GYORGYI YOVANÇEVSKI
SILVIYA NIKOLOVSKA

PROGRAMLAMA TEMELLERİ

II.SINIF DERS KİTABI
ELEKTROTEKNİK MESLEĞİ

Üsküp, 2023

Gyorgyi Yovançevski
Silviya Nikolovska

PROGRAMLAMA TEMELLERİ

II. SINIF DERS KİTABI

ELEKTROTEKNİK MESLEĞİ

Üsküp, 2023

PROGRAMLAMA TEMELLERİ

II. SINIF DERS KİTABI
ELEKTROTEKNİK MESLEĞİ

Yazarlar:
Gorgi Yovançevski,
Silviya Nikolovska,

Gözden geçirenler:
Deyan Spasov
Bilyana Pejovska
Viktoriya Dokaza

Orijinal baskının başlığı:

ОСНОВИ НА ПРОГРАМИРАЊЕ
УЧЕБНИК ЗА II ГОДИНА
ЕЛЕКТРОТЕХНИЧКА СТРУКА
Ѓорѓи Јованчевски
Силвија Николовска

Makedonca'dan Türkçe'ye çeviri
Ervin Salih

Lektör:
Abdülcan Feta

Mesleki redaksiyon:
Ervin Salih

Editör:
Ervin Salih

Grafik ve teknik düzenleme:
Vladanka Koleva, Evgeniya Pavlova – ARS STUDIO

Yayımcı:

Kuzey Makedonya Cumhuriyeti'nin Eğitim ve Bilim Bakanlığısok. "Aya Kiril ve Metodiy" No.54, 1000 Üsküp

Yer ve yayın tarihi: Üsküp, yıl 2023
Ulusal Ders Kitapları Komisyonu'nun 19.08.2022 tarih ve 26-548/1 sayılı kararıyla bu ders kitabının kullanımı onaylanmıştır.

İ Ç İ N D E K İ L E R

Giriş

MODÜL 1: C++ PROGRAMLAMA DİLİNE GİRİŞ 1

1.1 Programlar 4

1.2 Algoritmalar 5
Algoritma Adımları 6
Alıştırma Ödevleri 7

1.3 Algoritmaların Temsil Edilmesi 8
Yapılandırılmış Algoritmalar 10
Bilgiyi Kontrol Etme Soruları 11

1.4 Programlama 12
Programlama Aşamaları 12
Programlama Dilleri 14
Makine Dili 14
Sembolik Diller 15
Yüksek Seviyeli Programlama Dilleri 16
Programlama Dillerinin Nesilleri 21
Programlama Dillerinin Ayrımı 21
Etkinlikler Diyagramı 22
Bilgiyi Kontrol Etme Soruları 23

1.5 Tümleşik Geliştirme Ortamı 24
Microsoft Visual Studio 2019 Tümleşik Geliştirme Ortamı 27
Code::Blocks Tümleşik Geliştirme Ortamı 38

1.6 C++'ya Giriş 45
C++'nın Tarihçesi 45
C++ Dilinin Elemanları 45
C++ Programı 46
Yazdırma komutu 49
Alıştırma Ödevleri 53
Büyüklükler ve Veriler 53
Sabitler ve Değişkenler 54
Verilenin Adları 56
Veri Türleri 57
Sabit ve Değişken Verilerin Bildirilmesi 59
Alıştırma Ödevleri 63
Bilgiyi Kontrol Etme Soruları 63

1.7 Veri Türleri 65
Tamsayı Veri Türü int int 65
Operatörlerin Kısaltılmış Biçimi 68
Alıştırma Ödevleri 69
Gerçek Veri Türü float, double, long double 70
Ondalık Verileri Temsil Etme Biçimleri 73
Aritmetik İfadeler ve Veri Türünün Dönüştürülmesi 74
Alıştırma Ödevleri 78
Karakter Veri Türü char 78
Alıştırma Ödevleri 80
Mantıksal Veri Türü bool 81
Bitsel Operatörler 86
Alıştırma Ödevleri 88
Sayılı Veri Türü 88
Alıştırma Ödevleri 91
Dizeler 91
Alıştırma Ödevleri 93
Verilerin Türünü Yeniden Adlandırma 94
Türün Otomatik Belirlenmesi 95
Bilgiyi Kontrol Etme Soruları 95

1.8 Verilerin Okunması ve Yazdırılması 97
Sayısal Verilerini Okuma ve Yazdırma 97
Karakter ve Dize Verilerini Okuma ve Yazdırma 102
Biçimlendirilmiş Yazdırma 104
Verileri Okurken Ayrıntılar 105
Alıştırma Ödevleri 109
Bilgiyi Kontrol Etme Soruları 110

Terimler 111

Özet 117

MODÜL 2 C++'DA AKIŞ KONTROLÜ VE FONKSİYONLAR 123

2.1 Sıralı Kontrol Yapısı 126

2.2 Seçim İçin Algoritmik Kontrol Yapıları ve Seçim Kontrol Komutları 128
İki Olasılıklı Seçimi İçin Algoritmik Kontrol Yapısı **eğer-o zaman-değilse** 128
İki Olasılıklı Seçim İçin Kontrol Komutu if 130
İki Olasılıklı Seçim İçin Kontrol if-else 131
Çözülmüş Ödevler 134

Alıştırma Ödevleri 136
İf ve if-else Kontrol Komutların İç İçe Yerleştirilmesi 137
Çözülmüş Ödevler 138
Koşullu Operatör ?: 140
Alıştırma Ödevleri 141
Çok Olanaklı Seçim İçin Algoritmik Kontrol Yapısı durum ve Çok Olanaklı Seçim İçin Kontrol Komutu switch 141

Çözülmüş Ödevler 145
Alıştırma Ödevleri 147
Bilgiyi Kontrol Etme Soruları 147
Ödevler 148

2.3 Tekrarlama Algoritmik Kontrol Yapıları ve Tekrarlama Kontrol Komutları 150
Saymalı Tekrarlama İçin Algoritmik Kontrol Yapıları ve for Kontrol Komutu 151
Arttırma ve Azaltma Operatörleri 157
Sayaçın for Kontrol Komutunun Gövdesinde Kullanılması 159
for Kontrol Komutunun Özellikleri 160

Çözülmüş Ödevler 162
Alıştırma Ödevleri 165
Döngünün Başında Çıkışlı **iken-yürütür** Tekrarlama Algoritmik Kontrol Yapısı ve while Kontrol Komutu 166
Çözülmüş Ödevler 170
Alıştırma Ödevleri 173
Döngü Sonunda Çıkışlı Tekrarlama Algoritmik Kontrol Yapısı **yürütür-iken** ve do-while Kontrol Komutu 174

Çözülmüş Ödevler 177
Alıştırma Ödevleri 180
Tekrarlama Kontrol Komutlarının İç İçe Girmesi 180
Bilgiyi Kontrol Etme Soruları 183
Ödevler 185

2.4 Atlama İçin Algoritmik Kontrol Yapıları ve Atlama İçin Kontrol Komutları 186
continue Kontrol Komutu 186
break Kontrol Komutu 188
exit () Kontrol Komutu 189
goto Kontrol Komutu 191

Bilgiyi Kontrol Etme Soruları 192

2.5 Fonksiyonlar 192
Kütüphane Fonksiyonları 192
Matematiksel Fonksiyonlar 193
Rastgele Sayı Üretme Fonksiyonu 196
Karakterlerle Çalışma Fonksiyonları 199
sizeof() Operatörü 201
Alıştırma Ödevleri 202
Kullanıcı Fonksiyonları 203
Dönüş Değerli Kullanıcı Fonksiyonlar 207
Dönüş Değerli Fonksiyonun Tanımı ve Bildirimi 210
Fonksiyonun Sabit Parametreleri 212
Çözülmüş Ödevler 214
Alıştırma Ödevleri 215
Dönüş Değeri Olmayan Kullanıcı Fonksiyonları 216
Referans Parametreleri 220
Çözülmüş Ödevler 223
Alıştırma Ödevleri 226
Fonksiyonda Fonksiyon Çağırmak 226
Global ve Yerel Değişkenler 228
Statik Değişkenler 231
Yerleşik Fonksiyonlar 233
Alıştırma Ödevleri 234

Bilgiyi Kontrol Etme Soruları 235

Ödevler 237

Terimler 238

Özet 241

MODÜL 3 C++'DA KARMAŞIK VERİ TÜRLERİ 247

3.1 Tek Boyutlu Diziler 249
Tek Boyutlu Dizilerin Bildirimi 250
Dizinin Başlatılması ve Elemanlara Değer Atama 252

Çözülmüş Ödevler 258
Aralığa Dayalı for Komutu 264
Alıştırma Ödevleri 265
Bilgiyi Kontrol Etme Soruları 266

3.2 İki Boyutlu Diziler – Matrisler 266
İki Boyutlu Dizilerin Bildirimi, Başlatılması ve Elemanlara Değer Alama 268
İki Boyutlu Dizinin Boyutlarının Hesaplanması 271
Çözülmüş Ödevler 275
Alıştırma Ödevleri 278
Bilgiyi Kontrol Etme Soruları 279

3.3 İşaretçiler 280
İşaretçilerin Bildirimi 280
& Adres Operatörü 281
Referanssızlaştırma Operatörü * 282
İşaretçilerin Başlatılması 283
İşaretçiler ve Diziler 285
Adres Aritmetiği 287
new ve delete Komutları 291
Alıştırma Ödevleri 292
Bilgiyi Kontrol Etme Soruları 292

3.4 Dizeler 293
Dizelerle Çalışma Fonksiyonları 293
Çözülmüş Ödevler 301
Dizeyi Sayıya Dönüştürme Fonksiyonları 303
Sayıyı Dizeye Dönüştürme 305
Alıştırma Ödevleri 306
Bilgiyi Kontrol Etme Soruları 306

Terimler 307

Özet 307

EK

Basit Veri Türleri 311

ASCII Karakterler 312

Kaynakça 313

Giriş

Bu ders kitabı, programlama eğitimi için başlangıç dersi tanımlamaktadır. Makedonca özel sözde dil aracılığıyla problemlerin algoritmik ifadesinin temellerini ve C++ programlama dilinde algoritmaların kodlanmasını kapsayacak şekilde tasarlanmıştır.

Bu ders kitabı, elektroteknik mesleğinin II. sınıfı "Programlama Temelleri" dersi için eğitim programı ile uyumludur. Bu ders için, üç modüler birimden oluşan modüler programa göre yazılmıştır.

Birinci modüler birimi olan "C++ programlama diline giriş" öğrenci, algoritmaları ve onların metinsel ve grafiksel ifade edilmesini, programlama dillerini ve programlama ortamlarını tanır. Ayrıca, C++ dilinde değişkenler, veri türleri, okuma ve yazdırma komutları gibi C++ dilinde program yapısının temel öğelerini tanır.

İkinci modüler birimi olan "C++'daki Akış Kontrolü ve Fonksiyonlar" , algoritmarının yürütme akışını kontrol etmeye yönelik temel algoritmik kontrol yapıları açıklanmıştır: seçim kontrol yapıları ve tekrarlama kontrol yapıları. Bunlar herhangi bir algoritmayı ifade etmek ve yürütme sürecini kontrol etmek için yeterlidirler. Onların uygulamaları, C++ programlama dilinde karşılık gelen kontrol komutları ile açıklanmıştır ve gösterilmiştir. Ayrıca, bu modüler birimde, alt algoritmalar ve C++ dilinde fonksiyonlarla ilgili uygun uygulamalarr işlenmiştir. C++ kütüphane fonskiyonları incelenmiştir, kullanıcı tanımlı fonksiyonlara ise özellikle önem verilmiştir.

Üçüncü modüler birimi olan "C++'da Karmaşık Veri Türleri"'nde, öğrenciye C++ programlamada sayısal diziler, işaretçiler ve dizelerin kullanımı tanıtılmıştır.

Sunulan malzemenin programlamaya yeni başlayan her öğrenci için yeterli olduğunu düşünüyoruz ve öğrenilmesi zor değildir çünkü onu birçok örnek, alıştırma ve çözülmüş ödevlerlerle kolay ve erişilebilir bir şekilde sunmaya çalıştık. Her öğretim biriminden sonra, bilgiyi kontrol etmek için sorular ve bağımsız olarak çözmek için ödevler verilmiştir.

Yazarlar

Not: *Yazarlar lektör ile işbirliği içinde bazı bilişim terimlerini ve ayrıca C++ programlama dilinden terimleri Makedonca diline ifade etmeye çalıştılar. Bazı okuyucular için alışılmadık gelebilir, ancak bilişim terimlerinin Makedonca dilinin kullanımı sorunu üzerinde çalışmanın zamanının geldiğine inanıyoruz.*

Özelde, bu programlama dilinin özelliğinden dolayı bazı terimler sadece Makedonca dilinde yazılır. Ders kitabında yazarların, bu programlama dilindeki bazı terimlerin çoklu anlamları nedeniyle Makedonca dilinin kullanım kurallarına uymayan bir biçimde yazılması gerektiği yönündeki görüşlerine saygı duyulmaktadır.

MODÜL

C++ PROGRAMLAMA DİLİNE GİRİŞ

1

Bu bölümde aşağıdakiler hakkında bilgi edineceksiniz:

- Program, programlama ve programlama dilleri.
- Yazılım ve onun bölümü.
- Algoritma, algoritma adımları, genel ve ayrıntılı algoritma.
- Algoritmaların tanımlanması.
- Yapılandırılmış programlama.
- Sözde dil, blok-diyagram, kaynak program ve yürütülebilir program.
- C++ için tümleşik geliştirme ortamları.
- C++'nın tarihi.
- C++ dilinin elemanları.
- Büyüklük, veri, sabit, değişken ve değişken türü.
- Veri türü.
- Ondalık sayıları temsil etme biçimleri.
- Türünn yeniden adlandırılması, türün otomatik belirlenmesi.
- Verilerin okunması ve yazdırılması.

Anahtar kelimeler

bool	Ekleme operatörü (insersiyon)
char	Fonksiyon
cin	Fonksiyonel dil
Code::Blocks	Genel algortima
cout	Gerçek veri türü
double	Girdi veri akışı
endl	Girintileme
enum	Hata ayıklayıcı
enum	İfade
float	İlişkisel operatör
int	Karakter veri türü
iomanip	Karmaşık veri türü
iostream	Kayan nokta
istream	Kaynak program
long	Kısaltılmış operatör biçimi (karmaşık operatör)
long double	Kodlama
long long	Kullanıcı program
Microsoft Visual Studio	Makine bağımsız dil
namespace	Makine dili
ostream	Mantıksal dil
return	Mantıksal ifade
short	Mantıksal operatör
string	Mantıksal veri türü
Unified Modeling Languge – UML	Metin düzenleyici – editör
unsigned char	Metinsel veri
unsigned int	Modüler programlama
unsigned long	Nesne yönelimli dil
unsigned long long	Otomatik veri türü belirleme
unsigned short	Örtük dönüştürme (tür zorlaması)
Açık tür dönüştürme (tür atama)	Problem yönelimli dil
Adres veri türü	Programlama dili
Adlandırlmış sabit	Programlama dili dilbilgisi
Algoritma	Programlama dili semantiği
Algoritmik adım	Programlama dili sözdizimi

Algoritmik kontrol yapısı	Prosedürel programlama dili
Ana fonksiyon	Sabit nokta
Anahtar kelime	Sayılı veri türü
Artırma operatörü ++	Sembolik dil
Assembler	Sıralı kontrol yapısı
Aşırı doldurma	Sistem programları
Ayırma operatörü (ayıklama)	Sözde dil
Ayrılmış kelime	Tanımlayıcı
Ayrıntılı algoritma	Tam sayı veri türü
Azaltma operatörü – –	Temel veri türü
Bağlaç	Tümleşik geliştirme ortamı
Basit veri türü	Tür
Başlık dosyası	Tür belirleyici
Betik (Komut soyası) dili	Veri
Bildirimsel programlama dili	Veri türünü yeniden adlandırma
Bitsel operatörler	Yapılandırılmamış veri türü
Blok-dıyagram	Yapılandırılmış programlama
Boole fonksiyonları	Yapılandırılmış veri türü
Boyut	Yazılım
C++ standart kütüphanesi	Yeni satır işareti
Çevirici (derleyici)	Yerleşik veri türü
Çıkış (kaçış) dizisi	Yorum
Çıktı veri akışı	Yorumlayıcı
Değişken	Yukarıdan aşağıya programlama
Değişken bildirimi	Yüksek programlama dili
Değişken tanımlama	Yürütülebilir program
Değişkenin başlatılması	Uygulama
Değişmez	Uygulama programı
Dize	Zorunlu programlama dili
Dizelerin birleştirilmesi	

1.1 Programlar

Bilgisayar kendi başına herhangi bir iş yapamaz. Bilgisayarın herhangi bir işi yapabilmesi için, bir **programı** (İng. program) etkinleştiren ve çalıştıran uygun komutların verilmesi gerekir. Program doğru yazılmışsa, bilgisayar onu çalıştıracak ve uygun sonuçları verecektir

Bilgisayarın tüm çalışması, **sistem programları** (İng. system programs) olarak adlandırılan özel programların kontrolü altında gerçekleşir. Sistem programları üç gruba ayrılabilir:

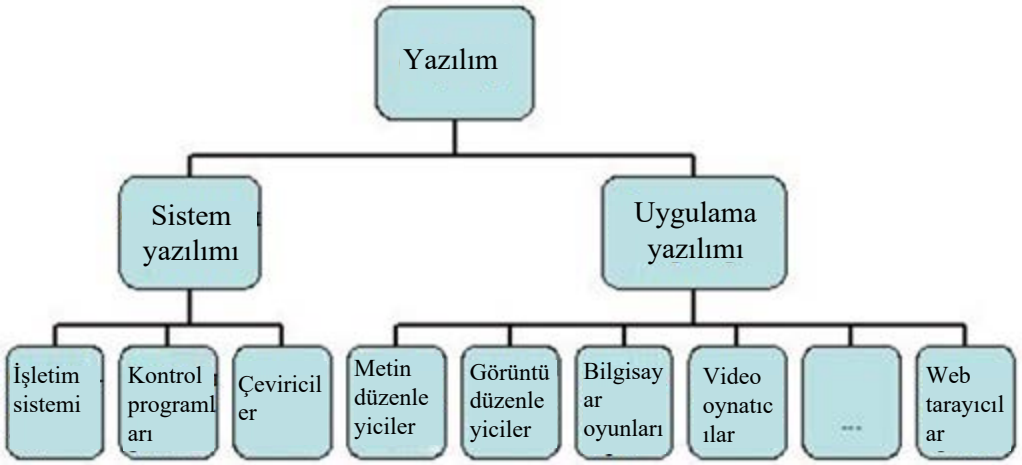
- **İşletim sistemleri** (İng. operating systems).
- **Kontrol programları** (İng. control programs).
- **Hizmet (yardımcı) programlar** (İng. utility programs).

Kullanıcılar için belirli görevleri yerine getiren programlara **uygulama programları** (İng. application programs) veya **kullanıcı programları** (İng. user programs) denir.

Bilgisayarımızda bulunan tüm uygulama programları belirli ödevler çözer, yani onlarla belirli işler yaparız. Örneğin, iyi bilinen uygulama programları, Microsoft Office paketindeki programlardır: MS Word, MS Excel, MS PowerPoint, vb.

Bilgisayarın çalıştırılabildiği tüm programlara tek bir adla **yazılım** (İng. software) denir.

Şekil 1.1.1'de, yazılımın olası bir bölümü verilmiştir.



Şekil 1.1.1

Program[1], bilgisayarın anlayabileceği ve yürütebileceği metni tanımlamaktadır. Programlar, insan ve bilgisayar arasındaki iletişimi sağlayan **programlama dilinde** (İng. programming language) yazılır. Programlar, bilgisayarın "anladığı" komutlarla yazılır ve komutlarda "ifade edilen" eylemleri gerçekleştirir.

Programlar metin düzenleyici ile düzenlenebilir (yazılabilir ve değiştirilebilir), dosya olarak kaydedilebilir ve çalıştırılabilir.

Program yazma süreci, **programlama** (İng. programming) olarak adlandırılır, programları yazan (programlayan) kişilere ise **programcı** (İng. Programmers) denir.

1.2 Algoritmalar

Her insan kahve yapması, belirli bir yemeği yapması, araba kullanması, bilgisayara program yüklemesi, ertesi gün için çantasını hazırlaması vs. gerektiğinde ne yapılması gerektiğini çok iyi bilir.

İnsan faaliyetleri, çoğunlukla, belirli sayıda ayrı küçük (temel) faaliyetlerin, yani gerçekleştirilmesi için açıklama gerektirmeyen faaliyetler dizisi olarak gerçekleştirilebilir.

Örneğin, kahve hazırlarken temel faaliyetler şunlar olabilir:

- Cezveye su koymak.
- Cezveye kahve koymak.
- Cezveyi ocağa koymak.
- Cezvenin yerleştirildiği ocağı açmak.
- Kahvenin kaynamasını beklemek.
- Cezveyi ocaktan kaldırmak.
- Ocağı kapatmak.

Bir kişinin (kahveyi hazırlayan kişi gibi) belirli bir etkinliği (kahve hazırlamak) gerçekleştirebilmesi için, temel faaliyetleri ve onların yerine getirme sürecini (sırasını) bilmesi gerekir. Böyle bir temel faaliyetler dizisine süreç veya **algoritma** (İng. algorithm) denir.

Şunu söyleyebiliriz:

Algoritma, sonlu sayıda kesin olarak tanımlanmış faaliyetlerden oluşan prosedür ve bunların tam olarak verilmiş yürütme sırasıdır.

Algoritma terimini daha ayrıntılı olarak, aşağıdaki süreçle üç sayıdan en büyüğünü bulma örneği ile açıklayacağız:

- Herhangi iki sayıyı karşılaştırmak ve bunlardan daha büyük olanı belirlemek.
- İlk karşılaştırmandan daha büyük sayıyı üçüncü sayı ile karşılaştırmak ve bunlardan daha büyük olan sayıyı belirlemek

[1] C++ programlama dilinde yazılmış bir program ***şekil 1.3.4.***'te verilmiştir.

İkinci karşılaştırmadaki daha büyük sayı, üç sayıdan en büyüğüdür.

Sürecin kesin olarak tanımlanmış sadece iki faaliyetten oluştuğu görülebilir.

Bu süreci herhangi üç numaraya uygulamak için önceden atanmaları gerekir. Sürecin sonucu en büyük sayıdır. Bu, bir görevi çözmenin, *girdi verilerine* belirli bir prosedür - bir algoritma uygulayarak *çıktı sonuçlarını* belirlemeyi içerdiği anlamına gelir.

Bu nedenle şunu söyleyebiliriz:

Algoritma, *kesin olarak belirlenmiş bir sırada girdi verilerine uygulanan ve çıktı sonuçlarına ulaşılan, tam olarak tanımlanmış sonlu faaliyetler kümesinden oluşan bir prosedürdür.*

Algoritma kelimesi Latince dilinden alınmıştır ve arap rakamlarla dört temel aritmetik işlemi gerçekleştirmek için kuralları ilk kez formüle eden 9. yüzyıl Arap matematikçisi Abu Ja'far Muhammad ibn Musa al–Khwarizmi 'nin, Al–Khwarizmi soyadının Latince çevirisidir.

Algoritma Adımları

Bir algoritmayı oluşturan faaliyetlere **algoritma adımları** (İng. algorithms steps) denir. Adımların daha genel veya daha ayrıntılı olmasına bağlı olarak, algoritma **genel** veya **ayrıntılı** olabilir. Örneğin, bir önceki noktadan ödev için (verilen üç sayıdan en büyüğünü belirleme) genel algoritmayı **şekil 1.2.1**'de gösterilmiş şekilde yazabiliriz.

Adım 1. Üç sayının verilmesi.
Adım 2. Herhangi iki sayıyı karşılaştırmak ve onlardan büyüğünü belirlemek.
Adım 3. Adım 2'de bulunan daha büyük sayıyı üçüncü sayıyla karşılaştırmak ve onlardan büyüğünü bulmak.
Adım 4. sonucu yazdırmak.

Şekil 1.2.1

Sayıları a, b ve c ile, a ve b'den büyük olanı p ile, p ve c'den büyük olanı n ile gösterirsek, **şekil 1.2.2**'deki gibi daha ayrıntılı algoritma yazabiliriz.

Adım1. a, b ve c sayılarını vermek
Adım 2. Eğer a b'den büyükse, o zaman p=a, değilse p=b.
Adım 3. Eğer p c'den büyükse, o zaman n=p, değilse n=c.
Adım 4. n'i yazdırmak.

Şekil 1.2.2

Her algoritmada girdi verileri ve çıktı sonuçları olduğunu söylemiştik. Çıktı sonuçları, algoritma adım adım yürütülürken girdi verilerinin işlenmesi ("dönüştürülmesi")

ile elde edilir. Bu arada, işlem her zaman doğrudan değil, **ara sonuçlar** aracılığıyla gerçekleştirilir. Böylece, yukarıdaki algoritmada ara sonuç **p,** girdi verileri **a, b** ve **c** , çıktı sonucu **n'**dir.

Algoritmanın doğruluğunu kontrol etmek için rastgele girdi verileri üzerinde test yapılır. Örneğin, önceki algoritmayı şu sayılar için test edelim: a = 37, b = 12 ve c = 44.

Adım 1. a = 37, b = 12, c = 44.
Adım 2. a (= 37) , b'den (=12) büyükür, o zaman p = a (= 37).
Adım 3. p (= 37) c'den (=44) büyük değildir, onun için n = c (= 44).
Adım 4. n'i (=44) yazdır.

Alıştırma Ödevleri

1. Pasta yapmak için algoritma yazın.
2. Evden çıkmadan önce sabah hazırlığı için algoritma yazın.
3. Dosyaları diskten USB'ye kopyalamak için algoritma yazın.
4. Üç sayının ortalama değerini belirlemenin olası adımları nelerdir?
5. Verilen üç basamaklı sayıdan ortadaki basamağı ayırmak için (algoritmik adımlarla), algoritma yazmaya çalışın.
6. n doğal bir sayısının çift ya da tek olduğunu belirleyen algoritma yazın. (Eğer sayı 2 ile bölünüyorsa çifttir, değilse tektir).
7. Bugün, yılın sırasıyla kaçıncı gününün olduğunu bulacağınız algoritma yazın. Örneğin: bugün 01.09.2020 ise bugün yılın 245. günüdür.
8. İki doğal sayının en büyük ortak bölenini (EBOB) bulmak için genel algoritma yazın.
9. İki doğal sayının en küçük ortak katını (EKOK) bulmak için genel algoritma yazın.
10. Bugün kaç yaş, ay ve gün olduğunuzu hesaplayacağınız algoritma yazın.

1.3 Algoritmaların Temsil Edilmesi

Algoritmalar üç şekilde temsil edilebilir:

- Metinsel.
- Grafiksel.
- Programlama dili ile.

Algoritmaların metinsel temsili, **1.2 Algoritmalar** bölümünde gösterildiği gibi adımlar halinde yapılır. Bu arada, algoritmaları tanımlamak için **sözde dil** de kullanılabilir. Biz, şu kelimelerden oluşan (Türkçe dilinden sözcüklerle) sözde dil kullanacağız: **algoritma, alt algoritma, başlangıç, bitiş, eğer, o zaman, değilse, iken (olduğu sürece), yürütür (çalıştır), kadar, için,adım, artır, azalt, oku, yazdır, kes, devam et, atla, çıkış.**

```
algoritma En büyük
başlangıç
        oku a, b, c;
        eğer a > b
                o zaman
                        p ← a
                değilse
                        p ← b
        son_eğer {a > b}
        eğer p > c
                o zaman
                        n ← p
                değilse
                        p ← c
        son_eğer { p > c};
        yazdır n;
bitiş { En büyük };
```

Şekil 1.3.1

Sözde dil için ayrılmış kelimeler olduklarını vurgulamak için bu kelimeleri kararmış (kalın) olarak yazacağız.

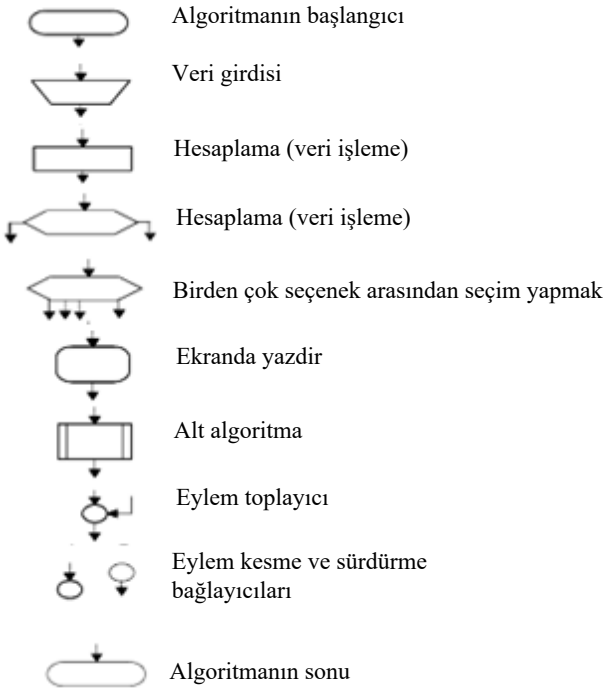
Böyle bir sözde dil ile bir algoritmanın metinsel temsil edilmesine ilişkin bir örnek olarak, ***şekil 1.2.2***'deki üç sayıdan en büyük sayıyı bulmak için algoritma ***Şekil 1.3.1***'de verilmiştir. Burada değer atama adımını ifade eden ok kullanılır. Örneğin, p ← a oku, p'ye a'nın değeri verildiği eylemi ifade edilir.

Algoritmaların grafik gösterimi **blok-diyagram** (İng. flowchart) ile yapılır. Blok diyagramda, belirli eylemler (işlemler) için ***şekil 1.3.2***'de verilmiş özel grafik semboller (bloklar) kullanılır. Algoritmaların bu grafik sembollerle gösterilmesine **standart grafik gösterim** denir. ***Şekil 1.3.1***'de metinsel olarak sunulan üç sayıdan en büyüğünü belirleme ödevi için algoritmanın blok diyagramla gösterilmesi şekil 1.3.3'te verilmiştir.

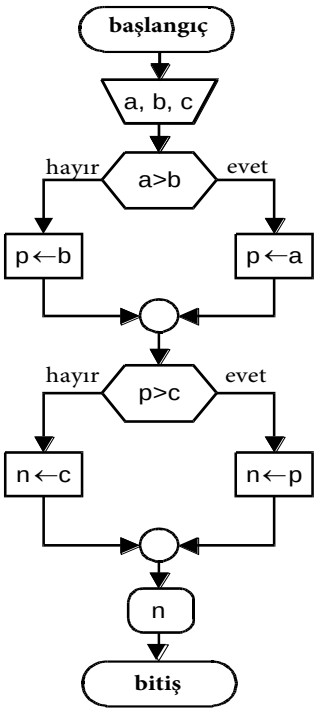
Algoritmaların grafik gösteriminin hem avantajları hem de dezavantajları vardır. Avantaj, algoritmadaki faaliyet akışının daha fazla görünür olmasıdır, çünkü insan görüntüyü daha kolay algılar. Diğer taraftan, daha büyük bir algoritma için blok diyagram birden fazla sayfa kaplayabilir ve algoritma okunmaz hale gelebilir.

Algoritmalar doğrudan programlama dili ile de temsil edilebilir. Bu arada, algoritmayı, biz programlama dilinin komutlarıyla **kodluyoruz** (İng. encoding), yani "ifade ediyoruz".

Bir programlama dilinin komutlarıyla kodlanmış algoritmaya **kaynak program** (İng. source program) denir. Kaynak programın çalıştırılabilmesi için yürütülebilir programa (İng. executive program) çevrilmesi gerekir.



Şekil 1.3.2



Şekil 1.3.3

Örneğin, i verilen 3 sayıdan en büyüğünü bulmak için C++ programlama dilinde kodlanmış algoritması *Şekil 1.3.4.*'te verilmiştir.

```
1    #include <iostream>
2    using namespace std;
3
4    int main() {     // Uc sayidan en buyugu:
5
6        double a, b, c, p, n;
7        cout << " İlk sayiyi girin: ";
8        cin >> a;
9        cout << " İkinci sayiyi girin:";
10       cin >> b;
11       cout << " Ucuncu sayiyi girin: ";
12       cin >> c;
13       if (a > b)
14           p = a;
15       else
```

Şekil 1.3.4

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

Şekil.1.3.4 (devam)

Programı çalıştırdıktan sonra çıktı şu olacaktır:

```
İlk sayiyi girin: 123
İkinci sayiyi girin: 321
Ucuncu sayiyi girin: 234
123, 321 ve 234 sayilarindan en buyugu:  321
```

Yapılandırılmış Algoritmalar

Daha karmaşık ödevler için algoritmalar çok uzun ve okunamaz olabilir. Bu yüzden, bu algoritmalara göre yazılacak programların da anlaşılması zor olabilir. Bu, bazı değişikliklerin veya eklemelerin yapılması gerektiğinde özellikle önemlidir. Büyük programlarda daha kolay bulunmak için, bugün bunlar **yapılandırılmış programlama** (İng. structured programming) olarak bilinen programlama tekniği kullanılarak yazılmaktadır.

Yapılandırılmış programlama adı, algoritmaların eylemlerini kontrol eden **algoritmik kontrol yapılarının** (İng. algorithm control structures) kullanımından gelmektedir.

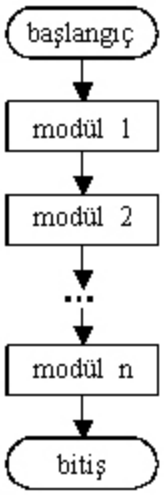
Yapılandırılmış programlamada iki programlama tekniği kullanır:

- Yukarıdan aşağıya programlama (İng. top-down programming).
- Modüler programlama (İng. modular programming).

Yukarıdan aşağıya programlama, ödevi daha küçük ve daha basit ödevlere, yani **alt ödevler** olarak adlandırılan ödevlere bölerek (ayrıştırarak) yapılır. Gerekirse, kolayca programlanabilir ödevler elde edilene kadar bu alt ödevler daha da basit alt ödevlere bölünür.

Bu şekilde ayrıştırılan ödevin her alt ödevi, diğer alt ödevlerden bağımsız olarak ayrı bir ödev olarak düşünülebilir. Her alt ödev için ayrı algoritma yazılabilir ve ardından algoritmaya göre ayrı bir program yazılabilir. Alt ödevler için bu tür programlara modüller (İng.modules) denir.

Her modülün yalnızca sadece bir giriş noktası (başlangıç) ve sadece bir çıkış noktası (bitiş) vardır, ***şekil 1.3.5.*** Programdaki tüm modüller sıralı (arka arkaya) düzenlenmiştirve hiçbiri atlatılamaz. Bir modülün gerçekleştirdiği fonksiyon başka bir modülde tekrarlanamaz, yani modüller arasında örtüşme olmamalıdır. Ayrıca, bir modül birkaç başka modülden oluşabilir ve tersi daha büyük bir modülün parçası olabilir.



Şekil 1.3.5

Modüler programlama özellikle programları değiştirirken veya hataları düzeltirken kullanışlıdır. Programın değiştirilmesi veya programdaki bir hatanın düzeltilmesi gerekiyorsa, sadece hatanın meydana geldiği modül değiştirilir, diğer modüller değişmez.

Yapılandırılmış programlamadaki algoritmaların grafiksel gösterimi, **standart gösterim** olarak adlandırılan yapılandırılmamış programlamadaki grafiksel gösterimden farklıdır.

Yapılandırılmış algoritmalar dikdörtgen bloklarla temsil edilirler. Bir dikdörtgen, kendi başlangıcı ve sonu olan ayrı bir birimi (modül) ifade etmektedir. Bir modül daha basit modüllere bölünüyorsa, daha büyük olanın içinde daha küçük dikdörtgenlerle temsil edilirler. Örneğin, verilen üç sayıdan en büyüğünü bulma ve ***şekil 1.3.3***'te gösterilmiş olan standart algoritmanın yapılandırılmış şekli, ***şekil 1.3.6'***da gösterilmiştir.

Bilgiyi Kontrol Etme Soruları

1. Program nedir, programlama nedir?
2. Programlamanın yapıldığı dillere ne denir?
3. Sistem programları hangileridir, uygulama programları ise hangileridir?
4. Sistem programları nasıl gruplandırılır?
5. Yazılımın ana bölümü nedir?
6. Microsoft Windows bir uygulama programı mıdır?
7. Microsoft Excel'i hangi uygulama programları grubuna yerleştirdiniz?
8. Bilgisayarınızda çalışan üç uygulama programı belirtin. Bunları ***Şekil 1.1.1'***deki ikinci seviyede uygun kategoriye yerleştirmeye çalışın.
9. Şekil 1.1.1'de verilmiş üçüncü seviye yazılım gruplarının her biri için birer örnek program vermeye çalışın.

```
oku a,b,c
eğer a > b
        o zaman
                p ← a
        değilse
                p ← b
eğer p > c
        o zaman
                n ← p
        değilse
                n ← c
yazdır n
```

Şekil.1.3.6

10. Algoritma nedir? Bir tanımını söyleyin.
11. Algoritmik adım nedir?
12. Ayrıntı düzeyi açısından algoritma nasıl olabilir?
13. Bir algoritma nasıl gösterilebilir?
14. Algoritmalar hangi diyagramlarla grafiksel olarak tanımlanır?
15. Algoritmaların standart grafik gösterimi için grafik sembolleri (blokları) listeleyin.
16. Kaynak program nedir?
17. Bilgisayar doğrudan kaynak programı çalıştırabilir mi?
18. Yapılandırılmış programlama nedir?
19. Yapılandırılmış programlamada hangi teknikler kullanılır?
20. Yapılandırılmış algoritmalar grafiksel olarak nasıl temsil edilir?
21. Program modülü nedir?
22. Modüler programlama tekniği ne için kullanışlıdır?
23. Yukarıdan aşağıya programlama tekniğini açıklayın (veya size açıklanmasını isteyin).

1.4 Programlama

Bir kişinin bilgisayar yardımıyla belirli bir ödevi çözecek bir program yazabilmesi için programlamayı bilmesi gerekir.

Programlama, program yazma sürecidir.
Programlayan kişilere programcı denir.

Programlama Aşamaları

Bir ödevi bilgisayar yardımıyla çözmek için sadece programlamayı bilmemiz ve program yazmamız yeterli değildir. Programlama, bu sürecin yalnızca bir parçasıdır ve bu süreç birkaç aşamadan oluşuyor:

- Ödevi kurma.
- Ödevi çözmek için algoritma (prosedür) tanımlama.
- Program yazma.
- Programı test etme.

Aşama I: Ödevi kurma

Ödev kurarken, hangi koşullar altında çözüleceği kesin olarak tanımlanmalı ve belirtilmelidir. Bu yüzden, ödev önce doğru bir şekilde anlaşılmalıdır. Bu, doğasını analiz ederek ve gerekirse ait olduğu profesyonel alanı derinlemesine inceleyerek yapılır.

Yukarıdaki söylediklerimizi daha ayrıntılı açıklamak için birkaç ödev kuracağız ve bunları inceleyeceğiz.

1. İlk 10 doğal sayının toplamı bulunsun.
2. İki bilinmeyenli iki lineer denklem sistemi çözülsün.
3. Bir mağazadaki eşyaların miktarları ve değerlerine göre liste yapılsın ve malın toplam değeri hesaplansın.
4. 16 kibrit çubuğundan iki oyuncu A ve B dönüşümlü olarak 1, 2, 3 veya 4 çubuk alır. En son çubuk alan oyuncu kazanır. Oyunculardan biri sürekli olarak kazanabilir mi?

İlk ödev için herkes bunun zor olmadığını söyleyecektir - toplamayı bilirsem, programı yazmayı da biliyorum.

İkinci ödev için, iki bilinmeyenli iki doğrusal denklem sistemini çözmek için, değiştirme yöntemi veya zıt katsayılar yöntemi gibi bazı yöntem bilinmelidir.

Üçüncü ödev için şunları bilmeniz gerekir:

- Her eşyanın adı.
- Her eşyanın miktarı.
- Kilogram, litre veya paket başına fiyatı.

Dördüncü ödevi çözmemiz için bilgisayara çubuklar koymayacağız, ancak 16 sayısından dönüşümlü olarak her oyuncunun söylediği kadar çıkaracağız- 1, 2, 3 veya 4. ödevin özü kaçar çubuk almanın yolunu bulmaktır. Rakibin ne kadar çubuk aldığını ve kaç tane kaldığını bilerek, sonuncuları alıp kazanmak.

Aşama II: Ödevi çözmek için algoritma tanımlama

Ödevin analizini gerçekleştirdikten sonra, onu çözmek için algoritma tanımlamak (bulmak/düşünmek) veya (daha fazla algoritma biliyorsak) seçmek gerekir. Çoğu zaman prosedür, ödev için gerçekleştirilen analizden ve ödevin ait olduğu alandaki danışılan profesyonel literatürden öne çıkar. Bu arada, prosedürün bilgisayarda yürütülmesi için geçerli olduğu dikkate alınmalıdır. Bilgisayarın doğru sonuç vermesi için gerçekleştirmesi gereken tüm faaliyetler (işlemler) açıkça belirtilmelidir. Her işlem tek bir şekilde tanımlanmalı ve işlem sırası tam olarak belirtilmelidir. Tabii ki, tüm prosedür nihai olmalıdır, yani sonlu sayıda işlemden sonra, sınırlı bir yürütme süresinden sonra sona ermelidir. Bir ödevi çözmek için bu şekilde tanımlanan prosedüre, daha önce bahsettiğimiz gibi, algoritma denir.

Aynı ödevi çözmek için birkaç farklı algoritmanın yazılabileceğini vurgulayalım.

Aşama III: Program yazma

Program yazma, algoritmayı bir programlama dilinin elemanlarıyla yazmaktır (ifade etmek, kodlamak). (Programlama diller hakkında bir sonraki alt başlıkta daha detaylı bahsedeceğiz).

Aşama IV: Programı test eme

Dördüncü aşamada, programın doğru sonuçlar verip vermediği test edilir. Test genellikle çözümlerini bildiğimiz veya çözümlerini (bir şekilde) kontrol edebileceğimiz değerlerle yapılır.

Programlama Dilleri

İnsanlar dil yardımıyla birbirlerini anlarlar ve iletişim kurarlar. Bu nedenle, işaretlerle, simgelerle veya sözle tanımlandığına bağlı olmadan, dil temel bir iletişim aracıdır.

Diller şöyle ayrılabilir:

- **Doğal.**
- **Yapay.**

İnsanlar birbirleriyle iletişim kurmak için Türkçe, Makedonca, İngilizce vb. gibi doğal dilleri kullanırlar. İnsanlar ve makineler arasındaki (veya iki makine arasındaki) iletişim için yapay diller kullanılır.

Özel bir yapay dil türü, insan-bilgisayar iletişimi için geliştirilen programlama dilleridir. Programlama dilleri ile, program adı verilen metin biçiminde yazılmış bir ödevi çözme prosedürü (algoritması) ifade edilir. Programlar, bilgisayarın "ifade edilen" eylemleri "anladığı" ve gerçekleştirdiği komutlarla yazılır.

Bilgisayarların ortaya çıkmasından bugüne kadar çok sayıda[2] programlama dili geliştirilmiştir. Bunlar üç gruba ayrılır:

- **Makine dili.**
- **Sembolik diller.**
- **Yüksek seviyeli programlama dilleri.**

Makine Dili

Makine dili (İng.machine language) bilgisayarın dilidir. Makine dili, her bilgisayarın anladığı ve bilgisayarların doğrudan çalıştığı tek dildir. Belirli sayıda, sadece sıfırlarla ve birlerle, yani sadece ikilik rakamlarla[3] ifade edilen makine talimatlarından

[2] Bugüne kadar 3000 üzerine programlama dilin geliştirildiği düşünülüyor.

(İng. Machine instructions) oluşmaktadır***, Şekil 1.4.1.***

Fakat, ikilik rakamlarla yazılmış kayıt çok büyük ve okunmazdır. Bu yüzden, makine programları (kağıtta) genellikle onaltılık sayı sisteminde yazılır[4], ***Şekil 1.4.1.***

Makine talimatları

İkilik kayıt	Onaltılık kayıt
10111000110101100010110	B8D616
1000111011011000	8ED8
10100000000000000000000	A00000
00000010000001100000000100000000	02060100
10111011000100001000000	BB1080
1000100000000111	8807
1011010001001100	B44C
1100110100100001	CD21

Şekil1.4.1

Sembolik Diller

Sembolik dil (İng. assembly language), makine dilinden daha üst seviyededir. İlk bilgisayarlarda, programlar sadece makine dilinde yazılıyormuş. Zamanla, verilen ödev için programın daha kolay ve daha hızlı temsil edilmesi için, programcılar, makine talimatlarını belirtmek için **semboller** veya **anımsatıcılar**[5] olarak adlandırılan kısa basit kelimeler kullanmaya başlamışlardır. Bu şekilde sembolik diller oluşmuştur. İsimlerini de zaten makine talimatlarının ve verilerin sembolik şekilde ifade edildiğinden almışlar.

Örneğin, şekil 1.4.1'deki makine programının, şekil.1.4.26'da gösterilmiş olduğu gibi sembolik kaydı var.

[3] İkili rakamlar 0 ve 1'dir. *Bunlar ikilik sayı sisteminin* rakamlarıdır. Biz, 10 rakamı-0, 1, 2, 3, 4, 5, 6, 7, 8 ve 9, olan *onlu sayı sistemine* alışığız.

[4] *Onaltılık sayı sisteminin* 16 rakamı var: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E ve F.

[5] Anımsatıcı (mnemonik) hafıza tekniğidir.

[6] ***Şekil 1.4.2'***deki her komutun başında, bu komutun hafızada bulunduğu yerin adresi gösterilmiştir.

Sembolik diller, makine diline göre daha yüksek anlaşılırlık derecesine sahiptir. Örneğin ***şekil 1.4.2***'den göründüğü gibi MOV, taşıma için kısaltmadır (İng.move), ADD ise ekleme yani toplama demektir (İng. add) vb.

Ancak sembolik diller bilgisayar için anlaşılmazdır, dolayısıyla sembolik dilde yazılan program bilgisayarda doğrudan çalıştırılamaz. Bu yüzden, programı sembolik dilden makine dilindeki programa çeviren **derleyiciler** (İng.compilers) adı verilen özel sistem programları geliştirilmiştir.

Sembolik diller, **makine yönelimli dillerdir** çünkü her bilgisayar veya bilgisayar ailesinin, bilgisayarın işlemcisine bağlı olarak kendi sembolik dili vardır.

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

Şekil 1.4.2

Sembolik diller daha çok **çevirici dilleri** (İng. assembly languages) olarak bilinir, bunların çeviricilerine ise **birleştirici** (İng.assemblers) denir.

Yüksek Seviyeli Programlama Dilleri

Yüksek seviyeli programlama dilleri (İng. high-level languages), programlama hakkında bahsettiğimizde aklımıza gelen dillerdir. Bu diller, bilgisayarların evrenselliği ile insan faaliyetinin birçok alanına girmeye başladığı 20. yüzyılın 50'li yıllarından başlayarak gelişmişler. Bilgisayar üreticileri ve kullanıcıları arasında şu soru ortaya çıkmıştır. Bilgisayarlar, makine ve sembolik programlama tekniğini bilmeyen insanlar tarafından kullanılabilmesi için insana nasıl yakınlaştırılabilirler?

Bu sorunu çözmedeki temel fikir, programları insanın doğal diline benzer bir dilde yazmakmış. Kısa süre içinde yüksek seviyeli programlama dilleri geniş çapta kabul edilmiş ve çok sayıda kullanıcı tarafından kullanılmaya başlanmıştır.

Bu dillerin hızla kabul edilmesi ve geniş çapta kullanılması aşağıdaki unsurlardan kaynaklanmaktadır:

- *Bu dillerin insanın doğal yazılı anlatım biçimine ve ödevin ait olduğu alanın terminolojisine yakınlığı nedeniyle programlama önemli ölçüde kolaylaştırılır.*
- *Algoritmaların hızlı ve verimli yazılmasını sağlar.*
- *Bilgisayarın teknik özellikleri hakkında bilgi gerektirmezler ve programın yürütüldüğü bilgisayarın türüne (tipine) bağlı değildirler, yani aynı program farklı bilgisayar türlerinde (tiplerinde) çalıştırılabilir.*
- *Öğrenmesi hızlı ve kolaydır.*
- *Kullanıcılar arasında program ve deneyim alışverişini sağlamaktadır.*

Belirtilen tüm unsurlar, makine, yani sembolik dillerin yapısından açıkça farklı olan yüksek seviyeli programlama dillerinin yapısına dayanmaktadır.

Yükesk seviyeli programlama dilleri yapay diller olmasına rağmen, doğal dillere benzer şekilde yapılmıştır. Şöyle ki, bu dillerin her birinin harflerden, sayılardan ve noktalama işaretleri, aritmetik işlemler, karşılaştırma işaretleri ve diğerleri7 gibi özel karakterlerden oluşan kendi alfabesi vardır.

Alfabenin işaretlerini birleştirerek, dilin temel yapıları - **dilin sözlüğünü** oluşturan **kelimeler** oluşturulur. Örneğin: read, do, while, new, OPEN, MULTIPLY, STOP, if vb.

Kelimelerin belirli bir anlamı vardır, ancak programlarda bilgisayarın çalışmasını doğrudan etkileyecek bağımsız öğeler olarak şekillenemezler, ancak **cümleler** veya **komutlar** (İng. statements) olarak adlandırılan dil yapılarında birleştirilirler. Komutların kesin olarak tanımlanmış anlamı vardır ve programda bağımsız bütünler olarak bulunabilirler, yani bilgisayarda belirli bir eyleme neden olabilirler.

Aşağıda birkaç komut örneği verilmiştir:

```
if (ti > jas) System.out.println("Tebrikler, Sen yendin. ");
throw new UserDefinedException("UserDefinedException olustu! ");
MULTIPLY SAYİ1 BY SAYİ2 GIVING CARPİM.
```

Alfabedeki her karakter dizisinin (kombinasyonunun) (doğal dillerde olduğu gibi) dilin bir kelimesini temsil etmediğini, yani kelimelerin her yapısının komut olmadığını vurgulamak önemlidir. Buna göre, yüksek seviyeli programlama dilleri de, doğal diller gibi, kelime ve komut oluşturma kurallarını içeren kendi **dilbilgisine** sahiptir.

Her dil, belirli **ayrılmış kelimeler** kümesi içerir (İng. reserved words). Bunlardan bazılarının programlama dili için özel anlamı vardır ve bunlara **anahtar kelimeler** (İng. key words) denir. Programcılar, ayrılmış kelimelerin dışında, belirli kurallara göre oluşturulmuş başka kelimeler de kullanabilirler. Bunlara **tanımlayıcılar** (İng. identificators) denir.

Dilin, **sözdizimsel kurallar** veya yalnızca **sözdizimi** (İng. syntax) olarak adlandırılan ve komutları oluşturan katı kuralları vardır. Bu kurallarla, programdaki her komutun doğruluğu kontrol edilir.

Programdaki her komutun doğru tanımlanmış anlamı olmalı ve doğru tanımlanmış eylemlere neden olmalıdır. Komutların anlamı dilin **semantiği** (İng. semantics) olarak adlandırılır.

Yüksek seviyeli programlama dillerinde yazılan programlar bilgisayarda doğrudan çalıştırılamaz. Bunun için, **derleyici** (İng. compilers) denilen makine diline çevirisini yapacak özel sistem programlarına ihtiyaç vardır. Her yüksek seviyeli programlama dilinin kendi derleyicisi vardır.

[7] Programlama dillerinde genellikle İngiliz alfabesi kullanılır.

Yüksek seviyeli programlama dilinde yazılan programa **kaynak program** (İng. source program) denir, çevrilen makine programına ise **yürütülebilir programı** (İng. executive program) denir.

Çeviri (derleme) şematik olarak aşağıdaki şekilde gösterilmiştir.

Şekil 1.4.3 a

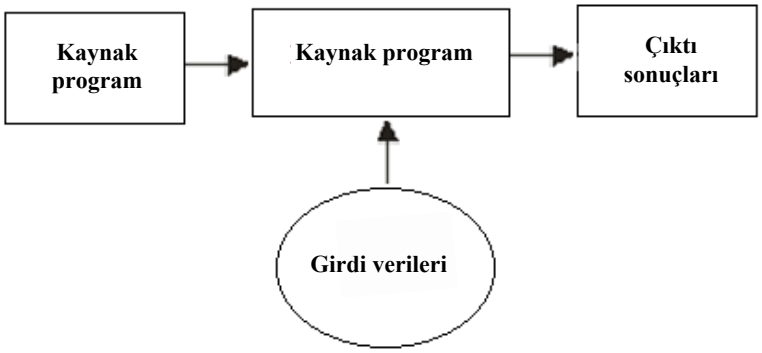
Yürütülebilir program genellikle bazı dış bellekte dosya olarak kaydedilir ve oradan gerektiğinde çağrılır ve çalıştırılır. Çağrıldığında, bilgisayarın çalışma belleğine yüklenir ve yürütülür.

Kaynak programların sıkça yürütülebilir programlara dönüştürülmeden doğrudan yürütülmesi de kullanılır. Bu, **yorumlayıcılar** (İng. interpreters) adı verilen diğer sistem programları ile yapılır.

Yorumlayıcılar, kaynak programın makine biçiminde dosyasını yapmazlar, bunun yerine, çalışan bellekteyken komutları yorumlarlar ve yürütürler.

Kaynak programın yorumlama şeması, ***Şekil 1.4.2*** b'de gösterilmiştir.

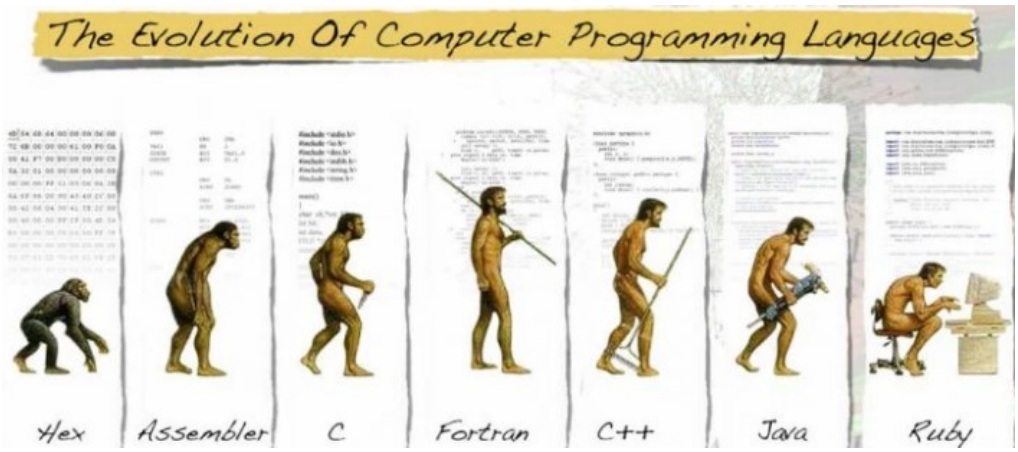
b.



Şekil 1.4.3 b

Yüksek seviyeli programlama dilleri, onlarla yazılan programın yürütüleceği makineye bağlı değildir ve bu nedenle **makineden bağımsız diller** olarak adlandırılır. Bu yüzden, yüksek seviyeli programlama dillerinin gelişmesi sırasında, bunlarla yazılan programların çalıştırılacağı bilgisayarın türü (tipi) dikkate alınmaz, ancak çözülen problemin türü dikkate alınır. Belli bir türdeki problemi (ekonomik, teknik, istatistiksel vb.) çözmeye yönelik olduklarını söylüyoruz ve onları **problem yönelimli diller** olarak adlandırılıyoruz.

Yüksek seviyeli programlama dillerinin gelişimi geçen yüzyılın 50'li yıllarında başlamış. Gelişimleri mecazi olarak literatürden alınan aşağıdaki görüntü ile temsil edilmektedir.



Tablo 1.4.1'de, daha önemli olan birkaç programlama dilinin kronolojik gelişimi gösterilmiştir.

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

Tablo 1.4.1

Günümüzde, 2010'dan sonra geliştirilen modern programlama dilleri, birkaç dilin iyi özelliklerini kullanarak, İnternette çalışan web uygulamalarını programlamak için tasarlanmıştır. Bunlar, özel programlar - web sayfalarında yürütülen, ancak diğer programların yürütülmesi sırasında yürütülen **komut dosyaları** (İng.scripts) olarak adlandırılan dosyalar oluşturmak için kullanılan **komut dosyası (betik) dilleridir** (İng. scripting languages). Komut dosyaları derlenmez, yorumlanır. Komut dosyası dilleri, HTML, Java, C++ vb. gibi diğer programlama dilleriyle bütünleşme ve iletişim için tasarlanmıştır

En ünlü komut dosyası dilleri şunlardır: JavaScript, PHP, Python, VBScript, vb.

Web programlama için en çok kullanılan programlama dilleri ***şekil 1.4.4***'te verilmiştir.

Şekil 1.4.4

Şekil 1.4.5 a, b, c, ç, d ve **e**'de, C++, Java, Pascal, FORTRAN, Ruby ve Python ile yazılmış ilk 10 doğal sayının toplamını bulma ödevini çözmek için basit programlar gösterilmiştir.

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

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

a

C++ programı

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

b

Java programı

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

c

PASCAL programı

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

ç

FORTRAN programı

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

d

Ruby programı

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

e

Python programı

Şekil 1.4.5

Programların yazılması metin düzenleyicide yapılır, daha sonra da bir dış belleğinde belirli bir adla dosya olarak kaydedilirler (saklanırlar). Program bu şekilde yürütülemez çünkü önce çevrilmesi gerekir.

Kaynak programların genellikle yazıldığı programlama dilinin adına göre son eki vardır ve program adından noktayla ayrılan programlama dilinin kısaltmasıdır: .pas, .bas, .for, .cpp, .java , vb. Örneğin: toplam.pas, ilk.bas, benimProgram.cpp, sirala.java vb.

Kaynak programların çevrilmesinden sonra elde edilen yürütülebilir programların şu uzantıları vardır: .exe, .com, .bat, .bin, .dll, vb. Örn: toplam.exe, ilk.com vb.

Programlama Dillerinin Nesilleri

Tarihsel gelişimine göre, programlama dilleri beş nesile ayrılır (İng. Generation Languages - GL), onlar şunlardır:

- Birinci nesil (1 GL): makine dili.
- İkinci nesil (2 GL): sembolik diller.
- Üçüncü nesil (3 GL): Yüksek seviyeli programlama dilleri (C++, Java, Fortran, vb.).
- Dördüncü nesil (4 GL): Üst yüksek seviyeli programlama dilleri (İng. Very high-level languages) (Perl, PHP, Python, SQL, vb.).
- Beşinci nesil (5 GL): yüksek programlama dilleri[8] (Mercury, Prolog, OPS5, vb.).

Programlama Dillerinin Ayrımı

Programlama dilleri, verilerin işlenme biçimine göre de ayrılabilir:

- **Komutsal.**
- **Bildirimsel.**

[8] Bazı yazarlar bu dilleri yapay zeka dilleri olarak adlandırır (İng. Artificial intelligence languages).

Komutsal şekil, programların veri işleme için komutlarla yazıldığı şekildir.

Bildirimsel şekil, sonuca nasıl ulaşılacağını açıklayan tanımlayıcı şekildir.

Komutsal diller şunlara ayrılır:

- **Prosedürel** (Pascal, C, Fortran, Basic, Ada, Modula ve diğerleri),
- **Nesne yönelimli** (C++, Java, C#, Delphi, Smalltalk ve diğerleri).

Bu dillerin temel özelliği değişkenlerin, komutların ve prosedürlerin[9] kullanılmasıdır. Prosedürel dillerde, verileri işleyen prosedürler ayrıdır, nesne yönelimli dillerde ise, prosedürler (fonksiyonlar veya yöntemler olarak adlandırılır) nesnelerin kendisinde saklıdır.

Bildirimsel diller şöyle ayrılır:

- **Fonksiyonel** (Lisp, Scheme, Miranda, Haskell, ML ve diğerleri),
- **Mantıksal** (Prolog ve diğerleri).

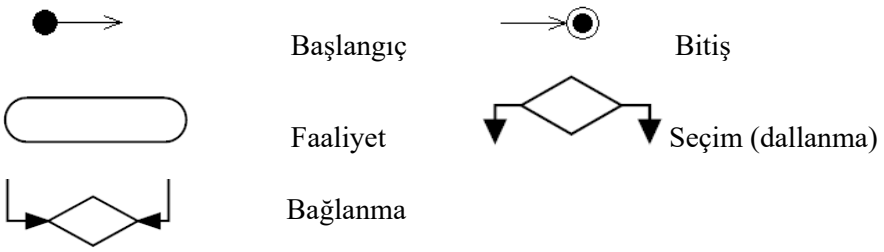
Fonksiyonel dillerin temel özelliği, ifadelerin ve fonksiyonların kullanılmasıdır. Program, daha basit fonksiyonlardan oluşan bir fonksiyondur (veya fonksiyonlar grubudur).

Mantıksal dillerde, temel hesaplama birimi, aynalamadan daha genel olan ilişkidir. Program şunlardan oluşur: gerçekler kümesi, izin verilir sonuçlar kümesi (ispatlama kuralları), sonuç çıkarma yöntemleri ve ispatın amacı.

Etkinlikler Diyagramı

Programların çalışmasını grafiksel olarak temsil etmek için çeşitli diyagramlar kullanılır. Diyagramların birleşik tanımlanması için, geçen yüzyılın 90'lı yıllarının sonlarında Unified Modeling Languga - UML adlı grafik dili oluşturuldu.

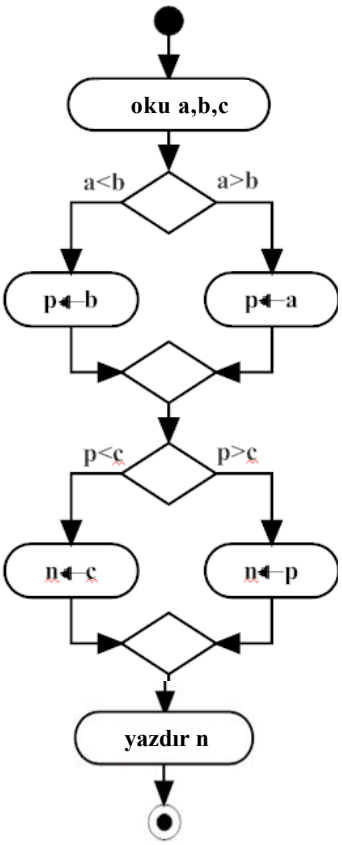
UML çok sayıda grafik sembolü içerir. Biz aşağıdakileri kullanacağız:



Şekil 1.3.3'teki diyagram, UML dilinde ifade edildiğinde ***Şekil 1.4.6'***daki gibi olacak.

[9] Değişkenler ve prosedürler hakkında daha sonra bahsedeceğiz.

Bilgiyi Kontrol Etme Soruları



Şekil 1.4.6

1. Programlama nedir?
2. Programlayan kişilere ne denir?
3. Bilgisayar yardımıyla ödev çözmenin aşamalarını açıklayın.
4. Program yazabilmek için neleri bilmeniz gerekiyor?
5. Hangi programlama dillerini duydunuz?
6. Makine dilinde programlar hangi rakamlarla yazılır? Neden makine dilinde programlama yapılmıyor?
7. Kağıtta makine programları hangi sayı sisteminde yazılır?
8. Makine dili mi sembolik diller mi daha basittir?
9. Programları sembolik dilden makine diline çevirme görevi olan sistem programlarına ne ad verilir?
10. Yüksek seviyeli programlama dillerinin yoğun kullanımı hangi unsurlara bağlıdır?
11. Doğal ve programlama dilinin alfabesi arasında fark var mıdır?
12. Yüksek seviyeli programlama dillerindeki kelimelerin anlamı var mıdır?
13. Yüksek seviyeli programlama dilinin sözlüğü neyden oluşuyor?
14. Program komutu sadece bir kelimeden oluşabilir mi?
15. Programcıların programlarda oluşturdukları kelimelere ne ad verilir?
16. Programlama dillerinde komutların oluşturulduğu kurallara ne denir?
17. Dilin semantiği nedir?
18. Yüksek seviyeli programlama dilinde yazılmış programa ne denir, makine diline çevrilmiş programın ise adı nedir? İlgili dosyaların hangi uzantıları vardır?
19. Çevirici ve yorumlayıcı arasındaki fark nedir?
20. Neden yüksek seviyeli programlama dillerinin makineden bağımsız diller olduğunu söylüyoruz?
21. Birkaç yüksek seviyeli programlama dili adlandırın.
22. Yüksek seviyeli programlama dillerinin nesillerini listeleyin.

23. Yüksek seviyeli programlama dillerinin bir ayrımını belirtin.
24. Prosedürel dillerinin ve nesneye yönelimli dillerinin ana özellikleri hangileridir?
25. Diyagramlar günümüzde hangi grafiksel dil ile temsil ediliyor? Temel grafik sembollerini çizin.

1.5 Tümleşik Geliştirme Ortamı

Önceki alt noktalarda bahsettiğimize göre, programlamaya başlayabilmemiz için, bit yüksek seviyeli programlama diliyle tanışmalıyız. Ardından, programı metin düzenleyicisinde yazıyoruz ve metin dosyası olarak kaydediyoruz.

Yazdığımız programın yazıldığı şeyi yapıp yapmadığını görmemiz için, onu makine diline çevirmemiz ve yürütmemiz gerekir. Bir makine dilini çevirirken bir derleyicinin kullanıldığını ve yürütülebilir programını/dosyasını elde ettiğimizi söyledik. Elde edilen yürütülebilir programını çalıştırmamız gerekir ve ancak o zaman programımızın ne yaptığını görebiliriz. Yorumlayıcı kullanıyorsak, program doğrudan çevrilir ve yürütülür, yani, yorumlanır.

Programcıların çalışmalarını kolaylaştırmak için, yukarıdaki işlevleri gerçekleştiren tüm programlar, ***tümleşik geliştirme ortamı*** (İng. Integrated Development Environment – IDE) adı verilen tek bir ortama yerleşir (entegre edilir). C++'da programlamak için, Microsoft Visual Studio, Code::Blocks, NetBeans, Eclipse, Visual Studio Code, CLion, CodeLite ve diğerleri gibi tümleşik geliştirme ortamları daha yaygın olarak kullanılır.

Her tümleşik geliştirme ortamı, programlama sırasında kullanılan uygulamalar kümesinden oluşuyor. Bu uygulama kümesi programlama diline bağlıdır ve mecburi olması gereken uygulamalar şunlardır: **metin düzenleyici** (İng. text editor), **derleyici** (İng. compiler), **hata ayıklayıcı** (İng.debugger) ve **bağlayıcı** (İng. linker).

Metin Düzenleyici

Metnin klavye aracılığıyla kaynak programına doğrudan girilmesini ve temel metin işleme işlemlerini gerçekleştirmesini sağlayan uygulamadır. Programları diskte kaydetme ve korunmuş programların yeniden açma,yani değiştirme olanaklarını sağlamaktadır.

Metin düzenleyicisi tüm standart metin giriş ve metin işleme araçlarına sahiptir. Metni düzenlerken, Microsoft Office Word, OpenOffice Writer ve diğerleri gibi ofis çalışması düzenleme programlarında neredeyse aynı komutları kullanabiliriz. Fare imlecini hareket ettirmek için de tüm komutlar geçerlidir.

Metin seçme, kopyalama, iletme ve silme komutları diğer metin düzenleyicilerde olduğu gibi aynıdır. Bu arada, genelde Edit olarak adlandırılan menüden komutlar kullanılıyor.

Birçok durumda, programda belirli bir metnin bulunması ve onun başka bir metinle değiştirilmesi gerekebilir. Bunlar, en sıkça Search olarak adlandırılan menüdeki komutlarla gerçekleştirilir.

Çevirici/Yorumlayıcı

Çevirici, (programlama dilinde yazılmış) kaynak programı bilgisayarın yürütmesi için yürütülebilir makine dili programına çeviren özel bir uygulamadır.

Çeviri yaparken çevirici, kaynak programının tamamını yürütülebilir programa çevirir. Bu program hemen çalıştırılabilir (çalışma belleğindeyse) veya dış bellekteki bir dosyaya kaydedilebilir ve yeniden çeviri yapmadan, daha sonra (veya hemen) çalıştırılabilir.

Yorumlayıcı, programı parça parça (komut komut), yani aynı anda birkaç komut birden çevirir ve yürütür. Programı yeniden yürütürürken, yorumlayıcı her komutu (veya komut grubunu) yeniden çevirmeli ve yürütmelidir.

Hem çeviricilerin hem yorumlayıcıların avantajları ve dezavantajları vardır. Günümüzde Java ve Visual Basic gibi bazı programlama dilleri çeviri ve

yorumlama kombinasyonu kullanırlar. Örneğin, Java çeviricisi kaynak programı **bayt kodu** (İng.bytecode) olarak bilinen ara koda çevirir. Bayt kodlu programı, herhangi bir bilgisayarda yorumlayıcı ile yürütülebilir.

Aynısı, C++'da bir programı çevirirken de yapılır, sadece kaynak programın ara koda çevirisi ön işleme aşaması olarak ele alınır. Ardından, programın ortaya çıkan şekli makine diline çevrilir, ***şekil 1.5.1.***

Çeviriciler ve yorumlayıcılar program hatalarını kontrol etmek için tasarlanmıştır. En yaygın olanları, komutlar yanlış yazıldığında ortaya çıkan dil (sözdizimi) hatalarıdır. Örneğin, close yerine cloce yazdıysak. Dil komutları yanlış kullanıldığında programsal (semantik) hataları da olabilir. Örneğin, do {... }while() yerine do {...}for() yazarsak, vb.

Hata Ayıklayıcı

Maalesef çeviriciler, sadece sözdizimsel veya semantik olarak programlama dilinin yanlış kullanımı nedeniyle programdaki hataları tespit edebilir. Meydana gelen başka bir hata türü ise özellikle yeni başlayan programcılar tarafından sıklıkla yapıldığı mantıksal hatalardır, yani yazdığımızda program amaçladığımız şeyi yapmaz. Bu hataların bulunması çok zordur ve bu nedenle programların yürütülme biçiminin, sonucu bilinen , önceden hazırlanmış girdi verileriyle kontrol edilmesi (test edilmesi) gerekir. Bu girdi ve çıktı verilerine **test-örnekleri** denir.

Hata ayıklayıcı, mantıksal hataları bulmaya yardımcı olan uygulamadır. Bu uygulama, test örnekleri üzerine programın yürütülmesini adım adım takip etmemizi ve böylece mantıksal hataları bulmamızı sağlar. Bu süreç sırasında hata ayıklayıcı, hatanın yerini ve türünü belirlemeye yardımcı olan bazı ara sonuçları da gösterebilir.

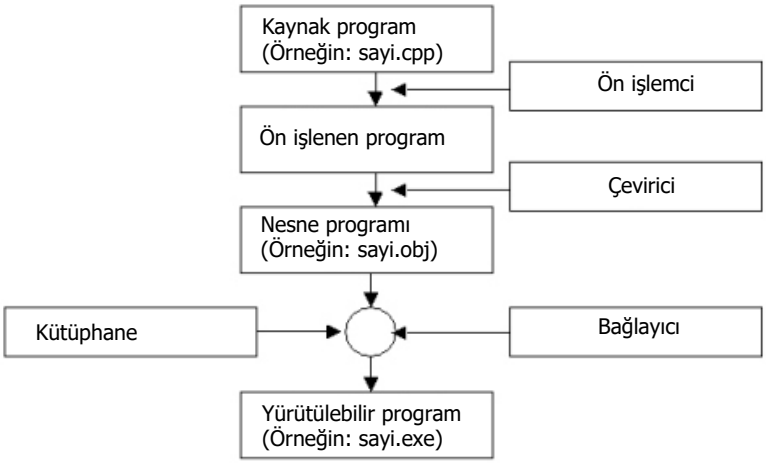
Bağlayıcı

Bazen program tek bir dosya olarak yazılamayacak kadar büyüktür. Ayrıca, farklı bölümler farklı programcılar tarafından yazılabilir. Bir programın bazı bölümleri başka bir programda kullanılabilir. Bu yüzden bu bölümler ayrı dosyalar olarak yazılmalı ve kaydedilmelidir. Bir programın birkaç bağımsız programdan oluşması gerekiyorsa, bunlar tek bir dosyaya, yani tek bir yürütülebilir programa bağlanarak birleştirilmelidir. Bu, **bağlayıcı** adı verilen özel bir uygulama ile yapılır.

Bağlayıcının diğer bir rolü de, kütüphane programlarını yazdığımız programla "bağlamak"tır. Her programlama dili, yazdığımız herhangi bir programda kullanılabilen hazır programların kütüphanelerine (veya başka adıyla modüllere) sahiptir. Örneğin, bir x gerçek sayısının karekökünü hesaplayan kütüphane programına birçok programlama dilinde sqrt(x) adı verilir. Programımızda y = sqrt(x) komutu varsa, bağlayıcı programımızı sqrt(x) programını içeren kütüphaneyle bağlar. (sqrt adı square root kısaltmasıdır).

Şekil 1.5.1'deki gösterilen şemada, ***Şekil 1.4.3*** a ve b'de verilen şemalardan daha gerçekçi kaynak programının, yürütülebilir programa çevirme şeması verilmiştir.

Tümleşik geliştirme ortamı, genellikle her gün birlikte çalıştığımız birçok uygulama kümesi gibi görünür. Çalışma alanı, menü çubuğu ve diğer ortak öğeler içeren temel pencereye sahiptir: pencereyi kapatan ve büyüklüğünü değiştiren düğmeler, pencereyi küçültme düğmeleri, vb. Çalışma alanında bireysel programların pencereleri açılır. Menü.



Şekil 1.5.1

çubuğunda, ortamın fonksiyonlarının çağrıldığı açılır menüler (komutlar listeleri) vardır: düzenleme menüsü, çeviri menüsü, bağlama menüsü, programları çalıştırma menüsü vb.Çoğu tümleşik programlama ortamında, genellikle programlama dilinin kurallarına ve programlama ortamının kendisiyle çalışmaya ilişkin genel bakış sağlayan yardım menüsü vardır.

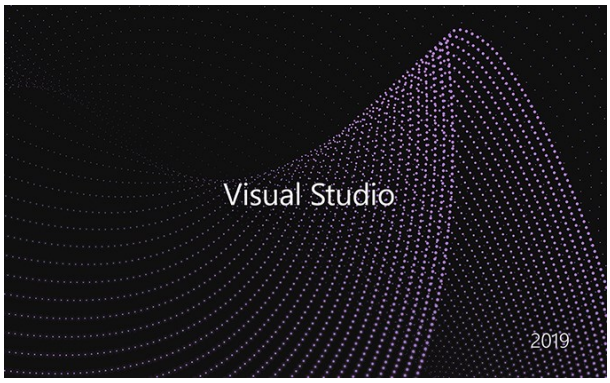
Devamda, Microsoft Visual Studio 2019 ve Code::Blocks tümleşik geliştirme ortamlarını açıklayacağız.

Microsoft Visual Studio 2019 Tümleşik Geliştirme Ortamı

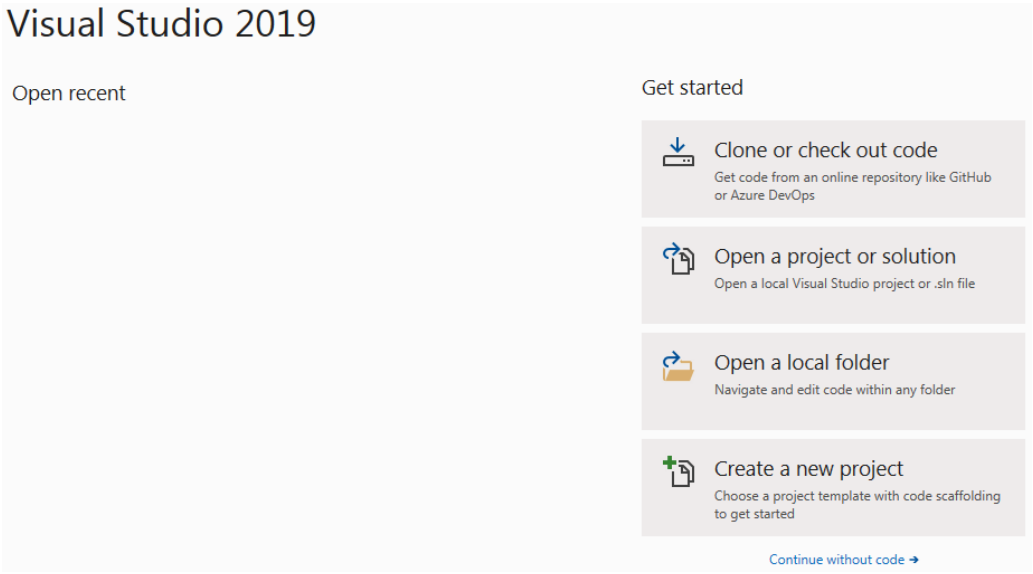
Microsoft Visual Studio 2019 (MVS 2019) çeşitli web sitelerinden indirilebilir.

Kurulum ve çalıştırmanın ardından, MVS 2019 genel görüntüsü elde ediliyor.

Ardından, MVS 2019'u başlatma şeklini seçme menüsü açılıyor, ***şekil 1.5.3.***



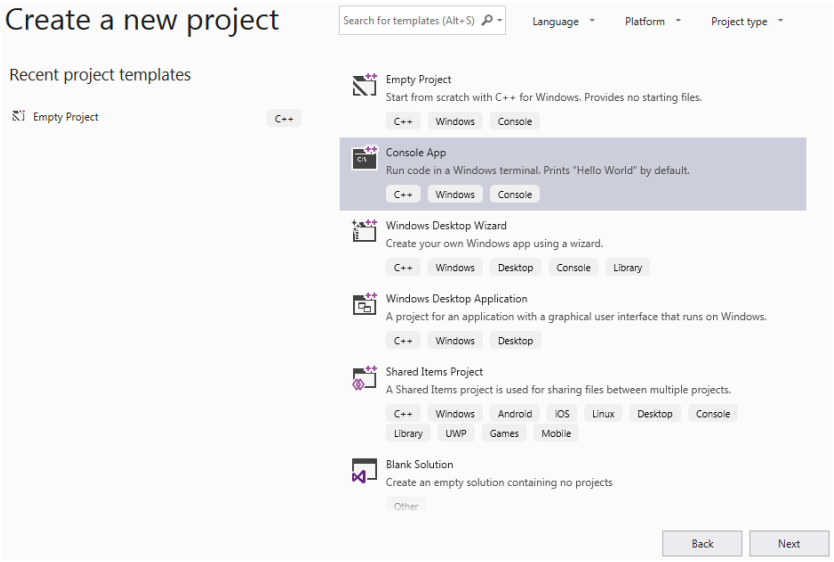
Şekil 1.5.2



Şekil 1.5.3

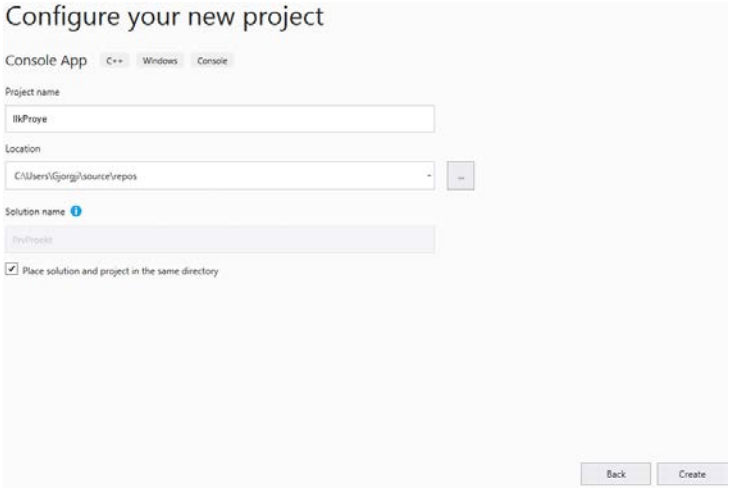
Yeni başlayanlar genellikle ikinci ve dördüncü başlama yöntemlerini seçerler.

Dördüncü yöntemin seçilmesiyle, yeni proje oluşturma menüsü açılır, ***şekil 1.5.4.*** Yeni başlayanlar için boş proje (Empty Project) veya C++ dilinde örnek program gösteren konsol uygulaması (Console App) oluşturmak en iyisidir. Ardından Next düğmesine tıklanır.



Şekil 1.5.4

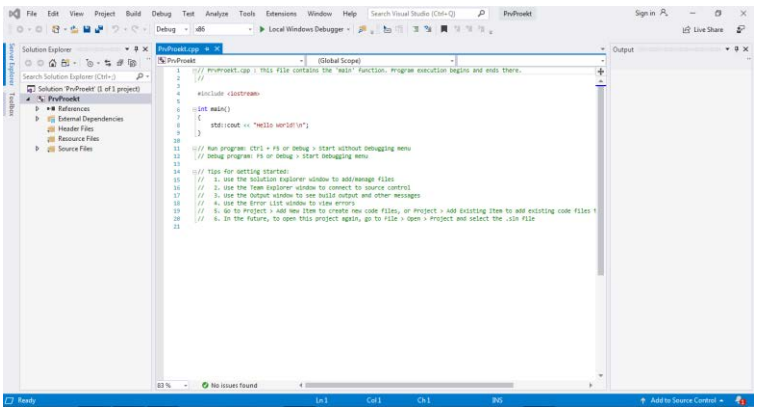
Yeni proje yapılandırma penceresi açılır, ***şekil 1.5.5.***



Şekil 1.5.5

İçinde yeni projenin adını gireceğiz, örneğin İlkProje ve Place solution and project in the same directory'in önündeki kutucuğu işaretleyeceğiz (üzerine tıklayarak).

Create düğmesine tıklandıktan sonra projenin oluşturulması başlar ve MVS 2019 tümleşik geliştirme ortamından pencere açılır, ***şekil 1.5.6.*** Bu pencerede, projeyle aynı adı altında C++ programı gösterilir. İlkProje.cpp. Programın üstünde ve altında, burada ele almayacağımız bir çok açıklama vardır.



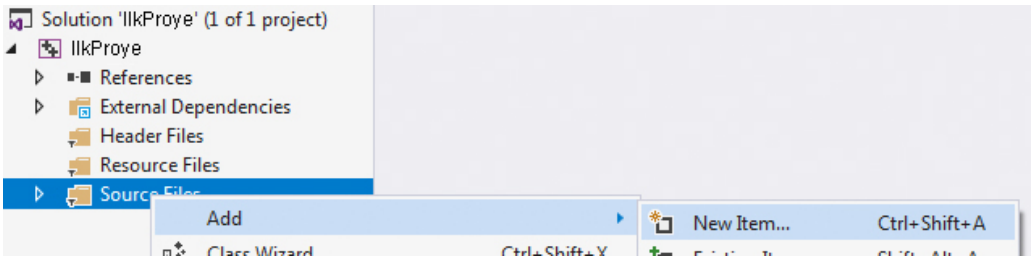
Şekil 1.5.6

Programın görünmesini istemiyorsak ***şekil 1.5.4'***deki menüden Empty Project'i seçmeliyiz.

İlkProje.cpp programı, programın adından sonra bulunan x işareti tıklayarak kapatılır.

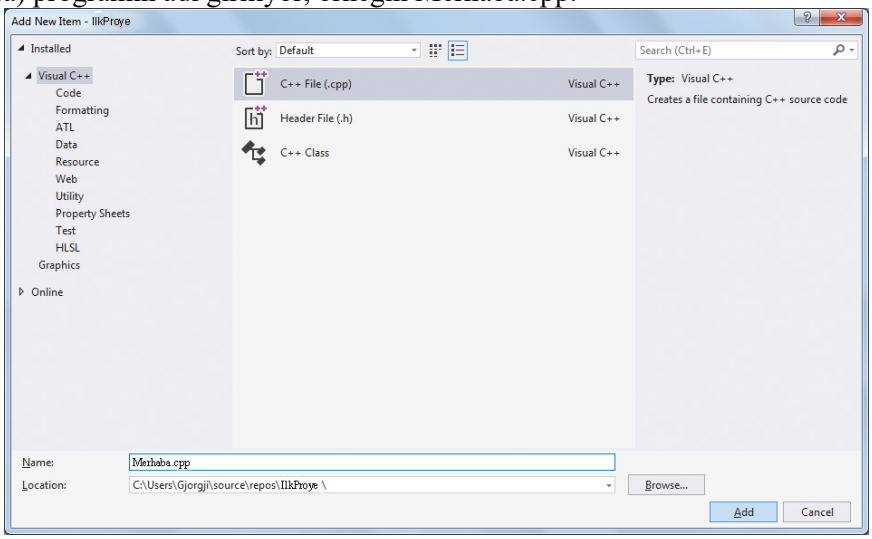
Solution Explorer penceresinde, MVS 2019 geliştirme ortamında oluşturulan projeler görüntülenir.Böylece, ***şekil 1.5.6'***da, alt dizinleriyle birlikte İlkProje projesinin dizini görünür.

Solution Explorer penceresinden Source Files alt dizinine sağ tıklayıp Add alt menüsünden New Item… komutunu seçerek (***şekil 1.5.7)*** Add New Item – İlkProje penceresi açılır, ***şekil 1.5.8.***



Şekil 1.5.7

Sağ çerçevede, C++ File (.cpp)'a tıklanır ve ardından Name alanına (alt kısımda) programın adı giriliyor, örneğin Merhaba.cpp.



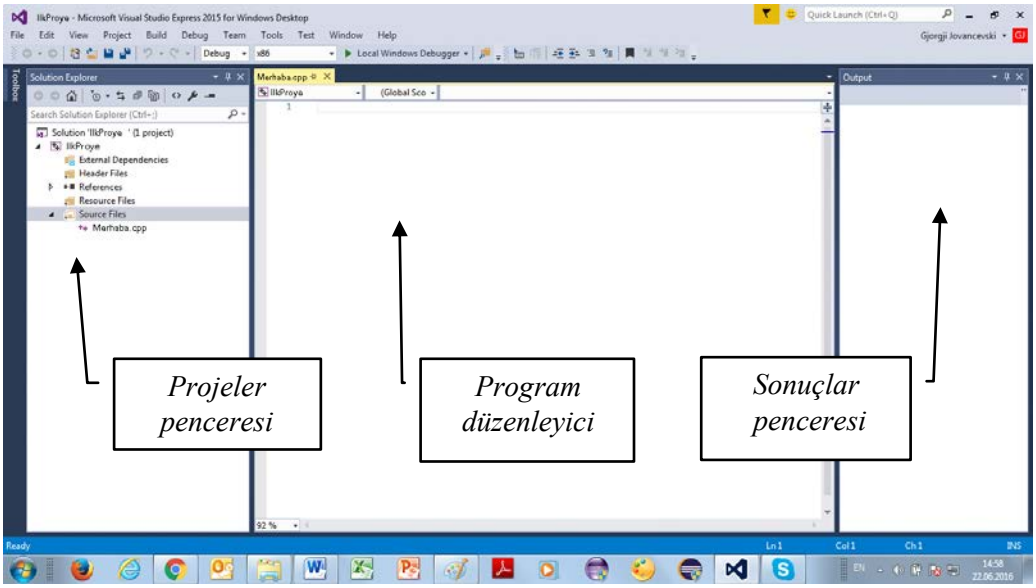
Şekil 1.5.8

Add düğmesine tıkladığınızda, Merhaba.cpp programını yazma ve düzenleme penceresi açılır, ***şekil 1.5.9.***

Ana pencere

Şekil 1.5.9'da ana pencere, yani MVS 2019'daki temel programlama ortamı gösterilmektedir. Ana pencere şu pencereleri içerir:

- Projeler penceresi (Solution Explorer).
- Program yazma ve düzenleme penceresi – düzenleyici.
- Sonuçları görüntüleme penceresi – çıktı penceresi (Output).



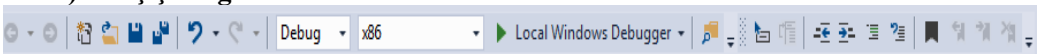
Şekil 1.5.9

Ana pencerenin temel elemanları şunlardır:

a) Menü çubuğu

Açılan menü çubuğu, pencerenin en üst kısmında, başlığın hemen altında bulunur. Menü çubuğunda şu menüler bulunmaktadır: File, Edit, View, Project, Build, Debug, Tools ve diğerleri. Her menü çevre komutları ve/veya diğer menüler içerir.

b) Araç çubuğu



Araç Çubukları (ortamda en yaygın kullanılan komutları başlatma düğmeleri), açılan menü çubuğunun hemen altında bulunur. Programlama ortamındaki araçlar aktif veya aktif değildir (soluk).

Yanlarında daha fazla komut içeren araçlar aşağıya doğru ok işaretine sahiptir ve üzerine tıklayarak açılan menü açılır. Aracın üzerine tıklarsak, aynı komutların verildiği pencere açılır.

İlk C++ Programı

Şekil 1.5.9' daki düzenleyicide Merhaba.cpp programını yazabiliriz ve düzenleyebiliriz, ***Şekil 1.5.10.***

Build alt menüsünden Compile komutu ile (veya CTRL + F7 ile) program çevrilir ve çıktı penceresinde (Output) programın başarıyla çevrilip çevrilmediği mesajı (***şekil 1.5.11***) yazılır.

Merhaba.cpp

IlkProye

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5
6         // Benim ilk programim Merhaba.cpp
7
8       cout << "Merhaba, nasilsiniz?" << endl;
9
10      system("pause");
11      return 0;
12  }
```

Şekil 1.5.10.

Output

Show output from: Build

```
1>------ Build started: Project: IlkProje, Configuration: Debug Win32 ------
1>Merhaba.cpp
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Şekil 1.5.11

Debug alt menüsünden Start Without Debugging komutu ile (veya Ctrl + F5 ile), program yürütülür ve sonuç ***şekil 1.5.12***'deki verilen pencerede gösteriliyor.

Şekil 1.5.12

Çıktı penceresini kapatmak için klavyedeki herhangi bir tuşa basılır.

Programı disk'te kaydetmek için, File alt menüsünden Save komutuna (veya Ctrl+s) basıyoruz. Programı diskte başka bir adla kaydetmek için File alt menüsünden Save As komutunu veriyoruz.

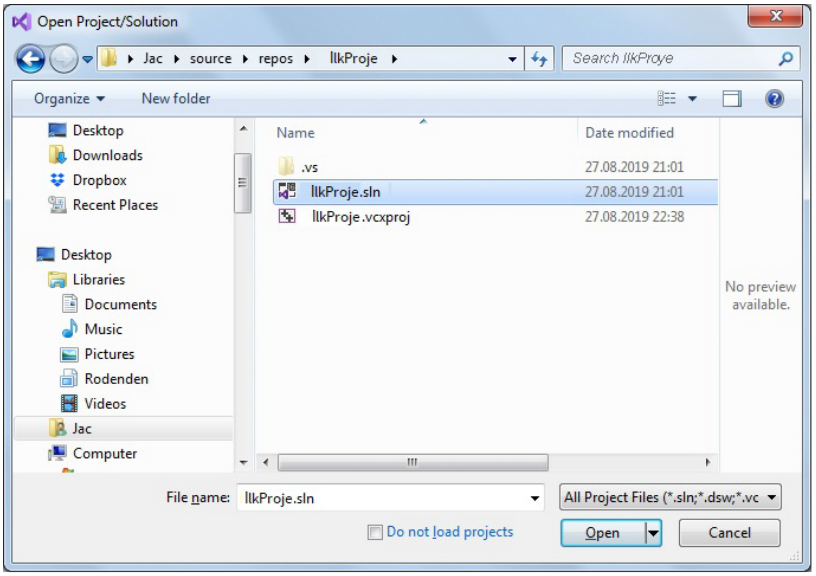
Bir projede dosya olarak yazdırılmış daha fazla program olabilir. Her dosyanın adı ve uzantısı olduğunu biliyoruz. C ++ dilinde yazılmış kaynak programların .cpp uzantısı vardır. Buna göre, C ++ 'daki bir programın adı: P1.cpp, progr.cpp, İlkProgram.cpp, benimProgram.cpp vb. olabilir.

Alıştırmalar

Alıştırma 1.5.1

a) Merhaba.cpp programını ***Şekil 1.5.15***'teki gibi olacak şekilde değiştirin.

MVS 2019'u ikinci kez başlatırken (ve ondan sonraki seferlerde), MVS 2019 sayfası görünür, ***şekil 1.5.3***. Hazır bir projeyi açmak istiyorsak Open'e project or solution linkine tıklamamız gerekiyor. Projeyi seçmek için ***şekil 1.5.13***'teki pencere açılacaktır.

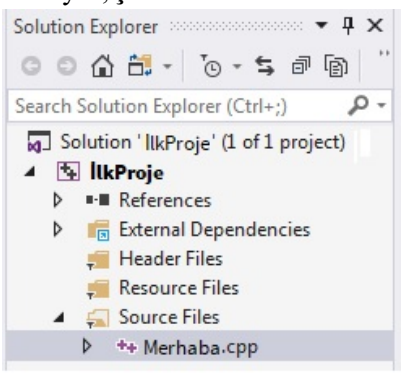


Şekil 1.5.13

Projelerden birine tıklandığında, örneğin İlkProje.sln, proje Solution Explorer penceresinde açılıyor ve içindeki tüm programlar gösteriliyor, ***şekil 1.5.14.***

Herhangi bir programın adına çift tıklayarak (örneğin, Merhaba.cpp'ye), o program düzenleyicide açılır. Düzenleyicide ***şekil 1.5.15***'te gösterildiği gibi Merhaba.cpp programını değiştiriyoruz.

b) Programı Save (veya Ctrl+S) komutuyla kaydedin.

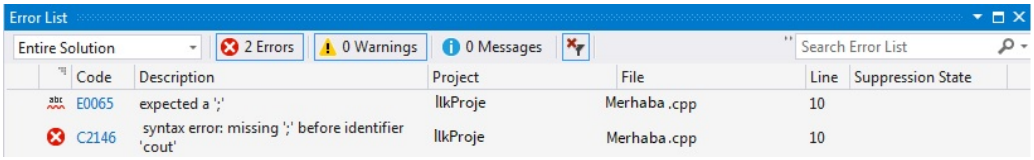


Şekil 1.5.14

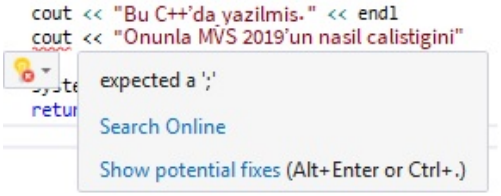
```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5
6       // Benim ilk programım Merhaba.cpp
7
8       cout << "Merhaba, nasilsiniz?" << endl;
9       cout << "bir program ornegidir" << endl;
10      cout << "Bu C++'da yazilmis      ." << endl;
11      cout << "Onunla MVS 2019'un nasil calistigini kontrol ediyoruz" << endl;
12
13      system( "pause" );
14      return 0;
15  }
```

Şekil 1.5.15

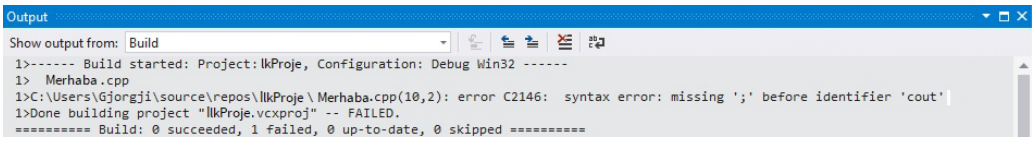
c) Programı yazarken bir komutta hata yaptıysanız, çeviri sırasında çıktı penceresinde hata mesajı karşımıza çıkacaktır. Örneğin, son komutun sonunda noktalı virgül koymayı unuttuysak, programı çevirirken hata listesi (İng.Error List) ve hata raporu içeren pencere açılacaktır.



Göründüğü gibi, hatanın oluştuğu komuttan sonraki komutun altı dalgalı çizgi ile çiziliyor. İmleci çizginin üzerine getirirsek, hata mesajı içeren çerçeve görüntülenecektir.



Çıktı penceresine (Output) hatanın türü ve konumu hakkında mesaj görüntülenir. Noktalı virgül koyduktan sonra program düzgün çalışacaktır.



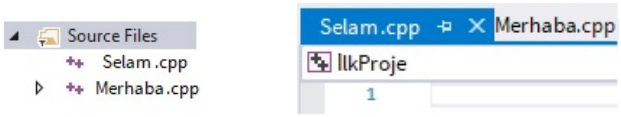
ç) Programda iki eğik çizgi // ile başlayan satırlar, programa yorum yazmak için kullanılır ve onlar

yürütülmez. Her programın başında ne yaptığına dair bir yorumun olması iyidir. Bunu biz de uygulayacağız.

Alıştırma 1.5.2

a) İlkProje projesinde başka bir program oluşturun, örneğin Selam.cpp adı altında.

Program, Merhaba.cpp programı ile aynı işlemle, yani Source Files' a sağ tı klayı p Add ▸ New İtem... seçeneğini seçerek oluşturulur. Açı lan pencerede (Add New İtem - İlkProje), C++ File(.cpp)' a tı klayı n ve Name alanı na (pencerenin alt kı smı nda) Selam.cpp adı girilir. En sonunda Add düğmesine tı klanı r. Solution Explorer penceresinde, Source Files alt dizininde yeni programı n adı gösterilecektir. Ayrı ca düzenleyicide yazma ve düzenleme için yeni program açı lacaktı r.



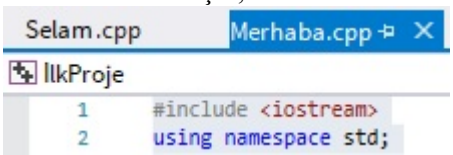
Düzenleyicide Merhaba.cpp programının da açık olduğunu görebiliriz.

Düzenleyicide bir programdan başka programa geçiş, programın adı üzerine tıklayarak yapılır.

Düzenleyicide açık olan bir programı kapatmak için, File alt menüsünden Close komutu verilir veya program adının sağında bulunan x işaretine tıklanır.

b) Merhaba.cpp programının başlığına tıkladıktan sonra program açılacaktır.

İlk iki komutu seçin,



onları Ctrl + C ile kopyalayın, Selam.cpp programının adı üzerine tıklayın ve Ctrl+V ile yapıştırın.

Enter ile bir boş satır bırakın , yorum ve mesaj yazdırma komutlarıyla birlikte ana fonkisyonu yazın, ***şekil 1.5.16.***

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       //Benim C++'da ikinci programim
6
7       cout << "Benim adim ..." << endl;
8       cout << "Benim soyadim  ..." << endl;
9       cout << "Benim doğum gunum ..." << endl;
10
11      system("pause");
12      return 0;
13  }
```

Şekil 1.5.16

Programı, Ctrl+S ile kaydedin, Ctrl+F7 ile derleyin ve Crtl+F5 ile yürütün.

Yürüttüğünüzde, aşağıdaki pencerede hata mesajının göründüğünü fark edeceksiniz.

[10]

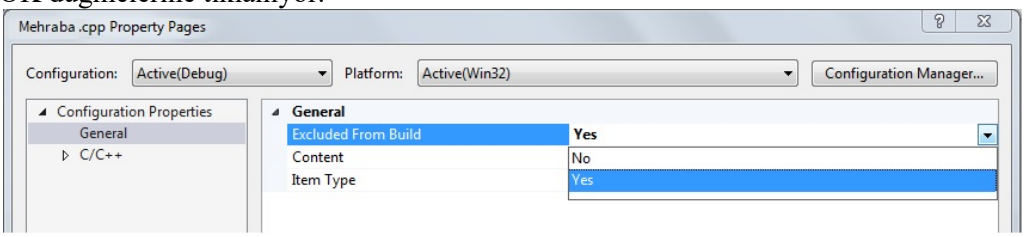
Mesaj, proje oluşturulurken bir hata oluştuğunu belirtir. Bir projede, yürütülebilir bir programa bağlanan birkaç program olabilir ("proje inşa ediliyor" diyoruz), ancak projenin yürütülmesini başlatan sadece bir program olmalıdır. Bu program main() fonksiyonunu içeren ve **uygulama** (İng. application) olarak adlandırılan programdır. Bizim İlkProje projemizde main() fonksiyonunu içeren iki program olduğu için yukarıdaki hata meydana geliyor.

Bu, farklı şekillerde çözülebilir ve en kolay çözümlerden biri, uygulama ve proje yürütmesini başlatmak istediğimiz program hariç, main () fonksiyonunu içeren tüm programları kapatmaktır.

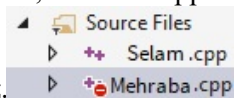
Merhaba.cpp programını kapatmamız için, üzerine sağ tıklayacağız ve açılan pencerede (Open) şu komutu vereceğiz:

[10] Yes'e tıklanırsa, son başarıyla yürütülen program yürütülecektir. No'ya tıklanırsa, son başarısız yürütülen program yürütülecektir.

Merhaba.cpp Property Pages penceresi açılacaktır ve General çerçevesindeki Excluded From Build menüsünde Yes seçiliyor. Ardından, sağ alt kısmında Apply ve OK düğmelerine tıklanıyor.



Solution Explorer penceresinde, Merhaba.cpp adı önünde ++ işaretlerinin

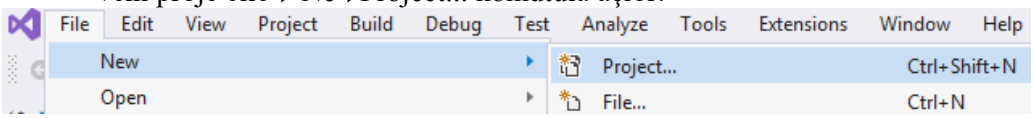


+ –’ ye değiştiğini fark edeceksiniz. Bu, Merhaba.cpp programının projeden kapatıldığı ve sadece Selam.cpp programının bir uygulama olduğu ve yürütülebileceği anlamına gelir.

Alıştırma 1.5.3

a) Yeni proje (örneğin, İkinciProje) ve içinde bir program oluşturun.

veni proje cile ▸ Ne ▸Project... komutula açlor.



Şekil 1.5.4’teki pencere açılacaktır ve Şekil 1.5.4 ve Şekil 1.5.5'te açıklandığı gibi Empty Project veya Console App bağlantısına tıklanıyor. Şekil 1.5.5'ten (alt tarafta) Name alanında projenin adı, örnek olarak İkinciProje yazılıyor. En sonunda, Create düğmesine tıklanıyor.

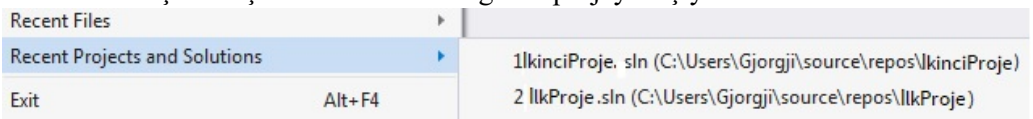
Solution Explorer penceresinde ikinci proje açılacaktır, ilki ise kapanacaktır.

Projede program, ***Alıştırma 1.5.2*** a’da açıklanan prosedüre göre açılıyor.

İlk projeye (veya herhangi başka bir açık projeye) geçmek istiyorsak, onu oluşturulan projeler listesinden aşağıdaki komutları kullanarak seçiyoruz:

File ▸ Recent Projects

Karşımıza çıkan listeden istediğimiz projeyi seçiyoruz.



Code::Blocks Tümleşik Geliştirme Ortamı

Code::Blocks tümleşik geliştirme ortamı http://www.codeblocks.org/ downloads internet sitesinde bulunabilir.

Sitenin merkezi kısmında üç bağlantı bulunmaktadır: **Download the binary release, Download the source code** ve **Retrieve source code from SVN**. En basit kurulum için, birinci bağlantı olan Download the binary release bağlantının seçilmesi öneriliyor.Bu bağlantıyı tıkladıktan sonra, bilgisayarınızın çalıştığı işletim sisteminin indirilmesi için sunulan Code::blocks ile yeni bir sayfa açılır. Yeni başlayanlar için MinGW setup içeren sürümünün indirilmesi önerilir. Bu ders kitabı yazarken, tüm Windows işletim sistemlerinin kullanıcıları için tasarlanan codeblocks- 20.03mingw-setup.exe bağlantısıdır. Code::Blocks'u indireceğiniz konuma tıkladıktan sonra (örneğin **Sourceforge.net**), yeni bir sayfa açılıyor, bu da sadece 5 saniye geçtikten sonra dosyayı kullanıcı tarafından seçilen bir konuma kaydetme seçeneği sunar. Dosyayı kaydettikten sonra yükleme talimatları izlenmelidir.

Başlatmak

Code::Blocks'un yüklenmesi sırasında, Start temel menüsünün Programs menüsünde, CodeBlocks adlı programlar grubu oluşturulur. Tümleşik programlama ortamını başlatmak için, bu gruptan CodeBlocks seçmeniz gerekir.

Ana Pencere

CodeBlocks'un ana penceresinde (***şekil 1.5.17***), programları kaydetme, arama, çevirme ve yürütme araçları, hata ayıklama araçları vb. bulunur. Devamın temel öğeleri açıklayacağız.

a) Menü Çubuğu

Menü çubuğu pencerenin en üst kısmında, hemen pencerenin başlığı altında bulunmaktadır. Menü çubuğunda şu menüler bulunmaktadır: File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, Plugins, DoxyBlocks, Settings ve Help. Her menü ortamdan komutlar ve/veya başka menüler içerir. Çoğunun anlamı daha geç incelenecektir.

b) Araç çubuğu

Araç Çubukları (ortamın en yaygın kullanılan komutlarını başlatma düğmeleri), açılan menü çubuğunun hemen altında bulunur.Code::Blocks programlama ortamındaki araçlar, anlamlarına bağlı olarak gruplara ayrılır.

Farklı araçlar gruplarının gösterilmesi ve gizlenmesi, View ▸ Toolbars seçeneğinin seçilmesiyle gerçekleştirilir. Her grubun kendi adı vardır ve adın önünde onaylama (✓) işareti varsa, grup ekranda görülür, eğer yoksa görülmez. Her araç, menü komutlarından bazısına karşılık olarak gelir.

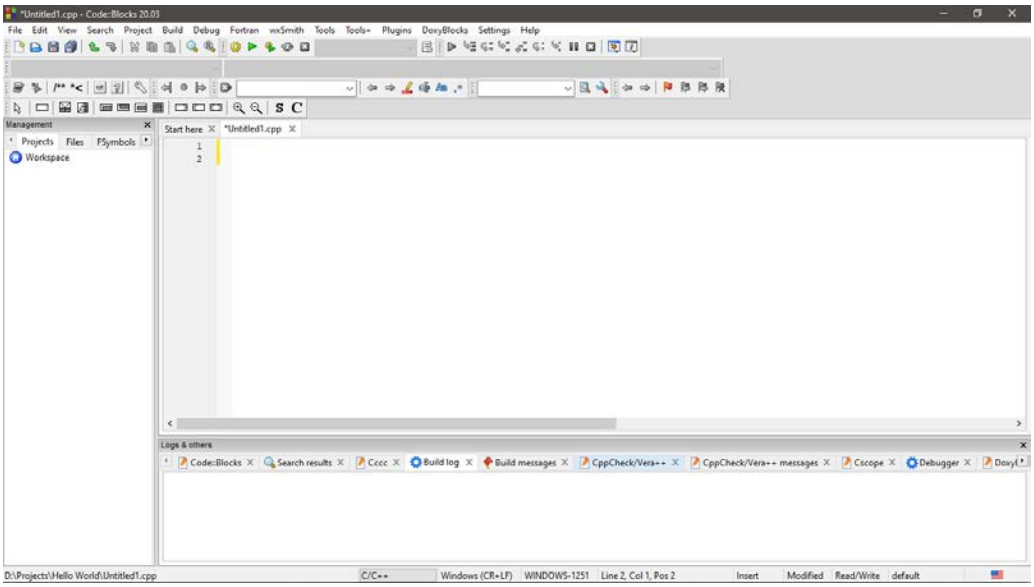
Ana pencerede ortamın uygulama pencereleri bulunur. Bunlardan en önemlileri şunlardır:

• Metin düzenleme penceresi – düzenleyici: Programcının belirli bir anda kaç programda çalıştığına bağlı olarak, içinde birkaç program olabilir. Programın adı, düzenleme penceresinin üst kısmında gösterilir. O anda aktif olan programın adı, diğer programların adlarından farklı olarak işaretlenir.

• Sonuçlar penceresi – çıktı penceresi: Bu pencerede programlar çeviri, bağlantı, başlatma vb. çalışmaları hakkında bilgi verilir. Çeviri sırasında hatalar tespit edilirse, hatalar bu pencerenin Build messages çerçevesinde gösterilir.

• Programın çalışmasını düzenleme penceresi:

Pencereler kapatılabilir, büyütülebilir veya küçültülebilir vb.



Şekil 1.5.17

Metin Düzenleyici

a) Metin girişi ve işleme

Metin düzenleyicisi, tüm standart metin giriş ve işleme araçlarını içerir. Metni düzenlerken, ofis işleri için metin düzenleme programlarındaki (Microsoft Office Word, OpenOffice Writer vb.) neredeyse aynı seçenekleri kullanabiliriz. Ayrıca, fare imleci için tüm hareket seçenekleri de geçerlidir.

b) Metin seçme, kopyalama ve taşıma

Bu işlemler, Edit menüsü komutları aracılığıyla yapılır.

c) Arama ve değiştirme

Çoğu durumda, program boyunca belirli bir metni aramak ve onu bulduktan sonra başka bir metinle değiştirmek gerekebilir. Bu, Search menüsünde bulunan Find, Replace vb. komutlarla gerçekleştirilir.

Programları derlemek ve yürütmek

Build menüsü, programları derlemek ve yürütmek için komutlar içerir. Pogram Build ▸ Build (veya Ctrl + F9) komutu ile derlenir. Sonuç olarak, kaynak program dosyasının bulunduğu klasörde, ilgili yürütülebilir program oluşturulur. Yürütülebilir program, Run komutu (veya F9) ile başlatılır.

Program yürütülürken, girdi verilerinin girildiği ve sonuçların görüntülendiği pencere açılır.

Ortam tarafından oluşturulan dosyalar

- .cpp uzantılı dosyalar kaynak programları içerir ve File ▸ Save veya Ctrl + S komutu çalıştırıldıktan sonra elde edilir.
- .exe uzantılı dosyalar, kaynak programına eşdeğer yürütülebilir programı içerir. Kaynak programın derlemesi sırasında oluşturulurlar.

Programın yürütülmesini izleme

Program yürütmenin izlenmesi, programdaki mantıksal hataları bulmak için çok önemlidir. Code::Blocks ortamı böyle izleme olanağını sağlamaktadır. Bu amaçla, programın yürütülmesini durdurmak, takip etmek istediğimiz verilerin içeriğini görüntülemek istediğimiz satırlar (kontrol noktaları olarak adlandırılan) metin içinde belirtilebilir. Programı izleme düzeninde çalıştırmaya

başladığımızda, kontrol noktaya gelene kadar normal çalışıyor ve sonra durduruluyor. O anda, programcılar gözlem için seçilen değişkenlerinin içeriğini gözden geçirebilirler. Ardından, program bir sonraki kontrol noktasına kadar yürütülmeye devam ediyor. Program izleme ile ilgili tüm komutlar Debug menüsünde bulunuyor.

Kontrol noktası, imleci programda seçilen satıra konumlandırarak ve Debug ▸Toggle Breakpoint (veya F5) komutunu vererek ayarlanır. Aynı zamanda satırın solunda kırmızı nokta koyulur. Nokta üzerine fare ile tıklanarak çıkartılır. Programın izlenmesi, Debug▸ Start komutunun verilmesiyle başlıyor. Program kontrol noktasına ulaşana kadar çalışır, o zaman durdurulur ve satırda sarı ok belirir. Oradan, programın izlenmesi menüden diğer komutları ile devam ediyor:

Debug – Next Step (sadece bir komut yürütülür, ardından sarı ok bir sonraki satıra geçiyor).

Continue (program bir sonraki kontrol noktasına kadar veya başka bir kontrol noktası yoksa sonuna kadar yürütülür).

Step into (sadece fonksiyonun çağrıldığı ve çağrılan fonksiyonun izlenmesi için gerekli olan komutlar için kullanılır).

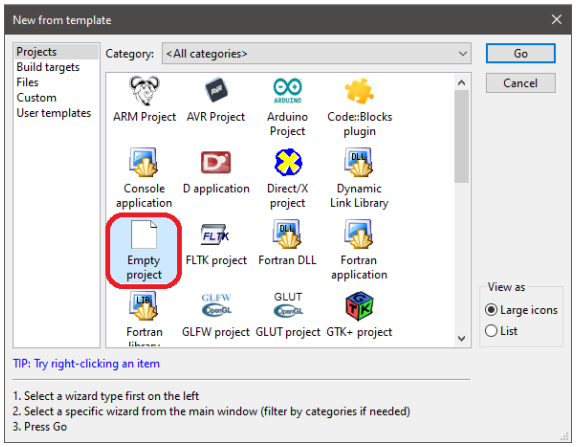
İzleme, Debug▸Stop Debugger komutuyla durdurulur.

Programcılar herhangi bir anda Debug▸Run to Cursor komutuyla (geçici) kontrol noktası ayarlamak için özel yol kullanabilirler. Komutu vermeden önce, imleç, programın yürütülmesini izlemeye devam etmek istediğimiz satırın başına yerleştirilmelidir.

Alıştırmalar

Alıştırma 1.5.4

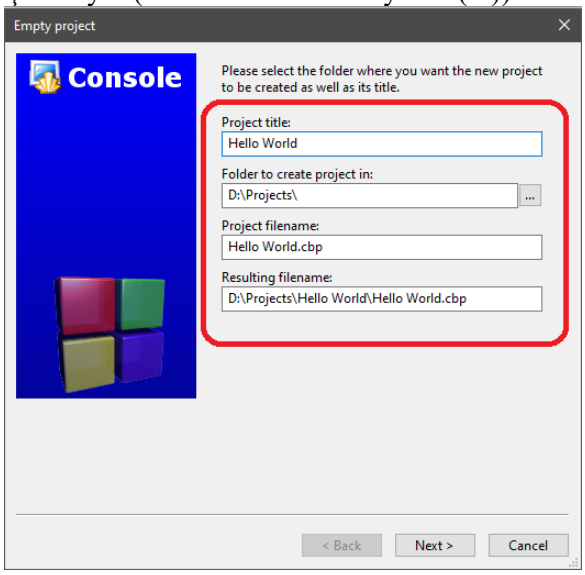
Code::Blocks'u başlatın. File►New▸Project▸EmptyProject ▸Go komutuyla yeni proje oluşturma seçin, ***şekil 1.5.18***.



Şekil 1.5.18

Bir sonraki pencerede (***şekil 1.5.19***) „Project Title", „Folder", „Project filename" ve „Resulting file Name" girin. Örneğin, projenin başlığı için İlkProje girin, geri kalan alanları ise olduğu gibi bırakın. Next düğmesine tıkladıktan sonra açılan pencerede ***(şekil 1.5.20),*** GNU GCC Compiler'ı seçin ve

"debug" ve "release" konfigürasyonlarını oluşturmak için aşağıdaki iki seçeneği işaretleyin (kutulu alanlara tıklayarak (✓)). Finish düğmesine tıklayın.

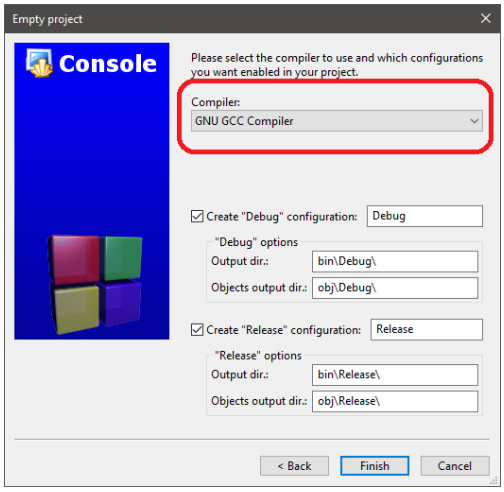


Şekil 1.5.19

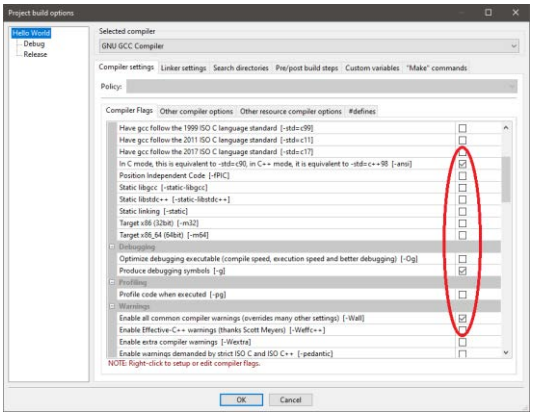
Projeye kaynak dosya ekleyin: File ▸ New ▸ File ▸ C/C++ Source. Ardından, projenin bulunduğu uygun klasöre dosya adını tam yol ile birlikte girin (örneğin, Alistirma1) ve "Add file to active project" seçeneğini açmayı unutmayın.

Her proje için aşağıdaki seçenekler ayarlanmalıdır: „Project ▸ Build Options.. ▸ Compiler Flags“.

Bununla, ilk programınızı yazmanız için hazır olan , (şimdilik) boş bir dosya içeren bir proje oluşturdunuz.



Şekil 1.5.20



Şekil 1.5.21

Alıştırma 1.5.5

a) **Alıştırma 1.5.4**'te açılan dosyada, aşağıdaki programı girin:

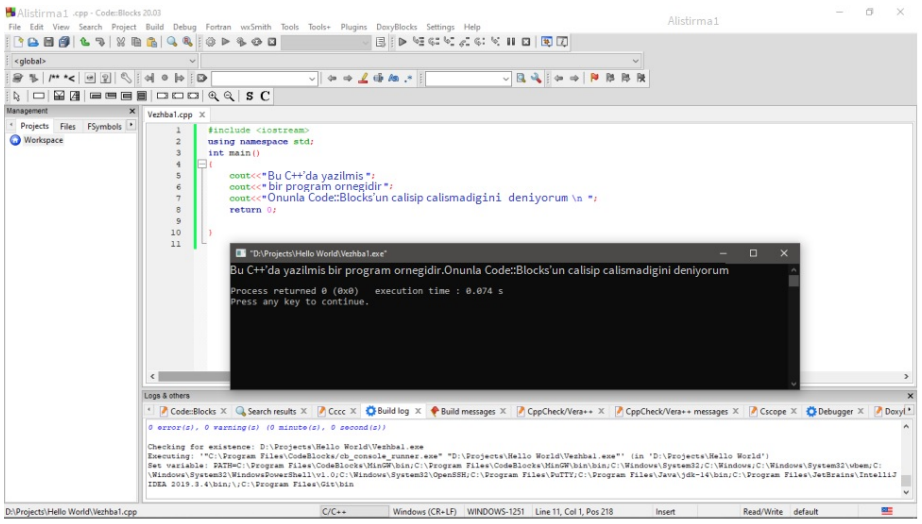
```
#include <iostream>
using namespace std;

int main()
{
        cout << "Bu C++'da yazilmis";
        cout << "bir program ornegidir.";
        cout << "Onunla Code::Blocks'un calisip calismadigini
        deniyorum"; return 0;
}
```

Şekil.1.5.22

b) Programı File ▸ Save, komutuyla kaydedin, örneğin;Alistirma1 adıyla. Vezhba1.

c) Alistirma1.cpp programını Ctrl + F9 derleyin.

ç) Alistirma1.cpp programını girerken a) altında verilen metinle ilgili bir şeyi kaçırdıysanız, hataları kaldırın ve Ctrl + S tuşlarına basarak değişiklikleri kaydedin. Ardından, F9 tuşuna basarak programı yürütün.



Şekil 1.5.23

Alıştırma 1.5.6

a) Yeni bir proje oluşturun ve önceki programı yeniden girin. Programı çalıştırın

b) Programda kasten hata yapın. Örneğin, yazdırma komutlarından birine cout yerine caut yazın. Şimdi programı çalıştırmayı deneyin. Ne oluyor? Hataları düzeltin ve programı yeniden yürütün.

c) Programın dördüncü satırını kopyalayıp beşinci satır olarak yapıştırın. Programı tekrar yürütün. Yürütmenin sonucu nedir?

Alıştırma 1.5.7

Yeni proje oluşturun, aşağıdaki programı yazın (***şekil 1.5.24***) ve yürütün:

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

int main()
{
	// Benim C++'da ikinci programim

	cout << "Benim adim... "  << endl;
	cout << "Benim soyadim... " << endl;
	cout << "Benim dogumgunum... " << endl;
	return 0;

}
```

Şekil 1.5.24

1.6 C+'ya Giriş

C++'nın Tarihçesi

1972'de, "AT&T Bell Labs" laboratuarlarında Dennis Ritchie tarafından C dili tasarlanmış. İçinde o zamanki programlama dillerinin sahip olmadığı çeşitli özellikler yerleştirilmiştir, yani:

- Çok alt düzey kaynaklara erişim sağlamak.
- Sistem programlama[11] için uygun olması.
- Farklı bilgisayar platformlarında çalışabilmesi.
- Farklı işletim sistemlerinde çalışmak.

1980'de C'ye **sınıflar** (İng.classes) ile çalışmak olanağı eklenmiş ve C'nin bu sürümüne "sınıflı C" adı verilmiş.

Bu sürümün daha da geliştirilmesi, yani onun 1983 yılında yükseltilmesi, C'nin bir yükseltmesi olduğu için C++ adlı yeni bir dile yol açmış. C++, **nesne yönelimli programlama** (İng. object-oriented programming) için tasarlanmış dildir.

C++, Bjarne Stroustrup tarafından tasarlanmış, amacı ise şunlardır:

- C'den daha iyi genel amaçlı bir dil oluşturmak,
- Soyut veri türlerini[12] (İng. abstract data types) desteklemek,
- Nesne yönelimli programlamayı desteklemek.

Günümüzde C++, çeşitli amaçlara yönelik kullanıcı uygulamalarından sistem araçlarına kadar değişen çeşitli türden programlar oluşturmak için en yaygın kullanılan diller arasında yer almaktadır.

C++ Dilinin Elemanları

C++ dili, izin verilen simgeler kümesi olan alfabeye sahiptir, yani:

- Alfabenin küçük harfleri: a – z.
- Alfabenin büyük harfleri: A – Z.
- Rakamlar: 0 – 9.
- Özel karakterler: ~ ! @ # $ % ^ & * ( ) – + = { } [ ] : ; '...' "..." < > ? /.

[11] Donanım seviyesinde programlama.

[12] Bunlar, gerçek dünyayı bir programa eşlemek için kullanılabilecek yapılardır. Örneğin bir programda öğrenci, daire, dikdörtgen vb. tanımlanabilir ve bunlarla uygun işlemleri yapılabilir, örneğin:öğrencinin başarısını, dikdörtgenin alanını hesaplamak vb. Soyut veri türleriyle nesne yönelimli programlamada çalışılır.

C++ alfabesinden, aşağıdaki türden olabilen kelimeler oluşuyor:

- Anahtar kelimeler (İng. keywords).
- Sayısal, sembolik ve metinsel sabitler.
- İsimler (tanımlayıcılar).
- Operatörler.

Ayırıcılar (separatörler), kelimeleri birbirinden ayırmak için kullanılan özel sembol türleridir. Bunlar:

- Boş yer (boşluk).
- Yeni satır.
- Sekme.

Tüm yüksek seviyeli programlama dilleri gibi, C++ da sözcük ve komut oluşturma kurallarını içeren kendi **dilbilgisine** sahiptir. C++ belirli sayıda **ayrılmış kelimeye** sahiptir. Bazılarının C++ için özel anlamı vardır ve bunlara **anahtar kelimeler** denir. Programcılar, **tanımlayıcılar** adı verilen belirli kurallara göre kelimeler oluşturabilirler. Ayrılmış kelimeler tanımlayıcı olamaz.

Kesin olarak tanımlanmış kelime ve sembol kombinasyonları ile dilin komutları oluşturulur. Bu arada **sözdizimsel kurallar** veya sadece **dilin söz dizimi** denilen katı kurallar kullanılır. Programı çevirirken, sözdizimSEL kuralları ile her komutun doğruluğu kontrol edilir.

Programdaki her komutun kesin olarak tanımlanmış anlamı ve kesin olarak tanımlanmış eylemlere neden olmalıdır. Komutların anlamı dilin **semantiği** olarak adlandırılır.

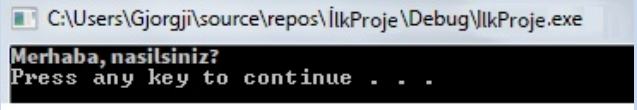
Örneğin, main() fonksiyonu için C++'daki sözdizimi kuralı şöyledir:

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

C++ Programı

***Şekil 1.6.1*'de, 1.5 Tümleşik Geliştirme Ortamı** bölümünde oluşturduğumuz Merhaba.cpp programı gösterilmiştir.

Program ekrana şunu görüntüler:



Programın her bir satırını açıklayacağız.

Satır 1: Yönerge

Programdaki girdi ve çıktı işlemleri için fonksiyonları içeren iostream kütüphanesini içeren yönerge.

Satır 2: Yönerge

Adların ad alanını (aralığını) oluşturuan yönerge.

Merhaba.cpp

İlkProje

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5
6       //Benim ilk programim Merhaba.cpp
7
8       cout << "Merhaba, nasilsiniz?" << endl;
9
10      system("pause");
11      return 0;
12  }
```

Şekil 1.6.1

Satır 3, 7 ve 9: Boş satır

Boş satırlar programı daha okunur hale getirmek için kullanılır, çevirici ise bunları ihmal eder.

Satır 4: Ana fonksiyon

C++ programları **fonksiyonlardan** oluşur[13]. Her C++ programının, main() adlı sadece bir ana fonksiyonu olmalıdır ve örnekte olduğu gibi bildirilmelidir. int kelimesi, main() fonskiyonunun tamsayı çıktısı olduğu anlamına gelir. Daha ileride, çıktı döndürmeyen fonksiyonlarını da bildirilebileceğini göreceğiz.

Satır 4: Büyük parantezlerin giriş kısmı – fonksiyonun başlangıcı

main() fonksiyonundan sonraki büyük parantezlerin giriş kısmı, **fonksiyon gövdesinin** başlangıcını gösteriyor. Her fonksiyonun gövdesi büyük parantezlerin çıkış kısmı ile biter, bizim örneğimizde 12. *Satırdaki* parantezlerin çıktı kısmıdır. Fonksiyon gövdesindeki satırlar (*Satır 6'dan Satır 11'e kadar*) sağa doğru çekilir ve buna girintileme (İng. indentation) denir.

Fonksiyonun başladığı büyük parantezlerin giriş kısmı, sıradaki *5.Satıra* da yerleştirilebilir.

Satır 6: Yorum

```
// Benim ilk programim Merhana.cpp
```

Programları yorumlamak iyi bir programlama pratiğidir. Bu yüzden, her programın başında programın ne yaptığı hakkında **yorum** koyulması önerilir. Yorumlar her foksiyonun başında da kullanışlıdır.

Verilen programda *Satır 6* çeviriciye hiçbir şey söylemez ve onu çevirmez. Bu nedenle, bu satıra yorum denir. Yorum, programı okuyan kişi için yararlı olan ve daha fazla görünürlük ve anlayış sağlayan metindir.

[13] Programlama dillerinde **fonksiyon**, yürütüldükten sonra sonuç veren bir programdır Örneğin, iki sayıyı çarpma, iki sayıyı bölme, bir sayının karekökünü bulma fonksiyonu vb.

Ayrıca yorumlar, programın düzenlemesi, değiştirmesi veya iyileştirmesi gereken diğer progrmacılar için de yararlıdır. Çoğu zaman programcı, daha geç kendisi baktığında daha kolay anlayabilmek için programda yorum yazıyor.

Her yorumun iki eğik çizgi // ile başladığını fark edebiliriz. Demek ki, //'den sonra satırın sonuna kadar yazılan her şey yorumdur. Programı derlerken, derleyici yorumları derlemez (çevirmez).

Satır 8: Yazdırma komutu.

```
cout << "Merhaba, nasilsiniz? " << endl;
```

`cout`	– yazdırma komutu
`<<`	– yazdırma operatörü
`"Merhaba, nasilsiniz?"`	– yazdırılan metin
`endl`	– mevcut satırı bitirme komutu
`;`	– komutun sonu işareti, yani komutların ayırıcısı.

Gördüğümüz gibi yazdırmak istediğimizi tırnak işareti içine koyuyoruz. Çoğu zaman tırnak işaretleri içine **karakter dizisi** temsil eden ve **dize**[14] (İng. string) olarak adlandırılan metin alınır.

Satır 10: Durdurma komutu

Çıktılar penceresini kapatmadan önce durdurma komutu. Bu komut zorunlu değildir, ancak geçmiş sonuçları görmek istiyorsak yararlıdır.

Satır 11: Değer dönme komutu

Dönen değer 0 ise (örneğimizde olduğu gibi), program doğru şekilde yürütülmüş anlamına gelir. Başka bir değer (1, 2...) döndürülürse, program çalışırken bir şeylerin yanlış olduğu demektir. Biz **return 0** komutunu kullanacağız.

Satır 12: main() fonksiyonunun kapanış parantezlerinin çıktı kısmı

Satır 1 ve satır 2 her C++ programına dahil edilmelidir.

main() fonksiyonunun standart biçimi vardır:

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

[14] Dizelerin ne olduğunu ve nasıl kullanıldığını daha sonra açıklayacağız.

C++ programları herhangi bir metin düzenleyici ile yazılabilir. Ardından, tümleşik geliştirme ortamına bağlı olarak tercüme edilir ve yürütülür.

Yazdırma komutu

Dizeleri yazdırma komutunun şu şekilde olduğunu söyledik:

```
cout << "Merhaba, nasilsiniz?" << endl;
```

Bu nedenle, ekranda başka bir şey yazılmasını istiyorsak, ***şekil 1.6.1'***deki programda Satır 8'deki komutun hemen altına, ***şekil 1.6.2***'de yapıldığı gibi, ***şekil 1.5.15*** ile aynı şekilde birkaç komut ekleyeceğiz.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5
6       // Benim ilk programim Merhaba.cpp
7
8       cout << "Merhaba, nasilsiniz?" << endl;
9       cout << "Bu C++'da yazilmis " << endl;
10      cout << "bir program ornegidir ." << endl;
11      cout << "Onunla MVS 2019'un nasil calistigini kontrol ediyoruz " << endl;
12
13      system( "pause" );
14      return 0;
15  }
```

Şekil 1.6.2

Programı çalıştırdıktan sonra aşağıdaki çıktıyı elde edeceğiz:

```
Merhaba, nasilsiniz?
Bu C++'da yazilmis
bir program ornegidir
Onunla MVS 2019'un nasil calistigini kontrol ediyoruz
Press any key to continue . . .
```

Komutlardaki her dizenin yeni satırda yazdırıldığını görüyoruz. Bu, cout komutunun sonuna endl[15] manipülatörü yerleştirilerek elde edilir.

Bu, sonunda endl manipülatörü bulunan yazdırma komutundan sonraki komutun yeni bir satırda (yeni sırada) yazdırılacağı anlamına gelir.

Aynı satırın devamında yazdırmak için, önceki komutun sonunda endl olmamalıdır.

Örneğin, ***şekil 1.6.2***'deki, satır 9'da komutun sonunda endl yoksa, şu yazdırılacaktır:

[15] C++'da veri girişini ve çıktısını kontrol etmek için çeşitli manipülatörler (yardımcı fonksiyonlar) vardır.

```
Merhaba, nasilsiniz?
Bu C++'da yazilmisbir program ornegidir .
Onunla MVS 2019'un nasil calistigini kontrol ediyoruz
Press any key to continue . . .
```

Görüldüğü gibi ikinci komutun son sözcüğü ile üçüncü komutun ilk sözcüğü birleştirilmiştir. Bunları ayırmak için, ikinci komuttaki dizenin sonunda veya üçüncü komuttaki dizenin başında boş yer bırakılmalıdır.

8, 9, 10 ve 11. satırlardaki komutlara bazen **ifadeler** de denir. Her ifadenin noktalı virgülle (;) bittiğini fark edebilirsiniz.

Yazdırma komutunun cout ("siaut" olarak okunur) sözcüğüyle başladığını, ardından **ekleme operatörü** (İng. insertion operator) olarak adlandırılan yazdırma operatörü << ve yazdırılacak ifadenin yazıldığını görüyoruz. Örneğimizde ifade, bilgisayarın standart çıkış cihazına, genellikle ekrana yazdırılan karakterler dizisidir (dize).

Yazdırma komutunun genel şekli şöyledir:

```
cout << ifade ;
```

<< operatörü, sol işleneni cout kelimesi ve sağ işleneni yazdırılması gerek ifade olan ikili operatördür.

<< operatörü aynı komutta birden çok kez kullanılabilir. Örneğin:

```
cout << "Merhaba, nasilsiniz? " << "Bu bir program ornegidir." << endl;
```

Çıktı olarak, iki dize aynı satırda yazdırılacaktır.

```
Merhaba, nasilsini? Bu bir program ornegidir
```

Bir satırda birden fazla yazdırma komutu yazılabilir ve birbirinden noktalı virgül işaretiyle ayrılmış olacaktır. Örneğin, ***şekil 1.6.2***'deki programın 9. ve 10. satırlarındaki komutları tek satırda yazarsak:

```
cout << Bu C++'da yazilmis. "; cout << "bir program ornegidir."
<< endl;
```

çıktı hiç değişmeden aynı kalacaktır.

Şekil 1.6.3'te gösterilmiş olduğu gibi, tek bir yazdırma komutuyla birden çok dize yazdırabiliriz. Ayrıca komut uzunsa << operatöründen önce veya sonra Enter tuşuna basarak bir sonraki satıra taşıyabiliriz.

Programı çalıştırmanın sonucu şöyle olacaktır:

```
Kales Angya kitabinin yazari Stale Popov'dur
veya, 2 komutla
Kales Angya kitabinin yazari Stale Popov'dur
veya, daha fazla komutla
Kales Angya kitabinin yazari Stale Popov'dur
Press any key to continue . . .
```

```
1     #include <iostream>
2     using namespace std;
3
4     int main() {     //Dizeler yazdirmak
5
6         cout << "Kales Angya" << " " << "kitabinin yazari"
7             << " Stale Popov'dur " << endl;
8         cout << "veya, 2 komutla " << endl;
9         cout << "Kales Angya " << " " << "kitabinin yazari ";
10        cout << "Stale Popov'dur " << endl;
11        cout << "veya, daha fazla komutla " << endl;
12        cout << "kitabinin yazari ";
13        cout << " ";
14        cout << "Kales Angya";
15        cout << "Stale Popov" << endl;
16        cout << "dur ";
17
18        system( "pause" );
19        return 0;
20    }
```

Şekil 1.6.3

Kitap adları tırnak işaretleri içinde yazıldığı için tırnak işaretleri "..." yazıldığında önüne ters eğik çizgi, yani \"..." konulur. "Kales Angja" dizesi "\"Kales Angja\"" şeklinde yazılmalıdır.

Ayrıca boş bir satır yazdırmak için şu komut verilir:

```
cout << endl;
```

Programda yapılan bu iki değişiklikle çıktı şöyle olacaktır:

```
Kales Angya" kitabinin yazari Stale Popov'dur
veya, 2 komutla
Kales Angya" kitabinin yazari Stale Popov'dur
veya, daha fazla komutla
Kales Angya" kitabinin yazari Stale Popov'dur

Press any key to continue . . .
```

Sıkça, daha uzun bir dizeyi birden çok satıra yazdırmamız gerekir. Bu amaçla, sonraki satırda yazdırmayı sağlayan \n özel dizgesi kullanılır. Örneğin, aşağıdaki dizeyi yazdırmak istiyorsak:

```
"Kales Angya"
kitabinin yazari
Stale Popov
'dur
```

bunu tek bit komutla yapabiliriz:

```
cout << "\"Kales Angya\"\n “kitabinin yazari \nStale Popov \n ’dur” << endl;
```

Komutta, tırnak işaretleri yazdırmak için \" dizgesi ve yeni satıra geçmek ve yazdırmak için \n dizgesi kullanılır.

C++'da bir çok kaçış dizgeleri (İng. escape sequences) olarak adlandırılan dizgeler vardır.

Birkaçını listeleyelim:

\n Yeni satıra geçiş.
\" Tırnak işaretleri yazdırmak "...".
\' Yarı tırnak işaretleri yazdırmak '...'
\t Sonraki sekme posizyonunda geçiş.
\\ Ters eğik çizgi yazdırmak \.

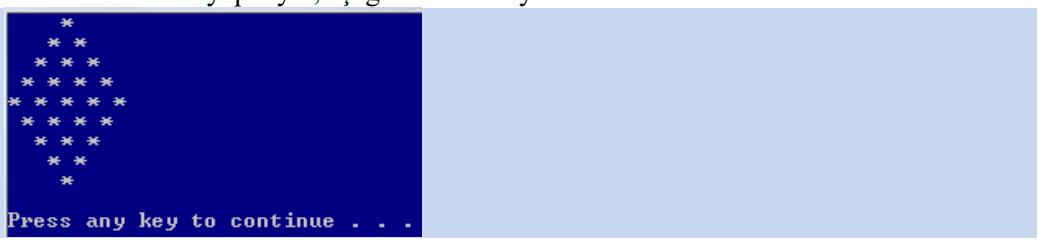
Şekil 1.6.3'teki programda yazdırma komutları arka arkaya dizilmiştir. Bilgisayar tarafından da bu şekilde yürütülecek, önce birinci, sonra ikinci, sonra üçüncü vb. yani komutlar, dizilmiş edildikleri sırayla yürütülecektir. Ardışık komutların böyle yapısına **sıralı yapı** veya **dizge** (İng.sequence) denir.

Örneğin, aşağıdaki programla

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

Şekil 1.6.4

komutların sıralı yapısıyla, aşağıdaki resim yazdırılacaktır:



Programda system („Color 17“) komutunu ekledik; bu komutla çıktının ekranda daha güzel görünümü için arka plan rengi mavi (1 numara) ve metin rengi beyaz (7 numara) ayarlanır. (Okuyucu, ilk sayı arka plan rengi ve

ikinci sayı metin rengi olacak şekilde bu komutu deneyebilir. Sayılar, ilk 16 onaltılık sayının herhangi bir kombinasyonu olabilir: 0, 1, 2, 3, 4, 5 , 6, 7, 8, 9, A, B, C, D, E, F).

Örneklerimizde son 4 komutu sürekli kullanacağız.

Alıştırma Ödevleri

1. Şekil 1.6.4'teki programda,
```
system("Color 17");
```
komutunda, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E,F olabilen arka plan renk numaralarını (programda 1 olan) ve metin renk numaralarını (programda 7 olan) değiştirin . En beğendiğiniz kombinasyonu seçiniz.
2. Ekranda aşağıdaki çıktıyı gösterecek program yazınız:
```
    *
   * *
  *   *
 *     *
* * * * *
```
3. Ekrandaki * işaretiyle veya başka bir işaretle isminizi yazacağınız program yazınız.
4. Adınızı ve soyadınızı yazdıracak program yazılsın ve ikinci satırda yaşınız yazdırılsın.
5. Önceki programı, kaç yaşında olduğunuzun yerine kaç aylık olduğunuzu yazdıracak şekilde düzeltin. Örneğin, Mayıs 2003'te doğduysanız ve şimdi Ekim 2020 ise, o zaman (17 yıl) x (12 ay) + (10 (Ekim) – 5 (Mayıs)) aylıksınız.
6. 1 234'ü 5 678 ile çarpma sonucunu yazdıracak program yazılsın. Hesap makinesi kullanarak doğru sonucu alıp almadığınızı kontrol edin.
7. 123 456 ile 7 891 011'in çarpma sonucunu ekrana yazdıracak program yazılsın. Sonuç olarak ne elde ettiniz? Bu doğru sonuç mu?

Büyüklükler ve Veriler

Algoritmaların veri işleme prosedürleri olduğunu, yani girdi verilerini kesin olarak belirlenmiş bir sırayla işlendiğini ve çıktı sonuçların elde edildiğini söyledik. Algoritmalarla hangi verilerin işlenebileceği sorusu ortaya çıkıyor. Bunun için, verilerin ne olduğunu görelim.

Büyüklük, belirli bir nesnenin veya olayın bazı özelliğinin ölçüsüdür. Örneğin, aşağıdaki özelliklerin kendi büyüklükleri vardır: uzunluk, hacim, kütle, renk, yaş, hız vb.

Büyüklükler, bir kümeden ölçü veya değerlerle ifade edilen değerlere sahiptir. Örneğin: cm

uzunluk ölçüsüdür, m^3 hacim ölçüsüdür, kg kütle ölçüsüdür, yıl yaş ölçüsüdür, m/sek hız ölçüsüdür, vb.

Büyüklüklerin ifade edildiği değerlere **veri** denir. Demek ki, verilerin değerleri vardır. Örneğin bir kişinin adım uzunluğu 70 cm'dir, kenarı 2 cm olan küpün hacmi 8 cm3'tür, bilişim profesörünün yaşı 47 vs. Veriler algoritmalarla, yani programlarla işlenir.

Sabitler ve Değişkenler

Çoğu durumda veriler, yani bir büyüklüğün değeri değişmez. Örneğin $\pi = 3.14$ sayısının değeri sabittir ve bu nedenle **sabit** (İng.constant) olduğunu diyoruz. Sabit değerler **değişmezler** (İng. literals)[16] olarak adlandırılır. Her programlama dilinde, programlarda sabit olarak kullanılabilen yerleşik değişmezler bulunur.

Örneğin, C++ programlama dilindeki π sabiti, PI olarak yazılabilir ve aşağıdaki gibi formüllerde kullanılabilir[17]:

```
cevre = 2 * r * PI;        alan = r * r * PI;        // * carpma isaretidir
```

Ancak programcılar programlarda kendi sabitlerini atayabilir (adlandırabilir). Örneğin, programda ülkemizin adı kullanılıyorsa, o zaman ulke adıyla ve "Makedonya"18 değeri ile sabit ayarlanacaktır. Veya, 12 değerini veya başka bir değeri alabilen aylar sabiti kullanılabilir. Programcılar tarafından tanımlanıp isimlendirilen ve değer atanabilen bu tür sabitlere **adlandırılmış sabitler** (İng.named constants) denir.

Sabitlerden farklı olarak, bazı büyüklüklerin veri değeri değişir. Örneğin, bir kişinin yaşı, günün saati, bir arabanın hızı vb. farklı değerler alabilir ve bu yüzden bu büyüklüklerin verilerinin değişken olduğunu söyleriz. Kardeşimin yaşı 10, benim yaşım 17, köpeğin yaşı 5 vs diyoruz. Demek ki 10, 17 ve 5 yaş verisinin değerleridir. Dolayısıyla yaş, değişken değerler alabilen veridir. Değişik değerler alabilen bu tür verilere **değişken** (İng. variables) denir.

Verileri bilgisayar yardımıyla işlerken, sabitlerin ve değişkenlerin kendi adları vardır.

---

[16] Programlarda çeşitli değişmezler yani verilerin sabit değerlerini kullanabiliriz. Örneğin: Dünya ivmesi, vatandaşın kimlik numarası, bir şehrin adı vb.

[17] C++'da sabitler büyük harflerle yazılır (tüm harfler büyüktür).

[18] Sabit bilgi metin ise, tırnak işaretleri içine alınır.

Değerlerinin girilmesi ve işlenirken hatırlanması gerekir. Örneğin: sayıPi, sayıE, yaz, zaman, hiz, renk, vb.

Veri sabitse, o zaman bellek konumunda verilen ad altındaki değer (örneğin, sayıPi), programın yürütülmesi sırasında değişmez.

Diğer taraftan veri değişken ise, değişkenin adının (örneğin: yas) yazılmış olduğu bellek konumunun değeri değişir. Örneğin; yas = 17, yas = 20, vb. yas veri değerinin değişken olduğunu söylüyoruz ve bundan ötürü yas verisi , "değişken" olarak adlandırılıyor.

Matematiksel formüller genellikle değişkenler içerir. y = f(x) fonksiyonunda x ve y'nin değişken olduğunu söylüyoruz çünkü x'in farklı değeri için farklı y değeri elde ediliyor. Veya kenarları a ve b olan bir dikdörtgenin alanını hesaplama formülünde (P = a * b), a ve b kenarları değişkendir. Bazen, formüllerde sabitler de dahil edilir. Örneğin, bir dairenin alanı için formül S = r * r * PI'dir. Formülde sayı da sabit olarak bulunabilir. Örneğin, V = 4 * r * r * r * PI / 3.

Programlama dilleri ile aşağıdaki şekilde ifade edilen farklı veriler işlenebilir:

- Tam sayılar (… –2, –1, 0, 1, 2, 3…),
- Gerçek sayılar (… –2,34… –1,89… 0… 0,573… 123,45…),
- Karakterler19 ('?', '*', '@', '7', '+'…),
- Metinler20 ("Makedonya", "21-ci yuzyildayiz", "Bilisim"...),
- Görüntüler, ses vb.

Bunların farklı **türden** (İng. type) veriler olduğunu diyoruz. Verinin türü dışında, **adı** ve **değeri** vardır. Bu yüzden, verilerin aşağıdaki özelliklere sahip olduğunu belirtiyoruz:

- Adı.
- Türü.
- Değeri.

Değişkenlere verilen değere göre, belirli türden olduklarını diyoruz. Örneğin, yas değişkeni (yıl olarak ifade edilirse) tamsayı türündendir, çünkü yalnızca tam sayı değerleri alabilir: yas = 18, yas = 10, yas = 123, vb. Buna benzer şekilde, alan (dairenin alanı) değişkeni ondalık sayıların değerlerini aldığı için gerçek türdendir, alana = (3^2 · PI = 9 · 3.14 = 28.2735). Isaret = '+', karakter türü değişkenidir (yarı tırnak işareti içine alınmış herhangi karakter değeri alabilir), metin türü değişkeni (tırnak işaretleri içindeki herhangi bir metin değeri alabilir) e `ulke = "Makedonya daimdir"` vb. . .

19 Programlama dillerinde karakterler yarı tırnak işareti ile belirtilir.
20 Programlama dillerinde metinler tırnak işareti ile belirtilir.

Verilenin Adları

C++ ' daki verilerin adları (tanımlayıcılar[21]) şunlar olabilir:

- Anahtar kelimeler veya ayrılmış kelimeler[22].
- Kullanıcı tanımlı adlar.

Ayrılmış kelimelerin özel anlamı ve uygulaması vardır. Bunlar:
const, double, float, int, struct, unsigned, break, continue, else, for ve diğerleridir.

Kullanıcı tanımlı adlar, kullanıcılar (programcılar) tarafından aşağıdaki kurallara uygun olarak oluşturulur:

• Ad, harf (A – Z, a – z) veya alt çizgi () ile başlar.
• Ad uzunluğu sınırsızdır.
• Küçük ve büyük harfler farklıdır. Örneğin, ben ve Ben farklı adlardır.
• Ad, alfabenin küçük ve büyük harflerini, 0 – 9 arasındaki sayıları ve alt çizgi içerebilir.
• Ad boşluk içeremez.
• Ad, !, @, #, $, ^ gibi özel karakter içeremez.
• Ad, anahtar kelime veya ayrılmış kelime olamaz.

Örneğin, geçerli veri adları şunlardır:

```
a, A1, veSen, i2j3,_ad, yaricap, R, sayı_120, _char, SenOku.
```

Geçersiz veri adları şunlardır:

```
5_li        - Ad rakamla başlayamaz.
do$lar      - Ad özel karakter içeremez.
i 2         - Ad içinde boşluk kullanılamaz.
char        - Ayrılmış kelime tanımlayıcı olamaz.
```

Veri adları için ayrılmış kelimeler , ayrılmış kelimelere benzer veya ayrılmış kelimeler içeren kelimeler önerilmez.

Örneğin:

```
CHAR        – char ayrılmış kelimeye benzerdir.
_float      – float ayrılmış kelimeyi içerir.
__          – İki alt çizgi.
```

[21] Tanımlayıcı (İng. identifier), kimliğin belirlenmesi anlamına gelen tanımlama kelimesinden gelmektedir. Veri adları, verilerin ne olduğunu tanımlamak için kullanılır ve bu nedenle tanımlayıcılar olarak adlandırılır.

[22] Anahtar kelimelerin dilde özel anlamı vardır. Ayrılmış kelimeler dil için ayrılmıştır ve tanımlayıcı olarak kullanılamazlar.

Veri Türleri

Veri **türü**, üzerinde sonlu işlem kümesinin tanımlandığı sonlu bir veri kümesini temsil etmektedir[23]. Her veri türünün değerleri, o tür için dil tarafından belirtilen sayıda bellek konumuna yazılır. Dolayısıyla, veri türü genellikle bir veri değerinin yerleştiği bir veya daha fazla ardışık bellek konumunun içeriğinin nasıl yorumlanacağını gösterdiği söylenir.

C++ ' da üç tür veri türü vardır:

- Basit (yapılandırılmamış).
- Karmaşık (yapılandırılmış).
- Adres.

Veri türleri		
Basit (yapılandırılmamış) veri türleri		
	Tümlevsel(integral)[24]	
	tamsayı	short, int, long, long long
	karakter	char
	mantıksal	bool
	gerçek	float, double, long double
	sıralandırılmış	enum
Karmaşık (yapılandırılmış) veri türleri		
	dizi	array
	yapı	struct
	birleşim	union
	sınıf	class
Adres		
	işaretçi	pointer
	referans	reference

Basit veya **yapılandırılmamış veri türleri,** parçalara bölünemeyen veri türleridir. Örneğin, 123, –456, 1234567890, 'W', 123.45, vb. basit (yapılandırılmamış, bölünmez) verilerdir. Diğer taraftan, "Sen", üç karakter (basit, yapılandırılmamış) veriden oluşan: 'S', 'e' ve 'n', metin dizi türünden 25' yapılandırılmış veridir. Bu nedenle, **karmaşık (yapılandırılmış) veri türlerinin** birden çok basit türden oluştuğunu söylüyoruz.

---

[23] Devamda açıklayacağız.
[24] Tüml evsel türler bir bütünü temsil eden, yani bölünemez olanlardır.
[25] Daha sonra açıklayacağız.

Adres veri türleri, değeri bir bellek konumunun adresi olan veri türleridir[26].

Tamsayı veri türü ve karakter veri türü işaretsiz de olabilir (İng.unsigned):

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

Short, int, long, char, bool, float, double, enum, signed ve unsigned sözcükleri **tür belirteçleri** olarak adlandırılır (İng. type specifiers).

Tamsayı, gerçek, karakter ve mantıksal türler, **ilkel veri türleri** (İng. primitive data types) veya t**emel veri türleri** (İng.fundamental data types) olarak da adlandırılır. Onlar yerleşik (oluşumiçi) veri türleridir (İng.built-in data types).

Devamda tam sayı, gerçek, karakter ve metinsel veri türlerini kısaca açıklayacağız.

Tam sayı veri türü, her alt tür için (short, int, long, long long) kesin olarak belirlenmiş alt kümesinden pozitif ve negatif tam sayıların değerleridir {... –2, –1, 0, 1, 2...}. (**Bkz. tablo 1.7.1, 1.7.2, 1.7.3**).

Gerçek veri türü her alt tür için (float, double, long double) kesin olarak belirlenmiş alt kümesinden pozitif ve negatif gerçek değerler, yani ondalık sayılardır {... –2.0..., –2.001..., –1.0..., 0.0..., 1.0..., 2.0. .. }.

Karakter veri türü, karakter kümesindeki değerlerdir, yani harfler, rakamlar ve özel karakterler. Bunları bir değer olarak kullanırken, diğeraynı karakterlerden ayırt etmek için yarı tırnak işareti içine alınırlar. Örneğin, 5 tam sayısı '5' karakterinden farklıdır. Karakteristik karakter verisi bir boş yeridir ' '.

Karakter veriler kümesi, her karakterin bir öncülü (birincisi hariç) ve bir ardılı (sonuncusu hariç) olduğu ASCII (American Standard Code for Information Interchange) tablosunda tam olarak sıralanmıştır. (Bkz. **Ek B: ASCII Karakterleri**).

Karakter verilerinin birleşmesiyle, karakter dizileri (dizgeler) oluşturulur ve bunlar, dize adı verilen özel veri türü olarak ele alınır.

Bunları programdaki sabitlerden ve değişkenlerinden ayırt etmek için dizeler tırnak işaretleri içine yazılıyor.

[26] Daha sonra açıklayacağız.

[27] Unsigned, verilerin yalnızca pozitif sayılar olduğunu ve 0 sayısının, negatif değerleri olamayacağı belirtmektir.

Örneğin:

```
"C++ Pregramlama Temelleri"
"Makedonya"
"2020"
"Pazartesi, 21.06.2045 yılı"
" "       // bu bir boş yerli dizedir -  boşluk (İng. blank) dizesi.
""        // bu boş dizedir (İng. null string) ve blank dizeden farklıdır.

"Dize tek satırda yazılmalı ve tırnak işaretleri ile bitmelidir, yeni bir satıra
gitmemelidir, bu nedenle bu bir dize değildir."
```

Örneklerin sonuncusunun dize olmadığına dikkat edin. Bir dizenin bir satırdan uzun olması gerekiyorsa, + işaretiyle kolayca birleştirilebilen birden çok diziye bölünür.

Örneğin, şu ifadeyle:

```
"Makedonya" + " " + "2021"
```

"Makedonya 2021" dizesi elde edilecektir.

Dizelerle yapılan bu işleme **birleştirme** (İng. concatenation) denir.

Dizeler, karmaşık (yapılandırılmış) veri türleridir. C++'da türleri **string** ile belirtilir.

Dizeler, C++ dilinde tanımlanmış veri türü değildir, ancak. C++ Standart kütüphanesinde (İng. C++ Standard library) dahil edilmiştir.

Sabit ve Değişken Verilerin Bildirilmesi

Verilerin bir programla işlenebilmesi için bilgisayarın çalışma belleğinde olmaları gerekir. Program, her bir verinin hangi bellek adresinde bulunduğunu ve ne kadar bellek yeri kapsadığını bilmelidir. Bir verinin kapsadığı bellek yerin sayısının veri türüne bağlı olduğunu söylemiştik (bkz. ***tablo 1.7.1, 1.7.2, 1.7.3***). Bu yüzden çeviricinin hangi verinin hangi türden olduğunu anlaması için verinin yanı sıra türünün de yazılması gerekir. Bu sürece **veri bildirimi** (İng. data declaration) denir. Bildirim sırasında her veri için adı ve türü verilir. Eğer bildirim sırasında verilerin başlangıç değeri de atanmışsa buna **bildirim ve başlatma** (İng. declaration and initialization) denir.

Programda herhangi bir sabit veya değişken kullanılmadan önce (hesaplama, kopyalama, yazdırma vb. gibi çeşitli işlemlerde) bildirilmelidir. Çeviri sırasında çevirici, bildirilen her sabit veya değişken için (türüne göre) uygun bellek yeri ayırır, bildirim ve başlatma gerçekleştirilirse, bellek yerinde değer ayırır ve ayarlar. Aynı zamanda, programcılar verilerin bellekte hangi konumda bulunduğunu bilmezler, ancak derleyici konumu verilerin sabitinin veya değişkeninin (tanımlayıcı) adıyla ilişkilendirir.

Verilerin sabit ve değişken olabileceğini söyledik.

Sabitler, programın yürütülmesi sırasında değeri (bellek konumunun içeriği) değişmeyen tanımlayıcılardır. (Bir sabitin değerini değiştirmeyi denersek hata oluşur). Sabitler, const sözcüğüyle bildirilir ardından türü ve adı belirtilir, ardından = atama operatörüyle değer atanmalıdır:

const *tür ad = değer*; [28]

Sabitleri değişkenlerden ayırmak için, onları büyük harfle yazma ve sözcükleri alt çizgi ile ayırma uygulaması kullanılır.

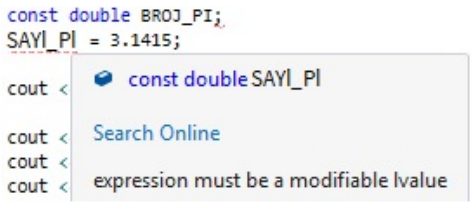
Örnekler:

```
const int M = 12;                          // Yildaki aylar
const float YER_İVMESİ = 9.81;
const double SAYI_PI = 3.1415;
const char EVET = 'E';
const string TARİH = "Pazartesi, 21.06.2036";
```

M sabiti int türündedir ve konumuna sadece tam sayı değeri yerleştirilebilir. EVET sabiti char türündedir, yani ona atanabilecek değer bir karakter olmalıdır. Başka sözlerle, sabitin türü, sabitin hangi değeri alabileceğini belirliyor.

Ayrıca, sabitin ne olduğunu bilmek için gerekirse sabitten sonra yorum yazmak çok faydalı olur.

Bir sabiti bildirirken, onun başlatılması gerektiğini bir kez daha not edelim. Bildirimden sonra başlatılırsa, hata oluşur.



Değişkenler, değeri (adlarının ilişkilendirildiği bellekteki konumun içeriği) değişebilen tanımlayıcılardır.

Değişkenler şöyle bildirilir:

tür ad;

veya:

tür ad=değer;

veya:

tür ad=ifade;

[28] Genel tanımlamalarda eğik harfler kullanacağız. Bir kelimelik ifadeleri birleştirilmiş kelimelerle ve birkaç kelimelik ifadeleri (listeler gibi) ise alt çizgi ile birleştirilmiş kelimelerle yazacağız.

Son yöntem, değişkeni aynı zamanda bildirmeyi ve başlatmayı (başlangıç değerinin atanmasını) temsil eder. Başlatılan değişkenin değeri (daha sonra), değişken atama değeri komutuyla programın herhangi bir yerinde değiştirilebilir.

Programda kullanabilmek için değişken tanımlanmış olmalıdır. Değişken, bildirim sırasında (başlatma ile) değer atanarak veya atama komutuyla **tanımlanır**.

Değişkene değer atamak için standart operatör = operatörüdür. Atamanın sağ tarafı değer veya ifade olabilir.

Bir değişkene değer atama komutu şu şekildedir:

```
ad = değer;
```

veya:

```
ad = ifade;
```

Değişkenlerin bildirimi, bildirimi ve başlatılması ve tanımlanması programın herhangi bir yerinde yapılabilir.

Örnekler:

```
short i;                    // I tamsayi degiskenin bildirimi.
int m = 10, n = 17;         // M ve n tamsayi degiskenlerin
                            // Bildirimi ve baslatilmasi.
float agirlik = 56.75f;     // Agirlik gercek degiskenin
                            // Bildirimi ve baslatilmasi.
double toplam;              // Toplam gerçek degiskenin bildirimi.
char isaret  = '+';         // Isaret karakter degiskenin
                            // Bildirimi ve baslatilmasi.
string devlet;              // Devlet dizenin bildirimi
i = 5;                      // I degiskenine deger atama
toplam  = 123.45;           // Toplam degiskenine deger atama
devlet = "MAKEDONYA";       // Devlet degiskenine deger
                            // Atama.
double ilkSayi = (1 + 2 * 7) / (7 - 5);   // Bildirim ve baslatma
```

Birinci komutla, çevirici i değişkeni için 2 bayt bellek ayırır (short türünden olduğu için), yani bu konumu i adı ile bağlar. Yalnızca bildirilen ancak başlatılmayan i değişkeni kullanmaya çalışırsak, aşağıdaki hata oluşacaktır.

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

komutuyla

```
i = 5;
```

i değişkenine 5 değeri atanır, yani bu değer bellekteki i değişkeni için ayrılan konuma yerleştirilir. Bu komuttan sonra, i değişkeni programın herhangi bir yerinde kullanılabilir.

Örnekler'deki ikinci satırda iki değişken bildirilmiştir. Bu sadece değişkenler aynı türden oldukları zaman mümkündür.

Son örnekte, atama komutunun sağ tarafında ilk önce hesaplanan aritmetik ifade bulunuyor ve ardından bu değer ilkSayi değişkenine atanır.

C++ aşağıdaki değer atama şekline de izin veriyor

```
x = (y = 10) * (z = 5);      // X'e 50 degeri verilir.
x = y = z = 20;              // X, y ve z'ye 20 degeri verilir.
```

C ++ 'da, **1.6 C++'ya Giriş** noktasındaki **Verilerin Adları** başlığında belirtilen sınırlamalar dışında, C ++ 'a değişkenlerin adlarını yazmak için belirli bir kural yoktur. Her programcı kendi tarzını oluşturur, ancak en yaygın olarak ilk harfi küçük ve addaki her kelime büyük harfle başlama tarzı kullanılır. Örneğin: sayiPİ, bireyselFiyat, adVeSoyad, kareninAlanı vb. Bu tarz C ++ programlama dilinde de kullanılır ve biz de kullanacağız.

Örnek 1.6.1

Aşağıdaki örnekte (***şekil 1.6.5***) SAYİ_Pİ sabiti bildirilmiştir ve ona 3.1415 değeri atanmıştır. Ardından, short türünden bireyselFiyat değişkeni bildirilmiştir ve onun değer bildirim sırasında değil atama komutuyla verilmiştir.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // Sabitlerin ve degiskenlerin bildirimi, baslatilmasi ve yazdirilmasi
6
7       const double  SAYI_PI = 3.1415;  //Sabitin bildirimi ve
8                                        //baslatilmasi
9       short bireyselFiyat ;            // Tamsayi degiskenin bildirimi
10      int dondurmaAdet = 100;          //Bildirme ve baslatma
11       bireyselFiyat = 50;             //Degiskene deger verme
12      cout << "Sabit PI =" <<SAYI_PI << endl;
13      cout <<  dondurmaAdet  << " dondurmanin fiyati, bir adeti"
14          <<  bireyselFiyat ; " denar" "olmak uzere";
15      cout << dondurmaAdet * << bireyselFiyat
16          << "denardir." << endl;
17
18      cout << endl;
19      system( "Color 17" );
20      system( "pause" );
21      return 0;
22  }
```

Şekil 1.6.5

Programın yürütülmesi sonucu olarak şu çıktı elde edilecektir:

```
Sabit Pİ = 3.1415
100 dondurmanın fiyati, bir adeti 50 denar olmak uzere 5000 denardir

Press any key to continue . . .
```

Alıştırma Ödevleri

1. Bildirim ve başlatma komutları içeren 3 program yazınız. Ardından, bildirilen tüm değişkenlerin değerlerini ekranda yazdırmak için programları yazdırma komutlarıyla tamamlayın. Başlatılmamış veya tanımlanmamış bir değişkenin değerini yazdırdığınızda ne olur?
2. int a = 12, b = 7; komutuyla iki değişken bildirip başlatacağınız program yazılsın. Ardından bunların toplamını, farkını ve çarpımını ekranda art arda üç satır şeklinde yazdırın.
3. Aşağıdakiler bildirilmiş/tanımlanmış olsun:

```
int x;
float y;
char z;
x = 2; y = 5.8; z = 'P';
```

   x, y ve z değişkenlerinin değerlerini tek bir boşlukla ayırarak tek bir satıra yazdıran ve ardından yeni satıra geçen program yazılsın.
4. Aşağıdakiler bildirilmiş/tanımlanmış olsun:

```
int gun = 21;
string ay = "Haziran";
int yıl = 2021;
string bugun = "Pazartesi";
```

Bildirilen ve başlatılan değişkenleri kullanan ve şunu yazdıran program yazılsın:

```
Bugün 21 Haziran 2021 Pazartesi.
```

Bilgiyi Kontrol Etme Soruları

1. C++ dilini kim tasarlamış?
2. C++'da karakter kümesi nelerden oluşur?
3. C++'da kullanılan bazı özel karakterleri listeleyin.
4. Tanımlayıcılar nelerdir?
5. Bir dilde sözdizimi ve semantik nedir?
6. Tanımlayıcılar nasıl ayrılır?
7. main() fonksiyonunun biçimini yazın.
8. C++'da komutlar hangi karakterle ayrılır?
9. Programın girintilenmesi neden yapılır?
10. Bir programda yorumlar nasıl işaretlenir?
11. C++'da yazdırma komutu nedir?
12. Yazdırma operatörü hangisidir?
13. Yeni bir satırda yazdırmaya geçmek hangi escape dizgesi ile yapılır?
14. Sıralı komut yapısı nedir?
15. Büyüklük nedir, veri nedir?

16. Sabit nedir ve değişken nedir?
17. Hangi değerlere değişmezler diyoruz?
18. C++'da bir değişkenin adı ne kadar uzun olabilir?
19. Verilerin özellikleri nelerdir?
20. C++'da veri adları nasıl olabilir?
21. Kullanıcı adlarını (tanımlayıcıları) tanımlama kurallarını belirtin.
22. Aşağıdaki veri adlarından hangileri doğru, hangileri yanlış yazılmış ve neden?

    a) _54abc b) ben_5T c) printf ç) ha Di
    d) benimAdim e) for f) a.7_b9 g) b9_c10

23. Veri türünün ne olduğunu açıklayınız?
24. C++'da hangi veri türlerini biliyorsunuz?
25. Basit ve karmaşık veri türlerinin ne olduğunu açıklayın.
26. Adres veri türlerinin içeriği ne olabilir?
27. Hangi tür belirteçleri biliyorsunuz?
28. Unsigned kelimesi, veri türünden önce belirtilmişse neyi belirtir?
29. Tamsayı türü, gerçek tür, karakter türü ve metin veri türleri hangi değerlere sahip olabilir? Bu türler hangi belirteçlerle belirtilir?
30. Hangi tanımlayıcının sabit, hangisinin değişken olduğunu söylüyoruz?
31. Programda aynı değişken kaç kez ve hangi yerlerde kullanılabilir?
32. Aynı programda aynı isimli sabit ve değişken kullanırsanız ne olur?
33. Programın adını değişken olarak kullanabilir misiniz?
34. Değişken bildirimi nedir, değişken tanımı nedir?
35. Bir değişkenin başlatılması ne anlama gelir?
36. Bildirme ve başlatma komutlarının genel şeklini yazınız.
37. Değişkenlerin bildirimi ile ilgili birkaç örnek ve değişkenlerin bildirimi ve başlatılmasıyla ilgili birkaç örnek verin.
38. Hangi operatör ile bir değişkene veya bir sabite değer atanabilir?
39. Yalnızca bildirilmiş bir değişken ne zaman kullanılabilir?
40. Aşağıdaki bildirim doğru mudur: int m = 10, n = m;

1.7 Veri Türleri

Tamsayı Veri Türü int

Tamsayı veri türü, Z = {... –2, –1, 0, 1, 2...} tamsayılar kümesinden herhangi bir tamsayı değerine sahip olabilir.

Dört tür tamsayı verisi olduğunu söylemiştik: short, int, long ve long long. Türüne bağlı olarak, değer kümesinin boyutu farklıdır.

Tablo 1.7.1'de, tamsayı veri türleri, değerlerinin aralığı (İng.data range) ve kapladıkları bellek[29] miktarı gösterilmiştir.

Tür	Değerler	Bellek
short	–32768... 32767	16 bit
int	–2147483648... 2147483647	32 bit
long	–2147483648... 2147483647	32 bit
long long	–9223372036854775808... 9223372036854775807	64 bit
unsigned short	0... 65535	16 bit
unsigned int	0... 4294967295	32 bit
unsigned long	0... 4294967295	32 bit
unsigned long long	0... 18446744073709551615	64 bit

Tablo 1.7.1

Her türün değer aralığı, programın yürütüldüğü bilgisayara bağlıdır. En büyük ve en düşük değerler sabit olarak tanımlanmıştır:

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

Tamsayı veri türleri ile işlemler için, şu aritmetik operatörler kullanılır:

+	Toplama
–	Çıkarma ve olumsuzluk[30]

[29] Veri türlerinin kapladığı bellek miktarı, bir bilgisayar platformundan diğerine değişebilir.

*	Çarpma
/	Tamsayı bölümü - iki tam sayının bölümünün tamsayı kısmı
%	Modulo bölümü - iki tam sayıyı bölmenin kalanı

İki tam sayının / operatörü ile bölünmesinin sonucu, bölme işleminin tam kısmıdır. Örneğin:

```
10 / 4 = 2,  -15 / 6 = -2,    20 / (-6) = -3,    (-14) / (-3) = 4
5 / 0 derleme sırasında hata gösterecek.
```

İki tamsayı üzerinde % operatörünün uygulanması sonucu, bölme işleminin kalanını verir. Örneğin[31]:

```
10 % 4 = 2,  -15 % 6 = -3,   20 % (-6) = 2,    (-14) % (-3) = -2
5 % 0 derleme sırasında hata gösterecek.
```

Tamsayı operatörlerin önceliği şöyledir:

1	*	/	%	aynı öncelik
2	+	–		aynı öncelik

Aritmetik ifadeler hesaplanırken işlemler soldan sağa doğru yapılır. İlk önce önceliği 1 olan işlemler (üç işlemden herhangi biri) yürütülür, ardından önceliği 2 olan işlemler (iki işlemden herhangi biri) yürütülür.

Örneğin:

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

Parantez kullanılarak operatörlerin önceliklerine göre belirlenen işlemlerin yürütülme sırası değiştirilir.

Örneğin:

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

Örnekler:

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

[30] Bir işlenenin önündeki eksi işareti (–) tekli operatör, yani tekli eksi olarak adlandırılır.

[31] Operatörlerden biri negatif ise geri kalanının işareti çeviriciye bağlıdır.

[32] Aşırı yüklemeyi daha sonra açıklayacağız.

Atama komutlarında değişkenlere değer atamak için standart operatör = operatörüdür. İşaretsiz bir sabit atanırken, u veya U son eki eklenmelidir, aksi takdirde int olarak ele alınacaktır.

Aşağıdaki sabit türleri için de son ek eklenir:

Tür	Son ek
long	l veya L
long long	ll veya LL
unsigned long	ul veya UL
unsigned long long	ull veya
Örnekler:	ULL.

```
int ilkSayi, ikinciSayi;
short rakam  = 8;
long arapskiCifri = 1234567890;
ilkSayı  = 1 + 2 * 3 + 45 % 6;           (= 10)
ilkSayı  = ilkSayı  + rakam;             (= 18)
ikinciSayi  = ilkSayı  / rakam;          (= 2)
ikinciSayi  = ikinciSayi % rakam;        (= 0)
const long GUNESTEN_SATURNE = 1424600000L;
long long aastronomikBirim = 149597870700LL;
unsigned int tumRakamlar = 1234567890u;
unsigned long maxLong = 4294967295ul;
unsigned long long maxLL = 18446744073709551615ull;
x = (y = 10) * (z = 5);
x = y = z = 20;
```

Sıradaki örnekte, tamsayı bölmeyi, modulo bölmeyi, yani tamsayı bölmenin kalanını ve tam sayılar için işlemlerin önceliğini sunacağız.

Örnek 1.7.1

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {    // Tamsayilarla islemler
5
6       // Tamsayi bolumu - iki tam sayinin bolumunun tamsayi kismi
7       cout << "10 / 4 = " << 10 / 4 << endl;
8       cout << "-15 / 6 = " << -15 / 6 << endl;
9       cout << "20 / (-6) = " << 20 / ( -6 ) << endl;
10      cout << "-14 / (-3) = " << ( -14 ) / ( -3 ) << endl;
11      // Modulo bolme - iki tamsayiyi bolme isleminin kalani
12      cout << "10 % 4 = " << 10 % 4 << endl;
13      cout << "-15 % 6 = " << -15 % 6 << endl;
14      cout << "20 % (-6) = " << 20 % ( -6 ) << endl;
```

Şekil 1.7.1

```
15      cout << "-14 % (-3) = " << ( -14 ) % ( -3 ) << endl;
16      // Tam sayilar icin operatorlerin onceligi
17      cout << "1 + 2 * 3 - 4 * 5 % 6 / 7 = "
18          << 1 + 2 * 3 - 4 % 5 * 6 / 7 << endl;
19      cout << "(1 + 2 * (3 - 4)) * (5 % 6 / 7) = "
20          << ( 1 + 2 * ( 3 - 4 ) ) % ( 5 * 6 / 7 ) << endl;
21
22      cout << endl;
23      system( "Color 17" );
24      system( "pause" );
25      return 0;
26  }
```

Şekil 1.7.1 (devam)

Şekil 1.7.1'deki programın yürütülmesinden elde edilen bir çıktı şudur:

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

Press any key to continue . . .
```

Operatörlerin Kısaltılmış Biçimi

C++'daki operatörler, aritmetik operatör ve atama operatörü = ile ifade edilen **kısaltılmış biçimde** kullanılabilir. Bunlara **bileşik atama operatörleri** (İng. compound assignment operators) denir.

Atama operatörlerin genel biçimi şöyledir:

aritmetikoperatör =

Tamsayı veriler için atama operatörlerinin kısaltılmış biçimleri şunlardır:

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

Örnekler:

```
ilkSayi += rakam;      komutu      ilkSayi = ilkSayi + rakam;
ikinciSayi %= rakam;   komutu      ikinciSayi = ikinciSayi % rakam;
```

Aşağıdaki örnekte operatörlerin kısaltılmış biçimlerinin kullanımı gösterilmiştir.

Örnek 1.7.2

Şekil 1.7.2'de, operatörlerin kısaltılmış biçiminin kullanımı gösterilmiştir.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {    // Operatorlerin kisaltilmis bicimleri
5
6       int rakam = 8;
7       cout << "Rakam= " << rakam << endl;
8
9       int ilkSayi, ikinciSayi, ucuncuSayi ;
10      ilkSayi   = ikinciSayi  = 100;        // Bunu C++'da yapabiliriz
11      cout << "IlkSayi = " <<ilkSayi << endl;
12      cout << "IkinciSayi  =" <<ikinciSayi<< endl;
13
14       ilkSayi  +=rakam;                    // + operatorunun kisaltilmis bicimi
15      ikinciSayi *= rakam;                  // * operatorunun kisaltilmis bicimi
16      cout << " ilkSayi  += rakam =   " << ilkSayi << endl;
17      cout << "ikinciSayi *= rakam =   " << ikinciSayi << endl;
18
19      // Degiskenlere ifadede deger atama
20      // Bunu da C++'da yapabiliriz
21      ucuncuSayi = ( ilkSayi = 123 ) + (ikinciSayi  = 321 );
22      cout << "Ucuncu sayi = "  << ilkSayi << " + " <<ikinciSayi  << " = "
23          << ucuncuSayi  << endl;
24
25      cout << endl;
26      system( "Color 17" );
27      system( "pause" );
28      return 0;
29  }
```

Şekil 1.7.2

Programın yürütülmesinden bir çıktı şudur:

```
Rakam  = 8
Ilk sayi      = 100
Ikinci sayi    = 100
ilkSayi    += rakam  =  108
ikinciSayi  *= rakam  =  800
Ucuncu sayi = 123 + 321 = 444

Press any key to continue . . .
```

Alıştırma Ödevleri

1. x = 123 ve y = 52 sayıları için toplamı, farkı, çarpımı, bölümün tam kısmını ve bölümün kalanını hesaplayıp yazdıracağınız program yazın. Bu arada, sadece bir değişken z kullanılsın.
2. Diğer değişkenler kullanmadan tamSayiX = 123 tamsayısının karesini ve küpünü hesaplayan ve yazdıran program yazılsın.

3. Üç haneli X = 274 sayısının orta rakamını ekrana yazdıran program yazılsın.
4. Manastırdan'dan Üsküp'e giden tren sabah 5:20'de kalkıyor. Sabah 8:12'de Üsküp'e varıyor. Trenin Manastırdan'dan Üsküp'e ne kadar süre yol yaptığını hesaplayan program yazılsın.
5. Aşağıdaki açıların toplamını ve farkını hesaplayan program yazılsın:
   $\alpha = 100° 47' 38''$ ve $= \beta 35° 53' 49''$.

Gerçek Veri Türü — float, double, long double

Gerçek veri türleri, gerçek sayılar kümesinden değer alan verilerdir. Gerçek veri türü genellikle ondalık sayıları ifade etmektedir.

C++'te üç gerçek veri türü yerleştirilmiştir: **float, double** ve **long double**.

Gerçek türdeki veriler sabit ve değişken olabilir.

Başlatma, tamsayı veri türünde olduğu gibi aynı şekilde gerçekleştirilir.
Float türünden değişkenleri bildirirken, f veya F son eki (uzantısı) eklenmelidir.
Aksi halde double türü olarak kabul edilecektir.

Ayrıca, long double türünün sonuna l veya L son eki eklenir.

Örnekler:

```
float x;
float toplamAgirlik;
double topluToplam = 12345.67;
long double PI = 3.14159265358979323846;
const float YERYUZU_İVMESİ = 9.81f;
float e_sayisi = 2.7182818284590452354F;
long double elektronYaricapi = 0.00000000000000282L;    //2.82x10-15
```

Float, double ve long double veri türlerinin kesinliği farklıdır, yani:

- **float** verileri 32 bit kapsar ve 7 önemli basamağa sahiptir.
- **double** verileri 64 bit kapsar ve 15 önemli basamağa sahiptir.
- **long double** verileri 80 bit kapsar ve 18 önemli basamağa sahiptir.

Gerçek veri türünün değer aralığı ***tablo 1.7.2***'de verilmiştir.Aralık, sayıları 32, 64 ve 80 bit ile temsil etme kesinliğine göre hesaplanmıştır. Tamsayıları ve ondalık sayıları ifade etmek için standart biçimler vardır[33].

[33] Kesinliğe, yani bit sayısına göre bilgisayarda tamsayıları ve ondalık sayıları temsil etme biçimlerini açıklayan IEEE (Institute of Electrical and Electronics Engineers) standardı vardır.

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

Tablo 1.7.2

C++'daki ondalık sayıların en büyük ve en küçük değerleri derleyiciye bağlıdır ve sabitler[35] olarak ayarlanır. Örneğin, benim bilgisayarımda bu değerler şunlardır:

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

Gerçek türdeki veriler için aritmetik operatörler şunlardır:

+	Toplama
–	Çıkarma veya olumsuzluk
*	Çarpma
/	Gerçek bölme (verilerden en az biri gerçek sayıysa)

0.0 ile bölünürken sayı türü aralığının dışına çıkarsa, hata oluşmaz, ancak sonuç olarak inf veya –inf (sonsuz, İng.infinity) elde edilir.

Aritmetik ifadelerde gerçek sayılar için operatörlerin önceliği aşağıdaki gibi soldan sağa incelenir:

1	*	/	aynı öncelik (çarpımsal operatörler)
2	+	–	aynı öncelik (toplamsal operatörleri)

Bu operatörler de standart ve kısaltılmış biçimde kullanılabilir:

+=	–=	*=	/=

[34] Bu tür, C++'ın önceki sürümlerinde kullanılmıştır. Bugünkü C++ standartlarında bu tip, double ile eşdeğerdir.

[35] Gerçek veri türleri için en küçük pozitif değerler sabit olarak tanımlanır. Bilgisayarımda bu değerler şunlardır:

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

Örnek 1.7.3

Bu örnek ile (***şekil 1.7.3***), float veri türü (32 bit ile temsil edilen) ile double veri türü (64 bit ile edilen) arasındaki kesinlik farkı gösterilmiştir.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {     // Gercek sayilarla islemler
5
6       // float veri turu icin
7       float ilkSayı, ikinciSayi, sonuc;
8       ilkSayı   = 1.000000000f;
9       ikinciSayi = 0.999898989f;
10      sonuc = ilkSayi – ikinciSayı;
11      cout <<“ float turunden ondalik sayilarin farki: “ << endl;
12      cout <<ilkSayi << " - " <<ikinciSayi  << " = " <<sonuc << endl;
13
14      // double veri turu icin
15      double ilk_Sayı, ikinci_Sayi, sonuc2;
16      ilk_Sayı = 1.000000000;
17      ikinci_Sayi  = 0.999898989;
18      sonuc2 = ilk_Sayi – ikinci_Sayı;
19      cout <<“ double turunden ondalik sayilarin farki: “ << endl;
20      cout << ilk_Sayi   << " - " << ikinci_Sayi << " = "
21          <<  sonuc2     << endl;
22
23      cout << endl;
24      system( "Color 17" );
25      system( "pause" );
26      return 0;
27  }
```

Şekil 1.7.3

Programın yürütülmesinin bir örneği şudur:

```
float turunden ondalik sayilarin farki:
1  - 0.999899  = 0.00010103
double turunden ondalik sayilarin farki:
1  - 0.999899  = 0.000101011

Press any key to continue . . .
```

Çıktıdan görüldüğü gibi, verilerin değeri aynı olmasına rağmen sonuç farklıdır.

Ondalık Verileri Temsil Etme Biçimleri

Programlama dillerinde ondalık veriler iki şekilde temsil edilir:

- Ondalık biçimde (f-biçimi, F-biçimi), **kaymayan(sabit)nokta** (İng.fixed point) olarak bilinen şekil.
- Üstel biçimde (e-biçimi, E-biçimi), **kayan nokta** (İng. floating point) olarak bilinen şekil.
  f harfi "float" kelimesinden, e harfi ise "exponent" kelimesinden gelir.

f-biçimindeki (veya F-biçimindeki) ondalık sayılar, tam sayıyı ve sayının ondalık kısmını ayıran sabit yerde bulunan ondalık nokta ile yazılır. Bu nedenle, ondalık sayıların bu şekilde temsil edilmesi kaymayan (sabit) nokta gösterimi olarak adlandırılır. Örneğin: 1234.56, -0.25, 0.5432, -123.45, vb.

e-biçiminde (veya E-biçiminde), üs kullanarak nokta hareket ettirilebilir. Örneğin, 123456789.12 sayısı, ondalık noktayı herhangi bir yere sola veya sağa kaydırarak yazılabilir:

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

Aynı sayılar e- veya E-biçiminde şu şekilde yazılıyor:

1.234568e+08, 1.234568E+08.

Ondalık sayılar e-biçiminde (veya E-biçiminde) yazdırıldığında, ondalık noktanın solunda sadece bir basamak yazdırılır, e veya E harfi ise üssün önüne yazılır. Üs - veya + işaretiyle yazılır.

f, F, e veya E biçiminde yazdırılmış ondalık sayılar 6 doğru basamak kesinliğine sahiptir, yedinci basamak ise yuvarlanır.

Örneğin, aşağıdaki değişkenler f-biçiminde ve e-biçiminde tanımlanırsa (bildirilir ve başlatılırsa):

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

onların f-biçiminde yazdırılması sırasında, şu elde edilir:

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

e-biçiminde yazdılırsa ise, şu elde edilir:

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

Benzer şekilde, ***şekil 1.7.3***'teki programda ikinciSayi değişkeninin değeri biraz değiştirilirse (0.999898989 yerine 0.999998989 değişirse), sonuc değişkeninin değeri e-biçiminde yazdırılacak:

```
float turunden ondalik sayilarin farki:
1 - 0.999999 = 1.01328e-06
double turunden ondalik sayilarin farki:
1 - 0.999999 = 1.011e-06

Press any key to continue . . .
```

Aritmetik İfadeler ve Veri Türünün Dönüştürülmesi

Hesaplamalar sırasında programlarda basit ve karmaşık aritmetik ifadeler belirir. Bunlar sabitler, değişkenler ve operatörler ile ifade edilirler.

Operatörler aşağıdakiler olabilir:

Tekli: + ve –
İkili: + – * / ve %

Örneğin:

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

Eğer bir ifadede birden fazla operatör varsa, bunlar aşağıdaki önceliğe göre yürütülür:

En yüksek	tekli +	tekli –
Orta:	* / %	
En düşük	+ –	

Örneğin, a=5, b=4, c=3, d=2 için, a+b/c%d ifadesi şu şekilde hesaplanacaktır:

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

Operatörlerin önceliği parantezlerle değiştirilebilir. Örneğin, bir önceki ifadeyi (a+b) / (c % d) olarak yazarsak, şöyle hesaplanacaktır:

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

İfadelerdeki tekli operatörler sağdan sola yürütülür. Örneğin, a=5 ve b = - 3 için

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

Değişken türünün, karşılık gelen bellek konumunun sadece o veri türünü içerebileceğini belirttiğini söylemiştik.

Örneğin, şunlar bildirilirse:

```
int tamSayi;
float gercekSayi;
```

Aşağıdaki atama komutları geçerlidir:

```
tamSayi = 5;
gercekSayi = 6.7;
```

Ancak, eğer değerleri değiştirirsek ne olur, yani:

```
tamSayi = 6.7;
gercekSayi = 5;
```

İlk komut ile tamSayi değişkeni sadece tam sayı değerleri alabildiğinden dolayı 6.7'nin ondalık kısmı kesilerek tamSayi değişkenine 6 değeri atanır.

İkinci komutla, gercekSayi değişkeni sadece gerçek sayıları alabildiğinden dolayı, 5 tam sayısı gerçek sayı 5.0'a dönüştürülür ve gercekSayi değişkenine 5.0 değeri atanır.

Uygun olmayan türdeki bir değişkene bu şekilde değer atama, **örtük** (otomatik) dönüştürme veya **tür zorlaması** (İng. type coercion) olarak adlandırılır.

Gördüğümüz gibi, tamsayı değişkenine gerçek değer atanırken veri ile ilgili bilgiler kayboluyor çünkü ondalık kısım kesiliyor. Bilgi kaybını önlemek için, **tür dökümü** olarak da bilinen (İng. type casting) **açık tür dönüştürme** (İng.type conversion) kullanılmaktadır.

Tür dökümü, dönüştürülecek değişken veya ifadeden önce tür belirtilerek yapılır. Böylece önceki iki komut döküm ile şu şekilde olacaktır:

```
tamSayi = int(6.7);
gercekSayi = float(5);
```

tamSayi değişkeni değerin 6.99 olması durumunda (ki bu 7'ye yakındır) bile 6 değerini alacağından dolayı, verilerle ilgili daha az bilgiyi kaybetmek için 0.5 değeri ekleyerek yuvarlamak iyi bir programlama uygulamasıdır.

Komut şu şekilde olacaktır:

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

Çoğu zaman, aritmetik ifadeler hem tam sayı hem de gerçek değişkenler içerir, yani karışık veri türleri içerir.

Aşağıdaki gibi karışık veri türlerine sahip ifadelerde:

```
gercekSayi1 = 5 + 7.34;
gercekSayi2 = 129.56 - 1.3e+2;
```

daha düşük kesinlikli türlerin daha yüksek kesinlikli türlere örtük (dolaylı) dönüşümü gerçekleştirilir.

Aşağıdaki komutlar grubuyla:

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

ilk önce k tam sayı değişkeninin (= 5) double (= 5.0) türü değişkenine örtük (otomatik) dönüşümü gerçekleştirilir, 7.34 ile toplanır ve ardından toplam, p değişkenine (= 12.34) atanır.

p'nin hesaplanması ifadesinde, açık dönüştürme de kullanılabilir:

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

Aşağıdaki komutla:

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

12.34'ün 12'ye örtük dönüşümü gerçekleştirilir ve bu değer m tamsayı değişkenine atanır.

Aynısı, açık dönüştürme ile de yapılabilir:

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

bu arada p değişkeninin değeri açık olarak int türüne dönüştürülür.

Aşağıdaki örnekler de dönüştürmeyi göstermektedir:

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

Tür dönüştürme için, C dili için tipik olan (İng. C-like), yani ondan alınan bir form da kullanılabilir. Bu arada, **cast** operatörü olarak adlandırılan ve aynı zamanda **döküm**

operatörü da denilen **( )** operatörü kullanılır. Bu operatör, önce ifade değerinin dönüştürüleceği tür, ardından türü değiştirilen ifade parantez içinde belirtilerek kullanılır.

Örneğin:

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

Önceki örnekler, aşağıdaki şekilde de yazılabilir:

```
k = (int)(5 + 7.34);         (= 12)
r = (double)(3 * 5);         (= 15.0)
cout << (int) 'a' << endl;   (97 yazdırılacaktır)[36]
```

İki tam sayı değişkenini bölerken sıkça hata yapılır. Örneğin 8 doğal sayının toplamı 100 ise ortalama değer 100 / 8 = 12,5'tir:

```
          int toplam = 100;
          int sayilar = 8;
          float ortalamaDeger = toplam / sayilar;//Yanlış, tamsayı bölümü)
şöyle olmalı  float ortalamaDeger = float(toplam) / sayilar;
veya      float ortalamaDeger = zbir / float(sayilar);
veya      float ortalamaDeger = 1.0 * toplam / sayilar; (tür dökümü yapmadan)
```

Önceki örneklerde, şöyle bir komut da kullanmıştık:

```
ilkSayi = ilkSayi + rakam;
```

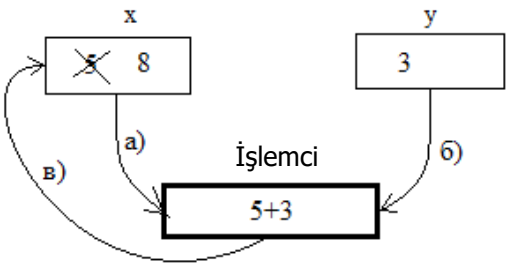
Bu komutu matematiksel biçimde yazarsak, imkansız bir denklem elde edeceğiz:

```
x = x + y
```

Bu denklem sadece y = 0 olduğunda doğrudur.

Ancak bu ifade, programlamada doğru şekilde yürütülür. İşlemci önce x değişkeninin o anki değerini y değişkeninin değeri ile toplayıp sağdaki ifadeyi hesaplıyor ve elde edilen toplamı x değişkeninin yeni değeri olarak atar.

Örneğin, değişkenlerin x = 5 ve y = 3 değerleri varsa, komut aşağıdaki diyagrama göre yürütülecektir.



a) x (= 5) değeri işlemciye aktarılır.
b) ona y (= 3) değeri eklenir.
c) Elde edilen toplam x'in yeni değeri (= 8) olarak yerleştirilir.

[36] 'a' karakterinin ASCII değeri 97'dir.

x değişkeninin değeri 1 için artarsa (artırma diyoruz), komut şöyle olacaktır:

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

Bu komutu yürütmek için, **arttırma operatörü**[37] (İng. increment operator) adı verilen ++ özel operatörü tanıtılmış. Önceki komut bu operatör ile, şu şekilde olacaktır:

```
++x;    veya    x++;
```

Arttırma operatörüne karşılık olarak, değişkenin değerini 1 için azaltan, **azaltma operatörü** – – da tanıtılmış (İng. decrement operator). Örneğin, aşağıdaki komutların:

```
– –x;    veya    x– –;
```

şu komutla aynı etkisi olacaktır:

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

Örnek 1.7.4

Şekil 1.7.4'teki programla örtük ve açık dönüştürme gösterilmiştir.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {     //Veri turlerin donusumu
5
6       cout <<"Ortuk donusturme (sadece daha ust ture)" << endl;
7       int k = 5;
8       double p;
9       p = k + 7.34;
10      cout << "5 + 7.34 = " << p << endl;
11      int m = p;
12      cout << "int m; \nm = 5 + 7.34 = " << m << endl;
13
14      cout << "\nAcik (zorunlu) donusturme " << endl;
15      cout << "m = int(5 + 7.34) = " << int( p ) << endl;
16      cout << "m = (int)(5 + 7.34) = " << ( int ) p << endl;
17      cout << "int('a') = " << int( 'a' ) << endl;
18      cout << "(int)'a' = " << ( int ) 'a' << endl;
19      int toplam  = 100;
20      int sayilar = 8;
21      cout << "100 / 8 = " << toplam/sayilar << endl;
22      cout << "float(100) / 8 = " << float(toplam) / sayilar << endl;
23      cout << "100 / float(8) = " << toplam/ float(sayilar) << endl;
24      cout << "1.0*100 / 8 = " << 1.0 * toplam/sayilar << endl;
```

Şekil 1.7.4

[37] Bu operatörleri daha geç daha ayrıntılı açıklayacağız.

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

Şekil 1.7.4 (devam)

Uygulamanın yürütülme çıktısı şu olacaktır:

```
Ortuk donusturme (sadece daha ust ture)
5 + 7.34 = 12.34
int m;
m = 5 + 7.34 = 12

Acik (zorunlu) donusturme
m = int(5 + 7.34) = 12
m = (int)(5 + 7.34) = 12
int('a') = 97
(int)'a' = 97
100 / 8 = 12
float(100) / 8 = 12.5
100 / float(8) = 12.5
1.0*100 / 8 = 12.5
```

Alıştırma Ödevleri

1. Yarıçapı r = 18,3 cm olan çemberin çevresini (O = 2rπ) ve alanını ($P = r^2\pi$) hesaplayan program yazılsın.
2. Kenarları a = 12.5 cm, b = 7.2 cm ve c = 15.8 cm olan paralelyüzün alanını ve hacmini hesaplayan program yazılsın.
3. 5.27x – 23.15 = 0 denklemini çözmek için program yazılsın.
4. α = 36.27° açısını, $\alpha^{rad} = \alpha\pi^{o} / 360$ formülüne göre radyana çeviren program yazılsın.
5. 465 kilometre uzunluğundaki yolu 5 saatte geçen otomobilin hızını hesaplayan program yazılsın.
6. a = 3 ve b = 16 için a ve b gerçek sayılarının bölümünün, 3. ondalık basamağını yazdıran program yazılsın.
7. π = 3.1415 sayısının değeri ile $4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\frac{1}{13}-\frac{1}{15}\right)$ toplamı arasındaki farkı hesaplayan program yazılsın.

Karakter Veri Türü **char**

Karakter türündeki veriler, herhangi bir karakter veya herhangi bir tam sayı değerine sahip olabilir. Onlar değişken veya sabit veri olabilirler. **char** kelimesi ile bildirilirler. Bu arada, metindeki aynı karakterden ayırt etmek için karakter tek tırnak işareti içine koyulur.

Örnekler:

```
const char hashtag = '#';
char a;
char isaret = '+';
char denar = 'd';
char ascii = 120;
```

C++ programlama dilinde kullanılan tüm karakterler, bilgisayarda ASCII (American Standard Code for Information Interchange) tablosuna göre tamsayılar - kodlar ile temsil edilir (belleğe yazılır). (Bkz. **Ek B: ASCII Karakterler**). Örneğin, Q karakterinin ikili değeri (kodu) 010100012 = 8110'dur, '=' karakterinin kodu ise 001111012 = 9910'dur. Bu kodları yazdırırken, karşılık gelen karakterler yazdırılır.

Karakter türündeki veriler, değer aralığı dikkate alınarak herhangi bir tam sayı türü gibi ele alınabilir. Onlarla aritmetik işlemler yapılabilir, okunabilir, yazdırılabilir ve karşılaştırılabilir.

Örneğin, aşağıdaki komutla yuzde değişkeni bildirilirse:

```
char yuzde;
```

% işareti aşağıdaki iki komuttan herhangi biriyle atanabilir:

```
char yuzde = '%';
yuzde = 37;
```

Aşağıdaki komutla:

```
yuzde++;
```

% işaretinin tamsayı değeri 37'den 38'e artar, şu komutla ise:

```
cout << yuzde << endl;
```

& karakteri yazdırılacaktır. Başka sözlerle, standart ASCII karakterleri sıralıdır ve onluk değerleri 0 (000000) ile 127 (1111111) arasındadır.

Genişletilmiş ASCII tablosu, 0'dan 255'e kadar kodlara sahip 256 karakter içerir. Genişletme, 128'den 255'e kadar kodlu karakterleri ile yapılmıştır.

C++ karakterleri ASCII[38] tablosuna göre tam sayı değerlere sahip olduğu için onlar karşılaştırılabilir. Örneğin, '+' karakterinin ASCII değeri 43_{16} veya 67_{10}'dur, ':' karakterinin ASCII değeri ise 58_{16} veya 88_{10}'dur. Bu nedenle, '+' < ':' yazılabilir.
Karakter türü verilerinin değerleri ve aralıkları ***tablo 1.7.3.***'te verilmiştir. –128 ila –1 ve 128 ila 255 arasındaki kodlar aynı karakterlerin kodlarıdır.

Tür	Değerler	Bellek
char	–128... 127	8 bit
unsigned char	0... 255	8 bit

Tablo 1.7.3

[38] Standart ASCII tablosu 128 karakter içerir, genişletilmiş ASCII tablosu ise 256 karakter içerir. Ancak birkaç dünya dilinden işaretleri kapsamak için bugün uluslararası kodlama standardı olan unicode (İng. Unicode) kullanılır.

Char türünden verilerinin en büyük ve en küçük değerleri şu sabitlerdir:

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

Örnek 1.7.5

Faiz oranı %7,5 ise, 100.000 denarlık yatırılan sermayenin faizi, faiz = sermaye · faiz oranı/100 formülüne göre hesaplansın. $\frac{\text{kapital} \cdot \text{kamatnaStapka}}{100}$.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {    //Karakter veri türü
5
6       char isaretYuzde = '%';
7       double sermaye = 100000.0;
8       float faizOrani = 7.5f;
9       double faizTutar;
10      faizTutar = sermaye * faizOrani / 100;
11      cout <<sermaye <<"denarlik yatirilan sermaye icin" << endl;
12      cout <<"yillik faiz orani " << isaretYüzde
13          << faizOrani << endl;
14      cout <<faizTutar <<"denar faiz kazanilir" << endl;
15
16      system( "Color 17" );
17      system( "pause" );
18      return 0;
19  }
```

Şekil 1.7.5

Programın yürütülmesiyle bir çıktı aşağıdaki gibi olabilir:

```
100000 denarlik yatirilan sermaye icin
Yillik faiz orani %7.5 ise
7500 denar faiz kazanilir
Press any key to continue . . .
```

Alıştırma Ödevleri

1. Adınızdaki harf sayısı kadar karakter değişkeni bildireceğiniz ve her birine birer harf atayacağınız program yazılsın. Ardından, çıktıda tam adı elde edeceğiniz şekilde onları yazdırın.
2. ASCII tablosunda 'S' ve 'D' harflerinin sıra sayıları arasındaki farkın bulunduğu program yazılsın.
3. Bir ürünün fiyatı 123.45 denardır,bir hafta içinde %15 pahalılaşmış ve ardından %8 ucuzlamıştır. Haftanın sonunda bu ürünün fiyatı ne kadardır? (% işareti için karakter değişkeni kullanın).

4. ASCII tablosunda @ karakterinden önceki ve sonraki karakter bulunsun ve yazdırılsın.
5. Alfabenin iki harfi arasında kaç harf olduğu belirlensin.
6. İki tamsayı değişkeni a ve b'yi 12 345 ve 678 değerlerine bildirilsin ve başlatılsın. Ayrıca operator karakter bir değişken de bildirilsin ve ona +, –, *, / ve %. işaretlerden biri atansın. Her işaret atamasından sonra, aralarında operatör işareti olacak şekilde a ve b'yi aralarındaki operatörü yazdırarak işlemin sonucu yazdırılsın. İşlemin sonucu için uygun bir değişken bildirilsin. Örneğin operator = '%' için 12 345 % 678 = 141 yazdırılsın.

Mantıksal Veri Türü **bool**

Değeri true veya false olabilen verilere **mantıksal türü verileri** (İng. logical type) denir. Ayrılmış sözcükler true ve false, C++'da özel sabitlerdir. Mantıksal ifadeleri hesaplarken, true ve false mantıksal sabitleri 1 ve 0 değerine sahiptir ve 1 ve 0 olarak yazdırılır.

Mantıksal veri türü **bool** kelimesi ile tanımlanır.

Örnekler:

```
bool evethayir;
bool trafikisigi = true;
```

Mantıksal türdeki verilerle **mantıksal operatörler** de kullanılabilir (İng. logical operators)[39]:

&&	mantıksal ***VE*** (AND)
||	mantıksal ***VEYA*** (OR)
!	mantıksal ***DEĞİL*** (NOT)

Bu operatörlere **koşullu operatörler** (İng.conditional operators) de denir, çünkü çoğunlukla daha karmaşık bir mantıksal ifadenin değerini (true veya false) hesaplarken kullanılırlar.

Mantıksal operatörler aslında, onları tanımlayan matematikçi George Boole'e göre **boole fonksiyonları** (İng. boolean functions)[40] olarak adlandırılan fonksiyonlardır.

Mantıksal operatörler aşağıdaki tabloda tanımlanmıştır

DEĞİL*/NOT/!*	***VE****/AND/&&*	***VEYA****/OR/||*
!false = true	false && false = false	false || false = false
!true = false	false && true = false	false || true = true
	true && false = false	true || false = true
	true && true = true	true || true = true

Tablo 1.7.4

[39] Başka mantıksal operatörler de vardır.
[40] Başka Boole fonksiyonları da vardır.

Örnekler:

a = true ve b = false için aşağıdaki ifadelerin değerleri şunlar olacak:

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

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

Daha önce belirtilen tüm veri türleri ile **ilişkisel operatörler** (İng. relational operators) kullanılabilir:

<	küçük
<=	küçük veya eşit
>	büyük
>=	büyük veya eşit
==	eşit
!=	farklı

= = operatörünün "iki eşittir" değil, iki işleneni karşılaştırmak için ikili operatör olduğunu vurgulayalım. Örneğin, z mantıksal bir değişkeni ise, aşağıdaki komutla:

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

ilk önce x ve y değerleri karşılaştırılır ve aynıysa z'ye true değeri, aynı değilse z'ye false değeri atanır.

İlişkisel operatörler kullanılırken işlenenlerin aynı türden olmasına dikkat edilmelidir. Aksi takdirde, bir türden başka bir türe zorunlu dönüştürme meydana gelebilir ve bu da hatalı sonuca veya hata mesajına yol açabilir.

Örneğin, aşağıdaki komutta

```
bool evetHayir = 1 == '1';
```

'1' karakterinin tam sayıya örtük dönüştürülmesi gerçekleştirilir ve ardından 1 sabiti ile karşılaştırılır. Sonuç false olacaktır. Neden?

İlişkisel operatörlerin işlenenleri başka bir basit türden de olabilir. İfadelerde kullanıldığında, onlar zorunlu olarak mantıksal türe dönüştürülürler, yani: 0 değeri false ve sıfır olmayan tüm değerler true olarak dönüştürülür.

Örneğin,aşağıdaki komutlar:

```
double yaricap = 1.23;
bool evetHayir = yaricap > 0;
```

doğru yürütülür ve evetHayir mantıksal değişkeni, true değerini alır.

Gerçek sayıları karşılaştırırken, tam değerlerine değil, yaklaşık değer ile hesaplanmalarına dikkat edilmelidir.

Örneğin, aşağıdaki komutlarla:

```
float a = 1.0f / 5;
evetHayir = a == 0.2;
```

```
cout << "a = 1.0/5 = " << a << " 0.2'ye esit midir? " << daNeevetHayir<< endl;
```

sonuç false olacaktır, yani evetHayir değişkeni için 0 olacaktır.

```
a  = 1.0/5  = 0.2 0.2'ye esit midir? 0
```

Ancak işlenenler ve tüm ifade doğru yazılmış olabilir (derleyici hata bildirmez) fakat sonuç yanlış olabilir. Örneğin, aşağıdaki ifadenin değeri:

```
bool eniyiBasari = not == 4 || 5
```

not değişkeninin değeri ne olduğuna bağımsız olarak true olacak, çünkü || operatörünün sağ işleneni her zaman true değerine sahiptir.

Aslında, doğru ifade şu şekilde olmalıdır:

```
bool basari = not == 4 || not == 5;
```

not 4'ten daha küçük olunca, ifade false değerini alacak.

Benzer olarak, aşağıdaki komut:

```
bool siralanmis = a < b < c;
```

sözdizimsel olarak doğrudur fakat anlamsal olarak değildir, Böylece a=3 ve c=5 değerleri için yanlış sonuç verecektir çünkü önce 3<b karşılaştırılarak true veya false sonucu verir yani, 1 veya 0 değeri. Ardından, 1 < 5 veya 0 < 5'i karşılaştırır ve bu her zaman true sonucu verir. Ancak b = 6 için sonuç olarak false elde edilmelidir.

Doğru komut şu olacaktır:

```
bool siralanmis  = a < b && b < c;
```

Mantıksal ve ilişkisel operatörler, değeri true veya false olabilen **mantıksal ifadelerde** (İng.logical expressions) kullanılır.

Örnekler:

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

Aşağıdaki tabloda, **De Morgan kuralları** (İng. De Morgan's Laws) adı verilen eşdeğer mantıksal ifadeler verilmiştir.

İfade	Eşdeğer ifade
!(a && b)	!a || !b
!(a || b)	!a && !b

Programlardaki ifadeler oldukça karmaşık olabilir ve bunlarda aritmetik, mantıksal ve ilişkisel operatörler kullanılabilir. Bunları hesaplarken operatörlerin önceliğine uyulur, yani:

En yüksek	değil (NOT) !	Tekli +			Tekli –
	aritmetiksel	*	/	%	
		+	–		
	ilişkisel	<	<=	>	>=
		==	!=		

	mantıksal	(AND) &&						
		(OR) ||						
En düşük	atama	=	+=	–=	*=	/=	%=	

Mantıksal ifadeler, operatörlerin ve parantezlerin önceliğine göre soldan sağa doğru hesaplanır.

Belirli durumlarda, C++, ifadenin koşullu değerlendirmesini (İng. conditional evaluation veya short-circuit) gerçekleştirir.

Örneğin, aşağıdaki ifade hesaplanırken:

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

alfa = –30 o için sol alt ifade (alfa > 0) false değerini alır ve sağ alt ifadenin değerinden (alfa < 90) bağımsız olarak tüm ifadenin değeri false olur. Bu nedenle, && operatörlü ifade olması durumunda, sağ alt ifade hesaplanmıyor.

Benzer olarak, aşağıdaki ifade hesaplanırken:

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

x=-2 için, sol alt ifadenin değeri true olur ve sağ alt ifadenin değerinden bağımsız olarak tüm ifadenin değeri true olur. Bu yüzden, sağ alt ifade hesaplanmıyor.

Örnek 1.7.6

Bu örnek ile, iki mantıksal değişken a ve b'nin değerlerine bağlı olarak hesaplanan AND, OR ve NOT mantıksal fonksiyonlarının tablosu yazdırılır.

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

Şekil 1.7.6

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

Şekil 1.7.6 (devam)

Program yürütüldüğünde, çıktı şu olacak:

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

Press any key to continue . . .
```

Örnek 1.7.7

Aşağıdaki örnekle mantıksal veri türlerin kullanımı gösterilmektedir.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {     //Mantiksal veri turu
5
6       const double G = 9.81;  // Gercek sabitin bildirimi ve baslatilmasi
7       int aciAlfa ;           // Tamsayi degiskenin bildirimi
8       double ivme ;           // Gercek degiskenin bildirimi
9       char isaret = '*';      // Karakter degiskenin bildirimi ve baslatilmasi
10      bool evetHayir;         // Mantiksal degiskenin bildirimi
11      evetHayir = isaret == '*'; // isaret degiskeninin degeri * ise, o zaman
12                              // evetHayir mantiksal degiskeni 1 (true)
13                              // aksi takdirde 0 (false) degeri alir.
14      cout << evetHayir  << "isareti " << isaret  <<  "'dir" << endl;
15      ivme = 9.9;
16      evetHayir = ivme < G; // ivme degiskeninin degeri G sabitinin
17                            // degerinden daha kucukse, o zaman evetHayir
18                            // mantiksal degiskenin  degeri 1 (true) olur
19                            // aksi takdirde degeri 0 (false) olur.
20      cout << evetHayir << ": " << ivme << " < " << G << endl;
21      aciAlfa = 60;
22      evetHayir = (aciAlfa > 0 ) && (aciAlfa < 90 );
23      cout << evetHayir << ": " << aciAlfa  << " dar acidir. " << endl;
```

Şekil 1.7.7

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

Şekil 1.7.7 (devam)

Program yürütüldükten sonra aşağıdaki çıktı elde edilecek:

```
1: isaret *'dir
0: 9.9 < 9.81
1: 60 daracidir

Press any key to continue . . .
```

Bitsel Operatörler

C++, bitsel operatörler (İng. bitwise operators) olarak adlandırılan başka bir mantıksal operatörler grubuna sahiptir. Bu operatörler ile mantıksal işlemler bit bit gerçekleştirilir.

&	bitler için mantıksal ***VE***	~	tümleyen
|	bitler için mantıksal ***VEYA***	<<	sola kayma
^	bitler için Özel ***VEYA***	>	sağa kayma

p ve q bitleri üzerine &, | ve ^ mantıksal operatörlerinin tablosu aşağıda verilmiştir:

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

İkili sayının tümleyeni bitlerin olumsuzlanmasıyla elde edilir, 1'den 0'a ve 0'dan 1'e. Örneğin,

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

sayısının tümleyeni

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

sayısıdır.

Kaydırma operatörleri, bitleri sola veya sağa kaydırmak için kullanılır (İng.shifting). Bu yüzden, kaydırma operatörleri veya **shift** operatörleri olarak adlandırılırlar. Örneğin, m << 1 ile m sayısının bitleri 1 basamak sola kaydırılır (= 000… 01101010 = 106_{10}) ve m >> 1 ile bitler 1 basamak sağa kaydırılır (= 000…00011010 = 26_{10}).

Örnek 1.7.8

a = 00000101 (= 5_{10}), b = 00001110 (= 14_{10}) ikili sayıları verilmiş olsun. Aşağıdaki işlemlerin değeri yazdırılsın:

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

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

int main() {     // Bitsel operatorler
    int a = 5, b = 14;
    cout << "-------------" << endl;
    cout << " a=00000101" << endl;
    cout << " b=00001110" << endl;
    cout << "-------------" << endl;
    cout << "a & b   | " << ( a & b ) << endl;
    cout << "a | b   | " << ( a | b ) << endl;
    cout << "a ^ b   | " << ( a ^ b ) << endl;
    cout << "~a      | " << ( ~a ) << endl;
    cout << "a << 1  | " << ( a << 1 ) << endl;
    cout << "a >> 1  | " << ( a >> 1 ) << endl;
    cout << "-a >> 1 | " << ( -a >> 1 ) << endl;
    cout << "-------------" << endl;

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

Şekil 1.7.8

Programla şu tablo elde edilecektir:

```
-------------
 a=00000101
 b=00001110
-------------
a & b   | 4
a | b   | 15
a ^ b   | 11
~a      | -6
a << 1  | 10
a >> 1  | 2
-a >> 1 | -3
-------------
```

Bitsel operatörlerde mantıksal işlem, karşılık gelen bitler üzerinde gerçekleştirilir. Sonuç, herhangi bir tamsayı değeri olabilir. Böylece,

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

Mantıksal operatörler ile bitsel operatörler arasındaki farka dikkat edilmelidir. Mantıksal operatörler, sıfırdan farklı herhangi bir değer ,mantıksal olarak doğru kabul edilerek bir veya ikiden fazla işlenen üzerine uygulanır.Bu arada, işlemin sonucu 0 veya 1 olabilir.

Örneğin bir önceki tabloda bulunan değerler için: a && b = 1 ve a || b = 1, çünkü a ve b'nin 0'dan farklı değerleri vardır

&& ve || mantıksal operatörleri kullanırken, bunların & ve | bitsel operatörleri ile değiştirilmemesine dikkat edilmelidir. Örneğin, a = 5 ve b = 14 için aşağıdaki ifadeler:

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

farklı sonuca sahiptir.

Bitsel operatörlerin kısaltılmış biçimleri şunlardır:

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

Alıştırma Ödevleri

1. a = true ve b = false için (a || !b) && !(a || b) mantıksal ifadesinin değerini (doğru veya yanlış, yani 1 veya 0) hesaplayacak program yazılsın.
2. a = 7, b = 3 ve c = 4 için aşağıdaki ifadenin değerini bulmak için program yazılsın: !(a – b < 5 * c) && (b * c % a + 1 == a * b) ||(c < b).
3. 12 ile 2^5 tam sayısının çarpımını ve bölümünü bulan program yazılsın. (Yönerge: m tam sayısının 2^n ile çarpımı ondalık noktasını n basamak sağa kaydırarak, bölme işlemi ise ondalık noktasını n basamak sola kaydırarak gerçekleştirilir.Dolayısıyla, kaydırma operatörleri >> ve << kullanılabilir).

Sayılı Veri Türü

Sayılı türdeki veriler, sayılar değil, tanımlayıcılar olması gereken önceden tanımlanmış sabitler kümesinden (listeden) değerler alabilir. Ayrıca, sabitlerin değerlerinin büyük harflerle yazıldığını biliyoruz.

Sıralı veri türü, **enum** (İng.enumerate) kelimesi ile tanımlanır, listedeki değerler ise büyük parantezler içinde koyulur. Büyük parantezlerin çıkış kısmından sonra noktalı virgül işareti koyulmaz.

Örnekler:

```
enum temelRenk {KİRMİZİ, YESİL, MAVİ, SARİ, KAHVERENGİ}
enum mevsim  {İLKBAHAR, YAZ, SONBAHAR, KİS}
enum dogruYanlis  {TRUE, FALSE}
```

Listedeki her ismin kendi sıra numarası vardır, yani: 0, 1, 2, 3, vb. Örneğin KİRMİZİ 'nın sıra numarası 0, YESİL'in 1, MAVİ'nin 2'dir.

Yukarıdaki örnekte, 3 sayılı tür tanımlanmıştır: temelRenk, mevsim ve dogruYanlıs.

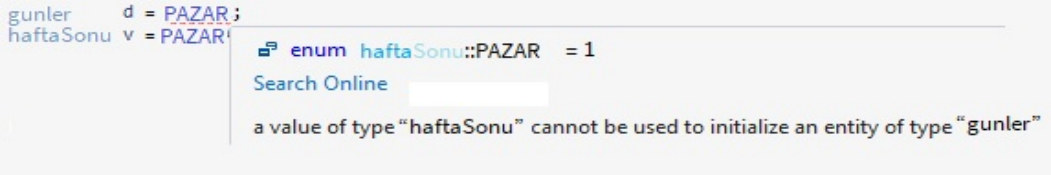
Aşağıdaki sıralı türlerin tanımları yanlıştır, çünkü listelerdeki sabitler tanımlayıcı değildir, yani C++'daki tanımlayıcılar için kurallara göre oluşturulmamıştır:

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

Sıralı türlerin aşağıdaki iki tanımı doğrudur:

```
enum gunler {PTS, SAL, CAR, PER, CUM, CTS, PAZAR}
enum haftaSonu {CUMARTESİ, PAZAR}
```

Ancak ikisi aynı programda kullanılamazlar, çünkü PAZAR sabiti ikinci tanımla yeniden tanımlanmaktadır.



Sıralı türden değişkenlerin bildirimi, türü belirtilerek,ardından değişkenler listesiyle yapılır.

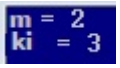
Örnekler:

```
temelRenk k, y, m;
mevsim  sb, ki, ya;
```

Bu değişkenler sadece türün tanımı sırasında listede belirtilmiş sabitlerden değerler alabilir:

```
m = MAVİ;
ki = KİS;
```

Sıralı türdeki değişkenler yazdırılamaz, m ve ki değişkenlerini yazdırırken, tanım listesindeki sıra numaraları yazdırılır:



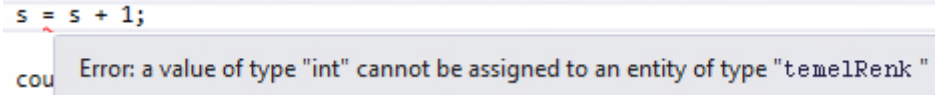
çünkü MAVİ'nin sıra numarası 2 ve KIS'ınki 3'tür.

Sıralı türdeki değişkenler dolaylı olarak artırılamaz.

Örneğin, aşağıdaki komut:

```
m = m + 1;
```

derlenmez bile. Hemen hata oluşur.



Hata, 1 ile toplarken, m değişkeninin zorunlu olaral tamsayı 2'ye (tanım listesindeki sıra numarası) dönüştürülmesi ve 1 ile toplanması nedeniyle oluşur. Sonuç, sıralı m değişkenine atanamayan tamsayı değeri 3'tür.

Bu, açık dönüşümle önlenebilir:

```
m = temelRenk(m+1)
```

bundan sonra m değişkeni SARİ değerini alacaktır.

Sıralı tür sabitlerine, tür tanımı sırasında da (sıra sayıları yerine) değerler atanabilir.

Örnekler:

```
enum not {ENİYİ = 5, ÇOKİYİ = 4, İYİ = 3, YETERLİ = 2};
enum aylar {OCAK = 1, OCAK , MART, NİSAN, MAYİS, HAZİRAN, TEMMUZ,
              AGUSTOS, EYLUL, EKİM, KASİM, ARALİK}
```

İkinci örnekte, OCAK'tan ARALIK'a kadar sıra sayıları 0'dan 11'e değil, 1'den 12'ye kadar olacaktır.

Sıralı türdeki veriler karşılaştırılabilir. Bu arada, tanım listelerindeki sıra sayıları karşılaştırılır.

Anonim sıralı türü de bildirilebilir (İng. anonymous enume ration type). Örneğin:

```
enum {elma, armut, seftali, erik, kiraz, visne, kayisi};
```

Sırali türün tanımı main() fonksiyonundan önce yapılır, ***şekil 1.7.9.***

Örnek 1.7.9

Bu örnekle, sıralı türün tanımını ve sıralı türdeki değişkenlerin bildirimi gösterilmiştir.

Şekil 1.7.9'daki program yürütüldükten sonra, aşağıdaki çıktı elde edilir:

```
kRenk  = 0,  mRenk = 5
Renkler ayni midir ?       0
ki =KIS'ın degeri 3'tur
```

```
1   #include <iostream>
2   using namespace std;
3
4   enum   temelRenk {
5          KIRMIZI, YESIL, MAVI
6   };
7   enum       mevsim {
8          İLKBAHAR, YAZ, SONBAHAR, KIS
9   };
10
11  int main() {     // Sayili veri turu
12      temelRenk kRenk, mRenk;
13      mevsim ki = KIS;
14      bool dogruYanlis;
15
16      kRenk = KİRMİZİ;
17      mRenk = MAVİ;
18      dogruYanlis = kRenk == mRenk;
19      cout << "kRenk = " << kRenk << ", mRenk = " << mRenk << endl;
20      cout << "Renkler ayni midir ? " << dogruYanlis << endl;
21
22      cout << " ki = KIS'ın değeri: " << ki << endl;
23
24      cout << endl;
25      system( "Color 17" );
26      system( "pause" );
27      return 0;
28  }
```

Şekil 1.7.9

Alıştırma Ödevleri

1. Sabitler listesi PAZARTESİ'den PAZAR'a kadar olan günlerin adlarını içeren, haftadakiGunler adlı sıralı türü tanımlayacağınız program yazın.Ardından, haftadakiGunler türünden yedi değişken bildirin ve her birine haftadakiGunler türünden rastgele sıralı bir sabit atayın. haftadakiGunler türünden başka bir değişken tanımlayın. Örneğin, gun ve ona CARSAMBA değeri atayın. Ondan sonra, dogruYanlis mantıksal değişkeni bildirin ve buna gun değişkeninin içeriğini yedi değişkenin her biri ile karşılaştırarak elde edilen değer atanacak. dogruYanlis değişkeninin alacağı değerleri yazdırın. (***Şekil 1.7.9***'daki programa bakınız).

2. Sıralı türünde gunlukDersler değişkeni bildireceğiniz program yazılsın:

```
enum gunlukDersler {
      PZR  = 4, SAL  = 6, CAR  = 5, PER T = 5, CUM  = 3
       };
```

Haftadaki toplam ders sayısı hesaplansın.

3. Sıralı türünden notlar adlı bir değişkeni bildireceğiniz program yazılsın. notlar türü tanımı, 12 dersin adını ve her ders için notu içerir. Örneğin, {MAK = 4, BİL = 5...}. Öğrencinin ortalama başarısı hesaplansın.

DİZELER

1.6 C++'a Giriş alt bölümünün **Yazdırma komutu** ve **Veri türleri** alt başlıklarında, **dize** (İng. string) terimini tanıdık. Dizeler, karakter dizgileri olarak karakter verilerinden oluşan verilerdir. Bu tür veriler, basit char türünden oluşan ayrı yapılar olarak ele alınır ve string türü olarak adlandırılır.

Bu verilere ayrıca **metinsel veriler** de denir, çünkü değerleri tırnak işaretleri içine alınmış metindir.Örneğin, metin (dize) sabiti. Örneğin, C++'daki dize sabitleri şunlardır:

```
"Merhaba, nasilsiniz "
"Goce Delcev cad. No.123."
"Makedonya incil ulkesi."
```

C++'da dize sabitleri harfler (alfabeden büyük ve küçük harfler), rakamlar ve +, –, *, / $ gibi özel karakterler içerebilir.

Dize sabitlerinin temel özellikleri şunlardır:

- Tırnak işareti içinde yazılır.
- Her karakterin en soldaki 0 dizini ile başlayarak bir dizini vardır.
- Metin sabitinin uzunluğu, boşluklar dahil olmak üzere karakter sayısına eşittir.

- Bir dize sabitinin maksimum uzunluğu 2048 bayttır.

Dize türündeki değişkenler, herhangi bir basit türün değişkenleri gibi bildirilebilir ve onlara dize sabitleri atanabilir.

Örnekler:

```
string benimAdim;           // benimAdim degiskenin bildirimi
string evetHayir;
string tarihBugun = "01.01.2017"; //Bildirme ve baslatma
string tarihBugun("01.01.2017");  // veya (aynen) bildirme ve
                                  // baslatma
string dyildizlar(60, '*'); // Bildirme ve baslatma
```

String türü olarak bildirilmiş değişkene değer atama, = atama işareti ile yapılır.

Örnekler:

```
benimAdim = "Gjorgji Jovancevski";
evetHayir = 'D';                  // Bunu yapabiliriz
tarihYarin  = "21 Haziran 2053";
```

Dizedeki her karakterin kendi dizini (indisi) olduğunu söylemiştik. Örneğin, benimAdim[0] 'G'dir, benimAdim[5] 'j''dir, tarihYarin[11] '3' 'tür vb.

"A" ve 'A' verilerinin aynı olmadığını vurgulayalım. İlk veri dize türündendir, ikinci veri ise char türündendir.

Dizelerle farklı işlemler gerçekleştirilebilir, örneğin: birleştirme (**Veri Türleri** alt başlığında bahsetmiştik), iki dizeyi karşılaştırma (İng. comparation), bir dizenin uzunluğunu belirleme (İng.length) ve diğerleri.

Bu işlemler, C++ standart kütüphanesinin <string> kütüphanesindeki fonskiyonlarla gerçekleştirilir. Bu yüzden, programlarda kullanabilmek ve dize türündeki değişkenleri bildirebilmek için bu kütüphane #include direktifi ile programın başında yer almalıdır:

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

Örnek 1.7.10

Şekil 1.7.10'daki programla dize değişkenlerin ve dize sabitlerin kullanımı gösterilmiştir.

Programın yürütülmesiyle ortaya çıkan bir çıktı şudur:

```
ki en buyuk Makedon:
skender Filip MAKEDON ve Goce Nikola MAKEDON
once True: İskender Filip MAKEDON, Goce Nikola MAKEDON'dan yasamis
Press any key to continue . . .
```

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

const string soyad = "MAKEDON";
const string bosYer = " ";

int main() {    // Dize verileri

    string benimAdim ' İskender Filip";
    string seninAdin = "Goce Nikola";

    string enbuyukMakedonlar = benimAdim + bosYer + soyad;
    enbuyukMakedonlar += " ve ";
    enbuyukMakedonlar += seninAdin + bosYer + soyad;
    cout << "İki en buyuk Makedon: " << endl;
    cout << enbuyukMakedonlar << endl;

    bool evetHayir = benimAdim < seninAdin;
    cout << boolalpha << evetHayir << ": " << benimAdim << soyad
        << seninAdin << bosYer << soyad << "'dan once yasamis"
        << endl;

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

Şekil 1.7.10

Alıştırma Ödevleri

1. Adınız ve soyadınızın değerlerinin atanacağı iki dize değişkeninin bildirileceği program yazılsın. Değeri olarak, ebeveyninizin adının ilk harfi atanacağı işaret değişkeni de bildirilsin. Üç değişken birleştirilsin ve aşağıdaki örnekteki gibi biçimde yazdırılsın:
   `Gjorgji J. Jovancevski`
2. Dize değişkenlerinin bildirileceği ve bunlara aşağıdaki dizi sabitlerinin atanacağı program yazılsın: `"GRA"`, `"LOJİ"`, `"BIYO"`, `"Fİ"`, `"CO"` `"FYA"` ve `"JEO"`. Ardından, birleştirerek şu kelimeler oluşturulsun ve yazdırılsın: `BİYOGRAFI`, `BİYOLOJİ`, `COGRAFYA` ve `JEOLOJİ`
3. İki dize değişkeni ve iki karakter değişkeninin bildirileceği ve bunlara : `"KAL"`, `"LAK"`, 'A' ve 'B'. sabitlerinin atanacağı program yazılsın. Ardından, onları birleştirerek anlamı olan enuzun

palindrom kelime oluşsun. (palindrom, hem ileri hem de geri aynı şekilde okunan kelime veya cümledir. Örneğin, palindromlar şunlardır: ada, küçük, el ele, vb.).

Verilerin Türünü Yeniden Adlandırma

Çoğu zaman, programın okunabilirliği ve içinde daha kolay bulunmamız için, bize bir şeyi hatırlatacak yerleşik veri türlerinin adları için eş anlamlıların (diğer adlar) kullanılması tavsiye edilir.

Örneğin, aşağıdaki bildirimlerle:

```
int i, j, k;
double x, y, z;
char a, b;
bool evethayır, bulundu;
```

int, double, char ve bool türünden değişkenler bildirilir.

C++'da var olan veri türü için başka bir ad da verilebilir. (Bu arada yeni tür oluşturulmaz). Yeni tür adının değer aralığı ve işlemleri, mevcut türle aynıdır.

Bir türü yeniden adlandırmak için **typedef** anahtar sözcüğü kullanılır.

Örneğin:

```
typedef int tamsayi;
typedef double gercek;
typedef char isaretler;
typedef bool mantiksal;
```

Ardından tamsayi türü int türüyle, gercek türü double türüyle eşit olarak bildirim gerçekleştirilebilir.

Örneğin:

```
int sayi = 5;
tamsayi sayi1 = 8;
char isaret = '$';
isaretler isaret1 = '@';
bool evetHayir = 1;
mantiksal evetHayır1 = true;
cout << sayi << ' ' << sayi1 << endl;
cout << isaret << ' ' << isaret1 << endl;
cout << evetHayir << ' ' << boolalpha << evetHayir1 << endl;
```

Yukarıdaki program bölümünün çıktısı şudur:

```
5 8
$ @
1 true

Press any key to continue . . .
```

Türün Otomatik Belirlenmesi

Bir değişkenin bildirimindeki türü, başlatıldığı veya bir değer atandığı veri türüne göre belirlenebilir. Bu, **auto** anahtar kelimesi ile yapılır.

Örneğin, ***Şekil 1.7.10***'daki programda değişken bildirimi auto kelimesiyle belirtilirse, program doğru şekilde yürütülecektir:

```
auto benimAdim = "İskender Filip";
auto seninAdin = "Goce Nikola";
auto enbuyukMakedonlar = benimAdim + bosYer + soyad;
auto evetHayir = benimAdim < seninAdin;
```

auto anahtar kelimesi, çoğunlukla, değişkenin başlatılması için ifade karmaşık olduğunda, yani içinde çeşitli veri ve fonksiyonlar kullanıldığında ve ifade hesaplanırken sonucun türü hemen belirlenemediğinde kullanılır. Derleyici, ifadeyi hesaplarken sonuç türünü belirler.

Bilgiyi Kontrol Etme Soruları

1. Hangi tamsayı veri türlerini (değer aralığına göre) biliyorsunuz?
2. unsigned olarak bildirilen tamsayı değişkenleri hangi değerleri alır?
3. Short değişkenine 100 000 değeri atanırsa ne olur?
4. Tamsayılarla işlemler için operatörleri listeleyin.
5. Tamsayı türü değişkenlerin bildirimiyle ve bildirme ve başlatılmasıyla ilgili birkaç örnek verin.
6. Tamsayılar için aritmetik operatörleri kısaltılmış biçimde belirtin.
7. % = operatörü ne anlama geliyor?
8. Hangi veriler için gerçek türden olduğunu söylüyoruz?
9. C++'da hangi gerçek veri türleri vardır?
10. Gerçek türü değişkenlerin bildirimiyle ve bildirme ve başlatılmasıyla ilgili birkaç örnek verin.
11. Kayan noktalı ondalık verilerin gösterimini açıklayın.
12. Gerçek veriler türünün yazdırma biçimlerini açıklayın.
13. 34400.811 numarasını e-biçiminde gösterin.
14. Gerçek türü değişkenlerin bildirimiyle ve bildirme ve başlatılmasıyla ilgili birkaç örnek verin.
15. Örtük tür dönüştürme nedir ve diğer adı nedir?
16. Açık tür dönüştürme nedir ve diğer adı nedir?
17. int türünün örtük dönüşümü hangi türlere yapılabilir?

18. Daha yüksek veri türünün daha düşük veri türüne açık dönüşümü nasıl gerçekleştirilir?
19. 0.12 * 5.3 + 123 ifadesini int türüne dönüştürme komutunu yazın?
20. Şu komutu açıklayın: x = x = y;
21. Arttırmanın ne anlama geldiğini ve azaltmanın ne olduğunu açıklayınız.
22. Hangisi arttırma operatörü, hangisi azaltma operatörüdür?
23. Karakter değişkenlerin bildirimi ile ilgili örnekler verin.
24. char türü ne kadar bellek alanı kaplar?
25. Hangi mantıksal sabitleri biliyorsun uz? Bunların dışında neden başkaları yok?
26. Mantıksal operatörleri sayın.
27. Hangi ilişkisel işlemleri biliyorsunuz? Uygun operatörleri belirtin.
28. !(a && b) || (!a || b) ifadesinin a = true ve b = false için sonucu ne olacak?
29. Aşağıdaki bildirme ve başlatma doğru mudur:
    bool evetHayir = x > y < z;?
30. Bitsel operatörleri sayın.
31. Aşağıdaki işlemlerin sonucunu bulun:
    a & b, a | b ve a ^ b eğer a = 00010110 ve b = 00010101
32. Sayılı tür verilerin değerleri nasıldır?
33. Sayılı veri türünün tanımı nereye yerleştirilir?
34. Sayılı türün tanımı ve o türdeki bir değişkenin bildirimi için örnek veriniz.
35. Sayılı türdeki veriler nasıl yazdırılır?
36. Metinsel veri türü basit türlere ait midir?
37. Dize sabitin bildirimi için örnek yazın.
38. Dize değişkenin bildirimi için örnek yazın.
39. Eğer typedef int tam;

```
typedef int celi;
```

tanımlanmışsa o zaman aşağıdaki değişkenler hangi basit tür ile bildirilecek?

```
tam a, b, c; ?
```

40. Derleyicinin bir değişkeni başlatırken türünü belirleyebilmesi için hangi sözcük kullanılır?

1.8 Verilerin Okunması ve Yazdırılması

Sayısal Verilerini Okuma ve Yazdırma

C++'da verilerin okunması, karakterler **akışı** (İng. stream) olarak düzenlenmiştir. Standart girdi ve çıktı akışı, <iostream> kütüphanesinde bulunan fonksiyonlar (rutinler[41], İng. routines) yardımıyla gerçekleştirilir ve her programda aşağıdaki yönergeyle dahil edilir:

```
#include <iostream>
```

<iostream> kütüphanesinde, girdi ve çıktı işlemlerini gerçekleştirmek için gerekli olan <<42 ve >> operatörleri, cin, cout, endl vb. komutları tanımlanmıştır.

Ayrıca, belirli işlemlerin gerçekleştirilmesi için fonksiyonlar içeren diğer kütüphanelerin de programa dahil edilmesi gerekir. Örneğin, programda dizeler kullanırsak, #include yönergesi ile <string> kütüphanesinin dahil edilmesi gerekir:

```
#include <string>
```

#include yönergesi ile başlayan program satırları çevirici tarafından çevrilmez, önişlemci (İng. preprocessor) olarak adlandırılan özel program tarafından işlenir. Önişlemci programda #include yönergelerinde listelenen tüm dosyaları (İng. files) içerir. Bu dosyalar köşeli parantezler içine koyulur ve onlarla önişlemciye, bunları **C++ standart kütüphanesinde** include dizini olarak adlandırılan dizininde araması bildirilir. Bunlara **başlık dosyaları** (İng. header files) denir.

Verileri okuma ve yazdırma operatörleri şunlardır:

>	girdi operatörü,
<<	çıktı operatörü

> << operatörü "şuraya gönder" anlamına gelir, >> operatörü ise "şuradan al" anlamına gelir. Girdi ve çıktı komutları şunlardır:

cin	Klavye aracılığıyla değerler girme komutu (standart girdi akışı için).
cout	Değerleri ekrana yazdırma komutu (standart çıktı akışı için).

Veri değerlerinin okunması sırasında boşluklar, sekmeler ve bitiş karakterleri (eng. enter) atlatılır. Değerlerin okunması boş olmayan ilk karakterden başlar.

Örneğin, i ve x değişkenleri bildirilirse:

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

[41] Kısa programlar.
[42] Bu operatörü şimdiye kadar da yazdırmak için kullanıyorduk.

Onlar için değerlerin klavyeden okunması, şu komut ile yapılır:

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

tamSayi ve gercekSayi değişkenlerine şu değerler atanırsa:

```
tamSayi = 123;
gercekSayi = 456.78;
```

o zaman aşağıdaki komutla:

```
cout << tamSayi << gercekSayi;
```

Şu yazdırılacak:

```
123456.78
```

Yazdırılan değerleri ayırmak için, karakter sabiti olarak boşluk kullanılabilir. Böylece, aşağıdaki komutla:

```
cout << tamSayi << ' ' << gercekSayi << endl;
```

Şu yazdırılacaktır:

```
123 456.78
```

endl (daha önce de kullandığımız) satır sonu için operatördür, yani sıradaki satıra geçmek için kullanılır:

Birden fazla boş yer bırakmak gerekirse, genellikle dize sabit kullanılır.

Örneğin, aşağıdaki komutla:

```
cout << tamSayi << "   " << gercekSayi << endl;
```

Şu yazdırılacaktır:

```
123    456.78
```

Yeni satıra yazdırmak için önceki örneklerde açıkladığımız ve kullandığımız ' \n' karakteri kullanılır.

Örneğin, aşağıdaki iki komutla:

```
cout << tamSayi  << '\n' << gercekSayi  << '\n';
cout << tamSayi  << endl << gercekSayi  << endl;
```

Aynısı yazdırılacak:

```
123
456.78
```

Çıktı komutunda özel anlamları olan birkaç özel karakter, **1.6 C++'ya Giriş** alt noktasında **Yazdırma Komutu** alt başlığında belirtilen çıkış (escape) dizileri kullanılarak yazdırılmalıdır.

C++'da tek satırda bulunan yorumlar, yorumdan önce // karakteri konularak yazılır.

Satır sonuna kadar olmayan yorum /* ve */ işaretleri arasına yerleştirilir.

Örneğin:

```
/*     Bu program ile temel bilgiler iceren
       en basit bir kartvizit yapılır:
       - Sirketin adı
       - telefon
       - E-posta
*/
```

```
#include <iostream>
```

önişlemci yönergesi, önişlemciye girdi-çıktı işlemleri için fonksiyon tanımlarını içeren <iostream> dosyasını dahil etmesini söyler.

C++'da değişkenler programın herhangi bir yerinde bildirilebilir ve sadece bildirim satırından sonra kullanılabilir, ondan önce kullanılamaz.

Aşağıdaki komut:

```
cout << "İlk tam sayiyi gir: ";
```

belirtilen metni monitör ekranına yazdırmak için standart çıktı akışı cout ve << operatörünü kullanır. Önceki komut yürütüldüğünde, cout standart çıktısına " İlk tam sayiyi gir" karakter akışı gönderir. Bu komutu şu şekilde okuyabiliriz: “ " İlk tam sayiyi gir " metni cout’a karakter akışı olarak gönderilir".

Sıradaki komut:

```
cin >> sayi1;
```

Standart girdi akışı cin ve klavye aracılığıyla girilen verileri karakter dizisi biçiminde almak için >> operatörünü kullanır. Bu komut "cin klavye aracılığıyla girilen karakter akışını okur ve sayı1'e atar" şeklinde okunabilir. Bu durumda, girilen karakterler tamsayıya dönüştürülen sayının rakamlarıdır ve sayı1 tamsayı değişkenine atanır.

Birkaç örnek vereceğiz.

Örnekler

Örnek 1.8.1

Şekil 1.8.1’de, metnin önceki bölümünde incelenen bazı özellikleri göstereceğimiz iki sayıyı toplamak için basit bir C++ programı gösterilmiştir.

```
1   // İki sayinin toplanmasi
2   #include <iostream>
3   using namespace std;
4
5   int main() {                    // main ( ) ana fonsiyonun baslangici.
6       int broj1;                  // sayi1 tamsayi degiskeninin bildirimi.
7       cout <<" İlk tam sayiyi girin: " ;
8       cin >> sayi1;               // sayi1 degiskeni degerinin okunması.
9       int sayi2, toplam;          // sayi2 ve toplam tamsayi
10                                  // degiskenlerinin bildirimi
11      cout <<" İkinci tam sayiyi girin: " ;
12      cin >> sayi2;               // sayi2 degiskeni degerinin okunması.
13      toplam = Sayi1 + sayi2        /* iki degiskenin degerlerini
14                                 toplama komutu */
15      cout << " Toplam: " << toplam  << endl;
```

Şekil 1.8.1

```
16
17          cout << endl;
18          system("Color 17");
19          system("pause");
20          return 0;
21
22      } //main ( ) ana fonkisyonun sonu
```

Şekil 1.8.1 (devam)

Örnek 1.8.2
Tam sayı türünden veriler bildirilsin, girilsin ve yazdırılsın.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // Klavye ile girilen tamsayi verileri okumak ve yazdirmak
6       unsigned short tarihAy ;
7       unsigned short tarihGun;
8       unsigned int  tarihYil;
9       long long sayi19basamakli;
10      cout << "Gun giriniz: "; cin >> tarihGun;
11      cout << "Ay giriniz: " ; cin >> tarihAy;
12      cout << "Yil giriniz: " ; cin >> tarihYil;
13      cout << endl;
14      cout << "Bugun tarih: " << tarihGun  << "." << tarihAy  << "."
15          <<  tarihYil << endl;
16      cout << "\n 19 rakam giriniz: ";        cin >> sayi19basamakli;
17      cout << "\nGirdiginiz sayi: " << sayi19basamakli << endl;
18
19      cout << endl;
20      system( "Color 17" );
21      system( "pause" );
22      return 0;
23  }
```

Şekil 1.8.2

Şekil 1.8.2'deki programın çıktısı şu olacak:

```
Gun giriniz:       01
Ay giriniz:          01
Yil giriniz:           2017

Bugun tarih: 1.1.2017

19 rakam giriniz: 1234567890123456789

Girdiginiz sayi:  1234567890123456789
Press any key to continue . . .
```

Örnek 1.8.3

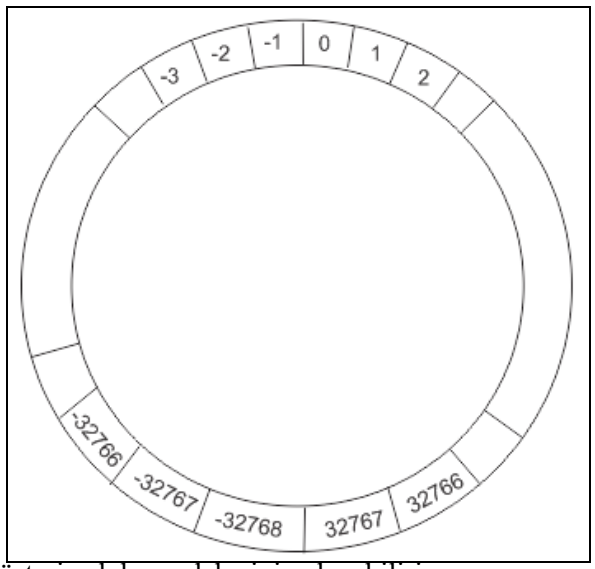
Gerçek türünden veriler bildirilsin, girilsin ve yazdırılsın.

Sıradaki programda (***Şekil 1.8.3***) iki gerçek sayı okunur ve onların aritmetik ortalaması yazdırılır.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       //      Klavye ile girilen tamsayi verileri okumak ve yazdirmak
6       float sayi1k;
7       double sayi1kinci, sayiOrtalama;
8       cout << "ilk ondalik sayiyi giriniz: "; cin >> sayi1k;
9       cout << "ikinci ondalik sayiyi giriniz: " ;  cin >> brojVtor;
10      sayiOrtalama = ( sayi1k + sayi1kinci ) / 2;
11      cout << sayi1k  << "ve " << sayi1kinci << "\nsayilarının ortalaması: "
12          << sayiOrtalama << endl;
13
14      cout << endl;
15      system( "Color 17" );
16      system( "pause" );
17      return 0;
18  }
```

Şekil 1.8.3

Örnek 1.8.4



Şekil1.8.4'te, değişkenlerin türüne göre verilen sayıların aralığı aşmasıyla ilgili örnek gösterilmiştir.

Bu ödev, tür tanımına göre değer aralığından daha büyük değer aldığı için toplam değişkeninin yanlış kullanımını gösterir. toplam değişkeni, maksimum değeri 32.767 olan short türündendir, hesaplamada ise 60.000 değerini alarak, short türündeki (-32768 ila -1) negatif tam sayı aralığına geçiyor. Tam sayı veri türünün kapsamını kapalı (dairesel) şekilde gösterirsek bunu daha iyi anlayabiliriz.

Görüldüğü gibi 32 768 sayısı (aralığın üzerinde olan) –32 768 sayısıyla temsil edilecek, 32 769 sayısı

ise –32 767 sayısıyla temsil edilecektir. Bu yüzden, toplam değişkeninin değeri içink şunu elde edeceğiz:

toplam = –32 768 + (60 000 – 32 767) – 1 = –5 536

Değişkenin değeri, değişkenin bildirildiği türün kapsamı dışına çıkması, **taşma (aşıma)** (İng. overflow) olarak adlandırılır.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       short birimFiyati, toplam;      /* Degiskenlerin bildirimi*/
6       short adet      = 1000;         /* Bildirme ve baslatma*/
7       birimFiyati     = 60;           /* Deger atama */
8       toplam = birimFiyati * adet;
9       cout << "Birim fiyati " <<birimFiyat << "denar olmak uzere ";
10      cout << adet << "adet icin toplam fiyat "
11          << toplam << "denardir." << endl;
12
13      cout << endl;
14      system( "Color 17" );
15      system( "pause" );
16      return 0;
17  }
```

Şekil 1.8.4

```
Birim fiyati 60 denar olmak uzere 1000 adet için toplam fiyat -5536 denardir.
Press any key to continue . . .
```

Karakter ve Dize Verilerini Okuma ve Yazdırma

Önceki bölümlerde, dizelerin birleştirme, karşılaştırma ve diğer gibi işlemlerin gerçekleştirilebildiği karakter dizileri olduğunu söylemiştik.

Birleştirme + işaretiyle gerçekleştirilir, işlenenler ise şunlar olabilir:

- Karakter veya dize değişmezleri:

```
' ', '@', "proglamlama"
```

- const sözcüğüyle bildirilen adlandırılmış karakter veya dize sabitleri:

```
const char bosluk = ' ';
const string artiarti = "++";
```

- Bildirilen karakter veya dize değişkenleri:

```
char CHarfi  = 'C';
string baslik  = "temelleri";
```

Aşağıdaki komutla:

```
baslik = CHarfi  + " " + "artiarti " + ' ' + "bosluk "
+ dilinde + programlama  + baslik;
```

Baslik değişkeni şu değeri alacaktır:

```
"C++ diline programlama temelleri"
```

Dize sabitlerini birleştirirken, + işaretinin işlenenlerinden en az biri, dize değişkeni veya adlandırılmış sabit olmalıdır. Aksi takdirde, hata meydana gelir.

```
string ab = "aaa" + "bbbb";
```

expression must have integral or unscoped enum type

Doğrusu şöyledir:

```
string a = "aaa";
string ab = a + "bbbb";
//  veya
string ab = string("aaa") + "bbbb";
```

Sayısal verilerin (int, float, double türü) dizelerle birleştirilemeyeceğini belirtelim.

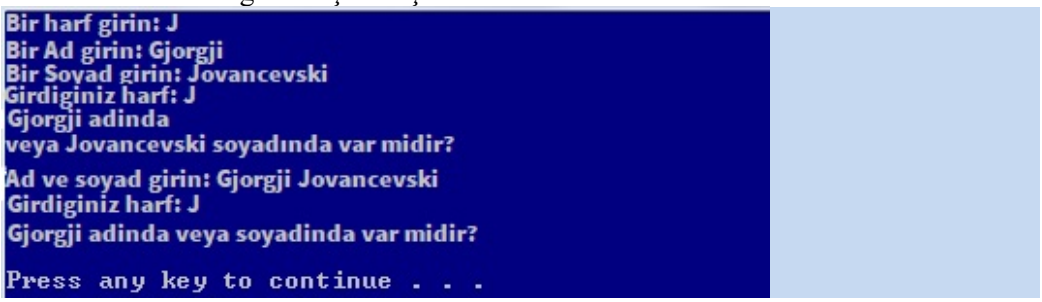
Örnek 1.8.5

Şekil 1.8.5'te karakter ve dize değişkenlerinin okunması ve yazdırılması gösterilmiştir.

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

int main() {
    char birHarf;
    string benimAdim, benimSoyadim, benimAdimVeSoyadim;
    cout << " Bir harf girin: " ; cin >> birHarf;
    cout << " Bir Ad girin: "; cin >> benimAdim;
    cout << " Bir Soyad girin: "; cin >>benimSoyadim;
    cout << " Girdiginiz harf: " <<birHarf << endl;
    cout <<  benimAdim <<"adında" << endl;
    cout <<  "veya" <<benimSoyadim <<" soyadında var midir? " << endl;
    cout << endl;
    cout << " Ad ve soyad girin: " ;
    char sonİsareti;
    cin.get(sonİsareti );
    getline( cin, benimAdimVeSoyadim );
    cout << " Girdiginiz harf: " << birHarf << endl;
    cout << benimAdimVeSovadım
        << " adinda veya soyadında var mıdır? " << endl;

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

Şekil 1.8.5

Programın çıktısı şu olacaktır:



Not: benimAdimVeSoyadım dize değişkeni için veri girerken "Gjorgji Jovancevski" dizesi girilmesine rağmen, aynı değişken yazdırılırken sadece "Gjorgji" yazdırılır. Bunu devamda açıklayacağız.

Biçimlendirilmiş Yazdırma

Veriler yazdırılırken, bir sonraki sekme konumunu yazdırmak için '\t' çıktı (escape) dizgesi veya yeni satıra yazdırmak için '\n' çıktı dizgesi kullanılarak bir veya daha fazla boşlukla ayrılır. Ayrıca, endl işleticisi de yeni satıra yazdırmak için kullanılır.

Yazdırma sırasında verileri biçimlendirmek için başka manipülatörler de kullanılır. Bazıları <iostream> kütüphanesinde, bazıları da programın başında yer alması gereken (<iostream> gibi) <iomanip> kütüphanesinde tanımlanmıştır:

```
#include <iostream>
#include <iomanip>.
```

Tablo 1.8.1'de, etkisi cout yazdırma komutunda olan yaygın olarak kullanılan birkaç manipülatör verilmiştir.

Manipülatör	Etkisi	Açıklama
endl	hemen	Yeni satır ('\n ' gibi aynı).
setw(n)	Sıradaki veriye	Çıktı komutundan sonraki verileri yazdırmak için bir alanın minimum genişliğini ayarlama. (Çok basamaklı sayıysa, birden çok sütunda yazdırılır). Verilerin sağa hizalı olduğu (varsayılan olarak) (ing.by default) alınır.
width(n)	Sıradaki veriye	setw(n) gibi aynı
left	Sıradaki veriye	setw(n) ile ayarlanmış alanda sola hizalı.

right	Sıradaki veriye	Setw(n) ile ayarlanmış alanda sağa hizalı (varsayılan olarak)
dec	Sıradaki veriye	Sayı ondalık biçimde (10 tabanlı) yazdırılır.
oct	Sıradaki veriye	Sayı sekizli biçimde (8 tabanlı) yazdırılır.
hex	Sıradaki veriye	Sayı on altılık biçimde (16 tabanlı) yazdırılır.
setprecision(n)	Tüm sonraki verilere	Ondalık noktadan sonra yazdırılacak basamak sayısını ayarlama.
fixed	Tüm sonraki verilere	Ondalık sayıları f biçiminde yazdırma. Varsayılan kesinlik 6 basamaktır.
scientific	Tüm sonraki verilere	Ondalık sayıları e biçiminde yazdırma.
uppercase	Tüm sonraki verilere	Ondalık sayıları e-formatında, ancak büyük E ile yazdırmak için. Örneğin, 1.234E+05. Bu etki, nouppercase manipülatörü ile iptal edilir.
boolalpha	Tüm sonraki verilere	true veya false sabitleri yazdırılır.
noboolalpha		1 veya 0 sabitleri yazdırılır.

Tablo 1.8.1

Verileri Okurken Ayrıntılar

Programlarda verilerin girdisi ve çıktısı karakter akışları üzerinden yapıldığını söyledik. Bu arada, programın başında, iki tür değişken: **ostream** ve **istream** tanımını içeren <iostream> kütüphanesinin dahil edilmesi gerekir.

ostream türü **çıktı akışı** (İng. output stream) ile ilişkilendirilirken , istream türü ise veri **girdi akışı** (İng. input stream) ile ilişkilendirilir.

<iostream> kütüphesinde aşağıdaki türden iki değişken bildirilmiştir:

```
istream cin;
ostream cout;
```

cout değişkeni, **ekleme** (araya koyma) **operatörü** (İng. insertion operator) << ile birlikte bilgisayarın standart çıkış cihazına, genellikle monitöre bağlıdır. Bu arada, tek bir yazdırma komutu ile yazdırılması gereken değişken ve sabitlerin değerleri çıktı akışına yerleştirilir.

Örneğin, aşağıdaki program bölümüyle:

```
int m;
double b;
m = 12;
```

```
b = 34.5;
cout << m << b << endl;
```

çıktı akışına m ve b değişkenlerinin değerleri eklenir ve monitörde şunlar yazdırılacaktır:

```
1234.5
Press any key to continue . . .
```

Benzer olarak, cin değişkeni >> **ayırma** (çıkarma) (İng. extraction operator) **operatörüyle** birlikte bilgisayarın standart giriş cihazıyla, genellikle klavyeyle ilişkilendirilir. Bu arada, girdi akışından veriler ayrılır ve programdaki uygun değişkenlere atanır.

Örneğin, önceki program bölümünde m ve b'yi okuma komutu eklersek:

```
cin >> m >> b;
cout << m << b;
```

ve klavye aracılığıyla, aşağıdaki girdi akışını girirsek:

```
6 7.8
```

Enter tuşuna basarak, m ve b değişkenlerine 6 ve 7.8 değerleri atanacaktır. Çıktı akışı 67.8 olacaktır:

```
1234.5
6 7.8
67.8
Press any key to continue . . .
```

Not Çoğu zaman, operatörün << veya >> olması gerektiğini karıştırırız. Operatörün << çıktıya (yazdırma için) veya >> girdiye (okuma için) yönlendirilmesi gerektiği unutulmamalıdır.

Girdi akışında 6 ve 7.8 verilerinin doğru bir şekilde boşlukla ayrıldığını fark edebilirsiniz. Boşluk olmasaydı, m'ye 67 değeri atanırdı, b'ye ise 0,8 değeri atanırdı.

Aynı girdi akışı

```
6 7.8
```

iki ayrı komutla da doğru okunacaktır:

```
cin >> m;
cin >> b;
```

Girdi akışının sonunda Enter tuşuna basarak, satır sonu karakteri oluşturur ve yeni satıra geçilir. Bu yüzden, **yeni satır karakteri** (İng. newline character) olarak adlandırılır. Girdi akışını okurken, yeni satır karakteri bulunana kadar okunur.

Benzer şekilde, çıktı akışı, (önceki örneklerde gördüğümüz gibi) endl manipülatörü veya '\n' çıkış dizgesi tarafından ayarlanan bitiş karakterine ulaşılana kadar yazdırılır. Ulaşıldığında, ekran işleçi yeni satıra geçiyor.

Boşluğun da karakter olarak da okunması için özel komut kullanılır:

```
cin.get(karakterDegiskeni);
```

Bu komut, cin değişkeninin bildirildiğini söylediğimiz istream türü ile ilişkilidir. Özel bir değişken olduğundan dolayı cin, <iostream> kütüphanesinde tanımlanan çeşitli fonksiyonlar onunla ilişkilendirilir. Bu fonksiyonlardan biri, nokta operatörü (İng.dot operator) ile cin değişkenine bağlanan get() fonksiyonudur. Bu yüzden, okuma komutu cin değişkeni, nokta ve get() fonksiyonundan oluşur.

Örneğin, önceki örnekten

```
6 7.8
```

girdi akışının okunması ve aralarındaki boşluk için, aşağıdaki program bölümünü yazmamız gerekiyor:

```
cin >> m;
cin.get(z);
cin >> b;
cout << m << ' ' << z << ' ' << b << endl;
```

Bu arada, z değişkeni boşluk değeri kazanacak ve çıktı şu olacak:

```
6 7.8
6   7.8

Press any key to continue . . .
```

Boşluklar içeren dizeleri okurken, boşluklarla ilgili daha büyük bir sorun meydana geliyor. ***Şekil 1.8.5***'teki örnekte, son kısımda:

```
cout << "Ad ve soyad girin: "; cin >> benimAdimveSoyadim;
cout << "Girdiginiz harf " << birHarf << endl;
cout << "benimAdimVeSoyadım " << adinda veya soyadında var mıdır << endl;
```

Bütün ad ve soyad girildiğine rağmen sadece ad yazdırılır:

```
Ad ve soyad girin: Gjorgji Jovancevski
Girdiğiniz harf: J
Gjorgi adinda veya soyadinda var midir?
Press any key to continue . . .
```

Bunun nedeni, >> operatörünün ilk boşluğa kadar okuması ve ad ve soyad girilmesine rağmen sadece ad ve soyad arasındaki boşluğa kadar okumasıdır. Bu nedenle, benimAdimVeSoyadim değişkenini "Gjorgji" alır ve bu değer yazdırılır.

Dizeleri doğru okumak için, yeni satır işaretine (Enter ile yerleştirilmiş) kadar tüm giriş satırını okuyan getline() fonksiyonu kullanılır. Bu arada, verileri okumak için okuma işaretçisi bir sonraki satırın başına yerleştirilir.

Komutun sözdizimi şöyledir:

```
getline(cin, stringDegisken);
```

Örneğin, ***Şekil 1.8.5***'teki programda şu komutu:

```
cin benimAdimVeSoyadim;
```

aşağıdaki komutlarla değiştireceğiz:

```
char sonİsareti;
cin.get(sonİsareti);        // '\n' işareti okunur
getline(cin, benimAdimVeSoyadim);
```

ve çıktı doğru olacaktır:

```
Bir harf girin: J
Bir Ad girin: Gjorgji
Bir Soyad girin: Jovancevski
Girdiginiz harf: J
Gjorgji adinda
veya Jovancevski soyadında var midir?

Ad ve soyad girin: Gjorgji Jovancevski
Girdiginiz harf: J
Gjorgji Jovancevski adinda veya soyadinda var midir?

Press any key to continue . . .
```

Açıklayalım. Aşağıdaki komut ile:

```
getline(cin, benimAdimVeSoyadim);
```

girilen dize okunuyor ve benimAdimVeSoyadim dize değişkenine atanır. Ancak, önceki okuma komutunda, bitiş işaretine kadar okuyan ancak son karaktere kadar okumayan >> operatörünü kullanır. Bu yüzden, şu komutla:

```
cin.get(sonİsareti);
```

bitiş işareti okunur ve okuma işaretçisi yeni okuma satırının başına yerleşir.

Not: Tüm program boyunca verileri okumak için getline() fonksiyonu kullanılıyorsa, bitiş karakterinin ayrıdan okunmasına gerek yoktur.

Örnek 1.8.6

Bu örnek ile, get() ve getline() fonksiyonlarının kullanımı gösterilmektedir. Program iki öğrencinin not ortalamasını hesaplar.

```
1   #include <iostream>
2   #include <string>
3   #include <iomanip>
4   using namespace std;
5
6   int main() {     // get( ) ve getline( ) fonksiyonlarin gosterimi
7
8       char enter;
9       string AdveSoyad_1      ;
10      int o1_1, o2_1, o3_1, o4_1, o5_1, o6_1, o7_1, o8_1;
11      cout <<"Birinci ogrencinin adini ve soyadini girin: "   ;
12      getline(AdveSoyad_1 );
13      cout << " Bosluklarla ayrilmis 8 not girin. "                  << endl;
14      cout <<"Notlar: " ;
15      cin >> o1_1 >> o2_1 >> o3_1 >> o4_1 >> o5_1 >> o6_1 >> o7_1 >> o8_1;
16      double ortalama _1 = float( o1_1 + o2_1 + o3_1 + o4_1 + o5_1 + o6_1 +
17          o7_1 + o8_1 ) / 8;
18      cin.get( enter );
19
20      string AdveSoyad_2 ;
21      int o1_2, o2_2, o3_2, o4_2, o5_2, o6_2, o7_2, o8_2;
```

Şekil 1.8.6

```
22      cout <<"İkinci ogrencinin adini ve soyadini girin: "    ;
23      getline( cin, AdveSoyad_2 );
24      cout << "Bosluklarla ayrilmis 8 not girin.."                 << endl;
25      cout << "Notlar: ";
26      cin >> o1_2 >> o2_2 >> o3_2 >> o4_2 >> o5_2 >> o6_2 >> o7_2 >> o8_2;
27      double ortalama _2 = float( o1_2 + o2_2 + o3_2 + o4_2 + o5_2 + o6_2 +
28          o7_2 + o8_2 ) / 8;
29
30      system( "cls" );
31      cout << left << setw( 20 ) <<"Ad ve Soyad" << right << setw( 25 )
32          << "  M  M  I  A  F  H  B  F" << "\tOrtalama " << endl;
33      cout << setw( 45 ) << right << "  a  a  n  n  i  e  i  i" << endl;
34      cout << setw( 45 ) << right << "  k  t  f  g  z  m  o  s" << endl;
35      cout << "-----------------------------------------------" << endl;
36      cout << left << setw( 21 ) <<AdveSoyad_1 << "  " << o1_1 << "  "
37          << o2_1 << "  " << o3_1 << "  " << o4_1 << "  " << o5_1 << "  "
38          << o6_1 << "  " << o7_1
39          << "  " << o8_1 << "\t" << setprecision( 2 ) << Ortalama_1  << endl;
40      cout << left << setw( 21 ) << AdveSoyad_2    << "  " << o1_2 << "  " << o2_2
41          << "  " << o3_2 << "  " << o4_2 << "  " << o5_2 << "  " << o6_2 << "  "
42          << o7_2 << "  " << o8_2 << "\t" << setprecision( 2 ) << AdveSoyad_2 < endl;
43
44      cout << endl;
45      system( "Color 17" );
46      system( "pause" );
47      return 0;
48
49  }
```

Şekil 1.8.6 (devam)

Programın çıktısı şudur:

```
Programın çıktısı şudur:    M  M  I  A  F  H  B  F  Ortalama
                            a  a  n  n  i  e  i  i
                            k  t  f  g  z  m  o  s
-----------------------------------------------------------
Goce Delcev                 5  5  4  5  4  4  5  5    4.6
Jane Sandanski              5  4  5  5  4  4  5  4    4.5

Press any key to continue . . .
```

Alıştırma Ödevleri

1. Aşağıdaki tabloyu yazdıran program yazılsın:

```
Baslik                  Yonetmen            Tur          Yil
-------------------------------------------------------------
Bal-kan-kan             Darko Mitrevski     Dram         2005
Yagmurdan Once          Milco Mancevski     Dram         1994
En Uzun Yol             Branko Gapo         Tarihi       1976
Corek Otu               Kiril Cenevski      Dram         1971
Kurtlar Gecesi          Franc Stiglic       Savaş        1955
```

Film adları için 5 dize değişkeni, yönetmenler için 5 ve tür için 3 dize değişkeni ile yıllar için 5 tamsayı değişkeni bildirilsin. Veriler klavye aracılığıyla aşağıdaki biçimde girilsin:

Bilgiyi Kontrol Etme Soruları

1. C++'da verilerin okunması ve yazdırılması nasıl düzenlenmiştir?
2. Verileri okuyabilmek ve yazdırabilmek için her programda hangi kütüphanenin bulunması gerekir?
3. Verileri okuma ve yazdırma operatörleri hangileridir?
4. Girdi (klavye yoluyla veri girme) ve çıktı (verileri ekrana yazdırma) için komutlar hangileridir?
5. Taşma (İng.overflow) nedir?
6. Hangi C++ kütüphanesi biçimlendirilmiş yazdırma fonksiyonlarını içerir?
7. Bir verinin yazdırıldığı alanın genişliği hangi manipülatör ile ayarlanır?
8. Yazdırma alanındaki verilerin hızalamasını (sola veya sağa) ayarlamak için hangi manipülatör kullanılır?
9. Yazdırılacak verinin kesinliğini (ondalık basamak sayısı) ayarlamak için hangi manipülatör kullanılır?
10. true ve false mantıksal sabitleri hangi manipülatör ile yazdırılrr?
11. Yazdırma sırasında ekleme (araya koyma) operatörü hangisidir, okurken ise ayırma (çıkarma) operatörü hangisidir?
12. Bir karakter okuma komutunu yazın.
13. Satırın tamamını okuma komutunu yazın.
14. Hangi komutla sadece bir kelime, hangi komutla tüm cümle okunur?

Terimler

- Bilgisayar tarafından çalıştırılabilen tüm programlara tek adla yazılım denir.
- Yazılım, sistem programları ve uygulama programları olarak ikiye ayrılır.
- Sistem programları, bilgisayarın çalışmasını yöneten programlardır.
- Uygulama programları veya kullanıcı programları, kullanıcılar için belirli görevleri yerine getiren programlardır.
- Programlama dili, insanlarla bilgisayarlar veya iki bilgisayar arasındaki iletişim için kullanılan yapay dildir.
- Algoritma, sonlu sayıda kesin olarak tanımlanmış faaliyetlerden oluşan ve bunların kesin olarak verilmiş sırayla yürütme prosedürüdür.
- Algoritma, kesin olarak belirlenmiş bir sırayla girdi verilerine uygulanan ve bunlarla çıktı sonuçlarına ulaşılan, kesin olarak tanımlanmış sonlu sayıda faaliyetler kümesinden oluşan prosedürdür.
- Algoritmik adım, algoritmanın bir eylemidir.
- Genel algoritma daha genel adımlardan oluşur.
- Ayrıntılı algoritma, tek bir komutla yazılabilen daha ayrıntılı adımlardan oluşur.
- Sözde dil, algoritmaların metinsel yazılması için kullanılan dildir.
- Algoritmaların grafik gösterimi blok diyagramlar ile yapılır.
- Kodlama algoritmaların bir programlama dilinden komutlarla "ifade edilmesi" yani yazılmasıdır.
- Kaynak program, bir programlama dilinden komutlarla kodlanmış algoritmadır.
- Yürütülebilir program, kaynak programdan çeviri yoluyla elde edilen ve işlemci tarafından yürütülebilen programdır.
- Çevirici, kaynak programını yürütülebilir programa çeviren sistem programıdır.
- Yorumlayıcı, kaynak programını yorumlayan ve yürüten sistem programıdır.
- Algoritmik kontrol yapıları, algoritmaların yürütülmesini kontrol eden özel yapılardır.
- Yapılandırılmış programlama, şu teknikleri kullanan programlamadır: yukarıdan aşağıya programlama ve modüler programlama.
- Yukarıdan aşağıya programlama tekniği ile bir ödev, alt ödevler adı verilen daha küçük ve daha basit ödevlere bölünür.
- Modüler programlama tekniği, algoritmanın bağımsız birimlerinin ayrı modüller olarak programlanmasından oluşur ve bunlar daha sonra sırayla tek bir bütüne birleştirilir.

- Makine dili, makine yönergelerinin ve verilerinin sıfırlar ve birlerle ifade edildiği dildir.
- Sembolik diller, makine komutlarının ve verilerinin sembollerle (anımsatıcılarla) ifade edildiği dillerdir.
- Birleştiriciler, sembolik dil programını makine dili programına çeviren çeviricilerdir.
- Yüksek seviyeli programlama dilleri, günümüzde (çoğunlukla) programlamanın yapıldığı, doğal dillere yakın dillerdir.
- Yüksek seviyeli programlama dilleri makineden bağımsız dillerdir çünkü programların yürütüleceği (önceden çeviri ile) makineye bağlı değildirler.
- Yüksek seviyeli programlama problem yönelimli dillerdir çünkü özel alanlardan (ekonomi, teknoloji, yapay zeka vb.) problemlere yöneliktirler.
- Programlama dilinin dil bilgisi, kelimeler ve komutlar oluşturmak için bir dizi kurallardır.
- Sözdizimi, programlama dillerinde komutlar oluşturmak için bir dizi kurallardır.
- Semantik, bir komutun anlamıdır, yani yürütme sırasında neden olan eylemlerdir.
- Ayrılmış kelimeler, sadece programlama dili için ayrılmış özel kelimelerdir.
- Ayrılmış kelimelerin bazıları, programlama dili için özel bir anlamı olduğu için anahtar kelimelerdir.
- Tanımlayıcılar, programcılar tarafından belirli kurallara göre oluşturulmuş kelimelerdir ve programlama dilinde ayrılmış kelimelerden farklı olmalıdır.
- Çevirici (derleyici), kaynak programı makine programına çevirmek için özel programdır.
- Yorumlayıcılar, kaynak programı doğrudan yürüten (yorumlayan) özel programlardır.
- Komut dosyası dilleri, diğer programlama dilleri ile entegrasyon ve iletişim için programlama dilleridir.
- Komutsal programlama dilleri, veri işleme için komutların kullanıldığı dillerdir.
- Bildirimsel programlama dilleri sonuca nasıl ulaşılacağını anlatan dillerdir.
- Prosedürel programlama dilleri, prosedürlerin (fonksiyonların) (bir bütün olarak ifade edilen komutlar grubu) kullanıldığı komutsal dillerdir.
- Nesne yönelimli diller verilerin, nesnelerin nitelikleri olarak ifade edildiği ve nesnelerin davranışını fonksiyonlarla ifade edildiği komutsal dillerdir.

- Fonksiyonel diller, ifadeleri ve fonksiyonları kullanan bildirim dilleridir.
- Mantıksal diller, hesaplamalar sırasında ilişkilerin, gerçeklerin, sonuçların ve çıkarım yöntemlerinin kullanıldığı bildirimsel dillerdir.
- Unified Modeling Languge – UML, farklı süreçleri (algoritma akışı, bir durumdan diğerine geçiş vb.) ifade etmek için özel grafik dilidir.
- Tümleşik geliştirme ortamı, programlamada kullanılan uygulamaların bulunduğu ortamdır.
- Microsoft Visual Studio, programları yazmak, düzenlemek ve yürütmek (test etmek) için tümleşik geliştirme ortamıdır.
- Code::Blocks, programları yazmak, düzenlemek ve yürütmek (test etmek) için tümleşik geliştirme ortamıdır.
- Metin düzenleyici – editör, program yazmak ve düzenlemek için kullanılan uygulamadır.
- Hata ayıklayıcı, programda mantıksal hataları aramak için kullanılan uygulamadır.
- Bağlayıcı, birden çok dosyayı (programı) tek bir dosyaya bağlamak için kullanılan uygulamadır.
- C++'da uygulama, main() ana işlevini içeren programdır.
- <iostream>, programlarda girdi ve çıktı işlemleri için fonksiyonlar içeren kütüphanedir.
- namespace, değişkenlerin, fonksiyonların ve diğer tanımlayıcıların adları için bellekte ad alanı (aralığı) oluşturmak için kullanılan yönergedir.
- Fonksiyon, yürütüldükten sonra sonuçlar veren programdır.
- Girinti, programların daha iyi görünür olması için komutların girintisini
- (çoğunlukla bir sekme konumu için) tanımlamaktadır.
- Ana fonksiyon, her uygulamanın yürütülmesini başlatan fonksiyondur.
- Yorum, satırda iki eğik çizgiden sonra veya başta /* ve sonda */ karakterleri arasına yerleştirilen herhangi bir metindir.
- cout yazdırma komutudur.
- cin okuma komutudur.
- << yazdırma operatörüdür.
- >> okuma operatörüdür.
- endl yeni satıra geçme komutudur.
- ; komut sonu karakteridir, yani komut ayırıcısıdır..
- return 0; uygulamanın doğru tamamlanması için komuttur.
- Sıralı yapı veya dizge, komutların sıralandıkları sırayla yürütüldüğü yapıdır.

- Çıktı (kaçış) dizgeleri, ters eğik çizgi ve bazı harf veya karakter kombinasyonlarıyla ifade edilen özel dizgelerdir. Örneğin, \n yeni satıra geçiş kaçış dizgesidir.
- Dize, özel bir veri türü olarak ele alınan, tırnak işaretleri içine alınmış karakterler dizisidir.
- İfade (genellikle), değeri olan matematiksel ifadedir. Çoğu zaman, programlama dillerindeki komutlara ifade denir.
- Büyüklük, belirli bir nesnenin veya olgunun bazı özelliklerinin ölçüsüdür.
- Veriler, büyüklüklerin ifade edildiği değerlerdir.
- Değişmezler, değişmeyen sabit değerlerdir.
- Adlandırılmış sabitler veya sadece sabitler, sadece bir kez değer atanabilen tanımlayıcılardır (büyüklüklerin adları) ve programın yürütülmesi sırasında değişmezler.
- Değişkenler, programda farklı değerler atanabilen tanımlayıcılardır
- (büyüklüklerin adları).
- Veri türü, üzerinde sonlu işlemler kümesinin tanımlandığı sonlu veri kümesini temsil eder.
- Basit veya yapılandırılmamış veri türleri, parçalara ayrılamayan veri türleridir.
- Karmaşık veya yapılandırılmış veri türleri birden çok basit türden oluşur.
- Adres veri türleri, değeri bazı bellek konumunun adresi olan veri türleridir.
- Temel veri türleri olarak da adlandırılan basit veri türleri veya yerleşik veri türleri, tamsayı, gerçek, karakter ve mantıksal türlerdir.
- Tür belirteçleri, tür adlarıdır.
- unsigned veriler, sadece negatif olmayan değerler alabilen verilerdir.
- Dize, tırnak işaretleri içine alınmış karakterler dizisidir.
- Birleştirme, iki dizeyi birbirine bağlamak için yapılan işlemdir.
- Değişkenin bildirimi, değişkenin türünü ve adını belirtmektir.
- Değişkenin bildirimi ve başlatılması, değişkenin türünü ve adını belirtmek ve değişkene başlangıç değeri atamaktır.
- Değişken tanımlama, bildirim ve başlatma sırasında değer atamak veya önceden bildirilmiş bir değişkene atama komutu ile değer atamaktır.
- Tamsayı veri türü, Z = {... –2, –1, 0, 1, 2... } tamsayılar kümesinden bir değere sahip verilerdir ve alt kümeye bağlı olarak, tamsayı veri türleri şu kelimelerle bildirilir: short, int, long, long long, unsigned short, unsigned int, unsigned long ve unsigned long long.

- Operatörlerin (veya karmaşık operatörlerin) kısaltılmış biçimi, aritmetik operatör ve atama operatörü = ile ifade edilir. Örneğin, +=, –=, *= vb.
- Gerçek veri türü, gerçek sayılar kümesinden (yani ondalık sayılar kümesinden) değerler alan ve float, double ve long double sözcüklerle bildirilen verilerdir.
- Ondalık sayının sabit nokta ile temsil edilmesi, ondalık sayının sabit bir yerde ondalık nokta ile yazılmasıdır; bu, tam sayıyı ve sayının ondalık kısmını tam olarak böler.
- Ondalık sayının kayan nokta ile temsil edilmesi, ondalık sayının, kaydırılan yerler için üs kullanılarak istenilen yerde ondalık nokta ile yazılmasıdır.
- Örtük (otomatik) dönüştürme veya tür zorlaması, uygun olmayan türdeki bir değişkene değer atanmasıdır.
- Açık tür dönüştürme veya tür dökümü, değişkenin veya ifadenin dönüştürülmesini istediğimiz türün belirtilmesidir.
- Arttırma operatörü ++'dır.
- Azaltma operatörü – –'dir.
- Karakter veri türü, herhangi bir karaktere veya herhangi bir tamsayı değerine sahip olabilir ve char sözcüğü ile bildirilir.
- Mantıksal veri türü, true veya false değerine sahip olabilir ve bool sözcüğüyle bildirilir.
- Mantıksal operatörler veya koşullu operatörler şunlardır: &&, || ve !.
- Boole fonksiyonlar şunlardır: NOT, AND ve OR.
- Mantıksal ifade, değeri true veya false olabilen ifadedir.
- İlişkisel operatörler şunlardır: <, <=, >, >=, ==, !=.
- Bitsel operatörler şunlardır: &, |, ^, ~, << ve >>.
- Sıralı veri türü, sayılar değil, tanımlayıcılar olması gereken önceden tanımlanmış sabitler kümesinden (değişmez değerler) değere sahip olabilir. enum sözcüğüyle bildirilir.
- Metinsel veri, tırnaklar işaretleri içine alınmış karakterler dizisidir.
- Veri türünün yeniden adlandırılması, tür için başka bir ad (eş anlam) vermektir.
- Otomatik tür belirleme, değişkene atanan ifadenin türüne göre çevirici tarafından değişkenin türünün belirlendiği auto kelimesi ile yapılır.
- Başlık dosyaları, C++ standart kütüphanesinden #include yönergesiyle dahil edilen ve açılı parantezler <> içine alınmış dosyalardır.
- Taşma değişkenin değeri, bildirildiği türün kapsamı dışında olduğunda meydana gelir.

- Ekleme (araya koyma) operatörü <<, bilgisayarın standart çıktı cihazıyla, genellikle monitörle ilişkilendirilir.
- Ayırma (çıkarma) operatörü >>, bilgisayarın standart giriş cihazıyla, genellikle klavyeyle ilişkilendirilir.
- Yeni satır işareti, Enter tuşuna basılarak oluşturulan işarettir.
- Çıktı veri akışı, yazdırılan verilerden karakterler dizisini tanımlamaktadır.
- Girdi veri akışı, okunan verilerden karakterler dizisini tanımlamaktadır.
- <string>, dizelerle çalışmak için fonksiyonlar içeren kütüphanedir.
- <iomanip>, biçimlendirilmiş yazdırma için manipülatörler içeren kütüphanedir.
- <ostream>, çıktı akışı komutlarını içeren kütüphanedir.
- <istream>, girdi akışı komutlarını içeren kütüphanedir.

Özet

- Bilgisayar tarafından çalıştırılabilen tüm programlara bir adla yazılım denir.
- Bilgisayarın çalışması sistem programları tarafından kontrol edilir.
- Herhangi bir programı bilgisayar yardımıyla çalıştırmak için, bilgisayarın çalıştırabileceği bir uygulama programı olmalıdır.
- Herhangi faaliyeti gerçekleştirebilmek için, gerçekleştirilecek (temel) faaliyetlerin tam sayısı ve bunların uygulanma sırası bilinmelidir. Yürütme sırasına göre yazılan temel faaliyetlerin (adımların) listesi algoritma tanımlamaktadır.
- Algoritmanın yürütülmesi sırasında girdi verileri girilir ve çıktı sonuçları elde edilir. Bazen çıktı sonuçları doğrudan değil, ara sonuçlar aracılığıyla elde edilir.
- Algoritmik adımlara bağlı olarak, algoritma genel veya ayrıntılı olabilir.
- Algoritmalar sözde dille metinsel olarak, grafiksel olarak blok diyagramla veya bir programlama dilinin komutlarıyla yazılabilir. Grafik gösterim, standart veya yapılandırılmış blok diyagramlar olabilir.
- Algoritmanın programlama dilinden gelen komutlarla yazılması işlemine kodlama denir, ortaya çıkan metne ise kaynak programı adı verilir.
- Bir programlama dilinde yazılmış kaynak program, çevirici tarafından, işlemcinin çalıştırabileceği yürütülebilir programa çevrilebilir veya yorumlayıcı tarafından eşzamanlı olarak çevrilebilir (yorumlanabilir) ve yürütülebilir.
- Algoritmaların uygulama seyrinin (izlenebilmesi) daha kolay kontrol edilebilmesi için algoritmik kontrol yapılarıyla yazılmıştır.Onlara göre, bu tür programlamaya yapısal programlama adı verildi.
- Yapılandırılmış programlama, şu yöntemleri kullanan bir programlama şeklidir: yukarıdan aşağıya programlama (bir görevi daha basit görevlere ayırma) ve modüler programlama (bağımsız birimler için ayrı programlar oluşturma).
- Bilgisayar yardımıyla bir ödevi çözmek birkaç aşamadan oluşur: ödevi belirleme, algoritma tanımlama, program yazma (programlama) ve programı test etme.
- Ödev kurulurken, hangi koşullar altında çözüleceği kesin olarak tanımlanmalı ve belirtilmelidir.
- Algoritma tanımlarken öncelikle görevi anlamak, işlenecek girdi verilerini analiz etmek, formüllere ve/veya yöntemlere ihtiyaç olup olmadığı ve hangi sonuçların elde edilmesi gerektiği analiz edilmelidir.

Ardından programlanabilen algoritma yazılır ve program bilgisayarda çalıştırılır.

- Programı geliştirdikten sonra, sonuçları bilinen farklı girdiler için doğru çalışıp çalışmadığı (doğru sonuçları alıp almadığı) test edilmelidir.
- Programlama dilleri, şunlara ayrılan yapay dillerdir: üst seviyeli programlama dilleri (programların yazıldığı), sembolik diller (programın çevrilebileceği ancak zorunlu olarak çevrilemeyeceği) ve makine dili (bilgisayarın çalıştırabilmesi için programın çevrildiği dil).
- Makine dilinde hem komutlar hem de veriler bit olarak, yani 0 ve 1 rakamları ile yazılır.
- Sembolik dillerde hem komutlar hem veriler için semboller
- kullanılır.Sembolik diller makine yönelimlidir çünkü her işlemci ailesinin kendi sembolik dili vardır.
- Her üst seviyeli programlama dili şunları içerir: dilbilgisi, ayrılmış (ve anahtar) kelimeler sözlüğü, tanımlayıcılar (sabitler ve değişkenler) oluşturmak için kurallar, komutlar oluşturmak için sözdizimsel kurallar (sözdizimi) ve komutların anlamını kontrol etmek için semantik kurallar.
- Program yazarken (programlama), programcılar işledikleri verileri adlandıracakları özel kurallara sahip sözcükler (tanımlayıcılar) oluşturabilir. Tanımlayıcılar ayrılmış kelimeler olmamalıdır.
- Kaynak program çevirici (derleyici) ile çevrilirse, programın yürütülebilir dosyası makine biçiminde yapılır.Kaynak program yorumlayıcı tarafından yorumlanırsa, yürütülebilir bir dosya yapılmaz, ancak yorumlayıcı komutları yorumlar ve yürütür.
- Komut dosyası dilleri, diğer programlama dilleri ile iletişim kurmak için kullanılır. Bunun çeviriciye değil, yorumlayıcıya ihtiyaçları vardır.
- Kaynak programların adından sonra yazıldıkları dilin kısaltması olan sonekleri vardır. Örneğin: .cpp, .java, .pas, .for vb. Yürütülebilir
- programların .exe, .com, .bat, .bin, .dll vb. uzantıları vardır.
- Birden fazla nesil programlama dili vardır.
- Veri işleme yöntemine göre, programlama dilleri ikiye ayrılır: zorunlu
- (prosedürel ve nesne yönelimli) ve bildirimsel (fonksiyonel ve mantıksal).
- Nesne yönelimli programlamada, kendi nitelikleri (verileri) ve davranışı veya durumu (fonksiyonlarla ifade edilen) olan nesneler kullanılır.
- Algoritmaların ve algoritmadaki süreçlerin grafik gösterimi için Unified Modeling Languge – UML kullanılır.
- Günümüzde C++ programlama için farklı tümleşik geliştirme ortamları bulunmaktadır ve bunlardan en çok bilinen şunlardır: Microsoft Visual Studio, Code::Blocks, NetBeans ve diğerleri.

- Her tümleşik geliştirme ortamında şunlar bulunur: metin düzenleyici, çevirici, bağlayıcı ve hata ayıklayıcı.
- C++ dili, Bjarne Stroustrup tarafından 1983 yılında, 1980 yılından "Sınıflı C" diline yükseltme olarak tasarlanmış, bu da Dennis Ritchie tarafından tasarlanan 1972 C diline yükseltme olan nesne yönelimli programlama dilidir.•
- C++ 'da yazılmış program (uygulama), yürütmenin başladığı ana fonksiyona (main()) ve ana fonksiyondan çağrılan diğer fonksiyonlara sahip olmalıdır.
- C++ ilk nesne yönelimli programlama dilleri arasındadır.
- Girdi-çıktı işlemlerinin gerçeklştirilmesi için programın başında <iostream> kütüphanesinin dahil olması gerekmektedir.
- Programın daha fazla okunabilir olması için, çevirisi yapılmayan boş satırlar ve yorumlar yerleştirilir ve ayrıca programın girintilenmesi yapılır
- C++'da her komut noktalı virgülle (;) biter.
- C++'da programlar herhangi bir metin düzenleyicide yazılabilir.
- C++'da bir program fonksiyonlardan oluşur. Ana fonksiyon, main () olarak adlandırılır.
- Her fonksiyonun gövdesi büyük parantezler içinde koyulur.
- Her uygulamanın yürütülmesini başlatan sadece bir ana fonksiyonu olmalıdır.
- C++'da yazdırma komutu, << yazdırma operatörüyle birlikte kullanılan cout'tur.
- C++'da okuma komutu, okuma operatörü >> ile birlikte kullanılan cin'dir.
- C++'daki ana fonksiyon, program doğru yürütülürse 0 sonucunu döndürür.
- Sonunda endl manipülatörü bulunan yazdırma komutundan sonraki komut yeni satırda (yeni sırada) yazdırılır.
- "...", '... ', \ gibi özel karakterleri yazdırmak için çıktı (kaçış) dizgeleri kullanılır.
- C++ dili, bir programlama dilinin dilbilgisi, sözdizimi, semantiği, ayrılmış kelimeleri, tanımlayıcıları (programcılar tarafından oluşturulan sözcükler) ve diğer öğelerini içerir.
- Programı çevirirken söz dizimi kuralları ile her komutun doğru yazılışı kontrol edilir.
- Programdaki her komutun tam olarak tanımlanmış anlamı olmalıdır ve tam olarak tanımlanmış eylemlere neden olmalıdır. Bu, dilin semantik kuralları tarafından Verilerle büyüklükleri değerleri ifade edilir, büyüklükler ise nesnelerin veya olayların özellik ölçüleridir.Veriler sabit (değişmez değere sahip) veya değişken (değeri değiştirilebilir) olabilir.

- C++'da sabitleri bildirirken, türünden önce const kelimesi koyulur.
- Her verinin adı, türü ve değeri vardır.
- C++'da veri adları (tanımlayıcılar) şunlar olabilir: ayrılmış kelimeler (anahtar kelimeler dahil) veya belirli kurallara göre oluşturulmuş kullanıcı tanımlı adlar (tanımlayıcılar).• Ayrılmış kelimeler tanımlayıcı olarak kullanılamaz.
- C++'da veri türleri basit (yapılandırılmamış), karmaşık (yapılandırılmış – basit türlerden oluşur) ve adres (değeri bellek konumunun adresidir) olabilir.
- Basit türler şunlardır: integral (tamsayı, karakter ve mantıksal), gerçek ve sıralı.
- Karmaşık türler şunlardır: dizi, yapı, birlik ve sınıf.
- Sabitler ve değişkenler türü ve adını belirtilerek bildirilir. Bildirimde sabite veya değişkene bir değer verilirse, bildirim ve başlatma işlemi gerçekleştirilir.
- Sabitleri değişkenlerden ayırmak için, hepsini büyük harflerle yazmak ve sabitlerin adlarındaki sözcükleri alt çizgi ile ayırmak uygulanır.
- Sabitler, const kelimesi ve ardından tür ve ad ile bildirilir ve ardından = atama operatörü ile sabite değer atanmalıdır.
- Bir programda kullanılmadan önce, değişken, başlatma veya atama deyimi ile değer atayarak tanımlanmalıdır.
- Bir değişkene, türünün aralığından daha büyük veya daha küçük değer atandığında, taşma veya eksik doldurma meydana gelir ve hata oluşur.
- Tamsayı türü ve karakter türü işaretsiz olabilir.
- Veri türü belirleyicileri şunlardır:

- Tamsayı türü ve karakter türü işaretsiz olabilir.
- Veri türü belirleyicileri şunlardır:

  - tamsayı türü için: short, int, long ve long long.
  - karakter türü için: char.
  - sıralı türü için: enum.
  - yapı için: struct.
  - sınıf için: class.
  - referans için: referans.
  - gerçek türü için: float, double ve long double.
  - mantıksal türü için: bool.
  - dizi için: array.
  - birlik için: union.
  - işaretçi için: pointer.

- Tamsayı veri türü, {... –2, –1, 0, 1, 2...} tamsayılar kümesinden değerlerdir.
- Aritmetik operatörlerin kısaltılmış biçimleri: +, –, * ve / tamsayı ve gerçek veriler için: +=, –=, *=, /= operatörleridir.

%= operatörü, tamsayı veri türü için % operatörünün kısaltılmış biçimidir. Bu operatörlere karmaşık operatörler denir.

- Gerçek veri türü, gerçek (ondalık) sayılar kümesindeki değerlerdir {... –2.0..., –2.001..., –1.0..., 0.0..., 1.0..., 2.0 ...} .
- float türü değişkenler tanımlanırken f veya F eki (soneki) eklenmelidir. Aksi halde double tip değişkenler gibi ele alınırlar.
- Programlama dillerindeki ondalık veriler, ondalık biçimde (f-format, F-format)(kaymayan (sabit) nokta olarak bilinen yöntem) ve üstel biçimde (e-format, E-format) (kayan nokta olarak bilinen yöntem) gösterilir.
- f harfi "float" kelimesinden, e harfi ise "exponent" kelimesinden gelir.
- f-formatındaki (veya F-formatındaki) ondalık sayılar sabit bir yerde ondalık noktayla yazılır.
- Ondalık sayılar e-format (veya E-format) olarak yazdırıldığında, ondalık noktanın solunda sadece bir basamak yazılır, e veya E harfi ise üssün önünde yazılır.
- Uygun olmayan türde bir değişkene değer atama dönüştürme ile yapılır ve örtük (otomatik) veya açık (tür dökümü) olabilir. İkinci durumda, uygun olmayan türün değerinin dönüştürülmesi gereken tür belirtilir, ilk durumda ise belirtilmez.
- Tamsayının veya işaretli bir değişkenin değerini 1 için artırmak/azaltmak için artırma operatörü (++) veya eksiltme operatörü (– –) kullanılır.
- Karakter türündeki veriler, herhangi bir karakter veya herhangi bir tamsayı değerine sahip olabilir ve yarı tırnak işareti içine koyulur.
- Mantıksal veri türü yalnızca true veya fasle özel sabitlerin değerine sahip olabilir.
- Mantıksal ifadeler hesaplanırken, true ve false mantıksal sabitleri 1 ve 0 değerine sahiptir.
- Mantıksal veri türü işlemleri için mantıksal (koşullu) operatörler kullanılır: &&, || ve !. Bunlar AND (VE), OR (VEYA) ve NOT (DEGİL) Boole fonkisyonların operatörleridir.
- İlişkisel operatörler şunlardır: <, <=, >, >=, == (eşit) ve != (farklı).
- Bitsel operatörler şunlardır: & (mantıksal VE), | (mantıksal VEYA), ^ (özel VEYA), ~ (tamamlayıcı), << (1 basamak sola kaydırma) ve >> (1 basamak sağa kaydırma).
- Sıralı veri türü enum kelimesi ile bildirilir ve sabitler büyük parantezler içinde kouylur ve büyük harflerle yazılır.Büyük parantezlerin çıkış kısmından sonra noktalı virgül (;) koyulmaz.
- Sıralı veri türü main() fonksiyonundan önce tanımlanır;

- Dizeler veya metinsel veriler, tırnak işaretleri içine yazılmış, karakterler dizisinden oluşan verilerdir.
- Dizelerle çalışmak için #include direktifi ile programa dahil edilmesi gereken <string> kütüphanesindeki fonksiyonlar kullanılır.
- Çoğu zaman, programın okunabilirliği ve içinde daha kolay gezmek için, yerleşik veri türlerinin adları için typedef anahtar kelimesiyle bize bir şeyi hatırlatacak eşdeger adların (diğer adlar) kullanılması tavsiye edilir.
- Değişkenin türünün kendisine atanan ifadenin türüne göre belirlenmesini istediğimizde, bu değişken auto sözcüğü ile bildirilir.
- C++'da veri okuma, karakter girdi akışı olarak düzenlenir, yazdırma ise karakter çıktı akışı olarak düzenlenir.
- Standart girdiden okuma komutu cin'dir, standart çıktıya yazdırma komutu ise cout'tur. cin ve cout, <iostream> kütüphanesinden alınan değişkenlerdir. cin, girdi veri akışıyla ilişkili istream türündendir, cout ise çıktı veri akışıyla ilişkili ostream türündedir.
- #include yönergesiyle başlayan program satırları çevirici tarafından çevrilmez, önişlemci adı verilen özel program tarafından işlenir.
- #include yönergesiyle programa dahil edilen dosyalar köşeli parantez içine alınır, ve bununla önişlemciye onları C++ standart kütüphanesindeki include dizininde araması için bilgi verir.
- Verileri biçimlendirmek için yazdırma sırasında manipülatörler kullanılır. Onlardan bazıları <iostream> kütüphanesinde tanımlanmıştır, bazıları ise
- (<iostream> gibi) programın başında yer alması gereken <iomanip> kütüphanesinde tanımlanmıştır.
- Enter tuşuna basarak yeni satır karakteri oluşturulur ve yeni satıra geçer. Girdi verilerini okurken, yeni satır karakteri bulunana kadar okunur.
- C++ standart kütüphanesi, benzer fonksiyon gruplarını içeren, başlık dosyaları adı verilen birden çok dosya içerir. En yaygın olarak kullanılan başlık dosyaları şunlardır: <iostream>, <iomanip>, <cmath>, <string>, <cctype> ve diğerleri.

MODÜL

C++'DA AKIŞ KONTROLÜ VE FONKSİYONLAR

2

Bu bölümde aşağıdakiler hakkında bilgi edineceksiniz:

- Algoritmik kontrol yapıları.
- Sıralı algoritmik kontrol yapısının kullanılması.
- **eğer-o zaman** ikili olasılık seçimi için algoritmik kontrol yapısı ve if-else ikili olasılık seçimi için kontrol komutu.
- if-else ikili olasılık seçim kontrol komutunun iç içe yerleştirilmesi.
- Çoklu **durum** olasılıklardan seçim yapmak için algoritmik kontrol yapısı ve çoklu olasılıklardan seçim yapmak için kontrol komutu switch.
- ?: koşullu operatörü kullanmak.
- Tekrarlama için algoritmik kontrol yapıları; i**ken–yürütür, yürütür–iken , için–kadar-adım** ve tekralama için kontrol komutları;while, do-while , for.
- Arttırma (++) ve eksiltme (– –) operatörlerini kullanma
- Atlama için algoritmik kontrol yapıları (**devam, ara, çıkış, atlama**) ve atlama kontrol komutları (continue, break, exit(), go to).
- C++ programlama dilinde kütüphane fonksiyonları.
- Kullanıcı fonksiyonlar: tanımlama, bildirme ve çağırma.
- Global (genel) ve yerel değişkenleri bildirmek ve kullanmak.
- Yerleşik fonksiyonlar

Anahtar kelimeler

:: görüş alanını çözme operatörü.	Fonksiyon
?: koşullu operatör	Fonksiyon alt algoritma
break	Fonksiyon alt programı
continue	Fonksiyon imzası
do-while	Fonksiyon prototipi
exit()	Gerçek argüman
for	Girdi biçimsel argüman
goto	Girdi parametresi
if	Girdi-çıktı biçimsel argüman
if-else	Girdi-çıktı parametresi
inline	Gizli değişken
switch	Global değişken
void	Görünürlük alanı
while	**için–kadar–adım**
Algortimik kontrol yapısı	**için–arttır–kadar**
Algoritmik blok	**için–azalt–kadar**
Alt algoritma	**iken- yürütür**
Alt ödev	İki olasılıklı seçim için algoritmik kontrol yapısı
ara (kesinti)	Kullanıcı fonksiyonu
Argüman	Kullanıcı tanımlı fonksiyon
Argümanlar listesi	Modüler programlama
atlayış	Parametreler
başlangış-bitiş	Parametreler listesi
Biçimsel argüman	Prosedür
Çıktı	Referans
Çıktı biçimsel argüman	Referans değişkeni
Çıktı parametresi	Referansa göre argüman iletme
Değere göre argüman iletme	Referansa göre iletilen argüman
Değere göre iletilen argüman	

Değişkenin yaşam süresi	Referanslama
devam	Sabit parametre
döndür	Sıralı algoritmik kontrol yapısı (dizge)
Döngü	Statik değişken
Döngülü tekrarlama	Standart kütüphane
Dönüş değeri olan fonksiyon	Türün biçimsel parametresi
Dönüş değeri olmayan fonksiyon	Yerel değişken
durum	Yerleşik fonksiyon
eğer–o zaman	Yukarıdan aşağıya programlama
eğer–o zaman–değilse	**yürütür – iken**
Erişim alanı	

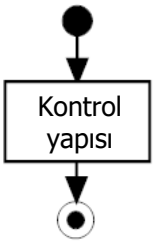
C++ Programlama Diline Giriş Modülünün 1.1, 1.2 ve **1.3** alt bölümlerinde, algoritma, algoritmik adımlar, algoritmaları temsil etme yolları ve yapılandırılmış programlama terimlerini tanıdık. Yapılandırılmış programlamada algoritma akışını tanımlamak için özel **algoritmik kontrol yapıları** kullanılır. Onlarla herhangi bir hesaplama yapılmaz, ancak algoritmik adımların yürütme sırası belirlenir, yani adımların yürütme sırası yönetilir.

Programlama teorisinde, her algoritmanın akışının aşağıdaki algoritmik kontrol yapıları kullanılarak kontrol edilebileceği bilinmektedir:

- Sıralı kontrol yapısı veya dizge (İng. sequence)..
- Seçim veya seçme kontrol yapısı (İng. selection).
- Tekrarlama veya yineleme kontrol yapısı (İng. iteration).

Birinci algoritmik kontrol yapısı aynı zamanda **lineer (doğrusal) kontrol yapısı** adıyla da bilinir, ikincisi **dallanmış kontrol yapısı,** üçüncüsü ise **döngüsel kontrol yapısı** adıyla da bilinir.

Her algoritmik kontrol yapısının sadece bir giriş noktası (●) ve sadece bir çıkış noktası (⊙) vardır, şekil 2.1



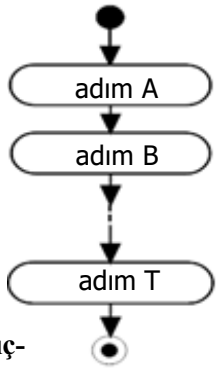
Şekil 2.1

2.1 Sıralı kontrol yapısı

Bu kontrol yapısı, belirtilen sırayla yürütülen algoritmik adımlardan oluşur. Bunu önceki tüm örneklerde kullandık.

Grafiksel olarak, bu yapı ***şekil 2.1.1***'de gösterilmiştir.

Her sıralı kontrol yapısı, ayrı bir bütünü temsil eder ve algoritmanın herhangi bir yerinde bulunabilir. Bu yüzden, yapının başlangıcını ve bitişini işaretlemek gerekir. Bu, **başlangıç** (İng. begin) ve **bitiş** (İng.end) kelimeleri ile yapılır, ***şekil 2.1.2.*** Sıralı algoritmik kontrol yapısı **başlangıç-bitiş** olarak adlandırılır.



Şekil 2.1.1

```
başlangıç
        adım A;
        adım B;
        ...
        adım T;
bitiş
```

Şekil 2.1.2

" ; " (noktalı virgül) işareti, **başlangıç-bitiş** sıralı kontrol yapısındaki adımları ayırmak için kullanılır. Sıralı kontrol yapısının içeriği bir adımlar **bloğudur**, daha iyi görünürlük için ise adımlar, **başlangıç** ve **bitiş** kelimelerinin sağına hafifçe girintili olarak yazılır. **Başlangıç** ve **bitiş** kelimeleri aynı dikey çizgide yazılır. Bu, tüm algoritmik kontrol yapıları için geçerlidir.

Algoritmaların ve programların girintili (çoğunlukla bir sekme pozisyonu için) şekilde yazılması **girintileme** (İng. indentation). olarak adlandırılır. Girintileme, algoritmik kontrol yapılarının ve programlarının daha iyi görünürlüğü ve daha kolay tanınması için kullanılır.

başlangıç-bitiş algoritmik kontrol yapısı, genellikle verilere başlangıç değerler atama adımlarında ve hesaplamalarda kullanılır.

Örneğin:

başlangıç { Başlangıç değerler}
    a ← 3;
    b ← –5.74;
    c ← '@';
bitiş

Bu arada, sola doğru ok ← sembolu değişkenlere değer atama için kullanılır. **Modül 1: C++ Programlama Diline Giriş'**te yazdığımız tüm programlarda komutları arka arkaya yazdığımız için **başlangıç-bitiş** algoritmik kontrol yapısını kullandık.

Bazı durumlarda, bir algoritmik adımın yürütüldüğü komutlar grubunun (örneğin, adım: verileri okuma) işaretlenmesi gerekiyorsa, grup büyük parantezler {} içine alınır. {} parantezleri yürütülmesi gereken komutlar bloğunu belirtmektedir.

Buna göre, **şekil 2.1.2**'deki algoritmik kontrol yapısı şekil **2.1.3'**teki gibi yazılacaktır.

```
{
        komut A;
        komut B;
        ...
        komut P;
}
```

Şekil 2.1.3

Örneğin; verilere başlangıç değer atama algoritma bölümü, aşağıdaki program bölümü ile yazılacaktır:

```
{       // Baslangic degerler
        a = 3;
        b = -5.74;
        c = '@';
}
```

Not: {} parantezlerinden sonra noktalı virgül işareti koyulmaz.

Aşağıdaki program bölümünü yürüttükten sonra

```
{
        a = 2;
        b = 3;
        p = a;
        a = b;
        b = p;
}
```

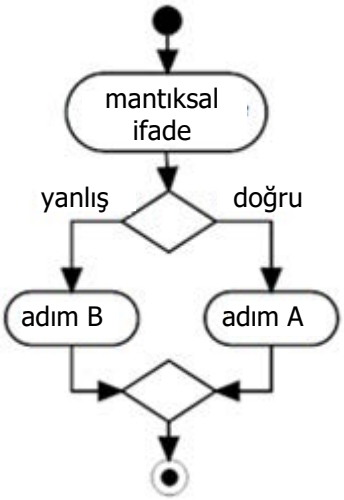
a ve b değişkenlerinin değerleri 3 ve 2 olacaktır.

2.2 Seçim İçin Algoritmik Kontrol Yapıları ve Seçim Kontrol Komutları

İki Olasılıklı Seçimi İçin Algoritmik Kontrol Yapısı
eğer-o zaman-değilse

İki olasılıklı seçim algoritmik kontrol yapısı, bir koşula bağlı olarak, algoritmadaki eylemin iki olası devam yönünden birini seçmek için kullanılır. Koşul bir mantıksal ifadedir. Mantıksal ifadenin yalnızca iki değeri olabilir: *doğru* ve *yanlış*. Bunlara *mantıksal değerler* denir ve bu nedenle ifadeye *mantıksal ifade* denir.

İki olasılıklı seçim için kontrol yapısı ***şekil 2.2.1***'de grafiksel olarak gösterilmiştir.



Şekil 2.2.1

Eğer mantıksal ifadenin değeri doğruysa **o zaman** A adımı, değilse (değer yanlış ise) B adımı yürütülür. Bu yüzden, bu kontrol yapısı **eğer-o zaman-değilse** (İng. if-then-else) olarak adlandırılır.

Yapının sözde dilde metinsel yazılışı ***şekil 2.2.2'***de verilmiştir.

```
eğer mantıksal ifade
        o zaman
                adım A;
        değilse
                adım B;
bitiş_eğer {mantıksal ifade}
```

Şekil 2.2.2

Yapı **eğer** ile başlayarak **bitiş_eğer** *{mantıksal ifade}* ile biter.

Örneğin, ***Şekil 2.2.3***'te bu yapı, a ve b sayılarından büyük olanı bulmak için kullanılmıştır.

```
eğer a > b
        o zaman
                dahabuyuk← a;
        değilse
                dahabuyuk← b;
bitiş_eğer {a > b}
```

Şekil 2.2.3

Bir algoritmada adımların daha genel olabileceğini ve birkaç başka adımdan veya bütün kontrol yapıdan oluşabileceğini söyledik. ***Şekil 2.2.2***'deki adım A ve adım B başka adımlardan oluşuyorsa, bunlar bir **algoritmik bloğu** temsil eder ve genellikle ***şekil 2.2.4***'teki gibi **başlangıç** ve **bitiş** sözcükleriyle işaretlenir. Demek ki, A1, A2... An ve B1, B2... Bm adımları **o zaman** olasılıkta, yani **değilse** olasılıkta algoritmik bloklardır.

```
eğer mantıksal ifade
        o zaman
                başlangıç
                        adım A1;
                        adım A2;
                        ...
                        adım An;
                bitiş
        değilse
                başlangıç
                        adım B1;
                        adım B2;
                        ...
                        adim Bm;
                bitiş
bitiş_eğer {mantıksal ifade}
```

Şekil 2.2.4

Örneğin, a ve b verilen iki sayıdan büyük olanı bulmak için bu kontrol yapısını kullanarak şu şekilde yazacağız:

```
dahabuyuk ← b;
eğer a > b
        o zaman
                dahabuyuk ← a;
bitiş_eğer {a > b}
```

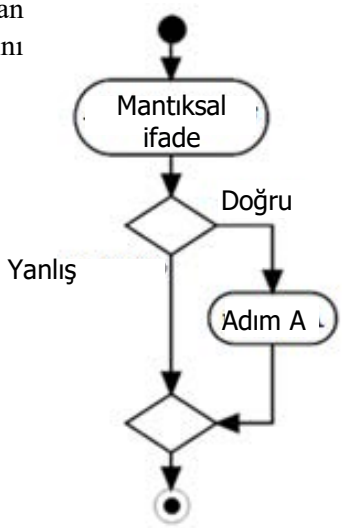
İki olasılıklı seçim için kontrol yapısı, mantıksal ifadenin olasılıklarından biri yürütme adımları içermediği zaman de kullanılabilir. Bu durumda, kontrol yapısının bir sonraki adımıyla devam ediliyor. Metinsel yazılışı ***şekil 2.2.5***'te verilmiştir.

```
eğer mantıksal ifade
        o zaman
                adim A;
bitiş_eğer {mantıksal ifade}
```

Şekil 2.2.5

Bu kontrol yapısına **eğer-o zaman** (İng.. if-then). denir.

eğer-o zaman kontrol yapısının grafiksel gösterişi ***şekil 2.2.6***'da verilmiştir.



Şekil 2.2.6

İki olasılıkli seçim için kontrol komutu if

if seçim kontrol komutu ile, **eğer- o zaman** iki olasılıklı seçimi için algoritmik kontrol yapısı gerçekleştirilir, ancak bu arada bir olasılıkta komut yoktur, ***şekil 2.2.5*** ve ***şekil 2.2.6.*** Mantıksal ifade, komut A'yı yürütmek veya yürütmemek için bir koşuldur, ***şekil 2.2.7.***

```
if (kosul)
        komut A;
```

Şekil 2.2.7

if kontrol komutunun yürütülmesi sırasında, önce *kosul* kontrol edilir ve geçerliyse (true değerine sahipse) komut A yürütülür , geçerli değilse (false değerine sahip ise), komut A yürütülmüyor ve programın yürütülmesi bir sonraki komutla devam ediyor.

Birden çok komutun yürütülmesi gerekirse, bunlar büyük parantezler içinde bir bloğa yerleştirilir, ***Şekil 2.2.8.***

kosul true değerine sahipse, o zaman{komut 1... komut m} bloğundaki komutlar yürütülür, aksi takdirde bloktaki komutlar yürütülmez.

kosul true veya false değerine sahip olabilen sadece mantıksal ifade olabilir.

```
if (kosul)  {
        komut 1;
        ...
        komut m;
}
```

Şekil 2.2.8

C++'da mantıksal ifadeler, ilişkisel ve mantıksal operatörlerle[1] oluşturulur.
Mantıksal ifade örnekleri şunlardır:

```
(a > b) && (a >= c)
((alfa < 90) && (beta > 0)) || (alfa – beta > 0)
!(x <= y)                    x > y                    ile aynısıdır
!((a > b) && (a >= c))       (a <= b) || (a < c)      ile aynısıdır
!((x != y) || (x > y))       (x == y) && (x <= y)     ile aynısıdır
```

Aşağıdaki üç örnek, **if** seçim kontrol komutunu göstermektedir:

a)

```
int x;
if(x < 0)
        xaps = –x;
```

b)

```
int x, y, pom;
if (true) {
      pom = x;
      x = y;
      y = pom;
}
```

c)

```
int x;
if (x != 0)
      x = 0;
```

[1] **1.7 Veri türleri** alt bölümünde **Mantıksal veri türleri** başlığında bunlardan bahsetmiştik.

Örnekler

Örnek 2.2.1

Girilen sayının pozitif olup olmadığını belirlemek için program bölümü yazılsın.

```
int sayi;
cout << "Sayi giriniz: ";
cin >> sayi;
if(sayi > 0) {
	cout << "Girilen sayi pozitiftir." << endl;
}
```

Örnek 2.2.2

Bir sayının mutlak değerini hesaplamak için program bölümü yazılsın.

```
int sayi;
cout << "Tam sayi giriniz: ";
cin >> sayi;
cout << "sayisinin mutlak degeri" << sayi;
if (sayi < 0) {
	sayi = -sayi;
}
cout << "sayisidir" sayi << endl;
```

İki olasılıklı seçim için kontrol komutu if-else

if-else seçim kontrol komutu ile **eğer- o zaman-değilse** iki olasılıklı seçim için algoritmik kontrol yapısı gerçekleştirilir, ***şekil 2.2.1*** ve ***şekil 2.2.2***.

if-else seçim kontrol komutu yürütülürken (***Şekil 2.2.9***), ilk önce *kosul* kontrol edilir ve geçerliyse (true değerine sahipse) komut A yürütülür, geçerli değilse (false değerine sahipse) komut B yürütülür.

Koşul geçerli veya geçerli olmayan durumda birden çok komutun yürütülmesi gerekiyorsa, bu komutlar büyük parantezler içinde blok içine alınır, ***şekil 2.2.10***.

kosul true değerine sahipse, {komut A1... komut Ap} bloğundaki komutlar yürütülür, *kosul* false değerine sahipse, {komut B1... komut Bq} bloğundaki komutlar yürütülür.

```
if (kosul)
	komut A;
else
	komut B;
```

Şekil 2.2.9

```
if (kosul) {
	komut A1;
	...
	komut Ap;
}
else {
	komut B1;
	...
	komut Bq;
}
```

Şekil 2.2.10

Örnekler

Örnek 2.2.3

Bir sorunun doğru cevabı bir mesajla, yanlış cevabı başka bir mesajla cevaplanması gerekiyorsa, bunu aşağıdaki program bölümü ile yazabiliriz:

```
char cevap;
cout << "Dogru icin D cevabi girin: ";
cin >> cevap;
if (cevap == 'D')
	cout << "cevap dogrudur." << endl;
else
	cout << "cevap yanlistir." << endl;
```

Örnek 2.2.4

alfa açısı 0°'den büyük ve 90°'den küçükse dar açıdır. Bu, aşağıdaki program bölümü ile yazılabilir.

```
int alfa;
cout << "Derece ile aci girin: ";
cin >> alfa;
if ((alfa > 0) && (alfa < 90))
	cout << "Açi dardir." << endl;
else
	cout << "Açi dar degildir." << endl;
```

Örnek 2.2.5

Aşağıdaki program bölümü doğrudur (derleyici hata göstermez), ancak yanlış çalışıyor:

```
bool evetHayir;
cin >> evetHayir;
if (evetHayir = true) {
	cout << "Cevap dogrudur." << endl;
	cout << "Ancak program bolumu hatalidir." << endl;
}
else
	cout << "Cevap yanlistir." << endl;
```

`EvetHayir` değişkeni için hangi değeri girersek girelim, bu program bölümü her zaman `Cevap dogrudur` yazdırır.

Hata, `evetHayir = true` koşulundadır, çünkü bu eşitlik koşulu değildir (== operatörüyle) atama komutudur (= operatörüyle). Yürütülmesinin sonucu her zaman doğrudur ve bu yüzden devamlı "`Cevap dogrudur`" yazdırılır.

Örnek 2.2.6

Sıkça, iki sayıyı bölerken paydanın değerini kontrol etmek gerekir çünkü 0' la bölmeye izin verilmez. Aşağıdaki program bölümü, 0' la bölmenin nasıl kaçınılması gerektiğini göstermektedir:

```
int n;
cout << "Tam sayi girin: ";
cin >> n;
if (n != 0)
    cout << "'in evrik değeri" << 1./ n << n << " 'dir." << endl;
else
       cout << "0 ile bolmeye izin verilmez." << endl;
```

Alıştırmalar

Alıştırma 2.2.1

Aşağıdaki program bölümlerinden hangileri aynı sonuç verir?

a)

```
if(a > b)
      cout << a;
Else
      cout << b;
```

b)

```
if(a > b)
      cout << a;
```

c)

```
if(a <= b)
      cout << b;
Else
      cout << a;
```

Alıştırma 2.2.2

Aşağıdaki program bölümleri arasındaki fark nedir?

a)

```
if((a == b) || (a == c))
       cout << a << endl;
else
       cout << b << ' ' << c << endl;
```

b)

```
if((a != b) && (a != c))
       cout << b << ' ' << c << endl;
else
       cout << a << endl;
```

Alıştırma 2.2.3

Aşağıdaki program bölümü ne yapar?

```
if(a < b)
      min = a;
else
      min = b;
 cout << min << endl;
```

Çözülmüş ödevler

Ödev 2.2.1
Girilen doğal sayının çift veya tek olup olmadığı kontrol edilsin.

```
1   // Girilen sayinin çift ya da tek
2   // olup olmadigi kontrol edilsin
3    #include <iostream>
4    using namespace std;
5
6   int main() {
7        int sayi;
8        cout <<  "Dogal sayi girin: ";
9        cin >> sayi;
10       if( sayi;  % 2 == 0 )
11           cout <<" Girilen sayi "  << sayi << " cifttir. " << endl;
12       else
13           cout << " Girilen sayi " << sayi << " tektir. " << endl;
14
15       cout << endl;
16       system( "Color 17" );
17       system( "pause" );
18       return 0;
19   }
```

Şekil 2.2.11

```
Dogal sayi girin: 123
Girilen sayi 123 tektir.

Press any key to continue . . .
```

Ödev 2.2.2
Girilen sayının pozitif, negatif veya sıfır olup olmadığı kontrol edilsin.

```
1   // Girilen sayinin pozitif, negatif veya sifir
2   // olup olmadigi kontrol edilsin.
3    #include <iostream>
4    using namespace std;
5
6   int main() {
7        float  sayi;
8        cout << " Bir sayi girin: ";
9        cin >> sayi;
10       cout << "Girilen sayi "  << sayi << "  : ";
11       if ( sayi    < 0) {
12           cout << " negatiftir." << endl;
13       }
14       if (broj > 0) {
15           cout <<" pozitiftir." << endl;
16       }
```

Şekil 2.2.12

```
17      if (sayi == 0) {
18          cout <<" sifirdir." << endl;
19      }
20
21      cout << endl;
22      system("Color 17");
23      system("pause");
24      return 0;
25  }
```

Şekil 2.2.12 (devam)

Programı yürüttükten sonra olası bir çıktı şudur:

```
Bir sayi girin: -12.345
Girilen sayi -12.345: negatiftir.

Press any key to continue . . .
```

Ödev 2.2.3

Girilen sayının karşılık gelen sayı türü aralığında olup olmadığı kontrol edilsin.

Not: **1.8 Verilerin Okunması ve Yazdırılması** alt bölümünde **Sayısal Verilerini Okuma ve Yazdırma** alt başlığında, taşmanın (İng. overflow) nasıl gerçekleştiğini açıklamıştık. Taşmanın meydana gelmemesi için, hatalı sonuçlar alınabileceğinden dolayı genellikle bir verinin, değişkenin bildirildiği türün değer aralığında olup olmadığını kontrol etmek gerekir.

Aşağıdaki örnekte, tamsayıların taşması gösterilmiştir.

```
1   // Tamsayilarin tasmasi
2   #include <iostream>
3
4   using namespace std;
5
6   int main() {
7
8       int tamSayi;
9       double ussu3;
10      cout << " int turunun araligi: [" << INT_MIN
11          << ", " << INT_MAX << "]" << endl;
12      cout << " 1290'da daha buyuk tamsayi girin: ";
13      cin >> tamSayi
14      na3 = 1.0 * tamSayi * tamSayi * tamSayi;
15      cout << tam Sayi   << " tamsayisi ussu 3: " ;
16      if( ( INT_MIN <= ussu3 ) && ( ussu3 <= INT_MAX ) )
17          cout << int( ussu3 ) << endl;
18      else {
```

Şekil 2.2.13

```
19          cout << "Tasma meydana geldi." << endl;
20          cout << " Cunku " <<tamSayi    <<"ussu 3 "
21              <<tamSayi *tamSayi*tamSayi<< "degildir."<< endl;
22      }
23
24      cout << endl;
25      system( "Color 17" );
26      system( "pause" );
27      return 0;
28  }
```

Şekil 2.2.13 (devam)

Programın yürütülmesiyle olası bir çıktı şudur:

```
int turunun araligi: [-2147483648, 2147483647] 'dir
1290'da daha buyuk tamsayi girin: 1291
1291 tamsayisi ussu 3: Tasma meydana geldi.
Cunku 1291 ussu 3 -2143282125 degildir

Press any key to continue . . .
```

Alıştırma ödevleri:

Aşağıdaki ödevler için programlar yazılsın:

1. n doğal sayısının karesi ve küpü hesaplansın.
2. r yarıçaplı dairenin çevresi ve alanı hesaplansın.
3. Üç basamaklı doğal sayı okunsun ve ortadaki rakam yazdırılsın.
4. Üç sayı okunsun ve bunların bir üçgenin kenarları olup olamayacağı belirlensin.
5. Üç sayı büyüklüğe göre sıralansın.
6. Doğrusal denklem çözülsün: $ax + b = 0$, $a \neq 0$ için.
7. Doğrusal eşitsizlik çözülsün: $ax + b > 0$, $a \neq 0$ için.
8. 180o'den küçük bir açı okunsun, dar veya geniş olup olduğu yazdırılsın.
9. Girilen tarihin doğru olup olmadığı kontrol edilsin. (Tarih; gün, ay ve yıl olarak üç tamsayı olarak girilsin).
10. Bir kişinin yaşı, bugünün tarih ile doğum tarihi arasındaki fark olarak hesaplansın. Tarihler gün, ay ve yıl için tam sayı olarak girilsin.

if ve if-else kontrol komutların iç içe yerleştirilmesi

if ve if-else kontrol komutları, if seçimi veya else seçimi içinde iç içe yerleşmiş (İng.nested) olabilir.

İç içe yerleştirme else seçiminde olduğunda, böyle ifadeye if-else-if denir.

Aşağıdaki örneği inceleyelim.

Örnek 2.2.7

Üç tam sayı a, b ve c dik üçgenin kenarları olabilir mi?

Çözüm: $a^2 + b^2 = c^2$ veya $a^2 + c^2 = b^2$ veya $b^2 + c^2 = a^2$ ise a, b ve c sayıları dik üçgenin kenarları olabilir.

Bu cümleyi aşağıdaki program bölümü ile ifade edebiliriz:

```
if(a * a + b * b == c * c)
       cout << "Evet";
else   if(a * a + c * c == b * b)
       cout << "Evet";
else   if (b * b + c * c == a * a)
       cout << "Evet";
else
       cout << "Hayir";
```

İç içe yerleştirme, if seçiminde de olabilir.

Aşağıdaki örneği inceleyelim.

Örnek 2.2.8

Verilen üç sayıdan a, b ve c, en büyüğü bulunsun.

Çözüm:

Eğer a > b ve a > c, ise o zaman en büyük sayı a'dır. Eğer b > a ve b > c, ise o zaman en büyük sayı b'dir Eğer c > a ve c > b, ise o zaman en büyük sayı c'dir. Program bölümü şöyle olacaktır:

```
if(a > b)
       if(a > c)
              cout << "En buyuk sayi" << a;
if(b > a)
       if(b > c)
              cout << "En buyuk sayi" << b;
if(c > a)
       if(c > b)
              cout << "En buyuk sayi" << c;
```

Not: İç içe yerleştirilmiş seçim komutu hem **if** seçiminde hem **else** seçiminde olamaz.

if, if-else ve if-else-if seçim kontrol komutlarını iç içe yerleştirirken, C++ çeviricinin her birini önceki if ile ilişkilendirmesine dikkat edilmelidir. Örneğin, aşağıdaki program bölümünde a < b için hiçbir şey yazdırılmayacaktır çünkü else birinci if'e değil ikinci if'e bağlıdır.

```
if(a > b)
	if(a > c)
		cout << "En büyük sayi" << a;
	else
		cout << "En buyuk sayi" << a;
```

a < b için aşağıdaki komutun yürütülmesini istersek.

```
cout << "En buyuk sayi" << a; << "degildir.";
```

o zaman else birinci if ile parantezler yardımıyla ilişkilendirilmelidir.

```
if(a > b) {
	if(a > c)
		cout << "En büyük sayi" << a;
}
else
	cout << "En buyuk sayi" << a;  << "degildir.";
```

Buna **sarkan-else problemi** denir (İng. dangling-else problem).

Alıştırmalar

Örnek 2.2.7 ve Örnek 2.2.8 için, C++'da ayrı ayrı programlar yazılsın.

Çözülmüş ödevler

Ödev 2.2.4

Derslerden alınan notların ortalaması biliniyorsa, öğrencinin genel başarısı yazdırılsın:

Ortalama	Genel başarı
1 ile 1.5 arası	Yetersiz
1.6 ile 2.5 arası	Yeterli
2.6 ile 3.5 arası	İyi
3.6 ile 4.5 arası	Çok iyi
4.6 ile 5.0 arası	En iyi

C++'daki program ***şekil 2.2.14***'teki gibi olacaktır.

```
1   // Ortalama not biliniyorsa, ogrencinin genel basarisi yazdirilsin.
2   #include <iostream>
3   using namespace std;
4
5   int main() {
6       float ortalama;
7       cout << " Ortalama notu girin: ";
8       cin >> ortalama;
9       cout << ortalama <<" ortalama notu icin genel basari sudur:";
10      if (ortalama <= 1.5)
11          cout << "Yetersiz" << endl;
12      else    if (ortalama <= 2.5)
13          cout << "Yeterli" << endl;
14      else    if (ortalama  <= 3.5)
15          cout << "Iyi"  << endl;
16      else    if (ortalama   <= 4.5)
17          cout << "Cok iyi" << endl;
18      else
19          cout << "En iyi" << endl;
20
21      cout << endl;
22      system("Color 17");
23      system("pause");
24      return 0;
25  }
```

Şekil.2.2.14

```
Ortalama notu girin: 3.56
3.56 ortalama notu icin genel basari sudur: Cok iyi
```

if veya else içinde yalnızca bir komut olduğunda, büyük parantezlerin konulabileceğinin ancak zorunlu olmadığını hatırlayalım.

Ödev 2.2.5
Verilen üç sayıdan en büyüğü belirlensin.

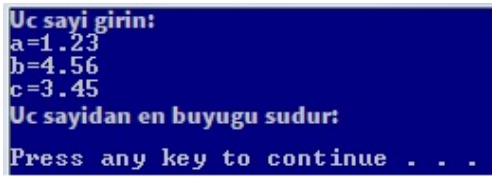
C++'daki program ***şekil 2.2.15***'te verilmiştir.

```
1   // Uc sayıdan en buyugunu belirlemek.
2
3   #include <iostream>
4   using namespace std;
5
6   int main() {
7       double a, b, c;
8       cout <<         "Uc sayi girin: \n";
9       cout << "a="; cin >> a;
```

Şekil.2.2.15

```
10      cout << "b="; cin >> b;
11      cout << "c="; cin >> c;
12      cout <<  "Uc sayidan en buyugu sudur: "  ;
13      if (a > b) {
14          if (a > c)
15              cout << a << endl;
16          else
17              cout << c << endl;
18      }
19      else {
20          if (b > c)
21              cout << b << endl;
22          else
23              cout << c << endl;
24      }
25
26      cout << endl;
27      system("Color 17");
28      system("pause");
29      return 0;
30  }
```

Şekil.2.2.15 (devam)



Koşullu Operatör ?:

Koşullu operatör, basit if-else komutlarını temsil etmek için kullanılır. Onun söz dizimi şöyledir:

kosul? ifade T : ifade N;

İlk olarak, değeri true veya false olabilen *kosul* hesaplanır. Eğer değeri true ise, T ifadesi hesaplanıyor, false ise N ifadesi hesaplanıyor.

Örneğin, aşağıdaki program bölümü:

```
if(m > n)
        max = m;
else
        max = n;
```

Şu şekilde daha kısa yazılabilir:

```
max = (m > n) ? m : n;
```

Aşağıdaki komutla, sayi değişkeninin çift değeri varsa "cifttir" yazdırılır veya sayi değişkeni tek değere sahipse "tektir" yazdırılır:

```
cout << "sayi" << sayi << " sayisi "
             << ((sayi % 2 == 0) ? "cifttir" : "tektir") << endl;
```

Örneğin, sayi = 23 için şu yazdırılacaktır:

```
23 sayisi tektir.
```

Alıştırma Ödevleri

Seçim kontrol komutların iç içe yerleştirilmesi ile aşağıdaki ödevleri çözmek için programlar yazılsın:

1. İki sayı ve bir aritmetik operatör girilsin (+, –, * veya /) ve işlem gerçekleştirilsin.
2. Romen rakamının Arapça karşılığı belirlensin. (Romen rakamları şunlardır: I = 1, V = 5, X = 10, L = 50, C = 100, D = 500 ve M = 1 000).
3. Değerlendirme aşağıdaki puan cetveline göre yapılırsa kazanılan puan sayısına göre öğrencinin notu belirlensin: 60'tan az - yetersiz, 60'tan 69'a kadar - yeterli, 70'ten 79'a kadar - iyi, 80'den 89'a kadar - çok iyi ve 90'dan fazla – en iyi.
4. Geçerli not için koşul 100 puan üzerinden 60 puan kazanmak ise öğrencinin geçerli not alıp almadığı koşullu operatör ?: ile belirlensin.
5. Koşullu operatörü ?: kullanarak $O(x_1, y_1, r_1)$ ve $O(x_2, y_2, r_2)$ çemberlerinin kesişip kesişmediği belirlensin. (İki nokta arasındaki mesafe $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ formülüne göre hesaplanır).

Çok olanaklı seçim için algoritmik kontrol yapısı durum ve çok olanaklı seçim için kontrol komutu switch

Algoritmada eylemi sürdürmek için birkaç olasılık olduğu zaman, **çok olanaklı seçim yapmak için algoritmik kontrol yapısı kullanılır,** ***şekil 2.2.16.*** Olasılıklardan birinin seçimi, bir verinin veya bir aritmetik ifadenin değerine bağlı olarak yapılır.

İfade'nin değeri a, b... k olabilir veya *ifade* bu değerlerden hiçbiri olmayabilir. *İfade'nin* a değeri alması durumunda işlem A adımıyla devam eder, *ifade* değeri b olması durumunda işlem B adımıyla devam eder vs. *ifade'nin* a, b... k değerlerinden hiçbirini almaması durumunda işlem

```
durum ifade
    a: adim A;
          çıkış;
    b: adim B;
          çıkış;
    . . .
    k: adim K;
          çıkış;
    değilse
        adim X;
bitiş_durum {ifade}
```

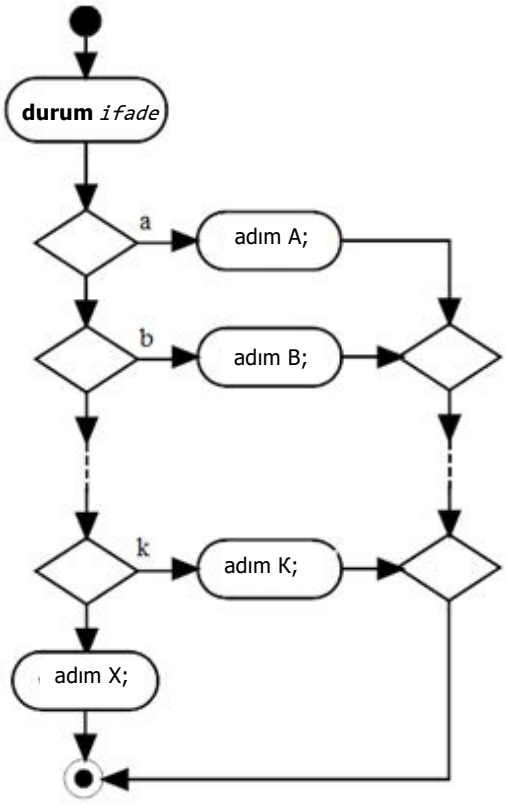
Şekil 2.2.16

X adımıyla devam eder.

Şimdiye kadar söylediklerimizden, bu algoritmik kontrol yapısının, işlemin birkaç olası devam etme durumundan birini seçtiği görülebilir. Bu yüzden bu yapıya **durum seçimi** veya kısaca sadece **durum** (İng. case) olarak adlandırılır.

a, b...k adımlarından sonra, durum kontrol yapısında kalan adımların yürütülmesinin kesildiği **çıkış,** algoritmik kontrol yapısı belirtilir ve işlem **durum** ardından gelen adımla veya kontrol yapıyla devam ediyor. (**çıkış** algoritmik kontrol yapısı hakkında **2.4 Atlama İçin Algoritmik Kontrol Yapıları ve Atlama İçin Kontrol Komutları** alt bölümünde daha ayrıntılı bahsedeceğiz).

Bu kontrol yapısı ***şekil 2.2.17***'de grafiksel olarak gösterilmiştir.



Şekil 2.2.17

Birkaç farklı ifade değeri için aynı adımı gerçekleştirmek gerektiğinde **durum** yapısı ***şekil 2.2.18'***deki gibi yazılır.

```
durum ifade
        a, c, d:    adim A;
                    çıkış;
        b, g:       adim B;
                    çıkış;
        . . .
        k, i, f, j: adim K;
                    çıkış;
        değilse
                    adim X;
    bitiş_durum {ifade}
```

Şekil 2.2.18

switch kontrol komutuyla, **durum** çok olanaklı seçim algoritmik kontrol yapısı uygulanır, sözdizimi ise ***şekil 2.2.19***'da verilmiştir.

Parantezdeki ifade (*ifade*), değeri tam sayı türünden (short, int, long, long long), karakter türünden (char), mantıksal türünden (bool) veya sayılı türünden (enum) olan herhangi bir ifade olabilir. a, b... k işaretleri, tam sayı, karakter, mantıksal veya sayılı türden sabitlerdir. Onlar değişken olamaz.

```
switch(ifade) {
     case a:       komut A;
                   break;
     case b:       komut B;
                   break;
     ...
     case k:        komut K;
                    break;

   default:
                  komut X;
}
```

Şekil 2.2.19

Her durum (case) için, eylemi switch kontrol komutunun sonunda atlama (çıkış) olan ilk break kontrol komutuna kadar tüm komutlar yürütülür.

Bu yüzden, her case'ten sonra break kontrol komutu kullanılmalıdır. break kontrol deyimi belirtilmemişse, yürütme akışı bir sonraki case[2] ile devam eder. Komut X'ten sonra break koyulmuyor.

Eğer *ifade*, case kelimesinden sonra belirtilen sabitlerden hiçbir değer almazsa, default kelimesinden sonraki komut X yürütülür. Default bölümü zorunlu değildir, belirtilmesi gerekmez. Eğer yoksa, herhangi bir case'i yürüttükten sonra, switch kontrol komutundan bir sonraki komuta atlanır.

Örnekler

Örnek 2.2.9

Genç veya yetişkin olup olmadığınızı onaylayan program bölümü şöyle olabilir:

```
bool EvetHayir;
char EH;
cout << "21 yuzyilda mi dogmussunuz? (E,H): "; cin >> EH;
EvetHayir = true;
if(EH == 'H')
       EvetHayir = false;
switch(EvetHayir) {
       case true:    cout << "Siz gencsiniz." << endl;
              break;
       case false:   cout << "Siz yetişkinsiniz." << endl;
}
```

Örnek 2.2.10

Aritmetik işlemn işareti girilsin: +, - * veya / ve sonuç yazdırılsın.

```
switch(isaret) {
case '+': cout << isaret << " arti " << endl;
       break;
case '-': cout << isaret << " eksi " << endl;
       break;
case '*': cout << isaret << " carpi " << endl;
```

[2] Bu sıkça yapılan bir hatadır, bu nedenle her case'in sonunda break koyulmasına dikkat edilmelidir.

```
        break;
case '/': cout << isaret << " bolu " << endl;
        break;
default:
        cout << isaret << " aritmetik islem isareti degildir." << endl;
}
```

Farklı durumlar (cases) için ilk break komutuna kadar aynı komutlar grubu yürütülebilir.

Örnek 2.2.11

Bir harf girilsin ve harfin sesli veya sessiz olup olmadığı yazdırılsın.

```
switch(bukva) {
case 'a':
case 'e':
case 'i':
case 'o':
case 'u':
        cout << "harf " << " harfi seslidir." << endl;
        break;
default:
        cout << "harf " <<  " harfi sessizsdir." << endl;
}
```

Örnek 2.2.12

Yılın bir ayı sıra numarasıyla verilmişse, switch kontrol komutuyla o ayın kaç gün olduğu belirlenebilir.

Not: Artık yılında Şubat 29 gündür. Artık yılı 4'e bölünen ve 100'e bölünmeyen veya 400'e bölünen yıldır.

```
switch(ay){
        case 1: case 3: case 5: case 7: case 8: case 10: case 12:
                denovi = 31;
                break;
        case 4: case 6: case 9: case 11:
                gun = 30;
                break;
        case 2:
                gun = (((yil % 4 == 0) && (yil  % 100 != 0)) ||
                (yil  % 400 == 0)) ? 29 : 28;
}
        cout << yil  << " yilinda " << ay << " -nci ayin "
        << gun << " gunu vardir." << endl;
```

Örnek 2.2.13

Aylar sıralandığında, önceki ödev için program bölümü şu şekilde olacaktır:

```
enum aylar  {OCAK, SUBAT, MART, NİSAN, MAYİS, HAZİRAN,
             TEMMUZ, AGUSTOS, EYLUL, EKİM, KASİM, ARALİK
}
aylar ayEnum = SUBAT;
switch(ayEnum) {
       case OCAK: case MART: case MAYİS: case TEMMUZ: case AGUSTOS:
       case EKİM: case ARALİK:
              gun = 31;
              break;
       case NİSAN: case HAZİRAN: case EYLUL: case KASİM:
              gun = 30;
              break;
       case SUBAT:
              gun = (((yil % 4 == 0) && (yil % 100 != 0)) ||
              (yil % 400 == 0)) ? 29 : 28;
}
       cout << yil << "yilinda " << ayEnum << "-nci ayin "
       << gun  << " gunu vardir." << endl;
```

Önceki 5 örneği bir programda yazın ve çalıştırın.

Çözülmüş ödevler

Ödev 2.2.6

5, 4, 3, 2 veya 1olarak sayısal notları verilmiş bir öğrencinin genel başarısını yazdıracak program yazılsın.

Program ***şekil 2.2.20***'de verilmiştir.

```
1   // Ogrencinin genel basarisi
2   #include <iostream>
3   using namespace std;
4
5   int main() {
6       int not;
7       cout << "Bir not girin (1,2,3,4,5): ";
8       cin >> not;
9       cout <<"Ogrencinin genel basarısı: ";
10      switch( not;  ) {
11          case 5:     cout <<"En iyi " << endl;
12              break;
13          case 4:     cout <<"Cok iyi " << endl;
14              break;
```

Şekil 2.2.20

```
15          case 3:     cout << "Iyi " << endl;
16              break;
17          case 2:     cout << "Yeterli " << endl;
18              break;
19          case 1:     cout <<"Yetersiz " << endl;
20              break;
21          default:    cout <<" Bu not degildir " << endl;
22      }
23
24      cout << endl;
25      system( "Color 17" );
26      system( "pause" );
27      return 0;
28  }
```

Şekil 2.2.20 (devam)

Programın yürütülmesiyle olası bir çıktı şudur:

```
Bir not girin       (1,2,3,4,5): 5
Ogrencinin genel basarısı: En iyi

Press any key to continue . . .
```

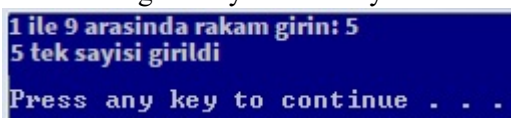
Ödev 2.2.7

1 ile 9 arasında girilen bir rakamın çift veya tek olup olmadığını yazdıracak program yazılsın. Rakam 1 ile 9 arasında değilse, bir yorum yazdırılsın.

```
1   // Cift ve tek rakamlar
2   #include <iostream>
3   using namespace std;
4
5   int main() {
6       short rakam;
7       cout <<  "1 ile 9 arasinda rakam girin: ";
8       cin >> rakam;
9       switch(rakam ) {
10          case 1: case 3: case 5: case 7: case 9:
11              cout <<rakam <<"tek sayisi girildi." << endl;
12              break;
13          case 2: case 4: case 6: case 8:
14              cout <<rakam <<"cift sayisi girildi." << endl;
15              break;
16          default: cout <<"Girilen karakter:" <<rakam << endl;
17      }
18
19      cout << endl;
20      system( "Color 17" );
21      system( "pause" );
22      return 0;
23  }
```

Şekil 2.2.21

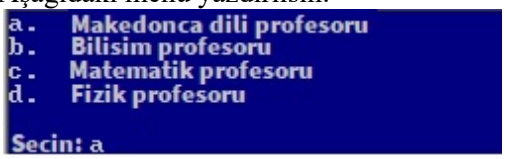
Programın yürütülmesiyle olası bir çıktı şudur:



Alıştırma ödevleri

Aşağıdaki ödevler için switch çok olanaklı seçim için kontrol komutunu kullanarak programlar yazılsın:

1. 10'dan küçük doğal sayı girin ve sayının asal[3], 2'yle bölünebilir, 3'le bölünebilir veya kusursuz[4] olup olmadığı yazdırılsın.
2. Aşağıdaki menü yazdırılsın:



Ardından girilen harfe göre (a, b, c, d) söz konusu profesörün adı yazdırılsın.

1. 1, 2, 3... 9, 0 rakamlardan biri girilsin ve rakam kelimelerle yazdırılsın. Örneğin: 1 için "bir" kelimesi yazdırılsın, 2 için "iki" kelimesi yazdırılsın vb.
2. Romen rakamının Arapça karşılığı belirlensin. (Romen rakamları şunlardır: I = 1, V = 5, X = 10, L = 50, C = 100, D = 500 ve M = 1 000).
3. 60 puandan az - yetersiz, 60'tan 69'a kadar - yeterli, 70'den 79'a kadar - iyi, 80'den 89'a kadar - çok iyi ve 90'dan 99'a kadar – en iyi puan ölçeğine göre değerlendirme yapılarak öğrencinin kazandığı puan sayısına göre notu belirlensin.

Bilgiyi Kontrol Etme Soruları

1. Sıralı algoritmik kontrol yapısı ne zaman (en sıkça olarak) kullanılır?
2. Hangi seçim için algoritmik kontrol yapılarını biliyorsunuz? Onların metinsel biçimlerini yazın.
3. C++'da hangi algoritmik kontrol komutuyla, **eğer-o zaman-değilse** iki olasılıklı seçim için algoritmik kontrol yapısı gerçekleştirilir? Bir kaç örnek verin.

[3] Asal sayı, yalnızca 1'e ve kendisine bölünebilen sayıdır. Asal sayılar şunlardır: 2, 3, 5, 7, 11 vb.

[4] Kusursuz sayı, kendisi hariç bölenlerinin toplamına eşit olan sayıdır. Kusursız sayılar şunlardır: 6 (= 1 + 2 + 3), 28 (= 1 + 2 + 4 + 7 + 14), 496, vb.

4. **eğer-o zaman-değilse** algoritmik kontrol yapısını grafiksel olarak gösterin.
5. Seçim algoritmik kontrol yapılarında ve seçim kontrol komutlarında koşul olarak kullanılan ifade türü nedir?
6. If ve/veya else'den sonra birden fazla komut olduğunda, iki seçenekli seçim için kontrol komutu nasıl yazılır? Örnekler verin.
7. Bir olasılıktan sonra komut yoksa, iki olasılıktan seçim yapmak için kontrol komutunun söz dizimini yazın. Örnekler verin.
8. Kontrol seçimi komutlarının iç içe yerleştirmenin hangi yollarını biliyorsunuz?
9. else seçiminin iç içine yerleştirilmiş seçim kontrol komutuna ne ad verilir?
10. Sarkan else'in ne olduğunu açıklayın. Örnek verin.
11. Koşullu operatörünün söz dizimini yazın. Örnekler verin.
12. Çok olanaklı seçim için algoritmik kontrol yapısını metinsel ve grafiksel olarak gösterin.
13. Çok olanaklı seçim kontrol komutunda ifade hangi türden olabilir?
14. Switch kontrol komutunda her case'in sonunda hangi kontrol komutu koyulur?
15. Bir case sonunda break komutu yoksa ne olur?

Ödevler

1. "n" değişkeninin değeri 78 ise aşağıdaki program bölümünün yürütülmesinden sonra ne yazdırılacaktır?

```
if((n % 2 == 0) && (n % 3 == 0))
   cout << n << " 6 ile bolunur." << endl;
else
   cout << n << " 6 ile bolunmez." << endl;
```

2. Aşağıdaki program bölümünde hangi hata yapılmıştır:

```
if(x = a)
   cout << "x'in a ile ayni degeri var.";
else
   cout << "x'in a'dan farkli degeri var.";
```

3. Aşağıdaki program bölümü, n'in değerinden bağımsız olarak neden her zaman "false" yazdırıyor?

```
cin >> n;
if(n = 0)
          cout << "true" << endl;
   else
          cout << "false" << endl;
```

4. Aşağıdaki program bölümüyle ne yazdırılacaktır?

```
int i = 0, n;
float a = 1;
char c = 'a';
a++;
if(c == a && i == 0)
   n = 1;
else
   n = 2;
cout << "n = " << n << endl;
```

5. Aşağıdaki iki program bölümü aynı sonucu veriyor mu?

```
if(a > b)
   cout << a << " > " << b << endl;
else if(b < c)
   cout << a << " <= " << b << "<" << c << endl;
else
   cout << a << " <= " << b << ">=" << c << endl;
```

```
if(a > b)
   cout << a << " > " << b << endl;
if(b < c)
   cout << a << " <= " << b << " < " << c << endl;
else
   cout << a << " <= " << b << " >= " << c << endl;
```

6. İki değişkenli mantıksal ifade için aşağıdaki koşullara göre program bölümü yazılsın:
   - ifade, degisken1 veya degisken2'den biri doğruysa doğrudur.
   - ifade hem degisken1 hem degisken2 doğruysa yanlıştır.
   - ifade hem degisken1 hem degisken2 yanlışsa yanlıştır.
7. Üç tam sayı girilsin ve en büyüğü; diğer ikisinden büyükse bir kez, üçüncüden büyük diğeriyle aynı değere sahipse iki kez, üç sayının hepsi eşitse üç kez yazdırılsın.
8. **Ödev 2.2.4**, girilen ortalama notunun [0.0, 5.0] aralığında olup olmadığını kontrol edecek şekilde tamamlansın.
9. **Ödev 2.2.6**, girilen notunun [1, 5] aralığında olup olmadığını kontrol edecek şekilde tamamlansın.
10. Üç gerçek sayının bir üçgenin kenarları olup olamayacağı kontrol edilsin?
11. Aşağıdaki işlemlerle basit bir hesap makinesini simüle edecek program yazınız: +, –, *, %, /, x^2.

2.3 Tekrarlama Algoritmik Kontrol Yapıları ve Tekrarlama Kontrol Komutları

Tekrarlama kontrol yapıları, bir grup algoritmik adımları birden çok kez yürütmek gerektiğinde kullanılır. Adımların bir uygulaması bir **döngü** olarak adlandırılır, ***şekil 2.3.1.***

```
döngü
      adim A;
      adim B;
        ...
      adim M;
bitiş_döngü
```

Şekil 2.3.1

Bu yüzden, bir grup algoritmik adımların yürütülmesinin bu şekilde tekrarlanmasına **döngüsel tekrarlama** denir.

Bu kontrol yapıları daha kolay anlamak için aşağıdaki ödevi inceleyelim.

Ödev: İlk 10 doğal sayının toplamı bulunsun.

Bir kabımız olduğunu düşünelim ve içine sayıları koyalım. Başlangıçta kap boştur, yani toplam 0'dır. Kabımızda koyduğumuz ilk doğal sayı 1'dir, yani toplam şimdi 0 + 1 = 1'dir. Ardından, kabta sonraki doğal sayı 2'yi koyuyoruz, demek ki kabta iki sayı olacak (1 ve 2), toplam ise 0 + 1 + 2 olacaktır. Ardından 3, 4, 5, 6, 7, 8, 9 ve 10 sayılarını koyuyoruz, ve toplam her eklenen sayi için artacaktır. Son toplam 0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 olacaktır.

Toplamı bulma sürecinde sadece iki işlemin gerçekleştirildiğini görebiliriz:

- Toplama sayı eklemek.
- Sayıyı 1 için arttırmak.

Sayı'nın 1'den 10'a kadar değiştiğinden ve toplamın 0, 1, 3, 6, 10 vb. olduğundan dolayı, sayi ve toplam olmak üzere iki değişken tanıtabiliriz. Başlangıçta toplam değişkeninin içeriği 0'dır, sayi'nin başlangıç değeri ise 1'dir. Bu yüzden, sayi ve toplam değişkenlerini 1 ve 0 değerleriyle başlatıyoruz. Algoritma, metinsel olarak şekil 2.3.2'deki gibi yazılabilir.

```
toplam ← 0;
sayi ← 1;
döngü 10 kez
        – sayiyi toplama ekle;
        – sayiyi 1 icin arttir;
bitiş_döngü
```

Şekil 2.3.2

Tekrarlamayı durdurma yöntemine bağlı olarak, tekrarlama için üç kontrol yapısı ayırt ediyoruz, yani:

- Döngü saymalı kontrol yapısı.
- Döngü başında çıkışlı kontrol yapısı.
- Döngü sonunda çıkışlı kontrol yapısı.

Saymalı tekrarlama için algoritmik kontrol yapıları ve for kontrol komutu

Bu kontrol yapıda döngünün tekrarlandığı sayısı önceden bilinmektedir. Tekrarlamalar, yani döngüler, ilk ve son değerlerin verildiği özel sayaç ile sayılır, ***şekil 2.3.3.***

Döngüler, sayac değişkeninin başlangıç değerinden son değerine kadar 1 için **artarak** değişkenin her değeri için yürütülür. Sayac değişkeni her döngüden önce 1 için artar. Bu yüzden, bu kontrol yapısı **için - arttır – kadar** olarak adlandırılır.

```
için sayac ← baslangic arttır sona kadar
        adim A;
        adim B;
        ...
        adim K;
bitiş_için {sayac}
```

Şekil 2.3.3

Sayac, baslangic ve son değişkenleri aynı türden değişkenler olmalıdır. baslangic değişkeni, son değişkeninden küçük veya ona eşit değere sahip olmalıdır. Aksi takdirde hiçbir döngü yürütülmez. Döngünün oluşmuş olduğu algoritmik adımlarda sayacın değeri değişmemelidir.

Başlangıç (baslangic) değerinin son (son) değerinden büyük olması gereken tekrarlama gerekirse, o zaman benzer bir kontrol yapısı kullanılır. Bu yapıda, döngüler sayac değişkeninin baslangictan sona doğru 1 için **azalarak** her değeri **için** yürütülür. Sayaç değişkeni her döngüden önce 1 için azaltılır.

```
için sayac ← baslangic azalt sona kadar
        adim A;
        adim B;
        ...
        adim J;
bitiş_için {sayac}
```

Şekil 2.3.4

Bu yüzden. bu tür bir kontrol yapısına **için-azalt-kadar** denir, ***şekil 2.3.4.***

Sayaçın 1 için değil, adım için verilmiş değer kadar arttığı veya azaldığı algoritmalarda (ödevlerde), kontrol yapısı **için - kadar - adım** olarak adlandırılır, ***şekil 2.3.5.***

```
için sayac ← baslangic sona kadar adım deger
        adim A;
        adim B;
        ...
        adim M;
bitiş_için {sayac}
```

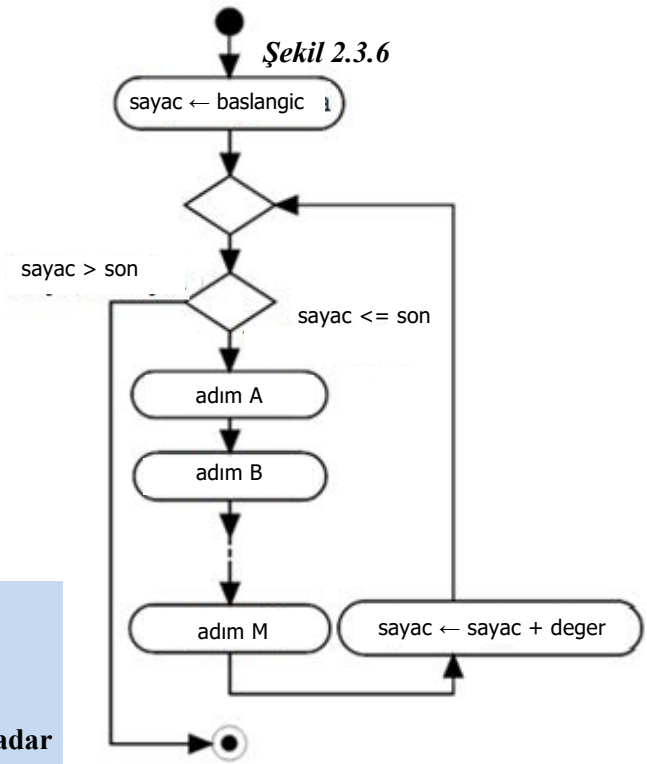
Şekil 2.3.4

Bu kontrol yapıda, sayac başlangic değerinden son değerine kadar her döngüde belirli bir değer kadar adım adım artar. Bu yüzden, bu tekrarlama kontrol yapısı **için - kadar - adım** olarak adlandırılır.

için - kadar - adım kontrol yapısındaki adım değeri (deger) +1 ise, o zaman **için - arttır - kadar** kontrol yapısı elde edilir, adim değeri -1 ise, **için - azalt - kadar** kontrol yapısı elde edilir.

için - kadar - adım kontrol yapısının grafik gösterimi ***şekil 2.3.6'*** da verilmiştir.

için - arttır - kadar kontrol yapısıyla ifade edilmiş ilk 10 doğal sayının toplamını bulma ödevinin algoritması ***şekil 2.3.7'*** de verilmiştir.



Şekil 2.3.6

```
algoritma Toplam10aKadar
başlangıç
        toplam ← 0;
        sayi ← 1;
        için sayac ← 1 arttır 10'a kadar
                toplam ← toplam + sayi;
                sayi ← sayi + 1;
        bitiş_için {sayac}
        yazdır toplam;
bitiş {Toplam10aKadar}
```

Şekil 2.3.7

için - arttır - kadar, için - azalt - kadar ve **için - kadar - adım** üç kontrol yapıyı gerçekleştirmek için C++'da for kontrol komutu kullanılır.

for kontrol komutunun genel formu ***şekil 2.3.8'***de verilmiştir.

Başlatma, sayaca başlangıç değerini atama bölümüdür. Bu bölüm, döngü başlamadan önce bir kez yürütülür.

Koşul, döngüyü tekrar çalıştırmadan önce koşulu kontrol etme kısmıdır. Koşul mantıksal ifadedir. *Koşul* true değerine sahipse döngü tekrar yürütülür, false değerine sahipse döngü yürütülmez ve işlem akışı bir sonraki komutla devam eder.

```
for (başlatma; koşul; güncelleme){
        adim A;
        adim B;
    . . .
        adim R;
}
```

Şekil 2.3.8

Koşul, ilk döngü başlamadan önce de karşılanmayabileceğinden dolayı, hiçbir döngü yürütülmez olabilir.

for kontrol komutundaki güncelleme kısmı, her döngünün yürütülmesinden sonra yürütülür ve içinde çoğunlukla sayaç güncellenir.

için-kadar-adım kontrol yapısı için for kontrol komutunun formu ***şekil 2.3.9***'da verilmiştir:

```
for (sayac = baslangic; sayac <= son; sayac = sayac + deger) {
                komut A;
                komut B;
                ...
                komut L;
}
```

Şekil 2.3.9

Başlatma bölümünde, sayac değişkeni başlangic değeriyle başlatılır. Bu bölüm sadece bir kez yürütülür. Sayac değerinin son değerinden küçük veya ona eşit (sayac <= son) olup olmadığı koşulunun kontrol edilmesi komutun kendisinde yerleşiktir. sayac <= son koşulu (kosul) her döngü başlamadan önce kontrol edilir ve true değerine sahipse döngü yürütülür, false değerine sahipse döngü yürütülmez.

Şekil 2.3.7'deki algoritmaya göre for komutunu kullanarak ilk 10 doğal sayının toplamını bulma ödevinin programı ***şekil 2.3.10***'da verilmiştir.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {     // for ile ilk 10 dogal sayinin toplami
5
6       int sayac,sayi,toplam;
7       toplam = 0;
8       sayi = 1;
9       for( sayac = 1; sayac <= 10;  sayac = sayac + 1 ) {
10        toplam = toplam + sayi ;
11           sayi = sayi + 1;
12      }
13      cout << " Ilk 10 dogal sayinin toplami:" << toplam << endl;
14
15      cout << endl;
16      system( "Color 17" );
17      system( "pause" );
18      return 0;
19  }
```

Şekil 2.3.10

```
Ilk 10 dogal sayinin toplami:55
Press any key to continue . . .
```

Örnekler

Örnek 2.3.1

10'dan başlayarak 1'e kadar ilk 10 doğal sayının toplamı bulunsun, yani 10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1.

Program ***şekil 2.3.11'***de verilmiştir. Programın ***Şekil 2.3.10***'daki ile aynı olduğunu görebiliriz, fark, sayi değişkeninin 10 olarak başlatılması, sayacın 10 olarak başlatılması, koşulun sayac >= 1 olarak değiştirilmesi ve for komutunun güncelleme bölümünde, sayacın 1 için azalmasıdır.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {    // for ile ilk 10 dogal sayinin toplami
5
6       int sayac,sayi,toplam;
7       toplam = 0;
8       sayi = 10;
9       for( sayac = 10; sayac >= 1; sayac = sayac - 1 ) {
10          toplam += broj;
11          sayi += 1;
12      }
13      cout << " İlk 10 dogal sayinin toplami:" << toplam << endl;
14
15      cout << endl;
16      system( "Color 17" );
17      system( "pause" );
18      return 0;
19  }
```

Şekil 2.3.11

Ornek 2.3.2

Girilen sayı için 10'a kadar çarpma tablosu yazdırılsın.

Program ***şekil 2.3.12***'de verilmiştir. Programda, bir sonraki değişkenin yazdırılacağı yerlerin sayısını belirten setw() fonksiyonun ve yazdırılan değeri sağa hizalayan right fonksiyonun kullanıldığı <iomanip> kütüphanesi dahil edilmiştir.

Örneğin, x'in değeri 1 234 ise aşağıdaki komutla

```
cout << setw(10 << right << x << endl;
```

x'in değeri 10 basamaklı alanda sağa hizalanmış olarak yazdırılacaktır:

␣␣␣␣␣␣123.

```
1   #include <iostream>
2   #include <iomanip>
3    using namespace std;
4
5   int main() {    //  for ile carpma tablosu
6
7       int sayac, sayi;
8       cout << " (1, 2,...,9) ile carpma tablosu: ";
9       cin >> sayi;
10      for(sayac = 1;  sayac  <= 10; sayac = sayac  + 1 ) {
11          cout << "\n" << setw( 3 ) << right << sayac   << " x " << sayi << " = "
12              << setw( 3 ) << sayac * sayi;
13      }
14
15      cout << endl;
16      system( "Color 17" );
17      system( "pause" );
18      return 0;
19  }
```

Şekil 2.3.12

```
(1, 2,...,9) ile çarpma tablosu: 7
  1 x 7 =   7
  2 x 7 =  14
  3 x 7 =  21
  4 x 7 =  28
  5 x 7 =  35
  6 x 7 =  42
  7 x 7 =  49
  8 x 7 =  56
  9 x 7 =  63
 10 x 7 =  70
Press any key to continue . . .
```

Örnek 2.3.3

Büyük ve küçük harfler arasındaki karakterler olmadan G'den g'ye kadar olan harfler yazdırılsın.

Harf sayacının başlatma bildirimi, aşağıdaki programda olduğu gibi for komutunun başlatma bölümünde de yapılabilir.

```
1    #include <iostream>
2    using namespace std;
3
4   int main() {    // G'den g'ye kadar harflerin yazdirilmasi (for ile)
5
6       for( char  harf = 'G';   harf <= 'g'; harf = harf + 1 ) {
7           cout << harf << " ";
8       }
9
```

Şekil 2.3.13

```
10      cout << endl;
11      system( "Color 17" );
12      system( "pause" );
13      return 0;
14  }
```

Şekil 2.3.13 (devam)

```
g f e d c b a Z Y X W V U T S R Q P O N M L K J I H G
Press any key to continue . . .
```

Örnek 2.3.4

f(x) = $3x^2$ – 2x + 4 için x∈ [–30,30] fonksiyonu, elde edilen çıktıda gösterildiği gibi, adımın değeri 4 ile tablolansın. Program ***şekil 2.3.14***'te verilmiştir.

```
1   #include <iostream>
2   #include <iomanip>
3   using namespace std;
4
5   int main() {     // fonksiyonun tablolaması (for ile)
6
7       double x, y;
8       cout << setw( 7 ) << "x" << setw( 15 ) << "y=3x^2-2x+4" << endl << endl;
9       for( int sayac = -30; sayac <= 30; sayac += 4 ) {
10          x = sayac;
11          y = 3 * x * x - 2 * x + 4;
12          cout << setw( 10 ) << x << setw( 10 ) << y << endl;
13      }
14
15      cout << endl;
16      system( "Color 17" );
17      system( "pause" );
18      return 0;
```

Şekil 2.3.14

Programın çıktısı şudur:

```
      x     y=3x^2-2x+4

    -30          2764
    -26          2084
    -22          1500
    -18          1012
    -14           620
    -10           324
     -6           124
     -2            20
      2            12
      6           100
     10           284
     14           564
     18           940
     22          1412
     26          1980
     30          2644
```

Arttırma ve Azaltma Operatörleri ++ , —

C++'da veri değerini 1 için artırma ve azaltma için özel operatörler getirilmiştir. Bunlara **arttırma operatörü** (İng. increment operator) ve **azaltma operatörü** (İng. decrement operator) denir. Arttırma operatörü ++'dır , azaltma operatörü ise – –'dir. x verisinin değerinin arttırılması veya azaltılması ön ek (++x veya – –x) veya son ek (x++ veya x– –) şeklinde yapılabilir. İlk durumda artırma veya azaltma, x verisinin katıldığı ifadenin hesaplanmasından önce (ön artırma), ikinci durumda ise, ifadenin hesaplanmasından sonra (arttırma sonrası) gerçekleştirilir.

Operatör	Şekli	Artma	Azalma
++	++x	ön ekli	
	x++	son ekli	
– –	– –x		ön ekli
	x– –		son ekli

Örnek:

x'in hesaplamadan önceki değeri	İfade	İfadenin değeri	x'in hesaplamadan sonraki değeri
5	++x	6	6
5	x++	5	6
5	– –x	4	4
5	x– –	5	4

Artırma ve azaltma komut olarak kullanılabilir.

Örnekler

Örnek 2.3.5

Önceki örneklerden ilk 10 doğal sayının toplamı aşağıdaki program bölümü ile hesaplanabilir:

```
int sayi = 1;
int toplam = 0;
for (int sayac = 1; sayac <= 10; sayac++) {
       toplam += sayi;
       ++sayi;       // sayi = sayi + 1; veya sayi += 1; ile aynidir.
}
```

Örnek 2.3.6

Aşağıdaki program bölümü, artırma operatörünün kullanımını göstermektedir:

```
int a, b, c;
a = b = c = 1;
cout << "a = " << a << ", b = " << b << ", c = " << c << endl;
b = ++a;
cout << "b = ++a \na = " << a << ", b = " << b << ", c = " << c << endl;
c = a++;
cout << "c = a++ \na = " << a << ", b = " << b << ", c = " << c << endl;
c = ++ ++b;
cout << "c = ++ ++b \na = " << a << ", b = " << b << ", c = " << c << endl;
```

Çıktısı şudur:

```
a = 1, b = 1, c = 1
b = ++a
a = 2, b = 2, c = 1
c = a++
a = 3, b = 2, c = 2
c = ++ ++b
a = 3, b = 4, c = 4

Press any key to continue . . .
```

Arttırma ve azaltma operatörleri, sadece ASCII karakterleri ve tam sayılar gibi doğrusal sıralı kümelerde kullanılabilir. (Doğrusal sıralı kümeler, her öğenin bir öncülü (ilki hariç) ve ardılı (sonuncusu hariç) olduğu kümelerdir).

Önceki örneklerde, aşağıdaki komutların yerine;

```
sayac = sayac + 1;
sayac = sayac - 1;
```

şu komutlar kullanılabilir ;

```
sayac++;      veya    ++sayac;
sayac--;      veya    --sayac;
```

Aşağıdaki komutlar yerine ise ;

```
harf = harf + 1;
harf = harf - 1;
```

şu komutlar kullanılabilir;

```
harf++;       veya    ++harf;
harf--;       veya    --harf;
```

Sayaçın for kontrol komutunun gövdesinde kullanılması

1'den 10'a kadar ilk 10 doğal sayının toplamı programında, sayi değişkeninin başlangıç değeri için 1 verildiği ve her döngüde 1 için arttığı görülmektedir. Ayrıca sayaç (sayac) 1 ile başlatılır ve her döngüde 1 için artar. Her döngüdeki değerlerini yazdırırsak aynı olduklarını göreceğiz.

```
Sayac   Sayi

  1       1
  2       2
  3       3
  4       4
  5       5
  6       6
  7       7
  8       8
  9       9
  10      10

İlk 10 dogal sayinin toplamı: 55
```

Böyle durumlarda sayac değişkeni, sayi değişkeni yerinde kullanılabilir. Fakat, programlamaya yeni başlayanlar, tekrarlama kontrol komutlarının gövdesinde sayaçları kullanırken çok dikkatli olmalıdır çünkü kolayca hata yapılabilir ve bir şey daha. Tekrarlama kontrol komutlarının gövdesinde her zaman sayaç kullanılamaz.

İlk 10 doğal sayının toplamı programında, for ifadesinin gövdesinde sayacın kullanımı ***Şekil 2.3.15***'teki gibi olacaktır.

Ayrıca, programda sayacın başlatılması bildirim bölümünde de yapılabileceği gösterilmiştir.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {     // for ile ilk 10 dogal sayinin toplami
5
6       int toplam = 0;
7       for( int   sayac = 1; sayac <= 10; sayac ++ ) {
8           toplam = toplam + sayac;
9       }
10      cout << "İlk 10 dogal sayinin toplami: " << toplam << endl;
11
```

Şekil 2.3.15

```
12      cout << endl;
13      system( "Color 17" );
14      system( "pause" );
15      return 0;
16  }
```

Şekil 2.3.15 (devam)

for Kontrol Komutunun Özellikleri

a) Birden fazla değişkenin başlatılması ve güncellenmesi

C++'da **for** kontrol komutu, birden fazla değişkenin başlatılmasına ve güncellenmesine izin verir. Bu arada, onlar birbirinden virgülle ayrılırlar.

Örnek 2.3.7

1'den n'e kadar ve n'den 1'e kadar sayıların toplamının aynı olup olmadığı kontrol edilsin.

Programda (***Şekil 2.3.16***), for ifadesinin başlatma bölümünde ve güncelleme bölümünde iki değişken başlatılır ve güncellenir.

Tekrarlamayı durdurma koşulu, ilişkisel ve mantıksal operatörlerin kullanılabileceği herhangi mantıksal ifade olabilir. Bu programda koşul i <= n && j >= 1'dir.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // 1+2+...+n toplami n+(n-1)+(n-2)+... 2+1 toplamiyla ayni midir(for ile)
6
7       int n, toplam1denNyeKadar=0, toplamNden1eKadar=0;
8       cout << "Hangi sayiya kadar, n= "              ;
9       cin >> n;
10      for( int i = 1, j = n; i <= n && j >= 1; i++, j-- ) {
11        toplam1denNyeKadar += i;
12        toplamNden1eKadar  += j;
13      }
14      cout << "1'den " << n << " 'e kadar dogal sayilarin toplamı" << endl;
15      cout << toplam1denNyeKadar << " 'dir" << endl;
16
17      cout << endl;
18      system( "Color 17" );
19      system( "pause" );
20      return 0;
21  }
```

Şekil 2.3.16

Programın yürütülmesiyle olası bir çıktı şudur:

```
Hangi sayiya kadar, n=100
1'den 100 'e kadar dogal sayilarin toplamı 5050'dir
100'den 1'e kadar dogal sayilarin toplamı 5050'dir

Press any key to continue . . .
```

b) Başlatma ve güncelleme bölümü boş olabilir

Örnek 2.3.8

Örnek 2.3.7'deki **program** ***Şekil 2.3.17***'deki gibi de yazılabilir. Başlatılmanın for komutu başlamadan önce yapıldığını ve güncellemenin for komutunun gövdesinde yapıldığını görüyoruz.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // 1+2+...+n toplami n+(n-1)+(n-2)+... 2+1 toplamiyla ayni midir (for ile)
6
7       int i, j, n, toplam1denNyeKadar=0, toplamNden1eKadar=0;
8       cout << "Hangi sayiya kadar, n= " ;
9       cin >> n;
10      i = 1; j = n;
11      for( ; i <= n && j >= 1;) {
12        toplam1denNyeKadar += i;
13        toplamNden1eKadar  += j;
14          i++;
15          j--;
16      }
17      cout << "1'den "  << n << " 'a kadar dogal sayilarin toplamı: " <<toplam1denNyeKadar << endl;
18      cout  << n << " 'dan 1'e kadar dogal sayilarin toplamı:" << toplamNden1eKadar << endl;
19
20      cout << endl;
21      system( "Color 17" );
22      system( "pause" );
23      return 0;
24  }
```

Şekil 2.3.17

```
Hangi sayiya kadar, n=99
1'den  99'a kadar dogal sayilarin toplamı: 4950
99'dan 1'e  kadar dogal sayilarin toplamı: 4950
Press any key to continue . . .
```

Alıştırmalar

Alışırma 2.3.1

Aşağıdaki program bölümü ne yapıyor?

```
toplam = 0;
for(int sayi = 2, sayi <= 10; sayi += 2)
	toplam += sayi;
```

Alıştırma 2.3.2

Aşağıdaki program bölümünün yürütülmesi sonucu olarak neyin yazdırılacağı belirlensin.

```
for(int x = 1, i = 0; x <= 10; x++)
	i += x;
cout << x << endl;
```

Alıştırma 2.3.3

Aşağıdaki program bölümünün yürütülmesi sonucunda neyin yazdırılacağı belirlensin:

```
for(int i = 0, j = 0; i < 5; i++, j--)
	cout << i + j << endl;
```

Çözülmüş ödevler

Ödev 2.3.1

n doğal sayısından küçük tüm Pisagor sayıları bulunsun.

Açıklama: Pisagor sayıları, x2 + y2 = z2 denkleminin geçerli olduğu x, y ve z doğal sayılarının herhangi üçlüsüdür. 3, 4, 5 ve 4, 3, 5 gibi üçlülerin tekrarlamaması için, x değişkeni 1'den y'den büyük olmayan değere kadar değişecektir, yani (eğer x = y) $\left[\sqrt{\frac{n^2}{2}}\right]$. Orta parantezler, ondalık sayının tam sayı kısmını göstermektedir. Örneğin, [3.56] = 3.

Not 1: Aşağıdaki programda (***Şekil 2.3.18***), $\left[\sqrt{\frac{n^2}{2}}\right]$ ve $x^2 + y^2$ değerlerinin hesaplandığı iki yardımcı değişken yard1 ve yard2 değişkenleri kullanılmaktadır. $\left[\sqrt{\frac{n^2}{2}}\right]$ değeri sabit olduğu için for ifadesinde bırakırsak her döngüde hesaplanacaktır ki bu gereksizdir. Ayrıca, $x^2 + y^2$ ifadesi her döngüde iki kez hesaplanır, ancak yard2'nin tanıtılmasıyla sadece bir kez hesaplanır. Bu şekilde $x^2 + y^2$ ifadesinin hesaplanma süresi iki kat kısalır.

Not 2: Bir gerçek sayının karekökünün sonucu gerçek sayı olduğundan ve döngüde tam sayıya ihtiyacımız olduğundan, (int) sqrt (n*n/2) ile gerçek sayının tam sayıya zorunlu dönüştürme (tür dökümü) gerçekleştiriyoruz.

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

int main() {
    // Pisagor ucluleri (for ile)

    int n, x, y, z,  yard1, yard2, sayac  = 0;
    cout <<"Hangi dogal sayiya kadar: "; cin >> n;
    cout <<  n  "\n sayisina kadar Pisagor ucluleri sunlardir" << endl;
    pom1 = ( int ) sqrt( n * n / 2 );
    for( x = 1; x <= yard1; x++ ) {
        for( y = x + 1; y <= n; y++ ) {
           yard2 = x * x + y * y;
            z = ( int ) sqrt( yard2 );
            if( ( z <= n ) && ( yard2 == z * z ) ) {
                ++ sayac;
                cout << setw( 10 ) << sayac  << ": " << setw( 2 ) << x << ", "
                    << setw( 2 ) << setw( 2 ) << y << ", " << setw( 2 ) << z << endl;
            }
        }
    }

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

Şekil 2.3.18

Programın bir çıktısı şudur:

```
Hangi sayiya kadar: 33

33 sayisina kadar Pisagor ucluleri sunlardir:
         1:  3,  4,  5
         2:  5, 12, 13
         3:  6,  8, 10
         4:  7, 24, 25
         5:  8, 15, 17
         6:  9, 12, 15
         7: 10, 24, 26
         8: 12, 16, 20
         9: 15, 20, 25
        10: 18, 24, 30
        11: 20, 21, 29

Press any key to continue . . .
```

Ödev 2.3.2

n doğal sayısından küçük tüm kusursuz sayılar bulunsun.

Açıklama: Kendisi hariç bölenlerinin toplamına eşit olan doğal sayılara kusursuz sayılar denir. Kusursuz sayılar şunlardır: 6, 28, 496, vb.

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

int main() {
    //   Kusursuz sayilar (for ile)

    int n, sayi, bolen, toplam;
    cout << "Hangi sayiya kadar, n= "; cin >> n;
    cout << '\n' << n << "sayisina kadar kusursuz sayilar sunlardir: " << endl;
    for( sayi  = 2; sayi  <= n; sayi ++ ) {
      toplam = 1;
        for( bolen = 2; bolen  <= sayi / 2; bolen ++ ) {
            if( sayi  %  bolen  == 0 )
            toplam += bolen;
        }
        if( toplam == sayi )
            cout << sayi << ", ";
    }
    cout << endl;

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

Şekil 2.3.19

Şekil 2.3.19'da verilen programın yürütülmesi sonucu şu olacaktır:

```
Hangi sayiya kadar, n =12345
12345 sayisina kadar kusursuz sayilar
sunlardir:
6, 28, 496, 8128,

Press any key to continue . . .
```

Alıştırma ödevleri

Aşağıdaki ödevler için for kontrol komutunu kullanarak programlar yazılsın:

1. $1 + 4 + 7 + 11 + \ldots + n$ toplamı hesaplansın.
2. $1^2 + 2^2 + 3^2 + \ldots + n^2$ toplamı hesaplansın.
3. $1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$ toplamı hesaplansın.
4. 1'den n'ye kadar doğal sayıların aritmetik ortalaması hesaplansın.
5. $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$ hesaplansın. 0! = 1 ve 1! = 1 varsayılır. (n! işareti "n faktöriyel" olarak okunur).
6. $1 \cdot \frac{1}{2} \cdot \frac{1}{3} \cdot \ldots \cdot \frac{1}{n}$ çarpımı hesaplansın.
7. A'dan Z'ye ve Z'den A'ya alfabenin harfleri yazdırılsın.
8. Aşağıdaki üç form yazdırılsın:

```
*
**
***
****
*****
******
*******
********
*********
```

```
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
6 5 4 3 2 1
7 6 5 4 3 2 1
8 7 6 5 4 3 2 1
9 8 7 6 5 4 3 2 1
```

```
                1
              1 2
            1 2 3
          1 2 3 4
        1 2 3 4 5
      1 2 3 4 5 6
    1 2 3 4 5 6 7
  1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
```

9. Aşağıdaki toplam hesaplansın:
   $1 + (1 + 2) + (1 + 2 + 3) + (1 + 2 + 3 + 4) + \ldots + (1 + 2 + \ldots + n)$.
10. Negatif olmayan n sayısının çift faktöriyeli bulunsun.
    n tek sayı için, $n!! = n(n-2)(n-4) \cdot \ldots \cdot 5 \cdot 3 \cdot 1$,
    n çift sayı için, $n!! = n(n-2)(n-4) \cdot \ldots \cdot 6 \cdot 4 \cdot 2$.
    0!! = 1 ve 1!! = 1 varsayılır.
11. n sayı girilsin ve onlardan kaç tanesinin çift, kaç tanesinin tek olduğu sayılsın.
12. n doğal sayısının bölenleri bulunsun.

Döngünün Başında Çıkışlı iken-yürütür Tekrarlama Algoritmik Kontrol Yapısı ve while Kontrol Komutu

Bir önceki alt başlıktan ilk 10 doğal sayının toplamını bulma ödevinde sadece iki işlemin tekrarlandığını söylemiştik:

- Toplama sayi eklemek.
- Sayıyı 1 için arttırmak.

Sayi değişkeni her döngüde 11 değerine kadar artar. Bu yüzden şöyle diyebiliriz: sayinin 10'dan küçük veya eşit değeri var **iken**, bu iki işlemi içeren döngü **yürütülür**. Böyle bir algoritmik kontrol yapısına **iken-yürütür** denir. Bu kontrol yapısı ***şekil 2.3.20***'de gösterilmiştir.

```
toplam ← 0;
sayi ← 1;
iken sayi ≤ 10 yürütür
        – sayi'ti toplam'a ekle;
        – sayi'yi 1 için arttır;
bitiş_iken {sayi ≤ 10}
```

Şekil 2.3.20

Bu kontrol yapının genel formunu, ***şekil 2.3.21***'deki gibi yazılabilir.

```
iken koşul yürütür
        adım A;
        adım B;
        ...
        adım M;
bitiş_iken {koşul}
```

Şekil 2.3.21

Bu yapı, A, B... M adımlarından oluşur. A'dan M'ye kadar olan adımların bir kez yürütülmesi bir döngüdür. Her döngünün başlangıcından önce, iki değere: *doğru* (true) veya *yanlış* (false) sahip olabilen mantıksal ifade tanımlayan koşulun (*koşul*) geçerli olup olmadığı kontrol ediliyor. *Koşul* geçerli (doğru) olduğunda, döngü adımları yürütülür ve işlem döngünün başına döner.

*Koşul'*un belirli bir kontrolünde, geçerli olmadığı (doğru olmadığı) belirlendiğinde, kontrol yapısı atlatılır ve işlem, **bitiş_iken**{*koşul*} ardından bir sonraki adımla veya bir sonraki algoritmik kontrol yapıyla devam ediyor.

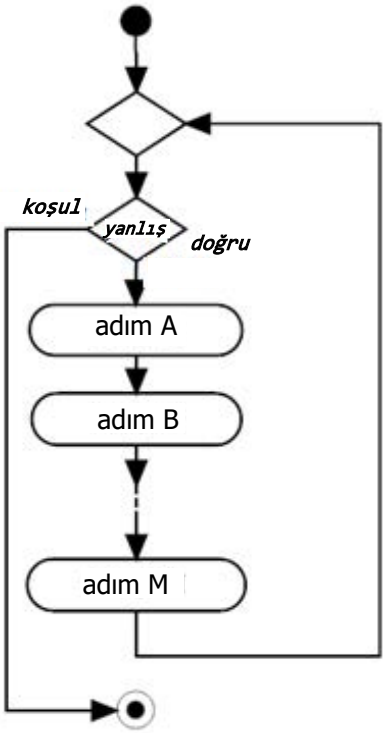
*koşul'*un her döngü başlamadan önce kontrol edildiğinden dolayı, ilk kontrolde bu *koşul* karşılanmayabilir ve bu nedenle *hiçbir döngü yürütülmeyebilir.*

iken–yürütür algoritmik kontrol yapısının grafik gösterimi ***şekil 2.3.22'***de verilmiştir.

iken-yürütür algoritmik kontrol yapısını uygulamak için, C++'da while kontrol komutu kullanılır. Yazılışı ***şekil 2.3.23***'te verilmiştir.

```
while (kosul) {
        komut B;
        komut B;
        ...
        komut H;
}
```

Şekil 2.3.23



Şekil 2.3.22

While kontrol komutu için, ilk döngüden önce *kosul* geçerli değilse, while komutu bir kez bile yürütülmez.

Örnekler

Örnek 2.3.9

iken-yürütür algoritmik kontrol yapısını kullanarak bir önceki alt başlıktan ilk 10 doğal sayının toplamını bulma ödevi için algoritma ***şekil 2.3.24***'deki gibi olacak, while komutu ile yazılan program ise ***şekil 2.3.25***'te verilmiştir.

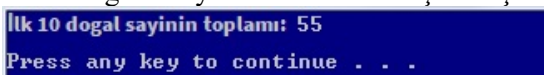
algoritma *Toplam10aKadar*
başlangıç
toplam ← 0;
sayi ← 1;
iken sayi ≤ 10 **yürütür**
toplam ← toplam + sayi;
sayi ← sayi + 1;
bitiş_iken {sayi ≤ 10}
yazdır toplam;
bitiş {*Toplam10aKadar*}

Şekil 2.3.24

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {    // Ilk 10 dogal sayinin toplami (while ile)
5
6       int  sayi, toplam;
7       toplam = 0;
8        sayi = 1;
9       while( sayi  <= 10 ) {
10      toplam = toplam + sayi;
11           sayi =  sayi  + 1;
12      }
13      cout << "Ilk 10 dogal sayinin toplamı: "  << toplam << endl;
14
15      cout << endl;
16      system( "Color 17" );
17      system( "pause" );
18      return 0;
19  }
```

Şekil 2.3.25

Programın yürütülmesinin bir çıktısı şudur:



Örnek 2.3.10
Pazarda ne kadar paran harcandığı hesaplansın.

Kaç ürün aldığımızı bilmemiz gerektiğinden dolayı her ürün için ödediğimiz tutarın miktarını gireceğiz. Girişin sonu için bir kontrol ayarlamamız gerekiyor. Her ürün için belirli bir miktar ödediğimizi biliyoruz ve bu pozitif bir rakamdır. Bu yüzden, girişin sonunda bir ürün için ödeme tutarı olamayan belirli bir sayı gireceğiz. Örneğin, tutar için -1, -99 veya 9999 gireceğiz (bir ürün için tam olarak bu tutarı ödemiş olmamız az olasıklıdır) vb. Tekrarlamanın önceden belirlenmiş bir değerle bu şekilde kontrolüne bitiş kontrollü tekraralama (İng. Sentinel-Controlled Repetition) denir.

While komutu ile ifade edilen program ***şekil 2.3.26***'da verilmiştir.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       //  Pazar (while ile)
6
7       double  hesapTutari, toplamHarcama = 0;
8       cout << " Bitis icin, hesap tutari olarak -99 girin" << endl;
9       cout << "\nBirinci hesabin tutarini girin: ";
10      cin >> hesapTutari;
11      while( hesapTutari !  = -99 ) {
12          toplamHarcama += hesapTutari;
13          cout << "Siradaki hesabin tutarini girin: ";
14          cin >> hesapTutari;
15      }
16      cout << "\n'Toplam olarak " << toplamHarcama  << " denar harcanmistir. " << endl;
17
18      cout << endl;
19      system( "Color 17" );
20      system( "pause" );
21      return 0;
22  }
```

Şekil 2.3.26

Programı yürütükten sonra, şunu elde ediyoruz:

Alıştırmalar

Alıştırma 2.3.4

Aşağıdaki program bölümünde hata bulunsun:

```
float x = 5, toplam = 0;
while(x >= 0)
	toplam = toplam + x;
```

Alıştırma 2.3.5

Aşağıdaki program bölümünü yürüttükten sonra neyin yazdırılacağı belirlensin:

```
int i = 0;
while(i < 5) {
	i = i + 2;
	cout << i << "\n";
}
```

Alıştırma 2.3.7

Aşağıdaki program ne yazdırıyor?

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

int main() {
	int n, s = 0, i = 1;
	double x;
	cout << "Dogal sayi girin: "; cin >> n;
	while(i <= n) {
		cout << "Ondalik sayi girin: "; cin >> x;
		s += x;
		i++;
	}
	cout << "s = " << s << endl;
	return 0;
}
```

Çözülmüş Ödevler

Ödev 2.3.3
Bir doğal sayının kaç basamaklı olduğu belirlensin.

Açıklama: n=174 sayısı verilsin.
n'yi (= 174) 10 ile bölersek bölüm 17 kalan 4 olur.
Bölümü n'ye atayacağız (n = 17).
n'yi (= 17) 10 ile bölersek bölüm 1 kalan 7 olur.
Bölümü n'ye (n = 1) atayacağız.
n'yi (= 1) 10 ile bölersek bölüm 0, kalan 1 olur.
Bölümü n'ye atayacağız (n = 0).
İki işlemle döngü oluşturabileceğimizi görüyoruz:

- n'yi 10 ile bölmek
- Elde edilen bölümün n'ye atanması.

Bölmenin kalanı 0 olduğunda (n = 0) tekralama biter.
Bu süreçle, n'nin başlangıç (girilen) değeri kaybolur. Algoritmayı çalıştırdıktan sonra da n'nin başlangıç değerini kullanmak gerekiyorsa, onu girdikten sonra başka bir değişkende kaydetmek gerekir.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       //  Tamsayinin basamak sayisi (while ile)
6
7       int  tamSayi, basamakSayisi, bolum;
8       cout <<" Tamsayi girin, n= " ;
9       cin >> tamSayi;
10      cout << tamSayi  <<"tamsayisinin"<<;
11      bolum = tamSayi  / 10;
12      basamakSayisi  = 1;
13      while( bolum !  != 0 ) {
14          tamSayi  = bolum;
15          bolum = tamSayi / 10;
16          basamakSayisi++;
17      }
18      cout << basamakSayisi << " basamagi vardir."  << endl;
19
20      cout << endl;
21      system( "Color 17" );
22      system( "pause" );
23      return 0;
24  }
```

Şekil 2.3.27

Program yürütüldükten sonra, şunu elde ediyoruz:

```
Tamsayi girin, n=2123456789
2123456789 tamsayisinin 10 basamagi vardir.

Press any key to continue . . .
```

Ödev 2.3.4

İki doğal sayının en büyük ortak böleni bulunsun.

Açıklama: İki doğal sayının en büyük ortak böleni (EBOB), eğer sayılar birbirine bölünebiliyorsa bunlardan küçük olanı olabilir veya her iki sayının da bölünebildiği daha küçük sayıdan küçük olan ilk sayı olabilir. Bu yüzden, önce EBOB'un iki sayıdan küçük olduğunu varsayıyoruz ,eğer değilse o zaman verilen iki sayıyla bölünebilen bir sayı bulana kadar EBOB'u her döngüde 1 için azaltıyoruz.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       //  Iki sayinin EBOB'unu (while ile)
6
7       int a, b, ebob;
8       cout << "Iki dogal sayi girin " << endl;
9       cout << "a = "; cin >> a;
10      cout << "b = "; cin >> b;
11      if( a < b )
12         ebob = a;
13      else
14         ebob = b;
15      while( ( a % ebob != 0 ) || ( b % ebob != 0 ) )
16        ebob --;
17      cout <<a << " ve " << b <<" sayilarinin en buyuk ortak boleni: "
18          <<ebob << endl;
19
20      cout << endl;
21      system( "Color 17" );
22      system( "pause" );
23      return 0;
24  }
```

Şekil 2.3.28

Programı yürüttükten sonra bir çıktı örneği şudur:

```
Iki dogal sayi girin
a = 1248
b = 2364
1248 ve 2364 sayilarinin en  buyuk ortak boleni: 12

Press any key to continue . . .
```

Ödev 2.3.5
Bilgisayarın düşündüğü doğal sayı tahmin edilsin.

Programın başında rand() fonksiyonu ile rastgele bir sayı üretilir. (Bu fonksiyon, **2.5 Fonksiyonlar** alt bölümünün **Kütüphane Fonksiyonları** alt başlığında işlenmiştir). rand() ile , 0 ile 32 767 arasında rastgele bir sayı üretilir. rand() % 100 ile, rastgele sayılar kümesi 0 ile 99 aralığına düşürülür, 1 + rand() % 100 ile ise, küme 1 ile 100 aralığına düşürülür.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {     // Dusunulen sayi (while ile)
5
6       int sayi, dusunulenSayi, deneme;
7       dusunulenSayi = 1 + rand() % 100;
8       cout << " 1 ile 100 arasinda dusundugum sayiyi bulun: " ;
9       cin >> sayi;
10      deneme = 1;
11      while( sayi  != dusunulenSayi ) {
12          if( sayi  > dusunulenSayi )
13              cout << " Dusunulen sayi " << sayi ;
14          else
15              cout << " Dusunulen sayi " << sayi ;
16          cout << "\nTekrar deneyin: ";
17          cin >> sayi;
18          deneme++;
19      }
20      cout << "Aferin! sayiyi "  <<deneme << " denemede buldunuz. "<< endl;
21
22      cout << endl;
23      system( "Color 17" );
24      system( "pause" );
25      return 0;
26  }
```

Şekil 2.3.29

```
1 ile 100 arasinda dusundugum sayiyi bulun: 50
Dusunulen sayi 50 sayisindan daha kucuktur
Tekrar deneyin: 25
Dusunulen sayi 25 sayisindan daha buyuktur
Tekrar deneyin: 37
Dusunulen sayi 37 sayisindan daha kucuktur
Tekrar deneyin: 30
Dusunulen sayi 30 sayisindan daha kucuktur
Tekrar deneyin: 26
Dusunulen sayi 26 sayisindan daha buyuktur
Tekrar deneyin: 28
Aferin! sayiyi  6 denemede buldunuz
Press any key to continue . . .
```

Alıştırma Ödevleri

Aşağıdaki ödevler için while kontrol komutunu kullanarak programlar yazılsın:

1. 0 sayısı girilene kadar klavyeden girilen tam sayıların çarpımı hesaplansın.
2. c = (b – a) / n adımı ile a'dan b'ye kadar olan sayıların toplamı, yani a + (a + c) + (a + 2c) + (a + 3c)... ifadesinin değeri hesaplansın.
3. n doğal sayısının rakamlarının toplamı hesaplansın.
4. İki doğal sayının en küçük ortak katı (EKOK) bulunsun.
5. Bir doğal sayının palindrom olup olmadığı kontrol edilsin. (Bir palindrom, geriye doğru okunduğunda ve ileriye doğru okunduğunda aynı değere sahip olan doğal sayıdır. Örneğin, 123 454 321 sayısı palindromdur).
6. n doğal sayısının asal sayı olup olmadığı kontrol edilsin. (Asal sayılar sadece 1'le ve kendilerinle bölünebilen sayılardır).
7. 10.000'e kadar tüm dost sayılar bulunsun.(Birinin bölenlerinin toplamı diğerine, diğerinin bölenlerinin toplamı birinciye eşit olan iki sayı dosttur. Bölenlere sayının kendisi ait değildir).
8. Negatif olmayan onlu tam sayı n, ikili sayıya dönüştürülsün.
9. Negatif olmayan ikili tam sayı b, onlu sayıya dönüştürülsün.
10. e = 2.718281 sayısının değeri, $e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots$ ifadesini kullanarak, belirli bir ε ("epsilon" olarak okunur) doğrulukla hesaplansın. ε doğruluğu son toplanan (toplama eklenmeyen) $\frac{1}{k!} < \varepsilon$ ulaştığında elde edilir. ε sayısı çok küçük bir değerdir, örneğin; $\varepsilon = 0{,}001$, $\varepsilon = 0{,}0000000001$, vb.
11. π = 3.14159 sayısının değeri, $\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \ldots$ fadesini kullanarak, $\frac{4}{2k+1} < \varepsilon$ olana kadar, verilen ε doğrulukla hesaplansın.

12. Aşağıdaki formlar yazdırılsın:

```
@@@@@@@@@@
@@@@@@@@@
@@@@@@@
@@@@@@
@@@@@
@@@@
@@@
@@
@
```

```
&&&&&&&&&&
&&&&&&&&
&&&&&&
&&&&
&&
&&
&&&&
&&&&&&
&&&&&&&&
&&&&&&&&&&
```

```
$
###
$$$$$
#######
$$$$$$$$$
###########
$$$$$$$$$
#######
$$$$$
###
$
```

Döngü Sonunda Çıkışlı Tekrarlama Algoritmik Kontrol Yapısı yürütür-iken ve do-while Kontrol Komutu

Önceki alt başlıklardan ilk 10 doğal sayının toplamını bulma ödevinde sadece iki işlemin tekrarlandığını söyledik:

- Toplama sayi eklemek.
- Sayıyı 1 için arttırmak.

Bu iki işlemin olduğu döngülerin, sayi değişkeninin 10'dan küçük veya eşit **iken yürütüldüğünü** söyleyebiliriz. Bu yüzden, bu algoritmik kontrol yapısına **yürütür-iken** denir. Bunu ***Şekil 2.3.30'***daki gibi gösterebiliriz.

```
toplam ← 0;
sayi ← 1;
yürütür
        – sayiyi toplama ekle;
        – sayıyı 1 için arttır;
iken sayi ≤ 10;
bitiş_ iken {sayi ≤ 10}
yazdır toplam;
```

Şekil 2.3.30

Bu algoritmik kontrol yapısında, her döngünün yürütülmesinden sonra, bir sonraki döngünün yürütülüp yürütülmeyeceği koşulu sonunda kontrol ediliyor. Bu, döngünün en az bir kez gerçekleştirileceği anlamına gelir.

yürütür-iken algoritmik kontrol yapısının metinsel yazılışı, ***şekil 2.3.31***'de ve grafiksel gösterimi ***şekil 2.3.32***'de verilmiştir.

```
yürütür
        adim A;
        adim B;
        ...
        adim M;
iken koşul;
bitiş_yürütür {koşul}
```

Şekil 2.3.31

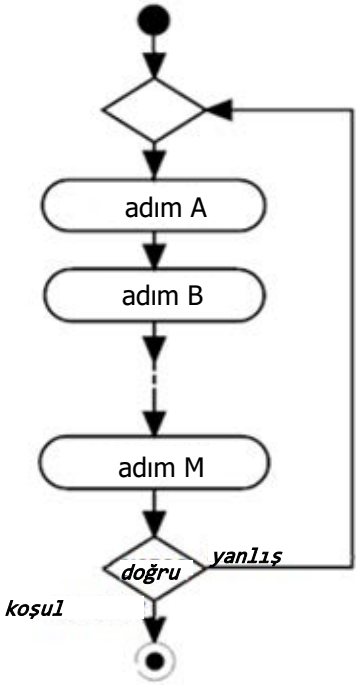
yürütür-iken algoritmik kontrol yapısını gerçekleştirmek için C++'da do-while kontrol komutu kullanılır, ***Şekil 2.3.33***.

do-while komutu, bir grup komutun en az bir kez yürütülmesi gerektiğinde ve bunların yeniden yürütülmesi bazı koşula bağlı olduğunda kullanılır.

Döngülerin yürütülmesi, *kosul* geçerli olana kadar tekrarlanır. *kosul* geçerli olmadığında tekrarlama sona eriyor ve işlem sonraki komutla devam ediyor.

```
do{
        komut A;
        komut B;
        ...
        komut T;
}while(kosul);
```

Şekil 2.3.33



Şekil 2.3.32

Örnekler

Örnek 2.3.11

yürütür-iken algoritmik kontrol yapısıyla ilk 10 doğal sayının toplamı ödevinin metinsel olarak ifade edilen algoritması ***şekil 2.3.34***'te verilmiştir, do-while kontrol komutu ile yazılan program ise ***şekil 2.3.35***'te verilmiştir.

```
algoritma Toplam
başlangıç
        toplam ← 0;
        sayi ← 1;
        yürütür
                toplam ← toplam + sayi;
                sayi ← sayi + 1;
        iken sayi ≤ 10;
        bitiş_yürütür {sayi ≤ 10}
        yazdir toplam;
bitiş {Toplam}
```

Şekil 2.3.34

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // İlk 10 dogal sayının toplamı (do-while ile)
6
7       int sayi, toplam;
8       toplam  = 0;
9       sayi = 1;
10      do {
11          toplam = toplam + sayi;
12          sayi = sayi +1;
13      } while( sayi <= 10 );
14      cout << "İlk 10 dogal sayinin toplami: " << toplam << endl;
15
16      cout << endl;
17      system( "Color 17" );
18      system( "pause" );
19      return 0;
20  }
```

Şekil 2.3.35

Programı yürüttükten sonra bir çıktı şudur:

```
Ilk 10 dogal sayinin toplami:55

Press any key to continue . . .
```

Örnek 2.3.12

Pazarda ne kadar para harcandığı hesaplansın.

Açıklama: **Örnek 2.3.10**'daki programın do-while komutuyla yazılması daha doğaldır çünkü en az bir ürün satın almışsak da, yani en az bir hesabımız varsa, pazar harcamalarını hesaplayacağız. Buna göre, en az bir döngü gerçekleştirilmelidir.

Program ***şekil 2.3.36***'da verilmiştir.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       //   Pazar (do-while ile)
6
7       double  hesapTutari, toplamHarcama=0;
8       cout << " Bitis icin, hesap tutari olarak -99 girin" << endl;
9       cout << " \nBirinci hesabin tutarini girin: " ;
10      cin >> hesapTutari ;
11      do {
12          toplamHarcama += hesapTutari;
13          cout << "Siradaki hesabin tutarini girin: ";
14          cin >> hesapTutari
15      } while( hesapTutari ! = -99 );
16      cout << "\nToplam olarak" <<toplamHarcama << " denar harcanmistir." << endl;
17
18      cout << endl;
19      system( "Color 17" );
20      system( "pause" );
21      return 0;
22  }
```

Şekil 2.3.36

Programın yürütülmesiyle bir çıktısı şu olacaktır, sonra şunu elde ediyoruz:

```
Bitis icin, hesap tutari olarak -99 girin
Birinci hesabin tutarini girin: 12
Siradaki hesabin tutarini girin: 34.5
Siradaki hesabin tutarini girin: 56.789
Siradaki hesabin tutarini girin: -99

Toplam olarak 103.289 denar harcanmistir
Press any key to continue . . .
```

Alıştırmalar

Alıştırma 2.3.8

Aşağıdaki program bölümünün ne yaptığı belirlensin:

```
int x = 0, i = 0;
do {
	x++;
	i += ++x;
} while(x < 10);
```

Alıştırma 2.3.9

Aşağıdaki program bölümü yürütüldükten sonra neyin yazdırılacağı belirlensin:

```
int i = 0;
do {
	i += i++;
	cout << i << ", ";
} while(i < 10);
```

Alıştırma 2.3.10

Aşağıdaki program bölümü yürütüldükten sonra neyin yazdırılacağı belirlensin:

```
int i = 12345;
do {
	cout << i % 10 << '\n';
	i /= 10;
} while(i > 0);
```

Çözülmüş Ödevler

Ödev 2.3.6

Bir doğal sayının kaç basamaklı olduğu belirlensin.

Açıklama: Bu ödev, **Ödev 2.3.3** ile aynıdır. Ancak do-while kontrol komutu ile ifade etmek daha doğaldır çünkü sayı tek haneli olsa bile kalanın 0 olup olmadığını anlamak için en az 10 ile bir bölme işlemi yapmalıyız.

do-while komutu ile yazılan program ***Şekil 2.3.37***'de verilmiştir.

Programın yürütülmesiyle bir çıktı şudur:

```
Dogal sayi girin, n=1234567890
1234567890 dogal sayisinin 10 basamagi vardir
Press any key to continue . . .
```

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

int main() {
    // Dogal sayinin basamak sayisi (do-while ile)

    int dogalSayi, basamakSayisi, bolum;
    cout << " dogal sayi girin, n= " ;
    cin >> dogalSayi;
    cout << tamSayi << "tamsayisinin"; <<
    basamakSayisi = 0;
    do {
        bolum=dogalSayi / 10;
        basamakSayisi++;
        dogalSayi = bolum;
    } while( bolum != 0 );
    cout << basamakSayisi << " basamagi vardir." << endl;

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

Şekil 2.3.37

Ödev 2.3.7

Euler[5] sayısının (Leonhard Euler, İsviçreli matematikçi), değeri e = 2.718281... $e \approx 1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}+\ldots+\frac{1}{n!}$ ifadesi yardımıyla ε doğrulukla hesaplansın.

Açıklama: 1 · 2 · 3 · ... · n çarpımının, n! ile gösterilirdiğini ve "n faktöriyel" olarak adlandırıldığını söylemiştik. Varsayılan olarak, 0! = 1 ve 1! = 1 tanımlanıyor ve 1'den büyük sayılar için faktöriyel n! = (n – 1)! · n formülüne göre hesaplanır.

Böylece, 2! = 1 · 2, 3! = 2! · 3 = 1 · 2 · 3 vb.

Faktöriyellerin bu özelliğini, ödevi çözmemis icin kullanabiliriz. Şöyle ki, $\frac{1}{2!}=\frac{1}{1!}\cdot\frac{1}{2}$, $\frac{1}{3!}=\frac{1}{2!}\cdot\frac{1}{3}$, $\frac{1}{4!}=\frac{1}{3!}\cdot\frac{1}{4}$ vb.yani $\frac{1}{n!}=\frac{1}{(n-1)!}\cdot\frac{1}{n}$.

Bu, toplamda her bir sonraki toplananın (örneğin, k'inci) bir önceki ((k-1)'inci) toplananını $\frac{1}{k}$ ile çarptığımızda elde edildiği anlamına gelir.

[5] Bazıları Napier sayısı (John Napier, 17. yüzyıl, İskoç matematikçi) olarak adlandırıyor.

ε kesinliğini, toplama eklenecek toplananın belirtilen kesinlikten ($\frac{1}{n!} < \varepsilon$) küçük olacağı durumda döngülerin tekrarlamanın sonu için kullanacağız. Aynı zamanda, belirlenen doğruluğa ulaşmak için kaç toplananın toplanması gerektiği önceden bilinmemektedir.

Program ***şekil 2.3.38***'de verilmiştir.

Programdaki kesin değerden, $\varepsilon = 10^{-50}$ doğruluk için 15 doğru ondalık basamaklı değerin elde edildiği görülebilir.

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

int main() {
    // e = 2.7182818284590452353602874713526624977572470936999...sayisinin
    // epsilon dogrulukla hesaplanmasi (do-while ile)

    int k;
    double eSayi, epsilon, toplanan;
    cout << " Dogruluk girin, epsilon = ";
    cin >>  epsilon ;
    eSayi  = 0;
    k = 0;
    toplanan = 1;
    do {
        eSayi += toplanan;
        ++k;
        toplanan /= k;
    } while(toplanan >= epsilon);
    cout << fixed << setprecision( 50 );
    cout <<" e=71 sayisi " << epsilon <<\n dogrulukla " <<eSayi<< " 'dir " << endl;

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

Şekil 2.3.38

Programın yürütülmesinde bir çıktı şudur:

```
Dogruluk girin, epsilon =1e-50
e=71 sayisi 0.00000000000000000000000000000000000000000000000001
dogrulukla 2.71828184590455348848081484902650117874145507812 5'dir
Press any key to continue . . .
```

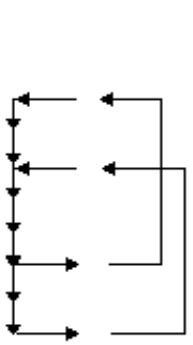
Alıştırma Ödevleri

Döngünün Başında Çıkışlı Tekrarlama Algoritmik Kontrol Yapısı ve while Kontrol Komutu alt başlığındaki **Alıştırma Ödevlerinden** hangilerinin do-while komutu kullanılarak yazılabileceği kontrol edilsin.

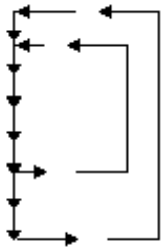
Tekrarlama Kontrol Komutlarının İç İçe Yerleştirilmesi

Bazı programlarda birden fazla tekrarlama kontrol komutu kullanmak gereklidir. Bu arada, bir kontrol komutu diğerinde bulunabilir. Buna **iç içe yerleştirme** (İng. nesting) denir.

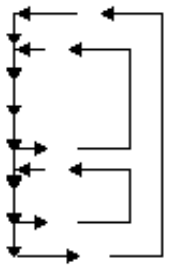
Tekrarlama kontrol komutlarını iç içe yerleştirirken, bir kontrol komutunun tüm komutları diğerinin içinde olmalıdır, yani ***şekil 2.3.39***'daki gibi kontrol komutlarının kesişmesi (örtüşmesi) olmamalıdır. ***Şekil 2.3.40*** a, b ve **c'de**, kontrol komutlarını iç içe yerleşmenin izin verilen şekilleri verilmiştir.



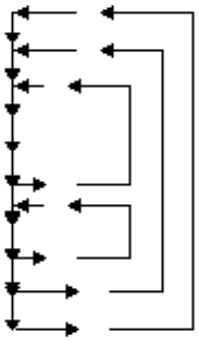
Şekil 2.3.39



a



b



c

Şekil. 2.3.40

Örnekler

Örnek 2.3.13
1'den 10'a kadar sayıların 10'a kadar çarpma tablosu için program yazılsın.

Program ***şekil 2.3.41'***de verilmiştir.

```
1   #include <iostream>
2   #include <iomanip>
3   using namespace std;
4
5   int main() {
6       //1'den 10'a kadar sayilarin 10'a kadar carpma tablosu. (for ile)
7
8       for( int i = 1; i <= 10; i++ ) {
9           cout<< i << " ile carpma " << endl << endl;
10          for( int j = 1; j <= 10; j++ ) {
11              int icarpij = i * j;
12              cout << setw( 2 ) << i << " x " << setw( 2 ) << j << " = "
13                  << setw( 3 ) << icarpij << endl;
14          }
15          cout << endl;
16      }
17
18      cout << endl;
19      system( "Color 17" );
20      system( "pause" );
21      return 0;
22  }
```

Şekil 2.3.41

```
1 ile carpma

 1 x  1 =   1
 1 x  2 =   2
 1 x  3 =   3
 1 x  4 =   4
 1 x  5 =   5
 1 x  6 =   6
 1 x  7 =   7
 1 x  8 =   8
 1 x  9 =   9
 1 x 10 =  10

2 ile carpma

 2 x  1 =   2
 2 x  2 =   4
 2 x  3 =   6
 2 x  4 =   8
 2 x  5 =  10
 2 x  6 =  12
 2 x  7 =  14
 2 x  8 =  16
```

...

Örnek 2.3.14

Rakam küplerinin toplamı o sayıya eşit olan n'den küçük tüm doğal sayılar bulunsun.

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

int main() {
    // Rakamlarinin kuplerine esit dogal sayilar
    system( "Color 17" );
    long long n, rakam, sayi, toplam, sayiN;
    bool varYok = false;
    cout <<" Hangi sayiya kadar, n = " ; cin >> n;
    cout <<"\n'<< n << "'den daha kucuk ve rakamlarinin kuplerinin" ;
    cout << "\n toplamina esit dogal sayilar \n sunlardır: " ;
    for( sayi = 1; sayi <= n; sayi ++ ) {
        toplam = 0;
        sayiN = sayi;
        do {
            rakam = sayiN % 10;
          toplam += ( int ) pow( rakam, 3 );
            sayiN /= 10;
        } while( sayiN > 0 );
        if( toplam == sayi ) {
            cout << sayi << ", ";
            varYok = true;
        }
    }
    if( !varYok ) {
        cout << "\n' << n << " 'e kadar oyle sayilar yoktur." << endl;
    }

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

Şekil 2.3.42

Programı yürüttükten sonra bir çıktı şudur:

```
Hangi sayiya kadar,  n = 1000

 1000'den daha kucuk ve rakamlarının kup
 toplamina esit dogal sayilar
 sunlardır: 1, 153, 370, 407,

Press any key to continue . . .
```

Bilgiyi kontrol etme soruları

1. Tekrarlama algoritmik kontrol yapısı ne zaman kullanılır?
2. Tekrarlama algoritmik kontrol yapısındaki adımların bir kez yürütülmesine ne ad verilir?
3. Tekrarlama algoritmik kontrol yapıları, tekrarlamanın çıkış şekline göre nasıl ayrılır?
4. Hangi döngüleri sayımlı tekrarlama algoritmik kontrol yapılarını biliyorsunuz?
5. **için-arttır-kadar** algoritmik kontrol yapısını metinsel gösterin.
6. **için-arttır-kadar** algoritmik kontrol yapısındaki sayac, baslangic ve bitis değişkenleri hangi türdendir?
7. **için-arttır-kadar** algoritmik kontrol yapısında, baslangic değişkeni bitis değişkeninden daha büyük değere sahipse, kaç döngü yürütülecek?
8. Döngüleri sayımlı tekrarlama algoritmik kontrol yapıları C++'da hangi kontrol komutu ile gerçekleştirilir?
9. for kontrol komutunun hangi kısımları vardır? Her kısımda ne yapılıyor?
10. Arttırma ve azaltma operatörlerinin işlemi nedir?
11. Değişkenin ön ek ve son ek arttırması ne anlama gelir? Örnek verin.
12. Arttırma ve azaltma operatörleri hangi veri kümeleri için kullanılabilir?
13. for kontrol komutunda sayaç döngünün gövdesinde kullanılabilir mi?
14. for kontrol komutunda kaç tane değişken başlatılabilir?
15. for kontrol komutundaki sayaç, döngü dışında başlatılabilir mi?
16. for kontrol komutunda koşul kısmı boş olabilir mi?
17. Aşağıdaki program bölümü ne yapıyor?

```
for(x = 1, i = 0; x <= 10; x++)
   i += x;
```

18. Aşağıdaki program bölümü yürütüldüğünde ne yazdırılacak?

```
for(int i = 0, j = 0; i < 5; i++, j--)
   cout << i + j << endl;
```

19. Aşağıdaki program bölümü yürütüldüğünde ne yazdırılacak?

```
for(int i = 1; i <= 5; i++)
   for(int j = 1; j <= 5; j++)
          for(int k = 1; k <= 5; k++)
                 cout << i + j + k << endl;
```

20. Aşağıdaki program bölümü yürütüldüğünde ne yazdırılacak?

```
for(int i = 1; i <= 5; i++) {
   for(int j = 1; j <= i; j++) {
          for(int k = 1; k <= j; k++)
                 cout << '?';
          cout << endl;
   }
   cout << endl;
}
```

21. **iken-yürütür** algoritmik kontrol yapısını metinsel ve grafiksel olarak gösterin.
22. **iken-yürütür** algoritmik kontrol yapısı C++'da hangi kontrol komutuyla gerçekleştirilir?
23. While kontrol komutu ile hiçbir döngüyü yürütmemek mümkün mü?
24. While kontrol komutunun sözdizimini yazın.
25. C++'daki hangi kontrol komutu ile, **yürütür-iken** algoritmik kontrol yapısı gerçekleştirilir?
26. do-while kontrol komutunun sözdizimini yazın ve nasıl çalıştığını açıklayın.
27. While ve do-while kontrol komutları arasındaki fark nedir?
28. Aşağıdaki program bölümünün yürütülmesi sonucunda ne yazdırılacaktır?

```
int i = 0;
while(i < 5) {
   i = i + 2;
   cout << i << endl;
}
```

29. Aşağıdaki program bölümü ne yapar?

```
int i, j;
i = 0;
while(i < 8) {
   j = 0;
   while(j < 8) {
          cout << setw(5) << (i + j) % 8;
          ++j;
   }
   cout << endl;
   ++i;
}
```

30. Aşağıdaki program bölümü ne yapar?

```
int x, i;
x = 0; i = 0;
do {
   x++;
   i += x;
} while(x < 10);
```

31. Aşağıdaki program bölümü yürütüldüğünde ne yazdırılacak?

```
int i = 0;
do {
   ++ ++i;
   cout << i << endl;
} while(i < 10);
```

32. Aşağıdaki program bölümü yürütüldüğünde ne yazdırılacak?

```
for(int i = 1; i <= 5; i++)
   switch(i) {
   case 2: cout << "gunler, "; break;
   case 4: cout << "siniz? "; break;
   case 3: cout << "nasil"; break;
   case 1: cout << "İyi"; break;
   }
```

33. Kontrol komutlarının iç içe yerleştirilmesi ne anlama gelir? Tekrarlama kontrol komutlarının izin verilen iç içe yerleştirilmesine ilişkin diyagramlar çizin.

Ödevler

1. Her satırda k yıldız, k = 1, 2... n olmak üzere aşağıdaki n satırdan oluşan üçgen oluşturulsun .

```
 *
* *
* * *
* * * *
...
* * * * * * * * *
```

2. Aşağıdaki üçgen oluşturulsun:

```
1
2 4
3 6 9
4 8 12 16
...
n ...
```

3. Aşağıdaki çizim yazdırılsın:

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

4. n sayısından küçük Fibonacci sayıları oluşturulsun: 1, 1, 2, 3, 5, 8, 13, 21, 34, vb.
5. Doğal sayı n girilsin ve basamak basamak sözcüklerle yazdırılsın. Örneğin 231 sayısı girilirse iki-uc-bir yazdırılsın
6. Doğal sayı n girilsin ve bu sayı kelimelerle yazdırılsın. Örneğin, 12 345 sayısı girilirse "on iki bin uc yuz kirk bes" yazdırılsın.

ε

7. $\frac{1}{1\cdot 2}+\frac{1}{2\cdot 3}+\frac{1}{3\cdot 4}+\cdots$ toplamı ε doğrulukla hesaplansın, $\left(\frac{1}{k(k+1)}<\varepsilon\right)$.
8. Klavye üzerinden sayılar girilsin ve girilen sayılardan en büyüğü ve en küçüğü bulunsun. Girişin bitişi için 999 999 sayısı girilsin.
9. $\frac{a}{b}$ kesiri kısaltılsın, a ve b doğal sayılardır.
10. Rakamlarının toplamıyla bölünebilen n'den küçük tüm sayılar yazdırılsın.
11. n doğal sayısının en büyük tek sayı böleni bulunsun.
12. Birler basamağının rakamlarını sayarak n doğal sayısının k'inci basamağı atlatılsın.

2.4 Atlama İçin Algoritmik Kontrol Yapıları ve Atlama İçin Kontrol Komutları

Programlama dillerinde atlama için üç tür algoritmik kontrol yapısı vardır, onlar da:

- *Döngünün sonuna atlama – **devam etme.***
- *Kontrol yapısının sonuna atlama – **kesinti.***
- *Algoritmanın sonuna atlama – **çıkış.***
- *Herhangi bir yere atlama - **atlayış.***

C++'daki bu algoritmik kontrol yapılarına karşılık gelen kontrol komutları şunlardır:

- **devam etme** için continue,
- **kesinti** için break,
- **çıkış** için exit(),
- **atlayış** için goto.

Komutları örneklerle açıklayacağız.

continue Kontrol Komutu

Bu kontrol komutu, döngünün sonuna koşulsuz atlama için kullanılır. Bu, döngünün bir komutunda, sona ermeden önce, belirli bir koşul karşılandığında ve döngünün diğer komutların yürütmesi gerekli olmadığında yapılır. Bu komutla mevcut döngüden çıkılır ve eylem bir sonraki döngüyle *devam eder.*

continue kontrol komutu, while, do-while ve for tekrarlama kontrol komutlarında mevcut döngüden çıkmak için kullanılır.

Örnek 2.4.1

Girilen sayıların karekökü hesaplansın. Negatif sayı girilirse, onun karekökü hesaplanmasın, bir sonraki girilen sayı ile devam edilsin. Girdi sonu için 999 999 sayısı girilsin.

Program ***şekil 2.4.1'***de verilmiştir. Sayı negatifse, negatif sayının kökü olmadığı mesajı yazdırılır ve bir sonraki döngüyle, yani bir sonraki sayının okunmasıyla devam edilir.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // Gercek sayinin karekoku (continue ile)
6
7       double x;
8       cout <<" Bitirmek icin sayi olarak 999999 girin."  << endl;
9       do {
10          cout <<"Gercek sayi girin, x = ";
11          cin >> x;
12          if( x < 0 ) {
13              cout << " Sayi negatiftir ve karekoku yoktur.          \n";
14              continue;
15          }
16          cout << x << " 'in karekoku: " <<sqrt( x ) << endl;
17
18      } while( x != 999999 );
19
20      cout << endl;
21      system( "Color 17" );
22      system( "pause" );
23      return 0;
24  }
```

Şekil 2.4.1

Programın yürütülmesiyle çıktı şudur:

```
Bitirmek icin sayi olarak 999999 girin.
Gercek sayi girin, x= 1
1'in karekoku: 1
Gercek sayi girin, x= 2
2'in karekoku: 1.41421
Gercek sayi girin, x= -3
Sayi negatiftir ve karekoku yoktur.
Gercek sayi girin, x=1234567890
1.234567e+09'in karekoku: 35136.4
Gercek sayi girin, x=999999
999999'in karekoku: 999999
Press any key to continue . . .
```

break Kontrol Komutu

break kontrol komutu, while, do-while ve for6 tekrarlama kontrol komutlarında, tekrarlama komutunun sonuna koşulsuz atlama ve tekrarlamaları durdurmak için kullanılır. Bu, döngünün bir komutunda, bitmeden önce, belirli bir koşul karşılandığında ve döngülerin tekrarlanmasının devam edilmesi gerekli olmadığında yapılır, yani tekralama *durdurulur*. İşlem bir sonraki komutla devam eder.

break kontrol komutu while, do-while ve for iç içe yerleştirilmiş kontrol komutlarının birinde bulunuyorsa, o zaman sadece gövdesinde bulunduğu iç içe yerleştirilmiş kontrol komutundan çıkar.

Bu kontrol komutu, genellikle sonsuz sayıda döngü içeren kontrol komutlarından çıkmak için kullanılır.

Not: Sonsuz tekrarlama, diğer komutlarla da gerçekleştirilebilir.

Örnek 2.4.2

Aşağıdaki örnekte, tekrarlama sadece bilgisayar tarafından (rastgele oluşturulmuş) düşünülmüş sayıyı tahmin edersek durdurulacaktır, aksi takdirde sonsuz olacaktır, ***şekil 2.4.2.***

Programda, **Ödev 2.3.5**'te açıklanan rastgele bir sayı üretmek için rand() fonskiyonu kullanılır. rand() ile farklı rastgele sayı dizisi oluşturmak için, argümanı klavye aracılığıyla girilen bir sayı olan srand() fonksiyonu kullanılır. srand() fonksiyonu, **2.5 Fonksiyonlar** alt bölümünün **Rastgele Sayılar Üretme Fonskiyonları** alt başlığında açıklanmaktadır.

Programın yürütülmesiyle bir çıktı şudur:

```
Dogal sayi girin: 12345
1 ile 10 arasinda dusundugum sayiyi bulun: 1
1 ile 10 arasinda dusundugum sayiyi bulun: 2
1 ile 10 arasinda dusundugum sayiyi bulun: 3
1 ile 10 arasinda dusundugum sayiyi bulun: 4
1 ile 10 arasinda dusundugum sayiyi bulun: 5
Aferin! dusundugum 5 sayisini 5'nci denemede buldunuz.
Press any key to continue . . .
```

[6] break kontrol komutu, switch kontrol komutunda da kullanılır, ancak burada switch komutunun tanımına göre zorunludur.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {     // 1 ile 10 arasinda dusunulen sayinin bulunmasi (break ile)
5
6       int   sayi, dusunulenSayi;
7       cout << `Dogal sayi girin: "; cin >> sayi ;
8       srand(sayi );
9       dusunulenSayi = 1 + rand() % 10;
10      for( int  denemeler = 1; ; denemeler ++ ) {
11          cout << "1 ile 10 arasinda dusundugum sayiyi bulun: " ;
12          cin >> sayi ;
13          if( sayi  == dusunulenSayi ) {
14              cout << " Aferin! dusundugum "  << dusunulenSayı << "sayisini "
15                   << denemeler  <<  "nci denemede buldunuz." << endl;
16              break;
17          }
18      }
19
20      cout << endl;
21      system( "Color 17" );
22      system( "pause" );
23      return 0;
24  }
```

Şekil 2.4.2

exit () Kontrol Komutu

Bazen programın zorunlu olarak sona erdirilmesi gerekir, bu da programın sonuna atlama ile yapılır. Bu, programın herhangi bir yerinde, yürütülmesi programın kesintiye uğramasına ve düzensiz bir şekilde sonlandırılmasına yol açabilecek bir işlem beklendiğinde gereklidir. Örneğin, bazı hesaplamalar sırasında karekökün altında olan büyüklüğün negatif değer alma olasılığı varsa, hata meydana geleceği ve program duracağı için karekökün hesaplanmasına izin verilmemelidir ve programın sonuna atlayarak düzenli olarak çıkılmalıdır.

Bu kontrol komutu sıkça var olmayan bir dosyadan okuma gibi yanlış sonuçlara veya uygulamanın durdurulmasına yol açabilecek hataları algılamak için kullanılır.

Programın sonuna atlamak için, C++'da EXIT_SUCCESS ve EXIT_FAILURE sabitleri olmak üzere bir argümana sahip olan exit() fonksiyonu kullanılır. Fonksiyon birinci argüman ile çağrılıyorsa programın başarılı bir şekilde sonlandığı, ikinci argüman ile çağrılırsa programın başarısızlıkla sonlandığı yani bir hata oluştuğu anlamına gelir.

exit(EXIT_SUCCESS) – programın normal (başarılı) tamamlanması için veya

exit(EXIT_FAILURE) – programın düzensiz (başarısız) tamamlanması için, yani EXIT_FAILURE sabiti yürütme sırasında hata oluştuğunu gösterir.

Örnek 2.4.3
Ogrenciler isimli dosyanın olup olmadığı kontrol edilsin.

Açıklama: Programda iki dizeyi karşılaştırmak için compare() fonksiyonu kullanılır. Bu fonksiyon, **3.5 Dizeler** alt bölümünde açıklanmıştır.

```
1   #include <iostream>
2   #include <string>
3    using namespace std;
4
5   int main() {
6       //  Programdan cikis (exit() ile)
7       string dosyaAdi;
8       while( true ) {
9           cout << "Dosyanin adini girin: ";
10          cin >>  dosyaAdi;
11          if( dosyaAdi .compare("ogrenciler") == 0 ) {
12              cout << dosyaAdi << "adli dosya vardır "
13              cout << " , Program devam edebilir. " << endl;
14              system( "pause" );
15              exit( EXIT_SUCCESS );
16          }
17          else {
18              cout << dosyaAdi << "adli dosya yoktur " ;
19              cout << ",program sona eriyor. " << endl;
20              system( "pause" );
21              exit( EXIT_FAILURE );
22          }
23      }
24
25      cout << endl;
26      system( "Color 17" );
27      system( "pause" );
28      return 0;
29  }
```

Şekil 2,.4.3

Programın çıktı örneği şudur:

```
Dosyanin adini girin: ogrenciler
ogrenciler adli dosya yoktur, program sona eriyor.
Press any key to continue . . .
```

goto Kontrol Komutu

İstenilen yerde atlama kontrol komutu goto, programda herhangi bir yerden herhangi bir yere atlamak için kullanılır. Bu arada, atlanacak yer de işaretlenmiş olmalıdır. İşaret sayısal veya metinsel olabilir. Bu işaret tanımlayıcı olarak kaydedilir ve sonunda iki nokta konulmalıdır.

İşaret, programın herhangi bir komutundan önce gelebilir.

Bu kontrol komutu, yapılandırılmamış programlama için kullanılıyormuş. Dezavantajı programın işleyiş akımını değiştirmesidir ve bu nedenle analiz edilmesi ve anlaşılması zordur. Bu yüzden, bu kontrol komutu yapısal programlamada kullanılmıyor..

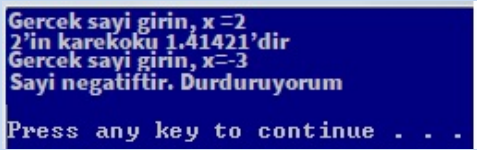
Örnek 2.4.4

goto kontrol komutu, ***Örnek 2.4.1***'de bir sayının karekökünü hesaplamak için programla gösterilmektedir. Program bir gerçek sayının karekökünü hesaplıyor ve negatif bir sayı girilirse yürütmeyi durduyor, ***şekil 2.4.4.***

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       //  Sayinin karekoku (goto ile)
6
7       double x;
8       while( true ) {
9           cout << " Gercek sayi girin, x = ";
10          cin >> x;
11          if( x < 0 )
12              goto son;
13          cout << x << " 'in karekoku: "  << sqrt( x ) << endl;
14      }
15  son:
16      cout << "Sayi negatiftir. Durduruyorum. " << endl;
17
18      cout << endl;
19      system( "Color 17" );
20      system( "pause" );
21      return 0;
22  }
```

Şekil 2.4.4

Programın yürütülmesiyle bir çıktı şudur:



Bilgiyi kontrol etme soruları

1. Hangi atlama algoritmik kontrol yapılarını biliyorsunuz?
2. Atlama algoritmik kontrol yapıları, C++'da hangi kontrol komutlar ile geçekleştirilir?
3. continue ve break kontrol komutları arasındaki fark nedir?
4. Hangi kontrol komutla (fonkisyonla) programın sonuna atlanır ve argümanı nasıl olabilir?
5. goto kontrol komutundan neden kaçınması gerektiğini açıklayın?

2.5 Fonksiyonlar

Kütüphane fonksiyonları

Matematikten, **fonksiyonların** (İng.functions) sadece **bir çıktı sonucuna** sahip oldukları bilinmektedir. Örneğin:

$y = 5$, $y = 4x - 2$, $y = 2x^2 - 3x + 1$ vb.

y değişkeninin değerinin x argümanına bağlı olduğunu söylüyoruz ve bu y = f (x) şeklinde yazılıyor.

Matematiksel fonksiyonları yanı sıra, çeşitli alanlarda kullanılan birçok başka fonksiyonlar da vardır. Programcıların işini kolaylaştırmak amacıyla,aynı fonksiyonları, örneğin x^n, $\sqrt{x}$, e^x ve diğerleri için her zaman **program yazmamaları** için, her programlama dilinde daha sık kullanılan fonksiyonlar için ayrı programlar yazılmıştır.Bu programlar, dilin program kütüphanelerinde saklıdır. Programcılar, programın başına #include yönergesi ile kütüphaneyi önceden dahil ederek bu programları kullanabilirler. Biz de, önceki tüm programlara <iostream> kütüphanesini dahil etmiştik çünkü veri girdisi ve çıktısı için bu kütühanedeki programları kullandık.

Kütüphanelerdeki programlar pratikte fonksiyonlar olarak adlandırılır.

C++ dilinin **C++ standart kütüphanesi (**İng. C++ Standard Library) vardır.

C++ kütüphanelerinin fonksiyonlarına **yerleşik fonksiyonlar** (İng. build-in functions) da denir. İsimleri ayrılmış kelimeler olduğu için isimler küçük harflerle yazılır. Örneğin, pow(), sqrt, rand(), vb.

C++ *standart kütüphanesi,* **başlık dosyaları** (İng.header files) adı verilen dosyalarda saklanan bir çok kütüphane içerir, bu dosyalar da benzer fonksiyon grupları içermektedir:

<iostream>	Girdi ve çıktı fonksiyonları.
<iomanip>	Biçimlendirme fonksiyonları.
<cstdlib>	Genel kütüphane: program kontrolü, rastgele sayılar, sıralama, isteme vb.
<cmath>	Genel matematiksel fonksiyonlar.
<string>	Dizelerle çalışma fonksiyonları.
<random>	Rastgele sayılar üretme fonksiyonları.

C++ kütüphanesinden bir fonksiyonu programda kullanmak için şimdiye kadar yaptığımız gibi #include yönergesi ile kütüphane programın başında dahil edilmelidir. Bu arada, kütüphane köşeli parantez içine alınır.

Devamda, daha sık kullanılan bazı kütüphane fonksiyonları inceleyeceğiz.

Matematiksel fonksiyonlar

Matematiksel fonksiyonları hakkında kısa bir tekrarlama yapalım.

Örneğin, f(x) = x fonksiyonu doğrusal fonksiyondur, $f(x) = x^2$ karesel fonksiyondur, $f(x) = \sqrt{x}$ x'in karekökünü bulmak için bir fonksiyondur, vb. f(x), bir değerin hesaplandığı kuralı belirtir. Örneğin, bir sayının karesini hesaplamanın kuralı, onu kendisi ile çarpmaktır ve $f(x) = x \cdot x$ veya kısaca x^2 şeklinde yazılır. x = 5 için elde edilecek değer 25'tir ve başka bir sayı ile, örneğin y ile gösterilir, yani y = 25. Her x sayısı için (**fonksiyonun argümanı** (bağımsız değişken) olarak adlandırılır), y değeri elde edilir. Bu, y = f(x) veya $y = x^2$ şeklinde de yazılır.

Programlamada, fonksiyonun argümanı ve değeri, belirli bir türden değişkenlerdir. Örneğin:

```
int x, y;
x = 5;
y = x * x;
veya
y = pow(x, 2);       // Üsse yükseltme fonksiyonu x²
```

C++'da en yaygın olarak kullanılan matematiksel fonksiyonlar şunlardır:

pow(x, y) – Üstel fonksiyon x^y.
exp(x) – Üstel fonksiyon e^x, e = 2.7182[7]'dir.
sqrt(x) – x'in karekökü $\sqrt{x}$
cbrt(x) – x'in küp kökü $\sqrt[3]{x}$
log(x) – e = 2,7182 tabanlı logaritmik fonksiyon, ln(x).
log10(x) – 10 tabanlı logaritmik fonksiyon log(x).
fabs(x) – x'in mutlak deÄŸeri,|x|.
ceil(x) – x'i x'ten küçük olmayan en küçük gerçek sayıya yuvarlama.
floor(x) – x'i x'ten büyük olmayan en büyük gerçek sayıya yuvarlama.
trunc(x) – Sadece ondalık noktanın solundaki gerçek kısmı kesme.
round(x) – x'e en yakın gerçek sayıya yuvarlama (ondalık basamaksız).
fmod(x, y) – gerçek sayı olarak x / y bölümünün kalanı.
modf(x, &integralen) – x ondalık sayısını bir integral bölüme ve ondalık bölüme ayırma. (& işareti hakkında daha sonra bahsedeceğiz).
sin(x) – sin(x) trigonometrik fonksiyonu.
cos(x) – cos(x) trigonometrik fonksiyonu.
tan(x) – tg(x) trigonometrik fonksiyonu.

Bu fonksiyonlar, kullanıcıya çok sayıda matematiksel hesaplamalar yapmasını sağlamaktadır. Fonksiyonlar, fonksiyonun adı ve küçük parantez içinde argümanlar listesi yazılarak kullanılır.

Önceki örnekte şu fonksiyonu yazmıştık:

```
y = pow(x, 2);
```

Komutun sağ tarafı, pow adında, x ve 2 argümanlı **fonskiyon çağrısı** (İng. function call veya function invocation) belirtir. Fonksiyon çağrılarak, <cmath> kütüphanesinde bulunan pow() adlı program çalıştırılır ve sonuç y değişkenine atanır. Sıkça **fonksiyonun değer döndürdüğü** söylenir.

Aşağıdaki komutla 900.0 sayısının karekökünü yazdırmak için, 900.0 argümanıyla sqrt adlı fonksiyon çağrılır:

```
cout << sqrt(900.0);
```

Fonksiyondan döndürülen değer yazdırılır.

Belirtilen matematiksel fonksiyonlar, her çağrıda bir değer döndürür. Bu yüzden **dönüş değerli fonskiyonlar** (İng. value-returning functions) olarak adlandırılırlar.

Fonskiyonların argümanları sabitler, değişkenler veya bütün ifadeler olabilir. Örneğin, c = 12, d = 3.0, g = 4.0 için

```
cout << sqrt(c + d * f);
```

komutu 12.0 + 3.0 * 4.0 = 25.0'ın karekökünü, yani 5.00 değerini hesaplayacaktır ve yazdıracaktır.

[7] Bu sayı doğal logaritmanın temelidir ve Euler sayısı veya Nepper sayısı olarak adlandırılır.

Aşağıdaki örnekte, matematiksel fonksiyonların kullanımı gösterilmektedir.

Örnek 2.5.1

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

int main()
{   //   pow( ), sqrt( ), cbrt( ), exp( ) ve log( ) matematiksel fonksiyonlari

    int a, b, c;
    cout << " x icin ilk değeri girin, x = " ;
    cin >> a;
    cout <<  " x icin ikinci değeri girin, x = " ;
    cin >> b;
    cout <<  " x icin ucuncu değeri girin, x = " ;
    cin >> c;
    cout << setprecision(4) << endl;
    cout << "-----------------------------------------------------------------
    cout << setw(5) << "x" << setw(7) <<  "onc."  << setw(7) << "ard."
        << setw(7) << "x^2" << setw(15) << karekok << setw(15)
        << "kupkok(x")     << setw(12) << "e^x" << setw(10) << "ln(x)" << endl;
    cout << "-----------------------------------------------------------------
    cout << setw(5) << a << setw(4) << a - 1 << setw(7) << a + 1 << setw(10)
        << pow(a, 2) << setw(15) << sqrt(a) << setw(15) << cbrt(a) << setw(12)
        << exp(a) << setw(10) << log(a) << endl;
    cout << setw(5) << b << setw(4) << b - 1 << setw(7) << b + 1 << setw(10)
        << pow(b, 2) << setw(15) << sqrt(b) << setw(15) << cbrt(b) << setw(12)
        << exp(b) << setw(10) << log(b) << endl;
    cout << setw(5) << c << setw(4) << c - 1 << setw(7) << c + 1 << setw(10)
        << pow(c, 2) << setw(15) << sqrt(c) << setw(15) << cbrt(c) << setw(12)
        << exp(c) << setw(10) << log(c) << endl;
    cout << "-----------------------------------------------------------------

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

Şekil 2.5.1

Programın yürütülmesiyle aşağıdaki çıktı elde edilir:.

```
   x  onc.   ard.    x^2   karekok (x)    kupkok (x)        e^x     ln(x)
   2     1      3      4         1.414          1.26      7.389    0.6931
   3     2      4      9         1.732         1.442      20.09     1.099
  12    11     13    144         3.464         2.289   1.628e+05     2.485
```

Rastgele Sayı Üretme Fonskiyonu

Programlamada, sıkça rastgele bir sayı üretme ihtiyacı ortaya çıkar. Bu nedenle, C++'da 0 ile 32 767 arasında rastgele bir sayı üreten özel **rand()** fonksiyonu vardır. |[8]

Örneğin:

```
0 ile 32767 arasinda rastgele sayilar:
19169 26500 6334 18467 41
```

{0, 1, 2, 3... 32 767} kümesini [0, max] belirli bir aralığa indirmek istiyorsak, aşağıdaki formülü kullanıyoruz:

```
rand() % (max + 1);
```

Örneğin, [0, 99] aralığındaki rastgele sayılar:

```
0 ile 99 arasinda rastgele sayilar:
64 62 58 78 24
```

Rastgele sayıların 1'den başlaması için formül şu şekildedir:

```
1 + rand() % (max + 1);
```

Örneğin, önceki örnekten [1, 100] aralığında rastgele sayılar şunlardır:

```
1 ile 100 arasinda rastgele sayilar:
62 28 82 46 6
```

Rastgele üretilen sayılar aralığının belirli bir sayıdan (örneğin baslangic) başlamasını istiyorsak, formül şöyledir:

```
baslangic + rand() % (maks + 1);
```

Örneğin, [1, 100] aralığı, [51, 150] şu şekilde olabilir:

```
51 + rand() % 100;
```

```
51 ile 150 arasinda rastgele sayilar:
87 78 93 146 142
```

Rastgele sayılar üretmek için aynı ifadeyi kullanan bir program çalıştırıldığında, aynı rastgele sayı dizisi üretilecektir. Bunun nedeni, C++ *standart kütüphanesinden* rand() fonksiyonu her çağrıldığında aynı programın çalıştırılmasıdır. Bu yüzden, üretilen rastgele sayılar gerçekten rastgele değil ve bunlar **sözde rastgele sayılar** (İng. pseudo-random numbers) olarak adlandırılır.

rand() fonksiyonuna yapılan her çağrıda yeni bir rastgele sayı dizisi oluşturmak için, **srand**() fonskiyonuna bağımsız değişken olarak farklı işaretsiz tam sayı değeri belirtilerek,

[8] Değeri yazdırılabilen RAND_MAX sabiti vardır.

rastgeleleştirme (İng. randomizing) gerçekleştirilir. Bu fonksiyon, rand() fonksiyonu çağrılmadan önce yürütülmelidir.

Örneğin, önceki örnekte rastgele sayılar oluşturulmadan önce aşağıdaki fonskiyon yürütülürse:

```
srand(123);
```

aşağıdaki rastgele sayılar üretilecektir:

```
51 ile 150 arasinda rastgele sayilar:
114 55 126 104 91
```

Eğer, fonksiyonu yeniden farklı argümanla yürütürsek:

```
srand(321);
```

farklı rastgele sayılar üretilecek:

```
51 ile 150 arasinda rastgele sayilar:
116 131 88 69 137
```

srand() fonksiyonunun yalnızca adıyla çağrıldığını, atama komutu olmadığını görebiliriz. Bu, fonksiyonun değer döndürmediği anlamına gelir. **Dönüş değeri olmayan (değer döndürmeyen) fonskiyonlar** (İng. non value-returning functions) veya **prosedürler** (İng. procedures) olarak adlandırılan bu tür başka fonksiyonlar de vardır.

Örnek 2.5.2

Farklı rastgele sayı dizileri oluşturmanın etkili yollarından biri bilgisayar saatini kullanmaktır. Bilgisayar saati, programa dahil edilmesi gereken <**ctime**> kütüphanesinde bulunan time(0) fonksiyonu ile okunur ve o anki zamanın işaretsiz tam sayısını saniye olarak döndürür.

```
1   #include <iostream>
2   #include <string>
3   #include <ctime>
4
5   using namespace std;
6
7   int main()
8   {   // Rastgele sayilar: random( ) ve srand( ) fonksiyonları
9
10      cout << " 0 ve " << RAND_MAX << "arasinda rastgele sayilar: \n"
11          << rand() << " " << rand() << " " << rand() << " "
12          << rand() << " " << rand() << " " << endl;
13      int max = 100;
14      cout << "\n0 ve " << max - 1 <<"arasinda rastgele sayilar: \n"
15          << rand() % max << " " << rand() % max << " " << rand() % max << " "
16          << rand() % max << " " << rand() % max << " " << endl;
17      cout << "\n1 ve " << max << "arasinda rastgele sayilar: \n"
18          << 1 + rand() % max << " " << 1 + rand() % max << " " << 1 + rand() % max
19          << " " << 1 + rand() % max << " " << 1 + rand() % max << " " << endl;
20      int baslangic = 51;
21      cout << "\n1 ve " << baslangi + max - 1 << "arasinda rastgele sayilar: << ": \n"
22          <<          + rand() % max << " " << baslangic + rand() % max << " "
```

Şekil 2.5.2

```
23          <<baslangic + rand() % max << " " << baslangic  + rand() % max << " "
24          << baslangic + rand() % max << " " << endl;
25      srand(123);
26      srand(time(0));
27      cout << "\n Saniye olarak zaman: " << time(0) << endl;
28      cout << "1 ve " << baslangic+max-1 << "arasinda rastgele sayilar: \n"
29          <<baslangic + rand() % max << " " << baslangic + rand() % max << " "
30          << baslangic  + rand() % max << " " << baslangic  + rand() % max << " "
31          << baslangic + rand() % max << " " << endl;
32
33      cout << endl;
34      system("Color 17");
35      system("pause");
36      return 0;
37  }
```

Şekil 2.5.2 (devam)

Programın bir çıktısı şudur:

```
0 ve 32767 arasinda rastgele sayilar:
19169     26500     6334   18467   41

0 ve 99 arasinda rastgele sayilar:
64    62    58    78        24

1 ve 100 arasinda rastgele sayilar:
62   28    82    46        6

1 ve 150 arasinda rastgele sayilar:
87    78    93   146       142

Saniye olarak zaman: 1469456833
1 ve 150 arasinda rastgele sayilar:
61     72   108   132     137

Press any key to continue . . .
```

Şekil 2.5.2'deki programda time(0) fonksiyonu kullanılarak, programin iki kez çalıştırıldığında aşağıdaki rastgele sayı dizileri üretilecektir:

```
Saniye olarak zaman: 1469456809
1 ve 150 arasinda rastgele sayilar:
Saniye olarak zaman: 1469456833
1 ve 150 arasinda rastgele sayilar:
```

Karakterlerle Çalışma Fonksiyonları

C++'da, karakterlerle çalışmak için birçok foksiyon tanıtılmıştır. Bu fonksiyonlar, programın başında bulunması gereken <cctype> kütüphanesinde bulunmaktadır.

isdigit(z)	– z rakamsa (0, 1... 9) true'dur.
isalpha(z)	– z harf ise true'dur.
islower(z)	– z küçük harf ise true'dur.
isupper(z)	– z büyük harf ise true'dur.
isblank(z)	– z boş yer ise true'dur.
isspace(z)	– z boşluksa true'dur: ' ','\t', '\n', '\v', '\f', '\r'[9].
isalnum(z)	– z harf veya rakamsa true'dur.
ispunct(z)	– z noktalama işareti ise true'dur: . , ; : ' " ? / ! – ( )[ ] { } < > … vb.
isprint(z)	– z yazdırılabilen karakterse true'dur. (Örneğin, '\n', '\t' ve diğerleri yazdırılmaz.)
tolower(z)	– z büyük harf ise, küçük harfe çevirir.
toupper(z)	– z küçük harfse, büyük harfe çevirir.

Örnek 2.5.3

```
#include <iostream>
#include <string>
#include <iomanip>

using namespace std;

int main() {    // Karakterler calisma fonksiyonlari

    char z1 = '5';
    char z2 = 'W';
    char z3 = '@';
    char z4 = '\n';
    char z5 = ' ';
    char z6 = '?';
    bool evetHayir;
    evetHayit = isdigit( z1 );
    cout      << z1 << " rakam midir? " << boolalpha << evetHayir << endl;
    evetHayit = isdigit( z2 );
    cout      << z2 << " rakam midir? " << boolalpha << evetHayir << endl;
    evetHayit = isalpha( z2 );
    cout      << z2 << " harf midir? " << boolalpha << evetHayir << endl;
    evetHayit = isalpha( z3 );
    cout      << z3 << " harf midir? " << boolalpha << evetHayir << endl;
    evetHayit = islower( z2 );
    cout      << z2 " kucuk harf midir? " << boolalpha << evetHayir << endl;
```

Şekil 2.5.33

[9] '\t' – yatay sekme, '\n' – yeni satır, '\v' – dikey sekme, '\f' – yeni sayfa, '\r' – – önceki satır.

```
26      evetHayir  = isupper( z2 );
27      cout << z2 <<" buyuk harf midir? " << boolalpha  << evetHayir << endl;
28      evetHayir  = isblank( z4 );
29      cout << z4 << " bos yer midir? " << boolalpha << evetHayir << endl;
30      evetHayir  = isblank( z5 );
31      cout  << z5 <<" bos yer midir? " << boolalpha << evetHayir << endl;
32      evetHayir  = isspace( z4 );
33      cout << z4 << " bosluk mudur? " << boolalpha << evetHayir << endl;
34      evetHayir = isspace( z5 );
35      cout << z5 << " bosluk mudur? "  << boolalpha << evetHayir << endl;
36      evetHayir = isalnum( z1 );
37      cout << z1 << " harf veya rakam midir? "  << boolalpha << evetHayir << endl;
38      evetHayir = isalnum( z2 );
39       cout  << z2 << " harf veya rakam midir? << boolalpha << evetHayir << endl;
40     evetHayir = isalnum( z6 );
41       cout  << z6 << " harf veya rakam midir? " << boolalpha << evetHayir << endl;
42     evetHayir = ispunct( z6 );
43       cout  << z6 << "noktalama isareti midir? << boolalpha << evetHayir << endl;
44     evetHayir = ispunct( z4 );
45       cout << z4 << noktalama isareti midir? << boolalpha << evetHayir << endl;
46    evetHayir = isprint( z4 );
47       cout << z4 << "yazdirilabilen karakter midir? "  << boolalpha << evetHayir << endl;
48            = isprint( z6 );
49       cout << z6 << "yazdirilabilen karakter midir?" << boolalpha << evetHayir << endl;
50       cout << z2 << " harfi";
51       z2 = tolower( z2 );
52       cout << "kucuk" << z2 << "harfine donusur" << endl;
53       cout << z2  << " harfi";
54       z2 = toupper( z2 );
55       cout << "buyuk"  << z2 "harfine donusur" << endl;
56
57       cout << endl;
58       system( "Color 17" );
59       system( "pause" );
60       return 0;
61   }
```

```
5 rakam midir? true
W rakam midir? false
W harf midir? true
@ harf midir? false
W kucuk harf midir? false
W buyuk harf midir? true
          bos yer midir? false
 bos yer midir? true
          bosluk mudur? true
 bosluk nudur? True
5 harf veya rakam midir? true
W harf veya rakam midir? true
? harf veya rakam midir? false
? noktalama isareti midir? true
noktalama isareti midir? false
          yazdirilabilen karakter midir? false
? yazdirilabilen karakter midir? true
W harfi kucuk w harfine donusur
w harfi buyuk W harfine donusur
```

Şekil.2.5.3 *(devam)*

Programı yürüttükten sonra çıktı şudur:

sizeof() operatörü

Bu operatör, bir veri türü tarafından kullanılan bellek boyutunu (bayt olarak) hesaplamak için kullanılır. Onun sözdizimi şöyledir:

sizeof(*tür*);

Parantez gerektirmeden tek bir değişkene veya ifadeye de uygulanabilir.

sizeof *ifade;*

Örnek 2.5.4

Şekil 2.5.4'teki programla sizeof() operatörünün kullanımı gösterilmiştir.

```
1   // Operator sizeof()
2
3   #include <iostream>
4   #include <string>
5   using namespace std;
6
7   int main() {
8       short a;    int b;  long c; long long d;
9       float e;    double f;   long double g;
10      char h;     string s;
11
12      cout << "sizeof(short) = " << sizeof(short) << endl;
13      cout << "sizeof(a) = " << sizeof a << endl;
14      cout << "sizeof(int) = " << sizeof(int) << endl;
15      cout << "sizeof(b) = " << sizeof b << endl;
16      cout << "sizeof(long) = " << sizeof(long) << endl;
17      cout << "sizeof(c) = " << sizeof c << endl;
18      cout << "sizeof(long long) = " << sizeof(long long) << endl;
19      cout << "sizeof(d) = " << sizeof d << endl;
20      cout << "sizeof(float) = " << sizeof(float) << endl;
21      cout << "sizeof(e) = " << sizeof e << endl;
22      cout << "sizeof(double) = " << sizeof(double) << endl;
23      cout << "sizeof(f) = " << sizeof f << endl;
24      cout << "sizeof(long double) = " << sizeof(long double) << endl;
25      cout << "sizeof(g) = " << sizeof g << endl;
26      cout << "sizeof(char) = " << sizeof(char) << endl;
27      cout << "sizeof(h) = " << sizeof h << endl;
28      cout << "sizeof(string) = " << sizeof(string) << endl;
29      cout << "sizeof(s) = " << sizeof s << endl;
30
31      cout << endl;
32      system("Color 17");
33      system("pause");
34      return 0;
35  }
```

Şekil 2.5.4

Programın yürütülmesiyle şu elde edilir:

```
sizeof(short) = 2
sizeof(a) = 2
sizeof(int) = 4
sizeof(b) = 4
sizeof(long) = 4
sizeof(c) = 4
sizeof(long long) = 8
sizeof(d) = 8
sizeof(float) = 4
sizeof(e) = 4
sizeof(double) = 8
sizeof(f) = 8
sizeof(long double) = 8
sizeof(g) = 8
sizeof(char) = 1
sizeof(h) = 1
sizeof(string) = 28
sizeof(s) = 28
```

Alıştırma Ödevleri

Kütüphane fonskiyonları kullanarak aşağıdaki ödevler için programlar yazılsın:

1. x için bir değer girilsin ve $1 + x + x^2 + x^3 + x^4 + x^5$ toplamı hesaplansın.
2. $e^0 + e^1 + e^2 + e^3 + e^4 + e^5$ toplamı hesaplansın.
3. Tam sayı n girilsin ve $\left[\sqrt{\frac{n^2}{2}}\right]$ hesaplansın. [] tam bölümü gösterir.
4. $ax^2 + bx + c = 0$ ikinci dereceden denklemin a, b ve c katsayıları girilsin ve aşağıdaki formüle göre kökleri hesaplansın:

$$D = b^2 - 4ac, \qquad x_1 = \frac{-b + \sqrt{D}}{2a}, \qquad x_2 = \frac{-b - \sqrt{D}}{2a}.$$

5. 0^o, 30^o, 45^o, 60^o, 90^o, 180^o, 270^o ve 360^o açıları için sin(x) ve cos(x) trigonometrik fonksiyonları için aşağıdaki değer tablosu yazdırılsın. (Açı radyan olarak verilmektedir $x^{rad} = \frac{\pi}{360} x^o$)ç

x	0^o	30^o	45^o	60^o	90^o	180^o	270^o	360^o
sin(x)								
cos(x)								

6. [31, 100] aralığından rastgele bir tek sayı üretilsin.
7. Bir harf girilsin ve büyük harf olup olmadığı belirlensin.
8. Bir karakter girilsin ve bunun noktalama işareti olup olmadığı belirlensin.

Kullanıcı Fonksiyonları

Yapılandırılmış programlamada (**1.2 Algoritmalar** noktasında açıklanmıştır) iki programlama tekniğinin kullanıldığını söylemiştik:

- **Yukarıdan aşağıya programlama** (İng. top-down programming).
- **Modüler programlama** (İng. modular programming).

Yukarıdan aşağıya programlama, ödevi **alt ödevler** olarak adlandıracağımız daha küçük ve daha basit ödevlere bölerek (ayrıştırarak) yapılır. Gerekirse, bu alt ödevler de kolayca programlanabilir ödevler elde edilene kadar daha da basit olanlara bölünür.

Bu şekilde ayrıştırılan ödevin her bir alt ödevi, diğerlerinden bağımsız, ayrı bir ödev olarak düşünülebilir. Her alt ödev için **alt algoritma** olarak adlandıracağımız ayrı algoritma yazılabilir.

Örneğin, verilen üç sayıdan en büyüğünü bulmak için algoritma aşağıdaki gibi yazılabilir,***şekil 2.5.5:***

```
algoritma ÜçSayıdanEnBüyüğü
başlangıç
        oku a, b, c;
        eğer a > b
                o zaman
                        dahaBüyük ← a;
                değilse
                        dahaBüyük  ← b;
        bitiş_eğer {a > b}
        p ← dahaBüyük;
        eğer p > c
                o zaman
                        dahaBüyük  ← p;
                değilse
                        dahaBüyük  ← c;
        bitiş_eğer {p > c}
        n ← dahaBüyük;
        yazdır n;
bitiş {ÜçSayıdanEnBüyüğü }
```

Şekil 2.5.5

Yukarıdaki algoritmada, iki sayıdan büyük olanın belirlenmesi iki kez aranır: a ve b ile, p ve c. Bu, ayrı algoritma, yani alt algoritma olarak ayrılabilir.

Ancak, bu alt algoritmanın ikili karşılaştırma yapılabilmesi için: ilk karşılaştırmada a ve b, ikinci karşılaştırmada p ve c için nasıl yazılacağı konusunda sorun ortaya çıkar.

Ayrıca, alt algoritma herhangi iki sayının karşılaştırmasını yapmalıdır. Bunu sağlamak için, alt algoritma genel biçimde yazılır.

Bir alt algoritma, diğer herhangi algoritma gibi yazılır. İçinde veri girebilir ve yazdırılabilir, değişkenler bildirilebilir, türler ve sabitleri tanımlanabilir, değişkenlere değer atanabilir, çeşitli hesaplamalar yapılabilir ve algoritmalarda da gerçekleştirilebilecek diğer işlemler yapılabilir.

Alt algoritmalar da algoritmalar gibi adlandırılır, örneğin: DahaBuyukSayi(), AsalSayi(), HepsindenEnKucuk(), Degistirme(), DiziKaydirma() vb.[10]

Örneğin, iki sayıdan daha büyük sayıyı bulmak için alt algoritma ***Şekil 2.5.6'***daki gibi tanımlanabilir:

```
altalgoritma DahaBuyukSayi(sayi1, sayi2)
başlangıç
        eğer sayi1 > sayi2
                o zaman
                        dahaBuyuk ← sayi1
                değilse
                        dahaBuyuk ← sayi2;
        bitiş_eğer{sayi1 > sayi2}
        döndür dahaBuyuk;
bitiş {DahaBuyukSayi}
```

Şekil 2.5.6

Alt algoritmanın adı DahaBuyukSayi()'dır ve sayi1 ve sayi2 olmak üzere iki girdi parametresi vardır.

Alt algoritmada hesaplanan sonuç, döndür DahaBuyuk;
adımında döndürülüyor.

Bu tür alt algoritmalara **fonksiyon alt algoritmaları** denir. ***Şekil 2.5.5'***teki algoritmada büyük sayıyı bulma adımlarını ***şekil 2.5.6'***daki alt algoritma ile değiştirirsek, algoritma ***şekil 2.5.7'***deki gibi görünecektir:

```
algoritma  ÜçSayıdanEnBüyüğü
başlangıç
        oku a, b, c;
        p ← DahaBuyukSayi(a, b);
        n ← DahaBuyukSayi(p, c);
        yazdır n;
bitiş {ÜçSayıdanEnBüyüğü}
```

Şekil 2.5.7

[10] Alt algoritmaların isimlerini ilk harfi büyük olacak şekilde yazacağız. Bazı isimlerin Latince, bazılarının da Kiril alfabesiyle yazıldığını fark edebilirsiniz. Bunu daha sonra açıklayacağız.

```
p ← DahaBuyukSayi(a, b);
n ← DahaBuyukSayi(p, c);
```

adımlarının sağ taraflarında, a,b ve p,c argümanları için DahaBuyukSayi() alt algoritması çağrılır. Buna göre, fonksiyonel alt algoritmanın adı ve parantez içinde argümanları ile çağrılır. Fonksiyonel alt algoritmanın sadece bir değer döndürdüğünden, alt algoritma atama adımında (***Şekil 2.5.7***'deki gibi), ifadede veya başka bir adımda çağrılabilir.

Fonksiyonel alt algoritma çağrısının genel biçimi şu şekildedir:

```
AltAlgoritmaAdi(girdi_argümanlar_listesi);
```

DahaBuyukSayi() alt algoritması dönüş değeri parametre olacak şekilde de yazılabilir, Şekil 2.5.6 a

```
altalgoritma DahaBüyükSayi (↓sayi1, ↓sayi2, ↑pogolem)
başlangıç
        eğer sayi1 > sayi2
                o zaman
                        dahaBuyuk ← sayi1
                değilse
                        dahaBuyuk ← sayi2;
        bitiş_eğer {sayi1 > sayi2}
bitiş {DahaBuyuk}
```

Şekil 2.5.6 a

Bu tür alt algoritmalara **prosedürel alt algoritmalar** veya sadece **prosedürler** (İng.procedures) denir. Parametre listelerinde girdi parametreleri, çıktı parametreleri ve girdi-çıktı[11] parametreleri bulunabilir.

Prosedürel alt algoritmalarda girdi, çıktı ve girdi-çıktı parametrelerini ayırt etmek için girdi parametrelerinden önce aşağıya ok ↓, çıktı parametrelerinden önce ise yukarıya ok ↑ koyacağız. Bir parametre girdi-çıktı ise önüne çift yönlü ok ↕ koyacağız (Bu işaretlemeler fonksiyon alt algoritmalarında kullanılmıyor çünkü onlarda tüm parametreler girdi parametreleridir).

Prosedürel alt algoritması sadece adıyla çağrılır, parantez içinde ise argümanlar listelenir:

```
AltAlgoritmaAdı (girdi çıktı ve girdi-çıktı_argümanlar listesi);
```

Not: Prosedürel alt algoritmanın, fonkisyon alt algoritmada olduğu gibi algoritmanın başka bir adımında çağrılamayacağını bilmek için, prosedürel alt algoritmaların adını Kiril harfleriyle yazacağız.

[11]Programlama dillerinde girdi, çıktı ve girdi-çıktı parametrelerini işaretlemenin farklı yolları vardır. En yaygın olarak in, out ve inout ile gösterilir.

Söylediklerimize göre, DahaBüyükSayı() prosedürel alt algoritmasını kullanarak verilen üç sayıdan en büyüğünü bulma algoritması ***şekil 2.5.7 a***'daki gibi olacaktır:

```
algoritma ÜçSayıdanEnBüyüğü
başlangıç
        oku a, b, c;
        DahaBüyükSayı (a, b, p);
        DahaBüyükSayı (p, c, n);
        Yazdır n;
bitiş {ÜçSayıdanEnBüyüğü}
```

Şekil 2.5.7 a

Prosedürel alt algoritmalar, genellikle alt algoritmanın birden fazla değer döndürmesi gereken durumlarda kullanılır.

Örneğin, iki bilinmeyenle iki doğrusal denklem sistemini çözerken, çözüm iki bilinmeyen için iki değerden oluşur veya ikinci dereceden denklem çözerken, çözüm karekökler için iki değerden oluşur ve benzeri.

Programlama dillerinde kodlanan alt algoritmalara **alt program** (İng. subprograms) denir. Alt programlar, **fonksiyonlar** (İng. functions) veya **prosedürler** (İng. procedures) olarak yazılabilir. Fonksiyonlar, sadece bir (yürütmeden sonra geri dönen) sonuç veren alt programlardır. Prosedürler ise, değer döndürmeyen alt programlardır. (Fakat prosedürlerle birden fazla sonuç alınmasını sağlayan mekanizmalar vardır).

Alt programların özü, farklı programlarda ve aynı programda birkaç yerde kullanılabilmesidir. Bu, **biçimsel argümanlar** (İng. formal arguments) ve **gerçek argümanlar** (İng. actual arguments) mekanizması aracılığıyla sağlanmaktadır. Literatürde, biçimsel argümanlara genellikle **parametreler** (İng. parameters) denir, gerçek argümanlar ise sadece **argümanlar** olarak adlandırılır. Biz son iki terimi kullanacağız.

C++'da sadece fonksiyonlar vardır. Fonksiyonlar (matematikte olduğu gibi) yalnızca bir sonuç, yani bir değer verir (döndürür). Bu yüzden (kütüphane fonskiyonlari gibi), bunlara **dönüş değerli fonksiyonlar denir**. Fonksiyonel alt algoritmalar bu tür fonksiyonlarla gerçekleştirilir. Prosedürel alt algoritmalar, C++'da **dönüş değeri olmayan fonksiyonlar** olarak adlandırılan prosedürler (kütüphane fonksiyonları gibi) ile gerçekleştirilir.

C++'da fonksiyonlar ikiye ayrılabilir:

- Kütüphane fonksiyonları.
- Kullanıcı tanımlı fonksiyonlar.

Kütüphane fonksiyonlarını **kütüphane fonksiyonlar** alt başlığında tanıdık.

Kullanıcı fonksiyonları (İng. user functions), kullanıcılar (programcılar) tarafından tanımlanır ve bu yüzden onlara **kullanıcı tanımlı fonksiyonlar** (İng. user-defined functions) da denir.

Dönüş Değerli Kullanıcı Fonksiyonlar

Daha önce açıklanan, verilen üç sayıdan en büyüğünü bulma örneğiyle başlayacağız.

Şekil 2.5.6'daki DahaBuyukSayi() fonskiyon alt algoritması, C++'da fonksiyon olarak tanımlanabilir, ***Şekil 2.5.8***:

```
int dahaBuyukSayi(int sayi1, int sayi2) {
	int DahaBuyuk;
	if(sayi1 > sayi2)		// Eğer sayi1 sayi2'den daha buyukse,
		DahaBuyuk = sayi1;	// o zaman sayi1 daha buyuktur,
	else				// degilse
		DahaBuyuk = sayi2;	// sayi 2 daha buyuktur.
	return DahaBuyuk;
}
```

Şekil 2.5.8

Fonkisyonun adı dahaBuyukSayi[12]'dır. Adın önünde, fonksiyonun döndürdüğü sonucun türü belirtilir. Bu örnekte, dahaBuyukSayi() fonksiyonunun sonucu int türündendir. Fonksiyonun adından sonra parantez içinde sayi1 ve sayi2 parametrelerinin isimleri ve türleri belirtilir. Parametreler, fonksiyon tanımında kullanılan değişkenlerdir. Fonksiyon, sonucu

```
return dahaBuyuk;
```

Üç sayıdan en büyüğünü bulmak için main() ana işlevi ***Şekil 2.5.9***'da verilmiştir:

```
int main() {
	int a; // ilk tamsayi
	int b; // ikinci tamsayi
	int c; // ucuncu tamsayi
	int p, n;
	cout << "Uc tamsayi girin: \n";
	cout << "İlk sayi:";
	cin >> a;
	cout << "İkinci sayi:";
	cin >> b;
	cout << "Ucuncu sayi:";
	cin >> c;
```

[12] Kullanıcı fonksiyonlarının isimlerini ilk harfi küçük olacak şekilde yazacağız. İsim birleştirilmiş birkaç kelimeden oluşuyorsa, her kelime (ilki hariç) büyük harfle yazılacaktır.

```
        // a, b ve c se argumlalardir
        p = dahaBuyukSayi(a, b);    // Fonksiyon cagrisi
        n = dahaBuyukSayi(p, c);    // Fonksiyon cagrisi
        cout  << a << ", " << b << "ve" << c <<
                "sayilarindan en buyugu" << n << "'dir" << endl;

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

Şekil 2.5.9

main() fonksiyonunu inceleyeceğiz.

a, b ve c sayılarını okuduktan sonra, atama komutunun sağında a ve b argümanlı fonskiyon çağrılır

```
p = dahaBuyukSayi ( a, b );
```

Bu arada a ve b argümanlarının değerleri sayi1 ve sayi2 parametrelerine aktarılır. Fonksiyon yürütülürken, sayi1 ve sayi2'nin daha büyük değeri dahaBuyuk değişkenine atanır ve fonksiyonun sonucunu belirler.

```
return dahaBuyuk;
```

komutu ile dahaBuyukSayi() fonksiyonun sonucunu, yani ana programdaki dahaBuyuk değişkeninin değerini döndürür. Bu nedenle, dahaBuyukSayi() fonksiyonu için dönüş değerli fonksiyon olduğunu söylüyoruz.

Atama komutuyla:

```
p = dahaBuyukSayi(a, b);
```

dahaBuyuk değişkeninin dönüş değeri p değişkenine atanır.

```
p = dahaBuyukSayi(a, b);
```

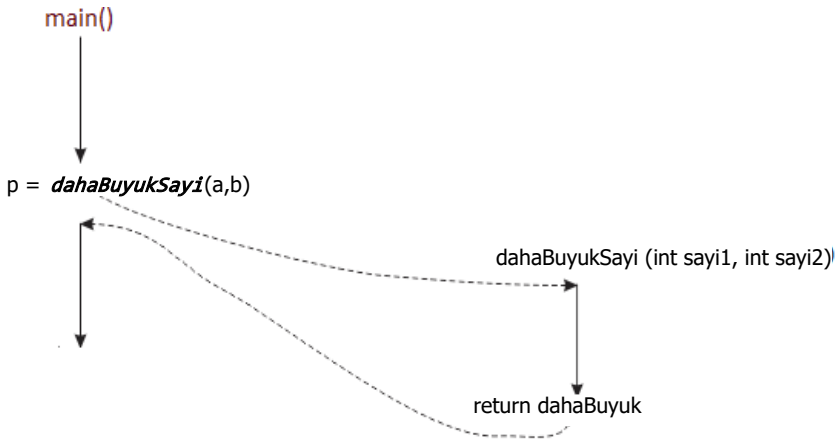
atama komutunun yürütülmesi ***Şekil 2.5.10***'da şematik olarak gösterilmiştir.

dahaBuyukSayi() fonksiyonunu yeniden çağıran bir sonraki atama komutunun yürütülmesi sırasında, n değişkenine p ve c sayılarından büyük olanın değeri atanır.

```
n = dahaBuyukSayi(p, c);
```

Bu komutu yürüttükten sonra, n değişkeni a, b ve c değişkenlerinin en büyük değerini içerir.

Fonksiyon argümanlarının türü, fonksiyon tanımındaki karşılık gelen parametrelerin türüyle aynı veya uyumlu olan sabitler, değişkenler veya ifadeler olabilir.



Şekil 2.5.10

Örneğin, dahaBuyukSayi() fonksiyonuna aşağıdaki çağrılar geçerlidir:

```
p = dahaBuyukSayi((int) (20 * 123 / 45), 6);
p = dahaBuyukSayi((int) 1.2345f, (int) Math.sqrt(1.5));
```

Her fonksiyon çağrısında, programın derlenmesi sırasında argümanların sayısı ve türü kontrol edilir.

- Argüman sayısı parametre sayısından farklıysa hata mesajı gelir.
- Parametrelerin türü karşılık gelen argümanların türüyle eşleşmiyorsa, çevirici önce argümanın türünü karşılık gelen parametrenin türüne "çevirmeye" (dönüştürmeye) çalışıyor.

Örneğin, double türünü int türüne. (Bu arada çağrı yapılabilir ancak sonuç yanlış olabilir). Türün "çevirisi" başarısız olursa, çevirici hata mesajı verir.

Dönüş değeri olan fonksiyon, programın farklı yerlerinde, farklı argümanlarla ve farklı şekillerde çağrılabilir.

Örneğin:

```
      // Fonksiyonu çagirmanin bir yolu:
p = dahaBuyukSayi(a, b);
      // Fonksiyonu çagirmanin başka bir yolu:
System.out.println("\ndahaBuyukSayi (a, b) : " + "sayisi daha buyuktur");
      // Fonksiyonu çagirmanin ucuncu yolu:
if(dahaBuyukSayi(a, b) > 99)
      System.out.print("\nBirincilik odulu.");
```

Şekil 2.5.11'deki gibi dahaBuyukSayi() fonksiyonunu ve main() ana fonksiyonunu bir programa koyarsak ve yürütürsek aşağıdaki çıktıyı elde edeceğiz:

```
Uc tamsayi girin:
Ilk sayi: 123
Ikinci sayi: 321
Ucuncu sayi: 231
123, 321 ve 231 sayilardan en buyuğu 321'dir
Press any key to continue . . .
```

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

// dahaBuyukSayi() fonksiyonunun tanimi .
// sayi1 ve sayi2 girdi parametreleridir.

       dahaBuyukSayi() fonksiyonu

// Ana fonksiyon

       main() fonksiyonu
```

Şekil 2.5.11

Dönüş Değerli Fonksiyonun Tanımı ve Bildirimi

C++'da dönüş değerli fonksiyonun tanımı şu şekilde yapılır:

```
    tür ad(parametreler_listesi) {
          fonksiyonun gövdesi (komutlar)
          return ifade;
    }
```

Fonksiyon başlığı

Notlar:

- Fonksiyonun başlığı ; **(noktalı virgül)** ile bitmez.
- *tür*, fonksiyonun türüdür, aslında dönüş değerinin türüdür.
- *ad*, fonksiyonun adıdır.
- parametreler_listesi (İng. parameters list), parametrelerin adlarını ve türlerini içermelidir.
- *ifade*, fonksiyonun sonucudur. Bu bir ifade (değişkenler ve/veya sabitler ve/ veya fonksiyonlardan oluşan) veya fonksiyonun türüyle aynı türde olması gereken bir değişken olabilir.
- Parametreler_listesindeki *parametreler,* **girdi parametreleri** (İng. input parameters) ve **girdi-çıktı parametreleri** (İng. input-output parameters) olabilir.

Fonksiyonun bildirimi C++'da aşağıda verilmiş biçimdeki komutla yapılır:

```
tür ad (parametreler_listesi);
```

tür – fonksiyonun döndürdüğü değerin türü,
ad – fonksiyonun adı,

parametreler_listesi – virgülle ayrılmış parametrelerin adlarını ve türlerini içerir.

Notlar:

- Fonksiyonun türü belirtilmezse, int türü varsayılır.
- Fonksiyonların adlarını (uzlaşıma göre) değişkenler[13] gibi, yani küçük baş harfle yazacağız. Fonksiyonun adı birkaç kelimeden oluşuyorsa, sonraki her kelimeyi büyük harfle yazacağız..
- Parametreler listesi parametre türlerini içermelidir, adları ise olabilir veya olmayabilir.
- Fonksiyonu bildirimi ; (noktalı virgül) işaretiyle bitiyor.

Fonksiyonun bildirimine **fonksiyon prototipi** (İng. function prototype) denir.

Fonksiyon prototipi ile, fonksiyon arayüzü tanımlanır, yani fonksiyonu çağırmadan önce bilinmesi gereken parametrelerin sayısı ve türü ile sonucun türü hakkında bilgi sağlanmaktadır.

Kullanıcı fonksiyonlarının fonksiyon prototipleriyle ilgili birkaç örnek vereceğiz:

```
int carpma(int m, int n);          – adı: carpma
                                   – parametreler: m ve n
                                   – fonkisyon türü: int
float kok(float x);                – fonksiyon türü: float
int dahaKucuk(int , int);          – parametre adları olmayan liste
int g();                           – boş bir parametre listesi olan bir işlev,
                                   – yani parametre yok.
int f(int a, b);                   – yanlış:int f(int a, int b);
                                     olması gerekir
long alan(long uzunluk,long kenar); – fonksiyon türü: long
int h(void);                       – parametresiz listeli fonksiyon [14]
void yazdir(int sayiMesaj);        – void[15] türünden fonksiyon
```

Fonksiyon prototipinin adını ve parametre listesini içeren kısmına, f**onksiyon imzası** (İng. function signature) denir.

Örneğin, fonksiyon imzaları şunlardır:

```
carpma(int m, int n);
kok(float b);
dahaKucuk(int, int);
alan(long, long);
```

[13] Fonksiyonların adları, ifadelerde ve diğer fonksiyonların argümanları olarak, yani değişkenler gibi kullanılabilir.

[14] void kelimesini daha sonra açıklayacağız.

[15] Daha sonra açıklayacağız.

Dönüş değerli fonksiyon, atama komutuyla çağrılıyor:

değişken = ad(argümanlar_listesi);

Ad fonksiyonunun argümanlar_listesi ve parametreler_listesi şunları içermelidir:

- eşit sayıda argümanlar ve parametreler,
- aynı argümanlar ve parametreler sıralaması,
- argümanların ve karşılık gelen parametrelerin aynı veya dönüştürülebilir türü.

Fonkiyonların çağrılmasıyla ilgili birkaç örnek vereceğiz:

```
carpım = carpma(a, b);
kareKok = kok(12.345);
n = dahaKucuk(c, p);
P = alan(uzunluk, genislik);
```

Fonksiyon çağrılırken, argümanların değerleri (önceden tanımlanmış olması gereken - bir değere sahiptirler) karşışık gelen parametrelere aktarılır (kopyalanır).

Örneğin, aşağıdaki çağrı sırasında:

```
carpım = carpma(a, b);
```

a ve b argümanlarının değerleri, carpma() fonksiyonunu tanımındaki m ve n parametrelerine aktarılır (kopyalanır)

```
int carpma(int m, int n) {...}
```

main() ana fonksiyonundan sonra tanımlanan fonksiyonlar, ana fonksiyonda çağrılabilmeleri, yani çeviricinin ana fonksiyonu çevirirken bu adların neyi tanımladığını bilmesi için ana fonksiyondan önce (prototiplerini belirterek) bildirilmelidir.

Fonksiyonun tanımında, hem ana main() fonksiyonundan hem başka fonksiyondan çağrılsa, yürütüldükten sonra sonuç olarak neyin döndürüleceği belirtilmesi gerekiyor.

Bir fonksiyon aynı programda sadece bir kez tanımlanabilir.

Fonksiyonun Sabit Parametreleri

Şekil 2.5.8'deki dahaBuyukSayi() fonksiyonunda, parametreler sayi1 ve sayi2'dir. Ancak bu parametrelerden birinin değerini fonksiyonun içinde değiştirmek mümkündür.

Örneğin fonksiyona şu komutu eklersek:

```
sayı1 = 99999;
```

o zaman ***şekil 2.5.11***'deki programda, 99 999'dan küçük olan sayi1, sayi2 ve sayi3 için girdiğimiz değerler ne olursa olsun sonuç 99 999 olacaktır:

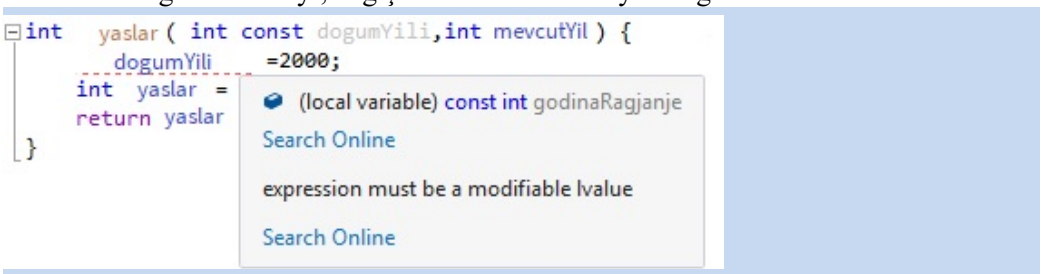
Fonksiyondaki parametrelerin, fonksiyonda değişikliklerden korunması için, **const** nitelendiriciyle (İng. qualifier) **sabit parametreler** olarak belirtilirler.

Aşağıdaki fonksiyonda (***şekil 2.5.12***), bir kişinin yaşı, içinde bulunduğumuz yıl ile doğum yılı arasındaki fark olarak hesaplanır. Doğum yılı değişmez olduğu için sabit olarak işaretlenmelidir.

```
1   #include <iostream>
2   using namespace std;
3
4   int vozrast( int const, int );
5
6   int main() {
7       int dogumYili, mevcutYil;
8       cout << "Dogum yilini girin: ";
9       cin >> dogumYili;
10      cout << "Simdiki yili girin: " ;
11      cin >> mevcutYil;
12
13      int yaslar  = yas ( dogumYili, mevcutYil );
14      cout << "\nSiz: << yaslar << "yasindasiniz."<< endl;
15
16      cout << endl;
17      system( "Color 17" );
18      system( "pause" );
19      return 0;
20  }
21
22  int  yas   ( int const dogumYili, int mevcutYil  ) {
23      int yaslar = mevcutYil - dogumYili;
24      return yaslar
25  }
```

Şekil 2.5.12

dogumYili parametresi, fonksiyonun tanımında sabit olarak nitelendirildiğinden dolayı, değiştirilemez - hata meydana gelir.



Şekil 2.5.12'deki örnekte, fonksiyonun main() ana fonksiyonundan sonra tanımlandığını ve bu nedenle fonksiyonun prototipinin main()'den önce belirtildiğini vurgulayalım.

Çözülmüş Ödevler

Ödev 2.5.1

Kütüphane fonksiyonu kullanmadan, xn'i hesaplamak için fonksiyon yazılsın.

Program ***şekil 2.5.13***'te verilmiştir.

```
1   // x^n fonksiyonu.
2   #include <iostream>
3   using namespace std;
4
5   int XussuN ( int n, double x ) {
6       double p = 1;
7       for( int i = 1; i <= n; i++ )
8           p = p * x;
9       return p;
10  }
11
12  int main() {
13      system( "Color 17" );
14      int n;
15      double a, kuvvet ;
16      cout << " Taban girin: a= " ;  cin >> a;
17      cout << " Kuvvet girin: n= " ; cin >> n;
18      stepen = XussuN ( n, a );
19      cout << '\n' << a << "^" << n << "kuvvetinin degeri: "
20          << " = " << kuvvet << endl;
21
22      cout << endl;
23      system( "Color 17" );
24      system( "pause" );
25      return 0;
26  }
```

Şekil 2.5.13

Ödev 2.5.2

Bir doğal sayının rakam sayısını ve rakamlarının toplamını hesaplamak için ayrı fonksiyonlar yazılsın.

Program ***şekil 2.5.14***'te verilmiştir.

```
1   //Dogal sayinin rakam sayisi ve rakamlarin toplami
2   #include<iostream>
3   using namespace std;
4
5   int rakamSayisi ( int m );
6   int rakamlarToplami ( int m );
7
8   int main() {
9       system( "Color 17" );
10      int n;
11      cout << " Dogal sayi girin: " ;  cin >> n;
12      cout << "\n Girilen sayi " << rakamSayisi ( n )
13          << "rakamlıdır " << endl;
14      cout << "\n Girilen sayinin rakamlarinin toplami " <<
15      rakamlarToplami( n ) << endl;
16
17      cout << endl;
18      system( "Color 17" );
19      system( "pause" );
20      return 0;
21  }
22
23  int rakamSayisi ( int m ) {
24      int sayi  = 0;
25      do {
26          sayi ++;
27          m /= 10;
28      } while( m > 0 );
29      return sayi;
30  }
31
32  int rakamlarToplami  ( int m ) {
33      int rakam, toplam = 0;
34      do {
35          rakam = m % 10;
36          toplam += rakam;
37          m /= 10;
38      } while( m > 0 );
39      return  toplam;
40  }
```

Şekil 2.5.14

Alıştırma Ödevleri

Aşağıdaki ödevler için dönüş değerli fonskiyonlar yazılsın:

1. n!'in hesaplanması.
2. İki sayının ortalamasını bulma.
3. Bir düzlemde iki nokta arasındaki mesafeyi hesaplamak için fonksiyon kullanarak bir çokgenin çevresini hesaplamak.
   (Eğer $A(x_a, y_a)$ ve $B(x_b, y_b$ ise), mesafe $d = \overline{AB} = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$).

4. a ve b'yi E.B.O.B ile bölerek $\frac{a}{b}$ kesirini (a ve b tam sayılardır) kısaltmak. a ve b'nin E.B.O.B.'ini bulmak için ayrı bir fonksiyon yazılsın.

5. İki sayının E.K.O.K.'ını bulmak.
6. n sayı için en büyük ortak böleni bulma. İki sayının E.B.O.B'ini bulmak için ayrı bir fonksiyon yazılsın.
7. Basamakları yalnızca 0 ve 1 olan ondalık tam sayılar olarak ifade edilen iki ikili sayının toplamını ve farkını hesaplamak için ayrı fonksiyonlar yazılsın. (Örneğin, x = 10011101 ve y = 01101001 için.)
8. Tabanı 8 olan bir sayı sisteminde iki sayının toplamının hesaplanması. (Bu sayı sistemindeki rakamlar: 0, 1, 2, 3, 4, 5, 6 ve 7'dir).
9. n doğal sayısının kusursuz olup olmadığını kontrol etme. (Kusursuz sayılar, sayının kendisi dışında bölenlerinin toplamına eşit olan sayılardır). n'den küçük tüm kusursuz sayılar bulunsun.
10. n doğal sayısının asal olup olmadığını kontrol etme. (Bir sayı sadece 1'e ve kendisine bölünebiliyorsa asaldır). n doğal sayısından küçük tüm asal sayıları bulunsun.
11. İlk k doğal sayının toplamı, yani 1 + 2 +...+ k için fonksiyonu kullanarak 1 + (1 + 2) + (1 + 2 + 3) +... + (1 + 2 + 3 +... + n) toplamı hesaplansın.
12. Verilen bir ikili tam sayının ondalık eşdeğeri bulunsun.

Dönüş Değeri Olmayan Kullanıcı Fonksiyonları

C++'da değer döndürmeyen fonksiyonlara, void türünden fonksiyonlar olduğunu diyoruz.

Örneğin, yazdirma () fonkisyonunun tanımı şöyle olabilir:

```
void yazdirma(int sayi, float sonuc) {
	cout << sayi << sonuc << endl;
	// sayi int turunden sayidir
	// sonuc float turunden sayidir
}
```

Bu fonksiyona yapılan çağrı, main() fonksiyonundan veya başka bir fonksiyondan olabilir. Dönüş değeri olmayan fonksiyonu çağırmak, dönüş değerli fonksiyonu çağırmaktan farklıdır. Dönüş değeri olmayan fonksiyon, sadece adı ve argümanlar listesiyle çağrılır.

Örneğin:

```
yazdirma (n , x);
```

void türünden bir fonksiyon parametrelersiz de olabilir.

Örneğin:

```
void girdi() {
	cout << "Girdi verilerin girilmesi\n ";
}
```

veya

```
void cikti(void) {   // void parametresiyle
                              // onsuz oldugu gibi aynidir
       cout << "Cikti verilerin yazdirilmasi\n ";
}
```

void türünden fonksiyona yapılan çağrı şöyle olabilir:

```
girdi();      girdi(void);         cikti();      cikti(void);
```

void türünden fonksiyonundan, yürütüldükten sonra, (dönüş değerli fonksiyondan farklı olarak) return komutu olmadan otomatik olarak çıkılır. Ancak, void türünden fonksiyonun gövdesinde herhangi bir yerden parametresiz return komutu ile çıkmak da mümkündür.

Örneğin:

```
void kareKok ( double sayi ) {
    if( sayi < 0 ) {
        cout << "Sayi negatiftir ve karekoku yoktur." << endl;
        return;
    }
    else
        cout << sayi << " sayisinin karekoku "
        << sqrt( sayi ) << endl;
    cout << "void fonksiyonun sonu." << endl;
}
```

Örneğin:

```
double sayi  = 2;
kareKok(broj);
```

ile şu yazdırılacak:

```
2 sayisinin karekoku 1.41421'dir
void fonksiyonun sonu.

Press any key to continue . . .
```

void türünden fonksiyon aşağıdaki şekilde çağrılırsa:

```
double sayi  = -2;
kareKok(sayi);
```

şu yazdırılacak:

```
Sayi negatiftir ve karekoku yoktur.

Press any key to continue . . .
```

İkinci çağrıda, birinci çağrıdan farklı olarak:

```
cout << "void fonksyonun sonu." << endl;
```

komutun yürütülmediğini görüyoruz, çünkü fonksiyon return komutuyla sona eriyor.

Dönüş değeri olmayan fonksiyonlar, genelde fonksiyonun birden fazla değer döndürmesi gereken durumlarda kullanılır. Örneğin, iki değişkenin içeriklerini değiştirmesi gerektiğinde, her iki değerin de döndürülmesi gerekir. n verinin sıralanması gerektiğinde, bunların döndürülmesi gerekir vb

void türünden fonksiyonun genel söz dizimi şöyledir:

```
void FonksiyonunAdı(girdi_çıktı_ve girdi-çıktı_parametreler_listesi) {

        fonksiyonun gövdesi
}
```

Dönüş değeri olmayan fonksiyonlar sadece adıyla çağrılıyor, parantezde ise argümanlar belirtiliyor:

```
FonksiyonunAdi (girdi çıktı ve girdi-çıktı_argümanlar_listesi);
```

Not: Dönüş değeri olmayan fonksiyon, dönüş değerli fonksiyon gibi ana fonksiyondan veya başka bir fonksiyondan başka bir komutta çağrılamaz.

Örnek 2.5.5

İki değişkenin değerlerini değiştirmek için void türünden fonksiyonun kullanılacağı uygulama yazılsın.

Program ***şekil 2.5.15***'te verilmiştir.

Programın bir çıktısı şudur:

```
a = 3
b = 4
a = 3 ve b = 4 degistirilirse a = 4 ve b = 3 olacak

Press any key to continue . . .
```

main() ana fonksiyonunda, void türünden degistir() fonksiyonuna yapılan çağrıdan sonra aşağıdaki komutu eklersek:

```
cout << "a ve b'nin değerleri void (degistir) fonksiyonundan dondukten sonra"
        << "a = " << a << "i b = " << b << endl;
```

Şu yazdırılacaktır:

```
a = 3
b = 4
a = 3 ve b = 4 degistirilirse a = 4 ve b = 3 olacak
a ve b'nin değerleri void (degistir) fonksiyonundan dondukten sonra
sunlar olacaktir: a = 3 ve b = 4
Press any key to continue . . .
```

Gördüğümüz gibi void türü fonksiyonda a ve b'nin değiştirilen değerleri yazdırılıyor, çağırıcı (main() fonksiyonu) döndükten sonra değerler değiştirilmiyor.

Örnek 2.5.5'teki (***Şekil 2.5.15***) programın yürütme sürecini, yani main()'den void türünden degistir() fonskiyonunu çağırmayı, fonksiyonun yürütülmesini ve main()'e geri dönmesini inceleyelim

```
1   #include <iostream>
2   using namespace std;
3
4   void degistir ( int, int );
5
6   int main() {
7       // iki degiskenin degerlerinin degisitirilmesi (void fonksiyonu ile)
8
9       int a, b;
10      cout << "a = ";      cin >> a;
11      cout << "b = ";      cin >> b;
12      cout << "a = " << a << " ve b = " << b << "degistirilirse";
13      degistir ( a, b );
14
15      cout << endl;
16      system( "Color 17" );
17      system( "pause" );
18      return 0;
19  }
20
21  void degistir ( int birinci, int ikinci ) {
22      int yard;
23      yard = birinci;
24      birinci = ikinci;
25      ikinci = yard;
26      cout <<"a : " << birinci<<"ve b = " << ikinci<<"olacak" << endl;
27  }
```

Şekil 2.5.15

Program çalıştırıldığında önce a ve b değişkenleri için bildirimler sırasında bellekte (**Şekil 2.5.16**) yer ayrılır ve ardından okuma sırasında girilen değerlerle doldurulur.

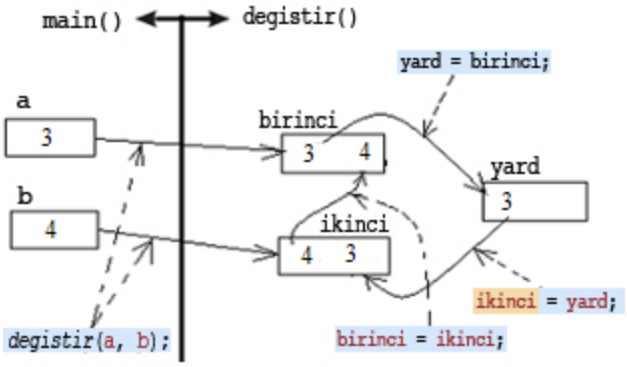
void türünde fonksiyon çağrıldığında işlem, fonksiyonun yürütülmesiyle devam ediyor. İlk olarak, bellekte birinci ve ikinci parametreleri için yer ayrılır ve a ve b argümanlarının değerleri bunlara aktarılır (kopyalanır). Ardından, yard değişkeni için yer ayrılır ve aşağıdaki komutlar yürütülür:

```
yard   = birinci;
birinci = ikinci;
ikinci = yard ;
```

Bu, **Şekil 1.5.16**'nın sağ tarafında gösterilmişti.

void türünden fonskiyonun sonunda, b ve a'nın değerlerini içeren birinci ve ikinci değişkenlerinin değerleri yazdırılıyor.

void türünden degistir() fonkisyonunun yürütülmesi ardından, işlem, ***şekil 2.5.16'***da görülebileceği gibi, a ve b değişkenlerinin aynı değerleri koruduğu main() fonksiyonuna geri dönüyor.



Şekil 2.5.16

Bu mekanizma **argümanların değere göre aktarılması** olarak adlandırılır, argümanlara ise **değere göre aktarılan argümanlar** (İng. pass-by-value arguments) denir.

void türünden bir fonksiyondaki parametre değerlerinin değiştirilebileceğini, ancak çağırıcıya (ana fonksiyonu veya başka fonksiyon) geri dönerken, karşılık gelen argümanlara kopyalanmadıklarını belirtmemiz önemlidir. Bu yüzden argümanlar, void türünden fonksiyon çağrılmadan önce sahip oldukları değerlerini korur. Buna göre, kopyalama tek yönde, yani sadece parametrelerdeki argümanlar üzerinde yapılır, tersi yapılmaz. Bu tür parametrelere **değer parametreleri** (İng. value parameters) denir.

void türünden degistir() fonksiyonu yürütüldükten sonra, yürütülmesi sırasında kullanılan tüm bellek (içinde bildirilen değişkenler ve parametreler listesi) serbest bırakılır. Bellekte sadece main() fonksiyonunun yürütülmesi sırasında oluşturulan değişkenler kalır.

Çoğu zaman, çağırıcıda (main() fonksiyonuna veya başka bir fonksiyona) void türünden fonksiyonunun birden fazla sonuç döndürmesi gerekir. C++'da void türdeki fonksiyonlar değer döndürmeyen fonksiyonlar olduğundan, birden fazla sonucun döndürülmesi, **referans parametreleri** (İng. referance parameters) tarafından sağlanmaktadır.

Referans Parametreleri

Referanslama (ing.referencing), kısaca, aynı değişkene başka bir isim vermektir. Örneğin:

```
int b = 3;
int & r = b;
```

İkinci komutla, b değişkenine yönlendiren r referansı bildirilir.

& (ampersan, ve) işareti "referans" olarak okunur. r değişkeni **referans değişkeni** (İng. reference variable) veya kısaca **referanstır** (İng. reference).

İkinci komut şöyle okunur: "r, b'ye başlatılan tamsayı referansıdır" veya "r, b'yi gösteren tam sayı referansıdır".

Referans bildirilirken, referansa gösterdiği değişken atanarak başlatılmalıdır. Referansı bildirdikten ve başlattıktan sonra, başka bir değişkene yönlendirmek için değiştirilemez. Ancak aynı değişkene yönlendiren birden çok referans bildirilebilir.

Ayrıca referansın türü yönlendirdiği değişkenin türü ile aynı olmalıdır.

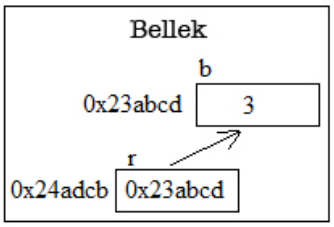
Referans bir değişkene yönlendirmesinden dolayı, çoğu zaman referans için değişkenin başka bir adı olduğunu diyoruz. Örneğin, b değişkenini ve ona yönlendiren r referansını yazdırırsak:

```
int b;
int& r = b;
b = 3;
cout << "Degisken  b = " << b << endl;
cout << "Referans  r = " << r << endl;
```

aynı değer elde edilecektir:

```
 Degisken b = 3
 Referans r = 3

Press any key to continue . . .
```

Aşağıdaki çizim (***şekil 2.5.17***), bellekteki b değişkenine yönlendiren r referansını göstermektedir. Hem b hem de r, bellekte yer kaplayan (bazı adreslerde bulunan) değişkenler olduğundan, yönlendirme, r referansı b değişkeninin adresini içerecek şekilde yapılır.



Şekil 2.5.17

Referansların dönüş değeri olmayan fonksiyonlarda referans parametreleri olarak kullanımını yine aynı **Örnek 2.5.5** ile açıklayacağız.

void türünden degistir() fonksiyonundan döndükten sonra, a ve b değişkenlerinin değerleri değişmediğinden dolayı, fonksiyonu önlerinde & işaretini koyarak referansları parametrelerle (**Şekil 2.5.18**) yazacağız:

```
void degistir(int &birinci, int &ikinci) {
	int yard;
	yard = birinci;
	birinci = ikinci;
	ikinci = yard;
	cout << "a = " << birinci  << "i b = " << ikinci  << endl;
}
```

Şekil 2.5.18

Ayrıca, void türünden fonksiyonun prototipinde de, parametrelerin referans olduklar belirtilmelidir:

```
void degistir(int &, int &);
```

Uygulamayı çalıştırdıktan sonra sonuç doğru olacaktır:

```
a = 3
b = 4
a = 3 ve b = 4 degistirilirse a = 4 ve b = 3 olacaktir
a ve b'nin degerleri, degistir void fonksiyonundan geri dondukten sonra a = 4
ve b = 3 olacaktır.
Press any key to continue . . .
```

& işareti, parametrenin türü ile adı arasında herhangi bir yere yerleştirilebilir:

```
int & birinci int& birinci  int &birinci
```

Fonksiyon çağırıcıdan değerleri aktarma ve tersine aktarma mekanizmasına **referanslara göre aktarma** (İng. pass-by-references) denir, argümanlara **ise referansa göre aktarılan argümanlar** (İng. pass-by-reference arguments) denir.

void türünden fonksiyona çağrı, fonksiyonun adından ve argümanlar listesinden oluşan komutla gerçekleştirilir. Örneğin, void türündeki degistir() fonksiyonu şu komutla çağrılacaktır:

```
değiştir(a, b);
```

Bu arada, argümanların sayısı ve sırası, parametrelerin sayısı ve sırası ile uymalıdır. Ancak

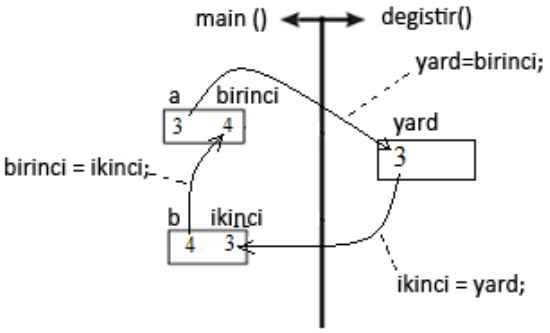
- değere göre aktarılan argümanların türleri, türleri aynı olan veya karşılık gelen değer parametrelerinin türleriyle dönüştürülebilen sabitler, değişkenler veya ifadeler olabilir,
- referansa göre aktarılan argümanların türleri , karşılık gelen referans parametreleriyle aynı türde olan değişkenler (sabitler veya ifadeler olamaz) olmalıdır.

void türünden bir fonksiyonu referans parametreleriyle çağırırken, referans parametreleri için değer parametrelerinde olduğu gibi yeni değişkenler oluşturulmaz. void türünden fonksiyonun yürütülmesi sırasında, referans parametreleri, çağrının karşılık gelen argümanlar için diğer adlar olarak ele alınır. Bu, referans parametrelerine karşılık gelen argümanların adresini atayarak elde edilir. Bu yüzden, referans parametrelerinde yapılan tüm değişiklikler, aslında karşılık gelen argümanlar üzerinde gerçekleştirilir çünkü onlar aynı bellek konumunun farklı adlarıdır.

Şekil 2.5.19'daki çizim, **şekil 2.5.15'**teki uygulamanın **şekil 2.5.18'**deki gibi void türü fonksiyonunun parametrelerinde yapılan değişikliklerle yürütülmesini göstermektedir.

Şekil 2.5.16 ve **şekil 2.5.19**'u karşılaştırarak, değiştir() fonksiyonunun dönüş değerli bir fonksiyon (**şekil 2.5.15**) olarak ve dönüş değeri olmayan fonksiyon olarak yazıldığında (**şekil 2.5.18**) yürütülmesindeki farkı görebiliriz.

Sabit Fonksiyon Parametreleri alt başlığında , dönüş değerli fonksiyonun sabit parametrelerinin fonksiyon içerisinde değiştirilemeyeceğini açıklamıştık. Aynısı , dönüş değeri olmayan fonksiyonlar için de geçerlidir. Parametrenin değer veya referans olmasına bağlı olmadan fonksiyonda değişmesini istemiyorsak const niteleyicisi ile sabit olarak işaretlenmelidir.



Şekil 2.5.19

Çözülmüş Ödevler

Ödev 2.5.13
Üç sayı büyüklüklerine göre sıralansın.

Program ***şekil 2.5.20***'de verilmiştir.

```
1   // Uc sayinin siralanmasi
2   #include <iostream>
3   using namespace std;
4
5   void degistir( double& birinci, double& ikinci) {
6       double yard = birinci;
7       birinci = ikinci;
8       ikinci  = yard;
9   }
10
11  int main() {
12      double a, b, c;
13      cout << "a, b ve c sayilarini girin." << endl;
14      cout << "a = "; cin >> a;
15      cout << "b = "; cin >> b;
16      cout << "c = "; cin >> c;
17      cout << "a << ", " << b << " ve "  << c <<
18           "sayilarinin siralanmis sekli sudur:"
19      if( a > b ) degistir( a, b );
20      if( a > c ) degistir( a, c );
21      if( b > c ) degistir( b, c );
22      cout << a << ", " << b << ", " << c << endl;
```

Şekil 2.5.20

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

Şekil 2.5.20 (devam)

Programı yürüttükten sonra bir çıktı şudur:

```
a, b ve c sayilarini girin.
a = 76.5
b = 65.4
c = 54.3
76.5, 65.4 ve 54.3 sayilarinin siralanmis sekli sudur:
54.3, 65.4, 76.5
Press any key to continue . . .
```

Ödev 2.5.14

İki bilinmeyenle iki lineer denklem sistemi çözülsün

$a_1x + b_1y = c_1$

$a_2x + b_2y = c_2$

Denklem sisteminin çözümü şudur:

$D = a_1b_2 - a_2b_1$; $D_1 = c_1b_2 - c_2b_1$; $D_2 = a_1c_2 - a_2c_1$;

eğer $D \neq$

o zaman

$x = \frac{D_1}{D}$; $y = \frac{D_2}{D}$;

değilse

eğer $D_1=0$ ***VE*** $D_2=0$

o zaman

yazdır „Sistemin ∞ çözümü vardır“;

değilse

yazdır „Sistemin çözümü yoktur.“;

Fonksiyon ve uygulama **şekil 2.5.21**'de verilmiştir.

Uygulamayı çalıştırdıktan sonra bir çıktı şudur:

```
a1, b1 ve c1 katsayilarini girin:
a1 = 3
b1 = -2
c1 = 5
a2, b2 ve c2 katsayilarini girin:
a2 = -4
b2 = 7
c2 = -1

Lineer deklem sisteminin:
     +3x - 2y = +5
     -4x + 7y = -1
Su cozumleri vardir: x = +2.53846, y = +1.30769

Press any key to continue . . .
```

```
1   // 2 bilinmeyeli iki lineer denklem sistemi
2   #include <iostream>
3   using namespace std;
4
5   void lineerDenklemSistemi( int a1, int b1, int c1, int a2, int b2,
6                              int c2, double& x, double& y ) {
7       int D = a1 * b2 - a2 * b1;
8       int D1 = c1 * b2 - c2 * b1;
9       int D2 = a1 * c2 - a2 * c1;
10      if( D != 0 ) {
11          x = ( double ) D1 / D;
12          y = ( double ) D2 / D;
13          cout<<"Su cozumleri vardir: x = " << x << ", y = "<< y << endl;
14      }
15      else {
16          if( D1 == 0 && D2 == 0 )
17              cout << " sonsuz cok cozumu vardir ." << endl;
18          else
19              cout << "celiskilidir ve cozumu yoktur." << endl;
20      }
21  }
22
23  int main() {
24      int a1, b1, c1, a2, b2, c2;
25      double x, y;
26      cout << "a1, b1 ve c1 katsayilarini girin: \n";
27      cout << "a1 = ";    cin >> a1;
28      cout << "b1 = ";    cin >> b1;
29      cout << "c1 = ";    cin >> c1;
30      cout << "a2, b2 ve c2 katsayilarini girin: \n";
31      cout << "a2 = ";    cin >> a2;
32      cout << "b2 = ";    cin >> b2;
33      cout << "c2 = ";    cin >> c2;
34      cout << "\nLineer deklem sisteminin" << endl;
35      cout << showpos << internal;
36      cout << "\t" << a1 << "x " << b1 << "y = " << c1 << endl;
37      cout << "\t" << a2 << "x " << b2 << "y = " << c2 << endl;
38      lineerDenklemSistemi( a1, b1, c1, a2, b2, c2, x, y );
39
40      cout << endl;
41      system( "Color 17" );
42      system( "pause" );
43      return 0;
44  }
```

Şekil 2.5.21

Alıştırma Ödevleri

Aşağıdaki ödevler için void türü fonksiyonlar yazılsın:

1. x ve y sayılarının kuvetleri x^y ve y^x hesaplanasın.
2. ve kutupsal koordinatlarını, aşağıdaki formüllere göre Kartezyen koordinatları x ve y'ye dönüştürülsün: · =ρcosφ,y=ρsinφ.
3. $ax^2+bx+c=0$ ikinci dereceden denklem çözülsün.
4. Üç sayı büyüklüğe göre sıralansın.
5. Verilen 5 basamaklı doğal sayının rakamlarından oluşturulabilecek en büyük ve en küçük sayı bulunsun.
6. Bir kişinin yaşını bugünkü gün ile doğum günü arasındaki farkı olarak bulunsun. Tarihler gün, ay ve yıl olarak ifade edilsin.
7. Girilen n sayının aritmetik, geometrik ve harmonik ortalaması hesaplansın.
8. n sayı girilsin ve bunların en büyüğü ve en küçüğü bulunsun.
9. Bir ondalık sayıyı ikili sayıya dönüştürerek, fonksiyon tüm parçanın ve ondalık bölümün ikili eşdeğerini ayrı ayrı döndürsün.
10. n = 5 için Pascal üçgeni oluşturulsun.

```
                    1                   n = 0
                1       1               n = 1
            1       2       1           n = 2
        1       3       3       1       n = 3
    1       4       6       4       1   n = 4
1       5       10      10      5       1 n = 5
```

Fonksiyonda Fonksiyon Çağırmak

Bir fonksiyon başka bir fonksiyonda çağrılabilir. Bunu aşağıdaki örnekle gösterceğiz.

Örnek 2.5.6
Verilen 5 sayıdan en büyüğü bulunsun

Ödevi çözmek için ***şekil 2.5.8'***deki iki sayıdan büyük olanı bulmak için dahaBuyukSayi() fonksiyonunu kullanacağız, ***şekil 2.5.9***'daki programı ise ucSayidanEnBuyugu( ) adıyla üç sayıdan en büyüğünü bulmak için fonksiyon olarak yazacağız. Ardından bu fonksiyonları aşağıdaki şekilde çağırarak 5 sayıdan en büyüğünü bulacağız:

```
dahaBuyukSayi(dahaBuyukSayi(a, b), ucSayidanEnBuyugu(c, d, e));
```

Not: Bu iki fonksiyona yapılan çağrıları birleştirerek, herhangi sayılardan en büyüğü bulunabilir.

Program **şekil 2.5.22'd**e verilmiştir

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

int dahaBuyukSayi( int sayi1, int sayi2 );
int ucSayidanEnBuyugu( int sayi1, int sayi2, int sayi3 );

int main() {
    int a, b, c, d, e, enBuyuk;
    cout << "Bes tamsayi girin : \n";
    cout << "Birinci sayi: ";   cin >> a;
    cout << "İkinci sayi: ";  cin >> b;
    cout << "Ucuncu sayi: ";  cin >> c;
    cout << "Dorduncu sayi: ";    cin >> d;
    cout << "Besinci sayi: "; cin >> e;

    enBuyuk = dahaBuyukSayi( dahaBuyukSayi ( a, b ),
                        ucSayidanEnBuyugu( c, d, e ) );
    cout <<a <<", " <<b <<", " <<c << ", " << d << " i   " <<e
           << "sayilarindan en buyugu sudur:"
        << enBuyuk << endl;

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

int dahaBuyukSayi( int sayi1, int sayi2 ) {
    int dahaBuyuk;
    if( sayi1 > sayi2 )     // sayi1 sayi2'den daha buyukse,
      dahaBuyuk = sayi1;    // o zaman dahaBuyuk sayi1'dir,
    else                    // degilse
      dahaBuyuk = sayi2;    // dahaBuyuk sayi2'dir.
    return dahaBuyuk;
}

int ucSayidanEnBuyugu( int sayi1, int sayi2, int sayi3 ) {
    int p, n;
    p = dahaBuyukSayi( sayi1, sayi2 );    // Fonksiyon cagrisi
    n = dahaBuyukSayi( p, sayi3 );        // Fonksiyon cagrisi
    return n;
}
```

Şekil 2.5.22

Programın çalıştırılmsıyls bir sonuç şudur:

```
Bes tamsayi girin:
Birinci sayi: 7
İkinci sayi: 3
Ucuncu sayi: 9
Dorduncu sayi: 2
Besinci sayi: 5
7, 3, 9, 2 ve 5 sayilarindan en buyugu sudur: 9

Press any key to continue . . .
```

Global ve Yerel Değişkenler

Programlarda ve alt programlarda (fonksiyonlarda) oluşturulan değişkenler, **erişim alanı** (İng. scope of access) olarak da adlandırılan belirli bir **görünürlük alanına** (İng. scope of visibility) sahiptir. Değişkenin görünürlük alanı, değişkene erişilebildiği, yani işlemlerde kullanılabileceği alan olarak kabul edilir.

Bazı değişkenler uygulamanın tamamında, bazıları fonksiyonun tamamında, bazıları ise sadece fonksiyonun bir kısmında kullanılabilir.

Görünürlük alanına göre değişkenler şöyle olabilir:

- **Global (Genel) değişkenler**.
- **Yerel değişkenler**.

Global değişkenlerin görünürlük alanı, yani bunlara erişim, uygulamanın tamamı, yani ana fonksiyon ve ana fonksiyon ile aynı dosyadaki tüm kullanıcı tanımlı fonksiyonlardır.

Bir uygulamadaki kullanıcı tanımlı fonksiyonların adları, global (genel) değişkenler olarak kabul edilir ve bu nedenle, bir fonksiyon uygulamadaki diğer herhangi bir fonkisyonda çağrılabilir. (C++'da fonksiyon başka bir fonksiyonun gövdesinde tanımlanamaz).

Yerel değişkenler şu erişim kapsamına sahip olabilir: sadece tanımlandıkları fonskiyon gövdesi, sadece fonksiyonun başlığı veya sadece parantezler {} arasındaki blok. Görünürlük, yerel değişkenin bildirilmesiyle başlar.

Fonksiyonun parametreleri, fonksiyonun tamamında yerel değişkenler olarak ele alınır. Ayrıca, bir fonksiyondaki yerel değişkenler tüm iç içe komutlarda görünür, bir komutun gövdesinde bildirilen yerel değişkenler ise, gövdenin dışında görünmezler.

Bir değişken için bir blokta veya tüm fonksiyonda **gizli olduğunu** (başka bir değişken tarafından kaplanmıştır) diyoruz, eğer o blokta veya o fonksiyonda görünmüyorsa (kullanılamazsa). Bunun anlamı, bir değişkenin görünürlük alanında (örneğin, int x = 1;) aynı ada sahip (int x = 2;) bir değişkenin bildirildiği (komutta) bir blok varsa, bu blokta sadece 2 değerine sahip x değişkeni görünür olacaktır.

Örnek 2.5.7

Bu örnekle değişkenin bir uygulamadaki görünürlüğünü göstereceğiz.

Şekil 2.5.23'teki programda, buYil değişkeni global değişken olarak bildirilmiştir, gecenYil ve gelecekYil değişkenleri ise, main() ve benimFonksiyonum() fonksiyonlarında yerel değişkenler olarak bildirilmiştir. buYil global değişkenine hem main() ana gonksiyonu hem de benimFunkciyonum() fonksiyonu tarafından erişilebilir. gecenYil yerel değişkenine sadece main() tarafından erişilir, benimFonksiyonum() tarafından erişilmez, gelecekYıl değişkenine ise sadece benimFonksiyonum() tarafından erişilir, main() tarafından erişilmez.

```
1   #include <iostream>
2   using namespace std;
3
4   void benimFonksiyonum();    // Prototip
5
6   int buYil = 2020;           // Global degisken
7
8   int main() {
9
10      int gecenYil = 2019;     // Yerel degisken
11      cout << "\tmain ( ) ana fonksiyondan yazdiriyorum " << endl;
12      cout << "Bu yil " << buYil << "yilidir, gecen yil ise " << gecenYil << endl;
13      benimFonksiyonum();
14      // 'gelecekYil' degiskenine main () fonksiyonundan erisilemez,
15      // cunku benimFonksiyonum ( ) fonksiyonda yerel degiskendir
16
17      cout << endl;
18      system( "Color 17" );
19      system( "pause" );
20      return 0;
21  }
22
23  void benimFonksiyonum () {
24      int gelecekYil = 2021;; // Yerel degisken
25      cout << "\n\tbenimFonksiyonum ( ) fonksiyonundan yazdiriyorum " << endl;
26      cout << "Bu yil " << buYil << " yilidir, gelecek yil ise "
27          << gelecek << endl;
28      int buYil = 2030;
29      cout << "Buradan fonksiyonun sonuna kadar 'buYil' global degiskeni "
30          << "\n'buYil' yerel degiskeni tarafindan ortulmustur " << endl;
```

Şekil 2.5.23

```
31      cout << "Bu yil " << buYil << "yilidir, gelecek yil ise"
32          << gelecekYil<< "yili olacaktir " << endl;
33
34      cout << "Yine de, ortulmus olan 'buYil' global degiskenine erisim"
35          << "\n:: operatoru ile mumkundur" << "\n Bu yil  "
36          << ::buYil  <<  " yilidir "  << endl;
37      // 'gecenYil' degiskenine benimFonksiyonum  () fonksiyonundan erisilemez,
38      // cunku main ( ) fonksiyonunda yerel degiskendir.
39  }
```

Şekil 2.5.23 (devam)

Yukarıdaki örnekte, 2020 değerine sahip buYil global değişkeni, 2030 değerine sahip yerel değişken olarak yeniden bildirildiği için benimFonksiyonum() içinde örtülmüştür (gizlenmiştir). Tüm bunları program çıktısından açıkça görebiliriz:

```
        main ( ) ana fonksiyondan yazdiriyorum
Bu yil 2020 yilidir, gecen yil ise 2019 yiliydi

        benimFonksiyonum ( ) fonksiyonundan  yazdiriyorum
Bu yil 2020 yilidir, gelecek yil 2021 yili olacaktir
Buradan fonksiyonun sonuna kadar 'buYil' global degiskeni
'buYil' yerel degiskeni tarafindan ortulmustur
Bu yil 2030 yilidir, gelecek yil ise 2021 yili olacaktir
Yine de, ortulmus olan 'buYil' global degiskenine erisim
:: operatoru ile mumkundur
 yil 2020 yilidir

Press any key to continue . . .
```

Aynı adlı yerel değişken buYil (= 2030) ile benimFonksiyonum() fonksiyonunda kapsanan global değişken buYil'a (= 2020), işareti :: olan **görünürlük kapsamı çözümleme operatörü** (İng. scope resolution operator) ile erişilmesi ilginçtir.

Bellekte bir değişkenin oluşturulması anından (bildirim komutunun yürütülmesinden hemen sonra) ortadan kaybolmasına (blok sonu, fonksiyon sonu veya uygulamanın sonu) kadar geçen süreye **değişkenin yaşam süresi** (İng. lifetime) denir.

Global değişkenler, uygulamanın çalıştığı süre boyunca yaşar.

Yerel değişkenler, oluşturma anından, tüm fonksiyon veya yalnızca parantez {} arasındaki bir blok olabilen görünürlük alanın sonuna kadar yaşar.

Değişkenlerin görünürlük alanı için aşağıdaki kurallar geçerlidir:

- Yerel değişkenler görünürlük alanı dışında kullanılamaz.
- Global değişkenler uygulama boyunca kullanılabilir
- Bir blokta bildirilen değişkenler (fonksiyon, komut gövdesi – for, while, do-while) sadece blokta kullanılabilir.

- Bir fonksiyonda bildirilen değişken başka bir fonksiyonda kullanılamaz.
- Bir değişken, görünürlük alanının bir bölümünde gizlenebilir, eğer o bölümde tekrar aynı adla bildirilmişse.
- Aynı adlı iki fonksiyon, sadece farklı argümanlar listelerine sahiplerse, aynı kapsama sahip olabilir.

Statik Değişkenler

Değişkenler **statik** (İng. static) de olabilir. Bu tür değişkenler **static** kelimesi ile bildirilirler.

Örneğin:

```
static int sayi  = 7;
static double tutar;
tutar = 123.45;
```

Yerel statik değişkenler, global değişkenler gibi davranır. Onlar bildirimden uygulama tamamlanana kadar yaşarlar. Bu arada, bildirim sırasında veya yerel değişken oldukları fonksiyona ilk çağrıda değer atanırken sadece bir kez başlatılırlar. Bildirildikleri fonksiyonu yürüttükten ve işlemi çağırıcıya (main() veya başka bir fonksiyona) döndürdükten sonra, statik yerel değişkenler aldıkları değeri tutar, yani statik olmayan yerel değişkenler gibi yok edilmezler. Fonksiyona her sonraki çağrıda, önceki çağrıda elde edilen duruma (değere) sahip olurlar.

Global değişkenler ve yerel statik değişkenler, tanımlarının ilk kez çalıştırılmasıyla yaşamaya başlar.

Örnek 2.5.8

Şekil 2.5.24'teki uygulamada f() fonksiyonu 5 kez çağrılıyor.

f() fonksiyonuna yapılan ilk çağrıda, statik değişken baslangic 0 değeriyle başlatılır, statik olmayan sayac değişkeni ise 100 değerine başlatılır. Arttırma komutları bu değişkenler 1 ve 101 değerini alır.

f() fonksiyonuna yapılan ikinci çağrıda, baslangic statik değişkeni yeniden başlatılmaz (bildirim ve başlatma komutu yeniden yürütülmesine rağmen), ancak (statik olduğu için) fonksiyonun ilk çağrısında aldığı 1 değerini korur. Bu yüzden, artırma komutuyla 2 değerini alır. Statik olmayan değişken olan sayac, 100 değerine yeniden başlatılır ve ardından 101'e arttırılır.

Aynısı, f() fonksiyonuna yapılan her çağrıda tekrarlanır.

Bu, statik değişkeninin sadece bir kez başlatıldığı (f() fonksiyonuna yapılan ilk çağrıda) ve uygulama sona erene kadar global bir değişken olarak yaşadığı anlamına gelir.

```
1   #include <iostream>
2   using namespace std;
3
4   void f();
5
6   int main() {
7       for( int k = 1; k <= 5; k++ )
8           f();
9
10      cout << endl;
11      system( "Color 17" );
12      system( "pause" );
13      return 0;
14  }
15
16  void f() {
17      static int baslangic = 0; // baslangic statik degiskenidir
18      int sayac = 100;
19
20      baslangic += 1;
21      sayac += 1;
22      cout << "baslangic =" << baslangic << "sayac =" << sayac << endl;
23  }
```

Şekil 2.5.24

Uygulanmanın yürütülmesiyle şu çıktı elde edilir

```
baslangic=1   sayac=101
baslangic=2   sayac=101
baslangic=3   sayac=101
baslangic=4   sayac=101
baslangic=5   sayac=101

Press any key to continue . . .
```

baslangic değişkeni statik değilse, çıktı şu olacaktır:

```
baslangic=1   sayac=101
baslangic=1   sayac=101
baslangic=1   sayac=101
baslangic=1   sayac=101
baslangic=1   sayac=101

Press any key to continue . . .
```

Global ve statik değişkenlerin kullanımı, uygulamadaki tüm fonksiyonlara erişilebilir oldukları için çok dikkatli olmalıdır. Bir fonksiyonda içeriklerini değiştirmek ,diğerinde hataya neden olabilir.

Yerleşik Fonksiyonlar

Bir uygulamayı çalıştırırken, içindeki tüm fonksiyonlar için bellekte bir kopya oluşturulur. Her uygulamanın yürütülmesi main() fonksiyonu ile başladığı bilinmektedir. İçinde başka bir fonksiyon çağrılırsa, eylem çağrı noktasında durur ve çağrılan fonksiyonun hafızada bulunduğu konuma atlanır. Fonksiyon yürütüldükten sonra, eylem main() fonksiyonuna döner ve fonskiyon çağrısı komutundan sonraki komutla devam eder.

Bir fonksiyonu çağırma ve çağırıcıya geri dönme mekanizmasının (bu, main() dışında başka bir fonksiyon da olabilir) çalışması belirli bir zaman alır. Çağrılan fonksiyonun küçük olması durumunda, fonksiyonu çağırma ve ondan geri dönüş süresi, fonksiyonun yürütme süresinden daha uzun olabilir. Küçük fonksiyon daha sık çağrılırsa, örneğin tekrarlama komutunda, zaman kaybı daha belirgindir.

Böyle durumlarda, fonksiyonun derlenmesi sırasında, fonskiyonun çağırıcının kodunda (örneğin main() içinde) yerleştirilmesi daha pratiktir. (C++'daki kütüphane fonksiyonlarının yerleşik fonksiyonlar olduğundan söylemiştik).

Bu amaçla, C++, çeviriciye çağrıldığı yerlerde onu çağıran programlara dahil etmesini söylemek için fonksiyon tanımının önüne yerleştirilen **inline** nitelendiricisi kullanır.

Örnek 2.5.9

Şekil 2.5.25'teki programda, oyunBonusu() fonksiyonu küçüktür ve birden çok kez yürütülür, bu yüzden yerleşik fonksiyon olarak nitelendirilebilir.

```
1   #include <iostream>
2   #include <string>
3   #include <ctime>
4   using namespace std;
5
6   inline int oyunBunusu( string adSoyad ) {
7       srand( time( 0 ) );
8       int bonus = 1 + rand() % 100;
9       return bonus;
10  }
11
12  int main() {
13      string adSoyad;
14      for( int i = 1; i <= 3; i++ ) {
15          cout << "Adinizi ve soyadinizi girin:";
16          getline( cin, adSoyad );
```

Şekil 2.5.25

```
17          cout << "\tSayin" << adSoyad
18              << " sizin bu oyun icin bonusunuz "
19              << igraBonus( adSoyad ) << endl;
20      }
21
22      cout << endl;
23      system( "Color 17" );
24      system( "pause" );
25      return 0;
26  }
```

Şekil 2.5.25 (devam)

Programın bir çıktısı devamda verilmiştir:

```
Adinizi ve soyadinizi girin: Aleksandar Makedonski
     Sayin Aleksandar Makedonski sizin bu oyun icin bonusunuz 41'dir
Adinizi ve soyadinizi girin: Car Samoil
     Sayin Car Samoil sizin bu oyun icin bonusunuz 67'dir
Adinizi ve soyadinizi girin: Goce Delcev
     Sayin Goce Delcev sizin bu oyun icin bonusunuz 90'dir

Press any key to continue . . .
```

Alıştırma Ödevleri

Aşağıdaki ödevler için uygulamalar ve fonksiyonlar yazılsın:

1. $S_n = a_1 + (a_1 + a_2) + (a_1 + a_2 + a_3) + ... + (a_1 + a_2 + ... + a_n)$ toplamı hesaplansın. Toplamı hesaplamak için $S_k = a_1 + a_2 + ... + a_k$ fonksiyon kullanılsın.
2. k! hesaplama fonksiyonunu kullanarak $S_n = \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots + \frac{1}{n!}$, toplamı hesaplansın!.
3. n doğal sayısının rakamlarının toplamını ve sayısını hesaplamak için ayrı ayrı fonksiyonlar yazılsın.
4. n'den küçük asal sayılar bulunsun. Bir doğal sayının asal olup olmadığını belirleyen fonksiyon yazılsın.
5. Gerçek ondalık sayıyı ikili sayıya dönüştürmek için void türünden fonksiyon yazılsın.
6. ikinci dereceden denklemi çözmek için void türünden fonksiyon yazılsın. $ax^2 + bx + c = 0$.
7. Ondalık sayı n'nin ikilik, sekizlik ve on altılık kaydını bulmak için void türünden fonksiyon yazılsın.
8. . n doğal sayısının rakamlarından oluşturulabilecek en büyük ve en küçük sayıyı bulan void türünden fonksiyon yazılsın.

9. Rakamlarının faktöriyelleri toplamına eşit olan üç basamaklı tüm doğal sayılar bulunsun. Bir sayının faktöriyelini hesaplamak için ayrı fonksiyon kullanılsın.
10. m'den n'ye kadar aralıktaki tüm dost sayılar bulunsun. (Birinci sayının bölenlerinin toplamı ikinci sayıya eşitse ve ikinci sayının bölenlerinin toplamı birinci sayıya eşitse iki sayıya dost sayı denir). Sayının bölenlerinin toplamı için ayrı fonksiyon yazılsın.

Bilgiyi Kontrol Etme Soruları

1. C++'da fonksiyonlar içeren kütüphanenin adı nedir?
2. C++ kütüphanesinden bir başlık dosyası belirtin.
3. Başlık dosyasından bir fonksiyon kullanmak istiyorsak programa neyin dahil edilmesi gerekir?
4. Klavye aracılığıyla girilen verilerin okunması için fonksiyonlar hangi C++ kütüphanesinde bulunur?
5. x^n'yi ve $\sqrt{x}$'i hesaplamak için hangi kütüphane fonksiyonları kullanılır?
6. C++'da rastgele sayı üretme fonksiyonu hangisidir?
7. Bir kelimenin tüm harflerini büyük harfe çeviren fonksiyon hangisidir?
8. Karakterin sayı olup olmadığını ve harf olup olmadığını kontrol etmek için hangi fonksiyon kullanılır?
9. Bir değişkenin kapladığı bellek boyutu hangi operatör ile belirlenir?
10. Algoritmaların bölümlerini alt algoritmalarda yazmanın avantajı nedir?
11. Alt algoritmalar nasıl olabilir? Onların ana özelliği nedir?
12. Fonksiyonel alt algoritmalar ve prosedürel alt algoritmalar nasıl çağrılır?
13. Aşağıdaki adım fonksiyon alt algoritmasında yazılmışsa

    döndür ←100;

    ne anlama geliyor?
14. Bir veya birden fazla değer döndürmesine göre C++'da hangi fonksiyon türleri vardır?
15. Dönüş değerli fonksiyonun genel tanımını yazın.
16. Dönüş değerli fonksiyon bildirimi nasıl yapılır? Genel biçimi yazın.
17. Fonksiyon türü nedir?
18. Fonksiyon prototipinin rolü nedir?
19. Fonksiyon imzası neyden oluşmaktadır?
20. main() fonksiyonundan önce tanımlanan fonksiyonlar için fonksiyon prototipinin belirtilmesi gerekir mi?
21. Fonksiyon prototipleri için bir kaç örnek verin.
22. Fonksiyon imzası için bir kaç örnek verin.

23. Dönüş değeri olan fonksiyonları çağırma yolları hangileridir?
24. Dönüş değerli fonksiyon çağrısında argümanların sayısı, fonksiyon tanımındaki parametrelerin sayısından farklıysa ne olur?
25. Fonksiyon çağrısındaki argümanların türü, fonksiyon tanımındaki karşılık gelen parametrenin türüyle eşleşmezse ne olur?
26. Aşağıdaki program bölümünde hata nerededir:

```
int f(void) {
   cout << "f ( ) fonksiyonunu cagirin ";
   int g(void) {
          cout << "g()fonksiyonunu cagirin ";
   }
}
```

27. Girilen kenar için, küpün hacmini hesaplayan dönüş değerli fonksiyon yazın.
28. Dönüş değerli fonksiyon cout yazdırma komutunda çağrılabilir mi?
29. Dönüş değeri olmayan fonksiyonların nasıl parametreleri olabilir?
30. C++'da değer döndürmeyen fonksiyonlar hangi türdendir?
31. Dönüş değeri olmayan fonksiyonlar return komutunu içerebilir mi?
32. Dönüş değerli fonksiyonlar hangi yollarla ve dönüş değeri olmayan fonksiyonlar hangi yollarla çağrılabilir?
33. Fonksiyona argumanların değere göre ve referansa göre nasıl aktarıldığını açıklayın.
34. Referans parametreler nasıl işaretlenir?
35. Dönüş değeri olmayan fonksiyon prototipinde referans parametrelerin işaretlenmesi gerekiyor mu?
36. Referansa göre aktarılan argüman ile karşılık gelen referans parametresi arasındaki fark nedir?
37. Dönüş değerli fonksiyonun referans parametreleri olabilir mi?
38. Hangi durumlarda dönüş değerli fonksiyonlar ve hangi durumlarda dönüş değeri olmayan fonksiyonlar kullanılır?
39. Aşağıdaki dönüş değeri olmayan fonksiyonda hata nerede yapılmış?

```
void P(int a) {
   int b;
   b = a++;
   return b;
}
```

40. Bir fonksiyondaki sabit parametrenin değerini değiştirmeye çalışırsak ne olur?
41. Bir fonksiyon başka bir fonksiyonu çağırabilir mi?
42. Bir fonksiyon içinde başka bir fonksiyon tanımlanabilir mi?

43. Global (genel) ve yerel değişkenler nedir?
44. Global değişkenlerin ve yerel değişkenlerden görünürlük alanı nedir?
45. Bir uygulamadaki fonksiyon adları nasıl değişkendir, global mi yerel mi?
46. Fonksiyon parametrelerinin görünürlük alanı nedir?
47. Global bir değişkenin fonksiyonda nasıl gizlenebileceğini (erişilemeyeceğini) açıklayın? Görülmesinin bir yolu var mıdır?
48. Değişkenin yaşam süresi nedir?
49. Global değişkenin ile statik değişkenin davranması arasında fark nedir?
50. Yerleşik fonksiyonlar hangi sözcükle belirtilir?

Ödevler

Tekrarlama Algoritmik Kontrol Yapıları ve Tekrarlama Kontrol Komutları noktasından tüm örnekler ve çözülen ödevler fonksiyonlar kullanarak yeniden çözülsün.

Aşağıdaki ödevler için fonksiyonlar kullanarak programlar yazılsın:

1. Bir cümlede kaç tane büyük harf, kaç tane küçük harf ve kaç tane noktalama işareti olduğu bulunsun.
2. (x, y) noktasının Kartezyen koordinatları aşağıdaki formüle göre kutupsal koordinatlara (,)dönüşsün: $\overline{\rho} = \sqrt{x^2 + y^2}$, $\varphi_x = arctg\,\frac{y}{}$.
3. İki sayı için EBOB bulunsun.
4. Tam negatif ondalık sayının ikili eşdeğeri bulunsun.
5. n sayı girerek bunların en küçüğünü ve en büyüğünü bulan fonksiyon yazılsın.
6. Verilen n doğal sayısından daha büyük ve daha küçük asal sayı bulunsun.
7. Saat (0 – 23), dakika (0 – 59) ve saniye (0 – 59) ile ifade edilen iki zaman süresini toplama fonksiyonu yazılsın.
8. Bir kişinin yaşını (bugünün tarihi ile doğum tarihi arasındaki farkı) hesaplamak için void türünden fonksiyon yazılsın. Tarihler tam sayı değişkenleri olarak girilsin: gün (1 – 31), ay (1 – 12), yıl (1 – 2345).

- **Algoritmik kontrol (yönetim) yapıları**, algoritmanın çalışma sırasını kontrol etmek için kullanılır.
- **Sıralı algoritmik kontrol yapısı veya dizgisi**, algoritmik adımların belirtildiği sırayla yürütüldüğü algoritmik yapıdır.
- Sıralı algoritmik kontrol yapısı **başlangıç-bitiş** olarak adlandırılır.
- **İki olasılıklı seçim için algoritmik kontrol yapısı**, bir koşula bağlı olarak (mantıksal ifade) algoritmadaki işlemin iki olası devam yönünden birini seçmek için bir yapıdır.
- **Algoritmik blok, başlangıç ve bitiş** sözcükleriyle belirtilen birden fazla adımdan oluşan bloktur.
- **eğer–o zaman–değilse**, iki olasılıklı seçim yapmak için algoritmik kontrol yapısıdır.
- **eğer–o zaman** iki olasılıklı seçim için algoritmik kontrol yapısı olup, olasılıklardan birinin yürütme adımları yoktur.
- **if, eğer-o zaman** algoritmik kontrol yapısını gerçekleştirmek için C++'da kontrol komutudur.
- **if-else, eğer-o zaman-değilse** algoritmik kontrol yapısını gerçekleştirmek için C++'da kontrol komutudur.
- **?:** koşullu operatör olarak adlandırılır.
- Çok olasılıklı seçim için algoritmik kontrol yapısı, bazı verilerin veya bazı aritmetik ifadelerin değerine bağlı olarak, algoritmadaki eylemin birkaç olası devam durumundan birinin seçildiği yapıdır.
- **durum,** çok olasılıklı seçim yapmak için algoritmik kontrol yapısıdır.
- **switch**, birden çok durum olasılıklı seçim için algoritmik kontrol yapısını gerçekleştirmek için C++'da kontrol komutudur.
- **Döngü,** bir grup algoritmik adımın bir yürütülmesidir.
- **Döngüsel tekrarlama**, aynı algoritmik adım grubunun birden çok kez yürütülmesidir.
- için-kadar-adım, döngülerin baslangic değerinden bitis değerine kadar adım değeri için sayıldığı algoritmik kontrol yapısıdır.
- **için-kadar-adım** algoritmik kontrol yapısında for adım değeri +1 ise, yapı **için–arttır–kadar** olarak adlandırılır.
- **için-kadar-adım** algoritmik kontrol yapısında for adım değeri - 1 ise, **yapı için–azalt–kadar** olarak adlandırılır.
- **for,** sayacı 1 için artırarak, sayacı 1 için azaltarak ve sayacı belirli bir değer için artırarak/azaltarak **için-arttır-kadar, için-azalt-kadar** ve **için-kadar-adım** kontrol yapılarını uygulamak için C++'da kontrol komutudur.

- ++ arttırma operatörüdür.
    - azaltma operatörüdür.
- **iken-yürütür,** her döngünün başlangıcından önce kontrol edilen ve bazı koşul geçerli iken döngülerin tekrarını gerçekleştirildiği algoritmik kontrol yapısıdır.
- **while, iken-yürütür** algoritmik kontrol yapısını gerçekleştirmek için C++'da kontrol komutudur.
- **yürütür–iken**, her döngünün sonunda kontrol edilen ve bazı koşulu geçerli iken döngülerin tekrarını gerçekleştirildiği algoritmik kontrol yapısıdır.
- **do-while, yürütür–iken** algoritmik kontrol yapısını gerçekleştirmek için C++'da kontrol komutudur.
- **devam**, döngünün sonunda atlamak ve bir sonraki döngü ile devam etmek için algoritmik kontrol yapısıdır.
- **continue, devam** algoritmik kontrol yapısının gerçekleştirilmesi için C++'da kontrol komutudur.
- **kesinti**, kontrol yapısının sonuna atlamak ve tekrarlamayı durdurmak için algoritmik kontrol yapısıdır.
- **break**, kesinti algoritmik kontrol yapısının gerçekleştirilmesi için C++'da kontrol komutudur.
- **çıkış**, algoritmanın sonuna atlama için algoritmik kontrol yapısıdır.
- **exit(), çıkış** algoritmik kontrol yapısının gerçekleştirilmesi için C++'da kontrol komutudur (fonksiyonudur).
- **atlayış**, algoritmada rastgele bir adıma atlamak için algoritmik kontrol yapısıdır.
- **goto, atlayış** algoritmik kontrol yapısının gerçekleştirilmesi için C++'da kontrol komutudur.
- **C++ standart kütüphanesi,** matematiksel fonksiyonlar, girdi ve çıktı fonksiyonlar, karakterle işlemleri, dizelerle işlemleri vb. içeren kütüphanedir.
- **Dönüş değerli fonksiyonlar**, yürütüldükten sonra çağrıldıkları yere değer döndüren fonksiyonlardır.
- **Dönüş değeri olmayan fonksiyonlar**, yürütüldükten sonra çağrıldıkları yere değer döndürmeyen fonksiyonlardır.
- **Yukarıdan aşağıya programlama**, bir ödevi alt ödevler adı verilen daha basit ödevlere bölmek için kullanılan tekniktir.
- **Modüler programlama**, alt ödevler ve onların sıralı yürütülmesi için programlar (modüller) oluşturmak için bir tekniktir.
- **Alt algoritma**, (kodlandıktan sonra) tek başına yürütülemeyen algoritmadır.

- **Fonksiyonlar** olarak da adlandırılan **fonksiyonel alt algoritmaların** sadece ***bir çıktı sonucu*** vardır.
- **Prosedürler** olarak da adlandırılan **prosedürel alt algoritmaların *birden fazla çıktı sonucu*** vardır.
- **Parametreler (biçimsel argümanlar)** alt algoritmaların tanımlanmasında kullanılır.
- **Argümanlar (bağımsız değişkenler) (gerçek argümanlar),** alt algoritma çağrılırken kullanılan değişkenlerdir.
- **Girdi parametreleri (girdi biçimsel argümanlar**), değerlerin çağırıcıdan (algoritma veya alt algoritma) alt algoritmaya aktarıldığı parametrelerdir. işareti ile işaretlenir.
- **Çıktı parametreleri (çıktı biçimsel argümanlar),** değerlerin alt algoritmadan çağırıcıya (algoritma veya alt algoritma) aktarıldığı parametrelerdir. işareti ile işaretlenir.
- **Girdi-çıktı parametreleri (girdi-çıktı biçimsel argümanlar),** değerlerin çağırıcıdan (algoritma veya alt algoritma) alt algoritmaya ve ters yönde aktarıldığı parametrelerdir. ↕ işareti ile işaretlenir.
- **döndür**, fonksiyonel alt algoritma sonucunun döndürüldüğü adımdır.
- **Parametreler listesi**, alt algoritma tanımının başlığındaki listedir.
- **Argümanlar listesi**, alt algoritma çağrısındaki listedir.
- **Fonksiyon** adı verilen **fonksiyonel alt programı,** programlama dilinde fonksiyonel alt algoritması için yazılmış programdır.
- **Kullanıcı tarafından tanımlı fonksiyonlar veya sadece kullanıcı fonksiyonlar**, programcılar tarafından yazılan fonksiyonlardır.
- C++'da **fonksiyon tanımı**, başlıktan (dönüş değerinin türü, fonksiyonun adı ve parametreler listesinden oluşur) ve (parantez içinde) gövdeden oluşur.
- C++'da dönüş değerli **fonksiyonun bildirimi**, (dönen değerin) türü , (fonksiyonun) adı ve parametreler listesinden oluşur.
- **Fonksiyon prototipi**, fonksiyon bildirimi olarak adlandırılır ve tür, ad ve parametreler listesinden oluşur.
- **Fonksiyon imzası**, fonksiyon adından ve parametreler listesinden oluşur.
- **Sabit parametreler**, fonksiyonda değeri değiştirilemeyen parametrelerdir.
- **void**, C++'da değer döndürmeyen fonksiyon türüdür.
- **Değere göre aktarılan argümanlar**, fonksiyona çağrıyla iletilen, ancak çağırıcıdaki (main() fonksiyonu veya başka bir fonksiyonu) değerleri değişmeyen argümanlardır. Bu mekanizma, argümanların değere göre aktarma olarak adlandırılır.
- **Referanslama**, değişkene başka bir isim vermektir.
- **Referans veya referans değişken**, bir değişkenin başka bir adıdır.

- **Referansa göre aktarılan argümanlar**, karşılık gelen referans parametreleri için başka ad oldukları çağrı ile fonksiyona aktarılan argümanlardır. Bu mekanizma, **argümanları referansa göre aktarma** olarak adlandırılır.
- **Erişim alanı** olarak da adlandırılan **görünürlük alanı**, değişkene erişimin olduğu yani onlarla işlemlerin yapılabildiği alandır.
- **Global (Genel) değişkenler**, tüm uygulama tarafından, yani ana fonksiyon tarafından ve ana fonksiyonla aynı dosyadaki tüm kullanıcı tanımlı fonksiyonlar tarafından erişilebilen değişkenlerdir.
- **Yerel değişkenler**, sadece tanımlandıkları fonksiyonun gövdesini veya sadece fonksiyonun başlığı veya sadece içinde tanımlandıkları bölüm tarafından erişim alanı olan değişkenlerdir.
- Bir blokta veya fonksiyonun tamamında bulunan gizli değişken (başka bir değişkenle kapsanan) ve o blokta veya fonksiyonda görünmeyen (kullanılamayan) değişkendir.
- **:: görünürlük alanını çözme operatördür.**
- **Değişkenin yaşam süresi**, bellekte oluşturulmasından (bildirim komutunun yürütülmesinden hemen sonra) kaybolmasına (blok sonu, fonksiyon sonu veya uygulama sonu) kadar geçen süredir.
- **Statik değişken**, bildirim anından itibaren global bir değişken gibi davranan değişkendir.
- **Yerleşik fonksiyon**, çağırıcıyla (main() veya başka bir fonksiyon) birlikte derlenen ve koduna yerleşen fonksiyondur.
- **inline**, yerleşik fonksiyonlar için bir nitelendiricidir.

Özet

• Yapılandırılmış programlamada algoritmaların akışını tanımlamak için, algoritmik adımların yürütme sırasını yöneten özel algoritmik kontrol (yönetim) yapıları kullanılır.
• Her algoritmanın akışı, aşağıdaki algoritmik kontrol yapıları kullanılarak kontrol edilebilir: **sıralı** veya **dizgesel**, **seçim** veya **seçme** ve **tekrarlama** veya **yineleme**.
• Sıralı algoritmik kontrol yapısında, adımlar belirtildikleri sırayla yürütülür. Bu yapıya **başlangıç-bitiş** denir.
• **eğer-o zaman-değilse,** iki olasılık arasından seçim yapmak için algoritmik kontrol yapısıdır ve C++'da if-else kontrol komutu ile gerçekleştirilir.
• **eğer-o zaman**, olasılıklardan biri çalıştırılabilir birinin hiçbir adım içermediğinde iki olasılık arasından seçim yapmak için algoritmik kontrol yapısıdır ve C++'da if kontrol komutu ile gerçekleştirilir

- If ve if-else kontrol komutları, hem if hem else seçenekleri içinde iç içe yerleştirilebilir.else seçimin içindeki iç içe yerleştirilmiş kontrol komutuna if-else-if adı verilir.
- ?: koşullu operatör, basit if-else komutlarını ifade etmek için kullanılır.
- Algoritmada işleme devam etmek için birden çok olasılık olduğunda, birden fazla **durum** olasılıklı algoritmik kontrol yapısı kullanılır. Olasılıklardan birinin seçimi, bazı verinin veya bazı aritmetik ifadenin değerine bağlı olarak yapılır.
- Birden fazla durum olasılıklı algoritmik kontrol yapısı, C++'da switch kontrol komutu ile gerçekleştirilir..
- Tekrarlama algoritmik kontrol yapıları, döngüsel tekrarlama olarak adlandırılan bir grup algoritmik adımının birden çok kez yürütülmesi gerektiğinde kullanılır. Adımların bir yürütülmesine bir döngü denir.
- Tekrarlama için üç algoritmik kontrol yapısını ayırt ediyoruz: döngü sayımı ile tekrarlama **(için-kadar-adım),** döngünün başında çıkışlı tekrarlama (iken–yürütür) ve döngü sonunda çıkışlı tekrarlama **(yürütür-iken).**
- **için-kadar-adım** algoritmik kontrol yapısında, döngüler özel sayacla, adımın baslangic değerinden bitis değerine kadar adım değeri ile sayılır. Adım değeri +1 ise bu yapı için-**arttır-kadar** olarak adlandırılır, adım değeri –1 ise bu yapı **için-azalt-kadar** olarak adlandırılır.
- **için-kadar-adım (için-arttır-kadar** ve **için-azalt-kadar)** algoritmik kontrol yapısını gerçekleştirmek için, C++'da for kontrol komutu kullanılır.
- for kontrol komutunun başlatma kısmı, koşul kısmı ve güncelleme kısmı vardır.
- C++'daki for kontrol deyimi, birden çok değişkenin başlatılmasına ve güncellenmesine izin verir.
- for kontrol deyiminde, başlatma ve güncelleme bölümü boş olabilir.
- **iken-yürütür** algoritmik kontrol yapısı, C++'da while kontrol komutu ile gerçekleştirilir. while içindeki mantıksal koşul geçerli olduğu sürece döngüler yürütülür. Koşul, her döngünün başlangıcından önce kontrol edilir.
- while kontrol komutu ile ilk döngü başlamadan önce koşul geçerli değilse hiçbir döngü yürütülmeyebilir.
- Tekrarlamayı önceden bilinen bir değerle kontrol edilmesine bitiş kontrollü tekraralama denir.

- **yürütür-iken** algoritmik kontrol yapısı, C++'da do-while kontrol komutu ile gerçekleştirilir.do-while döngüleri, bir mantıksal koşulu geçerli olduğu sürece yürütülür. Koşul, her döngünün tamamlanmasından sonra kontrol edilir.
- do-while komutu ile en az bir döngü yürütülmelidir.
- Tekrarlama kontrol komutlarını iç içe yerleştirirken (for, while, do-while), bir kontrol komutunun tüm komutları diğerinin içinde olmalıdır, yani kontrol komutlarının kesişmesi (birbiriyle örtüşmesi) olmamalıdır.
- Programlama dillerinde dört tür atlama algoritmik kontrol yapısı vardır, onlar da: döngünün sonunda atlama **- devam**, kontrol yapısının sonunda atla **- kesinti**, algoritmanın sonunda atla **- çıkış** ve herhangi bir yere atlama **- atlayış**.
- Atlama algoritmik kontrol yapıları, C++'da şu kontrol komutlarıyla gerçekleştirilir: continue, break, exit() ve goto.
- Yapılandırılmış programlamada goto kontrol komutunun kullanımından kaçınılır.
- Yapılandırılmış programlama iki programlama tekniği kullanır: yukarıdan aşağıya programlama ve modüler programlama. Yukarıdan aşağıya programlama, ödevi alt ödevlere ayırarak yapılır. Her alt ödev için alt algoritma adı verilen ayrı bir algoritma yazılabilir.
- Alt algoritmaların özü, farklı algoritmalarda veya alt algoritmalarda ve aynı algoritma ve/veya alt algoritmada birkaç yerde kullanılabilmesidir.
- Alt algoritmaların kullanımı, parametreler (biçimsel argümanlar) ve argümanlar (gerçek argümanlar) mekanizması aracılığıyla sağlanır.
- Parametreler girdi, çıktı ve girdi-çıktı olabilir.
- Çıktı sonuçlarının sayısına göre, iki tür alt algoritma vardır: fonksiyonel alt algoritmalar (sadece bir sonuç döndüren) ve prosedürel alt algoritmalar (daha fazla sonuç döndürebilen).
- Programlama dillerindeki alt algoritmalar alt programlar ile gerçekleştirilir. Alt programlar fonksiyonel (tek bir değer döndürdüklerinde) ve prosedürel (birden çok değer döndürdüklerinde) olabilir.
- C++'da alt programlar fonksiyonlarla gerçekleştirilir. Fonksiyonel alt programlar, dönüş değeri olan fonksiyonlarla gerçekleştirilir, prosedürel alt programlar ise dönüş değeri olmayan fonksiyonlarla gerçekleştirilir.
- C++'daki fonksiyonlar şunlar olabilir: kütüphane fonksiyonları ve kullanıcı tanımlı fonksiyonlar.
- C++'da dönüş değerli fonksiyon tanımlarken şunlara dikkat edilmelidir:

• Fonksiyonun başlığı ;(noktalı virgül) ile bitmez,
• Parametreler listesi, parametrelerin adlarını ve türlerini de içermelidir,
• Dönüş değeri, bir ifade (değişkenler ve/veya sabitler ve/veya fonksiyonlar) sonucu veya tek bir değişken olabilir ve fonksiyonun türüyle aynı türden olması gerekiyor.

• C++'da fonksiyon bildiririken şunlar bilinmelidir:
  • Fonksiyon türü belirtilmezse, int türü varsayılır.
  • Fonksiyonların adları (uzlaşmaya göre) küçük baş harfle yazılır. İsim birkaç kelimeden oluşuyorsa, sonraki her kelime büyük harfle yazılır.
  • Parametreler listesi, parametre türlerini içermeli ancak adları içermesi zorunlu değildir.
  • Fonksiyonun bildirimi ; (noktalı virgül) işareti ile biter.
• Fonksiyon çağrılırken, fonksiyonun argümanlar listesi ve parametreler listesi şunları içermelidir:
  • aynı sayıda argümanlar ve parametreler,
  • aynı argüman ve parametre sırası,
  • karşılık gelen argümanların ve parametrelerin aynı veya dönüştürülebilir türü.
• C++'da dönüş değerli fonksiyon, adı ve argümanlar listesi veya atama komutu veya başka bir komut veya ifade ile çağrılır.
• C++'da dönüş değeri olmayan fonksiyon, sadece adı ve argümanlar listesiyle çağrılır.
• Ana programdan sonra tanımlanan fonksiyonlar, ana fonksiyon main()'den önce (prototipleri belirtilerek) bildirilir.
• Aynı programda bir fonksiyon sadece bir kez tanımlanabilir.
• Fonksiyon parametreleri, fonksiyondaki değişikliklerden korunmak için const nitelendiricisi ile sabit olarak işaretlenirler.
• C++'da dönüş değeri olmayan fonksiyonlar, void türündedir.
• Dönüş değeri olan fonksiyondan çıkış parametreli return komutu ile yapılır, dönüş değeri olmayan fonksiyondan ise, çalıştırıldıktan sonra otomatik olarak veya parametresiz return komutu ile fonksiyon gövdesinin herhangi bir yerinden çıkış yapılır.
• Fonksiyon çağrılırken, argümlanların değere göre veya referansa göre aktarma mekanizması kullanılır.
• Argümanların değere göre aktarılması sırasında, onlar fonksiyonda değişmezler, yani çağırıcıda (ana fonksiyon main() veya başka bir fonksiyon) döndükten sonra aynı kalırlar.
• Argümanları referansa göre aktarırken, onlar fonksiyonda değiştirilebilirler ve çağırıcıda (ana fonksiyon main() veya başka bir fonksiyon) döndükten sonra değişmiş olabilirler.

- Argümanları referansa göre aktarırken, argümanlar için yeni değişkenler yaratılmaz, ancak çağrı argümanları, fonksiyon referans parametrelerinin diğer adlarını tanımlamaktadır. Referans parametrelerindeki tüm değişiklikler, aslında, aynı bellek konumunda farklı adlar oldukları için karşılık gelen argümanlarda yapılır.
- Fonksiyonda herhangi bir argümanın değişmesini istemiyorsak, değere göre veya referansa göre aktarıldığına bağımsız olarak, const nitelendiricisi ile sabit olarak işaretlenmelidir.
- Dönüş değerli fonksiyonlar, diğer fonksiyonların argümanları olabilir.
- Global değişkenler, uygulama genelinde ve uygulama içinde çağrılan fonksiyonlarda erişilebilir değişkenlerdir.
- Bir uygulamada kullanıcı tanımlı fonksiyonların adları, global değişkenler olarak ele alınır.
- Yerel değişkenler, sadece tanımlandıkları fonksiyonda veya blokta erişilebilir (görünür) ve kullanılabilen değişkenlerdir.
- Yerel değişkenin görünürlüğü, bildirimi ile başlar.
- Bir fonksiyonun parametreleri, fonksiyon boyunca yerel değişkenler olarak ele alınır.
- Bir blokta bildirilen değişkenler (fonksiyon, komut gövdesi – if, switch, for, while, do-while, vb.) yerel değişken oldukları için sadece blokta kullanılabilirler.
- Bir fonksiyonda bildirilen değişken başka bir fonksiyonda kullanılamaz.
- Aynı ada sahip iki fonksiyon, sadece farklı parametre listelerine sahiplerse aynı görünürlük alanına sahip olabilirler.
- Bir uygulama çalıştırılırken, tüm fonksiyonlar için bellekte birer kopya oluşturulur.

MODÜL

C++'DA KARMAŞIK VERİ TÜRLERİ

3

Bu bölümde aşağıdakiler hakkında bilgi edineceksiniz:

- Diziler ve onların programlamadaki önemi.
- Tek boyutlu dizinin bildirimi, başlatılması ve elemanlara değerlerin atanması.
- Aralığa dayalı for komutu ve onun kullanımı.
- iki boyutlu dizinin bildirimi, başlatılması ve değerlerin atanması.
- İki boyutlu dizinin boyutlarının hesaplanması.
- İşaretçiler (Göstergeler) ve onların programlamadaki önemi.
- & adres operatörünü ve * referanssızlaştırma operatörünü kullanmak.
- Diziler ve işaretleyiciler arasındaki ilişki.
- new ve delete komutlarını kullanma.
- Dizelerle çalışma fonksiyonları.
- Dizeden sayıya ve sayıdan dizeye dönüştürme fonksiyonları.

Anahtar Kelimeler

#define.	Dİnamik değişken
<string>	* Dolaylı operatör (referanssızlaştırma operatörü)
delete	Dizi
new	İki boyutlu dizi
stoi, stol, stoll, stoul, stoull, stof, stod, stold	İşaretçi
to_string()	İşaretçi değişkeni
& adres operatörü	Matris
Adlandırılmış sabit	Sembolik sabit
Aralığa dayalı for komutu	Tek boyutlu dizi
Başlatma listesi	

3.1 Tek Boyutlu Diziler

Çözülmesi için çok sayıda değişkenin kullanılmasını gerektiren problemler vardır. Örneğin, bir öğrencinin ortalama notunu hesaplamak için, tüm derslerin notunu girmek ve her not için bir tam sayı değişkeni girmek gerekir, örneğin:

not_1 not_2 . . . not_n

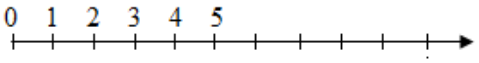
12 ders olduğunu tahmin edersek, bir öğrencinin notları için 12 değişkene ihtiyaç vardır. Sınıfta 30 öğrenci varsa, tüm öğrencilerin notları için 12 x 30 = 360 değişken gerekir. Bu kadar çok sayıda değişkene sahip program hem zor okunur hem de çok uzundur. Yazarken, değişkenlerin adlarını yazmak için uygulamanın kendisinden daha fazla zaman harcanacaktır.

Böyle durumlarda, aynı türden çok sayıda veri ile çalışırken, **dizi** (İng.arrays) adı verilen özel yapılarda düzenlenirler. Bir dizinin elemanları herhangi bir türden olabilir, ancak tüm elemanlar aynı türden olmalıdır. Örneğin, verilen örnekte not adlı dizideki tüm öğrenci notları tam sayı türünden olmalıdır.

Dizideki her elemanın kendi sıra numarası vardır. Örneğin not dizisinde eleman adları şunlarıdr: not_1, not_2… not_n. Onlar sadece 1, 2, 3... n sıra sayılarına göre farklılık gösterirler. Bu tür değerler matematikte sıra sayılarını indis (dizin) olarak yerleştirilecek şekilde yazılır: not1, not2... notn. İndis, dizideki elemanın sıra numarasını gösterir: 1 - ilk eleman, 2 - ikinci eleman, vb. n, n'inci elemanı belirtir.

Aritmetik ve geometrik dizileri hatırlayalım. a1, a2...an dizisine, herhangi ardışık elemanın farkı aynıysa aritmetik dizi denir, geometrik dizide ise herhangi iki ardışık elemanın bölümü aynıdır. Ayrıca tüm elemanların aynı ada sahip olduğunu ve indisleri 1, 2, 3... n olduğunu görüyoruz.

İndisler, sayısal bir eksende, yani tek yönde, tek boyutta noktalar olarak işaretlenebilir. Uzunluk bir boyuttur. Yüzeyin (dikdörtgenin) uzunluğu ve genişliği vardır, yani iki boyutu vardır. Mekanın (binanın) üç boyutu vardır: uzunluk, genişlik ve yükseklik.



Bu nedenle, tek indisli (tek boyutlu) dizilere **tek boyutlu diziler** (İng. one dimensional arrays) denir.

Dizinin n elemanı olduğunu biliyorsak, şu şekilde yazılır: a1, a2... an. Veya kısaca a[i]n veya sadece a[]n, yani i indisinin 1 ile n arasında değerlere sahip olduğu anlamına gelir. Aynı zamanda [a]n gösterimi de kullanılır, burada n, dizinin tek boyutlu olduğunu ve indislerinn 1'den n'ye kadar olduğu belirtilir. Literatürde a[1..n] gösterimi de kullanılmaktadır. Algoritmalarda son gösterimi kullanacağız.

Programlama dillerinde, tek boyutlu diziden veri değerleri alan değişkenler, indis orta parantez içine koyulacak şekilde yazılır: a[1], a[2]... a[n] veya not [1], not[2]... not[n]. Tüm elemanların adları aynı olmasına rağmen, indise göre farklılık gösteren farklı değişkenlerdir.

Birçok programlama dilinde, indisler 0'dan başlar ve n – 1'e kadar devam eder.

C++'daki diziler, indisler negatif olmayan tam sayılar ve ilk indis her zaman 0 olarak aynı şekilde belirtilir. n elemanlı bir dizi için indisler 0, 1, 2, 3... n – 1'dir. Örneğin, n elemanlı dizi için işaret a[]n'dir, elemanları ise şunlardır: a[0], a[1]... a[n – 1]. Eleman sayısı olmadan, sadece diziyi belirtmek istiyorsak, onu aynı adlı bazı değişkenden ayırmak için a[] gösterimini kullanırız.

Tek Boyutlu Dizilerin Bildirimi

Tek boyutlu diziler, C++'da değişkenler olarak aşağıdaki şekilde gibi bildirilir:

tür ad[sayı];

tür – dizi elemanlarının türüdür (short, int, long, long long, float, double, char, boolean, string, vb.).
ad – dizinin adıdır.
sayı – dizi elemanlarının sayısıdır ve değişmez değer veya 0'dan büyük adlandırılmış sabit olması gerekir.

Örnekler

Örnek 3.1.1

```
int a[5];
```

a[] dizisinin 5 elemanı vardır: a[0], a[1], a[2], a[3] ve a[4]. Dizi elemanları int türündendir ve onlara sadece tam sayı değerleri atanabilir.

Örnek 3.1.2

```
float br[25];
```

sayi[] dizisi float türündendir ve 25 elemanı vardır. Genel k-inci elemanı sayi [k] ile gösterilir ve k indisi 0, 1... 24 değerlerine sahip olabilir. Demek ki, elemanlar: sayi[0], sayi[1],... , sayi [24]'dir.

Örnek 3.1.3

```
string ime[15], prezime[30];
```

Elemanları yalnızca dize değerlerine sahip olabilen ad[] ve soyad[] adlı iki dizi bildirilmiştir.

Örnek 3.1.4

```
const int ELEMANSAYİSİ  = 100;
char c[ELEMANSAYİSİ];
```

Bu örnekte, c[] karakter dizisinin eleman sayısı önceden ELEMANSAYİSİ adıyla **adlandırılmış sabit** (İng. named constant) veya başka adıyla **sabit değişkeni** (İng. constant variable) olarak verilmiştir.

Dizi elemanlarının sayısını belirtmek için bu yol en sıkça kullanılır, çünkü adlandırılmış sabit bir kez başlatılır ve daha sonra uygulamada değiştirilemez.

Örnek 3.1.5

```
int k = 10;
float u[k];
```

Yukarıdaki bildirim yanlıştır çünkü elemanların sayısı sabit olmalıdır.

Dizide belirli bir elemanı belirtmek için, dizinin adını ve o elemanın dizideki konumunu belirtiriz. ***Tablo 3.1'de*** , d adında tam sayı dizisi verilmiştir. Bu dizi 10 elemandan oluşmaktadır. Bu elemanların her birine dizinin adı ve orta parantez [] içine alınmış dizideki elemanın indisini kullanarak erişilebilir.

Böylece, ilk eleman d[0]'dir, i-inci eleman d[i – 1]'dir.

0 dizideki birinci elemanın indisidir .

9 dizideki son elemanın indisidir.

Eleman	Elemanın değeri
d[0]	–567
d[1]	8
d[2]	9
d[3]	72
d[4]	345
d[5]	–78
d[6]	78
d[7]	9
d[8]	10
d[9]	987

Tablo 3.1.1

Bir dizide bulunan elemanların indisi, değeri negatif olmayan ve 0 ile n – 1 arasındaki indis aralığında (dizinin uzunluğu n ise) sabit, değişken veya ifade olabilir. Böylece, a = 7 ve b = 2 için

```
d[a + b - 1]
```

elemanı aslında d[8] elemanıdır.

Dizinin Başlatılması ve Elemanlara Değer Atama

Dizi elemanlarına değer, atama komutu ile veya başlatma sırasında atanabilir.

Aşağıdaki örneklerde, dizi elemanlarına değer atamanın farklı yolları gösterilmiştir.

Örnek 3.1.6

Değerler her elemana ayrı ayrı atanabilir:

```
int a[10];
a[0] = 1; a[9] = 10; a[3] = -4; a[5] = 5;
```

Örnek 3.1.7

Değer ataması, dizi indislerinin değerinin hesaplandığı ifadeler ile de yapılabilir. Örneğin, a[5] = 24 elemanı şu komutlarla:

```
int c = 3;
int d = 2;
a[c + d] += 6;
```

a[5] elemanının değeri 6 için artar, yani değeri şimdi 30 olacaktır.

Örnek 3.1.8

Dizi elemanlarına değer atama, bildirim sırasında, yani başlatma listesiyle (İng. initializer list) de yapılabilir:

```
 int a[] = {5, -2, 7, -3, 6};
```

Elemanların değerleri şöyle olacaktır: a[0] = 5, a[1] = –2, a[2] = 7, a[3] = –3, a[4] = 6. Dizinin uzunluğu (eleman sayısı) bu şekilde başlatma sırasında derleyici tarafından belirlenir.

Aşağıdaki dizinin bildirimi ve başlatılmasından sonra

```
char harfler[] = {'a', 'b', 'c'};
```

elemanların değerleri şöyle olacaktır: a[0] = 'a', a[1] = 'b', a[2] = 'c'.

Örnek 3.1.9

Başlatma listesinde dizideki elemandan daha az değerler varsa, kalan elemanlar sıfır olarak başlatılır.

Aşağıdaki bildirim ve başlatma ile:

```
int bazilari[10] = {3, -1, 4};
```

dizinin elemanları şu değerlerle başlatılacaktır: 3, –1, 4, 0, 0, 0, 0, 0, 0, 0. Sıfırlar derleyici tarafından doldurulur.

Benzer şekilde,

```
int sifirlar[100] = {};
```

ile tüm elemanları 0 olarak başlatılmış olan, sifirlar adlı tam sayı elemanlar dizisi bildirilmiştir.

Aşağıdaki bildirim ve başlatma ile:

```
char sesliHarfler[5] = {'a'};
```

'a' değeri, dizinin sıfırıncı elemanına atanır, diğer tüm elemanlar otomatik olarak boşluk olarak başlatılır.

Not: C++'da dizilerin otomatik olarak başlatılması yoktur. Diğer tüm elemanlar başlatılmadan önce en az bir eleman başlatılmalıdır.

Örnek 3.1.10

Elemanlar başlatma sırasında verilmişse, dizinin boyutu otomatik olarak hesaplanabilir. Aşağıdaki bildirimde ve başlatmada:

```
double bunlar[] = {2.34, -5.67, 3.45, 7.89};
```

dizinin uzunluğu verilmemiştir, ancak verilen eleman sayısına göre derleyici tarafından belirlenir, yani bunlar dizisinin uzunluğu 4 olacaktır.

Benzer şekilde, aşağıdaki bildirim ve başlatma ile:

```
string gunler[] = {"Pzt", "Sal", "Car", "Per", "Cum", "Cmt", "Paz"};
```

gunler dizisinin uzunluğu 7'dir.

Aşağıdaki bildirim ve başlatma ile:

```
int rakamlar[] = {1, 2, 3, 4, 5};
```

5 uzunluğunda dizi bildirilir ve tüm elemanlar başlatılır.

Diğer taraftan, aşağıdaki dizi bildirimi ve başlatılması ile

```
int v[5] = {1, 2, 4, 5, 6, 9};
```

Sözdizimi hatası oluşur, çünkü 6 başlatma değeri var ancak dizi en fazla 5 elemana sahip olabilir.

Örnek 3.1.11

Dizelerle şunları yapılmaz:

- • birbirine atanamaz:

```
int a[2] = {1,2};
int b[2];
b = a;                          // yanlis
```

- • birbiriyle başlatılamaz:

```
int c[2] = a;                   // yanlis
```

- • sadece adı aracılığıyla yazdırılamaz[1]:

```
cout << a;                      // yanlis
```

- • karşılaştırılamaz[2]:

```
if(a == b)                      // yanlis
```

- • Fonksiyonun dönüş değeri olamazlar[3].

```
return a;                       // yanlis
```

[1] Dizi adresi yazdırılır

[2] Dizi adresleri karşılaştırılır.

[3] Dizi işaretçisi döndürülür. İşaretçiler hakkında daha sonra bahsedeceğiz.

Dizi uzunluğunun belirtilmesi, **sembolik sabitle** de yapılabilir. Sembolik sabit, #**define** ön işlemci yönergesi ile belirtilir.

Örneğin,

```
#define PI 3.1415;
```

Örnek 3.1.12

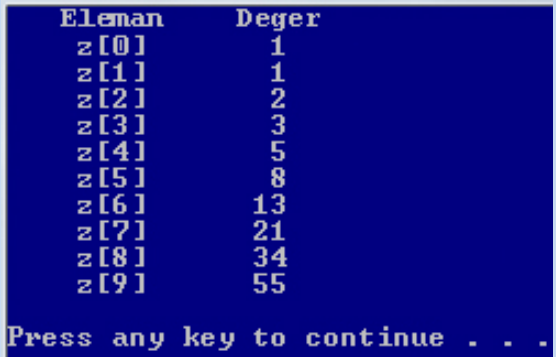
Bu örnekte, değeri 10 olan UZUNLUK sembolik sabitinin tanımlandığı #**define** ön işlemci yönergesi kullanılmıştır, **Şekil 3.1.1.**

Aşağıdaki program dizinin ilk iki elemanını 1, diğerlerini ise 0 değeri ile başlatır. for komutu ile üçüncü elemandan onuncu elemanına kadar değerler bir formüle göre atanır. (Hangi formül?)

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

#define DOLZINA 10

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

#define UZUNLUK 10

int main() {    // Sembolik sabitinin kullanimi

    int z[ UZUNLUK ] = { 1, 1 };// İlk iki elemanin 1 ile
                                // digerlerin 0 ile baslatilmasi

    // Ucuncuden onuncuya kadar olan elemanlara deger atama
    int j;
    for( j = 2; j < UZUNLUK; j++ )
        z[ j ] = z[ j - 2 ] + z[ j - 1 ];

    // z[] dizisinin tablo biciminde yazdirilmasi
    cout << setw( 10 ) << "Eleman" << setw( 10 ) << "Deger" << endl;
    for( j = 2; j < UZUNLUK; j++ )
        cout << setw( 6 ) << "z[" << j << "]" << setw( 8 ) << z[ j ] << endl;

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

Şekil 3.1.1

Programı çalıştırdıktan sonraki çıktı şudur:



Örnek 3.1.13

$a[]_n$, n elemanlı tam sayı dizisinin elemanlarını okumak için program bölümü şu şekildedir:

```
for(int i = 0; i < n; i++)
      cin >> a[i];
```

Örnek 3.1.14

$b[]_n$ dizisinin elemanlarını yazdırmak için program bölümü şudur:

```
for(int i = 0; i < n; i++)
      cout << a[i] << endl;
```

Örnek 3.1.15

n elemanlı $c[]_n$ dizisinin elemanlarının toplamını bulmak için program bölümü şöyledir:

```
toplam  = 0;
for(int i = 0; i < n; i++)
      toplam  += c[i];
```

Aynı program bölümü, bir dizinin elemanlarının çarpımını hesaplamak için de kullanılabilir, sadece şu komutların değiştirilmesi gerekir:

```
toplam  = 0;     komutunu  carpim  = 1;
toplam  += c[i]; komutunu  carpim  *= c[i];
```

Bu program bölümleri, dizi elemanlarıyla herhangi bir işlem için kullanılabilir.

Örnek 3.1.16

Şekil 3.1.2'deki örnekte, int, float, char ve string türü dizilerin bildirilmesi, başlatılması ve elemanlarına değer atanması gösterilmiştir.

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

int main() {    // Dizilerin bildirilmesi, baslatilmasi  ve
                // elemanlara deger atanmasi

    int uzunluk;
    int a[ 4 ] = {};
    a[ 0 ] = 123; a[ 2 ] = 12345;
    uzunluk = sizeof a / sizeof( a[ 0 ] );
    cout << uzunluk <<"uzunlugundaki tamsayi dizisi" << endl;
    cout << "indis" << setw( 10 ) << "deger " << endl;
    for( int indis = 0; indis < uzunluk; indis++ )
        cout << setw( 3 ) << indis << setw( 10 ) << a[ indis ] << endl;

    float b [] = { -3.5f, 2.7f, 7.3f };
    b[ 1 ] = 123.45f;
    uzunluk = sizeof b / sizeof( b[ 0 ] );
    cout << '\n' << uzunluk  << " uzunlugundaki gercek sayilar dizisi"<< endl;
    cout << "indis" << setw( 10 ) << "deger " << endl;
    for( int indis = 0; indis < uzunluk; indis++ )
        cout << setw( 3 ) << indis << setw( 10 ) << b[ indis ] << endl;

    char karakterler [] = { '@', '#', '$', '%', '^' };
    karakterler [ 1 ] =  ['; karakterler [ 2 ] = ']';
    uzunluk = sizeof karakterler / sizeof( karakterler [ 0 ] );
    cout << '\n' << uzunluk << "uzunlugundaki karakterler dizisiis " << endl;
    cout << "indis" << setw( 10 ) << "deger" << endl;
    for( int indis = 0; indis < uzunluk; indis++ )
        cout << setw( 3 ) << indis << setw( 10 ) << karakterler[ indis ] << endl;

    string isimler [] = { "Antonia", "Atanasia", "Mihaela", "Yana", "Yovana", "Teo" };
    uzunluk = sizeof isimler / sizeof(isimler[ 0 ] );
    cout << '\n' << uzunluk << "uzunlugundaki dizeler dizisi" << endl;
    cout << "indis" << setw( 10 ) << "deger" << endl;
    for( int indis = 0; indis < uzunluk; indis++ ) {
        cout << setw( 3 ) << indis << "\t";
        cout << setw( 10 ) << left << isimler[ indis ] << right << endl;
    }

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

Şekil 3.1.2

Programın çıktısı şudur:

```
4 uzunlugundaki tamsayi dizisi
indis     deger
  0         123
  1           0
  2       12345
  3           0

3 uzunlugundaki gercek sayilar dizisi
indis     deger
  0        -3.5
  1      123.45
  2         7.3

5 uzunlugundaki karaterler dizisi
indis      deger
  0          @
  1          [
  2          ]
  3          %
  4          ^

6 uzunlugundaki dizeler dizisi
indis     deger
  0       Antonia
  1       Atanasia
  2       Mihaela
  3       Jana
  4       Jovana
  5       Teo

Press any key to continue . . .
```

Kütüphane Fonksiyonları alt başlığında, bir tür veya ifade (değişken) tarafından kaplanan belleğin boyutunu elde etmek için kullanılabilen sizeof operatörüyle tanıştık.

sizeof operatörü, diziye de uygulanabilir, öyle ki dizinin büyüklüğünü, eleman sayısına bağlı olarak bayt ile verir. Dizinin eleman sayısını elde etmek için, dizinin büyüklüğünü (bayt ile) dizi elemanlarının türünün boyutuyla (bayt ile) bölmemiz gerekir.

Örneğin:

```
char karakterler[] = {'@','#','$','%','^'};
uzunluk = sizeof karakterler / sizeof(char);
veya
uzunluk = sizeof karakterler/sizeof(karakterler[0]);//karakterler[0] ilk elemandır
```

Çözülmüş Ödevler

Ödev 3.1.1

$a[]_n$ sayısal dizisinin her bir elemanın işaretini +'dan -'ye ve -'den +'ya değiştirecek algoritma ve program yazılsın.

Algoritma ve program aşağıda verilmiştir.

```
algoritma DizideİşaretlerinDeğiştirilmesi
başlangıç
        a[1..n];
        yazdır "Dizinin eleman sayısını girin, n = ";
        oku  n;
        için  i ← 1 arttır n'e kadar
                oku  ai;
        bitiş_için {i}
        için  i ← 1 arttır n'e kadar
                ai ← –ai;
        bitiş_için  {i}
        yazdır "Değiştirilmiş işaretlerle dizi şöyledir ";
        için  i ← 1arttır n'e kadar
                yazdır  ai;
        bitiş_için {i}
bitiş { DizideİşaretlerinDeğiştirilmesi}
```

Programın yürütülmesi ile çıktı şudur:

```
Dizinin eleman sayisini girin, n =3
Dizinin elemanlarini girin:
a[0] = 1
a[1] = -2
a[2] = 3

Girilen dizi sudur:
a[0]=1
a[1]=-2
a[2]=3

Degistirilmis isaretlerle dizi soyledir:
a[0]=-1
a[1]=2
a[2]=-3

Press any key to continue . . .
```

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // Dizi elemanlarinin isaretleri degistirilsin
6
7       const int m = 100;
8       double a[ m ] = {};
9       cout << "Dizinin eleman sayisini girin, n = ";
10      int n; cin >> n;
11      cout << "Dizinin elemanlarini girin: \n";
12      for( int i = 0; i < n; i++ ) {
13          cout << "a[" << i << "] = ";
14          cin >> a[ i ];
15      }
16
17      cout << "\nGirilen dizi sudur:" << endl;
18      for( int i = 0; i < n; i++ )
19          cout << "a[" << i << "]=" << a[ i ] << endl;
20      for( int i = 0; i < n; i++ )
21          a[ i ] = -a[ i ];
22
23      cout<<"\nDegistirilmis isaretlerle dizi soyledir:"<< endl;
24      cout<<"\nDegistirilmis isaretlerle dizi soyledir:"<< endl;
25      for (int i = 0; i<n; i++)
26          cout << "a[" << i << "]=" << a[i] << endl;
27
28      cout << endl;
29      system("Color 17");
30      system("pause");
31      return 0;
32  }
```

Şekil 3.1.3

Ödev 3.1.2

Aşağıdaki toplamı hesaplayacak program yazılsın:

$+ a_1 - a_2 + a_3 - a_4 + \ldots (+ / -) a_n$

Programın yürütülmesinin bir örneği şudur:

```
Dizinin eleman sayisini girin, n =5
Dizinin elemanlarini girin:
a[0] = 11
a[1] = -22
a[2] = -333
a[3] = 4444
a[4] = -55555

Girilen dizi sudur: 11, -22, -333, 4444, -55555,
+(11) - (-22) + (-333) - (4444) + (-55555) Dizisinin toplami: - 60299.

Press any key to continue . . .
```

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

int main() {
    // a1 - a2 + a3 - ... (+/-)an toplami bulunsun

    const int m = 100;
    double toplam = 0, a[ m ] = {};
    int i, k, n;
    cout << "Dizinin eleman sayisini girin , n = ";
    cin >> n;
    cout << "Dizinin elemanlarini girin : \n";
    for( i = 0; i < n; i++ ) {
        cout << "a[" << i << "] = ";
        cin >> a[ i ];
    }
    cout << "\nGirilen dizi sudur:";
    for( i = 0; i < n; i++ )
        cout << a[ i ] << ", ";
    k = 1;
    for( i = 0; i < n; i++ ) {
        zbir += k * a[ i ];
        k = -k;
    }
    cout << "\nDizisinin toplami:";
    for( i = 0; i < n; i++ ) {
        if( i % 2 == 0 )
            cout << '+' << "(" << a[ i ] << ")";
        else
            cout << '-' << "(" << a[ i ] << ")";
    }
    cout << " e " << toplam << endl;

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

Şekil 3.1.4

Ödev 3.1.3

Özel fonksiyonlarla a[]n sayısal dizisindeki en büyük ve en küçük elemanı bulacak program yazılsın.

Açıklama: Başlangıçta, en büyük eleman birincisi olan a1 olduğu varsayılıyor. Ardından ikinci eleman a2 en büyüğü (birinci olan) ile karşılaştırılır ve daha büyüğü en büyük olarak sayılır. Devamda diğer tüm a_3,

a_4...an elemanları o ana kadar bulunan en büyük elemanla karşılaştırılır ve ondan daha büyük bulunursa en büyük olarak kabul edilir. Bu arada en büyük elemanın dizideki konumu da hatırlanır.

Bir örnek vereceğiz. Dizi a = {2, –1, 7, 9, 3} olsun.

i	a_i	a_i > max	max	indis
1	2		2	1
2	–1	(–1 > 2) hayır		
3	7	(7 > 2) evet	7	3
4	9	(9 > 7) evet	9	4
5	3	(3 > 9) hayır		

Demek ki, en büyük eleman 9 değeriyle 4'üncü elemandır.

Dizide en küçük elemanı bulma süreci benzerdir. Bunu, aynı örnekle göstereceğiz

i	a_i	a_i < min	min	indis
1	2		2	1
2	–1	(–1 < 2) evet	–1	2
3	7	(7 < –1) hayır		
4	9	(9 < –1) hayır		
5	3	(3 < –1) hayır		

Demek ki en küçük eleman, -1 değeriyle 2'nci elemandır.

Argümanları dizi olan fonksiyonun tanımında, diziler sadece türü ve adlarıyla yazılır ve ardından bunun dizi olduğunu, değişken olmadığını belirtmek için parantezler koyulur.

Örneğin:

```
void EnBuyukEleman(double a[], double elemansayisi, double &konum,
double &enbuyuk)
```

Bu arada, & işareti yerleştirilmemiş olmasına rağmen dizi bir referans argümanı olarak ele alınır. Bu, fonksiyonun yürütülmesi sırasında dizinin elemanlarında gerçekleşecek tüm değişikliklerin ana fonksiyona döndükten sonra da kalacağı anlamına gelir.

Argümanı dizi olan fonksiyon çağrılırken, köşeli parantezler koymadan sadece dizinin adı yazılır:

```
EnBuyukEleman(a, n, siranumarasi, enbuyuk);
```

Bu arada, dizinin boyutu, değere göre argüman olarak aktarılır.

(Aslında, diziler için kullanılan özel **işaretçiler**[4](İng. pointer) mekanizması vardır. Bu mekanizma ile dizinin adı, dizinin ilk elemanının bellekteki adresini içeren işaretçi olarak ele alınır. (argüman adresinin karşılık gelen referans parametresine aktarıldığı gibi benzer bir şekilde)).

[4] **3.4 İşaretçiler** alt noktasına bakınız

Bazı durumlarda, dizinin geçtiği fonksiyondaki elemanlarının değerlerini değiştirmek istemediğimizde, referans olarak ele alındığı için const nitelendirici ile sabit olarak işaretlenmesi gerekir.

Dizideki en büyük ve en küçük elemanı bulma fonksiyonları ve bu fonksiyonların çağrıldığı ana fonksiyon **şekil 3.1.5**'te verilmiştir.

```
1   #include <iostream>
2   #include <iomanip>
3   using namespace std;
4
5   void diziOkuma( int&, double [] );
6   void diziYazdirma( const int, const double [] );
7   int enBuyukEleman( const int, const double [] );
8   int enKucukEleman( const int, const double [] );
9
10  int main() {    //Dizide en buyuk ve en kucuk eleman
11
12      double a[ 100 ];
13      int n, index;
14
15      diziOkuma( n, a );
16      system( "cls" );
17      cout << setw( 14 ) << "" << "Tek boyutlu dizi \n" << endl;
18      diziYazdirma( n, a );
19
20      index = enBuyukEleman( n, a );
21      cout << "\nEn buyuk eleman" << indis << "indisine ve"
22          << a[ indis ]  << "degerine sahiptir" << endl;
23
24      index = enKucukEleman( n, a );
25      cout << "\nEn kucuk eleman" << indis << "indisine ve"
26          << a[ indis ]  << "degerine sahiptir" << endl;
27
28      cout << endl;
29      system( "Color 17" );
30      system( "pause" );
31      return 0;
32  }
33
34  void diziOkuma( int& boyut, double x [] ) {
35      cout << "Eleman sayisini girin: ";
36      cin >> boyut;
37      cout << "Dizi elemanlarini girin: " << endl;
38      for( int i = 0; i < boyut; i++ ) {
39          cout << "x[" << i << "] = ";
40          cin >> x[ i ];
41      }
```

Şekil 3.1.5

```
42  }
43
44  void diziYazdirma( const int boyut, const double x [] ) {
45      cout << setw( 10 ) << left << "Indis:" << right;
46      for( int i = 0; i < boyut; i++ )
47          cout << setw( 5 ) << i;
48      cout << endl;
49      cout << setw( 10 ) << left << "Deger:" << right;
50      for( int i = 0; i < boyut; i++ )
51          cout << setw( 5 ) << x[ i ];
52      cout << endl;
53  }
54
55  int enBuyukEleman( const int elemansayisi, const double x [] ) {
56      int konum = 0;
57      double enBuyuk = x[ 0 ];
58      for( int i = 1; i < elemansayisi; i++ )
59          if( x[ i ] > enBuyuk ) {
60              enBuyuk = x[ i ];
61              konum = i;
62          }
63      return konum;
64  }
65
66  int enKucukEleman( const int elemansayisi, const double x [] ) {
67      int konum = 0;
68      double enKucuk = x[ 0 ];
69      for( int i = 1; i < elemansayisi; i++ )
70          if( x[ i ] < enKucuk ) {
71              enKucuk = x[ i ];
72              konum = i;
73          }
74      return konum;
75  }
```

Şekil 3.1.5 (devam)

Tek buyutlu dizi

```
               Tek buyutlu dizi

Dizin:         0    1    2    3    4    5    6    7    8    9
Deger:         3    7   12    0   -3  -23    0    5   25    4

En buyuk eleman 8 dizinine ve 25 degerine sahiptir
En buyuk eleman 5 dizinine ve -23 degerine sahiptir

Press any key to continue . . .
```

Aralığa Dayalı for Komutu

Örnek 3.1.15'te, n elemanlı $c[]_n$ dizisinin elemanlarının toplamı hesaplanıyor:

```
toplam  = 0;
for(int i = 0; i < n; i++)
      toplam  += c[i];
```

Elemanların i indisine göre sıralandığı şekilde baştan sona eklendiğini dikkat edebiliriz. Ancak herhangi bir sırayla toplarsak aynı toplam elde edilecektir.

Sıralamasına bağımsız olarak bir dizinin elemanları üzerinde bazı işlemlerin gerçekleştirildiği birçok problem vardır. Örneğin: bir dizideki en küçük (veya en büyük) elemanı bulma ödevinde, ilkinin en küçük olarak alınması gerekmiyor, herhangi bir eleman olabilir. Ondan sonra diğerleriyle karşılaştırılır, ancak herhangi bir sırayla olması gerekmez. Önceki en küçüğünden daha küçük eleman bulunduğunda, en küçüğü o olacaktır.

Bu tür problemleri çözmek için C++'da **aralığa dayalı for komutu** (İng. range-based for) kullanılır, **şekil 3.1.12.**

```
for(değişken : dizi){
        komut A;
        komut B;
        ...
        komut K;
}
```

Şekil 3.1.12

- *değişken*, dizi türünden değişkendir
- *dizi*, dizinin adıdır

Aralığa dayalı for komutunun gövdesinde indislerin işaretlemesi veya indislerle işlemler olamaz.

Birkaç örnek vereceğiz:

Örnekler

Örnek 3.1.17

c[] dizisinin elemanlarının toplamı için **Örnek 3.1.15**'teki program bölümü aşağıdaki gibi yazılabilir:

```
for(int sayi: c)
      toplam  += sayi;
```

Örnek 3.1.18

Ödev 3.1.2'de, elemanların toplamı için for komutu:

```
for(i = 0; i < n; i++) {
	toplam  += k * a[i];
	k = -k;
}
```

aralığa dayalı for komutu ile de yazılabilir.

```
for(double sayi: a) {
	toplam  += k * sayi;
	k = -k;
}
```

Örnek 3.1.19

Dizide en küçük veya en büyük elemanı bulmak için, elemanları indise göre karşılaştırmak değil, tüm elemanları karşılaştırmak gerekir. Bu nedenle, aralığa dayalı for komutu kullanılabilir.

```
double enKucuk  = a[0];
double enBuyuk  = a[0];
for(auto sayi: a) {
	if(sayi < enKucuk )
		enKucuk  = broj;
	if(sayi > enBuyuk )
		enBuyuk  = sayi;
}
```

Alıştırma Ödevleri

Aşağıdaki ödevler için tek boyutlu diziler kullanarak programlar yazılsın.

1. $a[]_n$ tam sayı dizisinin elemanlarının çarpımı bulunsun.
2. $a[]_n$ sayısal dizisinin aritmetik (A) ve harmonik (H) ortalaması hesaplansın.

$$A = \frac{a_1 + a_2 + \ldots + a_n}{n} \qquad H = \frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + \ldots + \frac{1}{a_n}}$$

3. $a[]_n$ sayılar dizisinden, çift sayıların toplamı ve tek sayıların toplamı ayrı ayrı hesaplansın.
4. $a[]_n$ dizisinin çift indisli ve tek indisli elemanları ayrı dizilere ayrılsın.
5. $a[]_n$ karakter dizisinin elemanları k yer için sağa veya sola döngüsel olarak kaydırılsın.
6. $a[]_n$ sayısal dizisinde d değerine sahip bir elemanın olup olmadığı kontrol edilsin.
7. $a[]_n$ sayısal dizisinin kaç elemanın d değerinden daha küçük ve kaç elemanın daha büyük olduğu bulunsun.
8. Elemanları, $a[]_n$ ve $b[]_n$ dizilerinin karşılık gelen elemanlarının toplamı olan yeni $c[]_n$ dizisi oluşturulsun, yani i = 0, 1... n – 1 için $c_i = a_i + b_i$.

9. a[]n ve b[]$_n$ sayısal dizileri elemanları çarpımının toplamı hesaplansın, yani
   $a_0b_0 + a_1b_1 + \ldots + a_{n-1}b_{n-1}$.
10. Aşağıdaki polinomun değeri hesaplansın
    $P_n(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1 + a_0$
    katsayılar ve argümanlar gerçek sayılardır, n ise doğal sayıdır.

Bilgiyi Kontrol Etme Soruları

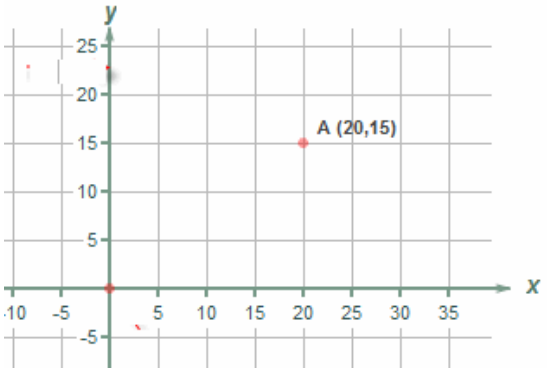
1. Tek boyutlu dizi nasıl bildirilir?
2. Tek boyutlu dizinin elemanlarının indisi hangi veri türünden olmalıdır?
3. Aşağıdaki tek boyutlu dizi bildirilmişse
   ```
   int a[10];
   ```
   dizi elemanlarının indisi hangi değerleri alabilir?
4. Tek boyutlu dizinin elemanlarına hangi yollarla değerler atanılabilir?
5. Tek boyutlu dizinin elemanları hangi yollarla başlatılabilir?
6. Sembolik sabit nasıl tanımlanır?
7. Tek boyutlu dizinin boyutu nasıl belirlenir?
8. Aralığa dayalı for komutu ne zaman kullanılır? Onun sözdizimini yazın.

3.2 İki Boyutlu Diziler – Matrisler

3.1 **Tek Boyutlu Diziler** alt bölümünde, tek boyutlu verilerin (matematiksel olarak) tek indisli dizilerle temsil edilebileceğini söylemiştik: a_1, a_2, ... , an veya kısaca a[]n. C++ da dahil olmak üzere programlama dillerindeki bu diziler, orta parantez içindeki indislerle yazılır: a[0], a[1]... a[n – 1]. Bu arada indisler 0'dan başlar.

Matematikte ve diğer alanlarda bazı verileri yazmak için 2, 3... ve daha fazla indisin kullanılması gereken örnekler vardır.

Örneğin, bir düzlemde her noktanın iki boyutla - koordinatlar ile belirlendiğini biliyoruz: x koordinatı ve y koordinatı, yani A(x, y), **şekil 3.2.1.**



Şekil 3.2.1

Benzer şekilde, bir sınıftaki öğrencilerin tüm derslerde (12 tahmin edelim) notları, satırlar (yatay, birinci boyut) ve sütunlar (dikey - ikinci boyut) olmak üzere iki boyutlu tablo ile gösterilebilir.

		1	2	3	4	5	6	7	8	9	10	11	12	
	Ders →	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	Ortalama
	Ogrenci ↓													
1	u_1	3	4	5	2	3	4	3	4	5	3	3	2	
2	u_2	4	5	4	5	3	4	5	4	5	5	5	4	
3	u_3	5	5	5	5	2	4	4	4	4	2	4	5	
…	. . .													
	Ortalama													

Şekil 3.2.2

Bu veriler, bir boyutun öğrenciler (o) ve diğer boyutun dersle (d) olduğu **iki boyutlu dizi** (İng. two-dimensional array) olarak yazılabilir. Buna göre, tabloda her notun iki indisi vardır. Diziyi not (notlar'dan kısaltılmış) ile adlandırırsak, o zaman iki boyutlu dizinin elemanlarının değerleri, öğrencilerin derslerden notlarıdır. Örneğin, $not(o_2, d_7) = 5$, $not(o_3, d_{10}) = 2$, vb. Veya kısaca, sadece indilerle, $not_{2,7} = 5$,$not_{3,10} = 2$ vb.

m öğrencinin ve n dersin olduğunu biliyorsak, o zaman dizi (matematiksel olarak) şu şekilde yazılır: $o_{1,1}, o_{1,2},..., o_{1,n}, o_{2,1}, o_{2,2},..., o_{2,n}, o_{3,1},..., o_{m,1}, o_{m,2},..., o_{m,n}$.

Veya daha açık bir şekilde iki boyutlu bir tablo olarak gösterebiliriz:

$$\begin{bmatrix} o_{1,1} & o_{1,2} & \cdots & o_{1,j} & \cdots & o_{1,n} \\ o_{2,1} & o_{2,2} & \cdots & o_{2,j} & \cdots & o_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ o_{i,1} & o_{i,1} & \cdots & o_{i,j} & \cdots & o_{i,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ o_{m,1} & o_{m,2} & \cdots & o_{m,j} & \cdots & o_{m,n} \end{bmatrix}$$

İki boyutlu dizi $not[i,j]_{m,n}$ ile işaretlenir, bu da i indisinin 1 ile m arasında değerlere sahip olduğu ve j indisinin 1 ile n arasında değerlere sahip olduğu anlamına gelir. Kısacası, dizi $not[]_{m,n}$ ile gösterilerek, m ve n dizinin iki boyutlu olduğunu gösterir ve birinci indisin 1'den m'ye kadar, ikinci indisin 1'den n'ye kadar olduğunu biliyoruz.

Literatürde not[1..m][1..n] yazılışı de kullanılmaktadır.

Genel eleman noti,j ile gösterilir.

(Matematikte iki boyutlu dizilere matris denir).

Bu tür iki boyutlu diziler, C++ da dahil olmak üzere programlama dillerinde not[m][n] ile yazılır, genel eleman ise, orta parantez içinde iki indis ile not[i][j] olur. Örneğin, not[2][7] = 5, not[3][10] = 2, vb. (not[2,7] gösterimi yanlıştır).

İki Boyutlu Dizilerin Bildirimi, Başlatılması ve Elemanlara Değer Atama

İki boyutlu dizinin bildirimi, tek boyutlu dizinin bildirimine benzerdir.

tür ad[sayı1] [sayı2];

tür – dizi elemanlarının türüdür (short, int, long, float, double, char, boolean, string .).
ad – dizinin adıdır,
sayı1 ve sayı2 – dizinin boyutlarıdır ve 0'dan büyük değişmez değerler veya adlandırılmış sabitler olmalıdır .

Daha kolay anlamak için, yukarıdaki öğrenci notları tablosunda sayı 1 sınıftaki öğrenci sayısı (satır sayısı) olacaktır, sayi 2 ise ders sayısı (sütun sayısı) olacaktır.

İki boyutlu dizinin her iki indisi de 0'dan başlar.
İki boyutlu dizinin ilk indisleri şunlardır: 0, 1, 2... sayı1 – 1.
İki boyutlu dizinin ikinci indisleri şunlardır: 0, 1, 2... sayı2 – 1.

Örnekler

Örnek 3.2.1
Aşağıdaki bildirimle

```
int a[3][4];
```

ilk indisi 0, 1 veya 2 ve ikinci indisi 0, 1, 2 veya 3 olan 12 elemandan oluşan $a[]_{3,4}$ dizisi bildirilir. Bu dizi grafiksel olarak, 3 satırlı ve 4 sütunlü tablo ile gösterilebilir:

		sütunlar (ikinci indis)			
		0	1	2	3
Satırlar (ilk indis)	0	$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
	1	$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
	2	$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$

Dizinin elemanları, indisleri orta parantez içinde olacak şekilde yazılır.

	0	1	2	3
0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

İki boyutlu dizinin elemanlarına değerler, atama komutu ile atanır.

Örnek 3.2.2

Aşağıdaki komutlarla:

```
float c[10][15];
c[0][4] = -5.23f; c[9][2] = 7.543f;
```

10 x 15, yani 150 elemanlı c[][] dizisi bildirilir ve c[0][4] ve c[9][2] elemanlarına değerler atanır. (Ancak diğer elemanların değerleri bilinmiyor).

İki boyutlu bir dizi, bildirim sırasında ve *başlatma listesiyle* başlatılabilir.

Örnek 3.2.3

Aşağıdaki

```
int d[3][2] = {};
```

d dizisinin tüm elemanları 0 değeriyle başlatılır.

Aşağıdaki komutla:

```
char harfler[3][2] = {'a', 'b', 'c'};
```

iki boyutlu harfler dizisi bildirilir ve başlatılır.

	0	1
0	a	b
1	c	
2		

Dizinin tüm elemanlarını başlatmak için başlatma listesinde yeterli değer yoksa, geri kalanı boşlukla başlatılır.

Şu komutla:

```
int d[3][2] = {-3,5,4};
```

üç eleman verilen değerlerle başlatılır, geri kalanları ise 0 ile başlatılır.

	0	1
0	–3	5
1	4	0
2	0	0

Örnek 3.2.4

Aşağıdaki komutla:

```
int d[3][2] = {{-3, 5},{4, 1},{7, -2}};
```

3 satırdan (her satır büyük parantez içine alınmıştır) ve 2 sütundan (her satırın 2'şer eleman vardır) oluşan dizi başlatma listesindeki verilere göre bildirilir ve başlatılır. Aynı bildirim aşağıdaki gibi daha açık bir şekilde de yazılabilir:

```
int d[3][2] = {
                    {-3, 5},
                    {4, 1},
                    {7, -2}
              };
```

	0	1
0	–3	5
1	4	1
2	7	–2

Benzer şekilde, bir diziyi başlatmak bazı satır için yeterli eleman yoksa dizide kalan elemanlar 0 ile başlatılır.

```
int d[3][2] = {{-3}, {4, 1}, {7}};
```

	0	1
0	–3	0
1	4	1
2	7	0

Aynı bildirimler ve başlatmalar aşağıdaki komutlarla da yapılabilir:

```
int d[][2] = {{-3, 5},{4, 1},{7, -2}};
```

ve

```
int d[][2] = {{-3}, {4, 1}, {7}};
```

bu arada birinci boyutun büyüklüğünün belirtilmesi gerekmiyor. Çevirici onu kendisi hesaplar.

Ancak, aşağıdaki komutla bildirilemez

```
int d[][] = {{-3, 5}, {4, 1}, {7, -2}};
```

çünkü çevirici her satırdan (iç parantez) kaç eleman alacağını bilmiyor. 1 veya 2 alabilir. Bu yüzden, sadece ilk boyutun büyüklüğü atlatılabilir.

Örnek 3.2.5

Dizinin boyutları sabitler yardımıyla da belirtilebilir.

Şu komutlarla:

```
const int m = 10;
const int n = 5;
float b[m][n];
```

10 satır ve 5 sütundan oluşan iki boyutlu dizi bildirilir, yani ilk elemanı b[0][0] ve son elemanı b[9][4]'tür.

Ancak tüm b[m][n] elemanlar dizisinin kullanılması gerekli değildir, yalnızca bir kısmı kullanılabilir. Örneğin, daha küçük dizi boyutları belirtebiliriz:

```
int satirlar  = 4;
int sutunlar  = 3;
```

ve sadece b[satirlar][sutunlar] bölümü kullanılabilir.

İki Boyutlu Dizinin Boyutlarının Hesaplanması

a[]$_{m,n}$ dizisi başlatma listesiyle başlatılırsa,o zaman uzunluğun (satır sayısının) belirlenmesi için C++'da sizeof() fonksiyonu kullanılır.

```
    baytEleman  = sizeof(a[0][0]); – bir elemanın bayt sayısı
baytSatir   = sizeof(a[0]);       – bir satırın bayt sayısı.
bajtiDizi   = sizeof(a)           – tüm dizinin bayt sayısı.
```

Ardından şunlar hesaplanabilir:

```
uzunluk  = bajtiDizi  / baytEleman ;
sutunlar  = baytSatir  / baytEleman ;
satirlar  = uzunluk  / sutunlar;
```

Örnek 3.2.6

a[]m,n iki boyutlu dizisinin elemanlarını okumak için program bölümü şöyledir:

```
for(int i = 0; i < m; i++)
      for(int j = 0; j < n; j++)
            cin >> a[i][j];
```

Örnek 3.2.7

b[]m,n İki boyutlu dizisini yazdırmak için program bölümü şöyledir:

```
for(int i = 0; i < m; i++) {
      for(int j = 0; j < n; j++)
            cout << "\tb[" << i << "," << j << "] = " << b[i][j];
      cout << endl;
}
```

Örnek 3.2.8

İki boyutlu c[]$_{m,n}$ tam sayı dizisinin elemanlarının toplamını bulmak için program bölümü:

```
int toplam  = 0;
for(int i = 0; i < m; i++) {
      for(int j = 0; j < n; j++)
            toplam  += c[i][j];
}
```

Örnek 3.2.9

Şekil 3.2.3'teki programla, iki boyutlu dizinin bildirimi, başlatılması, okunması ve yazdırılması gösterilmektedir.

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

const int m = 2;            //İlk dizinin boyutları
const int n = 3;

int main() {    //İki boyutlu dizinin bildirilmesi, baslatilmasi
                // okunmasi ve yazdirilmasi

    int i, j;   // satirlar ve sutunlar sayacları, dizinin
                // baslatma listesiyle uygun baslatilmasi
    int diziA[ m ][ n ] = { { 1, 2, 3 }, { 4, 5, 6 } };
    cout << "İlk dizi baslatilmistir" << endl;

    cout << "\nİlk dizinin yazdirilmasi" << endl;
    for( i = 0; i < m; i++ ) {
        for( j = 0; j < n; j++ )
            cout << setw( 5 ) << diziA[ i ][ j ];
        cout << endl;      //Her satirdan sonra yeni sira
    }

    const int m = 3;        // İkinci dizinin boyutlari
    const int n = 4;
    int diziB[ m ][ n ] = {};   // bildirim ve 0 ile baslatma
                                //dizide elemanlarin okunmasi
    cout << "\nİkinci dizinin elemanlarini girin:" << endl;
    for( i = 0; i < m; i++ ) {
        cout <<'\n ' << i + 1 <<".nci satir elemanlarini girin: " << endl;
        for( j = 0; j < n; j++ ) {
            cout << j + 1 << ".nci elemani girin";
            cin >> dizi[ i ][ j ];
        }
    }

    cout << "\nİkinci dizinin yazdirilmasi" << endl;
    for( i = 0; i < m; i++ ) {
        for( j = 0; j < n; j++ )
            cout << setw( 5 ) << diziB[ i ][ j ];
        cout << endl;       //Her satirdan sonra yeni sira
    }

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

Şekil 3.2.3

Örnek 3.2.10

Bu örnek ile, farklı türde iki boyutlu dizinin bildirilmesi, başlatılması ve elemanlarına değerlerin atanması ve ayrıca satır ve sütun sayısının hesaplanması gösterilmektedir.

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

int main()
{   // İki boyutlu dizinin bildirimi, baslatilmasi
    // ve elemanlarina degerlerin atanması

    int i, j, uzunluk, satirlar, sutunlar, baytEleman, baytSatir, baytDizi;
    int a[2][3] = {};
    cout << "                0 ile baslatilmis ve ardindan"
        << "                \n2 deger atanan tamsayi dizisi: " << endl;
    a[1][2] =                45;
    satirlar = 2;
    sutunlar = 3;
    for (i = 0; i < satirlar; i++)
        for (j = 0; j < sutunlar; j++)
            cout << "\ntam[" << i << "][" << j << "] = " << a[i][j];

    float b[][2] = { { -3.5f, 2.7f },{ 7.3f },{ 0.3f, -2.f } };
    b[0][1] = 133.f; b[2][1] = -57.f;
    baytEleman = sizeof(b[0][0]);
    baytSatir = sizeof(b[0]);
    baytDizi = sizeof(b);
    uzunluk = baytDizi / baytEleman;
    sutunlar = baytSatir / baytEleman;
    satirlar = uzunluk / sutunlar
    cout << "\cc"
        << "\n gercek sayilar dizisi:" << endl;
    for (i = 0; i < satirlar; i++)
        for (j = 0; j < sutunlar; j++)
            cout << "\ngercek[" << i << "][" << j << "] = " << b[i][j];

    char karakterler[][3] = { { '@','#','$' },{ '%','^' },{ '*' },{ '-','+','=' } };
    sutunlar = 3;
    satirlar = sizeof(karakterler ) sizeof(karakterler[0][0]) / sutunlar;
    cout << "\n\n Karakterler dizisi:" << endl;
    for (i = 0; i < satirlar ; i++)
        for (j = 0; j < sutunlar ; j++)
            cout << "\n karakter [" << i << "][" << j << "] = " << karakterler [i][j];

    string isimler[][1] = { { "Antonia" },{ "Atanasia" },{ "Mihaela" },{ "Yovana " },
    {},{ "Yana" } };
    sutunlar = 1;
    satirlar = sizeof(isimler) / sizeof(isimler[0][0]) / sutunlar;
    cout << "\n\n Dizeler dizisi: " << endl;
    for (i = 0; i < satirlar; i++)
        for (j = 0; j < sutunlar; j++)
            cout << "\n dize [" << i << "][" << j << "] = " << isimler[i][j];
```

(int)3
Search Online

Şekil 3.2.4

```
50
51      cout << endl;
52      cout << endl;
53      system("Color 17");
54      system("pause");
55      return 0;
56  }
```

Şekil 3.2.4 (devam)

Programı yürütme sonucu şudur:

```
0 ile baslatilmis ve ardindan
2 deger atanan tamsayi dizisi:

tam[0][0] = 0
tam[0][1] = 0
tam[0][2] = 0
tam[1][0] = -45
tam[1][1] = 0
tam[1][2] = 123

Baslatilmis ve ardindan 2 deger atanan
gercek sayilar dizisi:

gercek[0][0] = -3.5
gercek[0][1] = 133
gercek[1][0] = 7.3
gercek[1][1] = 0
gercek[2][0] = 0.3
gercek[2][1] = -57

Karakterler dizisi:

karakter [0][0] = @
karakter [0][1] = #
karakter [0][2] = $
karakter [1][0] = %
karakter [1][1] = ^
karakter [1][2] =
karakter [2][0] = *
karakter [2][1] =
karakter [2][2] =
karakter [3][0] = -
karakter [3][1] = +
karakter [3][2] = =

Dizeler dizisi:

dize[0][0] = Antonia
dize[1][0] = Atanasia
dize[2][0] = Mihaela
dize[3][0] = Yovana
dize[4][0] = Yana
dize[5][0] = Teo

Press any key to continue . . .
```

Çözülmüş Ödevler

Ödev 3.2.1
n'e kadar çarpma tablosu yazdırılsın.

Program **şekil 3.2.5'**te verilmiştir.

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

int main() {    // Carpma tablosu

    const int m = 20;
    const int n = 20;
    int t[ m ][ n ];
    int i, j, k;
    cout << "Carpma tablonun hangi sayiya kadar yazdirilmasini istediğiniz sayiyi girin:";
    cin >> k;
    for( i = 1; i <= k; i++ )
        for( j = 1; j <= k; j++ )
            t[ i ][ j ] = i * j;
    system( "cls" );
    cout << setw( 20 ) << "" << "CARPMA TABLOSU \n" << endl;
    cout << setw( 3 ) << "";
    for( j = 1; j <= k; j++ )
        cout << setw( 5 ) << j;
    cout << endl << endl;
    for( i = 1; i <= k; i++ ) {
        cout << setw( 3 ) << i;
        for( j = 1; j <= k; j++ )
            cout << setw( 5 ) << t[ i ][ j ];
        cout << endl;
    }

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

Şekil 3.2.5

Programın bir çıktısı şudur:

```
                    CARPMA TABLOSU

      1    2    3    4    5    6    7    8    9   10   11   12   13   14   15

 1    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
 2    2    4    6    8   10   12   14   16   18   20   22   24   26   28   30
 3    3    6    9   12   15   18   21   24   27   30   33   36   39   42   45
 4    4    8   12   16   20   24   28   32   36   40   44   48   52   56   60
 5    5   10   15   20   25   30   35   40   45   50   55   60   65   70   75
 6    6   12   18   24   30   36   42   48   54   60   66   72   78   84   90
 7    7   14   21   28   35   42   49   56   63   70   77   84   91   98  105
 8    8   16   24   32   40   48   56   64   72   80   88   96  104  112  120
 9    9   18   27   36   45   54   63   72   81   90   99  108  117  126  135
10   10   20   30   40   50   60   70   80   90  100  110  120  130  140  150
11   11   22   33   44   55   66   77   88   99  110  121  132  143  154  165
12   12   24   36   48   60   72   84   96  108  120  132  144  156  168  180
13   13   26   39   52   65   78   91  104  117  130  143  156  169  182  195
14   14   28   42   56   70   84   98  112  126  140  154  168  182  196  210
15   15   30   45   60   75   90  105  120  135  150  165  180  195  210  225

Press any key to continue . . .
```

Ödev 3.2.2

İki boyutlu dizide en küçük ve en büyük eleman bulunsun. Okumak, yazdırmak , en küçük ve en büyük elemanı bulmak için ayrı fonksiyonlar yazılsın.

Açıklama: Uygulamada okumak2DDizi (), yazdirmak2DDizi (), max2DEleman() ve min2DEleman() fonksiyonları kullanılır. En küçük ve en büyük elemanı ararken fonksiyonlardan çıkış değerleri elemanın indisleridir.

```
1   #include <iostream>
2   #include <iomanip>
3   using namespace std;
4
5   void okumak2Ddizi( int&, int&, double [][ 10 ] );
6   void yazdirmak2Ddizi( const int, const int, const double [][ 10 ] );
7   void max2dEleman( const int, const int, const double [][ 10 ], int&, int& );
8   void min2dEleman( const int, const int, const double [][ 10 ], int&, int& );
9
10  int main() {    //İki boyutlu dizide en kucuk ve en buyuk elemani
11      const int m = 10;
12      const int n = 10;
13      double a[ m ][ n ] = {};
14      int i, j, satirlar, sutunlar, imax, jmax, imin, jmin;
15      double min, max;
16
17      okumak2Ddizi (satirlar, sutunlar, a);
18
19      system( "cls" );
20      cout << "İki buyutlu dizi  " << endl;
21      yazdirmak2DDizi (satirlar, sutunlar, a );
22
```

Şekil 3.2.7

```
23      //En kucuk ve en buyuk elemanin bulunmasi
24      min2DEleman (satirlar, sutunlar, a, imin, jmin);
25      max2DEleman (satirlar, sutunlar, a, imax, jmax);
26      cout << "\n Dizide en buyuk eleman: a["
27          << imax << "][" << jmax << "] = " << a[ imax ][ jmax ] << endl;
28      cout << "\n Dizide en kucuk eleman: a ["
29          << imin << "][" << jmin << "] = " << a[ imin ][ jmin ] << endl;
30
31      cout << endl << endl;
32      system( "Color 17" );
33      system( "pause" );
34      return 0;
35  }
36
37  void okumak2DDizi(int& sat, int& sut, double a [][ 10 ] ) {
38      cout << "Satirlarin sayisini girin(<=10): "; cin >> sat;
39      cout << "Sutunlarin sayisini girin(<=10): "; cin >> sut;
40      cout << "Dizinin elemanlarini satirlara gore girin: " << endl;
41      for( int i = 0; i < sat; i++ ) {
42          for( int j = 0; j < sut; j++ ) {
43              cout << "a[" << i << "][" << j << "] = ";
44              cin >> a[ i ][ j ];
45          }
46      }
47  }
48
49  void yazdirmak2dDizi( const int sat, const int sut, const double a [][ 10 ]
50      cout << setw( 2 ) << "";
51      for( int j = 0; j < sut; j++ )
52          cout << setw( 5 ) << j;
53      cout << endl << endl;
54      for( int i = 0; i < sat; i++ ) {
55          cout << setw( 2 ) << i;
56          for( int j = 0; j < sut; j++ )
57              cout << setw( 5 ) << a[ i ][ j ];
58          cout << endl;
59      }
60  }
61
62  void min2DEleman(const int sat, const int sut, const double b [][ 10 ],
63      int& iEnKucuk, int& jEnKucuk) {
64      double enKucuk = b[ 0 ][ 0 ];
65      iEnKucuk = 0; jEnKucuk = 0;
66      for( int i = 0; i < sat; i++ )
67          for( int j = 0; j < sut; j++ )
68              if( b[ i ][ j ] < enKucuk ) {
69                  enKucuk = b[ i ][ j ];
70                  iEnKucuk = i; jEnKucuk = j;
```

Şekil 3.2.7 (devam 1)

```
73
74      void max2DEleman( const int sat, const int sut, const double b [][ 10 ],
75          int& iEnBuyuk, int& jEnBuyuk ) {
76          double enBuyuk = b[ 0 ][ 0 ];
77          iEnBuyuk = 0; jEnBuyuk = 0;
78          for( int i = 0; i < sat; i++ )
79              for( int j = 0; j < sut; j++ )
80                  if( b[ i ][ j ] > enBuyuk ) {
81                      enBuyuk = b[ i ][ j ];
82                      iEnBuyuk = i; jEnBuyuj = j;
83                  }
84      }
```

Şekil 3.2.7 (devam 2)

Programın yürütülmesiyle çıktı şudur:

```
İki boyutlu dizi
      0    1    2    3

 0    3   -2   -7    9
 1    4   -6    5   34
 2   23  -12  -56  -33
 3    0    3    4   -2
 4   -5   -7    0    4

Dizide en buyuk eleman: a[1][3] = 34

Dizide en kucuk eleman: a[2][2] = -56


Press any key to continue . . .
```

Alıştırma Ödevleri

1. a[]m,n iki boyutlu dizisinin elemanlarının aritmetik ortalaması bulunsun.
2. Aynı boyutlara sahip iki boyutlu dizinin toplamı hesaplansın,
   $c[]_{m,n} = a[]_{m,n} + b[]_{m,n}$. (i,j'inci elemanın toplamı $c_{i,j} = a_{i,j} + b_{i,j}$'dir).
3. 1'den n2'ye kadar sayıları yılan şeklinde içerecek iki boyutlu a[][]n,n dizisi oluşturulsun. Örneğin, n = 3 için dizi şöyle görünecek:

```
1 2 3
6 5 4
7 8 9
```

4. $a[]_{m,n}$ iki boyutlu dizisinde satırlara göre en büyük eleman ve sütunlara göre küçük eleman bulunsun.
5. $q[][]_{n,n}$ iki boyutlu dizisinin ana köşegenin elemanlarının toplamı ve ikincil köşegenin elemanlarının toplamı ayrı ayrı bulunsun.

6. $a[]_{n,n}$ iki boyutlu dizisinin ana köşegeninin üzerindeki elemanların toplamı bulunsun.
7. $a[]_{m,n}$ iki boyutlu dizisinden i-nci satır ve j-nci sütun silinsin.
8. $a[]_{m,n}$ iki boyutlu dizisinin ai,j elemanından geçen köşegenler üzerinde yatan elemanların toplamı hesaplansın.
9. $a[]_{m,n}$ iki boyutlu dizisinin sütunlarını k basamak sola (veya sağa) döndürülsün.
10. Aşağıdaki $a[]_{m,n}$ iki boyutlu dizisini oluşturulsun.

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

Bilgiyi Kontrol Etme Soruları

1. İki boyutlu dizi nasıl bildirilir ve başlatılır?
2. 4 satır ve 3 sütunlu b[][] iki boyutlu dizisini nasıl bildirebiliriz?
3. Önceki ödevde tanımlanan b[][] dizisinde şu elemanlar var mı:
   b[3][4], b[4][3], b[0][4], b[0][3], b[3][0] ve b[4][0]?
4. Dizisi verilmiş olsun
```
int a[5][5] = {1, 2, 3, 4, 5, 6, 7, 8};
```
   a[1][3] elemanının değeri nedir?
5. Aşağıdaki dizi bildirilmişse
```
double d[10][10];
```
   komutunda hata nedir?
```
d[5][10] = 4.55;
```
6. Aşağıdaki program bölümü ne yazdıracaktır:
```
int i, j, dizi[5][5];
for(i = 0, j = 0; i < 5; i++, j++)
   dizi[i][j] = i + j;
for(i = 0, j = 0; i < 5; i++, j++)
   cout << dizi[i][j] << endl;
```
7. $a[]_{m,n}$ iki boyutlu dizisinin elemanlarını okumak için program bölümü yazılsın.
8. $a[]_{m,n}$ iki boyutlu dizisini yazdırmak için program bölümü yazılsın.

3.3 İŞARETÇİLER

Değişkenler belleğe yerleştirildiklerinde ayrılan bellek büyüklüğü, türlerine bağlıdır. (Bunu, veri türleri hakkında konuştuğumuzda gördük). Örneğin, aşağıdaki değişkenler bildirilirse:

```
int tamSayi;
float gerçekSayi;
```

o zaman tamSayi değişkeni belleğe yerleştirildiğinde, int türüden tam sayılar için tanımlanmış aralığından en büyük tam sayıyı kaydedecek büyüklüğünde bellek ayrılır. Aynı şekilde, gerçekSayi değişkeni belleğe yerleştirildiğinde, float türünden gerçek sayılar için tanımlanmış aralığındaki en büyük gerçek sayıyı kaydedecek kadar bellek ayrılır.

Değişkenler belleğe iki şekilde yerleştirilebilir:

- Statik.
- Dinamik

Programın çalıştırıldığı sırasında, **statik değişkenler** (İng. static variables) bellekte sabit bir adrese yerleştirilir. Onlar bu adreste değerleri değişebilse de programın sonuna kadar kalırlar. Adresleri sabit olduğundan ve her zaman bilindiği için, yeni değer okurken veya yazarken bunlara erişmek çok hızlıdır.

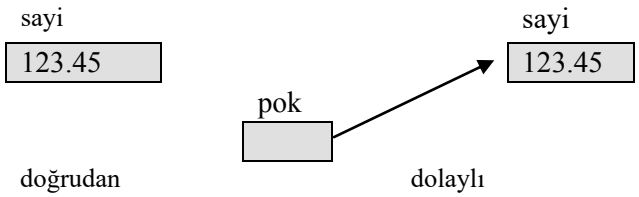
Dinamik değişkenler (İng. dynamic variables) ise oluşturuldukları anda (bildirimden sonra) belleğe yerleştirilir. Bu arada yerleşecekleri adres o anda belirlenir.

Modern programlama dillerinde, oluşturma sırasında herhangi bir boş adreste yerleşen dinamik değişkenlere erişmek için **işaretçiler** (İng. pointers) mekanizması kullanılır.

İşaretçilerin Bildirimi

İşaretçiler, değerleri bellek adresi olan değişkenlerdir. Bu tür değişkenlere **işaretçi değişkenleri** (İng. pointer variables) veya işaretçi türünden değişkenler denir. Değişkenlerin uygun türde veriler içerdikleri bilinmektedir: tam sayılar, gerçek sayılar, karakterler, dizeler, vb. İşaretçi değişkenleri, değişken adreslerinden başka veriler içeremez. Bu anlamda, işaretçi türü dışındaki herhangi türdeki değişkenlerin doğrudan bir değere (sayı, karakter, dize vb.) referanslandığını, işaretçi değişkenlerinin ise dolaylı olarak bir değere referanslandığını söyleyebiliriz.

Örneğin, sayi bir değişkense ve isar sayı değişkenine işaretçi ise, sayı değişkenine doğrudan ve dolaylı erişim, aşağıdaki şekilde gösterilmektedir.



İşaretçiler şu şekilde bildirilir:

```
tür *işaretçinin_adı;
```

tür – işaretçinin işaret edeceği değişkenlerin türüdür
* – işaretçi, yani işaretçi değişkeni bildirme işaretidir,

Örneğin

```
int *isar;
```

ile, sadece int türündeki değişkenleri işaret etmek için kullanılabilen isar işaretçisi bildirilir.

Aşağıdaki bildirim ile

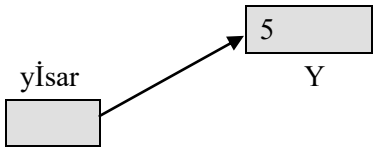
```
float *p;
```

sadece float türünden değişkenleri işaret etmek için kullanılabilen p işaretçisi bildirilir.

& Adres Operatörü

& operatörüne **adres operatörü** (İng. address operator) denir. Bu operatör tekli bir operatördür ve ondan sonra belirtilen işlenenin adresini döndürüyor. Örneğin, aşağıdaki bildirimler ve başlatmalar varsa:

```
int y = 5;
int *yPok;
```

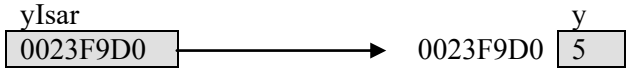


aşağıdaki komutla:

```
yİsar  = &y;
```

yİsar işaretçisine y değişkeninin adresi atanır. Bu yüzden, yİsar işaretçisinin "y'yi gösterdiği" söylenir.

y değişkeni 0023F9D0 adresinde bulunuyorsa, önceki komutu çalıştırdıktan sonra aşağıdaki durum elde edilir:



İşaretçiye adres atama, aşağıdaki gibi bildirilirken de yapılabilir:

```
int *yİsar  = &y;
```

```
yİsar  = y;
```

yazılamaz çünkü işaretçiye adresten başka bir değer atanamaz.

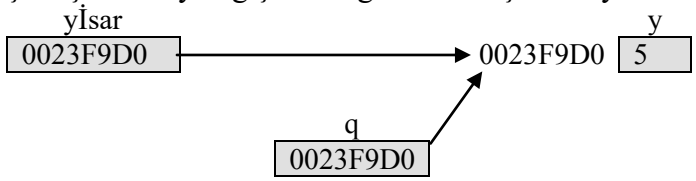
Eğer

```
int *q;
```

ile tam sayı değişkenlerini işaret etmek için kullanılabilecek q işaretçiyi bildirirsek, o zaman

```
q = yİsar;
```

komutuyla bu işaretçinin de y değişkenini gösterecek şekilde yönlendiriliyor.



Gerçek değişkenlere işaretçi bildirirsek

```
float *r;
```

o zaman

```
r = yİsar;
```

veya

```
r = &y;
```

yazılamaz çünkü r işaretçisi tam sayı değişkenleri göstermek için değil, sadece gerçek değişkenleri işaret etmek için kullanılır.

& adres operatörünün işleneni değişken olmalıdır.

Referanssızlaştırma Operatörü *

y değişkeni, *yİsar ile yİsar işaretçisi üzerinden da işaretlenebilir. ,

* operatörüne **dolaylı operatör** (İng. indirection operator) veya **referanssızlaştırma operatörü** (İng. dereferencing operator) denir. Bu operatör, işleneni tarafından işaret edilen değişkenin sahip olduğu değeri döndürür. Bu durumda *yİsar, yİsar işaretçisinin işaret ettiği değişkenin değerini döndürür.

Örneğin, aşağıdaki komut ile:

```
cout << *yİsar;
```

y değişkeninin değeri, yani 5 yazdırılacaktır. Aynı bu değer şu komutla da yazdırılacaktır.

```
cout << y;
```

Eğer

```
int *p = &x;
```

bildirirsek, o zaman aşağıdaki komutla

```
*p = *yİsar;
```

y değişkeninin değeri (= 5) x değişkenine atanacaktır.

Bir sonraki komut,

```
*p = *p + 3;
```

komutu ise, o zaman y değişkeninin değeri 3 için artacak ve 8 olacaktır.

Bir örnek daha verelim. Aşağıdaki komut sırasının olduğunu varsayalım:

```
int* ip;      // ip int degiskenin isaretcisidir
int b = 47;
int *ipb = &b;
```

Bu şekilde b ve ipb başlatılır, ipb tam sayısına işaretçi ise b'nin adresini içerir. İşaretçi aracılığıyla b değişkenine erişmek için, aşağıdaki gibi referanssızlaştırılır:

```
*ipb = 100;
```

Şimdi b, 47 yerine 100 değerini içeriyor.

İşaretçilerin başlatılması

İşaretçiler, bildirim sırasında veya bazı atama komutunda başlatılmalıdır. İşaretçi 0, NULL veya bir değişkenin adresi olarak başlatılabilir. NULL değerine sahip işaretçi herhangi bir değişkene işaret etmiyor. Bir işaretçiyi 0'a başlatmak, NULL'a başlatmakla eşdeğerdir, ancak programcılar arasında NULL daha yaygın olarak kullanılır. 0 değeri, bir işaretçiye atanabilen tek tam sayı değeridir.

Eş zamanlı başlatma olmadan işaretçi bildirildiğinde, hiçbir yere işaret etmediği vurgulamak gerekiyor. Bu nedenle, kullanılmadan önce belirli bir adresi işaret edecek şekilde ayarlanmalıdır.

Aşağıdaki örnek:

```
int *ip;
*ip=100;
```

„Null pointer assignment" türünden hata gösterecek çünkü ip herhangi bir bellek konumuna işaret etmediğinden dolayı, var olmayan bir bellek konumuna 100 değerini atamaya çalışmaktadır.

Doğru başlatma aşağıdaki gibidir:

```
int *ip, x;
ip = &x;
*ip = 100;
```

Örnekler

Örnek 3.3.1

& ve * operatörlerini kullanma.

Aşağıdaki programda (**Sekil 3.3.1**), * ve & operatörlerinin kullanımı gösterilmektedir. Bellek konumları, << operatörü yardımıyla on altılık sayılar olarak yazdırılır.

```
1   /* & ve * operatorlerin kullanimi */
2   #include <iostream>
3   using namespace std;
4
5   int main() {
6       int b;        /* b tam sayidir */
7       int* bİsar    /* bİsar tam sayiya gosteren isaretcidir */
8
9       b = 7;
10      bİsar = &b;  /* bİsar b'nin adresine gosteriyor */
11
12      cout << "b'nin adresi:" << &b
13          << "\nbİsar'in degeri: " << bİsar
14
15      cout << "\n\nb'nin degeri:" << b
16          << "\n*bİsar'in degeri:" << 0bİsar
17
18      cout << "\n\n* ve & birbirini tamamlayici"
19          << "oldugunu gosteriyoruz\n&*bİsar = " << &*bİsar
20          << "\n*&bİsar = " << *&bİsar << endl;
21
22      cout << endl;
23      system( "Color 17" );
24      system( "pause" );
25      return 0;
26  }
```

Şekil 3.3.1

Şunlar not edilmelidir:

a) a değişkeninin adresi ve aİsar işaretçisinin değeri aynıdır,

b) * ve & operatörleri birbirinin tamamlayıcısıdır.

Bu programın olası bir çıktısı şudur:

```
b'nin adresi: 002CF71C
bİsar'in degeri: 002CF71C

b'nin degeri: 7
*bİsari'in degeri: 7

* ve & birbirini tamamlayıcı oldugunu gosteriyoruz
&*bİsar = 002CF71C
*&bİsar = 002CF71C

Press any key to continue . . .
```

Örnek 3.3.2

İşaretçiyi fonksiyon argümanı olarak kullanma.

Şekil 3.3.2'deki programda, f() fonksiyonun işaretçi argümanı vardır. *p=5; komutu ile referanssızlaştırılır ve böylece x değişkeninin içeriği değişiyor.

```
1   // Fonksiyon argumani olarak isaretci
2   #include <iostream>
3   using namespace std;
4
5   void f( int* p );
6
7   int main() {
8       int x = 47;
9       cout << "x = " << x << endl;
10      cout << "&x = " << &x << endl;
11      f( &x );
12      cout << "x = " << x << endl;
13
14      cout << endl;
15      system( "Color 17" );
16      system( "pause" );
17      return 0;
18  }
19
20  void f( int* p ) {
21      cout << "p = " << p << endl;
22      cout << "*p = " << *p << endl;
23      *p = 5;
24      cout << "p = " << p << endl;
25  }
```

Şekil 3.3.2

Olası bir çıktı şu olabilir:

```
x = 47
&x = 0029F940
p = 0029F940
*p = 47
p = 0029F940
x = 5
```

İşaretçiler ve Diziler

Diziler ve işaretçiler arasında yakın bir ilişki vardır.

Örneğin, aşağıdaki dizi bildirilmiş ve başlatılmış olsun:

```
int z[] = {1, 2, 4, 8, 16};
```

z tanımlayıcısı, dizinin ilk elemanın adresinin değerine sahiptir ve *dizi elemanlarının türündeki değişkenlere işaretçi türündedir*. Bu durumda z, tam sayı değişkenlere işaretçidir, yani int türündedir.

Bu yüzden:

```
z;
```

ve

```
&z[0];
```

dizinin ilk elemanının adresini içerir.

İşaretçiler sıkça dizi elemanlarına erişmek için kullanılır. Bu, *dizi elemanlarının ardışık adreslere yerleştirildiği için mümkündür.*

Aşağıdaki bildirim verilmiş olsun

```
int z[10], *p;
```

Şu komutla:

```
p = z;        //dizinin adi elemana işaretçidir
              //bu, p = &z[0] ile aynidir
```

p işaretçisine z[] dizisinin ilk elemanının adres değeri atanır.

Dizinin ilk elemanının değeri şu şekilde elde edilebilir:

```
*z;
veya
z[0];
```

Örnekler

Örnek 3.3.3

Aşağıdaki programın çıktısı iki kez aynı on altılık adrestir.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       int a[ 10 ];
6       cout << "a = " << a << endl;
7       cout << "&a[0] = " << &a[ 0 ] << endl;
8
9       cout << endl;
10      system( "Color 17" );
11      system( "pause" );
12      return 0;
13
14  }
```

Şekil 3.3.3

```
a = 0038F938
&a[0] = 0038F938
```

Örnek 3.3.4

Aşağıdaki programla **(şekil 3.3.4)**, ilk elemana işaretçi aracılığıyla a[] dizisine değerlerin nasıl atandığı gösterilmektedir.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       int a[ 10 ];
6       int i;
7       int* ip = a;
8       for( i = 0; i < 5; i++ ) {
9           *( ip + i ) = 10 * i;
10          cout << *( ip + i ) << endl;
11          cout << a[ i ] << endl;
12      }
13
14      cout << endl;
15      system( "Color 17" );
16      system( "pause" );
17      return 0;
18
19  }
```

Şekil 3.3.4

Elde edilen çıktı şudur:

```
0
0
10
10
20
20
30
30
40
40

Press any key to continue . . .
```

Adres Aritmetiği

İşaretçiler üzerine aritmetik işlemler yapılabilir, Bu işlemler şunlardır:

- İşaretçiyi artırma (++).
- İşaretçiyi azaltma (– –).
- Tamsayı değeri ekleme (+ veya +=).
- Tamsayı değeri çıkarma (– veya –=).
- Bir işaretçiyi diğerine ekleme (çıkarma).

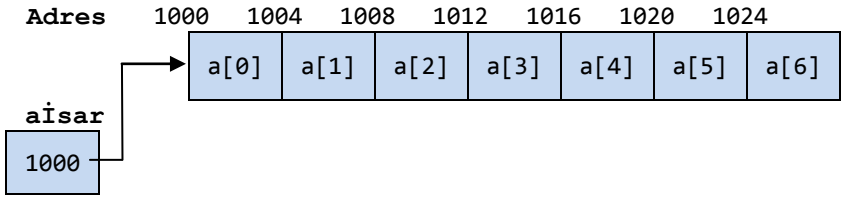
7 elemanlı a[] tamsayı dizisinin ve dizinin ilk elemanını gösteren aİsar işaretçisinin bildirimi verilmiş olsun

```
int a[7];
int* aİsar  = a;
```

a[] dizisinin ilk elemanının 1000 bellek adresinde bulunduğunu tahmin edelim. Bu, aİsar işaretçisinin 1000 değerine sahip olduğu anlamına gelir.

Ayrıca, aşağıdaki şekilde gösterildiği gibi, her tam sayı değişkenin 4 bayt kapladığını tahmin edeceğiz.

Aşağıdaki işaretçi işlemleri inceleyelelim.



a) İşaretçiyi artırma ve azaltma

Aşağıdaki komutla

```
++aİsar;
```

aİsar işaretçisi 1 için artacak, yani bir sonraki a[1] elemanına işaret edecektir. Demek ki, onun içeriği 1000 değil 1004 olacaktır.

Genel olarak, bir işaretçi 1 için artarsa (azalırsa), adresi, işaret ettiği değişkenin kapladığı kadar bayt için artar (azalır). Bizim durumumuzda, dizinin elemanlarının tam sayı değerler olduğundan ve her biri 4 bayt kapladığından dolayı, aİsar'ın içeriği 1004 olacaktır. a[] dizisi double türünden olsaydı ve double türündeki değişkenlerin 8 bayt kapladığını tahmin edersek, o zaman aİsar'ın içeriği 1008 olurdu.

b) İşaretçiye/işaretçiden tam sayı değeri ekleme ve çıkarma

Aşağıdaki komutla:

```
aİsar  = aİsar  + 3;
```

veya

```
aİsar  += 3;
```

işaretçinin içeriği 3 için artacak (1000 + 3 * 4 = 1012) ve (bizim örneğimizde) a[3] elemanına gösterecektir.

Bir sonraki komut şu ise

```
aİsar  = aİsar  - 2;
```

işaretçi a[1] elemanını gösterecektir.

c) Bir işaretçiyi diğerine eklemek ve bir işaretçiyi diğerinden çıkarmak

İki işaretçi daha bildirilmişse

```
int *aP1,*aP2;
```

ve dizinin 2. ve 5. elemanlarını gösterecek şekilde ayarlanmışsa:

```
aP1 = a+2;
aP2 = a+5;
```

O zaman şu komutla:

```
aİsar  = a + aP2 - aP1;
```

aİsar işaretçisi a[3] elemanına gösterecektir.

İşaretçi aritmetiğini, dizeler dışındaki diğer veri türleri üzerine gerçekleştirmenin anlamı yoktur.

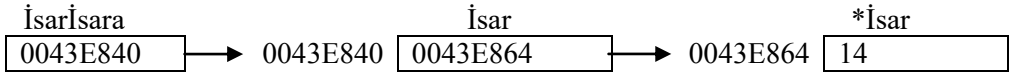
Eşitlik operatörleri veya ilişkisel operatörler, işaretçilere yazılan adresleri karşılaştırarak işaretçiler üzerinde de kullanılabilir. Örneğin, bir işaretçinin bir yere işaret edip etmediğini kontrol etmek için, NULL ile karşılaştırılmalıdır.

Bir işaretçi başka bir işaretçiye de işaret edebilir.

Örneğin,

```
int **İsarİsara  = &İsar;
```

ile İsarİsara işaretçisi, İsar işaretçisini gösterecek şekilde bildirilir ve başlatılır.



Not:

Gördüğümüz gibi, aşağıdaki bildirilmişse

```
int a[10], *ip;
```

o zaman aşağıdaki komutlar doğrudur:

```
ip = a;
ip++;
```

Ancak, aşağıdaki komutlar doğru değil:

```
a = ip;
ve
a++;
```

Örnekler

Örnek 3.3.5

Şekil 3.3.5'teki programla, tam sayı ve gerçek türdeki işaretçilerin içeriği gösterilir.

Elde edilen çıktı şöyle olabilir:

```
ip = 1243916
ip = 1243920
dp = 1243828
dp = 1243836

Press any key to continue . . .
```

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       int i[ 10 ];
6       double d[ 10 ];
7       int* ip = i;
8       double* dp = d;
9       //Tamsayi degerlere isaretciler
10      cout << "ip = " << ( long ) ip << endl;
11      ip++;
12      cout << "ip = " << ( long ) ip << endl;
13      // Gercek degerlere isaretciler
14      cout << "dp = " << ( long ) dp << endl;
15      dp++;
16      cout << "dp = " << ( long ) dp << endl;
17
18      cout << endl;
19      system( "Color 17" );
20      system( "pause" );
21      return 0;
22
23  }
```

Şekil 3.3.5

Görüldüğü gibi tam sayı değişkenleri
(1243920 – 1243916 =) 4 bayt yer kaplarken, gerçek değişkenler
(1243836 – 1243828 =) 8 bayt kaplamaktadır.

Örnek 3.3.6

Bu örnek, işaretçileri yönlendirmenin farklı yollarını ve işaret ettikleri değişkenlere değer atamasını göstermektedir.

```
1   // isaretcilerle isleme
2   #include <iostream>
3   using namespace std;
4
5   int main() {
6       int sayilar [ 5 ];
7       int* p;
8       // Diziye isaretcilerle deger atama
9       p = sayilar             *p = 10;            // p[ 0 ]
10      p++;                    *p = 20;            // p[ 1 ]
11      p = &sayilar [ 2 ];     *p = 30;            // p[ 2 ]
12      p = sayilar  + 3;       *p = 40;            // p[ 3 ]
13      p = sayilar             *( p + 4 ) = 50;    // p[ 4 ]
```

Şekil 3.3.6

```
14      for( int n = 0; n < 5; n++ )
15          cout << sayilar [ n ] << ", ";
16      cout << endl;
17      system( "Color 17" );
18      system( "pause" );
19      return 0;
20  }
```

Şekil 3.3.6 (devam)

Elde edilen çıktı şudur:

```
10, 20, 30, 40, 50,
```

new ve delete Komutları

new komutunun şu formu var:

işaretçi = new tür;

Bu komutla, bellekte uygun türden değişken oluşturulur ve işaretçiye oluşturulan değişkenin adresi atanır. (Aksi takdirde, işaretçi NULL değeri alır).

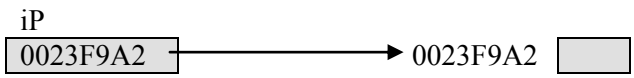
Örneğin, aşağıdaki işaretçi bildirilmişse

```
int *iP;
```

o zaman şu komutla:

```
iP = new int;
```

bellekte adlandırılmamış değişken oluşturulur ve onun adresi iP işaretçisine atanır.



Adlandırılmamış değişkenin değeri, oluşturulduğu sırasında şu şekilde atanabilir.

```
int *iP;
ip = new int(25);
```

veya birlikte

```
int *iP = new int(25);
```

Aşağıdaki bildirimle

```
int *aİsar  = new a[4];
```

int türündeki 4 elemanlık dizi için bellek ayrılır, aİsar işaretçisine ise dizinin ilk elemanın adresi atanmaktadır.

delete komutu, bellek konumunu silmek (serbest bırakmak) için kullanılır.

Örneğin

```
delete iP;
```

ile iP işaretçisi tarafından işaret edilen değişkenin belleği serbest bırakılır (silinir), ve bu arada tanımsız kalıyor.

Dizeleri silerken, şu sözdizimi kullanılarak:

```
delete [] işaretçi;
```

işaretçinin dizi elemanlarına gösterdiği vurgulanıyor.

Örneğin

```
delete [] aİsar;
```

Alıştırma ödevleri

1. int, float ve char türünde üç işaretçi ve karşılık gelen türden üç değişken a, b ve c'nin tanımlanacağı program yazın. Değişkenler başlatılsın ve ardından işaretçiler uygun değişkenleri gösterecek şekilde ayarlansın. İşaretçileri kullanarak a, b ve c değişkenlerinin değerleri ve adresleri yazdırılsın.
2. " Null pointer assignment " hatası oluşturcak bir program yazın ve ardından hatanın nasıl düzeltilebileceğini gösterin.
3. a[]n tamsayı eleman dizisinin girildiği program oluşturun ve ardından her elemanın değerini adresiyle birlikte yazdırın.
4. a[]n double eleman dizisinin girildiği program oluşturun ve ardından her elemanın değerini adresiyle birlikte yazdırın. Önceki ödevin çıktısında ne gibi fark görüyorsunuz?
5. x[]n karakter dizisinin ve c karakterinin tanımlandığı program yazın. Program, c karakterinin x[]n dizisinde bulunup bulunmadığı sorusuna cevap vermesi lazım. Bu arada, dizinin elemanlarına erişmek için işaretçi kullanılsın.

Bilgiyi Kontrol Etme Soruları

1. Değişkenlere bellek atamanın hangi yollar vardır? Onları açıkla.
2. İşaretçinin bildirilmesi nasıl yapılır?
3. * operatörünüm adı nedir ve neyi tanımlamaktadır?
4. & operatörünün adı nedir ve neyi tanımlamaktadır?
5. İşaretçinin içeriği nedir ve işaretçi değişkeninin içeriği nedir?
6. Herhangi bir işaretçi herhangi bir türde değişkene işaret edebilir mi? Cevabı açıklayın.
7. Aşağıdaki program bölümünde yanlış olan nedir:

```
int *isar;
char a;
isar = &a;
```

8. Aşağıdaki program bölümünde yanlış olan nedir:

```
int *isar;
*isar = 5;
```

9. Aşağıdaki komutlar dizisinde:

```
float *fp, f=100f;
```

```
fp = &f;
f = 200f;
*f = 300;
```

f değişkeninin değeri ne olacaktır?

10. Aşağıdaki komutlar dizisinde yanlış olan nedir?

```
char *c, ch = 'A';
cout << *c;
```

11. İsar işaretçisi a değişkenini gösteriyorsa, &*isar ifadesi neyi temsil eder?
12. İşaretçiler üzerine adres aritmetiğini kullanmak ne zaman anlamlı olur?
13. new komutuyla işaretçi nasıl bildirilir (ve başlatılır)?
14. new komutuyla dizi işaretçisi nasıl bildirilir?
15. delete komutuyla değişkene ve diziye bir işaretçi nasıl silinir

3.4 Dizeler

1.6 C++'ya Giriş alt bölümündeki **Yazdırma Komutları ve Veri Türleri** alt başlıklarında ve ayrıca **1.8 Verileri Okuma ve Yazdırma** alt bölümündeki **Karakter ve Dize Verilerini Okuma ve Yazdırma** alt başlığında, **dize** (İng. string) terimiyle karşılaştık. Ayrıca birçok örnekte dize değişkenleri veya dize dizileri kullandık.

Dizelerin, karakter dizileri[5] olarak karakter verilerinden oluşan veriler olduğunu söyledik. Bu tür veriler, basit char türünden oluşan ayrı yapılar olarak ele alınır ve string türü olarak adlandırılır.

Dizelerle, birleştirme, karşılaştırma, bir dizenin uzunluğunu belirleme ve daha çok sayıda farklı işlemler gerçekleştirilebilir. Genellikle, dize işlemleri, **C++ standart kütüphanesinin <string>** kütüphensinde bulunan fonksiyonlarla gerçekleştirilir. Bu fonksiyonları programlarda kullanmamız için #include yönergesiyle programın başında <string> kütüphanesinin bulunması gerekmektedir.

```
#include <string>
```

Dizelerle Çalışma Fonksiyonları

C++'da, dizelerle çalışma çok yaygın kullanılır. <string> kütüphanesinde çok sayıda fonksiyon vardır. Burada en sıkça kullanılan fonskiyonları açıklayacağız.

Örneklerde aşağıdaki değişkenleri kullanacağız:

```
string s, s1, s2;
s = "C++ en iyi programlama dilidir";
```

[5] Dizeler, metinsel dizelerinden, sonunda sıfırıncı karakter olmadığından farklıdır.

```
s1 = "İyi";
s2 = "gunler";
```
s3 – dize değişkeni
sonuc – dize değişkeni
altdize – dize değişkeni
karakter – karakter değişkeni
n – tamsayı değişkeni
dogru – mantıksal değişken

Dizedeki her karakterin 0 (ilk karakter), 1 (ikinci karakter) vs. konumu olduğunu hatırlayalım. Son karakterin, karakter sayısı (dize uzunluğu) eksi 1'e eşit konumu vardır. Örneğin, s dizesinin uzunluğu 33'tür. Bu nedenle, karakter konumları 0, 1, 2... 32'dir.

Şimdiye kadar metinde, dizelerle şu iki işlemi kullandık:

– Bir dizeyi = operatörü ile diğerine atama.

Örnek: `s3 = s1;`

–Dizilerin + operatörü ile birleştirilmesi.

Örnek: `s3 = s1 + s2;`

Bu iki dize işlemi için aşağıda açıklayacağımız özel fonksiyonlar vardır:

- **Bir dizeyi başka dizeye atama** **assign()**

`s3.assign(s1)` s1 dizenin değeri, s3 dizesine atanır.

Etkisi, şu komutla aynıdır

```
s3 = s1;
```

Örnek:

```
s1 = "İyi";
s3.assign(s1);
```
s3 dizisine "İyi" sabiti atanır.

- **İki dizeyi birleştirme** **append()**

`s1.append(s2)` s1 ve s2 dizelerini birleştiriyor.

Etkisi, şu komutla aynıdır

```
s1 = s1 + s2;
```

Örnekler

```
s1 = "İyi";
s2 = "günler";
s1.append(s2);
```
s1 dizesi "İyigunler" değerini alacaktır

Bunları bir boşlukla ayırmak için:

```
s1.append(" ");
```

```
s1.append(s2);
```
s1 dizesi "İy igunler" değerini alacaktır s1
veya tek bir komutla:
```
(s1.append(" ")).append(s2);
```
s1 dizesi "İy igunler" değerini alacaktır

Etkisi, şu komutla aynıdır
```
s1 = s1 + " " + s2;
```

- **Dizenin uzunluğu** **length()**

```
s.length()
```
s dizesinin uzunluğu belirleniyor

Örnek:
```
n=s.length();
```
n 33 değerini alacaktır.

- **Dizenin uzunluğu** **size()**

```
s.size()
```
s dizesinin uzunluğu belirleniyor. Bu fonksiyonun lehgt () fonksiyonuyla aynı etkisi vardır.

Örnek:
```
n=s.size();
```
n 33 değerini alacaktır.

- **Dizenin en büyük olası uzunluğu** **max_size()**

```
s.max_size()
```
Uygulamanın çalıştırıldığı sisteme bağlı olarak, s dizesinin ulaşabileceği maksimum uzunluğu belirleniyor.

Örnek:
```
n=s.max_size();
```
n 2 147 483 647 değerinin alacaktır.

- **Dizeleri karşılaştırmak** **compare()** s1

```
s1.compare(s2)
```
dizesinin s2 dizesine eşit, ondan büyük veya küçük olup olmadığını kontrol ediyor.

Karşılaştırma, ASCII tablosuna göre dize karakterlerinin sayısal değerleri karşılaştırılarak sözlüksel olarak gerçekleştirilir.

pozitiftir, eğer s1 > s2,
negatiftir, eğer s1 < s2.
0 eğer s1 = s2,

Dizelerin tüm karakterleri eşitse ve birinin sonunda boşluk varsa, daha uzun olan dize daha büyüktür. Örneğin, "iyi" < "iyi ".

Örnekler
```
s1 = "iki";
s2 = "iyi";
```

Aşağıdaki komutla:

```
n = s1.compare(s2);
```
n tamsayı değişkeni -1 değerini alacak çünkü ikinci pozisyondaki karakterler farklıdır, 'a'= 97 < 'r'= 114.

Şu komutlarla:

```
n = s1.compare("iki");
```
n, 0 değerini alacaktır

```
n = s1.compare(s2);
```
n, -1 değerini alacaktır ('a' < 'r').

```
n = s1.compare("İyi");
```
n, 1 değerini alacaktır ('d' = 100 > 'D' = 68).

- **İki dizenin değerlerinin değiştirilmesi** **swap()**

```
s1.swap(s2)
```
s1 ce s2 dizelerinin içeriklerini değiştiriyor.

Örnek:

```
s1 = "iki";
s2 = "iyi";
s1.swap(s2);
```
s1, "iyi" değerini, s2 ise "iki" değerini alacaktır.

- **Dizenin alt dizesi** **substr()**

```
s.substr(konum, uzunluk)
```
s dizesinden, konum (= 0, 1... lenght(w) –1) konumundan (= 0, 1... lenght(w) –1) uzunluk değişkeni (> 0) ile belirtilen uzunlukla alt dize ayrılır.

Örnekler

```
s1 = s.substr(17, 7);
```
s1, "program" değerini alacaktır.

```
s1 = s.substr(17, 20);
```
s1, "programlama dilidir" değerini alacaktır.

```
s1 = s.substr(3, 1);
```
s1, "" değerini alacaktır.

```
s1 = s.substr(37, 3);
```
Çalıştırma sırasında hata oluşacaktır.

- **Dizeden karakter ayırmak** **at()**

```
s.at(konum)
```
konum (= 0, 1, 2... s.length() – 1) değişkeniyle verilmiş konumdaki karakterin ayrılması

Örnek:

```
karakter  = s.at(9);
```
karakter, 'i' değerini alacaktır.

- **Dizede alt dize veya karakter arama6** [6] **find() rfind()**

`s.find(dize veya karakter)`

Soldan sağa aranırsa, dize veya karakterin ilk geçtiği konumu döndürür.

`s.rfind(string veya karakter`

Sağdan sola aranırsa, dize veya karakterin ilk geçtiği konumu döndürür.

Örnekler

```
karakter  = 'j';
n = s.find(karakter ); n
= s.rfind(karakter );
```

Karakter boşluksa `' '`,

n 7 değerini alacaktır.
n 28 değerini alacaktır
n 3 ve 27 değerlerini alacaktır.

```
altdize  = "gram";
n = s.find(potstring);
```

n 14 değerini alacaktır..

```
altdize  = "graf";
n = s.find(potstring);
```

n, -1 değerini alacaktır çünkü alt dize bulunamadı

- **Karakterlerin bir konumdan dizenin sonuna kadar silinmesi** **erase()**

`s.erase(konum)`

konum değişkeni ile belirtilen konumdan (kendisi dahil) dizedeki karakterleri dizenin sonuna kadar siliyor.

Örnek:

```
s.erase(16);
```

s "C++ en iyi" değerini alacak

- **Alt dizeyi dizeyle değiştirme** **replace()**

`s.replace(konum, karaktersayisi, dize)`

s dizesinde, *konum* konumundan başlayarak, *karaktersayisi* konumundaki alt *dize*, *dize* dizesiyle değiştirilir. (*karaktersayisi* konumundan alt dize siliniyor).

`s.replace(konum, karaktersayisi, dize, konumdan, karaktersayisi1)`

Alt dizeyle yer değiştirme, aynı zamanda *dize* dizesinden, *konumdan* konumundan ve *karaktersayisi1* uzunluğunda gerçekleştirilebilir.

[6] gibi başka arama fonksiyonları da vardır. find_first_of, find_last_of, find_first_not_of.

Örnek

```
s3 = " en en";
s.replace(9, 7, s3);
```

s, şu değeri alacaktır:

```
"C++ en en en programlama dilidir".
```

```
s.replace(9, 7, s3, 1, 3);
```

s şu değeri alacaktır:

```
"C++ en en programlama dilidir". .
```

- **Dizeye başka bir dize (veya alt dize) ekleme** **insert()**

```
s.insert(konum, dize)
```

s dizisinde, *konum* konumundan başlayarak, *dize* dizesi eklenir.

```
s.insert(konum, dize, konumdan, karaktersayisi1);
```

s dizesinde, *konum* konumundan başlayarak, *konumdan* başlayarak *karaktersayisi1* karakterden oluşan *dize* dizesinden alt dize eklenir.

Örnekler

```
s3 = " ve en kolay";
s.insert(16, s3);
```

s şu değeri alacaktır:

```
" C++ en iyi ve en kolay programlama
dilidir ".
```

```
s.replace(16, s3, 0, 6);
```

s şu değeri alacaktır

```
"C++ en iyi ve en programlama
dilidir".
```

- **• Dizenin boş olup olmadığını kontrol etme** **empty()**

```
s.empty()
```

s dizesi boşsa true değeri döndürür.

Örnek

```
string str;
do {
       cout << "Adinizi girin: "; getline(cin,
dize); } while(str.empty());
```

Dizelerle çalışmak için listelenen fonksiyonlardan bazıları aşağıdaki örneklerde gösterilmiştir.

Örnekler

Örnek 3.4.1

```
// Dizelerle calisma fonksiyonlari: assign( ), append( ), length( ) ve compare( )

#include <iostream>
#include <string>
#include <iomanip>

using namespace std;

int main() {

    string s, s1, s2, s3, sonuc, altdize;
    char karakter;
    int n, m;
    bool dogru;
    s = " C++ en iyi programlama dilidir";
    s1 = "İyi";
    s2 = "gunler";
    s3 = " ";
    cout << "Bir dizeyi baska dizeye atama" << endl;
    cout << "s1 = " << s1 << ", s3 = "<< s3;
    s3.assign (s1);
    cout << ", s3 = (s1) " << s3 << endl;
    cout << "Dizelerin birlestirilmesi" << endl;
    s1.append (" ");
    cout << "s1 = " << s1;
    s1.append(s2);
    cout << ", s2 = " << s2 << ", s1 s2 =  " << s1 << endl;
    cout << "Dizenin uzunlugu" << endl;
    n = s.length( );
    cout << " \ " << s << "\ dizesinin uzunlugu "<< n << "karakterdir." << endl;
    n = s1.size( );
    cout << " \ " << s1 << "\ dizesinin uzunlugu "<< n << "karakterdir." << endl;
    cout << " Dizeleri karsılastirma" << endl;
    s1 = "iki";
    s2 = "iyi";
    n = s1.compare ("iki");
    cout << " << s1 << " dizesini " \"iki\" dizesiyle karsilastirirsak sonuç"
       <<  n << "verir" << endl;
    n = s1.compare (s2);
    cout << s1 << "\dizesini \" << s2 << " \ dizesiyle karsilastirirsak sonuç"
        <<  n << "verir" << endl;
    n = s1.compare ("iyi");
    cout << s1<< "dizesini " \"iyi\" dizesiyle karsilastirirsak sonuç"
        <<  n << "verir" << endl;

```

Şekil 3.4.1

```
46      cout << endl;
47      system("Color 17");
48      system("pause");
49      return 0;
50  }
```

Şekil 3.4.1 (devam)

```
 Bir dizeyi baska dizeye atama
s1 = İyi, s3 =  , s3(s1) İyi
 Dizelerin birlestirilmesi
s1 = İyi, s2 = gunler, s1 s2 = İyi gunler
 Dizenin uzunlugunu
"C++ en iyi programlama dilidir" dizesinin uzunlugu 30 karakterdir.
" İyi gunler" dizesinin uzunlugu 10 karakterdir
  Dizeleri karsılastırma
"iki" dizesini "iki" dizesiyle karsılastırırsak sonuc 0 verir
"iki" dizesini "iyi" dizesiyle karsılastırırsak sonuc -1 verir
"iyi" dizesini "iyi" dizesiyle karsılastırırsak sonuc 1 verir
```

Örnek 3.4.2

```
1   // Dizelerle calısma fonksiyonlari: swap( ), substr( ) ve at( )
2
3   #include <iostream>
4   #include <string>
5   #include <iomanip>
6   using namespace std;
7
8   int main() {
9
10      string s, s1, s2, s3, sonuc, altdize;
11      char karakter;
12      int n,m;
13      bool dogru;
14      s = " C++ en iyi programlama dilidir";
15      s1 = "İyi";
16      s2 = "gunler";
17      s3 = " ";
18      cout << " İki dizenin degerini degistirme" << endl;
19      cout << "Degistirmeden once: s1 = " << s1 << ", s2 = " << s2 << endl;
20      s1.swap (s2);
21      cout << "Degistirmeden sonra: s1 = " << s1 << ", s2 = " << s2 << endl;
22      cout << "Dizeden altdize ayirmak" << endl;
23      cout << "Dize: \ " << s << " \"  << endl;
24      s1 = s.substr (11,7);
25      cout << "Altdize (11,7): " << s1 << endl;
26      s1 = s.substr (11,20);
27      cout << "Altdize (11,20): " << s1 << endl;
28      s1 = s.substr(3,1);
29      cout << "Altdize (3,1): "  << s1 << endl;
30      cout << " Dizeden karakter ayirmak" << endl;
31      karakter = s.at(9);
```

Şekil 3.4.2

```
32      cout <<"9-ncu konumdaki karakter sudur:" << karakter << endl;
33
34      cout << endl;
35      system("Color 17");
36      system("pause");
37      return 0;
38  }
```

Şekil 3.4.2 (devam)

```
 İki dizenin degerini degistirme
Degistirmeden once: s1 = iyi, s2 = gunler
Degistirmeden sonra: s1 = gunler, s2 = iyi
 Dizeden altdize ayirmak
Dize: "C++ en iyi programlama dilidir"
Altdize (11.7): program
Altdize (11,20): programlama dilidir
Altdize (3,1):
 Dizeden karakter ayirmak
9-ncu konumdaki karakter sudur: i
```

Çözülmüş Ödevler

Ödev 3.4.1
Dizede bir karakterin kaç kez geçtiği belirlensin.

Şekil 3.4.3'teki programda, girilen karakterin dizede (cümlede) kaç kez geçtiğini belirleyen dizedeKarakter() fonksiyonu, main() ana fonskiyondan sonra tanımlanır. Bu nedenle, dizedeKarakter() fonksiyonun prototipi main() fonksiyonundan önce belirtilir.

```
1   #include <iostream>
2   #include <string>
3   using namespace std;
4
5   int dizedeKarakter( char, string );
6
7   int main() {     //dizede (fonskiyon ile) bir karakterin tekrarlanma sayisi
8
9       string cumle;
10      char karakter
11      cout << "Bir cumle girin: ";
12      getline (cin, cumle);
13      cout << "Cumlede aramak istediginiz karakteri girin: ";
14      cin >> karakter;
15      int tekrarlanmaSayisi = dizedeKarakter (karakter, cuml);
16      if (tekralanmaSayisi > 0)
17         cout << "\n" << karakter << "karakteri" << cümle << "cumlesinde \n\""
18          << tekrarlanmaSayisi << "\ "\n kez tekralanir." << endl;
```

Şekil 3.4.3

```
19          else
20              cout << "\n" << karakter << "karakteri" << cumle << "cumlesinde \n"
21              <<  "\n yoktur"   << endl;
22
23          cout << endl;
24          system( "Color 17" );
25          system( "pause" );
26          return 0;
27      }
28
29      int dizedeKarakter ( char karakter, string s) {
30          int kez = 0;
31          int uzunluk = s.length( );
32          for (int konum = 0; konum < uzunluk; konum++) {
33              if (s.at(konum) == karakter)
34                  kez++;
35          }
36          return kez;
37      }
```

Şekil 3.4.3 (devam)

Programın çalıştırılmasıyla bir çıktı şudur:

```
Bir cumle girin: Ben Koprulu dogumluyum
Cumlede aramak istediginiz karakteri girin:

m karakteri
"Ben Koprulu dogumluyum" cumlesinde
3 kez tekrarlanir

Press any key to continue . . .
```

Ödev 3.4.2

Bir dize girilsin ve kaç küçük harf ve kaç büyük harf olduğu sayılsın.

Dizede bir karakterin harf olup olmadığını test etmek için isalpha() fonksiyonu kullanılır. isupper() fonksiyonu harfin büyük harf olup olmadığını kontrol etmek için kullanılır, harfin küçük olup olmadığını kontrol etmek için ise islower() fonksiyonu kullanılır.

Ödevin programı **şekil 3.4.4**'te verilmiştir.

Programın çalıştırılmasıyla bir çıktı şudur:

```
Dize girin: C++ en iyi Programlama dilidir

 ++ en iyi Programlama dilidir dizesinde
 2 buyuk harf ve 23 kucuk harf vardir

Press any key to continue . . .
```

```
1   #include <iostream>
2   #include <string>
3    using namespace std;
4
5   int main() {    //Dizede buyuk ve kucuk harflerin sayisi (for ile)
6
7       char karakter:
8       int buyukHarfSayisi = 0; kucukHarfSayisi = 0;
9       string s;
10      cout << "Dize girin: "; getline( cin, s );
11      int uzunluk = s.length( );
12      for( int sayac = 0; sayac < uzunluk; sayac ++ ) {
13          karakter = s.at( sayac );
14          if( isalpha( karakter ))
15              if( isupper ( karakter ))
16                  buyukHarfSayisi++;
17              else
18                  kucukHarfSayisi++;
19      }
20      cout << s <<  "\n dizesinde \n" << buyukHarfSayisi
21          << " buyuk harf ve " << kucukHarfSayisi << "kucuk harf vardir" << end
22
23      cout << endl;
24      system( "Color 17" );
25      system( "pause" );
26      return 0;
27  }
```

Şekil 3.4.4

Dizeyi Sayıya Dönüştürme Fonksiyonları

Dizeleri okurken >> operatörü ile sadece ilk kelimenin, yani ilk boşluğa kadar okunduğunu gördük. Bu yüzden, tüm dizeyi okuyan getline() fonskiyonunu inceledik.

Ayrıca, bir programda verileri hem >> operatörü hem getline() fonksiyonuyla okursak, >> operatörüyle sayısal değer okuduktan sonra yeni satır karakterinin okunmamış kalmasına dikkat etmemiz gerektiğini de gördük ve ayrıdan cin.get() fonksiyonu veya cin.getline() fonksiyonuyla okunmalıdır.

Okuma hataları konusunda endişelenmemek için, belki giriş akışındaki sayısal değerler de karakter dizileri (sayılar veya rakamlar ve diğer karakterler) olduğundan, getline() fonksiyonunu kullanmak en basit olabilir. Bu, sayısal değerin okunması durumunda, dizeden sayıya dönüştürme özel fonksiyonlarla sağlanır.

Aşağıdaki fonksiyonlar sunulmaktadır:

stoi(s)	– s dizesinin int türünde tamsayıya dönüştürülmesi
stol(s)	– s dizesinin long türünde tamsayıya dönüştürülmesi,
stoll(s)	– – s dizesinin long long türünde tamsayıya dönüştürülmesi,
stoul(s)	– s dizesinin unsigned long türünde tamsayıya dönüştürülmesi,
stoull(s)	– s dizesinin unsigned long long türünde tamsayıya dönüştürülmesi,
stof(s)	– s dizesinin float türünde gerçek sayıya dönüştürülmesi,
stod(s)	– s dizesinin double türünde gerçek sayıya dönüştürülmesi,
stold()	– s dizesinin long double türünde gerçek sayıya dönüştürülmesi.

Bu fonksiyonlar aşağıdaki örnekte gösterilmiştir.

Örnek 3.4.3

```
1     // Dizenin sayiya donusturulmesi
2
3   #include <iostream>
4    #include <string>
5    using namespace std;
6
7   int main() {
8       string s;
9       int sayiİnt;
10      long sayiLong;
11      long long sayiLongLong;
12      unsigned long sayiİsaretsizLong;
13      unsigned long long sayiİsaretsizLongLong;
14      float sayiFloat;
15      double sayiDouble;
16      long double double sayiLongDouble;
17      cout << "Dize girin \"2123456789\":  "; getline (cin,s);
18      sayiİnt = stoi (s);
19      cout << "\t\tint turunden sayi sudur: " << sayiİnt << endl;
20      cout << "Dize girin \"2123456789\":  "; getline (cin,s);
21      sayiLong = stol (s);
22      cout << "\t\ltong turunden sayi sudur: " << sayiLong << endl;
23      cout << "Dize girin \"9123456789123456789\":  "; getline (cin,s);
24      sayiLongLong = stoll (s);
25      cout << "\t\tlong long turunden sayi sudur: " << sayiLongLong << endl;
26      cout << "Dize girin \"4123456789\":  "; getline (cin,s);
27      sayiİsaretsizLong = stoul (s);
28      cout << "\t\tunsigned long turunden sayi sudur: " << sayiİsaresizLong << endl;
29      cout << "Dize girin \"18123456789123456789\":  "; getline (cin,s);
30      sayiİsaretsizLongLong = stoull(s);
```

Şekil 3.4.5

```
31      cout << "\t\tusigned long long turunden sayi sudur: "
32           << sayiİsaretsizLongLong << endl;
33      cout << "Dize girin \"1.23456789\":  "; getline (cin,s);
34      sayiFloat = stof(s);
35      cout << "\t\tfloat turunden sayi sudur: " << sayiFloat << endl;
36      cout << "Dize girin \"2.123456789e-308\":  "; getline (cin,s);
37      sayiDouble = stod(s);
38      cout << "\t\tdouble turunden sayi sudur: " << sayiDouble << endl;
39      cout << "Dize girin \"1.123456789e+308\":  "; getline (cin,s);
40      sayiLongDouble = stold(s);
41      cout << "\t\tlong double turunden sayi sudur: " << sayiLongDouble << endl;
42
43      cout << endl.;
44      system("Color 17");
45      system("pause");
46      return 0;
47  }
```

Şekil 3.4.5 (devam)

Programı yürüttükten sonra bir çıktı şudur

```
Dize girin    "2123456789": 2123456789
              int turunden sayi sudur: 2123456789
Dize girin    "2123456789": 2123456789
              long turunden sayi sudur: 2123456789
Dize girin    "9123456789123456789": 9123456789123456789
              long long turunden sayi sudur: 9123456789123456789
Dize girin    "4123456789": 4123456789
              unsigned long turunden sayi sudur: 4123456789
Dize girin    "18123456789123456789": 18123456789123456789
              unsigned long long turunden sayi sudur: 18123456789123456789
Dize girin    "1.23456789": 1.23456789
              float turunden sayi sudur: 1.23457
Dize girin    "2.123456789e-308": 2.123456789e-308
              double turunden sayi sudur: 2.12346e-308
Dize girin    "1.123456789e+308": 1.123456789e+308
              long double turunden sayi sudur: 1.12346e+308
```

Sayıyı Dizeye Dönüştürme

Bir sayıyı dizeye dönüştürmek için to_string() fonksiyonu kullanılır:

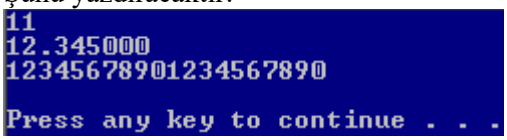
```
to_string(sayi)
```

sayı: int, uzun, uzun uzun, unsigned int, unsigned long, unsigned long long, float, double ve long double türünden olabilir.

Örneğin, aşağıdaki program bölümü:

```
int sayi1  = 11;
double sayi2 = 12.345;
unsigned long long sayi3 = 12345678901234567890;
cout << to_strin(sayi1) << endl;
cout << to_strin(sayi2) << endl;
cout << to_strin(sayi3) << endl;
```

Şunu yazdıracaktır:



Alıştırma Ödevleri

1. Bir kelime girilsin ve onun ters kelimesi yazdırılsın. (ters kelime arkadan öne yazılan kelimedir. Örneğin otelo, oleto'nun zıttıdır).
2. Aşağıdaki dizide Makedon kelimesi The Great ile değiştirilsin.
"En buyuk makedon kralı İskender Makedon, II Filipin ogluymus."
3. Ad (ad), soyad (soyad) ve okulun adı (okul) girilsin ve şu yazdırılsın:
Benim ismim ad soyad, okul okulu ogrencisiyim.
(ad, soyad ve okul değişkenleri büyük harflerle yazılmalsın).
4. Önceki ödevde soyadın cümlede hangi konumda başladığı belirlensin.
5. Belirli bir dizede, içinde bulunan iki verilmiş kelime arasındaki mesafe bulunsun. (Mesafe, kelimeler arasındaki karakter sayısıdır). Örneğin, 3. ödevde ad ve okul kelimeleri arasındaki mesafe 8'dir.
6. Girilen cümlenin (dize) nokta ile bitip bitmediği kontrol edilsin.
7. "123456789" dizisi girilsin ve aşağıdaki ifadeler hesaplansın:

```
s1 = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9;
s2 = 1 - 2 + 3 - 4 + 5 - 6 + 7 - 8 + 9;
p1 = 1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9;
```

8. 5 basamaklı tam sayı dize olarak girilsin ve rakamlarının toplamı hesaplansın.
9. Bir dizenin karakterlerini döngüsel olarak k basamak sola (veya sağa) kaydırılsın. Dizenin fonksiyon argümanı olsun, dönüş değeri ise karakterleri kaydırılmış dize olsun.

Bilgiyi Kontrol Etme Soruları

1. Dizelerle çalışma fonksiyonları hangi C++ kütüphanesinde bulunur?
2. İki dizenin değerleri hangi fonksiyon ile değiştirilir?
3. Bir dizenin uzunluğu hangi fonksiyonla belirlenir?
4. Dizeden sayıya dönüştürme fonksiyonlarından bazılarını söyleyin?
5. to_string() fonksiyonu ne için kullanılır?

Terimler

• **Dizi**, yapılandırılmış eş kökenli veri türüdür.
• **Tek boyutlu dizi**, tek indisli dizidir.
• **Sabit değişken** olarak da adlandırılan **adlandırılmış sabit,** programda değiştirilemeyen tanımlanmış sabittir.
• **Başlatma listesi,** bir dizinin elemanlarına atanan değerlerin listesidir.
• **Sembolik sabit,** #define ön işlemci yönergesi ile belirtilir.
• **Aralığa dayalı for komutu,** sıralarına bakılmadan bir dizinin tüm elemanları üzerine işlem yapılması gerektiğinde kullanılır.
• **İki boyutlu dizi,** iki indisi olan dizidir.
• **Matris**, iki boyutlu dizi için matematiksel terimdir.
• **Statik değişken,** bellekte sabit adreste yerleşen değişkendir.
• **Dinamik değişken,** oluşturulduğu sırada boş olan bellekteki adreste yerleşen değişkendir.
• **İşaretçi veya işaretçi değişkeni,** değeri bellek adresi olan değişkendir.
• **Adres operatörü &,** kendisinden sonra belirtilen işlenenin adresini döndüren operatördür.
• **Dolaylı operatör * (referanssızlaştırma operatörü),** işaretçinin işaret ettiği kendisinden sonraki değişkenin değerini döndüren operatördür.
• **new**, dinamik değişken oluşturma komutudur.
• **delete**, dinamik değişkeni silme komutudur.
• **<string>**, dizelerle çalışma fonksiyonları içeren C++ kütüphanesidir.
• **stoi, stol, stoll, stoul, stoull, stof, stod, stold,** dizeyi int, long, long long, unsigned long, unsigned long long, float, double ve long double türden sayıya dönüştürmek için kullanılan fonksiyonlardır.
• **to_string()** sayıyı dizeye dönüştürme fonksiyonudur.

Özet

• Diziler, elemanları aynı türden veriler olan yapılandırılmış veri türüdür.
• Dizideki her elemanın indis adı verilen kendi sıra numarası vardır.
• Bir dizi tek boyutlu (bir indisli), iki boyutlu (iki indislii), üç boyutlu (üç indisli) vb. olabilir.
• Dizi elemanlarına, atama deyimi ile veya bildirim sırasında başlatma listesi ile değer atanabilir.

• Tek boyutlu dizi şu şekilde bildirilir: tür adı [sayı], burada sayı tam sayı veya adlandırılmış sabit olabilir.
• Başlatma listesiyle başlatılan dizinin boyutu çevirici tarafından otomatik olarak belirlenir.
• Başlatma listesinde dizideki elemanlardan daha az değerler varsa, dizinin geri kalan elemanları dizinin türüne göre varsayılan bir değerle başlatılır. Başlatma listesinde dizideki elemanlarda daha fazla değerler varsa, sözdizimi hatası oluşur.
• Bir dizinin boyutu, #define yönergesiyle belirtilen sembolik sabit aracılığıyla da belirtilebilir.
• C++'da dizilerin otomatik olarak başlatılması yoktur.
• C++'da n uzunluğundaki tek boyutlu dizinin elemanlarının indisleri şunlardır: 0, 1... n – 1.
• Aralığa dayalı for komutu, elemanların sırasına bakılmadan bir dizinin tüm elemanlarının üzerinde işlemler için kullanılır.
• İki boyutlu dizi, iki indisi olan dizidir.
• C++'da boyutları [m][n] olan iki boyutlu dizinin elemanlarının indisleri şunlardır: birinci boyut için 0, 1... m – 1 ve ikinci boyut için 0, 1... n – 1.
• İki boyutlu dizi şu şekilde bildirilir: tür adı [sayı1] [sayı2], burada sayı1 ve sayı2 tam sayı veya adlandırılmış sabit olabilir.
• İki boyutlu dizi bildirim sırasında veya başlatma listesiyle başlatılabilir.
• Başlatma listesinde iki boyutlu dizinin tüm elemanlarını başlatmak için yeterli değer yoksa, geri kalan dizi türünün varsayılan değeriyle başlatılır.
• İki boyutlu dizi, başlatma listesiyle başlatılırsa, C++'da satır ve sütun sayısını belirlemek için sizeof() operatörü kullanılır.
• İki boyutlu dizi, her satırı tek boyutlu dizi olduğunu alarak dizi dizisi olarak ele alınabilir.
• Statik değişkenler bellekte sabit adreste yerleşir ve programın sonuna kadar aynı adreste kalır.
• Dinamik değişkenler, oluşturuldukları sırada boş olan adreslere yerleştirilir.
• Dinamik değişkenlere erişmek için işaretçi mekanizması kullanılır.
• İşaretçiler sadece bellek adresi içerebilirler.
• İşaretçiler, sadece işaretçiyle aynı türde olan değişkenlere işaret edebilir.
• Adres operatörü &, kendisinden sonra belirtilen işlenenin adresini döndürür.

• Bir işaretçi başka bir işaretçiye atanabilir ve o zaman her ikisi de aynı değişkeni işaret ediyor.
• İşaretçiler aracılığıyla bir değişkenin değerini elde etmek için, değişkeni işaret eden işaretçiden önce belirtilen dolaylı operatör * kullanılır.
• Bir işaretçi 0, NULL veya bir değişkenin adresi olarak başlatılabilir.
• NULL değerine sahip işaretçi hiçbir değişkene işaret etmez.
• 0 değeri, işaretçiye atanabilen tek tam sayı değeridir.
• Dizi adı, dizinin ilk eleman adresinin değerine sahiptir ve dizi elemanlarının türündeki değişkenler işaretçi türündendir.
• İşaretçilerle şu işlemler yapılabilir: artırma, eksiltme, tam sayı değeri ekleme veya çıkarma ve bir işaretçiyi başka işaretçiye ekleme veya bir işaretçiyi diğerinden çıkarma.
• Bir işaretçiye işaretçi bildirilebilir.
• new komutu ile bellekte dinamik değişken oluşturulur.
• delete komutu ile değişken silinir ve kapladığı bellek boşaltılır.
• Programda dizeler kullanılıyorsa, #include yönergesi ile <string> kütüphanesi dahil edilmelidir.
• C++'da dizelerle çalışmak için çok sayıda fonksiyon vardır.
• C++'da dizeden sayıya dönüştürme fonksiyonları şunlardır: stoi, stol, stoll, stoul, stoull, stof, stod, stod.
• Sayıyı dizeye dönüştürme için to_string() fonksiyonu kullanılır.

EK A

BASİT VERİ TÜRLERİ

Tür	Bit	Aralık
Tam sayılar		
short	16	-32768'den 32767'ye kadar
unsigned short	16	0'dan 65535'e kadar
int	32	-2147483648'den 2147483647
long	32	-2147483648'den 2147483647
unsigned int	32	0'dan 4294967295
long long	64	-9e18'den +8e18'e kadar
Karakterler		
char	8	'A'-'Z', 'a'-'z', '0'-'9'.....
Gerçek sayılar		
float	32	+/-10E-37'den +/-10E37'ye kadar
double	64	+/-10E-307'den +/- 10E307'ye kadar
long double	32	+/-10E-307'den +/- 10E307'ye kadar

ASCII KARAKTERLER

B

ASCII	Onaltılık	Karakter	ASCII	Onaltılık	Karakter	ASCII	Onaltılık	Karakter	ASCII	Onaltılık	Karakter
0	0	NUL	32	20	(boşluk)	64	40	@	96	60	`
1	1	SOH	33	21	!	65	41	A	97	61	a
2	2	STX	34	22	"	66	42	B	98	62	b
3	3	ETX	35	23	#	67	43	C	99	63	c
4	4	EOT	36	24	$	68	44	D	100	64	d
5	5	ENQ	37	25	%	69	45	E	101	65	e
6	6	ACK	38	26	&	70	46	F	102	66	f
7	7	BEL	39	27	'	71	47	G	103	67	g
8	8	BS	40	28	(	72	48	H	104	68	h
9	9	TAB	41	29	)	73	49	I	105	69	i
10	A	LF	42	2A	*	74	4A	J	106	6A	j
11	B	VT	43	2B	+	75	4B	K	107	6B	k
12	C	FF	44	2C	,	76	4C	L	108	6C	l
13	D	CR	45	2D	-	77	4D	M	109	6D	m
14	E	SO	46	2E	.	78	4E	N	110	6E	n
15	F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

KAYNAKÇA

Cormen, T., Leiserson. C., Rivest, R., Stein, C.: Introduction to ALGORITHMS, 3th Ed., MIT Press, 2009.
Deitel, P., Deitel, H.: C++20 for Programmers, Pearson, 3rd edition, 2020.
Deitel, P., Deitel, H.: C++ How to Program, Pearson, 10 edition, 2016.
Deitel, P., Deitel, H.: C++ How to Program (Early Objects Version), Pearson, 9 edition, 2013.
Eckel, B.: Thimking in C++: Introduction to Standard C++, Vol.1, Prentice Hall, 2000.
Yovançevski, G.: C++ Programlama, üniversite ders kitabı, Gocmar, Üsküp 2018.
Yovançevski, G., Ackovska, N., Stoyçevska, B, Yovanov, M..: C++ Algoritmalar ve Programlar Derlemesi, üniversite ders kitabı, Gocmar, Üsküp 2017, 2007.
Yovançevski, G., Ackovska, N., Stoyçevska, B, Yovanov, M..: Yeni Başlayanlar İçin C++ Programlama, Gocmar, Üsküp 2011.
Yovançevski, G., Stoyova, R.: Programlama dilleri, IV. sınıf lise eğitimi için ders kitabı, Gocmar, Üsküp 2009, 2006, 2004;
Yovançevski, G., Lazarevska, J.: Programlama dilleri, III. sınıf lise eğitimi için ders kitabı, Gocmar, Üsküp 2009, 2006;
Yovançevski, G.: Bilişim, I. sınıf lise eğitimi için ders kitabı, Gocmar, Üsküp 2009, 2002.
Yovançevski, G., Ackovska, N., Stoyçevska, B.: C++ Programlama temelleri, üniversite ders kitabı, Gocmar, Üsküp 2007.
Yovançevski, G.: Algoritmalar ve programlar, Gocmar, Üsküp 1993.
Knuth, D.: The Art of Computer Programming, Volumes 1-4A, 1st Ed., Addison-Wesley Professional, 2011.
Lippman, S., Lajoie, J., Moo, B.: C++ Primer, Addison Wesley Professional, 5 edition, 2012.
Horton, I.: Visual C++ 2013, Wiley, 2014.
Sedgewick, R.: Algorithms in C++, 3th Ed., Addison Wesley Professional, 1998.
Sedgewick, R., Wayne, K.: Algorithms, 4th Ed., Addison Wesley Professional, 2011.
Stroustrup, B: The C++ Programming Language, 4th Ed., Addison Wesley Professional, 2013.
Stroustrup, B: Programming: Principles and Practice Using C++, 2th Ed., Addison Wesley Professional, 2014.
Wilf, H.: Algorithms and Complexity, Taylor & Francis, 2002.

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

bool palindrom( int n, string kelime )
{	if( kelime.at( 0 ) != kelime.at( n - 1 ) )
		return false;
	else {
		if( n > 2 ) {
			kelime = kelime.substr( 1, n - 2 );
			return palindrom( n - 2, kelime );
		}
		else
			return true;
	}
}

int main() {
	string kelime;
	cout << "Kelime veya cumle girin: ";
	getline( cin, kelime );
	cout << kelime << "kelimesi \"" << "\" ";
	int n = kelime.length();
	if( palindrom( n, kelime ) )
		cout << "Palindromdur " << endl;
	else
		cout << "palindrom DEGILDIR? " << endl;

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