4. Fonksiyonlar - I

C++'da alt programlara fonksiyon denilmekte ve "alt program", "prosedür", "subroutine" sözcüklerinin karşılığı olarak kullanılmaktadır. Fonksiyonlar belirli bir amaç için yazılmış program parçalarıdır ve programcının programını modüler hale getirmesini sağlamaktadır. Bir C++ programı kendi içinde var olan hazır fonksiyonlar (cout, cin, sqrt(x), exp(x) …) ile kullanıcının tanımladığı fonksiyonlardan oluşmaktadır.

Bir programı fonksiyonlar ile yazmanın bir çok nedeni vardır.

- Fonksiyonlar yapısal programlamanın temel yapı taşlarıdır. Programı küçük parçalara ayırarak, yönetilebilir, daha anlaşılabilir programlar yazabilmeyi sağlamaktadır. Fonksiyonlar bir başka ifade ile böl ve yönet (*divide and conquer*) prensibinin uygulanmasına imkan vermektedir.

- Başka bir sebep ise yazılımın yeniden kullanılmasıdır. Bu sayede, önceden yazılmış fonksiyonlar bloklar halinde birbirleri ardına yerleştirilerek yeni programlar yazılabilir.

- Üçüncü sebep ise programda kodları tekrar etmekten kurtulmaktır. Kodları fonksiyon haline getirerek paketlemek, kodların program içinde birçok noktada yalnızca fonksiyonu çağırarak kullanılmasını sağlar.

C++' da bir fonksiyon aşağıda gösterildiği şekilde tanımlanmaktadır.

```
geri dönüş tipi   fonksiyonun ismi  ( parametre listesi )
{
        Değişken tanımları
        Komutlar
        Return değer
}
```

Kullanıcının tanımladığı her fonksiyonun, amacına yönelik bir ismi vardır. Örneğin fonksiyon toplama işlemi yapıyorsa ismi "topla" olabilir. Fonksiyon geriye değer dönderiyorsa örneğin toplam değerini geriye dönderiyorsa bu değer return ile geriye dönderilir.

Geri dönüş tipi fonksiyonun geriye dönderdiği değerin tipini belirtir. Örneğin int, float, char, bool veya string olabilir. Eğer fonksiyon geriye değer döndermiyorsa geri dönüş tipi "void" olarak belirtilir.

Parametre listesi fonksiyona gönderilen veya girilen değişkenlerin sayısını ve tipini belirtir. Örnek olarak aşağıda tanımlanan int Kare(int x) şeklinde verilen fonksiyon kendisine gönderilen tamsayı(int) türünde değişken değerinin karesini alarak geriye döndermektedir.

```
int Kare(int x)
{
   int deger;
  deger=x*x;
   return deger;
}
```

Fonksiyon tanımlamalarında geri dönüş değerini yazım hatası oluşturur.

Bir değer ile dönmesi beklenen geri dönüş tipi belirtilmiş bir fonksiyonun, geri dönüş değerinin belirtilmemesi beklenmeyen hatalara yol açabilir

Geri dönüş tipi ***void*** *olarak bildirilmiş bir fonksiyonun, bir değer geri döndürmesi bir yazım hatasıdır.*

Bir fonksiyon içinde başka bir fonksiyon tanımlamak yazım hatasıdır.

Anlamlı fonksiyon isimleri ve anlamlı parametre isimleri kullanmak programları daha okunur yapar ve yorumların çok fazla kullanılmasını engeller

Eğer fonksiyonun tanımı main fonksiyonundan sonra yapılmış ise fonksiyon ismi parametre listesi ile birlikte main() fonksiyounundan önce programın başlangıç kısmında kütüphane dosyalarından sonra belirtilmelidir. Bu şekilde bildirime fonksiyon prototipi denir. Bir fonksiyon prototipi, derleyiciye fonksiyon tarafından döndürülen verinin tipini, fonksiyonun almayı beklediği parametre sayısını, parametrelerin tiplerini ve parametrelerin sırasını bildirir.

maksimum fonksiyonunun prototipi

```
int maksimum ( int, int, int );
```

şeklindedir.

Bu fonksiyon prototipi, **maksimum** fonksiyonunun **int** tipinde 3 argüman alacağını ve sonuç olarak **int** tipinde bir sonuç döndüreceğini belirtir. Dikkat edilirse, fonksiyon prototipiyle **maksimum** fonksiyonunun tanımının ilk satırı, parametrelerin isimlerinin (**x**, **y** ve **z**) bulunmayışı haricinde aynıdır.

Fonksiyon içinde tanımlanan bütün değişkenler yerel değişkenlerdir.bu değişkenler sadece tanımlandığı fonksiyon içinde geçerlidir. Parametre belirten değişkenlerde yerel değişkendir.

Aşağıda verilen programda maksimum isimli fonksiyona üç tamsayı parametre olarak gönderilmekte ve en büyük değer geriye döndürülmekte ve ana program içinde dönen değer ekrana yazdırılmaktadır.

```
//Üç tamsayının en büyüğünü bulmak
#include <iostream.h>
#include <conio.h>
int maksimum( int, int, int );    //fonksiyon prototipi
int main( )
{
   int a, b, c;
   cout<< "3 tamsayı giriniz: ";
   cin>>a>>b>>c ;
   cout<< "Maksimum : "<< maksimum( a, b, c )<<" dir\n" ;
   gech();
  return 0;
}

//maksimum fonksiyonunun tanımı
int maksimum( int x, int y, int z )
{
   int maks = x;
   if ( y > maks )
      maks = y;
   if ( z > maks )
      maks = z;
   return maks;
}
```

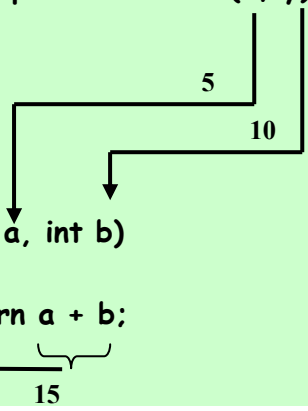
```
3 tamsayi giriniz: 3 7 45
Maksimum : 45 dir
_
```

Fonksiyonlara parametre aktarım yöntemleri
Fonksiyonlara parametre aktarımı iki şekilde yapılmaktadır.

Değer ile çağırma(Call by value)
Adres ile çağırma(Call by reference)

Değer ile çağırma yönteminde fonksiyon parametrelerine değişkenlerin değeri kopyalanır. Gerçek değişkenin değerinde herhangi bir değişiklik olmaz. Değer ile çağırma yönteminde geriye tek bir değer dönderilmektedir.

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

int add(int, int);

int main()
{
int x,y;

x=5;  y=10;

cout << "Toplam = " <<add(x, y)<< endl;

getch();
return 0;
}

int add(int a, int b)
{
      return a + b;
}
```




```
Toplam = 15
```

Adres ile çağırma

Geriye birden fazla değer döndermek gerekirse veya fonksiyon içinde yapılan işlemlerin fonksiyona gönderilen parametre değişkenini etkilemesi isteniyorsa adres ile çağırma yöntemi kullanılır. Bu yöntem ile değişkenlerin değerleri değil adresleri fonksiyona parametre olarak gönderilir. Aşağıda verilen programda

degistir1(….) fonksiyonunda fonksiyona a ve b değerlerin kopyası yerel değişkenler x ve y ye geçirilmektedir . Değişim bu yerel değişkenler üzerinde olmaktadır. Fonksiyon sonlandığında yerel değişkenlerde yok olmaktadır. Dolayısı ile a ve b değişkenleri yer değişmemiş lmaktadır.

Degistir2 (…) fonksiyonunda ise a ve b değişkenlerinin adresleri fonksiyona geçirilmektedir.adreslerin içindeki değerler yer değiştiğinden ve fonksiyondan çıktıktan sonrada bu değişken adresleri kaybolmayacağından değişim işlemi gerçekleşmektedir.

```
#include <iostream.h>
#include <conio.h>
void degistir1( int x, int y )
{
	int gecici;
	gecici = x;
	x = y;
	y = gecici;
}

void degistir2( int *x, int *y )
{
	int gecici;
	gecici = *x;
	*x = *y;
	*y = gecici;
}

int main(  )
{
    int a, b;
    a = 12;
    b = 27;
    cout<< "degistir1 fonk dan once a : "<<a<<"  b : "<<b<<endl;
    degistir1(a, b);
    cout<< "degistir1 fonk dan sonra a : "<<a<<"  b : "<<b<<endl;

    cout<< "degistir2 fonk dan once a : "<<a<<"  b : "<<b<<endl;
    degistir2(&a, &b);// Argumanları aktarırken, baslarina & konur
    cout<< "degistir2 fonk dan sonra a : "<<a<<"  b : "<<b<<endl;

    getch();
    return 0;
}
```

```
degistir1 fonk dan once a : 12  b : 27
degistir1 fonk dan sonra a : 12  b : 27
degistir2 fonk dan once a : 12  b : 27
degistir2 fonk dan sonra a : 27  b : 12
```

Aşağıda verilen programda ise birden fazla değer fonksiyondan geriye dönderilmektedir.

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

void hesapla( int x, int y , int *top, int *fark, int *carp)
{
       *top=x+y;
       *fark=x-y;
       *carp=x*y;

}

int main(  )
{
       int a, b,top1,fark1,carp1;
       a = 3;
       b = 10;
       hesapla(a,b,&top1,&fark1,&carp1);
    cout<<"a : "<<a<<" b: "<<b<<endl;
    cout<< "toplam : "<<top1<<" fark : "<<fark1
         <<" carp : "<<carp1<<endl;
       getch();
    return 0;
}
```

```
a : 3 b: 10
toplam : 13 fark : -7 carp : 30
```