BILGISAYAR PROGRAMLAMA II

BAHAR- 4. HAFTA

Structure of Program (Programlamanın Yapısı)

```
1 // My first program in C++
2 # include <iostream>
3
4 int main ()
5 {
6
7 }    std::cout << Hello World! ";
```

```
Hello World!
```

→ programın 4. satır ile 7 satır arasını aşağıdaki gibi tek satırda yazabiliriz

```
4 int main () { std::cout << Hello World! "; }
```

→ iki eğik çizgi (//) anlamı, programcının programı etkilemeyen bir yorum yapmasını sağlar.

→ 5. satırdaki açık parantez ({) ona fonksiyon tanımının başlangıcını ve 7. satırdaki kapatma parantezi (}) ise sonucunu gösterir.

→ 4. satırdaki int main () kodu bir fonksiyon bildirimini başlatır.

```
1 // my second program in C++
2 #include < iostream >
3
4 int main()
5 {
6    std:: cout << "Hello World! ";
7    std :: cout << "I'm a C++ program";
8 }
```

```
Hello world! I'm a C++ program
```

→ Programdada 4. satır ile 8. satır arasını aşağıdaki gibi tek satırda yazabiliriz.

```
4 int main () { std :: cout << "Hello World" ; std::cout << I'm a C++
  program"; }
```

→ ya da kodu bölerek de yazabiliriz.

```
4 int main ()
5 {
6    std :: cout <<
7      "Hello World! ";
8    std :: cout
9    << "I'm a C++ program";
10 }
```

- using namespace std

```
using namespace std;
```

```
1 // my second program in C++
2 #include <iostream>
3 using namespace std;
4
5 int main ()
6 {
7    cout << "Hello World! ";
8    cout << "I'm a C++ program";
9 }
```

```
Hello World! I'm a C++ program
```

Variables and Types (Değişkenler ve Tipler)

```
a=5;
b=2;
a = a+1;
result = a-b;
```

- identifiers (Tanımlayıcılar)

→ Geçerli bir tanımlayıcı bir ya da daha fazla karakter, sayı ya da alt çizgi dizesidir.

- Fundamental data types (Genel data tipleri)

→ Character types (Karakter Tipleri) : Tek karakter örneğin: 'A', 'a', '$'.
   Numerical integer types (Tam sayı tipi)
   Floating - point types (Ondalık tipi)
   Booleon types (Booleon tipi)

- Decloration of Variables (Değişkenlerin Deklarasyonu)

```
int a;
float mynumber
```

```
int a,b,c;
```

```
int a;
int b;
int c;
```

```
//operating with variables

# include <iostream>
using namespace std;

int main ()
{
   // declaring variables (değişken bildiren) :
   int a,b;
   int result;

   // process (yönlendirmek);
   a=5;
   b=2;
   a=a+1;
   resut =a-b;

   // print out the resut (sonucu yazdır) :
   cout << resut;

   // terminate the program (programı sonlandırma) :
   return 0;
}
```

```
4
```

→ return 0 kalıbı, program hatasızsa hata 0'dır. Bu kalıp ile çalışır.

- initialization of variables (Değişkenlere değer atamak)

```
int x=0;
```

→ x'e ilk değer olarak 0 atar.

```
int x (0);
```

```
int x {0};
```

• introduction to strings

```
1  // my first string
2  #include <iostream>
3  #include <string>
4  using namespace std;
5
6  int main ()
7  {
8
9      string mystring;
10     mystring = "This is a string";
11     cout << mystring;
       return 0;
12 }
```

```
This is a string
```

→ 8. ve 9. satırı aşağıdaki gibi kodlar ile de yazabiliriz.

```
8 string mystring = "This is a string";
```

```
8 string mystring ("This is a string");
```

```
8 string mystring {"This is a string"};
```

BİLGİSAYAR PROGRAMLAMA II

BAHAR - 5. HAFTA

Operators

- assigment operator (=) (Eşittir operatörü)

```
x=5;
```

```
x=y;
```

- arithmetic operators (+, -, *, /, %)

→ + (addition) (toplama)
- (subtraction) (çıkarma)
* (multiplication) (çarpma)
/ (division) (bölme)
% (modulo) (kalan)

- compound assignment (+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=)

→ y += x ;	ifadenin anlamı	y = y + x;
x -= 5 ;	" "	x = x - 5;
x /= y ;	" "	x = x / y;

- increment and decrement (++, --)

```
1 ++x;
2 x+=1;
3 x=x+1;
```

→ Yukarıdaki 1., 2. ve 3. satırdaki kodların anlamları birbirlerinin aynısıdır. x sayısını 1 arttırır.

- Relational and comparison operators (==, !=, >, <, >=, <=)
(İlişkisel ve Karşılaştırma Operatörleri)

	tanımı
→ ==	eşittir
!=	eşit değildir
<	küçüktür
>	büyüktür
<=	küçük yada eşittir
>=	büyük yada eşittir.

- Logical operators (!, &&, ||) (Mantıksal Operatörler)

→ && operator (and)

a	b	a&&b
true	true	true
true	false	false
false	true	false
false	false	false

|| operator (or)

a	b	a||b
true	true	true
true	false	true
false	true	true
false	false	false

```
( (5==5) && (3>6) )    // false olarak değerlendirir (true && false)
( (5==5) || (3>6) )    // true   "   "            (true || false)
```

```
!(5==5)    // false olarak değerlendirir çünkü ifade doğru olduğu için
!(6<=4)    // true    "    "             çünkü ifade yanlış olduğu için
!true      // false   "    "
!false     // true    "    "
```

→ ! işareti ingilizce'deki "not" anlamındadır Yani değildir.

- sizeof (Boyut)

```
x = sizeof (char);
```

Basic Input / Output

- Standard output (cout)

→ Standart olarak varsayılan çıktı ekranıdır ve erişmek için tanımlanan C++ akış nesnesi cout'tur.

```
cout << "output sentence";  // çıktı cümlesini ekrana yazdırır
cout << 120;                // 120 sayısını ekrana yazdırır.
cout << x;                  // x değerini ekrana yazar
```

- standard input (cin)

```
int age;
cin >> age;
```

→ Standart girdi varsayılan olarak klavyedir ve erişmek için tanımlanan C++ stream nesnesi cin'dir.

```
cin >>a>>b;
```

```
cin >> a;
cin >> b;
```

- cin and strings

```
string mystring;
cin >> mystring;
```

```
// cin with strings
#include <iostream>
#include <string>
using namespace std;

int main ()
{
  string mystr;
  cout << what's your name ? ";
  getline (cin, mystr);
  cout << "Hello" << mystr << ".\n";
  cout << "what is your favourite team? ";
  getline (cin, mystr);
  cout << "I like" << mystr << "too!\n";
  return 0;
}
```

```
what's your name? Homer Simpson
Hello Homer Simpson
what is your favorite team? The Isotopes
I like The Isotopes too!
```

BILGISAYAR PROGRAMLAMA II

BAHAR - 6. HAFTA

Statements and flow Control (İfadeler ve Akış Kontrolleri)

● Selection statements : if and else

```
if (x==100)
cout << "x is 100";
```

→ x tam olarak 100 değil ise, bu ifade yok sayılır ve hiçbir şey yazılmaz.

```
if (x==100)
{
    cout << "x is ";
   cout << x ;
}
```

→ Koşul yerine getirildiğinde tek bir ifadeden daha fazlasını dahil etmek istiyorsanız, bu ifadeler bir blok oluşturan parantez içine alınacaktır ({}).

→ Bu programı tek satırda da yazabiliriz.

```
if (x==100) { cout << "x is "; cout << x ; }
```

```
if (x==100)
   cout << "is 100";
else
   cout << "x is not 100";
```

→ Bu programdaki söz dizimi ;

```
if (condition) (şart)
    statement 1 (cümle 1)
else
    statement 2 (cümle 2)
```

Statement 1, koşulun doğru olması durumunda yürütülür değilse statement2 yürütülür.

```
if (x>0)
    cout << "x is positive";
else if (x<0)
    cout << "x is negative";
else
    cout << "x is 0";
```

→ else if yapısı, bir değer aralığının kontrol edilmesi amacıyla birleştirilebilir.

● Iteration statements (Loops) (Yineleme Cümleleri (Döngüler))

1- The while loop

→ Bu programdaki söz dizimi ;

```
while (expression) (ifade)
    statement (cümle)
```

2- The do-while loop

→ Söz dizimi ; do statement while (condition)
(şart)

3- The for lop

→ Söz dizimi ; for (initialization ; condition ; increase) statement ;
(başlatma) (şart) (arttırmak)

// countdown using a for lop
#include <iostream>
using namespace std;
 	
int main()
{
  for (int n=10; n>0; n--) {
    cout << n << ", ";
}
  cout "liftoff!\n";
}
	10,9,8,7,6,5,4,3,2,1, liftoff!

for (int $\boxed{n=10}$; $\boxed{n>0}$; $\boxed{n--}$)

→ condition (şart)

→ increase (arttırmak)

→ initialization (başlatma)

● Jump statements (Atlama Cümleleri)

1- The break statement

→ Sonsuz döngüyü sonlandırmak veya yarıda kesmeye zorlamak için kullanılabilir.

2- The continue statement

→ Sayıyı atlamayı sağlar.

3- The goto statement

● Another selection statement : switch

→ Amacı, birkaç olası sabit ifade arasında bir değeri kontrol etmektir. Bu if-else ifadelerini birleştirmeye benzer, ancak sabit ifadelerle sınırlıdır.

Söz dizimi ;

```
switch (expression) (ifade)
{
    case constant1:
       group-of-statements-1 ;
       break;
    case constant2 :
       group-of-statements-2 ;
       break;
    .
    .
    default;
       default-group-of-statements
}
```

switch example	if-else example

switch(x) {
  case 1 :
    cout << "x is 1";
    break ;
  case 2 :
    cout << "x is 2";
    break ;
  default
    cout << "value of x unknown";
}
	if (x==1 {
	    cout << "x is 1";
	}
	else if (x==2) {
	    cout << "x is 2";
	}
	else {
	    cout << value of x unknown";
	}

```
switch (x) {
    case 1;
    case 2;
    case 3;
       cout << "x is 1,2 or 3";
       break ;
   default :
      cout << "x is not 1,2 nor 3";
}
```

Functions

type name (parameter 1 , parameter 2, ...) { statements }

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

int addition (int a, int b)
{
  int r;
  r = a+b;
  return r;
}

int main()
{
   int z;
   z = addition (5,3);
   cout << "The result is " << z;
}
```

```
The result is 8
```

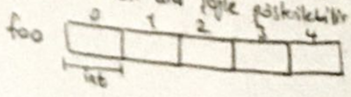
```
int addition (int a , int b)
               ↑       ↑
z addition (   5   ,   3  )
```

BİLGİSAYAR PROGRAMLAMA II

Arrays

BAHAR - 9. HAFTA

Örneğin, fo olarak adlandırılan int türünde 5 tam sayı değerini içeren bir dizi şöyle gösterilebilir:

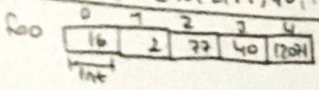


type name [elements];

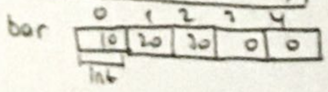
```
int foo[5];
```

- Initializing arrays

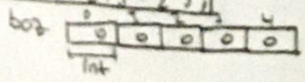
```
int foo[5] = { 16, 2, 77, 40, 12071 };
```



```
int bar[5] = { 10, 20, 30 };
```



```
int baz[5] = { };
```


baz
0
4
0
0
0
0
0
int