АУТОР

Мр Андријана Томовска

ИНФОРМАТИКА

ЗА VII РАЗРЕД ДЕВЕТОГОДИШЊЕГ ОСНОВНОГ ОБРАЗОВАЊА

2021

Аутор: мр Андријана Томовска

Издавач: Министарство просвете и науке Републике Северне Македоније

Рецензенти: проф. др Марика Апостолова Трпковска
Жаклина Наумовска
Сенар Фазлија

Илустратор: Елена Трпчевска

Превод: Анета Пауновска, Снежана Ивановић

Лектор: Јелена Иванишевић

Стручна редакција: Катерина Рајовска ООШ „Св. Кирил и Методиј“,
с. Кучевиште, Скопје

Одлуку о одобрењу уџбеника за предмет „Информатика“ за VII разред деветогодишњег основног образовања, бр. 26- 521/1 од 16. 3. 2020, донела је Национална комисија за уџбенике.

Штампа: Полиестердеј дооел, Скопље

Тираж: 13

CIP - Каталогизација во публикација
Национална и универзитетска библиотека "Св. Климент Охридски", Скопје

373.3.016:004(075.2)=163.41

ТОМОВСКА, Андријана
Информатика за VII разред деветогодишњег основног образовања / аутор Андријана Томовска ; [илустратор Елена Трпчевска ; превод Анета Пауновска, Снежана Ивановић]. - Скопље : Министарство образовања и науке Републике Северне Македоније, 2021. - 162 стр. : илустр. ; 30 см

Библиографија: стр. 160. - Речник кључих појмова (стручна терминологија): стр. 157-159

ISBN 978-608-226-968-9

COBISS.MK-ID 54313477

ПРЕДГОВОР

У складу са наставним планом и програмом сачињеним и дефинисаним од стране Бироа за развој образовања и одобреним од стране Министарства образовања и науке Републике Северне Македоније, креиран је овај уџбеник из информатике за VII разред за деветогодишње обавезно основно образовање. Према званичном наставном програму, уџбеник има пет тема.

У првој теми „Програм за прорачунске табеле" ученици ће се упознати са радним окружењем Microsoft Office Excel 2016 и моћи ће да креирају и уређују прорачунске табеле, обављају математичке прорачуне користећи формуле и функције и, на крају, графички приказ података. За практично увежбавање ових функционалности програма за прорачунске табеле, осмишљена су релевантна питања и практични задаци.

Следећа тема која ће бити обрађена је „Упознавање са информатичким концептима решавањем логичких такмичарских задатака", у којој ће ученици помоћу конкретних задатака развијати логичко размишљање како би пронашли одговарајуће решење. Док решавају логичке такмичарске задатке, ученици ће научити везу између логичког мишљења и решавања помоћу информатичких концепата и њиховог места у раду са рачунарима.

У трећој теми „Напредно програмирање у визуелном окружењу" биће направљена веза са претходном темом, и помоћу логике креираћемо блокове исказа који ће извршавати одређену активност. На тај начин, користећи визуелно програмско окружење Scratch, креираћемо једноставне програме, интерактивне игре, приче и слично.

Четврта тема посвећена је: „Програмирању помоћу стандардног структурног програмског језика". У ту сврху користићемо интегрисано програмско окружење Code::Blocks и програмски језик C++. Помоћу бројних примера и практичних задатака, ученици ће стећи вештине креирања програма.

У последњој теми: „Живот на мрежи" ученици ће моћи да креирају веб-дневнике, односно блогове са посебним нагласком на поштовање правила етичке употребе и безбедности интернета.

У овом уџбенику практични радови су основа за усвајање и савладавање нових наставних јединица и зато је припремљен посебан део са питањима и задацима који ће ученику омогућити да прошири своја знања и вештине о поменутим темама. Свака наставна јединица почиње истицањем нових

кључних појмова са којима ће се ученици сусрести при учењу, а завршава се са запамти као рубриком која указује на најважније елементе наставне јединице. После сваке лекције наведена су питања која служе за понављање наученог. Путем лекција, ученици ће наићи на подсетнике, савете, белешке који су смештени у посебне оквире и посебно су графички обележени. У прилогу је листа значења графичких елемената коришћених у уџбенику.

Важно/Савет

Да се подсетимо

Коментар

Задатак

Запамти

Кључни појмови

Покушајте

Питања

Шта ћемо научити/циљеви

На крају сваке теме постоји рубрика „Поновимо! Увежбајмо!“, као посебан додатак са питањима и задацима у вези са темом.

На крају уџбеника постоји мали речник са објашњењима стручних појмова и прилозима који су потребни за усвајање наставних јединица. Поред уџбеника, направљен је и CD са свим практичним радовима дефинисаним у директоријумима за сваку тему.

Вежбе, задаци, практични рад и питања припремљени су у складу са узрастом ученика VII разреда деветогодишњег основног образовања и у складу са њиховим предзнањем стеченим у претходним школским годинама у оквиру овог наставног предмета.

Аутор

САДРЖАЈ

Програм за прорачунске табеле

Увод у програм за прорачунске табеле
Програм за прорачунске табеле – Microsoft Office Excel 2016
Рад са подацима у табели
Уређивање табеле
Форматирање табеле
Аутоматско попуњавање података у табели
Формуле и функције у програму за прорачунске табеле
Сортирање података
Израда графикона
ПОНОВИМО! УВЕЖБАЈМО!

1. Увод у програм за прорачунске табеле

Шта ћемо научити?

Научићемо да креирамо табеле са различитим врстама података и низовима прорачуна користећи формуле и функције, уређене алаткама за форматирање и графички представљене графиконима.

Људи се у свакодневном животу често суочавају са различитим подацима који су уређени у табелама. На пример: наставници у школама праве табеле да би пратили и бележили постигнућа ученика из неког предмета, да би могли да израчунају просечну оцену по ученику и оцену целог одељења, да би могли да праве поређења постигнућа по тромесечјима, да би направили распоред часова и тако даље. Особље школске библиотеке може да креира табелу књига којима располаже школа. Ученици ће овај програм моћи да користе за различите школске и ваннаставне намене, као што су: креирање табела са подацима о ученицима у разреду, креирање телефонског именика, креирање табела о редовности у школи, преглед редовности домаћих задатака, прикупљање и обрада података приликом израде пројектног задатка, графички приказ прикупљених података итд. Родитељи могу да користе овај програм да би пратили породичне приходе, месечне трошкове итд.

Због свега тога сврха проучавања теме „Програм за прорачунске табеле **MS Office Excel 2016**“ јесте рад са прорачунским табелама, извршавање прорачуна према одређеним функцијама и креираним формулама, као и графички приказ података.

Изучавајући ову тему ученици ће се оспособити за:

- креирање табеларног документа,
- унос података различитих формата,
- уређивање/форматирање табела;
- извршавање прорачуна помоћу формула и функција,
- сортирање података у табели,
- филтрирање података у табели,
- креирање различитих врста графикона за презентацију података.

Поред теоријског учења појмова и материјала у целини, посебно је наглашен практични рад. Да бисте све то урадили, потребно је да на рачунару инсталирате пакет Microsoft Office 2016.

Слика 1: Икона Microsoft Office 2016

Кључни појмови

радна свеска, радни лист, табела, колона, ред, ћелија,
узастопне – неузастопне ћелије, активна ћелија, формула, функција, графикон.

1.1 Програм за прорачунске табеле – Microsoft Office Excel 2016

Шта ћемо научити?

Појам табеле упознали смо прошле школске године, када смо проучавали тему „Програм за обраду текста".
Табела представља скуп података који су организовани у вертикалне делове који се називају колоне и хоризонталне делове који се називају редови. Свака колона и ред се пресецају и њихов пресек назива се ћелија.

Апликација **Microsoft Office Excel 2016** представља део канцеларијског пакета Microsoft Office 2016 и омогућава представљање података на табеларни начин, уређивање прорачунских табела, извођење табеларних прорачуна и графички приказ података, креирање прегледних и професионалних извештаја итд.

111. Покретање програма. Елементи радног прозора

Да бисте креирали документ прорачунске табеле, покрените програм Microsoft Office Excel 2016 преко менија Старт и изаберите одговарајућу икону која представља програм. Појавиће се следећи прозор:

Слика 1: Отварање новог документа у програму Excel 2016

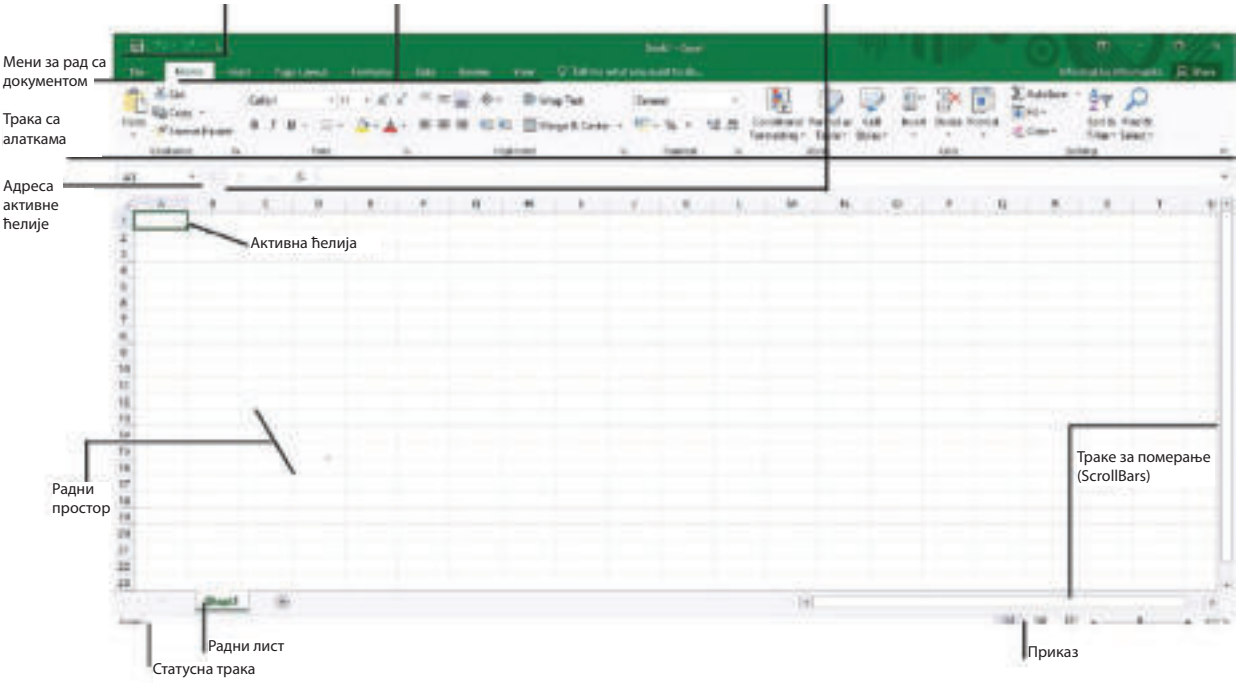
Овај прозор је подељен на два дела. На левој страни прозора приказана је листа докумената која је корисник недавно отворио и имају својство линкова, односно кликом на име документа одмах се активира изабрани документ. Са десне стране прозора корисник може отворити празан радни документ, или може одабрати готови образац (Template) документа за прераду или дораду.

Да се подсетимо!

Шта су документи? Како се још називају документи? Шта је карактеристично за њих?

Документи креирани у програму MS Excel 2016 називају се радне свеске (Workbook) и добијају наставак .xlsx. Свака радна свеска се састоји од радних листова. Сваки радни лист састоји се од табела. Окомити делови табеле називају се колоне и означени су словима (A, B, C ..... Z, AA, AB, AC ...), а хоризонтални делови називају се редови и означавају се бројевима (1, 2, 3...). Ћелија је место где се уносе подаци. Свака ћелија има име које се састоји од слова колоне и броја реда у коме се налази. На пример: A1, B5, D16 итд. Сваки радни лист састоји се од 256 колона и 65.536 редова.

Када одаберете **Blank Workbook**, активира се радни прозор програма **Microsoft Office Excel 2016**. На следећој слици приказани су делови радног документа програма.



Слика 2: Радни прозор MS Excel 2016

1. Quick Access Toolbar – трака за брзи приступ командама које се најчешће користе. Може се модификовати у складу са потребама корисника.

2. Трака менија (Menu Bar) – мени-трака која представља груписање команди према томе који су њихови задаци, у ствари, то је командна трака за одговарајућу картицу.

3. Трака са алаткама – појављује се или мења у зависности од тога који је мени изабран.

4. Адреса активне ћелије (Name Box) – представља поље за приказ имена ћелије која је изабрана, односно приказује активну ћелију.

5. Formula Bar – трака за унос формула и функција, приказ података или садржај активне ћелије.

6. Активна ћелија – ћелија у коју корисник уноси податке.

7. Sheet Tabs – трака са радним листовима.

8. Scroll Bar – траке за кретање кроз радни прозор: водоравна и вертикална.

9. Статусна трака (Status Bar) – налази се у доњем делу прозора и може се прилагодити приказу неколико опција, од којих већина пружа кориснику информације о тренутном листу.

10. Мени File – садржи основне команде за рад са документом у Excel-у.

1.1.2 Рад са документима у програму Excel

Поступак за рад са документима у програму Excel је потпуно исти као у осталим програмима канцеларијског пакета Microsoft Office, тј. све се задаје преко менија File. Али, да се подсетимо основних команди и операција са документима:

Активности / операција	Значење активности / операције
New	Отварање новог документа
Open	Отварање постојећег документа
Save	Сачувати документ
Save As	Креирати копију документа на другој локацији или под другим називом
Print	Штампање документа
Share	Дељење документа са више корисника или послати документ имејлом
Export	Промена типа документа или конвертовање документа у .pdf или . xps формат
Close	Затварање документа

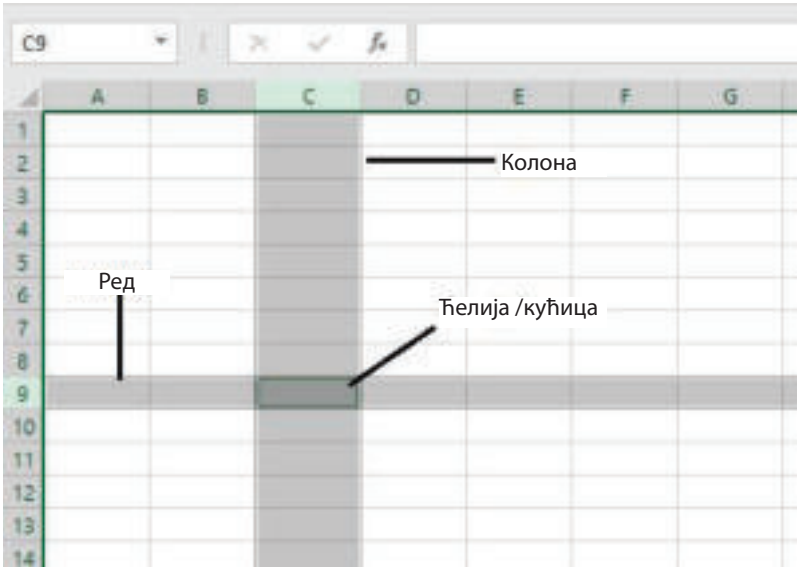
Табела 1: Активности / операције са радним документом у програму Excel 2016

Коментар!

За пресек колона и редова у овом уџбенику користићемо реч ћелија. Иако се у различитој литератури може наћи и под називом ћелија и под називом кућица.

1.1.3 Рад са основним елементима радног листа

Следећи прозор приказује основне елементе радног листа: табела, ред, колона и ћелија:



Слика 3: Основни елементи радног прозора

Да бисмо започели унос података, прво означавамо ћелију и у њу уносимо податке. Та ћелија која је означена и спремна за унос података назива се активна ћелија. На пример:

Задатак:

Направите директоријум на радној површини рачунара. Назовите га вашим именом и презименом. У овом директоријуму сачуваћете своје практичне вежбе.

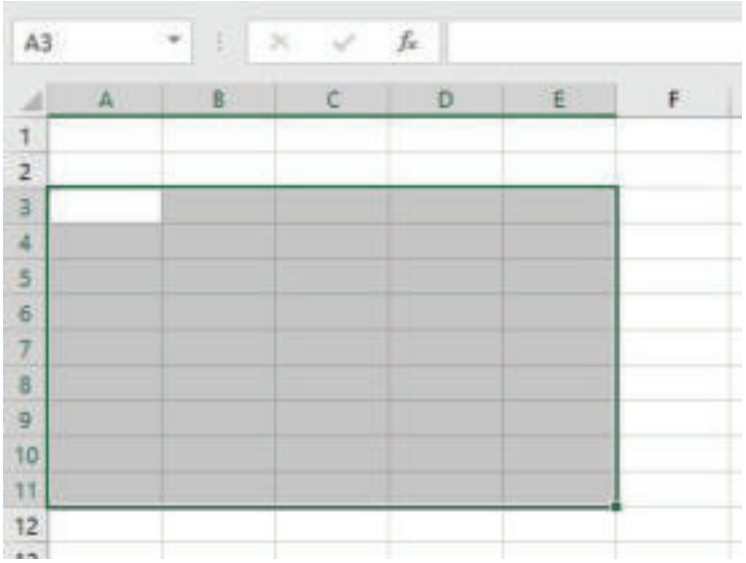
Отворите нови радни документ.
Кликните на ћелију A1 и унесите податак: вежба 1.
Кликните на ћелију A2 и унесите податак: своје име.
Кликните на ћелију B2 и унесите ваш разред.
Сачувајте документ под називом **„Vezba1.xlsx“** у директоријуму који је креиран на радној површини.

Када радите са елементима на радном листу, увек их треба изабрати и обележити. Избор или обележавање значи да покажемо на којим елементима ћемо изводити активности, као што су: избор фонта, величина фонта, боја фонта, оквири итд., као и копирање, одсецање, брисање итд.

Ћелија се обележава кликом на њу. Зависно од потребе, понекад можемо да одаберемо више ћелија одједном. **Суседне ћелије** бирамо тако што ћемо кликом на прву ћелију коју желимо да обележимо држати притиснут

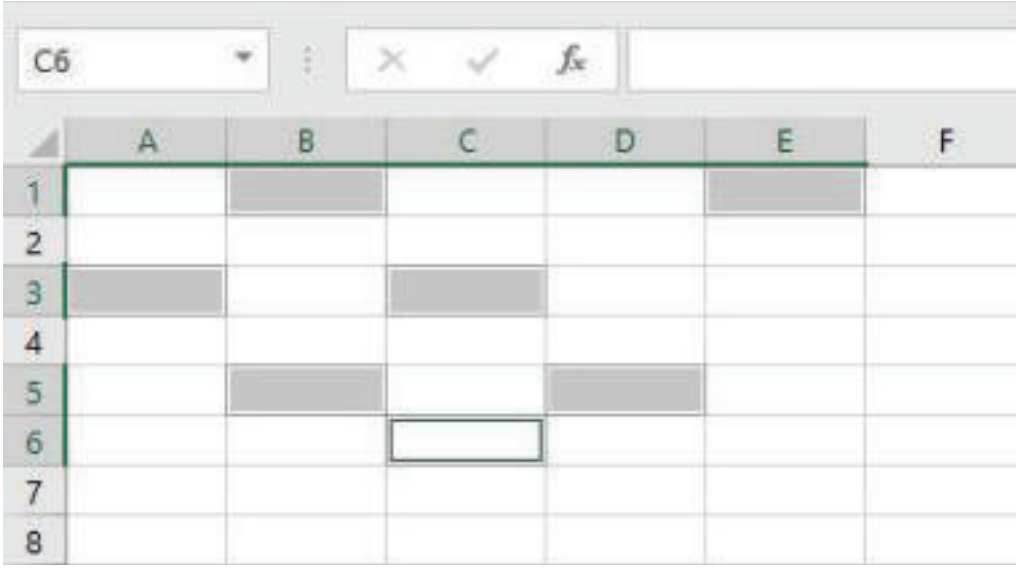
тастер **Shift** на тастатури и потом кликнемо на последњу ћелију, или кликом на прву ћелију повучемо миша до последње ћелије. На тај начин смо обележили ранг ћелија.

Ранг ћелије представља збир узастопних или суседних ћелија и обележава се на следећи начин, на пример: A3: E11.



Слика 4: Избор узастопних / суседних ћелија

Неконзистентне или **несуседне ћелије** можемо изабрати кликом на ћелију држећи тастер **Ctrl** на тастатури, а затим кликните на следећу ћелију итд.



Слика 5: Избор несуседних ћелија

За кретање кроз радни документ користићемо траке за померање (Scrooll Bars), које се налазе на десној страни и у доњем делу радног прозора, као и тастере на тастатури:

Дугме на тастатури	Кретање
Enter или ↓	Једна ћелија испод
↑	Једна ћелија изнад
←	Једна ћелија улево
→	Једна ћелија удесно
Ctrl + ←	Прва ћелија у реду
Ctrl + →	Последња ћелија у реду
Home	Почетак реда
End	Крај реда
Ctrl + Home	Почетак табеле
CTRL + End	Крај табеле

Табела 2: Кретање кроз радни прозор

Вежба 1

У документу вежбе 1, на радном листу Sheet 1 изаберите ћелије B4:F8 и помоћу алатке Fill Color, из почетног менија Home, обојите их у плаво. Именујте радни лист: суседни. Сачувајте промене направљене у документу.

У исти документ додајте нови радни лист. Именујте радни лист као несуседни. Изаберите ћелије: A2, C2, E2, G2, B3, D3, F3, A4, C4, E4, G4, B5, D5, F5, A6, C6, E6, G6, B7, D7, F7. Сачувајте промене које су направљене у документу.

Упамти!

Програм **Microsoft Office Excel 2016** користи се за презентацију података у табелама, за обављање прорачуна и графичког приказа података. Документи који се креирају у програму Excel 2016 називају се радне свеске (Workbook) и имају наставак .xlsx. Радне свеске или документи садрже радне листове (Sheet1, Sheet2.). Радни листови садрже табеле. Табела је скуп пода-

така који су организовани у вертикалне делове који се називају колоне и хоризонталне делове који се називају редови. Свака колона и ред се секу и њихов пресек назива се ћелија или кућица. Најчешће активности или операције са радним документом су: отварање новог документа, отварање постојећег документа, чување документа и штампање документа.

Питања

1. Чему служи програм Excel 2016?
2. Како се зову документи креирани у програму за прорачунске табеле?
3. Који су елементи радног прозора програма за прорачунске табеле?
4. Како су водоравни делови означени у активном документу у програму за прорачунске табеле?
5. Како се назива пресек колона и редова у програму за прорачунске табеле?

1.2 Рад са подацима у табели

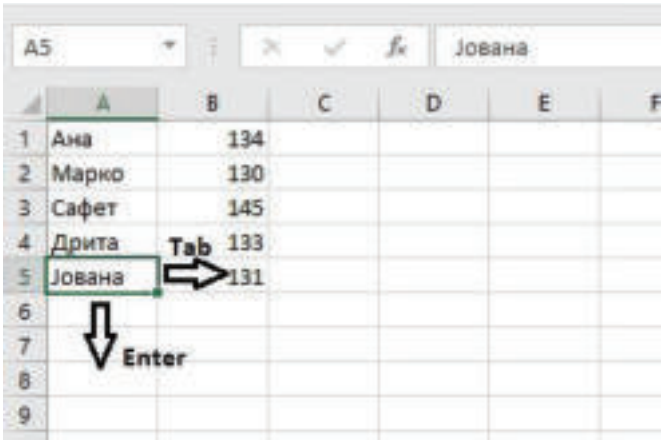
Коментар!

Алфабетски подаци и алфанумерички подаци распоређују се на левој страни ћелије, док су нумерички подаци распоређени према десној страни ћелије. Наравно, алатима за форматирање можемо их распоредити по жељи.

Да бисмо креирали табелу у програму за прорачунске табеле, најпре треба унети податке. Подаци у табели се уносе у активној ћелији, а затим настављамо са уносом помоћу тастатуре, користећи тастере са стрелицама за кретање по радном прозору као што смо приказали у претходној наставној јединици.

На пример, да бисмо направили списак ученика и њихове висине изводимо следеће кораке:

1. кликните на ћелију A1 и унесите име ученика;
2. притисните **Enter** на тастатури да бисте прешли у ћелију A2;
3. поново унесите име и понављајте поступак све док не унесете десет (10) имена;
4. кад унесете последње име, притисните тастер **Tab** на тастатури, које ће вас одвести до једне ћелије удесно. Тамо бројем унесите висину ученика;
5. помоћу тастера за кретање кроз радни документ наставите да уносите податке и за остале ученике на списку;
6. сачувајте документ у директоријуму на радној површини под називом **„Vezba2.xlsx“**.



Слика 1: Унос података у табелу

На основу практичног задатка можемо приметити да се подаци које уносимо у активну ћелију такође појављују на **Formula Bar**, односно на **траци за формуле**. Кликом на било коју ћелију која садржи податак, адреса ћелије се појављује у пољу **Name Box**, док се садржај ћелије приказује у **Formula Bar**.

1.2.1 Типови података

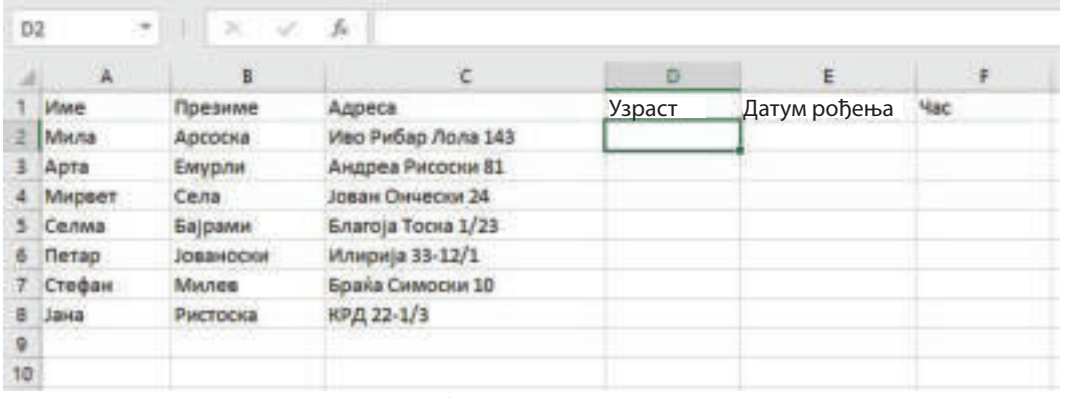
Подаци које уносимо у табелу могу бити **алфабетски**, **нумерички и алфанумерички**. Алфабетски подаци садрже само слова, тј. текст, нумерички подаци су низови бројева, док алфанумерички подаци представљају комбинација слова/текста и бројева.

Алфабетски и алфанумерички подаци

За унос текстуалних података (**алфабетски**) и комбинација текста и бројева (**алфанумерички подаци**) урадићемо следећи задатак. У документу **„Vezba2.xlsx“**, додајмо нови радни лист у коме ћемо креирати табелу: основни подаци о ученицима. Преименујте радни лист у назив **„Подаци“**.

1. У ћелијама A1, B1, C1, D1, E1, F1 и G1 креираћемо насловне ћелије: име, презиме, адреса, године узраста, датум рођења и сат.

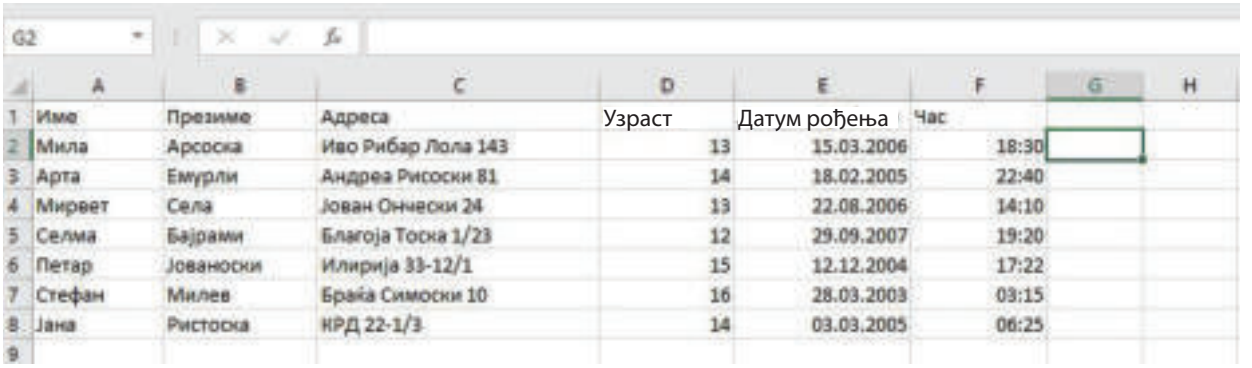
2. Помоћу тастера за кретање кроз радни документ, у колоне унесите податке: име, презиме, адреса.



Слика 2: Унос алфабетских и алфанумеричких података

Нумерички подаци

Када говоримо о нумеричким подацима, можемо утврдити да постоје различите **врсте нумеричких података**, као што су: циљни бројеви, позитивни и негативни бројеви, датум, сат, валута итд. Додајте табелу „**Подаци**“. Пошто су унесени сви подаци сачувајте промене у документу.



	A	B	C	D	E	F	G	H
1	Име	Презиме	Адреса	Узраст	Датум рођења	Час		
2	Мила	Арсоска	Иво Рибар Лола 143	13	15.03.2006	18:30		
3	Арта	Емурли	Андреа Рисоски 81	14	18.02.2005	22:40		
4	Мирвет	Села	Јован Ончески 24	13	22.08.2006	14:10		
5	Селма	Бајрами	Благоја Тоска 1/23	12	29.09.2007	19:20		
6	Петар	Јованоски	Илирија 33-12/1	15	12.12.2004	17:22		
7	Стефан	Милев	Браќа Симоски 10	16	28.03.2003	03:15		
8	Јана	Ристоска	КРД 22-1/3	14	03.03.2005	06:25		
9								

Слика 3: Унос нумеричких података

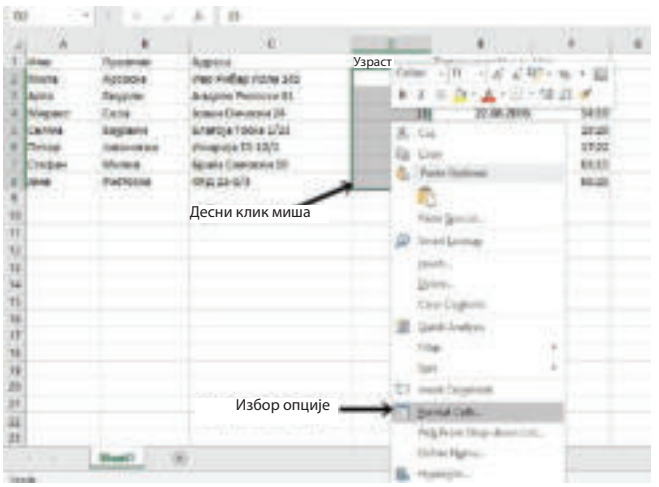
Можемо приметити да смо приликом уноса датума дан одвојили од месеца тачком (.). Одвојили смо и месец од године такође тачком. Када уносимо датум, такође можемо користити косу црту (/) или само црту (-). Excel затим препознаје ове податке као датуме. Када уносимо сат, раздвајамо сате од минута и секунди знаком две тачке (:).

1.2.2 Формат бројних (нумеричких) података

У табелама, приликом уноса података, можемо да изаберемо различит формат података. Избор формата података изводимо на следећи начин:

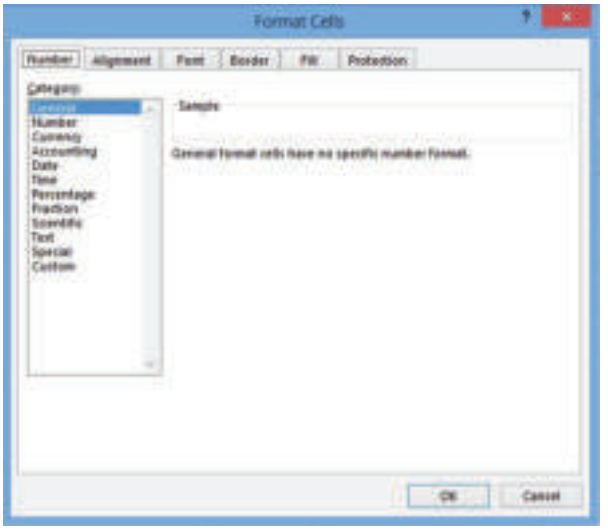
1. бирамо податке чији формат желимо да променимо;
2. десним кликом на тастеру миша код изабране ћелије изаберемо опцију **Format Cells**.

Појавиће се следећи прозор:



Слика 4: Избор опције Format Cells

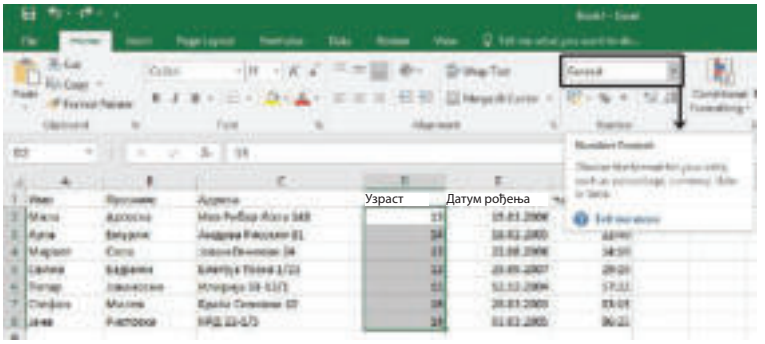
Након избора опције Format Cells, појавиће се прозор из кога бирамо картицу **Number**, а затим бирамо категорије:



Слика 5: Прозор форматирања ћелија Format Cells

1. **General** – општи формат.
2. **Number** – цели бројеви и децимални бројеви, позитивни и негативни бројеви.
3. **Currency** – формат валуте: денари, евра, долари, лире, фунте итд.
4. **Accounting** – приказује знакове валута на левој страни вредности и приказује цртицу (-) код вредности нула (0).
5. **Date** – формати датума.
6. **Time** – формати времена.
7. **Percentage** – приказ формата процента.
8. **Fraction** – формати фрагмената.
9. **Scientific** – приказ броја у експоненцијалном формату.
10. **Text** – низови текста.
11. **Special** – специјални/посебни формати.
12. **Custom** – избор додатних формата података.

Осим коришћења приказаног поступка, за одабир другог формата бројчаних података можемо користити и мени Home и опцију Number Format, као што је приказано на следећој слици:



Слика 6: Избор формата података путем менија Home

Бирамо формат датума: дд.мм.гг и формат времена, односно сата, у коме ће се приказати сат и минути, као и опција преподне/поподне. Сачувајте промене у документу.

Запамти!

Приликом креирања табеле у програму за прорачунске табеле, подаци се уносе у активну ћелију. Постоје различите врсте података: алфабетски, нумерички и алфанумерички. Нумерички подаци могу имати различите формате: цео број, децимални број, позитивни и негативни број, датум, време, валута итд. Формате података бирамо путем опције Format Cells, десним кликом миша, или путем опције Number Format у менију Home.

Питања

1. У коју ћелију се уноси податак?
2. Који тастер или тастере треба кликнути након уноса податка у ћелију?
3. Наброј различите формате бројчаних података!

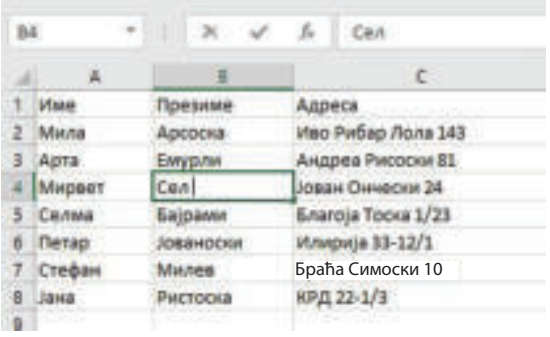
Експоненцијални формат значи замену дела броја са E + n, у коме је E (експонент) помножен са претходним бројем 10 n-тог степена. На пример, дводецимални научни формат приказује 12345678901 као 1,23E + 10, што је 1,23 пута 10 на 10-и степен.

1.3 Уређивање табеле

Када у програму за прорачунске табеле креирамо табелу, на први поглед изгледа да је све у реду док не проверимо јесмо ли направили техничку грешку, да ли смо унели погрешан податак, недостаје ли цео ред или цела колона у табели. Наравно да нећемо избрисати целу табелу и почети изнова, већ ћемо исправити учињене грешке. Да бисмо извршили промене, отворите **„Vezba2.xlsx“**.

1.3.1 Промена и уређивање унетих података

Ако желимо да направимо промене податка у ћелији B4, односно да унесемо нови податак уместо постојећег, једноставно кликнемо на ћелију и почнемо да пишемо. У том тренутку на статусној траци пише **Ready**, а то значи да се налазимо у режиму уношења новог податка. На пример: уместо Села, написаћемо Селими.

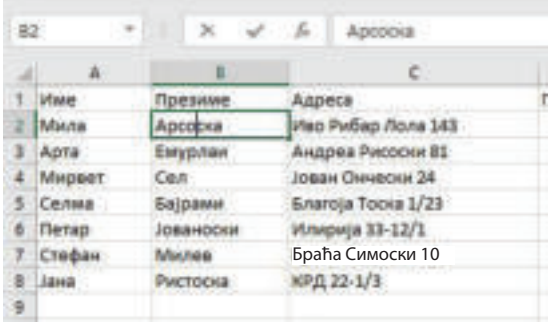


	A	B	C
1	Име	Презиме	Адреса
2	Мила	Арсоска	Иво Рибар Лола 143
3	Арта	Емурли	Андреа Рисоски 81
4	Мирвет	Сел	Јован Ончески 24
5	Селма	Бајрами	Благоја Тоска 1/23
6	Петар	Јованоски	Илирија 33-12/1
7	Стефан	Милев	Браћа Симоски 10
8	Јана	Ристоска	КРД 22-1/3

Слика 1: Промена податка у ћелији

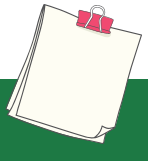
Такође, у ћелији B2 треба да додамо један знак, односно слово у презимену.

Тада, двапут кликнемо мишем у ћелију B2, поставимо курсор тамо где треба да буде уметнуто слово и тастатуром упишемо слово, у нашем случају слово „в“. Тада на статусној траци пише **Edit**, што значи да смо у режиму промене података.



	A	B	C
1	Име	Презиме	Адреса
2	Мила	Арсовска	Иво Рибар Лола 143
3	Арта	Емурлан	Андреа Рисоски 81
4	Мирвет	Сел	Јован Ончески 24
5	Селма	Бајрами	Благоја Тоска 1/23
6	Петар	Јованоски	Илирија 33-12/1
7	Стефан	Милев	Браћа Симоски 10
8	Јана	Ристоска	КРД 22-1/3
9			

Слика 2: Додавање знака / слова приликом уноса



Коментар!

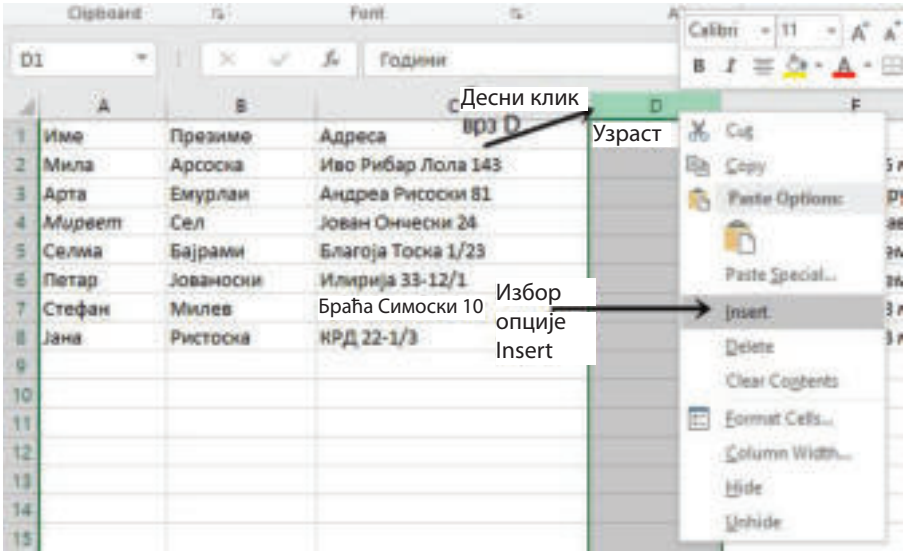
Да бисте брзо уметнули више од једне колоне или реда, изаберите више редова/колона, кликните десним тастером миша на њих и изаберите опцију Insert.

Податак или податке из ћелија можемо избрисати на следећи начин: прво ћемо одабрати које податке желимо да избришемо, а затим кликнемо на тастер **Delete** на тастатури.

1.3.2 Активности са колонама и редовима

У креирану табелу можемо додавати и брисати колоне и редове, у зависности од потребе корисника. Поступак **уметања нове колоне** је следећи:

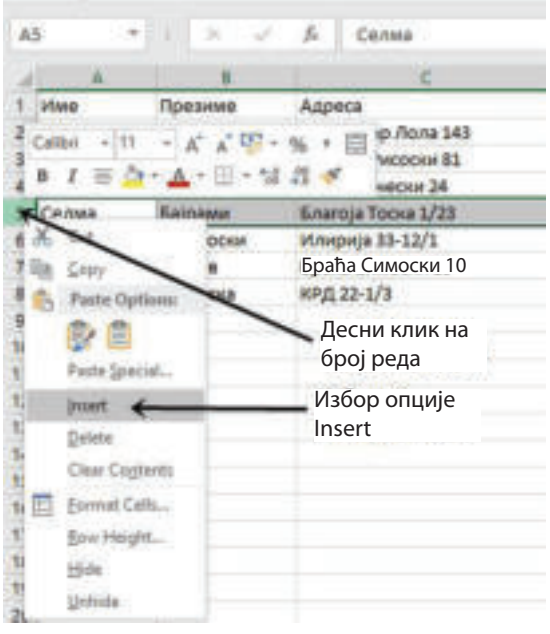
1. означимо колону испред које желимо да додамо нову колону;
2. кликнемо десним кликом на слово означене колоне;
3. изаберемо опцију **Insert**.



Слика 3: Уметање колоне у табелу

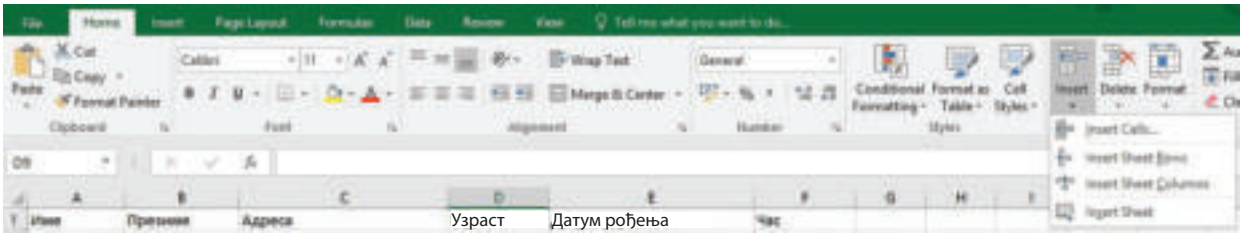
Поступак је исти за **уметање реда у табелу**:

1. означимо колону испред које желимо да додамо нови ред;
2. кликнемо десним кликом на број означеног реда;
3. изаберемо опцију **Insert**.



Слика 4: Уметање реда у табелу

Такође, може се извршити уметање колоне и реда у табелу путем менија **Home** и избором одговарајућег алата за уметање колоне и реда:



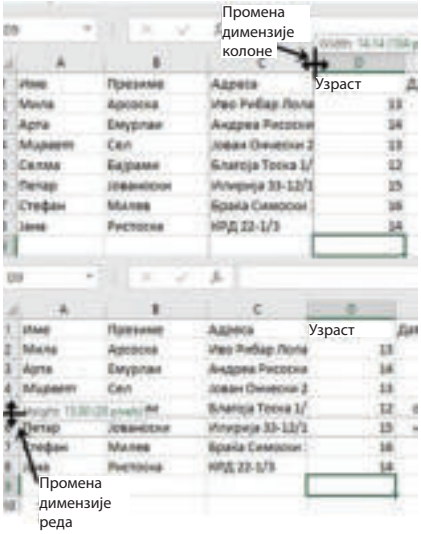
Слика 5: Уметните колону и ред путем менија

Покушајте!

Пошто смо научили процедуру за уметање колоне и реда у табели, покушајмо да обришемо колону или ред, ако знамо да би уместо опције Insert требало да изаберемо Delete.

1.3.3 Промена димензија колона и редова

Приликом уноса података у ћелије, често је неопходно да **мењамо димензије колона и редова**. Поступак за произвољну промену димензија колоне, односно реда, приказан је на следећој слици:



Слика 6: Промена димензије колоне и реда

На основу слике можемо приметити да прво постављамо показивач између слова колоне, тј. између бројева редова, добијамо курсор, притиснемо и повучемо. Код промене димензије реда, курсор се мења за 90°, а поступак је исти.

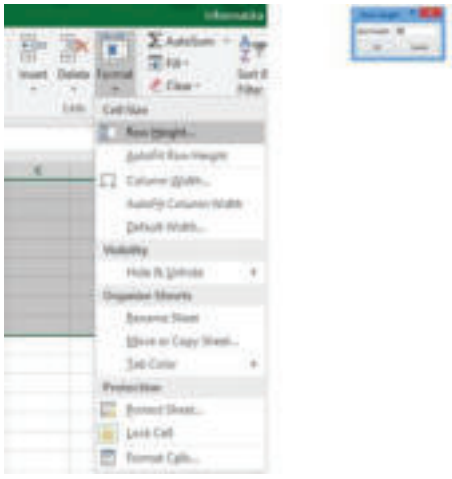
Коментар!

Можете брже променити димензије колона/редова помоћу двоструког клика на линију после ознаке колоне/реда.

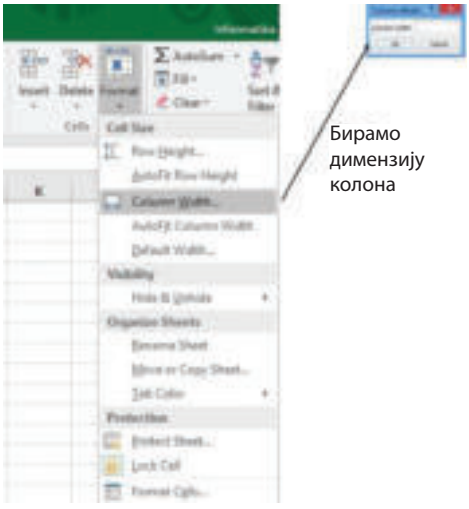
Задатак

Отворите документ „Vezba2.xlsx“ и промените димензије колоне у којој се налазе датум рођења и насловни ред. Затим изведите поступак промене величине колоне/реда путем опције Format из менија Home, како бисте добили исте димензије за све колоне и редове.

Али када корисник жели да повећа или смањи колоне и редове за тачно одређену вредност, онда путем главног менија Home изаберемо алатку Format:



Слика 7: Промена димензија редова преко опције Format из менија Home



Слика 8: Промена димензије реда преко опције Format из менија Home

Запамти!

Уређивање или едитовање документа је поступак корекције. У те намене податке можемо променити кликом на ћелију и одмах почињемо да пишемо или двоструким кликом и постављањем курсора на место где желимо да урадимо корекцију. Путем менија Home можемо уметнути колону и ред да бисмо избрисали колону и ред, али и да бисмо променили димензије.

Питања

1. Какав је поступак за промену података у ћелији?
2. Како променити димензије колона и редова: Наведите кораке!

1.4 Форматирање табеле

Да се подсетимо

Подсетите се шта је форматирање! Који се алати најчешће користе за форматирање? Можете ли објаснити о чему највише треба водити рачуна приликом форматирања?

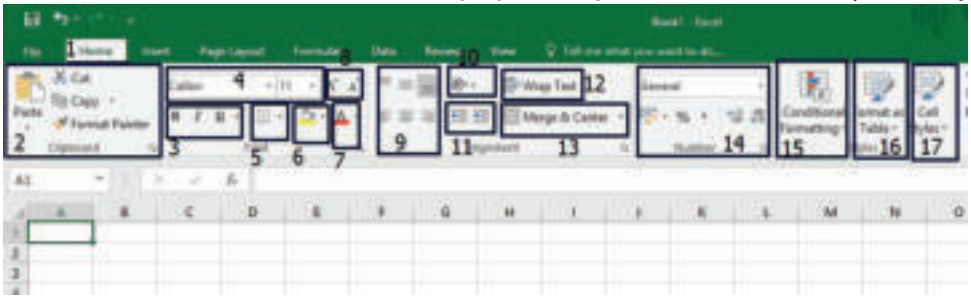
Изглед ћелија, колона и редова завршава се форматирањем изгледа табеле.

Форматирање података заправо представља додавање ефеката који побољшавају изглед табеле и тако омогућавају већу прегледност података у табели.

Форматирање ћелија значи:

- променити атрибуте фонта;
- поравнање података у ћелијама;
- приказивање текста под углом;
- спајање и дељење ћелија;
- додавање оквира и линија у једну или више ћелија;
- додавање боје ћелији и додавање ефеката.

Али пре него што се предузме било каква радња, ћелије прво морају бити селектоване. Већина алатки за форматирање се налази у менију **Home**:

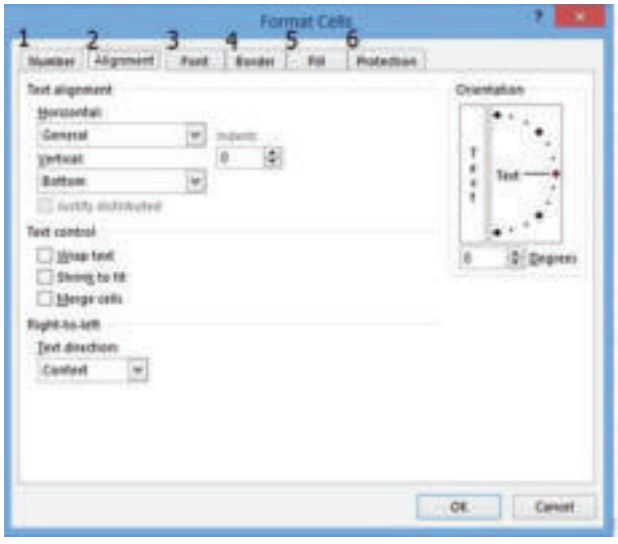


Слика 1: Алатке за форматирање помоћу менија Home

1. Мени Home.
2. Наредбе за копирање, одсецање, уметање и алатке за форматирање.
3. Стилови: подебљано, косо и подвучено.
4. Избор фонта и величине фонта.
5. Избор ивица.
6. Избор боја ћелија.
7. Избор боје фонта.
8. Алати за повећање/смањење слова.
9. Оријентација података.
10. Увлачење.
11. Подела текста.

12. Спајање ћелија и подешавање центра.
13. Избор типа података.
14. Условно форматирање.
15. Форматирање табеле.
16. Стилови ћелије.

Форматирање табеле, осим кроз почетни мени **Home**, може се извршити помоћу опције **Format Cells** пошто се претходно одаберу ћелије из табеле:



1. Избор формата податка
2. Редослед података
3. Избор фонта
4. Избор ивица и опције за ивице
5. Попуњавање ћелија/кућица бојом
6. Чување радног листа

Слика 2: Форматирање података путем менија Format Cells

Користећи таб Alignment, корисник може направити редослед садржаја ћелија, тј. уредити податке у ћелијама, тако да у Text Alignment изабере редослед података хоризонтално и вертикално, као и оријентацију или смер текста у ћелији.

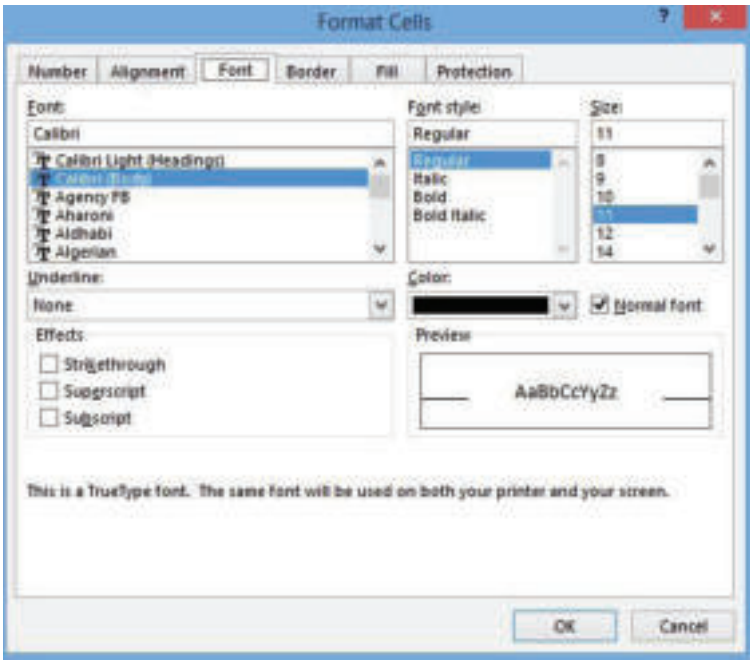
Додатне измене врше се и избором следећих опција:

Wrap Text – приказује текст који ће бити распоређен у више редова.

Shrink To Fit – смањује текст како би се уклопио у ћелију без промене димензија ћелије.

Merge Cells – омогућава спајање одабраних ћелија.

Одабиром таба **Font** бира се облик писања слова, стилови, величина и остали ефекти повезани са фонтом:

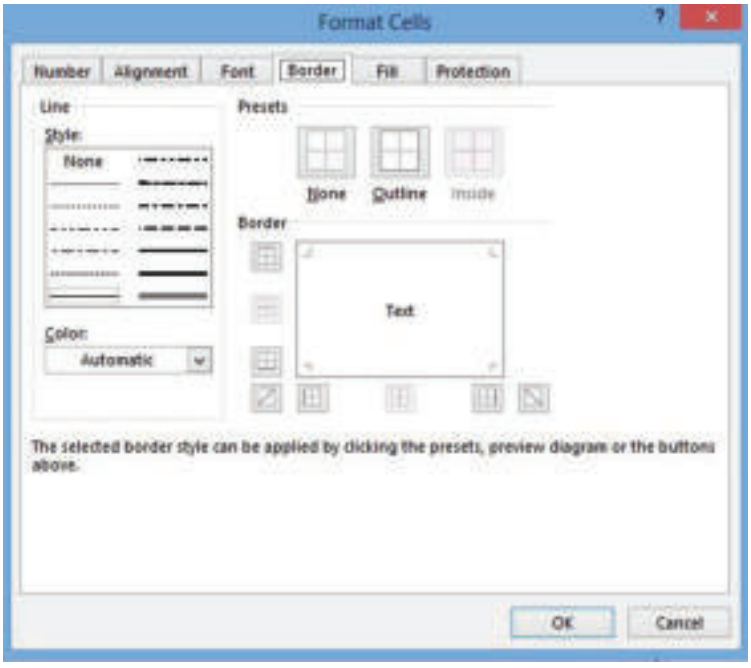


Слика 3: Избор алатки за фонт

Коментар!

Опције за фонт, стил фонта и величину фонта можете одабрати и помоћу менија Home.

У опцији **Border**, бира се стил, дебљина и боја ивица на ћелијама у табели:

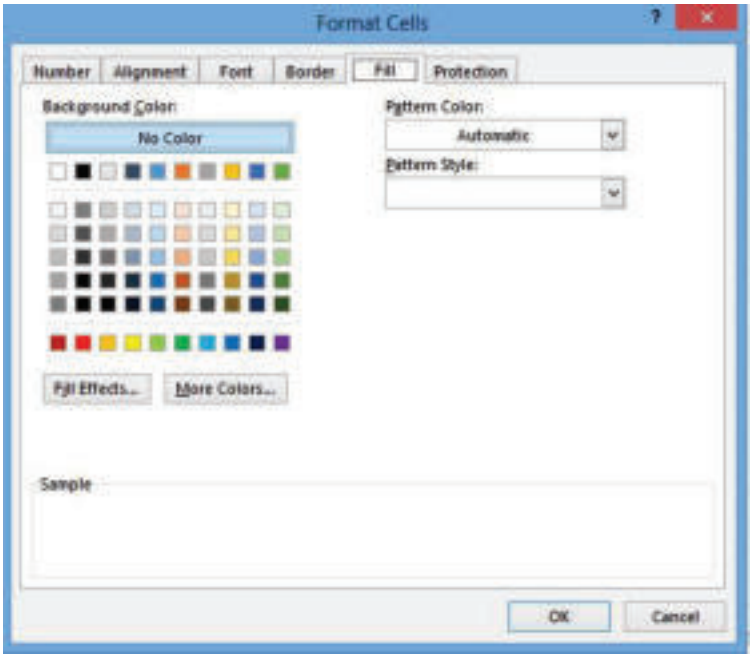


Слика 3: Избор алатки за фонт

Коментар!

Мени Home садржи алатке за уређивање ивица табеле, као могућност за бржи и лакши приступ.

Помоћу опције Fill бира се боја за попуњавање ћелија:



Слика 5: Избор боје позадине ћелије

Задатак

Отворите нови радни документ и на првом радном листу (Sheet1) направите распоред часова:

- у ћелије A1, B1, C1, D1, E1 унесите имена дана у седмици у којима похађате наставу;
- у ћелијама које се налазе испод насловних ћелија унесите предмете према распореду часова;
- додајте нову колону испред колоне понедељак и унесите редни број часова;
- додајте нови ред изнад насловних ћелија;
- спојте првих шест ћелија и напишите наслов „Распоред часова“;
- насловним ћелијама у којима су записана имена дана задајте фонт Calibri, величина четрнаест (14), подебљани и искошени стил, боја слова тамноплава и боја позадине ћелија светлоплава;
- целој табели додајте ивице;
- подесите димензије колона/редова по потреби;
- сачувајте документ у директоријуму на радној површини.

Такође, може се изабрати форматирање ћелија готовим форматом избором алата **Cells Styles** из менија **Home**, као што је приказано на слици. Али ћелије треба претходно селектовати да би стил који је корисник изабрао од понуђених опција био примењен:

Слика 6: Избор стилова ћелија

Осим готових формата ћелија, корисник може изабрати готов формат табела помоћу алатке Format As Table из менија Home, приказан на следећој слици:

Слика 6: Избор стилова ћелија

Коментар

Осим стандардног форматирања табеле, постоји и посебно форматирање, тј. условно форматирање. Њему приступамо из менија Home и помоћу опције Conditional Formatting. Покушајте да употребите овај начин форматирања у некој од креираних табела.

Задатак

Додајте нови радни лист и креирајте табелу у коју ћете унети податке за почетак и крај часова и време трајања одмора између часова. Форматирајте табелу примењујући стилове ћелија (**Cell Styles**) из менија **Home** или избором припремљеног формата табеле користећи опцију **Format Table** опет из менија **Home**.

Запамти!

Форматирање табела значи додавање ефеката табели, који се односе на промену начина, формата, оријентације и распоређивања података у ћелији. Приликом форматирања табела, најчешће се користе алатке из менија Home или наредба **Format Cells**. Постоје и унапред припремљене форме ћелија и табеле и њих бирамо из менија **Home** и помоћу команди **Cells Styles** и **Format As Table**.

Питања

1. Што је форматирање?
2. Која је разлика између форматирања и уређивања табела?
3. Које алатке се најчешће користе за форматирање табела?
4. Како се уређују распоређивање и оријентација текста у ћелијама?
5. Које опције можемо изабрати помоћу наредбе Format Cells? Можемо ли приступити овим опцијама на други начин? Објасни!
6. Који је поступак за раздвајање текста?
7. Чему служи наредба Merge Cells?
8. Чему служи опција Cells Styles из менија Home?
9. Да ли користите унапред припремљен формат табела? Који је поступак за избор?

1.5 Аутоматско попуњавање података у табели

Да се подсетимо

Да ли се сећате шта је рачунарска обрада података? Које су предности рачунарске обраде података? Можете ли да објасните појам „аутоматска“ обрада?

Веома корисна алатка за аутоматско попуњавање података у табели је **Excel**. Ова алатка нам омогућава да уместо ручног уношења података, за попуњавање ћелија користимо функцију Autofill. Аутоматско попуњавање је једна од главних функција табеларног процесора зато што се потребни подаци уносе у табелу брже него ручно.

У ствари, **Autofill** у **Microsoft Excel-y** нам омогућава ефикасније креирање табела, чиме брзо попуњавамо ћелије низом података и то не само једноставне нумеричке вредности, већ и друге типове података и формула. Осим тога, можемо креирати појединачне листе које кориснику омогућавају да не губи време изнова уносећи исте вредности.

Али, пре него што упознамо функцију и применљивост аутоматског попуњавања ћелија, научимо значења различитих форми показивача.

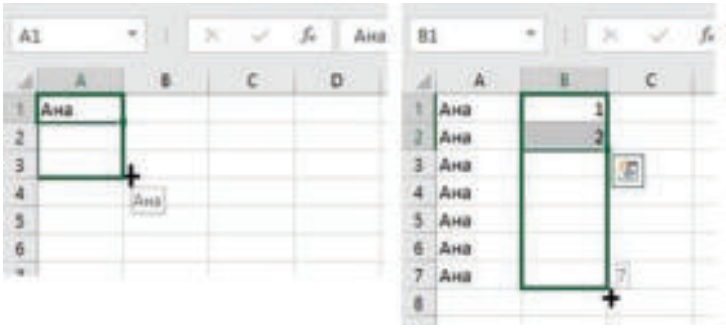
Показивач	Значење
+	Аутоматско попуњавање ћелија
	Копирање садржаја ћелије
	Премештање садржаја ћелије
	Мењање димензије колоне
	Мењање димензије реда
	Означавање колоне
	Означавање реда
	Означавање ћелија

Табела 1: Форме показивача

Да бисмо могли да ћелије попуњавамо аутоматски, користићемо следећи показивач: +. Притом је поступак за аутоматско попуњавање ћелија следећи:

1. кликнемо на икону коју ћемо означити као активну ћелију и уносимо податак;

2. после уношења податка на тастатури кликнемо Enter;
3. мишем приступимо доњем десном углу ћелије у којој се налази податак;
4. добија се показивач за аутоматско попуњавање;
5. кликнемо и повучемо до ћелије у којој желимо да извршимо аутоматско попуњавање.



Слика 1: Аутоматско попуњавање ћелија

Коментар!

У програму за прорачунске табеле могу се креирати листе, као на пример: дани у недељи, месеци у години и слично. Поступак креирања листе полази од менија File, Options, менија Advance и бирамо Edit Custom List. Покушајте да креирате једну припремљену листу!

Задатак

1. отворите нови радни документ у Excel-у 2016;
2. у радном листу 1 (Sheet 1) у колони A, креирајте низ сачињен од парних бројева користећи аутоматско попуњавање ћелија, почевши од броја два (2) до броја двадесет и два (22);
3. у колони B на истом радном листу креирајте низ од непарних бројева, почевши од броја три (3) па до двадесет и три (23);
4. у колони C креирајте низ од имена Ана, у колони D низ од имена Ана Марија;
5. у колони E креирајте низ који ће се мењати за вредност пет (5) почевши од нуле (0);
6. у колони F попуните низ 100, 99, 98… 89.

Запамти!

Аутоматско попуњавање ћелија је једна од многих предности програма за прорачунске табеле јер нам омогућава да на лакши и бржи начин уносимо податке, креирамо низове, попуњавамо ћелије. Да бисмо извршили аутоматско попуњавање ћелија, користимо овај показивач: +. Аутоматско попуњавање се може обављати не само текстуалним подацима већ и бројевима, бројним подацима, формулама и функцијама различитих формата. У програму за прорачунске табеле могу се креирати и посебне листе које ће бити искоришћене за аутоматско попуњавање ћелија.

1.6 Формуле и функције у програму за прорачунске табеле

Као што смо навели у самој дефиницији, осим што нам омогућава табеларни и прегледни приказ података, програм Excel нам омогућава и да обављамо прорачуне. Према томе, главни циљ примене формула и функција у програму за прорачунске табеле је обављање брзих и прецизних прорачуна.

1.6.1 Формуле

Формуле у програмима за прорачунске табеле креирају се од података и различитих математичких операција између њих. Excel 2016 у формулама користи стандардне знакове који су приказани у следећој табели:

Знак	Математичка операција
+	Сабирање
-	Одузимање
*	Множење
/	Дељење
^	Степен/Експонент

Табела 1: Математичке операције за извршавање прорачуна

Белешка!

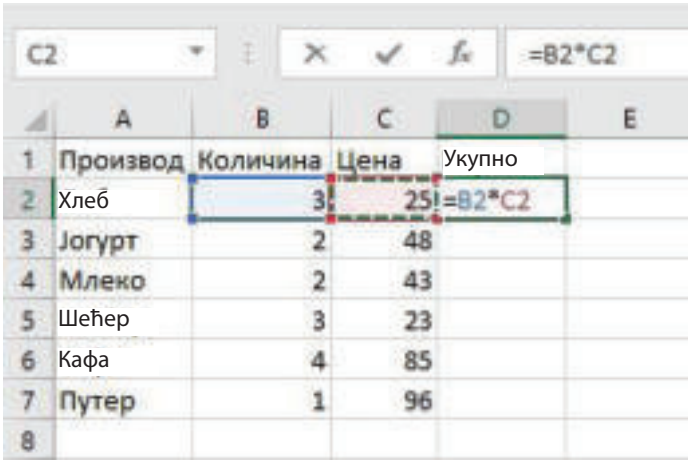
Могу се користити и логички оператори као што су <, >, =, <=, >=, < >, који враћају вредност тачно (True) и нетачно (False). При креирању формула могу се користите и заграде.

Формуле и функције које се уносе у ћелију у табели увек почињу са знаком „=“.

На пример:

Да бисмо применом формула израчунали колико треба да платимо на каси у супермаркету, после уноса података вршимо прорачун:

1. кликнемо на ћелију D2 и унесемо знак „=“;
2. уносимо адресу ћелије у којој је уписана количина;
3. додамо знак за множење и адресу ћелије у којој је уписана цена производа;
4. на тастатури кликнемо Enter.



Слика 1: Прорачун применом формуле

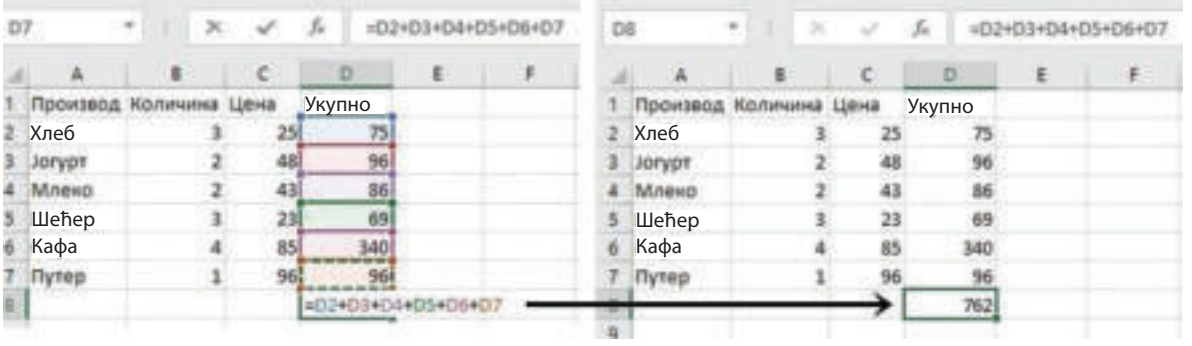
Коментар!

Осим уписивањем, адреса ћелије у формули се може додати и кликом на ћелију.

На основу задатка примећујемо да при уносу формуле у активној ћелији, кликом тастера **Enter** са тастатуре у активној ћелији приказује се резултат, а у **траци за формуле (Formula Bar)** приказује се формула коју смо употребили да дођемо до резултата. Прорачун за све остале артикле можемо извршити аутоматским попуњавањем ћелија.

Израчунајмо и укупну суму коју треба платити!

1. Кликнемо у D8 и уписујемо знак „=“.
2. Уписујемо адресе ћелија од D2:D7 са знаком за сабирање између.
3. Кликнемо Enter са тастатуре и добијемо укупни резултат.



Слика 2: Прорачун укупне суме применом формуле

Коментар!

Приликом креирања формула, нећемо радити са вредностима у ћелијама већ са њиховим адресама да би се при промени вредности података аутоматски променио и резултат.

Задатак

Попуните следећу табелу:

	A	B	C	D
1	X	Y	z=x*2-2*y	s=(x+y)/4
2	0	3		
3	1	6		
4	2	9		
5	3	12		
6	4	15		
7	5	18		
8	6	21		
9	7	24		
10	8	27		
11	9	30		

1.6.2 Функције

Функције су припремљене формуле које се могу користити за извођење неких прорачуна са подацима у табели. Функције имају своје име и груписане су по категоријама: математичке, статистичке, логичке и слично. Функције раде са адресама ћелија, рангом ћелија или несуседним ћелијама.

Испред функције пише се знак једнако (=), следи име функције и на крају аргументи функције. Аргументи функције су у ствари ознаке ћелија које ће учествовати у прорачуну.

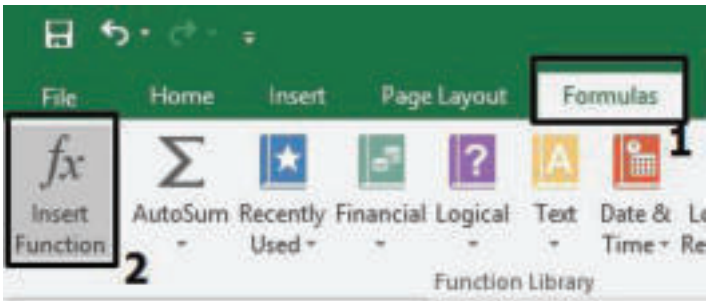
Слика 3: Елементи функције

Најчешће коришћене функције су:

SUM – израчунава збир бројних вредности у групи ћелија.
MIN – даје најмању нумеричку вредност групе ћелија.
MAX – даје највећу нумеричку вредност групе ћелија.
AVERAGE – израчунава средњу вредност нумеричких вредности у групи ћелија.

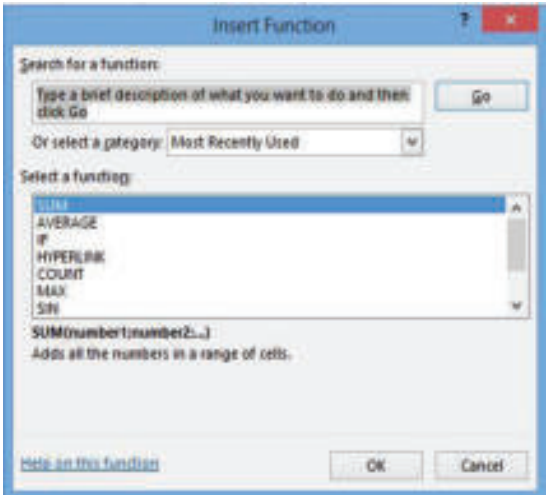
COUNT – даје број нумеричких записа у изабраним ћелијама.
COUNTIF – даје број записа у изабраним ћелијама ако је испуњен одређени услов.

Поступак уношења функције почиње селектовањем ћелије у којој ће се прорачун извршити, а после тога менијем **Formulas** и командом **Insert Function** уносимо функцију.



Слика 4: Поступак за уношење функција

Притом, појављује се следећа слика на основу које бирамо категорију и функцију:



Слика 5: Избор функција

Да би се извршио прорачун помоћу изабране функције, додајемо и аргументе са следећег прозора:

Слика 6: Избор аргумената функције

Кликом на дугме **OK** на прозору, добићемо резултат прорачуна.

Међутим, понекад се у ћелији у којој би требало да се прикаже резултат из формуле или функције, приказују симболи за грешку: #####, #DIV/0!, #NAME?, #REF!, #VALUE!. Они се још зову и поруке за грешку у формули или функцији (**Error message**).

Поруке за грешке	Значење
#####	Колона је тесна да би приказала резултат
#DIV/0!	Дељење нулом је немогуће
#NAME?	У формули је погрешно уписана адреса ћелије
#REF!	У формули постоји адреса избрисане ћелије
#VALUE!	У формули једна од ћелија садржи текст

Табела 2: Поруке за грешке у формули или функцији

Задатак

Креирајте табелу у програму за прорачунске табеле као што је приказано на следећој слици:

	A	B	C	D	E	F	G
1		Број прочитаних књига по месецима					
2	Име и презиме ученика	Јануар	Фебруар	Март	Април	Мај	Укупно прочитаних књига
3	Ана Михајлоска	4	3	2	1	0	
4	Сара Етеми	5	2	2	1	1	
5	Арта Тахири	4	3	1	2	1	
6	Јован Спироски	3	1	2	1	1	
7	Дритон Емини	6	1	3	2	0	
8							
9	Укупно прочитаних књига по месецима						
10	Највише прочитаних књига						
11	Најмање прочитаних књига						
12	Средња вредност прочитаних књига						
13	Укупно подаци						
14	Колико ученика НИЈЕ прочитало ниједну књигу?						

Извршити тражене прорачуне користећи функције **MIN, MAX, AVERAGE, COUNT** и **COUNTIF**.

Запамти!

Формуле у програмима за прорачунске табеле су математички изрази креирани од података и различитих математичких операција. У формулама се не ради са подацима у ћелијама, већ са њиховим адресама. Функције су припремљене формуле које се могу користити за извођење одређених прорачуна са подацима у табели. Функције имају своје име и груписане су по категоријама: математичке, логичке, статистичке и слично.

Питања

1. Шта су формуле, а шта функције?
2. Опиши разлику између формула и функција!
3. Које су најчешће коришћене функције?
4. Истражите, набројте и наведите друге Excel функције поред оних које смо учили у лекцији!
5. Напишите синтаксу функције која израчунава збир неколико вредности из узастопних ћелија!

1.7 Сортирање података

Ако радни лист има пуно садржаја, то може бити тешко за брзо проналажење информација. Филтери се могу користити само да би се сузили подаци на радном листу, омогућавајући да се виде информације које су потребне кориснику.

Подаци у табели се могу организовати и да се притом креирају извештаји према потребама корисника. Ово се ради са следећим операцијама:

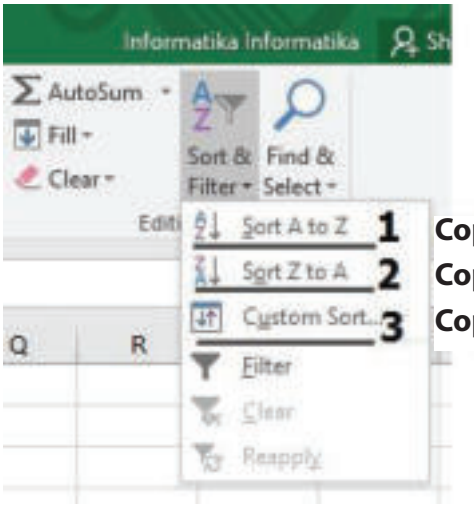
- сортирање података у задатом редоследу;
- филтрирање података према одређеним критеријумима;
- стварање збирних и подзбирних података.

У овој наставној јединици усредсредићемо се на поступак сортирања, што је посебна алатка која олакшава приступ траженим информацијама издвојеним од осталих.

Сортирање значи промену редоследа редова у табели да би подаци у њима били поређани према одређеном редоследу. У програму Excel могуће су следеће методе сортирања:

- сортирање по опадајућем или растућем реду;
- сортирање по абецеди, нумерички, хронолошки или редоследно кориснички дефинисано;
- сортирање према дефинисаном условном форматирању.

Ако сортирање треба извршити према једном критеријуму, тј. према једној колони, онда је довољно да се кликне само једно поље у колони у коме корисник жели да сортира, а затим из менија Home се бира.



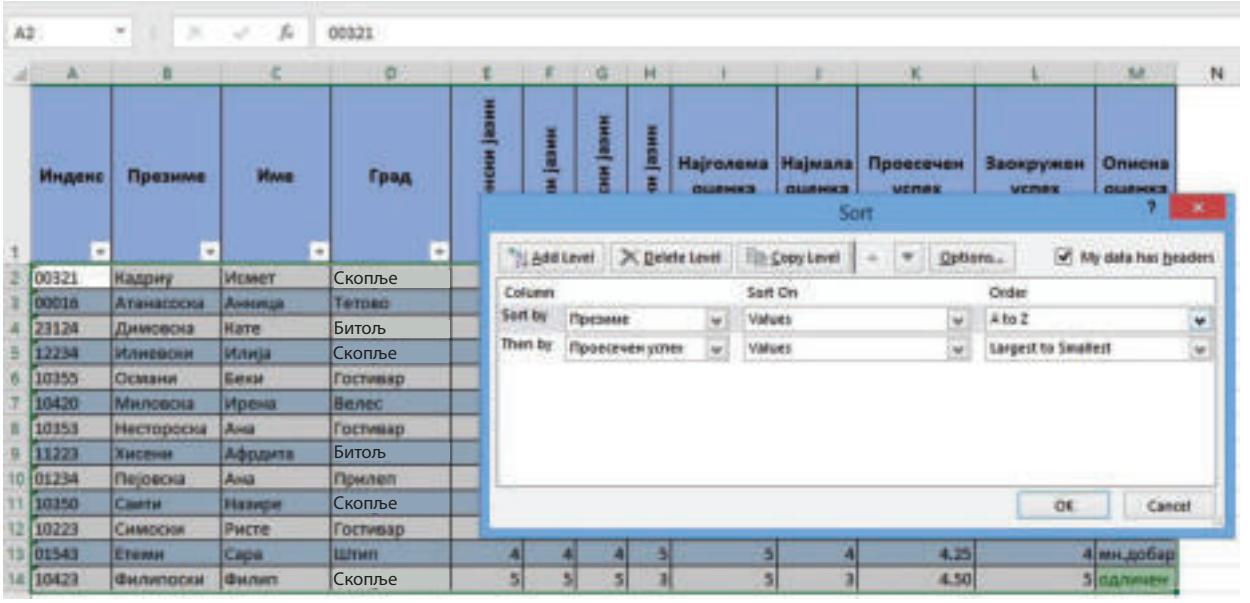
Сортирање по абецедном реду
Сортирање по реду обрнутом од абецедног
Сортирање по одређеном услову

Слика 1: Сортирање

Ако корисник одабере опцију **Custom Sort**, онда се у пољу **Column (Sort by)** бира критеријум по којем ће се сортирање извршити, у пољу Sort on бира врсту података који ће се сортирати, а у пољу 3 бира се редослед сортирања. На пример, у следећој табели сортирање је извршено према следећим критеријумима:

1. Према презимену ученика по абецедном реду;

2. Према просечном успеху као другом критеријуму, по реду од највишег просека према најмањем.



Слика 2: Сортирање података помоћу опције Custom Sort

Задатак

У програму за прорачунске табеле креирати следећу табелу „Подаци из библиотеке“:

Подаци из библиотеке

Ред бр	Лектира	Име аутора	Презиме аутора	Разред	Изнајмљена
1	Мајка	Јован	Јовановић - Змај	II	84
2	Зоки Поки	Оливера	Николовска	II	68
3	Поезија	Рифат	Кукај	III	35
4	Шекерна приказна	Славко	Јанески	III	44
5	Сказна за детето Вилен	Глигор	Прличев	IV	54
6	Училиште	Драган	Лукић	III	30
7	Орхан	Нецати	Зекирија	IV	57
8	Летни цвекиња	Наим	Фрашери	VI	67
9	Силјан Штркот	Марко	Цепенков	VII	46
10	Принцезата Арѓиро	Исмаил	Кадаре	VII	44
11	Бегалка	Видое	Подгорец	VIII	70

Извршити сортирање према категорији **„Лектира“** по абецедном реду, а затим колико је пута **„Изнајмљена“** почевши од највеће до најмање вредности!

	A	B	C	D	E	F
1	Подаци из библиотеке					
2	Ред бр	Лектира	Име аутора	Презиме аутора	Разред	Изнајмљена
3	1	Мајка	Јован	Јовановиќ - Змај	II	84
4	2	Зоки Поки	Оливера	Николовска	II	68
5	3	Поезија	Рифат	Кукај	III	35
6	4	Шеќерна приказна	Славко	Јанески	III	44
7	5	Сказна за детето Вилен	Глигор	Прличев	IV	54
8	6	Училиште	Драган	Лукиќ	III	30
9	7	Орхан	Нецати	Зекирија	IV	57
10	8	Летни цвеќиња	Наим	Фрашери	VI	67
11	9	Силјан Штркот	Марко	Цепенков	VII	46
12	10	Принцезата Арѓиро	Исмаил	Кадаре	VII	44
13	11	Бегалка	Видое	Подгорец	VIII	70

Запамти!

Сортирање је процес распоређивања података према правилима. Распоређивање може бити по абецедном реду, према реду обрнутом од абецедног или према условима дефинисаним опцијом **Custom Sort**.

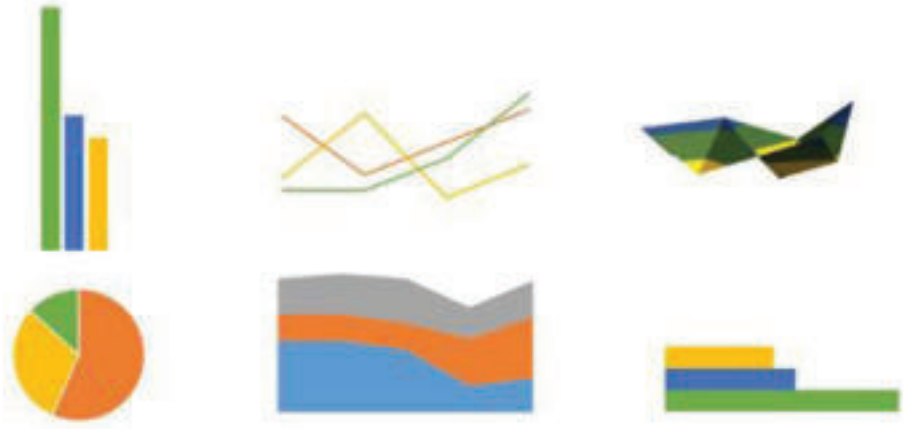
Питања

1. Шта је сортирање?
2. Какав је поступак сортирања података по абецедном реду?
3. Како можемо сортирати податке под датим условима?

1.8 Израда графикона

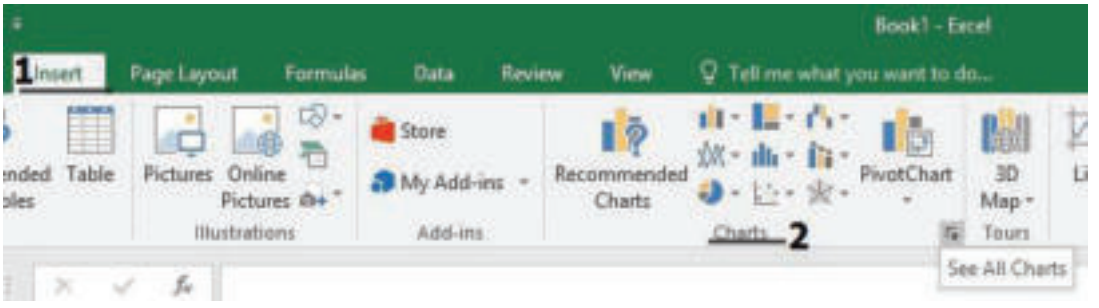
Када се једном креира табела, форматирање се врши на основу поступака које смо навели у претходним наставним јединицама, а често је потребно да подаци буду представљени сликовито, односно путем графикона. Уз сликовити приказ података, кориснику се пружа брз и јасан увид у вредности података, њихово поређење и анализу.

Стога, **графикони су сликовито приказивање података из табеле**. Заправо, графикон претвара податке у слику. Графикони могу бити: стубасти, у облику линија, пите, круга и слично и користе се за различите циљеве, као што је, на пример: за поређење по категоријама, за поређење вредности у процентима и слично, као што је приказано на следећој слици:



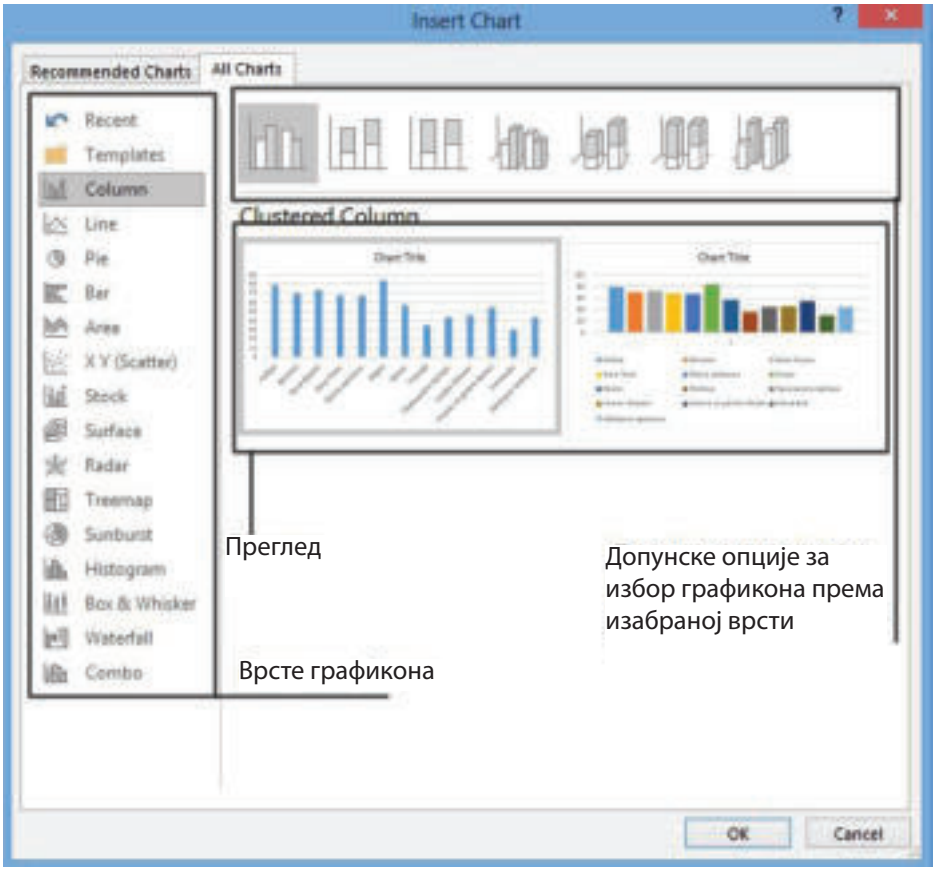
Слика 1: Врсте графикона

Поступак за израду графикона у програму за прорачунске таблице је једноставан ако су подаци правилно написани и организовани у радном листу. Затим, бира се ранг ћелија са подацима које треба приказати графиконом. Након избора података који ће бити графички приказани, можемо применити команду за креирање графикона у програму **Excel 2016** са менија **Insert** и одељка **Charts**:



Слика 2: Поступак за уметање графикона

На слици примећујемо да приликом уметања графикона одмах можемо бирати категорију, тј. врсту графикона путем Quick Launcher-a, смештеног на левој страни рубрике Chart, при чему се приказује следећи прозор:



Слика 3: Поступак за избор врста графикона

Кликом на дугме **OK**, графикон ће се појавити на радном листу у коме се налази извор података, односно изворна табела. Тада одлучујемо да ли је графикон прегледан да би остао у истом радном листу или га треба пренети на нови радни лист. Такође, док је графикон селектован, појављују се менији **Design** и **Format** који омогућавају форматирање и уређивање изгледа графикона. На пример, из менија **Design** можемо применити следеће:

- додавање елемената на графикону,
- распоред елемената на графикону,
- избор боје,
- стил графикона,
- избор података,
- промена типа графикона,
- преношење графикона на нови радни лист / у исти радни лист са табелом.

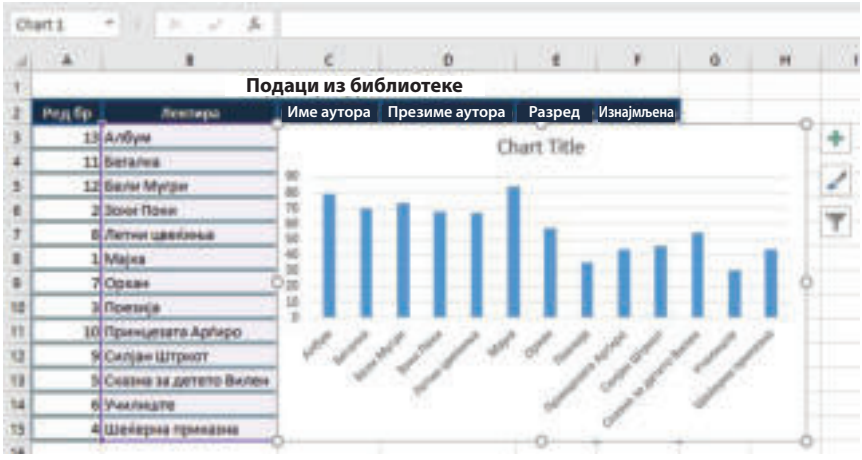
Помоћу менија **Format** можемо додати облике деловима графикона, бојама линије, бојењем елемената, стиловима слова, димензијама елемената и слично.

Коментар!

Приликом креирања формула, нећемо радити са вредностима у ћелијама, већ са њиховим адресама, у циљу да се при промени вредности податка аутоматски промени и резултат.

Задатак

Отворимо табелу „Подаци библиотеке“, која је сачувана у директоријуму на радној површини. Направимо графикон који ће илустровати која је књига највише читана, користећи графикон **Columns**, као што је приказано у наставку на сликама



Графикон треба да буде на новом радном листу и да се форматира помоћу алатки из менија **Design** и **Format**.

Запамти!

Графикони су графички прикази података из табеле. Пружају брз и јасан увид у вредности података, њихово поређење и анализу. Графикони могу бити: стубасти, у облику линија, пите, круга итд.

Питања

1. Шта је графикон?
2. Опиши поступак израде графикона?
3. Које типове графикона пружа програм Excel 2016?
4. Који се алати могу користити из менија Design и Format за уређивање и форматирање графикона?

ПОНОВИМО! УВЕЖБАЈМО!

Задатак 1

Отворите радни документ у MS Excel 2016. Први радни лист именујте „кућа“ и у њему применом различитих начина селектовања обојте ћелије тако да добијете кућу. Документ сачувајте под својим презименом и разредом!

Задатак 2

У програму прорачунских табела у радном листу 1 (Sheet 1) креирајте телефонски именик ученика у разреду. Табела треба да садржи следеће податке: редни број, име, презиме и телефон. Осим броја телефона вашег друга, додајте и датум његовог рођења.

Задатак 3

Направите списак књига у својој библиотеци. Притом унесите следеће податке: аутор, наслов књиге и година издања.

Задатак 4

Направите табелу у којој ћете унети податке: месец, приходе, расходе. Након уноса података додајте нову колону: разлика (разлика између прихода и расхода). Додајте нови ред изнад насловних ћелија. Ћелије новог реда обојте помоћу опције Fill color. Редовима задајте димензије двадесет (20), а колонама димензије двадесет и пет (25). Сачувајте документ!

Задатак 5

У радном документу програма за прорачунске табеле направите следећу табелу:

E5 | 0

	A	B	C	D	E
1	Успех по предметима				
2	Предмет	Оцењени	Неоцењени	Позитивне оцене	Негативне оцене
3	Математика	12	1	10	2
4	Информатика	11	2	11	0
5	Енглески језик	10	3	10	0
6	Матерњи језик				

Форматирајте табелу помоћу алата за форматирање: фонт, величина, стилови, редослед, оријентација и примена стилова ћелија или припремљени формати табела. Сачувајте документ!

Задатак 6

У програму за прорачунске табеле креирати следећу табелу и применом формула за израчунавање обима и површине извршити прорачун:

	A	B	C	D	E
1		***Геометријске фигуре***			
2		**Страна 1**	**Страна 2**	**Обим**	**Површина**
3	Правоугаоник	6	4		
4	Коцка	3	3		
5	Квадрат	5	5		
6					

Задатак 7

Креирајте табелу у програму за прорачунске табеле у којој су приказане просечне температуре у земљи, по месецима. Помоћу функција одредите који је месец најтоплији, а који најхладнији у години!

	A	B
1	Јануар	5.6
2	Фебруар	6.2
3	Март	8.8
4	Април	12
5	Мај	17
6	Јун	21
7	Јул	23
8	Август	23
9	Септембар	18
10	Октобар	15
11	Новембар	10
12	Децембар	6.9
13		

Задатак 8

Направите табелу са подацима својих пријатеља: име, презиме, узраст и сортирајте их редом обрнутим од абецедног.

Задатак 9

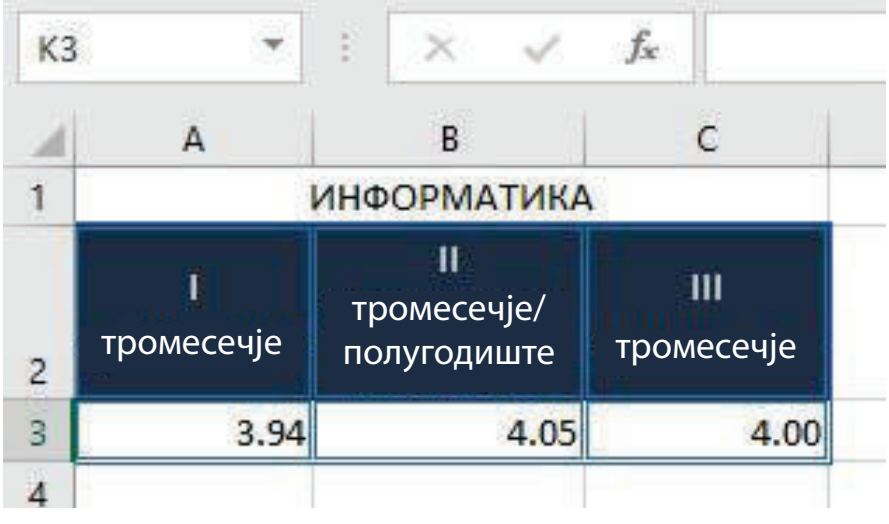
Креирајте и форматирајте табелу у програму за прорачунске табеле. Табела треба да садржи податке ваших другова: име, презиме, успех. Сортирајте их по презимену, абецедним редом, а затим по успеху, почев од најбољег до најслабијег.

Задатак 10

Направите графикон у програму за прорачунске табеле у коме ће се упоређивати успех одељења од VI до IX разреда. На тај начин примените графикон питу креиран у новом радном листу.

Задатак 11

Направите табелу која приказује просечан успех на крају сваког тромесечја, као што је приказано на слици. Затим, податке графички прикажите помоћу графикона Columns.



Задатак 12

Истражите трошкове воде по квадратном метру у вашем граду и израчунајте колики рачун треба да платите!

Пројектни задатак

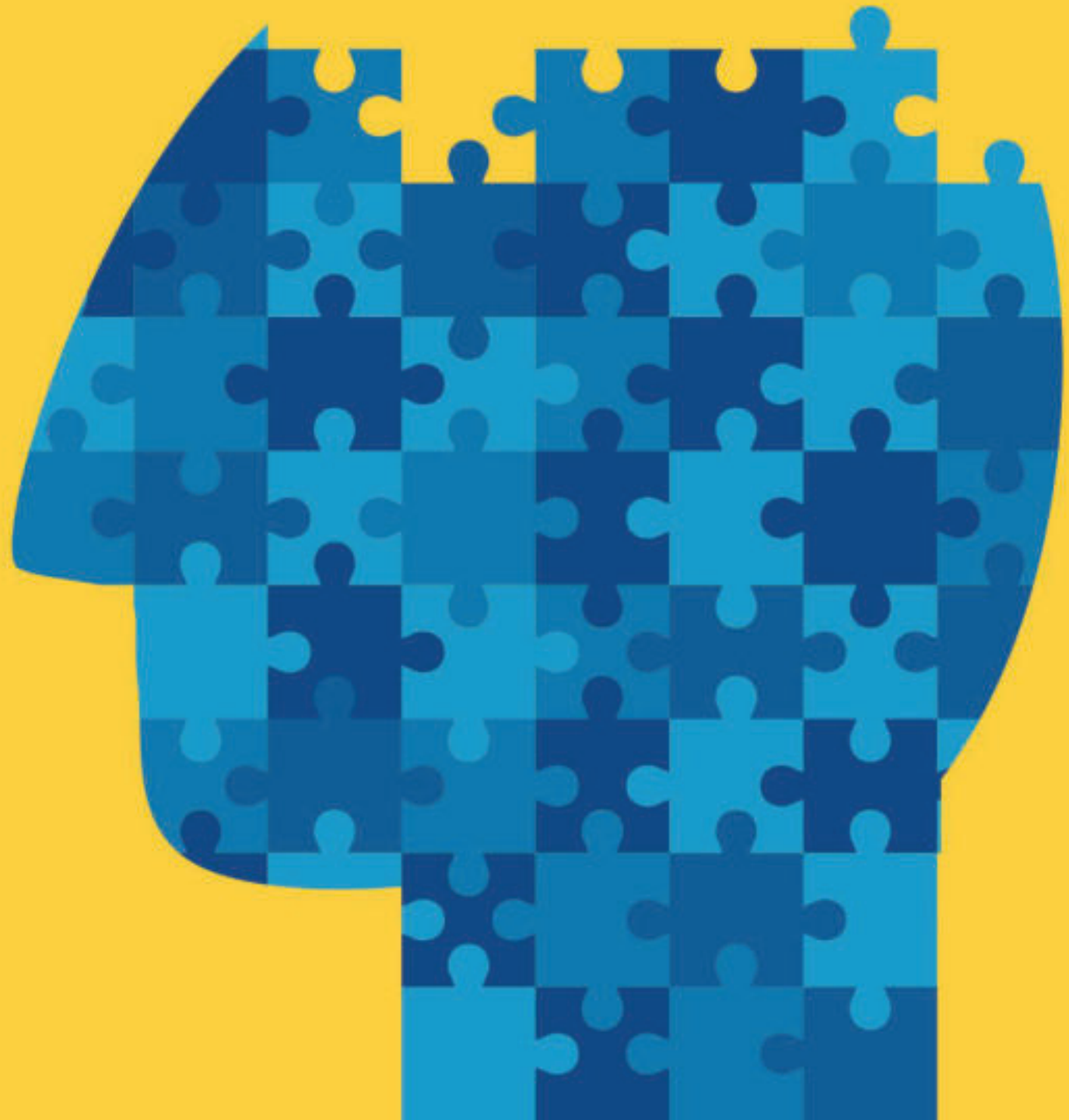
У програму за прорачунске табеле, у радном листу 1, креирајте следећу табелу:

Редни број	Презиме	Име	Град	Македонски језик	Енглески језик	Немачки језик	Албански језик
1	Кадриу	Исмет	Тетово	4	5	5	3
2	Атанасоска	Анкица	Скопје	5	5	5	5
3	Димовска	Кате	Прилеп	4	4	2	3
4	Илиевски	Илија	Битола	3	3	2	2
5	Османи	Бакија	Скопје	3	3	5	4
6	Миловска	Ирена	Штип	4	4	4	5
7	Несторовска	Ана	Гостивар	5	4	4	2
8	Хисени	Афродита	Велес	2	2	3	3
9	Перовски	Ана	Битола	2	3	4	5
10	Сапти	Назире	Скопје	5	3	4	2
11	Симоски	Ристе	Гостивар	5	2	3	4
12	Етеми	Сара	Скопје	5	5	5	3
13	Филиповски	Филип	Гостивар	3	3	5	5
14	Спироски	Ристе	Велаздово	3	4	4	4

- Наслови ћелија са језицима треба да имају вертикални правац текста и примените опцију Wrap Text.
- Поред колоне „Албански језик“ додајте колону „Највиша оцена“ и помоћу функције израчунајте највишу оцену.
- Додајте колону „Најмања оцена“ поред колоне „Највиша оцена“ и помоћу функције израчунајте најмању оцену.
- На крају табеле израчунајте просечни успех за сваког ученика/студента понаособ.
- Спој ћелије од A15 до J15 и напиши просек разреда.
- Ученике са просечном оценом „добар“ обој црвеном, а „одличне“ ученике зеленом бојом користећи условно форматирање.
- Израчунајте просек оцена у разреду у ћелији K15.
- Испод колона у којима су предмети треба избројати колико петица, четворки, тројки и двојки има за сваки предмет.
- Форматирајте табелу користећи: фонт, боју слова, стилове слова, боју позадине ћелије, врсту оквира и боју оквира да би се приказала прегледност. Може се користити и припремљен формат/дизајн табеле.
- Поређајте податке по презимену почевши од A, а други услов је према успеху почев од најбољих.
- Направите графикон на посебном радном листу за приказ успеха ученика. Форматирајте графикон по жељи.
- Сачувајте документ!

Упознавање информатичких концепата помоћу логичког решавања такмичарских задатака

Упознавање информатичких концепата помоћу логичког решавања такмичарских задатака
Анализа и решавање логичких такмичарских задатака
Повезаност задатка са концептима из рачунарства – информатички концепти
Структуре података
Бинарни бројеви
Криптографија
ПОНОВИМО! УВЕЖБАЈМО!



2 Упознавање информатичких концепата помоћу логичког решавања такмичарских задатака

Шта ћемо научити?

Научићемо да креирамо табеле са различитим типовима података и низовима прорачуна применом формула и функција, уређеним помоћу алатки за форматирање и графички представљеним графиконима.

У проучавању рачунарства, између осталог, једна од најузбудљивијих и најзанимљивијих ствари јесте учење да се размишља на нов начин и да се решавају проблемски задаци и ситуације под називом **рачунарско размишљање**. Овакав начин размишљања је пресудан за 21. век као основа за успех у многим ситуацијама у свакодневном животу, током образовања, а тако и за професионални напредак и усавршавање.

Пре него што се рачунари употребе за решавање проблема или проблемске ситуације, неопходно је разумети суштину проблема. Стога се може рећи да рачунарско размишљање представља скуп вишеструких вештина, као што су: креативност, способност објашњавања и доказивање, тимски рад и сарадња.

Главни елементи рачунарског размишљања су:

- логичко размишљање,
- алгоритамско размишљање,
- ефикасно решавање,
- научно размишљање,
- иновативно размишљање.

У овој теми ћемо ставити посебан акценат на **логичко размишљање**, као посебну и основну вештину рачунарског размишљања. Логичко размишљање је најчешће повезано са рачунарским концептима, али у новије време је синоним за интеграцију рачунарске идеје и у другим дисциплинама.

Кад кажемо да је нешто логично, заправо мислимо да то нешто има смисла. Способност појединца да размишља на начин утемељен на чињеницама и доказима је познат као логичка способност размишљања. Према томе, **логичко размишљање се може дефинисати као процес у коме се користи објективно расуђивање да би се дошло до одређеног закључка.**

Покушајте!

Истражите појам логика! Дефинишите и покушајте да истражите разлике у логичком и критичком размишљању? По чему су ова размишљања слична, а по чему се разликују?

Проблемски задаци или ситуације које се решавају помоћу логичког размишљања имају структуру, везе између чињеница и имају смисла, односно логичку узрочно-последичну везу. Овај начин размишљања укључује дубинску **анализу**, као на пример: мерење свих доступних опција, коришћење чињеница и бројки и доношење важних одлука које се заснивају на предностима и недостацима, а то значи да се у таквом процесу не узимају у обзир елементи као што су осећања или емоције. Најчешћи пример логичког мишљења је СУДОКУ. Задате су информације о вредностима садржаним у неколико ћелија. Да би се решила слагалица, треба закључити које су вредности у свим осталим ћелијама, поштујући правило приликом решавања овог проблемског задатка. Ако направимо и најмању грешку, тада ћемо бити далеко од решења.

Иако је овај процес неупадљив, сви ми се свакодневно суочавамо са њим у различитим ситуацијама и условима које превазилазимо захваљујући сопственим вештинама расуђивања. **На пример**: приликом израчунавања цена у супермаркету да бисмо проверили можемо ли добити све што нам треба по нижој цени или док покушавамо да уклопимо све наше обавеза у једном дану, наша машина за размишљање се непрестано окреће да би нашла одговарајуће решење. Сходно томе, у логичком резоновању, најважније ствари које морамо испоштовати су следеће:

- **Не гледајте на ствари само из своје перспективе.** То може бити субјективно и не води нас ка циљу, односно до коначног решења. На пример: Стефан и Алмир ручају у ресторану. Стефанов тањир има непријатан и лош мирис, док Алмир ужива у ручку. Пошто се Стефану није допао мирис хране, брзо је закључио да оброк није хранљив, није здрав и није добро припремљен. Ово није логичан начин да се дође до закључка, јер Стефан нема доказа да храна није здрава и лоше је припремљена. Да бисте дошли до логичког закључка, морате да искључите сопствена субјективна мишљења и да се усредсредите на доказане информације, као што је, на пример, које састојци се користе за припрему овог оброка, да сазнате поступак припреме, а не да размишљате на основу претпоставки.

- **Размислите пре него што започнете – креирајте стратегију**. Стратегија игра велику улогу у процесу размишљања. Започните истраживањем, постављајући питања која ће помоћи у тумачењу чињеница. Активно тражите детаље и научите како они функционирају као засебни делови, а како као група, пре него што дођете до опште слике.

● **Обратите пажњу на значење речи**. Мале језичке варијације чине велику разлику у значењу постављеног задатка или проблемске ситуације. Утврђивање разлике између изјава, дефинитивно одбацују недоследности и нејасноће у логичком размишљању. На пример: реч **„неопходно“** је различита од **„довољно“**. Неопходно значи да услов или поступак мора бити испуњен, за разлику од речи довољно, што значи учинити минималан напор за постизање решења.

На крају, да закључимо да је логичко размишљање као скуп вештина и техника за решавање проблемских задатака и ситуација, корак пре програмирања као процеса стварања софтвера/програма за различите намене.

Кључни појмови!

рачунарско размишљање, логичко размишљање, анализа, програмирање, структура података, бинарни бројеви, кодирање, криптографија, бесплатни софтвер.

Запамти!

Рачунарско размишљање је нови начин решавања проблемских задатака и ситуација, којим се помоћу логичког мишљења постиже објективно решење. Логичко размишљање се заснива на чињеницама и доказима до којих се долази на различит начин применом различитих техника и метода. Овакав начин размишљања никада се не заснива на претпоставкама, жељама, осећањима и емоционалним стањима.

2.1 Анализа и решавање логичких такмичарских задатака

Анализа проблемског или логичког задатка је први и основни корак који морамо објективно предузети да бисмо започели пут до коначног решења. Помоћу **анализе** разлажемо задатак или проблемску ситуацију, откривамо чињенице или елементе целине, откривамо везе како бисмо дошли до резултата.

Анализа задатака укључује три корака која се могу сажети на следећи начин:

Слика 1: Кораци у анализи проблемског задатка

Ако је анализа задатака пажљиво организована и следи одговарајуће кораке, може се применити на решавање многих задатака и довести до коначног решења, управо због укључивања детаљне пажљиве и интегрисане анализе.

Покушајте!

Покушајте да истражите који је обрнути или супротни процес анализе ако знамо да је анализа процес разградње задатка на посебне елементе или делове!

Анализирајмо неколико проблемских задатака применом неких техника које смо изнели, користећи притом стечено знање за логичко мишљење.

Задатак

Јован, Астрет и Сами су вршњаци и похађају исту школу. Сва тројица су добри атлетичари који су освојили више државних награда и међународних такмичења. Који је од њих најспорији, ако су све наведене изјаве тачне?

1. Јован није најспорији.
2. Астрет је најбржи.
3. Сами није најбржи.

Најспорији		Најбржи
		
Јован	Астрет	Сами

Један од исказа каже да је **Астрет** најбржи, а самим тим и лик **Астрета** повезујемо са оквиром означеним као „најбржи“. Сада два поља остају непопуњена. Такође се наводи да **Јован** „није најспорији“ и зато га повезујемо са другим квадратом. Остаје празно само прво поље означено као „најспорији“ и тамо повезујемо **Самија**. Стога је тачан одговор на питање ко је најспорији – **Сами**.

У решавању постављеног задатка користили смо технику која се назива **апстракција**, јер смо издвојили најважније одлике задатка. Идентификоване карактеристике водиле су нас где треба да започнемо испитивање проблемског задатка, којим путем да кренемо и како да дођемо до могућег решења. Иста ствар се дешава и решавањем проблемске ситуације у **информатици**. Увек се ради са информацијама које су важне за решавање проблемске ситуације, а непотребне информације се одбацују или игноришу да би се решио проблем.

Задатак

Слично претходном задатку, покушајте да дођете сами до тачног решења следећег задатка!

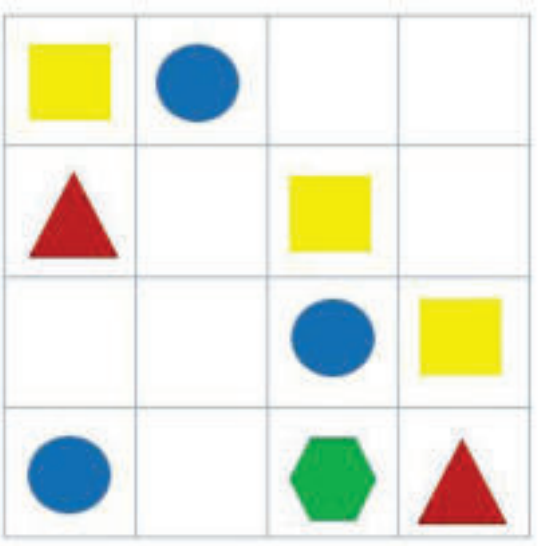
Ако су сви искази тачни, где се налази богатство?

1	2	3
		
Богатство није у 2.	Богатство је у 1 или 3.	Богатство није ту.

Питања и задаци

Судоку је једна од најпопуларнијих игара које укључују логичко размишљање и процес уклањања чињеница које нису од важности за долажење до резултата.

Решимо једноставан Судоку задатак!

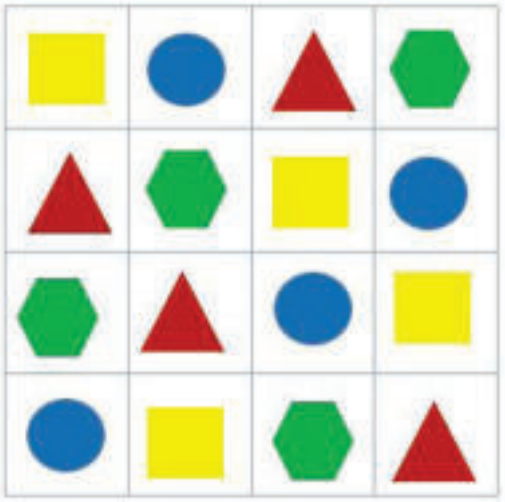


У табели су геометријске фигуре распоређене у нека поља, а нека поља су празна. Празна поља треба попунити следећим фигурама:

Свака геометријска фигура треба да има једнак број фигура, распоређених тако да хоризонтално и вертикално увек има само једну фигуру исте врсте. Имајте на уму да се круг може додати у табелу само једном и зато треба одредити одговарајуће место, тј. у хоризонталном и вертикалном смеру треба да буде само једна таква фигура. Нађимо право место. У првом, трећем и четвртом реду (водоравно), већ постоји фигура круга, тако да нам као избор остаје други ред.

Али у ком пољу у другом реду? У којој колони?

Колоне један, два и три већ садрже фигуру круга и оне су искључене из опције. Сходно томе, први, трећи и четврти ред, као и прву, другу и трећу колону изостављамо као могућност. Остаје једина могућност да додамо фигуру круга у други ред и четврту колону и то је тачан одговор. Ако попунимо сва поља, као што је у примеру са кругом, онда долазимо до следећег резултата:



Покушајте!

Покушајте да решите Судоку применом правила за попуњавање поља бројевима! Постоји много електронских примера, али у штампаним медијима се могу наћи у штампаним верзијама.

Процес **елиминације**, када се решавају логички такмичарски задаци, омогућава одбацивање чињеница које нису значајне и које нас не воде до коначног решења проблемске ситуације, као у случају игре Судоку. У свакодневном животу, елиминација се најчешће користи у решавању тестова у којима је понуђено више могућих одговора, при чему се елиминишу нетачни одговори да би се смањио број могућих опција за тачан одговор.

Стога смо у анализи логичких задатака користили технике логичког размишљања да бисмо дошли до коначног решења. Овде смо користили технике апстракције, елиминације и посматрања задатих правила. Али, приликом решавања проблемских ситуација и задатака, као део информатичких или рачунарских процеса, користе се друге технике које произилазе из потребе и природе задатка, са једним циљем, да се на најоптималнији начин постигне коначно решење.

Запамти!

Анализа проблемских ситуација и задатака треба да буде објективна а не пристрасна, односно да не укључује елементе субјективности. Поступак анализе укључује три корака: разлагање задатка на саставне елементе, одређивање везе између елемената и реструктурирање према добијеним резултатима. Најчешће технике када се анализира задатак или проблемска ситуација су: апстракција, елиминација, поштовање правила, проналажење сличности итд.

2.2 Повезаност задатка са концептима из рачунарства – информатички концепти

Логичко размишљање и решавање логичких такмичарских задатака су вештине које се могу користити у различитим научним дисциплинама, али су најчешће повезане са рачунарима и **информатичким концептима**. Стога, није случајно што се каже да је логично размишљање корак пре **програмирања** као процеса креирања апликативних решења.

Сходно томе, **програмирање је дефинисано као процес писања програма помоћу програмских језика**. Људи који креирају програме зову се **програмери**.

На почетку стварања програма, програмери изврше бројне кораке, од којих нам сваки на свој начин отвара пут према решењу, односно води до стварања програма. Током овог процеса програмери се сусрећу са информатичким концептима. Због тога ћемо се у овој наставној јединици упознати са некима од њих.

2.2.1 Структуре података

Структуре података су посебан облик или начин прикупљања, организације, обраде и чувања података. Најчешће коришћене структуре су: низови, повезане листе, стекови/гомиле, редови, приоритетни редови, бинарно стабло итд.

Стога се структуре података могу користити код стварања:

- листе корака за постизање одређеног решења,
- низа упутстава како доћи до неког одредишта,
- упутства за попуњавање обрасца или шеме итд.

Задатак

Направите низ корака користећи наведене ознаке како би се добила следећа слика:


Један квадратић лево


Један квадратић десно


Један квадратић горе


Један квадратић доле


Обој квадратић

Корак 1	2	3	4	5	6	7	8	9	10
Корак 11	12	13	14	15	16	17	18	19	20
Корак 21	22	23	24	25	26	27	28	29	30

Помоћу решеног задатка успели смо да створимо низ упутстава за коначно решење, односно бојење мреже квадрата према приказаној слици.

Задатак

Јесте ли чули за Ханојске куле? Ево примера! Овај пример ће вам показати кораке за извршење задатка који садржи само три диска.

Циљ је предузети кораке за пренос дискова са једног на друго место, а да се притом води рачуна о редоследу.

Корак 1: Пренесите диск број 1 у трећи ред.
Корак 2: Пренесите диск број 2 у други ред.
Корак 3: Пренесите диск број 1 у други ред, одмах изнад диска број 2.
Корак 4: Пренесите диск број 3 у трећи ред.
Корак 5: Пренесите диск број 1 у први ред.
Корак 6: Пренесите диск са бројем 2 у трећи ред, изнад диска са бројем 3.
Корак 7: Пренесите диск број 1 у трећи ред, изнад диска са бројем 2.

Према наведеним корацима, успели смо да све дискове пренесемо са једног места на друго, поштујући правила померања која задаје игра. У седам корака, дискове из првог реда пребацили смо у последњи ред, при чему је редослед дискова остао исти.

Током извођења задатка **„Ханојске куле“**, срели смо се са структуром података стек/гомила, чија је главна карактеристика слагање тако да се прво узму они елементи који су на врху гомиле. Наиме, за стек кажемо да се поштује принцип **„први уђе, последњи излази“**. Наравно, приликом преноса поштују се правила која произилазе из природе самог задатка.

Покушајте!

Уз помоћ наставника биологије покушајте да направите низ у коме ћете приказати животни циклус лептира. Колико периода има животни циклус лептира? Може ли се прескочити неки од наведених циклуса?

2.2.2 Бинарни бројеви

Да се подсетимо!

Који је језик рачунара? Како су подаци представљени помоћу овог језика?

Бинарни систем бројева састоји се од две нумеричке вредности, цифре 0 и 1. Само једна цифра 0 или 1 назива се **бит**. Једним битом можемо да представимо „тачно“ или „нетачно“, „да“ или „не“, „заузето“ или „слободно“ и многе друге ситуације.

Помоћу бинарних бројева подаци у рачунарима се представљају у **низовима нула и јединица**. Претварање слова у реч у бинарним бројевима представља **бинарни код**. Сваки природни број се може представити у низу бинарних бројева, на пример:

Природни бројеви	0	1	2	3	4	5	6	7	8	9	10
Бинарни приказ	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010

Да бисмо претворили природни број у бинарни приказ, користимо базу 2 са степенима: 0, 1, 2, 3, итд. На пример, број 5 у бинарном приказу би изгледао овако: 0101, а овај резултат потиче из:

5	Степен са основом 2	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
		$4 + 1 = 5$			
	Приказ у битовима	0	1	0	1

Попунимо следећу слагалицу тако да у сваком пољу запишете бинарни приказ природног броја:

Хоризонтално:

1) 2
3) 5
4) 34
5) 20

Вертикално:

1) 17
2) 10
3) 10

Помоћу логичког размишљања решимо следећи такмичарски задатак:

Задатак

Ученици VII разреда радили су тест из географије који се састојао од питања са више понуђених одговора. Наставница је у табели приказаној испод бележила тачне одговоре са 1, а нетачне са 0:

	Питање 1	Питање 2	Питање 3	Питање 4	Питање 5	Питање 6
Ана	0	0	1	1	1	0
Сара	1	1	1	1	1	1
Алмир	1	0	0	0	1	1
Сафет	0	0	0	0	1	0
Елона	0	0	1	1	0	1
Амира	1	0	1	0	1	1
Јелена	0	0	1	1	1	1

На основу табеле наставница је желела да сачини извештај о томе које градиво је најслабије усвојено, а које најбоље како би додатно омогућила потпуно усвајање градива.

Одговорите на питања у складу са подацима датим у табели!

- На које питање је одговорила већина ученика?
- На које питање је одговорило најмање ученика?
- Који ученик има најбољи резултат?
- Ко је најслабији ученик?
- Који ученици треба да похађају додатну наставу судећи по резултатима?

2.2.3 Криптографија

Криптографија је научна дисциплина која се бави проучавањем метода слања и примања порука на начин који је разумљив само онима којима је намењен. У ствари, криптографија омогућава употребу тајних кодова који гарантују корисницима:

- заштиту података од неовлашћеног приступа;
- сигурну и заштићену комуникацију;
- поузданост и тајност порука;
- провера порекла података, односно информација итд.

Криптографија се користи када се ради са кредитним картицама, рачунарским лозинкама, при е-трговини и док обављате друге активности, при чему је заштита корисника од посебног значаја.

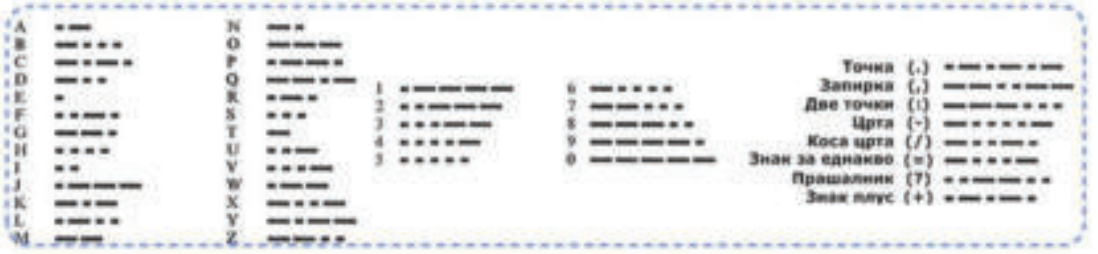
За криптографију се може рећи да је процес који укључује трансформације података или порука:

- шифровање,
- дешифровање.

Шифровање или **енкрипција** је процес трансформације порука у шифре, а **дешифровање** или **декрипција** је обрнути процес који шифровану поруку трансформише у свој изворни облик. Овај процес се назива криптографски процес и он је приказан на следећој слици:

Слика 2: Криптографски процес

Најпознатији пример криптографије је Морзеова азбука која је коришћена у телеграфској комуникацији. Следећа слика приказује свако слово, број и интерпункцијски знак представљен Морзеовом азбуком:



Међутим, најчешће коришћена енкрипција је Цезаров код. Дешифрујмо тј. да декриптујмо поруку користећи Цезаров модел!

Задатак

Јана и Азра су другарице из VII разреда. Потребно је да заједно направе пројектни задатак „Безбедност при коришћењу интернета“ и у ту сврху потребан им је интернет. Приликом форматирања рачунара избрисана је лозинка за приступ интернету. Јанин брат памти лозинку, али жели да се поигра с њима дајући им шифровану лозинку и неколико упутстава:

1. Дешифровање се може извршити Цезаровим моделом;
2. Цезаров модел користи тајни број (кључ);
3. При шифровању се користи неколико целих бројева, као што је кључ за померање.

Лозинка	?	?	?	?	?	?	?	?	?	?	?
Кључ	1	1	3	3	2	1	2	2	3	1	2
Шифра	S	M	T	M	O	Љ	Џ	P	Ж	J	Џ

Хајде да почнемо! Абецеду ћемо написати у табели, а онда ћемо користити кључеве. У нашем случају су дати кључеви за декодирање, односно померање за 1, 2 и 3 места:

Табела 1: Цезаров модел са кључем 1

Слово	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц	Ч	Џ	Ш
Кључ 1	1	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц	Ч	Џ
Шифра	Ш	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц	Ч	Џ	

Табела 2: Цезаров модел са кључем 2

Слово	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц	Ч	Џ	Ш
Кључ 2	1	2	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц	Ч
Шифра	Џ	Ш	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц	Ч	

Табела 3: Цезаров модел са кључем 3

Слово	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц	Ч	Џ	Ш
Кључ 3	1	2	3	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц
Шифра	Ч	Џ	Ш	А	Б	В	Г	Д	Ѓ	Е	Ж	З	Ѕ	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ќ	У	Ф	Х	Ц	

У задатку су нам задати шифра и кључ, пронађимо лозинку!

Кључ	1	1	3	3	2	1	2	2	3	1	2
Шифра	S	M	T	M	O	Љ	Џ	P	Ж	J	Џ
Лозинка	И	Н	Ф	О	Р	М	А	Т	И	К	А

Запамти!

Логичко размишљање и решавање логичких такмичарских задатака су корак испред програмирања. Програмирање је процес писања програма. Људи који креирају програме зову се програмери. Логички задаци и њихово решење су повезани са информатичким концептима, као што су: низови, листе, стекови/гомиле, редови, приоритетни редови, бинарни бројеви, криптографија итд.

ПОНОВИМО! УВЕЖБАЈМО!

Задатак 1

Направите низ упутстава која ће показати пут од улаза у школу до ваше учионице.

Задатак 2

Претворите природне бројеве 12, 18 и 36 у бинарне бројеве!

Задатак 3

Претворите бинарни број 11100111 у природни број!

Задатак 4

Претворите број 236 у бинарни број!

Задатак 5

Направите шифровану поруку користећи Цезаров модел!

Задатак 6

Истражите карактеристике игара Packman и Tetris. Са којим информатичким концептима можемо да их повежемо? Објасни!

Задатак 7

Алма жели да пошаље поруку Ани: „ИЗЛАЗИМО“ Цезаровим кодом. Како гласи порука коју ће послати Алма?

Задатак 8

Стефан је добио поруку од Амира: „ПГОП“ кодирану у Цезаровом коду. Шта је Амир желео да каже Стефану, ако Стефан зна да је кључ број три (3)?

Задатак 9

Јасмин је Томију послао поруку: „УНК ОДДСА“. Међутим, поруку је пресрео Елвин. Елвин зна да се Јасмин и Томи допису̀ју тако што се свако слово писма помера за шест (6) места. Тако је дешифровао поруку. Какву поруку је Елвин примио током дешифровања?

Допунски задаци:

За вежбање логичког размишљања, за решавање логичких такмичарских задатака могу се користити приручници са задацима и објашњењима из Дабара који су објављени на званичној страници **www.talent.mk**.

Интернет адресе за решавање логичких такмичарских задатака и игара које можете користити су:

www.bebras.org
https://www.bbc.com/bitesize/articles/zqnc4wx
http://digit.mile.mk/

Напредно програмирање у визуелном окружењу

3. Увод у програмирање у визуелном окружењу

Да се подсетимо!

Шта су програми? Наведите дефиниције за програмирање! Како се прави програм? Које програмске језике знате?

Логичко размишљање и решавање логичких такмичарских задатака је увод у **програмирање**. Није случајно што се каже да је логично размишљање корак испред **програмирања**.

Слика 1: Лого Scratch-a

Сходно томе, **програмирање се може дефинисати као начин или вештина размишљања како би се дошло до коначног решења проблемског задатка или проблемске ситуације**.

Сходно томе, сврха програмирања је створити скуп упутстава којих се рачунар мора придржавати да би извршио одређене операције или реализацију жељене акције.

Када се пишу и развијају програми, програмер користи групу програма који чине **програмско окружење**. Ти програми су:

- **едитор** – у коме се пише изворни код програма;
- **компајлер** – претвара изворни код у машинску инструкцију;
- **библиотека готових програма** – збирка или комплет већ написани малих и често коришћених програма у програмирању;
- **дебагер** – користи се за отклањање грешака.

Најзанимљивије и најатрактивније је **програмирање у визуелном окружењу**, јер се при решавању проблемских ситуација, креирању корака и упутства користе визуелни објекти, као што су: ликови, позадине, звукови, покрети, ефекти итд., помоћу којих програмирање постаје лако и забавно.

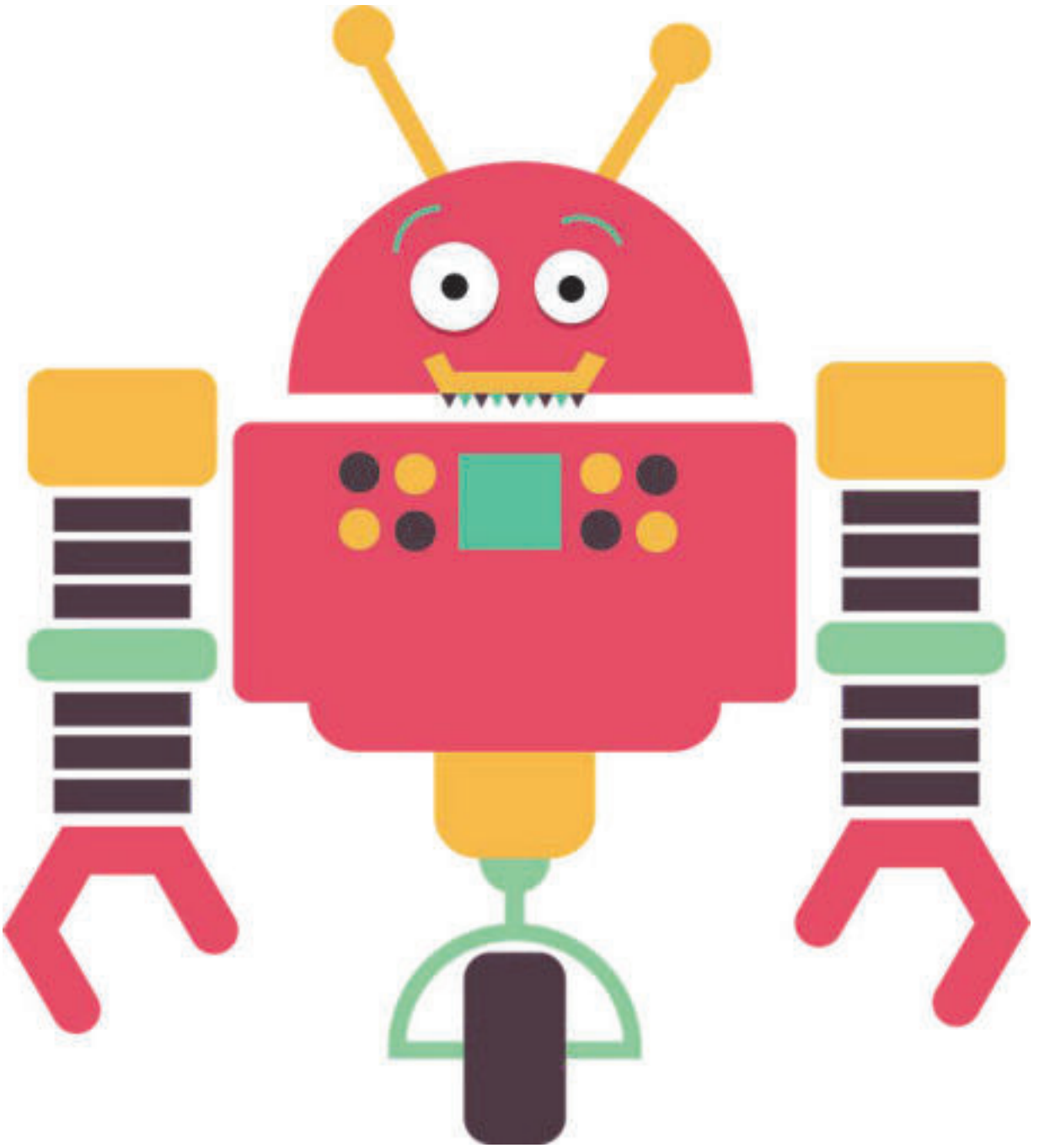
У процесу учења програмирања постоји више апликација које су примери визуелног окружења за креирање програма, од којих се Scratch најчешће користи.

Scratch је визуелни програмски језик којим се креирају занимљиве приче, анимације, игре итд. Програмирање у Scratch-у се заснива на изради пројеката помоћу објеката, такозваних фигура. Његов сет наредби подсећа на слагалицу, пружа широке могућности за стварање сложених програмских решења и савладавање проблемског и алгоритамског начина размишљања.

У овој теми ћемо се упознати са програмским језиком Scratch, његовим радним окружењем и начином стварања пројектних активности попут решења задатака.

Кључни појмови!

графички приказ, координате, константе, променљиве, искази (наредбе), интерактивни програми, објекти, догађаји.



3.1 Графичко програмирање. Увод у програм Scratch

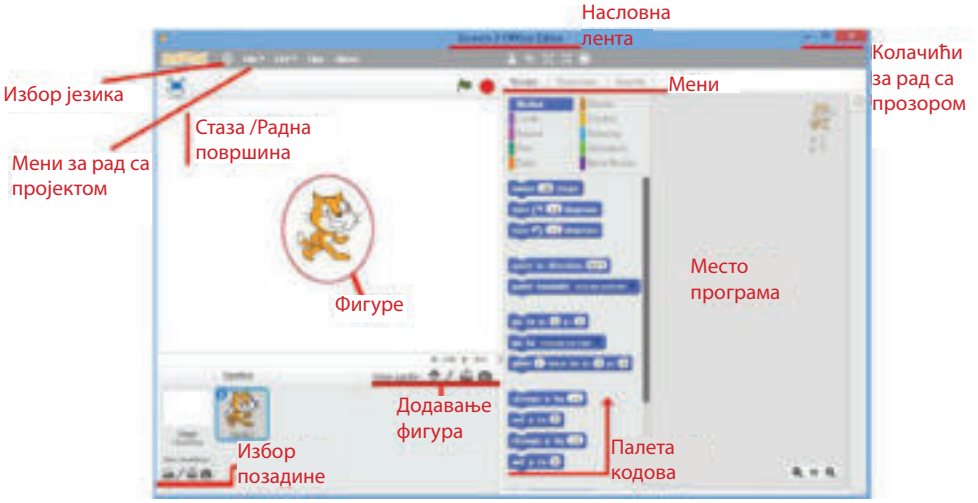
Програмирање у визуелном окружењу представља занимљив начин учења концепта и корака у стварању програма. Главна карактеристика овог начина програмирања је то што су све алатке, фигуре, наредбе, па чак и кодови **графички приказани**. Овај интерфејс омогућава ученицима најлакши и најједноставнији начин за креирање пројеката. Приликом израде пројеката ученици развијају своју креативност и креативно размишљају, почињу систематски да расуђују и да раде заједно. Најчешће коришћени програмски језик који се имплементира у визуелно окружење за програмирање је **Scratch.**

Scratch је програмски језик који вам омогућава да креирате интерактивне приче, анимације, игре, музичке инструменти итд. Пошто су све алатке графички приказане одговарајућим иконама, ученици могу да вуку и комбинују блокове кодова, да укључе ефекте анимације, аудио-ефекте итд., без учења синтаксе или текстуалних наредби програмског језика.

Визуелно окружење за програмирање **Scratch-a** може да се користи *online* на следећем линку: **https://scratch.mit.edu.** Ова страница представља званичну страницу **Scratch-a.** Прво се креира кориснички налог помоћу кога се корисник пријављује под својим корисничким именом и лозинком, а затим започиње стварањем пројеката. Постоји и *offline* верзија која се бесплатно преузима са званичне странице.

3.1.1 Покретање Scratch-a

Ванмрежна верзија Scratch-a отвара се двоструким кликом на њену икону која се налази на радној површини **рачунара (Desktop)**, што отвара следећи прозор преко кога приступамо алаткама за рад:



Слика 1: Почетни екран Scratch-a

Опишимо детаљно радни прозор:

- насловна трака приказује име програма;
- дугмад за прозор омогућавају минимизацију, максимизацију и затварање пројекта;
- алат за избор језика омогућава кориснику да бира на којем језику ће бити приказ менија и наредби. Scratch је развијен на више језика;
- преко менија *file* корисник врши активности на пројекту: чување, отварање новог пројекта, отварање постојећег пројекта и сличне активности;
- преко траке менија приступамо наредбама и уређивању објеката који ће се користити у пројекту;
- на радној површини су постављене фигуре у којима се налазе ликови у пројекту;
- кроз одељак за додавање фигура, корисник може направити избор ликова који ће бити део пројектног задатка;
- избор позадине омогућава примену одговарајуће теме пројектног задатка;
- палета наредби садржи блокове наредби или исказе које корисник може да изабере. Пошто они имају различите функције, односно различито се примењују у различите сврхе, различито су обојени. Помоћу боја корисник зна у којој категорији припада наредба;
- локација програма је скуп сетова инструкција које је корисник одабрао и комбиновао превлачењем до показују и изводе активност.

3.1.2 Упознавање са основним алатима програма Scratch

Објекти (Sprite)

Када корисник почне да креира програм, након покретања визуелног окружења на екрану прво уочава фигуру или лик који има доминантну улогу у пројекту, односно који ће бити програмиран за обављање делатности. У почетку је лик Scratch логотип, али може се променити кликом на икону Choose Sprite From Library и избором из галерије, као што је приказано на слици:

Слика 2: Галерија ликова у Scratch-у

Са слике се види да су ликови подељени на различите категорије. Објекат (**sprite**) се додаје кликом на категорију, после на објекту и на крају на дугме **OK**.

Објекти који нису у галерији могу се додавати са рачунара, камере или лика који је креирао корисник.

Позадина (Background)

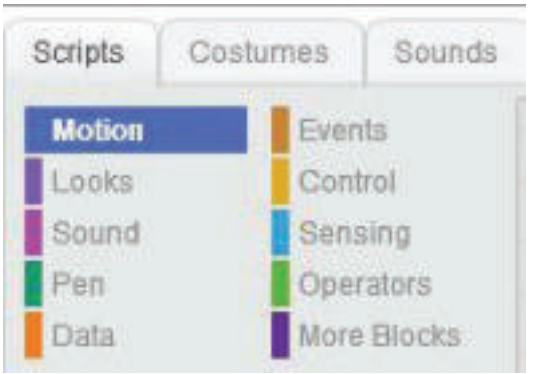
Позадина је постављена на исти начин као и ликови. Кликните на икону за промену позадине и појављује се прозор за избор позадине, као што је приказано на слици:

Слика 3: Галерија тема за позадину

Теме позадине такође се могу додати са рачунара или се могу самостално направити.

Наредбе (Искази)

У програмирању искази представљају наредбе које говоре рачунару шта треба да уради. У Scratch-у су искази у одељку скрипте:



Слика 4: Различити искази (наредбе) у Scratch-у

На слици примећујемо да су искази, односно наредбе подељене у категорије: покрет, изглед, звук, оловка, догађај, контрола итд. Свака категорија догађаја или наредба означена је засебном бојом. Кодови ових наредби су такође у облику блокова у боји, као и категорија.

Креирање програмског корака врши се кликом и повлачењем блока и повезивањем блокова, односно корака у једну целину. Када се поставе ове наредбе, почиње се покретањем креираног програма.

3.1.3 Креирање једноставних програма у Scratch-у

Да би се разумеле могућности и функционалности алата Scratch, направићемо једноставан програм. Тема пројектног задатка је плес. Хајде да почнемо!

Прво започињемо програм Scratch и бирамо лик. За ову пројектну активност бирамо лик Cassy Dance и смештамо га у средини радне површине. Како да нађемо средину радне површине?

Покушајте!

Уз помоћ наставника математике покушајте да дефинишете шта је координатни систем? Како је представљен?

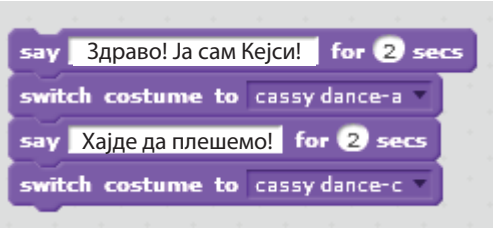
Радна површина у програму **Scratch** представљена је у облику координатног система који се састоји од оса x и y. Координатни систем је у облику математичке мреже или обрасца који се састоји од многих тачака које се називају **координатама**. Локација тачака заправо одређује вредности x и y. Центар или средина радне површине има вредност: x = 0 и y = 0. Свака тачка изван средине има своју засебну вредност x и y. У ствари, кроз вредности координата одређује се место, односно положај предмета.

Лик **Cassy Dance** је предмет у пројектном задатку. Да бисмо га поставили на средину радне површине, путем менија **motion** бирамо блокове: **set x to 0 и set y to 0**, кроз које пролазе вредности x и y и означићемо их са 0:

Слика 5: Додељивање вредности x и y

При избору блокова помоћу наредбе за подешавање вредности на осама примећујемо да је место на коме ће се одредити вредност у облику круга, односно поља за унос. Указује нам на то да према потребама вредност можемо променити ручно у складу са потребама. Стога, у програмирању ова резервисана места за вредности називају се **варијабле** или **променљиве**.

На почетку, објекат, у нашем случају Cassy Dance, може дати уводне поруке. То радимо кроз мени Scripts и опцију Looks, при чему наредбама бирамо следеће блокове:



Слика 6: Сет наредби за писање поруке

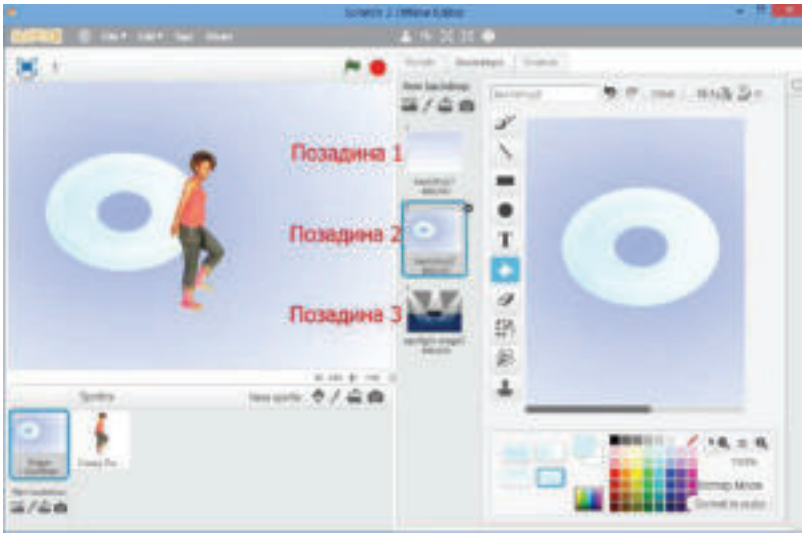
На основу кода се примећује да, поред давања наредбе за поруку, промене се врше и у положају предмета, чиме се стиче утисак да се лик креће. Да почнемо реализацију сета наредби, пре сета љубичастих кодова, кроз мени **Events** додајемо следећи блок са наредбом:

Слика 7: Додавање почетка за извршавање сета инструкција

Уз поступак додавања **позадине (Backdrop)** бирамо **Spotlight Stage 2**. Док је позадина изабрана, изаберите на траци менија **Backdrop** и добићемо овакав изглед:

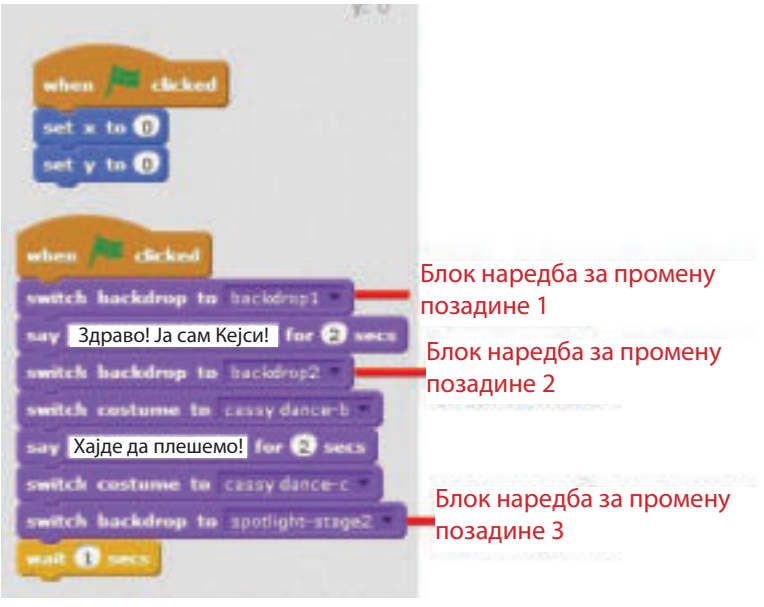
Слика 8: Уређивање позадине

Направите копију **backdrop 1** десним тастером миша и опцијом **duplicate**. Појављује се још једна бела позадина. Можемо уредити ове две беле позадине избором алатке **Fill with color**, одабиром боје са палете са бојама које се налазе на дну прозора и бојимо позадину. Понављамо исто са **backdrop 2.**



Слика 9: Креирање и уређивање позадина

Даље, ове додате позадине, креиране и уређене, додајемо у код пројектне активности. Помоћу менија **Look**, додајемо блокове са кодовима за промену позадине. У прилогу следи редослед наредби:



Слика 10: Сет наредби за промену позадине у пројектној активности

Да бисмо објекат поставили на сцену, додајемо вредности координата, путем менија **motion** команде set x to 15 и set y to 100. Овако постављени објекат биће програмиран за извођење плеса који ће бити праћен музиком. Блок наредбе, односно искази које треба додати имају следећи редослед:

Слика 11: Додавање понављајућих циклуса

Да бисмо завршили код, додајемо сет упутстава. На пример, свој пројектни задатак ћемо довршити поруком и заустављањем свих процеса:

Слика 12: Завршетак програма

На тај начин смо завршили наш први пројекат. Хајде да погледамо!

На левој страни радног прозора, у одељку где су приказани објекти, у горњем десном углу су иконе за почетак и завршетак пројекта:

Слика 12: Завршетак програма

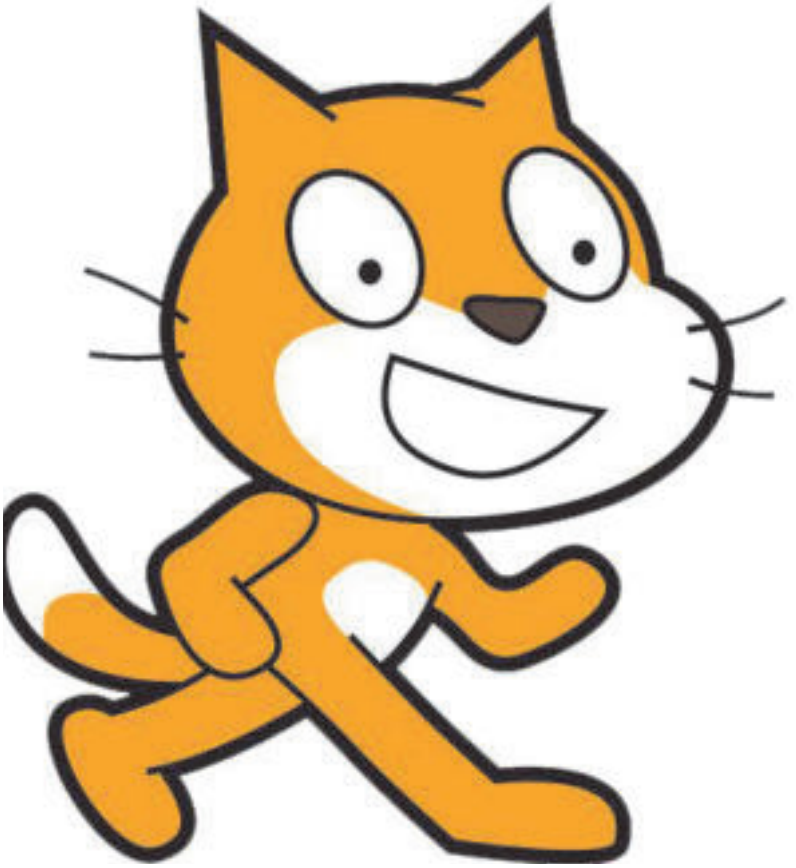
Кликом на зелену заставицу започињемо извршење сета упутстава, при чему визуелно видимо шта смо урадили, а ако желимо поново да зауставимо приказ, кликнемо на црвено дугме за заустављање.

Запамти!

Програми за графичко програмирање омогућавају креирање секвенце наредби за извршење програма помоћу алата, фигуре, наредбе итд., који имају графички приказ. Scratch је програмски језик који вам омогућава да креирате интерактивне приче, анимације, игре итд. Команде у Scratch-у приказују се у облику блокова који личе на слагалице. Комбинују се повлачењем и спајањем. Радна површина Scratch-а може се математички представити у виду мреже или шеме састављене од тачака – координате. Додати објекти у задатку у пројекту Scratch добијају блокове са исказима (наређења). Искази (команде) говоре рачунару шта да уради. Неки искази садрже резервисана места за вредности и називају се варијабле или променљиве.

Питања

1. Шта значе блокови у Scratch-у? Како изгледају?
2. Како се деле блок наредбе? Наведи неколико категорија!
3. Шта значи зелена застава у горњем делу приказа пројекта?
4. Шта нам показују координате објекта у пројектном задатку?
5. Како се поставља позадина пројектног задатка?



3.2 Интерактивни програми са догађајима

Да се подсетимо!

Да ли се сећате када смо први пут споменули појам „интерактивни програми“? У којим темама обраде? Можете ли се сетити како смо дефинисали овај појам?

Термин „**интерактивни**“ се већ раније спомињао у различитим темама и у различитим областима. На пример, час информатике можемо окарактерисати као **интерактивни** јер постоје активности, комуникација за реализацију активности. Често се дискутује на часовима за решавање задатака и решавање проблемских ситуација, што значи да сви размишљамо наглас, износимо закључке, ставове, али и одговоре на многа питања.

Иста ствар се дешава када користимо образовни софтвер. Наш програм нам даје упутства, а ми као корисници следимо упутства и у складу са њима предузимамо одговарајуће активности.

Што се тиче рачунарске игре, придржавање правила игре заправо представља ту интеракцију између корисника и ње саме. Сваки корисник жели да постигне циљ, односно да победи и зато мора да делује у складу са правилима игре.

У овој наставној јединици креираћемо пројекат који укључује догађаје **интерактивне природе**.

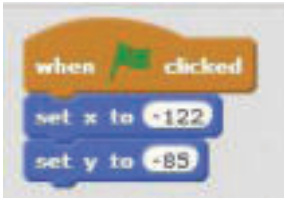
Главни лик у нашој интерактивној игри је **Dinoaur1**. Он ће извршавати математичке прорачуне задатака које ћемо задати. За почетак, креирамо низове блок наредби које ће рачунати **количник од два унета броја**.

Почнимо са програмирањем!

Прво започињемо програм Scratch. Коришћењем алата маказе (**delete**) избриши објекат, односно лик Cat. Алатом за додавање објеката, у одељку **spirit** додајемо објекат из постојеће галерије. За наш пројекат ћемо одабрати објекат **Dinoaur1**. Следеће, додајемо позадине (**backdrop**). Направићемо и уредити једну позадину помоћу алата и палете боја које нуди програм, а друга позадина је слика из нашег рачунара. Почетни екран би изгледао овако:

Слика 1: Додавање објеката и позадине у пројекат

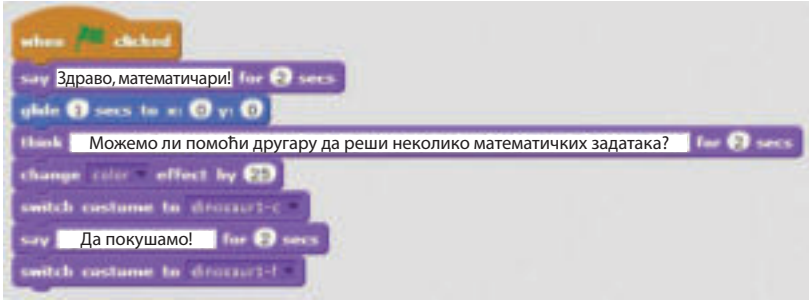
Кликните на Dinoaur1 и започните са креирањем низова блок наредби:



Слика 2: Утврђивање локације објекта

Наведени редослед блок наредби значи да кликом на зелену заставу која означава почетак извршења низова инструкција, вредност **x** и **y** треба да буде **-122** и **-85**, што значи да смо утврдили локацију **Dinoaur-a 1**.

Следећи блок кодови су почетни догађаји у пројекту:



Слика 3: Низ блок наредби које објављују поруку

А то значи да ће се, када кликнемо на зелену заставицу, извршити секвенца блок наредби. Пре свега, поздравља нас Dinoaur1 са „**Здраво математичари**“, помера се, односно „**склизне**“ на средину радне површине. Када Dinoaur1 дође до средине радне површине, приказује се као да размишља, мења своју боју и његов приказ је сличан кретању.

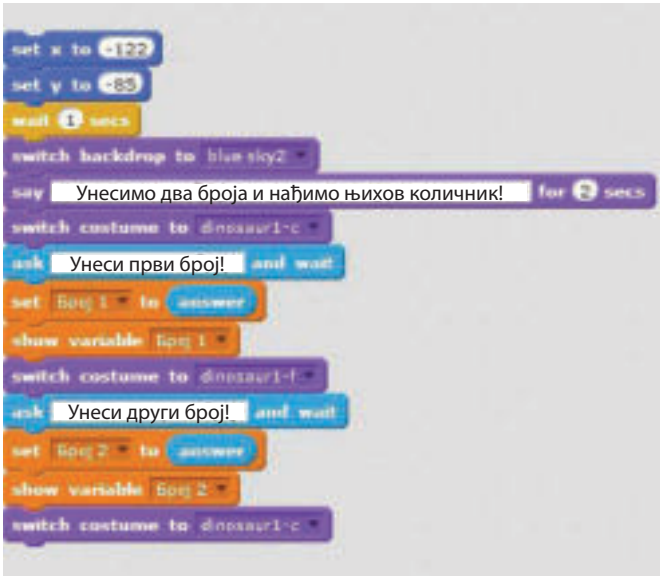
Даље, настављамо са редовима блокова који ће нас довести до нове ситуације. Dinoaur1 ће давати упутства, а ми помоћу тастатуре уносимо потребне податке. У ствари, Dinoaur1 је наш математичар и он ће нам рећи резултат израчунавања количника два унета броја. У том циљу ћемо прво дефинисати три променљиве, односно променљиве које ће имати вредности унете помоћу тастатуре и резултат прорачуна. Поступак за креирање променљивих је следећи: преко менија Scripts приступамо категорији Data и тамо кликните на Make variable. Зато што ћемо користити ову променљиву и у другим ситуацијама у пројектној активности, кликнемо на опцију и променљиву именујемо.

Коментар!

Можете дати произвољно име варијабли односно променљивој, али подсетимо се какву ће вредност имати и која је улога. На пример, у задатку смо дефинисали променљиве именима Број 1 и Број 2, али би се могле дефинисати и као дељеник и делилац по математичким називима.

Слика 4: Креирање варијабли/променљивих

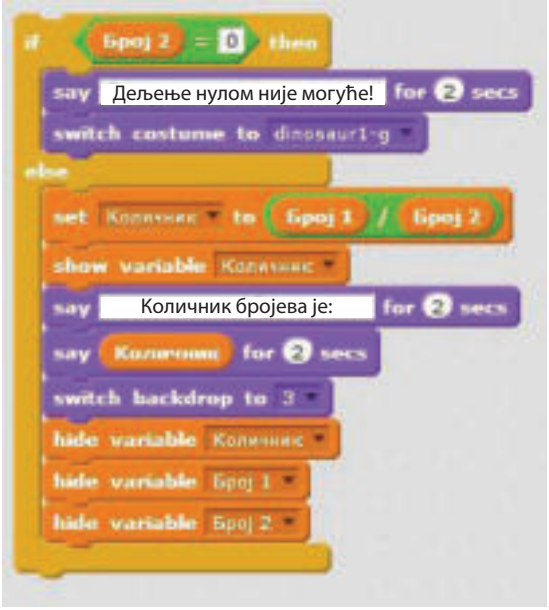
У конкретном пројектном задатку креирамо **три променљиве**. Прва променљива ће садржати вредност броја (**број 1**) коју прво унесемо са тастатуре, друга променљива ће садржати вредност унету за други број (**број 2**) и трећа променљива (**количник**) приказаће вредност која произилази из решења. Применимо креиране променљиве у складу с тим!



Слика 5: Примена блок наредби из категорије Data

Користећи мени **Motion**, дајемо вредности **x** и **y** да бисмо вратили **Dionaur1** у првобитни положај, односно на почетну тачку. Из менија **Looks** дајемо блок наредбу за промену позадине (**backdrop**) и блокирај команду за приказ поруке. **Dionaur1** мења његов изглед и каже нам да унесемо први број. Овде се примењује блок наредба преко које нам се даје могућност уноса броја са тастатуре и приказ поља са унетом вредношћу. Исто поновите за други број, као што је приказано на слици.

Следећа ствар коју треба да урадимо јесте да проверимо да ли је вредност другог броја 0, јер математичко правило каже да дељење нулом није могуће. У ту сврху, користећи мени **Control**, примењујемо блок наредбу: „ако је услов задовољен, онда треба извршити следеће инструкције“ и „ако услов није задовољен, треба извршити следеће инструкције“, односно:

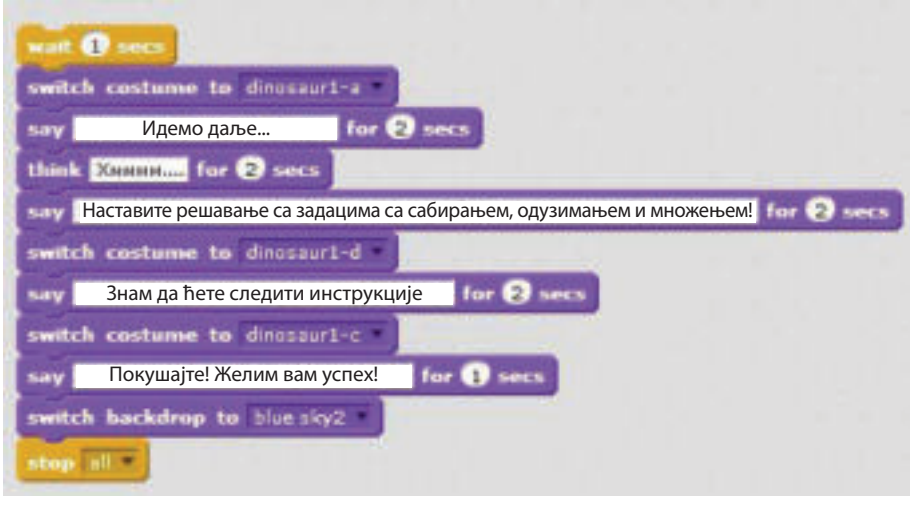


Слика 6: Примена блок наредбе If/else

Или, ако је вредност другог броја једнака нули, приказује се порука: „дељење нулом није могуће“ и **Dionaur1** мења изглед. Ако услов није испуњен, онда:

- извршити израчунавање количника;
- приказати вредност количника;
- приказати количник као поруку од **Dionaur-a1**;
- прећи на нову тему, односно позадину;
- када прећемо на нову позадину, променљиве неће бити видљиве.

Тиме смо завршили израчунавање количника оба броја. Пројекат можете довршити произвољно. На пример, у нашој ситуацији **Dionaur1** наставља са порукама:



Слика 7: Блок наредбе за приказ порука

И претходно смо применили поступак приказивања порука на екрану. Поруке зависе од креативности корисника и како је он визуелно замишљао решење пројектног задатка. Наредбом „**stop all**“, означавамо крај извршења низова наредби.

Кликом на зелену заставицу започињемо извршење низова блок наредби и тако вршимо проверу онога шта смо направили. Овај пројекат је интерактиван јер **Dionaur1** задаје инструкције, говори нам како да поступамо, а ми следимо та упутства предузимањем активности са тастатуре.

Задатак

Dionaur1 нам је помогао да пронађемо количник два унета броја. Наставите задатак тако што ћете створити низове блок наредби, који ће рачунати и друге математичке операције, као што је збир два броја, разлику два броја и производ.

Размисли!

Да ли се може креирати секвенца блок наредби за извршење прорачуна математичких операција са три броја? Шта нам недостаје у програму Dionaur 1 који смо креирали за извршење прорачуна уносом два броја? Објасните поступак! Покажите где бисте направили измене!

Запамти!

Програми у које корисник може да унесе вредности променљивих којима се приказују резултати и зависе од улазних јединица називају интерактивни програми. Варијабле су променљиве које добијају вредности приликом уноса података са тастатуре или као резултат прорачуна. Варијабле/променљиве се креирају из менија Scripts и категорије Data.

Питања

1. Који се програми називају интерактивним?
2. Шта су променљиве? Каква је процедура за креирање променљивих?
3. Којом се командом омогућава унос вредности променљиве?
4. Како корисник уноси у програм вредност променљиве?
5. Чему служи команда If... Else?

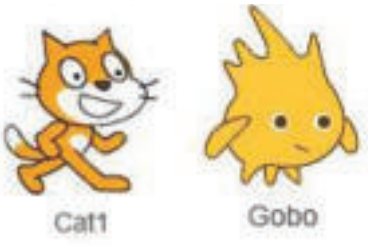
3.3 Израда програма са сложенијим проблемским ситуацијама

Креирање игре у **Scratch**-у је мало сложеније јер укључује више **објеката** и **догађаја**. Сваком објекту се додељују инструкције или сетови инструкција да би се постигао циљ, односно да се дође до коначне победе у игри. Притом, обавезно треба предвидети догађаје који би се могли догодити. Свакако, очекује се да се за њих понуде одговарајућа решења. На пример, ако се победи, добити поруку или чути победнички звук, а ако се изгуби, вратити објекат на почетак.

На пример, направићемо игру која има два објекта: **Cat** и **Gobo**. Cat се креће према кретању миша – лево и десно. Више Gobo-a пада с неба, а Cat мора сакупити што више укупног броја поклона.

Покренимо нови пројекат у **Scratch**-у.

Кликом на дугме **Choose Sprite from library** додајемо нови објекат/карактер. Из галерије са објектима/ликовима бирамо Cat и Gobo:



Слика 1: Објекти/ликови у пројекту

Алатком **Shrink** мењамо димензије објеката/ликова. Кликните на алатку Shrink, а затим на објекат, односно лик, онолико пута колико желите да смањимо објекат. У нашем случају смањићемо димензије два објекта:

Слика 2: Промена димензија помоћу алата Shrink

Тему или позадину додајемо помоћу алата **choose backdrop from library**. За нашу игру користићемо тему **Gingerbread**:

Слика 3: Тема пројекта

Даље, сваком објекту треба доделити сетове или низове упутстава да би омогућили анимирање објеката и да извршавају неку акцију. На пример, објекат **Sprite1** треба да се креће лево и десно и да сакупља **Gobo**:

Слика 4: Сет упутстава за померање Sprite-а 1

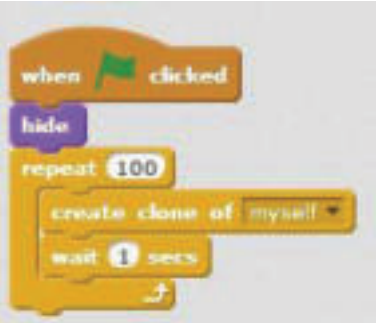
Коментар!

На основу наведених блок наредби примећујемо да постоје блок наредбе представљене у различитим бојама, што нам показује да припадају различитим категоријама наредби.

Објаснимо их блок по блок командама!

Кликом на зелену заставицу покреће се процес извршавања упутстава која следе. Притом дајемо им вредности на координатама **x** и **y**, **0** и **-130**. Означавамо правац кретања **90 удесно**. Објекат/лик ће се кретати дуж **x-осе** према кретању миша. Исто тако, кликом на зелену заставицу почеће емитовање **звучне датотеке**. У нашем случају, звук се бира из галерије звукова и бирамо звук **dance slow mo**.

Такође треба задати инструкције и објекту/лику **Gobo**. По нашем задатку, требало је да више **Gobo-a** или клонова **Gobo-a** падне са неба. Иако смо дефинисали само један објекат – поклон, ипак у игри је тај предмет бројнији. Да видимо поступак:



Слика 5: Креирање клонова објекта

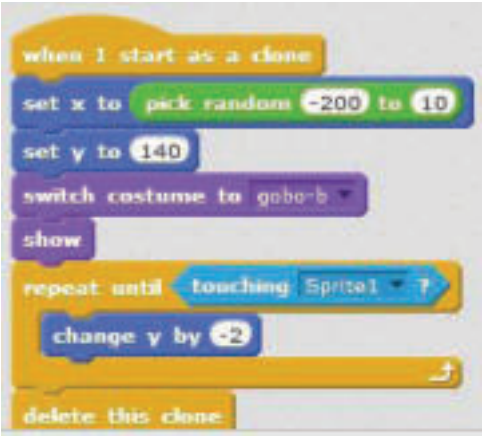
Ова секвенца или скуп наредби значи да се кликом на зелену заставицу означава почетак извршења низа наредби за стварање клона, односно копија објеката/ликова. Блок наредбом repeat омогућује се да креирате клонове сваке секунде све док се не достигне вредност 100.

Следеће што треба урадити је креирање блокова са наредбама које ће рећи објекту – клону како да поступа и које радње треба да изврши:

Слика 6: Инструкције за објекат – клон

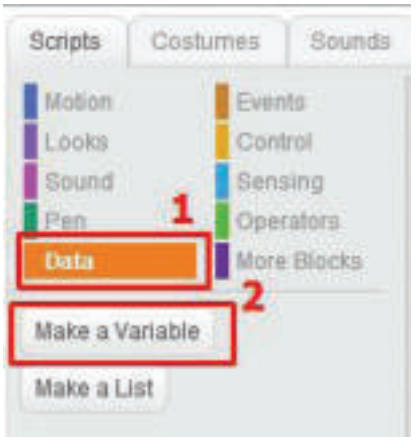
То би значило следеће: када се појави клон – објекат, помераће се наделе до објекта **Spirit1**. Када додирне **Spirit1**, почеће емитовање музичке датотеке која се зове **zoop**. Цикличном наредбом **repeat**, сваке секунде промениће се четири различита облика објекта **Gobo**. Одмах затим, објекат ће се избрисати, односно нестаће објекат – клон.

Следећи сет инструкција указује на то да ће се објекат – клон појављивати на произвољним координатама у опсегу од -200 до 200 у односу на **x-осу** и кретаће се почев од вредности 140 у односу на **y-осу**. Притом ће мењати свој приказ све док га не додирне објекат **Spirit1**. На крају ће се објект – клон изгубити.



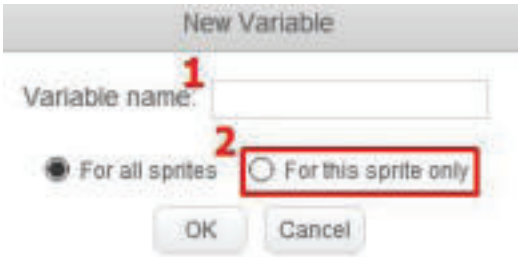
Слика 7: Постављање произвољних координата за појављивање објекта – клона

Недостају нам бодови у игри. Следећим редоследом наредби омогућићемо бројање. Када **Gobo** додирне предмет **Spirit1**, тада ће се догодити промена бодова. Прво, морамо додати нову променљиву која ће на почетку имати вредност 0, а затим ће се повећавати за 1. Додавање променљиве врши се кликом на опцију **Data** из менија **Scripts**, затим на опцију **Make variable**:



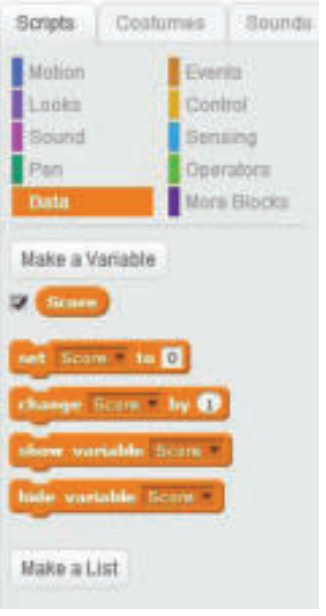
Слика 8: Креирање нове променљиве

Појавиће се следећи прозор путем кога корисник даје име променљивој и означава да ће се креирана променљива користити за изабрани лик:



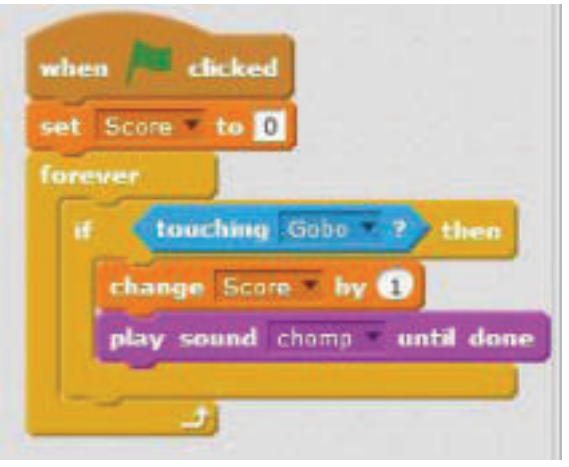
Слика 9: Дефинисање нове променљиве

Кликом на OK додају се следеће команде блок наредби које се односе на нову променљиву:



Слика 10: Блок наредбе које се односе на нову варијаблу **Score**

Након креирања променљиве, креирамо секвенцу блок наредби које треба да се изврше, тј. да броје бодове:



Слика 11: Бројање поена

Да видимо како функционише игра! Кликните на зелену заставицу и почните да играте.

Задатак

Направите једноставну игру у **Scratch-y**. Игре које сте креирали поделите са пријатељима и играјте се пратећи упутства.

Запамти!

Пројекти који укључују више објеката и више догађаја су сложенији за креирање. Сваки објекат се програмира одвојено, тј. задају му се низови блок наредби које треба да изврши како би се креирало кретање, активност или радња. Објекти у програмима су описани са условом и начином понашања. У стварном свету објекат може да буде: аутомобил, бицикл, лопта или било који лик који је дефинисан стањем и понашањем. Догађаји се дешавају у зависности од ситуације или понашања објекта. Треба их предвидети и понудити одговарајуће решење.

Питања

1. Шта су објекти?
2. Како се програмирају објекти?
3. Каква је веза између објеката и догађаја?

ПОНОВИМО! УВЕЖБАЈМО!

Задатак 1

Покушајте да направите пројектни задатак у **Scratch-y** применом блокова са исказима (наредбама) на којима смо практично радили на часу. Уметните позадину, додајте објекат, преместите објекат, додајте звук итд.

Задатак 2

Креирајте једноставну игрицу у **Scratch-y** која укључује више објеката и догађаја.

Задатак 3

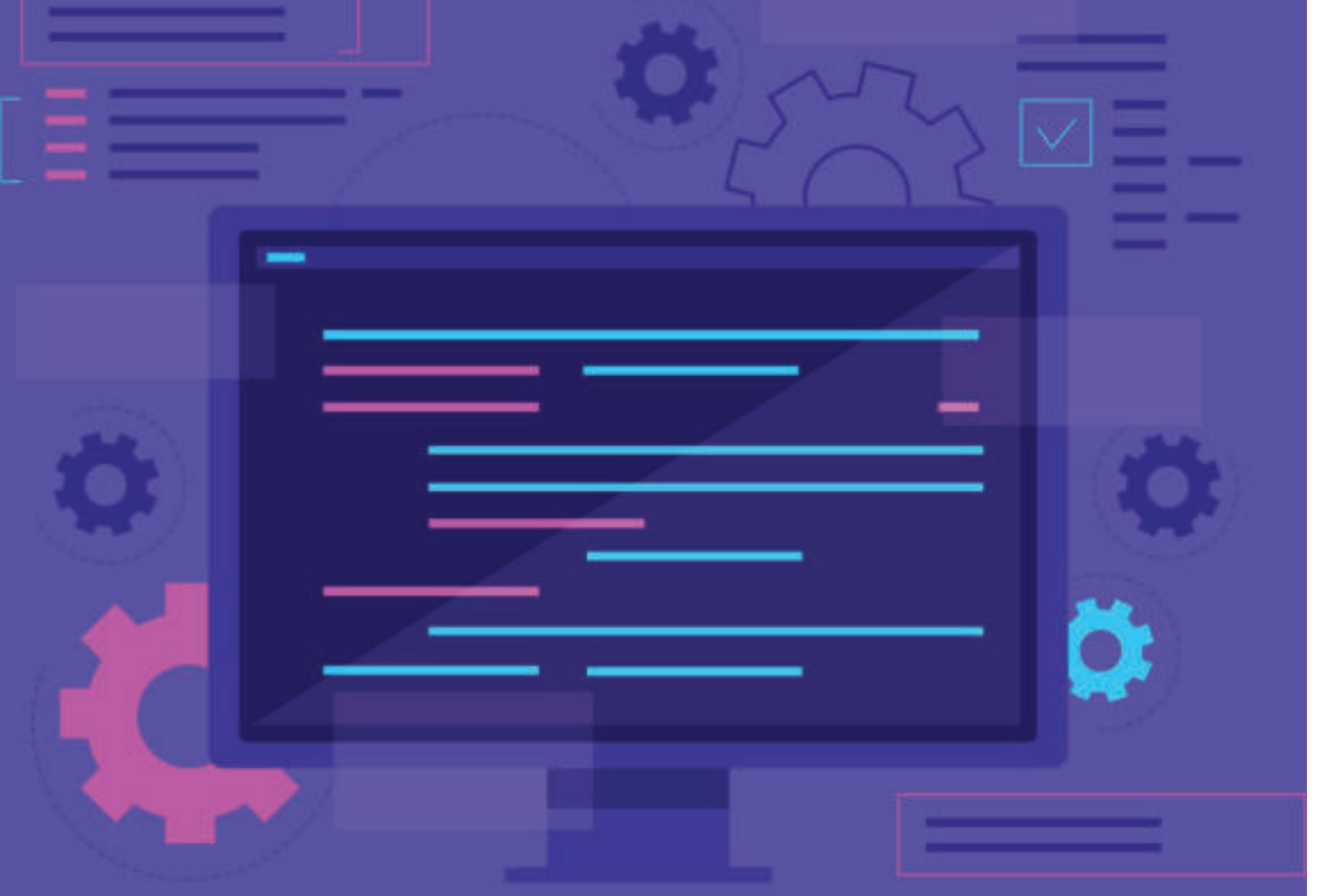
Креирајте посебне програме који ће израчунавати површину и обим квадрата, правоугаоника и круга.

Задатак 4

Креирајте пројект у Scratch-у са интерактивним карактером, у коме ће се при уносу броја са тастатуре проверавати да ли је позитиван или негативан.

Програмирање путем стандардног структурног програмског језика

Увод у програмирање путем стандардног структурног програмског језика
Процес израде једног програма
Упознавање основних елемената интегрисане околине програмирања
Изглед готових примера програмских кодова
Извршавање готових примера програма
Основни елементи програмског језика C++
Искази
Израда програма
Аритметичке операције и изрази
Константе и променљиве
Искази (технике) за уношење података у програм
Упоредни искази
Структура понављања циклуса до испуњавања услова
ПОНОВИМО! УВЕЖБАЈМО!



4 Увод у програмирање путем стандардног структурног програмског језика

Шта ћемо научити?

Научићемо да радимо у интегрисаној околини за програмирање Code::-Block и да креирамо програм у програмском језику C++.

Живимо у време брзих промена и још бржег технолошког развоја у свим областима нашег друштва. Такорећи, не може се замислити ниједан посао који ће се обавити без примене рачунара. Полазећи од чињенице да је рачунар електронски уређај који се употребљава за прикупљање, чување и обраду података помоћу програма који су инсталирани у њему, долазимо до закључка да он не би био најмоћнија алатка која нуди решење за сваку проблемску ситуацију, за различите циљеве и из различитих области.

У нашој свакодневици користимо различите програме, као на пример:

- играмо забавне игрице
- пишемо есеје у програму за обраду текста (MS Word, Open Office Writer и др.);
- вршимо прорачуне у програму за табеларне прорачуне (MS Excel, Open Office Calc и др.);
- креирамо презентације у програму за креирање и уређивање мултимедијалних презентација (MS Power Point, Open Office Impress, Preci и др.);
- гледамо видео-датотеке помоћу адекватних програма (Winamp, VLC Player, Real Player и др.);
- креирамо и уређујемо фотографије у графичким едиторима (MS Paint, In Design, Corel Draw, Photoshop и др.);
- комуницирамо са рођацима и пријатељима путем програма за комуникацију (Viber, Messenger, WhatsApp и др.).

Постоје и други програми које корисник може да наручи да би се креирали у сагласности са његовим потребама. У суштини, програми представљају везу између корисника и рачунара.

Према томе, рачунарски програми представљају низ инструкција које рачунар треба да изврши. Процес осмишљавања, креирања и повезивања низова са инструкцијама назива се програмирање. Људи који креирају програме помоћу програмских језика називају се програмери.

Програмски језици представљају мноштво правила, симбола и кључних речи, који се комбинују на различите начине у циљу долажења до решења задатка или проблемске ситуације. Учење програмирања развија логички начин размишљања који је користан и применљив у свакодневним ситуацијама.

Кључни појмови!

преводилац, изворни програм, извршни програм, едитор, компајлер, дебагер, исказ, упоредни израз, циклус.

4.1 Процес израде једног програма

Израда једног програма било да је намењен рачунару, телефону, таблету или другим уређајима представља сложен процес зато што се кораци за његово креирање пишу неким програмским језиком. Постоји много програмских језика и сваки од њих има карактеристично мноштво инструкција и синтакси у писању. **Програмски језици** се обично деле на **више** и **ниже**.

Нижи програмски језици су машински оријентисани, јер зависе од машине на којој се израђују, а то значи да проблем написан на једном процесору не мора да функционише на другом процесору. Зато је програмирање у машинском коду веома тешко, јер треба добро познавати структуру и грађу рачунара.

За разлику од нижих програмских језика, виши програмски језици се чешће употребљавају, јер не зависе од машине на којој се раде, а то значи да могу да се изводе на рачунарима са различитим процесорима. С друге стране, њихови симболи, знаци али и синтакса су веома слични природним језицима и лако су разумљиви. Најчешће употребљавани виши програмски језици су: **C, C ++, Fortran, Basic, Pascal, Java** и други. Без обзира на то који виши програмски језик употребљавамо при креирању програма, процес се свакако одвија у четири **фазе**:

Ред. бр.	Фаза	Значење
1	Писање изворног кода	Писање изворног кода се врши у програмском језику у интегрисаном окружењу за програмирање. Притом се овај програм назива **изворни код** (**Source Code**) који се памти у меморијским уређајима на рачунару као датотека са наставком .cpp, на пример: Vezba.cpp
2	Превођење изворног кода	Помоћу програма за превођење који се називају компајлери преводи се изворни код. При превођењу изворног кода могу се уочити грешке. Ове грешке се називају синтаксне грешке и најчешће се односе на погрешно написане речи програмског језика, неправилно написане или заборављене интерпункцијске знакове и сл. При превођењу се добија датотека објектног кода који има наставак Obj, на пример: Vezba.obj
3	Повезивање у извршном коду	Помоћу програма под називом повезивачи (**Linkers**), повезују се објектни кодови у извршни код (**Executeble code**). Извршни код извршава рачунар. Притом се креира извршна датотека која има наставак **.exe**, на пример: vezba.exe
4	Тестирање програма	Тестирање програма служи за проверу програма о томе да ли испуњава постављене услове и долази до тачног решења, односно представља један тип истраживања који обезбеђује информације о квалитету производа и евентуалном проналажењу грешака. Ова фаза представља и процес валидности и верификације програма, тј. испуњава захтеве наведене у документацији, ради према очекиваном и може да се примени на све његове карактеристике.

Табела 1: Фазе при писању програма

Покушај!

Подсети се на Offline визуелни програм Scratch. Упореди где и како сте писали кодове, како се компајлирало и како су се проналазиле грешке при креирању програма. Уочи сличности и разлике.

У процесу креирања програма у овој тематској јединици креираћемо програме помоћу C++ програмског језика у Code::Block интегрисаном окружењу за програмирање.

Запамти!

Рачунарски програми су низови инструкција које извршава рачунар. Процес стварања програма назива се програмирање. Људи који креирају програме помоћу програмских језика називају се програмери. Програмски језици се деле на ниже и више. Фазе при креирању програма су: писање изворног кода, превођење изворног кода, повезивање у извршни код, тестирање и верификација програма.

Питања

1. Како се зове поступак за креирање програма?
2. Ко креира програме? Како?
3. Како се деле програмски језици?
4. Који су нижи програмски језици?
5. Наброј неке више програмске језике.
6. Која је разлика између виших и нижих програмских језика?
7. Дефиниши шта је синтакса програмског језика!
8. Шта је изворни програм? Какав наставак има датотека изворног програма?
9. Како се зове програм који извршава рачунар?
10. Наброј фазе при креирању програма!

4.2 Упознавање са основним елементима интегрисаног окружења за програмирање

Интегрисано окружење за програмирање (IDE – Integrad Development Environment) представља софтверски пакет који садржи основне алатке потребне за писање и тестирање софтвера. Ова апликација обично нуди бројне функције за писање, модификовање, састављање, поделу и дебаговање софтвера. Главна намена овог пакета алатки јесте да поједностави развој софтвера и да минимализује грешке у кодирању.

При креирању програма, корисници употребљавају бројне алатке за креирање, грађење и тестирање софтверског кода. Развојне алатке често **укључују уреднике (Editor) изворног кода, библиотеке кодова, компајлере и платформе за тестирање.** У суштини, програмер пише и уређује изворни код у уреднику кодова (Editor), компајлер преводи изворни код на читљив језик који рачунар може да изврши, а дебагер тестира софтвер да би решио било какав проблем или грешку.

Покушај!

Замисли једну проблемску ситуацију приликом свакодневних школских обавеза. Покушај да размишљаш и функционишеш као интегрисано окружење за програмирање: креирај кораке за решавање, компајлирај те кораке да би их разумели твоји другови и разговарај са њима о могућим допунским елементима, покушај да сазнаш да ли је то право решење.

Интегрисано окружење за програмирање ћемо употребљавати при креирању програма са **C++ програмским језиком Code::Blocks**.

Code::Blocks је интегрисано окружење за програмирање које подржава компајловање и тестирање више програмских језика. Ради са низом компајлера. То је бесплатни пакет алатки за развој софтвера у C++ програмском језику отвореног кода и садржи следеће алатке: едитор за текст, програме за превођење и повезивање, као и програме за откривање грешака.

4.2.1 Инсталирање програма Code::Blocks

Будући да је програм Code::Blocks бесплатан програм са отвореним кодом можемо да га преузмемо (Download) са следеће странице:

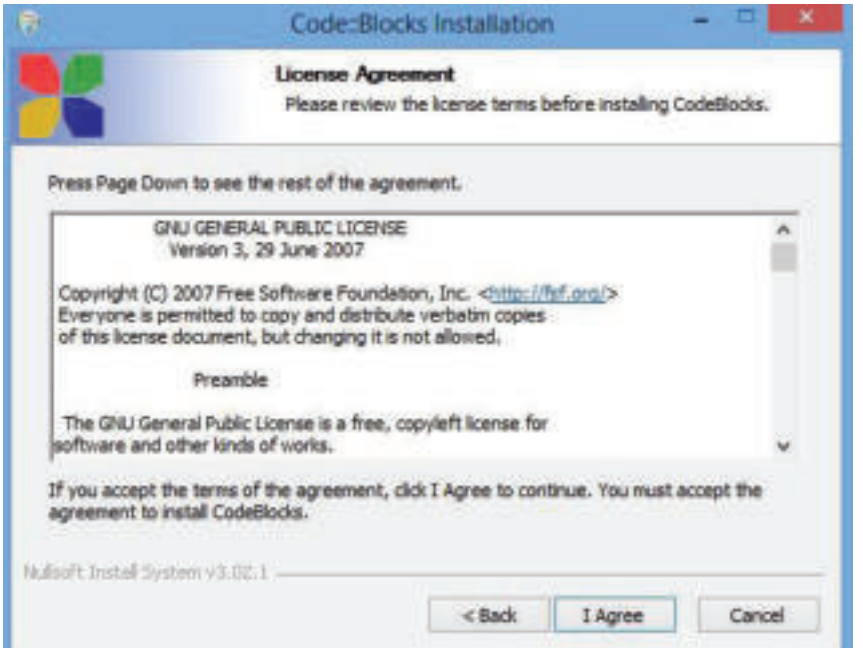
https:// //www.codeblocks.org/downloads/binaries.

На преузету извршну датотеку додајемо наредбу да буде извршена, односно да започне процес инсталирања двокликом на њега.

Појављује се чаробњак (**Wizard**) за наставак процеса инсталирања, као што је приказано на слици.

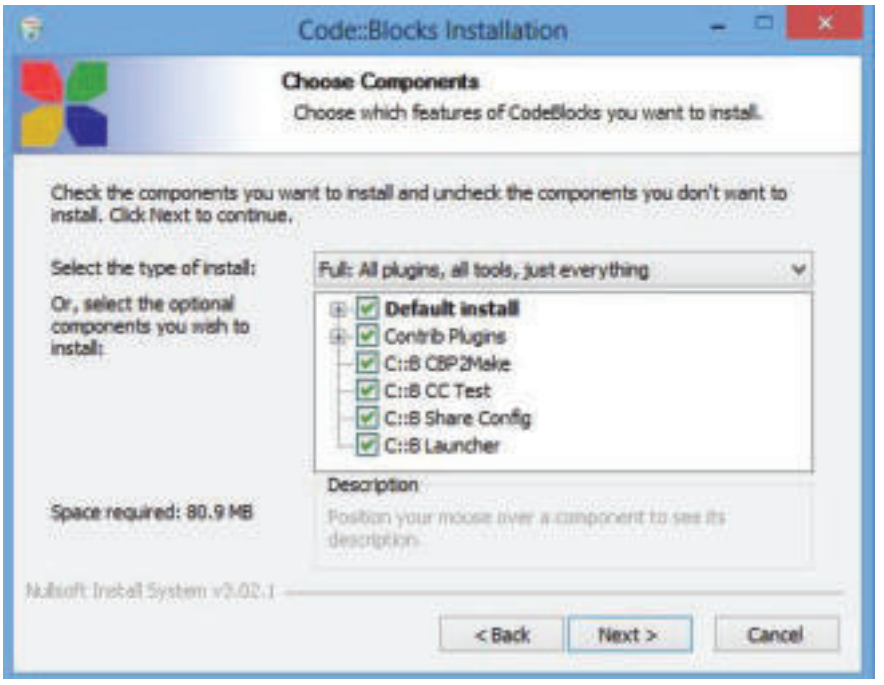
Слика 1: Почетак процеса инсталирања програма Code::Blocks

Кликом на тастер **Next** приступамо следећем кораку инсталационог процеса, а то је да ли прихватамо начела употребе овог програма:



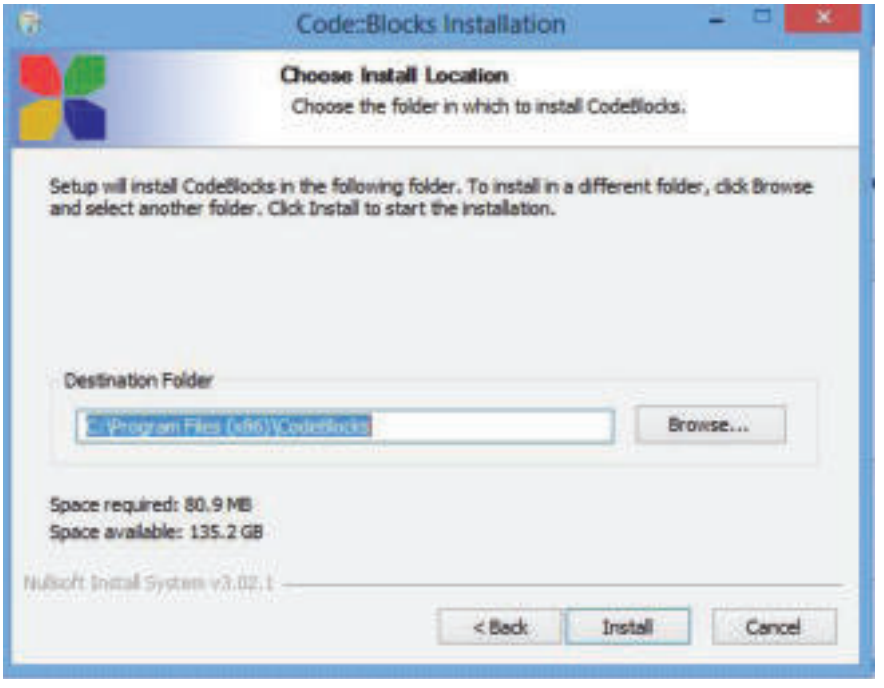
Слика 2: Прихватање правила за инсталирање програма

Да бисмо наставили процес, кликнемо на **I Agree** и крећемо ка следећем кораку, а то је избор компонената које желимо да инсталирамо као пакет овог програма.



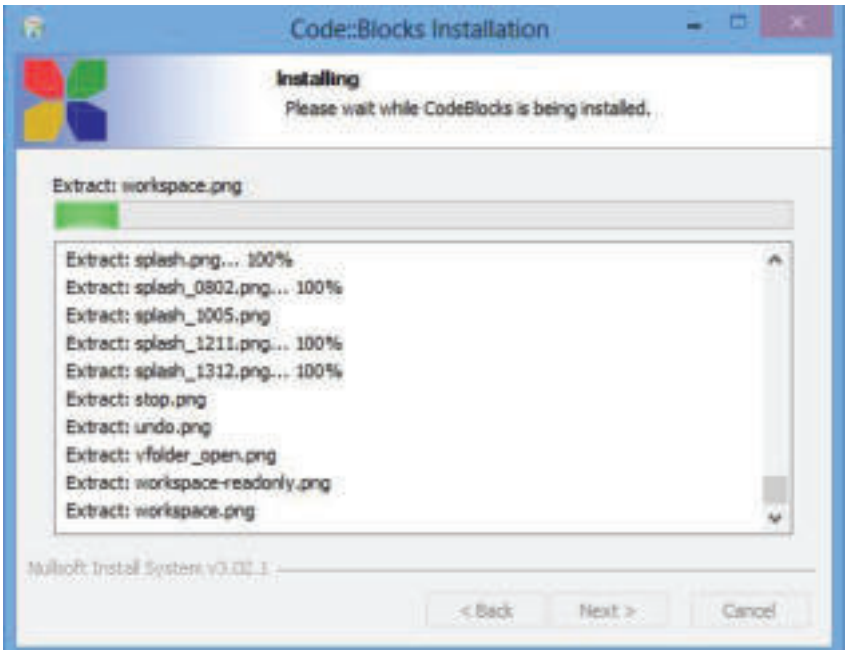
Слика 3: Избор допунских компонената у радном окружењу

Кликом тастера **Next**, отвара се следећи прозор помоћу кога ћемо изабрати директоријум у који ће се сместити датотеке које су потребне за нормално и несметано функционисање програма:



Слика 4: Избор директоријума где ће се сместити датотеке програма

Када завршимо са почетним инсталационим корацима, кликнемо на **Install** при чему отпочиње пребацивање програмских датотека у дестинациони директоријум:



Слика 5: Инсталација програма

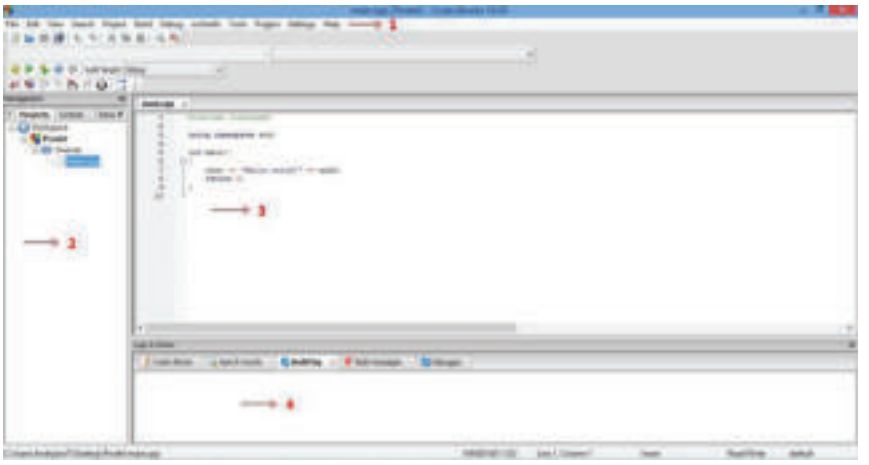
Када се заврши овај процес, кликнемо на опцију Next да бисмо приступили следећем кораку, на крају **Finish**, при чему се појављује прозор за покретање Code::Blocks:

Слика 6: Дијалог – прозор за покретање програма Code::Blocks

Из овог дијалог – прозора бирамо опцију **Yes**, притом покреће се програм и видимо радну површину интегрисаног окружења за програмирање Code::Blocks.

4.2.2 Радно окружење Code::Blocks

Након покретања програма, појављује се радни прозор програма Code::Blocks. Хајде да прегледамо радно окружење и да опишемо функционалности алатки!

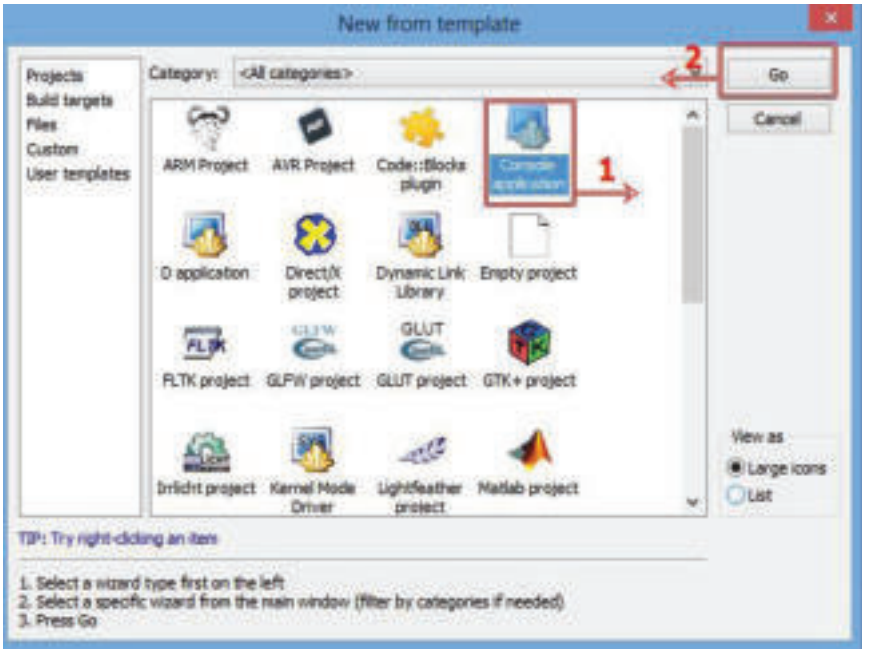


Слика 7: Радно окружење програма Code::Blocks

Радно окружење обухвата следеће елементе:

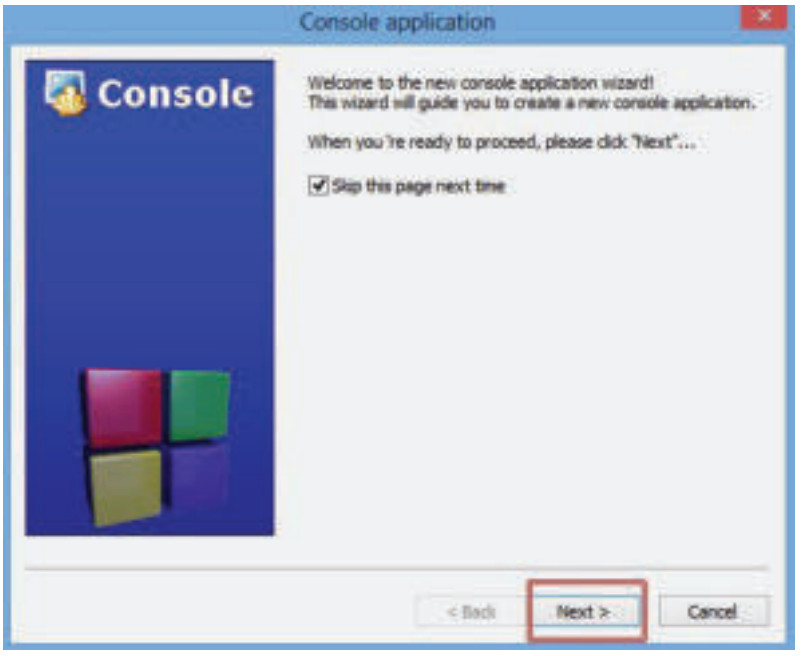
1. мени – трака са наредбама;
2. трака за управљање;
3. едитор за уношење изворног кода (**Source Code**);
4. прозор за писање порука о грешкама.

Први корак при креирању новог пројекта у интегрисаном окружењу за програмирање Code::Blocks, почиње кликом на мени **File** →**New** →**Project**, при чему се појављује следећи прозор:



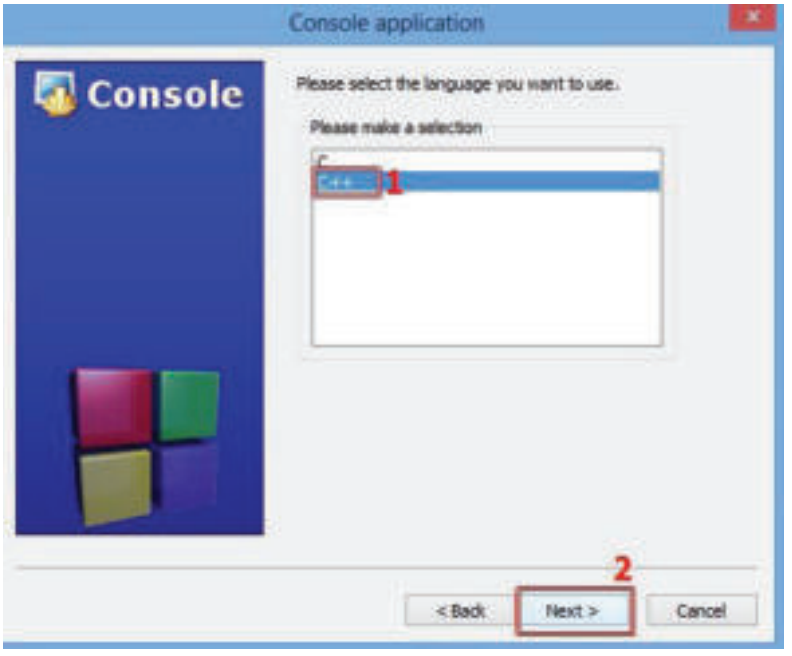
Слика 8: Креирање новог пројекта

Из прозора бирамо **Console Application** и кликнемо на тастер **Go** да бисмо наставили ка другом кораку.



Слика 9: Прилагођавање радном окружењу

Кликом на **Next**, појављује се прозор из којег бирамо који ћемо програмски језик употребити, као што је приказано на следећој слици:

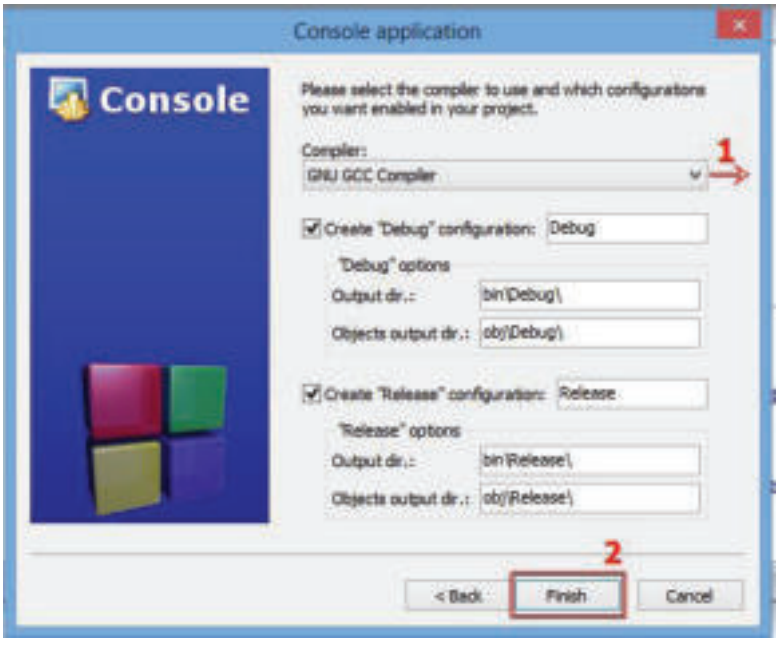


Слика 10: Избор програмског језика за писање

Кликом на тастер **Next**, приступамо додељивању имена пројекту и локацији где ће бити смештен, односно сачуван тај пројекат:

Слика 11: Додељивање имена и локације пројекту

Прво дајемо име пројекту, затим дефинишемо у који директоријум ће бити смештен пројекат, а такође дајемо име пројектној датотеци која се креира, и на крају кликнемо на тастер **Next.** Кликом на тастер **Next** приступамо избору компајлера:



Слика 12: Избор компајлера

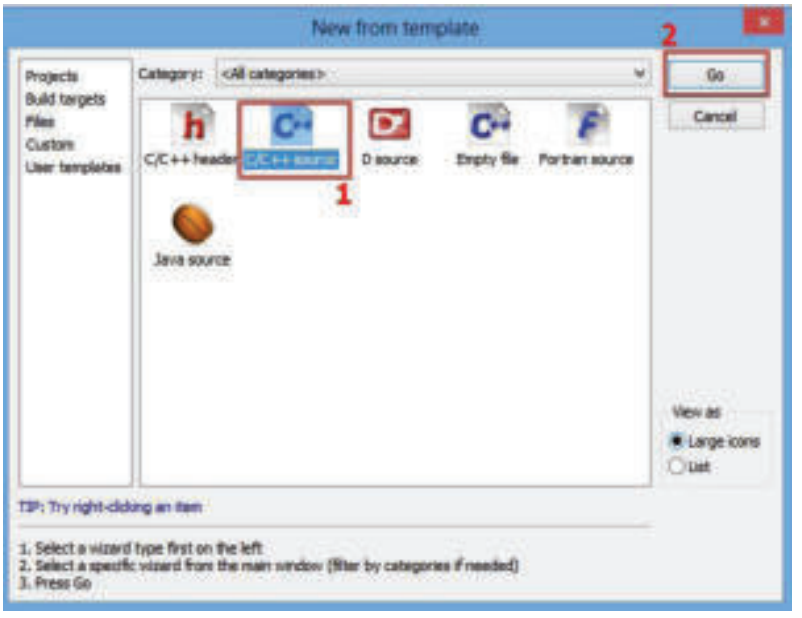
Кликом на тастер **Finish** почињемо поступак креирања изворних кодова у едитор интегрисаног окружења за програмирање.

4.2.3 Креирање нове датотеке за изворни код

Савет

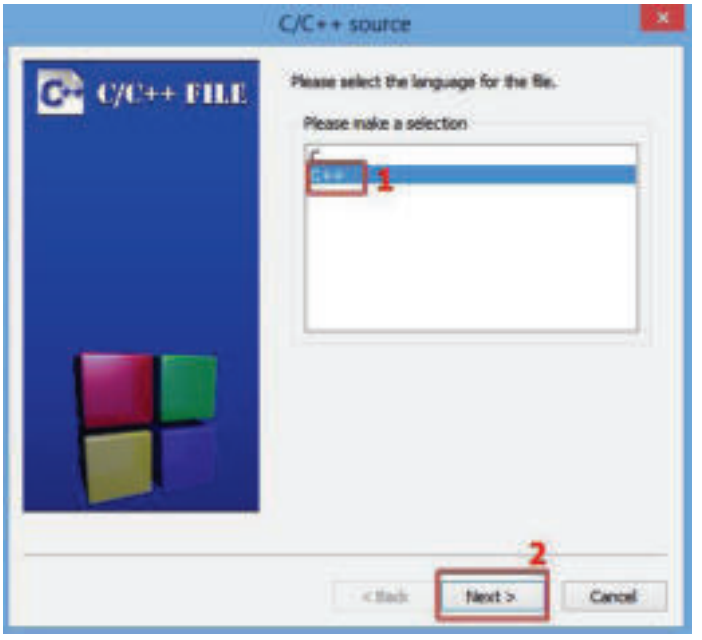
На радној површини свог рачунара креирај директоријум у коме ћеш сачувати све креиране пројекте.

Пре свега, креирајмо **нову датотеку** да бисмо написали изворни код. Нова датотека се креира помоћу менија **File** → **New** → **File**, при чему почиње поступак корак по корак. Први корак је избор типа датотеке:



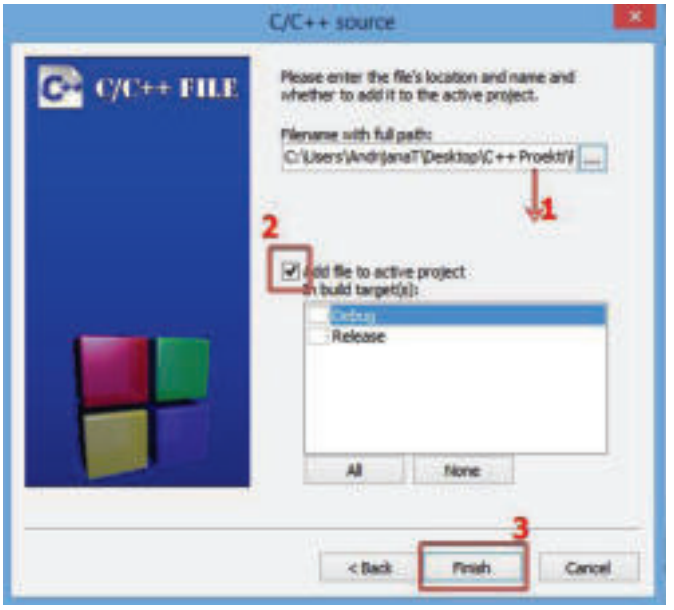
Слика 13: Избор C++ датотеке

Бирамо **C/C++Source** датотеку и кликнемо на **Go**. Затим, приступамо другом кораку, а то је избор програмског језика:



Слика 14: Избор програмског језика

Бирамо **C++** програмски језик и кликнемо на тастер **Next** да бисмо приступили следећем кораку поступка:



Слика 13: Избор C++ датотеке

У прозор кликнемо типку **Browse** и путем отвореног прозора додајемо име и локацију новокреиране датотеке. Кликом на тастер **Finish**, улазимо у едитор изворног кода (**Source Code**) и почињемо са писањем програма.

Запамти!

Интегрисано окружење за програмирање је софтверски пакет који садржи алатке које су потребне за креирање програма. Такве алатке су: едитор изворног кода, библиотеке кодова, компајлери и платформе за тестирање. Code::Blocks представља бесплатно интегрисано окружење за програмирање са отвореним кодом.

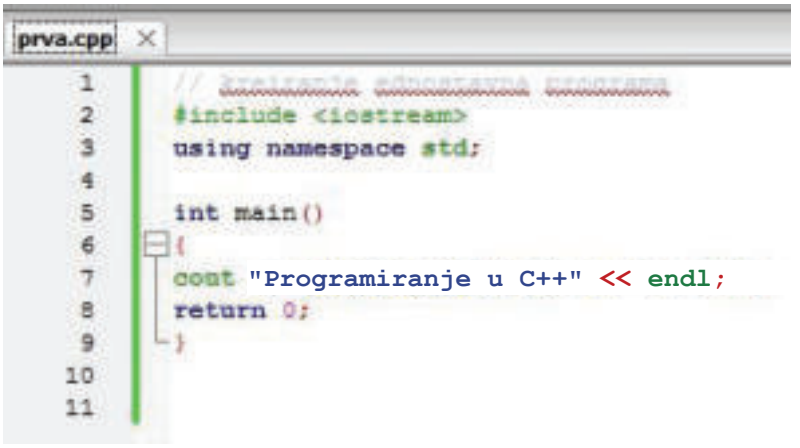
Питања:

1. Шта представља интегрисано окружење за програмирање?
2. Која је главна намена интегрисаног окружења за програмирање?
3. Које алатке укључује интегрисано окружење за програмирање?
4. У ком делу прозора се уписује изворни код?
5. Која је улога компајлера у радном окружењу за програмирање?
6. Која је улога дебагера?
7. Опиши радни прозор интегрисаног окружења за програмирање Code::Blocks?

4.3 Изглед готових пример – програмских кодова

Да бисмо почели са писањем изворног кода програма, потребно је да знамо **структуру** и **синтаксу** програмског језика. Како програмски језик представља мноштво правила, симбола и специјалних знакова који се употребљавају при креирању програма, постоје правила о **синтакси (граматици)** и **семантици (логици)** која морају да се поштују, у јединственом циљу стварања програма који ће се несметано и тачно извршавати.

Изворни код се пише у уредницима за изворни код, односно едиторима. На слици је дат пример за изворни код:



Слика 1: Пример изворног кода

На основу едитора изворног кода уочавамо да је свака линија означена бројем. Поред бројева се налази зелена линија која нам показује да је овај код преведен. Претходно, пре него што смо извршили превођење, та линија је била жута. Хајде да обележимо значење кода у свакој линији!

Број линија	Значење изворног кода
Л1	Знаком // почиње се писање коментара или се даје опис онога што следи
Л2	Наредба за укључивање библиотеке **iostream**
Л3	**Using namespace std** приказује да ће се користити стандардни елементи из библиотеке
Л4	Празна линија која се игнорише при извршењу
Л5	Главна функција **main**0
Л6	Отворена велика заграда која означава почетак и затворена велика заграда која означава крај извршења наредби главне функције **main**0
Л7	**Cout** је наредба за штампање на екрану << су оператори за штампање **Programiranje u C++** је текст за штампање на екрану **endl** је наредба за крај линије **„ ;“** означава з наредбе
Л8	**return 0;** је порука оперативном систему да је програм успешно завршен
Л9	Затварање велике заграде значи затварање главне функције **main**0

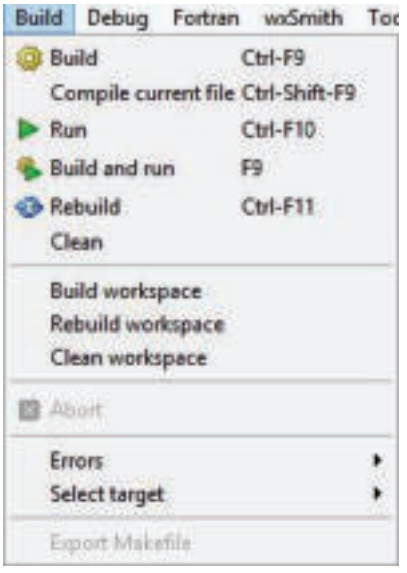
Слика 1: Пример изворног кода

Савет

После сваког исказа ставља се тачка запета (;) и то представља знак који нам указује да се исказ ту завршава, осим у случајевима који ће бити приказани при креирању програма.

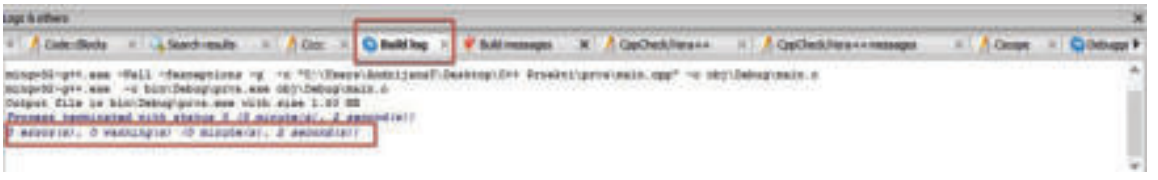
Код писања изворног кода, програмери воде рачуна о прегледности написаних наредби са циљем да се не ствара хаотичност и конфузија при анализи изворног кода. На пример, за преводиоца не значи ништа да ли ће се наредбе писати у једној линији или ће се организовати у посебне линије, јер знак тачка запета (;) јасно указује на то да је ту крај наредбе.

Превођење и повезивање у извршном коду одвија се као једна фаза, а одвија се уз помоћ менија **Build** → **Build** или комбинацијом тастера са тастатуре **Ctrl + F9:**



Слика 2: Наредба **Build**

Уколико се при овој фази појаве грешке, програмер приступа њиховом решавању и поново врши превођење и повезивање изворног кода. Порука да се појавила грешка приказује се у доњем делу радног прозора у посебном раму **Build log:**



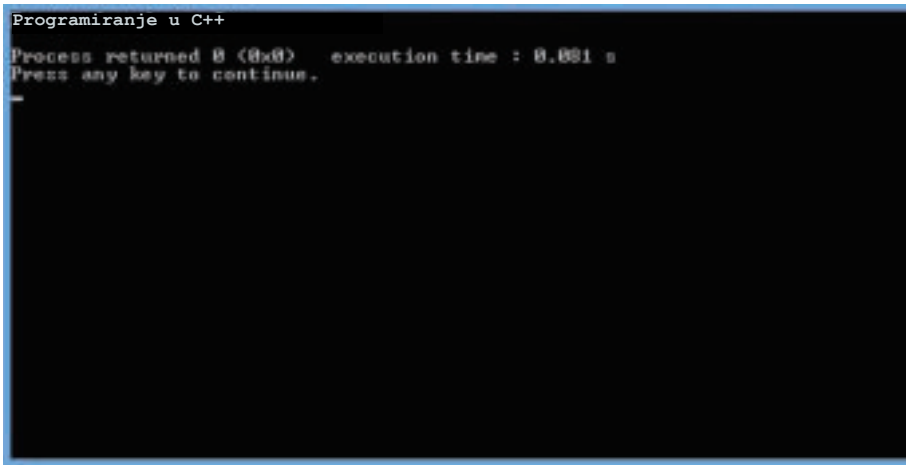
Слика 3: **Build log** рам

Програм ћемо завршити наредбом **Build** → **Run** или комбинацијом тастера на тастатури **Ctrl + F10.** Након овог процеса у креираном директоријуму постоје три датотеке, односно датотеке: изворна, објектна и извршна датотека.

Позивање наредби **Build** и наредбе **Run** може да се обави и помоћу траке са алаткама која се налази одмах испод мени-ленте:

Слика 4: Иконе алатки за Build и Run

Кликом на **Run**, почиње извршење програма који се приказује и завршава у посебном прозору, као што је приказано на слици:



Слика 5: Прозор извршне датотеке

Из овог прозора примећујемо да је порука исписана на почетку прозора, а да је испод ње сам рачунар генерисао поруку „**Press any key to continue…**“, што значи да ће се, ако кликнемо на неки тастер тастатуре, овај програм затворити.

Запамти!

При писању програма увек морамо да водимо рачуна о **синтакси**, **семантици** и **структури** програмског језика. Изворни код се пише у едитору, у којем је свака линија означена бројем. Изворни код се преводи и повезује и то се одвија као једна фаза. Уколико се појаве грешке, појављују се поруке **Build log.** Извршење програма се приказује у посебном прозору и као резултат се појављују три датотеке: изворна, објектна и извршна.

Питања

1. Шта је изворни код?
2. Којом наредбом се преводи изворна датотека?
3. Колико се датотека креира после превођења изворне датотеке? Које су то датотеке?
4. Како се извршава програм?

4.4 Извршавање готових примера програма

У претходној наставној јединици сазнали смо значење и функционалност основних и стандардних елемената у једноставном програму који једино извршава писање порука на екрану. Да бисмо упознали и друге елементе, креираћемо програм у коме ћемо имати могућност да поступамо интерактивно при извршењу програма, односно да уносимо податке или вредности са тастатуром. У едитору изворног кода, у интегрисаном окружењу за програмирање **Code::Block**, упишимо следеће низове наредби:

```
1   // ...
2   #include <iostream>
3   using namespace std;
4
5   int main()
6   {
7       int Бр. 1, Бр. 2, збир
8       cout << "Izračunajmo zbir dva broja!" << endl;
9       cout << "Unesi prvo broj" << endl;
10      cin >> br1;
11      cout << "Unesi drugi broj" << endl;
12      cin >> br2;
13      zbir=br1+br2;
14      cout << "Zbir dva broja je" << збир << endl;
15      return 0;
16  }
```

Слика 1: Изворни код програма за израчунавање збира

У првој линији програма написали смо **коментар**, односно **објашњење** о томе какав ћемо програм креирати и шта ће израчунавати тај програм. У нашем случају израчунаћемо збир два броја.

Сваки програм почиње уводом у коме дефинишемо које ћемо **библиотеке** укључити и које наредбе ћемо употребити:

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

Слика 2: Увод у програм

У овом случају је укључена библиотека iostream, а то значи да ће се примењивати стандардне наредбе за уношење података и приказ резултата на екрану. Следи главна функција у којој се извршавају наредбе које у суштини сачињавају програм:

```
1   // Сабирање два броја
2   #include <iostream>
3   using namespace std;
4
5   int main()
6   {
7       int br1, br2, zbir;
8       cout << "Izračunajmo zbir dva broja!" << endl;
9       cout << "Unesi prvi broj" << endl;
10      cin >> br1;
11      cout << "Unesi drugi broj" << endl;
12      cin >> br2;
13      zbir=br1+br2;
14      cout << "Zbir dva broja je" << endl;
15      return 0;
16  }
```

Слика 3: Главна функција Main

Пошто ћемо у овај програм прво уносити податке са тастатуре, најпре у главном програму дефинишемо варијабле, односно променљиве чије вредности зависе од уноса са тастатуре. Помоћу **int** означавамо да вредности које уносимо са тастатуре буду **цели бројеви** (**integer**). Варијаблама додајемо имена **br1**, **br2** и **zbir**.

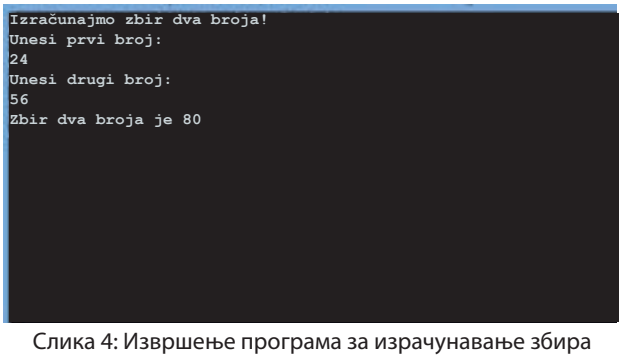
У линији (8) је исписана наредба **cout** којом ће се извршити писање поруке на екрану. Иза наредбе се записују оператори штампања (<<), а затим порука која треба да се прикаже на екрану записана са горњим наводницима. Наредбом **endl** означавамо крај линије, а знаком тачка запета (;) крај исказа, односно наредбе. Исту функционалност и значење има и линија број девет (9).

У линији број десет (10), исписана је наредба **cin,** која последња са операторима (>>) „<<“ означава наредбу за уношење података са тастатуре. Свакако да наредба завршава са знаком тачка запета (;). Исти поступак је поновљен у линијама под бројем једанаест (11) и дванаест (12).

У линији број тринаест (13), написана је формула за израчунавање збира од два унесена броја или у буквалном преводу, варијабла **zbir** ће се приказати у следећем реду од низа наредби.

Тако ће се, у линији четрнаест (14) са **cout** и операторима за штампање исписати реченица написана под наводницима, а иза ње вредност варијабле **zbir.** Ту се са **endl** завршава наредба. Главну функцију завршавамо са **return0**; чиме и указујемо процесору да је програм успешно завршен. Затварањем велике заграде означавамо крај наредбе у главној функцији **main**.

После исписивања изворног кода у едитору интегрисане средине за програмирање, приступамо компајлирању и дебаговању, тј. превођењу и повезивању изворне датотеке у извршној датотеци. Помоћу менија **Build** и наредбе **Build** започињемо са фазом превођења. Уколико се не појаве грешке у **Build log** настављамо са наредбом **Run** за извршење датотеке. У супротном, коригујемо грешке и понављамо процес. Извршна датотека се извршава у следећем прозору:



Слика 4: Извршење програма за израчунавање збира

На овом екрану у суштини видимо резултат нашег рада и то, да ли се процес одвијао према дефинисаним корацима.

Међутим, понекад не иде све у складу са планираним. Могуће је да се догоде грешке при превођењу за које програмер мора да пронађе решење и да се процес настави. Рачунарске грешке се називају **багови** (**Bugs**), па одатле долази и име процеса за изналажење и поправка грешака – **дебаговање** (**Debug**). Овај процес ћемо практично извежбати путем креирања изворног кода.

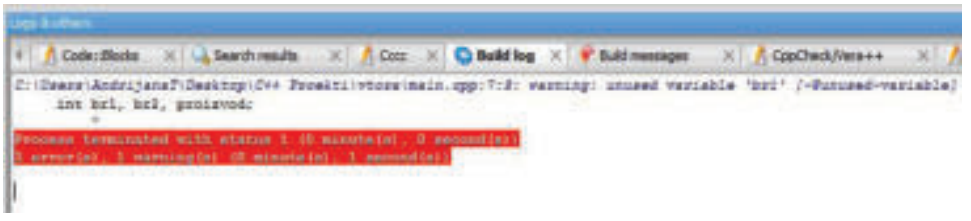
У едитору за изворни код у програму **Code::Block** напишимо следеће низове наредби:

```
1   // Сабирање два броја
2   #include <iostream>
3   using namespace std;
4
5   int main()
6   {
7       int br1, br2, proizvod;
8       cout << "Izračunajmo zbir dva broja!" << endl;
9       cout << "Unesi prvi broj:" << endl;
10      cin >> br1;
11      cout << "Unesi drugi broj" << endl;
12      cin >> br2
13      proizvod=br1*br2;
14      cout << "Zbir dva broja je" << endl;
15
16      system ("PAUSE");
17      return 0;
18  }
```

Слика 5: Изворни код за израчунавање збира

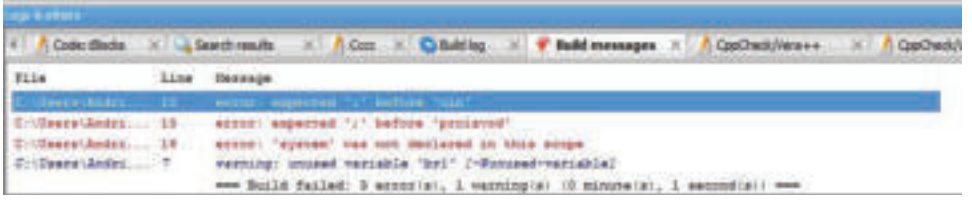
Покушајмо да уочимо грешке које смо направили при креирању програма. Хајде да их наведемо!

При процесу превођења изворне датотеке добићемо следећи резултат:



Слика 6: Порука о грешкама

У пољу **Build log** примећујемо да постоје грешке при превођењу програма. Кликом на опцију **Build message** видећемо грешке – једну, по једну:

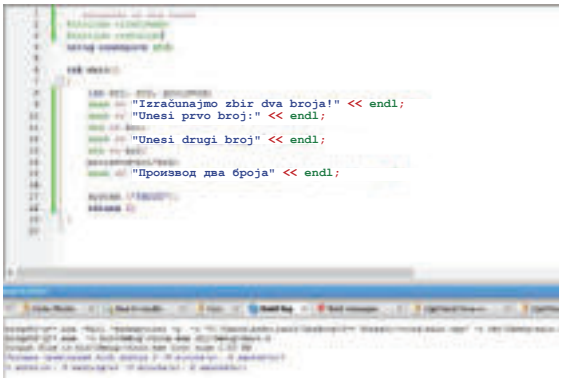


Слика 7: Издвојене грешке по линијама

У прозору уочавамо да постоје грешке:

- У линији број десет (10) недостаје нам знак „;“ пре позивања наредбе **cin**;
- У линији број тринаест (13) недостаје нам знак „;“ одмах пре дефинисања вредности „**Збир**“;
- На крају не препознаје наредбу **system (“Pause“)**, зато што она припада библиотеци **iostream**.

Приступамо решавању грешака, односно додајемо знак „;“ на местима на којима недостају и у уводном делу додајемо библиотеку **cstdlib** у циљу да се омогући функционисање наредбе **system (“Pause“)**.



Слика 8: Изворни код након извршених корекција

Сада, како више нема грешака приступамо извршавању, при чему ће се на посебном екрану приказати извршна датотека и резултат програма.

Запамти!

Претпроцесорске наредбе започињу знаком „#“. Путем **# include** најављујемо библиотеке чије ћемо наредбе употребљавати у изворном коду. **Main** је главна функција коју има сваки **C++** програм и она почиње са отвореном великом заградом **„{“** и завршава се затвореном заградом **„}“**. Између две заграде се дефинишу варијабле, пишу се искази као део корака у програму, знаци, симболи и сл. При превођењу и повезивању могу да се појаве грешке које се називају **Bugs,** а процес откривања и поправљања грешака – дебаговање (**Debug**).

Питања:

1. Како почиње сваки програм у C++ програмском језику?
2. Дефиниши увод у програм!
3. Како се називају линије које почињу са две косе црте (//)?
4. Наведи која је стандардна библиотека!
5. Како се зове главна функција? Шта она садржи?
6. Како се зову грешке у програму? Како се уклањају?
7. Шта значи return0;?
8. Којим знаком се одвајају наредбе једна од друге?

4.5 Основни елементи програмског језика C++

Запамти!

У нашој свакодневној комуникацији примењујемо неки језик, односно размењујемо информације, податке, идеје, ставове. Како ћемо се међусобно разумети? Који језик употребљавамо? Од чега је састављен језик којим говоримо? Како употребљавамо слова да бисмо разговарали?

Ова се питања намећу и при обављању једноставних радњи помоћу рачунара. И рачунари комуницирају: корисник – рачунар – корисник или пак рачунар – рачунар. Корисник комуницира помоћу рачунара применом улазних уређаја преко којих он уноси податке и информације, даје наредбе, употребљава програме и сл.

Ове активности се преводе у **бинарни језик** који је разумљив рачунару, и обратно, када рачунар треба да одговори на неку активност, тада се бинарни изрази преводе на језик разумљив за корисника и тако резултат видимо на екрану.

За комуникацију између рачунара можемо да кажемо да се дешава када су они мрежно повезани и могу да размењују податке, информације, документе, програме, заједничке уређаје и сл.

Програмски језици, као и природни језици, имају своју азбуку. **Азбука програмских језика се састоји од:**

- великих и малих слова абецеде,
- цифри од нуле (0) до девет (9),
- посебних знакова,
- знака за празно место,
- комбинација од два или три знака.

Адекватну комбинацију и примену ових елемената креирају искази, односно наредбе и као такве се називају **градбени елементи програмског језика.** Према томе, градбени елементи програмског језика су:

- резервисане речи,
- идентификатори,
- оператори,
- интерпункцијски знаци,
- коментари.

Сваки од наведених градбених елемената има своје место у структури и синтакси програмског језика. Хајде да објаснимо описно! Резервисане речи се још називају и кључне речи. Оне су унапред дефинисане и при превођењу изворне датотеке преводилац тачно зна шта значе. Такве су, на пример: int, delete, if, then, else, true, false, while и др.

Идентификаторе у програмском језику дефинише програмер. Они су комбинација знакова који могу да се комбинују уз поштовање следећих правила:

- име почиње словом или доњом цртом;
- мала и велика слова се разликују, на пример: „Zbir“ је различито од „zbir“;
- име може да садржи цифру али не и да почне са цифром;
- име не може да буде резервисана реч;
- име не може да садржи специјални знак као !, @, # и сл.

Оператори се у програмском језику употребљавају за означавање аритметичких, логичких и других операција које се изводе у програму. Интерпункцијски знаци се користе за одвајање елемената програмског језика, као што се, на пример, знаком „;“ означава крај исказа, односно наредбе, тј. раздвајају се искази.

Програмер при креирању програма може да напише **коментар** или **више коментара**. Писање коментара започиње са две косе црте „ // “. У ствари, коментарима се описује чему је намењен програм или се опет објашњавају појединачне линије са исказима, односно наредбама. Коментари се не извршавају, они само стоје као смерница или објашњење о значењу елемента, линије наредбе или програма у целини.

Различити програмски језици се међусобно разликују најчешће по синтакси, односно различитим резервним речима, правилима о њиховој примени и сл.

Будући да ми радимо са C++ програмским језиком, његови градбени елементи су:

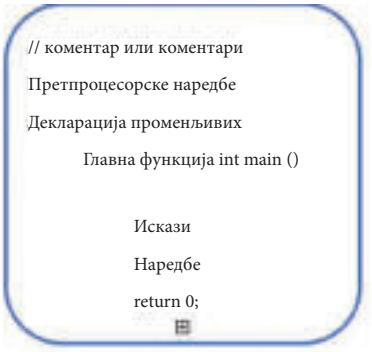
1. **Подаци** – програмским језиком могу да се обрађују различите врсте података и сви подаци имају следеће карактеристике: име, врста и вредност. Примери за различите врсте података: **char, int, bool, float, double** и сл.;
2. **Константе** представљају величине које не мењају своју вредност;
3. **Променљиве** су величине које мењају своју вредност;
4. **Декларације** у суштини дају опис променљиве, односно какву врсту података ће садржати променљива, на пример: **int x, float a,** и сл.
5. **Изрази служе** за вршење некаквог прорачуна у зависности од задатка који треба да се реши. Пример: sum + i и сл.;

6. **Искази** представљају наредбе које је ће бити извршене у програму, као на пример: **sum =sum+i, whilw, for, if – else, switch, break** и др.;

7. **Функције – у C++** главна функција је **main „( )“**. Искази, односно наредбе се пишу између две велике заграде на главној функцији, при чему сваки исказ завршава знаком „;“. Постоје функције које може да креира програмер, али се најчешће употребљавају готове функције:

8. **Модули –** при писању програмског изворног кода, сваки елемент се приказује у различитој боји. Преко боје утврђујемо да ли смо тачно написали реч у изразу.

Начин на који се комбинују градбени елементи програмског језика, њихов ток и редослед дефинишу **структуру програмског језика.** На пример, структура једног програма изгледа овако:



Слика 1: Структура од C++

Запамти!

Градбени елементи програмског језика су: резервисане речи, идентификатори, оператори, интерпункцијски знаци и коментари. Програмски језик C++ карактеришу следећи основни елементи: подаци, константе, променљиве, декларације, изрази, искази, функције и модули.

Питања

1. Наброј гредбене елементе програмског језика C++!
2. Наброј основне елементе програмског језика C++!
3. Шта су резервисане речи? Како се креирају?
4. Шта су идентификатори? Како се креирају?
5. Наведи неколико примера тачно креираних идентификатора према правилима о њиховом креирању!
6. Шта су функције? Коју функцију мора да има програм?
7. Шта су коментари? Наведи пример за коментар!

4.6 Искази

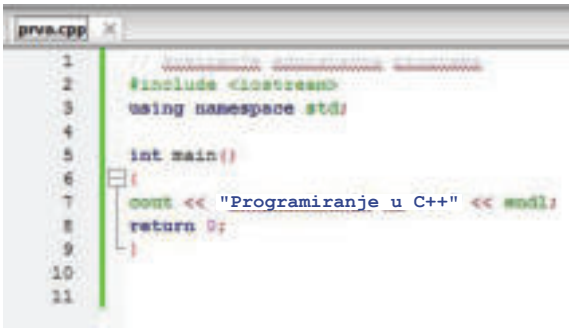
Искази представљају основни елемент сваког програмског језика. Без исказа рачунар не би знао које кораке треба извршити да би се дошло до коначног резултата. Према томе, **исказе можемо да дефинишемо као наредбе које казују рачунару шта да извршава, односно које радње треба да преузме да би дошао до решења.**

Искази се у програму одвајају знаком ; "Знак", нам казује да је ту крај наведеног исказа.

У програмским језицима постоји много исказа и сви имају различито значење, функцију и циљ. Код програмирања се најчешће срећемо са исказима за приказ на екрану и исказима за доделу вредности.

4.6.1 Исказ за приказ на екрану

При прегледању готових програма, први програм са којим смо се суочили био је једноставан програм који исписује текст или поруку на екрану.



Слика 1: Исказ за приказ на екрану

Након превођења и повезивања овог изворног фајла у извршном прозору приказује се порука "Programiranje u C++". Према примеру:

```
cout << "Programiranje u C++" << endl;
```

исказ представља исказ који омогућава приказ на екрану. Хајде да прегледамо елементе које садржи овај исказ:

Елемент	Значење
Cout	Израз за приказ на екрану
<<	Оператор за исписивање поруке на екрану
""	Све што је написано између наводника приказује се на екрану
Programiranje u C++	Порука између два наводника која ће се исписати на екрану
Endl	Завршавање линије
;	Завршавање исказа

Табела 1: Елементи исказа за приказ на екрану

Израз **cout** можемо да употребимо и на друге начине. Хајде да видимо шта ће приказати на екрану исказ **cout,** уколико је представљен на следеће начине

Исказ **cout“<<Broj a“;**	Приказ на екрану
Cout<<br;	На екрану ће се појавити порука која се налази између наводника
Cout<<br;	На екрану ће се појавити вредност променљиве **br**
Cout<<55;	На екрану ће се и штампати број 55
Cout <<3*br;	На екрану ће се приказати резултат прорачуна

Табела 2: Различити начини употребе исказа cout

Путем интегрисаног окружења за програмирање **Code..Block** креирајмо програм који ће нам на екрану приказати поруку, као у следећем примеру изворног кода:

```
1   // programa za prikaz na ekran
2   #include <iostream>
3   #include <cstdlib>
4   using namespace std;
5   int main()
6   {
7   cout << "Drugari!";
8   cout << "Danas ćemo na informatici";
9   cout << "učiti";
10  cout <<"programiranje u C++";
11  system ("pause");
12  return 0;
13  }
14
```

Слика 2: Изворни код за приказ текста на екрану

Резултат овог програма је приказан на следећој слици:

```
\Drugari!
Danas ćemo iz informatike učiti programiranje u C++"
Press any key to continue . . . _
```

Слика 3: Исписана порука на екрану

Са слике, уочавамо да је порука исписана у једном реду, сви делови су спојени и овако приказани не приказују прегледност и јасност. Да бисмо их издвојили у посебне линије, употребићемо наредбу **endl:**

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
cout << "Drugari!" << endl;
cout << "Danas ćemo na informatici" << endl;
cout << "učiti" << endl;
cout <<"programiranje u C++" << endl;
system ("pause");
return 0;
}
```

```
Drugari!
Danas ćemo iz informatike učiti
programiranje u C++ Press any key to continue ...
```

Слика 4: Примена наредбе endl

У изворном коду у примеру примећујемо да има више исказа за штампање и они ће се извршавати један по један, као што су наведени. Овако написани искази се називају секвенца или низ исказа.

4.6.2. Исказ за додељивање вредности

Исказ за додељивање вредности примењујемо при израчунавању збира на два унесена броја. Додела вредности променљиве може да буде обављена уносом уз помоћ тастатуре или, пак, вредношћу која се добија прорачуном.

Хајде да погледамо изворни код програма за прорачун збира од два броја:

```
using namespace std;

int main()
{
    int br1, br2, zbir;
    cout << "Izračunajmo zbir dva broja!" << endl;
    cout << "Unesi prvi broj:" << endl;
    cin >> br1;
    cout << "Unesi drugi broj:" << endl;
    cin >> br2;
    zbir=br1+br2;
    cout << "Zbir od dva broja je" << endl;
    return 0;
}
```

Слика 5: Додељивање вредности променљиве

На слици уочавамо да програм почиње коментаром као описом програма. Испод коментара следе претпроцесорска дешавања и декларисање да ће се употребљавати стандардни искази који се налазе у **namespace.** Затим, почиње главна функција main **( )**.

У главној функцији је приказана секвенца од исказа који ће се извршавати по редоследу. Поред декларисања променљиве и исказа на екрану, ту је примењен и исказ за додељивање вредности. Поступак за примену исказа **cin,** као исказа који додељује вредност променљиве уносом са тастатуре је приказан на следећој слици:

```
cout << "Unesi prvi broj:" << endl;
cin >> br1;
```

Слика 6: Додељивање вредности са тастатуре

Према томе, **cin** је исказ којим ћемо доделити вредност променљивој са тастатуре. Затим, следе оператори за унос ">>" и променљива која је дефинисана за први број, односно **br1**.

Променљива **br1** ће имати вредност коју је унео корисник тастатуре. Али, прво, пре него што се промени овај израз, мора да се дефинише променљива **br1**. На претходној слици променљиве **br1**, **br2** и **zbir** су дефинисане као **integer** вредности, односно као цели бројеви. Према томе, да би доделио вредност **br1** и **br2** са тастатуре, корисник ће унети целе бројеве. То представља **технику за унос података**. Променљивој **zbir** неће се додати вредност са тастатуре, већ ће њена вредност бити резултат израчунавања збира два броја, као што је приказано на следећој слици:

```
zbir=br1+br2;
cout << "Zbir od dva broja je" << endl;
```

Слика 7: Додељивање вредности променљивој као резултат израчунавања

Из приказа, утврђујемо да вредност променљиве **zbir** зависи од вредности **br1** и **br2** које ће корисник унети тастатуром. Збир два броја одређује вредност варијабле **zbir**. Резултат се на крају приказује на екрану исказом **cout**.

Запамти!

Искази су наредбе које одређују које радње треба да предузме рачунар да би се дошло до решења. Знак тачка запета (;) после сваког исказа означава крај исказа. Исказ **cout** се користи за приказ на екрану. Он је прослеђен операторима (<<), наводницима за исписивање поруке и завршава се са тачком запетом. Исказ за додељивање вредности променљиве са тастатуре, односно за унос података је **cin**. После исказа **cin** следе оператори (>>) за унос података променљиве и завршава тачком запетом. Ове врсте програма се називају интерактивни програми. Ипак, поред уноса са тастатуром, променљива може да добије вредност као резултат израчунавања.

Питања

1. Шта је исказ?
2. Који се исказ користи за приказ на екрану?
3. Шта омогућава наредба **endl**?
4. Који је исказ за додељивање вредности?
5. Како може да се одреди вредност тастатуром?

4.7 Израда програма

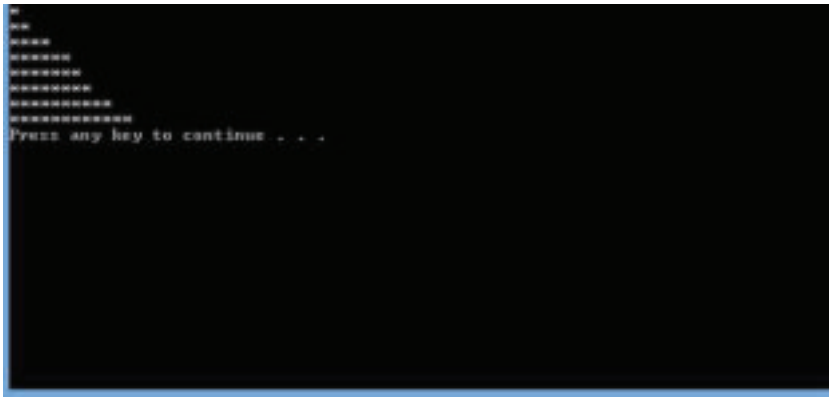
У следећим задацима ћемо направити програм у коме је више исказа за приказ **наплаћено**. Написани искази у програмском коду ће се извршавати један по један редоследно и овај начин извршавања се назива **техника редоследног извршавања**.

На пример, помоћу исказа за приказ на екран можемо да одштампамо **облик правоуглог троугла.** Ту ћемо користити технику редоследног извршавања. У том циљу, креирамо нови пројекат у **Code..Blocks** и почињемо писање програма:

```
1   //наплаћени искази за приказ на екран
2   #include <iostream>
3   #include <cstdlib>
4   using namespace std;
5
6   int main()
7   {
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      system ("pause");
17      return 0;
18  }
```

Слика 2: Приказ наплаћених исказа за приказ на екрану

Резултат следећег изворног кода је приказан на следећој слици:



Слика 2: Приказ наплаћених исказа за приказ на екрану

Да покушамо да на исти начин креирамо квадрат и да при приказу квадрата дијагонала буде уочљива. У том циљу, поред већ искоришћених звездица, употребићемо други знак. Изворни код овог програма би изгледао овако:

```
1  //naredbeni iskazi za prikaz na ekran - kvadrat
2  #include <iostream>
3  #include <cstdlib>
4  using namespace std;
5
6  int main()
7  {
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     system ("pause");
17     return 0;
18 }
19
```

Слика 3: Изворни код за приказ квадрата

Резултат извршења програма је приказан на посебном екрану, као што је представљено на слици:

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

Слика 4: Приказ извршеног фајла за приказ квадрата

По истом принципу, покушајмо да будемо креативни и исцртамо објекте помоћу исказа за приказ на екрану. На пример, можете да исцртате прво слово свог имена, објекат, срце и сл.

Запамти!

Писање исказа у програмском коду у коме се извршавају сви и редоследно назива се техника редоследног извршавања.

4.8 Аритметичке операције и изрази

При креирању програма у програмском језику C++ најчешће су употребљаване аритметичке операције и изрази. Аритметичке операције и изрази у суштини садрже математичке прорачуне. Према томе, **аритметички изрази у програмском језику C++ могу да се дефинишу као запис од две или више бројних вредности које су повезане математичким операторима**

Математички оператори су симболи који представљају специфичну акцију. Основни аритметички оператори који се користе у C++ су:

Аритметички оператор	Значење
+	Сабирање
-	Одузимање
*	Множење
/	Дељење
%	Остатак од дељења

Табела 1: Аритметичке операције са C++

Коментар!

У програмском језику C++ не постоји оператор за експонент. Међутим, постоји уграђена функција **pow**, која је дефинисана као библиотека **cmath**.

Да бисмо видели како се примењују аритметички оператори, креираћемо једноставан програм који врши једноставне математичке операције.

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

int main()
{
    int broj1, broj2, zbir, razlika, proizvod, kolicnik;
    cout << "Izračunati zbir, razliku, proizvod i količnik dva broja!" << endl;
    cout << "Unesite prvi broj" << endl;
    cin >> broj1;
    cout << "Unesite drugi broj" << endl;
    cin >> broj2;
    zbir=broj1+broj2;
    cout << "Zbir dva broja je: " << endl;
    razlika=broj1-broj2;
    cout << "Razlika dva broja je:" << endl;
    proizvod=broj1*broj2;
    cout << "Proizvod dva broja je:" << endl;
    kolicnik=broj1/broj2;
    cout << "Količnik dva broja je:" << endl;
    system ("pause");
    return 0;
}
```

Слика 1: Изворни код применом аритметичких операција

Уводни део програма, као и у другим примерима о програмима укључује библиотеке и команде које су дефинисане у **namespace**. **Главна функција main ()** је у суштини главни програм који садржи исказе који ће се извршавати да би се дошло до решења.

Као што примећујемо, у главном програму су дефинисане варијабле или променљиве којима ће бити додељене вредности. Вредности које ће им бити додељене су **integer**, односно цели бројеви. Варијабле **b1** и **b2** имаће вредност коју ће корисник доделити уносом са тастатуре, док ће варијабле **zbir, razlika, proizvod I količnik** добити вредности као резултат прорачуна. При прорачуну се употребљавају основне аритметичке операције. Резултат прорачуна видимо на извршном прозору:

```
Izračunati zbir, razliku, proizvod i količnik dva broja!
Unesite prvi broj
897
Unesite drugi broj
760
Zbir dva broja je:  1657
Razlika dva broja je:  137
Proizvod dva broja je:  681720
Količnik dva broja je:  1
```

Слика 2: Аритметичке операције са два броја

Запамти!

Аритметичке операције су записи са две или више бројних вредности које су повезане математичким операторима са циљем да се изврши прорачун. У програмском језику C++ се употребљавају следеће математичке операције: (+) за сабирање, (-) за одузимање, (*) за множење и (/) за дељење. Комбинацијом променљивих и оператора креирају се изрази.

4.9 Константе и променљиве

Константе и променљиве су елементи програмских језика. Употреба константи и променљивих при креирању програма изискује поштовање правила ради њихове правилне примене. Хајде да се упознамо са њима!

4.9.1 Константе

Константе представљају податке чија се вредност за време извршавања не мења. Константа C++ се у програмском језику креира на два начина. Један начин је коришћењем **# define** наредбе, а други је додавањем речи **const** пре врсте и имена константе. На пример:

```
1   // дефинисање константа на C++
2   #include <iostream>
3   #include <cstdlib>
4   using namespace std;
5   #define pi 3.141
6   int main()
7   {
8       const int n=5;
9       cout << n*pi << endl;
10      system ("pause");
11      return 0;
12  }
13
```

Слика 1: Дефинисање константи

Прва константа коју дефинишемо у програму је константа **pi** која има вредност 3.141. Њу дефинишемо коришћењем наредбе **#define** и иста је смештена пре него се појави почетак главне функције **main ( ).** Зато што је ова наредба смештена пре почетка функције **main ( )**, сматра се да се извршава у претпроцесорској фази и управо због тога на крају не садржи знак тачка запета, који у суштини означава крај исказа.

Друга константа је **n**. Она је дефинисана у главној функцији **main ( ).** Наиме, поред имена **n** и врсте **int** употребљава се и реч **const int n=5,** као што је приказано у програму. У неким случајевима, уколико се не наведе врста константе, подразумева се да припада врсти **int**. Резултат прорачуна је у ствари **5*3.141 = 15.705**, који је приказан на екрану преко исказа **cout.** Према томе, можемо да закључимо да се константе међусобно разликују према врсти података којима је исказана њихова вредност.

4.9.2 Променљиве

При изучавању математике, физике, хемије и других сличних наука, често се срећемо са много променљивих које имају неко значење. На пример: Обим се обележава са **L**, површина се обележава са **P**, радијус са **r**, дијагонала са **d**, температура са **T**, брзина са **a**, и сл. **Променљиве су подаци чија вредност може да се мења за време извршавања програма.** Оне се декларишу врстом и именом или, врстом именом и вредношћу. На пример:

```
int I, j;
int m=15;
float e=2,78;
int a=5, b=10
```

4.9.2.1 Врста променљиве

Свака променљива која се користи у програму мора да се најави, односно да се декларише, тј. да се дефинише њена врста. У следећој табели приказане су најчешће коришћене врсте промељивих са њиховим значењем:

Врсте променљивих	Значење
Char	Знаковна променљива која може да има вредност знака или целобројну вредност
Int	Целобројна променљива која може да има вредност тачно (true) ...-2, -1, 0, 1, 2...
Bool	Логичка променљива која може имати једну од две вредности тачно (true) и нетачно (false)
Float	Реална променљива са обичном прецизношћу
Double	Реална променљива са двоструком прецизношћу

Табела 1: Врсте променљивих

4.9.2.2 Додавање вредности променљивој

Поред врсте и имена, променљивој се може доделити вредност помоћу оператора „=“, као што је приказано у следећим примерима: **c=8, a=10, a=b*3, x=x+6, x=d=f=2** и сл. Овај поступак се назива **иницијализација** или **додавање почетне вредности**. То значи да ће на променљиву **c** бити додата вредност осам (8), променљива **a** ће добити вредност што је резултат прорачунавања производа **b*3**, променљива x ће добити резултат из прорачунавања збира **x+6,** променљива **f** ће добити вредност два (2), **d** ће добити вредност од **f,** док ће опет **x** имати вредност од **d**. На пример:

```
1  // inicijalizacija i deklaracija na promenliva
2  #include <iostream>
3  #include <cstdlib>
4  using namespace std;
5
6  int main()
7  {
8      //inicijalizacija
9      int a=6, b=18;
10     int suma;
11
12     //deklaracija
13     suma=a+b;
14     cout <<"Suma= " << suma <<endl;
15     system ("pause");
16     return 0;
17 }
18
```

Слика 2: Иницијализација и декларација променљивих

У овом програму у главној функцији, у **линији број девет (9)** дефинисане су и иницијализоване променљиве **a** и **b: int a=6, b=18.** У **линији број десет (10)** дефинисана је променљива **suma,** али није иницијализована. Она ће добити вредност после извршења прорачуна: **suma = a + b**, а резултат ће бити приказан на екрану.

Запамти!

Промрнљиве и константе при употреби у програм морају де се најаве, односно да се декларишу, а то значи да им се одреди врста и име. Променљиве добијају вредност помоћу оператора једнако (=). Додељивање почетне вредности променљивој назива се иницијализација.

Питања

1. Шта је константа, а шта променљива? Која је разлика између њих?
2. Шта је декларација? Да ли се може декларисати више променљивих једним исказом? Пример?
3. Шта је иницијализација? Наведи пример!

4.10 Искази (технике) за уношење података у програм

Да би разумели суштину технике за унос података у програм, креираћемо неколико програма.

Задатак

Креирати програм за израчунавање квадрата унесеног броја!

```
1  // proračun kvadrata za uneseni broj
2  #include <iostream>
3  #include <cstdlib>
4  using namespace std;
5
6  int main()
7  {
8      int broj;
9
10     cout << "*************************************************" << endl;
11     cout << „izračunati kvadrat unesenog broja“ << endl;
12     cout << "*************************************************" << endl;
13     cout << "unesi jedan broj" << endl;
14     cin >> broj;
15     cout << "Kvadrat broja" << broj << "e:" << broj*broj << endl;
16     system ("pause");
17     return 0;
18 }
19
```

Слика 1: Прорачун квадрата за унесени број

При креирању програма за прорачун квадрата унесеног броја, поред коментара и претпроцесорских наредби које се налазе у уводу програма, у главној функцији употребљавамо исказе за приказ на екрану и исказе за унос података са тастатуре. У ствари, ту употребљавамо технику за унос података. Променљива за коју треба да се унесе податак са тастатуре, прво се дефинише које је врсте, а затим, адекватно се примењује. Резултат извршавања видимо на посебном екрану:

```
*************************************************
  Izračunati kvadrat unesenog broja
*************************************************
Unesi jedan broj
5
Kvadrat broja 5 je:  25
Press any key to continue . . .
```

Слика 2: Резултат прорачуна квадрата броја

Задатак

Израчунати површину и обим круга!

Овде је реч о готовим математичким формулама које је потребно да употребимо при прорачуну. На пример: плоштина круга се израчунава формулом P = r * r * pi, док се обим израчунава следећом формулом L = 2 * r * pi.

```
// proračun površine i obima kruga
#include <iostream>
#include <cstdlib>
using namespace std;

#define pi 3.141

int main()
{
    float radius, P, L;

    cout << "**************************************************" << endl;
    cout << "Izračunati površinu i obim kružnice" << endl;
    cout << "**************************************************" << endl;
    cout << "Unesi radijus kružnice" << endl;
    cin >> radius;
    P=radius*radius*pi;
    cout << "Površina kružnice iznosi " << P << endl;
    L=2*radius*pi;
    cout << "Obim kružnice iznosi" << L << endl;
    system ("pause");
    return 0;
}
```

Слика 3: Израчунавање површине и обима круга

Полазећи од готових математичких формула, **pi** има константну вредност **3.141** и зато је у претпроцесорском делу дефинишемо као константу користећи наредбу **#define**. Променљиве у главној функцији дефинишемо као **float,** односно реалне бројеве који садрже цео и децимални број, јер **pi** има децималну вредност и очекује се да је резултат прорачуна децимални број. Затим се примењују искази за приказ на екрану и исказ за додељивање вредности које се редоследно извршавају и дају коначни резултат у одвојеном прозору:

```
**************************************************
 Izračunati površinu i obim kružnice
**************************************************
Unesi radijus kružnice:
6
Površina kružnice iznosi  113.076
Obim kružnice iznosi 37.692
Press any key to continue . . .
```

Слика 4: Резултат прорачуна површине и обима кружнице

Задатак

Израчунати површину троугла користећи математичку формулу P=(a*h)/2

```cpp
// proračun površine trougla
#include <iostream>
#include <cstdlib>
using namespace std;

#define pi 3.141

int main()
{
    float a, h, P;
    cout << "*************************************************" << endl;
    cout << "Izračunati površinu trougla" << endl;
    cout << "*************************************************" << endl;
    cout << "Unesi veličinu strane a" << endl;
    cin >> a;
    cout << "Unesi veličinu trougla" << endl;
    cin >> h;
    P=a*h/2;
    cout << "Površina trougla iznosi: " << P << endl;
    system ("pause");
    return 0;
}
```

Слика 5: Прорачун површине троугла

Да бисмо креирали програм и добили тачан резултат, код прорачуна примењујемо математичку формулу за израчунавање површине троугла. Дефинишемо врсту променљиве, помоћу исказа за приказ на екрану иштампали смо текст као увод у задатак, уносимо вредности променљивих тастатуром и вршимо прорачун применом формуле. Добијамо следећи резултат при извршавању датотеке:

```
*************************************************
 Izračunati površinu trougla
*************************************************
Unesi veličinu strane a:
7
Unesi veličinu trougla:
4
Površina trougla iznosi: 14
Press any key to continue . . .
```

Слика 6: Резултат прорачуна површине троугла

Задатак

Креирај програм који ће при уносу броја са тастатуре приказати његовог претходника и следбеника.

4.11 Упоредни изрази

У нашој свакодневици често се дешавају ситуације када је потребно да донесемо одлуке, и у зависности од тих одлука знамо којим путем да кренемо даље. На пример, после завршавања обавезне школе, налазимо се на једној раскрсници на којој треба да одлучимо да ли ћемо образовање да наставимо на факултету или ту завршавамо. Ту имамо две могућности. Прва могућност је да ће нам се, ако наставимо образовање на универзитету, отворити широке могућности да проширимо своје знање, а самим тим ћемо и бити конкурентни при изналажењу добро плаћеног и стручног посла. Друга могућност је да одлучимо да не наставимо са образовањем и останемо са до сада стеченим знањем, чиме се и могућност за конкурентност умањује. Ипак, ово су тешке животне одлуке, али су добар пример за увод у оно о чему ћемо говорити у овој наставној јединици.

Коментар!

Да ли сте се некада нашли у ситуацији када је потребно донети одлуку? Како сте одлучили? Шта вас је натерало да тако одлучите? Опишите ситуацију!

Сличне ствари се дешавају и код рачунара. На пример: Када неки програм одлучује да ли је програм задовољен и ако је задовољен, која ће се радња предузети, и обратно, ако није задовољен услов, шта даље?!

Овакви изрази се називају **упоредни изрази** и њих корисник бира када има алтернативу. То значи да ће се предузети једна радња уколико је тврдња тачна, а друга радња уколико није тачна. На пример, када почињемо са креирањем програма, истичемо да се **искази (наредбе)** извршавају редоследно један, за другим. Код примене оваквих упоредних израза губи се редослед програма, зато што његово извршење зависи од тачности дефинисане тврдње.

Ови упоредни изрази у програмском језику C++ називају се **логички изрази** или **Boolean изрази.** Код креирања оваквих израза користе се оператори за поређење:

Оператор у C++	Значење	Математички оператор
==	Једнако на	=
!=	Неједнако на	≠
>	Веће од	>
<	Мање од	<
>=	Веће или једнако на...	≥
<=	Мање или једнако на...	≤

4.11.1 Структура за избор између две могућности

При креирању програма са упоредним изразима, најчешће је употребљавана **структура за избор између две могућности.** У зависности од вредности израза, у овој структури се врши избор између две могућности. Израз може имати вредност или **тачно** или **нетачно,** то значи да, ако је услов израза тачан, тада ће се ова наредба извршити, а ако није тачан, тада ће се извршити друга наредба или ће се програм ту завршити.

Према горенаведеном, можемо констатовати да постоје две врсте гранања израза, у зависности од тога да ли је услов задовољен или није: **једнократно** и **двократно.**

Једнократно грањање се примењује ако је услов задатка испуњен, тада ће се извршити нека наредба, а уколико услов није испуњен, тада ће се програм завршити, односно **„ако услов, онда наредба".** У том циљу се примењује исказ **if..then.**

На пример: када корисник унесе године, програм ће показати да ли је малолетан и[illegible]

proračun površine trougla

```
// If...else naredba
#include <iostream>
#include <cstdlib>
using namespace std;

int main()
{
    int Godine;
    cout << "Koliko imate godina?" << endl;
    cin >> Godine;

    if (Godine <=18)
        cout << " Maloletni ste!" << endl;
    else
        cout << "Punoletni ste!" << endl;

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

Слика 1: If…Then исказ

Важно!

После исказа if не ставља знак тачка запета зато што ће се у овом исказу извршавати наредбе које следе. Додавањем знака тачка запета, означићемо крај исказа.

Из главног програма можемо да уочимо да је дефинисана променљива **године** из врсте целобројна вредност тј. **int.** После приказа на екрану **„Koliko godina imate?"**, корисник уноси податак са тастатуре и кликне **Enter**. Тада се врши провера да ли је унесени податак мањи или једнак осамнаест (18), и, ако је задовољен услов, тада ће се на екрану приказати порука **„Maloletni**

ste!“. Уколико услов није задовољен, односно ако се унесе податак већи од осамнаест (18), тада се неће извршити никаква радња.

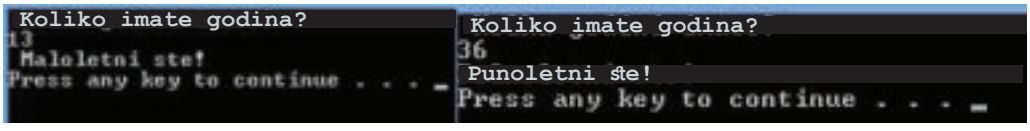
Двократно грањање, у суштини, значи да се тада изврши дефинисана наредба ако је неки услов задовољен, и изврши се друга наредба ако није задовољен, односно **„ако услов, тада наредба1, иначе наредба2“**. У том циљу се примењује исказ **if…else.**

На пример, задатак који смо претходно урадили бисмо наставили са дефинисањем друге наредбе.

```
1   // if...then наредба
2   #include <iostream>
3   #include <cstdlib>
4   using namespace std;
5
6   int main()
7   {
8       int godini;
9       cout << "Koliko imate godina?" << endl;
10      cin >> godini;
11
12      if (godini <=18)
13          cout << " Maloletni ste!" << endl;
14
15      system ("pause");
16      return 0;
17  }
18
```

Слика 2: if...Else наредба

Према томе, при уносу податка „**godine**“, програм проверава да ли је унесени податак мањи или једнак са осамнаест (18). Ако је услов испуњен, односно унесени податак се налази у рангу између нуле (**0**) и осамнаест (18), тада ће се на екрану исписати порука: **„Maloletni ste!“.** Уколико услов није задовољен, односно унесени податак је већи од броја осамнаест (18), тада се изврши радња и на екрану се испише **„Punoletni ste!“**



Слика 3: Резултат извршеног фајла if…else

При употреби исказа **if…else**, може се десити да наредба садржи више исказа. Овај начин писања наредби се назива блок исказа и они се стављају у велике заграде. На пример, да додамо блок исказ у програм у коме смо радили:

```
//if...else naredba sa blok iskazima
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int godine;
    cout << "Koliko imate godina?" << endl;
    cin >> godine;

    if (godine <=18)
      {
          cout << "Maloletni ste!" << endl;
          cout << "Vi ste srednjoškolac!" << endl;
      }
    else
      {
          cout << "Punoletni ste!" << endl;
          cout << "Vi ste završili obrazovanje!" << endl;
      }
    system ("pause");
    return 0;
}
```

Слика 4: Блок искази у if...else

То значи да ће се после уношења податка о променљивој у **„godine“** направити провера да ли је унесени податак мањи или једнак са осамнаест (18). Ако је услов задовољен, тада ће се на екрану исписати две поруке једна испод друге, а уколико није задовољен услов, поново ће се на екрану испи-сати две друге наредбе, у зависности од испуњености услова. Блок исказа се увек пише између две велике заграде.

4.11.2 Израда програма са структуром за избор између две могућности.

Хајде да креирамо неколико програма у којима ће се поставити услов. Уносом података проверићемо да ли је услов задовољен или није и у складу са одговором ће се извршавати наредбе.

Задатак

Израчунати површину троугла користећи математичку формулу: P=(a*h)/2

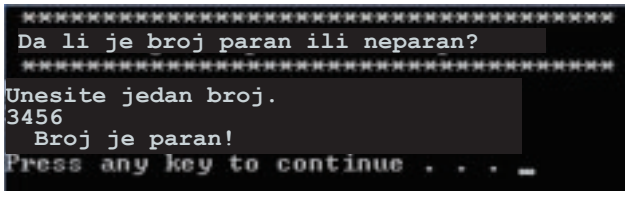
```
   paran ili neparan broj
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int broj;
    cout << " ************************************ " << endl;
    cout << "Da li je broj paran ili neparan?" << endl;
    cout << " ************************************ " << endl;
    cout << "Unesi jedan broj?" << endl;
    cin >> broj;
    if (broj%2==0)
          cout << "Broj je paran!" << endl;

    else
          cout << "Broj je neparan!" << endl;

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

Слика 5: Провера да ли је број паран или непаран

У овом програму се после уношења броја са тастатуре проверава да ли је број паран или непаран. Према математичком правилу: **„сваки број који је подељен са бројем два (2) има резултат цео број и без остатка, тада је паран број“,** утврђујемо услов. Ту примењујемо **оператора „%“ (mod)** који приказује остатак дељења. Унесени број подељен са бројем два има остатак једнак нули, тада ће се на екрану исписати порука **„Broj je paran!“** Ако услов није задовољен, исписаће се порука **„Broj je neparan!“** Резултат извршавања програма је приказан на следећем прозору:



Слика 6: Резултат програма за проверу броја да ли је паран или непаран

Задатак

Креирати програм у који ћемо унети два броја. Притом ћемо створити услов који ће проверавати који је број од унесених већи.

```
1   // Koji je broj veći
2   #include <iostream>
3   #include <cstdlib>
4   using namespace std;
5   int main()
6   {
7       int broj1, broj2;
8       cout << " ************************************ " << endl;
9       cout << "Koji je broj veći?" << endl;
10      cout << " ************************************ " << endl;
11      cout << "Unesite prvi broj:" << endl;
12      cin >> broj1;
13      cout << "Unesite drugi broj:" << endl;
14      cin >> broj2;
15      if (broj1>broj2)
16            cout << "Prvi broj je veći od drugog!" << endl;
17      else
18            cout << "Drugi broj je veći od prvog!" << endl;
19      system ("pause");
20      return 0;
21  }
```

Слика 7: Програм који проверава који је број већи

После уноса оба броја у програм употребљавамо исказ **if ... else** у коме креирамо услов за проверу да ли је први број већи од другог. Ако је испуњен услов, исписује се порука на екрану да је први број већи од другог, а уколико није испуњен, исписана је порука да је други број већи од првог. Резултат провере је следећи:

```
**********************************
Koji je broj veći?
**********************************
Unesite prvi broj:
345
Unesite drugi broj:
123
 Prvi broj je veći od drugog!
Press any key to continue . . .
```

Слика 8: Резултат програма који је број већи

Задатак

Креирати програм који проверава да ли је број позитиван или негативан!

```
1   // Pozitivan, negativan ili jednak 0
2   #include <iostream>
3   #include <cstdlib>
4   using namespace std;
5   int main()
6   {
7       int broj;
8       cout << " ************************************ " << endl;
9       cout << "Da li je broj pozitivan ili negativan?" << endl;
10      cout << " ************************************ " << endl;
11      cout << "Unesite jedan broj?" << endl;
12      cin >> broj;
13      if (broj<0)
14          cout << "Broj je negativan!" << endl;
15      else
16          if (broj==0)
17              cout << "Broj nije niti pozitivan niti negativan!" << endl;
18          else
19              cout << "Broj je pozitivan?" << endl;
20
21      system ("pause");
22      return 0;
23  }
```

Слика 9: Провера са if...else да ли је број позитиван, негативан или једнак 0

На основу овог програма можемо да приметимо да после уноса броја са тастатуре креирамо услов, ако је унесени број мањи од нуле, исписује се порука да је то негативан број. У супротном, ако услов није задовољен, треба приступити провери услова да ли је унесени број једнак нули и, ако је овај услов задовољен, на екрану треба буде исписана порука да број није ни позитиван ни негативан. У супротном је исписана порука да је унесени број позитиван. Примећујемо да смо овде **два пута** применили исказе **if ... else,** који се извршавају у зависности од испуњења претходног услова. Ова техника се назива **техника угњежђивања исказа.** Резултат прорачуна видимо на екрану:

```
**********************************
Da li je broj pozitivan ili negativan?
**********************************
Unesite jedan broj.
2346
Broj je pozitivan!
Press any key to continue . . .
```

Слика 10: Резултат провере да ли је број позитиван, негативан

Запамти!

Упоредни изрази се користе при избору између две могућности. Они се називају логички или Boolean изрази и при њиховом креирању користе се оператори за упоредбу. Структура програма која садржи избор две могућности може да буде једнократно разграната и двократно разграната. У првом случају користи се израз if, док се у другом случају користи израз if...else. Ови искази у својој структури могу да садрже више исказа и они се записују између великих заграда.

Питања

1. Дефиниши упоредне изразе!
2. Наведи операторе који се употребљавају за поређење!
3. Какво може да буде гранање у програму у коме може да се направи избор између две могућности? Која је разлика између њих?
4. Напиши синтаксе од if и if... else!

4.12 Структура понављања циклуса до испуњавања услова

До сада смо креирали много програма који имају различиту структуру, различите исказе, различите циљеве и решења. При креирању програма у програмском језику C++, искази се извршавају један по један редоследно. Овај начин извршавања исказа назива се **физички редослед** извршавања.

Када смо креирали програме уз примену **if... else,** редослед извршавања исказа се мењао у зависности од испуњења услова који је дефинисан. Међутим, често је потребно да се неки искази понављају, односно да се извршавају више пута. Овакво понављање исказа представља **циклус**, а ове структуре се називају **цикличне структуре** или **структуре за понављање.** Редослед извршавања исказа у циклусу назива се **логички редослед**, јер он зависи од испуњености услова.

Према томе, једна или више инструкција извршава циклус неколико пута, све док се не испуни један или група услова. Најчешће коришћени искази који извршавају циклусе су**: while, do – while i for** циклус.

При примени структуре за понављање могуће су две ситуације и то:

- унапред се зна колико ће се пута циклус поновити;
- број понављања зависи од неког услова и тај број није унапред познат.

Услов може да се нађе на почетку циклуса или, пак, на крају циклуса.

Циклус **while** је најједноставнији циклус који се завршава испуњавањем датог услова, а то значи да **while** инструкција тестира услов при којем се извршавају низови наредби све до његовог испуњавања. Његова синтакса је следећа:

```
While (услов)
{
наредба 1;
наредба 2;
наредба 3;...
}
```

На основу приказане синтаксе, **while** инструкција је контролна структура која се назива циклус или **loop**. Инструкција или наредба која треба да се изврши сваки пут при завршетку циклуса назива се **тело циклуса**. Притом, пре него што се почне извршавање циклуса, прво проверава да ли је услов задовољен. Ако је услов задовољен, започиње се са извршавањем исказа, а уколико није задовољен, уопште се не започиње извршавање исказа.

На пример:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int i=1;
    while (i<=10)
    {
        cout << i <<"\n";
        i++;
    }
    system ("pause");
    return 0;
}
```

Слика 1: Програм за исписивање назива бројева од један (1) до десет (10)

На основу програма, примећујемо да је главна функција променљива **„I“** дефинисана врстом „цео број“ и иницирана је вредношћу један (1). Применом структуре за понављање исписују се бројеви у низу од један (1) до десет (10), све до испуњавања услова задатог у циклусу **while**. На екрану ће се приказати следеће:

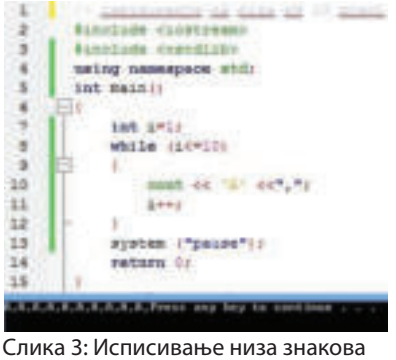
```
1
2
3
4
5
6
7
8
9
10
Press any key to continue . . .
```

Слика 2: Исписивање низа бројева од један (1) до десет (10)

Напомена!

У овом програму низ од бројева је вертикално постављен. Да би се представили бројеви хоризонтално, бришемо израз „\n“ и под наводницима додајемо запету.

Сличан овакав програм који на екрану исписује знаке до одређеног броја, приказан је у следећем програму:



Слика 3: Исписивање низа знакова

Циклуси који користе **while** инструкцију могу да буду:

- циклуси контролисани бројачем,
- циклуси контролисани ситуацијом.

Циклуси контролисани бројачем користе променљиву која контролише циклус. У том циљу се иницира контролна променљива бројача и при сваком кругу извршавања, бројач се повећава. На пример:

```
ciklus kontrolisan brojacem
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int i=1;
    while (i<5)
    {
        cout << "Programiranje u C++" << endl;
        i++;
    }
    system ("pause");
    return 0;
}
```

```
Programiranje u C++
Programiranje u C++
Programiranje u C++
Programiranje u C++
Programiranje u C++
Press any key to continue . . .
```

Слика 4: Циклус контролисан бројачем

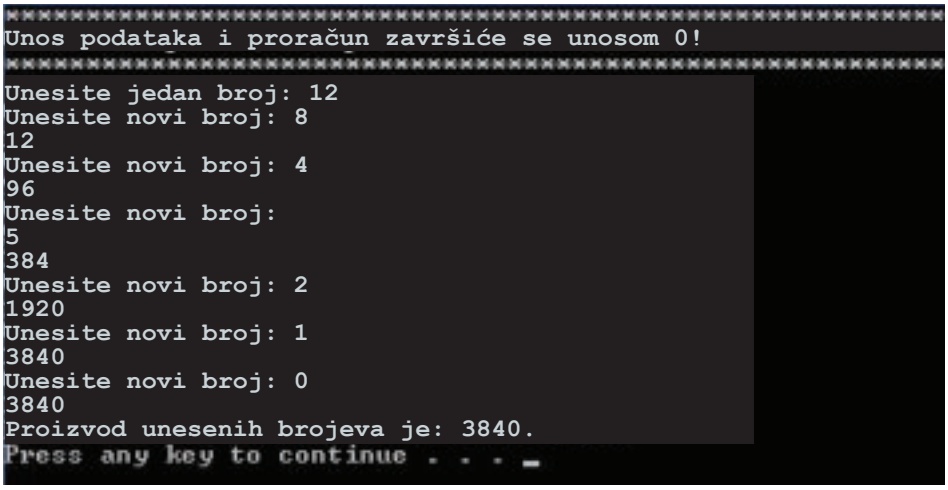
Према задатку, реченица **„Programiranje u C++“** ће се исписивати све док се не задовољи услов: **да се напише пет пута.** Сваког круга бројач **I** се повећава за један (1).

Код **циклуса који су контролисани ситуацијом**, услов за прекид зависи од неке ситуације која се извршава док се тело циклуса извршава. На пример: Израчунати производ унесених бројева, а унос ће се прекинути уношењем нуле (0) на тастатури:

```
ciklus kontrolisan situacijom
#include <iostream>
#include <cstdlib>
using namespace std;
main()
{
    int broj, p=1, br=1;
    cout << "*************************************************************" << endl;
    cout << "Unos podataka i proračun završiće se unosom 0!" << endl;
    cout << "*************************************************************" << endl;
    cout <<"Unesite jedan broj:" ;
    cin >> broj;
    while (broj!=0)
    {
        p=p*broj;
        br++;
        cout << "Unesite novi broj:" ;
        cin >> broj;
        cout << p << endl;
    }
    if (br==0)
    p=0;
    cout << "Proizvod unesenih brojeva je:" << p << "." << endl;
    system ("pause");
    return 0;
}
```

Слика 5: Циклус контролисан ситуацијом

Овај програм циклуса **while** ће се извршавати све док се не унесе нула (0) са тастатуре. А то значи да унос нуле (0) представља ситуацију која ће прекинути, односно завршити циклус. Зато се каже да су ови циклуси контролисани ситуацијом. Резултат извршавања кода је приказан на следећој слици:



Слика 6: Резултат извршавања циклуса

4.12.1 Израда једноставних програма са структуром за понављање до испуњења услова

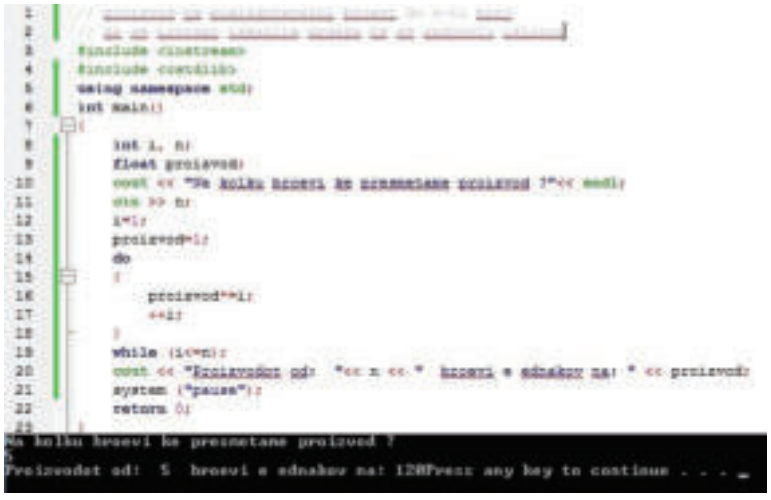
Код структура за понављање до задовољења услова, услов може бити дефинисан на **почетку циклуса** или на **крају циклуса.** Код првог се најпре проверава да ли је услов задовољен да би се извршили низови исказа, а код другог се искази извршавају све док се не задовољи услов. На пример, израчунаћемо производ узастопних бројева до **n**-тог броја, на два начина:

Први начин: Услов се налази на почетку исказа за понављање циклуса:

```
uzastopnih brojeva do n-tog broja
Uslov postavljen na početku
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int i, n;
    float proizvod;
    cout << "Za koliko brojeva ćemo proračunati proizvod?" << endl;
    cin >> n;
    i=1;
    proizvod=1;
    while (i<=n)
    {
        proizvod*=i;
        ++i;
    }
    cout << "Proizvod od: " << n << "broja je jednako:" << Proizvod;
    system ("pause");
    return 0;
}
```

Слика 7: Услов у циклусу је дефинисан на почетку

Други начин: Извршиће се низови из исказа док се не задовољи услов.



Слика 8: Програм у коме се извршавају искази до задовољавања услова

Задатак

Примењујући структуре за понављање у циклусу док се не задовољи услов, креирати следеће програме:

1. Исписати на екрану парне бројеве до **n**-тог броја!
2. Израчунати збир непарних бројева до **n**-тог броја!
3. Израчунати производ парних бројева до **n**-тог броја!

Запамти!

Редослед извршавања исказа може да буде физички и логички. Понављање исказа неколико пута представља циклус, а структура цикличне структуре или структуре за понављање. Најчешће коришћени искази за циклусе су: while, do-while и for. Код употребе while прво се проверава услов, а затим се извршавају искази. Код do-while искази се извршавају све док се не постигне услов.

Питања

1. Шта су структуре за понављање? Дефиниши!
2. Који су најчешће коришћени искази за циклусе?
3. Наведи синтаксе од while и do-while!
4. Како се контролишу циклуси код промене while и do-while!
5. Да ли се зна колико пута ће се поновити циклус? Објасни!

ПОНОВИМО! УВЕЖБАЈМО!

Задатак 1

Напиши програм помоћу кога ћеш добити следећи приказ на екрану:
Pvc čas: INFORMATIKA
Programiranje u C++

Задатак 2

Које су вредности променљивих a и b, након извршавања следећих наредби?

```
a = 15
b = 23
a = a + b
b = a - b
a = a - b
```

Задатак 3

Шта ће се приказати на екрану после извршавања следећег кода:

```
int a, b, c;
cin >> a;
cin >> b;
cin >> c;
cout << a << "," << b << "," << c << "." << endl;
```

Ако корисник унесе следеће вредности: пет (5), шест (6) и седам (7) једну за другом?

Задатак 4:

Шта ће се исписати на екрану по извршавању следећег низа наредби?

```
int a = 8;
int b = 10;
cout << (a+b)/2
```

Задатак 5:

Колико пута ће се извршити следећи циклус?

```
int i = 2
do
{
cout << i <<  " " ;
i ++;
}
While (i<=10)
```

Задатак 6:

Напиши програм за израчунавање површине правоугаоника тако да излаз на екрану прикаже димензије страна и резултат прорачуна површине. Формула за израчунавање правоугаоника је: p = a * b

Задатак 7:

Напиши програм који ће израчунати аритметичку средину два броја. Како би се извршио прорачун аритметичке средине за осам бројева?

Задатак 8:

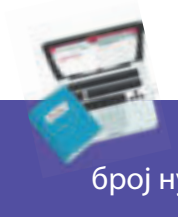
Креирати програм који проверава да ли је унесени број паран или непаран.

Задатак 9:

Креирати програм који израчунава обим квадрата, уколико је унесена вредност стране „a“ позитивна, тако да у буде исписана порука „Прорачун се не може извршити! Унесите позитиван број!“.

Задатак 10:

Применом циклуса do – while, израчунати збир парних бројева до n-тог броја.



Задатак 11:

Израчунати збир бројева који се циклично уносе са тастатуре, све док се не унесе број нула (0).

Онлајн живот

Увод у веб-дневнике (блогове)
Појам о блогу и његовој примени
Израда блога
Коментарисање садржина блога
ПОНОВИМО И УВЕЖБАЈМО!

5. Увод у веб-дневнике (блогове)

Поновимо!

Шта су програми? Наведи дефиниције за програмирање! Како се израђује програм? Које програмске језике познајеш?

Модерно друштво и тренд масовне глобализације намећу потребу примене **интернет технологије** у процесу образовања, али и у свакодневици уопште. Применом интернет технологије корисницима се омогућава употреба различитих сервиса, као што су: комуникација, размена порука, подела информација, куповина путем интернета, преузимање слика, музике, видео-садржаја и сл.

Према томе, **интернет** је најпознатија светска мрежа сачињена од милиона рачунара широм света, који су међусобно повезани на најразличитије начине у циљу међусобне комуникације и размене информација. Ипак, најкориснији интернет сервис је комуникација, односно размена података и информација путем електронске поште, односно онлајн сервиса за комуникацију, друштвених мрежа и, у новије време, применом **веб-дневика** или **блогова**. Будући да се ради о комуникацији, корисник треба да поштује правила етичког коришћења интернета:

- не дели личне податке, као ни податке своје породице, као што су: име и презиме, адреса, телефон или фотографије са непознатим саговорницима на интернету, јер никад не знамо ко стоји са друге стране. Нису добронамерни сви људи који комуницирају на интернету;
- никада не уноси број кредитне картице родитеља да би купио или поручио нешто без њихове дозволе преко интернета;
- не откривај лозинке које користиш за приступ интернет сервису да не би биле злоупотребљене;
- не користи увредљиве и непристојне речи у разговору путем интернета;
- при комуникацији путем интернета поштуј ставове других људи, иако се можда разликују од твојих. Понашај се пријатељски у онлајн комуникацији;
- при писању текста на интернету, не пиши великим словима, то је исто као да вичеш.

У новије време, актуелизацијом друштвених мрежа појављује се термин **дигитални отисак**, као резултат објава које корисник објављује на интернету. При нашем свакодневном сурфовању убеђени смо да смо заштићени и анонимни, али није тако. То значи да активности корисника на друштвеним мрежама прати много више људи него што мислимо. Предузимањем радњи од стране корисника: свиђање страница и објава, дељење линкова, статуса,

слика, видеа, коментирање и сл., формира се **дигитални отисак** корисника, односно креира се база активности сваког корисника преко кога се сазнаје о каквој је личности реч, која су њена интересовања и занимљивости и шта утиче на њу.

Према томе, **дигитални отисак су све интернет информације (и позитивне и негативне) које су случајно или намерно остављене.**

Дигитални отисак има и позитивне и негативне стране, али њих треба контролисати. На пример, позитивна страна дигиталног отиска је када тражимо неки рецепт на интернету и путем тог рецепта можемо да дођемо до бројних кулинарских страница. Међутим, постоје и негативне стране које су последица неодговорне употребе интернета. На пример, фотографије за које смо мислили да су приватне постале су јавне, лични подаци у одређеним моментима нису добро заштићени и сл. Зато, пре него што се објави нешто на интернету, корисник треба добро да размисли које су предности објаве и како она може да му наштети. Пре сваке објаве, корисник треба да одмери позитивне и негативне вредности и донесе исправну одлуку пре објаве. Најчешћи савети о позитивном дигиталном отиску су следећи:

- Одјавите се при напуштању друштвених мрежа или електронске поште;
- Не откривајте лозинке;
- Не записујте јавно лозинке;
- При креирању лозинке користите слова, бројеве и знаке;
- Често бришите архиву;
- Не делите личне податке на интернету.

У овој теми ћемо ставити нагласак на **веб-дневнике**, односно **блогове**, као моћну онлајн алатку која омогућава везу између наставника и ученика, поделу наставних јединица, вежби, видео-записа, доделу допунског материјала онима који желе да науче више од онога што се нуди у уџбеницима и сл.

Да би уочили функционалности и могућности које нуде блогови помоћу претраживача, покушајмо да пронађемо блогове различитих аутора о различитим темама. Постоји много блогова који садрже управо теме из градива које изучавамо, **информатици**.

Кључни појмови

веб-дневник, блог, блогер, логовање, корисничко име, лозинка, запис, коментар, регистрација, дигитални отисак.

5.1 Појам о блогу и његова примена

Блог или **веб-дневник** представља онлајн дневник, у коме се садржина приказује хронолошким редом, односно најновије вести или садржаји се приказују први, а остали иду доле, тј. сортирани су тако да се од најновијих иде према старијим објавама.

Реч блог потиче од речи **web** што значи мрежа и **log** што значи унос.

У суштини, блогови представљају комуникациону алатку која повезује кориснике, пословне групе и друге веб-истомишљенике.

Аутор блога се назива **блогер.** Он не само да креира блог, већ стално обнавља и уређује његову садржину, Креирање, обнављање и уређивање садржине блога назива се **блоговање**. Блогер може да буде свако ко познаје рад на рачунару и интернет технологије и јесте спреман да дискутује на одређену тему.

Садржина веб-дневника може бити различита: текстови, белешке, дискусије, фотографије, видео-клипови, адресе ка другим локацијама на интернету и сл. Блогери од самог почетка знају о чему желе да пишу, да коментаришу, односно блогују. Најчешће теме за блоговање су следеће:

- приручници о образовању и теме из образовања,
- познате личности: биографије и њихова достигнућа,
- спорт,
- здравље и лепота,
- теме о родитељима и деци,
- компјутерске игрице,
- домаћи љубимци и њихов узгој,
- рецепти,
- политика,
- водич кроз различите области,
- животна искуства,
- услуге и производи,
- путовања,
- мода,
- уређење дома и сл.

Свакако, постоје и многе теме које интересују кориснике. Блогер одлучује о којој теми ће креирати блог. Блогери се најчешће одлучују за теме које су им познате, о којима имају предзнања и помоћу дискусије проширују и продубљују знање, намењени су јавности или пак одређеној циљној групи која дискутује у вези с темом или темама које су постављене на блогу. Диску-

сије могу бити забавне и поучне, а такође, могу се наћи и садржине које су од великог значаја за интернет популацију.

Да би сазнали могућности које омогућава блоговање, оспособићемо се да креирамо блог и будемо део блог заједнице.

Запамти!

Блог или веб-дневник представља локацију на интернету у облику електронског часописа у коме се садржине приказују обратним временским редоследом. Блогер је аутор веб-дневника, односно блога. Блоговање је процес креирања, уређивања, коментарисања и одржавања блога. Поштовање правила о етичком понашању при комуникацији преко интернета важи и приликом блоговања и остављања дигиталног отиска. Дигитални отисак представља траг који остављамо после примене различитих сервиса на интернету.

Питања

1. Шта је блог?
2. Шта су блогери?
3. Ко може да буде аутор блога?
4. Шта је блоговање?
5. Какве теме могу да се користе при креирању блога?
6. Ко пише садржине блогова?
7. Наброј неколико правила о етичком коришћењу блога!
8. Да ли се блогови могу примењивати као алатка у школским циљевима?
9. Шта је дигитални отисак?

5.2 Креирање блога

Креирање блога је једноставан процес. Најједноставнији, најбржи и најјефтинији начин за примену његових могућности, специјалности и функционалности је путем бесплатних блог-сервиса. Постоји много блог-сервиса за израду блогова:

1.Blogger;
2.Drupal;
3.Ghost;
4.Joomla;
5.Magento;
6.Medium;
7.Squarespace;
8.Tumblr;
9.Typepad;
10.Weebly;
11.Wix;
12.WordPress.

Најпопуларнији блог сервиси су **Blogger** и **WordPress.**

Blogger представља једну од најстаријих платформи на **Google-**у и зато што је део **Google** пакета, лако приступамо употреби нашег личног налога **(account**) на **Gmail-y.** Према томе, да би отпочели са блоговањем на блог сервису Blogger, потребно је да прво имамо своје корисничко име и лозинку (**Gmail account**).

Слика 1: **Blogger** лого

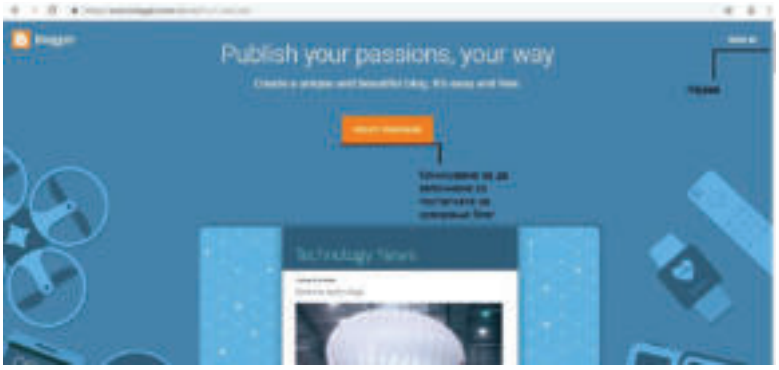
Како је **Blogger** хостован онлајн, поред њега корисник добија и поддомен који изгледа овако: имесајта.blogspot.com.

Блог сервис **Blogger** може да се користи за: писање дневника, постављање слика, писање личних радова, портала са вестима и новостима, приручника и сл. Притом, није ограничен број блогова које аутор може да креира и одржава. Поред тога, корисник помоћу блога може да оствари и зараду, путем рекламирања, промоција различитих производа и сл.

5.2.1 Кораци ка креирању блога.

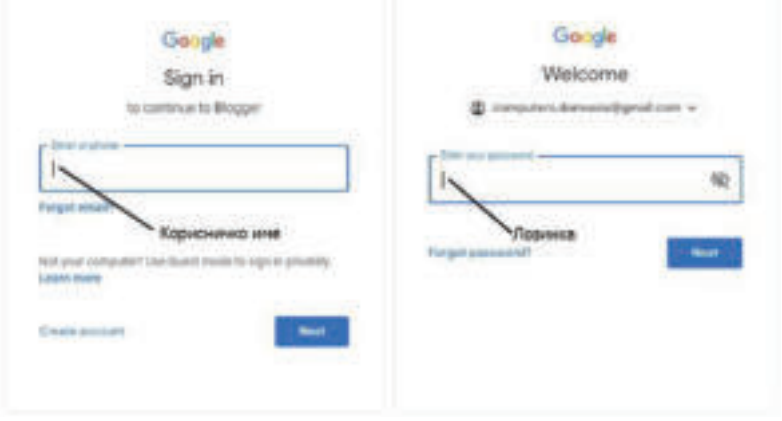
Да бисмо започели поступак креирања блога на бесплатном сервису **Blogger,** потребно је да најпре креирамо рачун на Gmail-у, затим покренемо веб-претраживач (Browser), као на пример: Google Chrome, Internet Explorer, Mozilla Firefox, Opera и сл. У траци за адресе претраживача пишемо адресу

бесплатног блог сервиса, www.blogger.com, при чему се појављује следећи прозор:



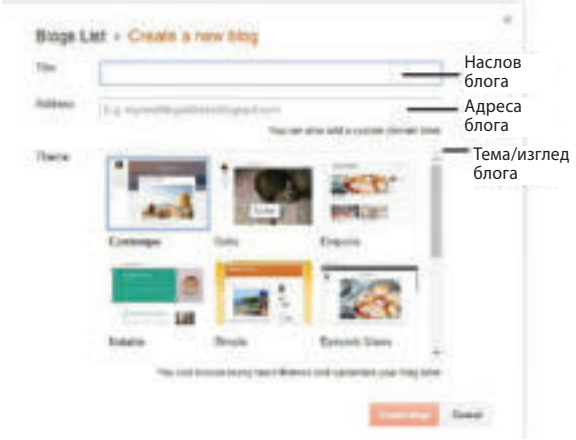
Слика 2: Почетни екран при приказивању странице за креирање блога

Уколико корисник нема свој налог (**Account**), кликне на опцију **Sing in** и почиње поступак регистрације на **Gmail,** и обратно, уколико има рачун на Gmail-у, тада бира опцију **Create New Blog**, када се отвара прозор за пријаву:



Слика 3: Прозор за пријаву

После пријаве, корисник даје име блогу, адресу и бира тему, односно изглед блога. То се ради попуњавањем поља са следеће слике:



Слика 4: Креирање блога

Коментар!

Поред тема или изгледа који су дати као опција, корисник може да дода тему која је изван листе понуда.

Када корисник упише адресу блога, тада се на левој страни у близини рама за унос појављује порука:

Checcking Adress Availability, а то значи да се врши провера да ли је унесена адреса доступна, тј. да ли је користи други корисник. Затим се бира тема или изглед блога, који је адекватан садржини блога. На крају се бира дугме **Create Blog** и приступа се контролној табли (**Dashboard**) блога. Помоћу ове контролне табле корисник уређује садржину блога, додаје нове садржине, управља коментарима и сл.

Слика 5: Контролна табла (Dashboard) блога

5.2.2 Уређивање садржине блога

После појављивања (Dashboard) на контролној табли, корисник може да започне са уређивањем садржине блога. Да би се креирала објава (**Post)**, кликне се на тастер **New post,** при чему се појављује следећи прозор:

Слика 6: Креирање објаве (Post) на блогу

Кликом на тастер **Publish**, креира се објава на садржини блога. Објава може бити сачувана кликом на дугме **Save**, прегледана кликом на дугме

preview, и затворена кликом на опцију **Close**. Са десне стране прозора примећујемо да се налази опција за допунска уређивања поставки (**Post setting**) објаве (**Post)**, као на пример: додавање ознака да ли ће се објава аутоматски приказати кликом на тастер **Publish** или ће корисник поставити датум и време објаве, додати локацију са које је направљена објава и сл.

Коментар!

Када се објава прикаже, поново се као главни екран појављује Dashboard. Приступ контролној табли (Dashboard) има само администратор блога. Кликом на линк Post може се видети цела листа објава.

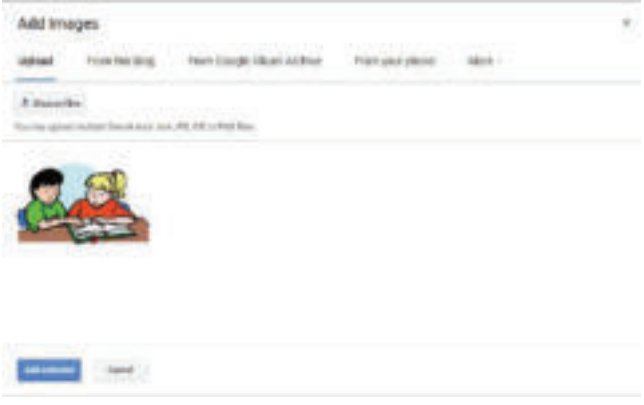
На самом прозору за креирање објава примећује се да је писање, едитовање и формирање садржине објаве исто као и у **MS Word**, што значи да корисник може да изабере фонт, боју фонта, стилове фонта, додавање линкова, додавање слика и слично.

5.2.3 Додавање слика и видеа у блог

Док је отворен прозор за креирање објава, прво се смешта курс на место где треба да се дода слика и бира се алатка за додавање слике:

Слика 7: Алатка за додавање слика

Затим се отвара следећи прозор, преко којег се слика качи



Слика 8: Избор слике за објаву

Помоћу опције **Chose Files** бира се слика која ће се прикачити из рачунара. Селектована слика се објављује кликом на дугме **Add Selected,** које се налази у левом доњем углу прозора.

Слика 9: Уређивање слике у објави

Коментар!

Поред тога што можемо да објавимо слику из рачунара, при креирању објава са сликом могу се употребити и слике које су већ искоришћене у блогу, слике из Google архиве, као и слике са телефона.

У прозору се при селектовању додате слике појављују опције за њено уређење, као на пример: уређење димензија слике, подређивање слике, додавање ознаке слици, додатне поставке и брисање.

Поступак додавања видеа у блог је исти као и додавање слике. Да би се додао видео у објаву, потребно је да је видео снимљен и да је сачуван у рачунару. Кликом на опцију **Choose video to upload,** у објави почиње да се приказује видео. По завршетку приказивања, кликне се на **Publish**.

Покушај!

Шта су програми? Наведи дефиниције програмирања! Како се прави програм? Које програмске језике познајеш?

5.2.4 Додавање линкова у блог

Подсетимо се!

Шта су линкови? Како се може препознати да ли је један текст линк? Наведи неколико карактеристика линкова!

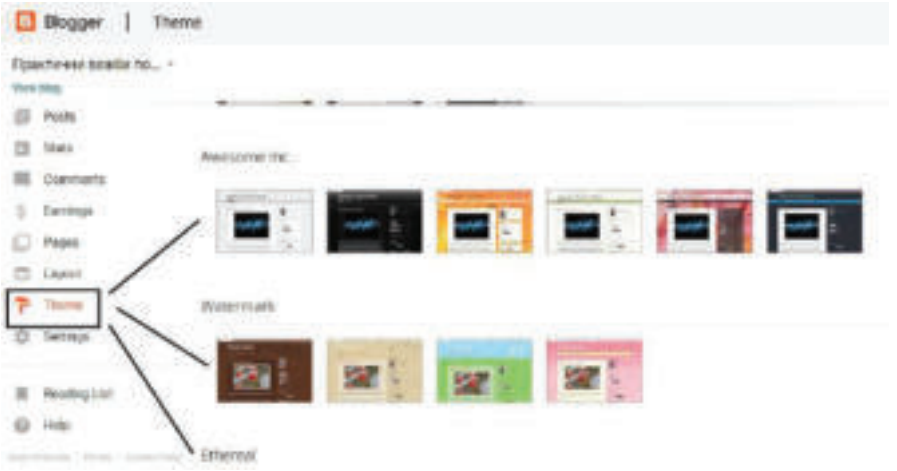
После објаве (**Post)** на блогу, корисник може да дода и линкове, односно да се повеже са другим страницама постављеним на интернету. У том циљу се селектује текст у објави у којој треба да се нађе линк и кликне се на алатку из прозора за додавање линкова:

Слика 10: Додавање линкова у објаву на блогу

У прозору за уређивање линка (**Edit link),** у пољу **Text to Display,** појављује се текст који је селектован у објави, а корисник је потврдио да жели да буде линк. Затим, селектује се дугме **Web Address** и у поље се уписује адреса странице са којом корисник жели да се повеже. Уколико је потребно да се успостави веза са имејл адресом, тада се означава дугме **Email Address** и на крају се кликне на дугме **Publish.**

5.2.5 Мењање теме/изгледа блога

Корисник може изменити тему блога током уређивања иако ју је изабрао на самом почетку. Поступак мењања теме/изгледа блога се врши преко контролне табле (**Dashboard**) и опције **Theme:**



Слика 11: Избор теме/изгледа блога

Када је тема/изглед изабран, приказује се како би блог изгледао и уколико је изглед прихватљив, кликнемо на дугме **Add** за додавање теме.

На крају се на контролној табли кликне на линк **View Blog** да би могао да се направи преглед објава и изглед блога.

Запамти!

Најчешће коришћени блог сервис је Blogger и он припада Google-у. Корисник мора имати кориснички налог (account) на Gmail-у, а то подразумева корисничко име и лозинку за најаву. Објаве или постови су садржине које се објављују на блогу. Преко контролне табле (dashboard) аутор блога може да креира поставке, да додаје објаве, уређује објаве, да мења теме/изглед блога и сл.

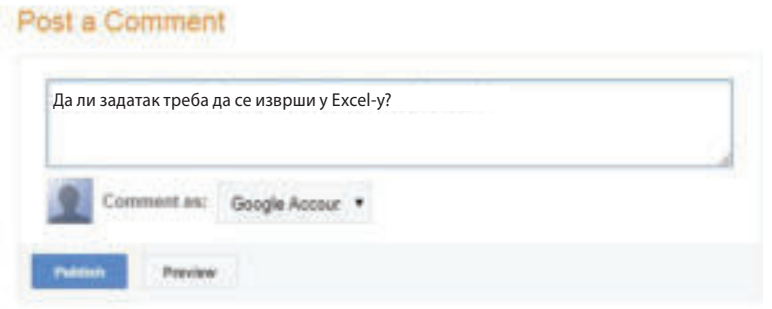
Питања

1. Набројте неколико блог сервиса!
2. Који је најјпопуларнији блог сервис?
3. Чему служи контролна табла (dush table) при креирању блога?
4. Шта су постови?
5. Шта је поступак за креирање поста?
6. Шта све може да буде садржина блога?

5.3 Коментарисање садржине блогова

Као што је речено на почетку теме, блогови нам омогућују да будемо део заједнице са одређеном тематиком. У ствари, можемо да учествујемо у дискусији о било којој теми уколико покажемо интересовање за то да проширимо знања, делимо искуства, изналазимо решења о проблемима, приказујемо добре праксе и сл. Из наведених разлога можемо да коментаришемо објаве, односно постове у блогу.

Да бисмо могли да коментаришемо садржај блога, потребно је да прво позовемо на прегледање садржаја блога уписивањем адресе блога у адресну траку на веб-претраживачу. Притом, прегледамо његов садржај. Да бисмо додали коментар објаве, кликнемо на **Comment**, при чему се појављује образац за додељивање коментара, као што је приказано на слици:



Слика 1: Додавање коментара на објаву на блогу

У празно поље уписује се текст коментара и бира се профил преко кога ће се коментарисати. Затим имамо две опције: **Publish** и **Preview**. Уколико се изабере **Publish**, коментар ће бити објављен испод садржаја објаве, а са **Preview** се прегледају коментари.

Да би администратор или креатор блога могао да одговори на постављене коментаре или пак да се укључи у започету дискусију, прво прави преглед приступајући преко контролне табле (**Dashboard)** и опције **Comment.**

Слика 2: Преглед коментара садржаја блога

Кликом на коментар, опција **Reply** омогућава да администратор блога да одговор на коментар. У празно поље уписује се одговор и кликне се **Publish** да би се коментар објавио.

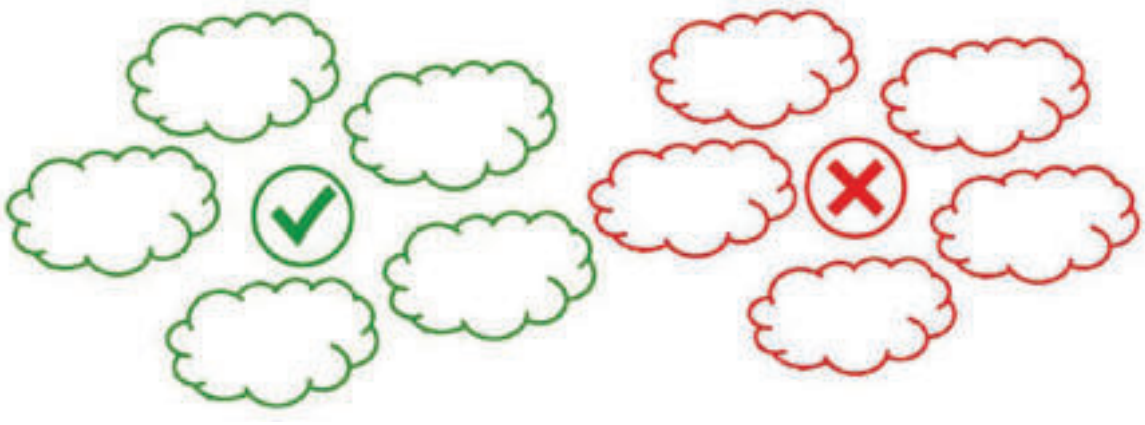
Запамти!

Приликом блоговања и корисници и администратор додају коментаре. Преко коментара корисници започињу разговор, деле искуства, постављају питања о нечем непознатом, одговарају на питања. Коментари омогућавају да се буде део заједнице на различиту тему.

ПОНОВИМО! УВЕЖБАЈМО!

Задатак 1:

Попуни следећи дијаграм тако што ће дозвољена правила о коришћењу интернета бити написана у зеленом делу, а недозвољена у црвеном делу.



Задатак 2:

Наброј савете за остављање позитивног дигиталног отиска.

- _____
- _____
- _____
- _____
- _____
- _____

Задатак 3:

Објасни шта представља веб-дневник!

_____

_____

_____

Задатак 4:

Допуни реченице да би добио тачне тврдње.

Људи који уређују веб-дневнике називају се_____

Поступак уређивања блогова назива се_____.

Задатак 5:

У веб-дневнику вести/садржаји се објављују:

а) према броју слика објављених у блогу;
б) без неког посебног реда;
в) према хронолошком реду, односно по датуму објаве.

Задатак 6:

Пре него што се започне поступак за израду блога,

а) неопходно је да корисник има имејл адресу, тј. електронску пошту,
б) корисник може да има имејл адресу, али то није неопходно,
в) није потребна имејл адреса.

Задатак 7:

Корисничко име власника (аутора блога) користи се као:

а) име за потпис,
б) име за најаву у веб-дневнику;
в) тему за дискусију.

Задатак 8:

Записи на блогу се називају:

а) статуси;
б) објаве;
в) твитови.

Задатак 9:

Посетилац блога може да остави поруку испод једне или више објава на блогу помоћу опције за додавање:

а) линк,
б) архива блога,
в) коментари.

Задатак 10:

Шта је потребно да се уради да би се објавио нови запис у блогу?

_____

_____

_____

Задатак 11:

1. _____ 5. _____
2. _____ 6. _____
3. _____ 7. _____
4. _____ 8. _____

Задатак 12:

Како треба да се понашаш при писању или посети одређених блогова на интернету?

_____

_____

_____

Пројектни задатак

Креирајте веб-дневник/блог на неком од бесплатних сервиса за креирање блогова. Изаберите тему која одговара некој од пројектних активности у школи или, пак, која одговара темама из наставних садржаја, као на пример: еко-интеграција, међуетничка интеграција и интеркултуралност у школи, спорт и спортске активности, ликовна дела и догађаји, играње, програмирање и сл. Пажљиво одаберите тему/изглед блога који одговара садржини блога. Блог треба да обухвати различите садржаје: текст, линкове, видео-клипове, слике и сл. Адресе блогова размените са друговима да бисте могли да остављате коментаре испод објава, а затим одговорите на следећа питања:

1. Која је адреса твог блога?
2. Који је наслов твог блога?
3. Коме је блог намењен?
4. Ког датума је написан последњи запис у блогу?
5. Ког датума је написан први запис у блогу?
6. Осим записа, којих других информација има на блогу?

РЕЧНИК КЉУЧИХ ПОЈМОВА
(стручна терминологија)

Активна ћелија	Ћелија која је у моменту означена за унос података назива се активна ћелија.	Тема 1
Анализа	Анализа је разлагање проблемског задатка на посебне елементе.	Тема 3
Бинарни бројеви	Бинарни бројеви су 0 и 1. Они чине бинарни језик у коме се подаци, информације и комаде претварају у низове од 0 и 1.	Тема 2
Блог	Блог или веб-дневник представља онлајн дневник у коме се садржај приказује хронолошким редом, односно најновије вести или садржине се приказују прве, а остале иду ниже.	Тема 5
Графикон	Графикон представља графички приказ података.	Тема 1
Графички приказ	Сликовит приказ	Тема 3
Дебагер	Дебагер је проналазач и одстрањивач грешака (bugs).	Тема 4
Дигитални отисак	Дигитални отисак су све интернет информације (позитивне и негативне) које су случајно или намерно остављене.	Тема 5
Едитор	Едитор је уређивач изворног кода.	Тема 4
Елиминација	Елиминација је процес избацивања чињеница које нису важне и које нас не воде ка коначном решењу проблемске ситуације.	Тема 3
Запис	Записи су објаве на блогу.	Тема 5
Извори програм	Изворни програм је листа текстуалних команди које треба превести у извршни рачунарски програм.	Тема 4
Извршни програм	Извршни програм је датотека која може да се извршава као програм на рачунару.	Тема 4
Иницијализација	Иницијализација је додељивање вредности променљивој при њеном дефинисању.	Тема 4
Интерактивни програми	Интерактивни програми су програми у којима постоји некаква врста комуникације (интеракције) међу субјектима.	Тема 3

Искази	Искази представљају наредбе које казују рачунару шта да уради.	Тема 3
Кодирање	Превођење података, информација и програма у језик разумљив рачунару и обратно назива се кодирање.	Тема 2
Колона	Колоне су вертикални делови табеле	Тема 1
Коментар	Коментари су најједноставнија алатка за интеракцију на блогу, односно за успостављање комуникације са заједницом.	Тема 5
Компајлер	Компајлер је преводилац изворне датотеке у извршну.	Тема 4
Константе	Константа је стална и непроменљива величина.	
Координате	Геометријски елементи који се користе за тачно утврђивање положаја у некој тачки.	Тема 3
Криптографија	Криптографија је научна дисциплина која се бави проучавањем методологије за слање и примање порука на начин који је разумљив само онима којима је намењена.	Тема 2
Логичко размишљање	Логичко размишљање је начин долажења до решења путем поштовања тачности тврдњи, односно исказа.	Тема 3
Ситуација	Ситуација у програмском језику представља нови градбени елемент који омогућава интерактивност у програму путем активности у програму помоћу миша или притискањем на одговарајуће дугме на тастатури.	Тема 3
Неследбене ћелије	Неследбене ћелије су несуседне ћелије.	Тема 1
Следбене ћелије	То су ћелије које се налазе једна до друге или једна поред друге.	Тема 1
Програмирање	Писање програма помоћу програмског језика назива се програмирање.	Тема 2
Променљиве	Променљиве или варијабле које се мењају у зависности од унесеног податка.	Тема 3
Радна свеска	Радне свеске су документи који се креирају у програм за табеларна израчунавања. На пример: MS Office, Excel, Open, Office.org, Apple и Work Numbers.	Тема 1
Радни лист	Свака радна свеска у програму за табеларне прорачуне састоји се од радних листова.	Тема 1

Ред	Редови су хоризонтални делови табеле.	Тема 1
Редослед	Поредак података према неком услову је редослед.	Тема 1
Слободан софтвер	Слободан софтвер је софтвер који може да се користи, проучава и именује без ограничења, који може да се копира и редистрибуира у измењеном или неизмењеном облику без ограничења или уз минимална ограничења.	Тема 2
Упоредни израз	Упоредни израз је наредба или исказ који омогућава компарацију вредности.	Тема 4
Структура података	Структуре података су групе података дефинисане под једним именом.	Тема 2
Табела	Табела је начин организације података у колоне и редове.	Тема 1
Ћелија	Пресек колоне и реда назива се ћелија.	Тема 1
Формуле	Формуле су аритметички и логички изрази које креира корисник у циљу извршавања прорачуна.	Тема 1
Функције	Функције су готове формуле које садрже име и аргументе за прорачун.	Тема 1
Циклус	Понављање низа наредби до испуњења услова је циклус.	Тема 4

КОРИШЋЕНА ЛИТЕРАТУРА

Davis, S. (2014). Beginning Programming with C++ for Dummies. Wiley Publishing

Дејтел П., Дејтел Х. (2010). C++ Како се програмира – седмо издање. Арс Ламина

Freun S., Starks J., Schmieder E. (2016). Microsoft Office 365 Excel 2016 comprehensive. University Idianapolis

Gardner S. (2005). Buzz Marketing with Blogs for Dummies. Wiley Publishing

Marji M. (2014). Learn to program with Scratch. William Pollock

www.bebras.org

https://education.microsoft.com

www.bbc.co.uk

www.code.org