

Fonksiyonlar

Ders 6

İçerik

- ▶ Fonksiyon tanımlama ve modüler programlama
- ▶ Hazır kütüphane fonksiyonları
- ▶ Kendi fonksiyonunu yazma



Fonksiyonlar

▶ Fonksiyon Kullanmanın Sebepleri

- ▶ Büyük hesaplama işlerini küçük parçalara bölme
- ▶ Daha önceden başkalarının (veya kendinizin) yaptıklarının üzerine inşa etme
- ▶ Bazı işlemlerin içeriğini programın diğer parçalarından gizleyebilme

▶ C de fonksiyonlar

- ▶ C programları genelde küçük küçük fonksiyonlardan oluşur
- ▶ Tüm program bir veya daha fazla kaynak dosyasından meydana gelir
- ▶ Kaynak dosyaları tek başına derlenebilirler ve daha önceden derlenmiş kütüphane dosyaları ile birlikte yüklenirler.



Fonksiyonlar

- ▶ Fonksiyon bir işlev gören ve farklı fonksiyonlardan çağrılabilen organize edilmiş bir kod bloğudur.

```
return_tip isim(argüman deklarasyonları){
```

```
    /* fonksiyon vucudu */
```

```
    return ifade;
```

```
}
```

- ▶ **main()** ve **printf()** fonksiyonu dahil olmak üzere şimdiye kadar bir kaç fonksiyonu zaten gördük.

```
int main (){
```

```
    /* bir seyler yap */
```

```
    return 0; /* basarılı */
```

```
}
```

Fonksiyonun return tipi
int verildiği için fonksiyon
tam sayı bir değer
döndürmek zorunda
Burada bu değer 0.

Örnek

- ▶ Verilen a, b, c katsayılarını kullanarak denklemin köklerini bulan bir program yazınız.
- ▶ Discriminant

$$\Delta = b^2 - 4ac$$

- ▶ Eğer $\Delta > 0$, kökler

$$x_1 = \frac{-b + \sqrt{\Delta}}{2a} \quad x_2 = \frac{-b - \sqrt{\Delta}}{2a}$$

Karekökü `sqrt` kütüphane fonksiyonunu çağırarak hesaplayabiliriz



Kütüphane Fonksiyonunu Çağırma: sqrt

Fonksiyon Çağırısı:

- sqrt fonksiyonu argüman 4.0 ile çağrıldı
- Argüman fonksiyona gönderilen bir “mesajdır”.

Programın çalışması sqrt fonksiyonundan devam eder. Bu çağrı ile $x = 4.0$ şeklinde düşünebilirsiniz

Fonksiyonun Return değeri

`sqrt(4.0);`

`double sqrt(double x);`

`2.0`

```
sqrt(4);
```

```
double d1 = sqrt(4);  
double d2 = sqrt(4)/2;
```

```
double r = 4.0;  
printf("%lf\n", sqrt(r));
```

```
printf("%lf\n", sqrt(4));  
printf("%lf\n", d1);  
printf("%lf\n", d2);
```

Sqrt Fonksiyonu ile Denklemin Kökleri

```
#include <stdio.h>
#include <math.h>
int main()
{
    double a = 1.0, b = 2.0, c = 1.0;
    double d; /* discriminant: b^2 - 4ac */
    double x1, x2; /* outputs: roots (kokler) */

    d = pow(b, 2) - 4*a*c;

    if (d >= 0){
        x1 = (-b + sqrt(d))/(2*a);
        x2 = (-b - sqrt(d))/(2*a);

        printf("the roots (denklemin kokleri)\n"
               "x1: %lf\t x2: %lf", x1, x2);
    }
    return 0;
}
```

Kendi Fonksiyonumuzu Tanımlama

```
#include <stdio.h>

int mutlak(int a){
    if (a > 0)
        return a;
    else
        return -a;
}

int main(){
    int b=mutlak(-5);
    printf("%d", b);
    return 0;
}
```

- ▶ Fonksiyonun gerçekleştirilmesi
fonksiyon tanımlamasıdır (function definition).
- ▶ Fonksiyonun return tipi
 - ▶ **int**
- ▶ Fonksiyonun ismi
 - ▶ **mutlak**
- ▶ Argüman deklarasyonları (fonksiyon parametreleri)
 - ▶ **a**

Fonksiyon Tanımlaması

```
#include <stdio.h>

//A-65 küçük a-97
char buyuk_cevir(c) {
    return c-32;
}

int main(){
    char c= 'k';
    char b=buyuk_cevir(c);
    printf("%c", b);
    return 0;
}
```

- ▶ Fonksiyonun gerçekleştirilmesi
fonksiyon tanımlamasıdır (function definition).
- ▶ Fonksiyonun return tipi
 - ▶ **char**
- ▶ Fonksiyonun ismi
 - ▶ **buyuk_cevir**
- ▶ Argüman deklarasyonları (fonksiyon parametreleri)
 - ▶ **char c**

Fonksiyon Deklarasyonu

```
#include <stdio.h>

int mutlak(int a);

int main(){
    int b = mutlak(-5);
    printf("%d", b);
    return 0;
}

int mutlak(int a){
    if (a > 0)
        return a;
    else
        return -a;
}
```

- ▶ Fonksiyon kullanılmadan önce ya tanımlanmalı yada deklare edilmelidir
- ▶ **int mutlak(int a);**
 - ▶ Fonksiyon prototipidir.
- ▶ Birçok C fonksiyon prototipleri **header dosyalarında** verilmektedir.

Farklı Dosyalarda

► myprogram.c

```
#include <stdio.h>

int mutlak(int a);

int main(){
    int b = mutlak(-5);
    printf("%d", b);
    return 0;
}
```

► mymath.c

```
int mutlak(int a){
    if (a > 0)
        return a;
    else
        return -a;
}
```

- Beraber derleyip tek bir exe oluşturuyoruz

```
gcc mymath.c myprogram.c -o program.exe
```



Birden Fazla Parametre

```
#include <stdio.h>
```

```
float hacimHesapla (float a, float b);
```

```
int main(){
```

```
//koni hacim  $\pi * r * r * h / 3$ 
```

```
    float r=5.2;
```

```
    float h=10.3;
```

```
    printf("%f", hacimHesapla (r,h));
```

```
    return 0;
```

```
}
```

```
float hacimHesapla (float r, float h)
```

```
{
```

```
    float pi = 3.14;
```

```
    float hacim= $\pi * r * r * h / 3$ ;
```

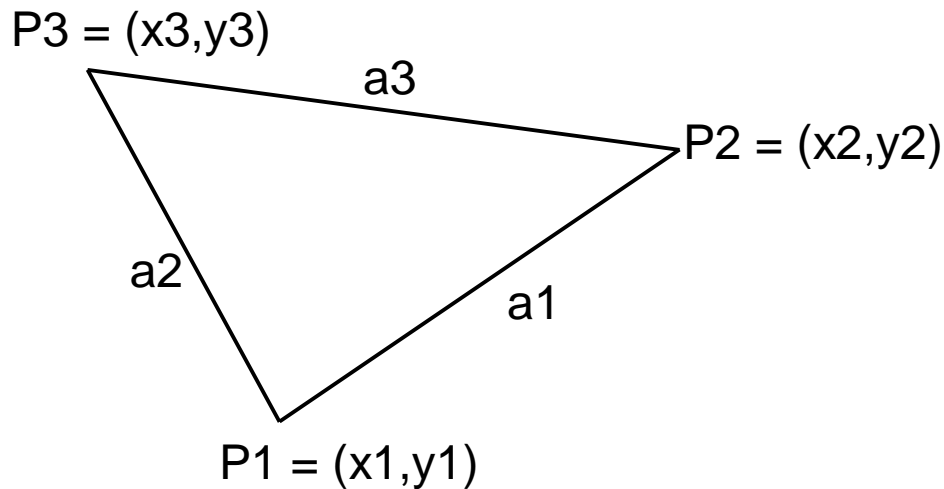
```
    return hacim;
```

```
}
```



Vaka Çalışması: Üçgenin Alanı

- Üçgenin üç noktasını kullanarak alanını hesaplayan bir program yazınız.



$$a1 = |P2 - P1|$$

$$a2 = |P3 - P1|$$

$$a3 = |P3 - P2|$$

$$a1 = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

$$a2 = \sqrt{(x3 - x1)^2 + (y3 - y1)^2}$$

$$a3 = \sqrt{(x3 - x2)^2 + (y3 - y2)^2}$$

$$A = \sqrt{p(p - a1)(p - a2)(p - a3)}$$

$$p = \frac{a1 + a2 + a3}{2}$$

İki Nokta Arasındaki Mesafe

- İki nokta arasındaki mesafeyi bulan fonksiyon

```
double mesafe(double x1, double y1, double x2, double y2)
{
    double d;
    d = sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
    return (d);
}
```



Üçgenin alanı

► Alan fonksiyonunun tanımı

```
double alan(double x1, double y1, double x2, double y2,  
            double x3, double y3) {  
    double a1, a2, a3; /* kenar uzunluklari */  
    double p;          /* Heron formulu icin */  
    double a;          /* alan */  
  
    a1 = mesafe(x1, y1, x2, y2);  
    a2 = mesafe(x1, y1, x3, y3);  
    a3 = mesafe(x2, y2, x3, y3);  
    p = (a1 + a2 + a3) / 2;  
    a = sqrt(p * (p - a1) * (p - a2) * (p - a3));  
  
    return a;  
}
```



Ana Program Fonksiyonu

```
/* Deklarasyonlar */
```

```
double mesafe(double x1, double y1, double x2, double y2);  
double alan(double x1, double y1, double x2, double y2,  
            double x3, double y3);
```

```
int main(){  
    double x1 = 0, y1 = 0;      /* point (nokta) 1 */  
    double x2 = 7, y2 = 10;     /* point (nokta) 2 */  
    double x3 = -5, y3 = 5;     /* point (nokta) 3 */  
  
    /* ucgenin alani */  
    double a = alan(x1, y1, x2, y2, x3, y3);  
    printf("%lf\n", a);  
    return 0;  
}
```

```
/* onceki slaytlarda verilen alan ve mesafenin tanimlamalari */
```



Void

- ▶ Void hiçbirşey manasına gelmektedir. Eğer fonksiyon bir değer döndürmüyorsa veya hiç bir argüman kabul etmiyorsa kullanılır.

```
void tamsayiyazdir(int a){  
    printf("%d", a);  
}
```



Fonksiyona Değerler Geçirilir – pass by value

- Bir fonksiyonu çağırdığımızda argümanın **değerleri** parametre değişkenlerine **kopyalanır**.

```
/* This function takes two int arguments, and returns an int. */  
int myfunc(int x, int y)
```

```
{  
    x *= 3;  
    ++y;  
    return x + y;  
}
```

```
/* A function that calls myfunc() */  
void caller_func(void)
```

```
{  
    int a=1, b=2, c, d;  
  
    c = myfunc(a,b); /* c = 6 */  
  
    d = a + b; /* d = 3 */
```


```
}
```

- ▶ Çağırana fonksiyona değeri döndürülmesi ve fonksiyonun çalışmasının bitirilmesi **return** ile olur.
- ▶ Çağırana fonksiyon bu değeri ister kullanır, isterse gözardı eder.

```
/* Prototype: two int arguments, and returns an int. */  
int an_algorithm(int, int);
```

```
void caller_func(void)  
{  
    int a=1, b=2, c;  
    c = an_algorithm(a,b); /* use return value */  
    an_algorithm(a,b); /* ignore return value (implicitly) */  
    (void)an_algorithm(a,b); /*ignore return value(explicitly) */  
}
```

```
int an_algorithm(int x, int y)  
{  
    return x*2 + x/y;  
}
```



Alıştırma

```
#include <stdio.h>
double f(double x1, double y1, double x2, double y2){
    x1 = (x1 - x2);
    y1 = (y1 - y2);
    return (x1 + y1);
}

int main(){
    double t1 = 4.2, t2 = 3, t3 = 2.3, t4 = -3.2;
    double fr = f(t1, t2, t3, t4);

    printf("%lf %lf %lf %lf", t1, t2, t3, t4);
    return 0;
}
```



Örnek

- Verilen tamsayının faktoriyelini hesaplayıp veren bir fonksiyon yazınız.

```
int faktoriyel(int n) {  
    int fact = 1;  
    for(int i = 1; i <= n; i++){  
        fact *= i;  
    }  
    return fact;  
}
```



Örnek

- ▶ Verilen tamsayının faktoriyelini hesaplayan fonksiyonun çağırılması.

```
#include <stdio.h>
int faktoriyel(int n);

int faktoriyel(int n){
    int fact = 1;
    for(int i = 1; i <= n; i++){
        fact *= i;
    }
    return fact;
}

int main(){
    int f1 = faktoriyel(5);
    int f2 = faktoriyel(10);
    printf("\n 5! = %d 10! = %d\n", f1, f2);
    return 0;
}
```

Örnek

- Verilen bir sayının üssünü bulan bir fonksiyon yazınız.

```
double ussu(double sayi, int us){  
    int i = 1;  
    double sayi2 = sayi;  
  
    if (us == 0){  
        return 1;  
    }  
  
    while( i < us ){  
        sayi2 = sayi*sayi2;  
        i++;  
    }  
    return sayi2;  
}
```



Örnek ussu fonksiyonunun kullanımı

```
#include <stdio.h>
```

```
double ussu(double sayi, int us);
```

```
double ussu(double sayi, int us){
```

```
    int i = 1;
```

```
    double sayi2 = sayi;
```

```
    if (us == 0){
```

```
        return 1;
```

```
    }
```

```
    while( i < us ){
```

```
        sayi2 = sayi*sayi2;
```

```
        i++;
```

```
    }
```

```
    return sayi2;
```

```
}
```

```
int main(){
```

```
    int a = 5, b = 2;
```

```
    double c = ussu(a, b);
```

```
    printf(" besin karesi: %lf", c);
```

```
    return 0;
```

```
}
```


Değişkenlerin Kapsamı

```
int b1 = 0;
/* bu noktada sadece b1 erişilebilir */
{
    int b2 = 0;
    /* bu noktada hem b1 ve hem de b2 */
    {
        int b3 = 0;
        /* bu noktada b1, b2 and b3 */
    }
    /*bu noktada sadece b1 and b2 */
    {
        int b4 = 0;
        /* bu noktada b1, b2 ve b4 */
    }
    /* bu noktada sadece b1 ve b2 */
}
/* sadece b1 */
```



C de Değişkenler

- ▶ **lokal değişkenler:**
Fonksiyonun veya bloğun içerisinde tanımlı.
- ▶ **formal parametreler:**
Fonksiyon tanımında ki parametreler.
- ▶ **global değişkenler:**
Bütün fonksiyonların dışında tanımlanmış değişkenler.

```
#include <stdio.h>

float hacimH (float r, float h);
float pi = 3.14;

int main(){
    //koni hacim pi*r*r*h/3
    float r=5.2;
    float h=10.3;
    printf("%f", hacimH(r,h));
    return 0;
}

float hacimH (float r, float h)
{
    float hacim=pi*r*r*h/3;
    return hacim;
}
```



Lokal Değişkenler ve Formal Parametreler

```
double alan(double x1, double y1, double x2, double y2,  
            double x3, double y3)  
{  
    double a1, a2, a3; /* side lengths: kenar uzunluklari */  
    double p;          /* for Heron's formula */  
    double a;          /* the area: alan */  
    a1 = mesafe(x1, y1, x2, y2);  
    a2 = mesafe(x1, y1, x3, y3);  
    a3 = mesafe(x2, y2, x3, y3);  
    p = (a1 + a2 + a3) / 2;  
    a = sqrt(p * (p - a1) * (p - a2) * (p - a3));  
    return a;  
}
```



GLOBAL Değişkenler

```
#include <stdio.h>
```

```
#include <math.h>
```

```
/* global variables */
```

```
double px1 = 0, py1 = 0; /* point (nokta) 1 */
```

```
double px2 = 7, py2 = 10; /* point (nokta) 2 */
```

```
double px3 = -5, py3 = 5; /* point (nokta) 3 */
```

```
/* Declarations */
```

```
double alan(void);
```

```
int main()
```

```
{
```

```
    double a = area();
```

```
    printf("%lf\n", a);
```

```
    return 0;
```

```
}
```

```
double alan()
```

```
{
```

```
    double a1, a2, a3, p, a;
```

```
    a1 = sqrt((px1 - px2) * (px1 - px2) + (py1 - py2) * (py1 - py2));
```

```
    a2 = sqrt((px1 - px3) * (px1 - px3) + (py1 - py3) * (py1 - py3));
```

```
    a3 = sqrt((px3 - px2) * (px3 - px2) + (py3 - py2) * (py3 - py2));
```

```
    p = (a1 + a2 + a3) / 2;
```

```
    a = sqrt(p * (p - a1) * (p - a2) * (p - a3));
```

```
    return a;
```

```
}
```

Static anahtar kelimesi

► Fonksiyonun içerisinde

- Static değişken değerini fonksiyon çağrılarında tutar.

```
#include <stdio.h>
void foo(){
    static int ne = 0;
    int n = 1;

    ne++;

    printf("n: %d, ne: %d \n",
           n, ne);
}

int main(){
    for (int i = 0; i < 10; i++)
        foo();
}
```

► Global değişken veya fonksiyon tanımlamalarında

- Sadece o dosya içinde erişilebilir.
- Farklı bir dosyadan fonksiyon çağrılmaz

```
#include <stdio.h>
```

```
static void foo(){
}
```

```
int main(){
    foo();
}
```

Örnek: Programın Çıktısı?

```
#include <stdio.h>
int a = 20;

int toplama(int a, int b) {
    printf ("%d\t", a);
    printf ("%d\t", b);
    a = a + 10;
    return a + b;
}

int main () {
    int a = 10;
    int b = 20;
    int c = toplama( a, b);

    printf ("%d\t", a);
    printf ("%d\t", c);
    return 0;
```



Alıştırmalar

- ▶ Feet cinsinden verilen bir uzunluğu metreye çevirip döndüren bir fonksiyon yazınız.
- ▶ Kartezyen koordinatları verilen bir noktanın polar koordinatlarını bulan bir fonksiyon yazınız.



Özet

- ▶ **Fonksiyonlar**

- ▶ Deklarasyon
- ▶ Tanımlama
- ▶ Fonksiyon çağrıları
- ▶ Fonksiyon parametreleri

Cuma gününe:

- ▶ **Diziler**

