndryshore dhe operator.

Operatorët mund të jenë:

Unare: + dhe –
Binare: + – * / dhe %

Për shembull:

–5 + 3, –y, a + b / c % d, 1.2 * x * x – 3.4 * x + 5.67

Nëse te shprehja ka më shumë operatoro, ata realizohen sipas prioritetit:

më i lartë: unar + unar –
mesatar: * / %
më i ulët: + –

Për shembull, për a = 5, b = 4, c = 3, d = 2, shprehja a + b / c % d do të njehsohet:

```
a + b / c % d = 5 + 4 / 3 % 2 = 5 + 1 % 2 = 5 + 1 = 6
```

Prioriteti i operatorëve mund të ndryshojë me kllapa. Për shembull, nëse shprehjen paraprake e shkruajmë (a + b) / (c % d), do të njehsohet:

```
(a + b) / (c % d) = (5 + 4) / (3 % 2) = 9 / 1 = 9
```

Operatorët unar te shprehjet realizohen prej të djathtës në të majtë. Për shembull, për a = 5 dhe b = –3:

```
a + ( –b ) = 5 + (– (–3 ) ) = 5 + ( +3 ) = 8
```

Nuk mundet `5 (+ -) (-3) )`

Thamë se lloj i ndryshores cakton se lokacioni i memories përkatëse mund të përmbajë vetëm lloj të atillë të të dhënave.

Për shembull, nëse është deklaruar:

```
int numërIPlotë;
float numërReal ;
```

këto urdhëra për shoqërim janë në rregull:

```
numërIPlotë  = 5;
numërReal  = 6.7;
```

Por çka ndodh nëse i ndryshojnë vlerat, d.m.th.,.:

```
numërIPlotë  = 6.7;
numërReal  = 5;
```

Me urdhrin e parë, pasi ndryshorja numërIPlotë mund të fitojë vlerë vetëm nujmra të plotë, pritet pjesa dhjetore prej 6.7 dhe të ndryshores numërIPlotë dhe i shoqërohet vlera 6.

Me urdhrin e dytë, pasi ndryshorja numërReal mund të fitojë vlerë vetëm numra real, konvertohet numri i plotë 5 në real 5.0 dhe të ndryshores numriReal i shoqërohet vlera 5.0.

Mënyra e këtillë e shoqërimit të vlerës së ndryshores prej lloj jo përkatës quhet **konversion implicit** (automatik) ose **detyrimi i llojit** (angl. type coercion).

Vërejmë se gjatë shoqërimit të vlerës reale të ndryshores numër i plotë humbin informatat për të dhënën pasi pritet pjesa dhjetore. Për t'iu larguar humbjes së informatës, shfrytëzohet **lloj eksplicit i konversionit** (angl. type conversion), të njohur sikurse **hedhja e llojit** (angl. type casting).

Hedhja kryhet me përmendjen e llojit para ndryshores ose shprehjes që duhet të knverton. Kështu dy urdhërat paraprake me hedhje do të jenë:

```
numërIPlotë  = int(6.7);
numërReal  = float(5);
```

Pasi ndryshorja numërIPlotë d të jetë vlerë 6 edhe në rastin vlera të jetë 6.99 (që është më afër deri te 7), për t'u humbur më pak informata për të dhënën, praktika e mire programore është ai të rrumbullakon me shtuarje vlere 0.5.

Urdhri do të jetë:

```
numërIPlotë  = int(6.7 + 0.5);
```

Shpesh, shprehjet aritmetike përmbajnë edhe ndryshore numra të plotë dhe nunra real, d.m.th., përmbajnë lloje të përziera të të dhënave.

Te shprehjet me lloje të përziera të të dhënave, sikurse:

```
numërReal1  = 5 + 7.34;
numërReal2  = 129.56 - 1.3e+2;
```

kryhet konversioni implicit i llojeve me precizitet më të vogël në llojet me recizitet më të madh.

Me këtë grup vijues të urdhërave:

```
int k = 5;
double p;
p = k + 7.34                (= 5.0 + 7.34 = 12.34)
```

së pari kryhet konversioni implicit (automatik) i ndryshores numër i plotë k (= 5) te ndryshorja prej llojit double (= 5.0), mblidhet me 7.34 dhe pastaj shuma i shoqërohet ndryshores p (= 12.34).

Te shprehja për njehsim të p, mund të shfrytëzohet edhe knversioni eksplicit:

```
p = float(k) + 7.34;
```

Me këtë urdhër:

```
int m = p;                          (12 ← 12.34)
```

kryhet konversioni implicit i 12.34 në 12, vlera që i shoqërohet ndryshores numër i plotë m.

E njëjta mund të bëhet edhe me klnversioni eksplicit:

```
m = int(p);                         (= 12)
```

ku vlera e ndryshores p në mënyrë eksplicite shndërrohet në lloj int.

Këto shembuj gjithashtu ilustrojnë knversin:

```
r = double(3 * 5);          (= 15.0)
cout << int('a') << endl;   (Do të shtypet 97)
```

Për konversionin e llojit, mund të shfrytëzohet edhe frma që është tipikë për gjuhën C (angl. C-like), d.m.th., e trashëguar prej atij. Poashtu, shfrytzohet operatori ( )

i cili quhet operator cast, të quajtur edhe operator për hedhje. Ai përdoret ashtu që së pari jepet lloj te i cili shndërrohet vlera e shprehjes, por pas tij te kllapat vijn shprehja lloj i të cilës ndryshon.

Për shembull:

```
m = (int)(5 + 7.34)          (= 12)
```

Shembujt paraprak mund të shkruhen edhe në këtë mënyrë

```
k = (int)(5 + 7.34);         (= 12)
r = (double)(3 * 5);         (= 15.0)
cout << (int) 'a' << endl;   (do të shtypet )[36]
```

Gabi mi shpeshtë bëhet gjatë pjesëtimit të dy ndryshoreve numra të plotë. PËr shembull, nëse shuma e 8 numrave natyrorë është 100, vlera mesatare është 100 / 8 = 12.5:

```
        int shuma = 100;
        int numra = 8;
        float vleraMesatare = shuma / numra;//E pasaktë, pjesëtimi numra të plotë)
duhet   float vleraMesatare  = float(shuma ) / numra;
ose     float vleraMesatare  = shuma  / float(numra);
ose     float vleraMesatare  = 1.0 * shuma  / numra;     (Pa hedhje)
```

Te shembujt paraprak shfrytëzuam edhe urdhër të këtillë:

```
numriIParë  = numriIParë  + shifra;
```

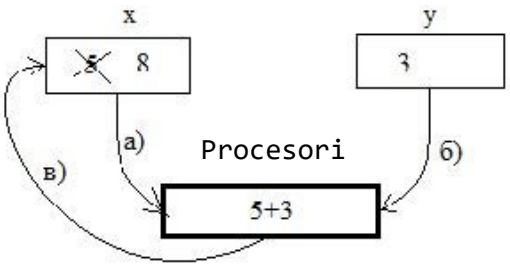
Nëse e shkruajmë këtë urdhër në formën matematikore, do të fitojmë barazim të pamundshëm:

```
x = x + y
```

e cila është e saktë nëse y = 0.

Por kjo shprehje në programim realizohet në mënyrë korrekte. Procesori së pari e jehson anën e djathtë, ashtu që e mbledh vlerën momentale të ndryshores x me vlerën e ndryshores y dhe shuma e fituar i shoqërohet si vlerë e re e ndryshores x.

Për shembull, nëse ndryshoret kanë vlera x = 5 dhe y = 3, urdhri do të realizohet sipas kësaj skeme.



a) Vlerat e x (= 5) barten te procesori.
b) Te ai shtohet vlera e y (= 3).
c) Shma e fituar vendohet si vlerë e re e x (= 8).

[36] Vlera e ASCII në shenjën 'a' është 97.

Nëse vlera e ndryshores x zmadhohet për 1 (themi, janë të inkriminuara), urdhri do të jetë:

```
x = x + 1;
```

Për realizimi e këtij urdhri, është future operator i veçantë, të quajtur o**perator për inkriminim**[37] (nagl. increment operator) ++. Urdhri paraprak i shkruar me këtë operator do të jetë:

```
++x;    ose     x++;
```

Përkatësisht me operatorin për inkriminim, është future edhe **operatori për dekrementim** (angl. decrement operator) – –, me të cilin zvogëlhet vlera e ndryshores për 1. Për shembull, urdhërat:

```
– –x;    ose     x– –;
```

kanë efekt të njëjtë sikurse edhe urdhri:

```
x = x - 1;
```

Shembulli 1.7.4

Me programin prej ***figura 1.7.4*** është ilustruar konversioni implicit dhe eksplicit.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {    // Konversioni i llojit të të dhënave
5
6       cout << "Konversioni implicit (vetëm në llojin e lartë)" << endl;
7       int k = 5;
8       double p;
9       p = k + 7.34;
10      cout << "5 + 7.34 = " << p << endl;
11      int m = p;
12      cout << "int m; \nm = 5 + 7.34 = " << m << endl;
13
14      cout << "\nKonversioni eksplicit (i detyruar) " << endl;
15      cout << "m = int(5 + 7.34) = " << int( p ) << endl;
16      cout << "m = (int)(5 + 7.34) = " << ( int ) p << endl;
17      cout << "int('a') = " << int( 'a' ) << endl;
18      cout << "(int)'a' = " << ( int ) 'a' << endl;
19      int zbir = 100;
20      int broevi = 8;
21      cout << "100 / 8 = " << shumë / numra   << endl;
22      cout << "float(100) / 8 = " << float( shumë   / numra   << endl;
23      cout << "100 / float(8) = " << shumë / float( numra   ) << endl;
24      cout << "1.0*100 / 8 = " << 1.0 * shumë / numra   << endl;
```

Figura 1.7.4

[37] Këto operacione më detalisht do t'i shqyrtojmë më vonë.

```
25
26      cout << endl;
27      system( "Color 17" );
28      system( "pause" );
29      return 0;
30  }
```

Figura 1.7.4 (vazhdim)

Dalja prej realizimit zë aplikacionit është:

```
Konversioni implicit (vetëm në llojin më të lartë)
5 + 7.34 = 12.34
int m;
m = 5 + 7.34 = 12

Konversionin eksplicit (të detyruar)
m = int(5 + 7.34) = 12
m = (int)(5 + 7.34) = 12
int('a') = 97
(int)'a' = 97
100 / 8 = 12
float(100) / 8 = 12.5
100 / float(8) = 12.5
1.0*100 / 8 = 12.5
```

Detyra për ushtrime

1. Të shkruhet program për njehsimin e perimetrit të ($P = 2r\pi$) dhe syprina ($S = r^2\pi$) të rrethit me rreze r = 18.3 cm.
2. Të shkruhet program për njehsimin e syprinës dhe vëllimit të paralelpipedit me brinjë a = 12.5 cm, b = 7.2 cm dhe c = 15.8 cm.
3. Të shkruhet program për zgjidhjen e barazimit: $5.27x - 23.15 = 0$.
4. Të shkruhet program për shndërrimin e këndit $\alpha = 36.27°$ në radian sipas formulës $\alpha^{rad} = \alpha\pi^{o} / 360$.
5. Të shkruhet program për njehsimin ei shpejtësisë së autombilit që për 5 rë kalon largësinë prej 465 kilometra.
6. Të shkruhet program për shtypjen e dhjetorit të tretë prej herësit të numrave real a dhe b, për a = 3 dhe b = 16.
7. Të shkruhet program për njehsimin e ndryshimit ndërmjet vlerave të numrit $\pi = 3.1415$ dhe shumës. $4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\frac{1}{13}-\frac{1}{15}\right)$.

Lloji i shenjës të të dhënave **char**

Të dhënat prej **llojit të shenjës** mund të kenë vlerë që çfarëdo qftë shenje numër i plotë. Atom mund të jenë ndryshore ose konstante të të dhënave. Deklarohen me fjalën **char**. Poashtu, shenja vendohet te gjysmëthnjzat, që të dallohen prej shenjës së njëjtë në tekst.

Shembuj:

```
const char taraba = '#';
char a;
char shenjë  = '+';
char denar = 'd';
char ascii = 120;
```

Të gjitha shenjat të cilat shfrytëzohen në gjuhën programore C++, në kompjuter reperzentohen (shkruajnë në memorie) me numra të plotë – kode, sipas tabelës ASCII (American Standard Code for Information Interchange). (Shiko **Datotekën B: Shenja ASCII**). Për shembull, shenja Q ka vlerë binare (kod) $01010001_2 = 81_{10}$, ndërsa shenja '=' ka kod $0011111_2 = 99_{10}$. Gjatë shtypjes së këtyre kodeve, shtypen shenjat përkatëse.

Të dhënat prej llojit të shenjës mund të trajtohen sikurse çfarëdo lloj numër të plotë, duke pasur llogari për vargun e vlerave. Me atom und të realizohen operacione aritmetike, ato mund të lexohen, të shtypen dhe të krahasohen.

Për shembull, nëse deklarohen ndryshorja procent me:

```
char përqindja;
```

shenja % mund të shoqëroj me çfarëdo qoftë prej këtyre dy urdhërave:

```
char përqindje  = '%';
përqindje  = 37;
```

Me këtë urdhër:

```
përqindja++;
```

inkrementohet vlerë numër i plotë të shenjës % prej 37 në 38, por me urdhrin:

```
cout << përqindje  << endl;
```

do të shtypet shenja &. Me të tjera fjalë, shenjar standarde ASCII janë renditur dhe vlera e tyre dekade është prej 0 (000000) deri 127 (1111111). 111).

Tabela e zgjeruar ASCII përmban 256 shenja me kode prej 0 në 255. Zgjerimi është me shenjat me kode prej 128 deri 255.

Pasi shenjat te C++ kanë vlera numra të plotë sipas tabelës ASCII[38], atom und të krahasohen. Për shembull, sheja '+' ka ASCII-vlerë 43_{16} ose 67_{10}, por ASCII-vlera e shenjës ':' është 58_{16} ose 88_{10}. Prandaj, mund të shkruhet '+' < ':'.

Vlerat dhe vargu i të dhënave prej llojit të shenjës janë dhënë te ***tabela 1.7.3.*** Kdet prej–128 deri –1 dhe prej 128 deri 255 janë kode të shenjave të njëjta.

Lloj	Vlera	Memoria
char	–128... 127	8 bit
unsigned char	0... 255	8 bit

Tabela 1.7.3

[38] Tabela standarde ASCII përmban 128 shenja, ndërsa tabela e zgjeruar ASCII përmban 256 shenja. Por për të përfshirë shenjat prej më shumë gjuhëve botërore, sot shfrytëzohen të ashtuquajtura unikod (angl. Unicode), i cili paraqet standard internacioal për kodim.

Vlera minimale dhe maksimale e të dhënave prej tipto char janë konstante:

```
char                    CHAR_MIN  = -128
                        CHAR_MAX  = 127
unsigned char           UCHAR_MAX = 255
```

Shembulli 1.7.5

Të njehsohet kamata për kapital të depnuar prej 100 000 denar nëse norma e kamatës është 7.5 %, sipas formulës kamatë $= \frac{\text{kapita} \cdot \text{norma e interest}}{100}$.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {    // Lloj i shenjës të të dhënave
5
6       char shenjaEPërqindjes '%';
7       double kapitali   = 100000.0;
8       float normaEKamatës  = 7.5f;
9       double sasiaEKamatës;
10      sasiaEKamatës = capital*normaEKamatës/100
11      cout << "Për kapitalin e deponuar prej" << capital    << "  denar i" << endl;
12      cout << "me nrmë të kamatës vjetore prej " << normaEKamatës
13          << "                << endl;
14      cout << "shenjaEPërqindjes    " <<fitohet kamata prej "  denar ." << endl;
15
16      system( "Color 17" );
17      system( "pause" );
18      return 0;
19  }
```

Figura 1.7.5

Një dalje sipas realizimit të tij është:

```
Për kapitalin e deponuar prej 100000 denar  me  .
nrmë të kamatës vjetore prej 7.5% fitohet
kamata prej 7500 denarë
Press any key to continue . . .
```

Detyra për ushtrime

1. Të shkruhet program me të cilin do të deklaroni aq shenja të ndryshoreve aq sa ka shkronja emri Juaj dhe çdonjërës do t'i shoqëroj nga një shkronjë. Pastaj, shtypni ashtu që në dalje do ta fitoni gjithë emrin.
2. Të shkruhet program për gjetjen e ndryshimit të numrave rendor të shkronjave 'S' dhe 'D' te tabela ASCII.
3. Çmimi i ndonjë prodhimi është 123.45 denarë, por gjatë një jave është shtrenjtuar për 15 % dhe pastaj është liruar për 8 %. Sa ka qenë çmimi në fund të javës? (Për shenjën % të shfrytëzohet ndryshorja e shenjës).

4. Të gjendet edhe shtypet shenja që i paraprin edhe shenja që e përcjell pas shenjës @ te tabela ASCII.
5. Të caktohet sa shkronja ka ndërmjet dy shkronjave prej alfabetit.
6. Të deklarohen dhe inicializohen dy ndryshore numra të plotë a dhe b të vlerave 12 345 dhe 678. Të deklarohet edhe në shenjë e ndryshores operator dhe t'i shoqërohet nga një prej shenjave: +, –, *, / dhe %. Pas çdo shoqërimi të shenjës, të shtypen a dhe b me shenjën e operatorit ndërmjet tyre, si edhe rezultati prej operacionit. Për rezultatin prej operacionit, të deklarohet ndryshorja përkatëse. Për shembull, për operator = '%' të shtypen 12 345 % 678 = 141.

Lloj logjik i të dhënave **bool**

Të dhënat vlera e të cilëve mund të jetë ose true ose false janë të **llojit logjik** (angl. logical type). Fjalët e rezervuara true dhe false janë konstante special në C++. Gjatë njehsimit të shprehjeve, konstantat logjike true dhe false kanë vlerë 1 ose 0 dhe lexohen sikur 1 dhe 0.

Të dhënat e llojeve logjike deklarohen me fjalën **bool**.

Shembuj:

```
bool Po Jo;
bool semafor = true;
```

Me të dhënat prej llojeve logjike mund të shfrytëzohen edhe **operatorët logjik** (angl. logical operators)[39]:

&& logjike ***DHE*** (AND)
|| logjike ***OSE*** (OR)
! logjike ***JO*** (NOT)

Këta operatorë quhen edhe **operator të kushtëzuar** (angl. conditional operators) pasi shpesh shfrytëzohen gjatë vlerave (true ose false) të ndonjë shprehje logjike të përbërë.

Operatorët logjik janë, në realitet, funksione, të quajtura **funksione të Bulit** (angl. boolean functions)[40], sipas matematikanit Xhorxh Bul (George Boole) i cili i ka përkufizuar.

Operatorët logjik janë përkufizuar me këtë tabelë.

JO/*NOT*/*!*	***DHE***/*AND*/*&&*	***OSE***/*OR*/*||*
!false = true	false && false = false	false || false = false
!true = false	false && true = false	false || true = true
	true && false = false	true || false = true
	true && true = true	true || true = true

Tabela 1.7.4

[39] Ekzistojnë edhe operator tjerë logjik.
[40] Ekzistojnë edhe të tjera funksione të Bulit.

Shembuj:

Vlera e këtyre shprehjeve për a = true dhe b = false do të jetë:

```
a)
(a || b) && (!a || b) = (true || false) && (false || false) = true && false = false

b)
(a && !b) && (!a || b) || (a || b) = (true && !false) && (!true || false) || (true || false) = (true && true) && (false || false) || (true || false) = true && false || true = false || true = true
```

Me të gjitha llojet paraprake të përmendura të të dhënave mund të shfrytëzohen **operatorët e relacionit** (angl. relational operators):

<	më i vogël
<=	më i vogël ose i barabartë
>	më e madhe
>=	më e madhe ose e barabartë
==	e barabartë
!=	e ndryshme

Të përmendim se operatori == nuk është „dy e barabartë", por operator binary për krahasim të dy operandëve. Për shembull, nëse z është ndryshore logjike, me urdhrin:

```
z = x == y;
```

së pari krahasohen vlerat e x dhe y dhe nëse janë të njëjta, z i shoqërohet vlerë true, por nëse nuk janë të njëjtë, z i shoqërohet vlera false.

Gjatë shfrytëzimit të operatorëve të relacioneve, duhet të kemi kujdes operndët të jenë të llojit të njëjtë. Në të kundërtën, mund të ndodh konversioni i detyruar i një lloj në tjetrin, i cili mund të sillet deri te rezultati i gabuar ose deri te dukuria e porosisë për gabim.

Për shembull, te urdhri:

```
bool Po Jo  = 1 == '1';
```

kryhet konversioni implicit i shenjës '1' në numër të plotë dhe pastaj krahasohet me konstante 1. Rezultati është false. Pse?

Operandët e relacioneve mund të jenë edhe prej llojit tjetër të thjeshtë. Gjatë shfrytëzimit të shprehjeve, ato në mënyrë të detyrueshme konvertojnë në lloj logjik, edhe atë: vlera 0 në false, por të gjitha vlerat jozero në true.

Për shembull, këto urdhëra:

```
double rreze  = 1.23;
bool Po Jo  = rreze  > 0;
```

realizohet ndryshorja korrekte dhe logjike Po Jo fiton vlerë true.

Gjatë krahasimit të numrave real, duhet të kemi kujdes se ato njehsohen përafërsisht, por jo saktë.

Për shembull, me këto urdhëra:

```
float a = 1.0f / 5;
Po Jo  = a == 0.2;
```

```
cout << "a = 1.0/5 = " << a << " a është e barabartë 0.2? " << Po Jo<< endl;
```

rezultati do të jetë false, d.m.th., 0 për ndryshoren Po Jo:

```
a = 1.0/5 = 0.2 a është e barabartë 0.2? 0
```

Por mund të ndodh operndët dhe gjithë shprehja të jenë shkruar në mënrë korrekte (përkthyesi të mos paraqet), por rezultati të jetë i gabuar. Për shembull, vlera e shprehjes:

```
bool suksesIShkëlqyeshëm = nota  == 4 || 5
```

është true, pavarësisht si është vlera e ndryshores nta pasi operandi i djathtë i operatorit || gjithmonë ka vlerë true.

Në realitet, gjykimi i drejtë duhet të jetë:

```
bool sukse  = nota  == 4 || nota  == 5;
```

që fiton vlerë false nëse nota është më e vogël se 4.

Ngjashëm, urdhri:

```
bool të renditur  = a < b < c;
```

është e saktë sintksisht por semantike nuk është. Kështu, për vlerën a = 3 dhe c = 5, do të japë rezultat të gabuar pasi së pari krahasohet 3 < b që jep rezultat true ose false, d.m.th., vlera 1 ose 0. Pastaj, krahasohet 1 < 5 ose 0 < 5 që gjithmonë jep rezultat true. Por për b = 6 duhet të fitohet rezultati false.

Urdhri i drejtë është:

```
bool të renditur  = a < b && b < c;
```

operatorët logjik dhe të relacionit shfrytëzohen në **shprehjet logjike** (angl. logical expressions), vlera e të cilit mund të jetë true ose false.

Shembuj:

```
int alfa = 60, x = 7, a = 3, b = 8, c = 2;
bool p = (alfa > 0) && (alfa < 90);
bool q = (x < -1) || (x > 1);
bool r = !(a < b + 5 * c) && (2 * c + 1 == 3 * b) || (c >= b);
cout << "p = " << p << "\nq = " << q << "\nr = " << r << endl;
```

Me këtë tabelë janë paraqotur shprehjet logjike ekuivalente të quajtura **rregullat e De Morganit** (angl. De Morgan's Laws).

Shprehje	Shprehje ekuivalente
!(a && b)	!a || !b
!(a || b)	!a && !b

Shprehjet te programet mund të jenë edhe maft të përbërë edhe tea to të shfrytëzohen operatorët aritmetik, logji dhe të relacionit. Gjatë njehsimit të tyre, respektohet prioriteti i operatrëve, edhe atë:

Me i lartë	Negacioni (NOT) !	unar +			unar –
	aritmetike	*	/	%	
		+	–		
	Të relacionit	<	<=	>	>=
		==	!=		

	logjike	(AND) &&						
		(OR) ||						
të ulët	Shqërim më	=	+=	–=	*=	/=	%=	

Njehsimi i shprehjeve logjike kryhet prej të majtës në të djathtë, me respektimin e prioritetit të operatorëve dhe kllapave

Në rastet e caktuara, C++ kryen vlerësim të kushtëzuar (angl. conditional evalua-tion, ose short-circuit) shprehjes.

Për shembull, gjatë njehsimit të shprehjes:

```
(alfa > 0) && (alfa < 90)
```

për alfa = -30^0, nënshprehja e majtë (alfa > 0) fiton vlerë false dhe pavarësisht prej vlers së nënshprehjes së djathtë (`alfa < 90`), vlera e gjithë shprehjes është `false`. Prandaj, në rastin e shprehjes me operatorin &&, nënshprehja e djathtë nuk njehsohet.

Ngjashëm, gjatë njehsimit të shprehjes:

```
(x < -1) || (x > 1)
```

për x = –2, vlera e nënshprehjes është true dhe pavarësisht prej vlerës së nënshprehjes së djathtë, gjithë shprehja fiton vlerë true. Prandaj, nënshprehja e djathtë nuk njehsohet.

Shembulli 1.7.6

Me këtë shembull shtypet tabela e funksioneve logjike AND, OR dhe NOT, të njehsuara varësisht prej vlerave të dy ndryshoreve logjike a dhe b.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {    // Lloj logjik i të dhënave
5
6       bool a, b;
7       cout << "----------------------------------------" << endl;
8       cout << "a    b     a AND b  a OR b   NOT a   NOT b" << endl;
9       cout << "----------------------------------------" << endl;
10      a = true;
11      b = true;
12      cout << a << "    " << b << "        " << ( a && b ) << "       "
13          << ( a || b ) << "        " << ( !a ) << "       " << ( !b ) << endl;
14      b = false;
15      cout << a << "    " << b << "        " << ( a && b ) << "       "
16          << ( a || b ) << "        " << ( !a ) << "       " << ( !b ) << endl;
17      a = false;
18      b = true;
19      cout << a << "    " << b << "        " << ( a && b ) << "       "
20          << ( a || b ) << "        " << ( !a ) << "       " << ( !b ) << endl;
21      b = false;
```

Figura 1.7.6

```
22      cout << a << "     " << b << "         " << ( a && b ) << "         "
23          << ( a || b ) << "         " << ( !a ) << "         " << ( !b ) << endl;
24      cout << "-----------------------------------------" << endl;
25
26      cout << endl;
27      system( "Color 17" );
28      system( "pause" );
29      return 0;
30  }
```

Figura 1.7.6(vazhdim)

Pas realizimit të programit, dalja është:

```
-----------------------------------------
a     b      a AND b   a OR b    NOT a    NOT b
-----------------------------------------
1     1         1         1         0         0
1     0         0         1         0         1
0     1         0         1         1         0
0     0         0         0         1         1
-----------------------------------------

Press any key to continue . . .
```

Shembulli 1.7.7

Me këtë shembull ilustrohet shfrytëzimi i llojit logjik të të dhënave.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {    // Lloj logjik i të dhënave
5
6       const double G = 9.81;  // Deklarimi dhe inicializimi i konstantes reale nxitimi
7       int këndiAlfa;          // Deklarimi i ndryshores numra të plotë
8       double nxitimi          // Deklarimi i ndryshores numra real
9       char shenja= '*';       // Deklarimi dhe inicializimi i ndryshores me shenjë
10      bool Po Jo;             // Deklarimimi ndryshores logjike
11      Po Jo = shenja== '*';   // Nëse vlera e ndryshores shenjë është *, atëherë
12                              // Ndryshorja logjike Po Jo fiton vlerë 1 (true)
13                              // në të kundërtën fiton vlerë 0 (false)
14      cout << PoJo << ": shenja është " << shenja << endl;
15      nxitimi       = 9.9;
16      PoJo = nxitimi       < G; // Nëse vlera e ndryshores nxitim është më e vogël
17                              // prej vlerës së konstantës G, atëherë ndryshorja logjike
18                              // Ndryshorja Po Jo fiton vlerë 1 (true)
19                              // në të kundërtën fiton vlerë 0 (false)
20      cout << PoJo << ": " << nxitimi << " < " << G << endl;
21              = 60;
22      PoJo = ( këndiAlfa > 0 ) && ( këndiAlfa < 90 );
23      cout << PoJo << ": " << agolAlfa << " kënd i ngushtë " << endl;
```

Figura 1.7.7

```
24
25      cout << endl;
26      system( "Color 17" );
27      system( "pause" );
28      return 0;
29  }
```

Figura 1.7.7 (vazhdim)

Pas realizimit të programit, fitohet kjo dalje:

```
1: Shenja është   *
0: 9.9 < 9.81
1: 60 e kënd i ngushtë.

Press any key to continue . . .
```

Operatorët me bit

C++ ka edhe një grup operator logjik, të quajtur **operator me bite** (angl. bitwise operators). Me këta operatorë, operacionet logjike realizohen bit pas biti.

&	logjike ***DHE*** për bite	~	komplement
|	logjike ***OSE*** për bite	<<	zhvendosje majtas
^	në veçanti ***OSE*** për bite	>	zhvendosje djathtas

Tabela e operacioneve logjike &, | dhe ^ mbi bitet p dhe q është:

p	q	p & q	p | q	p ^ q
1	1	1	1	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

Komplement i numri logjik fitohet me negationin e bitëve, 1 në 0 dhe 0 në 1. Për shembull, komplement i numri

$m = 000\ldots\ 00110101\ (= 53_{10})$

është numri

$\bar{m} = 111\ldots\ 11001010\ (= -54_{10})$.

Operatorët për zhvendosje shfrytëzohen për zhvendosje (angl. shifting) të bitëve majtas se djathtas. Prandaj, quhen operator për zhvendosje ose operator **shift**. Për shembull, me m << 1, zhvendose n bitët të numrit m për 1 vend majtas (= 000… 01101010 = 106_{10}), por me m >> 1, bitët zhvendose n për 1 vende djathtas.

Shembulli 1.7.8

Le të jenë dhënë numrat binar a = 00000101 (= 5_{10}), b = 00001110 (= 14_{10}). Të shtypet vlera e operacioneve:

a & b, a | b, a ^ b, ~a, a << 1, a >> 1 dhe –a >> 1.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {    // Operator me bit
5       int a = 5, b = 14;
6       cout << "-------------" << endl;
7       cout << " a=00000101" << endl;
8       cout << " b=00001110" << endl;
9       cout << "-------------" << endl;
10      cout << "a & b   | " << ( a & b ) << endl;
11      cout << "a | b   | " << ( a | b ) << endl;
12      cout << "a ^ b   | " << ( a ^ b ) << endl;
13      cout << "~a      | " << ( ~a ) << endl;
14      cout << "a << 1  | " << ( a << 1 ) << endl;
15      cout << "a >> 1  | " << ( a >> 1 ) << endl;
16      cout << "-a >> 1 | " << ( -a >> 1 ) << endl;
17      cout << "-------------" << endl;
18
19      cout << endl;
20      system( "Color 17" );
21      system( "pause" );
22      return 0;
23  }
```

Figura 1.7.8

Me programin fitohet kjo tabelë:

Te operatorët me bite, operacioni logjik realizohet mbi bitët përkatës. Rezultati mund të jetë çfarëdo vlere numër i plotë. Prandaj,

5 & 14 = 00000101 & 00001110 = 00000100 = 4.

Duhet të vërehet ndryshimi ndërmjet operatorëve logjik dhe operatorëve me bite. Operatorët logjik zbatohen mbi një ose mbi dy operand, ku çdo vlerë e ndryshueshme prej zeros trajtohet si pikë logjike. Poashtu, rezultati i operacionit mund të jetë 0 ose 1.

Për shembull, për vlerat e tabelës paraprake: a && b = 1 dhe a || b = 1, pasi a dhe b kanë vlerë të ndryshme prej 0.

Gjatë shfrytëzimit të operatorëve logjik && dhe ||, duhet pasur kujdes të mos zëvendësohen me operatorët me bite & dhe |. Për shembull, për a = 5 dhe b = 14, shprehjet:

```
a | b = 15
a || b = 1
```

kanë rezultat të ndryshëm.

Forma e shkurtër e operatorëve me bit është:

```
&=      |=      ^=      <<=     dhe     >>=.
```

Detyra për ushtrime

1. Të shkruhet program me të cilin do të njehsohet vlera (true ose false, d.m.th., 1 ose 1) e shprehjes logjike (a || !b) && !(a || b) për a = true dhe b = false.
2. Të shkruhet program për gjetjen e vlerës së shprehjes:
   !(a – b < 5 * c) && (b * c % a + 1 == a * b) || (c < b) për a = 7, b = 3 dhe c = 4.
3. Të shkruhet program për gjetjen e prodhimit dhe herësit të numrit të plotë 12 dhe 25. (*Udhëzim*: Shumëzimi i numrit të plotë m me 2n kryhet me zhvendosjen e pikës dhjetore për n vende djathtas, por pjesëtimi me zhvendosje të pikës dhjetore për n vende majtas. Prandaj, mund të shfrytëzohen operatorët për zhvendosje >> dhe <<).

Lloj i numërueshëm i të dhënave

Të dhënat prej llojit të numërueshëm mund të fitojë vlera prej përpara të bashkësisë së dhënë (lista) e konstantave, të cilat patjetër të jenë identifikator, por jo numra. Gjithashtu, e dimë se vlerat e konstantave shkruhen me shkronja të mëdha.

Lloj i numërueshëm i të dhënave përkufizohet me fjalën **enum** (angl. enumerate), por vlerat te lista vendohen në kllapa të mëdha. Pas pjesës dalëse prej kllapave të mëdha, nuk vendohet shenja e pikëpresjes.

Shembuj:

```
enum ngjyraThemelore  {E KUQE, E GJELBËR, E KALTËR, E VERDHË, KAFENE}
enum kohëVjetore  {PRANVERË, VERË, VJESHTË, DIMËR }
enum saktëEPasaktë  {TRUE, FALSE}
```

Çdo emër te lista ka numrin e vet rendor, edhe atë: 0, 1, 2, 3 etj. Për shembull, numri rendor e KUQE është 0, e GJELBËR është 1, e KALTËR është 2.

Te shembulli i dytë përkufizohen 3 lloje të numërueshme, edhe atë: ngjyraThemelore, kohëVjetore dhe saktëEPasaktë.

Përkufizimet vijuese të llojeve të numërueshme nuk janë të drejta pasi konstantat te listat nuk janë identifikator, d.m.th., nuk janë formuar sipas rregullave për identifikator në C++:

```
enum numriINotave  {1, 2, 3, 4, 5}
enum numriIOperatorëve  {'+', '-', '*', '/', '%'}
```

Këto dy përkufizime të llojeve të numërueshme janë të drejta:

```
enum ditë  {HËNË, MART, MËR, EJT, PREM, SHT, DIE}
enum vikend {SHTUN, DIELË}
```

Por nuk mund të shfrytëzohen në programin e njëjtë pasi konstanta E DIEL ripërkufizohet me përkufizimin e dytë.

```
denovi d = E_DIEL;
vikend v = E_DIEL
```

enum vikend::E_DIEL = 1

Search Online

a value of type "vikend" cannot be used to initialize an entity of type " ditë "

Deklarata e ndryshoreve prej llojit të numërueshëm kryhet me specifikuar llojin, por pas tij specifikohet lista e ndryshoreve.

Shembuj:

```
ngjyraThemelore  ku, gj, ka;
kohëVjetore  vj, di, ve;
```

Këto ndryshore mund të fitojnë vlera vetëm prej atyre konstantave që janë përmend te lista gjatë përkufizimit të llojit:

```
ka  = KALTËR;
di  = DIMËR;
```

Ndryshoret prej llojit të numërueshëm nuk mund të shtypen. Gjatë shtypjes së ndryshoreve ka dhe di, shtypen numrat e tyre rendore te lista me përkufizim:

```
s  = 2
zi = 3
```

pasi numri rendor i KALTËR është 2, por në DIMËR është 3.

Ndryshoret prej llokit të numërueshëm nuk mund të inkremntohen në mënyrë implcite. Për shembull, ky urdhër:

```
ka = ka + 1;
```

nuk mund as të kompajlohet. Menjëherë paraqitet gabimi.

```
s = s + 1;
```

cou Error: a value of type "int" cannot be assigned to an entity of type "ngjyraThemelore"

Gabimi paraqitet pasi gjatë mbledhjes me 1, ndryshorja ka detyrimisht konvertohet në numër të plotë 2 (numri rendor te lista e përkufizimit) dhe mblidhet me 1. Rezultati është vlerë numër i plotë 3 e cila nuk mund të shoqërohet te ndryshorja e numërueshme ka.

Kjo mund të shmanget me konversion eksplicit

```
ka  = ngjyraThemelore (ka  + 1);
```

sipas të cilës ndryshorja ka do të fitn vlerën e VERDHË.

Konstantave prej llojit të numërueshëm mund t'u jepen vlera (vend numra rendor) dhe gjatë përkufizimit të llojit.

Shembuj:

```
enum nota  {SHKËLQYSHËM  = 5, SHMIRË  = 4, MIRË  = 3, MJAFTUESHËM  = 2};
enum muaj  {JANAR  = 1, SHKURT, MARS, PRILL, MAJ, QERSHOR, KORRIK,
            GUSHT, SHTATOR, TETOR, NËNTOR, DHJETOR}
```

Te shembulli i dytë, numrat rendor prej JANARI deri DHJETOR do të jenë prej 1 deri 12, por jo prej 0 deri 11.

Të dhënat prej llojit të numërueshëm mund të krahasohen. Poashtu, krahasohen numrat e tyre rendor te listat e përkufizimit.

Mund të deklarohen dhe lloje anonime të numërueshme (angl. anonymous enume-ration type). Për shembull:

```
enum {mollë, dardhë, pjeshkë, kumbull, qershi, vishnje, kajsi};
```

Përkufizimi i llojit të numërueshëm kryhet para funksionit main(), ***figura 1.7.9***.

Shembulli 1.7.9

Me këtë shembull ilustrohet përkufizimi për llojin e numërueshëm dhe deklarimi i ndryshoreve prej llojit të numërueshëm.

Pas realizimit të programit prej ***figura 1.7.9***, fitohet kjo dalje:

```
c Ngjyra = 0, s Ngjyra = 5
Ngjyrat a janë të njëjta ?     0
Vlera e di =           = DIMRI është 3
```

```
1   #include <iostream>
2   using namespace std;
3
4   enum ngjyraThemelore{
5       KUQE, GJELBËR, KALTËR
6   };
7   enum kohëVjetore
8       PRANVERË, VERË, VJESHTË,DIMËR
9   };
10
11  int main() {    // Të dhënat e llojit të numërueshëm
12    ngjyraThemelore gjyraku,Ngjyraka;
13     kohëVjetore di  = DIMËR
14       bool saktëPaSaktë ;
15
16        Ngjyraku = KUQE
17        Ngjyraka =KALTËR
18        saktëPaSaktë = Ngjyraku == Ngjyraka
19        cout << Ngjyraku= " <<Ngjyraku<< ",Ngjyraka = " <<Ngjyraka<< endl;
20        cout << "Ngjyrat a janë të njëjta"<< saktëPaSaktë   << endl;
21
22        cout <<    "vlera e di=DIMËR është   " << zi << endl;
23
24        cout << endl;
25        system( "Color 17" );
26        system( "pause" );
27        return 0;
28  }
```

Figura 1.7.9

Detyra për ushtrime

1. Shkruani program te i cili do të përkufizoni lloj të numërueshëm ditëNëJavë, lista e konstantave të cilave do t'i përmban emrat e ditëve prej të HËNËS deri të DIELËN. Pastaj deklaroni shtatë ndryshore të llojit ditëNëJavë dhe çdonjërës shoqëroni nga një konstante prej llojit të numërueshëm ditëNëJavë sipas zgjedhjes së rastit. Deklaroni edhe një ndryshore të llojit ditëNëJavë, për shembull, ditë dhe shoqëroni vlerë MËRKUR. Pastaj, deklaroni ndryshore logjike saktëPaSaktë, të cilës do t'i shoqëroni vlerë që fitohet prej krahasimit të përmbajtjes së ndryshores ditë me çdonjërën prej shtatë ndryshoreve. Shtypni vlerat që d t'i fiton ndryshorja saktëPaSaktë. (Shiko programin prej ***figura 1.7.9***).
2. Të shkruhet program te i cili do të deklaroni ndryshore prej llojit të numërueshëm orëNëDitë:

```
enum orëNëDitë  {
   HËN  = 4, MAR  = 6, MËR  = 5, EJT  = 5, PRE  = 3 };
```

   Të njehsohet numri i përgjithshëm i orëve në javë.

3. Të shkruhet program te i cili do të deklaroni ndryshore prej llojit të numërueshëm nota. Përkufizimi i llojit nota përmban emra të 12 lëndëve me nota për çdo lëndë. Për shembull, {MAQ = 4, INF = 5...}. Të njehsohet suksesi mesatar i nxënësit.

Stringje

Te nëntitujt **Urdhri për shtypje** dhe **Llojet e të dhënave** prej nënpikës **1.6 Hyrje në C++**, u njohtëm me konceptin **string** (angl. string). Stringjet janë të dhëna të përbëra prej shenjave të të dhënave si sekuenca prej shenjave. Të dhënat e këtilla trajtohen si struktura të veçanta prej llojit të thjeshtë char, por quhen lloj string.

Këto të dhëna quhen edhe **të dhëna tekstuale** pasi vlera e tyre është tekst i vendosur në thonjëza, d.m.th., konstante tekstuale (string). Për shembull, string-konstante në C++ janë:

```
"Përshëndetje, si jeni ? "
"rr. Goce Delçev numër 123."
"Maqedonia vend biblik."
```

String-konstantat në C++ mund të përmbajnë shkronja (të mëdha dhe të vogla prej alfabetit), shifra dhe shenja special, sikurse +, –, *, / dhe $.

Karakteristikat themelore të string-konstantave janë:

- Shkruen në zhonjza.
- Çdo shenjë ka indkes, duke filluar prej më të majtit i cili ka indeks 0.
- Gjatësia e konstantes tekstuale është e barabartë me numrin e shenjave, duke future edhe vendet e zbrazëta.

- Gjatësia maksimale e një string-konstante është 2 048 bajt.

Ndryshoret e llojit string mund të deklarohen si edhe ndryshore prej çfarëdo lloj dhe atyre mund t'u shoqërohet string-konstante.

Shembuj:

```
string emriIm;              // Deklarata e ndryshores emriIm
string Po Jo;
string dataSot  = "01.01.2017"; //Deklarata dhe inicializimi
string dataSot("01.01.2017");  // ose (njëjtë) deklarata dhe
                                  // inicializimi
string dyje(60, '*');       // Deklarata dhe inicializimi
```

Shoqërimi i vlerës së ndryshores së deklaruar prej llojit string kryhet me shenjën për shoqërim =.

Shembuj:

```
emriIm  = "Gjorgji Jovancevski";
Po Jo  = 'D';                       // Kjo është e lejuar
dataNesër  = "21 qershor viti 2053 ";
```

Thamë se çdo shenjë te stringu ka indeksin e vet. Për shembull, emriIm[0] është 'G', emriIm[5] është 'j' , dataNesër[11] është '3' etj.

Të theksojmë se të dhënat "A" dhe 'A' nuk janë të njëjta. E dhëna e parë është prej llojit string, ndërsa e dhëna e dytë është i llojit char.

Me stringet mund të kryhen operacione të ndryshme, sikurse: bashkimi (e përmendëm te nëntitulli **Llojet e të dhënave**), krahasimi (angl. comparation) i dy stringeve, caktimi i gjatësisë (angl. length) të stringut dhe shumë të tjera.

Këto operacione realizohen me funksione prej bibliotekës <string> të *bibliotekave standard* në ***C++***. Prandaj, për t'i shfrytëzuar te programet, por edhe për programet të kyçet kjo bibliotekë, me direktivën #include:

```
#include <iostream>
#include <string>
using namespace std;
```

Shembulli 1.7.10

Me programin te ***figura 1.7.10*** është ilustruar shfrytzimi i string ndtyshoreve dhe string-konstanta.

Një dalje prej realizimit të programit është:

```
Dy maqedonasit më të nëdhenj kanë qenë:
Aleksandër Filip MAKEDONSKI dhe Goce Nikolla MAKEDONSKI
Aleksandër Filip MAKEDONSKI ka jetuar para Goce Nikolla MAKEDONSKI
Press any key to continue . . .
```

```
#include <iostream>
#include <string>
using namespace std;

const string MBIEMRI  = "MAKEDONSKI";
const string vendIZbrazët   = " ";

int main() {     // String të të dhënave

    string emriIm  = "Aleksandar Filip";
    string emriYt   = "Goce Nikola";

    string maqedonasMëTëMëdhenj = emriYt  + vendIZbrazët + mbiemri
    maqedonasMëTëMëdhenj += " i ";
    maqedonasMëTëMëdhenj += emriYt  + vendIZbrazët  + mbiemri
    cout <<Dy maqedonasit më të mëdhenj kanë qenë " << endl;
    cout <<meqedonasitMëTëMëdhenj<< endl;

    bool Po Jo = emriIm emriYt > mbiemri;
    cout << boolalpha <<Po Jo << ": " <<       : " <<
        <<          << "                " <<          <<
        <<          << endl;

    cout << endl;
    system( "Color 17" );
    system( "pause" );
    return 0;
}
```

Figura 1.7.10

Detyra për ushtrime

1. Të shkruhet program te i cili do të deklaroni dy stringje të ndryshueshme te të cilat u shoqërohen vlera emri Juaj dhe mbiemriJuaj. Të deklarohet edhe ndryshorja e shenjës të cilës i shoqërohet vlerë shkronja e parë prej emrit të prindit Tuaj. Të bashkohen të tre nryshoret dhe të shtypen në formë sikurse ky shembull:
   `Gjorgji J. Jovancevski`
2. Të shkruhet program te i cili do të deklarohen ndryshoret stringje dhe do t'u shoqërohen këto stringje-konstante: "GRAF", "LOG", "BIO", "IJA" dhe "GEO". Pastaj, me bashkim të formohen dhe shtypen fjalët: BIOGRAFIJA, BIOLOGIJA, GEOGRAFIJA dhe GEOLOGIJA.
3. Të shkruhet program te i cili do të deklarohen dy stringje të ndryshoreve dhe dy ndryshore të shenjave dhe t'u shoqërohen konstantat: "KAL", "LAK", 'A' dhe 'B'. Pastaj, me bashkimin e tyre të formohet fjala më e gjatë palindrome që ka

domethënie. (Palindron është fjalë ose fjali, e cila lexohet njëjtë edhe prej përpara edhe prej prapa. Për shembull, palindrome janë: oko, dovod, elelesevesele etj.).

Riemërtimi i llojit të të dhënave

Shpesh, për shkak të pastërtisë të programit dhe gjetja më e lehtë në atë, për emrat të llojeve të ndërtuara të të dhënave është e dësirueshme të shfrytëzohen sinonime (emra tjerë) të cilët do të na kujtojnë në diçka.

Për shembull, me këto deklarata:

```
int i, j, k;
double x, y, z;
char a, b;
bool Po Jo, gjetur;
```

deklarohen ndryshoret të llojit int, double, char dhe bool.

Në C++ mund të hyhet edhe tjetër emër për ndonjë lloj ekzistues. (Poashtu, nuk krijohet lloj i ri). Vargu i vlerave dhe peracioneve me emrin e ri të llojit janë identike me llojin ekzistues.

Për riemërtimin e lojit, shfrytëzohet fjala kyçe **typedef**.

Për shembull:

```
typedef int numra të plotë;
typedef double real;
typedef char shenja;
typedef bool logjike;
```

Pastaj, mund të kryhet deklarimi me llojin numra të plotë të barabartë sikurse me llojin int, me llojin real të barabartë sikurse lloj double etj.

Për shembull:

```
int numri  = 5;
numër i plotë1  = 8;
char shenja  = '$';
shenja  shenjë1  = '@';
bool Po Jo  = 1;
logicki Po Jo1  = true;
cout << numër  << '' << numër1 << endl;
cout << shenjë  << ' ' << shenjë1 << endl;
cout << Po Jo  << ' ' << boolalpha << Po Jo1 << endl;
```

Dalja prej sistemit programor lart është

```
5 8
$ @
1 true

Press any key to continue . . .
```

Caktimi automatik i llojit

Lloj i ndryshores gjatë deklarimit të saj mund të caktohet sipas llojit të të dhënës me të cilën inicializohet ose i shoqërohet vlera. Kjo bëhet me fjalën kyçe **auto**.

Për shembull, nëse deklarata e ndryshores te program prej ***figura 1.7.10*** jepet me fjlën auto, program do të Realizon në mënyrë korrekte:

```
auto emriIm  = "Aleksandar Filip";
auto emriYt  = "Goce Nikola";
auto maqedonasMëTëMëdhenj  = emriIm  + vendIZbrazët  + mbiemri;
auto Po Jo  = emriIm  < emriYt ;
```

Fjala kyçe auto më së shpeshti shfrytëzohet kur shprehja për inicializim të ndryshores është e përbërë, d.m.th., kur te ai shfrytëzohen lloje të ndryshme të të dhënave dhe të funksioneve dhe kur nuk mund menjëherë të caktohet rezultati gjat të njehsuarit e shprehjes. PËrkthyesi e cakton llojn e rezultatit gjatë njehsimit të shprehjes.

Pyetje për kontroll të njohurive

1. Cilat lloje të të dhënave numra të plotë (sipas vargut të vlerave) i dini?
2. Çfarë vlera fitojnë ndryshoret me numra të plotë të deklaruara si un-signed?
3. Ça ndodh nëse ndryshores short i shoqërohet vlera 100 000?
4. Përmend operatorët për operione me numra të plotë.
5. Përmend disa shembuj për deklarim dhe deklarimi dhe inicializimi të ndryshoreve prej llojit të numrave të plotë.
6. Përmendi operatorët aritmetik për numrat e plotë në formën e shkurtër.
7. Çka do të thotë operatori % =?
8. Për cilat të dhëna themi se janë të llojit real?
9. Cilat lloje të dhënave reale i ka në C++?
10. Përmend disa shembuj për deklarim dhe deklarimi dhe inicializzimi i ndryshoreve prej llojit real.
11. Sqaro paraqitjen e të dhënave dhjetore me pikën lëvizëse.
12. Sqaro formatet për shtypje të llojit real të të dhënave.
13. Paraqitni numrin 34400.811 në e-format.
14. Përmend disa shembuj për deklarim dhe deklarimi dhe inicializzimi i ndryshoreve prej llojit real.
15. Çka është konversioni implicit i llojit dhe se si quhet ndryshe?
16. Çka është konversioni eksplicite i llojit dhe se si quhet ndryshe?
17. Te cilat lloje mund të realizohet konversioni implicit i llojit int?

18. Si realizohet konversioni eksplicit prej llojit më të lartë në më të ulët të të dhënës?
19. Shkruani urdhër për konversion të shprehjes 0.12 * 5.3 + 123 në llojin int?
20. Sqaro urdhrin: x = x = y;
21. Sqaro çka do të thotë inkrementimi, kurse çka dekrementimi.
22. Cili është operatori për inkremenim, por cili për dekrementim?
23. Përmend shembuh për deklarim të ndryshoreve me shenja.
24. Sa hapësirë të memories zen lloj char?
25. Cilat konstante i dini? Pse nuk ka të tjera?
26. Përmend operatorët logjik.
27. Cilat operacione të relacioneve i dini? Përmend operatorët përkatës.
28. Cili do të jetë rezultati prej shprehjes: !(a && b) || (!a || b), për a = true dhe b = false?
29. A janë të drejta deklarimi dhe inicializimi: bool Po Jo = x > y < z;?
30. Përmend operatorët me bite.
31. Cakto rezultatin prej operacioneve: a & b, a | b dhe a ^ b, për a = 00010110 dhe b = 00010101.
32. Si janë vlerat e të dhënave prej llojit të numërueshëm?
33. Ku vendohet përkufizimi i llojit të numërueshëm të dhënës?
34. Përmend shembull për përkufizim të llojit të numërueshëm dhe për deklarim të ndryshores prej atij lloj.
35. Si shtypen të dhënat prej llojit të numërueshëm?
36. Lloj tekstual i të dhënave a bën pjesë në llojet e thjeshta?
37. Shkruaj shembull për deklarim të string-konstantes.
38. Shkruaj shembull për deklarim të string ndryshore.
39. Nëse është përkufizuar

```
typedef int celi;
```

atëherë prej cilit lloj të thjesht do të jenë ndryshoret të deklaruara me të

```
plota a, b, c; ?
```

40. Cila fjalë shfrytëzohet gjatë inicializimit të ndryshores që të mund ta cakton llojin e tij?

1.8 Leximi dhe shtypja e të dhënave

Leximi dhe shtypja e të dhënave numerike

Leximi i të dhënave në C++ është organizuar si **përrua** (angl. stream) prej shenjve. Përrua e hyrjes dhe daljes standarde kryhet me ndihmën e funksioneve (rutinat[41], angl. routines), të cilat gjenden në bibliotekë <iostream>, që, tani, kyçet në çdo program me direktivat:

```
#include <iostream>
```

Në bibliotekën <iostream> janë përkufizuar operatorët <<[42] dhe >>, urdhërat cin, cout, endl dhe të tjera, të nevojshme për realizimin e operacioneve hyrëse dhe dalëse.

Gjithashtu, te program patjetër të kyçen edhe biblioteka të tjera të cilat përmbajnë funksione për realizim të operacioneve të caktuara. Për shembull, nëse në program shfrytëzojmë stringjet, duhet të kyçet biblioteka <string> me direktivën #include:

```
#include <string>
```

Vijat programore të cilat fillojnë me direktivën #include nuk i përkthen përkthyesi, por përpunohen me program të veçantë të quajtur **paraprocesor** (angl. preprocesor). Ai i kaç në program të gjitha datotekat (angl. files) të cilat janë përmendur te direktivat #include. Këto datoteka vendohen në kllapa këndore, me të cilat informohen paraprocesorët e tyre t'i kërkon në të ashtuquajturat direktoriume include prej ***bibliotekës standard të C++***. Ato quhen **heder-datoteka** (angl. header files).

Operatorët për lexim dhe shtypje të të dhënave janë:

```
  >     operatori hyrës ,
 <<     operatori dalës .
```

Operatori << domethënë „dërgoe në", por operatori >> domethënë „merre prej". Urdhërat për hyrje dhe dalje janë:

```
cin Urdhri për futje të vlerave nëpërmjet tastaturës
        (për përrur hyrëse standarde),
cout    Urdhri për lexim të vlerave në ekran
        (për përrur dalës standarde).
```

Gjatë leximit të vlerave të të dhënave, kapërcehen vendet e zbrazëta, tabulatorët dhe shenjave për fund të rreshtit (angl. return). Leximi i vlerave fillon prej shenjës së parë që nuk është blanko.

Për shembull, nëse janë deklaruar ndryshoret i dhe x:

```
int i;
double x;
```

[41] Programe të shkurtëra.
[42] Këtë operator e shfrytëzuam dhe deri tani për shtypje.

leximi i vlerave për ato prej tastaturës kryhet me urdhër:

```
cin >> i >> x;
```

Nëse te ndryshoret numëriplotë dhe numrireal u shoqërojmë vlera:

```
numriplotë  = 123;
numrireal  = 456.78;
```

atëherë me urdhrin:

```
cout << numriplotë  << numrireal;
```

do të shtypet

```
123456.78
```

Për shqyrtimina e vlerave të shtypura shfrytëzohen një vend i zbrazët si shenjë konstante. Kështu, me këtë urdhër

```
cout << numriplotë  << ' ' << numrireal  << endl;
```

do të shtypet:

```
123 456.78
```

endl (të cilin e shfrytëzuam edhe pararakisht) është operatori për fund të vijës, d.m.th., shërben për kalimin në vijën vijuese.

Nëse duhet të lehet më shumë se një vend të zbrazët, zakonisht shfrytëzohet string-konstanta.

Për shembull, me urdhrin:

```
cout << numriPlotë  <<""<< numriReal  << endl;
```

do të shtypet:

```
123    456.78
```

Për shtypjen në rresht të ri, shfrytëzohet shenja '\n', që e sqaruam she e shfrytëzuam në shembujt e deritanishëm.

Për shembull, me të dy urdhërat:

```
cout << numriplotë  << '\n' << numrireal  << '\n';
cout << numriplotë  << endl << numrireal  << endl;
```

do të shtypet:

```
123
456.78
```

Disa shenja specifike, të cilat kanë domethënie speciale te urdhri për dalje patjetër të shtypen, me shfrytëzimin e sekuencave hyrëse (eskejp) sekuenca, të përmendura te nëntitujt **Urdhri për shtypje** prej nëntitullit **1.6 Hyrje në C++.**

Kmentimet në C++ të cilët janë në një vijë shkruhen me vendosjen e shenjës // para komentimit.

Komentimi i cili nuk është deri në fund të vijës vendohet ndërmjet shenjave /* dhe */. Për shembull:

```
/*     Me këtë program përpunohet vizita më e thjeshtë
       me të dhënat themelore:
       - emri i firmës
       - telefoni
       - e-mail
*/
```

Direktiva paraprocesorit:

```
#include <iostream>
```

i tregon paraprocesorit ta kyç datotekën <iostream>, e cila përmban përkufizime të funksioneve për operacione hyrëse-dalëse.

Ndryshoret te C++ mund të deklaraohen kudo te programi, por mund të shfrytëzohen sipas vijës deklarimit, por jo para saj.

Urdhri:

```
cout << "Futni numrin e parë të plotë: ";
```

e shfrytëzon përruan dalëse standarde cout dhe operatorin << që të shtypet teksti i përmendur në ekran prej monitorit. Kur urdhri paraprak do të realizohet, ai i dërgon përrua të shenjave„Futni numrin e parë të plotë" te dalja standarde cout. Këtë urdhër do të mund ta lexojmë sikurse: te cout dërgohet teksti„Futni numrin e parë të plotë" si përrua prej shenjave.

Urdhri vijues:

```
cin >> numri1;
```

e shfrytëzon përruan e hyrjes standarde cin dhe operatori >> për marrjen e të dhënave të cilat futen nëpërmjet tastaturës si varg prej shenjave. Ky urdhër mund të lexohet si „cin e lexon përruan prej shenjave të cilat futen nëpërmjet tastaturës dhe ia shoqëron numri1". Në rastin konkret, shenjat të cilat futen janë shifrat e numrit, të cilat konvertohen në numër të plotë dhe ai i shoqërohet ndryshorja numri i plotë nurmi1.

Do të përmendim disa shembuj.

Shembuj

Shembulli 1.8.1

Te ***figura 1.8.1*** është dhënë program i thjeshtë në C++ për mbledhjen e dy numrave, ku do të ilustrojmë disa prej vetive rë shqyrtuara te pjesa paraprake prej tekstit.

```
1   // Mbledhja e dy numrave
2   #include <iostream>
3   using namespace std;
4
5   int main() {                    // Fillimi i funksionit kryesor
6       int numri1;                 // Deklarimi i ndryshores numër i plotë1
        cout <<                           ";
8       cin >> numri1;              // Leximi i vlerës së ndryshores
9       int numri2 shuma;           // Deklarimi i numrave të plotë të ndryshëm
10                                  // Futni numrin e dytë të plotë
11      cout << '                         : ";
12      cin >>  numri2;             //Leximi i vlerës së ndryshores
13      shuma = numri1+numri2;      /* Urdhri për mbledhje të vlerave të dy
14                                     ndryshoreve */
15      cout <<          " << shuma << endl;
```

Figura 1.8.1

```
16
17          cout << endl;
18          system("Color 17");
19          system("pause");
20          return 0;
21
22      } // Fundi i funksionit kryesor main()
```

Figura 1.8.1(vazhdim)

Shembulli 1.8.2

Të deklarohen, futen dhe shtypen të dhënat prej llojt numra të plotë.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // Leximi dhe shtypja e të dhënave numra të plotë futni nëpërmjet tastaturës
6       unsigned short dataMuaj   ;
7       unsigned short dataDita ;
8       unsigned int    dataViti  ;
9       long long               ;
10      cout <<   "Futni ditë"    ;    cin >> dataDita;
11      cout <<   "Futni muaj"     ;  cin >> dataMuaj;
12      cout <<   "Futni vit:"       ; cin >> dataViti;
13      cout << endl;
14      cout << "Sot është" << dataDita << "." << dataMuaj  << "."
15          << dataViti     << endl;
16      cout << "\n Futni 19 shifra"      ; cin >> numriMe19shifra;
17      cout << "\n E futët numrin;"         <<numriMe19shifra << endl;
18
19      cout << endl;
20      system( "Color 17" );
21      system( "pause" );
22      return 0;
23  }
```

Figura 1.8.2

Dalja prej programit te ***figura 1.8.2*** është:

```
Futni ditë       01
Futni muaj         01
Futni vit            2017
Sot është 1.1.2017

Futni 19 shifra:      1234567890123456789
E futt numrin:          1234567890123456789
Press any key to continue . . .
```

Shembulli 1.8.3

Të deklarohen, futen dhe shtypen të dhënat prej llojit numri i plotë.

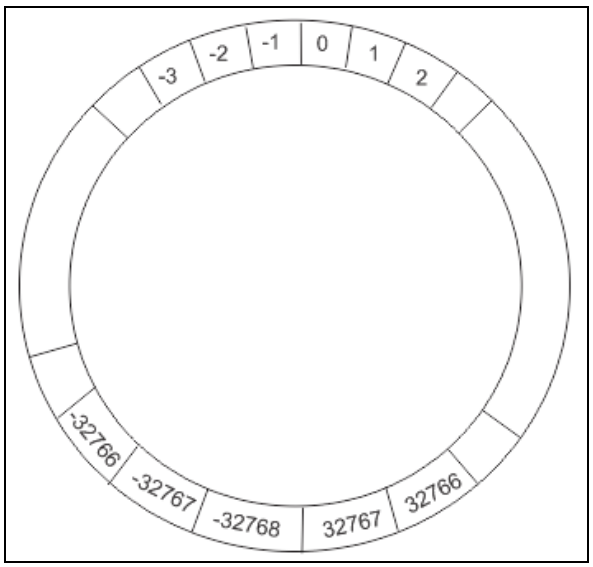
Te program vijues (***figura 1.8.3***) lexohen dy numra të plotë dhe shtypet mesi i tyre aritmetik.

```
#include <iostream>
using namespace std;

int main() {
    // Leximi dhe shtypja e të dhënave.reale fut nëpërmjet tastaturës
    float numrilParë;
    double numrilDytë numriMesi
    cout << " Fut numrin e parë dhjetor;"       ;      cin >>numrilParë,
    cout << "`Fut numrin e dytë dhjetor"         ;     cin >> numrilDytë;
    numriMesi    = (numrilParë + numrilDytë)/ 2;
    cout << "\n Vlera mesatare prej   " <<numrilParë< " i " << dhe numrilDytë;
        << "është "<< brojsredina << endl;

    cout << endl;
    system( "Color 17" );
    system( "pause" );
    return 0;
}
```

Figura 1.8.3

Shembulli 1.8.4

Te ***figura 1.8.4*** është dhënë shembull për kapërcimin e vargut të numrave, të dhënë sipas llojit të ndryshoreve.



Detyra ilustron shfrytëzimin jo korrekt të ndryshores gjithsej pasi fiton vlerë e cila është më e madhe prej vargut të vlerave sipas përkufizimit të llojit. Ndryshorja gjithsej është i llojit short, vlera maksimale të cilës është 32 767, por te njehsimi fiton vlerë 60.000, e cila kalon në vargun negative të numrave të plotë prej llojit short (–32768 në –1).

Kjo do të jetë më e qartë nëse vargu i llojit të numrave të plotë të dhënat e paraqesin të mbyllur (rrumbullakt). Prej atij vërehet e numri 32 768 (i cili është jashtë vargut) do të paraqitet me

numrin –32 768, por numri 32 769, do të paraqitet me numrin –32 767. Pasi, për vlerë të ndryshores gjithsej do të fitohet:

gjithsej = –32 768 + (60 000 – 32 767) – 1 = –5 536

Kjo dukuri e daljes së vlerës jashtë prej vargut të llojit prej të cilit është ndryshorja e deklaruar quhet **mbushje** (angl. overflow).

```
#include <iostream>
using namespace std;

int main() {
    short çmimiPërNjësi, gjithsej;  /* Deklarata e ndryshoreve      */
    short     copa = 1000;          /* Deklarata me inicializim      */
    çmimiPërNjësi= 60;              /*  Shqërim  */
    gjithsej = çmimiPërNjësi
    cout << " gjithsej çmimiPër "       <<copa;
    cout << "  copa me    " << çmimiPërNjësi << "denarë është "
        << gjithsej << " denarë." << endl;

    cout << endl;
    system( "Color 17" );
    system( "pause" );
    return 0;
}
```

Figura 1.8.4

```
Çmimi i përgjithshëm për 1000 copa nga 60 denarë është -5536 denarë
Press any key to continue . . .
```

Leximi dhe shtypja e të dhënave me shenja dhe stringe

Te shqyrtimet paraprake thamë se stringet janë vargje, me të cilët mund të realizohen operacione, sikurse: bashkim, krahasim dhe të tjera.

Bashkimi realizohet me shenjën +, por operandët mund të jenë:

- Literalër e shenja ose string:

```
' ', '@', "programimi në"
```

- Konstanta të emërtuara me shenja ose string të deklaruara me fjalën const:

```
const char e zbrazët  = ' ';
const string plusplus = "++";
```

- Ndryshore të deklaruara me shenja ose string:

```
char CShkronja = 'C';
string Titulli = "Bazat e";
```

Me urdhrin:

```
titull  = titull  + " " + "programim në" + ' ' + "gjuha"
+ e zbrazët  + CShkronja  + plusplus;
```

ndryshorja titulli do të fitn vlerë:

```
"Bazat e rgramimit në gjuhën C++."
```

Gjatë bashkimit të string-konstanteve, të paktën njëri prej operandëve të shenja + patjetër të jetë ndryshore ose konstante e emërtuar. Në të kundërtën, do të paraqitet gabimi.

```
string ab = "aaa" + "bbbb";
```

expression must have integral or unscoped enum type

Është e drejtë:

```
string a = "aaa";
string ab = a + "bbbb";
//  ose
string ab = string("aaa") + "bbbb";
```

Të përmendim se të dhënat numerike (të llojit int, float, double) nuk mund të bashkohen me stringe.

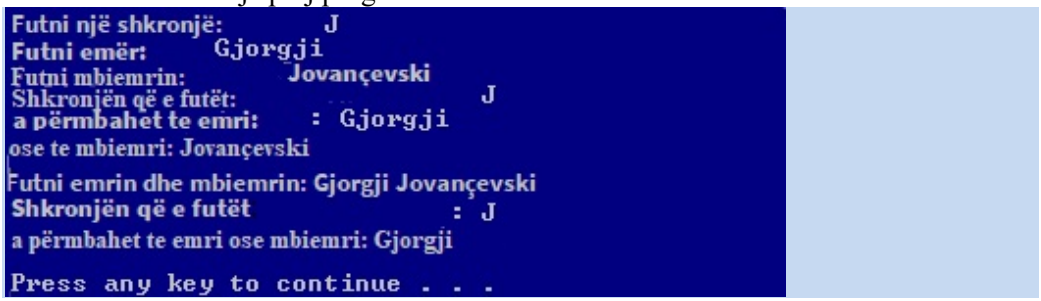
Shembulli 1.8.5

Te ***figura 1.8.5*** janë ilustruar leximi dhe shtypja e ndryshoreve me shenja dhe stringe.

```
1   #include <iostream>
2   #include <string>
3   using namespace std;
4
5   int main() {
6       char  njëshkronjë;
7       string emriIm, mbiemriIm, emriDHEMbiemriIm  ;
8       cout << "Futni një shkronjë: "  ; cin >>njëshkronjë;
9       cout << "Futni emër emriIm:"  cin >> emriIm;
10      cout <<"Futni mbiemër mbiemriIm:"  cin >> mbiemriIm;
11      cout <<"Vallë shkronjë që e futët njëShkronjë :"  <<njëshkronjë; << endl;
12      cout << "përfshihet te emri emriIm:" << emriIm << endl;
13      cout << "ose te mbiemri mbiemriIm:" <<  mbiemriIm; << endl;
14      cout << endl;
15      cout << "Futni emrin dhe mbiemrin>";
16      char shenjaPërFund;
17      cin.get(shenjaPërFund);
18      getline( cin, emriMbiemriIm  );
19      cout << "Vallë shkronja që e futët njëShkronjë:"<<njëshkronjë;<< endl;
20      cout << "përfshihet te emri ose mbiemri:"
21          << emriMbiemriIm  << endl;
22
23      cout << endl;
24      system( "Color 17" );
25      system( "pause" );
26      return 0;
27  }
```

Figura 1.8.5

Dalja prej programit është:



Vërejtje: Edhe pse futja e të dhënë për ndryshore string emriImMbiemriIm futet stringu „Gjorgji Jovançevski“, gjatë shtypjes së ndryshores së njëjtë, shtypet vetëm „Gjorgji“. Këtë do ta sqarojmë në vazhdim.

Shtypja e formatuar

Gjatë shtypjes të të dhënave, ato ndahen me një ose më shumë vende të zbrazëta, me shfrytëzimin e sekuencës (eskejp) dalëse '\t' për shtypje të pozitën vijuese tabulatore ose me sekuencën dalëse '\n' për shtypje në rreshtin e ri. Gjithashtu, për shtypje në rresht të ri shfrytëzohet edhe manipulatori endl.

Për formatimin e të dhënave gjatë shtypjes, shfrytëzohen edhe manipulator të tjerë. Disa prej tyre janë përkufizuar në bibliotekë <iostream>, por disa janë përkufizuar në bibliotekë <iomanip>, e cila (sikurse edhe <iostream>) duhet të kyçet në fillim prej programit:

```
#include <iostream>
#include <iomanip>.
```

Te ***tabela 1.8.1*** janë dhënë disa më së shpeshti manipulator të shfrytëzuar veprimi i të cilëve është në urdhrin për shtypje cout.

Manipulator	Veprimi	Përshkrimi
endl	momental	Vija e re (njëjtë sikurse '\n ').
setw(n)	të të dhënës vijuese	Vendosja e gjeësisë minimale në fushë për shtypje të të dhënës vijuese në urdhrin për dalje. (Nëse numri është me më shumë shifra, shtypet në shumë shtylla). Nënkuptohet (sipas difolt) (angl. by default) se e dhëna është mbështetur djathtas.
width(n)	të të dhënës vijuese	Njëjtë me setw(n).
left	të të dhënës vijuese	Mbështetje majtas te fusha e vendosur me setw(n).

right	të të dhënës vijuese	Mbështetje djathtas (sipas difolt) te fusha e vendosur me setw(n).
dec	të të dhënës vijuese	Numri shtypet në format dhjetor (baza 10).
oct	të të dhënës vijuese	Numri shtypet në format oktal (baza 8).
hex	të të dhënës vijuese	Numri shtypet në heksadhjetor (baza 16).
setprecision(n)	të gjitha të dhënat vijuese	Vendosja e numrit të shifrave të cilat shtypen sipas pikës dhjetore.
fixed	të gjitha të dhënat vijuese	Shtypja e numrave dhjetorë në f-format. Difolt preciziteti është 6 shifra.
scientific	të gjitha të dhënat vijuese	Shtypja e numrave dhjetorë në e-format.
uppercase	të gjitha të dhënat vijuese	Për shtypjen e numrave dhjetorë në e-format, por me E të madhe. Për shembull, 1.234E+05. Efekti eliminohet me manipulatorin noupperacase.
boolalpha	të gjitha të dhënat vijuese	Shtypen konstantat true ose false.
noboolalpha		Shtypen konstantat 1 se 0.

Tabela 1.8.1

Specifikat gjatë të lexuarit të të dhënave

Thamë se dalja dhe hyrja e të dhënave te programet kryhet nëpërmjet të dhënave prej shenjave. Poashtu, në fillim të programit duhet të kyçet biblioteka <**iostream**>, e cila përmban përkufizimin e dy llojeve të ndryshoreve: **ostream** dhe **istream**.

Lloj ostream është lidhur me **përruan dalëse** (angl. output stream), ndërsa lloj istream është lidhur me **përruan hyrëse** (angl. input stream) të të dhënave.

Te biblioteka <iostream> janë deklaruar dy ndryshore prej këtyre llojeve:

```
istream cin;
ostream cout;
```

Ndryshorja cout së bashku me **operatorin për futje** (insertim) (angl. insertion operator) <<, është lidhur me pajisjen dalëse standard të kompjuterit, më së shpeshti monitori. Poashtu, me një urdhër për shtypje, vlerat e ndryshoreve dhe konstantave të cilat duhet të shtypen futen te përruai daljes.

Për shembull, me këtë segment programor:

```
int m;
double b;
m = 12;
```

```
b = 34.5;
cout << m << b << endl;
```

Në përruen dalëse vendose n vlerat e ndryshoreve m dhe b dhe të monitorit do të shtypen:

```
1234.5
Press any key to continue . . .
```

Njashëm, ndryshorja cin së bashku me **operatorin për ndarje** (ekstrakcim) (angl. extraction operator) >>, është lidhur me pajisjen hyrëse standard të kompjuterit, më së shpeshti tastatura. Poashtu, prej përruas hyrëse ndahen të dhënat të cilat shoqërojnë ndryshoreve te programi.

Për shembull, nse te segmenti programor paraprak shtojmë urdhër për lexim të m dhe b:

```
cin >> m >> b;
cout << m << b;
```

dhe nëpërmjet tastaturës usi të dhënë hyrëse:

```
6 7.8
```

me shtypjen e butonit Enter, të ndryshoreve m dhe b d t'u shoqëroj verat 6 dhe 7.8. Përrua dalëse do të jetë 67.8:

```
1234.5
6 7.8
67.8
Press any key to continue . . .
```

Vërejtje: Shpesh, ngatërrohemi në cilën anë duhet të jetë operatori, << ose >>. Duhet të mbahet mend se operatori duhet të rientohet kah *dalja* << (për shtypje) ose prej *hyrjes* >> (për lexim).

Vëreni se te përrua hyrëse të dhënat 6 dhe 7.8 janë ndarë e vend të zbrazët, që është e saktë. Nëse nk kishte vend të zbrazët, atëherë te m do t'i shoqërohet vlera 67, por b vlera 0.8.

I njëjti përrua hyrëse:

```
6 7.8
```

saktë do të lexohet edhe me dy urdhëra:

```
cin >> m;
cin >> b;
```

Me shtypjen e butonit Enter në fund të përruas hyrëse, gjenerohet shenja për fund të vijës aktuale dhe kalon në vijën e re. Prandaj, ai quhet **shenja për vijën e re** (angl. newline character). Gjatë leximit të përruas hyrëse, lexohet ndërsa nuk gjendt shenja për vijëne re.

Ngjashëm, përrua dalëse shtypet deri sa nuk vjen deri te shenja për fund, i cili (vërejtëm te shembujt deritanishëm) parashtrohet me manipulatorin endl ose me eskejp sekuencën '\n'. Kur do të vjen deri te ai, treguesi i ekranit kalon në vijë tëre.

Për të lexuar dhe vend ii zbrazët si shenjë, shfrytëzohet urdhri i veçantë:

```
cin.get(ndryshorjaMeShenjë);
```

Ky udhr është lidhur me llojin istream, te i cili theksuam se është deklaruar ndryshorja cin. Pasi cin është ndryshore e veçantë, me atë janë lidhur më shumë funksione, të cilat janë përkufizuar në bibliotekën <iostream>. Njëri prej atyre funksioneve është edhe funksioni get(), e cila e lidh me ndryshoren cin me të ashtuquajturë **operator pika** (angl. dot operator). Prandaj, urdhri për leximin përbëhet prej ndryshores cin, pika dhe funksioni get().

Për shembull, për të lexuar përruan hyrëse prej shembullit paraprak:

```
6 7.8
```

dhe vendit të zbrazët ndërmjet tyre, duhet ta shkruajmë segmentin programor vijues:

```
cin >> m;
cin.get(z);
cin >> b;
cout << m << ' ' << z << ' ' << b << endl;
```

Poashtu, ndryshorja z do të fitojë vlerë vend të zbrazët dhe dalja do të jetë:

```
6 7.8
6   7.8

Press any key to continue . . .
```

Problem më i madh me vende të zbrazëta paraqitet gjatë leximit të stringeve të cilat përmbajnë edhe vende të zbrazëta. Kështu, te shembulli prej ***figura 1.8.5***, te pjesa e

```
cout << "Futni emrin dhe mbiemrin: "; cin >> emriMbiemriIm;
cout << "shkronja që e futët a është: " << njëShkronjë  << endl;
cout << "përmbahet te emri ose mbiemri: " << emriMbiemriIm  << endl;
```

futet gjithë emri dhe mbiemti, por shtypet vetëm emri:

```
Futni emrin dhe mbierin: Gjorgji Joançevski
Vall shkronjën që e futët: J
përmbahet te emri ose mbiemri:Gjorgji
Press any key to continue . . .
```

Kjo ndodh pasi me operatorin >> lexohet deri te vendi i parë i zbrazët dhe, edhe pse është utur emri dhe mbiemri, lexohet vetëm deri te vendi i zbrazët, ndrmjet emrit dhe mbiemrit. Prandaj, ndryshorja emriMbiemriIm fiton vlerën „Gjorgji“, e cila edhe shtypet.

Për leximin e drejtë të stringeve, shfrytëzohet funksioni getëine(), me të cilin lexohet gjithë vija hyrëse deri te shenja për vijën e re (të vendosu me Enter). Poashtu, marker për lexim vendoset në fillim të vijës vijuese për lexim të të dhënave.

Sintaksa e urdhrit është:

```
getline(cin, stringNdryshorja);
```

Për shembull, te program prej ***figura 1.8.5***, urdhri:

```
cin >> emriMbiemriIm;
```

do ta zëvendësojmë me urdhrin:

```
char shenjëPërFund;
cin.get(shenjëPërFund);     // lexohet shenja  '\n'
getline(cin, emriMbiemriIm);
```

edhe dalja do të jetë e drejtë:

```
Futni një shkronjë:          J
Futni emër:        Gjorgji
Futni mbiemrin:           Jovançevski
Shkronjën që e futët:                    J
a përmbahet te emri:      : Gjorgji
ose te mbiemri: Jovançevski

Futni emrin dhe mbiemrin: Gjorgji Jovançevski
Shkronjën që e futët                   : J
a përmbahet te emri ose mbiemri: Gjorgji

Press any key to continue . . .
```

Të sqarojmë. Me urdhrin:

```
getline(cin, emriMbiemriIm);
```

lexohet stringu i future dhe shoqërohet stringu të ndryshores emriMbiemriIm. Megjithatë, te urdhri paraprak për lexim shfrytëzohet operatori >>, me të cilin lexohet deri te shenja për fund, por jo edhe shenja për fund. Prandaj, me urdhrin:

```
cin.get(shenjëPërFund);
```

lexohet shenja për fund dhe marker për lexim pozicionohet në fillim të vijës së re për lexim.

Vërejtje:: Nëse për lexim të të dhënave te gjithë program shfrytëzohet funksioni getëine(), atëheë nuk është e nevojshme në veçanti leximi i shenjës për fund.

Shembulli 1.8.6

Me këtë shembull ilustroht shfrytëzimi i funksioneve get() dhe getëine(). Programi njehson note mesatare të dy nxënësve.

```
1   #include <iostream>
2   #include <string>
3   #include <iomanip>
4   using namespace std;
5
6   int main() {    //Ilustrimi i funksioneve get() dhe getëine():() i getline()
7
8       char enter;
9       string emriDheMbiemri_1
10      int o1_1, o2_1, o3_1, o4_1, o5_1, o6_1, o7_1, o8_1;
11      cout << "Futni emrin dhe mbiemrin e nxënësit të parë;
12      getline( cin, emriDheMbiemri_1);
13      cout << "  Futni 8 nta të ndara me vend të zbrazët       " << endl;
14      cout << "  Nota ";
15      cin >> o1_1 >> o2_1 >> o3_1 >> o4_1 >> o5_1 >> o6_1 >> o7_1 >> o8_1;
16      double mesatare = float( o1_1 + o2_1 + o3_1 + o4_1 + o5_1 + o6_1 +
17          o7_1 + o8_1 ) / 8;
18      cin.get( enter );
19
20      string imeIPrezime_2;
21      int o1_2, o2_2, o3_2, o4_2, o5_2, o6_2, o7_2, o8_2;
```

Figura 1.8.6

```
22      cout <<"Futni emrin dhe mbiemrin e nxënësit të dytë ";
23      getline( cin, Emri dhe miemri);
24      cout << "  Futni 8 nota të ndara me vend të zbrazët     " << endl;
25      cout << " Nota: ";
26      cin >> o1_2 >> o2_2 >> o3_2 >> o4_2 >> o5_2 >> o6_2 >> o7_2 >> o8_2;
27      double Mesatare_2 float( o1_2 + o2_2 + o3_2 + o4_2 + o5_2 + o6_2 +
28          o7_2 + o8_2 ) / 8;
29
30      system( "cls" );
31      cout << left << setw( 20 ) << " Emri dhe miemri << right << setw( 25 )
32          << "  M  M  I  A  F  H  B  F" << "\tMesatare" << endl;
33      cout << setw( 45 ) << right << "  a  a  n  n  i  e  i  i" << endl;
34      cout << setw( 45 ) << right << "  k  t  f  g  z  m  o  s" << endl;
35      cout << "-----------------------------------------------" << endl;
36      cout << left << setw( 21 ) << emriDheMbiemri_1 << "  " << o1_1 << "  "
37          << o2_1 << "  " << o3_1 << "  " << o4_1 << "  " << o5_1 << "  "
38          << o6_1 << "  " << o7_1
39          << "  " << o8_1 << "\t" << setprecision( 2 ) << mesatar_1  << endl;
40      cout << left << setw( 21 ) << emriDheMbiemri_2 << "  " << o1_2 << "  " << o2_2
41          << "  " << o3_2 << "  " << o4_2 << "  " << o5_2 << "  " << o6_2 << "  "
42          << o7_2 << "  " << o8_2 << "\t" << setprecision( 2 ) << mesatar_2  << endl;
43
44      cout << endl;
45      system( "Color 17" );
46      system( "pause" );
47      return 0;
48
49  }
```

Figura 1.8.6 (vazhdim)

Dalja prej programit është

```
Emri dhe mbiemri      M  M  I  A  F  H  B  F  Mesatare
                      a  a  n  n  i  e  i  i
                      k  t  f  g  z  m  o  s
-----------------------------------------------------
Goce Delçev           5  5  4  5  4  4  5  5    4.6
Jane Sandanski        5  4  5  5  4  4  5  4    4.5

Press any key to continue . . .
```

Detyra për ushtrime

1. Të shkruhet program për shtypje të kësaj tabele:

Titulli	Regjisor	Zhanr	Viti
Bal-kan-kan	Darko Mitrevski	drama	2005
Pred dozhdot	Milço Mançevski	drama	1994
Najdolgiot pat	Branko Gapo	historic	1976
Crno seme	Kiril Cenevski	drama	1971
Volçja noç	Franc Stiglic	luftarak	1955

Të deklarohen 5 stringet ndryshore për titujt e filmave, 5 për regjisorët, 5 për string ndryshoret për titujt e filmave, 5 për regjisorët dhe 3 për zhynrin, si edhe 5 ndryshore me numra të plotë për vitet. Të dhënat të futen nëpërmjet tastaturës në formë:

Pyetje dhe kontrolli i njohurive

1. Si është organizuar leximi dhe shtypja e të dhënave në C++?
2. Cila bibliotekë është e detyrueshme të kyçet në çdo program që të mund të lexohet dhe të shtypen të dhënat?
3. Cilët janë operatorët për lexim dhe për htypje të të dhënave?
4. Cilat janë urdhërat për hyrje (futje e të dhënave nëpërmjet tastaturës) dhe për dalje (shtypja e të dhënave në ekran)?
5. Çka është mbushje (angl. overflow)?
6. Cila bibliotekë prej C++ i përmban funksionet për formatim të shtypjes?
7. Me cilin manipulator vendoset gjerësia e fushës në të clën shtypet ndonjë e dhënë?
8. Me cilin manipulator vendoset mbështetja e të dhënës (majta ose djathtas) te fusha për shtypje?
9. Me cilin manipulator vendoset preciziteti i të dhënës (numri i vendeve dhjetore) e cila do të shtypet?
10. Me cilin manipulator shtypen konstantat logjike true dhe false?
11. Cili është operatori për futje (insertim) gjatë shtypjes, por cili është operatori për ndarje (ekstrakcion) gjatë leximit?
12. Shkruani urdhrin për lexim në shenjë.
13. Shkruani urdhrin për lexim të gjithë vijës.
14. Me cilin urdhër lexohet vetëm në fjalë, por me cilën gjithë fjalia?

Terminet

- Të gjithë programet të cilët mund t'i realizojë kompjuteri, me një emër quhet **softuer.**
- Softueri ndahet në **programe sistemore** dhe **programe aplikative**.
- **Programet sistemore** janë ato të cilët udhëheqin me punën e kompjuterit.
- **Programet aplikative** ose **programe shfrytëzuese** janë ato të cilët realizojnë punë të caktuara për shfryëzuesit.
- **Gjuhë programore** është gjuhë artificial e cila shërben për komunikim ndërmjet njerit dhe kompjuterit ose për komunikim ndërmjet dy kompjuterëve.
- **Algoritmi** është mënyrë prej numrit të fundshëm rigorozisht të përkufizuar aktivitete dhe saktë renditje të dhënë të realizimit të tyre.
- **Algoritëm** paraqet mënyrë që përbëhet prej bashkësisë së fundshme të aktiiteteve saktë të përkufizuara, të zbatuara mbi të dhënat hyrëse sipas renditjes rigoroze hyrëse, me të clat arrihet deri te rezultatet dalëse.
- **Hapi algoritmik** është një veprim prej algoritmeve.
- **Algoritmi i përgjithshëm** përbëhet prej hapave më të përgjithshme.
- **Algoritmi detal** përbëhet prej hapave më detale të cilët mund të shkruhen me një urdhër.
- **Pseudogjuha** është gjuhë për shkruarje tekstuale të algoritmeve.
- Paraqitja grafike e algoritmeve kryhet me të ashtuquajturën **bllk-diagrame**.
- **Kodimi** është „të shprehurit", d.m.th., të shkruarit e algoritmeve me urdhërat prej ndonjë gjuhe programore.
- **Programi burimor** paraqet algoritëm të koduar me urdhëra prej ndonjë gjuhe programore.
- **Programi realizues** është program i fituar prej programit burimor me përkthyes, ku mund procesori ta realizojë.
- **Përkthyes** është program sistemor që e përkthen programin burimor në program realizues.
- **Interpretator** është program sistemor që e interpreton de realizon programin burimor.
- **Struktura kontrolluese algoritmike** janë struktura speciale me të cilat kontrollohet realizimi i algoritmeve.
- **Programimi i strukturuar** është programim te i cili shfrytëzohen teknikat: programimi prej lart pshtë dhe programimi modular.
- Me teknikën **programimi prej lart poshtë** ndahet (zbërthehet) detyra në detyra më të vogla dhe më të thjeshta të ashtuquajtura **nëndetyra**.
- Teknika **programimi modular** përbëhet në programimin e tërësive të pavarua pre algoritmit si module të veçanta, të cilët pastaj bashkohen sekuencijalisht në një tërësi.

- **Gjuha makinerike** është gjuhë te e cila imstuksionet makinerike dhe të dhënat shprehen me zero dhe njëshe.
- **Gjuhë simbolike** janë gjuhë te të cilat instruksionet makinerike dhe të dhënat shprehen me simbole (mnemonic).
- **Asemblere** janë përkthyes të cilët programin prej gjuhës simbolike e përkthejnë në program të gjuhës makinerike.
- **Gjuha programore e lartë** janë gjuh të gjuhëve natyrore, te t clat sot (më së shpeshti) programohet.
- Gjuhët programore të larta janë **gjuhë makinerike të pavarura** pasi nuk vren prej makinës te e cila realizohen programet (me përkthim të mëparshëm).
- Gjuhët programore të larta janë **gjuhë problemore të orientuara** pasi impnohen për problem prej fushave të veçanta (ekonomi, teknikë, inteligjenca artificijale etj.).
- **Gramatika** e gjuhës programore është bashkësi e rregullave për ndërtimin e fjalëve dhe urdhërave.
- **Sintaksa** është bashkësi e rregullave për ndërtimin e urdhërave te gjuhët programore.
- **Semantika** është domethënia e ndonjë udhëri, d.m.th., veprime që ajo i shkakton gjatë realizimit.
- **Fjalët e rezervuara** janë fjalë të veçanta të rezervuara për gjuhën programore.
- Disa prej fjalëve të rezervuara janë **fjalët kyçe** pasi kanë domethënie të veçantë për gjuhën programore.
- **Identifikator** janë fjalë të cilët i formojnë programuesit sipas rregullave të caktuara dhe patjetër të jenë të ndryshme prej fjalëve t rezervuara te gjuha programore.
- **Përkthyes (kompajler)** është program special për përkthim të programit burimit në programin makinerik.
- **Interpretator** janë programe speciale të cilët direkt e realizojnë (interpretojnë) programin burimor.
- **Gjuhë skripte** janë gjuhë programore për integracion dhe komunikim me gjuhë tjera programore.
- **Gjuhë programore imperative** janë ato te të cilat për përpunimin e të dhënave shfrytëzohen urdhëra.
- **Gjuhë programore deklarative** janë ato te të clat përshkruhet se si të vjen deri te rezultati.
- **Gjuhë programore procedurale** janë gjuhë imperative te të cilat shfrytëzohen **procedura** (funksione) (grup prej urdhërave të shprehura si një tërësi).
- **Gjuhë objekte të orientuar** janë gjuhë imperative te të cilat të dhënat janë shprehr si atribut të objekteve, por sjelloja e objekteve shprehet me funksione.

- **Gjuhë funksionale** janë gjuhë deklarative te t cilt shfrytëzohen shprehje dhe funksione.
- **Gjuhë logjike** janë gjuhë deklarative te të cilat gjatë njehsimeve, shfrytëzohen relacione, fakte, përfundime dhe metoda për përfundim.
- **Unified Modeling Languge – UML** është gjuhë e veçanr grafike për të shprehur procese të ndryshme (vijimi i algoritmit, kalimi prej njërës gjendje në tjetrën dhe të ngjasme).
- **Rrethina e integruar zhvillimore** është rrethina me aplikime të cilat shfrytëzohen gjatë programimit.
- **Microsoft Visual Studio** është rrethinë e integruar zhvillimore për të shkruar, redaktur dhe realizimi (testimi) i programeve.
- **Code::Blocks** është rrethinë e integruar zhvillimore për të shkruar, redaktur dhe realizimi (testimi) i programeve.
- **Redaktor tekstual – editor** është aplikacioni për të shkruar dhe redaktuar programe.
- **Dibager** është aplikacion për kërkesë të gabimeve logjike te programi.
- **Lidhës** është aplikim për lidhje të shumë datotekave (programeve) në datotekë të vetme.
- **Apikacioni** në C++ është program që e përmban funksionin kryesor main().
- **<iostream>** është bibliotekë që përmban funksione për operacione hyrëse dhe dalëse te programet.
- **namespace** është direktiva për formimin e hapësirës së emrit (varg) te memoria për emra të ndryshoreve, funksioneve dhe identifikatorëve të tjerë.
- **Funksioni** është program që pas realizimit jep rezultate.
- **Dhëmbëzimi** paraqet dhëmbëzim të urdhërave (më së shpeshti për një pozitë tabultorësh) për shkak të pamjes më të madhe rë programeve.
- **Funksioni kryesor** është ai me të cilin fillon realizimi i çdo aplikacioni.
- **Komentim** është çfarëdo qoftë tekst i venduar me dy viza të pjerrëta ose ndërmjet shenjave /* në fillim dhe */ në fund.
- **cout** është urdhër për shtypje.
- **cin** është urdhër për lexim.
- **<<** është operator për shtypje.
- **>>** është operator për lexim.
- **endl** është urdhër për kalimin në ndonjë vijë të re.
- **;** është shenjë për në fund të urdhrit, d.m.th., ndarës i urdhërave.
- **return 0;** është urdhër për mbarimin korrekt të aplikimit.
- **Struktura rendore** ose **sekuenca** është struktura te e cila urdhërat realizohen nërenditje sikurse që janë renditur.

- **Sekuenca (eskejp) dalëse** janë sekuenca speciale të shprehura me kombinim të vizës së pjerrtë të kundërt dhe ndonjë shkronje ose shenje. Për shembull, \n është eskejp sekuencë për kalimin në rresht të ri.
- **String** është varg prej shenjave të vendosura në thonjëza, të cilat trajtojnë sikurse lloj i veçantë i të dhënës.
- **Gjykim** (më së shpeshti) paraqet shprehje matematikore që ka vlerë. Shpesh, urdhërat te gjuhët programore quhen gjykime.
- **Madhësia** është masa për ndonjë karakteristikë të objektit ose dukurisë së dhënë.
- **Të dhëna** janë vlera me të cilat shprehen madhësitë.
- **Literale** janë vlera fikse të cilat nuk ndryshojnë.
- **Konstantat e emërtuara** ose vetëm **konstante** janë identifikator (emra të madhësive) te të cilat mund vetëm njëherë t'u jepet vlerë dhe ajo nuk guxon të ndryshojë gjatë realizimit të programit.
- **Ndryshore** janë identifikator (emra të madhësive) te të cilat mund t'u jepen vlera të ndryshme të programit.
- **Lloj** e dhënë paraqet bashkësi të fundshme të të dhënave mbi të cilën është përkufizuar bashkësi e fundshme e operacioneve.
- **Llojet e thjeshta** ose **të pastrukturuara të të dhënave** janë ato të cilat nuk mund të ndahen në pjesë.
- **Llojet e përbëra** ose **të strukturuara të të dhënave** përbëhen prej më shumë llojeve të thjeshta.
- **Llojet e adresave tëtë dhënave** janë ato vlera e të cilave është adresa e ndonjë lokacioni të memories.
- **Llojet e thjeshta të të dhënave** ose **lloje të ndërtuara të të dhënave** të quajtura dhe **llojet themelore t të dhënave** janë lloje numra të plotë, real, me shenja dhe logjike.
- **Specifikatorët e llojit** janë emra të llojeve.
- Të dhëna **unsigned** janë ato të cilat mund të kenë vetëm vlera jonegative.
- **String** është varg prej shenjave të vendosura në thonjëza.
- **Lidhja** është operacion për lidhje të dy stringeve njëri me tjetrin.
- **Deklarimi i ndryshores** është dhënia e llojit dhe të emrit të ndryshores.
- **Deklarimi dhe inicializimi i ndryshores** është dhënia e llojit dhe të emrit dhe shoqërimi i vlerës fillestare të ndryshores.
- **Përkufizimi i ndryshores** është shoqërimi i vlerës gjatë deklarimit dhe inicializimit se shoqërimi i vlerës me urdhër për shoqërim të tani më ndryshores së deklaruar.
- **Lloj numër i plotë i të dhënave** është e dhënë me vlerë prej bashkësisë së numrave të plotë Z = {... –2, –1, 0, 1, 2... }, por varësisht prej bashkësisë, llojeve të të dhënave numra të plotë deklarohen e fjalët: **short**, **int**, **long, longlong, unsigned short, unsigned int, unsigned long dhe unsigned long long**.

- **Forma e shkurtuar e operatorëve** (ose **operatorët** e **përbërë**) shprehet me operator aritmetik dhe operatori për shoqërim =. Për shembull, +=, –=, *= etj.
- **Lloj real i të dhënës** është ajo e dhënë që fitn vlera prej bashkësisë së numrave real (mendohet në numra dhjetor), por deklarohet me fjalët: **float, double** dhe **long double**.
- Paraqitja e numrit dhjetor me **pikë të palëvizshme** është kur numri dhjetor shkruhet me pikë dhjetore në vendin e fiksuar, që saktë i ndan pjesën e plotë dhe dhjetore të numrit.
- Paraqitja e numrit dhjetor me **pikë lëvizëse** është kur numri dhjetor shkruhet me pikë dhjetore në çfarëdo vendi, që shfrytëzohet eksponent për vendet e zhvendosura.
- **Konversioni implicit** (automatik) ose **lloj i detyruar** është shoqërimi i vlerës së ndryshores prej llojit jo përkatës.
- **Konversioni i llojit eksplicit** ose **hidhja e llojit** është listimi i llojit te i cili duam të kovertojmë ndryshoren ose shprehjen.
- **Operatori për inkrementim** është ++.
- **Operatori për dekrementim** është – –.
- **Lloj me shenja të të dhënës** mund të ketë vlerë çfarëdo qoftë shenjë ose çfarëdo qoftë vlerë numër i plotë dhe deklarohet me fjalën **char**.
- **lloj logjik** i të **dhënës** mund të ketë vlerë ose true ose false, por deklarohet me fjalën **bool**.
- **Operator logjik** ose **operator i kushtëzuar** janë: &&, || dhe !.
- **Funksionet e Bulit** janë: NOT, AND dhe OR.
- **Shprehje logjike** është shprehje vlera e të cilës mund të jetë true ose false.
- **Operator të relacionit** janë: <, <=, >, >=, ==, !=.
- **Operator me bite** janë: &, |, ^, ~, << dhe >>.
- **Lloj i numërueshëm** i të **dhënës** mund të ketë vlerë prej përpara të bashkësisë së përkufizuar të konstantave (vlerat literale), të cilat patjetër të jenë identifikator, por jo numra. Deklarohet me fjalën **enum**.
- **E dhëna tekstuale** është varg prej shenjave të vendosura në thonjëza.
- **Riemërtimi** i llojit të të dhënës paraqet dhënie në tjetër emër (sinonim) për llojin.
- **Caktimi automatik i llojit** kryhet me fjalën auto, ku lloj i ndryshores e cakton përkthyesin, sipas llojit të shprehjes i cili i shoqërohet ndryshores.
- **Heder-datoteka** janë datoteka prej ***bibliotekës standard të C++*** të cilët janë të kyçur me direktivën #include dhe vendohen në thonjëza këndore <>.
- **Mbushja** ndod kur vlera e ndryshores është jashtë prej vargut të llojit prej të cilit është deklaruar.

- **Operatori për futje** (insertim) << është lidhur me pajisjen dalëse standarde të kompjuterit, më së shpeshti monitorin.
- **Operatori për ndarje** (ekstrakcion) >> është lidhur me pajisje hyrëse standarde të kompjuterit, më së shpeshti monitorin.
- **Shenja për vijën e re** është shenja që gjeneron me shtypjen e butonit Enter.
- **Përrua dalëse** e të dhënave paraqet varg prej shenjave të të dhënave që shtypen.
- **Përrua hyrës** i të dhënave paraqet varg prej shenjave të të dhënave që lexohen.
- **<string>** është biblioteka e cila përmban funksione për punë me stringe.
- **<iomanip>** është biblioteka e cila përmban manipulator për shtypjen e formatimit.
- **<ostream>** është biblioteka e cila përmban urdhëra për përrua dalëse.
- **<istream>** është biblioteka e cila përmban urdhëra për përrua hyrëse

Përmbledhje

- Të gjitha programet të cilat munet të Realizon kompjuteri, me një emër quhet softuer.
- Puna e kompjuterit është e kontrolluar me të ashtuquajturën sistem programor.
- Për t'u realizuar çfarëdo qoftë program me ndihmën e kompjuterit, patjetër të ekziston program aplikativ që kompjuteri mund ta realizojë.
- Që të mund të realizohen çfarëdo aktivitete, duhet të njihet numri i saktë i aktiviteteve (elementare) që duhet të realizohen dhe renditja e tyre e realizimit. Lista e aktiviteteve elementare (hapave) të shkruara sipas renditjes së realizimit paraqet algoritëm.
- Gjatë realizimit të algoritmit, futen të dhëna hyrëse, por fitohen rezultate dalëse. Ndonjëherë, rezultatet dalëse nuk fitohen direkt, por nëpërmjet ndërmjet rezultateve.
- Varësisht prej hapave algoritmike, algoritmi mund të jetë i përgjithshëm ose detal.
- Algoritmet mund të shkruhen tekstualisht me pseudo gjuhë, grafikisht me bllok-diagram ose me urdhërat e ndonjë gjuhe programore. Paraqitja grafike mund të jetë me bllok-diagrame standarde ose me të strukturuara.
- Procesi i të shkruarit të algoritmit me urdhër prej ndonjë gjuhe programore quhet kodim, por teksti i fituar quhet program burimor.
- Programi burimor e shkruar te ndonjë gjuhë programore mund të përkthehet me përkthyes në program realizues, që procesori mund ta realizojë ose njëkohësisht ta përkthen (interpreton) dhe realizon me interpretator.
- Për kontroll më të lehtë gjatë realizimit të algoritmeve, ato shkruhen me struktura kontrolluese algoritmike. Sipas tyre, programimi i atillë e ka fituar emrin programimi i strukturuar.
- Programimi i strukturuar është mënyrë e programimit te i cili shfrytëzohen metodat: programimi prej lart poshtë (zbërthimi i detyrës në detyra më të thjeshta) dhe programimi modular (përpunimi i programeve të veçanta për tërësi të pavarura).
- Të zgjidhurit e detyrës me ndihmën e kompjuterit përbëhet prej disa fazave: vendosja e detyrës, përkufizimi i algoritmit, të shkruarit e programit (programimi) dhe testimi i programit.
- Gjatë vendosjes së detyrës, saktë duhet të përkufizohen dhe të precizohen kushtet nën të cilat ajo zgjidhet.
- Gjatë përkufizimit të algoritmit, paraprakisht duhet të kuptohet detyra, të realizohet analiza e të dhënave hyrëse që duhet të përpunohen, a janë të nevojshme formula dhe/ose metoda dhe çfarë rezultate duhet të fitohen.

Pastaj, shkruhet algoritmi i cili mund të programohet dhe program të realizohet në kompjuter.

- Pas përgatitjes së programit, ai duhet të testohet a punon drejtë (a fitohen rezultate të sakta) për të dhëna të ndryshme hyrëse, për të cilët njihen rezultatet.
- Gjuhët programore janë gjuhë artificial të cilat ndahen në: gjuhë programore të larta (te të cilat shkruhen programet), gjuhë simbolike (te të cilat mund, por jo patjetër të përkthehet programi) dhe gjuha makinerike (te e cila përkthehet program që të mund kompjuteri ta realizojë).
- Në gjuhën makinerike, dhe instruksionet dhe të dhënat shkruhen me bite, d.m.th., me shifra 0 dhe 1.
- Te gjuhët simbolike, dhe për instruksionet dhe për të dhënat shfrytëzohen simbole. Gjuhët simbolike janë të orientuara në mënyrë makinerike pasi çdo familje e procesorëve kanë gjuhë simbolike personale.
- Çdo gjuhë programore e lartë ka: gramatikë, fjalor prej fjalëve të rezervuara (dhe kyçe), rregulla për formimin e identifikatorëve (konstante dhe të ndryshueshme), rregulla sintaksore (sintaksa) për konstruksionin e urdhërave dhe rregulla semantike për kontroll të domethënies së urdhërave.
- Gjatë të shkruarit e programit (programimi), programuesit mund të formojnë fjalë (identifikator) me rregulla të veçanta, me të cilat do t'i emërtojë të dhënat që përpunohen. Identifikatorët nuk guxojë të jenë fjalë të rezervuara.
- Nëse program burimor përkthehet me përkthyes (kompajler), atëherë bëhet datoteka realizuese e programit në formën makinerike. Nëse program burimor interpretohet me interpretator, atëherë nuk bëhet datoteka e realizueshme, por interpretatori i interpreton dhe realizon urdhërat.
- Gjuhët e skripteve shërbejnë për komunikim me gjuhë të tjera programore. Për ato nuk është e nevojshme përkthyes, por interpretator.
- Proraet burimore ka prapashtesë sipas emrit që është shkurtesë prej gjuhës në të cilën janë shkruar. Për shembull:.cpp,.java,.pas,.for etj. Programet e realizuara kanë prapashtesë.exe,.com,.bat,.bin,.dll etj.
- Ka shumë gjenerata të gjuhëve programore.
- Sipas mënyrës së përpunimit të të dhënave, gjuhët programore ndahen në: imperative (procedural dhe objekte të orientuara) dhe deklarative (funksionale dhe logjike).
- Në programimin objekt të orientuar shfrytëzohen objekte të cilat kanë atribute të veta (të dhëna) dhe sjelloja ose gjendja (e shprehur me funksione).
- Për paraqitje grafike të algoritmeve dhe proceset te ndonjë algoritëm, shfrytëzohet gjuha Unified Modeling Languge – UML.
- Sot ekzistojnë rrethina të ndryshme të integruara zhvillimore për programim në C++, por më të njohura janë: Microsoft Visual Studio, Code::Blocks, NetBeans dhe të tjera.

- Çdo rrethinë integruese zhvillimore ka: redactor të tekstit, përkthyes, lidhës dhe dibager.
- Gjuha C++ është gjuha për programim objekt të orientuar, të dizajnuar prej Bjarne Stroustrup (Bjarne Stroustrup) në vitin 1983, si përmirësim të gjuhës „C me klasa“ prej vitit 1980, që, pra, është përmirësim të gjuhës C prej vitit 1972 të dizajnuar prej Dennis Ritchie (Denis Riçi)).
- Programi (aplikimi) i shkruar në C++ patjetër të ketë një funksion kryesor (main()) prej të cilit fillon realizimi edhe të funksioneve të tjera të cilat thirren prej funksionit kryesor.
- C++ është ndërmjet gjuhëve të para programore objekte të orientuara.
- Për realizimin e operacioneve hyrëse-dalëse, në fillim prej programit patjetër të kyçet biblioteka <iostream>.
- Për pamje më të madhe të programit, vendohen vija të zbrazëta dhe komentime, të cilat nuk përkthehen, por kryhet dhe dhëmbëzimi i programit.
- Çdo urdhër në C++ mbaron me pikëpresje (;).
- Programet në C++ mund të shkruhen me çfarëdo redaktues të tekstit.
- Programet në C++ përbëhen prej funkioneve. Funksioni kryesor quhet main().
- Trupi i çdo funksioni vendohet në kllapa të mëdha.
- Çdo aplikim patjetër të ketë vetëm një funksion kryesor, me të cilën fillon realizimi i plikimit.
- Urdhri për shtypjes në C++ është cout, që shfrytëzohet me operatorin për shtypje <<.
- Urdhri për shtypjes në C++ është cin, që shfrytëzohet me operatorin për lexim>>.
- Funksioni kryesor në C++ kthen rezultat 0 nëse është realizuar në mënyrë korrekte.
- Urdhri vijues pas urdhrit për shtypje që në fund e ka manipulatorin end shtyp në vijën e re (rresht i ri).
- Për shtypjen e shenjave special, sikurse”...“, ’... ’, \ etj., shfrytëzohet sekuenca dalëse (eskejp).
- Gjuha C++ ka gramatikë, sintaksë, semantik, fjalë të rezervuara, identifikator (fjalë të formuara prej programuesve) dhe elemente të tjera të një gjuhe programore.
- Gjatë përkthimit të programit, me rregulla sintaksore kontrollohet të shkruarit e drejtë të çdo urdhri.
- Çdo urdhër te program patjetër të ketë saktë domethënie të përkufizimit dhe të shkakton saktë veprime përkufizuese. Kjo kontrollohet me rregulla semantike të gjuhës.
- Me të dhënat shprehet vlerat e madhësive, por madhësitë janë masa për karakteristikat e objektee ose të dukurive. Të dhënat mund të jenë konstante (kanë një vlerë të pandrysheueshme) ose ndryshore (vlera mund të ndryshojë).

- Gjatë deklarimit të konstantave në C++, para llojit të tyre vendohet fjala const.
- Çdo e dhënë ka: emër, lloj dhe vlerë.
- Emrat (identifikatorët) e të dhënave në C++ mund të jenë: fjalë të rezervuara (te të cilat bëjn pjesë edhe fjalët kyçe) ose emra të përkufizuara shfrytëzuese (identifikator), të cilët formojnë sipas rregullave të caktuara. Fjalët e rezervuara nuk mund të shfrytëzohen si identifikator.
- Llojet e të dhënave në C++ mund të jenë të thjeshta (të pastrukturuara), të përbëra (të strukturuara – të cilat përbëhen prej llojeve të thjeshta) dhe me adresë (vlera e të cilës është adresa e lokacionit të memories).
- Llojet e thjeshta janë: integral (numra të plotë, me shenja dhe logjike), reale dhe të numërueshme.
- Lloj të përbërë janë: varg, struktura, unioni dhe klasa.
- Konstantat dhe ndryshoret deklarohen duke përmendur llojin dhe emrin. Nëse te deklarimi jepet vlera e konstantes ose ndryshorja, atëherë kryhet deklarimi dhe inicializimi.
- Për t'u dalluar konstantat prej ndryshoreve, praktikohet ato të shkruhen me të gjitha shkronjat e mëdha, por fjalët te emri i konstantave të ndahen me vizë të gjatë.
- Konstantat deklarohen me fjalën const sipas të cilës përmendet lloj dhe emri, por pastaj, patjetër të shtohet vlera e konstantes me operatorin për shoqërim =.
- Para se të shfrytëzohet te program, ndryshorja patjetër të jetë e përkufizuar me dhënien e vlerës, qoftë me inicializim, qoftë me urdhër për shoqërim.
- Kur ndonjë ndryshoreje i shoqërohet vlerë më e madhe ose më e vogël prej vargut të llojit të tij, vjen deri te mbimbushje ose nënmbushje dhe paraqitet gabimi.
- Lloj numër i plotë dhe lloj me shenja mund të jenë edhe të pa shënuara.
- Specifikuesit e llojeve të taksave:
  - për lloj numër të plotë: short, int, long dhe long long.
  - për lloj me shenja: char.
  - për lloj të numërueshëm: enum.
  - për strukturë: strukt.
  - për klasë: class.
  - për referencë: reference.
  - për lloj real: float, double dhe long double.
  - për llojlogjik: bool.
  - për varg: array.
  - për union: union.
  - për tregues: pointer.
- Lloji i të dhënave integer janë vlera nga grupi i numrave të plotë {... –2, –1, 0, 1, 2...}.
- Format e shkurtuara të operatorëve aritmetikë: +, –, * dhe / për të dhëna të plota dhe reale janë: +=, –=, *=, /=. Operatori %= është një formë e shkurtër e

operatori % për lloj numër të plotë të të dhënave. Këta operatorë quhen operator të përbërë.

- Lloj real i të dhënave janë vlera prej bashkësisë së numrave realë (dhjetor) {... –2.0..., –2.001..., –1.0..., 0.0..., 1.0..., 2.0...}.
- Gjatë deklarimit të ndryshoreve të llojit float, duhet të vendohet prapashtesa (sufiksi) f ose F. Në të kundërtën, ato do të trajtohen si ndryshore të llojit double.
- Të dhënat dhjetore në gjuhët programore paraqiten në format dhjetor (f-format, F-format) (mënyra e njohur si pikë e palëvizshme) edhe në formatin eksponencial (e-format, E- format) (mënyra e njohur si pikë lëvizëse).
- Shkronja f vjen prej fjalës „float“, por shkronja e vjen prej fjalës „exponent“.
- Numrat dhjetor te f- format ose F- format) shkruhen me pikë dhjetore në vendin fiks.
- Kur numrat dhjetor shtypen në e- format (ose E- format), shkruhen v*etëm një shifër* majtas prej pikës dhjetore, por shkronja e ose E shkruhen para eksponentit.
- Shoqërimi i vlerës së ndryshores prej llojit jopërkatës kryhet me konversionin, por mund të jetë implicit (automatik) ose eksplicit (me hedhje). Në rastin e dytë, përmendet lloj te i cili duhet të konvertohet vlera prej llojit jopërkatës, por në rastin e parë, nuk përmendet.
- Për zmadhim/zvogëlim vlera e ndryshores numër i plotë ose me shenja për 1, shfrytëzohet operatori për inkrementim (++) ose operatori për dekerementim (– –).
- Të dhënat prej llojit me shenja mund të ketë vlerë çfarëdo shenje ose çfarëdo vlerë numër i plotë. Ato vendohen në gjysmëthonjëza.
- Lloj logjik i të dhënave mund të ketë vetëm vlerë të konstantave speciale true ose false.
- Gjatë njehsimit të shprehjeve logjike, konstantat logjike true dhe false kanë vlerë 1 dhe 0.
- Për operacionet me lloj logjik të të dhënave, shfrytëzohen operatorët logjikë (të kushtëzuar): &&, || dhe !. Ato janë operator për funksionet e Bulit AND (**DHE**), OR (**OSE**) dhe NOT (**JO**).
- Operatorët e relacionit janë: <, <=, >, >=, == (e barabartë) dhe != (e ndryshme).
- Operatorët me bite janë: & (logjike DHE), | (logjike OSE), ^ (jashtëzakonshme OSE), ~ (komplement), << (zhvendosje majtas për 1 vend) dhe >> (zhvendosje djathtas për 1 vend.
- Lloj i numërueshëm deklarohet me fjalën enum, por konstanta vendohet në listë me kllapa të mëdha dhe shkruhet me shkronja të mëdha. Pas pjesës dalëse prej kllapave të mëdha, nuk vendohet pikëpresje (;).
- Lloj i numërueshëm përkufizohet para funksionit main();

- Stringet ose të dhënat tekstuale janë të dhëna të përbëra prej vargut të shenjave të vendosura në thonjëza.
- Për punën me stringe, shfrytëzohen funksione prej bibliotekës <string>, e cila patjetër të kyçet në programin me direktivë #include.
- Shpesh, për shkak të lexueshmërisë së programit dhe gjendshmëria më e lehtë në atë, për emrat e llojeve të ndërtuara të të dhënave është e dëshirueshme të shfrytëzohen sinonime (tjera emra) të cilat do të na përkujtojnë në diçka, me fjalën kyçe typedef.
- Kur duam llojin e ndryshores të caktohet sipas llojit të shprehjes që i shoqërohet, ai deklarohet me fjalën auto.
- Leximi i të dhënave në C++ është organizuar si përrua hyrëse prej shenjave, ndërsa shtypja është organizuar si përrua dalës prej shenjave.
- Urdhri për lexim prej hyrjes standard është cin, por urdhri për shtypje të daljes standard është cout. cin dhe cout janë ndryshore prej bibliotekës <iostre-am>. cin është prej llojit istream që është lidhur me përruan hyrëse të të dhënave, por cout është prej llojit ostream që është lidhur me përruan dalëse të të dhënave.
- Vijat programore të cilat fillojnë me direktivën `#include` përkthyesi nuk i përkthen, por ato përpunohen me program të veçantë të quajtur paraprocesor.
- Datotekat të cilat janë kyçur te program me direktivën #include vendohen te kllapat këndore, me të cilat informohen paraprocesorët ato t'i kërkon në të ashtuquajturën direktorium include prej *bibliotekës standard të C++*.
- Për formatimin e të dhënave, gjatë shtypjes shfrytëzohen manipulator. Disa prej tyre përkufizohen në bibliotekë <iostream>, por disa përkufizohen në bibliotekë <iomanip>, e cila (sikurse edhe <iostream>) duhet të kyçet në fillim prej programit.
- Me shtypjen e butonit Enter, gjenerohet shenja për vijë të re dhe kalon në vijë të re. Gjatë leximit të të dhënave hyrëse, lexohet deri sa nuk gjendet shenja për vijë të re.
- *Biblioteka standard e C++* përmban më shumë datoteka, të quajtura heder-datoteka të cilat përmbajnë grupe prej funksioneve të ngjashme. Më së shpeshti të shfrytëzuara heder-datoteka janë: <iostream>, <iomanip>, <cmath>, <string>, <cctype> dhe të tjera.

MODULI

KONTROLLI I RRJEDHJES DHE FUNKSIONET NË C++

2

Në këtë kapitull do të njiheni me këtë:

- Strukturat kontrolluese algoritmike.
- Shfrytëzimi i renditjes së strukturës kontrolluese algoritmike.
- Struktura kontrolluese algoritmike për zgjedhje prej dy mundësive **nëse–atëherë–ndryshe** dhe urdhri kontrollues për zgjedhje prej dy mundësive if-else.
- Folezimi i urdhrit kontrollues për tgjedhje prej dy mundësive if-else.
- Struktura kontrolluese algoritmike për zgjedhje prej më shumë mundësive **rast** dhe urdhri kontrollues për zgjedhje prej më shumë mundësive switch.
- Shfrytëzimi i operatorit të kushtëzuar?:.
- Struktura kontrolluese algoritmike për përsëritje **derisa–realizo, realizo–derisa dhe për–deri–hapi** dhe urdhri kontrollues për përsëritje while, do-while dhe for.
- Shfrytëzimi i operatorëve për inkrementim (++) dhe për dekrementim (—)
- Struktura kontrolluese algoritmike për kërcim (**vazhdo, ndërpre, dalje, kërcim**) dhe urdhëra kontrolluese për kërcim (continue, break, exit(), goto).
- Funksione biblioteke në gjuhën programore C++.
- Funksione shfrytëzuese: përkufizimi, deklarimi dhe thirrja.
- Deklarimi dhe shfrytëzimi i ndryshoreve globale dhe lokale.
- Funksione të përmirësuara.

Fjalë kyçe

:: operator për zgjidhjen e fushës së pamjes.	Funksion shfrytëzues
?: operator i kushtëzuar	Funksion shfrytëzues i përkufizuar
break	Lista e argumenteve
continue	Lista e parametrave
do-while	Ndryshorja lokale
exit()	Funksioni shfrytëzues
for	Programimi modular
goto	Parametra
if	Nnalgoritme
if-else	Nëndetyra
inline	Fusha e pamjes
switch	Fusha e qasjes
void	**fillimi–fund**
while	**ndërprerje**
nëse-atëherë	Bartja e argumentit sipas vlerës
nëse-atëherë-ndryshe	Bartja e argumentit sipas referencës
Struktrura kontrolluese algoritmike	**Programimi prej lart poshtë**
Struktrura kontrolluese algoritmike për zgjedhje prej dy mundësive	**vazhdo**
Bllok algoritmik	Procedura
Argument	Nënalgoritmi procedural
Argumenti i bartur sipas vlerës	Struktrura kontrolluese algoritmike rendore (sekuenca)
Argumenti i bartur sipas referencës	**Ndryshorja referente**
Funksione të përmirësuara	Referenca
Argument i vërtetë	Referencimi
Parametri hyrës	Biblioteka standarde
Argument hyrës formal	**kërcim**
Parametri hyrës-dalës	Ndryshorja e fshehur
Argumenti formal hyrës-dalës	**rast**
dyer	Ndryshorja statike

Ndryshoret globale	Argumenti formal
derisa–realizo	Parametri formal i llojit
Jeta e ndryshores	Funksioni
për–deri–hapi	Funksioni pa vlerë kthyese
për–zmadho–deri	Funksioni me vlerë kthyese
për–zvogëlo–deri	Nënprogrami funksional
realizo–deri	Nënalgoritmi funksional
dalje	Nënshkrimi funksional
Parametri dalës	Prototip funksional
Argumenti formal dalës	Përsëritja ciklike
Parametri konstant	Cikli

Te nënpikat **1.1, 1.2** dhe **1.3** prej **MODULIT 1: Hyrje në gjuhën programore C++**, u njohtëm me konceptet algoritëm, hapat algoritmike, mënyra e paraqitjes së algoritmeve dhe programimin e strukturuar. Për përshkrim të rrjedhës së algoritmeve gjatë programimit të strukturuar, shfrytëzohen të ashtuquajtura **struktura kontrolluese algoritmike** të veçanta. Me ato nuk realizohet kurfar njehsime, por caktohet renditja e realizimit të hapave algoritmike d.m.th., drejton me renditjen e hapave të realizuara.

Në teorinë e programimit është e njohur se rrjedha e çdo algoritmi mund të kontrollohet me shfrytëzimin e këtyre strukturave kontrolluese algoritmike:

- **Struktura kontrolluese rendore** ose **sekuenca** (angl. sequence).
- **Struktura kontrolluese për zgjedhje** ose **selektim** (angl. selection).
- **Struktura kontrolluese për përsëritje** ose **iteracion** (angl. iteration).

Struktrura kontrolluese algoritmike e parë është e njohur edhe nën emrin **struktura kontrolluese vijore**, e dyta është e njohur edhe nën emrin **struktura kontrolluese e degëzuar**, por e treta është e njohur edhe nën emrin **struktura kontrolluese ciklike**.

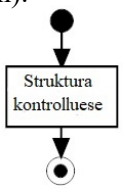


figura 2.1

Çdo strukturë kontrolluese algoritmike ka vetëm një pikë hyrëse (●) dhe vetëm një pikë dalëse (⊙), ***figura 2.1***.

2.1 Struktura kontrolluese rendore

Kjo strukture kontrolluese përbëhet prej hapave algoritmik të cilët realizojnë sipas asaj renditje me të cilën janë përmendur. Atë e shfrytëzojmë te të gjithë shembujt e deritanishëm.

Grafikisht, struktura është paraqitur te ***figura 2.1.1***.

Çdo strukture kontrolluese rendore paraqet tërësi të veçantë dhe mund të paraqitet kudo te algoritmi. Prandaj, është e nevojshme të shënohet fillimi dhe fundi i strukturës. Ajo bëhet e fjalët **fillim** (angl. begin) dhe **fund** (angl. end), ***figura 2.1.2.*** Struktura kontrolluese algoritmike rendore quhet **fillim–fund**.

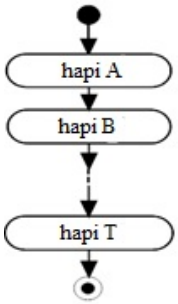


Figura 2.1.1

```
fillim
        hapi A;
        hapi B;
        ...
fund
```

***Figura* 2.1.2**

Shenja ; (pikëpresje) shfrytëzohet për ndarje të hapave te strukturë kontrolluese rendore **fillim–fund.** Përmbajtja e strukturës kontrolluese rendore është një **bllok** prej hapave, por për pamje më të madhe hapat shkruhen pak ë tërhequr djathtas prej fjalëve **fillim** dhe **fund**. Fjalët fillim dhe fund shkruhen në vertikalen e njëjtë. Kjo vlen për të gjitha strukturat kontrolluese algoritmike.

Kjo mënyrë e të shkruarit të algoritmeve dhe programeve me tërheqje (më së shpeshti për një pozitë tabulatore), quhet **dhëmbëzi** (angl. indentation). Dhëmbëzimi shfrytëzohet për pamje më të madhe dhe për njohje më të lehtë të strukturës kontrolluese algoritmike dhe të programeve.

Struktura kontrolluese algoritmike **fillim–fund** më së shpeshti shfrytëzohet në hapat për dhënien e vlerave fillestare dhe të të dhënave edhe gjatë njehsimeve.

Për shembull:

```
fillim  {Vlerat fillestare}
    a ← 3;
    b ← –5.74;
    c ← '@';
fund
```

Poashtu shigjeta majtas ← cshfrytëzohet se symbol për shoqërim të vlerave të ndryshoreve.

Te të gjitha programet që i shkruajtëm te **Moduli 1: Hyrje në gjuhën programore C++**, shfrytëzuam strukturë kontrolluese algoritmike **fillim–fund** pasi urdhërat i shkruajtëm njëra ps tjetrës.

Në disa raste, nëse duhet të shënohet grupi i urdhërave me të cilët realizohej një hap algoritmik (për shembull, hapi: leximi i të dhënave), grupi vendohet në kllapa të mëdha {}. Kllapat {} e shënojnë bllokun e urdhërave të cilat duhet të realizohen.

Kështu,, struktura kontrolluese algoritmike **fillim–fund** prej ***figura 2.1.2***, do të shkruhet si te ***figura 2.1.3***.

```
{
        urdhri A;
        urdhri B;
        ...
        urdhri P;
}
```

Figura 2.1.3

Për shembull, segmenti algoritmik për dhënie vlera fillestare të të dhënave, do të shkruhet me këtë segment programor:

```
{       // Vlera fillestare
        a = 3;
        b = -5.74;
        c = '@';
}
```

Vërejtje: Pas kllapave {}, nuk vendohet shenja pikë

Pas ekzekutimit të segmentit të programit të mëposhtëm

```
{
        a = 2;
        b = 3;
        p = a;
        a = b;
        b = p;
}
```

vlerat e ndryshoreve a dhe b do të jenë 3 dhe 2.

2.2 Strukturat kontrolluese algoritmike për zgjedhje dhe urdhëra kontrolluese për zgjedhje

Struktura kontrolluese algoritmike për zgjedhje prej dy mundësive **nëse–atëherë–ndryshe**

Struktura kontrolluese algoritmike për zgjedhje prej dy mundësive shfrytëzohet për zgjedhje të njërës prej dy kaheve të mundshme të vazhdimit të veprimit te algoritmi, varësisht prej ndonjë kushti. Kushti i ndonjë shprehje logjike. Shprehja logjike mund të ketë vetëm dy vlera: *saktë* dhe *pasaktë*. Ato quhen *vlera logjike* dhe prandaj, shprehja quhet *shprhje logjike*.

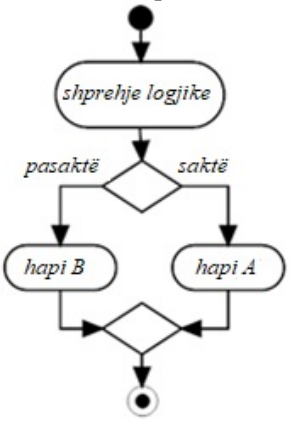


figura 2.2.1

Struktura kontrolluese për zgjedhje prej dy mundësive grafikisht është paraqitur te ***figura 2.2.1.***

Nëse vlera e s*hprehjes logjike është e saktë*, **atëherë** realizohet hapi A, **ndryshe** (nëse vlera është *e saktë*) realizohet haçi B. Prandaj, kjo strukture kontrolluese quhet **nëse–atëherë–ndryshe** (angl. if-then-else).

Shënimi tekstual strukturës të pseudo gjuhë është dhënë te ***figura 2.2.2***.

```
nëse  shprehje logjike
        atëherë
                hapi  A;
        ndryshe
                hapi  B;
fund_nëse  {shprehje logjike}
```

figura 2.2.2

Struktura fillon me **nëse**, por mbaron me **fund_nëse** {*shprehe logjike*}. Për shembull, te ***figura 2.2.3*** është shfrytëzuar kjo strukturë për gjetjen e nmrit më të madh prej numrave a dhe b.

```
nëse  a > b
        atëherë
            më i madh ← a;
        ndryshe
            më i madh ← b;
fund_nëse {a > b}
```

Figura 2.2.3

Treguam se hapat te një algoritëm mund të jenë më të përgjithshme dhe të përbëhen prej më shumë hapave tjerë ose madje edhe prej strukturave kontrolluese të plota. Nëse hapi A dhe hapi B prej ***figura 2.2.2*** përbëhen prej hapave tjerë, atëherë ato paraqesin një bllok **algoritmik** dhe më së shpeshti shënohet me fjalët **fillim** dhe **fund**, si te ***figura 2.2.4***. Domethënë, hapat A1, A2... An dhe B1, B2... Bm janë blloqe algoritmike në mundësi **atëherë**, përkatësisht në mundësinë **ndryshe**.

```
nëse  shprehje logjike
        atëherë
                fillimi
                        hapi  A1;
                        hapi  A2;
                        ...
                        hapi  An;
                fund
        ndryshe
                fillimi
                        hapi  B1;
                        hapi  B2;
                        ...
                        hapi  Bm;
fund
fund_nëse  {shprehje logjike}
```

Figura 2.2.4

Për shembull, për gjetjen e numrit më të madhprej dy numrave të dhënë a dhe b, me shfrytëzimin e kësaj strukture kontrolluese, do të shkruajmë:

```
më i madh ← b;
nëse  a > b
        atëherë
                më i madh ← a;
fund_nëse  {a > b}
```

Struktura kontrolluese për zgjedhje prejdy mundësive mund të shfrytëzohet edhe kur njëra prej mundësive të shprehjes logjike nuk përmban hapa realizues. Në këtë rast, vazhdohet te hapi vijues pas strukturës kontrolluese. Shënimi tekstual është dhënë te ***figura 2.2.5.***

```
nëse  shprehje logjike
        atëherë
                hapi  A;
fund_nëse {shprehje logjike}
```

Figura 2.2.5

Struktura kontrolluese e këtillë quhet nëse-atëherë (angl. if-then). then).

Paraqitja grafike e strukturës kontrolluese **nëse-atëherë** është dhënë te ***figura 2.2.6.***

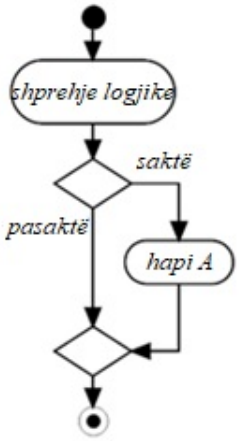


Figura 2.2.6

Urdhri kontrollues për zgjedhje prej dy mundësive if

Me urdhrin kontrollues për zgjedhje if, realizohet struktura kontrolluese algoritmike për zgjidhje prej dy mundësive **nëse–atëherë**, ku njëra mundësi nuk a urdhër, ***figura 2.2.5*** dhe ***figura 2.2.6***. Shprehja logjike është ndonjë kusht për realizmin e udhëtarit A ose për mos realizimin, ***figura 2.2.7***.

```
if(kusht)
        urdhër A;
```

Figura 2.2.7

Gjatë realizimit të urdhrit kontrollues për zgjedhje if, së pari shqyrtohet kushti dhe nëse është plotësuar (ka vlerë true), realizohet urdhri A, por nëse nuk është plotësuar (ka vlerë false), nuk realizohet urdhri A, por realizimi i programit vazhdon me këtë urdhër.

Nëse duhet të realizohen më shumë urdhëra,ato veëndohen në bllok me kllapa të mëdha, ***figura 2.2.8.***

```
if(kusht) {
        urdhër 1;
        ...
        urdhër m;
}
```

Figura 2.2.8

Nëse *kushti* ka vlerë true, atëherë realizohen urdhërat prej bllokut {urdhri 1... urdhri m}, ndryshe urdhërat prej bllokut nk realizohen.
kusht mund të jetë vetëm shprehje logjike, i cili mundet të ketë vlerë true ose false.

Shprehjet logjike në C++ ndërtohen me operatorët e relacionit dhe logjike[1].
Shembuj përshprehje logjike janë:

```
(a > b) && (a >= c)
((alfa < 90) && (beta > 0)) || (alfa – beta > 0)
!(x <= y)                    njëjtë me    x > y
!((a > b) && (a >= c))       njëjtë me    (a <= b) || (a < c)
!((x != y) || (x > y))       njëjtë me    (x == y) && (x <= y)
```

Këta tre shembuj e ilustrojnë urdhrin kontrollues për zgjedhje **if**:

a)

```
int x;
if(x < 0)
        xaps = –x;
```

b)

```
int x;
if (x != 0)
     x = 0;
```

c)

```
int x, y, pom;
if (true) {
     pom = x;
     x = y;
     y = pom;
}
```

[1] Për ato folëm te **Llojet logjike të të dhënave** te nënpika **1.7. Llojet etë dhënave.**

Shembuj

Shembulli 2.2.1

Të shkruhet segment programor për konstatimin numri i future a është pozitiv.

```
int numri;
cout << "Futni nmër: ";
cin >> numër ;
if(numër  > 0) {
	cout << "Numri i future është pozitiv." << endl;
}
```

Shembulli 2.2.2

Të shkruhet segment programor për njehsimin e vlerës absolute.

```
int numri;
cout << "Futni numër të plotë: ";
cin >> numri;
cout << "Vlera absolte e numrit  " << numri;
if (numri < 0) {
	numri = -numri;
}
cout << " e " numri << endl;
```

Urdhri kontrollues përzgjedhje prej dy mundësive if-else

Me urdhrin kontrollues për zgjedhje if-else, realizohetstruktura kontrolluese algoritmike për zgjedhje prej dy mundësive **nëse-atëherë-ndryshe**, ***figura 2.2.1*** dhe ***figura 2.2.2.***

Gjatë realizimit të urdhrit kontrollues për zgjedhje if-else (***figura 2.2.9***), së pari shqyrtohet kushti dhe nëse është plotësuar (ka vlerë true), realizhet urdhri A, por nëse nuk është plotësuar (ka vlerë false), realizohet urdhri B.

Nëse duhet tërealizohen më sumë urdhëra kur është plotësuar kushti ose kur ai nuk është plotësur, ato urdhëra vendohen në bllok me kllapa të mëdha, f***igura 2.2.10***.

Nës *kushti* ka vlerë true, atëherë realizohen urdhërat prej bllokut { urdhër A1... urdhër Ap}, por nëse *kushti* ka vlerë false, realizohen urdhërat prej bllokut {urdhër B1... urdhër Bq}.

```
if(kusht)
	urdhër A;
else
	urdhër B;
```

Figura 2.2.9

```
if(kusht) {
	urdhër A1;
	...
	urdhër Ap;
}
else {
	urdhër B1;
	...
	urdhër Bq;
}
```

Figura 2.2.10

Shembuj

Shembulli 2.2.3

Nëse gjatë përgjigjes së saktë të ndonjë prerje duhet të përgjiget me një porosi, por gjatë përgjigjes të pasaktë me tjetrën, atëherë atë mundemi ta shkruajmë me këtë segment programor:

```
char përgjigje;
cout << "Futni përgjigjen P për të saktë: ";
cin >> përgjigje;
if (përgjigje == 'D')
	cout << "Përgjigja është e sakë." << endl;
else
	cout << "Përgjigja është e sakë." << endl;
```

Shembulli 2.2.4

Këndi alfa është këndi ngushtë nëse është më i madh prej 0^0 he më i vogël se 90^0. Kjo mund të shkruhet me këtë segment programor:

```
int alfa;
cout << "Futni kënd në shkallë: ";
cin >> alfa;
if ((alfa > 0) && (alfa < 90))
	cout << "Këndi është i ngushtë." << endl;
else
	cout << "Këndi është i ngushtë." << endl;
```

Shembulli 2.2.5

Segmenti programor vijues është i rregullt (përkthyesi nuk paraqet gabim), por punon gabimisht:

```
bool Po Jo;
cin >> Po Jo;
if (Po Jo = true) {
	cout << "Përgjigja është e saktë." << endl;
	cout << "Por segmenti është gabimisht." << endl;
}
else
	cout << "Përgjigja nuk është e saktë." << endl;
```

Çfarëdo vlere ta fsim për ndryshore Po Jo, segmenti gjithmonë shtyp `Përgjigja është e saktë.`

Gabimi është te kushti `Po Jo = true` pasi ai është urdhër për shoqërim (me operatorin =), por jo kusht për bari (me operator ==). Rezultati prej realizimit të tij gjithmonë është true dhe prandaj, gjithmnë shtypet `Përgjigja është e saktë.`

Shembulli 2.2.6

Shpeshherë është e nevojshme kontrolli i vlerës të emëruesit gjatë pjesëtimit të dy numrave pasi nukështë e lejuar pjesëtimi me 0. Segmenti programor vijues ilustron se si dhet të shmanget pjesëtimi me 0:

```
int n;
cout << "Futni numër të plotë: ";
cin >> n;
if (n != 0)
    cout << "Vler reciproke e  " << n << "është: " << 1./ n << endl;
else
       cout << "Nuk është e lejuar pjesëtimi me 0." << endl;
```

Ushtrime

Ushtrimi 2.2.1

Cilat prej këtyre segmenteve programore japin rezultat të njëjtë?

a)

```
if(a > b)
      cout << a;
Else
      cout << b;
```

b)

```
if(a > b)
      cout << a;
```

c)

```
if(a <= b)
      cout << b;
Else
      cout << a;
```

Ushtrimi 2.2.2

Cili është dryshimi ndërmjet këtyre segmenteve programore?

a)

```
if((a == b) || (a == c))
       cout << a << endl;
else
       cout << b << ' ' << c << endl;
```

b)

```
if((a != b) && (a != c))
       cout << b << ' ' << c << endl;
else
       cout << a << endl;
```

Ushtrimi 2.2.3

Çka punon ky segment programor?

```
if(a < b)
      min = a;
else
      min = b;
 cout << min << endl;
```

Detyra të zgjidhura

Detyra 2.2.1

Të kontrollohet se numri natyror i futur a është numër çift se tek.

```
1   // Të kontrollohet numri i futur a është
2   // çift ose tek.
3    #include <iostream>
4    using namespace std;
5
6   int main() {
7        int numër;
8        cout << "Futni numër natyror:"      ;
9        cin >>numër;
10       if(numër % 2 == 0 )
11           cout << "Numri i futur"      << numër<< "është çift ." << endl;
12       else
13           cout << "Numri i futur"      << numër<< "është tek."     << endl;
14
15       cout << endl;
16       system( "Color 17" );
17       system( "pause" );
18       return 0;
19   }
```

Figura 2.2.11

```
Futni numër natyror: 123
Numri i futu 123 është tek.
Press any key to continue . . .
```

Detyra 2.2.2

Të kontrollohet numri i futur a është pozitiv, negative ose zero.

```
1   // Të provhet vall numri i future është
2   //  pozitiv, negative ose zero
3    #include <iostream>
4    using namespace std;
5
6   int main() {
7       float numri;
8       cout << " Futni numër  ";
9       cin >> numri;
10      cout << "  Numri i futu  " << numri << "është;";
11      if (numri< 0) {
12          cout << " negative ." << endl;
13      }
14      if (numri> 0) {
15          cout << "pozitiv ."   << endl;
16      }
```

Figura 2.2.12

```
17      if(numri == 0) {
18          cout << "zero ." << endl;
19      }
20
21      cout << endl;
22      system("Color 17");
23      system("pause");
24      return 0;
25  }
```

Figura 2.2.12 (vazhdim)

Një dalje prej realizimit të programit është:

```
Numri i futur      -12.345
Numri i future      -12.345 është: negativ
Press any key to continue . . .
```

Detyra 2.2.3

Të kontrollohet vall numri i future është në vargun e llojit përkatës të numrave.

Vërejtje: Te nëntitulli **Leximi dhe shtypja e të dhënave numerike** prej nënpikës **1.8 Leximi dhe shtypja e të dhënave**, sqaruam sesi ndodh mbushja (ang. overflow). Për të mos ardhur deri tejmbushja, pasi mund të fitohen rezultate të gabuara, shpesh është e nevojshme kontrolli i ndonjë të dhënë a është ai në vargune vlerave të llojit prej të cilit është ndryshorja e deklaruar.

Te shembulli vijues është ilustruar mbushja e numrave të plotë.

```
1   //Mbushja e numrave të plotë
2   #include <iostream>
3
4   using namespace std;
5
6   int main() {
7
8       int numërIPlotë;
9       double të3;
10      cout << "Vargu i llojit është       [" << INT_MIN
11          << ", " << INT_MAX << "]" << endl;
12      cout << "Futni numër të plotëmë të madh se 1290: ";
13      cin >> numërIPlotë;
14      na3 = 1.0 *numërIPlotë*numërIPlotë * numërIPlotë;
15      cout << "Numri i plotë " << numërIPlotë << "në kub është";
16      if( ( INT_MIN <= të3 ) && ( të3 <= INT_MAX ) )
17          cout << int( të3 ) << endl;
18      else {
```

Figura 2.2.13

```
19          cout << " Erdh deri në mbushje     ." << endl;
20          cout << "   Pasi     " << numërIPlotë << "në kub nuk është "
21              << numërIPlotë*numërIPlotë* numërIPlotë   << endl;
22      }
23
24      cout << endl;
25      system( "Color 17" );
26      system( "pause" );
27      return 0;
28  }
```

Figura 2.2.13 (vazhdim)

Një dalje prej realizimit të programit është:

```
Vargu i llojit int është:       [-2147483648, 21474836471
Futni numer te plote me të madh prej 1290:  1291
Numri i plotë    1291 në kub është:Erdh deri në mbushje .
Pasi 1291 në kub nuk është-2143282125

Press any key to continue . . .
```

Detyra për ushtrime

Të shkruhen programe për këto detyra:

1. Të njehsohet katrori dhe kubi i numrit natyror n.
2. Të njehsohet perimetri i vijës rrethore dhe syprina e rrethit me rreze r.
3. Të lexohet numri natyror teshifror dhe të shtypet shifra e mesme.
4. Të lexohen tre numra dhe të konstatohet a mund të jenë brinjë të trekëndëshit.
5. Të renditen tre numra sipas madhësisë.
6. Të zgjidhet barazimi linear: $ax + b = 0$, për $a \neq 0$.
7. Të zgjidhet pabarazimi linear: $ax + b > 0$, për $a \neq 0$.
8. Të lexohet këndi më i vogël se 180^0 dhe të shtypet a është i ngushtë ose gjerë.
9. Të kontrollohet data e futur a është e drejtë. (Data të futet sit re numra të plotë për ditën, muajin dhe vitin).
10. Të njehsohet mosha e njeriut si ndryshim të datës sotshme të lindjes. Datat të futen i numra të plotë për ditën, muaji dhe vitin.

Folezimi i urdhërave kontrolluese if dhe if-else

Urdhërat kontrolluese if dhe if-else mund të folezohen (angl. nested) qoftë në zgjedhjen if qoftë në zgjedhjen else.

Kur folezimi është në zgjedhjen else, urdhri i këtillë quhet if-else-if.

Ta shqyrtojmë këtë shembull.

Shembulli 2.2.7

Tre numra të plotë a, b dhe c a mund të jenë brinjët e trekëndëshit kënddrejtë?

Zgjidhje: Numrat a, b dhe c mund të jenë brinjët e trekëndëshit kënddrejtë nëse $a^2 + b^2 = c^2$ ose nëse $a^2 + c^2 = b^2$ ose nëse $b^2 + c^2 = a^2$.

Këtë fjali mundemi ta shprehim me këtë segment programor:

```
if(a * a + b * b == c * c)
       cout << "Po";
else   if(a * a + c * c == b * b)
       cout << "Po";
else   if (b * b + c * c == a * a)
       cout << "Po";
else
       cout << "Jo";
```

Folezimi mund të jetë edhe në zgjedhjen e if.

Ta shqyrtojmë këtë shembull.

Shembulli 2.2.8

Të gjendet nuri më i madh prej numrave të dhënë a, b dhe c.

Zgjidhje:

Nëse a > b dhe a > c, atëherë numri më i madh është numri a.

Nëse b > a dhe b > c, atëherë numri më i madh është numri b.

Nëse c > a dhe c > b, atëherë numri më i madh është numri c.

Sementi programor do të jetë:

```
if(a > b)
       if(a > c)
              cout << "Numri më i madh është  " << a;
if(b > a)
       if(b > c)
              cout << "Numri më i madh është  " << b;
if(c > a)
       if(c > b)
              cout << "Numri më i madh është  " << c;
```

Vërejtje: Nuk është e lejuar folezim i urdhrit programor për zgjidhje të jetë edhe te zgjedhja **if** edhe te zgjedhja **else**.

Gjatë folezmit të urdhërave kontrollues për zgjedhje if, if-else dhe if-else-if, duhet pasur kujdes se përkthyesi në C++ çdo else elidh me parprakun if. Për shembull, te segmenti programor vijues,për a < b, nuk ka asgjë të shtypet pasi else është lidhur me të dytën if, por jo me të parën.

```
if(a > b)
	if(a > c)
		cout << "Numri më i madh është  " << a;
	else
		cout << "Numri më i madh është  " << a;
```

Nëse dum për a < b të realizohet urdhri

```
cout << "Numri më i madh nuk është  " << a;
```

atëherë duhet else të lidhet të parën if me kllapa.

```
if(a > b) {
	if(a > c)
		cout << "Numri më i madh është  " << a;
}
else
	cout << "Numri më i madh nuk është  " << a;
```

Kjo quhet **problem i varur else** (angl. dangling-else problem).

Ushtrime

Për **Shembullin 2.2.7** dhe **Shembulli 2.2.8** të shkruhen programe të veçanta në C++.

Detyra të zgjidhura

Detyra 2.2.4

Të shtypet suksesi i përgjithshëm i nxënësit nëse është mesatarja e notave prej lëndëve, dh atë:

Mesatarja	Suksesi i përgjithshëm
1 deri 1.5	Pamjaftueshëm
1.6 deri 2.5	Mjaftueshëm
2.6 deri 3.5	Mirë
3.6 deri 4.5	Shumë mirë
4.6 deri 5.0	Shkëlqyeshëm

Prormi në C++ do te jetë sikurse te ***figura 2.2.14.***

```
1   //Të shtypet suksesi i përgjithshëm të nxënësit nëse dihe nota mesatare.
2   #include <iostream>
3   using namespace std;
4
5   int main() {
6       float mesatare;
7       cout << "Futni notën mesatare  : ";
8       cin >>  mesatare;
9       cout << "Suksesi i prgjithshëm për notën mesatare " << mesatare << " e ";
10      if ( mesatare  <= 1.5)
11          cout << "Pamjaftueshëm << endl;
12      else    if ( mesatare  <= 2.5)
13          cout << " Mjaftueshëm " << endl;
14      else    if ( mesatare <= 3.5)
15          cout << " Mirë " << endl;
16      else    if ( mesatare <= 4.5)
17          cout << "Shumë mire" << endl;
18      else
19          cout << " Shkëlqyeshëm " << endl;
20
21      cout << endl;
22      system("Color 17");
23      system("pause");
24      return 0;
25  }
```

Figura 2.2.14

```
Futni ntë mesatare 3.56
Suksesi i përgjithshëm për notën mesatare 3.56 është Shumë
Mirë
Press any key to continue . . .
```

Të kujtohemi se kur ka vetëm një urdhër në if ose në else, atëherë mudet, por jo patjetër, të vendohen në kllapa të mëdha.

Detyra 2.2.5
Të caktohet numri më i madh prej tre numrave të dhënë.

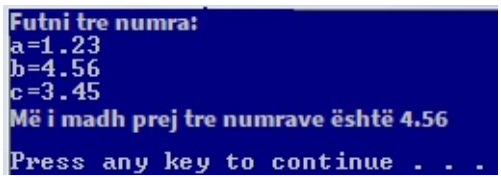
Proami në C++ është dhënë te ***figura 2.2.15.***

```
1   //Të caktohet numri më i madh prej tre numrave
2
3   #include <iostream>
4   using namespace std;
5
6   int main() {
7       double a, b, c;
8       cout << "  Futni tre numra:   \n";
9       cout << "a="; cin >> a;
```

Figura 2.2.15

```
10      cout << "b="; cin >> b;
11      cout << "c="; cin >> c;
12      cout << "Më i madh prej tre numrave është ";
13      if (a > b) {
14          if (a > c)
15              cout << a << endl;
16          else
17              cout << c << endl;
18      }
19      else {
20          if (b > c)
21              cout << b << endl;
22          else
23              cout << c << endl;
24      }
25
26      cout << endl;
27      system("Color 17");
28      system("pause");
29      return 0;
30  }
```

Figura 2.2.15 (vazhdim)



Operator i kushtëzuar?:

Operatori i kushtëzuar shfrytëzohet për paraqitjen e urdhërave të zakonshme if-else. Sintaksa e tij është:

kushti? shprehja S : shprehja J ;

Së pari njehsohet *kushti* vlera e të cilit mund të jetë true ose false. Nëse është true, atëherë njehson shprehje S, por nëse është false njehsohet shprehja J.

Për shembull, segmenti programor:

```
if(m > n)
        max = m;
else
        max = n;
```

mund të shkruhe shkurt

```
max = (m > n) ? m : n;
```

Me këtë urdhër do të shtypet „çift.“ nëse ndryshorja numër ka vlerë çift, në të kundërtën do të shtypet „tek“ nëse ndryshorja numër ka vlerë tek:

```
cout << "Numri  " << numri  << "është  "
              << ((numri  % 2 == 0) ? "çift" : "tek") << endl;
```

Për shembull, për numrin = 23 do të shtypet:

```
Numri 23 është numër tek
```

Detyra për ushtrime

Të shkruhen programe për folezim për zgjidhjen e këtyre detyrave me folezim te urdhërat kontrollues për zgjedhje:

1. Të futen dy numra dhe operatori për operacionin aritmetik (+, –, * ose /) dhe të kryhet operacioni.
2. Të cktohet ekuivalenti arab të shifrës romake. (Shifrat romak janë: I = 1, V = 5, X = 10, L = 50, C = 100, D = 500 dhe M = 1 000).
3. Të caktohet nota për nxënësin sipas pikëvetë fituara nëse vlerësi kryhet sipas kësaj shkalle të pikëve: më pak se 60 – pamjaftueshëm, prej 60 deri 69 – mjaftueshëm, prej 70 deri 79 – mirë, prej 80 deri 89 – shumë mire dhe më shumë prej 90 – shkëlqyeshëm.
4. Të caktohet nxënësi që ka marrë notë kaluese nëse kushti për të kaluar është të fitojë 60 pikë prej gjithsej 100, me shfrytëzimin e operatorit të kushtëzuar?:.
5. Të caktoet se vija rrethore $O(x_1, y_1, r_1)$ dhe $O(x_2, y_2, r_2)$ a priten, me shfrytëzimin e operatorit të kushtëzuar?:. (Largësia ndërmjet dy pikave njësohet sipas formulës $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$).

Struktura kontrolluese algoritmike për zgjedhje prej më shumë mundësive RAST dhe urdhri kontrollues për zgjedhje prej më shumë mundësive switch

Kur ka më shumë mundësi për vazhdimin e veprimit te algoritmi, shfrytëzohet **struktura kontrolluese algoritmike për zgjedhje prej më shumë mundësive**, ***figura 2.2.16***. Zgjedha e njërës prej mundësive kryhet, varësisht prej vlerës të ndonë të dhëne ose të ndonjë *shprehje* aritmetike.

Vlera e *shprehjes* mund të jetë a, b... k ose mund *shprehja* të os fiton asnjë prej atyre vlerave. Në rasti shprehja të fitojë vlerë a, veprimi vazhdon me hapin A, në rastin kur vlera e *shprehjes* të jetë b, veprimi vazhdon me hapin B etj. Në rastin *shprehja* të mos fiton asnjë

```
rast shprehja
    a: hapi A;
       ndërprerje;
    b: hapi B;
       ndërprerje;
    . . .
    k: hapi K;
       ndërprerje;
    ndryshe
       hapi X;
fund_rast {shprehje}
```

Figura 2.2.16

prej vlerave a, b... k, atëherë veprimi vazhdon me hapin X.

Prej asaj që u tha deri më tani, vërehet se kjo strukture kontrolluese algoritmike kryhet zgjedhje të një ose më shumë rasteve të mundshme të vazhdimit të veprimet. Prandaj, ai quhet **zgjedhje në rast** ose., më shkurt, vetëm **rast** (angl. case).

Pas hapave a, b... k, është përmendur struktura kontrolluese algoritmike **ndërprerje**, me tëcilën ndërpritet realizimi i hapave tjerë te struktura kontrolluese rast dhe veprimi vazhdon me hapin ose struktura kontrolluese sipas **rast**. (Për strukturën kontrolluese algoritmike ndërprerje do të flasim te nënpika **2.4 Struktura kontrolluese algoritmike për kërcim dhe urdhëra kontrolluese për kërkim**).

Kjo strukture kontrolluese grafikisht është paraqitur te ***figura 2.2.17***.

Kur është e nevojshme të kryhet hap i njëjtë për më shumë vlera të *shprehjes*, atëherë struktura **rasti** shkruhet si te ***figura 2.2.18***.

```
rast shprehje
    a, c, d:  hapi A;
              ndërprerje;
    b, g:     hapi B;
              ndërprerje;
    . . .
    k, i, f, j: hapi K;
              ndërprerje;
    ndryshe
              hapi X;
fund_rast {shprehje }
```

Figura 2.2.18

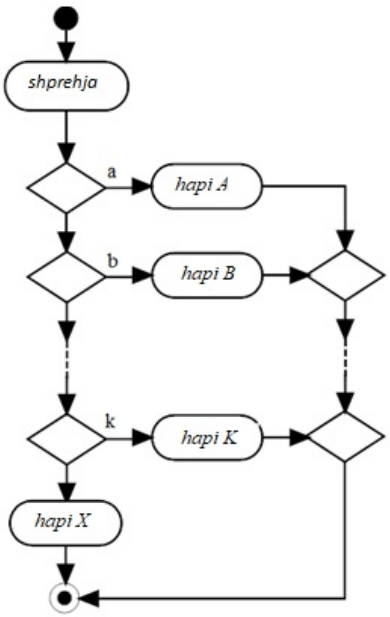


Figura 2.2.17

Me urdhrin kontrollues **switch** realizohet struktura kontrolluese algoritmike për zgjedhje prej më shumë mundësive **rast**, por sintaksa e saj është dhënë te f***igura 2.2.19.***

Shprehja në kllapa (shprehja) mund të jetë çfarëdo shprehje vlerat e të cilës janë lloj numra të plotë (short, int, long, long long), lloj me shenja (char), lloj logjik (bool) ose lloj të numërueshëm (enum). Shenjat a, b... k janë konstante prej numrave të plotë, me shenja, lloje logjike ose të numërueshme. Ato nuk mund të jenë ndryshore.

```
switch(shprehje) {
          case a: urdhri A;
                  break;
          case b: urdhri B;
                  break;
          ...
          case K: urdhri K;
                  break;
          default:

                  urdhri X;
 }
```

Figura 2.2.19

Për çdo rast (case), realizohen tëgjitha urdhërat deri te urdhri kontrollues i arë break, veprimi i të cilit është kërcim (dalje) në fund të urdhrit kontrollues switch. Prandaj, patjetër të shfrytëzohet urdhri kontrollues break pas çdo case. Nëse nuk përmendet urdhri kontrollues break, realizimi vazhdon me këtë case2. Pas urdhrit X, nuk vendohet break.

Nëse *shprehja* nuk fiton asnjë vlerë prej konstantave të përmendura pas fjalës case, atëherë realizohet urdhri X pas fjalës default. Departamenti default është opcopnal, jo patjetër të përmendet. Nëse nuk ekziston, pas realizimit të çfarëdo qoftë case, klrcen në urdhrin vijues sipas urdhrit kontrollues switch.

Shembuj

Shembulli 2.2.9

Segmenti programor për vërtetim a jeni të ri (e re) ose jeni i rritur (e rritur) është:

```
bool Po Jo;
char PJ;
cout << "A jeni të lindur në shekullin 21? (P, J: "; cin >> PJ;
Po Jo = true;
if(PJ == 'J')
       Po Jo = false;
switch(Po Jo) {
       case true:    cout << "Ju jeni i ri (e re)." << endl;
              break;
       case false:   cout << "Ju jeni i rritur (e rritur)." << endl;
}
```

Shembulli 2.2.10

Të futet shenja për operacionin aritmetik: +, –, * ose / dhe të shtypet rezultati.

```
switch(shenja) {
case '+': cout << shenja << "plus " << endl;
       break;
case '-': cout << shenja << "minus " << endl;
       break;
case '*': cout << shenja << "nga  " << endl;
```

[2] Ky është gabim i shpeshtë, prandaj duhet të keni kujdes të vendohet break në fund të çdo case.

```
        break;
case '/': cout << shenja  << " mbi  " << endl;
        break;
default:
       cout << shenja  << " nuk është shenjë për operacion aritmetik." << endl;
}
```

Grupi i njëjtë i urdhërave mund të realizohet për raste të ndryshme (cases) deri te urdhri i parë break.

Shembulli 2.2.11

Të futet shkronja dhe të shtypet a është zanore ose është bashkëtingëllore.

```
switch(shkronjë) {
case 'a':
case 'e':
case 'i':
case 'o':
case 'u':
        cout << "Shkronja  " << shkronja  << " është zanore." << endl;
        break;
default:
        cout << "Shkronja  " << shkronja  << " është bashkëtingëllore." << endl;
}
```

Shembulli 2.2.12

Nëse ndonjë muaj i vitit është dhënë me numrin e tij rendor, atëherë me urdhrin kontrollues switch, mund të caktohet sa ditë ka ai muaj.

Vërejtje: Në vitin e brishtë muaj shkurt ka 29 ditë. Viti i brishtë është ai vit i cili plotpjesëtohet me 4 dhe nuk plotpjesëtohet me 100 ose e cila plotpjesëtohet me 400.

```
switch(muaj){
        case 1: case 3: case 5: case 7: case 8: case 10: case 12:
                ditë  = 31;
                break;
        case 4: case 6: case 9: case 11:
                ditë  = 30;
                break;
        case 2:
                ditë  = (((viti  % 4 == 0) && (viti  % 100 != 0)) ||
                (viti  % 400 == 0)) ? 29 : 28;
}
        cout << muaj  << "-muaj në  " << viti  << " viti ka  "
        << ditë  << " ditë ." << endl;
```

Shembulli 2.2.13

Segmenti programor për detyrën paraprake kur muajt prej llojit të numërueshëm do të jetë:

```
enum muaj  {JANAR, SHKURT, MARS, PRILL, MAJ, QERSHOR,
              KORRIK, GUSHT, SHTATOR, TETOR, NËNTOR, DHJETOR
}
muaj  muajEnum = SHKURT;
switch(muajEnum) {
       case JANAR: case MARS: case MAJ: case QERSHOR: case GUSHT:
       case TETOR : case DHJETOR :
              ditë  = 31;
              break;
       case PRILL: case QERSHORI: case SHTATOR: case NËNTOR:
              ditë  = 30;
              break;
       case SHKURT:
              ditë = (((viti  % 4 == 0) && (viti  % 100 != 0)) ||
              (viti  % 400 == 0)) ? 29 : 28;
}
       cout << muajEnum  << "-muaj në " << viti  << "viti ka  "
       << ditë  << " ditë ." << endl;
```

Shkruani 5 shembujt paraprak në një program dhe realizoni.

Detyra të zgjidhura

Detyra 2.2.6

Të shkruhet program me të cilën do të shtypet suksesi i përgjithshëm i nxënësit kur është dhënë numri i suksesit 5, 4, 3, 2 ose 1.

Programi është dhënë te ***figura 2.2.20.***

```
1   // Suksesi i përgjithshëm i nxënësit
2   #include <iostream>
3   using namespace std;
4
5   int main() {
6       int nota;
7       cout << "Futni nota  (1,2,3,4,5): ";
8       cin >> nota;
9       cout << "Suksesi i përgjithshëm i nxënësit";
10      switch( nota; ) {
11          case 5:     cout <<"Shkëlqyeshëm" << endl;
12              break;
13          case 4:     cout << "Shumë mirë"  << endl;
14              break;
```

Figura 2.2.20

```
15          case 3:     cout << "Mire"  << endl;
16              break;
17          case 2:     cout <<"Mjaftueshëm" << endl;
18              break;
19          case 1:     cout << "Pamjaftueshëm" << endl;
20              break;
21          default:    cout << "Kjo nuk është notë"  << endl;
22      }
23
24      cout << endl;
25      system( "Color 17" );
26      system( "pause" );
27      return 0;
28  }
```

Figura 2.2.20 (vazhdim)

Një dalje gjatë realizimit të programit është:

```
Futni nota       (1,2,3,4,5): 5
Suksesi i përgjithshëm i nxënësit është:
Shkëlqyeshëm
Press any key to continue . . .
```

Detyra 2.2.7

Të shkruhet program me të cilin për shifrën e future ndërmjet 1 dhe 9 do të shtypen a është çift ose tek. Nëse shifra nuk është ndërmjet 1 dhe 9 të shkruhet komenti.

```
1   // Shifrat çift dhe tek
2   #include <iostream>
3   using namespace std;
4
5   int main() {
6       short shifra;
7       cout <<"Futni shifër prej 1 deri 9 " ;
8       cin >> shifra;
9       switch( shifra; ) {
10          case 1: case 3: case 5: case 7: case 9:
11              cout <<"Është future shifra tek" << cifra << endl;
12              break;
13          case 2: case 4: case 6: case 8:
14              cout << "Është future shifra çift "  << cifra << endl;
15              break;
16          default: cout <<"Shenja e future nuk është shifër:" << cifra << endl;
17      }
18
19      cout << endl;
20      system( "Color 17" );
21      system( "pause" );
22      return 0;
23  }
```

Figura 2.2.21

NJë dalje prej realizimit të programit është:

```
Futni shifër prej 1 do 9: 5
Është future shifra tek 5
Press any key to continue . . .
```

Detyra për ushtrime

Për këto detyra të shkruhen programe me shfrytëzimin e urdhrit kontrollues për zgjedhje prej më shumë mundësive switch:

1. Të futet numri natyror më i vogël se 10 dhe të shtypet a është i thjeshtë[3], i plotpjesëtueshëm me 2, i plotpjesëtueshëm me 3 ose i përsosur[4].
2. Të shtypet kjo meny:

```
a. Profesor nga gjuha maqedonishte
b. Profesor nga informatika
c. Profesor nga matematika
d. Profesor nga fizika

Zgjidhni:    a
```

Pastaj, varësisht prej shkronjave të futura (a, b, c, d), të shzypet emri i profesorit të lëndës.

1. Të futet shifra 1, 2, 3... 9, 0 dhe të shtypet shifra me fjalë. Për shembull: për 1 të shtypet fjala „një“, për të shtypet fjala „dy“ etj.
2. Të caktohet ekuivalenti arab me shifrën romake (Shifra romak janë: : I = 1, V = 5, X = 10, L = 50, C = 100, D = 500 dhe M = 1 000).
3. Të caktohet nota për nxënësin sipas pikëve të fituara nëse vlerësimi kryhet sipas kësaj shkalle të pikëve 60 – pamjaftueshëm, prej 60 deri 69 – mjaftueshëm, prej 70 deri 79 – mirë, prej 80 deri 89 – shumë mirë dhe prej 90 deri 99 – shkëlqyeshëm.

Pyetje për kontroll të njohurive

1. Kur (më së shpeshti) shfrytëzohet struktura kontrolluese algoritmike rendore?
2. Cilat struktura kontrolluese algoritmike për zgjedhje i dini? Shkruani formën tekstuale të tyre.
3. Me cilin urdhër kontrollues algoritmik në C++ realizohet struktura kontrolluese algoritmike për zgjedhje prej dy mundësive **nëse–atëherë–ndryshe**? Përmend disa shembuj.

[3] Numër i thjeshtë është ai numër që është i plotpjesëtueshëm vetëm me numrin 1 dhe me vetveten. Numra të thjeshtë janë: 2, 3, 5, 7, 11 etj.

[4] Numër i përsosur është a ii cili është i barabartë me shumën e pjesëtuesve të vet, pa vet numrin. Numrat e përsosur janë: 6 (= 1 + 2 + 3), 28 (= 1 + 2 + 4 + 7 + 14), 496 etj.

4. Paraqitni grafikisht strukturën kontrolluese algoritmike nëse–atëherë–ndryshe.
5. Si është lloj i shprehjes që shfrytëzohet si kusht te struktura kontrolluese algoritmike për zgjedhje dhe urdhërat kontrolluese për zgjedhje?
6. Si shkruhet urdhri kontrollues për zgjedhje prej dy mundësive, kur sipas if dhe/ose sipas else ka më shumë se një urdhër? Përmend shembuj.
7. Shkruani sintaksën e urdhrit kontrollues për zgjedhje prej dy mundësive nëse nga një mundësi nk ks urdhëra. Përmend shembuj.
8. Cilat mënyra të folezimit te urdhërat kontrolluese për zgjedhje i dini?
9. Si quhet urdhri kontrollues për zgjedhje te e cila ka folezim te zgjedhja else?
10. Sqaro çka është varëse else. Përmend shembuj.
11. Shkruaj sintaksën e operatorit të kushtëzuar. Përmend shembuj.
12. Paraqitni tekstualisht dhe grafikisht strukturën kontrolluese algoritmike për zgjedhje prej më shumë mundësive.
13. Si mund të jetë lloj i shprehjes te urdhri kontrollues për zgjedhje prej më shumë mundësive?
14. Cili urdhër kontrollues vendohet në fund të çdo case te urdhri kontrollues switch?
15. Çka do të ndodh nëse në fund të çdo case nuk ka urdhër break?

Detyra

1. Çka do të shtypet pas realizimit të këtij segmenti programor nëse vlera e ndryshores n është 78?

```
if((n % 2 == 0) && (n % 3 == 0))
   cout << n << " plotpjesëtohet me 6." << endl;
else
   cout << n << " nuk plotpjesëtohet me  6." << endl;
```

2. Cili është gabimi te ky sistem programor:

```
if(x = a)
   cout << "x ka vlerë të njëjtë me a.";
else
   cout << "x ka vlerë të ndryshme me a.";
```

3. Pse ky segment programor gjithmonë shtyp „false“, pavarësisht prej vlerës së n?

```
cin >> n;
if(n = 0)
          cout << "true" << endl;
   else
          cout << "false" << endl;
```

4. Çka do të shtypet me këtë segment programor?

```
int i = 0, n;
float a = 1;
char c = 'a';
a++;
if(c == a && i == 0)
   n = 1;
else
   n = 2;
cout << "n = " << n << endl;
```

5. Këto segmente programore a japin rezultat të njëjtë?

```
if(a > b)
   cout << a << " > " << b << endl;
else if(b < c)
   cout << a << " <= " << b << "<" << c << endl;
else
   cout << a << " <= " << b << ">=" << c << endl;
```

```
if(a > b)
   cout << a << " > " << b << endl;
if(b < c)
   cout << a << " <= " << b << " < " << c << endl;
else
   cout << a << " <= " << b << " >= " << c << endl;
```

6. Të shkruhet segmenti programor për shprehjen logjike prej dy ndryshoreve:
   - shprehja është e saktë nëse ose ndr1 ose ndr2 është e saktë.
   - shprehja është e pasaktë nëse ose ndr1 ose ndr2 është e saktë.
   - shprehja është e pasaktë nëse edhe ndr1 edhe ndr2 është e pasaktë.
7. Të futen tre numra të plotë dhe të shtypet numri më i madh: njëherë nëse është më i madh prej dy të tjerëve, dy herë nëse të dy më të mëdhenj prej të tretit kanë vlerë të njëjtë, treherë nëse të tre numrat janë të barabartë.
8. Të plotësohet **Detyra 2.2.4** me kontrollin se nota mesatare e futur a është në intervalin [0.0, 5.0].
9. Të plotësohet **Detyra 2.2.6** me kontrollin se nota mesatare e future a është në intervalin [1, 5].
10. Të shkruhet program për shtypje të kësaj dalje, varësisht prej ndryshoreve.
11. Të kontrollohet se tre numra të ndryshëm a mund të jenë brinjë të trekëndëshit?
12. Shkruani programin që do të simulon kalkulator të zakonshëm me kto operacione: +, –, *, %, /, x^2.

2.3 Struktura kontrolluese algoritmike për përsëritje dhe urdhëra kontrollues për përsëritje

Struktura kontrolluese algoritmike për përsëritje shfrytëzohet atëherë kur është e nevojshme një grup hapa algoritmike të kryejë shumë here. Një realizim i hapave quhet një cikël, ***figura 2.3.1***. Prandaj, përsëritja e këtillë e realizimit të grupit të hapave algoritmike quhet **përsëritje ciklike**.

Për më lehtë të kuptohen këto struktura kontrolluese, do ta shqyrtojmë këtë detyrë.

```
cikli
        hapi A;
        hapi B;
        ...
        hapi M;
fundi_cikli
```

Figura 2.3.1

Detyra: Të caktohet shuma e 10 numrave të parë natyror.

Të mendojmë se kemi ndonjë enë dhe në atë i vendojmë numrat. Në fillim ena është e zbrazët, d.m.th., shuma është 0. Numri i parë natyror të cilin e vendojmë te ena është 1, domethënë tani shuma është 0 + 1 = 1. Pastah, te ena e vendosim numrin natyror vijues 2, domethënë në enë do të kemi dy numra (1 dhe 2), por shumë do të jetë 0 + 1 + 2. Pastaj, i vendosim numrat 3, 4, 5, 6, 7, 8, 9 dhe 10, por shuma do të zmadhohet me shtuarjen e çdo numri. Shuma e fundit do të jetë 0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10.

Mund të vërejmë se te mënyra për gjetjen e shumës realizohen vetëm dy veprime:

- Shtuarja e numri shumës.
- Zmadhimi i numrit për 1.

Pasi numri ndryshon prej 1 deri 10, por shuma është 0, 1, 3, 6, 10 etj., mundemi të futim dy numra të ndryshueshëm numri dhe shuma. Në fillim, përmbajtja e ndryshores shumë është 0, por vlera fillestare e numrit është 1. Prandaj, ndryshoret numri dhe shuma i inicializojmë vlert 1 dhe 0. Algoritmi mund të shkruhet tekstualisht si te ***figura 2.3.2.***

```
shuma ← 0;
numri ← 1;
cikli 10 herë
        – shtoje numrin shumës;
        – zmadhoje numrin për 1;
fund_cikli
```

Figura 2.3.2

Varësisht prej mënyrës së ndërprerjes së përsëritjes, dallojmë tri struktura kontrolluese për përsëritje, edhe atë:

- Struktura kontrolluese me numërim të cikleve.
- Struktura kontrolluese me dalje në fillim të ciklit.
- Struktura kontrolluese me dalje në fund prej ciklit

Strukturat kontrolluese algoritmike për përsëritje me numërim të cikleve dhe urdhërt kontrolluese for

Në këtë strukture kontrolluese, numri i përsëritjeve të ciklit është prej përpr i njohur. Përsëritjet, d.m.th., ciklet numërohen me numërues të veçantë, për të cilët janë dhënë vlera fillestare dhe e fundit, ***figura 2.3.3.***

Ciklet realizohen **për** çdo vlerë të ndryshores numërues prej fillestares, me **zmadhimin** nga 1, **deri** te fundi. Ndryshorja numërues zmadhohet për 1 para çdo cikli. Prandaj, struktura kontrolluese e këtillë quhet **për–zmadho–deri.**

```
për numërues ← fillestare zmadho deri   fundi
        hapi  A;
        hapi  B;
        ...
        hapi  K;
fund_për  {numërues}
```

Figura 2.3.3

Ndryshoret numërues, fillestare dhe e fundit patjetër të jenë ndryshore të llojit të njëjtë. Ndryshorja vlera fillestare ka vlerë më të vogël ose të barabartë prej ndryshores së fundit. Në të kundërtën, nuk realizohet asnjë cikël. Vlera e numëruesit nuk guxon të ndryshojë te hapat algoritmike prej të cilëve përbëhet cikli.

Nëse është e nevojshme përsëritja te e cila vlera fillestare (fillestare) duhet të jetë më e madhe prej të fundit (e fundit), atëherë shfrytëzohet strukture kontrolluese e ngjashme. Te ajo, cikli realikzohet **për** çdo vlerë të ndryshores numërues prej fillestares, e **zvogëlim** sipas 1, **deri** e fundit. Ndryshorja numërues zvogëlohet për 1 pra çdo cikli. Prandaj struktura kontrolluese e këtillë quhet **për–zvogëlo–deri,** ***figura 2.3.4*****.**

```
për numërues ← fillestare zvogëlo  deri fundit
        hapi  A;
        hapi  B;
        ...
        hapi  J;
fund_për  {numërues}
```

Figura 2.3.4

Te algoritmet (detyrat) ku numëruesi nuk zmadhohet ose zvogëlohet sipas 1, por sipas ndonjë sasie të hapave, struktura kontrolluese **quhet për–deri–hapi,** ***figura 2.3.5.***

Te kjo strukture kontrolluese, **për** ndonjë vlerë fillestare të numëruesit, **deri** ndonjë vlerë të fundit, ai zmadhohet në çdo cikël me **hapi** për sasi të caktuar. Prandaj, kjo strukture kontrolluese për përsëritje quhet **për–deri–hapi**.

```
për numërues ← fillestare deri fundit hapi sasia
        hapi  A;
        hapi  B;
        ...
        hapi  M;
fund_për  {numërues}
```

Figura 2.3.5

Nëse sasia (sasia) e hapit te struktura kontrolluese **për–deri–hapi** është +1, fitohet struktura kontrolluese **për–zmadho–deri**, por nëse sasia e hapit është –1, fitohet struktura kontrolluese **për–zvogëlo–deri**.

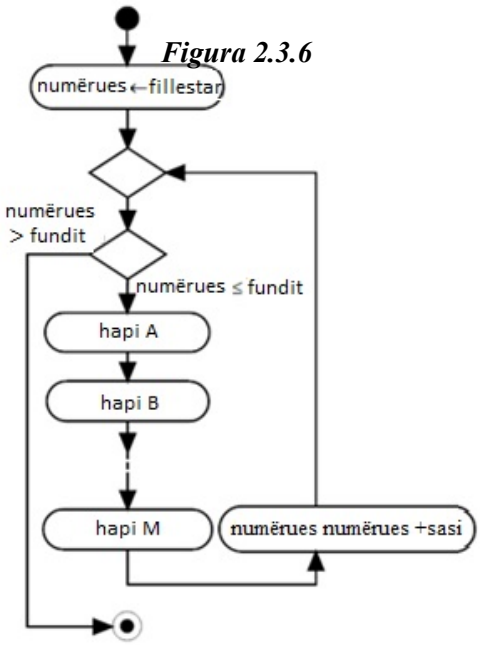
Paraqitja grafike e strukturës kontrolluese **për–deri–hapi** është dhënë te ***figura 2.3.6.***

Algoritmi për detyrën për gjetjen e shumës së 10 numrave të parë natyror, e shprehur me strukturën kontrolluese **për–zmadho–deri** është dhënë te ***figura 2.3.7.***

```
algoritëm ShumaDeri10
fillimi
        shuma ← 0;
        numri ← 1;
        për numërues ← 1 zmadho deri 10
                shuma ← shuma + numri;
                numri ← numri + 1;
       fund_për {numërues}
      shtyp shuma;
fund { ShumaDeri 10}
```

Figura 2.3.7

Figura 2.3.6



Për realizimin e tri strukturave kontrolluese: **për–zmadho–deri**, **për–zvogëlo–deri** dhe **për–deri–hap**, në C++ shfrytëzohet urdhri kontrollues for.

Forma e përgjithshme e urdhrit kontrollues for është dhënë te ***figura 2.3.8.***

Kushti është pjesë e shqyrtimit të kushtit para përsëri të realizojë ciklin. Ai është shprehje logjike.

Nëse *kushti* ka vlerë true, atëherë cikli përsëri do të realizojë, por nëse ka vlerë false, atëherë cikli nuk realizohet, por veprimi vazhdon me urdhrin vijues. Pasi kushti mund të mos plotësohet dhe para fillimit të ciklit të parë, mund të mos realizon asnjë cikël.

```
for (inicializimi; kusht; azhurimi){
        urdhër A;
        urdhër B;
          ...
        urdhër R;
}
```

Figura 2.3.8

Pjesa *azhurim* e urdhrit kontrollues for realizohet pas realizimit të çdo cikli, por më së shpeshti te ai azhurohet numëruesi.

Forma e urdhrit kontrollues for për strukturën kontrolluese **për–deri–hap** është dhënë te **figura 2.3.9:**

```
for (numërues = fillestar; numërues <= fundit; numërues = numërues + sasia) {
                    urdhër A;
                    urdhër B;
                    ...
                    urdhër L;
}
```

Figura 2.3.9

Te pjesa për inicializim, ndryshorja numërues inicializohet në vlerën fillestare. Ajo pjesë realizohet vetëm njëherë. Shqyrtimi i kushtit se vlera e numëruesit a është më e vogël prej ose e barabartë me vlerën e fundit (e fundit) (numërues <= fundit), është përmirësuar në vet urdhrin. Kushti (kusht) numërues <= fundit shqyrtohet para fillimit të çdo cikli edhe nëse vlera true, cikli realizohet, por nëse vlera false, cikli nuk realizohet.

Programi për detyrë për gjetjen e 10 numrave të parë natyror, sipas algoritmit prej **figura 2.3.7**, me shfrytëzimin e urdhrit for, është dhënë te **figura 2.3.10.**

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {     // Shuma e 10 numrave të parë natyror me
5
6       int numërues, numër, shuma;
7     shuma = 0;
8     numër = 1;
9     for(numërues = 1;numërues<= 10;numërues =numërues+ 1 ) {
10        shuma = numër + numër;
11        numër =numër+ 1;
12      }
13      cout <<   "Shuma e 10 numrave të parë natyror është"    <<shumë<< endl;
14
15      cout << endl;
16      system( "Color 17" );
17      system( "pause" );
18      return 0;
19  }
```

Figura 2.3.1.0

```
Shuma e 10 numrave të parë natyror është 55
Press any key to continue . . .
```

Shembuj

Shembulli 2.3.1

Të gjendet shuma e 10 numrave të parë natyror, me mbledhje prej 10 deri 1, d.m.th., 10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1.

Programi është dhënë te **figura 2.3.11**. Vërejmë se program është i njëjtë me atë te **figura 2.3.10**, me atë dallim që ndryshorja numër inicializohet në 10, numërues numërues inicializohet në 10, kushti është ndryshuar në numërues >= 1 edhe në pjesën për azhurim të uerdhërit for, numëruesi zvogëlohet për 1.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {     // Shuma e 10 numrave të parë natyror me
5
6       int numërues, numër, shuma;
7      shuma = 0;
8      numër = 10;
9       for( numërues  = 10; numërues >= 1; numerues = numërues  - 1 ) {
10                 += numërues;
11     numërues += 1;
12      }
13      cout <<"Shuma e 10 numrave të parë natyror është"     << shuma << endl;
14
15      cout << endl;
16      system( "Color 17" );
17      system( "pause" );
18      return 0;
19  }
```

Figura 2.3.11

Shembulli 2.3.2

Të shtypet tabela për shumëzim deri në 10 me numër të futur.

Programi është dhënë te ***figura 2.3.12***. Te ai është kyçur biblioteka <iomanip> prej të cilës shfrytëzohet funksioni setw(), me të cilën jepet numri i vendeve te të cilat shtypet ndryshorja vijuese dhe funksioni right, me të cilën vlera që shtypet mbështetet në anën e djathtë.

Për shembull, nëse vlera e x është 1 234 me urdhrin:

```
cout << setw(10 << right << x << endl;
```

vlera e x do të shtypet djathtas të mbështetur në fushën prej 10 vendeve:

␣␣␣␣␣␣123.

```
1   #include <iostream>
2   #include <iomanip>
3   using namespace std;
4
5   int main() {     //Tabela e shumëzimit me
6
7       int numërues, numër;
8       cout <<"Tabela e shumëzimit me (1,2,...,9): ";
9       cin >>numër;
10      for(numërues = 1;numërues <= 10; numërues= numërues + 1 ) {
11          cout << "\n" << setw( 3 ) << right <<numërues<< " x " <<numër<< " = "
12              << setw( 3 ) <<numërues * numër;
13      }
14
15      cout << endl;
16      system( "Color 17" );
17      system( "pause" );
18      return 0;
19  }
```

Figura 2.3.12

```
Tabela e shumëzimit me (1,2,...,9): 7

  1 x 7 =   7
  2 x 7 =  14
  3 x 7 =  21
  4 x 7 =  28
  5 x 7 =  35
  6 x 7 =  42
  7 x 7 =  49
  8 x 7 =  56
  9 x 7 =  63
 10 x 7 =  70
Press any key to continue . . .
```

Shembulli 2.3.3

Të shtypen shkronjat prej G deri g pa shenjat ndërmjet shkronjave të mëdha dhe të vogla.

Deklarimi me inicializim të numëruesit shkronja, mund të bëhet edhe në pjesën për inicializim të urdhrit for, sikurse është në programin vijues.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {     // Shtypja e shkronjave prej G deri g
5
6       for( char shkronja = 'G';shkronja <= 'g';shkronja : shkronja + 1 ) {
7           cout <<shkronja << " ";
8       }
9
```

Figura 2.3.13.

```
10      cout << endl;
11      system( "Color 17" );
12      system( "pause" );
13      return 0;
14  }
```

Figura 2.3.13 (vazhdim)

```
g f e d c b a Z Y X W V U T S R Q P O N M L K J I H G
Press any key to continue . . .
```

Shembulli 2.3.4

Të tabelohet funksioni $f(x) = 3x^2 - 2x + 4$ për x [–30,30] me sasinë e hapit 4, sikurse është treguar te dalja e fituar. Programi është dhënë te figura 2.3.14.

```
1   #include <iostream>
2   #include <iomanip>
3   using namespace std;
4
5   int main() {    // Tabelimi i funksionit me
6
7       double x, y;
8       cout << setw( 7 ) << "x" << setw( 15 ) << "y=3x^2-2x+4" << endl << endl;
9       for( int numërues = -30; numërues <= 30; numërues  += 4 ) {
10          x = numërues;
11          y = 3 * x * x - 2 * x + 4;
12          cout << setw( 10 ) << x << setw( 10 ) << y << endl;
13      }
14
15      cout << endl;
16      system( "Color 17" );
17      system( "pause" );
18      return 0;
19  }
```

Figura 2.3.14

Dalja prej programit është

```
         x     y=3x^2-2x+4

       -30            2764
       -26            2084
       -22            1500
       -18            1012
       -14             620
       -10             324
        -6             124
        -2              20
         2              12
         6             100
        10             284
        14             564
        18             940
        22            1412
        26            1980
        30            2644
```

Operatorët për inkrementim dhe dekrementim ++ , —

Në C++ shfrytëzohen operatorët për zmadhimin ose për zvogëlimin e vlerës së ndonjë ndryshore për 1. Ato quhen **operator për inkrementim** (angl. increment operator)dhe **operator për dekrementim** (angl. decrement operator). Operatori për inkrementim është ++, por për dekrementim është – –. Inkrementimi ose dekrementimi i vlerës së ndryshores x mund të kryhet në formën prefikse (++x ose – –x) ose posfikse (x++ ose x– –). Në rastin e parë, inkrementimi ose dekrementimi realizohet para, por në rastin e dytë sipas njehsimit të shprehjes (postinkrement).

Operatori	Forma	Inkrement	Dekrement
++	++x	prefiks	
	x++	postfiks	
– –	– –x		prefiks
	x– –		postfiks

Shembulli

Vlera e x para njehsimit njehsimit	Shprehja postfiks	Vlera e shprehje	Vlera e x pas njehsimit
5	++x	6	6
5	x++	5	6
5	– –x	4	4
5	x– –	5	4

Inkrementimi dhee dekrementimi mund të shfrytëzohet si urdhër.

Shembuj

Shembulli 2.3.5

Shuma e 10 numrave të parë natyror prej shembujve paraprak mund të njehsohet me këtë segment programor:

```
int numër  = 1;
int shuma  = 0;
for (int numërues = 1; numërues <= 10; numërues ++) {
       shuma  += numër;
       ++ numër      // njëjtë me numër = numër + 1; ose numër += 1;
}
```

Shembulli 2.3.6

Segmenti programor vijues ilustron përdorimin e operatorit për inkrementim:

```
int a, b, c;
a = b = c = 1;
cout << "a = " << a << ", b = " << b << ", c = " << c << endl;
b = ++a;
cout << "b = ++a \na = " << a << ", b = " << b << ", c = " << c << endl;
c = a++;
cout << "c = a++ \na = " << a << ", b = " << b << ", c = " << c << endl;
c = ++ ++b;
cout << "c = ++ ++b \na = " << a << ", b = " << b << ", c = " << c << endl;
```

Dalja e tij është:

```
a = 1, b = 1, c = 1
b = ++a
a = 2, b = 2, c = 1
c = a++
a = 3, b = 2, c = 2
c = ++ ++b
a = 3, b = 4, c = 4

Press any key to continue . . .
```

Operatorët për inkrementim dhe për dekrementim mund të shfrytëzohen vetëm për bashkësi lineare të renditura, sikurse janë shenjat ASCII dhe numrat e plotë. (Bashkësi lineare të renditura janë ato te të cilat çdo element ka paraardhësin e tij (përveç të parit) dhe pasardhsin e vet (përveç të fundit)).

Te shembujt paraprak, në vend të urdhërave

```
numërues = numërues + 1
numërues = numërues - 1;
```

mund të shfrytëzohen urdhërat

```
numërues ++;   ose    ++ numërues;
numërues --;   ose    -- numërues
```

por në vend të urdhërave

```
shkronja = shkronja + 1;
shkronja = shkronja - 1;
```

mund të shfrytëzohen urdhërat

```
shkronja ++;  ose     ++ shkronja;
shkronja --;  ose     -- shkronja;
```

Shfrytëzimi i numëruesit te trupi i urdhrit kontrollues for

Te program për shumën e 10 numrave të parë natyror prej 1 deri 10, mund të vërehet se ndryshorja numër fton vlerë fillestare 1 dhe inkrementohet për 1 në çdo cikël. Gjithashtu, edhe numëruesi (numërues) inicializohet në 1 dhe në çdo cikël inkrementohet për 1. Nëse i shtypim vlerat e tyre në çdo cikël, do të vërejmë se janë të njëjtë.

```
numërues, numër
 1       1
 2       2
 3       3
 4       4
 5       5
 6       6
 7       7
 8       8
 9       9
 10      10
Shuma e 10 numrave të parë natyror është 55
```

Në rastet e këtilla, ndryshorja numërues mund të shfrytëzohet në vend të ndryshores numër. Megjithatë, fillestarët e programimit duhet të jenë shumë të kujdesshëm gjatë shfrytëzimit të numëruesit te trupi i urdhrit kontrollues për përsëritje pasi lehtë bëhen gabime. Edhe, akoma diçka. Numëruesi nuk mund gjithmonë të shfrytëzohet në trupin e urdhërave kontrolluese për përsëritje.

Programi për shumën e 10 numrave të parë natyror, me shfrytëzimi e numëruesit te trupi i urdhrit for, do të jepet si te ***figura 2.3.15*.**

Gjithashtu, te ai është treguar se deklarimi i numëruesit mund të kryhet edhe te pjesa për inicializim.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {    // Shuma e 10 numrave të parë natyror me
5
6       int shuma  = 0;
7       for( int numërues = 1;numërues= 10; numërues ++ ) {
8         shuma =shuma+ numërues;
9       }
10      cout << "Shuma e 10 numrave të parë natyror është" <<shuma<< endl;
11
```

Figura 2.3.15

```
12      cout << endl;
13      system( "Color 17" );
14      system( "pause" );
15      return 0;
16  }
```

Figura 2.3.15 (vazhdim)

Specifikat e urdhrit kontrollues for

a) Inicializimi dhe azhurimi i më shumë ndryshoreve

Urdhri kontrollues **for** në C++ lejon inicializimin dhe azhurimin e më shumë ndryshoreve. Poashtu atë ndahen me presje.

Shembulli 2.3.7

Të kontrollohet se shuma e nurave prej 1 deri n dhe prej n deri 1 a është i njëjtë?

Te programi ***(figura 2.3.1***6), te pjesa për inicializim edhe te pjesa për azhurim të urdhrit for, inicializohet dhe azhurohen dy ndryshore.

Kushti për ndërprerje të përsëritjeve mund të jetë çfarëdo shprehje logjike ku mund të shfrytëzohen operator të relacionit dhe logjik. Në këtë program kushti është i <= n && j >= 1.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       //     Se shuma 1+2+...+n a  është e njëjtë me shumën     n+(n-1)+(n-2)+...+2+1?  me (for)
6
7       int n, shuma1deriN   = 0, shumaNderi1  = 0;
8       cout <<Deri te cili numër n=";
9       cin >> n;
10      for( int i = 1, j = n; i <= n && j >= 1; i++, j-- ) {
11       shuma1deriN += i;
12       shumaNderi1 += j;
13      }
14      cout <<   "Shuma e numrave natyrorë prej 1 deri"   << n << " është  " << shuma1deriN << endl;
15      cout <<    "Shuma e numrave natyrorë prej "  << n << "  deri 1 është     " << shumaNderi1  << endl;
16
17      cout << endl;
18      system( "Color 17" );
19      system( "pause" );
20      return 0;
21  }
```

Figura 2.3.16

Një dalje prej realizimit të programit është:

```
Deri te cili numër n=100
Shuma e numrave natyrorë prej 1 deri 100 është 5050
Shuma e numrave natyrorë prej 100 deri 1 është 5050

Press any key to continue . . .
```

b) **Pjesa e inicializimit dhe azhurimit mund të jetë edhe i zbrazët**

Shembulli 2.3.8

Programi prej **Shembulli 2.3.7** mund të shkruhet edhe si te ***figura 2.3.17***. Vërejmë se inicializimi kryhet para fillimit të urdhrit for, por azhurimi kryhet te trupi i urdhrit for.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       //      Shuma 1+2+...+n a është e njëjtë me shumën n+(n-1)+(n-2)+...+2+1? (me for)
6
7       int i, j, n, shuma1deriN = 0, shumaNderi1 = 0;
8       cout << " Deri te cili numër n=";
9       cin >> n;
10      i = 1; j = n;
11      for( ; i <= n && j >= 1;) {
12       shuma1deriN += i;
13       shumaNderi1 += j;
14          i++;
15          j--;
16      }
17      cout << "  Shuma e numrave natyrorë prej 1 deri  " << n << " është" << shuma1deriN  << endl;
18      cout << "  Shuma e numrave natyrorë prej  " << n << "  deri 1 është  " << shumaNderi1 << endl;
19
20      cout << endl;
21      system( "Color 17" );
22      system( "pause" );
23      return 0;
24  }
```

Figura 2.3.17

```
Deri te cili numër n=99
Shuma e numrave natyrorë prej 1 deri 99 është 4950
Shuma e numrave natyrorë prej 99 deri 1 është 4950

Press any key to continue . . .
```

Ushtrime

Ushtrimi 2.3.1

Çka bën ky segment programor?

```
shuma = 0;
for(int numër = 2, numër <= 10; numër += 2)
	shuma += numër;
```

Ushtrimi 2.3.2

Të konstatohet çka do të shtypet si rezultat i realizimit të segmentit programor vijues:

```
for(int x = 1, i = 0; x <= 10; x++)
	i += x;
cout << x << endl;
```

Ushtrimi 2.3.3

Të konstatohet çka do të shtypet si rezultat i realizimit të segmentit programor vijues:

```
for(int i = 0, j = 0; i < 5; i++, j--)
	cout << i + j << endl;
```

Detyra të zgjidhura

Detyra 2.3.1

Të gjenden të gjitha Numrat e Pitagorës më të vegjël prej numrit natyror n.

Sqarim: Numrat e Pitagorës janë çdo treshe e numrave natyrorë x, y dhe z që e kënaqin barazimin $x^2 + y^2 = z^2$. Për të mos përsëriten treshat, sikurse 3, 4, 5 dhe 4, 3, 5, ndryshorja x do të ndryshojë prej 1 deri te vlera jo më e madhe prej y, d.m.th., deri (nëse x = y) $\left[\sqrt{\frac{n^2}{2}}\right]$. Për të mos përsëriten treshat, sikurse 3, 4, 5 dhe 4, 3, 5, ndryshorja x do të ndryshojë prej 1 deri te vlera jo më e madhe prej y, d.m.th., deri (nëse x = y).

Kllapat e mesme shënojnë pjesën e plotë prej numrit dhjetor. Për shembull, [3.56] = 3.

Vërejtje 1: Te program vijues (figura 2.3.18) shfrytëzohen dy ndryshore ndihmëse nd1 dhe nd2 me të cilat njehsohet vlera e $\left[\sqrt{\frac{n^2}{2}}\right]$ dhe x2 + y2. Pasi vlera $\left[\sqrt{\frac{n^2}{2}}\right]$. Pasi vlera është konstanta, nëse e lejmë te urdhri for, do të njehsohet te çdo cikël, që është e panevojshme. Gjithashtu, shprehja $x^2 + y^2$ njehsohet dy here te çdo cikël, por me futjen e nd2, vetëm njëherë. Në këtë mënyrë, koha për njehsim të shprehjes $x^2 + y^2$ shkurohet dyfish.

Vërejtje 2: Pasi rezltati prej rrënjës katrore të numrit real është numr real, port e cikli na duhet numër i plotë, me (int) sqrt(n * n / 2), kryejmë klonversion të detyrueshëm (të quajtur hedhje) të numri real në numr të plotë.

```
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
    //    Treshet e Pitgorës (me for)

    int n, x, y, z,    nd1, nd, numërues  = 0;
    cout << "Deri te cili numër natyror:";    cin >> n;
    cout << "\nTreshet e Pitagorës " << n << "janë:" << endl;
    nd1 = ( int ) sqrt( n * n / 2 );
    for( x = 1; x <= nd1 ; x++ ) {
        for( y = x + 1; y <= n; y++ ) {
             nd2 = x * x + y * y;
            z = ( int ) sqrt( nd2  );
            if( ( z <= n ) && ( nd2  == z * z ) ) {
                ++numërues;
                cout << setw( 10 ) <<numërues;<< ": " << setw( 2 ) << x << ", "
                    << setw( 2 ) << setw( 2 ) << y << ", " << setw( 2 ) << z << endl;
            }
        }
    }

    cout << endl;
    system( "Color 17" );
    system( "pause" );
    return 0;
}
```

Figura 2.3.18

Një dalje prej programit është:

```
Një dalje prej programit është: 33

Treshet e Pitagorës deri te numri 33janë:
         1:  3,  4,  5
         2:  5, 12, 13
         3:  6,  8, 10
         4:  7, 24, 25
         5:  8, 15, 17
         6:  9, 12, 15
         7: 10, 24, 26
         8: 12, 16, 20
         9: 15, 20, 25
        10: 18, 24, 30
        11: 20, 21, 29

Press any key to continue . . .
```

Detyra 2.3.2
Zë gjenden të gjithë numrat e përsosur më të vegjël se numri natyror n.

Sqarim: Numra natyrorë të cilt janë të barabartë e shumën e pjesëtuesve të tij, pa vet numrin, quhen numea të përsosur. Numrat e përsosur janë: 6, 28, 496 etj.

```
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
    // Numrat e përsosur (me for)

    int n, numër, pjesëtues, shuma;
    cout << " Deri te cili numër  n=":  cin >> n;
    cout << "\nNumrat e përsosur deri   " << n << "jane:" << endl;
    for(numër= 2; numër<= n; numër  ++ ) {
       shuma = 1;
        for(pjesëtues  = 2;pjesëtues <= numër/ 2;pjesëtues++ ) {
            if(numër %pjesëtues == 0 )
               shuma +=pjesëtues;
        }
        if(shuma== numër)
            cout <<numër<< ", ";
    }
    cout << endl;

    cout << endl;
    system( "Color 17" );
    system( "pause" );
    return 0;
}
```

Figura 2.3.19

Rezultati prej realizimit të programit është dhënë te ***figura 2.3.19*** është:

```
Deri te cili numër n =12345

Numra të përsosur deri 12345 janë:
6, 28, 496, 8128,

Press any key to continue . . .
```

Detyra për ushtrime

Për këto detyra të shkruhen programe me shfrytëzimin e urdhrit kontrollues for:

1. Të njehsohet shuma $1 + 4 + 7 + 11 + \ldots + n$.
2. Të njehsohet shuma $1^2 + 2^2 + 3^2 + \ldots + n^2$.
3. Të njehsohet shuma $1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$.
4. Të njehsohet mesi aritmetik i numrave natyror prej 1 deri n.
5. Të njehsohet $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$. Sipas përkufizimit $0! = 1$ dhe $1! = 1$. (Shënimi n! lexohet „n faktoriel“).
6. Të njehsohet prodhimi: $1 \cdot \frac{1}{2} \cdot \frac{1}{3} \cdot \ldots \cdot \frac{1}{n}$.
7. Të shtypen shkronjat prej alfabetit prej A deri te ZH dhe prej ZH deri te A.
8. Të shtypen shkronjat këto tri forma:

```
*
**
***
****
*****
******
*******
********
*********
```

```
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
6 5 4 3 2 1
7 6 5 4 3 2 1
8 7 6 5 4 3 2 1
9 8 7 6 5 4 3 2 1
```

```
                1
              1 2
            1 2 3
          1 2 3 4
        1 2 3 4 5
      1 2 3 4 5 6
    1 2 3 4 5 6 7
  1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
```

9. Të njehsohet shuma:
   $1 + (1 + 2) + (1 + 2 + 3) + (1 + 2 + 3 + 4) + \ldots + (1 + 2 + \ldots + n)$.
10. Të gjendet faktoriel i dyfishtë i numrit jonegativ n.
    $n!! = n(n-2)(n-4) \cdot \ldots \cdot 5 \cdot 3 \cdot 1$ për n tek, ,
    $n!! = n(n-2)(n-4) \cdot \ldots \cdot 6 \cdot 4 \cdot 2$ për n çift.
    Sipas përkufizimit $0!! = 1$ dhe $1!! = 1$.
11. Të futen n numrat dhe të numërohet sa prej tyre janë çift, por sa janë tek.
12. Të gjenden pjesëtuesit e numrit natyror n.

Struktura kontrolluese algoritmike për përsëritje me dalje në fillim të ciklit derisa–realizo dhe urdhri kontrollues while

Te detyra për gjetjen e shumës së 10 numrave të parë natyror prej nëntitullit paraprak, theksuam se përsëriten vetëm dy veprime:

- Shtuarja numër shumës.
- Zmadhimi i numrit për 1.

Ndryshorja numër zmadhohet në çdo cikël deri te vlera 11. Prandaj, mundemi të themi: **derisa** numri ka vlerë më të vogël ose të barabartë me 10, **realizohen** cikle me këto veprime. Struktura kontrolluese algoritmike e këtillë quhet **deri–realizo.**

```
shuma ← 0;
numër ← 1;
 deri numër ≤ 10 realizo
        – shto numër shumës;
        – zmadhonumrin për 1;
– fund_ deri {numër ≤ 10}
```

Figura 2.3.20

Ajo është paraqitur te ***figura 2.3.20.***

```
deri kusht realizo
        hapi A;
        hapi B;
          ...
        hapi M;
fund_deri {kusht}
```

Figura 2.3.21

Struktura përbëhet prei hapave A, B... M. Një realizim i hapave prej A deri M është një cikël. Para fillimit të çdo cikli, shqyrtohet a është plotësuar kushti (*kusht)*, i cili paraqet shprehje logjike me dy vlera: *saktë* (true) ose pasaktë (false). Derisa është plotësuar (saktë) kushti, realizohen hapat prej ciklit dhe veprimi kthehet në fillim të ciklit.

Kur te ndonjë shqyrtim të kushtit, do të konstatohet se ai nuk është plotësuar (nuk është i saktë), kapërcehet struktura kontrolluese dhe veprimi vazhdon me hapin vijues ose me strukturën kontrolluese algoritmike vijuese **fund_deri** {kusht}.

Pasi *kusht* shqyrtohet para fillimit të çdo cikli, mund të ndodh *kushti* të mos jetë plotësuar qysh gjatë shqyrtimit të parë dhe poashtu, ë*shtë e mundshme të mos realizohet asnjë cikël.*

Paraqitja grafike e strukturës kontrolluese algoritmike dhënë te ***deri–realizo është figura 2.3.22***.

Për realizimin e strukturës kontrolluese algoritmike pa **deri–realizo**, në C++ shfrytëzohet urdhri kontrollues while. Shënimi i tij është dhënë te ***figura 2.3.23.***

```
while(kusht)
        hapi A;
        hapi B;
        ...
        hapi M;
}
```

Figura 2.3.23

Për urdhrin kontrollues while vlen se nëse *kushti* nuk është plotësuar para ciklit të parë, atëherë urdhri while nuk realizohet asnjëherë.

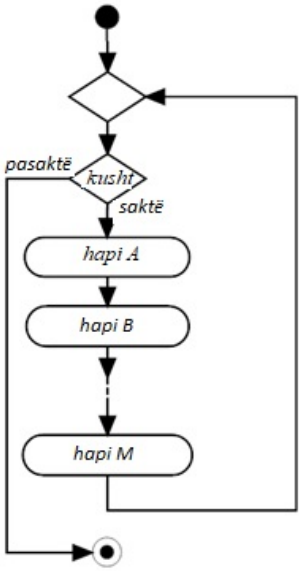


Figura 2.3.22

Shembuj

Shembulli 2.3.9

Me shfrytëzimin e strukturës kontrolluese algoritmike **deri–realizo**, algoritmi për detyrën për gjetjen e shumës së 10 numrave të parë natyror prej nëntitullit paraprak, do të jetë si te ***figura 2.3.24***, por program i shkruar me urdhrin while është dhënë te ***figura 2.3.25***.

algoritëm *ShumaDeri10*
fillimi
shuma ← 0;
numër ← 1;
deri numër ≤ 10 **realizo**
shuma← shuma + numër;
numër ← numër + 1;
fund_deri {numër ≤ 10}
shtyp shuma
fund {*ShumaDeri10*}

Figura 2.3.24

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {     // Shuma e 10 numrave të parë natyror  (me while)
5
6       int numër, shuma;
7      shuma = 0;
8      numër = 1;
9       while(numër <= 10 ) {
10         shuma =shuma+ numër;
11         numër =numër + 1;
12      }
13      cout << " Shuma e 10 numrave të parë natyror është " <<shuma<< endl;
14
15      cout << endl;
16      system( "Color 17" );
17      system( "pause" );
18      return 0;
19  }
```

Figura 2.3.25

Një dalje prej realizimit të programit është:

```
Shuma e 10 numrave të parë natyror është 55
Press any key to continue . . .
```

Shembulli 2.3.10
Të njehsohet sa para kemi shpenzuar në treg.

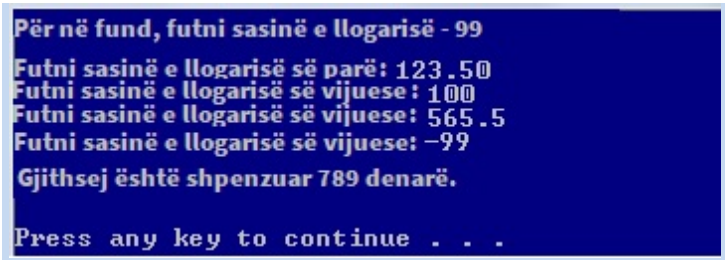
Pasi është e nevojshme të dimë sa prodhime kemi blerë, për çdo prodhim do të futim sasinë e shumës që e kemi paguar. Për në fund futja, duhet të vendosim ndonjë kontrollë. E dimë se për çdo prodhim kemi paguar shumë të caktuar, që është numër pozitiv. Prandaj, në fund të futjes do t futim ndonjë numër i cili nuk mund të paraqet sasi të paguar për ndonjë prodhim. Për shembull, për sasinë do të futim –1, –99 ose 9 999 (e vogël është gjasa të kemi paguar saktë këtë sasi për ndonjë prodhim) dhe të ngjashme. Kontrolli i këtillë i përsëritjes me ndonjë vlerë përpara të njohur quhet *përsëritje kontrollim me roje* (angl. Sentinel-Controlled Repetition).

Program ii shprehr me urdhrin while, është dhënë te ***figura 2.3.26.***

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // Treg  (me while)
6
7       double  sasiELlogarisë, gjithsejShpenzim   = 0:
8       cout << " Për në fund, fut sasinë e llogarisë -99" << endl;
9       cout << "\nFutni sasinë e llogarië së parë";
10      cin >> sasiELlogarisë ;
11      while(  sasiELlogarisë != -99 ) {
12          sasiELlogarisë += gjithsejShpenzim ;
13          cout << " Futni sasinë e llogarisë vijuese ";
14          cin >> sasiELlogarisë ;
15      }
16      cout << "\nGjithsej është shpenzuăr<< gjithsejShpenzim  << " denarë " << endl;
17
18      cout << endl;
19      system( "Color 17" );
20      system( "pause" );
21      return 0;
22  }
```

Figura 2.3.26

Pas realizimit të programit, fitohet:



Ushtrime

Ushtrimi 2.3.4

Të gjendet gabimi në segmentin programor vijues:

```
float x = 5, suma = 0;
while(x >= 0)
	suma = suma + x;
```

Ushtrimi 2.3.5

Të konstatohet çka do të shtypet pas realizimit të segmentit programor vijues:

```
int i = 0;
while(i < 5) {
	i = i + 2;
	cout << i << "\n";
}
```

Ushtrimi 2.3.7

Çka shtyp program vijues?

```
#include <iostream>
using namespace std;

int main() {
	int n, s = 0, i = 1;
	double x;
	cout << "Futni numrin natyror:"; cin >> n;
	while(i <= n) {
		cout << "Futni numrin dhjetor:"; cin >> x;
		s += x;
		i++;
	}
	cout << "s = " << s << endl;
	return 0;
}
```

Detyra të zgjidhura

Detyra 2.3.3
Të konstatohet sa shifra ka ndonjë numër natyror.

Sqarim: Le të jetë dhënë numri n = 174.
Nëse n (= 174) e pjesëtojmë me 1, herësi do të jetë 17, por mbetja 4.
Herësin do t'ia shoqërojmë te n (n = 17).
Nëse n (= 17) e pjesëtojmë me 1, herësi do të jetë 1, por mbetja 7.
Herësin do t'ia shoqërojmë te n (n = 1).
Nëse n (= 1) e pjesëtojmë me 1, herësi do të jetë , por mbetja 1.
Herësin do t'ia shoqërojmë te n (n =).
Vërejmë se mund të formojmë cikël me dy veprime:

- Pjesëtimi i n me 10.
- Shoqërimi i herësit të fituar te n.

Përsëritje ndërpritet kur mbetja prej pjesëtimit do të jetë 0 (n = 0).
Me këtë mënyrë humbet vler fillestare (e future) e n.
Nëse është nevojshme edhe pas realzimit të algoritmit të shfrytëzohet vlera fillestare e n, është e nevojshme pas futjes të mbahet mend te ndryshorja tjetër.

```
#include <iostream>
using namespace std;

int main() {
    // Numri i shifrave të numrit të plotë (me while)

    int numriPlotë, numriShifra, herës;
    cout << "Futni numër të plotë, n=";
    cin >> numriPlotë;
    cout << "Numrit të plotë" <<numriPlotë<< " ka ";
        herës = numriPlotë / 10;
    numriShifra = 1;
    while( herës != 0 ) {
      numriPlotë = herës ;
           herës = numriPlotë / 10;
       numriShifra ++;
    }
    cout << numriShifra << " shifra." << endl;

    cout << endl;
    system( "Color 17" );
    system( "pause" );
    return 0;
}
```

Figura 2.3.27

Pas realizimit të programit, fitohet:

```
Futni numër natyror   n=2123456789
Numri natyror 2123456789 ka 10 shifra

Press any key to continue . . .
```

Detyra 2.3.4

Të gjendet pjesëtuesi më i madh i përbashkët për dy numra natyrorë.

Sqarim: Pjesëtuesi më i madhi përbashkët (PMP) për dy numra natyrorë mund të jetë më i vogli prej tyre nëse numrat plotpjesëtohen me pjesëtuesin e përbashkët (PMP) për dy numra natyrorë mund të jetë më i vogli prej tyre nëse numrat plotpjesëtohen njëri me tjetrin ose numri i parë më i vogël prej më të voglit, me të cilin plotpjesëtohen të dy numrat. Prandaj, supozojmë se PMP është më i vogli prej tyre, por nëse nuk është, atëherë e zvogëlojmë PMP për 1 në çdo cikël deri sa nuk gjejmë numër me të cilin plotpjesëtohen të dy numrat e dhënë.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // PMP u dy nurave    (me while)
6
7       int a, b,pmp;
8       cout << "Futni dy numra natyrorë" << endl;
9       cout << "a = "; cin >> a;
10      cout << "b = "; cin >> b;
11      if( a < b )
12          pmp=a ;
13      else
14          pmp=b ;
15      while( ( a % pmp != 0 ) || ( b % pmp != 0 ) )
16        pmp --;
17      cout <<"Pjesëtuesi më i madhi përbashkët për numrat"
18          << a << " i " << b << " e " << pmp<< endl;
19
20      cout << endl;
21      system( "Color 17" );
22      system( "pause" );
23      return 0;
24  }
```

Figura 2.3.28

Shembull i daljes pas realizimit të programit është:

```
Futni dy numra natyrorë
a = 1248
b = 2364
Pjesëtuesi më i madhi përbashkët për numrat 1248 dhe 2364 është e 12

Press any key to continue . . .
```

Detyra 2.3.5
Të qëllohet numri natyror që e ka menduar kompjuteri.

Në fillim prej programit gjenerohet numër i rastit me funksionin rand(). (Ky funksion është përpunuar te nëntitulli **Funksionet e bibliotekës** prej nënpikës **2.5 Funksione**). Me rand() gjenerohet numër i rastit prej 0 deri 32 767. Me rand() % 100 bashkësia e numrave të rastit sillet në intervalin prej 0 deri 99, por me 1 + rand() % 100, sillet në bashkësinë prej 1 deri 100.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {     //Numri i menduar (me while)
5
6       int numri, numrilMenduar,   përpjekje;
7     numrilMenduar  = 1 + rand() % 100;
8       cout <<"Qëlloni cilin numër e kam menduar ndërmjet 1 dhe 100" ;
9       cin >> numër ;
10    përpjekje = 1;
11      while(numër !=numrilMenduar ) {
12          if(numër >numrilMenduar )
13              cout <<"Numri i menduar është më i vogël se" <<numri;
14          else
15              cout <<"Numri i menduar është më i madh se" <<numri;
16          cout << "\nQëlloni përsëri ";
17          cin >> numër;
18       përpjekje ++;
19      }
20      cout << "Bravo! E qëlluat në" << përpjekje   << "përpjekje"   << endl;
21
22      cout << endl;
23      system( "Color 17" );
24      system( "pause" );
25      return 0;
26  }
```

Figura 2.3.29

```
Qëlloni cilin numër e kam menduar ndërmjet 1 dhe 100:50
Numri i menduar është më i vogël se 50
Qëlloni përsëri:  25
Numri i menduar është më i madh se  25
Qëlloni përsëri: 37
Numri i menduar është më i vogël se 37
Qëlloni përsëri:30
Numri i menduar është më i vogël se 30
Qëlloni përsëri: 26
Numri i menduar është më i madh se 26
Qëlloni përsëri:28
Bravo! E qëlluat në përpjekje  e 6-të.

Press any key to continue . . .
```

Detyra për ushtrime

Për këto detyra të shkruhen programe me shfrytëzimin e urdhrit kontrollues while:

1. Të njehsohet prodhimi i numrave të plotë të future nëpërmjet tastaturës deri sa nuk futet numri 0.
2. Të njehsohet shuma e numrave prej a deri te b me hap c = (b – a) / n, d.m.th., vlera e shprehjes a + (a + c) + (a + 2c) + (a + 3c) +...
3. Të njehsohet shuma e shifrave të numrit natyror n.
4. Të gjendet shumëfishi më ivogël i përbashkët (SHVP) për dy numra natyrorë.
5. Të kontrollohet ndonjë numër natyror a është palindrom. (Palindrom është ai numër natyror që ka vlerë nëse lexohet prej prapa, sikurse edhe prej përpara. Për shembull, numri 123 454 321 është palindrom).
6. Të kontrollohet numri natyror n a është numër i thjeshtë ose nuk është. (Numra të thjeshtë janë ato numra të cilët plotpjesëtohen vetëm me 1 dhe me vetveten).
7. Të gjenden të gjithë numrat miqësor deri te 10 000. (Dy numra janë miqësor nëse shuma e pjesëtuesve të njërit është i barabartë me tjetrin dhe shuma e pjesëtuesve të tjetrit është i barabartë me të parin. Te pjesëtuesit e numrit nuk bën pjesë vet numri).
8. Të shndërrohet numri jonegativ i plotë dekad n në numër binar.
9. Të shndërrohet numri jonegativ i plotë bina n në numër dekad.
10. Të njehsohet vlera e numrit e = 2.718281 me saktësi të dhënë ε (lexohet „epsilon“),nëpërmjet shprehjes $e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \dots$ Saktësia ε arrihet kur mbledhësi i fundit (i cili nuk i shtohet shumës). $\frac{1}{k!} < \varepsilon$ Numri ε është vlerë shumë e vogël, për shembull, ε = 0.001, ε = 0.0000000001 etj.
11. Të njehsohet vlera e numrit π = 3.14159 me saktësi të dhënë ε, nëpërmjet shprehjes ,$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} \cdots$,përkatësisht deri sa $\frac{4}{2k+1} < \varepsilon$.
12. Të shtypen këto forma:

```
@@@@@@@@@@      &&&&&&&&&&      $
@@@@@@@@@       &&&&&&&&       ###
@@@@@@@@        &&&&&&        $$$$$
@@@@@@@         &&&&         #######
@@@@@@          &&          $$$$$$$$$
@@@@@           &&          ##########
@@@@            &&&&        $$$$$$$$
@@@             &&&&&&      #######
@@              &&&&&&&&    $$$$$
@               &&&&&&&&&&  ###
                            $
```

Struktura kontrolluese algoritmike për përsëritje me dalje në fund prej ciklit realizo–deri dhe urdhri kontrollues do-while

Te detyra për gjetjen e shumës së 10 numrave të parë natyror prej nëntitullit paraprak, theksum se përsëriten vetë dy veprime:

- Shtuarja e numrit shumës.
- Zmadhimi i numrit për 1.

Mund të themi se cikli i këtyre dy veprimeve janë realizojnë derisa numri ka vlerë më të vogël ose të barabartë me 10.

Prandaj, kjo strukture kontrolluese quhet **realizo-deri**. Këtë mundemi ta paraqesim si te ***figura 2.3.30***.

```
shumë ← 0;
numër ←
realizo
        – shto numër shumës;
        – zmadho numrin për 1;
deri numër ≤ 10;
fund_ deri  {broj ≤ 10}
shtyp  shumë;
```

Figura 2.3.30

Në këtë strukturë kontrolluese algoritmike, kushti se ciklivijues a do të realizojë ose jo, shqyrtohet në fund sipas realizimit të çdo cikli.

Kjo do të thotë se patjetër të realizohet të paktën një cikël Shënimi tekstual i strukturës kontrolluese realizo-derisa dhe paraqitja grafike është dhënë te ***figura 2.3.31***, dhe ***figura 2.3.32***.

```
hapi A;
hapi B;
hapi M;
derisa (kushti);
fundi_realizo {kushti}
```

Figura 2.3.31

Urdhri kontrollues do-while shfrytëzohet kur ndonjë grup urdhëtash duhet të kryejë të paktën njëherë, por realizimi përsëri varet prej ndonjë kushti.

Realizimi i ciklit përsëritet derisa është plotësuar *kushti*. Kur *kushti* nuk do të jetë plotësuar, përsëritja mbaron dhe veprimi vazhdon me urdhrin vijues.

```
deri{
urdhri A;
urdhri B;
urdhri T;
}while(kushti);
```

Figura 2.3.33

Shembuj

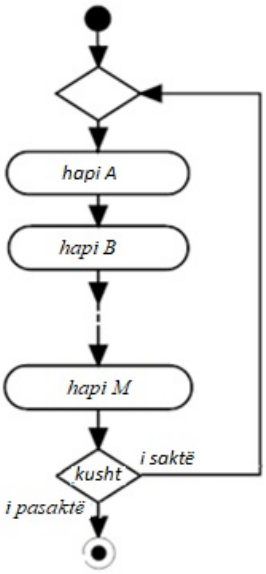


Figura 2.3.32

Algoritmi për detyrën për shumën e 10 numrave të parë natyror, të shprehur tekstualisht me strukturën kontrolluese algoritmike **realizo–derisa** është dhënë te ***figura 2.3.34*****,** por program është shkruar me urdhrin kontrollues do-while është dhënë te ***figura 2.3.35.***

algoritëm Shuma
fillimi
shuma ← 0;
numër ← 1;
realizo
shuma← shuma + numër;
numër ← numër + 1;
deri numër ≤ 10;
fund_realizo {numër ≤ 10}
shtyp shuma;
fund { *Shuma* }

Figura 2.3.34

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // Shuma e 10 numrave të parë natyror    (me do-while)
6
7       int numër, shuma;
8     shuma = 0;
9     numër = 1;
10      te {
11          shuma=shuma+numër;
12          Numër=numër+1
13      } while(numër <= 10 );
14      cout << "Shuma e 10 numrave të parë natyror është"    << shuma << endl;
15
16      cout << endl;
17      system( "Color 17" );
18      system( "pause" );
19      return 0;
20  }
```

Figura 2.3.35

Një dalje pas realizimit të programit është:

```
Shuma e 10 numrave të parë natyror është 55
Press any key to continue . . .
```

Shembulli 2.3.12

Të njehsohet sa para kemi shpenzuar në treg.

Sqarim: Programit prej **Shembulli 2.3.10** më e natyrshme është ta shkruajmë me urdhrin kontrollues do-while pasi njehsimi për shepnzim në treg do të bëjmë edhe nëse kemi blerë të paktën një prodhim d.m.th., nëse kemi të paktën një llogari. Domethënë, patjetër të realizohet të paktën një cikël.

Programi është dhënë te ***figura 2.3.36.***

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       //  Treg (me do-while)
6
7       double sasiELlogaris, gjithsejShpenzim;
8       cout <<"Për në fund, futni sasinë e llogarisë -99" << endl;
9       cout <<"Futni sasinë e llogarisë së parë:"              ;
10      cin >>  sasiELlogaris ;
11      do {
12          gjithsejShpenzim  += sasiELlogaris ;
13          cout <<"Futni sasinë e llogarisë vijuese";
14          cin >> sasiELlogaris   ;
15      } while(    sasiELlogaris != -99 );
16      cout << "\nGjithsej është shpenzuar"<< gjithsejShpenzim   << "  denar ." << endl;
17
18      cout << endl;
19      system( "Color 17" );
20      system( "pause" );
21      return 0;
22  }
```

Figura 2.3.36

Një dalje prej realizimit të programit është:

```
Për në fund. Futni sasinë e llogarisë  -99
Futni sasinë e llogarisë së parë:12
Futni sasinë e llogarisë vijuese:34.5
Futni sasinë e llogarisë vijuese: 56.789
Futni sasinë e llogarisë vijuese: -99

Gjithsej janë shpenzuar 103,289 denarë.

Press any key to continue . . .
```

Ushtrime

Ushtrimi 2.3.8

Të konstatohet çka punon segmenti programor vijues:

```
int x = 0, i = 0;
do {
	x++;
	i += ++x;
} while(x < 10);
```

Ushtrimi 2.3.9

Të konstatohet çka do të shtyp pas realizimit të segmentit programor vijues:

```
int i = 0;
do {
	i += i++;
	cout << i << ", ";
} while(i < 10);
```

Ushtrimi 2.3.10

Të konstatohet çka do të shtyp pas realizimit të segmentit programor vijues:

```
int i = 12345;
do {
	cout << i % 10 << '\n';
	i /= 10;
} while(i > 0);
```

Detyra të zgjidhura

Detyra 2.3.6

Të konstatohet sa shifra ka ndonjë numër natyror.

Sqarim: Detyra është e njëjtë me **Detyrën 2.3.3**. Megjithatë, më e natyrshme është ta shprehim me urdhrin kontrollues do-while pasi edhe të jetë numër njëshifror, patjetër të realizojmë të paktën një pjesëtim me 10 për të shikuar mbetja a është 0.

Programi, i shkruar me urdhrin do-while, është dhënë te ***figura 2.3.37.***

Një dalje prej realizimit të programit është:

```
Futni numër natyror n=1234567890
Numri natyror 1234567890 ka 10 shifra.

Press any key to continue . . .
```

```
#include <iostream>
using namespace std;

int main() {
    // Numri i shfrave të numrit natyror (me do-while)

    int numërNatyror, numërShifrash, herësy;
    cout << "Futni numër natyror  n=";
    cin >> numërNatyror;
    cout << "Numri natyror" <<numërNatyror  << " ka ";
numërShifrash = 0;
    do {
           herës = numërNatyror / 10;
    numërShifrash ++;
        numërNatyror = herës ;
    } while(herës != 0 );
    cout <<numërShifrash << "shifra"  << endl;

    cout << endl;
    system( "Color 17" );
    system( "pause" );
    return 0;
}
```

Figura 2.3.37

Detyra 2.3.7

Të njehsohet vlera e numrit të Ojlerit5 (Leonhard Euler, matematikan zvieran) e = 2.718281... nëpërmjet shprehjes

$$e \approx 1+ \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \ldots + \frac{1}{n!}$$

me saktësi ε.

Sqarim: Thamë se prodhimi 1 · 2 · 3 ·... · n shënohet me n! dhe quhet „n faktoriel“. Sipas përkufizimit, 0! = 1 dhe 1! = 1, por faktoriel për numra më të mëdhenj se 1 njehsohet sipas formulës n! = (n – 1)! · n.

Kështu, 2! = 1 · 2, 3! = 2! · 3 = 1 · 2 · 3 etj.

Kjo veti e faktorielëve mundemi ta shfrytëzojmë për zgjidhjen e detyrës.

Përkatësisht $\frac{1}{2!} = \frac{1}{1!} \cdot \frac{1}{2}$, $\frac{1}{3!} = \frac{1}{2!} \cdot \frac{1}{3}$, $\frac{1}{4!} = \frac{1}{3!} \cdot \frac{1}{4}$, etj., $\frac{1}{n!} = \frac{1}{(n-1)!} \cdot \frac{1}{n}$.

Kjo detyrë do të thotë se çdo mbledhës vijues (për shembull, k) fitohet

kur mbledhësi paraprak ((κ – 1) do ta shumëzojmë me $\frac{1}{k}$ ·

[5] Disa e quajnë numër të Neperit (*John Napier, shekulli XVII, matematikan skocez*).

Aktësinë ε do ta shfrytëzojmë për në fund të përsëritjes së ciklit kur mbledhësi që duhet t'i shtohet shumës do të jetë më i vogël prej saktësisë së dhënë $(\frac{1}{n!} < \varepsilon)$. Poashtu, nuk dihet prej përpara sa mbledhës duhet të mblidhen për të arritur saktësinë e dhënë.

Programi është dhënë te figura 2.3.38.

Prej vlerës së saktë te program, vërehet se për saktësinë $\varepsilon = 10^{-50}$ fitohet vlera me 15 dhjetore të sakta.

```
#include <iostream>
using namespace std;

int main() {
    // Numri i shfrave të numrit natyror (me do-while)

    int numërNatyror, numërShifrash, herësy;
    cout << "Futni numër natyror  n=";
    cin >> numërNatyror;
    cout << "Numri natyror" <<numërNatyror  << "ka ";
numërShifrash = 0;
    do {
            herës = numërNatyror / 10;
    numërShifrash ++;
        numërNatyror = herës ;
    } while(herës != 0 );
    cout <<numërShifrash << "shifra"  << endl;

    cout << endl;
    system( "Color 17" );
    system( "pause" );
    return 0;
}
```

Figura 2.3.38

Një dalje prej realizimit të programit është:

```
Futni saktë. epsilon = 1e-50
Numri është  e=2.71 me saktësi 0.000000000000000000000000000000000000000000000000001
është 2.718281828459045534884808148490265011787414550781 25

Press any key to continue . . .
```

Detyra për ushtrime

Të kontrollohet cilat **detyra prej Detyra** për ushtrime prej nëntitullit **Struktura kontrolluese algoritmike për përsëritje me dalje në fillim të ciklit dhe urdhri kontrollues** mund të shkruhen me shfrytëzimin e urdhrit kontrollues do-while.

Folezimi i urdhërave kontrolluese për përsëritje

Te disa programe është e nevojshme të përdoren më shumë udhëra kontrollues për përsëritje. Poashtu, ndonjë urdhër kontrollues mund të gjendet te tjetra. Ai quhet **folezim** (angl. nesting).

Gjatë folezimit të urdhërave kontrolluese për përsëritje, të gjitha urdhërat e një urdhri kontrollues patjetër të gjendet te tjetra, përkatësisht nuk guxon të ketë prerje (mbivendosje) të urdhërave kontrolluese, si te ***figura 2.3.39***. Te ***figura 2.3.40*** a, b dhe c janë dhënë mënyra të lejuara të folezimit të urdhërave kontrolluese.

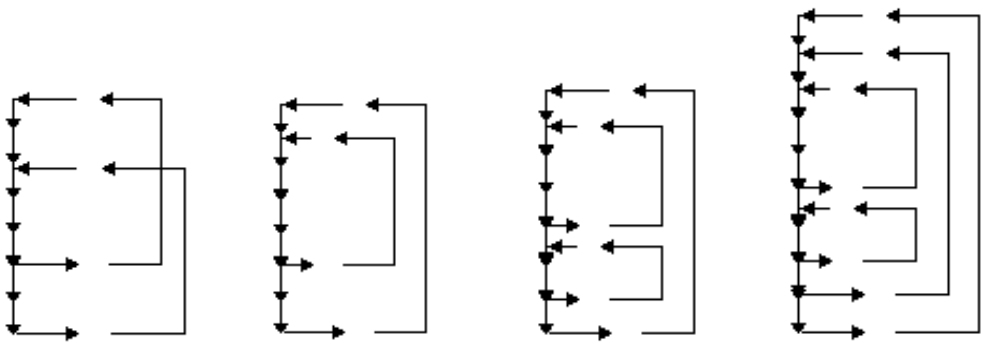


Figura 2.3.39

a ***b*** ***c***

Figura 2.3.40

Shembuj

Shembulli 2.3.13

Të shkruhet program për shumëzim tabular deri më 10 për numrat prej 1 deri 10

Programi është dhënë te ***figura 2.3.41.***

```
1   #include <iostream>
2   #include <iomanip>
3   using namespace std;
4
5   int main() {
6       // Tabela e shumëzimit deri 10, për numrat prej 1 deri 10.   (me for)
7
8       for( int i = 1; i <= 10; i++ ) {
9           cout <<"Shumëzimi me" << i << endl << endl;
10          for( int j = 1; j <= 10; j++ ) {
11              int ipoj = i * j;
12              cout << setw( 2 ) << i << " x " << setw( 2 ) << j << " = "
13                  << setw( 3 ) << ipoj << endl;
14          }
17
18      cout << endl;
19      system( "Color 17" );
20      system( "pause" );
21      return 0;
22  }
```

Figura 2.3.41

```
Shumëzimi me 1

 1 x  1 =   1
 1 x  2 =   2
 1 x  3 =   3
 1 x  4 =   4
 1 x  5 =   5
 1 x  6 =   6
 1 x  7 =   7
 1 x  8 =   8
 1 x  9 =   9
 1 x 10 =  10

Shumëzimi me 2

 2 x  1 =   2
 2 x  2 =   4
 2 x  3 =   6
 2 x  4 =   8
 2 x  5 =  10
 2 x  6 =  12
 2 x  7 =  14
 2 x  8 =  16
```

...

Shembulli 2.3.14

Të gjenden të gjithë numrat natyrorë më të vegjël se n shuma e të cilit të kubeve të shifrave është e barabartë me numrin.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // Numrat natyrorë të barabartë me kubet e shifrave të vet.
6       system( "Color 17" );
7       long long n, shifra, numër, shuma, numriN;
8       bool   Nukka = false;
9       cout <<"Deri te cilin numër" n = ";     cin >> n;
10      cout << "\nNumra natyrorë më të vegjël se" << n;
11      cout << "\ndhe të barabartë me shumën e kubeve të shifrave të vet  \n janë ";
12      for(numër = 1;numër<= n; numër ++ ) {
13        shuma = 0;
14        numriN = numër;
15          do {
16               shifra = numërN % 10;
17             shuma += ( int ) pow( shifra 3 );
18              numriN /= 10;
19          } while(numriN > 0 );
20          if(shuma ==numër) {
21              cout <<numër << ", ";
22                nukka = true;
23          }
24      }
25      if( !  nukka ) {
26          cout << "\nNuk ka numra të atillë deri"<< n << endl;
27      }
28
29      cout << endl << endl;
30      system( "Color 17" );
31      system( "pause" );
32      return 0;
33  }
```

Figura 2.3.42

Një dalje prej realizimit të programit është:

```
Deri te cili numër, n  = 1000

Numra natyrorë më të vegjël se 1000
dhe të barabartë me shumën e kubeve të shifrave të vet
janë  1, 153, 370, 371, 407,

Press any key to continue . . .
```

Pyetje për kontroll të njohurive

1. Kur shfrytëzohet struktura e kontrollit algoritmik për përsëritje?
2. Si quhet një realizim i hapave te struktura e kontrollit algoritmik për përsëritje?
3. Si ndahen struktura e kontrollit algoritmik për përsëritje, sipas mënyrës së ndërprerjes së përsëritjes?
4. Cilat struktura të kontrollit algoritmik për përsëritje me numërim të cikleve i dini?
5. Paraqitni tekstualisht strukturën e kontrollit algoritmik **për–zmadhim–deri**.
6. I cilit lloj janë ndryshoret numërues, struktura e kontrollit algoritmik fillestar dhe të fundit te struktura e kontrollit algoritmik **për–zmadhim–deri?**
7. Sa cikle do të realizohen te strukturën e kontrollit algoritmik **për–zmadhim–deri** nëse ndryshorja fillestare ka vlerë më të madhe prej ndryshores e fundit?
8. Me cilin urdhër kontrolluese në C++ realizohet struktura e kontrollit algoritmik për përsëritje me numërimin e cikleve?
9. Cilat janë pjesët e urdhrit kontrollues for? Çka bëhet te çdo pjesë?
10. Cili është veprimi i operatorëve për inkrementim dhe për dekrementim?
11. Çka do të thotë prefix, kurse çka inkrementkm të postfiksit të ndryshores?Përmend shembull.
12. Për cilat bashkësi të dhënat mund të shfrytëzohen operatorët për inkrementim dhe për dekrementim?
13. A mund numëruesi te urdhri kontrollues for të shfrytëzohet te trupi i tij?
14. Sa ndryshore mund të inicializohen te urdhri kontrollues for?
15. Numëruesi te urdhri kontrollues for a mund të inicializohet jashtë prej tij?
16. A mund pjesa për kusht te urdhri kontrollues for të jetë i zbrazët?
17. Çka punon segmenti programor vijues?

```
for(x = 1, i = 0; x <= 10; x++)
   i += x;
```

18. Çka do të shtyp gjatë realizimit segmenti programor vijues?

```
for(int i = 0, j = 0; i < 5; i++, j--)
   cout << i + j << endl;
```

19. Çka do të shtyp gjatë realizimit segmenti programor vijues?

```
for(int i = 1; i <= 5; i++)
   for(int j = 1; j <= 5; j++)
          for(int k = 1; k <= 5; k++)
                 cout << i + j + k << endl;
```

20. Çka do të shtyp gjatë realizimit segmenti programor vijues?

```
for(int i = 1; i <= 5; i++) {
   for(int j = 1; j <= i; j++) {
          for(int k = 1; k <= j; k++)
                 cout << '?';
          cout << endl;
   }
   cout << endl;
}
```

21. Paraqitni tekstualisht dhe grafikisht strukturën e kontrollit algoritmik **deri–realizo**.
22. Me cilin urdhër kontrollues në C++ realizohet struktura e kontrollit algoritmik **deri–realizo**?
23. A mund me urdhrin kontrollues while të mos realizohet asnjë cikël?
24. Shkruani sintaksën e urdhrit kontrollues while.
25. Me cilin urdhër kontrollues në C++ realizohet struktura e kontrollit algoritmik **realizo–deri**
26. Me cilin urdhër kontrollues në C++ realizohet struktura e kontrollit algoritmik **realizo–deri?**
27. Shkruani sinataksën dhe sqaroni se si punon urdhri kontrollues do-while.
28. Çka do të shtypet si rezultat i realizimit të këtij segmenti programor?

```
int i = 0;

  i = i + 2;
  cout << i << endl;
}
```

29. Çka punon ky segment programor?

```
int i, j;
i = 0;
while(i < 8) {
   j = 0;
   while(j < 8) {
          cout << setw(5) << (i + j) % 8;
          ++j;
   }
   cout << endl;
   ++i;
}
```

30. Çka punon ky segment programor?

```
int x, i;
x = 0; i = 0;
do {
   x++;
   i += x;
} while(x < 10);
```

31. Çka do të shtyp pas realizimit ky segment programor?

```
int i = 0;
do {
   ++ ++i;
   cout << i << endl;
} while(i < 10);
```

32. Çka do të shtyp pas realizimit ky segment programor?

```
for(int i = 1; i <= 5; i++)
   switch(i) {
   case 2: cout << "den, "; break;
   case 4: cout << "ste? "; break;
   case 3: cout << "kako"; break;
   case 1: cout << "Dobar"; break;
   }
```

33. Çka do të thotë folezimi i urdhërave kontrolluese? Vizatoni skema të folezimeve të urdhërave kontrolluese për përsëritje.

Detyra

1. Të gjenerohet ky trekëndësh prej n vijave, port e çdo vijë për k yllza, к = 1, 2... n. *

```
* *
* * *
* * * *
...
* * * * * * * * *
```

2. Të gjenerohet ky trekëndësh:

```
1
2 4
3 6 9
4 8 12 16
...
n ...
```

3. Të shtypet ky vizatim:

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

4. Të gjenerohen numrat e Fibonacci: 1, 1, 2, 3, 5, 8, 13, 21, 34 etj., më të vegjël se numri n.
5. Të futet numri natyror n dhe të shtypet shifër pas shifre me fjalët. Për shembull, nëse futet numri 231, të shtypet dy-tre-një.
6. Të futet numri natyror n dhe të shtypet me fjalë. Për shembull, nëse futet numri 12 345, të shtypet „dymbëdhjetë mijë e treqind e dyzet e pesë".

7. Të njehsohet shua $\frac{1}{1\cdot 2}+\frac{1}{2\cdot 3}+\frac{1}{3\cdot 4}+\cdots$ ma saktë ε, $\left(\frac{1}{k(k+1)}<\varepsilon\right)$.
8. Të futen numrat nëpërmjet tastaturës dhe të gjendet numri më i madh dhe më i vogël. Për në fund të futjes, të futet numri 999 999.
9. Të thjshtohet thyesa $\frac{a}{b}$, a dhe b janë numra natyrorë.
10. Të shtypen të gjitha numrat më të vegjël se n, të cilët janë të plotpjeëtueshëm me shumën e shifrave të tij.
11. Të gjendet numri tek më i madh pjesëtues të numrit natyror n.
12. Të anashkalohet shifra k e numrit natyror n, duke numëruar shifrat prej shifrës së njësheve.

2.4 Strukturat kontrolluese algoritmike për kapërcim dhe urdhëra kontrollues për kapërcim

Te gjuhët programore ekzistojnë tre lloje të strukturës së kontrollit algoritmik për kapërcim, edhe atë:

- *Kapërcim në fund të ciklit* – **vazhdo.**
- *Kapërcim në fund të strukturës kontrolluese* – **ndërprerje.**
- *Kapërcim në fund të algoritmit* – ***dalje***.
- *Kapërcim në çfarëdo vend* – **kapërcim,**

Urdhërat kontrolluese përkatëse për këto struktura të kontrollit algoritmik në C ++ janë:

- continue për **vazhdim**,
- break për **ndërprerje**,
- exit() për **dalje**,
- goto për **kapërcim**.

Urdhërat do t'i sqarojmë te shembujt.

Urdhri kontrollues continue

Ky urdhër kontrollues shfrytëzohet për kapërcim pa kusht në fund të ciklit. Kjo bëhet kur ndonjë urdhër prej ciklit, para se ai të mbarojë, është plotësuar kusht i caktuar she nuk është e nevojshme të realizohen urdhërat tjera prej ciklit. Me këtë dilet prej ciklit actual dhe veprimi vazhdon me ciklin përkatës.

Urdhri kontrollues **continue** shfrytëzohet për dalje prej cikit actual te urdhërat kontrollues për përsëritje while, do-while dhe for.

Shembulli 2.4.1

Të njehsohet rrënja katrore prej numrave të futur. Nëse futet numri jonegativ, të mos njehsohet rrënja katrore prej, por të vazhdohet me këtë numër vijues të futur. Për në fund të futjes, të futet numri 999 999.

Programi është dhënë te ***figura 2.4.1***. Nëse numri është negative, shtypet porosia se nuk ka rrënjë katrore prej nurit negative dhe vazhdon me ciklin vijues, d.m.th., me leximin e numrit vijues.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // Rrënja katrore prej numrit real (me continue)
6
7       double x;
8       cout <<"Për fund, futni 999999 për numër" << endl;
9       do {
10          cout <<"Futni numër real,  x=";
11          cin >> x;
12          if( x < 0 ) {
13              cout <<"Numri është negative dhe nuk ka rrënjë katrore . \n";
14              continue;
15          }
16          cout <<"Rrënja katrore prej" << x << "është " << sqrt( x ) << endl;
17
18      } while( x != 999999 );
19
20      cout << endl;
21      system( "Color 17" );
22      system( "pause" );
23      return 0;
24  }
```

Figura 2.4.1

Një dalje prej realizimit të programit është:

```
Për fund, futni 999999 për numër.
Futni numër real x=1
Rrënja katrore prej 1 është 1
Futni numër real x=2
Rrënja katrore prej 2  është 1.41421
Futni numër real x=-3
Numri është negative dhe nuk ka rrënjë katrore
Futni numër real x=1234567890
Rrënja katrore prej 1.23457e+09 është 35136.4
Futni numër real x=999999
Rrënja katrore prej 999999 është 999.999

Press any key to continue . . .
```

Urdhri kontrollues **break**

Urdhri kontrollues break shfrytëzohet te urdhërat kontrollues për përsëritje while, do-while dhe for[6], si kapërcim i pakushtëzuar në fund të urdhrit për përsëritje dhe ndërprerje të përdëritjes. Kjo bëhet kur ndonjë urdhër prej ciklit, para se ai të mbarojë, është plotësuar kusht i caktuar dhe nuk është e nevojshme të vazhdon me përsëritje ciklet, d.m.th., *ndërpritet* përsëritja. Veprimi i përsëritjes me këtë urdhër.

Nëse urdhri kontrollues **break** gjendet te ndonjë prej urdhërave kontrolluese të folezuar while, do-while dhe for, atëherë me atë dilet vetëm prej urdhrit kontrollues të folezuar te trupi i të cilit gjendet.

Ky urdhër kontrollues më së shpeshti shfrytëzohet për dalje prej urdhërave kontrollues me numër të pafundshëm të urdhërave.

Vërejtje: Përsëritjet e pafundshme mund të realizohen edhe e urdhëra tjerë.

Shembulli 2.4.2

Te shembulli vijues, përsëritja do të ndërpritet vetëm nëse e qëllojmë numrine menduar prej kompjuterit (rastësisht i gjeneruar), ndryshe do të jetë i pafundshëm, ***figura 2.4.2.***

Te program shfrytëzohet funksioni rand() për gjenerimin e numrit të rastit, i sqaruar te **Detyra 2.3.5.** Për të gjeneruar varg të ndryshëm të numrave të rastit me rand(), shfrytëzohet funksioni srand(), argument i të cilit është numër i future nëpërmjet tastaturës. Funksioni srand() është sqaruar te nëntitulli **Funksioni për gjenerim të numrave të rastësishëm** prej nënpikës **2.5 Funksione.**

Një dalje prej realizimit të programit është:

```
Futni numër natyror 12345
Qëlloni cili numër e mendova ndërmjet 1 dhe 10: 1
Qëlloni cili numër e mendova ndërmjet 1 dhe 10: 2
Qëlloni cili numër e mendova ndërmjet 1 dhe 10: 3
Qëlloni cili numër e mendova ndërmjet 1 dhe 10: 4
Qëlloni cili numër e mendova ndërmjet 1  i 10: 5
Bravo! E qëlluat numrin 5 e menduar në përpjekjen e 5-të
Press any key to continue . . .
```

[6] Urdhri kontrollues break shfrytëzohet edhe te urdhri kontrollues switch, por atje është e detyrueshme sipas përkufizimit të urdhrit switch.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {     //Të qëlluarit e numrit të menduar ndërmjet 1 dhe 10 (me break)
5
6       int numri, numrilMenduar;
7       cout <<"Futni numër natyror" ; cin >>numër;
8       srand(numër);
9      numrilMenduar = 1 + rand() % 10;
10      for( int përpjekje = 1; ;përpjekje ++ ) {
11          cout <<"Qëlloni cilin numër e mendova ndërmjet 1 dhe 10";
12          cin >>numër;
13          if(numër ==numrilMenduar ) {
14              cout <<"Bravo! E qëlluat numrin e menduar"
15                  <<numrilMenduar << " në " <<përpjekje << " përpjekje" << endl;
16              break;
17          }
18      }
19
20      cout << endl;
21      system( "Color 17" );
22      system( "pause" );
23      return 0;
24  }
```

Figura 2.4.2

Urdhri kontrollues exit()

Ndonjëherë është e nevojshme program detyrimisht të mbarojë, që arrihet me kapërcim në fund të programit. Kjo është e nevojshme kur ndonjë vend te program pritet operacion me realizimin e të cilit mund të arrihet deri në ndërprerje të programit dhe deri te mbarimi jorregullar i tij. Për shembull, nëse ekziston mundësi gjatë ndonjë njehsimi madhësia nënrrënjësore e rrënjës katrore të fitojë vlerë negative, nuk duhet lejuar njehsim të rrënjës katrore pasi do të paraqitet gabim dhe program do të ndërpritet, por duhet në mënyrë rregullare të dilet prej programit, me kapërcim te fundi i tij.

Ky urdhër kontrollues shpesh shfrytëzohet gjatë detektimit të gabimeve të cilat mund të sjellin deri te rezultatet e gabuara ose deri te ndërprerja e aplikimit, sikurse që është rasti të lexohet prej datotekës që nuk ekziston.

Për kapërcimnë fund të programit, në C++ shfrytëzohet programi exit(), që ka një argument dhe atë konstantat EXIT_SUCCESS dhe EXIT_FAILURE. Nse funksioni thirrt me argumentin e parë, domethënë se program ka mbaruar me sukses, por nëse thirret argument i dytë, domethënë se programi ka mbaruar pa suksese, d.m.th., ka ndodh gabim.

exit(EXIT_SUCCESS) – për mbarim të suksesshëm (me sukses) të programit, ose

exit(EXIT_FAILURE) – për mbarim jo rregulluar (pa sukses) të programit, d.m.th., konstanta EXIT_FAILURE tfregon se ka ndodh gabim gjatë realizimit.

Shembulli 2.4.3

Të kontrollohet a ekziston datoteka me emrin Nxënësja

Sqarim: Te program shfrytëzohet funksioni comparë() për krahasim të dy stringeve. Funksioni është sqaruar te nënpika **3.5 Stringe.**

```
#include <iostream>
#include <string>
 using namespace std;

int main() {
    //Dalje prej programit (so exit() )
    string emriDatoteka;
    while( true ) {
        cout << "Futni ermine datotekës" ;
        cin >> emriDatoteka;
        if(emriDatoteka .compare( "Nxënës " ) == 0 ) {
            cout << "Ekziston datotekë me emrin " << emriDatoteka;
            cout << ",program mund të vazhdon " << endl;
            system( "pause" );
            exit( EXIT_SUCCESS );
        }
        else {
            cout << " Nuk ka datotekë me emrin " << emriDatoteka;
            cout << ", program mbaron " << endl;
            system( "pause" );
            exit( EXIT_FAILURE );
        }
    }

    cout << endl;
    system( "Color 17" );
    system( "pause" );
    return 0;
}
```

Figura 2.4.3

Shembull për dalje prej programit është:

```
Futni emrin e datotekës: nxënës
Nuk ka datotekë me emër nxënësa, program_mbaron.
Press any key to continue . . .
```

Urdhri kontrollues goto

Urdhri kontrollues për kapërcim në çfarëdo vend goto shfrytëzohet për kapërcim prej çfarëdo vendi dhe në çfarëdo vend në program. Poashtu, vendi në të cilin kapërcen patjetë të jetë shënuar. Shënimi mund të jetë numerike ose tekstuale. Ajo shkruhet si identifikator dhe detyrimisht në fund vendose n dy pika.

Shënimi mund të jetë para çfarëdo urdhri te programi.

Ky urdhër kontrolli shfrytëzohet për programimin e pastrukturuar. Mangësia e tij është që e ndryshon rrjedhën e realizimit të programit dhe prandaj, është vështirë ajo të të analizohet dhe të kuptohet. Prandaj ky urdhër kontrollues nuk shfrytëzohet në programimin e strukturuar.

Shembulli 2.4.4

Urdhri kontrollues goto është ilustruar me program për njehsimin e rrënjës katrore te shembulli **Shembulli 2.4.1**. Programi e njehson rrënjën katrore prej numrit real dhe ndërpret me realizimin nëse futet numër negative, ***figura 2.4.4***.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // Rrënja katrore  (me goto)
6
7       double x;
8       while( true ) {
9           cout << "Futni numër real  x=";
10          cin >> x;
11          if( x < 0 )
12              goto fund ;
13          cout << "Rrënja katrore prej " << x << " është " << sqrt( x ) << endl;
14      }
15  kraj:
16      cout << " Numri është negative. Ndërpritet " << endl;
17
18      cout << endl;
19      system( "Color 17" );
20      system( "pause" );
21      return 0;
22  }
```

Figura 2.4.4

Një dalje prej realizimit të programit është:

```
Futni numër real  x=2
Rrënja katrore prej 2 është 1.41421
Futni numër real, x=-3
Numri është negativ. Ndërpritet

Press any key to continue . . .
```

Pyetje për kontrollin e njohurisë

1. Cilat struktura kontrolluese algoritmike për kapërcim i dini?
2. Me cilat urdhëra kontrolluese në C++ realizohet struktura kontrolluese algoritmike për kapërcim?
3. Cili është ndryshimi ndërmjet urdhërave kontrolluese continue dhe break?
4. Me cilin urdhër kontrollues (funksion) kapërcehet në fund të programit dhe si mund të jetë argument i tij?
5. Sqaroni pse duhet të shmangni urdhrin kontrollues goto?

2.5 Funksionet

Funksionet e bibliotekës

Është e njohur prej matematikës se **funksionet** (angl. functions) kanë vetëm **një rezultat dalës**. Për shembull:

$y = 5$, $y = 4x - 2$, $y = 2x^2 - 3x + 1$ etj.

Themi se vlera e ndryshore y varet prej argumentit x dhe atë shkruhet me $y = f(x)$.

Përveç funksioneve matematikore, ekzistojnë edhe shumë funksione të tjera të cilat shfrytëzohen në fusha të ndryshme. Për t'u lehtësuar puna e programuesve të mos shkruajnë gjithmonë programe për funksione të njëjta, për shembull për x^n, $\sqrt{x}$, e^x edhe të tjera, në çdo gjuhë programore janë shkruar programe të veçanta për funksionet më shpesh të shfrytëzuara. Ato programe ruhen në të ashtuquajturën **biblioteka programore** të gjuhës. Programuesit mund t'i shfrytëzojnë ato programe me kyçjen paraprakisht në bibliotekë në fillim të programit, me direktivën #include. Kështu, te të gjitha programet ne e kyçim bibliotekën <iostream> pasi shfrytëzuam programe prej asaj për hyrje dhe dalje të të dhënave.

Praktika e programeve prej bibliotekave të quhet funksione.
Gjuha C++ ka të ashtuquajtura ***biblioteka standard e C++*** (angl. C++ Stan-dard Library).

Funksionet prej bibliotekës të C++ quhen edhe **funksione të përmirësuara** (angl. build-in functions). Emri i tyre shkruhet me shkronja të vogla pasi emrat janë fjalë të rezervuara. Për shembull, pow(), sqrt, rand() etj.

Bibioteka standard e C++ përmban më shumë biblioteka të cilat ruhen në datotekë, të quajtur **heder-datoteka** (angl. header files), të cilat, pra, përmbajnë grupe prej funksioneve të ngjashme, sikurse:

<iostream>	Funksionet për hyrje dhe për dalje.
<iomanip>	Funksionet për formatim.
<cstdlib>	Bibliotea e përgjithshme: kontrolli i programit, numrat e rastit, klasifikimi, kërkesa etj.
<cmath>	Funksionet matematikore të përgjithshme.
<string>	Funksionet për punë me stringe.
<random>	Funksionet për gjenerim të numrave të rastit.

Për t'u shfrytëzuar ndonjë funksion prej bibliotekës së C++ te program, në fillim të programit duhet të kaçet biblioteka me direktivën #include, sikurse që e bwnim atë edhe deri tani. Poashtu, biblioteka vendohet në kllapa këndore.

Në vazhdim do të shqyrtojmë disa funksione të bibliotekës më shpesh të shfrytëzuara.

Funksionet matematikore

Të përsërisim shkurtimisht për funksionet matematikore.

Për shembull, funksioni f(x) = x është funksion linear, $f(x) = x^2$ është funksion katror, $f(x) = \sqrt{x}$ është funksion për gjetjen e rrënjës katrore prej x e.t jf. (x) paraqet rregull sipas të cilës njehsohet ndonjë vlerë e re. Për shembull, rregulla për njehsimin e katrorit të ndonjë numri është ai numër të shumëzohet me vetveten, por shkruhet me $f(x) = x \cdot x$ ose shkurtimisht x^2. Vlera e cila do të fitohet për x = 5 është 25 dhe shënohet me ndonjë numër tjetër, për shembull me y, d.m.th., y = 25.

Për çdo numër x (të quajtur argument i funksionit), fitohet vlera e y. Ajo shkruhet edhe me y = f(x) ose $y = x^2$.

Te programimi, argument dhe vlera e funksionit janë ndryshore të llojit të caktuar. Për shembull:

```
int x, y;
x = 5;
y = x * x;
ose
y = pow(x, 2);       // Funksioni për fuqizim  x^2
```

Më së shpeshti funksione matematikore në C++ janë:

pow(x, y) – Funksioni eksponencial x^y.
exp(x) – Funksioni eksponencial e^x, e = 2.7182[7].
sqrt(x) – Rrënja katrore prej x, $\sqrt{x}$.
cbrt(x) – Rrënja kubike prej x, $\sqrt[3]{x}$.
log(x) – Funksioni logaritmik ln(x), me bazë e = 2.7182.
log10(x) – Funksioni logaritmik log(x), me bazë 10.
fabs(x) – Vlera absolute prej x, |x|.
ceil(x) – Rrumbullakimi i x në numrin më të vogël real jo më i vogël se x.
floor(x) – Rrumbullakimi i x në numrin më të madh real jo më i madh se x.
trunc(x) – Prerja vetëm të pjesës reale majtas prej pikës dhjetore.
round(x) – Rrumbullakimi i numrit real më të afërm deri te x (pa dhjetore).
fmod(x, y) – Mbetja prej herësit x / y si numër real.
modf(x, &integralen) – Ndarja e numrit dhjetor x në pjesë integrale dhe dhjetor. (Për shenjën & do të flasim më vonë).
sin(x) – Funksioni trigonometrik sin(x).
cos(x) – Funksioni trigonometrik cos(x).
tan(x) – Funksioni trigonometrik tg(x).

Këto funksione i mundësojnë shfrytëzuesit të realizojë një varg njehsime matematikore. Funksionet shfrytëzohen ashtu që shkruhet emri i funksionit dhe lista me argument të vendosur në kllapa të vogla.

Te shembulli paraprak shkruajtëm

```
y = pow(x, 2);
```

Ana e djathë prej urdhrit paraqet thirrje të funksionit (angl. func-tion call ose function invocation) me emrin pow dhe me argumente x dhe 2. Me thirrjen e funksionit, realizohet programi me emrin pow(), i cili ruhet në bibliotekën <cmath> dhe rezultati i shoqërohet ndryshores y. Shpesh thuhet se **funksioni kthen vlerën.**

Me urdhrin vijues për shtypje të rrënjës katrore prej numrit 900.0, thirret funksioni me emrin sqrt dhe me argument 900.0:

```
cout << sqrt(900.0);
```

Vlerën që e kthen funksioni shtypet.

Funksionet matematikore të përmendura gjatë çdo thirrje kthejnë vlerën. Prandaj quhwn funksione me vlerë kthyese (angl. value-returning functions).

Argumentet e funksioneve mund të jenë konstante, të ndryshueshme ose shprehje të plota. Për shembull, për c = 12, d = 3.0, g = 4.0, urdhri

```
cout << sqrt(c + d * f);
```

do t'i njehson dhe shtyp rrënjën katrore 12 + 3.0 * 4.0 = 25.0, d.m.th., është 5.00.

[7] Ky numër është baza e logaritmeve natyrore, por quhet numri i Ojlerit ose Neperit.

Te shembujt vijues është ilustruar shfrytëzimi i disa prej funksioneve matematikore.

Shembulli 2.5.1

```
#include <iostream>
#include <string>
#include <iomanip>
using namespace std;

int main()
{   //Funksione matematikore pow(), sqrt(), cbrt(), exp() dhe log()

    int a, b, c;
    cout << "Futni vlerën e parë për x=";
    cin >> a;
    cout << "Futni vlerën e dytë për x=";
    cin >> b;
    cout << "Futni vlerën e tretë për  x=";
    cin >> c;
    cout << setprecision(4) << endl;
    cout << "-------------------------------------------------------------------
    cout << setw(5) << "x" << setw(7) << "paraardh." << setw(7) <<"paraardh."
        << setw(7) << "x^2" << setw(15) << "Rrënja katrore (x)" << setw(15)
        << "Rrënja kubike"<< setw(12) << "e^x" << setw(10) << "ln(x)" << endl;
    cout << "-------------------------------------------------------------------
    cout << setw(5) << a << setw(4) << a - 1 << setw(7) << a + 1 << setw(10)
        << pow(a, 2) << setw(15) << sqrt(a) << setw(15) << cbrt(a) << setw(12)
        << exp(a) << setw(10) << log(a) << endl;
    cout << setw(5) << b << setw(4) << b - 1 << setw(7) << b + 1 << setw(10)
        << pow(b, 2) << setw(15) << sqrt(b) << setw(15) << cbrt(b) << setw(12)
        << exp(b) << setw(10) << log(b) << endl;
    cout << setw(5) << c << setw(4) << c - 1 << setw(7) << c + 1 << setw(10)
        << pow(c, 2) << setw(15) << sqrt(c) << setw(15) << cbrt(c) << setw(12)
        << exp(c) << setw(10) << log(c) << endl;
    cout << endl;
    system("Color 17");
    system("pause");
    return 0;
}
```

Figura 2.5.1

Gjatë realizimit të programit fitohet kjo dalje:

x	paraardh.	pasardh.	x^2	rrënja katrore (x)	rrënja kubike (x)	e^x	ln(x)
2	1	3	4	1.414	1.26	7.389	0.6931
3	2	4	9	1.732	1.442	20.09	1.099
12	11	13	144	3.464	2.289	1.628e+05	2.485

Funksioni për gjenerimin e numrave të rastësishëm

Në programim shpesh paraqitet nevoja prej gjenerimit të numrit të rastit. Prandaj, te C++ ekziston formulë e veçantë **rand**(), e cila gjeneron numër të rastësishëm ndërmjet 0 dhe 32 767. [8]

Për shembull:

```
Numra të rastit ndërmjet 0 dhe 32767:
19169 26500 6334 18467 41
```

Nëse duam këtë bashkësi {0, 1, 2, 3... 32 767} ta sjellim në interval të caktuar [0, max] atëherë do ta shfrytëzojmë formulën:

```
rand() % (max + 1);
```

Për shembull, numra të rastit prej intervalit [0, 99]:

```
Numra të rastit ndërmjet 0 dhe 99:
64 62 58 78 24
```

Për të filluar numrat e rastit prej 1, formula është

```
1 + rand() % (max + 1);
```

Për shembull, numra të rastit prej shembullit paraprak në intervalin [1, 100] janë:

```
Numra të rastit ndërmjet 1 dhe 100:
62 28 82 46 6
```

Nëse duam intervalin e numrave të gjenerur të rastit të fillon prej numrit të caktuar (për shembull fillestar), atëherë formula është:

```
fillestar + rand() % (max + 1);
```

Për shembull, intervali [1, 100] mund të jetë [51, 150] me:

```
51 + rand() % 100;
```

```
Numra të rastit ndërmjet 1 dhe 150:
87 78 93 146 142
```

Gjatë realizimit të programit te i cili shfrytëzohet shprehja e njëjtë për gjenerimin e numrave të rastit, do të gjenerohet vargu i njëjtë i numrave të rastit. Kjo është ashtu pasi pas çdo thirrje të funksionit rand() prej *bibliotekës standard* të C++, realizohet program i njëjtë. Prandaj, numrat e rastit të gjeneruara nuk janë me të vërtet të rastësishëm, por quhen **numra pseudo të rastësishme** (angl. pseudo-random numbers).

Për të gjeneruar varg të ri të numrave të rastit gjatë çdo thirrje të funksionit rand(), kryhet randomizimi (angl. randomizing) me dhënien e vlerës

[8] Ekziston konstante RAND_MAX vlera e të cilës duhet të shtypet.

së ndryshme të numrave të plotë të pa shënuara si argument i funksionit standard(). Ky funksion duhet të realizohet para thirrjes së funksionit rand().

Për shembull, nëse te shembulli paraprak, para gjenerimit të numrave të rastit, kryhet funksioni:

```
srand(123);
```

do të gjenerohen këto numra të rastit:

```
Numra të rastit ndërmjet 1 dhe 150:
114 55 126 104 91
```

Nëse, përsëri kryhet funksioni, por me tjetër argument:

```
srand(321);
```

do të gjenerohen numra tjerë të rastit:

```
Numra të rastit ndërmjet 1 dhe 150:
116 131 88 69 137
```

Vërejmë se funksioni srand() thirret vetëm me emrin e tij, pa urdhër për shoqërim. Kjo do të thotë se ky funksion nuk kthen vlerë. Ka edhe të tjera funksione të atilla të cilat quhen **funksione pa vlerë kthyese** (angl. non value-returning functions) ose **procedura (**angl. procedures).

Shembulli 2.5.2

Njëri prej mënyra efikase për gjenerimin e vargjeve të ndryshme të numrave të rastit është shfrytëzimi i ores së kompjuterit. Ora lexohet me funksionin time(0), e cila gjendet te biblioteka <**ctime**> dhe e cila patjetër të kyçet në program, por kthen numër të plotë të kohës në atë moment të shprehur në sekonda.

```
1   #include <iostream>
2   #include <string>
3   #include <ctime>
4
5   using namespace std;
6
7   int main()
8   {   // Numra të rastësishëm: funksione random() dhe  srand()
9
10      cout <<"Numra të rastësishëm ndërmjet 0 dhe" << RAND_MAX << ": \n"
11          << rand() << " " << rand() << " " << rand() << " "
12          << rand() << " " << rand() << " " << endl;
13      int max = 100;
14      cout << "\nNumra të rastësishëm ndërmjet  0 dhe " << max - 1 << ": \n"
15          << rand() % max << " " << rand() % max << " " << rand() % max << " "
16          << rand() % max << " " << rand() % max << " " << endl;
17      cout << "\nNumra të rastësishëm ndërmjet 1 dhe" << max << ": \n"
18          << 1 + rand() % max << " " << 1 + rand() % max << " " << 1 + rand() % max
19          << " " << 1 + rand() % max << " " << 1 + rand() % max << " " << endl;
20      int poceten = 51;
21      cout << "\nNumra të rastësishëm ndërmjet 1 dhe" << fillestar  + max - 1 << ": \n"
22          << fillestar  + rand() % max << " " << fillestar  + rand() % max << " "
```

Figura 2.5.2

```
23          << fillestar + rand() % max << " " << fillestar + rand() % max << " "
24          << fillestar  + rand() % max << " " << endl;
25      srand(123);
26      srand(time(0));
27      cout << "\n Koha në sekonda " << time(0) << endl;
28      cout <<"Numra të rastësishëm ndërmjet 1 dhe" << fillestar + max - 1 << ": \n"
29          << poceten + rand() % max << " " << fillestar + rand() % max << " "
30          << poceten + rand() % max << " " << fillestar + rand() % max << " "
31          << poceten + rand() % max << " " << endl;
32
33      cout << endl;
34      system("Color 17");
35      system("pause");
36      return 0;
37  }
```

Figura 2.5.2(vazhdim)

Një dalje prej realizimit të programit është:

```
Numra të rastësishëm ndërmjet 0dhe 32767:
19169 26500 6334 18467 41

Numra të rastësishëm ndërmjet 0 dhe 99:
64 62 58 78 24

Numra të rastësishëm ndërmjet 1 dhe 100:
62 28 82 46 6

Numra të rastësishëm ndërmjet 1  dhe 150:
87 78 93 146 142

       Koha në sekonda:1469456833
Numra të rastësishëm ndërmjet 1 dhe 150:
61 72 108 132 137

Press any key to continue . . .
```

Me shfrytëzimin e funksionit time(0), te program prej ***figura 2.5.2***, do të gjenerohen këto vargje të numrave të rastit gjatë realizimit të programit dy herë:

```
 Koha në sekonda: 1469456809
Numra të rastësishëm ndërmjet 1dhe150:
56 108 52 114 59
      Koha në sekonda: 1469456833
Numra të rastësishëm ndërmjet:1dhe150:
61 72 108 132 137
```

Funksione për punë me shenja

Në C++ janë future më shumë funksione për punë me shenja. Ato gjenden te biblioteka <cctype>, e cila patjetër të kyçet në fillim të programit.

isdigit(z)	– true nëse z është shifra (0, 1... 9).
isalpha(z)	– true nëse z është shkronja.
islower(z)	– true nëse z është shkronja e vogël.
isupper(z)	– true nëse z është shkronja e madhe.
isblank(z)	– true nëse z është vend i zbrazët.
isspace(z)	– true nëse z është hapësirë e zbrazët: ' ','\t', '\n', '\v', '\f', '\r'[9].
isalnum(z)	– true nëse z është shkronja ose shifra.
ispunct(z)	– true nëse është shenjë interpuktive: . , ; : '' „" ? / ! – ( )[ ] { } < > … etj
isprint(z)	– true nëse z është shenjë që shtypet. (Për shembull, '\n', '\t' dhe të tjera nuk shtypen)
tolower(z)	– nëse z është shkronjë e madhe e konverton në shkronjë të vogël.
toupper(z)	– nëse z është shkronjë e vogël e konverton në shkronjë të madhe.

Shembulli 2.5.3

```
#include <iostream>
#include <string>
#include <iomanip>

using namespace std;

int main() {    // Funksione për punë me shenja

    char z1 = '5';
    char z2 = 'W';
    char z3 = '@';
    char z4 = '\n';
    char z5 = ' ';
    char z6 = '?';
    bool daNe;
    daNe = isdigit( z1 );
    cout <<"A është"   << z1 << " shifër? " << boolalpha << poJo << endl;
    daNe = isdigit( z2 );
    cout << "A është " << z2 << " shifër? " << boolalpha << poJo << endl;
    daNe = isalpha( z2 );
    cout << "A është " << z2 << " shkronjë? " << boolalpha << poJo << endl;
    daNe = isalpha( z3 );
    cout << "A është" << z3 << "shkronjë? " << boolalpha << poJo << endl;
    daNe = islower( z2 );
    cout << "A është" << z2 << "shkronjë e vogël ?  " << boolalpha << poJo << endl;
```

Figura 2.5.3

[9] '\t' – tabulator horizontal, '\n' – vijë e re, '\v' – tabulator vertical, '\f' – faqe e re, '\r' – vija paraprake.

```
26      pojo = isupper( z2 );
27      cout << "A është"  << z2 << " shkronjë e madhe "  << boolalpha << pojo << endl;
28      pojo = isblank( z4 );
29      cout << "A është"  << z4 << " vend i zbrazët? " << boolalpha << pojo << endl;
30      pojo = isblank( z5 );
31      cout << "A është"  << z5 << " vend i zbrazët? " << boolalpha << pojo << endl;
32      pojo = isspace( z4 );
33      cout <<"A është"  << z4 << " hapësirë e zbrazët " << boolalpha << pojo << endl;
34      pojo = isspace( z5 );
35      cout << "A është"  << z5 << "hapësirë e zbrazët" << boolalpha << pojo << endl;
36      pojo = isalnum( z1 );
37      cout << "A është"  << z1 << "shkronjë ose shifër?" << boolalpha << pojo << endl;
38      pojo = isalnum( z2 );
39      cout << "A është"  << z2 << " shkronjë ose shifër? " << boolalpha << pojo << endl;
40      pojo = isalnum( z6 );
41      cout << "A është"  << z6 << " shkronjë ose shifër?" << boolalpha << pojo << endl;
42      pojo = ispunct( z6 );
43      cout << "A është"  << z6 << "shenjë e interpuksionit?  " << boolalpha << pojo << endl;
44      pojo = ispunct( z4 );
45      cout << "A është"  << z4 << " shenjë e interpuksionit?  " << boolalpha << pojo << endl;
46      pojo = isprint( z4 );
47      cout << "A është"  << z4 << " shenjë që shtypet ?  " << boolalpha << pojo << endl;
48      pojo = isprint( z6 );
49      cout << "A është"  << z6 << " shenjë që shtypet ?  " << boolalpha << pojo << endl;
50      cout << " Shkronja " << z2;
51      z2 = tolower( z2 );
52      cout << " e shndërruar në të vogël është " << z2 << endl;
53      cout << " Shkronja " << z2;
54      z2 = toupper( z2 );
55      cout << " e shndërruar në të madhe është " << z2 << endl;
56
57      cout << endl;
58      system( "Color 17" );
59      system( "pause" );
60      return 0;
61  }
```

```
A është 5 shifra?   true
A është W shifra?   false
A është W shkronjë? true
A është @ shkronjë? false
A është W shkronjë e vogël? false
A është W shkronjë e madhe  true

   vend i zbrazët  false
A është   vend i zbrazët      true
A është
  hapësirë e zbrazët   true
A është     hapësirë e zbrazët ?  true
A është  5  shkronjë ose shifër ?   true
A është  W  shkronjë ose shifër ?   true
A është  ?  shkronjë ose shifër?    false
A është  ?  shenjë e interpunksionit   true

  shenjë e interpunksionit ? false
A është
   shenjë që shtypet        false
A është       shenjë që shtypet ?       true
Shkronja  W e shndërruar në të vogël është  w
Shkronja  w e shndërruar në të madhe është W
```

Figura 2.5.33 3 *(vazhdim)*

Një dalje prej realizimit të programit është:

Operatori sizeof()

Ky operator shfrytëzohet për njehsimin e madhësisë së memories (në bajt) që e zen ndonjë lloj të të dhënave. Sintaksa e tij është

sizeof(*miobjek*); ;

Ai mund të zbatohet edhe në një ndryshore ose shprehje, ku janë të nevojshme kllapa

sizeof shprehje;

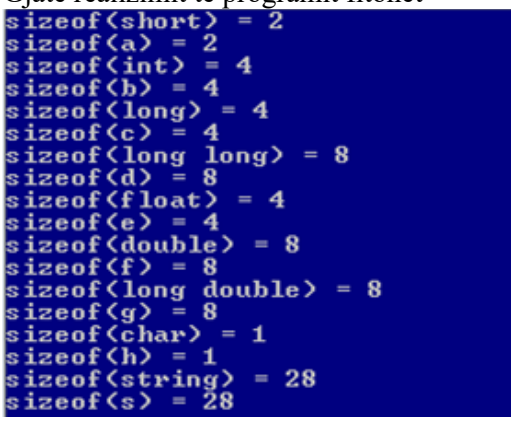
Shembulli 2.5.4

Me programin e ***figura 2.5.4*** është ilustruar shfrytëzimi i operatorit sizeof().

```
1   // Operator sizeof()
2
3   #include <iostream>
4   #include <string>
5   using namespace std;
6
7   int main() {
8       short a;    int b;  long c; long long d;
9       float e;    double f;   long double g;
10      char h;     string s;
11
12      cout << "sizeof(short) = " << sizeof(short) << endl;
13      cout << "sizeof(a) = " << sizeof a << endl;
14      cout << "sizeof(int) = " << sizeof(int) << endl;
15      cout << "sizeof(b) = " << sizeof b << endl;
16      cout << "sizeof(long) = " << sizeof(long) << endl;
17      cout << "sizeof(c) = " << sizeof c << endl;
18      cout << "sizeof(long long) = " << sizeof(long long) << endl;
19      cout << "sizeof(d) = " << sizeof d << endl;
20      cout << "sizeof(float) = " << sizeof(float) << endl;
21      cout << "sizeof(e) = " << sizeof e << endl;
22      cout << "sizeof(double) = " << sizeof(double) << endl;
23      cout << "sizeof(f) = " << sizeof f << endl;
24      cout << "sizeof(long double) = " << sizeof(long double) << endl;
25      cout << "sizeof(g) = " << sizeof g << endl;
26      cout << "sizeof(char) = " << sizeof(char) << endl;
27      cout << "sizeof(h) = " << sizeof h << endl;
28      cout << "sizeof(string) = " << sizeof(string) << endl;
29      cout << "sizeof(s) = " << sizeof s << endl;
30
31      cout << endl;
32      system("Color 17");
33      system("pause");
34      return 0;
35  }
```

Figura 2.5.4

Gjatë realizimit të programit fitohet



Detyra për ushtrime

Të shkruhen programe për këto detyra me shfrytëzimin e funksioneve të bibliotekës:

1. Të futet vlera për x dhe të njehsohet shuma $1 + x + x^2 + x^3 + x^4 + x^5$.
2. Të njehsohet shuma $e^0 + e^1 + e^2 + e^3 + e^4 + e^5$.
3. Të futet numr i plotë b dhe të njehsohet $\left[\sqrt{\frac{n^2}{2}}\right]$, [] paraqet pjesën e plotë.
4. Të futen koeficientët a, b dhe c të barazimit të katrorit $ax^2 + bx + c = 0$ dhe të njehsohen rrënjët sipas formulës.

   $D = b^2 - 4ac, \qquad x_1 = \frac{-b + \sqrt{D}}{2a}, \qquad x_2 = \frac{-b - \sqrt{D}}{2a}$.
5. Të shtypet kjo tabelë; me vlera për funksionet trigonometrike sin(x) dhe cos(x) për këndet prej 0^o, 30^o, 45^o, 60^o, 90^o, 180^o, 270^o dhe 360^o. (Këndi jepet në radian)

   $x^{rad} = \frac{\pi}{360} x^o$

x	0^o	30^o	45^o	60^o	90^o	180^o	270^o	360^o
sin(x)								
cos(x)								

6. Të gjenerohet numër tek i rastësishëm prej intervalit [31, 100].
7. Të futet shkronjë he të konstatohet a është e madhe.
8. Të futet shenjë dhe të konstatohet a është shenjë e interpurksionit.

Funksione shfrytëzuese

Tam se te programimi i strukturuar (e sqaruar te pika **1.2 Algoritme**) shfrytëzohen dy teknika të programimit, edhe atë:

- **Programimi prej lart poshtë** (angl. top-down programming).
- **Programimi modular** (angl. modular programming).

Orogramimi prej lart poshtë kryhet me ndarjen (zbërthim) të detyrës në pjesë më të vogla dhe detyra më të thjeshta, të cilat do t'i quajmë **nëndetyra.** Nëse është e nevojshme, edhe ato nëndetyra më tej ndahen akoma në të thjeshta ndërsa nuk fitohen detyra lehtë të programohen.

Çdo nëndetyrë prej detyrës së atillë të zbërthyer mund të shqyrtohet si detyrë e veçantë, pavarësisht prej të tjerave. Për çdo nëndetyrë mund të shkruhet algoritëm i veçantë, të cilin do ta quajmë **nënalgoritëm**.

Për shembull, algoritmi për gjetjen e numrit më të madh prej tre numrave të dhënë mund të shkruhet edhe në këtë mënyrë, ***figura 2.5.5*:**

```
algoritëm  MëIMadhiPrejTreNumrave
fillimi
        lexo a, b, c;
        nëse a > b
                atëherë
                        më i madh ← a;
                ndryshe
                        më i madh  ← b;
        fund_nëse  {a > b}
        p ← më i madh;
        nëse p > c
                atëherë
                        më i madh  ← p;
                ndryshe
                        më i madh  ← c;
        fund_nëse {p > c}
        n ← më i madh
        shtyp n;
fund { MëIMadhiPrejTreNumrave }
```

Figura 2.5.

Te algoritmi dy herë kërkohet caktimi i numrit më të madh prej dy numrave: a dhe b, dhe p dhe c. Kjo mund të veçohet si algoritëm i veçantë, përkatësisht nënalgoritëm.

Megjithatë, paraqitet problem se si të shkruhet ai nënalgoritëm që të mund me atë të realizohen të dy krahasimet: krahasimi a dhe b te i pari dhe p dhe c te i dyti.

Gjithashtu, nënalgoritmi duhet të kryejë krahasim të çfarëdo dy numrave. Që të mundësohet kjo, nënalgoritmi shkruhet në formën e përgjithshme.

Një nënalgoritëm shkruhet sikurse edhe çdo algoritëm tjetër. Edhe te ai mund të futen edhe të shtypen të dhëna, të deklarohen ndryshore, të përkufizohen lloje dhe konstante, të shoqërohen vlera ndryshoreve, të realizohen njehsime të ndryshme edhe operacione të tjera të cilat mund të realizohen edhe te algoritmet.

Nënalgoritmet, gjithashtu edhe algoritmet, emërtohen. Për shembull: NumërMëIMadh(), NumërIThjeshtë(), MëIVogëlPrejTëGjithëve(), *Zëvendësim*(), *ZhvendosjeEVargut*() etj.[10]

Për shembull, nënalgoritmi për gjetjen e numrit më të madh prej dy numrave mund të përkufizohet si te ***figura 2.5.6***:

```
nënalgoritmi NumriMëIMadh(numri1, numri2)
fillimi
        nësenumri1 > numri2
                atëherë
                        më i madh ← numri1
                ndryshey
                        më i madh ← numri 2;
        fund_nëse { numri1 > numri2}
        ktheje më i madh;
fund {Më i madh}
```

Figura2.5.6

Emri i nënalgoritmit është NumriMëIMadh(), por ka dy parametra hyrës numri1 dhe numri2.

Rezultati, i cili njehsohet te nënalgoritmet, kthehet me hapin

```
        ktheje më të madhin;
```

Nënalgoritmet e këtilla quhen **nënalgoritme funksionale**.

Nëse hapat për gjetjen e numrit më të madh te algoritmi prej ***figura 2.5.5*** i zëvendësojmë me nënalgoritmet prej ***figura 2.5.6***, algoritme do të duket si te ***figura 2.5.7*:**

```
algoritm MëIMadhiPrejTreNumrave
fillimi
        lexoa a, b, c;
        p ← MëIMadh(a, b);
        n ← MëIMadh ((p, c);
        shtyp n;
fund { MëIMadhiPrejTreNumrave }
```

Figura 2.5.7

[10] Emrat e nënalgoritmeve do t'i shkruajmë me shkronjën e madhe të parë. Do të vërejmë se disa emra janë shkruar alfabetin latin. Këtë do ta sqarojmë më vonë.

Te ana e djathtë prej hapave:

p ← NumriMëIMadh(a, b);
n ← NumriMëIMadh(p, c);

thirret nënalgoritmi NumriMëIMadh() për argumentet a, b dhe p, c. Kjo do të thotë se nënalgoritmi funksional thirret me emrin dhe argumentin te kllapat. Pasi nënalgoritmi funksional kthe vetëm një vlerë, ai mund të thirret në hap për shoqërim (si te ***figura 2.5.7***), në shprehje ose në hap tjetër.

Forma e përgjithshme e thirrjes së nënalgoritmeve funksionale është

EmriINënalgoritmit(l*ista_e_hyrjeve_argumente*);

Nënalgoritmi NumriMëIMadh() mund të shkruhet edhe kështu që vlera kthyese të jetë parametër, ***figura 2.5.6 a***:

```
nënalgoritëm NumriMëIMadh (↓numri1, ↓numri12, ↑
fillimi
        nëse numri1 > numri2
                atëherë
                        më i madhm ← numri1
                ndryshe
                        më i madh ← numri2;
        fund_nëse { numri1 > numri2}
fund {Më i madh}
```

Figura 2.5.6 a

Nënalgoritmet e këtilla quhen **nënalgoritme procedurale** ose vetëm **procedura** (angl. procedures). Lista e tyre e parametrave mund të ketë parametra hyrës, parametra dalës dhe hyrëse-dalëse[11].

Për t'u dalluar parametrat hyrëse, dalëse dhe hyrëse-dalëse te nënalgoritmet procedural, para parametrave hyrëse do të vendojmë shigjetë poashtë↓, por para parametrave dalëse shigjeta lart ↑. Nëse ndonjë parametër është hyrëse-dalëse, atëherë para tij do të vendojmë shigjetë dykahëshe ↕. (Këto shenja nuk shfrytëzohen te nënalgoritmet funksionale pasi tea to të gjitha parametrarjanë hyrëse).

Nënalgoritmi procedural thirret vetëm me emrin e tij, por në kllapa përmenden argumentet:

EmriINënalgoritmit(lista_e_hyrjeve_të_daljeve_dhe_të_hyrje_dalje_argumente);

Vërejtje: Emrin e nënalgoritmeve procedural do ta shkruajmë me alfabetin tone që të njihet se nënalgoritmi procedural nuk mund të thirret në tjetër hap prej algoritmit, sikurse që mund nënalgoritmi funksional.

[11] Në gjuhët programore ekzistojnë mënyra të ndryshme për shënimin e parametrave hyër, dalës dhe hyrë-dalës. Më së shpeshti shënohen me in, out dhe inout.

Sipas asaj që u tha, algoritmi për gjetjen e numrit më të madh prej tre numrave të dhënë, me shfrytëzimin e nënalgoritmeve procedurale NumiMëIMadh(), do të jetë si te ***figura 2.5.7 a*****:**

```
algoritmi MëIMadhiPrejTreNumrave
fillimi
        lexo a, b, c;
        NumriMëIMadh(a, b, p);
        NumriMëIMadh(p, c, n);
        shtyp n;
fund { MëIMadhiPrejTreNumrave }
```

Figura 2.5.7 a

Nënalgoritmet procedural shfrytëzohen më së shpeshti në raste kur nënalgoritmi duhet të kthen më shumë se një vlerë.

Për shembull, gjatë zgjidhjes së sistemit prej dy barazimeve lineare me dy të panjohura, zgjidhja përbëhet prej dy vlerave për dy të panjohura. Ose, gjatë zgjidhjes së barazimit katror, zgjidhja përbëhet prej y vlerave për rrënjët katrore dhe të ngjashme.

Nënalgoritmet të cilat kodohen te gjuhët programore quhen **nënprograme** (angl. subprograms). Nënprogramet mund të jenë të shkruara sikurse **funksione** (ang. functions) ose si **procedurale** (angl. procedures). Funksionet janë nënprograme të cilat japin (kthejnë pas realizimit) vetëm një rezultat. Procedurat janë nënprograme të cilët nuk kthejnë vlerë, (Por ekzistojnë mekanizma të cilat mundësojnë me procedurat të fitojë më shumë rezultatet).

Thelbi i nënprogrameve është në atë që atom und të shfrytëzohen në programe të ndryshme, por edhe te shumë vende në programin e njëjtë. Kjo është mundësuar nëpërmjet mekanizmit të **argumenteve formale** (angl. formal arguments) dhe **argumente të vërteta** (angl. actual arguments). Në literature, argumentet formale shpesh quhen **parametra** (angl. parametërs), por argumentet e vërteta vetëm **argumente**. Ne do t'i shfrytëzojmë dy terminet e fundit.

Nëo C++ ka vetëm funksione. Funksionet (sikurse edhe në matematikë) japin (kthejnë) vetëm një rezultat, d.m.th., një vlerë. Prandaj (sikurse edhe te funksionet e bibliotekës), quhen **funksione me vlerë kthyese.** Me funksionet e këtilla realzohen nënalgoritmet funksionale. Nënalgoritmet procedural realizohen me procedura (sikurse edhe funksonet e bibliotekës), të cilat në C++ quhen **funksione pa vlerë kthyese**.

Funksionet në C++ mund të jenë:

- Funksionet e bibliotekës.
- Funsione shfrytëzuese të përkufizuara.

Me funksionet e bibliotekës u njohtëm te nëntitulli **Funksionet e bibliotekës.**

Funksionet shfrytëzuese (angl. user functions) janë përkufizuar prej shfrytëzuesve (programuesit) dhe prandaj, quhen edhe funksione shfrytëzuese të përkufizuara (amgl user-defined functions).

Funksione shfrytëzuese me vlerë kthyese

Dë të fillojmë me shembullin për gjetjen e numrit më të madh prej Funsione shfrytëzuese të përkufizuara. tre numrave të dhënë, të sqaruar paraprakisht.

Nënalgoritmet funksiona NumriMëIMadh() prej ***figura 2.5.6***, mund të përkufizohet në C++ si funksion, ***figura 2.5.8*:**

```
int numriMëIMadh(int numri1, int numri2 {
	int më i madh;
	if(numri1 > numri2)		// Nëse numri1 është më i madh se numri2,
		më i madh = numri1	// atëherë është më i madh numri1,
	else				// ndryshe
		më i madh = numri2	// është më i madh numri2.
	return më i madh;
}
```

Figura 2.5.8

Emri funksion është numërMëIMadh[12]. Para emrit jepet lloj i rezultatit që e kthen funksioni numërMëIMadh () është i llojit int. Pas emrit të funksionit, në kllapa, jepen emrat dhe llojet e parametrave numri1 dhe numri2. Parametra janë ndryshore të cilat shfrytëzohen në përkufizimin e funksionit. Funksioni e jep (kthen) rezultatin me urdhër

```
return më i madh;
```

Funksioni kryesor main() për gjetjen e numrit më të madh prej tre numrave, është dhënë te **figura 2.5.9**:

```
int main() {
	int a; // numri i plotë i parë
	int b; // numri i plotë i dytë
	int c; // numri i plotë i tretë
	int p, n;
	cout << Vlera e tre numrave të plotë: \n";
	cout << "Numri i parë:";
	cin >> a;
	cout << "numri i dytë: ";
	cin >> b;
	cout << "Numri itretë: ";
	cin >> c;
```

[12] Emrat e funksioneve shfrytëzuese do t'i shkruajmë me shkronjën e parë të vogël. Nëse emri përbëhet prej më shumë fjalë të bashkuara, atëherë do fjalë (përveç të parit) do ta shkruajmë me shkronjë të madhe.

```
        // a,b dhe c janë argumente
        p = numriMëIMadh(a,b);      // Thirrje funksionit
        n = numriMëIMadh(p, c);     // Thirrje funksionit
        cout << "Më i madhi prej numrave" << a << ", " << b << "i"
                << c << "e" << n << endl;

        cout << endl;
        system("Color 17");
        system("pause");
        return 0;
}
```

Figura 2.5.9

Do ta anallizojmë funksionin main().

Pas leximit të trenumrave a, b dhe c, thirret funksioni me argumentet a dhe b në anën e djathtë te urdhri për shoqërim

```
p = numriMëIMadh(a, b);
```

Poashtu, vlerat e argumenteve a dhe b barten te parametrat numri1 dhe numri2. Gjatë realizimit të funksionit, vlera më e madhe prej numri1 dhe numri2 shoqërohet te ndryshorja dhe paraqet rezultat prej funksionit.

Me urdhrin

```
return më i madh;
```

funksioni numriMëIMadh() e kthen rezultatin, d.m.th., vlera e ndryshores më i madh te program kryesor. Prandaj, themi se funksioni numriMëIMadh() është funksioni me vlerë kthyese.

Me urdhrin për shoqërim

```
p = numriMëIMadh(a, b);
```

vlera e kthyer e ndryshores më e madhe i shoqërohet ndryshorja p.

Realizimi i urdhrit për shoqërim

```
p = numriMëIMadh(a, b);
```

është paraqitur në mënyrë skematike te ***figura 2.5.10***.

Gjatë realizimit të urdhrit vijues për shoqërim, te e cila përsëri thirret funksioni numriMëIMadh (), i ndryshores n i shoqërohet vlera e numrit më t madh prej numrave p dhe c c);

```
n = numriMëIMadh(p, c);
```

Pas realizimit të këtij urdhri, ndryshorja n e përmban vlerën më të madhe prej ndryshoreve a, b dhe c.

Argumentet e funksionit mund të jenë konstante, ndryshore ose shprehje, lloj i të cilit ose kmpatibile me llojin e parametrave përkatës në përkufizimin e funksionit.

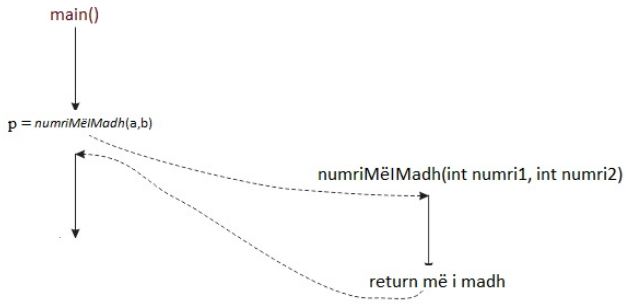


Figura 2.5.10

Për shembull, këto thirrje të funksionit numriMëIMadh() janë valid:

```
p = numriMëIMadh((int) (20 * 123 / 45),
p = numriMëIMadh((int) 1.2345f, (int) Math.sqrt(1.5));
```

Gjatë çdo thirrje të funksionit, kontrollohet numri dhe llo i arguumeneteve për kohën e përkthimit të programit.

- Nëse numri i argumenteve është i ndryshëm prej numrit të parametrave, paraqitet porosia për gabim.
- Nëse nuk dakordohet lloj i parametrave me llojin e argumenteve përkatëse, së pari përkthyesi përpiqet ta „përkthen“ (konverton) llojin e argumentit në lloj të parametrit përkatës. Për shembull, double në int. (Poashtu, mund të kryhet thirrje, por rezultati të jetë i gabuar). Nëse nuk ka sukses „përkthimi“ të llojit. Atëherë përkthyesi paraqet porosi për gabim. Një funksion me vlerë kthyese mund të thirr në vende të ndryshme te program, me argumente

të ndryshme dhe mënyra të ndryshme.

Për shembull:

```
      // Një mënyrë e thirrjes së funksionit:
p = numriMëIMadh(a, b);
      // Mënyra tjetër e thirrjes:
System.out.println("\nMë i madh është numri: " + numriMëIMadh (a, b));
      // Mënyra e tretë e thirrjes së funksionit:
if(numriMëIMadh(a, b) > 99)
      System.out.print("\nShpërblimi i parë.");
```

Nëse i vendojmë në një program funksionin numriMëIMadh () dhe funksionin kryesor main(), si te ***figura 2.5.11*** dhe nëse e realizojmë, do ta fitojmë këtë dalje:

```
Futni tre numra të plotë:
Numri i parë:123
Numri i dytë: 321
Numri i tretë: 231

Më i madh prej numrave 123, 321 dhe 231 është 321
Press any key to continue . . .
```

```
#include <iostream>
using namespace std;

// Përkufizimi i funksionit numriMëIMadh ().
// numri1 dhe numri2 janë parametra hyrës.

       Funksioni numriMëIMadh()

// Funksioni kryesor

       Funksioni main()
```

Figura 2.5.11

Përkufizimi dhe deklarimi i funksionit me vlerë kthyese

Përkufizimi dhe deklarimi i funksionit me vlerë kthyese në C++ kryhet në këtë mënyrë:

```
lloj i emrit (lista_e_parametrave)
        trupi i funksionit (urdhërat)
        return shprehja;
}
```

Titulli i funksionit

Vërejtje:

- Titulli i funksionit nuk mbaron me **;** (**pikëpresje**),
- *lloj* është lloj i funksionit, i cili është, në realitet, lloj i vlerës kthyese,
- *emri* është emri i funksionit.
- *lista_e_parametrave* (angl. parametërs list) patjetër t'i përmba edhe emrat dhe llojet e parametrave.
- *shprehja* është rezultat prej funksionit i cili mund të jetë shprehje (prej ndryshoreve dhe/ose funksione) ose një ndryshore dhe e cila patjetër të jetë e llojit të njëjtë me lloj të funksionit.
- Parametrat te lista_e_parametrave mund të jenë parametra hyrës (angl input parametërs) dhe parametra hyrës-dalës (angl. input-output parametërs).

Deklarimi i funksionit në C++ kryhet me urdhrin e formës

```
Lloj emri (lista_e_parametrave);
```

lloj – lloj i vlerës e cila e kthen funksioni,
emri – emri i funksionit,

lista_e_parametrave – i përmban emrat e parametrave dhe llojet e tyre, të ndara me presje.

Vërejtje:

- Nëse nuk përmendet *lloj* i funksionit, nënkuptohet lloj int.
- Emrat e funksioneve (sipas konventës) do t'i shkruajmë sikurse edhe ndryshoret[13], d.m.th., me shkronjë fillestare të vogël. Nëse emri përbëhet prej më shumë fjalëve, ço fjalë vijuese do ta shkruajmë me shkronjë të madhe.
- Lista e parametrave patjetër t'i përmban llojet e parametrave, por emrat mund, por jopatjetër.
- Deklarimii funksionit mbaron me shenjën ; (pikëpresje).

Deklarimi i funksionit quhet **prototip i funksionit** (angl. function prototype).

Me protipin funksional përshkruhet interfejsi i funksionit, përkatësisht ai jep informata për numrin dhe llojin e parametrave të cilat duhet të jenë të njohur para se të thirret funksioni, si edhe për llojin e rezultatit.

Do të përmendim disa shembuj të prototipeve funksional të funksioneve shërbyese:

```
int shumëzo(int m, int n);          – emri: shumëzo
                                    – parametra: m dhe n
                                    – lloj i funksionit: int
float rrënja(float x);              – lloj i funksionit është float
int më i vogël(int, int);           – lista pa emra të parametrave
int g();                            – funksioni me listë të zbrazët të parametrave,
                                      d.m.th., pa parametra.
int f(int a, b);                    – jo e drejtë, duhet:
                                      int f(int a, int b);
long syprina(long gjatësia,long gjerësia);  – lloj i funksioit: long
int h(void);                        – funksioni me listë pa parametra [14]
void shtyp(int numriPorosisë);      – funksioni i llojit void[15]
```

Pjesa prej prototipit funksional i cili i përmban emrin dhe lista e parametrave quhet **nënshkrimi funksional** (angl. function signature).

Për shembull, nënshkrime funksionale janë:

```
shumëzo(int m, int n);
rrënja(float b);
më i vogël(int, int);
syprina(long, long);
```

[13] Emrat e funksioneve mund të shfrytëzohen te shprehjet, por edhe si argument të funksioneve të funksioneve të tjera, d.m.th., njëjtë sikurse edhe ndryshoret.
[14] Fjalën void do ta sqarojmë më vonë.
[15] Do ta sqarojmë më vonë.

Funksioni me vlerë kthyese thirret me urdhrin për shoqërim:

ndryshore = emri(lista_e_argumenteve);

*Lista_e_argumenteve dhe lista*_e_paramentrave të funksionit *emri* patjetër të kenë:

- numr të njëjtë të argumenteve dhe parametrave,
- renditje të njëjtë të argumenteve dhe parametrave,
- lloj të njëjtë konvertibil dhe parametra përkatës.

Do të përmendim shembuj për thirrje të funksioneve:

```
prodhim = shumëzo(a, b);
rrKatrore = rrënja(12.345);
n = mëIVogël(c, p);
S = syprina(gjatësia, gjerësia);
```

Gjatë thirrjes së funksionit, vlerat e argumenteve (të cilët parprakisht patjetër të përkufizohen – kanë vlerë) bartet (kopjohet) në parametra përkatës.

Për shembull, gjatë thirrjes

```
prodhim = shumëzo(a, b);
```

vlerat e argumenteve a dhe b barten (kopjohen) te parametrat m dhe n prej përkufizimit të funksionit shumëzo ()

```
int shumëzoi(int m, int n) {...}
```

Funksionet të cilat janë të përkufizuara sipas funksionit kryesor main(), patjetër të deklarohen (duke përmend prototipin e tyre) para funksionit kryesor që të mund të thirren te funksioni kryesor, përkatësisht që të dihet përkthyesi kur e përkthen funksionin kryesor që paraqesin ato emra.

Te përkufizimi i funksionit, patjetër të përmendet çka kthehet si rezultat pas realizimit të tij, qoftë të thirret prej funksionit kryesor ma-in() ose prej tjetër funksioni.

Një funksion mund vetëm njëherë të përkufizohet te programi i njëjtë.

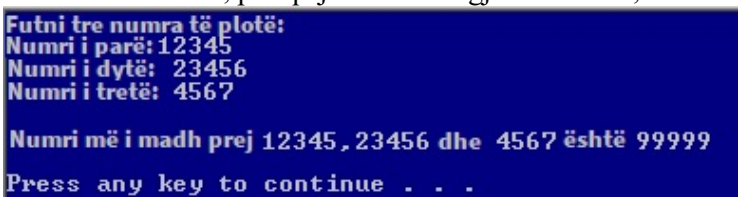
Parametrat konstant të funksionit

Te funksioni numriMëIMadh() prej ***figura 2.5.8***, parametra janë numri1 dhe numri2. Por është e mundshme te vet funksioni të ndryshojë vlera e nërit prej këtyre parametrave.

Për shembull, nëse te funksioni e shtojmë urdhrin:

```
numri1 = 99999;
```

atëherë te program prej ***figura 2.5.11*****,** pa marrë parasysh vlerat që i futim për numri1, numr2 dhe numri3, por që janë mëtë vegjël se 99 999, rezultati do të jetë 99 999:



Për të mbrojtur parametrat te funksioni prej ndryshimeve te ai, ato shënohen sikurse parametra konstant me kvalifikatorin (angl. qualifier) const.

Te funksioni vijues (***figura 2.5.12***) njehsohet mosha e njeriut, si rezultat prejrrjedhës dhe vitit të lindjes. Pasi viti i lindjes është i pandryshueshëm, duhet të shënohet si konstante.

```
1   #include <iostream>
2   using namespace std;
3
4   int mosha ( int const, int );
5
6   int main() {
7       int vitiLindjes, vitiSot;
8       cout << "Futni vitin e lindjes:";
9       cin >> vitiLindjes;
10      cout << " Futni vitin e tanishëm: ";
11      cin >> vitiSot       ;
12
13      int   vitin  = mosha ( vitiLindjes, vitiSot; );
14      cout << "\nMosha juaj është"      << vite     << " vite " << endl;
15
16      cout << endl;
17      system( "Color 17" );
18      system( "pause" );
19      return 0;
20  }
21
22  int  mosha ( int constvitiLindjes , int vitiSot) {
23      int     vite = vitiSot vitiLindjes;
24      return vite;
25  }
```

Figura 2.5.12

Pasi paramet vitiLindjes është kualifikuar si konstante te përkufizimi i funksionit, ai nuk mund të ndryshojë te ai – do të paraqitet gabim.

Të përmendim se te shembulli prej ***figura 2.5.12***, funksioni është përkufizuar sipas funksionit kryesor main() dhe prandaj para main() është përmend prototipi i funksionit

Detyra të zgjidhura

Detyra 2.5.1

Të shkruhet funksioni për njehsimin e xn, pa shfrytëzimin e funksionit të bibliotekës.

Programi është dhënë te ***figura 2.5.13***.

```
1   // Funksioni x^n.
2   #include <iostream>
3   using namespace std;
4
5   int XnaN( int n, double x ) {
6       double p = 1;
7       for( int i = 1; i <= n; i++ )
8           p = p * x;
9       return p;
10  }
11
12  int main() {
13      system( "Color 17" );
14      int n;
15      double a, fuqi;
16      cout << "Futni bazën:" a =";  cin >> a;
17      cout << "Futni fuqinë: n = "; cin >> n;
18      stepen = XnaN( n, a );
19      cout << "\nVlera e fuqisë " << a << "^" << n
20          << " = " << fuqia << endl;
21
22      cout << endl;
23      system( "Color 17" );
24      system( "pause" );
25      return 0;
26  }
```

Figura 2.5.13

Detyra 2.5.2

Të shkruhen funksione të veçanta për njehsimin e numrit dhe të shumës së shifrave të numrit natyror.

Programi është dhënë te ***figura 2.5.14.***

```
1   // Numri dhe shua e shifrave të numrit natyror
2   #include<iostream>
3   using namespace std;
4
5   int  numrilShifrave ( int m );
6   int  shumaEShifrave  ( int m );
7
8   int main() {
9       system( "Color 17" );
10      int n;
11      cout << "Futninumër natyror ";  cin >> n;
12      cout << "\nNumri i future është " << numrilShifrave ( n )
13           << " - shifror " << endl;
14      cout << "\nShuma e shifrave të numrit të future është " <<
15       numrilShifrave ( n ) << endl;
16
17      cout << endl;
18      system( "Color 17" );
19      system( "pause" );
20      return 0;
21  }
22
23  int numrilShifrave ( int m ) {
24      int numër = 0;
25      do {
26        numër ++;
27          m /= 10;
28      } while( m > 0 );
29      return numër;
30  }
31
32  int numrilShifrave ( int m ) {
33      int shifra, shuma = 0;
34      do {
35          shifra = m % 10;
36        shuma += shifra;
37          m /= 10;
38      } while( m > 0 );
39      return shuma;
40  }
```

Figura 2.5.14

Detyra për ushtrime

Për ktëto detyra të shkruhen funksionet me vlerë kthyese:

1. Njehsimi i n!
2. Gjetja e vlerës mesatare të dy numrave.
3. Njehsimi i perimetrit të shumëkëndëshit, ku shfrytëzohet funksioni për njehsimin e largësisë ndërmjet dy pikave në rrafsh.
   (Nëse është A(x_a, y_a) dhe B(x_b, y_b), largësia $d = \overline{AB} = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$).

4. Thjeshtimi i thyesës $\frac{a}{b}$ (a dhe b janë numra të plotë) me pjesëtim të a dhe b me PVP të tyre. Për gjetjen e PVP prej a dhe b, të shkruhet funksion i veçantë.
5. Gjetja e SHMP të dy numrave.
6. Gjetja e pjesëtuesit më të madh të përbashkët për n numra. Për gjetjen e PMP për dy numra, të shkruhet funksion i veçantë.
7. Të shkruhen funksione të veçanta për njehsimin e shumës dhe ndryshimit të dy numrave binary të shprehur si numra të plotë dekad shifrat e të cilëve janë vetëm 0 dhe 1. (Për shembull, për x = 10011101 dhe y = 01101001.)
8. Njehsimi i shumës së dy numrave në sistemin numerik me bazë 8. (Shifrat në këtë sistem numerik janë: 0, 1, 2, 3, 4, 5, 6 dhe 7).
9. Kontrolli numri natyror n a është numër i përsosur. (Numrat e përsosur janë ato numra të cilët janë të barabartë me shumën e pjesëtuesve të tyre pa vet numrin). Të gjenden të gjithë numrat e përsosur më të vegjël se numri n.
10. Kontrolli numri natyror n a është numër i thjeshtë ose nuk është, (Numri është i thjeshtë nëse është i plotpjesëtueshëm me 1 dhe me vetveten). Të gjenden të gjithë numrat e thjeshtë më të vegjël se numri natyror n.
11. Të njehsohet shuma 1 + (1 + 2) + (1 + 2 + 3) +... + (1 + 2 + 3 +... + n) me y-nevojë e funksionit për shumën e k-numrave të parë natyror, d.m.th., 1 + 2 +...+ k.
12. Të gjendet ekuivalenti dekad i numrit të plotë binar.

Funksione shfrytëzuese pa kthim t vlerës

Funksionet në C++ të cilët nuk kthejnë vlerë themi se janë funksione të llojit **void.**

Për shembull, përkufizimi i funksionit shtypje() mund të jetë:

```
void shtypje (int numër, float rezultat) {
	cout << numër<< rezultat << endl;
	// numri është numri i llojit etj
	// rezultat është numri i llojit float
}
```

Thirrja e këtij fuksioni mund të jetë funksion i main() ose prej tjetër funksioni. Thirrja pa vlerë kthyese dallohet prej thirrjes së funksionit me vlerë kthyese. Funksioni pa vlerë kthyese thirret vetëm me emrin dhe me listën e argumenteve.

Për shembull:

```
shtypja (n , x);
```

Funksioni i llojit void mund të jetë pa parametra.

Për shembull:

```
void vlez() {
	cout << "Futja e të dhënave hyrëse \n  ";
}
```

Ose

```
void dalje(void) {   // Me parametrin void është njëjtë
                              // sikurse edhe pa atë
       cout << " Shtypja e të dhënave dalëse \n ";
}
```

Thirrja e funksionit prej llojit void mund të jetë:

```
hyrje();      hyrje(void);         dalje();      dalje(void);
```

Prej funksionit të llojit void dilet pa urdhrin return (sikurse prej funksionit me vlerë kthyese), automatikisht sipas realizimit të tij. Por mund të dilet edhe prej kudo në trupin e funksionit të llojit void, me urdhrin return pa parametër.

Për shembull:

```
void rrënjakatrore ( double numër ) {
    if(numër < 0 ) {
        cout << "Numri është negative dhe nuk ka rrënjë katrore" << endl;
        return;
    }
    else
        cout << " Rrënja katrore prej " << numër << " është "
        << sqrt(numër ) << endl;
    cout << "Fundi i void-funksioni " << endl;
}
```

Për shembull, me

```
double numri = 2;
```

Rrënja Katrore(numër);

do të shtypet:

```
Rrënja katrore prej 2 është 1.41421
Fundi i void-funkionit

Press any key to continue . . .
```

Nëse funksioni i llojit void thirret me

```
double numri = -2;
```

rrënjaKatrore(numri);

do të shtypet:

```
Numri është negative dhe nuk ka rrënjë katrore.

Press any key to continue . . .
```

Vërejmë se te thirrja e dytë nuk realizohet urdhër

```
cout << "Fund ii procedurës." << endl;
```

sikurse gjatë thirrjes së parë pasi funksioni mbaron me urdhrin return.

Funksionet pa vlerë kthyese shfrytëzohen më së sheshti në rast kur funksioni duhet të kthen më shumë se një velrë. Për shembull, kur duhet ndryshoret t'i shkëmbejnë përmbajtjet, duhet të dy vlerat të kthehen. Kur duhet të klasifikohen n të dhëna, ato duhet të kthehen dhe ngjashëm.

Sintaksa e përgjithshme e funksionit të llojit void është:

```
void EmriIFunksionit(lista_e_hyrjeve_ të_daljeve_ dhe_të_hyrje_daljeve_
                                                        _parametra) {
        trupi i funksionit
}
```

Funksionet pa vlerë kthyese thirren vetëm me emrin e tyre, por në kllapa përmenden argumentet:

```
EmriIFunksionit(lista_e_hyrjeve_ të_daljeve_ dhe_të_hyrje_daljeve
                                                        _argumenteve);
```

Vërejtje: Funksioni pa vlerë kthyese nuk mund të thirret në tjetër urdhër prej funksionit kryesor ose prej tjetër funksioni, sikurse që mundeshte funksioni me vlerë kthyese.

Shembulli 2.5.5

Të shkruhet aplikimi te i cili do të shfrytëzohet funksioni i llojit void për zëvendësim të vlerave të dy ndryshoreve.

Programi është dhënë te ***figura 2.5.15.***

Një dalje prej programit është:

```
a = 3
b = 4
Të zëvendësuara a = 3 dhe b = 4 janë
a = 4 dhe b = 3
Press any key to continue . . .
```

Nëse te funksioni kryesor main(), pas thirrjes së funksionit zëvendësim() të llojit void e shtojmë urdhrin:

```
cout << "Vlerat e a dhe b pas kthimit prej funksionit void zëvendësim() janë: "
     << "a = " << a << "i b = " << b << endl;
```

do të shtypet:

```
a = 3
b = 4
Të zëvendësuara a = 3 dhe b = 4 janë a = 4 dhe b = 3
Vlerat e dhe pas kthimit prej void funksioni zëvendësim ()janë: a = 3 dhe b = 4
Press any key to continue . . .
```

Vërejmë se funksioni i llojit void shtypen vlera e zëvendësuara të a dhe b, por pas kthimit te thirrësi (funksioni main()), vlerat e tyre nuk janë të zëvendësuara.

T'i analizojmë procesin e realizimit të programit prej S**hembulli 2.5.5 (*figura 2.5.15*)**, d.m.th., thirrja e funksionit zëvendësim() të llojit void prej ma-in(), realizimi i funksionit dhe kthimi prapa në main().

```
1   #include <iostream>
2   using namespace std;
3
4   void zëvendësim ( int, int );
5
6   int main() {
7       // Zëvendësimi i vlerave të dy ndryshoreve (me void funksionin)
8
9       int a, b;
10      cout << "a = ";     cin >> a;
11      cout << "b = ";     cin >> b;
12      cout << " Të zëvendësuara   a = " << a << "dhe b = " << b << "janë";
13      zëvendësimi ( a, b );
14
15      cout << endl;
16      system( "Color 17" );
17      system( "pause" );
18      return 0;
19  }
20
21  void zëvendësim ( int i pari, int i dyti ) {
22      int pom;
23      pom = i pari;
24      i pari = i dyti;
25      i dyti = pom;
26      cout << "a = " <<i pari<< " dhe b = " << i dyti << endl;
27  }
```

Figura 2.5.15

Gjatë realizimit të programit, së pari te memoria rezervohet hapësira (***figura 2.5.16***) për ndryshoret a dhe b gjatë deklarimit të tyre, por pastaj plotësohet me vlerat që janë future gjatë leximit.

Me thirrjen e funksionit prej llojit void, veprimi vazhdon me realizimin e funksionit. Së pari, te memoria rezervohen hapësirë për parametrat i parë dhe i dytë dhe tea to barten (kopjohen) vlerat e argumenteve a dhe b. Pastaj, rezervohet hapësira për ndryshoren pom dhe realizohen urdhërat:

```
pom = i parë;
i parë = i parë;
i dytë = pom
```

Kjo është treguar në anën e djathtë te ***figura 2.5.16.***

Në fund të funksionit prej llojit void, shtypen vlerat e të parit dhe të dytit, që i përmbajnë vlerat e b dhe a.

Pas realizimit të funksionit zëvendësim() të llojit void, veprimi kthehet te funksioni main(), ku ndryshoret a dhe b i mbajnë vlerat e njëjta, që vërehet edhe prej ***figura 2.5.16.***

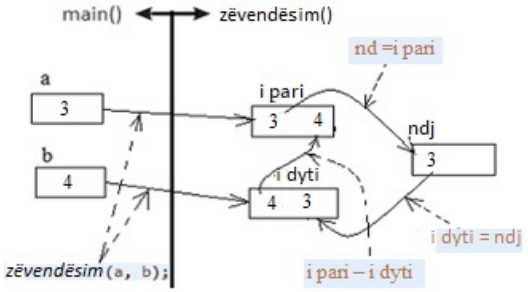


Figura 2.5.16

Ky mekanizëm quhet **bartja e argumenteve sipas vlerës**, por argumentet quhen **argument të bartura sipas vlerës** (angl. pass-by-value arguments).

Është e rëndësishme të përmendim se vlerat e parametrave te funksionet të llojit void mund të ndryshojnë, por gjatë kthimit te thirrësi (funksioni krysor ose funksioni tjetër), ato nuk kopjohen në argument përkatëse. Prandaj, argumentet e mbajnë vlerën të cilën e kanë pasur para thirrjes të funksionit të llojit void. Domethënë, kopjimi kryhet njëkahëshe, d.m.th., vetëm të argumenteve te parametrave, por jo edhe të kundërtën. Parametrat e këtilla quhen **parametra e vlerave** (angl. value parametërs).

Pas realizimit të funksionit zëvendësim() të llojit void, memoria e përgjithshme e cila ka qenë e zënë gjatë realizimit të tij (ndryshoret e deklaruara te aid he te lista e saj të parametrave) janë lirohet. Te memoria ngelin vetëm ndryshoret të krijuara gjatë realizimit të funksionit main().

Shpeshherë,është e nevojshme të kthehen më shumë rezultate prej funksionit të llojit void te thirrësi (funksioni main() ose tjetër funksion). Pasi funksionet e llojit void në C++ janë funksione të cilat nuk kthejnë vlera, kthimi i më shumë rezultateve është mundësuar me të ashtuquajturën **parametra referent** (angl. reference parametërs).

Parametra referent

Referencimi (ang. referencing), thënë shkurt, është shoqërim emër tjetër të ndryshores së njëjtë. Për shembull:

```
int b = 3;
int & r = b;
```

Me urdhrin e dytë, deklarohet referenca r, e cila referohet te ndryshorja b.

Shenja & (ampersand) lexohet „referenca“. Ndryshorja r është **ndryshorja e referencës** (angl. reference variable) ose shkurt **referenca** (angl. reference).

Urdhri i dytë lexohet: „r është reference numër i plotë inicializim i b“ ose „r është reference numër i plotë e cila referon b“.

Gjatë deklarimit të referencës, ajo patjetër të inicializohet ashtu që shoqërohet ndryshorja të cilës i referohet. Pas deklarimit dhe inicializimit të referencës, ajo nuk mund të ndryshojë të referon në tjetër ndryshore. Por mund të deklarohen më shumë referenca të cilat referojnë në ndryshoren e njëjtë.

Gjithashtu, lloj i referencës patjetër të jetë e njëjtë me llojin e ndryshores te e cila referohet.

Pasi referenca referon në ndonjë ndryshore, shpesh themi se referenca është tjetër emër për ndryshoren. Për shembull, nëse i shtypim ndryshoret b dhe referenca r e cila referon te ajo:

```
int b;
int& r = b;
b = 3;
cout << "Ndryshorja b = " << b << endl;
cout << "Referenca r = " << r << endl;
```

do të fitohet vlerë e njëjtë:

```
Ndryshorja   b = 3
Referenca    r = 3

Press any key to continue . . .
```

Te skema vijuese (***figura 2.5.17***) është ilustruar referenca r e cila ndryshore b te memoria. Pasi edhe b edhe r janë ndryshore të cilat zënë hapësirë te memoria (gjenden te disa adresa), referimi kryhet ashtu që referenca r e përmban adresën e ndryshores b.

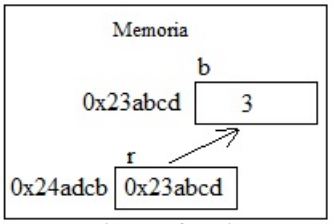


Figura 2.5.17

Zbatimi i referencave sikurse parametrat referent te funksionet pa vlerë kthyese, do ta sqarojmë te i njëjti **Shembull 2.5.5**.

Pasi pas kthimit prej funksionit zëvendësim() të llojit void, vlerat e ndryshoreve a dhe b nuk janë ndryshuar, funksionin do ta shkruajmë me parametra referente (***figura 2.5.18***) para të cilave vendohet shenja &:

```
void zëvendësimi(int &i pari, int &i dyti) {
	int pom;
	pom = i pari;
	prv = i dayti ;
	i dayti  = pom;
	cout << "a = " << i pari  << "dhe b = " << dyti << endl;
}
```

Figura 2.5.18

Gjithashtu prototipi i funksionit të llojit void duhet të shënohet se parametrat janë referent:

```
void zëvendësimi(int &, int &);
```

Rezultati sipas realizimit të aplikimit do të jetë i saktë:

```
a = 3
b = 4
Të zëvendësuar a = 3 dhe b = 4 janë a = 4 dhe b = 3
Vlerat e a dhe b pas kthimit prej void funksioni zëvendësim () janë a = 4 dhe b = 3

Press any key to continue . . .
```

Shenja & mund të vendohet kudo qoftë ndërmjet llojit dhe emrit të parametrit:

```
int & i parë  int& i parë   int & i parë
```

Ky mekanizëm i bartjes së vlerave prej thirrësit të funksionit dhe e kundërta quhet **bartës sipas referencës** (angl. pass-by-referencës), por argumentet quhen **argument të bartur sipas referencës** (angl. pass-by-reference arguments).

Thirrja e funksionit të llojit void, kryhet me urdhrin që përbëhet prej emrit të funksionit dhe lista e argumenteve. Për shembull, funksioni zëvendësim() të llojit void do të thirret me urdhrin

```
zëvendësim(a, b);
```

Poashtu, numri dhe renditja e argumenteve patjetër të përgjigjet numrit dhe renditjes së parametrave. Por

- llojet e argumenteve të cilat barten sipas vlerës mund të jenë konstante, ndryshore ose shprehje, llojet e të cilëve janë të njëjtë ose konvertibile me llojet e vlerave përkatëse të parametrave,
- llojet e argumenteve të cilat barten sipas referencës patjetër të jenë ndryshore (jo konstante ose shprehje) dhe atë të llojit të njëjtë sikurse parametrat përkatës referent.

Gjatë thirrjes së funksionit të llojit void me parametrat referent, nuk krijohen ndryshore të reja për parametra referent. Gjatë kohës së realizimit të funksionit të llojit, *parametrat referente janë trajtuar sikurse emrat tjera për argument përkatëse prej thirrjes.* Kjo arrihet ashtu që parmetrat refernte u shoqërohen adresa e argumenteve përkatëse. Prandaj, *të gjitha ndryshimet te parametrat referente, në realitet, kryhen mbi argumentet përkatëse* pasi ato janë emra të ndryshëm të lokacionit të njëjtë të memories.

Skema te ***figura 2.5.19*** e ilustron realizimin e aplikimit prej ***figura 2.5.15*** me ndryshimet e bëra për parametra e funksionit të llojit void si te ***figura 2.5.18***.

Me krahasimin e ***figura 2.5.16*** dhe ***figura 2.5.19***, vërehet ndryshimi gjatë realizimit të funksionit zëvendësim() kur është shkruar si funksion me vlerë kthyese ***(figura 2.5.15)*** dhe sikur funksioni pa vlerë kthyese (***figura 2.5.18*****).**

Te nëntitulli **Parametrat konstante të funksionit**, sqaruam se parametrat konstante të funksionit me vlerë kthyese nuk mund të ndryshohen te funksionet. E njëjta vlen edhe për funksionet pa vlerë kthyese. Pavarësisht a është parametrik me vlerë ose referent, nëse duam të mos ndryshon te funksioni, duhet të shënohet si konstante me kualifikatorin const.

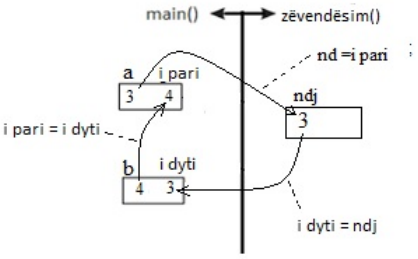


Figura 2.5.19

Detyra të zgjidhura

Detyra 2.5.13
Të renditen tre numra sipas madhësisë.

Programi është dhënë te ***figura 2.5.20***.

```
1   //Renditja e tre numrave
2   #include <iostream>
3   using namespace std;
4
5   void zëvendësim ( double& i pari, double& i dyti ) {
6        double pom = i pari ;
7        i pari = i dyti ;
8        i dyti = pom;
9   }
10
11  int main() {
12       double a, b, c;
13       cout << "Futni numra a, b dhe c." << endl;
14       cout << "a = "; cin >> a;
15       cout << "b = "; cin >> b;
16       cout << "c = "; cin >> c;
17       cout << "Numrat e renditur " << a << ", " << b << " dhe "
18            << c << " se: ";
19       if( a > b ) zëvendësim ( a, b );
20       if( a > c ) zëvendësim ( a, c );
21       if( b > c ) zëvendësim ( b, c );
22       cout << a << ", " << b << ", " << c << endl;
```

Figura 2.5.20

```
23      cout << endl;
24      system( "Color 17" );
25      system( "pause" );
26      return 0;
27  }
```

Figura 2.5.20 (vazhdim)

Një dalje e programit është:

```
Futni numra a, b dhe c
a  = 76.5
b  = 65.4
c  = 54.3
Numrat e renditur 76.5, 65.4 dhe 54.3 janë 54.3, 65.4, 76.5

Press any key to continue . . .
```

Detyra 2.5.14

Të zgjidhet sistemi prej dy barazimeve lineare me dy të panjohura

$a_1x + b_1y = c_1$

$a_2x + b_2y = c_2$

Zgjidhje e sistemit të barazimeve është:

$D = a_1b_2 - a_2b_1$; $D_1 = c_1b_2 - c_2b_1$; $D_2 = a_1c_2 - a_2c_1$;

nëse $D \neq$

atëherë

$x = \frac{D_1}{D}$; $y = \frac{D_2}{D}$;

ndryshe

nëse $D_1=0$ ***dhe*** $D_2=0$

atëherë

shtyp „Sistemi ka $\propto$ zgjidhje.“;

ndryshe

shtyp „Sistemi nuk ka zgjidhje.“;

Funksioni dhe aplikimi janë të dhënë te ***figura 2.5.21.***

Një dalje pas realizimit të aplikimit është:

```
Futni koeficientët a1, b1 dhe c1:
a1  = 3
b1  = -2
c1  = 5
Futni koeficientët a2, b2 dhe c2:
a2  = -4
b2  = 7
c2  = -1

Sistemi i barazimeve lineare:

         +3x -2y = +5
         -4x +7y = -1
     Ka zgjidhje: x = +2.53846, y = +1.30769

Press any key to continue . . .
```

```
1   // Sistemi me dy barazime lineare me 2 të pnjohura
2   #include <iostream>
3   using namespace std;
4
5   void sistemBarazimeLin ( int a1, int b1, int c1, int a2, int b2, int c2,
6                                                 double& x, double& y ) {
7       int D = a1 * b2 - a2 * b1;
8       int D1 = c1 * b2 - c2 * b1;
9       int D2 = a1 * c2 - a2 * c1;
10      if( D != 0 ) {
11          x = ( double ) D1 / D;
12          y = ( double ) D2 / D;
13          cout << "Ka zgjidhje : x = " << x << ", y = " << y << endl;
14      }
15      else {
16          if( D1 == 0 && D2 == 0 )
17              cout << "Ka pakufi shumë zgjidhje " << endl;
18          else
19              cout << "Është kundërthënëse dhe nuk ka zgjidhje " << endl;
20      }
21  }
22
23  int main() {
24      int a1, b1, c1, a2, b2, c2;
25      double x, y;
26      cout << "Futni koeficientët a1, b1 i c1: \n";
27      cout << "a1 = ";    cin >> a1;
28      cout << "b1 = ";    cin >> b1;
29      cout << "c1 = ";    cin >> c1;
30      cout << "Futni koeficientët a2, b2 i c2: \n";
31      cout << "a2 = ";    cin >> a2;
32      cout << "b2 = ";    cin >> b2;
33      cout << "c2 = ";    cin >> c2;
34      cout << "\n Sistemi i barazimeve lineare" << endl;
35      cout << showpos << internal;
36      cout << "\t" << a1 << "x " << b1 << "y = " << c1 << endl;
37      cout << "\t" << a2 << "x " << b2 << "y = " << c2 << endl;
38      sistemBarazimeLin ( a1, b1, c1, a2, b2, c2, x, y );
39
40      cout << endl;
41      system( "Color 17" );
42      system( "pause" );
43      return 0;
44  }
```

Figura 2.5.21

Detyra për ushtrime

Të shkruhen funksione të llojit void për këto detyra:

1. Të njehsohen shkallë e numrave x dhe y, x^y dhe y^x.
2. Të shndërrohen koordinatat polare dhe në koordinata të Dekartit x dhe y, sipas formulave: $x = \rho\cos\varphi, y = \rho\sin\varphi$.
3. Të zgjidhet barazimi katror $ax^2 + bx + c = 0$.
4. Të renditen tre numra sipas madhësisë.
5. Të gjenden numri më i madh dhe më i vogël që mund të formohen prej shifrave të numrit natyror 5-shifror.
6. Të gjendet mosha e njeriut në ditën e sotshme dhe ditëlindja. Datat të shprehen me ditë, muaj dhe vite.
7. Të njehsohet mesi aritmetik, gjeometrik dhe harmonic i n numrave të futur.
8. Të futen n numra dhe të gjenden numri më i madh dhe më ivogël prej atyreve.
9. Të shndërrohet numri dhjetor në binary, ashtu që funksioni ta kthen ekuivalentin binary të gjithë pjesës dhe të pjesës dhjetore në veçanti
10. Të gjenerohet trekëndëshi i Paskalit për n = 5

```
                    1                       n = 0
                1       1                   n = 1
            1       2       1               n = 2
        1       3       3       1           n = 3
    1       4       6       4       1       n = 4
1       5       10      10      5       1   n = 5
```

Thirrja e funksionit në funksion

Një funksion mund të thirret në tjetër funksion. Atë do ta tregojmë me këtë shembull.

Shembulli 2.5.6
Të gjendet numri më i madh prej 5 numrave të dhënë.

Për zgjidhjen e detyrës do ta shfrytëzojmë funksionin numriMëIMadh() për gjetjen e numrit më të madh prej dy numrave prej ***figura 2.5.8*****,** por program prej ***figura 2.5.9*** do ta shkruajmë si funksion për gjetjen e numrit më të madh prej tre numrave me emrin numriMëIMadh Prej3Numrave(). Pastajm numri më i madh prej 5 numrave do ta gjejmë me thirrjen e këtyre funksioneve në këtë mënyrë:

```
numriMëIMadh(a, b), numriMëIMadhPrej3Numrave(c, d, e));
```

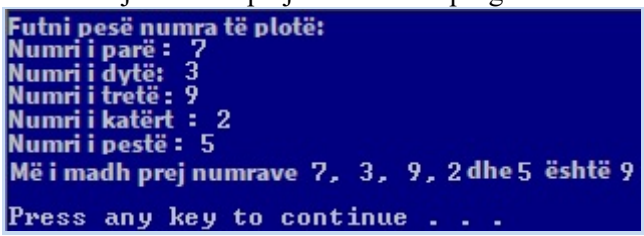
Vërejtje: Me kombinimin e thirrjeve të ëtyre dy funksioneve, mund të gjendet numri më i madh prej çfarëdo numrave.

Programi është dhënë te ***figura 2.5.22***.

```
1   #include <iostream>
2   using namespace std;
3
4   int numriMëlMadh ( int numri1 , int numri2 );
5   int numriMëlMadhPrej3Numrave ( int numri1., int numri 2, int numri3  );
6
7   int main() {
8       int a, b, c, d, e, më i madh;
9       cout << "Futni pesë numra të plotë \n";
10      cout << " Numri i parë: ";   cin >> a;
11      cout <<"Numri i dytë: ";  cin >> b;
12      cout << Numri i tretë:    ;  cin >> c;
13      cout << "Numri i katërt:";    cin >> d;
14      cout << "Numri i pestë:"; cin >> e;
15
16       më i madh  = numriMëlMadh ( numriMëlMadh  ( a, b ),
17                          numriMëlMadhPrej3Numrave  ( c, d, e ) );
18      cout << "Numri më i madh prej numrave " << a << ", " << b << ", "
19          << c << ", " << d << " dhe " << e << " është "
20          <<më i madh << endl;
21
22      cout << endl;
23      system( "Color 17" );
24      system( "pause" );
25      return 0;
26  }
27
28  int numriMëlMadh ( int broj1, int broj2 ) {
29      int më i madh;
30      if( numri1 > numri2 )      // Nlse numri1 është më i madh se numri2
31        më i madh =numri1;     // më i madh=numri1
32      else                       // atëherë
33        më i madh =numri2;     // më i madh është numri 2
34      return  më i madh;
35  }
36
37  int  numriMëlMadhPrej3Numrave  ( int numri1, int numri2 , int numri3 ) {
38      int p, n;
39      p = numriMëlMadh (numri1, numri2 );     //Thirrje funksionale
40      n = numriMëlMadh    ( p, numri3 );         // Thirrje funksionale
41      return n;
42  }
```

Figura 2.5.22

Një rezultat prej realizimit të programit është:



Ndryshoret globale dhe lokale

Ndryshoret të cilat krijohen te programet edhe te nënprogramet (funksionet) kanë **fushë të dukshmërisë** së caktuar (angl. scope of visibility) të quajtur edhe **fushë e qasjes** (angl. scope of access). Fusha e pamjes së ndonjë ndryshore llogaritet ajo fushë te ecila ka qasje deri te ndryshorja, d.m.th., mund të shfrytëzohet te operacionet.

Disa ndryshore mund të shfrytëzojë të gjithë aplikimin, disa në gjithë funksionin, por disa vetëm te pjesët e funksionit.

Sipas fushës së dukshmërisë ndryshoret mund të jenë:

- **Ndryshore globale.**
- **Ndryshore lokale.**

Fusha e dukshmërisë së ndryshoreve globale, përkatësisht qasja deri tea to është gjithë aplikimi, d.m.th., funksioni kryesor i të gjitha funksioneve të përkufizuara shfrytëzuese në të njëjtën datotekë te e cila është funksioni kryesor.

Emrat e funksioneve të përkufizuara shfrytëzuese në një aplikim trajtohen si ndryshore globale dhe prandaj, një funksion mund të thirret te çfarëdo qoftë tjetër funksion te aplikimi. (Funksioni në C++ nuk mund të përkufizohet te trupi i funksionit tjetër).

Ndryshoret lokale mund të kenë fushë të qasjes: vetëm trupi i funksionit te i cili deklarohen, vetëm titulli i funksionit ose vetëm blloku ndërmjet kllapave {}. Dukshmëria fillon me deklarimin e ndryshores lokale.

Parametrat e funksionit trajtohen si ndryshore lokale te gjithë funksioni. Gjithashtu, ndryshoret lokale në një funksion janë të dukshme te të gjitha urdhërat e folëzuar, por ndryshoret lokale të deklaruara te trupi i ndonjë urdhri nuk janë të dukshme jashtë prej trupit të tij.

Për një ndryshore themi se është **fshehur** (e mbuluar me tjetër ndryshore) te ndonjë bllok ose gjithë funksioni nëse nuk është e dukshme (nuk mund të shfrytëzohet) në atë bllok ose në atë funksion. Kjo do të thotë se nëse te fusha e dukshmërisë së ndonjë ndryshore (për shembull, int x = 1;) ka bllok (te urdhri) te i cii është deklaruar ndryshorja me emrin e njëjtë (int x = 2;), në atë bllok do të jetë e dukshme vetëm ndryshorja x me vlerë 2.

Shembulli 2.5.7

Me këtë shembull do ta demonstrojmë dukshmërinë e ndryshoreve te një aplikim.

Te program prej ***figura 2.5.23***, ndryshorja kyVit është deklaruar si ndryshore globale, por ndryshoret vjet dhe vitinEArdhshëm janë deklaruar si ndryshore lokale te funksioni main() dhe funksioniIm(). Deri te ndryshorja globale kyVit ka qasje edhe prej funksionit kryesor main() dhe prej funksionit funksiononiIm(). Deri te ndryshorja lokale vjet ka qasje vetëm prej main(), por nuk ka prej funksioniIm(), por deri te ndryshorja lokale vitiArdhshëm ka qasje vetëm prej funksioniIm(), por nuk ka prej main().

```
1   #include <iostream>
2   using namespace std;
3
4   void  funksionilm ();         // Prototip
5
6   int       kyVit = 2020;       //Ndryshorja globale
7
8   int main() {
9
10      int  vjet = 2019;     // Ndryshore lokale
11      cout << "\t Shtyp prej funksionit kryesor ()" << endl;
12      cout << "Ky vit është " << kyVit << ",por vjet ishte" << vjet<< endl;
13       funksionilm ();
14      // Deri te ndryshorja 'vitiArdhshëm'  nuk ka qasje në main()
15      // pasi është lokale në funksioniIm
16
17      cout << endl;
18      system( "Color 17" );
19      system( "pause" );
20      return 0;
21  }
22
23  void funksioniIm ()  {
24      int vitiArdhshëm = 2021;; // Ndryshore lokale
25      cout << "\n\tShtyp prej funksionit kryesor funksioniIm" << endl;
26      cout << " Ky vit është   " <<  kyVit      << ", por vitin e ardhshëm "
27          <<vitiArdhshëm << endl;
28      int vitiArdhshëm = 2030;
29      cout << " Prej këtu deri në fund funksionit  ndryshorja globale  'kyVit' "
30          << "\n është mbuluar me ndryshoren lokale 'kyVit' " << endl;
```

Figura 2.5.23

```
31      cout << "Këtë vit është " <<  kyVit  << "por vitin e ardhshëm eshtë "
32          <<vitiArdhshëm << endl;
33
34      cout << "Megjithatë, qasja deri te ndryshorja e mbuluar globale      'kyVit '  "
35          << "\nnuk mund me operatorin ::" << "\nKëtë vit është "
36          << ::kyVit << endl;
37      //  deri te ndryshorja 'vjet' nuk ka qasje prej funksionilm()
38      //  pasi është lokale te main()
39  }
```

Figura 2.5.23 (vazhdim)

Te shembulli, ndryshorja globale kyVit me vlerë 2020 është mbuluar (fshehur) te funksioniIm() pasi është përsëri e deklaruar si ndryshore lokale me vlerë 2030. E gjithë kjo është e qartë prej daljes së programit:

```
          Shtyp prej funksionit krysor main ()
Ky vit është 2020, por vjet ishte 2019

          Shtyp prej funksionit funksionilm ()
Ky vit është 2020, por viti ardhshëm është 2021
Prej këtu deri në fund të funksionit ndryshorja globale, 'kyVit'
është mbuluar me ndryshoren lokale, 'kyVit'
Ky vit është 2030, por viti ardhshëm është 2021
Megjithatë, qasja deri te ndryshoria e mbuluar globale 'kyVit'
është i mundshëm me operatorin ::
Ky vit është 2020

Press any key to continue . . .
```

Është interesante se ndryshorja globale kyVit(= 2020), që është mbuluar te funksioni funksioniIm() me ndryshoren lokale me të njëjtin emër kyVit(= 2030), i qaset me të ashtuquajturën **operator për zgjidhjen e fushës së dukshmërisë** (angl. scope resolution operator), shenja e të cilit është **::**.

Koha e krijimit të një ndryshore te memoria (menjëherë pas realizmit të urdhrit për deklarim) deri te zhdukja (fundi i bllokut, fundi i funksionit ose fundi i programit) quhet **jeta e ndryshores** (angl. lifetime).

Ndryshoret globale jetojnë gjithë kohën deri sa realizohet aplikimi.

Ndryshoret lokale jetojnë prej momentit të krijimit deri në fund të fushës së dukshmërisë, që mund të jetë gjithë funksioni ose vetëm blloku ndërmjet dy kllapave {}.

Mund të vendoset n këto rregulla për fushën e dukshmërisë së ndryshoreve:

- Ndryshoret lokale nuk mund të shfrytëzojnë jashtë prej fushës së dukshmërisë.
- Ndryshoret globale mund të shfrytëzohen në gjithë aplikimin
- Ndryshoret të deklaruara në një bllok (funksioni, trupi i urdhrit – for, while, do-while) mund të shfrytëzohen vetëm te ai.

- Ndryshorja e deklaruar në një funksion nuk mund të shfrytëzohet në funksion tjetër.
- Ndryshorja mund të jetë e fshehur në ndonjë pjesë prej fushës së saj të dukshmërisë nëse te ajo pjesë është deklaruar përsëri me emrin e njëjtë.
- Dy funksione me emrin e njëjtë mund të kenë fushë të njëjtë të dukshmërisë vetëm nëse kanë lista të ndryshme të argumenteve.

Ndryshoret statike

Ndryshoret mund të jenë edhe **statike** (angl. statik). Ato deklarohen me fjalën **statik.**

Për shembull:

```
static int numri = 7;
static double sasia;
sasia = 123.45;
```

Ndryshoret statike lokale qëndrojnë sikurse ndryshoret globale. Ato jetojnë prej deklarimit të tyre deri në mbarimin e aplikimit.

Poashtu, inicializohen vetëm njëherë gjatë deklarimit ose gjatë shoqërimit vlerë në thirrjen e parë të funksionit te i cili janë ndryshoret lokale. Pas realizimit të funksionit te i cili janë deklaruar edhe kthimi i veprimit te thirrësi (në main() ose tjetër funksion), ndryshoret statike lokale e ruajnë vlerën e fituar, d.m.th., nuk zhduken si ndryshore jo statike lokale. Gjatë çdo thirrje vijuese të funksionit, e kanë gjendjen (vlerën) e fituar te thirrja paraprake.

Ndryshoret globale dhe ndryshoret statike lokale fillojnë të jetojnë me realizimin e parë të përkufizimit të tyre.

Shembulli 2.5.8

Te aplikimi prej ***figura 2.5.24***, funksioni f() thirret 5 herë.

Te thirrja e parë e funksionitf(), ndryshorja statike fillimi inicializohet në vlerën 0, ndryshorja jo statike numëruesi i vlerës 100. Me urdhërat për inkrementimin, ato fitojnë vlerë 1 dhe 101.

Gjatë thirrjes së dytë të funksionit f(), ndryshorja statike fillimi nuk incicializohet përsëri (edhe pse realizohet urdhri për deklarim dhe inicializim), por (pasi është statike) e ruan vlerën 1 prej thirrjes së parë të funksionit. Prandaj, me urdhrin për inkriminim, fiton vlerë 2. Ndryshorja jo statike numëruesi përsëri inicializohet në vlerën 100 dhe pastaj inkrementohet në 101.

E njëjta përsëritet gjatë çdo thirrje të funksionit f(). Kjo do të thotë se ndryshorja statike numërues inicializohet vetëm një here (gjatë thirrjes së parë të funksionit f()) dhe jeton si ndryshore globale deri në mbarimin e aplikimit.

```
1   #include <iostream>
2   using namespace std;
3
4   void f();
5
6   int main() {
7       for( int k = 1; k <= 5; k++ )
8           f();
9
10      cout << endl;
11      system( "Color 17" );
12      system( "pause" );
13      return 0;
14  }
15
16  void f() {
17      static int fillimi = 0; // fillimi është ndryshore statike
18      int numërues = 100;
19
20      fillim += 1;
21      numërues += 1;
22      cout << "fillim =" <<fillim << " brojac=" <<numërues<< endl;
23  }
```

Figura 2.5.24

Dalja prej realizimit të aplikimit është:

```
fillimi   =1 numërrues=101
fillimi   =2 numërrues=101
fillimi   =3 numërrues=101
fillimi   =4 numërrues=101
fillimi   =5 numërrues=101

Press any key to continue . . .
```

Nëse ndryshorj numërues nuk është statike, atëherë dalja do të jetë:

```
fillimi   =1 numërrues=101
fillimi   =1 numërrues=101
fillimi   =1 numërrues=101
fillimi   =1 numërrues=101
fillimi   =1 numërrues=101

Press any key to continue . . .
```

Shfrytëzimi i ndryshoreve globale dhe statike duhet të jetë e kujdesshme pasi ato janë të qasura në të gjitha funksionet e aplikimit. Është e mundshme të vjen deri te ndryshimi i përmbajtjes së tyre të nj funksioni, port e tjetri ai të shkakton gabim.

Funksionet e integruara

Gjatë realizimit të një aplikimi, për të gjitha funksionet te ai bëhet nga një kopje te mmemoria. Është e njohur dr realizimi i çdo aplikimi fillon me funksionin main(). Nëse te ai thirret tjetër funksion, atëherë veprimi ndërpritet te vend ii thirrjes dhe kapërcen te lokacioni i memories ku gjendet funksioni i thirrur. Pas realizimit të tij, veprimi kthehet në main() dhe vazhdon me urdhrin vijues pas urdhrit për thirrje të funksionit.

Ky mekanizëm për thirrje të funksionit dhe kthimi te thirrësi (i cili mund të jetë edhe tjetër funksion përveç main()), kërkon kohë të caktuar për realizim. Në rastin kur funksioni i thirrur është i vogël, mund të ndodh koha e thirrjes së funksionit dhe kthimi prej atij të jetë më i madh prej kohës së realizimit të vet funksionit. Humbja e kohës është më e shprehur nëse funksioni i vogël thirret më shpesh, për shembull, te urdhri për përsëritje.

Në rastet e këtilla, më praktike është funksioni të integrohet në kodin e thirrësit (për shembull në main()) gjatë përkthimit të tij. (Përmendëm se funksionet e bibliotekës në C++ janë integruar funksionet).

Për këtë qëllim, në C++ shfrytëzohet kualifikatori **inline,** i cili vendohet para përkufizimit të funksionit për t'i treguar përkthyesit ta integron te programet të cilët e thirrin, në vendet te të cilët thirren.

Shembulli 2.5.9

Te program te ***figura 2.5.25*** funksioni luanBonus() është i vogël dhe realizohet shumë herë, prandaj mund të kualifikohet si funksion i integruar.

```
1   #include <iostream>
2   #include <string>
3   #include <ctime>
4   using namespace std;
5
6   inline int luanBonis ( string emriDheMbiemri) {
7       srand( time( 0 ) );
8       int bonus = 1 + rand() % 100;
9       return bonus;
10  }
11
12  int main() {
13      string emriDheMbiemri ;
14      for( int i = 1; i <= 3; i++ ) {
15          cout << "Futni emrin dhe mbiemrin tuaj";
16          getline( cin,emriDheMbiemri);
```

Figura 2.5.25

```
17          cout << "\t I nderuar " << emriDheMbiemri
18              << " bonusi juaj për këtë lojë është "
19              << igraBonus(emriDheMbiemri ) << endl;
20      }
21
22      cout << endl;
23      system( "Color 17" );
24      system( "pause" );
25      return 0;
26  }
```

Figura 2.5.25(vazhdim)

Një dalje prej programit është:

```
Futni emrin dhe mbiemrin tuaj:Aleksandar Makedonski
        I nderuar Aleksandar Makedonski bonusi juaj për këtë lojë është 41
Futni emrin dhe mbiemrin tuaj:Car Samoil
        I nderuar Car Samoil bonusi juaj për këtë lojë është 67
Futni emrin dhe mbiemrin tuaj:Goce Delçev
        I nderuar Goce Delçev bonusi juaj për këtë lojë është 90
Press any key to continue . . .
```

Detyra për ushtrime

Të shkruhen të gjitha aplikimet dhe funksionet n për këto detyra:

1. Të njehsohet shuma $S_n = a_1 + (a_1 + a_2) + (a_1 + a_2 + a_3) + ... + (a_1 + a_2 + ... + a_n)$. Të shfrytëzohet funksioni për njehsimin e shumës $S_k = a_1 + a_2 + ... + a_k$.
2. Të njehsohet shuma $S_n = \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots + \frac{1}{n!}$ e shfrytëzimin e funksionit për njehsimin k!.
3. Të shkruhen funksionet e veçanta për njehsimin e numrit dhe të shumës së shifrave të numrit natyror n.
4. Të gjenden numrat e thjeshtë më të vegjël se n. Të shkruhen funksionet për caktimin ndonjë numër natyror a është i thjeshtë ose jo.
5. Të shkruhet funksioni i llojit void për konversionin e numrit real dekad në binar.
6. Të shkruhet funksioni i llojit void për zgjidhjen e barazimit katror $ax^2 + bx + c = 0$.
7. Të shkruhet funksioni i llojit void për gjetjen e shënimit binar, oktal dhe heksadhjetor të numrit dekad n.
8. Të shkruhet funksioni i llojit void për gjetjen e numrit më të madh dhe më të vogël të cilët mund të formojnë prej shifrave numrin natyror n.

9. Të gjenden të gjithë numrat treshifror të cilët janë të barabartë me shumën e faktorielëve të shifrave të tij. Për njehsimin e faktorielëve të shifrave, të shfrytëzohet funksioni i veçantë.
10. Të gjenden të gjithë numrat miqësor në intervalin m deri në n. (Për dy numra themi se janë numra miqësor nëse shuma e pjesëtuesve të numrit të parë është i barabartë me numrin e dytë, por shuma e pjesëtuesve të numrit të dytë është e barabartë me numrin e parë). Për shumën e pjesëtuesve të numrit të shkruhet funksion i veçantë.

Pyetje për kontroll të njohurive

1. Si quhet biblioteka me funksioneve në C++?
2. Përmend ndonjë heder-datotekë prej bibliotekës së C++.
3. Çka duhet të kyçet te program nëse duam të shfrytëzojmë funksion prej ndonjë hder-datoteke?
4. Në cilën bibliotekë të C++ gjenden funsionet për lexim të të dhënave të future nëpërmjet tastaturës?
5. Me cilin funksion të bibliotekës njehsohet x^n, por me cilën $\sqrt{x}$?
6. Cili është funksioni në C++ për gjenerimin e numrit të rastit?
7. Me cilin funksion shndërrohen të gjitha shkronjat prej një fjale në të mëdha?
8. Me cilin funksion kontrollohet ndonjë shenjë a është shifër, por me cilën a është shkronjë?
9. Me cilin operator caktohet madhësia e memories që e zen ndonjë ndryshore?
10. Cila është përparësia e të shkruarit të pjesve të algoritmeve te nënalgoritmet?
11. Si mund të jenë nënalgoritmet? Cila është karakteristika kryesore e tyre?
12. Si thirren nënalgoritmet funksionale, por si proceduralet?
13. Çka do të thotë
    ktheje ←100;
    nëse është shkruar në nënalgoritëm funksional?
14. Çfarë lloje të funksioneve ekzistojnë në C++, sipas asaj a kthejnë një ose më shumë vlera?
15. Shkruani formën e përgjithshme të përkufizimit të funksionit me vlerë kthyse.
16. Si kryhet deklarimi i funksionit me vlerë kthyese? Shkruani formën e përgjithshme.
17. Çfarë është lloj i funksionit?
18. Cili është roli i prototipit funksional?
19. Prej çka përbëhet nënshkrimi funksional?
20. Funksionet të përkufizuara para funksionit main() a duhet të përmendet prototipi funksional?
21. Përmend disa shembuj të prototipit funksional.
22. Përmend disa shembuj të nënshkrimit funksional.

23. Në çfarë mënyrash mund të thirren funksionet me vlerë kthyese?
24. Çka do të ndodh nëse numri i argumenteve te thirrja e funksionit me vlerë kthyese është i ndryshëm prej numrit të parametrave te përkufizimi i funksionit?
25. Çka ndodh nëse lloj i ndonjë argument te thirrja nuk bënë pjesë me llojin e prametrit përkatës te përkufizimi i funksionit?
26. Ku është gabimi te këto segment programor:

```
int f(void) {

   cout << "Thirrje e funksionit f()";
   int g(void) {
          cout << "Thirrje e funksionit ()";
   }
}
```

27. Shkruani funksion me vlerë kthyese e cila për futjen e anës e njehson vëllimin e kubit.
28. A mund funksioni me vleë kthyese të thirret në urdhër për shtypje cout?
29. Çfarë parametra mund të kenë funksionet pe vlerë kthyese?
30. Prej cilit lloj janë funksionet në C++ cilët nuk kthejnë vlerë?
31. Funksionet pa vlerë kthyese a mund ta përmbajnë urdhrin return?
32. Në çfarë mënyra mund të kryhet thirrja e funksioneve me vleë kthyese, por me cilët funksione pa vlerë kthyese?
33. Sqaro se si barten në funksion argumentet sipas vlerës, por si sipas referencës.
34. Si shënohen parametrat e referencës?
35. Te prototipi i funksionit pa vlerë kthyese patjetër të shënohen parametrat referent?
36. Cili është ndryshimi ndërmjet argumentit të bartur sipas referencës dhe parametrit referent përkatës?
37. Funksioni a mund të ketë parametra referente?
38. Në cilat raste shfrytëzohen funksionet me vlerë kthyese, por në cilat funksione pa vlerë kthyese?
39. Ku është gabimi te ky funksion pa vlerë kthyese?

```
void P(int a) {

   int b;
   b = a++;
   return b;
}
```

40. Çka do të ndodh nëse përpiqemi ta ndryshojmë vlerën e parametrit konstant te funksioni?
41. A mund një funksion të thirr tjetër funksion?
42. A mund te një funksion të përkufizohet tjetër funksion?

43. Çfarë janë ndryshoret globale, kurse çka janë lokale?
44. Cila është fusha e dukshmërisë së ndryshoreve globale, por cila e ndryshoreve lokale?
45. Çfarë ndryshore janë emrat e funksioneve në një aplikim, globale ose lokale?
46. Cila është fusha e dukshmërisë së parametrave të funksionit?
47. Sqaro si mund ndonjë ndryshore globale të fshihet (të mos jetë e qasur) te ndonjë funksion? A ka mënyrë, megjithatë, të shikohet?
48. Çfarë është jeta e ndryshores?
49. Cili është ndryshimi në sjelljen e ndryshores globale dhe të ndryshores statike?
50. Me cilën fjalë shënohen funksionet e integruara?

Detyra

Të zgjidhen të gjithë shembujt dhe detyrat prej pikës **2.3 Struktura kontrolluese algoritmike për pëesëritje dhe urdhëra kontrollues për përsëritje**, me shfrytëzimin e funksioneve.

Për këto detyra të shkruhen programe me shfrytëzimin e funksioneve:

1. Të gjenden sa ka shkronja të mëdha, shkronja të vogla dhe shenja të interpuksionit te një fjali.
2. Të shndërrohen koordinatat e Dekartit në pikë (x, y) në polare (ρ, φ), sipas formulave: $\rho = \sqrt{x^2 + y^2}$, $\varphi = arctg\,\frac{y}{x}$.
3. Të gjendet PMP për dy numra.
4. Të gjendet ekuivalenti binary i numrit të plotë dekad negativ.
5. Të shkruhet funksioni me të cilin do të futen n numra dhe të gjendet numri më i vogël dhe më i madh prej tyre.
6. Të gjendet numri i thjeshtë më i madh dhe më i vogël prej numri natyror të dhënë n.
7. Të shkruhet funksioni për mbledhje të dy kohërave, të shprehura nëpërmjet orëve (0 – 23), minuta (0 – 59) dhe sekonda (0 – 59).
8. Të shkruhet funksion të llojit void për njehsimin e moshës së njeriut (ndryshimi prej dates së sotshme të lindjes). Datat të futen si ndryshore numra të plotë: dita (1 – 31), muaj (1 – 12), viti (1– 2345).

Termine

- **Struktura kontrolluese (menaxhuese) algoritmike** shërbejnë për kontrollë të realizimit të algoritmit.
- **Struktura kontrolluese algoritmike rendore** ose **sekuenca** është strukture algoritmike te e cila hapat realizohen sipas asaj renditje sipas të cilës janë përmendur.
- Struktura kontrolluese algoritmike rendore quhet **fillim–fund.**
- **Struktura kontrolluese algoritmike për zgjedhje prej dy bashkësive** është struktura për zgjedhje të njërës prej dy kaheve të mundshme të vazhdimit të veprimit në algoritëm, varësisht prej ndonjë kushti (shprehje logjike).
- **Bllok algoritmik** është bllok prej më shumë hapave të shënuara me fjalët **fillim** dhe **fund.**
- **nëse–atëherë–ndryshe** është strukturë kontrolluese algoritmike për zgjedhje prej dy mundësive.

∇ **nëse–atëherë** është strukturë kontrolluese algoritmike për zgjedhje prej dy mundësive, ku njra prej mundësive nuk ka hap realizues.

∇ **if** është urdhër kontrollues në C++ për realizimin e strukturës kontrolluese algoritmike **nëse–atëherë.**

∇ **if-else** është urdhër kontrollues në C++ për realizimin e strukturës kontrolluese algoritmike nëse–atëherë–ndryshe.

∇ **?:** quhet operator i kushtëzuar.

∇ **Struktura kontrolluese algoritmike për zgjedhje prej më shumë mundësive** është struktura te e cila kryhet zgjedhje e njërit prej më shumë rasteve të mundshme të vazhdimit të veprimit në algoritëm, varësisht prej vlerës së ndonjë datoteke ose të ndonjë shprehje aritmetike.

∇ **rasti** është strukturë kontrolluese algoritmike për zgjedhje prej më shumë mundësive.

∇ **switch** është urdhër kontrollues në C++ për realizimin e strukturës kontrolluese algoritmike për zgjedhje prej më shumë mundësive **rast.**

∇ **Cikël** është një realizim i grupit të hapave algoritmik.

∇ **Përsëritja ciklike** është realizimi i grupit të njëjtë të hapave algoritmike shumë herëshe.

∇ **për–deri–hap** është strukturë kontrolluese algoritmike te e cila ciklet numërohen prej vlerës së fillimit deri në fund me sasi të hapit.

∇ Nëse te struktura kontrolluese algoritmike **për–deri–hap** sasia është +1, struktura quhet **për–zmadho–deri.**

∇ Nëse te struktura kontrolluese algoritmike **për–deri–hap** sasia është –1, struktura quhet **për–zvogëlo–deri**.

∇ **for** është urdhër kontrollues në C++ për realizimi e strukturave kontrolluese për–zmadho–deri, për–zvogëlo–deri dhe për–deri–hap, me inkrementim të

numëruesit sipas 1, dekrementimi i numëruesit sipas 1 dhe zmadhimi/ zvogëlimi i numëruesit për sasi të caktuar.

- ++ është operator për inkrementim.
- – – është operator për dekrementim.
- **deri–realizo** është strukturë kontrolluese algoritmike te e cila përsëritja e cikleve kryhet deri sa është plotësuar ndonjë kusht, i cili shqyrtohet para fillimit të çdo cikli.
- **while** është urdhër kontrollues në C++ për realizimi e strukturës kontrolluese algoritmike deri–realizo.
- **realizo–deri** është strukturë kontrolluese algoritmike te e cila përsëritje e cikleve kryhet deri sa është plotësuar ndonjë kusht, që shqyrtohet në fund prej çdo cikli.
- **do-while** është urdhër kontrollues në C++ për realizimin e strukturës kontrolluese algoritmike realizo–deri.
- **vazhdo** është strukturë kontrolluese algoritmike për kapërcim në fund të ciklit dhe vazhdimi me ciklin vijues.
- **continue** është urdhër kontrollues në C++ për realizimin e strukturës kontrolluese algoritmike **vazhdo.**
- **ndërprerje** është strukturë kontrolluese algoritmike për kapërcim në fund të strukturës kontrolluese dhe ndërpritja e përsëritjes.
- **break** është urdhër kontrollues në C++ për realizimin e strukturës kontrolluese algoritmike **ndërprerje.**
- **dalje** është strukturë kontrolluese algoritmike për kapërcim në fund të algoritmit.
- **exit**() është urdhër kontrollues në (funksion) në C++ për realizimin e strukturës kontrolluese algoritmike **dalje.**
- **kapërcim** është strukturë kontrolluese algoritmike për kapërcim të çfarëdo hapi te algoritmi.
  goto është urdhër kontrollues në C++ për realizimin e strukturës kontrolluese algoritmike kapërcim.
- **Biblioteka standard** e C++ është biblioteka e cila përmban funksione matematikore, funksione për hyrje dhe dalje, për operacione me shenja, me stringe etj.
- **Funksione me vlerë kthyese** janë ato funksione të cilat pas realizimit kthejnë vlerë në vendin prej ku janë thirrur.
- **Funksione pa vlerë kthyese** janë ato funksione të cilat pas realizimit kthejnë vlerë në vendin prej ku janë thirrur.
- **Programimi prej lart poshtë** është teknika e zbërthimit të detyrës në detyra më të thjeshta, të quajtura **nëndetyra.**
- **Programimi modular** është teknika për përpunimin e programeve (moduleve) për nëndetyrat dhe realizimi i tyre sekuenciale.
- **Nënalgoritme** është algoritëm i cili nuk mund të realizon (sipas kodimit) në mënyrë të pavarur.

- **Nënalgoritmet funksionale** të quajtura edhe **funksione** kanë vetëm *një rezultat dalës*.
- **Nënalgoritmet procedurale** të quajtura edhe **procedura** kanë *më shumë rezultate dalëse*.

- **Parametra (argumente formale**) shfrytëzohen te përkufizimi i nënalgoritmeve.
- Argumente (argument të vërteta) janë ndryshore të cilat shfrytëzohen gjatë thirrjes së nënalgoritmeve.
- **Parametrat hyrëse (argument formale hyrëse)** janë ato nëpërmjet të cilëve barten vlerat prej thirrësit (algoritëm ose nënalgoritëm) te nënalgoritmet. Ato shënohen me shenjën ↓.
- **Parametra dalës (argument formale dalëse)** janë ato nëpërmjet të cilëve barten vlerat prej nënalgoritmit deri te thirrësi (algoritëm ose nënalgoritëm). Ato shënohen me shenjën ↑.
- **Parametra hyrëse-dalëse (argument formale hyrëse-dalëse**) janë ato nëpërmjet të cilëve barten vlerat prej thirrësit (algoritëm ose nënalgoritëm) në nënalgoritëm dhe e kundërta. Ato shënohen me shenjën ↕.
- **kthej** është hapi me të cilin kthehet rezultati prej nënalgoritmit funksional.
- **Lista e parametrave** është lista e titullit të përkufizimit të nënalgoritmeve.
- **Lista e argumenteve është lista te thirrja e nënalgoritmit.**
- **Nënprogrami funksional** i quajtur **funksion** është program te gjuha programore e shkruar për nënalgoritmin funksionale.
- **Funksione të përkufizuara shërbyese** ose vetëm **funksion shërbyes** janë funksione të shkruara prej programuesve
- **Përkufizimi i funksionit** në C++ përbëhet prej titullit (i cili përbëhet prej llojit tëvlerë kthyese, emri i funksionit dhe lista e parametrave) dhe trupi (në kllapa tëmëdha).
- **Deklarimi i funksionit** me vlerë kthyese në C++ përbëhet prej llojit (të vlerës kthyese), emri (i funksionit) dhe lista e parametrave.
- **Prototip funksional** quhet deklarimi i funksionit, por përbëhet prej llojit, emri dhe lista e parametrave.
- **Nënshkrimi funksional** përbëhet prej emrit të funksionit dhe liste e parametrave.
- **Parametrat konstant** janë ato vlerat e të cilave nuk mund të ndryshojnë te funksioni.
- **void** është lloj i funksionit në C++ të cilët nuk kthejnë vlerë.
- **Argumentet e bartura sipas** vlerës janë ato të cilët nuk barten te funksioni me thirrje, por vlera e tyre te thirrësi (funksioni main() ose tjetër funksion) nuk ndryshon. Mekanizimi quhet bartja e argumenteve.
- **Referencimi** është shqërimi i emrit tjetër të ndryshores
- **Referenca** ose **ndryshorja referente** është tjetër emër e ndonjë ndryshore

- **Argumetet të bartur sipas referencës** janë ato të cilët barten në funksion me thirrje, ku ato janë meër tjetër të parametrave referentë përkatës. Mekanizimi quhet **bartje të argumenteve sipas referencës.**
- **Fusha e dukshmërisë** të quajtur edhe **fusha e qasjes** është ajo fushë te e cila ka qasje deri te ndryshoret, d.m.th., mund të kryejnë operacione me atë.
- **Ndryshore globale** janë ato deri te të cilat ka qasje prej gjithë aplikimit, d.m.th., prej funksionit kryesor dhe prej të gjitha funksioneve të përkufizuara shërbyese në datotekën e njëjtë te e cila është funksioni kryesor.
- **Ndryshoret lokale** janë ato të cilat kanë fushë të qasjes vetëm trupi i funkionit te e cila janë deklaruar ose vetëm titulli i funksionit ose vetëm blloku te i cili janë deklaruar.
- **Ndryshorja e fshehur** (e mbuluar me ndryshore tjetër) te ndonjë bllok ose te gjithë funksioni është ajo e cila nuk është dukshme (nuk mund të shfrytëzohet) në atë bllok ose në atë funksion.
- **::** është **operator për zgjidhje të fushës së dukshmërisë.**
- **Jeta e ndryshores** është koha prej krijimit në memorien (menjëherë pas realizimit të urdhrit për deklarim) deri te zhdukja e tij (fund ii bllokut, fund ii funksionit ose fundi i aplikimit).
- **Ndryshorja statike** është ajo e cila prej momentit të deklarimit sillet si ndryshore globle.
- **Funksion i integruar** është a ii cili përkthen së bashku me thirrësin (main() ose tjetër funksion) dhe integrohet në kodin e tij.
- **inline** është kualifikator për funksione të integruara.

Përmbledhje

- Për përshkrim të rrjedhës së algoritmeve gjatë programimit të strukturuar, shfrytëzohen të veçanta të ashtuquajtura struktura kontrolluese (menaxhuese) algoritmike me të cilat menaxhohet me renditjen e realizimit të hapave algoritmik.
- Rrjedha e çdo algoritmi mund të kontrollohet me shfrytëzimin e strukturës kontrolluese algoritmike: **rendore** ose **sekuenca, zgjedhje** ose **selektim** dhe përsëritje ose **iterim.**
- Te strukturës kontrolluese algoritmike rendore hapat realizohen sipas renditjes sikurse është përmendur. Ajo quhet **fillim–fund.**
- **nëse–atëherë–ndryshe** është struktura kontrolluese algoritmike për zgjedhje prej dy mundësive, por në C++ realizohet me urdhrin kontrollues if-else.
- **nëse–atëherë** është struktura kontrolluese algoritmike për zgjedhje prej dy mundësive nuk përmban hap realizues, por në C++ realizohet me urdhrin kontrollues if.

- Urdhërat kontrollues if dhe if-else mund të folezohen edhe në zgjedhjen if dhe në zgjedhjen else. Urdhri kontrollues i folezuar te zgjedhja else quhet if-else-if.
- Operatori i kushtëzuar?: shfrytëzohet për paraqitjen e urdhërave të zakonshëm if-else.
- Ku ka më shumë mundësi për vazhdimin e veprimit te algoritmi, shfrytëzohet struktura kontrolluese algoritmike për zgjedhje prej më shumë mundësive **rasti.**
- Zgjedhja e njërit prej mundësive kryhet, varësisht prej vlerës të ndonjë të dhëne ose të ndonjë shprehje algoritmike
- Struktura kontrolluese algoritmike për zgjedhje prej më shumë mundësive **rasti**, në C +
  realizohet me urdhrin kontrollues switch.
- Struktura kontrolluese algoritmike për përsëritje shfrytëzohet atëherë kur është e nevojshme një grup të hapave algoritmik të kryhen më shumë here, e cila quhet përsëritje ciklike. Një realizim i hapave quhet një cikël.
- Dallojmë tri struktura kontrolluese algoritmike për përsëritje, edhe atë: përsëritje me numërim të cikleve (**për–deri–hapi)**, përsëritja me dalje në fillim prej ciklit (deri–realizo) dhe përsëritja me dalje në fund prej ciklit (**realizo–deri)**.
- Te struktura kontrolluese algoritmike **për–deri–hapi**, ciklet numërohen e numërues të veçantë, me ndonjë sasi të hapit prej vlerës fillestare deri në fund.
- Nëse sasia e hapit është +1, struktura quhet **për–zmadho–deri**, nëse sasia është –1, struktura quhet **për–zvogëlo–deri.**
- Për realizimin e strukturës kontrolluese algoritmike **për–deri–hapi** (**për–zmadho–deri** dhe **për–zvogëlo–deri**), në C++ shfrytëzohet urdhri kontrollues for.
- Urdhri kontrollues for ka pjesë për inicializimin, pjesë për kusht dhe pjesë për azhurim.
- Urdhri Urdhri kontrollues for ka pjesë për inicializimin, pjesë për kusht dhe pjesë për azhurim. kontrollues for në C++ lejon inicializimin dhe azhurimin e më shumë ndryshoreve.
- Te urdhri kontrollues for pjesa për inicializim dhe azhurimin mund dtë jetë edhe e zbrazët.
- Struktura kontrolluese algoritmike **deri–realizo,** në C++ realizohet, me urdhrin kontrollues while. Deri sa është plotësuar ndonjë kusht logjik në while, ciklet realizohen, Kushti shqyrtohet para fillimit të çdo cikli.
- Me urdhrin kontrollues while mund të mos realizohet asnjë cikël nëse kushti nuk është plotësuar qysh para fillimit të ciklit të parë.
- Kontrollimi i përsëritjet me ndonjë vlerë prej përpara të njohur quhet *përsëritje e kontrolluar me roje.*

- Struktura kontrolluese algoritmike **realizo–deri,** në C++ realizohet me urdhrin kontrollues do-while. Ciklet në do-while realizohen deri sa është plotësuar ndonjë kusht logjik. Kushti shqyrtohet pas barimit të ndonjë cikli.
- Me urdhrin do-while, patjetër të kryhet të paktën një cikël.
- Gjatë folezimit të urdhërave kontrollues për përsëritje (for, while, do-while), të gjitha urdhërat e njërit urdhër kontrollues patjetër të gjendet te tjetra, përkatësisht nuk guxon të ketë prerje (mbivendosje) të urdhërave kontrollues.
- Te gjuhët programore ekzistojnë katër lloje të strukturave kontrolluese algoritmike për kapërcim, edhe atë: kapërcim në fund të ciklit – **vazhdo,** kapërcim në fund të strukutrës kontrolluese – ndërprerje, kapërcim në fund të algoritmit – **dalje dhe kapërcim të çfarëdo vendi – kapërcim.**
- Struktura kontrolluese algoritmike për kapërcim, në C++ realizohet për urdhërat kontrollues: continue, break, exit() dhe goto.
- Shmanget shfrytëzimi i urdhrit kontrollues goto në programimin e strukturuar.
- Në programimin e strukturuar shfrytëzohen dy teknika për programim, edhe atë: programim prej lart dhe programim modular. Programimi prej lart poshtë kryhet me ndarje (zbërthim) të detyrës në nëndetyra. Për çdo nëndetyrë mund të shkruhet algoritëm i veçantë, i cili quhet nënalgoritëm.
- Thelbi i nënalgoritmeve është në atë që atom und të shfrytëzohen në algoritme të ndryshme ose nënalgoritme, por edhe në shumë vende në algoritmine njëjtë dhe /ose nënalgoritëm.
- Shfrytëzimi i nënalgoritmeve është mundësuar nëpërmjet mekanizmit të parametrave (argument formal) dhe argument (argument të vërtet).
  Parametrat mund të jenë hyrës, dalës dhe hyrës-dalës.
- Sipas numrit të rezultateve dalëse, ekzistojnë dy nënalgoritme, edhe atë: nënalgoritme funksionale (të cilët kthejnë vetëm një rezultat) dhe nënalgoritme procedural (të cilët mund të kthejnë më shumë rezultate).
- Nënalgoritmet te gjuhët programore realizohen e nënprograme. Nënprogramet mund të jenë funksionale (kur kthejnë një vlerë) dhe procedural (kur kthejnë më shumë vlera).
- Nënprogramet në C++ realizohen me funksione. Nënprogramet funksionale realizohen me funksione me vlerë kthyese, por nënprogramet procedural realizohen me vlerë kthyese funksionale.
- Funksionet në C++ mund të jenë: funksione biblioteke dhe funksione të përkufizuara shërbyese.
  Gjatë përkufizimit të funksionit me vlerë kthyese në C++, duhet të dihet se:

- titulli i funksionit nuk mbaron me ; (pikëpresje),
- lista e parametrave patjetër t'i përmban edhe emrat dhe llojet e parametrave,
- vlera kthyese është rezultat prej funksionit i cili mund të jetë shprehje (prej ndryshoreve dhe/ose konstantave dhe/ose funksioneve) ose një ndryshore dhe e cila patjeër të jetë i llojit të njëjtë me llojin e funksionit. .

- Për deklarimin e funksionit në C++, duhet të dihet:
- Nëse nuk përmendet lloj i funksionit, nënkuptohet lloj int.
- Emrat e funksioneve (sipas konventës) shkruhen me shkronjë të vogël fillestare. Nëse emri përbëhet prej më shumë fjalëve çdo fjalë vijuese shkruhet me shkronjë të madhe.
- Lista e parametrave patjetër t'i përmban llojet e parametrave, por emrat jo patjetër.
- Deklarimi i funksionit mbaron me shenjën ; (pikëpresje).
- Gjatë thirrjes së funksionit, lista me argument dhe lista me parametra të funksionit patjetër të kenë:
- numër të njëjtë të argumenteve dhe parametrave,
- renditje të njëjtë të argumenteve dhe parametrave,
- të njëjtë ose lloj konvertibil të argumenteve dhe parametrave përkatës.
- Funksioni me vlerë kthyese në C++ thirret me emrin dhe lista e argumenteve ose me urdhër për shqërim ose me urdhër tjetër ose shprehje.
- Funksioni pa vlerë kthyese në C++ thirret vetëm me emrin e vet dhe lista e argumenteve.
- Funksionet të cilët janë përkufizuar sipas programit kryesor, deklarohen (duke përmendur prototipin e tyre) para funksionit kryesor main().

- Një funksion mund vetëm njëherë të përkufizohet në programin e njëjtë.
- Për t'u mbrojtur parametrat e funksionit prej ndryshimeve në arë, ato shënohen si konstante me kualifikatorin const.
- Funksionet pa vlerë kthyese në C++ kanë llojin **void.**
- Prej funksionit me vlerë kthyese dilet me urdhrin return me parametër, deri sa prej funksionit pa vlerë kthyese dilet automatikisht pas realizimit të tij ose prej çfarëdo vendi te trupi i funksionit me urhërin return pa parametër.
- Gjatë thirrjes së funksionit, shfrytëzohet mekanizmi për bartjen e argumenteve sipas vlerës ose sipas referencës.
- Gjatë bartjes së argumenteve sipas vlerës, ato nuk ndryshojnë te funksioni, m.th., ngelin të njëjtë edhe pas kthimit te thirrësi (funksioni kryesor main() ose tjetër funksion).
- Gjatë bartjes së argumenteve sipas referencës, atom und të ndryshojnë te funksioni dhe ps kthimit te thirrësi (funksioni kryesor main() ose tjetër funksion) mund të jenë të ndryshueshëm.

- Gjatë bartjes së argumenteve sipas referencës, nk krijohen ndryshore të reja për argument, por argumentet prej thirrësit paraqesin emra të tjerë të parametrave referent të funksionit. Të gjitha ndryshimet te parametrat e referencës, në realitet, kryhen mbi argumentet përkatës pasi ato janë emra të ndryshëm të lokacionit të memories së njëjtë.
- Nëse nuk duam të ndryshojë ndonjë argument te funksioni, pavarësisht a është bartur sias vlerës ose sipas referencës, duhet të shënohet si konstant me kualifikatorin const.
- Funksionet me vlerë kthyese mund të jenë argument të funksioneve të tjera.
- Ndryshoret globale janë ato të cilat janë të qasura në gjithë aplikimim dhe te funksionet të cilat thirren te ajo.
- Emrat e funksioneve të përkufizuara shfrytëzuese në një aplikim trajtohen si ndryshore globale.
- Ndryshoret lokale janë ato të cilat janë të qasura (të dukshme) dhe mund të shfrytëzohen vetëm te funksioni ose blloku në të cilin janë deklaruar.
- Dukshmëria e ndryshores lokale fillon me deklarimin e vet.
- Parametrat e funksionit trajtohen si ndryshore lokale në gjithë funksionin.
- Ndryshoret e deklaruara në një bllok funksioni, trupi i urdhrit – if, switch, for, while, do-while etj.) mund të shfrytëzohen vetëm te ati pasi janë ndryshore lokale.
- Ndryshorja e deklaruar në një funksion nuk mund të shfrytëzohet te funksioni tjetër.
- Dy funksione me ermine njëjtë mund të kenë fushë të njëjtë të dukshmërisë vetëm nëse kanë lista të ndryshme të parametrave.
- Gjatë realizimit të një aplikimi, për të gjitha funksionet bëhet nga një kopje te memoria.

MODULI

LLOJET E TË DHËNAVE TË PËRBËRA NË C++

3

Në këtë kapitull do të njiheni me këto:

- Vargjet dhe domethënia e tyre te programimi.
- Deklarimi, inicializimi dhe shoqërimii vlerave të elementev të vargut njëdimensional.
- Urdhri for i bazuar në vargun dhe shfrytëzimi i tij.
- Deklarimi, inicializimi dhe shoqërimii vlerave të elementev të vargut dydimensional.
- Njehsimi i dimensioneve të vargut dydimensional.
- Treguesë dhe rëndësia e tyre te programimi.
- Shfrytëzimi i operatorit të adresës & dhe operacioni për dereferencim *.
- Lidhja ndërmjet vargjeve dhe treguesve.
- Shfrytëzimi i urdhërave new dhe delete.
- Fuksionet për punë me stringe.
- Funksionet për konvertim të string te numri dhe për konvertimin e numrit të stringëve.

Fjalë kyçe

#define.	Konstanta e emërtuar
<string>	Operatori indirekt (operator për dereferencim)*
delete	Matricaa
new	Lista inicializuese
stoi, stol, stoll, stoul, stoull, stof, stod, stold	Urdhri for i bazuar në varg
to_string()	Vargu
Operatori i adresësë &	Tregues
Vargu dydomenzional	Ndryshorja e treguesit
Vargu njëdimensonal	Konstanta simbolike
Vargu njëdimensonal	

3.1 Vargjet njëdimensionale

Ekzistojnë provleme për zgjidhjen e të cilave është e nevojshme të futet numër i madhi ndryshoreve. Për shembull, për të njehsuar suksesin mesatar të një nxënësi, është nevojshme të futen notat nga të gjitha lëndët dhe për çdo note të futet nga një ndryshore numër të plotë, sikurse

nota_1 nota_2 . . . nota_n

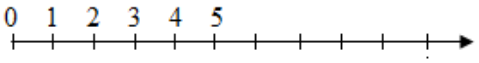
Nëse supozojmë se ka 12 lëndë, janë të nevojshme 12 ndryshore për notat e një nxënësi. Nëse në klasë ka 30 nxënës, atëherë janë të nevojshme 12 x 30 = 360 ndryshore për notat e të gjithë nxënësve. Programi me kaq numër të ndryshoreve jo vetëm që është e pa shqyrtuar por do të jetë shumë e gjatë. Gjatë të shkruarit e tij, shumë kohë do të shpenzohet për të shkruar emrat e ndryshoreve se sa për vet aplikimin.

Në rastet e këtilla, kur punohet me numër të madh të të dhënave të llojit të njëjtë, ato organizohen në struktura të veçanta rë quajtura **vargje** (angl. arrays). Elementet e vargut mund të jenë i cilit do lloj, por të gjitha elementet patjetër të jenë të llojit të njëjtë. Për shembull, të gjitha notat e nxënësve te vargu i emërtuar me nota, te shembulli i përmendur, patjetër të jenë të llojit numër i plotë.

Çdo element te vargu ka numrin e tij rendor. Për shembull, te vargu nota, emrat e elementeve janë: nota_1, nota_2… nota_n. Ato dallohen vetëm sipas numrave rendor 1, 2, 3... n. Madhësitë e këtilla në matematikës shkruhen në atë mënyrë që numrat rendor vendohen si indekse: nota1, nota2... notan. Indeksi e paraqet numrin rendor të elementeve te vargu: 1 – elementi i parë, 2 – elementi i dytë etj. n e paraqe elementi n.

Të përkujtohemi për vargjet aritmetike he gjeometrike. Vargu a_1, a_2... a_n quhet varg aritmetik nëse ndryshimi i çfarëdo dy elementeve fqinje është i njëjtë, port e vargu gjeometrik herësy i çfarëdo dy elementeve fqinje është i njëjtë. Gjithashtu, vërejmë se të gjithë elementet kanë emrin e njëjtë, por indeksat i kanë 1, 2, 3... n.

Indeksat mund të hënohen si pika në boshtin numerik, d.m.th., në një drejtim, një dimension. Gjatësia është një dimension. Syprina (drejtkëndëshit) ka gjatësi dhe gjerësi, domethënë dy dienzione. Hapësira (ndërtesa) ka tre dimensione: gjatësia, gjerësia dhe lartësia.



Prandaj, vargjet me një indeks (një dimension) quhen **vargje njëdimensionale** (angl. one dimensional arrays).

Nëse e dimë se vargu ka n elemente, atëherë shkruhet me: a_1, a_2... a_n. Ose, shkurt me $a[_i]_n$ ose vetëm me $a[]_n$, që do të thotë se indeksi i ka vlerë prej 1 deri n. Shfrytëzohet edhe shenja vijuese $[a]_n$, ku me n shënohet se vargu është njëdimensional sge se indeksat prej 1 deri n. Në literaturl shfrytëzohet edhe shënimi a[1..n]. Shënimin e fundit do ta shfrytëzojmë te algoeiret.

Në gjuhët programore, ndryshoret që fitojnë vlera të të hënave prej vargut njëdimensional shkruhen në atë mënyrë që indeksi vendohet në kllapa tmesme: a[1], a[2]... a[n] ose nota[1], nota[2]... nota[n]. Edhe pse emri i të gjitha elementeve është i njëjtë, ato janë ndryshore të ndryshme të cilat dallohen sipas indeksit.

Te shumë gjuhë programore indekset fillojnë prej 0 deri te n – 1. Vargjet teo C++ shënohen në të njëjtën mënyrë, ku indekset janë numra të plotë jonegativ, por indeksi i parë gjithmonë është 0. Për vargun prej n elementeve, indekset janë 0, 1, 2, 3... n – 1. Për shembull, për vargun prej n elementeve, shenja është a[]n, por elementet janë: a[0], a[1]... a[n – 1]. Nëse duam ta shënojmë vetëm vargun, pa numrin e elementeve, atëherë do të shfrytëzojmë shënimin a[] për të dalluar prej ndryshores me emrin e njëjtë.

Deklarimi i vargjeve njëdimensional

Vargjet njëdimensionale si ndryshore në C++ deklarohen në këtë mënyrë

```
lloj emri[numër];
```

lloj – është lloj i elementeve të vargut (short, int, long, long long, float, double, char, boolean, string etj.). emri – është emri i vargut.
emri – është emri i vargut.
numër – është numri i elementeve të vargut dhe ai patjetër të jetë literal ose konstante e emërtuar më e madhe se 0.

Shembuj

Shembulli 3.1.1

```
int a[5];
```

Vargu a[] ka 5 elemente dhe atë: a[0], a[1], a[2], a[3] dhe a[4]. Lloj i elementeve të vargut është int dhe te ato mund t'u shoqëroj vetëm vlera numra të plotë.

Shembulli 3.1.2

```
float br[25];
```

float nr[25]; Vargu nr[] është i llojit float dhe ka 25 elemente. Elementi i përgjithshëm k shënohet me nr[k], por indeksi k mund të ketë vlera 0, 1... 24. Domethënë elementet janë: nr[0], nr[1]... nr[24].

Shembulli 3.1.3

```
string embri[15], mbiemri[30];
```

Deklarohen dy vargje emri[] dhe mbiemri[] elementet e të cilit mund të kenë vlera vetëm stringa.

Shembulli 3.1.4

```
const int NUMRIELEMENTEVE = 100;
char c[NUMRIELEMENTEVE];
```

Te shembulli, numri i elementeve të vargut c[] është paraprakisht i dhënë si konstante e emërtuar (angl. named konstant), të quajtur edhe ndryshorja konstante (angl. konstant variable), `NUMRIELEMENTEVE`.

Kjo mënyrë e dhënies së numrit të elementeve shfrytëzohet më së shpeshti pasi konstanta e emërtuar inicilizohet njëherë dhe nuk mund të ndryshojë më tej te aplikimi.

Shembulli 3.1.5

```
int k = 10;
float u[k];
```

Kjo deklaratë është e gabuar pasi numri i elementeve patjetër të jetë konstante.

Për të shënuar elementin specifik të vargut, i japim emrin e vargut dhe pozitën e atij elementi te vargu.Te ***tabela 3.1*** është dhënë vargu numër të plotë me emrin d. Ky varg përbëhet prej 10 elementeve. Deri te çdonjëri prej këtyre elementeve mund të qaset nëpërmjet emrit të vargut dhe indeksit të vendosur në kllapa të mesme

Kështu, elementii i parë është d[0], por elementi i është d[i – 1].

0 është indeks i elementit të parë të vargut

Element	Vlera e elementit
d[0]	–567
d[1]	8
d[2]	9
d[3]	72
d[4]	345
d[5]	–78
d[6]	78
d[7]	9
d[8]	10
d[9]	987

9 është indeks i elementit të fundit të vargut. d[9] 987

Tabela3.1.1

Indeksi i elementit të vargut mund të jetë konstanta, ndryshorja ose shprehja vlera e të cilës është jonegative dhe te vargu i indekseve prej 0 deri n – 1 (nëse gjatësia e vargut është n). Kështu, për a = 7 dhe b = 2 elementi d[a + b – 1] është, në realitet,

```
d[a + b - 1]
```

elementi d[8].

Inicializimi i vargut dhe shoqërimi i vlerave të elementeve

Elementeve të vargut mund t'u shoqërohen vlera me urdhër për shoqërim ose gjatë inicializimit.

Te shembujt vijues janë ilustruar mënyra të ndryshme të shoqërimit vlera të elementeve të vargut..

Shembulli 3.1.6

Shoqërimi i vlerave mund të kryhet veçanërisht çdo elementi

```
int a[10];
a[0] = 1; a[9] = 10; a[3] = -4; a[5] = 5;
```

Shembulli 3.1.7

Shoqërimi i vlerave mund të kryhet edhe nëpërmjet shprehjeve te të cilat njehsohet vlera e indekseve të vargut. Për shembull, nëse elementi a[5] = 24, me urdhërat:

```
int c = 3;
int d = 2;
a[c + d] += 6;
```

të elementit a[5] i zmadhohet vlera për 6, d.m.th., vlera e tij tani do të jetë 30

Shembulli 3.1.8

Shoqërimi i vlerave të elementeve të vargut mund të kryhet edhe gjatë deklarimit, d.m.th., me ***listë të inicializimit*** (angl. initializer list):

```
 int a[] = {5, -2, 7, -3, 6};
```

Vlerat e elementeve do të jenë: a[0] = 5, a[1] = –2, a[2] = 7, a[3] = –3, a[4] = 6. Gjatësia e vargut (numri i elementeve) gjatë inicializimit të këtillë e cakton vet përkthyesi.

Pas deklarimit dhe inicializimit të vargut

```
char shkronja[] = {'a', 'b', 'c'};
```

vlerat e elementeve do të jenë: a[0] = 'a', a[1] = 'b', a[2] = 'c'.

Shembulli 3.1.9

Nëse ka më pak vlera te lista e inicializimit se sa që ka elemente te vargu, elementet e të tjera janë inicializuar në zero.

Me këtë deklarim dhe inicializim

```
int disa[10] = {3, -1, 4};
```

elementet e vargut do të inicializohen me këto vlera: : 3, –1, 4, 0, 0, 0, 0, 0, 0, 0. Zerot i plotëson përkthyesi.

Ngjashëm, me

```
int zero[100] = {};
```

deklarohet vargu prej elementeve numra të plotë me emrin zero, ku të gjitha elementet janë inicializuar në 0.

Me deklarimin dhe inicializimin

```
char zanore[5] = {'a'};
```

në mënyrë eksplicite vlera 'a' e elementit zero prej vargut, por të gjitha elementet e të tjera automatikisht janë inicializuar në vendet e zbrazëta.

Vërejtje: Te C++ nuk ka inicializim automatik të vargjeve. Patjetër të paktën një element të jetë i inicializuar, që pastaj të gjithë elementet tjerë të jenë të inicializuar.

Shembulli 3.1.10

Dimensioni i vargut mund të njehsohet automatikisht nëse elementet janë dhënë gjatë inicializimit. Gjatë kësaj deklarate dhe inicializimi

```
double këto[] = {2.34, -5.67, 3.45, 7.89};
```

nuk është dhënë gjatësia e vargut, por e cakton përkthyesi, sipas numrit të dhënë të elementeve, d.m.th., vargu këto do të ketë gjatësi 4.

Ngjashëm, me deklarimin dhe inicializimin

```
string ditë[] = {"Hën", ""Mar", "Mër", "Ent", "Pre", "Sht", "Die"};
```

gjatësia e vargut ditët është 7.

Me deklaratën vijuese dhe inicializuese

```
int shifra[] = {1, 2, 3, 4, 5};
```

janë deklaruar vargu me gjatësi 5 dhe janë inicializuar të gjitha elementet.

Nga ana tjetër, me këtë deklarim dhe inicializim të vargut

```
int v[5] = {1, 2, 4, 5, 6, 9};
```

paraqitet sintaksa e gabimit pasi ka 6 vlera të inicializuara, por vargu mund të ketë më së shumti 5 elemente.

Shembulli 3.1.11

Vargjet nuk mund:

- të shoqëron njëri tjetrës:

```
int a[2] = {1,2};
int b[2];
b = a;                          // pasaktë
```

- të inicializohet njëra me tjetrën:

```
int c[2] = a;                   // pasaktë
```

- të shtypen vetëm nëpërmjet :

```
cout [1]<< a;                   // pasaktë
```

- të krahasohen[2]:

```
if(a == b)                      // pasaktë
```

- tëmjenë vlera kthyese prej funksionit[3].

```
return a;                       pasaktë
```

[1] Shtypen adresat e vargut.
[2] Krahasohen adresat e vargjeve.
[3] Kthehet treguesi ni vargut. Për treguesit do të fklasim më vonë.

Specifikimi i gjatësisë së vargut mund të kryhet edhe me **konstante simbolike.** Konstanta simbolike jepet me direktivën paraprocesore #**define**.

Për shembull:

```
#define PI 3.1415;
```

Shembulli 3.1.12

Në këtë shembull shfrytëzohet direktiva paraprocesore #define, me të cilën përkufizohet konstanta simbolike GJATËSIA vlera e të cilës është 10, **figura 3.1.1.**

Programi i inicializon dy elementet e para të vargut me vlerë 1, por të tjerat me 0. Me urdhrin for, elementeve prej të tretit deri te i dhjeti u shoqërohen vlera sipas formulës. (Cila?).

```
1   #include <iostream>
2   #include <iomanip>
3   using namespace std;
4
5   #define GJATËSI 10
6
7   int main() {     // Shfrytëzimi i konstantes simbolike
8
9       int z[GJATËSI ] = { 1, 1 };// Inicializimi i dy elementeve të para me 1,
10                                 //  por të tjerët me 0
11
12      // Shoqërimi i vlerave të elementeve prej të tretit deri te i dhjeti
13      int j;
14      for( j = 2; j < GJATËSI ; j++ )
15          z[ j ] = z[ j - 2 ] + z[ j - 1 ];
16
17      // Shtypja e vargut në mënyrë tabelare
18      cout << setw( 10 ) << "Element" << setw( 10 ) << "Vlera" << endl;
19      for( j = 2; j <GJATËSI ; j++ )
20          cout << setw( 6 ) << "z[" << j << "]" << setw( 8 ) << z[ j ] << endl;
21
22      cout << endl;
23      system( "Color 17" );
24      system( "pause" );
25      return 0;
26  }
```

Figura 3.1.1

Dalja pas realizimit të programit është:

```
   Element      Vlera
     z[0]         1
     z[1]         1
     z[2]         2
     z[3]         3
     z[4]         5
     z[5]         8
     z[6]        13
     z[7]        21
     z[8]        34
     z[9]        55

Press any key to continue . . .
```

Shembulli 3.1.13

Segmenti programor për lexim të elementeve të vargut me numra të plotë prej n elementeve $a[]_n$ e:

```
for(int i = 0; i < n; i++)
       cin >> a[i];
```

Shembulli 3.1.14

Segmenti programor për lexim të elementeve të vargut $b[]_n$ është:

```
for(int i = 0; i < n; i++)
       cout << a[i] << endl;
```

Shembulli 3.1.15

Segmenti programor për gjetjen e shumës së elementeve të vargut prej n elementeve $c[]_n$ është: :

```
shuma = 0;
for(int i = 0; i < n; i++)
       shuma += c[i];
```

I njëjti segment programor mund të shfrytëzohet edhe për njehsimin e prodhimit të elementeve të vargut, vetëm duhet të ndryshojnë urdhërat:

```
shuma = 0;          me     prodhim = 1;
zbir += c[i];       me     prodhim *= c[i];
```

Këto segmenta mund të shfrytëzohen për çfarëdo operacion me elementet e vargut.

Shembulli 3.1.16

Te shembulli i ***figura 3.1.2*** ilustrohet deklarimi, inicializimi dhe shoqërimi i elementeve të vargjeve të llojit: int, float, char dhe string.

```cpp
#include <iostream>
#include <string>
#include <iomanip>
using namespace std;

int main() {     // Deklarata, inicializimi dhe shoqërimi i vlerave
                 // të elementeve të vargut

    int gjatësia ;
    int a[ 4 ] = {};
    a[ 0 ] = 123; a[ 2 ] = 12345;
    gjatësia = sizeof a / sizeof( a[ 0 ] );
    cout << "Vargu me numra të plotë me gjatësi " << gjatësi << endl;
    cout << "index" << setw( 10 ) << "vlerë" << endl;
    for( int index = 0; index < gjatësi  ; index++ )
        cout << setw( 3 ) << index << setw( 10 ) << a[ index ] << endl;

    float b [] = { -3.5f, 2.7f, 7.3f };
    b[ 1 ] = 123.45f;
     gjatësi  = sizeof b / sizeof( b[ 0 ] );
    cout << "\nVarg real me gjatësi " << gjatësi << endl;
    cout << "index" << setw( 10 ) << "vlerë " << endl;
    for( int index = 0; index < gjatësi index++ )
        cout << setw( 3 ) << index << setw( 10 ) << b[ index ] << endl;

    char znaci [] = { '@', '#', '$', '%', '^' };
    shenja[ 1 ] = '[';shenja [ 2 ] = ']';
    gjatësia = sizeof shenja / sizeof(shenja [ 0 ] );
    cout << "\Varg me simbole me gjatës   " << gjatësia << endl;
    cout << "index" << setw( 10 ) << " vlerë   " << endl;
    for( int index = 0; index < dolzina; index++ )
        cout << setw( 3 ) << index << setw( 10 ) << shenja[ index ] << endl;

    string emra _  [] = { "Antonia", "Atanasia", "Mihaela", "Jana", "Jovana", "Teo" };
    gjatësia = sizeof   emra / sizeof( emra  [ 0 ] );
    cout << "\nTeo  Varg prej stringave me gjatësi " << gjatësi << endl;
    cout << "index" << setw( 10 ) << "vlerë " << endl;
    for( int index = 0; index < gjatësi ; index++ ) {
        cout << setw( 3 ) << index << "\t";
        cout << setw( 10 ) << left << emra [ index ] << right << endl;
    }

    cout << endl;
    system( "Color 17" );
    system( "pause" );
    return 0;
}
```

Figura 3.1.2

Dalja prej programit është:

```
Varg me numra të plotë me gjatësi 4
 index        vlera
  0          123
  1            0
  2        12345
  3            0
Varg real me gjatësi 3
 index        vlera
  0         -3.5
  1       123.45
  2          7.3
Varg me simbole me gjatësi 5
 index        vlera
  0            @
  1            [
  2            ]
  3            %
  4            ^
Vargu me stringa me gjatësi 6
 index        vlera
  0       Antonia
  1       Atanasia
  2       Mihaela
  3       Jana
  4       Jovana
  5       Teo

Press any key to continue . . .
```

Te nëntitulli **Funksionet e bibliotekës**, u njohtëm me operatorin sizeof, me të cilin mund të fitohet madhësia e memories të zënë prej ndonjë lloj ose prej shprehjes (ndryshore).

Operatori sizeof mund të zbatohet edhe te vargu, ku do të fitohet madhësia e vargut në bajt, që varet prej numrit të elementeve. Për ta fituar numrin e elementeve të vargut, duhet madhësin e vargut (në bajt) ta pjesëtojmë me madhësinë e llojit të elementeve të vargut (në bajt).

Për shembull:

```
char shenja[] = {'@','#','$','%','^'};
gjatësiaa = sizeof shenja / sizeof(char);
ose
gjatësia = sizeof shenja/sizeof(shenja[0]); shenja [0] është elemeni i parë
```

Detyra të zgjidhura

Detyra 3.1.1

Të shkruhen algoritmet dhe programi me të cilin çdo element prej vargut numerik a[]$_n$ do ta ndryshojë shenjën, + në – dhe – në +.

Algoritmi dhe program janë dhënë në vazhdim.

```
Algoritmi NdryshimiIShenjaveTëVargut
fillimi
        a[1..n];
        shtyp „Futni numrin e elementeve të vargut, n =“;
        lexo n
        për i ← 1 zmadhoje deri n
                lexo ai;
        fundi_për{i}
         për i ← 1 zmadhoje deri n
                ai ← –ai;
        fundi_për{i}
        shtyp„Vargu me shenja t ndryshuara të elementeve është:”;
         për i ← 1 zmadho deri n
                shtyp ai;
        fund_për  {i}
fund{NdryshimiIShenjaveTëVargut}
```

Një dalje prej realizimit të programit është:

```
Futni numrin e elementeve të vargut          n = 3
Futni elementet e vargut :
a[0] = 1
a[1] = -2
a[2] = 3

Vargu i futur është :
a[0]=1
a[1]=-2
a[2]=3

Vargu me shenja të ndryshuara është :
a[0]=-1
a[1]=2
a[2]=-3

Press any key to continue . . .
```

```
#include <iostream>
using namespace std;

int main() {
    // Të ndryshohen shenjat e elementeve të vargut

    const int m = 100;
    double a[ m ] = {};
    cout << " Futni numrin e elementeve të vargut  n = ";
    int n; cin >> n;
    cout << " Futni elementet e vargut  \n";
    for( int i = 0; i < n; i++ ) {
        cout << "a[" << i << "] = ";
        cin >> a[ i ];
    }

    cout << "\nVargu i futur është " << endl;
    for( int i = 0; i < n; i++ )
        cout << "a[" << i << "]=" << a[ i ] << endl;
    for( int i = 0; i < n; i++ )
        a[ i ] = -a[ i ];

    cout << "\n Vargu me shenja të ndryshuara është : " << endl;
    cout << "\n Vargu me shenja të ndryshuara është :" << endl;
    for (int i = 0; i<n; i++)
        cout << "a[" << i << "]=" << a[i] << endl;

    cout << endl;
    system("Color 17");
    system("pause");
    return 0;
}
```

Figura 3.1.3

Detyra 3.1.2

Të shkruhet program për njehsimin e shumës
$+ a_1 - a_2 + a_3 - a_4 + \ldots (+ / -) a_n$.

Një shembull prej realizimit të programit është:

```
Futni numrin e elementeve të vargut          n = 5
Futni elementet e vargut :
a[0] = 11
a[1] = -22
a[2] = -333
a[3] = 4444
a[4] = -55555

Vargu i futur është :    11, -22, -333, 4444, -55555,
Shuma e vargut është : +(11)-(-22)+(-333)-(4444)+(-55555) e -60299

Press any key to continue . . .
```

```cpp
#include <iostream>
using namespace std;

int main() {
    // Të gjendet shuma a1 - a2 + a3 - . . . (+/-)an

    const int m = 100;
    double shuma = 0, a[ m ] = {};
    int i, k, n;
    cout << " Futni numrin e elementeve të vargut , n = ";
    cin >> n;
    cout << "Futni elementet e vargut: \n";
    for( i = 0; i < n; i++ ) {
        cout << "a[" << i << "] = ";
        cin >> a[ i ];
    }
    cout << "\n Vargu i future është ";
    for( i = 0; i < n; i++ )
        cout << a[ i ] << ", ";
    k = 1;
    for( i = 0; i < n; i++ ) {
       shuma += k * a[ i ];
        k = -k;
    }
    cout << "\n Shuma e vargut: ";
    for( i = 0; i < n; i++ ) {
        if( i % 2 == 0 )
            cout << '+' << "(" << a[ i ] << ")";
        else
            cout << '-' << "(" << a[ i ] << ")";
    }
    cout << " e " <<shuma << endl;

```

Figura 3.1.4

Detyra 3.1.3

Të shkruhet program për gjetjen e elementit më të madh dhe më të vogël te vargu numerik a[]n, me funksione të veçanta.

Sqarim: Në fillim merret se elementi më i madh është i pari, a_1. Pastaj, krahasohet elementi i dytë a_2 me më të madh (por ai është i pari) dhe më i madhi prej tyre merret për më të madh. Më tutje, të gjithë elementet e të tjera a_3, a_4... a_n

krahasohen me elementin më të madh deri atëherë i gjetur dhe nëse gjendet më i madhi prej tij, merret ai të jetë më i madh. Poashtu, mbahet mend pozita e tij te vargu.

Do të përmendim shembull. Le të jetë vargu a = {2, –1, 7, 9, 3}.

i	a_i	a_i > max	max	index
1	2		2	1
2	–1	(–1 > 2) jo		
3	7	(7 > 2) po	7	3
4	9	(9 > 7) po	9	4
5	3	(3 > 9) jo		

Domethënë, më i madhi është elementi i 4-të me vlerë 9.

Mënyra për gjetjen e elementit më të madh te vargu është i ngjashëm, Do ta ilustrojmë te shembulli i njëjtë.

i	a_i	a_i < min	min	index
1	2		2	1
2	–1	(–1 < 2) po	–1	2
3	7	(7 < –1) jo		
4	9	(9 < –1) jo		
5	3	(3 < –1) jo		

Domethënë, më i vogël është elementi i 2-të me vlerë –1.

Te përkufizimi funksioni që ka argument vargje, vargjet shkruhen vetëm me llojin dhe emrin, sipas të cilit vendohet në kllapa të mesme për të ditur se është varg, por jo ndryshore

Për shembull:

```
void ElementiMëIMadh(double a[], double numri i elementeve, double &pozita,
double & më i madh)
```

Poashtu, vargu trajtohet si argument referent edhe pse nuk është venduar shenja &. Kjo do të thotë se të gjitha ndryshimet e elementeve të vargut të cilat kanë ndodhur gjatë realizimit të funksionit, ngelin edhe kthimi te funksioni kryesor.

Gjatë thirrjes së funksionit me argument varg, shkruhet vetëm emri i vargut dhe atë pa kllapa:

```
ElementiMëIMadh(a, n, numërrendor, më i madh);
```

Poashtu, madhësia e vargut bartet si argument sipas vlerës.

(Në realitet, ekziston mekanizëm i veçantë i **treguesve**[4] (angl. pointers) i cili shfyrëzohet në rastin e vargjeve. Me atë mechanize, emrii vargut trajtohen si tregues, i cili e përmban adresën e elementit të parë të vargut në memorie. (Ngjashëm sikurse bartet adresa e argumentit te parametrit referent përkatës).

[4] Shiko nënpikën **3.4 Tregues.**

Te disa raste, kur duam të ndryshojnë vlerat e elementeve te funksioni në të cilin është bartur vargu, pasi trajtohet si referente, duhet të shënohet si kontsnta me kualifikator const.

Funksionet për gjetjen e elementit më të madh dhe më të vogël të vargut, sikurse edhe funksioni kryesor te i cili thirren ato funksione, janë dhënë te ***figura 3.1.5.***

```
1   #include <iostream>
2   #include <iomanip>
3   using namespace std;
4
5   void leximiVargut    ( int&, double [] );
6   void shtypjaEVargut   ( const int, const double [] );
7   int elementiMëlMadh   ( const int, const double [] );
8   int elementiMëlVogël ( const int, const double [] );
9
10  int main() {     // Elementi më i madh dhe më i vogël te vargu
11
12      double a[ 100 ];
13      int n, index;
14
15      leximiVargut ( n, a );
16      system( "cls" );
17      cout << setw( 14 ) << "" << "Vargu njëdimensional \n" << endl;
18      shtypjaEVargut( n, a );
19
20      index = elementiMëlMadh ( n, a );
21      cout << "\nMë i madh është elementi me indeks << index
22          << "dhe me vlerë" << a[ index ] << endl;
23
24      index =elementiMëlVogël  ( n, a );
25      cout << "Më i vogël është elementi me indeks " << index
26          << "dhe me vlerë " << a[ index ] << endl;
27
28      cout << endl;
29      system( "Color 17" );
30      system( "pause" );
31      return 0;
32  }
33
34  void leximiVargut ( int& dimension, double x [] ) {
35      cout << "Futni numrin e elementeve";
36      cin >> dimension;
37      cout << "Futni numrin e elementeve" << endl;
38      for( int i = 0; i < dimension    ; i++ ) {
39          cout << "x[" << i << "] = ";
40          cin >> x[ i ];
41      }
```

Figura 3.1.5

```
42  }
43
44  void shtypjaEVargut ( const int dimension, const double x [] ) {
45      cout << setw( 10 ) << left << "Index:" << right;
46      for( int i = 0; i < dimension ; i++ )
47          cout << setw( 5 ) << i;
48      cout << endl;
49      cout << setw( 10 ) << left << "Vlera:" << right;
50      for( int i = 0; i < dimension ; i++ )
51          cout << setw( 5 ) << x[ i ];
52      cout << endl;
53  }
54
55  int elementiMëlMadh  ( const int numrielementeve, const double x [] ) {
56      int pozicion  = 0;
57      double mëimadh = x[ 0 ];
58      for( int i = 1; i < numrielementeve ; i++ )
59          if( x[ i ] > mëimadh ) {
60              mëimadh = x[ i ];
61              pozicion = i;
62          }
63      return pozicion ;
64  }
65
66  int elementiMëlVogël  ( const int numrielementeve  , const double x [] ) {
67      int pozicion = 0;
68      double meivogël  = x[ 0 ];
69      for( int i = 1; i < numrielementeve ; i++ )
70          if( x[ i ] < mëivogël ) {
71              mëivogël = x[ i ];
72              pozicion = i;
73          }
74      return pozicion ;
75  }
```

Figura 3.1.5 (vazhdim)

Shembull për dalje prej programit është:

```
                    Vargu njëdimensional
Index:      0     1     2     3     4     5     6     7     8     9
Vlera:      3     7    12     0    -3   -23     0     5    25     4

Më i madh është elementi me index 8 dhe me vlerë 25
Më i vogël është elementi me index 5 dhe me vlerë -23

Press any key to continue . . .
```

Urdhri for i bazuar në varg

Te Shembulli 3.1.15 njehsohet shuma e elementet e vargut prej n elementeve $c[]_n$:

```
shuma = 0;
for(int i = 0; i < n; i++)
	shuma += c[i];
```

Vërejmë se elementet mblidhen prej të parit deri te i fundit, sikurse janë renditur sipas indeksit i. Megjithatë, do të fitohet shuma e njëjtë nëse i mbledhim edhe sipas çdo renditje.

Ka shumë problem te t cilët realizohet ndonjë operacion mbi elementet e vargut, pavarësisht prej renditjes së tij. Për shembull: te detyra për gjetjen e elementit mëtë bogël (ose më të madh) te vargu, jo patjetër elementi i parë të merret për më të vogël, por mund çfarëdo. Pastaj, i krahasohet me të tjerët, por jo patjetër sipas ndonjë renditje. Kur do të gjendet ndonjë element më i vogël prej deri atëherë më i vogli, ai do të jetë më i vogël.

Për zgjidhjen e problemeve të këtilla, në C++ shfrytëzohet e ashtuqajtura **urdhri for i bazuar në varg** (ang. range-based for), ***figura 3.1.12***.

```
for(ndryshore : varg){
		urdhri A;
		urdhri B;
		...
		urdhri K;
}
```

Figura 3.1.12

- *ndryshorja* është ndryshorja e llojit të vargut,
- *vargu* është emri i vargut.

Te trupi i urdhrit for të bazuar në varg nuk mund të ketë edhe urdhër edhe urdhër për shënime ose operacione me indeksat.

Do të përmendim disa shembuj.

Shembuj

Shembulli 3.1.17

Segmenti programor prej **Shembulli 3.1.15** për shumën e elementeve të vargut c[], mund të shkruhet në këtë mënyrë:

```
for(int numër: c)
	shuma += numër; ;
```

Shembulli 3.1.18

Te **Detyra 3.1.2,** urdhri for për shumën e elementeve:

```
for(i = 0; i < n; i++) {
	shuma += k * a[i];
	k = -k;
}
```

Mund të shkruhet edhe me urdhrin for të bazuar në varg.

```
for(double numri: a) {
	shuma += k * numri;
	k = -k;
}
```

Shembulli 3.1.19

Për gjetjen e elementit më të vogël ose më të madh te vargu, nuk është e nevojshme të krahasohen elementet sipas indeksit, por duhet të krahasohen të gjithë elementet. Prandaj, mund të shfrytëzohet urdhri for të bazuar në varg.

```
double mëivogël = a[0];
double  mëimadh  = a[0];
for(auto numër: a) {
	if(numër < mëivogël)
		mëivogël = broj;
	if(numër > mëimadh)
		mëimadh = numër;
}
```

Detyra për ushtrime

Të shkuhen programe për këto detyra me shfrytëzimin e vargjeve njëdimensionale:

1. Të gjendet prodhimi i elementeve të vargut numra të plotë njëdimensional $a[]_n$.
2. Të gjendet mesi aritmetik (A) dhe harmonik (I) të vargut numerik $a[]_n$.

$$A = \frac{a_1 + a_2 + ... + a_n}{n} \qquad H = \frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + ... + \frac{1}{a_n}}$$

3. Prej vargut të numrave $a[]_n$ të njehsohet në veçanti shuma e numrave çift dhe shuma e numrave tek.
4. Të veçohen vargjet elementet e vargut a[]n me indekse çift dhe me indekse tek.
5. Të kryhet zhvendosja ciklike e elementeve të vargut prej shenjave a[]n për k vende djathtas ose majtas.
6. Të kontrollohet vargu numerik a[]n a gjendet elementi me vlerë v.
7. Të gjendet sa elemente prej vargut numerik a[]n kanë vlerë më të vogël prej v, por sa më e madhe.
8. Të formohet vargu i ri c[]n, elementet e të cilit janë shumë e elementeve përkatëse e vargut a[]n dhe b[]n, d.m.th., ci = ai + bi, për i = 0, 1... n – 1.

9. Të njehsohet prodhimi i elementeve të vargjeve numerike a[]n dhe b[]n, d.m.th., $a_0b_0 + a_1b_1 + \ldots + a_{n-1}b_{n-1}$.
10. Të njehsohet vlera e polinomit

$$P_n(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1 + a_0$$

te i cili koeficientët dhe argument janë numra real, por n është numr natyror.

Pyetje për kontroll të njohurive

1. Si deklarohet vargu njëdimensional?
2. I cilit lloj të të dhënës duhet të jetë indeksi i elementeve të një vargu njëdimensional?
3. Nëse është deklaruar vargu njëdimensional

```
int a[ 10 ];
```

4. cilat vlera mund t'i pranon indeksi i elementeve të vargut?
5. Në çfarë mënyra mund të kryhet shoqërimi i vlerave të elementeve të vargut njëdimensional?
6. Në çfarë mënyra mund të kryhet inicializimi i elementeve të vargut njëdimensional?
7. Si përufizohet konstanta simbolike?
8. Si caktohet dimensioni i vargut njëdimensional?
9. Kur shfrytëzohet urdhri for e bazuar në varg? Shkruani sintaksën e tij.

3.2 Vargjet dydimensionale – matricat

Te nënpika **3.1 Vargjet njëdimensionale** treguam se të dhënat të cilat kanë një dimension mund (matematikisht) të paraqiten me vargje me një indeks: a1, a2... an, ose shkurtimisht a[]n. Këto vargje te gjuhët programore, pra edhe te C++, shkruhen me indeksat në kllapat e mesme: a[0], a[1]... a[n – 1]. Poashtu, indeksat fillojnë prej 0.

Ka shembuj në matematikë edhe në fushat tjera, ku për të shkruar disa të dhëna patjetër të shfrytëzohen 2, 3... dhe më shumë indekse.

Për shembull, e dimë se çdo pikë në rrafsh është caktuar me dy dimensione – koordinata: x-koordinata dhe y-koordinata, d.m.th., A(x, y), ***figura 3.2.1.***

Ngjashëm, notat e nxënësve prej një klase nga të gjitha lëndët

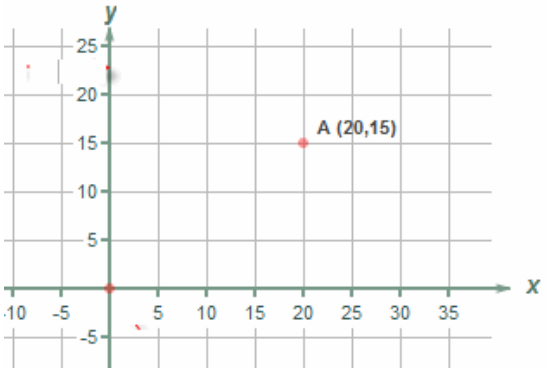


Figura 3.2.1

(të supozojmë 12) mund të tregohen me tabelë me dy dimensione, me rreshta (horizontale, dimensioni i parë) dhe shtylla (vertikale, dimensioni i dytë).

		1	2	3	4	5	6	7	8	9	10	11	12	
	Lënda →	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	Prosek
	Nxënësi ↓													
1	u_1	3	4	5	2	3	4	3	4	5	3	3	2	
2	u_2	4	5	4	5	3	4	5	4	5	5	5	4	
3	u_3	5	5	5	5	2	4	4	4	4	2	4	5	
…	. . .													
	mesatare													

Figura 3.2.2

Këto të dhëna mund të shkruhen si varg dydimensional (angl. two-dimensional array) ku njëri dimension është nxënësit (u), por dimensioni tjetër është lëndët (p). Domethënë, çdo note në tabelë ka indekse. Nëse vargun e emërtojmë me no (shkurt prej nota), atëherë vlerat e elementeve të vargut dydimensional janë notat e nxënësve sipas lëndëve. Për shembull, no(n2, p7) = 5, o(p3, p10) = 2 etj. Ose, shkurt, vetëm me indeksat, $o_{2,7} = 5$, $o_{3,10} = 2$ etj.

Nëse dimë se ka m nxënës dhe n lëndë, atëherë vargu (maematikor) shkuhet me: $o_{1,1}$, $o_{1,2}$..., $o_{1,n}$, $o_{2,1}$,$o_{2,2}$..., $o_{2,n}$, $o_{3,1}$..., $o_{m,1}$, $o_{m,2}$..., $o_{m,n}$.

Ose, më qartë si tabelë dydienzionale:

$$\begin{bmatrix} o_{1,1} & o_{1,2} & \cdots & o_{1,j} & \cdots & o_{1,n} \\ o_{2,1} & o_{2,2} & \cdots & o_{2,j} & \cdots & o_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ o_{i,1} & o_{i,1} & \cdots & o_{i,j} & \cdots & o_{i,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ o_{m,1} & o_{m,2} & \cdots & o_{m,j} & \cdots & o_{m,n} \end{bmatrix}$$

Vargu dydimensional shënohet me $o_{[i,j]_{m,n}}$ që do të thotë se indeksi i ka vlerë prej 1 deri m, por indeksi j ka vlerë prej 1 deri n. Shkurt, vargu shënohet me $o_{[]_{m,n}}$, ku m, n tregojnë se vargu është dydimensional dhe se dihet indeksi i parë është prej 1 deri m, por i dyti prej 1 deri n.

Në literature shfrytëzohet edhe shënimi o[1..m][1..n].

Elementi i përgjithshëm shënohet me oi,j.

(Vargjet dydimensionale në matematikë quhen **matrica)**.

Vargjet e këtilla dydimensionale te gjuhët programore, pra edhe te C++, shkruhen me o[m][n], por elementi i përgjithshëm është me dy kllapa të mesme, o[i][j]. Për shembull, o[2][7] = 5, o[3][10] = 2 etj. (Është e gabueshme paraqitja o[2, 7]).

Deklarimi, inicializimi dhe shoqërimi i vlerave të elementeve të vargjeve dydimensionale

Deklarimi i vargut dydimensional është i ngjashëm me deklaratën e vargut njëdimensional:

lloj emri [*numri1*] [*numri2*];

lloj – është lloj i elementeve të vargut (short, int, long, float, double, char, boolean, string etj.).
emri – është emri i vargut.
numri1 dhe *numri2* – janë dimensionet e vargut dhe ato patjetër të jenë literale ose konstante të emërtuara më të mëdha se 0

Për të qenë menjëherë më e qartë, te tabela e sipërme me notat për nxënësit, *numri1* do të jetë numri i nxënësve në klasë (numri i rreshtave), por *numri2* do të jetë numri i lëndëve (numri i shtyllave).

Të dy indeksat e vargut dydimensional fillojnë prej 0.
Indeksat e parë të vargut dydimensional janë: 0, 1, 2..., *numri1* – 1.
Indeksat e dytë i vargut dydimensional janë: 0, 1, 2... *numri2* –

Shembuj

Shembulli 3.2.1

Me deklaratën

```
int a[3][4];
```

deklarohet vargu $a[]_{3,4}$ prej 12 elemente, të të cilët undeksi i parë është 0, 1 ose 2, por indeksi i dytë është 0, 1, 2 ose 3. Ky varg grafikisht mund të paraqitet me tabelën me 3 rreshta dhe 4 shtylla:

rreshta (i parë indkesi)	shtylla (i dytë indeksi) 0	1	2	3
0	$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
1	$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
2	$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$

Elementet e vargut shkruhen me indeksin me kllapa të mesme.

	0	1	2	3
0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Vlerat e elementeve të vargut dydimensional shoqërojnë me urdhrin për shoqërim.

Shembulli 3.2.2

Me kto urdhëra:

```
float c[10][15];
c[0][4] = -5.23f; c[9][2] = 7.543f;
```

deklarohet vargu c[][] prej 10 x 15, d.m.th., 150 elementeve u shoqërohen vlera të elementeve c[0][4] dhe c[9][2]. (Po nuk janë të njohura vlerat e elementeve tjerë).

Vargu dydimensional mund të inicializohet gjatë deklarimit edhe me listën inicjalizuese.

Shembulli 3.2.3

Me urdhrin

```
int d[3][2] = {};
```

iniciaizohen të gjithë elementet e vargut d të vlerës 0.

Me urdhrin

```
char shkronja[3][2] = {'a', 'b', 'c'};
```

deklarohet dhe inicializohet vargu dydimensional shkronja.

	0	1
0	a	b
1	c	
2		

Nëse nuk ka mjaft vlera te lista inicializuese për inicializim të gjitha elementeve të vargut, mbetja inicializohet me vend të zbrazët.

Me urdhrin

```
int d[3][2] = {-3,5,4};
```

inicializohen tre elemente me vlera të përbashkëta, por të tjerat me 0.

	0	1
0	–3	5
1	4	0
2	0	0

Shembulli 3.2.4

Me urdhrin

```
int d[3][2] = {{-3, 5},{4, 1},{7, -2}};
```

deklarohen dhe inicializohet vargu prej 3 rreshtave (çdo rresht vendohet në kllapa të mëdha) dhe 2 shtylla (çdo rresht ka nga 2 elemente), sipas të dhënave prej listës inicializuse. Deklarata e njëjtë mund të shkruhet edhe më qartë:

```
int d[3][2] = {
                    {-3, 5},
                    {4, 1},
                    {7, -2}
              };
```

	0	1
0	–3	5
1	4	1
2	7	–2

Gjithashtu, nëse nuk ka mjaft elemente për inicializim të ndonjë rreshti, elementet e të tjera te ai inicializohen me 0.

```
int d[3][2] = {{-3}, {4, 1}, {7}};
```

	0	1
0	–3	0
1	4	1
2	7	0

Deklarimet e njljta mund të bëhen edhe me këto urdhëra:

```
int d[][2] = {{-3, 5},{4, 1},{7, -2}};
```

dhe

```
int d[][2] = {{-3}, {4, 1}, {7}};
```

ku mahësia e dimensionit të parë jo patjetër të përmendet. Përkthyesi vet e njehson.

Megjithatë, nuk mund të deklarohet me urdhrin

```
int d[][] = {{-3, 5}, {4, 1}, {7, -2}};
```

pasi përkthyesi nuk di nga sa elementet të marrë prej çdo rreshti (kllapa e brendshme). Mund të merret nga 1 ose nga 2. Prandaj, mund të anashkalohet vetëm madhësia e dimensionit të parë.

Shembulli 3.2.5

Dimensionet e vargut mund të jepen edhe nëpërmjet konstantave

Me urdhërat:

```
const int m = 10;
const int n = 5;
float b[m][n];
```

deklarohet vargu dydimensional me 10 rreshta dhe 5 shtylla, d.m.th., element ii parë është b[0][0], por i fundit është b[9][4].

Po jo patjetër të shfrytëzohet gjithë vargu b[m][n] elemente, por mund vetëm pjesë e tij. Për shembull, mundemi të japim dimensione më të vogla të vargut:

```
int rreshta = 4;
int shtylla = 3;
```

dhe të shfrytëzohet vetëm pjesa b[rreshta][shtylla].

Njehsimi i dimensioneve të vargut dydimensional

Nëse vargu $a[]_{m,n}$ është inicializuar me listën inicializuese, atëherë për caktimin e gjatësisë (numri i rreshtave), në C++ shfrytëzon funksionin sizeof().

```
numriIElementeve = sizeof(a[0][0]); – numri i bajtëve të një elementi.
numriIRreshtit = sizeof(a[0]);      – numri i bajtëve në një rresht.
bajtVargu = sizeof(a) më pas        – numri i bajtëve në gjithë vargun.
```

Pastah mund të njehsohet:

```
gjatësia = numriVargut/ bajtElement;
shtylla = bajtRreshti / bajtElement;
rreshta = gjatësia / shtylla;
```

Shembulli 3.2.6

Segmenti programor për lexim të elementeve të vargut dydimensional $a[]_{m,n}$ e:

```
for(int i = 0; i < m; i++)
	for(int j = 0; j < n; j++)
		cin >> a[i][j];
```

Shembulli 3.2.7

Segmenti programor për shtypje të elementeve të vargut dydimensional $b[]_{m,n}$ është:

```
for(int i = 0; i < m; i++) {
	for(int j = 0; j < n; j++)
		cout << "\tb[" << i << "," << j << "] = " << b[i][j];
	cout << endl;
}
```

Shembulli 3.2.8

Segmenti programor për gjetjen e shumës së elementeve të numrit të plotë të vargut dydimensionalc$[]_{m,n}$ është: :

```
int shuma  = 0;
for(int i = 0; i < m; i++) {
	for(int j = 0; j < n; j++)
		shuma += c[i][j];
}
```

Shembulli 3.2.9

Me programin prej ***figura 3.2.3*** janë ilustruar deklarimi, inicializimi, leximi dhe shtypja e vargut dydimensional.

```
#include <iostream>
#include <iomanip>
using namespace std;

const int m = 2;            // Dimensionet e vargut të parë
const int n = 3;

int main() {    // Deklarimi. inicializimi. leximi dhe
                // shtypja e vargut dydimensional

    int i, j;   //Numëruesi i rreshtave dhe të shtyllave, përkatësisht
                //Inicializimi i vargut me listë inicializuese
    int VarguA [ m ][ n ] = { { 1, 2, 3 }, { 4, 5, 6 } };
    cout << "Vargu i parë është inicializuar" << endl;

    cout << "\nShtypja e vargut të parë" << endl;
    for( i = 0; i < m; i++ ) {
        for( j = 0; j < n; j++ )
            cout << setw( 5 ) << VarguA [ i ][ j ];
        cout << endl;       //Rreshti i ri pas çdo rreshti
    }

    const int m = 3;        // Dimensionet e vargut të dytë
    const int n = 4;
    int Vargu B [ m ][ n ] = {};    // Deklarimi dhe inicializimi i 0
                                    // Leximi i elementeve të vargut
    cout << "\nFutni elementet e vargut të dytë:" << endl;
    for( i = 0; i < m; i++ ) {
        cout << "\nFutni element në " << i + 1 << "- rresht" << endl;
        for( j = 0; j < n; j++ ) {
            cout << "Futni " << j + 1 << "element";
            cin >>vargu [ i ][ j ];
        }
    }

    cout << "\nShtypja e vargut të dytë" << endl;
    for( i = 0; i < m; i++ ) {
        for( j = 0; j < n; j++ )
            cout << setw( 5 ) << Vargu B  [ i ][ j ];
        cout << endl;       //Rreshti i ri pas çdo rreshti
    }

    cout << endl;
    system( "Color 17" );
    system( "pause" );
    return 0;
}
```

Figura 3.2.3

Shembulli 3.2.10

Me këtë shembull është ilustruar deklarimi, inicializimi dhe shoqërimi i vlerave të elementeve të vargut dydimensional prej llojit të ndryshëm, si edhe njehsimi i numrit të rreshtave dhe të shtyllave.

```
#include <iostream>
#include <string>
using namespace std;

int main()
{   // Deklarimi, inicializimi dhe shoqërimi i vlerave
    // të elementeve të vargut dydimensional

    int i, j, gjatësia  reshta shtylla, bajtElement, bajtRreshti, bajtVarg;
    int a[2][3] = {};
    cout << "V                    inicializuar prej  0 "
        << "\n                  2  vlerave :" << endl;
    a[1][2] =                   45;
    Rreshta = 2;
    Shtylla  = 3;
    for (i = 0; i <  Rreshta ; i++)
        for (j = 0; j < Shtylla; j++)
            cout << "\ncel[" << i << "][" << j << "] = " << a[i][j];

    float b[][2] = { { -3.5f, 2.7f },{ 7.3f },{ 0.3f, -2.f } };
    b[0][1] = 133.f; b[2][1] = -57.f;
    bajtElement   = sizeof(b[0][0]);
    bajtRreshti   = sizeof(b[0]);
    bajtVarg    = sizeof(b);
    gjatësia  =   bajtVarg / bajtElement ;
    shtylla  = bajtRreshti / bajtElement   ;
    Rreshta = gjatësia  / shtylla  ;
    cout << "\n\nVargu i nurave real i inicializuar "
        << "\n dhe pastaj të shoqëruar 2 vlera " << endl;
    for (i = 0; i < rreshta ; i++)
        for (j = 0; j < shtylla ; j++)
            cout << "\nreal[" << i << "][" << j << "] = " << b[i][j];

    char shenjë[][3] = { { '@','#','$' },{ '%','^' },{ '*' },{ '-','+','=' } };
    shtylla = 3;
    rreshta = sizeof( shenjë) / sizeof( shenjë [0][0]) / shtylla ;
    cout << "\n\nVarg prej shenjash :" << endl;
    for (i = 0; i < rreshta ; i++)
        for (j = 0; j < shtylla ; j++)
            cout << "\nshenjë [" << i << "][" << j << "] = " <<shenjë [i][j];

    string emrat   [][1] = { { "Antonia" },{ "Atanasia" },{ "Mihaela" },{ "Jovana" },
    {},{ "Jana" } };
    shtylla  = 1;
    rreshta = sizeof( emrat ) / sizeof( emrat  [0][0]) / shtylla
    cout << "\n\nVarg prej stringa " << endl;
    for (i = 0; i < rreshta ; i++)
        for (j = 0; j < shtylla ; j++)
            cout << "\nstring[" << i << "][" << j << "] = " << emrat [i][j];
```

(int)3
Search Online

Figura 3.2.4

```
50
51          cout << endl;
52          cout << endl;
53          system("Color 17");
54          system("pause");
55          return 0;
56      }
```

Figura 3.2.4 (vazhdim)

Rezultati i realizimit të programit është:

```
Vargu prej numrave të plotë të inicializuar në 0
dhe pastaj të shoqërura 2 vlera:
plotë [0][0] = 0
plotë [0][1] = 0
plotë [0][2] = 0
plotë [1][0] = -45
plotë [1][1] = 0
plotë [1][2] = 123

Vargu prej numrave të realë të inicializuar
dhe pastaj të shoqërura 2 vlera

real [0][0] = -3.5
real [0][1] = 133
real [1][0] = 7.3
real [1][1] = 0
real [2][0] = 0.3
real [2][1] = -57


Varg prej shenjash:
shenja [0][0] = @
shenja [0][1] = #
shenja [0][2] = $
shenja [1][0] = %
shenja [1][1] = ^
shenja [1][2] =
shenja [2][0] = *
shenja [2][1] =
shenja [2][2] =
shenja [3][0] = -
shenja [3][1] = +
shenja [3][2] = =


Varg prej stringeve:
string[0][0] = Antonia
string[1][0] = Atanasia
string[2][0] = Mihaela
string[3][0] = Jovana
string[4][0] = Jana
string[5][0] = Teo

Press any key to continue . . .
```

Detyra të zgjidhura

Detyra 3.2.1
Të shtypet tabela për shumëzim deri n.

Programi është dhënë te **figura 3.2.5.**

```
#include <iostream>
#include <iomanip>
using namespace std;

int main() {     // Tabela e shumëzimit

    const int m = 20;
    const int n = 20;
    int t[ m ][ n ];
    int i, j, k;
    cout << "Futni deri te cili numër doni të shtypet tabela";
    cin >> k;
    for( i = 1; i <= k; i++ )
        for( j = 1; j <= k; j++ )
            t[ i ][ j ] = i * j;
    system( "cls" );
    cout << setw( 20 ) << "" << " TABELA E SHUMËZIMIT \n" << endl;
    cout << setw( 3 ) << "";
    for( j = 1; j <= k; j++ )
        cout << setw( 5 ) << j;
    cout << endl << endl;
    for( i = 1; i <= k; i++ ) {
        cout << setw( 3 ) << i;
        for( j = 1; j <= k; j++ )
            cout << setw( 5 ) << t[ i ][ j ];
        cout << endl;
    }

    cout << endl;
    system( "Color 17" );
    system( "pause" );
    return 0;
}
```

Figura 3.2.5

Një dalje prej programit është:

```
                                Tabela e shumëzimit
      1    2    3    4    5    6    7    8    9   10   11   12   13   14   15

 1    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
 2    2    4    6    8   10   12   14   16   18   20   22   24   26   28   30
 3    3    6    9   12   15   18   21   24   27   30   33   36   39   42   45
 4    4    8   12   16   20   24   28   32   36   40   44   48   52   56   60
 5    5   10   15   20   25   30   35   40   45   50   55   60   65   70   75
 6    6   12   18   24   30   36   42   48   54   60   66   72   78   84   90
 7    7   14   21   28   35   42   49   56   63   70   77   84   91   98  105
 8    8   16   24   32   40   48   56   64   72   80   88   96  104  112  120
 9    9   18   27   36   45   54   63   72   81   90   99  108  117  126  135
10   10   20   30   40   50   60   70   80   90  100  110  120  130  140  150
11   11   22   33   44   55   66   77   88   99  110  121  132  143  154  165
12   12   24   36   48   60   72   84   96  108  120  132  144  156  168  180
13   13   26   39   52   65   78   91  104  117  130  143  156  169  182  195
14   14   28   42   56   70   84   98  112  126  140  154  168  182  196  210
15   15   30   45   60   75   90  105  120  135  150  165  180  195  210  225

Press any key to continue . . .
```

Detyra 3.2.2

Të gjendet elementi më i vogël dhe më i madh te vargu dydimensional. Për lexim, për shtypje dhe për gjetjen e elementit më të vogël dhe më të madh të shkruhen funksionetë veçanta.

***Sqarim*:** Te aplikimi shfrytëzohen funksionet leximD2Vargu(), shtypjaD2Vargu(), maxD2Element() dhe minD2Element(). Gjatë kërkimit të elementit më të vogël dhe më të madh, vlerat dalëse prej funksioneve janë indeksat e elementit.

```
1   #include <iostream>
2   #include <iomanip>
3   using namespace std;
4
5   void leximD2Vargu ( int&, int&, double [][ 10 ] );
6   void shtypjaD2Vargu ( const int, const int, const double [][ 10 ] );
7   void maxD2Element  ( const int, const int, const double [][ 10 ], int&, int& );
8   void minD2Element  ( const int, const int, const double [][ 10 ], int&, int& );
9
10  int main() {     // Elementi më i vogël dhe më i madh te vargu dydimensional
11      const int m = 10;
12      const int n = 10;
13      double a[ m ][ n ] = {};
14      int i, j, rreshti, shtylla, imax, jmax, imin, jmin;
15      double min, max;
16
17      leximD2Vargu (rreshti, shtylla, a );
18
19      system( "cls" );
20      cout << "Vargu dydimensional " << endl;
21      shtypjaD2Vargu (rreshti, shtylla, a );
22
```

Figura 3.2.7

```
23      // Kërkimi i elementit më të vogël dhe më të madh
24      min2DElement(rreshta  shtylla a, imin, jmin );
25      max2DElement(rreshta  shtylla , a, imax, jmax );
26      cout << "\nElemeti më i madh te vargu është a["
27          << imax << "][" << jmax << "] = " << a[ imax ][ jmax ] << endl;
28      cout << "\nElemeti më i vogël te vargu është a["
29          << imin << "][" << jmin << "] = " << a[ imin ][ jmin ] << endl;
30
31      cout << endl << endl;
32      system( "Color 17" );
33      system( "pause" );
34      return 0;
35  }
36
37  void leximI2DVargut ( int& rresht , int& shtyllave double a [][ 10 ] ) {
38      cout << "Futni numrin e rreshtave (<=10): "; cin >> rresht;
39      cout << "Futni numrin e shtyllave(<=10): "; cin >>shtyllave;
40      cout << " Futni elementet e vargut sipas rrshtave : " << endl;
41      for( int i = 0; i < rresht ; i++ ) {
42          for( int j = 0; j < shtyllave  ; j++ ) {
43              cout << "a[" << i << "][" << j << "] = ";
44              cin >> a[ i ][ j ];
45          }
46      }
47  }
48
49  void shtypjaD2Vargu ( const int rresht , const int shtyllave, const double a [][ 10 ] ) {
50      cout << setw( 2 ) << "";
51      for( int j = 0; j <rresht  ; j++ )
52          cout << setw( 5 ) << j;
53      cout << endl << endl;
54      for( int i = 0; i < shtyllave; i++ ) {
55          cout << setw( 2 ) << i;
56          for( int j = 0; j <shtyllave; j++ )
57              cout << setw( 5 ) << a[ i ][ j ];
58          cout << endl;
59      }
60  }
61
62  void min2DElement( const int rresht , const int shtyllave, const double b [][ 10 ],
63      int& imëivogël, int& jmëivogël) {
64      double Mëivogël = b[ 0 ][ 0 ];
65     imëivogël = 0;imëivogël= 0;
66      for( int i = 0; i < rresht; i++ )
67          for( int j = 0; j < shtyllave ; j++ )
68              if( b[ i ][ j ] <Mëivogël) {
69                Mëivogël = b[ i ][ j ];
70               imëivogël = i;jmëivogël= j;
```

Figura 3.2.7 (vazhdim 1)

```
71              }
72      }
73
74      void max2DElement( const int rresht , const int shtylla, const double b [][ 10 ],
75          int& imëimadh, int& jmëimadh ) {
76          double mëimadh = b[ 0 ][ 0 ];
77          imëimadh, = 0; jmëimadh = 0;
78          for( int i = 0; i < rresht   ; i++ )
79              for( int j = 0; j < shtylla  ; j++ )
80                  if( b[ i ][ j ] > mëimadh ) {
81                      mëimadh = b[ i ][ j ];
82                      imëimadh = i; jmëimadh = j;
83                  }
84      }
```

Figura 3.2.7 (vazhdim 2)

Dalja prej realizimit të programit është:

```
Vargu dydimensional
      0    1    2    3

 0    3   -2   -7    9
 1    4   -6    5   34
 2   23  -12  -56  -33
 3    0    3    4   -2
 4   -5   -7    0    4

Elementi më i madh te vargu është a[1][3] = 34

Elementi më i vogël te vargu është  a[2][2] = -56


Press any key to continue . . .
```

Detyra për ushtrime

1. Të gjendet mesi aritmetik i elementeve të vargut dydimensional $a[]_{m,n}$.
2. Të njehsohet shuma e të dy vargjeve dydimensionale me dimensione të njëjta, $c[]_{m,n} = a[]_{m,n} + b[]_{m,n}$. (Shuma e elementit $c_{i,j} = a_{i,j} + b_{i,j}$).
3. Të formohet vargu dydimensional a[][]n,n që i përmban numratprej 1 deri n2 në formën e gjarpërit. Për shembull, për n = 3 vargu e ka këtë pamje:

```
1 2 3
6 5 4
7 8 9
```

4. Të gjenden elementet maksimale sipas rreshtave dhe elementet minimale sipas shtyllave te vargu dydimensional $a[]_{m,n}$.
5. Të gjendet në veçanti shuma e elementeve të diagonals krysore dhe shuma e elementeve të diagonales dytësore të vargut dydimensional $q[][]_{n,n}$.

6. Të gjendet shuma e elementeve mbi diagonalen dydimensionale a[]$_{n,n}$.
7. Të fshihen rreshti- id he shtylla- j prej vargut dydimensional a[]$_{m,n}$.
8. Të njehsohet shuma e elementeve të cilat shtrihen në diagonalet të cilat kalojnë nëpër elementin ai,j të vargut dydimensional a[]$_{m,n}$.
9. Të rrotullohen shtyllat e vargut dydimensional a[]m,n për k vende majtas (ose djathtas).
10. Të gjenerohet ky varg dydimensional a[]$_{m,n}$.

1	4	9	16	25	…
2	3	8	15	24	…
5	6	7	14	23	…
10	11	12	13	22	…
17	18	19	20	21	…
26	27	…	…	…	…

Pyetje për kontroll të njohurive

1. Si deklarohet dhe inicializohet vargu dydimensional?
2. Si mundemi ta deklarojmë vargun dydimensional b[][] me 4 rreshta dhe 3 shtylla?
3. Re vargu i përkufizuar b[][] te detyra paraprake a ekzistojnë elementet: b[3][4], b[4][3], b[0][4], b[0][3], b[3][0] dhe b[4][0]?
4. Le të jetë dhënë

```
int a[5][5] = {1, 2, 3, 4, 5, 6, 7, 8};
```

Cila është vlera e elementit a[1][3]?

5. Nëse është deklaruar

```
double d[10][10];
```

ku është gabimi te urdhri

```
d[5][10] = 4.55;
```

6. Çka do të shtyp ky segment programor:

```
int i, j, niza[5][5];
for(i = 0, j = 0; i < 5; i++, j++)

   niza[i][j] = i + j;
for(i = 0, j = 0; i < 5; i++, j++)
   cout << niza[i][j] << endl;
```

7. Të shkruhet segmenti programor për lexim të elementeve të vargut dydimensional a[]$_{m,n}$.
8. Të shkruhet segmenti programor për shtypje të elementeve të vargut dydimensional a[]$_{m,n}$.

3.3 TREGUESIT

Madhësia e cila u shoqëron ndryshoreve gjatë vendosjes së tyre te memoria varet prej llojit të tyre. (Atë e vërejtëm kur folëm për llojin e të dhënave). Për shembull, nëse janë deklaruar ndryshoret:

```
int numërIPlotë;
float realenBroj;
```

atëherë gjatë vendosjes së ndryshores numërIPlotë në memorie, i shoqërohet memories te e cila mund të shkruhet numri i plotë më i madh prej vargut të përkufizuar në numrat e plotë të llojit int. Gjithashtu, gjatë vendosjes së ndryshores numërReal te memoria, asaj i shoqërohet aq memorie që të mund në atë të shkruhet numri real më i adh prej vargut të përkufizuar të numrave real të llojit float.

Ndryshoret mund të vendoset n te memoria në dy mënyra:

- Statike.
- Dinamike.

Gjatë realizimit të programit, ndryshoret statike (angl. statik variables) vendose n në adresë fukse te memoria. Ato ngelin në të adresë deri në mbarimin e programit edhe pse vlera e tyre mund të ndryshojë. Pasi adresa është fikse dhe gjithmonë e njohur, qasja deri tea to gjatë leximit ose gjatë të shkruarit e vlerës së re është shumë i shpejtë.

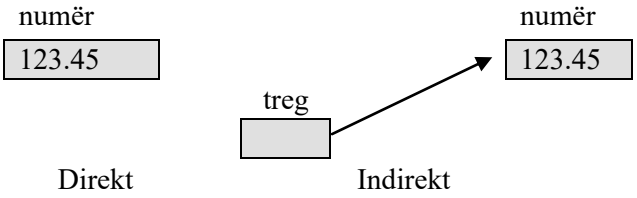
Ndryshoret dinamike (angl. dynamic variables), pra, vendose n te memoria në momentin e krijimit të tyre (sipas deklarimit). Poashtu. Adresa e vendosjes cktohet në atë moment.

Në gjuhët programore bashkëkohore, për qasje deri te ndryshoret dinamike të cilat vendose n në çfarëdo adresë të lire në momentin e krijimit, shfrytëzohen mekanizma **të treguesve** (angl. pointers).

Deklarimi i treguesve

Treguesit janë ndryshore vlerat e të cilit janë adresa memorie. Ndryshoret e këtilla quhen **ndryshore të treguesve** (angl. pointer variables) ose ndryshore të llojit tregues. Është e njohur se ndryshoret përmbajnë të dhëna të llojit përkatës: numra të plotë, numra real, shenja, stringe etj. Ndryshoret e treuesve nuk mund të përmbajnë të dhëna të tjera, përveç adresa të ndryshoreve. Në këtë kuptim, mund të themi se ndryshoret të cilit do lloj, përveç prej llojit të treguesit. Direkt referojnë vlerë (nmër, shenjë string etj.), por ndryshoret prej llojit tregues në mënyrë indirekte referojnë vlerë.

Për shembull, nëse numri është ndryshore, por tregk tregues të ndryshores numër, atëherë qasja në mënyrë direkte dhe indirekte deri te ndryshorja numër është paraqitur me këtë figurë.



Treguesit deklarohen në këtë mënyrë:

```
lloj *emri_i_tregues;
```

llojobjek – është lloj i ndryshoreve te të cilat do të tregojë treguesi,
* – është shenjë për deklarimin e treguesit, d.m.th., të ndryshores së treguesit.

Për shembull, me

```
int *pok;
```

deklarohet treguesi treg, i cili mund të shfrytëzohet vetëm për të treguar ndryshoreve të llojit int.

Me deklarimin

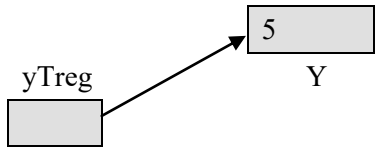
```
float *p;
```

deklarohet treguesi p, i cili mund të shfrytëzohet vetëm për të treguar ndryshoreve të llojit float.

Operatori i adresës &

Operatori & quhet **operator i adresës** (angl. address operator). Ai është operator unar, i cili e kthen adresën e operandit të përmendur sipas atij. Për shembull, nëse i kemi këto deklarime dhe inicializime:

```
int y = 5;
int *yPok;
```

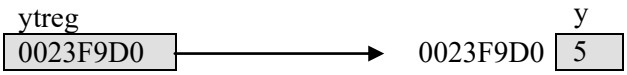


me urdhrin

```
yPok = &y;
```

treguesit yTreg i shoqërohet adresa e ndryshore y. Prandaj, për treguesin yTreg thuhet se „tregon në y“.

Nëse ndryshorja y gjendet te adresa 0023F9D0, pas realizimit të urdhrit paraprak, fitohet kjo situatë:



Shoqërimi i adresës së treguesit mund të bëjë edhe gjatë deklarimit të tij në këtë mënyrë:

```
int *yPok = &y;
```

Nuk mund të shkruhet

```
yTreg = y;
```

pasi treguesit nuk mund t'i shoqërohet tjetër vlerë, përveç adresës

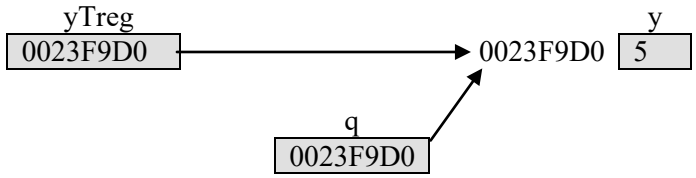
Nëse me

```
int *q;
```

deklarojmë tregues q i cili mund të shfrytëzohet për të treguear në ndryshore numra të plotë, atëherë me urdhrin

```
q = yTreg;
```

orientohet edhe ky tregues të tregojë ndryshoren y.



Nëse deklarojmë në ndryshore reale

```
float *r;
```

nuk mund të shkruhet

```
r = yTreg;
```

ose

```
r = &y;
```

pasi treguesi r nuk mund të shfrytëzon për të treguar ndryshore numra të plotë, por vetëm ndryshore reale.

Operand i operatorit të adresës & patjetër të jetë ndryshore.

Operatori për dereferencim*

Ndryshorja y mund të shënon edhe nëpërjet treguesit yTreg me *yTreg.

Operatori * quhet operator indirekt (angl. indirektion opera-tor) ose operator për derefrencim (angl. dereferencing operator). Ai e kthen vlerën të cilën e ka ndryshorja të cilës i tregon operandin e vet. Në rastin, *yTreg e kthen vlerën e ndryshores të cilës i tregon treguesit yTreg.

Për shembull, me urdhrin

```
cout << *yTreg;
```

do të shtypet vlera e ndryshores y, d.m.th., 5. Vlera e njëjtë do të shtypet edhe me urdhrin

```
cout << y;
```

Nëse deklarojmë

```
int *p = &x;
```

atëherë me urdhrin

```
*p = *yTreg;
```

Vlera e ndryshore y (= 5) do t'i shoqëron ndryshores x.

Nëse urdhri vijues është

```
*p = *p + 3;
```

Atëherë vlera e ndryshores y do të zmadhohet për 3 dhe do të fitohet 8.

Të përmendim edhe një shembull. Le ta kemimkëtë sekuencë prek urdhërave:

```
int* ip;      // ip është tregues kah int ndryshorja
int b = 47;
int *ipb = &b;
```

Në këtë mënyrë b dhe ipb janë inicializuar, por treguesi kah numri i plotë ipb e përmban adresën e b. Për të qasur kah ndryshorja b nëpërmjet treguesit, ai dereferencon në këtë mënyrë:

```
*ipb = 100;
```

Tani b përmban vlerë 100 në vend 47.

Inicializimi i treguesit

Treguesit duhet të inicializohen ose gjatë deklarimit ose ndonjë urdhri për shoqërim. Treguesi mund të jetë i inicializuar në 0, NULL ose në adresën e ndonjë ndryshore. Treguesi me vlerë NULL nuk tregon asnjë ndryshore. Inicializimi i treguesit në 0 është ekuivalenti me inicializimin e NULL, por ndërmjet programuesve më shumë shfrytëzohet NULL. Vlera 0 është vlerë e vetme e numrave të plotë e cila mund t'i shoqëron ndonjë treguesi.

Duhet të theksohet se gjatë deklarimit të treguesit pa inicializim të njëkohësishëm, ai nuk tregon askundi. Prandaj, para se të përdoret, ai patjetër të vendoset të tregojë në adresë konkrete.

Shembulli që vijon:

```
int *ip;
*ip=100;
```

do të paraqet gabim të llojit „Null pointer assignment", pasi paraqet përpjekje të shoqërohet vlera 100 e memories të pa ekzistueshme në lokacionin e memories pasi që ip nuk tregon në asnjë lokacion të memories.

Iniciativa korrekte është kjo:

```
int *ip, x;
ip = &x;
*ip = 100;
```

Shembuj

Shembulli 3.3.1
Shfrytëzimi i operatorëve & dhe *.

Në këtë program (***figura 3.3.1***) është ilustruar shfrytëzimi i operatorëve * dhe &. Lokacionet e memories shtypen me ndihmën e operatorit << sikurse numrat heksadhjetor.

```
1   /* Shfrytëzimi i & dhe * operatorët */
2   #include <iostream>
3   using namespace std;
4
5   int main() {
6       int b;          /* b është numër i plotë */
7       int*bTreg;      /* bTreg është tregues kah numri i plotë*/
8
9       b = 7;
10      bTreg = &b;     /*bTreg tregon në adresën e b */
11
12      cout << "Adresa e b është " << &b
13          << "\nVlera e bTreg është " <<bTreg;
14
15      cout << "\n\nVlera e b është" << b
16          << "\nVlera e *btreg është " << *bTreg;
17
18      cout << "\n\nTregojmë se * dhe & janë komplementar"
19          << "njëri në tjetrin\n&*bTreg= " << &* bTreg
20          << "\n*& bTreg = " << *& bTreg << endl;
21
22      cout << endl;
23      system( "Color 17" );
24      system( "pause" );
25      return 0;
26  }
```

Figura 3.3.1

Duhet të vërehet se:

a) adresa e ndryshores a dhe vlera e treguesit aTreg janë të njëjtë,
b) operatorët * dhe & janë komplementar njëri në tjetrin.

Një mundësi e daljes prej këtij program është

```
Adresa e b është 002CF71C
Vlera e bTreg është 002CF71C

Vlera e b është 7
Vlera e *bTreg është 7

Tregojmë se * dhe & janë komolementar njëri në tjetrin
&&bTreg = 002CF71C
*&bTreg = 002CF71C

Press any key to continue . . .
```

Shembulli 3.3.2

Shfrytëzimi i treguesit si argument i funksionit.

Te program i ***figura 3.3.2***, funksioni f() ka argument tregues. Me urdhrin *p = 5; e dereferencim, me të cilën ndryshon përmbajtja e ndryshores x.

```
1   //Tregues si argument i funksionit
2   #include <iostream>
3   using namespace std;
4
5   void f( int* p );
6
7   int main() {
8       int x = 47;
9       cout << "x = " << x << endl;
10      cout << "&x = " << &x << endl;
11      f( &x );
12      cout << "x = " << x << endl;
13
14      cout << endl;
15      system( "Color 17" );
16      system( "pause" );
17      return 0;
18  }
19
20  void f( int* p ) {
21      cout << "p = " << p << endl;
22      cout << "*p = " << *p << endl;
23      *p = 5;
24      cout << "p = " << p << endl;
25  }
```

Figura 3.3.2

Një mundësi e daljaes sështë:

```
x = 47
&x = 0029F940
p = 0029F940
*p = 47
p = 0029F940
x = 5
```

Tregues dhe vargje

Ekziston lidhje e ngushtë ndërmjet vargjeve dhe treguesve.

Për shembull, le të jetë deklaruar dhe inicializuar vargu

```
int z[] = {1, 2, 4, 8, 16};
```

Idntifikatori i tij z ka vlerë të adresës së elementit të parë prej vargut dhe është i llojit tregues i ndryshoreve prej llojit të elementeve të vargut. Në këtë rast, z është regues i nryshoreve numra të plotë, d.m.th., i llojit int.

Prandaj:

```
z;
```

dhe

```
&z[0];
```

e përmban adresën e elementit të parë të vargut.

Treguesit shpesh shfrytëzohen për qasje deri te elementet e vargut. Kjo është emundshme pasi *elementet e vargjeve te memoria vendose n në adresa të njëpasnjëshme.*

Le ta kemi këtë deklarim

```
int z[10], *p;
```

Me urdhrin

```
p = z;        //emri i vargut është tregues kah elementi i parë
              //kjo është e njëjtë sikurse edhe p = &z[0]
```

treguesit p i shoqërohet vlera e adresës së elementit të parë të vargut z[].

Vlera e elementit të parë prej vargut mund të fitohet me:

```
*z;
```

ose

```
z[0];
```

Shembuj

Shembulli 3.3.3

Dalja prej këtij program është dy herë adresa e njëjtë heksadhterore.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       int a[ 10 ];
6       cout << "a = " << a << endl;
7       cout << "&a[0] = " << &a[ 0 ] << endl;
8
9       cout << endl;
10      system( "Color 17" );
11      system( "pause" );
12      return 0;
13
14  }
```

Figura 3.3.3

```
a = 0038F938
&a[0] = 0038F938
```

Shembulli 3.3.4

Me këtë program (***figura 3.3.4***) tregohet se si shoqërohen vlerat e vargut a[], nëpërmjet treguesit të elementit të parë te ai.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       int a[ 10 ];
6       int i;
7       int* ip = a;
8       for( i = 0; i < 5; i++ ) {
9           *( ip + i ) = 10 * i;
10          cout << *( ip + i ) << endl;
11          cout << a[ i ] << endl;
12      }
13
14      cout << endl;
15      system( "Color 17" );
16      system( "pause" );
17      return 0;
18
19  }
```

Figura 3.3.4

Dalja e cila fitohet është:

```
0
0
10
10
20
20
30
30
40
40

Press any key to continue . . .
```

Aritmetika e adresave

Mbi treguesit mund të realizohen operacione aritmetikore, edhe atë:

- Inkrementimi i treguesit (++).
- Dekrementimi i treguesit (– –).
- Shtuarja e vlerës numër të plotë (+ ose +=).
- Zbritja e vlerës së numrit të plotë (– ose –=).
- Shtuarja (zbritja) një tregues prej tjetrit.

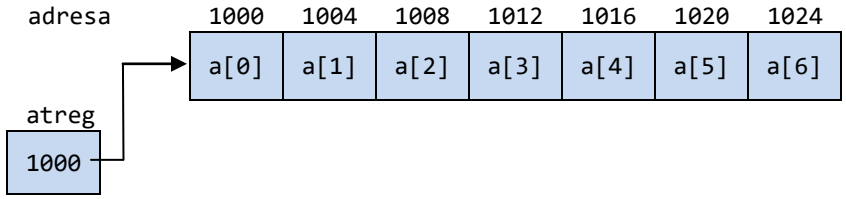
Le të jetë vargu i deklaruar numër i plotë a[] prej 7 elementeve dhe tregues aTreg i cili tregon elementin e parë të vargut:

```
int a[7];
int* aTreg = a;
```

Do të supozojmë se elementi i parë i vargut a[] gjendet te adresa e memories 1000. Kjo do të thotë se treguesi aTreg ka vlerë 1000.

Gjithashtu, do të supozojmë se ndryshoret numr i plotë zen nga 4 bajt, sikurse është paraqitur në këtë figurë.

T'i shqyrtojmë operacionet e përmedura me tregues.



a Inkrementimi dhe dekrementimi i treguesit

Me urdhrin

```
++aTreg;
```

treguesi aTreg do të inkrementohet për 1, d.m.th., do të tregojë në këtë element a[1]. Domethënë, përmbajtja e tij nuk do të jetë 1000, por do të jetë 1004.

Në rastin e përgjithshëm, nëse treguesi inkrementohet (dekrementohet) për 1, adresa e tij zmadhohet (zvogëlohet) për aq bajta sa zen ndryshorja e cila tregon. Në rastin tone, pasi elementet e vargut janë vlera numra të plotë dhe zënë nga 4 bajt, përmbajtja e aTreg do të jetë 1004. Nëse vargu a[] ishte i lojit double dhe nëse supozojmë se ndryshoret e llojit double zënë memorie prej 8 bajt, atëherë përmbajtja e aTreg do të jetë 1008.

b Shtuarja dhe zbritja e vlerës numër të plotë i/prej treguesit

Me urdhrin

```
aTreg = aTreg + 3;
```

ose

```
aTreg += 3;
```

përmbajtja e treguesit do të zmadhohet për 3 (1000 + 3 * 4 = 1012) dhe ai do të tregojë (në shembullin tone) elementin a[3].

Nëse urdhri vijues është

```
aTreg = aTreg – 2;
```

treguesi do të tregojë elementin a[1].

c Shtuarja e një treguesi në tjetrin dhe zbritja e një treguesi prej tjetrit

Nëse janë deklaruar edhe dy tregues

```
int *aP1,*aP2;
```

dhe të vendosur të tregojnë në elementin e 2-të dhe të 5-të të vargut:

```
aP1 = a+2;
aP2 = a+5;
```

atëherë me urdhrin

```
aPoк = a + aP2 – aP1;
```

treguesi aTreg do të tregojë elementin a[3].

Aritmetika e treguesit nuk ka kuptim të realizohet për lloje të tjera përveç për vargjet.

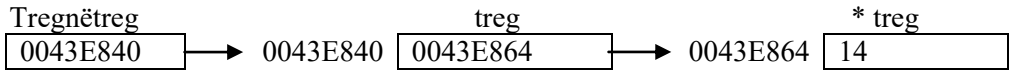
Operatorët për barazi ose operator të relacionit mund të shfrytëzojnë edhe për treguesit, ku krahasohen adresat të shkruar te treguesit. Për shembull, gjatë shqyrtimit ndonjë tregues diku, a duhet të krahasohet me NULL.

Treguesi mund të tregojë edhe në tregues të tjerë.

Për shembull, me

```
int **tregnëtreg = &treg;
```

deklarohet dhe inicializohet treguesi tregnëtreg të tregojë treguesit treg.



Vërejtje:

Vrejtëm se, nëse është deklaruar

```
int a[10], *ip;
```

atëherë këto urdhëra janë korrekte:

```
ip = a;
ip++;
```

Megjithatë, këto urdhëra nuk janë korrekte:

```
a = ip;
```

dhe

```
a++;
```

Shembuj

Shembulli 3.3.5

Me programin te figura 3.3.5 tregohet përmbajtja e treguesve prej llojit numër të plotë dhe real.

Dalja e cila fitohet mund të jetë:

```
ip = 1243916
ip = 1243920
dp = 1243828
dp = 1243836

Press any key to continue . . .
```

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       int i[ 10 ];
6       double d[ 10 ];
7       int* ip = i;
8       double* dp = d;
9       // Tregues të vlerave numra të plotë
10      cout << "ip = " << ( long ) ip << endl;
11      ip++;
12      cout << "ip = " << ( long ) ip << endl;
13      // Tregues të vlerave numra real
14      cout << "dp = " << ( long ) dp << endl;
15      dp++;
16      cout << "dp = " << ( long ) dp << endl;
17
18      cout << endl;
19      system( "Color 17" );
20      system( "pause" );
21      return 0;
22
23  }
```

Figura 3.3.5

Vërehet se ndryshoret numra të plotë zënë
(1243920 – 1243916 =) 4 bajt, ndërsa ndryshoret numra real zënë
(1243836 – 1243828 =) 8 bajt.

Shembulli 3.3.6

Me këtë shembull tregohen mënyra të ndryshme të orientimit të treguesve dhe shoqërimi ivlerave të ndryshoreve të cilave u tregojnë

```
1   // manipulimi me treguesët
2   #include <iostream>
3   using namespace std;
4
5   int main() {
6       int numra [ 5 ];
7       int* p;
8       // Shoqërimi i vlerave të vargut me tregues
9       p = numra ;             *p = 10;            // p[ 0 ]
10      p++;                    *p = 20;            // p[ 1 ]
11      p = &numra [ 2 ];       *p = 30;            // p[ 2 ]
12      p = numra + 3;          *p = 40;            // p[ 3 ]
13      p = numra ;             *( p + 4 ) = 50;    // p[ 4 ]
```

Figura 3.3.6

```
14      for( int n = 0; n < 5; n++ )
15          cout <<numra [ n ] << ", ";
16      cout << endl;
17      system( "Color 17" );
18      system( "pause" );
19      return 0;
20  }
```

Figura 3.3.6 (vazhdim)

Dalja e cila fitohet është:

```
10, 20, 30, 40, 50,
```

Urdhërat new dhe delete

Urdhri **new** e ka formën

tregues = new lloj;

Me atë te memoria krijohet ndryshorja prej *llojit* përkatës dhe të *treguesit* i shoqërohet adresa e ndryshores së krijuar. (Në të kundërtën, treguesi fiton vlerë NULL).

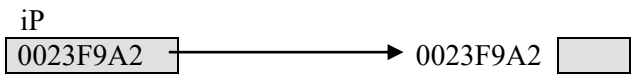
Për shembull, nëse është deklaruar treguesi

```
int *iP;
```

atëherë me urdhrin

```
iP = new int;
```

te memoria krijohet ndryshorja e pa emërtuar dhe adresa e tij i shoqërohet treguesit iP



Vlera e ndryshores së pa emërtuar mund t'i shoqëron gjatë krijimit me

```
int *iP;
ip = new int(25);
```

ose njëkohësisht

```
int *iP = new int(25);
```

Me deklarimin

```
int *aPok = new a[4];
```

rezervohet memoria për varg prej 4 elemente të llojit int, por treguesit aTreg i shoqërohet adresa e elementit të parë prej vargut.

Urdhri delete shfrytëzohet për fshirje (lirim) e lokacionit të memories.

Për shembull, me

```
delete iP;
```

lirohet (fshihet) memoria e ndryshore të cilës i tregohet treguesit iP, ku ai ngel i pa përkufizuar.

Gjatë fshirjes së vargut, shfrytëzohet sintaksa

```
delete [] tregues; ;
```

për t'u theksuar se treuesi tregon në vargun prej elementeve.

Për shembull,

```
delete [] aTreg;
```

Detyra për ushtrime

1. Shkruani program te i cili do të përkufizohen tre tregues të llojit int, float dhe char dhe tre ndryshore a, b dhe c prj llojeve përkatëse. Ndryshoret të inicializohen, por pastaj të vendoset n tregusit të tregojnë në ndryshoret përkatëse. Të shtypen vlerat dhe adresat e ndryshoreve a, b dhe c, duke shfrytëzuar treguesit.
2. Shkruani program e cila gjeneron gabim „Null pointer assignment", por pastaj tregoni se si mund të përmirësohet gabimi.
3. Krijoni program te i cili futet vargu a[]n prej elementeve numra të plotë, por pastaj shtypni vlerën e çdo elementi së bashku me adresën e tij.
4. Krijoni program te i cili futet vargu a[]n prej double elemente, por pastaj shtypni vlerën e çdo elementi së bashku me adresën e tij. Cili ndryshim e vërejti në dalje, në lidhje me detyrën paraprake?
5. Shkruani program te e cila përkufizohen vargjet me shenja x[]n dhe shenja c. Programi duhet të përgjigjet në pyetjen se shenja a përmbahet te vargu x[]n. Poashtu, për qasjen deri te elementet e vargut të shfrytëzohet treguesi.

Pyetje për kontroll të njohurive

1. Çfarë mënyra ekzistojnë për shoqërim të memories ndryshoreve?
   Sqaroni.
2. Si kryhet deklarimi i treguesit?
3. Si quhet operatori * dhe çka paraqet ai?
4. Si quhet operatori & dhe çka paraqet ai?
5. Çka është përmbajtja e treguesit, kurse çka ndryshores së treguesit?
6. Çfarëdo tregues a mund të tregojë ndryshores prej cilit do lloj?
   Sqaroni përgjigjen.
7. Çka është gabim në këtë segment programor:
   ```
   int *pok;
   char a;
   pok = &a;
   ```
8. Çka është gabim në këtë segment programor:
   ```
   int *pok;
   *pok = 5;
   ```
9. Pas vargut urdhëra:
   ```
   float *fp, f=100f;
   ```

```
fp = &f;
f = 200f;
*f = 300;
```

cila do të jetë vlera e ndryshores f?

10. Çka është gabim në këto vargje të urdhërave?

```
char *c, ch = 'A';
cout << *c;
```

11. Nëse treguesi tre tregon në ndryshoren a, atëherë paraqet shprehjen &*treg?
12. Kur ka kuptim të shfrytëzohet adresa aritmetrike te treguesit?
13. Si deklarohet (dhe inicializohet) tregues me urdhrin new?
14. Si deklarohet treguesi i vargut me urdhrin new?
15. Si fshihet treguesi i ndryshores, por si te vargu, me urdhrin delete?

3.4 Stringe

Te nëntitujt **Urdhri për shtypje** dhe **Llojet e të dhënave** prej nënpikës **1.6 Hyrje në** C++, si edhe nëntitulli Leximi dhe shtypja e të dhënave me shenja dhe string prej nënpikës **1.8 Leximi dhe shtypja e të dhënave, u nohtëm me konceptin string** (angl. string). Gjithashtu, te shumë shembuj shfrytëzuam string ndryshore ose vargje prej stringeve.

Thma se stringet janë të dhëna të përbëra prej të dhënave me shenja si sekuenca prej shenjave5. Të dhënat e këtilla trajtohen si struktura të veçanta të përbëra prej llojit të thjeshtë char, por quhen lloj string.

Me stringet mund të kryhen operacione të ndryshme sikurse: bashkimi, krahasimi, caktimi i gjatësisë së string dhe shumë të tjera. Më së shpeshti, operacionet me stringe realzohen me funksione të cilat hjenden te biblioteka <**string**> të ***bibliotekës standard të C++***. Për t'i shfrytëzuar këto funksione te programet, patjetër në fillim prej programit të kyçet biblioteka <string>, me direktivën #include.

```
#include <string>
```

Funksionet për punë me stringe

Te C++ shumë punohet me stringe. Ekzistojnë shumë funksione në bibliotekë <string>. Këtu do të sqarojmë ato të cilat më së shpeshti shfrytëzohen.

Te shembujt do t'i shfrytëzojmë këto ndryshore:

```
string s, s1, s2;
s = "C++ është gjuha më e mire programore";
```

[5] Stringet dallohen prej vargjeve tekstuale sipas asaj që ato nuk kanë fundin shenjën zero.

```
s1 = "Mirë";
s2 = "dita";
```
s3 – ndryshore string
rezultat – ndryshore string
nënstring – ndryshore string
shenja – ndryshore me shenjë
n – ndryshore numr të plotë
saktë – ndryshore logjike

Tëpërkujtohemi se çdo shenjë në string ka pozitën prej 0 (shenja e parë), 1 (shenja e dytë) etj. Shenja e fundit ka pozitën sa është numri i shenjave (gjatësia e stringut) minus 1. Për shembull, stringu s e ka gjatësinë 33. Prandaj pozitat e shenjave janë 0, 1, 2... 32.

Te teksti i deritanishëm shfrytëzuam dy operacione me stringe, edhe atë:

– Shoqërimi i një stringu në tjetrin me operatorin =.

Shembull: `s3 = s1;`

– Bashkimi i stringjeve me operatorin +.

Shembull: `s3 = s1 + s2;`

Për këto dy operacione me stringje, ekzistojnë funksione të veçanta, të cilat t'i sqarojmë në vazhdim:

- **Shoqërimi i një stringu në tjetër** **assign()**

`s3.assign(s1)` Vler e stringut s1 i shoqërohet stringut s3.

Efekti është njëjtë me urdhrin
```
s3 = s1;
```

Shembulli:
```
s1 = "Mirë";
s3.assign(s1);
```
Stringut s3 i shoqërohet konstanta "Mirë".

- **Bashkimi i stringujve** **append()**

`s1.append(s2)` I bashkon stringjet s1 dhe s2.

Efekti është njëjtë me urdhrin
```
s1 = s1 + s2;
```

Shembuj
```
s1 = "Mirë";
s2 = "dita";
s1.append(s2);
```
Stringu s1 do të fitojë vlerë "Mirdita".

Për t'i ndarë me vend të zbrazët:
```
s1.append(" ");
```

```
s1.append(s2);
```
s1 do të fitojë vlerë "Mirë dita".

Ose me urdhrin:

```
(s1.append(" ")).append(s2);
```
s1 do të fitojë vlerë "Mirë dita".

Efekti është njëjtë me urdhrin

```
s1 = s1 + " " + s2;
```

- **Gjatësia e stringut** **length()**

```
s.length()
```
Caktohet gjatësia e stringut s.

Shembulli:

```
n=s.length();
```
n do të fitojë vlerë 33.

- **Gjatësia e stringut** **size()**

```
s.size()
```
Caktohet gjatësia e stringut s.
Funksioni ka efekt të njëjtë sikurse funksioni length().
n do të fitojë vlerë 33.

Shembulli:

```
n=s.size();
```

- **Gjatësia maksimale e mundshme e string** **max_size()**

```
s.max_size()
```
Caktohet gjatësia maksimale e stringut e cila mund ta arrin stringun
s, varet prej sistemit te i cili realizohet aplikimi

Shembulli:

```
n=s.max_size();
```
n do të fitojë vlerë 2 147 483 647.

- **Krahasimi i stringëve**

```
s1.compare(s2)
```
compare()

Kontrollon se stringu s1 a është i barabartë, më i madh ose më i vogël se stringu s2.

Krahasimi kryhet në mënyrë leksokografike, ku krahasohen vlerat e shenjave të stringjeve, sipas tabelës ASCII.

Vlera pozitive, nëse s1 > s2,
Vlera negative,nëse s1 < s2.
0 nëse s1 = s2,

Nëse të gjitha shenjat prej stringjeve janë të barabarta, por njëri a vend të zbrazët, më i madh është stringu më i gjatë. Për shembull, "mirë" < "mirë ".

Shembulli

```
s1 = "mirë";
s2 = "mirë";
```

Me urdhrin

`n = s1.compare(s2);` ndryshore numër të plotë n do të fitojë vlerë –1 pasi janë të ndryshëm shenjat e pozitës së 3 -te, 'a' = 97 < 'r' = 114.

Me urdhrit:

```
n = s1.compare("mirë");          n do të fitojë vlerë 0.
n = s1.compare(s2);              n do të fitojë vlerë –1 ('a' < 'r').
n = s1.compare("mirë");          n do të fitojë vlerë 1.
                                 ('d' = 100 > 'D' = 68).
```

- **Zëvendësimi i vlerave të dy stringëve** **swap()**

  `s1.swap(s2)` I zëvendësomë përmbajtjet e stringjeve s1 dhe s2.

  Shembulli:

```
s1 ="mirë";
s2 = "mirë";
s1.swap(s2);          s1 do të fitohet vlera " mirë ", por s2 vlera
                      " mirë ".
```

- **Nënnstring prej string** **substr()**

  `s.s.substr(pozita, gjatësia)`

  Prej stringut s ndahet nënstringu prej pozitës pozita (= 0, 1... length(s) –1) me gjatësi të dhënë me ndryshoren gjatësi (> 0).

  Shembulli:

```
s1 = s.substr(17, 7);      s1 do të fitojë vlerë "program".
s1 = s.substr(17, 20);     s1 do të fitojë vlerë "gjuhë programore".
s1 = s.substr(3, 1);       s1 do të fitojë vlerë "".
s1 = s.substr(37, 3);      Do të paraqitet gabimi gjatë realizimit.
```

- **Ndarja e shenjës prej stringut** **at()**

  `s.s.at(pozita)` Ndarja e shenjës së pozitën të dhënë me ndryshoren pozita (= 0, 1, 2... s.length() – 1).

  Shembulli:

```
shenja = s.at(9);          shenja do të fitojë vlerën 'd'.
```

- **Kërkesa e nënstringut ose shenja në string [6]** **find() rfind()**

  `s.find(`*`string ose shenjë`*`)` JE jep pozitën e paraqitjes së parë të *stringut* ose *shenja* nëse kërkohet prej majtas në të djathtë.

  `s.rfind(string ose `*`shenja`* E jep pozitën e paraqitjes së parë të *stringut* ose *shenja* nëse kërkohet prej djathtas në të majtë.

  Shembuj

  ```
  znak = 'j';
  n = s.find(shenja);
  ```
  n do të fitojë vlerë 8.

  ```
  n = s.rfind(shenja);
  ```
  n do të fitojë vlerë 28.

  Nëse shenja është vend i zbrazët `' '`, n do të fitojë vlera 3 dhe 27.

  ```
  potstring = "gram";
  n = s.find(potstring);
  ```
  n do të fitojë vlerë 20.

  ```
  potstring = "grad";
  n = s.find(potstring);
  ```
  n do të fitojë vlerë –1 pasi nënstringu nuk është gjetur.

- **Fshirja e shenjave prej ndonjë pozite deri në fund të stringut** **erase()**

  `s.erase(`*`pozita`*`)` Fshirja e shenjave te stringu prej pozits së dhënë me ndryshoren *pozita* (duke kyçur edhe atë), deri në fund të stringut.

  Shembulli:

  `s.erase(16);` s do të fitojë vlerë "C++ është më i miri".

- **Zëvendësimi i nënstringut me string** **replace()**

  `s.s.replace(pozita, numrishenjave, string)`
  Te stringu s, duke filluar prej pozitës pozita, nënstringu prej këtij numërshenjave shenja zëvendësohet me stringun strinf. (Nënstringu prej numërshanje fshihet).

  `s.replace(pozita, numrishenjave, string, prejpozitës, numërshenja1)`
  Zëvendësimi mund të realizohet edhe me nënstringe prej stringut *string,* prej pozitës prejpozite dhe me gjatësin *numërshenje1*.

[6] Ekzistojnë edhe funksione të tjera për kërkesë, sikurse: find_first_of, find_last_of, find_first_not_of.

Shembulli

```
s3 = " më më";
s.replace(9, 7, s3);          s do të fitojë vlerë
                              "C++ është më më më gjuhë programore".

s.replace(9, 7, s3, 1, 3);    s do të fitojë vlerë
                              "C++ është më më  gjuhë programore".
```

- **Futja e stringut (ose nënstring) në string** **insert()**

`s.s.insert(pozita, string)`
Te stringu s, duke filluar prej pozitës pozita futet stringu *string*.

`s.insert(pozita, string, prejpozitës, numërshenja1);`
Te stringu s, duke filluar prej pozitës pozita futet nënstringu `string`, i cili përbëhet prej shenja, duke filluar prej pozitës `njëpozitë`.

Shembulli

```
s3 = " dhe më ilehtë";
s.insert(16, s3);             s do të fitojë vlerë
                              "C++ është më i miri dhe më i lehtë gjuhë
                              programore ".

s.replace(16, s3, 0, 6);      s do të fitojë vlerë
                              "C++ është më i miri dhe më gjuhë
                              programore ".
```

- **Kontrolli se stringu se është i zbrazët** **empty()**

`s.empty()` Kthen vlerën true nëse stingu s është i zbrazët.

Shembulli

```
string str;
do {
       cout << "Futni: "; getline(cin, str); }
while(str.empty());
```

Disa prej funksioneve të përmendura për punë me stringe janë ilustruar në këto shembuj

Shembuj

Shembulli 3.4.1

```
1   //Funksionet për punë me string: assign(), append(), length() dhe  compare()
2
3   #include <iostream>
4   #include <string>
5   #include <iomanip>
6
7   using namespace std;
8
9   int main() {
10
11      string s, s1, s2, s3, rezultat, potstring;
12      char znak;
13      int n, m;
14      bool saktë ;
15      s = "C++ është më e mira gjuhë programore ";
16      s1 = "Mirë";
17      s2 = " dita ";
18      s3 = "";
19      cout << " Shoqërimi një string në tjetër string " << endl;
20      cout << "s1 = " << s1 << ", s3 = " << s3;
21      s3.assign(s1);
22      cout << ", s3=(s1) " << s3 << endl;
23      cout << "Bashkimi i stringjeve" << endl;
24      s1.append(" ");
25      cout << "s1 = " << s1:
26      s1.append(s2);
27      cout << ", s2 = " << s2 << ", s1 s2 = " << s1 << endl;
28      cout << " Gjatësia e stringut " << endl;
29      n = s.length();
30      cout << " Stringut  \"" << s << "\"është i gjatë << n << " shenja.  " << endl;
31      n = s1.size();
32      cout << " Stringut  \"" << s1 << "\"është i gjatë << n << " shenja. " << endl;
33      cout << " Krahasimi i stringeve " << endl;
34      s1 = "mirë ";
35      s2 = " mirë";
36      n = s1.compare("mirë ");
37      cout << "String  \"" << s1 << "\" i krahasuar me string  \" mire \" jep    "
38          << n << endl;
39      n = s1.compare(s2);
40      cout << "String \"" << s1 << "\" i krahasuar me string  \"" << s2 << "\" jep   "
41          << n << endl;
42      n = s1.compare("Dobre");
43      cout << "String \"" << s1 << "\" i krahasuar me string \"Dobre\"  jep    "
44          << n << endl;
45
```

Figura 3.4.1

```
46      cout << endl;
47      system("Color 17");
48      system("pause");
49      return 0;
50  }
```

Figura 3.4.1 (vazhdim)

```
Shoqërimi i një stringu në tjetër string
s1 - Mirë, s3 - , s3<s1> Mirë
     Bashkimi i stringjeve
s1 = Mirë, s2 = dita, s1 s2 = Mirë dita
     Gjatësa e stringut
Stringu "C++ është gjuha prograore më e mire" është i gjatë 33 shenja
Stringu "Mirë dita" është i gjatë 9 shenja
     Krahasimi i stringjeve
Stringu "mire" i krahasuar me stringun "mire" jep 0
Stringu "mire" i krahasuar me stringun "mire" jep -1
Stringu "mire" i krahasuar me stringun "mire" jep   1
```

Shembulli 3.4.2

```
1   //Funksionet për punë me stringje : swap(), substr() i at()
2
3   #include <iostream>
4   #include <string>
5   #include <iomanip>
6   using namespace std;
7
8   int main() {
9
10      string s, s1, s2, s3, rezultat, nënstring ;
11      char shenja;
12      int n, m;
13      bool sakte;
14      s = "C++ është gjuhë programore më e mire ';
15      s1 = " Mirë ";
16      s2 = " dita ";
17      s3 = "";
18      cout << " Zëvendësimi i vlerave të dy stringjeve " << endl;
19      cout << "Para zëvendësimit s1 = " << s1 << ", s2 = " << s2 << endl;
20      s1.swap(s2);
21      cout << "Pas zëvendësimit: s1 = " << s1 << ", s2 = " << s2 << endl;
22      cout << "Ndarja e stringut prej stringut " << endl;
23      cout << "String: \"" << s << "\"" << endl;
24      s1 = s.substr(17, 7);
25      cout << "nënstring (17,7): " << s1 << endl;
26      s1 = s.substr(17, 20);
27      cout << "nënstring (17,20): " << s1 << endl;
28      s1 = s.substr(3, 1);
29      cout << "nënstring (3,1): " << s1 << endl;
30      cout << "Ndarja e shenjës prej stringut " << endl;
31    sakte= s.at(9);
```

Figura 3.4.2

```
32      cout << "Shenja në pozitën e 9-të është   << znak << endl;
33
34      cout << endl;
35      system("Color 17");
36      system("pause");
37      return 0;
38  }
```

Figura 3.4.2 (vazhdim)

```
Zëvendësimi i vlerave të dy stringjeve
Para zëvendësimit: s1 = Mirë, s2 = dita
Pas zëvënëdismir: s1 = dita, s2 = Mirë
     Ndarja e nëntringjeve prej stringut
Stringu: "C++ është gjuha programore më e mire"
Nënstring <17,7>: program
Nënstring <17,20>: gjuha programore
Nënstring <3,1>:
     Ndarja e shenjës prej stringut
Shenja në pozitën e 9-të është: d
```

Detyra të zgjidhura

Detyra 3.4.1

Të caktohet sa here paraqitet ndonjë shenjë te stringu.

Te program ***figura 3.4.3*** funksioni shenjëNëString() me të cilën caktohet sa herë paraqitet shenja e future në string (fjalia) është përkufizuar sipas funksionit kryesor main(). Prandaj, para main() është theksuar prototipi i funksionit shenjëNëString().

```
1   #include <iostream>                                    (Ctrl)
2   #include <string>
3   using namespace std;
4
5   int    shenjëNëString  ( char, string );
6
7   int main() {     // Numri i paraqitjeve të shenjës në string (me funksion)
8
9       string fjalia ;
10      char   shenjë ;
11      cout << "Futni një fjali ";
12      getline(akt , fjalia );
13      cout << "Futni cilën shenjë e kërkoni në fjali ";
14      cin >> shenjë ;
15      int numrilParaqitjeve  = shenjëNëString (shenja, fjalia );
16      if( numrilParaqitjeve > 0 )
17          cout << "\n Shenjën " << shenjë   << " në fjali  \n\"" << fjali
18          << "\"\n paraqitet " << numrilParaqitjeve    << " herë." << endl;
```

Figura 3.4.3

```
19      else
20          cout << "\nShenja " << shenja   << " te fjalia \n" << fjalia
21          << "\n nuk paraqitet." << endl;
22
23      cout << endl;
24      system( "Color 17" );
25      system( "pause" );
26      return 0;
27  }
28
29  int shenjëNëString ( char shenja , string s ) {
30      int  here = 0;
31      int gjatësia = s.length();
32      for( int pozita = 0; Pozita  < gjatësia ; pozita    ++ ) {
33          if( s.at( pozita ) ==shenja)
34             here ++;
35      }
36      return herë;
37  }
```

Figura 3.4.3 (vazhdimi)

Një dalje prej realizimit të programit është:

```
Futni një fjali: Unë jam prej Velelsi
Futni cilën shenjë e kërkoni te fjalia:  s

Shenja  s te fjalia
"Unë jam prej Velesi"

paraqitet 3 herë
Press any key to continue . . .
```

Detyra 3.4.2

Të futet string dhe të numërohet sa here ka shkronja të vogla, por sa ka shkronja të mëdha.

Për shqyrtimin se ndonjë shenjë prej stringut a është shkronjë, shfrytëzohet funksioni isalpha(). Për shqyrtimin se shkronja a është e madhe shfrytëzohet funksioni isupper(), por a është shkronjë e vogël shfrytëzohet funksioni islower().

Programi për detyrën është dhënë te ***figura 3.4.4.***

Një dalje prej realizimit të programit është:

```
Futni string: C++ është gjuha programore më e mirë
   Stringu: C++ është gjuha programore më e mire
   ka 12 shkronja të mëdha dhe 23 shkronja të vogla

Press any key to continue . . .
```

```
#include <iostream>
#include <string>
using namespace std;

int main() {     // Numri i shkronjave të mëdha dhe të vogla në string (me for)

    char shenjë ;
    int numrilShkronjaveTëMëdha = 0, numrilShkronjaveTëvogla  = 0;
    string s;
    cout << " Futni string "; getline( akt , s );
    int  gjatësi = s.length();
    for( int numërues = 0; numërues < gjatësi ; numërues ++ ) {
       shenjë = s.at( numërues );
        if( isalpha(shenjë ) )
            if( isupper(shenjë ) )
      numrilShkronjaveTëMëdha ++;
            else
     numrilShkronjaveTëVogla ++;
    }
    cout << "\n String : " << s << "\n ka  " << numrilShkronjaveTëMëdha
        << " shkronja të mëdha dhe  " << numrilShkronjaveTëVogla << "shkronja të vogla  " << endl;

    cout << endl;
    system( "Color 17" );
    system( "pause" );
    return 0;
}
```

Figura 3.4.4

Funksionet për konversion të stringut në numër

Vërejtëm se gjatë të lexuarit e stringjeve, me operatorin >> lexohet vetëm fjala e parë, d.m.th., eri te vend ii zbrazët. Prandaj, e shqyrtojmë funksionin getëine(), me të cilën lexohet gjithë stringu.

Gjithashtu, vërejtëm se nëse në një program lexojmë të dhëna edhe me operatorin >> edhe me funksionin getëine(), duhet pasur llogari se pas leximit të vlerës numerike me operatorin >>, ngel e pa lexuar shenja për vijë të re, e cila në veçanti duhet të lexohet me funksionin cin.get() ose me funksionin cin.getëine().

Që të mos kujdesemi për gabimet gjatë leximit, ndoshta është më e zakonshme ta shfrytëzojmë vetëm funksionin getëine() pasi edhe vlerat numerike te hyrja e përroit janë vargje prej shenjave (shifra ose shifra dhe shenja të tjera). Kjo është mundësuar me funksione të veçanta për konversionin e stringjeve në numra, në rast kur lexohet vlera numerike.

Janë future edhe këto funksione:

stoi(s)	– konversioni i stringut s në numër të plotë të llojit int,
stol(s)	– konversioni i stringut s në numër të plotë të llojit long,
stoll(s)	– konversioni i stringut s në numër të plotë të llojit long long
stoul(s)	– konversioni i stringut s në numër të plotë të llojit unsigned long,
stoull(s)	– konversioni i stringut s në numër të plotë të llojit unsigned long long,
stof(s)	– konversioni i stringut s në numër real të llojit float,
stod(s)	– konversioni i stringut s në numër real të llojit double,
stold()	– konversioni i stringut s në numër real të llojit long double.

Funksionet janë demostruar në këtë shembull.

Shembulli 3.4.3

```
1   // Konversioni i stringut në numër
2
3   #include <iostream>
4   #include <string>
5   using namespace std;
6
7   int main() {
8       string s;
9       int numriInt;
10      long numriLong;
11      long long numriLongLong;
12      unsigned long numriLongIPaShënuarLong;
13      unsigned long long numriLongIPaShënuarLongLong
14      float numriFloat ;
15      double numriDouble;
16      long double numriLongDouble ;
17      cout << " Futni string \"2123456789\": ";  getline(cin, s);
18      numriInt = stoi(s);
19      cout << "\t\t Numri prej llojit int është : " << numriInt << endl;
20      cout << " Futni string \"2123456789\": ";  getline(akt, s);
21      numriLong = stol(s);
22      cout << "\t\t Numri i llojit long është " << numriLong << endl;
23      cout << " Futni string \"9123456789123456789\": "; getline(akt, s);
24      numriLongLong  = stoll(s);
25      cout << "\t\t Numri i llojit long long është : " <<numriLongLong<< endl;
26      cout << "numriLongLong \"4123456789\": ";  getline(cin, s);
27      numriLongIPaShënuarLong = stoul(s);
28      cout << "\t\tBrojot od tip unsigned long e: " << numriLongIPaShënuarLong << e
29      cout << "Futni string \"18123456789123456789\": ";    getline(cin, s)
30      numriLongIPaShënuarLongLong = stoull(s);
```

Figura 3.4.5

```
31      cout << "\t\t Numri i llojit unsigned long long është :
32          <<numriLongIPaShënuarLongLong<< endl;
33      cout << "Futni string \"1.23456789\": ";  getline( akt, s);
34      numriFloat = stof(s);
35      cout << "\t\tNumri i llojit float është :  " <<numriFloat << endl;
36      cout << "Futni string \"2.123456789e-308\": ";    getline(akt , s);
37      numriDouble = stod(s);
38      cout << "\t\tNumri i llojit double është:  " <<numriDouble << endl;
39      cout << "Futni string \"1.123456789e+308\": ";    getline(akt , s);
40      numriLongDouble = stold(s);
41      cout << "\t\tNumri i llojit long double është:  " <<numriLongDouble << endl;
42
43      cout << endl;
44      system("Color 17");
45      system("pause");
46      return 0;
47  }
```

Figura 3.4.5 (vazhdimi)

Një dalje pas realizimit të programit është:

```
Futni string        "2123456789": 2123456789
 Numri i llojit int është:      2123456789
Futni string        "2123456789": 2123456789
 Numri i llojit long është:     2123456789
Futni string        "9123456789123456789": 9123456789123456789
 Numri i llojit long long është :       9123456789123456789
Futni string        "4123456789": 4123456789
 Numri i llojit unsigned long është:        4123456789
Futni string        "18123456789123456789": 18123456789123456789
 Numri i llojit unsigned long long është:           18123456789123456789
Futni string        "1.23456789": 1.23456789
 Numri i llojit float është:        1.23457
Futni string        "2.123456789e-308": 2.123456789e-308
 Numri i llojit unsigned double është :     2.12346e-308
Futni string        "1.123456789e+308": 1.123456789e+308
 Numri i llojit long double është:        1.12346e+308
```

Konversioni i numrit të stringut

Për konversionin e numrit te string shfrytëzohet funksioni to_string() :

```
to_string(numri)
```

numri mund të jetë illojit: int, long, long long, unsigned int, unsigned long, unsigned long long, float, double dhe long double.

Për shembull, segmenti programor vijues:

```
int numri1 = 11;
double numri2 = 12.345;
unsigned long long numri3 = 12345678901234567890;
cout << to_strin(numri1) << endl;
cout << to_strin(numri2) << endl;
cout << to_strin(numri3) << endl;
```

do të shtyp:

```
11
12.345000
123456789012345678 90

Press any key to continue . . .
```

Detyra për ushtrime

1. Të futet fjala dhe të shtypet fjle tij e kundërt. (Fjala e kundërt është ai që është shkruar prej prapa përpara. Për shembull, otelo e kundërta oleto).
2. Te stringu vijues të zëvendësohet fjala Makedonski me The Great.
3. „Mbreti më i madh maqedonas ka qenë Aleksandër i Maqedonisë, djali i Filip II Maqedonas.“
   Të futen emri (*emri*), mbiemri (*mbiemri*) dhe emrii shkollës (*shkolla*) dhe të shtypet:
   Unë jam emri mbiemri, nxënës në shkollën shkolla. (Ndryshoret emri, mbiemri dhe shkolla të shtypen me shkronja të mëdha).
4. Të caktohet në cilën poztë te fjalia nga detyra paraprake fillon mbiemri.
5. Te stringu i dhënë të caktohet largësia dërmjet dy fjalëve të dhëna të cilët përmbahen te ai. (Largësia është numër i shenjave ndërmjet fjalëve). Për shembull, te detyra 3 largësia ndërmjet fjalëve *mbiemri* dhe *shkolla* është 23.
6. Të kontrollohet se fjalia e future (string) a mbaron me pikë.
7. Të futen stringu "123456789" dhe të njehsohet:

```
s1 = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9;
s2 = 1 - 2 + 3 - 4 + 5 - 6 + 7 - 8 + 9;
p1 = 1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9;
```

2. Të futet numri i plotë 5 shifrorë si string dhe të njehsohet shuma e shifrave të tij.

15 Të krahet zhvendosja ciklike e shenjave të stringut për k vende majtas (ose djathtas). Stringu të jetë argument i funksionit, por vlera kthyese të jetë string me shenja të zhvendosura.

Pyetje për kontroll të njohurive

1. Te cila bibliotekë e C++ gjenden funksionet për punë me stringje?
2. Me cilin funksion kryhet zëvendësimi i vlerave të dy stringjeve?
3. Me cilin funksion kryhet gjatësia e stringut?
4. Përmend disa funksione për konversion të stringut në numëer?
5. Për çka shfrytëzohet funksioni to_string().

Termine

- **Vargu** është lloj i strukturuar prej të dhënave të afërme.
- **Vrgu** njëdimensional është vargu me një indeks.
- **Konstanta e emërtuar** e quajtur edhe **ndryshorja e konstantes** është përkufizuar konstante e cila nuk mund të ndryshojë te programi.
- **Lista e inicializuar** është lista prej vlerave të cilat shoqërojnë elementet e vargut.
- **Konstanta simbolike** jepet me direktivën paraprocesore #**define.**
- **Urdhri for i bazuar në varg** shfrytëzohet kur duhet të kryhet ndonjë operacion me të gjitha elementet e vargut, pa pasur llogari për renditjen e tyre.
- **Vargu dydimensional** është varg me dy indeksa.
- **Matrica** është termin matematikor për varg dydimensional.
- **Ndryshorja statike** është ajo e cila vendoset në adresl fikse te memoria.
- **Ndryshorja dinamike** është ajo e cila vendoset në adresë të memories e cila është e lirl në momentin e krijimit të saj.
- **Tregues** ose **ndryshore e treguesit** është ndryshorja vlera e të cilit është adresa e memories.
- **Operator i adresës &** është operator i cili e kthen adresën e iperandit i cili është përmend te ai.
- **Operatori indirekt * (operator për dereferencim)** është operator i cili e kthen vlerën e ndryshores te e cila tregon treguesi pas tij.
- **new** është urdhri për krijimin e ndryshores dinamike.
- **delete** është urdhër për fshirjene ndryshores dinamike.
- **<string>** është biblioteka në C++ e cila i përmban funksionet për punë me stringje.
- **stoi, stol, stoll, stoul, stoull, stof, stod, stold** janë funksione për konversion të stringut në numër prej llojit: int, long, long long, unsigned long, unsigned long long, float, double dhe long double.
- **to_string()** është funksioni për konversion të numrit të stringjeve.

Përmbledhje

- Vargjet janë lloje të strukturuara të të dhënave elementet e të cilave janë të dhënat prejllojit të njëjtë.
- Çdo element te vargu ka numrin e tij rendor, të quajtur indeks.
- Vargumund të jetë njëdienzional (me një indeks), dydimensional (me dy indeksa), tredimensional (me tre indeksa) etj.
- Elementeve të vargut mund t'u shoqërojnë vlera me urdhrin për shoqërim ose gjatë deklarimit me listë inicializuese.

- Vargu njëdimensional deklarohet me: *lloj emër [numër],* ku numri mund të jetë numër i plotë oe konstante e emërtuar.
- Dimensioni i vargut të inicializuar me listë të inicializuar caktohet automatikisht prej përkthyesit.
- Nëse ka më pak vlera te lista inicializuese se sa që ka elemente te vargu, elementet e të tjera të vargut janë inicializuar në vlerë difolt, sipas llojit të vargut. Por, nëse ka më shumë vlera te lista inicializuese se sa që ka elementet te vargu, paraqitet sintaksa e gabimeve.
- Dimensioni i vargut mund të jepet edhe nëpërmjet konstantes simbolike me direktivën #define.
- Në C++ nuk ka inicializim automatik të vargjeve.
- Indekset e elementeve të vargut njëdimensional me gjatësi n në C++ janë: 0, 1... n – 1.
- Urdhri for i bazuar në varg shfrytëzohet për operacione me të gjitha elementet e vargut, pa pasur llogari për renditjen e elementeve.
- Vargu dydimensional me dy indeksa.
- Indeksat e elementeve të vargut dydimensional me dimensione [m][n] në C++ janë: 0, 1... m – 1 për dimensionin e parë she 0, 1... n – 1 për dimensionin e dytë.

- Vargu dydimensional deklarohet me: l*loj emri [numri1] [numri2]*, ku *numri1* dhe *numri2* mund të jenë numër i plotë ose konstante e emërtuar.
- Vargu dydimensional mund të inicializohet gjatë deklarimit ose listës inicializuese.
- Nëse ka mjaft vlera te lista inicializuese për inicializimin e të gjitha elementeve të vargut dydimensional, tjetra inicializohet me vlera difolt të llojit të vargut.
- Nëse vargu dydimensional është inicializuar me listë inicializuese, atëherë për caktimine numrit të rreshtave dhe të shtyllave në C++ shfrytëzohet operatori sizeof().
- Vargu dydimensional mund të trajtohet si varg prej vargjeve, ashtu që çdo rresht të jetë varg njëdimensional.
- Ndryshoret statike vendose n në adresë fikse te memoria dhe ngelin në adresën e njëjtë deri në mbarimin e programit.
- Ndryshoret dinamike vendose n në adresa të cilat janë të lira në momentin e krijimit të tyre.
- Për qasje deri te ndryshoret dinamike shfrytëzohen mekanizma të treguesve.
- Treguesit mund të përmbajnë vetëm adresa të memories.
- Treguesit mund të tregojnë vetëm ndryshore të cilat kanë lloj të njëjtë si edhe treguesi.
- Operator ii adresës & e kthen adresën e operatorit pas tij.

- Një tregues mund të shoqërohet treguesit tjetër dhe atëherë të dy treguesit e ndryshores së njëjtë.
- Për fitimin e vlerave të ndryshores nëpërmjet treguesve, shfrytëzohet operatori inirekt *, i cili përmendet para treguesit i cili tregon ndryshoren.
- Treguesi mund të jetë i nicializuar në 0, NULL ose në adresën e ndonjë ndryshore.
- Treguesi me vlerë NULL nuk tregon asnjë ndryshore.
- Vlera 0 është vlera e vetme numër të plotë e cila mund t'i shoqërohet ndonjë treguesi.
- Emri i vargut ka vlerë të adresës së elementit të parë prej vargut dhe është prej llojit tregues të ndryshoreve të llojit të elementeve të vargut.
- Me treguesit mund të kryhen këto operacione: inkrementimin, dekrementimi, shtuarja e vlerës numër të plotë dhe shtuarja e në treguesi tjetrit ose zbritja e një treguesi prej tjetrit.
- Mund të deklarohet tregues në tregues.
- Me urdhrin new krijohet ndryshore dinamike te memoria.
- Me urdhrin delete fshihet ndryshorja dhe lirohet memoria që e zenë.
- Nëse te program shfrytëzohen stringje, atëherë patjetër të kaçet biblioteka <string> me direktivën #include.
- Te C++ ekzistojnë shumë funksione për punë me stringje.
- Funksionet për konversion të stringut te numri në C++ janë: stoi, stol, stoll, stoul, stoull, stof, stod, stold
- Për konversionin në string, shfrytëzohet funksioni to_string().

SHTESË

LLOJET THEMELORE TË TË DHËNAVE

Lloi	Bit	Vargu		
Numra të plotë				
short	16	-32768	te	32767
unsigned short	16	0	te	65535
int	32	-2147483648	te	2147483647
long	32	-2147483648	te	2147483647
unsigned int	32	0	te	4294967295
long long	64	-9e18	te	+ 8e18
Shenja				
char	8	'A'-'Z', 'a'-'z','0'-'9'...		
Numra real				
float	32	+/- 10E-37	te	+/- 10E38
double	64	+/- 10E-307	te	+/- 10E308
long double	32	+/- 10E-307	te	+/- 10E308

SHENJA ASCII

B

ASCII	Heksadh hjetore	Shenja	ASCII	Heksadh jetore	Shenja	ASCII	Heksadh jetore-	Shenja	ASCII	Heksadhj jetore	shenja
0	0	NUL	32	20	E zbrazët)	64	40	@	96	60	`
1	1	SOH	33	21	!	65	41	A	97	61	a
2	2	STX	34	22	"	66	42	B	98	62	b
3	3	ETX	35	23	#	67	43	C	99	63	c
4	4	EOT	36	24	$	68	44	D	100	64	d
5	5	ENQ	37	25	%	69	45	E	101	65	e
6	6	ACK	38	26	&	70	46	F	102	66	f
7	7	BEL	39	27	'	71	47	G	103	67	g
8	8	BS	40	28	(	72	48	H	104	68	h
9	9	TAB	41	29	)	73	49	I	105	69	i
10	A	LF	42	2A	*	74	4A	J	106	6A	j
11	B	VT	43	2B	+	75	4B	K	107	6B	k
12	C	FF	44	2C	,	76	4C	L	108	6C	l
13	D	CR	45	2D	-	77	4D	M	109	6D	m
14	E	SO	46	2E	.	78	4E	N	110	6E	n
15	F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

LITERATURA

Cormen, T., Leiserson. C., Rivest, R., Stein, C.: *Introduction to ALGORITHMS*, 3th Ed., MIT Press, 2009.

Deitel, P., Deitel, H.: *C++20 for Programmers, Pearson, 3rd edition, 2020.*

Deitel, P., Deitel, H.: *C++ How to Program*, Pearson, 10 edition, 2016.

Deitel, P., Deitel, H.: *C++ How to Program (Early Objects Version)*, Pearson, 9 edition, 2013.

Eckel, B.: *Thimking in C++: Introduction to Standard C++,* Vol.1, Prentice Hall, 2000.

Јованчевски, Ѓ.: *C++ Програмирање,* универзитетски учебник, Гоцмар, Скопје 2018.

Јованчевски, Ѓ., Ацковска, Н., Стојчевска, Б, Јованов, М..: *C++ Збирка алгоритми и програми,* универзитетски учебник, Гоцмар, Скопје 2017, 2007.

Јованчевски, Ѓ., Ацковска, Н., Стојчевска, Б, Јованов, М..: *C++ Програмирање за почетници*, Гоцмар, Скопје 2011.

Јованчевски, Ѓ., Стојова, Р.: *Програмски јазици*, учебник за IV година гимназиско образование, Гоцмар, Скопје 2009, 2006, 2004;

Јованчевски, Ѓ., Лазаревска, Ж.: *Програмски јазици*, учебник за III година гимназиско образование, Гоцмар, Скопје 2009, 2006;

Јованчевски, Ѓ.: *Информатика*, учебник за I година гимназиско образование, Гоцмар, Скопје 2009, 2002.

Јованчевски, Ѓ., Ацковска, Н., Стојчевска, Б.: *C++ Основи на програмирање,* универзитетски учебник, Гоцмар, Скопје 2007.

Јованчевски, Ѓ.: *Алгоритми и програми,* Гоцмар, Скопје 1993.

Knuth, D.: *The Art of Computer Programming,* Volumes 1-4A, 1st Ed., Addison-Wesley Professional, 2011.

Lippman, S., Lajoie, J., Moo, B.: *C++ Primer*, Addison Wesley Professional, 5 edition, 2012.

Horton, I.: Visual *C++ 2013,* Wiley, 2014.

Sedgewick, R.: *Algorithms in C++,* 3th Ed., Addison Wesley Professional, 1998.

Sedgewick, R., Wayne, K.: *Algorithms,* 4th Ed., Addison Wesley Professional, 2011.

Stroustrup, B: *The C++ Programming Language*, 4th Ed., Addison Wesley Professional, 2013.

Stroustrup, B: *Programming: Principles and Practice Using C++*, 2th Ed., Addison Wesley Professional, 2014.

Wilf, H.: *Algorithms and Complexity,* Taylor & Francis, 2002.

```cpp
#include <iostream>
#include <string>
using namespace std;

bool palindrom( int n, string fjala ) {
	if( fjala.at( 0 ) != fjala.at( n - 1 ) )
		return false;
	else {
		if( n > 2 ) {
			fjalë = fjalë.substr( 1, n - 2 );
			return palindrom( n - 2, fjalë );
		}
		else
			return true;
	}
}

int main() {
	string fjalë;
	cout << "Futni fjalë ose fjali: ";
	getline( cin, fjalë );
	cout << "Fjala \"" << fjalë << "\" ";
	int n = fjalë.length();
	if( palindrom( n, fjalë ) )
		cout << "Është palindrom? " << endl;
	else
		cout << "NUK ËSHTË palindrom? " << endl;

	cout << endl;
	system( "Color 17" );
	system( "pause" );
	return 0;
}
```