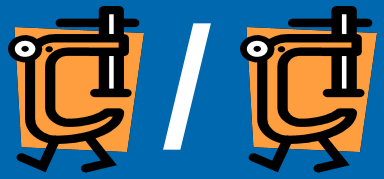
C / C++ Eğitimi

Akademi: DEPAR AKADEMİ
https://www.deparakademi.com.tr/

Eğitmen: **Bülent Çobanoğlu** (Bülend Hoca)



/ ++ Eğitimi-2

–: C/ C++Konsoldan Veri Çıkışı (scanf, cin,...)

--: Operatörler

–: Aritmetiksel Operatörler

–: ++a ile a++ farkı

–: Aritmetik Atama Operatörleri

–: Karşılaştırma Operatörleri

–: Mantıksal Operatörler

--: Bitsel Operatörler ile Aritmetiksel İşlemler

--: Mantıksal Operatörler ile Bitsel (Bitwise) operatörlerinin Farkı

–: Tümleyen Aritmetiği

–: Sabit Tanımlamaları (const, #define,...)

–: Değişken Tanımlamaları

–: C/C++ Keywords

--: Karar Yapıları (if, if else, switch-case, ….)

–: ?: operatörü

--: Döngüler (while-do while,..)

--: Döngüler (for ...)

–: break, continue deyimleri

--: İç içe Döngüler (Nested Loops)

--: Seriler

Konsoldan Veri Girişi (scanf, cin)

- Veri girişi için C dilinde `scanf`, C++ dilinde `scanf ve cin>>` komutları kullanılır

<table>
<tr><th>C dili</th><th>C++ Dili</th></tr>
<tr><td rowspan="2">int a;

scanf ("%d",&a);</td><td>int a;

scanf("%d", &a);</td></tr>
<tr><td>int a;

cin>>a;</td></tr>
<tr><td colspan="2">•getchar(): Klavyeden girilen tek bir karakteri okur ve Enter tuşuna basılmasını bekler.

• getche(): Klavyeden tek bir karakteri okur ve bir değişkene aktarır. Basılan karakter ekranda gözükür.

• getch(): Klavyeden tek bir karakteri okur ve bir değişkene aktarır. Basılan karakter ekranda gözükmez.

• gets(): Klavyeden girilen string (sözel) ifadeyi bir değişkene aktarmada kullanılır.

• getline(): Klavyeden girilen string (sözel) ifadenin belli bir karakterini (veya tek bir satırını) bir değişkene aktarmada kullanılan C++ komutudur.</td></tr>
</table>

Konsoldan Veri Girişi (scanf, cin)

- `int tamsayi=5;`
- **scanf()**

```
scanf("%d", &tamsayi);   //tipe ait format hatırlanmalı
scanf("%c",&karakter);
scanf("%s", karakter_dizisi);
```

- **cin>>**

```
cin>>tamsayi;   //tipe ait formatın hatırlanması gerekmez
cin>>reel;
cin>>karakter_dizisi;
```

Konsoldan Veri Girişi (scanf, cin)

scanf kullanırken şunlara dikkat etmek gerekir:

- İki tırnak içerisinde **yalnızca format karakteri** bulunmalıdır. \n gibi çıkış(escape) karakterleri bulunmamalıdır.
- Değişkenlerin önündeki **& adres** operatörü unutulmamalıdır.
- scanf fonksiyonuyla birden fazla değer girilirken girişler arasında istenildiği kadar boşluk karakterleri bırakılabilir.

scanf komutunun kullanım örnekleri;

Komut Satırı	Açıklaması
`scanf (%u, &b);`	Veri girişi yaparken işaretsiz tam sayı girmelisiniz.
`scanf ("%c %c %f", &c1, &c2, &a);`	Veri girişi yaparken girilen değerler arasında (a h 5.4 gibi) boşluk kullanmalısınız.
`scanf ("%d-%c- %f", &s1, &c1, &s2);`	Veri girişi yaparken girilen değerler arasında (4-a-5.4 gibi) tire '-' işareti kullanmalısınız.

Task:

a, b, c isimli double tipinde 3 değişken tanımlayınız. Sonra a ve b için klavyeden scanf fonksiyonu ile değerler alınız. Bu iki değerin çarpımını c'ye atayınız ve c'yi yazdırınız.

Operatörler (İşleçler)

Bir işleme yol açan, işlem sonucunda bir değer üretilmesini sağlayan atomlara (tokens) operatör denir.

Operatör Tipi	Sembolik Gösterimi
Aritmetiksel operatörler	+, -, *, /, %
Artırma ve azaltma operatörleri	++, --
Aritmetiksel atama operatörleri	=, +=, -=, *=, /=, %=, >>=, <<=, >>>=, &=, ^=, |=
Mantıksal operatörler	&&, ||, !, ^
Karşılaştırma operatörleri	>, <, = =, >=, <=, !=
Bit işlem operatörleri	&, |, ^, >>, <<, >>>, ~
İkili karşılaştırma operatörü	? :
Ayırma operatörü	, (Virgül)

Aritmetiksel Operatörler

İşleç	C++ Kullanımı
+	6+3;
-	6-3;
*	6*3;
/	6/3;
%	6%3

6+3= 9

6-3= 3

6*3= 18

6/3= 2

6%3= 0

"A % B = Kalan" işlemi; "A = (B * Çarpan) + Kalan" şeklinde ifade edilebilir.

Örneğin; 9 % 4 işleminin sonucu 1'dir. Eş değeri ise `9 = (4 * 2) + 1`'dir. Yani Çarpan = 2, Kalan = 1'dir. Bu işlemde Kalan, "`mod`" olarak adlandırılır.

Aritmetik Atama Operatörleri

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;

++ ve -- Operatörleri

++ operatörüne artırma (increment), -- operatörüne eksiltme (decrement) operatörü denilmektedir. Önek kullanımda sonraki işleme nesnenin artırılmış ya da eksiltilmiş değeri sokulurken, sonek kullanımda artırılmamış ya da eksiltilmemiş değeri sokulur. Örnek kullanımlar;

	++, - - operatörler	Eşdeğeri
sonek	b= a++; //Önce değeri al sonra artır	b =a; a=a+1;
önek	b= ++a; //Önce değeri al sonra artı	a=a+1; b=a;
sonek	b= a--; //Önce değeri al sonra artır	b =a; a=a-1;
önek	b= --a; //Önce değeri al sonra artı	a=a-1; b=a;

Not. Eğer bu operatörler tek başlarına başka bir operatör olmaksızın kullanılıyorsa veya başka bir değişkene atanmıyorsa, önek ve sonek biçimleri arasında bir fark yoktur. Örnek;

```
for(k=15; k>=0; --k) veya for(k=15; k>=0; k--) ifadelerinin arasında bir fark yoktur.
```

Karşılaştırma Operatörleri

İşleç	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 Operatörlerler

İşleç	Kullanımı	Anlamı	Açıklama
&&	a&&b	a ve b	Sonuç DOĞRU(1) veya YANLIŞ (0) tır
||	a||b	a veya b	Sonuç DOĞRU(1) veya YANLIŞ (0) tır
!	!a	a DEĞİL	a'nın mantıksal değili alınır

Bitsel (Bitwise) Operatörler

İşleç	Kullanımı	Anlamı	Açıklama
&	a&b	a AND b	Her iki değer bitsel VE yapılır
|	a|b	a OR b	Her iki değer bitsel VEYA yapılır
~	~a	NOT a	a'nın tüm bitleri terslenir
^	a^b	a XOR b	Her iki değer bitsel XOR yapılır

Tümleyen Aritmetiği

- Bilgisayarlar işaretli(signed) tam sayıları ifade etmek için **ikiye tümleyen (2's complement)** sistemini kullanır. Bu sistemde:
  - Sayının en solundaki bit işaret bitidir. Bu bit 1 ise s**ayı negatif**, 0 ise **pozitiftir.**
  - Negatif ve pozitif sayılar birbirlerinin ikiye tümleyeni (yani tamlayanı) dirler.
- Bir sayının ikiye tümleyeni sayının 1'e tümleyenine(1's complement) 1 eklenerek elde edilir (Sayının 1'e tümleyeni 1'lerin 0, 0'ların 1 yapılmasıyla elde edilir.) Buna göre;
- Sayı: 0001 0011 (19 sayısı)
- 1'e tümleyeni: 1110 1100
  +1
- 2'ye tümleyeni= 1110 110**1 (-19 sayısı)**

- Bu işlemi yani 19 sayısının negatif değerini hesaplayan kod satırını bitsel operatörler ile aşağıdaki gibi kodlayabiliriz;

```
int sayi = 19;
int neg_sayi = ~sayi + 1;
```

Mantıksal Veri Tipi: bool

C++ dili için standart bir veri tipi olan bool veri tipi, C dilinde standart bir veri tipi değildir. Bu nedenle C dilinde bool veri tipini kullanmak için #include <stdbool.h> kütüphanesini eklemek gerekir.

-
- Java, Python gibi dillerde bool veri tipi “true/false” sonucunu üretirken C/C++ dillerinde “1/0” sonucunu üretir.

C Dili Kodlaması	C++ Dili Kodlaması

`#include <stdio.h>`
`#include <stdlib.h>`
`#include <stdbool.h> //C'de bool tipi için gerekli`
`int main()`
`{`
`  bool b1=true;`
`  bool b2=false;`
`  printf("b1..:%d\n", b1);`
`  printf("b2..:%d\n", b2);`
`  return 0;`
`}`
	`#include <iostream>`
	`#include <stdlib.h>`
	`using namespace std;`
	`int main()`
	`{`
	`  bool b1=true;`
	`  bool b2=false;`
	`  cout<< "b1..:" <<b1 <<endl;`
	`  cout<< "b2..:" <<b2 <<endl;`
	`return 0;`
	`}`

Ekran Çıktısı	**Ekran Çıktısı**
b1..:1 b2..:0	b1..:1 b2..:0

İşleç Öncelikleri

En Yüksek	( )	Parantez içi
1	pow( ), sqrt( ), sin(), …	Fonksiyonlar
2	++degisken, --degisken	Ön artırma ve azaltma
3	-degisken	Negatifleştirme
4	*, /, %	Soldan sağa doğru
5	+, -	Soldan sağa doğru
6	<=, >=, ==, !=,….	Karşılaştırma operatörleri
7	|, &, ^, …	Bitsel operatörler
8	||, &&, ….	Mantıksal operatörler
9	?:	Sağdan sola doğru
En düşük	=, +=, *=,….	Atama operatörleri

Sabit Tanımlamaları

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

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

- #define PI 3.14
- #define OCAK 1
- #define uyarı “İkaz var”

Sabit Tanımlamaları

Sayı nokta içermiyorsa ve sayının sonunda hiçbir ek yoksa sabit int, long ve unsigned long türlerinin hangisinin sınırları içerisinde ilk kez kalıyorsa o türdendir.

```
0 ---> int
100 ---> int
```

```
123 ---> int
3000000000 ---> long
```

Sayı nokta içermiyorsa ve sayının sonunda küçük harf ya da büyük harf **L, U, UL, LU** varsa sabit long, unsigned int, unsigned long türdendir.

```
100L ---> long
0L ---> long
```

```
123U ---> unsigned int
1000u ---> unsigned int
```

```
100ul ---> unsigned long
1LU ---> unsigned long
```

Sabit Tanımlamaları

Sayı nokta içeriyorsa ve sayının sonunda hiçbir ek yoksa sabit **double** türdendir.

```
123.65 ---> double
0.5 ---> double
```

```
.10 ---> double
10 ---> int
10. ---> double
```

Sayı nokta içeriyorsa ve sayının sonunda küçük harf ya da büyük harf
F varsa sabit **float,**
L varsa sabit **long double türündedir.**

```
12.34F ---> float
10F ---> geçersiz
10.f ---> float
```

```
100.23L ---> long double
10.l ---> long double
.10L ---> long double
10L ---> long
```

Değişken Tanımlamaları

- İçerisinde veri sakladığımız, ismini ve tipini bizim belirlediğimiz bellek alanlarına değişken (variable) adı verilmektedir.

```
int x = 5;
```

Değişken adı	C	C++	Açıklama
Ali7	Geçerli	Geçerli	Bir değişkenin ilk karakteri mutlaka harf olmalıdır.
7Ali	Geçersiz	Geçersiz	
Ali_Veli	Geçerli	Geçerli	Değişken isminde '_' alt tire özel karakteri kullanılabilir.
Ali Veli	Geçersiz	Geçersiz	Değişken isminde boşluk karakteri kullanılmaz.
do	Geçersiz	Geçersiz	Değişken isimleri kullanılan programlama diline ait komutları içeremez.
_do	Geçerli	Geçerli	Değişken isminde '_' alt tire özel karakteri kullanılabilir ve 'do' ile '_do' farklıdır.
not	Geçerli	Geçersiz	C++ diline ait bir komut C dilinde geçerli değişken adı olarak kullanılabilir.
Sa%at	Geçersiz	Geçersiz	Değişken ismi içerisinde '.,@ ? * : ; !, (/ - + = % & " #' gibi özel karakterler kullanılmaz.

C Anahtar Kelimeleri

Anahtar Sözcükler (Keywords): Dil için özel anlamı olan, değişken olarak kullanılması yasaklanmış sözcüklerdir. Örneğin **if, do, int, return** gibi.

auto	float	signed
break	for	sizeof
case	goto	static
char	if	struct
const	inline (C99)	switch
continue	int	typedef
default	long	union
do	register	unsigned
double	restrict (C99)	void
else	return	volatile
enum	short	while
extern		

C++ Anahtar Kelimeleri

Anahtar Sözcükler (Keywords): Dil için özel anlamı olan, değişken olarak kullanılması yasaklanmış sözcüklerdir. Örneğin **if, do, int, return** gibi.

alignas (C++11)
alignof (C++11)
and
and_eq
asm
atomic_cancel (TM TS)
atomic_commit (TM TS)
atomic_noexcept (TM TS)
auto (1)
bitand
bitor
bool
break
case
catch
char
char8_t (C++20)
char16_t (C++11)
char32_t (C++11)
class (1)
compl
concept (C++20)
const
consteval (C++20)
constexpr (C++11)
constinit (C++20)
const_cast
continue
co_await (C++20)
co_return (C++20)
co_yield (C++20)
decltype (C++11)
default (1)
delete (1)
do
double
dynamic_cast
else
enum
explicit
export (1) (3)
extern (1)
false
float
for
friend
goto
if
inline (1)
int
long
mutable (1)
namespace
new
noexcept (C++11)
not
not_eq
nullptr (C++11)
operator
or
or_eq
private
protected
public
reflexpr (reflection TS)
register (2)
reinterpret_cast
requires (C++20)
return
short
signed
sizeof (1)
static
static_assert (C++11)
static_cast
struct (1)
switch
synchronized (TM TS)
template
this (4)
thread_local (C++11)
throw
true
try
typedef
typeid
typename
union
unsigned
using (1)
virtual
void
volatile
wchar_t
while
xor
xor_eq

Büler

Task:

İki değişkenin ali = ‘A’; veli = ‘V’; içeriğini yer değiştiren uygulamayı kodlayınız.

Task:

İki tam sayı değişkenin içeriğini 3. değişken kullanmadan (yerinde) değiştiren bir program yazınız.

Karar Yapıları(if, if-else, switch-case)

Yapı-1:

```
if (koşul)
    işlemler
```

Yapı-2:

```
if (koşul)
    işlemler-1;
else
    işlemler-2;
```

Yapı-3:

```
switch(ifade) {
    case d1: { işlemler-1;} break;
    case d2: { işlemler-2;} break;
    …..
}
```

```
if (a < 0){
   printf("Negatif");
}
```

```
if (bNotu >= 50) {
    printf ("Geçti");}
else{
    printf ("Kaldı");}
```

```
switch( sayi % 2) {
case 0:
    printf ("Çift\n"); break;
case 1:
    printf ("Tek\n"); break;
}
```

if-else yapıları-1

Tek komutlu if yapıları:

```
if(x>3)
  x++;
```

Çok komutlu if yapıları:

```
if(x>3)
{
  x++;
  y--;
}
```

if-else özel yapıları

- if parantezinin içerisindeki ifadede değişken veya bir karşılaştırma operatörü olması gibi bir zorunluluk yoktur.
- Kullanılmaları tavsiye edilmeyen ancak geçerli olan if yapıları

if(0) :Her zaman YANLIŞ (false)

if(10) :Her zaman DOĞRU (true)

if(-10) :Her zaman DOĞRU (true)

if (a) : a sıfır ise false, değilse true

if-elseif-else yapısı

```
/* 100 lük sistemde verilen not değerini harfli sisteme dönüştüren
   programı kodlayınız…*/
    if(nott<50)
       cout<<"FF aldınız";
    else if(nott>=50 && nott<=60)
       cout<<"DD aldınız";
    else if(nott>60 && nott<=75)
       cout<<"CC aldınız";
    else if(nott>75 && nott<=90)
       cout<<"BB aldınız";
    else
       cout<<"AA aldınız";
```

TASK:

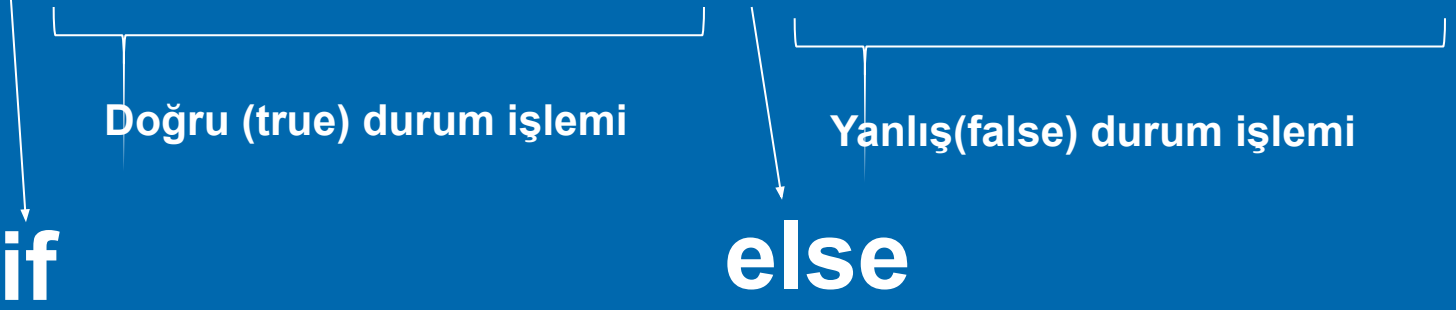
- ***100 lük sistemde verilen not değerini harfli sisteme dönüştüren programı*** **100 den büyük ve sıfırdan küçük notlarda hata mesajı verecek şekilde yeniden kodlayınız…**

Aşağıdaki program parçasının ekrana "Selin" yazması için test değişkeni hangi aralıkta olmalıdır? (2019 TÜBİTAK Bilgisayar Olimpiyatları Sorusu)

```
if (test <= 200)
   if (test < 100)
       if (test <= 300)
           printf("Ali\n");
       else
           printf("Burak\n");
   else
       printf("Selin\n");
else
   printf("Demet\n");
```

Kısa if-else operatörü ? :

a%2==0 ? printf(“çift") : printf(“tek");



printf () fonksiyonu ile kullanımı:

```
printf(“%s”,a%2==0 ? ”Çift” : ”Tek”);
```

Çoklu Dallanma: switch-case

```
switch(dallanma değişkeni){
    case n0:işlemler;break;
    case n1:işlemler;break;
    ……..
    case nn:işlemler;break;
    default:işlemler;
}
```

- Dallanma değişkeni: tamsayı, karakter olabilir, belirli bir aralık olamaz. O an değeri belli olmalıdır.

switch-case örneği

Tamsayı değişkenli:

```
int ay=1;
switch(ay){
   case 1: cout<<"OCAK";break;
   case 2: cout<<"SUBAT";break;
   case 3: cout<<"MART";break;
   case 4: cout<<"NİSAN";break;
   case 5: cout<<"MAYIS";break;
   //…
   default: cout<<"Error";
}
```

Karakter değişkenli:

```
int main()
{
 char *s = "bilgisayar";
 switch(*s + 1)
 {
        case 'b':
         printf("b");
         break;
        case 'c':
         printf("c");
         break;
        case 'i':
         printf("i");
         break;
        default:
         printf("diger");
         break;
 }
 return 0;
}
```

TASK:

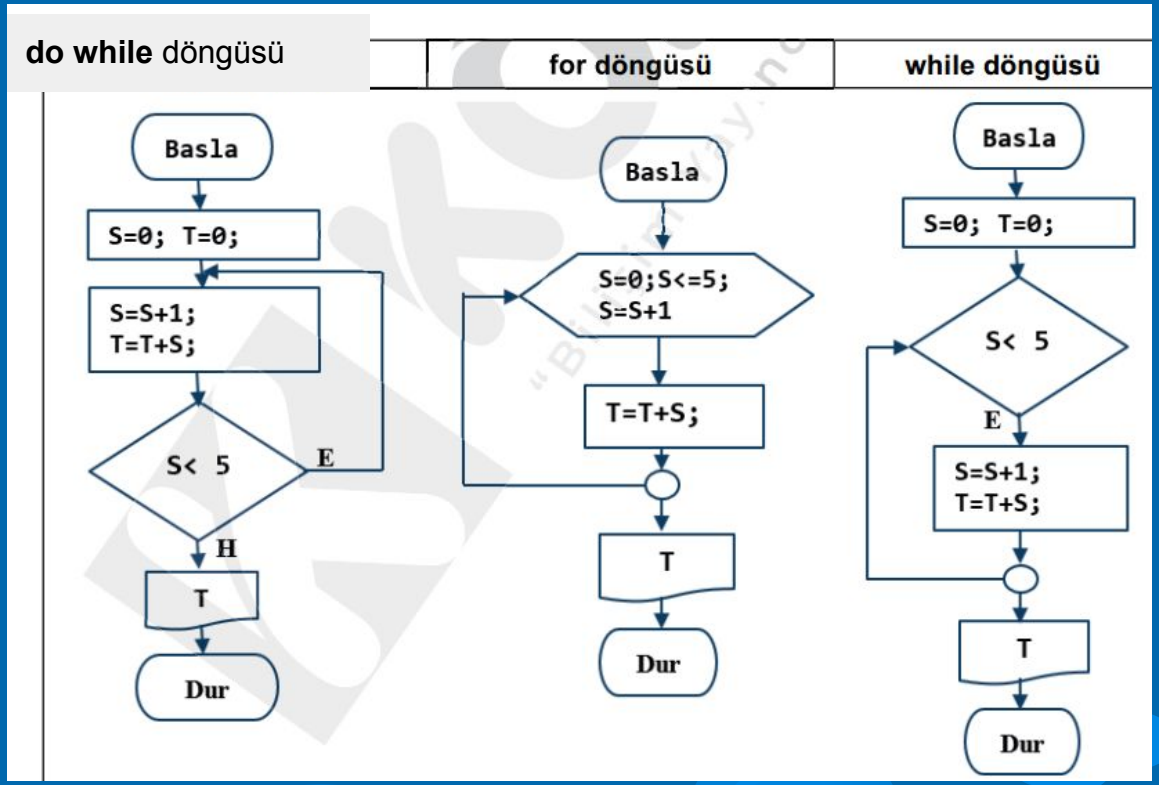
- **Girilen dereceyi fahrenayta veya fahrenaytı dereceye çeviren programı tasarlayınız. Çevirimin hangi birimden hangi birime olacağı program başında sorulmalıdır.**

Döngüler

- 1.Sayıcı kontrollü döngü
  - döngü değişkeni
  - döngü değişkeni başlangıç değeri
  - döngü değişkeni artması veya azalması
  - döngü değişkenin son değerinin test edilmesi
- 2.Belirsiz kontrollü döngü
  - belirsiz döngü adedi
  - döngü şartını sağlayan değişken değişimi

Döngülerin Çalışma Mantığı

1 den 5 kadar sayıların toplamını hesaplayan programın farklı döngü yapıları ile gerçekleştirimi:



Döngüler- while

```
while (döngü şartı)
{
   döngü gövdesi işlemleri;
}
```

- Döngü şartı sıfırdan (YANLIŞ) farklı olduğu sürece döngü işletilir.

Örnek: 1'den 10'a kadar olan tam sayıları toplayan program.

```
#include <iostream.h>
main(){
    int sayac=1,toplam=0;      //sayac, döngü kontrol değişkeni
    while(sayac<=10)           //döngü adedi belirli
    {
       toplam+=sayac;
       cout<<toplam<<"\t";
       sayac++;                //döngü kontrol değişkeni artımı
    }
    return 0;
}
```

while döngüsü özel durumlar

```
Sonsuz Döngüler:
   while(1) { }
   while(10) { }
   while(-10) { }        //Kullanım yeri?
Girilmez Döngü:
   while(0) { }          //Kullanma yeri?
Değişkene bağlı Döngü:
   while(a) { }          //Fazla serbest?
Tehlikeli Form:
   while(a=3)            //Tehlike ne?
```

Döngüler- for

```
for (başlangıç atamaları; döngü şart ifadesi; döngü sonu işlemleri)
{
    döngü gövdesi işlemleri;
}
```

- Başlangıç atamaları döngü başında sadece bir kez çalışır.
- Döngü şart ifadesi, başlangıç atamalarından sonra kontrol edilir
- Döngü sonu işlemleri her çevrimden sonra işletilir
- Döngü şart ifadesi her döngü sonu işleminden sonra kontrol edilir.

Döngü şartı sıfırdan (YANLIŞ) farklı olduğu sürece döngü işletilir

for döngüsü özel durumlar

```
Sonsuz Döngüler:
   for(;;) { }
   for(;1;){ }            // Sıfırdan farklı bir sayı
   for(;-9;){ }           // 9'un ve 1'in bir özelliği yok

Girilmez Döngü:
   for(;0;) { }

Değişkene bağlı Döngü:
   for(;a;) { }

Tehlikeli Form:
   for(;a=3;)
```

Döngüler: for

Örnek: 1'den 10'a kadar olan tamsayıları toplayan program.

```
#include <iostream.h>
main(){
    for(int sayac=1,toplam=0;sayac<=10;sayac++)
    {
      toplam+=sayac;
      cout<<toplam<<"\t";
    }
    return 0;
}
```

Döngüler: for

Örnek: Verilen sayıyı ikilik (binary) sayı sistemine çeviren program.

```
//dec_to_bin.cpp: Sayı ötelendikten sonra LSB biti maskelenmektedir.
#include<stdio.h>
int main()
{
    int sayi = 255;
    int k, bit;
    for(k=15; k>=0; --k)          // 16 bit şeklinde göster
    {
        bit = (sayi>>k) & 1;      // sayi k basamak sağa ötelenir ve 1 ile and işlemine tabi tutulur...
        printf("%d", bit);
        if(k%4==0)
            putchar(' ');         // Her 4 basamakta bir boşluk bırakılır....
    }
}
```

Döngüler: do-while

Kontrolün sonda yapıldığı while döngüsü;

```
do
{
    döngü gövdesi işlemleri;

} while (döngü şartı);
```

Örnek: 1'den 10'a kadar olan tam sayıları toplayan program.

```
#include <iostream.h>
main(){
    int sayac=0,toplam=0;      //sayac, döngü kontrol değişkeni
    do
    {
       toplam+=sayac;
       cout<<toplam<<"\t";
       sayac++;                //döngü kontrol değişkeni artımı
    } while(sayac<=10);        // döngü şartı kontrolü
    return 0;
}
```

break- continue

Belirli bir şartta veya sonsuz döngülerin terk edilmesinde **break**,

Belirli bir şartta döngü başına dönülmek istenmesinde **continue** kullanılır.

- **break** program akışını döngü sonrası komuta,
- **continue** ise döngü başına getirir.
- **break** döngünün sonlanmasını sağlar

break-continue örneği

```
#include <iostream.h>
main(){
   for(int x=1;x<=10;x++){
    if(x<5)
        continue;          //döngü başına döner
    if(x==9)
        break;             //döngüyü sonlandırır
    cout<<x;
   }
   return 0;
}
```

Çıktısı:
5 6 7 8

İç-içe Döngüler (Nested Loops)

- Bir döngünün içerisinde başka bir döngü olabilir. Tüm döngü yapıları iç-içe yapılandırılabilir.

Kullanım Alanları

- Çok boyutlu dizilerde,
- Satır ve sütunlardan oluşan, istenen çıktı desenini (paterni) oluşturmada,
- Seri hesaplamalarında,
- İlişkili döngülerde,

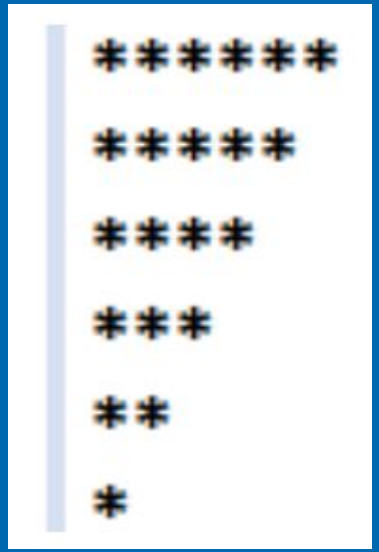
İç-içe Döngüler

- Çarpım tablosunu oluşturup ekranda gösteren programı kodlayınız.

```cpp
#include <iostream>
using namespace std;
int main() {
int i, j;
//Dıştaki i döngüsü
for (i = 1; i <= 10; i++) {
    //İçteki j döngüsü
    for (j = 1; j <= 10; j++) {
        cout<<j<<"*"<<i<<"="<<i*j<<"\t";
    } //for j sonu
    cout<<endl; //bir satır atla
} //for i sonu
return 0;
} //main sonu
```

```
1*1=1   2*1=2   3*1=3   4*1=4   5*1=5   6*1=6   7*1=7   8*1=8   9*1=9   10*1=10
1*2=2   2*2=4   3*2=6   4*2=8   5*2=10  6*2=12  7*2=14  8*2=16  9*2=18  10*2=20
1*3=3   2*3=6   3*3=9   4*3=12  5*3=15  6*3=18  7*3=21  8*3=24  9*3=27  10*3=30
1*4=4   2*4=8   3*4=12  4*4=16  5*4=20  6*4=24  7*4=28  8*4=32  9*4=36  10*4=40
1*5=5   2*5=10  3*5=15  4*5=20  5*5=25  6*5=30  7*5=35  8*5=40  9*5=45  10*5=50
1*6=6   2*6=12  3*6=18  4*6=24  5*6=30  6*6=36  7*6=42  8*6=48  9*6=54  10*6=60
1*7=7   2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49  8*7=56  9*7=63  10*7=70
1*8=8   2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64  9*8=72  10*8=80
1*9=9   2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81  10*9=90
1*10=10 2*10=20 3*10=30 4*10=40 5*10=50 6*10=60 7*10=70 8*10=80 9*10=90 10*10=100
```

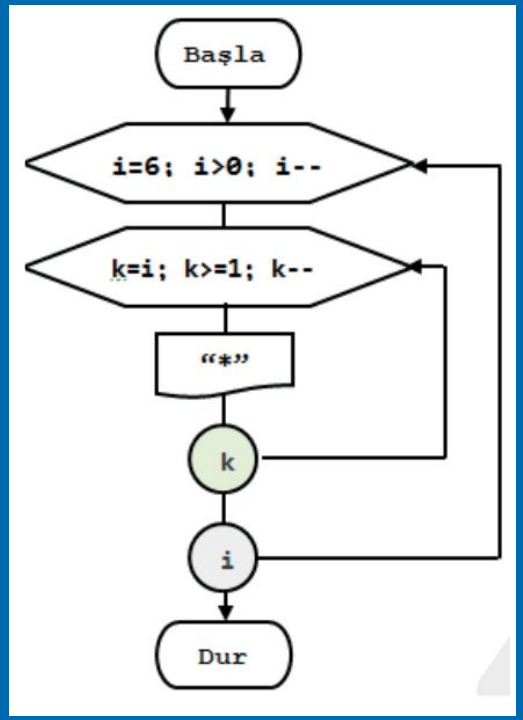
Aşağıdaki deseni ekrana yazdıracak programı kodlayınız.



İstenen çıktı:

```
******
*****
****
***
**
*
```

Algoritması:



C / C++ Kodu:

```
#include <stdio.h>
int main() {
for(int i=6; i>0; i--) {
  for(int k=i; k>=1; k--) {
    printf("*");
  } //for k sonu
  printf("\n"); //Bir satır atla
} // for i sonu
return 0;
} //main sonu
```

TASK: Aşağıdaki çıktıyı verecek kodu yazınız.

```
1 1 1 1 1 0 1 1 1 1 1

1 1 1 1 0 0 0 1 1 1 1

1 1 1 0 0 0 0 0 1 1 1

1 1 0 0 0 0 0 0 0 1 1

1 0 0 0 0 0 0 0 0 0 1
```

Çıktı Sorusu

```
//Verilen programda j=j-2; deyimi kaç kez çalıştırılır

#include <stdio.h>
int main() {
   int i, j, s=0;
   for (i=0; i<29; i++) {
        if (i%2==1) {
            j=i;
        while (j>0)
            j=j-2;
            s++; // sayac değişkeni
        } // if sonu
   }
   //printf("\'j=j-2\' %d kez çalıştırılır", s);
return 0; }
```

Çıktı Sorusu

Aşağıdaki program parçası ekrana kaç tane 'A' karakteri yazar?

```
for (i=0;i<N;i++)
     for (j=i;j<N;j++)
          printf("A\n");
```

A) N^2
B) $N^2 + 2N + 1$
C) $\frac{N^2}{2} + \frac{N}{2}$
D) $\frac{N^2}{2} + \frac{3N}{2} + 1$
E) N^3

Aşağıdaki kod parçası ekrana kaç kere "Tubitak" yazacaktır?

```
int i = 0;

  while (i < 5)
  {
    for(int j = (++i); j < 5; j=j+2)
      printf("Tubitak\n");
      break;
    i++;
  }
```

Seriler

- $\sum_{k=0}^{\infty} \frac{1}{k!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots = e$

`St=1+1/1!+1/2!+1/3!+1/4!+..... +1/n!`

şeklinde verilen serinin toplamını bulunuz

Çözüm:

Bu seri e sayısının açılımını vermektedir.

Bu serinin her bir teriminin paydasının faktöriyeli alınmakta ve seri toplam değişkenine (ST) eklenmektedir.

Seriler Çözümü

```
#include <iostream>
#include <math.h>
using namespace std;
int main() {
float ST=0;
int n, f=1;
cout<<"Terim sayisi.:";
cin>>n;
//Döngü içinde seri elemanlarını topla
for (int x=1; x<=n; x++){
   f=f*x;
   ST=ST+(1.0/f);
}
cout<<"Toplam=" << 1+ ST << endl; //ST yi Yaz
//Bu seri aslında e sayısını hesaplamakta, sağlaması:
cout<<"Math e sayısı..:"<< M_E << endl;
return 0;
}
```

TASK: Aşağıdaki seriyi hesaplayacak kodu yazınız.

$$\sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)!} = \frac{1}{1!} - \frac{1}{3!} + \frac{1}{5!} - \frac{1}{7!} + \frac{1}{9!} + \cdots$$