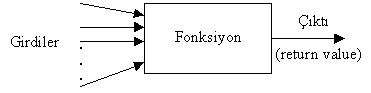
**FONKSİYONLAR**



C Programlama Dili fonksiyon olarak adlandırılan alt programların birleştirilmesi kavramına dayanır. Bir C programı bir ya da daha çok fonksiyonun bir araya gelmesi ile oluşur. C Dilini öğrenmek için ilk önce fonksiyon oluşturmayı ve onların diğerleri ile birlikte kullanılmasını öğrenmek gerekir.

Gerçek hayattaki problemlerin çözümü için geliştirilen programlar çok büyük boyutlardadır. Daha büyük programlar yazmanın en kolay yolu onları küçük parçalar halinde yazıp sonra birleştirmektir. Böylece çok büyük boyutlardaki program kodlarını yönetmek daha kolay olacaktır. Fonksiyonların faydaları:

* Kodun gereksiz yere büyümesini engeller.
* Okunabilirliği artırarak algılamayı kolaylaştırır.
* Programın test edilmesini ve hataların bulunmasını kolaylaştırır.
* Güncelleştirilebilir olmasını ve yeniden kullanabilme kolaylığı sağlar.

Her fonksiyonun bir adı ve fonksiyona gelen değerleri gösteren argümanları (bağımsız değişkenleri) vardır.C dilinde hazırlanan bir fonksiyonun genel yapısı şöyledir:

*FonksiyonTipi FonksiyonAdı(arguman listesi) argumanların tip bildirimleri*

*{*

*Yerel değişkenlerin bildirimi*

*...*

*fonksiyon içindeki deyimler veya diğer fonksiyonlar*

*...*

*}*

**Örnek 1:**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

void myfunction()

{

printf("Merhaba Fonksiyon2\n");

}

int main()

{

printf("Merhaba Fonksiyon1\n");

myfunction();

getch();

}

**Örnek 2:**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

int topla(int x, int y) {

int sonuc;

sonuc = x + y;

return sonuc;

}

/\* veya \*/

int topla(int x, int y) {

return (x+y);

}

int main(){

int a,b,t;

printf("1. Sayı Giriniz...:");scanf("%d",&a);

printf("2. Sayı Giriniz...:");scanf("%d",&b);

t=topla(a,b);

printf("%d",t);

getch();

}

**Örnek 3:**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

int topla(int x, int y) {

int sonuc;

sonuc = x + y;

return sonuc;

}

int cikar(int x, int y) {

int sonuc;

sonuc = x - y;

return sonuc;

}

int carpma(int x, int y) {

int sonuc;

sonuc = x \* y;

return sonuc;

}

float bolme(int x, int y) {

float sonuc;

sonuc = (float)x / y;

return sonuc;

}

int main(){

int a,b,top,cik,carp;

float bol;

printf("1. Sayı Giriniz...:");scanf("%d",&a);

printf("2. Sayı Giriniz...:");scanf("%d",&b);

top=topla(a,b);

cik=cikar(a,b);

carp=carpma(a,b);

bol=bolme(a,b);

printf("%d , %d , %d , %f",top,cik,carp,bol);

getch();

}

**FONKSİYON BİLDİRİMİ**

Main bloğundan önce ve main bloğundan sonra olmak üzere fonksiyon bildirimi iki şekilde yapılmaktadır. Eğer fonksiyon main bloğundan sonra yazılacaksa main bloğundan önce fonksiyonunun ön bildiriminin aplması gerekmektedir.

1. Main bloğundan önce fonksiyonun yazılması

...

**int topla(int x,int y)** /\* fonksiyon \*/

**{**

**...**

**}**

...

main()

{

...

}

1. Main bloğundan sonra fonksiyonun yazılması

...

int topla(int x, int y); /\* fonksiyon ön bildirimi \*/

...

main()

{

...

}

...

**int topla(int x, int y)** /\* fonksiyon \*/

**{**

**...**

**}**

**Örnek 4:**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

int faktoriyel(int n); // ön bildirim

/\* veya \*/

int faktoriyel(int); // ön bildirim

int main()

{

int a,f;

printf("Sayı Giriniz...:");scanf("%d",&a);

f=faktoriyel(a);

printf("%d",f);

getch();

}

int faktoriyel(int n)

{

int i=1, f=1;

while(i<=n)

f \*= i++;

return f;

}

**Void Fonksiyonlar**

Fonksiyonlar her zaman geri dönüş değeri göndermeyebilirler. Bu gibi durumlarda fonksiyon tipi *void* olarak belirlenmektedir. Bu tip fonksiyonlarda *return* değimi kullanılmaz ve fonksiyon blok sonuna kadar işletilir.

**Örnek 5:**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

void tek\_mi\_cift\_mi( int sayi )

{

if( sayi%2 == 0 )

printf( "%d, çift bir sayıdır.\n", sayi );

else

printf( "%d, tek bir sayıdır.\n", sayi );

}

int main( void )

{

int girilen\_sayi;

printf( "Lütfen bir sayı giriniz> " );

scanf( "%d",&girilen\_sayi );

tek\_mi\_cift\_mi( girilen\_sayi );

getch();

return 0;

}

**Lokal ve Global Değişkenler**

Bir fonksiyon içerisinde tanımladığınız değişkenler, sadece o fonksiyon içerisinde tanımlıdır ve main bloğundan bu değişkenlere ulaşamayız. Bu tip değişkenlere lokal değişkenler denir.  Programın herhangi bir noktasından erişebileceğiniz ve nerede olursa olsun aynı bellek adresini işaret eden değişkenler, global değişkenlerdir. Hep aynı bellek adresi söz konusu olduğun için, programın herhangi bir noktasında yapacağınız değişiklik, global değişkenin geçtiği bütün yerleri etkiler.

**Örnek 6:** Verilen örnekte kendi yazdığımız iki fonksiyonda ve main fonksiyonunda a lokal değişkeni tanımlanmıştır. Bu değişkenlerin hepsi farklı bir bellek bölgesini işaret etmektedir.

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

// Verilen sayinin karesini hesaplar

void kare\_hesapla( int sayi )

{

// kare\_hesapla fonksiyonunda

// a degiskeni tanimliyoruz.

int a;

a = sayi \* sayi;

printf( "Sayının karesi\t: %d\n", a );

}

// Verilen sayinin kupunu hesaplar

void kup\_hesapla( int sayi )

{

// kup\_hesapla fonksiyonunda

// a degiskeni tanimliyoruz.

int a;

a = sayi \* sayi \* sayi;

printf( "Sayının küpü\t: %d\n", a );

}

int main( void )

{

// main( ) fonksiyonunda

// a degiskeni tanimliyoruz.

int a;

printf( "Sayı giriniz> ");

scanf( "%d",&a );

printf( "Girdiğiniz sayı\t: %d\n", a );

kare\_hesapla( a );

// Eger a degiskeni lokal olmasaydi,

// kare\_hesapla fonksiyonundan sonra,

// a'nin degeri bozulur ve kup yanlis

// hesaplanirdi.

kup\_hesapla( a );

getch();

return 0;

}

**Örnek 7:** Örnek 6 daki problem global değişken kullanılarak yapılmıştır.

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

int sonuc=0;

// Verilen sayinin karesini hesaplar

void kare\_hesapla( int sayi )

{

sonuc = sayi \* sayi;

}

// Verilen sayinin kupunu hesaplar

void kup\_hesapla( int sayi )

{

sonuc = sayi \* sayi \* sayi;

}

int main( void )

{

int a;

printf( "Sayı giriniz> ");

scanf( "%d",&a );

printf( "Girdiğiniz sayı\t: %d\n", a );

kare\_hesapla( a );

printf( "Sayının karesi\t: %d\n", sonuc );

kup\_hesapla( a );

printf( "Sayının küpünü\t: %d\n", sonuc );

getch();

return 0;

}

**Örnek 7:** Örnek 6 daki problem 2 adet global değişken kullanılarak yapılmıştır.

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

int sonuc=0,a=0;

// Verilen sayinin karesini hesaplar

void kare\_hesapla( void)

{

sonuc = a \* a;

}

// Verilen sayinin kupunu hesaplar

void kup\_hesapla( void )

{

sonuc = a \* a \* a;

}

int main( void )

{

printf( "Sayı giriniz> ");

scanf( "%d",&a );

printf( "Girdiğiniz sayı\t: %d\n", a );

kare\_hesapla();

printf( "Sayının karesi\t: %d\n", sonuc );

kup\_hesapla();

printf( "Sayının küpünü\t: %d\n", sonuc );

getch();

return 0;

}

**Main fonksiyonuna Parametre Aktarımı**

**Örnek 8:**

#include <stdio.h>

#include <stdlib.h>

int main(int argsay, char \*argvek[]){

    int toplam;

    if(argsay < 3){

      printf("Eksik parametre !\n");

      exit(1);

    }

    if(argsay > 3){

       printf("Cok fazla parametre !\n");

       exit(1);

    }

    toplam = atoi(argvek[1]) + atoi(argvek[2]);

    printf("Toplamlari %d\n",toplam);

  return 0;

}

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <string.h>

int main( int argc, char \*argv[])

{

int i,t=0,x;

for (i = 0; i < argc; ++i){

puts(argv[i]);

t=t+atoi(argv[i]);

}

printf("%d",t);

return t;

}