SAKARYA ÜNİVERSİTESİ

Programlamaya Giriş
HAFTA 3
C++ Programlama Dili

Prof. Dr. Cemil ÖZ

Doç. Dr. Cüneyt BAYILMIŞ

Dr. Öğretim Üyesi Gülüzar ÇİT

Konu & İçerik

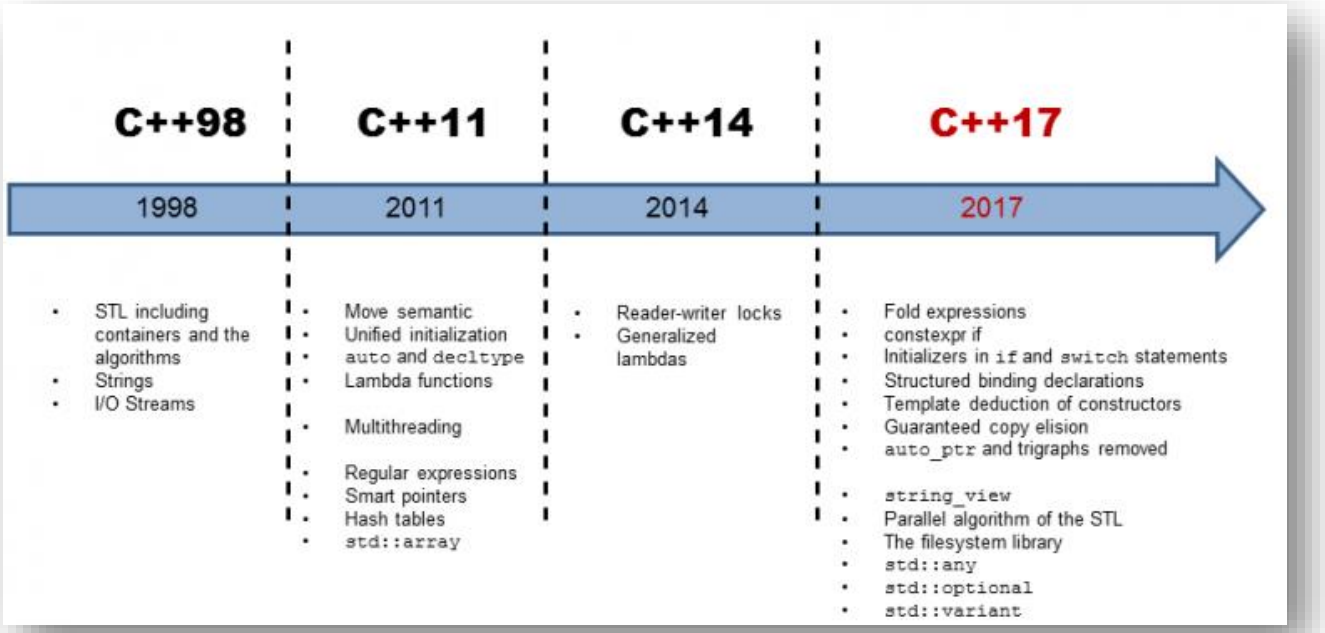
- C++ - Genel Bakış
- C++ Ortamı Bileşenleri
- Bir C++ Programının Gelişim Aşamaları
- Bir C++ Programının Derlenme Aşamaları
- Büyük Resim – C++ Genel Program Yapısı
- İlk C++ Programı
- İsim Uzayı (Namespace)
- Değişkenler (Variables)
- Değişmez/Sabit Tanımlama
- Setw (Manipulator)
- Tip Dönüşümü
- Aritmetik Atama, Artırma, Azaltma İşleçleri
- İlişkisel İşleçler
- İşlem Önceliği
- Kitaplık Fonksiyonları
- Sorular
- Kaynaklar

C++ - Genel Bakış

- C programlama dili, Denis Ritchie tarafından Bell Laboratuvarları'nda B dilinin geliştirilmesi/evrimleştirilmesi sonucu ortaya çıkmıştır. Bu tasarım için ise 1972 yılında DEC PDP-11 tipi bir bilgisayar kullanılmıştır.
- C, ilk olarak UNIX işletim sistemini yazarken kullanılan dil olarak meşhur olmuş ve günümüzde ise halen hemen hemen tüm işletim sistemleri C veya C++ ile yazılmaktadır.
- C'nin en büyük özelliklerinden biri donanımdan bağımsız olmasıdır, yani dikkatli bir tasarım ile herhangi bir donanımda C uygulaması geliştirmek mümkündür.
- C'nin farklı makinelerdeki hızlı gelişimi çok farklı varyasyonlara neden olmuştur. Bunlar arasında büyük benzerlikler olsa da çelişkiler de mevcuttur.
- Bu nedenle, C'nin standart bir versiyonunun yayınlanması zorunlu hale gelmiştir. 1983 yılında, ANSI (American National Standarts Enstitute) tarafından bir çalışma yayınlanmış ve 1988 yılında ikinci sürümü yayınlanan C versiyonu **ANSI C** olarak adlandırılmıştır.
- 1970'li yılların sonlarına doğru C gelişimini tamamlamış ve şu anda geleneksel C olarak bilinen halini almıştır.

C++ - Genel Bakış…

- C++ programlama dili ise C programlama dilinin devamı niteliğindedir.
- 1979 yılında yine Bell Laboratuvarları'nda Bjarne Stroustup tarafından geliştirilmiştir.
- Başlangıçta "C with classes" olarak adlandırılan dil, 80'li yılların başında C++ olarak isimlendirilmiştir.
- ANSI ve ISO (International Organization for Standardization) standartlarına uygun olan C++ programlama dilinin en büyük avantajı nesne yönelimli bir programlama dili olmasıdır.
-

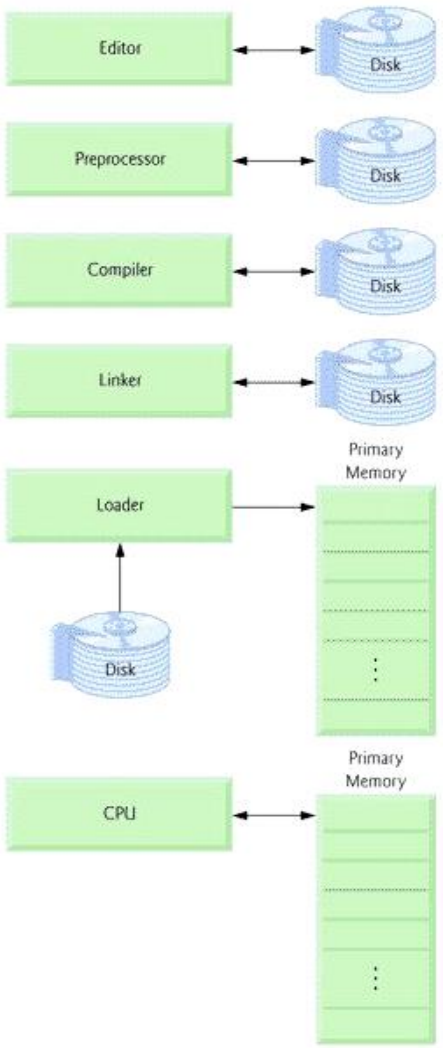


C++ Ortamı Bileşenleri

- Program Geliştirme Ortamı
- C++ Dili
- C++ Standard Kitaplığı

Bir C++ Programının Gelişim Aşamaları

1. Biçimlendirme (Edit)
2. Önişleme (Preprocess
3. Derleme (Compile)
4. Bağlama (Link)
5. Yükleme (Load)
6. Çalıştırma (Execute)



Program Editörde oluşturulur, diskte depolanır.

Önişlemci program kodu işletir.

Derleyici nesne kodu oluşturur ve diskte depolar.

Bağlayıcı, nesne kodu ile kütüphaneyi bağlar ve *.out dosyası oluşturur.

Yükleyici programı belleğe koyar.

CPU her bir komutu alır ve çalıştırır.

Bir C++ Programının Derlenme Aşamaları

- Program ilk olarak bir editörde yazılır. Bu programa kaynak (source) program ve içindeki kodlara kaynak kod (source code) denir. Kaynak program, sabit diske dosya şeklinde kaydedilmelidir. Bir çok C++ derleyicisi kaynak programın .cpp (.c) uzantılı bir dosya olmasını bekler. Örneğin: ornek1.cpp
- Derlendikten sonra programın makine dilindeki versiyonu nesne (object) program adını taşır ve .obj uzantılı olur.
- Derlendikten sonra çalışmaya hazır programın uzantısı .exe olur ve buna exe programı denir. {execute: yürütme}

Büyük Resim – C++ Program Genel Yapısı

```
// C++ genel program yapısı
#include <iostream>
#include <math.h>
using namespace std;
#define sabit  100
#define pi     3.14
char kar;
char dizi[10]; int
sayi=5;
void alt_program()
{
…
}
main()
{
int x;
float y;

Komutlar
……
}
```

kütüphane dosyalarının yazılması

İsim uzayı

Sabit tanımlanması

Global değişkenlerin tanımlanması

Alt programların yazılması

Yerel değişkenler

Ana programın yazılması

İlk C++ Programı

⇒ [1]_merhaba_dünya.cpp

```
//=========================================
// Adı                : Programlamaya Giriş
// Yazar              : Gülüzar ÇİT
// Versiyon           : 1.0
// Telif Hakkı        : @Copyright 2018
//=========================================
// Ekrana "Merhaba Dünya" yazan program
//=========================================

#include <iostream>

using namespace std;


int main()
{
    cout<<"!!!!Merhaba Dünya!!!"<<endl;
    return 0;
}
```

Kütüphane ekleme

main ve diğer tüm fonksiyon blokları **{** ile başlar ve **}** biter

return 0 programın başarı ile sonlandığını gösterir

- Anlaşılabilirliği artırmak için programların ilk satırlarında programın kısa tanıtımı, yazarı, tarihi gibi bilgiler yer almalıdır.
- Boşluklar, boş satırlar ve hiyerarşik hizalama yine anlaşılabilirliği artırma açısından son derece önemlidir.

İlk C++ Programının Çalıştırılması

- Derleyici
  - GNU C++ Compiler (Windows sistemler için MinGW), XCode, Visual C++ Compiler, vs.
- Geliştirme ortamı
  - NotPad ++, Eclipse CDT, Dev C++, Visual Studio, vs.

İsim Uzayı

- Program boyutları büyüdükçe tüm verilere farklı isimler bulmak zorlaşır.
- Nesne ya da fonksiyon arşivlerinin kullanımında, arşivlerdeki değişkenler ile kendi yazdığınız programınız içerisindeki değişkenlerin isimleri çakışabilir. Bu durumda derleme hatası ile karşılaşabilirsiniz.
- C++ bu problemi **isim uzayı (namespace)** ile çözmüştür.
- Her programcı kendi tanımlamalarını kendine ait bir isim uzayı içerisinde yapar.
- Bir isim uzayındaki verinin ismi başka bir uzaydaki ile aynı olsa dahi çakışma olmaz.

İsim Uzayı

İsim Uzayı Tanımlama

```
namespace programci1
{                                   //programci1'in isim uzayı
     int tamsayi;                   // programci1'e ait tamsayi
     void fonksiyon(int);
     :                              // Diğer değişkenler
}                                   // programci1 isim uzayının sonu

namespace programci2
{                                   // programci2'nin isim uzayı
     int tamsayi;                   // programci2'ye ait tamsayi
     :                              // Diğer değişkenler
}                                   // İsim uzayının sonu
```

İsim Uzayındaki Tanımlayıcılara Erişilmesi

```
programci1::tamsayi=3;          //programci1'in uzayındaki tamsayi
programci2::tamsayi=-345;       //progamci2'nin uzayındaki tamsayi
programci1::fonksiyon(6);       //progamci1'in uzayındaki fonksiyon
```

İsim Uzayı...

- **"using" Bildirimi:**

```
using programci1::tamsayi; // programci1 İsim uzayındaki
                           // bir değişken için geçerlidir
tamsayi = 3;               // programci1'deki tamsayi
programci2::tamsayi = -345;// Diğer tamsayi için isim
                           // uzayını belirtmek gerekli
programci1:: fonksiyon(6); // fonksiyon() fonksiyonu için
                           // isim uzayını belirtmek gerekli
```

- Bu bildirim istenirse tek bir veri yerine isim uzayının tamamı için geçerli yapılabilir. Bunun için **using namespace** sözcükleri birlikte kullanılır.

```
using namespace programci1;     // İsim uzayının tamamı için geçerli
tamsayi = 3;                    // programci1'deki tamsayi
fonksiyon(6);                   // programci1'deki fonksiyon
programci2::tamsayi = -345;     // Diğer isim uzayında
                                // isim belirtmek için gerekli
```

Değişkenler

- Üzerinde işlem yapılan verinin saklandığı bellek konumunu gösterir.
- Aşağıdaki özelliklere sahiptir.
  - **Ad (name)**
    - harf, rakam ve _
    - rakam ile başlayamaz
    - büyük/küçük harf duyarlı
  - **Tip (type)**
  - **Değer (value)**
  - **Ömür (lifetime)**
    - Değişkenin kullanılabildiği süre
  - **İsim Uzayı (namespace)**
    - Değişkene ismi ile erişilebilen program bloğu

Değişkenler…

- **Değişken isimlendirme kuralları**
  - Değişken isimleri harf, rakam ve '_' ifadelerinden oluşur.
  - İlk karakter rakam olamaz. Türkçe karakterler kullanılamaz.
  - Ayrılmış kelimeler (reserved words) kullanılamaz. İf, else, for v.s.
  - Büyük harf-küçük harf duyarlılığı vardır (case-sensetive).
  - Değişken isminin içerdiği veri ile ilgili olması büyük kolaylıklar sağlar.
- Değişkenlerin ilk harfi küçük, diğer kelimelerin baş harfi büyük olmalı.
  - ogrencininNotOrtalamasi, ogrenciSayisi gibi…
- Kodlama türleri
  - Linux Coding Style, Linus Torvalds
  - Hungarian Notation,
  - MSDN GNU Coding Standards
  - Java Coding Style Guide

Değişkenler…

Değişken Tipleri ve Değer Aralıkları

Veri Tipi	Bit	Bayt	Değer Aralığı
bit	1		0 / 1
bool	8	1	true / false
signed char	8	1	-126 to +127
unsigned char	8	1	0 to 256
Enum	16	2	-32768 ile +32767 arası
signed short	16	2	-32768 ile +32767 arası
unsigned short	16	2	0 to 65535
signed int	16	4	-2147483648 to +2147483647
unsigned int	16	4	0 – ~4 Milyar
signed long	64	8	-2147483648 to +2147483647
unsigned long	64	8	0 to 4294967295
float	32	4	+/-10E38 (hassasiyet 7 basamak)
double	64	8	+/- 10E308(hassasiyet 15 basamak)

- **int** tipi **short** tipinden büyük olmasına karşın daha hızlı erişilir (**NEDEN?**)
- int_8, _int16, _int32, _int64

Değişkenler…

Tamsayı veri Tipi / int

ÖRNEK: ⇒ [2]_tamsayi.cpp

```
#include <iostream>

using namespace std;

int main()
{
   int sayi1;
   int sayi2;
   int toplam = 0;
   //Yukarıdaki üç satırlık tamsayı tanımlama
   //aşağıdaki şekilde de yazılabilir:
   //int sayi1, sayi2, toplam = 0;

   cout <<" iki sayiyi giriniz: ";
   cin >> sayi1 >> sayi2;

   toplam=sayi1+sayi2;
   //char satirbasi='\n';
   cout << "islemin sonucu:\n"<< toplam <<endl;
   //???????
   //system("pause");
   return 0;
}
```

Tamsayılar	Bayt Miktarı
(unsigned) int	derleyici bağımlı*
(unsigned) short	2-bayt

* int veri tipi, kullanılan işletim sisteminin kelime uzunluğuna bağlıdır. Örneğin 32 bit kelime uzunluğuna sahip işletim sisteminde 4 bayt.

Değişkenler…

- **Karakter Veri Tipi - char**
  - **ÖRNEK:** ⇒ [3]_char.cpp

```
#include <iostream>

using namespace std;

int main()
{
    char ch1, ch2 ;
    char ch3 = '\t';

    ch1 = 'B';
    cout << ch3<< ch1<<'\t'<<"\\ t"<<ch2<< endl ;

    system("pause");

    return 0;
}
```

Değişkenler…

Karakter Veri Tipi – char

cout komutu içerisinde kullanılan özel karakterlerin anlamı

Karakter	Açıklama
\a	bip
\b	geri silme
\f	ileri sarma (yazıcı)
\n	Sonraki (bir alt) satır başına git*
\r	Geçerli satır başına git
\t	Yatay sekme (tab)
\v	Dikey sekme
\\	Ters bölü karakteri
\?	Soru işareti karakteri
\’	Tek tırnak karakteri
\"	Çift tırnak karakteri

*** \n yerine endl komutu da kullanılabilir**

Değişkenler…

- **Klavyeden değişken değeri alma – cin komutu**
  - ÖRNEK: ⇒ [4]_cin.cpp

```
#include <iostream>

using namespace std;

int main()
{
   int fSicaklik, cSicaklik;

   cout << "Sicaklik \"fahrenheit cinsinden\" giriniz \n";
   cin >> fSicaklik;

   cSicaklik = (fSicaklik - 32) * 5 / 9;

   cout << "Derece olarak: " << cSicaklik << " C dir"<<endl;

   system("pause");

   return 0;
}
```

Değişkenler…

- **Kayan Noktalı Veri Tipi - float**
  - ÖRNEK: ⇒ [5]_float.cpp

```
#include <iostream>

using namespace std;

int main()
{
    float yaricap;
    const float PI = 3.14159F;

    float alan;

    cout << "Dairenin yari capi: ";
    cin >> yaricap;

    alan = PI * yaricap * yaricap;
    //float alan = PI * yaricap * yaricap;

    cout << "Alan Degeri: " << alan << endl;

    system("pause");

    return 0;
}
```

Değişmez/Sabit Tanımlama

- Tüm program boyunca aynı kalan ve değiştirilmesi mümkün olmayan değişken veya tanımlamalardır.

```
#define pi 3.14
#define OCAK 1
#define uyarı "İkaz var"

const float g = 9.81;
const  int x = 5;
```

Değişmez/Sabit Tanımlama

- **Enum**
  - **enum enum-ismi { isimler listesi} değ_listesi;**
  - ÖRNEK: ⇒ [6]_enum.cpp

```
#include <iostream>

using namespace std;

enum haftanın_gunleri { Pzt, Sa, Car, Pers, Cu, Cts, Pz };


int main()
{
   haftanın_gunleri gun1, gun2;

   gun1 = Pzt;
   gun2 = Sa;

   int fark = gun2 - gun1;
   cout << "Günler arasındaki fark = " << fark << endl;

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

"setw" manipülatörü

- **setw (değer)**
  - Belirtilen değer kadar ekranda yer açar.
  - Tablo şeklinde görüntüler oluşturmak için kullanılır
  - **setw**'ye neden ihtiyaç duyulur?
  - ÖRNEK: ⇒ [7_1]_setw.cpp, [7_2]_setw.cpp

```
#include <iostream>
using namespace std;
int main()
   {
   long A=1234567, B=123, C=2345;
   cout << "A" << A << endl
        << "B" << B << endl
        << "C" << C << endl;
   system ("pause");
   return 0;
```

```
A1234567
B123
C2345
Devam etmek için bir tuşa basın . . .
```

```
#include <iostream>
#include <iomanip>     // setw kullanımı için
using namespace std;
int main()
   {
   long A=1234567, B=123, C=2345;
   cout << setw(4)<<"A"<<setw(10)<< A << endl
        << setw(4)<<"B"<<setw(10)<< B << endl
        << setw(4)<<"C"<<setw(10)<< C << endl;
   system ("pause");
   return 0;
   }
```

```
   A   1234567
   B       123
   C      2345
Devam etmek için bir tuşa basın . . .
```

Tip Dönüşümü

- **Otomatik Tip Dönüşümü**
  - ÖRNEK: ⇒ [8_1]_tipdonusumu.cpp

```
#include <iostream>

using namespace std;

int main()
{
   int      tamsayi = 7;
   float    kayannokta = 155.5F;

   double sonuc = tamsayi * kayannokta;
   cout << "SONUÇ.....: " << sonuc << endl;

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

Tip Dönüşümü (Otomatik)

Tiplerin Dönüşüm Sırası

Veri Tipi	Açıklama
long double	
double	
float	
unsigned long int	unsigned long ile eş anlamlı
long int	long ile eş anlamlı
unsigned int	unsigned ile eş anlamlı
int	
unsigned short int	unsigned short ile eş anlamlı
short int	short ile eş anlamlı
unsigned char	
char	
Bool	

Tip Dönüşümü

- **Açık Tip Dönüşümü**
  - ÖRNEK: ⇒ [8_2] _tipdonusumu.cpp

```
#include <iostream>
#include <conio.h>

using namespace std;

int main()
{
    char ch1 = _getch();

    cout << ch1 << " karakterinin ASCI karsiligi : " << static_cast<int>(ch1) << endl;
    // (int)ch1 -  C standardı tip dönüştürme ifadesi
    //cout << ch1 << " karakterinin ASCI karsiligi:" << (int)ch1 << endl;

    system("pause");

    return 0;
}
```

- **cast** operatörü temel sayısal veri tipleri (**int** ve **double** gibi) arasında dönüşüm yapmak için kullanılabilir.
- Yanlış veri tipine dönüştürme derleme veya çalışma zamanı hatalarına neden olabilir.

Aritmetik İşleçler

Operatörler

C++ İşlemi	Aritmetik İşleç	Matematiksel Gösterim	C++ Karşılığı
Toplama	+	$a + b$	a+b
Çıkarma	-	$a - b$	a-b
Çarpma	*	ab	a*b
Bölme	/	${}^{a}/_{b}$ veya $\frac{a}{b}$	a/b
Mod Alma	%	$a \bmod b$	a%b

Aritmetik İşleçler…

Özel İşleçler

İşleç	C++ Kullanımı	Anlamı
+=	a += 3;	a = a + 3;
-=	a -= 3;	a = a - 3;
*=	a *= 3;	a = a * 3;
/=	a /= 3;	a = a / 3;
++	a++; veya ++a;	a = a + 1;
--	a--; veya --a;	a = a - 1;

Aritmetik İşleçler…

➢ **ÖRNEK:** ⇒ [9]_islecler.cpp

```
int sayi = 27;

sayi += 10;                          // a = a + 10;
sayi -= 7;                           // a = a - 7;
sayi *= 2;                           // a = a * 2;
sayi /= 3;                           // a = a / 3;
sayi %= 3;                           // a = a % 3;
cout << sayi << endl;

sayi = 10;

cout << "Sayı =" << sayi << endl;        // 10
cout << "Sayı =" << ++sayi << endl;      // 11 (önceden arttırma)
cout << "Sayı =" << sayi << endl;        // 11
cout << "Sayı =" << sayi++ << endl;      // 11 (sonradan arttırma)
cout << "Sayı =" << sayi << endl;        // 12
```

➢ ***NOT:*** Kısaltılmış atama operatörleri kullanılarak programcılar daha hızlı kod yazabilirler ve derleyiciler kodu biraz daha hızlı derleyebilirler. Bazı derleyiciler kısaltılmış atama operatörleri için daha hızlı çalışan kod üretirler

İlişkisel İşleçler

İşleç	C++ Kullanımı	Anlamı
==	a == b;	a, b'ye eşit mi?
!=	a != b;	a, b'den farklı mı?
>	a > b;	a, b'den büyük mü?
<	a < b;	a, b'den küçük mü?
>=	a >= b;	a, b'den büyük eşit mi?
<=	a <= b;	a, b'den küçük eşit mi?

Mantıksal ve Bitsel İşleçler

İşleç	C++ Kullanımı	Anlamı	Açıklama
&&	a && b	a **VE** b	a **DOĞRU(1)** **VE** b **DOĞRU(1)**
||	a || b	a **VEYA** b	a **VEYA** b **DOĞRU(1)**
!	!a	a **DEĞİL**	A'nın mantıksal değili
&	a&b	a **AND** b	bitsel **VE** operatörü
|	a|b	a **OR** b	bitsel **VEYA** operatörü
~	~a	**NOT** a	a'nın tüm bitlerinin tersi
^	a^b	A **XOR** b	a ve b için bitsel **XOR** operatörü

İşleç Öncelikleri

İşleç	Yön	İşleç Grubu
()	soldan sağa	Parantez
++, --	sağdan sola	Tekli
*, /, %	soldan sağa	Çoklu çarpma, bölme, mod alma
+, -	soldan sağa	Çoklu toplama, çıkarma
<, <=, >, >=	soldan sağa	İlişkisel
==, !=	soldan sağa	Eşitlik
&&	soldan sağa	İlişkisel
||	soldan sağa	İlişkisel
=, +=, -0, *=, /=, %=	soldan sağa	Atama

İşlem Önceliği

- **ÖRNEK:**
  - Matematiksel ifade:

$$z = pr\%q + \frac{w}{x} - y$$

  - C++ Karşılığı

```
z = p  *  r  %  q  +  w  /  x  –  y;
```

6 1 2 5 3 2

- *NOT:* Kompleks aritmetik ifadelerde fazladan parantez kullanmak kodun okunabilirliğini arttır.

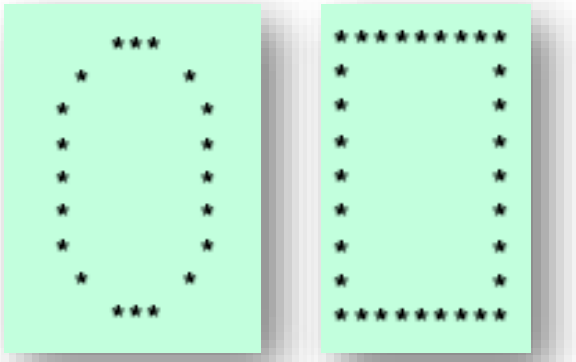
Kitaplık Fonksiyonları

➢ **ÖRNEK:** ⇒ [10]_karekok.cpp

```
#include <iostream>
#include <locale.h>                 // Diller ve karakter setleri kütüphanesi
#include <cmath>                    // sqrt() - karekök alma fonksiyonu için

using namespace std;

int main()
{
    // Sonuç ekranında Türkçe karakterleri kullanabilmek için
    setlocale(LC_ALL, "Turkish");

    double sayi, sonuc;             //sqrt() fonksiyonu double veri tipi alır.

    cout << "Bir sayi girin: ";
    cin >> sayi;
    sonuc = sqrt(sayi);         //sayının karekökünü hesapla

    cout << sayi << " sayısının karekökü : " << sonuc << endl;

    system("pause");

    return 0;
}
```

Çalışma Soruları

- Kullanıcıdan istediği iki tamsayıyı okuyan ve bu değerlerin toplamını, farkını, ortalamasını, çarpımını, bölümünü ve bölümünden kalanını bulup ekrana yazdıran programı yazınız.
- Aşağıdaki çıktıları alt alta üreten programı yazınız.



- Kullanıcıdan istediği iki tamsayıyı okuyan ve bu tamsayıları karşılaştırıp sonucu ekrana yazdıran programı yazınız(ÖR: "10, 20'den küçüktür". NOT: 10 ve 20, kullanıcı tarafından girilecek olan örnek sayıları temsil etmektedir.).
- Kullanıcıdan istediği yarıçap bilgisini kullanarak bir dairenin çapını, çevresini ve alanını hesaplayan programı yazınız (NOT: sabit olarak tanımlayacağınız π için 3.14159 değerini kullanınız).

KAYNAKLAR

- Deitel, C++ How To Program, Prentice Hall
- Horstmann, C., Budd,T., Big C++, Jhon Wiley&Sons, Inc.
- Robert Lafore, Object Oriented Programming in C++, Macmillan Computer Publishing
- http://www.modernescpp.com/index.php/cpp17-core