

Bilgisayar Bilimlerine Giriş II

-2-

Dokuz Eylül Üniversitesi
Bilgisayar Bilimleri Bölümü

BİL 1012



Fonksiyonlar

- Gerçek hayattaki yazılım problemlerini çözen çoğu bilgisayar programları ve yazılımları, şu ana kadar öğrendiklerimizden çok daha geniş ve karmaşık bir yapıya sahiptir.
- Tecrübeler bu tür geniş programları yazmanın en iyi yolunun, küçük parçaları ya da her biri orijinal programdan daha kolay kullanılabilecek modülleri (daha önceden hazırlanmış program parçacıkları) birleştirmek olduğunu göstermiştir.
- Bu tekniğe, **böl ve yönet** (**divide & conquer**) denir.

C ve Fonksiyonlar

- ▶ C programları fonksiyonlardan oluşurlar.
- ▶ Şu ana dek kullandığımız **main()** de bir fonksiyondur. Bu fonksiyonun bir başka fonksiyon içinden çağrılmasına gerek yoktur.
- ▶ Her C programında bir **main()** fonksiyonun yer alması gerekmektedir.
- ▶ **main()** fonksiyonu, program çalıştırıldığında otomatik olarak çağrılan bir fonksiyondur.
- ▶ Bir **main()** fonksiyonu içinden bir başka fonksiyon çağrılabilir.

main() Fonksiyonu

- ▶ **main()** fonksiyonu da geri dönüş değeri kullanabilir.
- ▶ **main()** fonksiyonunun geri dönüş değerinin görevi, programın çalışması bittikten sonra sonucu işletim sistemine göndermektir.
- ▶ Program içinde *return* deyimi ile iletilen değer 0 olduğunda, bu işletim sistemi tarafından "*program başarılı olarak sonlandı*" olarak değerlendirir.
- ▶ Başka bir deyişle, *return 0;* program, kullanıcının talebi doğrultusunda (olumlu anlamda) "yapması gereken işi yaptı" mesajını işletim sistemine bildirilir.
- ▶ 0'dan farklı herhangi bir değer ise programın sorunlu sonlandığı anlamına gelecektir.

C ve Fonksiyonlar

- Bir fonksiyon içinden bir başka fonksiyon çağrılabilir.
- Örneğin, **fonksiyon1()** isimli fonksiyondan **fonksiyon2()** isimli bir başka fonksiyon çağrılabilir.



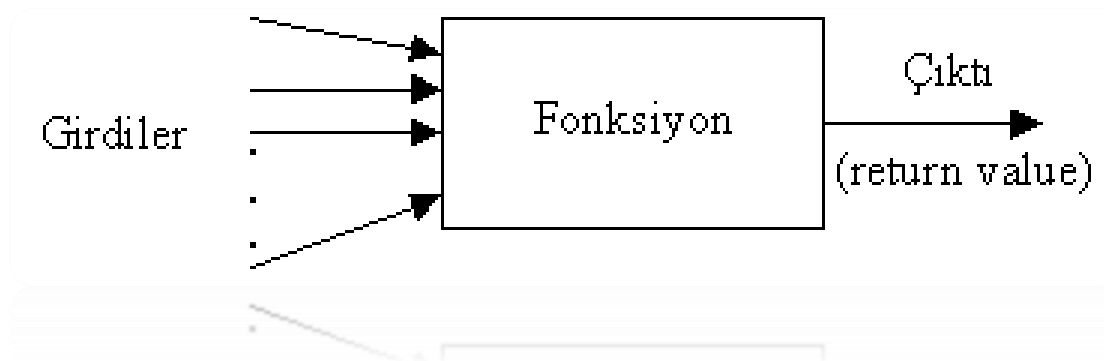
Fonksiyon Tanımı

- ▶ Fonksiyon, belirli sayıda verileri kullanarak bunları işleyen ve bir sonuç üreten komut grubudur.
- ▶ Her fonksiyonun bir **adı** ve fonksiyona gelen değerleri gösteren **parametreleri** (bağımsız değişkenleri) vardır.
- ▶ Bir fonksiyon bu parametreleri alıp çeşitli işlemlere tabi tutar ve bir değer hesaplar.

```
fonksiyonTipi fonksiyonAdı (arguman listesi)
{
    yerel bildirimler;
    ...
    (fonksiyon içindeki deyimler ve diğer fonksiyonlar)
    return değer;
}
```

Fonksiyon Tanımı

- Bu değer, çıktı veya geri dönüş değeri (**return value**) olarak adlandırılır.
- Bir fonksiyonun kaç girişi olursa olsun **sadece bir çıkışı** vardır.



Fonksiyon Geri Dönüş Değerleri

- ▶ Fonksiyon geri dönüş değeri **return** anahtar sözcüğü kullanılarak gerçekleştirilir.
- ▶ **return** anahtar sözcüğünün iki önemli işlevi vardır:
 - ▶ fonksiyonun geri dönüş değerini oluşturur
 - ▶ fonksiyonu sonlandırır
- ▶ **return** deyiminden sonra bir değişken, işlem, sabit veya başka bir fonksiyon yazılabilir.

Dikkat Edilmesi Gerekenler

- ▶ Fonksiyonun tipi ile **return** ifadesinin sağında yazılan değer aynı tipte olmalıdır.
- ▶ Eğer fonksiyon her hangi bir değer döndürmüyorsa, fonksiyonun tipi **void** olur.
- ▶ Bir fonksiyonun parametre alma zorunluluğu da yoktur.
- ▶ Fonksiyonlar normalde ana fonksiyonun (main) üzerinde tanımlanırlar. Yoksa derleyici bu fonksiyonlar çağırıldığında hata verir.
 - ▶ Ana fonksiyonun altında tanımlanan bir fonksiyonu derleyiciye tanıtarak düzgün derlemesini sağlamak için ana fonksiyonun üzerinde bir '**prototip**' tanımlanabilir.

Fonksiyon Tanımlama Örnekleri

| Örnek | Açıklama |
|---------------------------------|---|
| <code>int islem();</code> | Tam sayı değer dönen ve parametre içermeyen bir fonksiyon |
| <code>int islem(void);</code> | Tam sayı değer dönen ve parametre içermeyen bir fonksiyon |
| <code>int islem(int x);</code> | Tam sayı değer dönen ve tam sayı türünde parametre girdisi olan bir fonksiyon |
| <code>void islem();</code> | Değer dönmeyen ve parametre girdisi olmayan bir fonksiyon |
| <code>void islem(int x);</code> | Değer dönmeyen ve tam sayı türünde parametre girdisi olan bir fonksiyon |

Örnek: İki sayının toplamı

- Fonksiyon tipi: **int**
- Fonksiyon adı: **topla**
- Parametreler: **x ve y**
- Geri dönüş değeri: **x+y**



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

```
int topla(int x,int y)
{
    return x+y;
}
```

Fonksiyon bildiriminde,
fonksiyona girdi olarak,
kullanılan değişkenlere
Parametre denir.

```
int main()
{
    int sonuc;
    sonuc=topla(20,15);
    printf("Toplam sonucu=%d",sonuc);
    getch();
    return 0;
}
```

Fonksiyon çağılırken
gönderilen değerlere
Argüman denir.

► Yerel (local) Bildirim

```
int topla(int a,int b)
{

    int c; /* c yerel değişken*/
    c = a + b;
    return c;
}
```

Yerel değişkenler kullanıldığı fonksiyon içerisinde bildirilir. Yalnızca bildirildiği fonksiyon içerisinde tanınır ve kullanılabilir. Farklı blokların yerel değişkenleri aynı isimde olabilir.

► Genel (general) Bildirim

```
int m,n; /* m genel değişken*/
main()
{
    m=7;
}
```

Genel değişkenler bütün fonksiyonun dışında bildirilir. Program içindeki blok ve fonksiyonların hepsine aittir ve program çalıştığı sürece bellekte saklanır. Aynı isimde herhangi başka bir değişken bildirimi yapılamaz.

Değişken Bildirim Yerleri ve Türleri

► Dışsal (Extern) Bildirim

- Programlar parçalara ayrılıp ayrı ayrı derlenerek tekrar birleştirilebilirler.
- Böyle bir program yazma sürecinde, tüm program parçalarında aynı genel (global) değişkenler kullanılmak istenebilir.
- Bu durumda ana fonksiyonda normal bir genel değişken bildirimi yapılırken diğer programlarda *dışsal bildirim* yapılmalıdır.

| <i>Ana Parça</i> | <i>Diğer Parçalar</i> |
|--|--|
| <pre>... int x; main() { ... } fonk () { ... }</pre> | <pre>... extern int x; fonkA() { ... } fonkB() { ... }</pre> |

Değişken Bildirim Yerleri ve Türleri

► Statik (Static) Bildirim

- Bir fonksiyonun yerel değişkenleri için bellekte ayrılan yerler, fonksiyon sonlandığında boşaltılırlar. Bu da o yerel değişkenlerin değerlerinin kaybolmasına neden olur.
- Bir yerel değişkenin belleğe yerleştikten sonra programın tümü sonlanana kadar saklanması için *statik bildirim* yapılmalıdır.
- Böylece o fonksiyon tekrar çağırıldığında yerel değişkenler eski değerlerini koruyacaktır.
- Genel (global) değişkenler için statik bildirim yapıldığında ise o değişkenlerin başka bir program parçasında dışsal olarak tanımlanması engellenmiş olur.

Örnek:

```
int fibonacci()  
{  
    static int f0 = 0, f1 = 1; //static yerel değişkenler  
    int f2;  
  
    f2 = f0 + f1;  
    f0 = f1;  
    f1 = f2;  
  
    return f2;  
}
```

Bu fonksiyon her çağırıldığında bir sonraki fibonacci sayısını verecektir. Eğer yerel değişkenler statik bildirimle tanımlanmazlarsa, fonksiyon her zaman 1 değerini döndürür.

Fonksiyonları Çağırarak: Değere Göre ve Referansa Göre

► Değer ile Çağırma

- parametre olarak *değişkenin kendisi tanımlanır* ve fonksiyon çağırılırken ilgili parametreye bir *değer* gönderilir.

► Referans ile Çağırma

- parametre olarak bir *işaretçi değişken tanımlanır* ve fonksiyon çağırılırken ilgili parametreye bir *adres* gönderilir.

Not: Şimdilik sadece, değere göre çağırma üzerine yoğunlaşacağız.

void() Fonksiyonlar

- ▶ Bir fonksiyonun her zaman geri dönüş değerinin olması gerekmez. Bu durumda *return* deyimini kullanılmayabilir.
- ▶ Eğer bu anahtar kelime yoksa, fonksiyon ana bloğu bitince kendiliğinden sonlanır. Böyle fonksiyonların tipi void (boş, hükümsüz) olarak belirtilmelidir.
- ▶ Bu tip fonksiyonlar başka bir yerde kullanılırken, herhangi bir değişkene atanması söz konusu değildir, çünkü geri dönüş değeri yoktur.
- ▶ Ancak, void fonksiyonlara parametre aktarımı yapmak mümkündür.

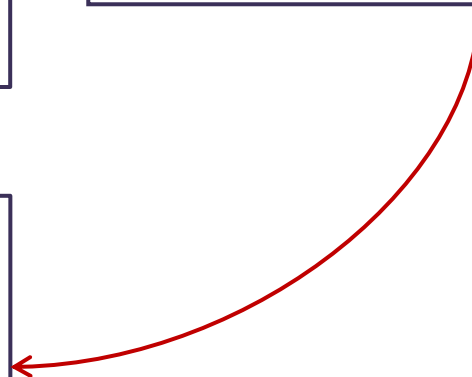
Örnek: İç içe birden fazla fonksiyon çağırımı

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

```
void mesaj2()
{
    printf("  Giris II");
}
```


```
main()
{
    printf("  Bilgisayar");
    mesaj();
    getch();
}
```

```
void mesaj()
{
    printf("  Bilimlerine");
    mesaj2();
}
```



Örnek: Klavyeden girilen iki sayıdan büyük olanını bulup main()'de yazdıran C programı yazınız.

```
#include <stdio.h>
#include <conio.h>
int max (int, int); // prototip
int main()
{
    int x, y;
    scanf("%d %d", &x, &y);
    printf("buyuk sayi = %d", max(x, y));
    getch();
    return 0;
}
```



```
int max (int a, int b)
{
    if (a > b) return a;
    else return b;
}
```

Sıklıkla Kullanılan Matematik Kütüphanesi Fonksiyonları - 1

| Fonksiyon | Tanım | Örnek |
|-------------------|--|--|
| sqrt(x) | x'in karekökü | sqrt(900.0) = 30.0 sqrt(9.0) = 3.0 |
| exp(x) | e^x üssel fonksiyonu | exp(1.0) = 2.718282 exp(2.0) = 7.389056 |
| log(x) | x'in e tabanına göre logaritması | log(2.718282) = 1.0 log(7.389056) = 2.0 |
| log10(x) | x'in 10 tabanına göre logaritması | log10(1.0) = 0.0 log10(10.0) = 1.0 log10(100.0) = 2.0 |
| fabs(x) | x'in mutlak değeri | fabs(5.0) = 5.0 fabs(0.0) = 0.0 fabs(-5.0) = 5.0 |
| ceil(x) | x kendinden büyük ilk tamsayıya yuvarlar | ceil(9.2) = 10.0 ceil(-9.8) = -9.0 |

Sıklıkla Kullanılan Matematik Kütüphanesi Fonksiyonları - 2

| Fonksiyon | Tanım | Örnek |
|---------------------|---|---|
| floor(x) | <i>x kendinden küçük ilk tamsayıya yuvarlar</i> | floor(9.2) = 9.0 floor(-9.8) = -10.0 |
| pow(x, y) | x^y | pow(2, 7) = 128.0 pow(9, .5) = 3.0 |
| fmod(x, y) | <i>x/y işleminin kalanını bulur</i> | fmod(13.657, 2.333) = 1.992 |
| sin(x) | <i>x'in sinüsünü hesaplar(x radyan)</i> | sin(0.0) = 0.0 |
| cos(x) | <i>x'in kosinüsünü hesaplar(x radyan)</i> | cos(0.0) = 1.0 |
| tan(x) | <i>x'in tanjantını hesaplar(x radyan)</i> | tan(0.0) = 0.0 |

Ön İşlemci Direktifleri

- ▶ Başlık dosyaları, derleyicinin kütüphane fonksiyonu çağrılarının doğru yapılıp yapılmadığını anlamasında yardımcı olan bilgiler içerir.
- ▶ ANSI C'deki standart başlık dosyaları aşağıdaki gibidir:

❖ assert.h

❖ ctype.h

❖ errno.h

❖ float.h

❖ limits.h

❖ locale.h

❖ math.h

❖ setjmp.h

❖ signal.h

❖ stdarg.h

❖ stddef.h

❖ stdio.h

❖ stdlib.h

❖ string.h

❖ time.h

Örnek: İki kenarı verilen bir dik üçgenin hipotenüsün hesaplayan fonksiyonu tanımlayınız. Fonksiyon **double** türünden iki argüman almalı ve hipotenüsü **double** türünden döndürmeli.

```
#include <stdio.h>
#include <math.h>
#include <conio.h>
double hipotenus(double s1, double s2)
{
    return sqrt( pow(s1,2)+pow(s2,2));
}
```

```
int main()
{
    double kenar1, kenar2;
    printf("Ucgenin kenarlarini giriniz: ");
    scanf("%lf %lf", &kenar1,&kenar2);
    printf("Hipotenus: %1f\n\n", hipotenus(kenar1,kenar2));
    return 0;
}
```


Makro Fonksiyon

- ▶ Başlık dosyaları içinde *define* önışlemcisi kullanılarak tanımlanmış olan fonksiyonlar makro fonksiyon olarak adlandırılır.
- ▶ Makro fonksiyonlar, gerçek anlamda altprogram benzeri fonksiyon değildir; belirli bir iş yapan program parçasına verilen simgesel adlardır.

```
#define kare(x)      ( (x) * (x) )  
#define buyuk(a,b)  ( (a) > (b) ) ? (a) : (b)  
#define dairealan(r) 3.14*kare(r)
```

Örnek:

```
#include<stdio.h>
#include<conio.h>
#define kare(x) ((x)*(x))
#define dairealan(r) 3.14*kare(r)
int main()
{
    float yc,x,alan;
    printf("YARICAPI GIRIN:");
    scanf("%f",&yc);

    alan=dairealan(yc);

    printf("\nDAIRENIN ALANI=%f\n",alan);
    getch();
    return 0;
}
```

} makro fonksiyonların tanımlanması

Dizilerin Fonksiyona Aktarılması

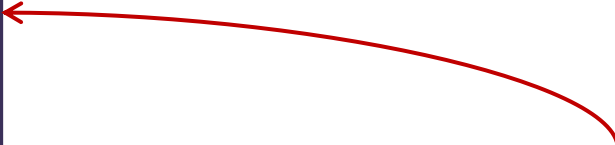
- Dizileri fonksiyona aktarmak için *dizinin adını* yazmak yeterlidir. Dizi kaç boyutlu olursa olsun, çağrılan bir fonksiyona gönderilirken *yalnızca adı* yazılır.
- Gönderilen dizinin boyut bilgisi, o fonksiyon tanımlanırken formal değişken bildirim kısmında verilir.

```
void yaz(float gram[], ... )  
int ver(int ayin_gunleri[][2], ... )  
float topla(int say[][4][6], ... )
```

Formal parametre bildirim kısmında değişkenlerin dizi olduğu köşeli parantezler ile belirtilir. Her boyut için ayrı ayrı aç ve kapa köşeli parantezleri kullanılır. Bir boyutlu diziler için eleman sayısı yazılmayabilir.

Örnek: Bir dizinin en büyük elemanının bulunması

```
#include <stdio.h>
#include <conio.h>
int enBuyuk(int a[], int n)
{
    int k, enb;
    enb = a[0];
    for(k=1; k<=n; k++)
        if(a[k]>enb)
            enb = a[k];
    return enb;
}
```



```
int main()
{
    int a[5] = { 100, -250, 400, 125, 300 };
    int eb;

    eb = enBuyuk(a,5);
    printf("En buyuk eleman = %d\n",eb);
    return 0;
}
```

Rekürsif Fonksiyonlar

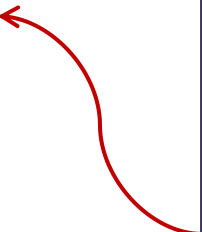
- ▶ Bazı problem tipleri için fonksiyonların kendi kendilerini çağırması kullanışlı olabilir.
- ▶ Bir yineleme fonksiyonu (**recursive function**), kendi kendini doğrudan ya da bir başka fonksiyon içinden çağırarak fonksiyondur.
- ▶ Yineleme fonksiyonu, **bir problemi çözmek için çağırılır**.
- ▶ Bu fonksiyon, yalnızca en basit durumu ya da temel durum olarak adlandırılan durumu nasıl çözeceğini bilmektedir.

Örnek: n değerine kadar olan sayıları toplayan recursive program yazınız.

```
#include<stdio.h>
#include<conio.h>
int toplam (int n)
{
    if (n==0) return 0;
    else return n+toplam(n-1);
}
```


```
int main()
{
    int n;
    printf("Hangi sayiya kadar:");
    scanf("%d",&n);

    printf("Sonuc=%d",toplam(n));
    getch();
    return 0;
}
```



Örnek: $n!$ özyinelemeli olarak hesaplanan C programı

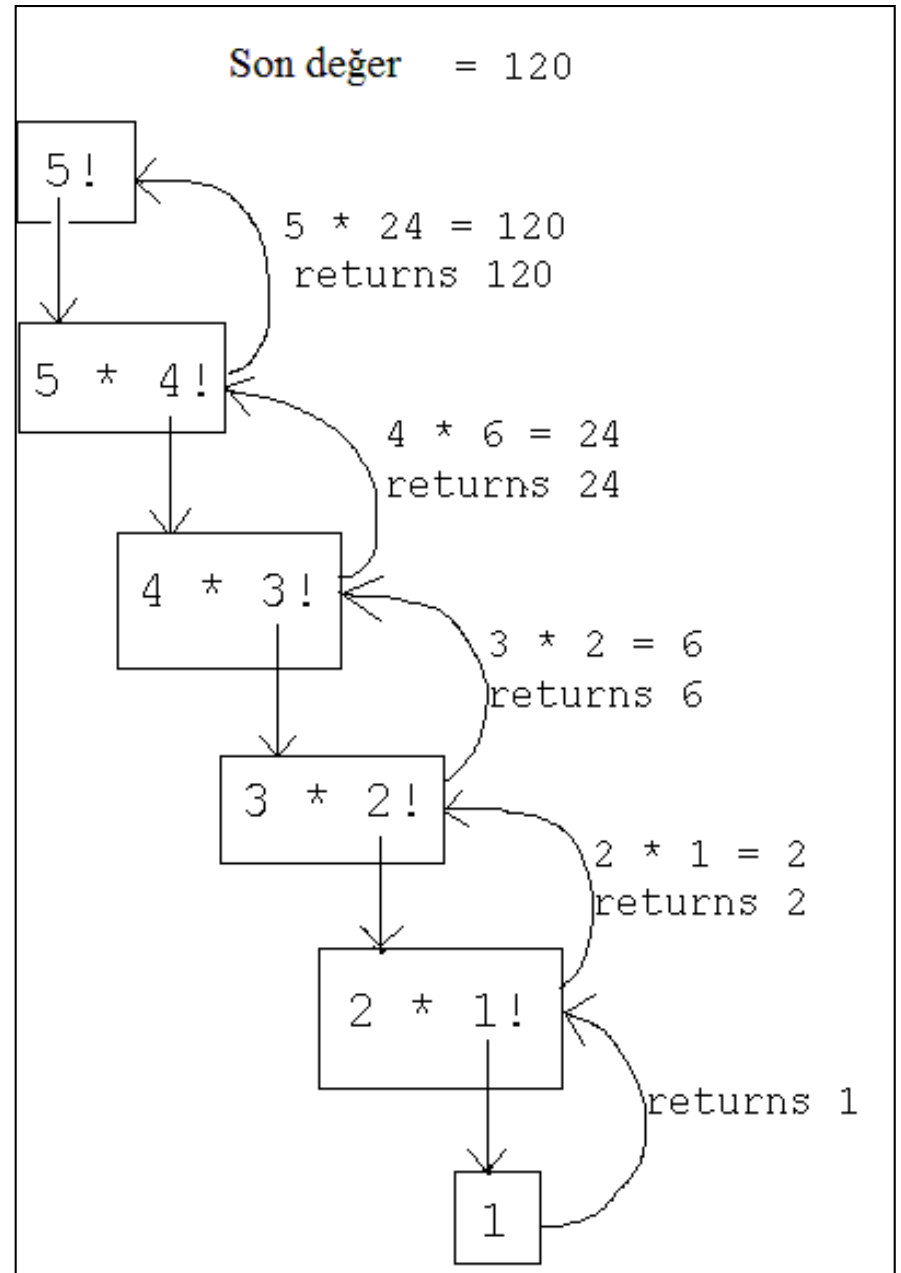
```
#include<stdio.h>
#include<conio.h>
int fakt (int sayi)
{
    if(sayi==0) return 1;
    else return sayi*fakt (sayi-1);
}
```



```
int main()
{
    int sayi;
    printf("sayiyi giriniz:");
    scanf("%d",&sayi);

    printf("Sonuc=%d",fakt (sayi));
    getch();
    return 0;
}
```

n! değerinin özyinelemeli
olarak bulunması
işlemi




ÖRNEKLER

Soru 1: a^b özyinelemeli olarak hesaplatan C programı yazınız.

```
#include<stdio.h>
#include<conio.h>
int us (int a,int b)
{
    if(b==0) return 1;
    else return a*us(a,b-1);
}
```

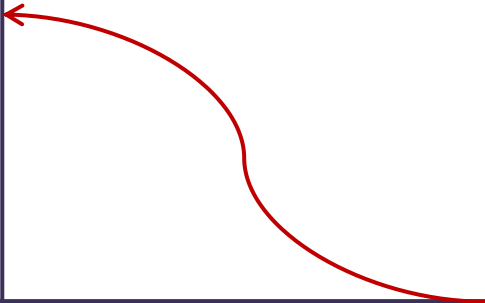
```
main()
{
    int a,b;
    printf("Sayi ve us degerini giriniz:");
    scanf("%d %d",&a,&b);

    printf("Sonuc=%d",us(a,b));
    getch();
}
```



Soru 2: Fibonacci dizisinin klavyeden girilen n sırada bulunan elemanını bulup ekranda yazdıran recursive C programı yazınız.

```
#include<stdio.h>
#include<conio.h>
int fibonacci (int n)
{
    if((n==1) || (n==2)) return 1;
    else return fibonacci(n-1)+fibonacci(n-2);
}
```



```
int main()
{
    int n;
    printf("kacinci sirada bulunan eleman:");
    scanf("%d",&n);
    printf("Sonuc=%d",fibonacci(n));
    return 0;
}
```

Soru 3

- ▶ Klavyeden girilen 5 (a, b, c) değeri için,
 - ▶ $a < b < c$ ise $M = (a+b)c/(b+4)$ işlemini **mat1** isimli fonksiyonda hesaplatan,
 - ▶ $b < a < c$ ise $M = (c-a)b/(a+1)$ işlemini **mat2** isimli fonksiyonda hesaplatan,
 - ▶ Diğer durumlarda $M = (a+b+c)/(2b+c)$ işlemini **mat3** isimli fonksiyonda hesaplatan
- ve sonucu main()'de yazdıran C programı yazınız.

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

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

```
float mat3(int a,int b,int c)
```

```
{
```

```
    float m;
```

```
    m=(float)(a+b+c)/(2*b+c);
```

```
    return m;
```

```
}
```

```
float mat2(int a,int b,int c)
```

```
{
```

```
    float m;
```

```
    m=(float)((c-a)*b)/(a+1);
```

```
    return m;
```

```
}
```

```
float mat1(int a,int b,int c)
```

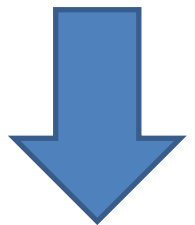
```
{
```

```
    float m;
```

```
    m=(float)((a+b)*c)/(b+4);
```

```
    return m;
```

```
}
```



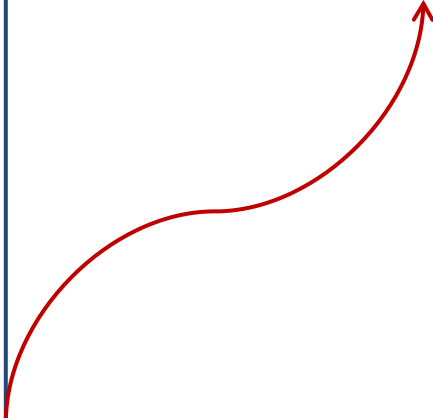
```
main()
{
    int a,b,c,i;
    float m;
    for(i=1;i<5;i++){

        printf("\na,b,c degerlerini giriniz:");
        scanf("%d %d %d",&a,&b,&c);

        if((a<b)&&(b<c))
        {
            m=mat1(a,b,c);
            printf("\nSonuc=%f",m);
        }

        else if((b<a)&&(a<c))
        {
            m=mat2(a,b,c);
            printf("\nSonuc=%f",m);
        }
    }
}
```

```
else {
    m=mat3(a,b,c);
    printf("\nSonuc=%f",m);
}
}
getch();
}
```



Soru 4

- ▶ **main()** fonksiyonunda verilen bir x değerini f isimli bir fonksiyona aktarıp,
 - ▶ $f(x)=3x-1$ değerini hesaplatan ve bu değeri g isimli bir fonksiyona gönderen,
 - ▶ g isimli fonksiyonda, $g(x)=x^2+2x-3$ değerini hesaplatan
 - ▶ sonucu **main()** fonksiyonunda yazdıran

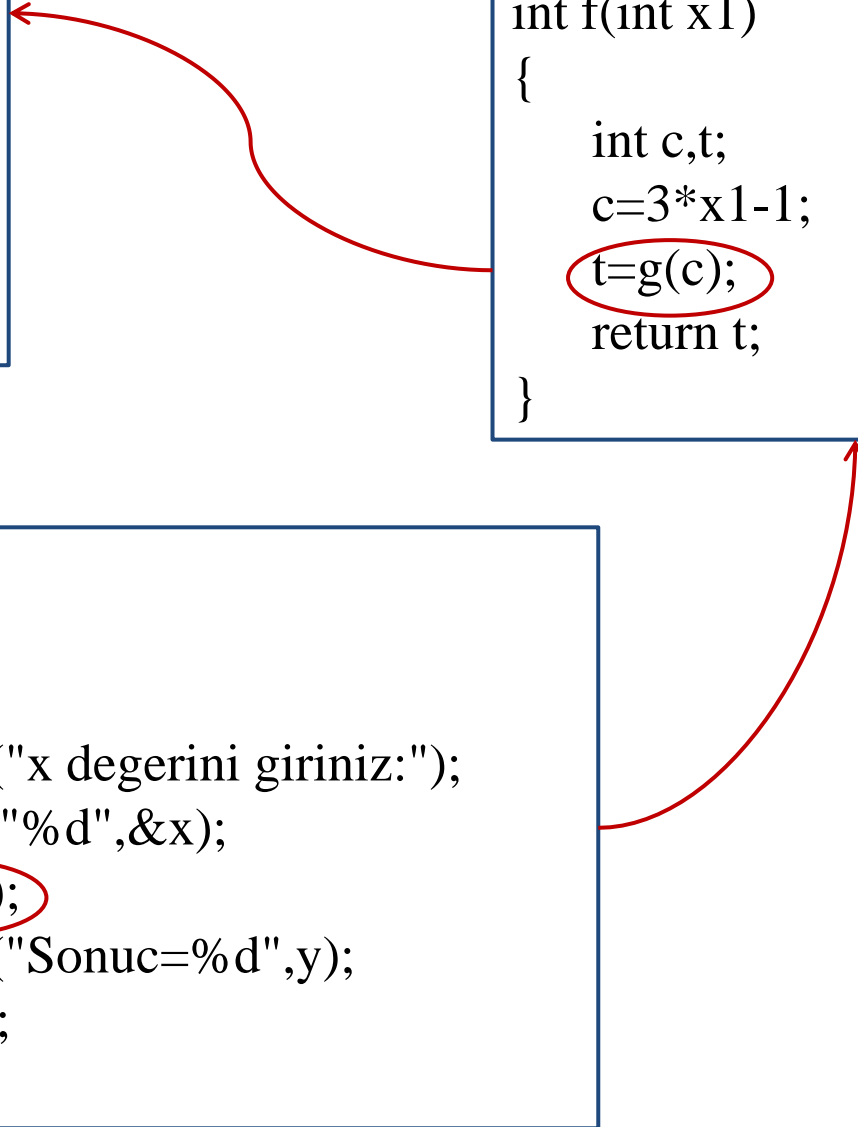
C programı yazınız.

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

```
int g(int c1)
{
    int z;
    z=c1*c1+2*c1-3;
    return z;
}
```

```
int f(int x1)
{
    int c,t;
    c=3*x1-1;
    t=g(c);
    return t;
}
```

```
main()
{
    int x,y;
    printf("x degerini giriniz:");
    scanf("%d",&x);
    y=f(x);
    printf("Sonuc=%d",y);
    return 0;
}
```

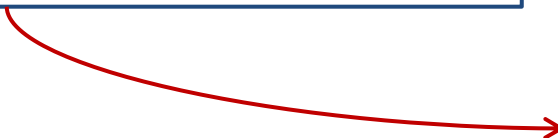


Soru 5

- ▶ Eğer bir sayının kendisi hariç, bütün çarpanlarının toplamı yine o sayıya eşitse bu sayıya **MÜKEMMEL SAYI** denir.
 - ▶ Örneğin, 6 bir mükemmel sayıdır. Çünkü $6 = 1 + 2 + 3$.
- ▶ Bir sayının mükemmel bir sayı olup olmadığını tespit eden bir fonksiyon yazınız.
- ▶ Bu fonksiyonu 1-1000 arasındaki tamsayılardan mükemmel olanlarını bulmak için bir program içinde kullanınız.

```
#include <stdio.h>
#include <conio.h>
int perfect( int value );
int main()
{
    int j;
    printf( "1 ile 1000 arasi tamsayilarda\n" );
    for ( j = 2; j <= 1000; j++ )
    {
        if ( perfect( j )==1 )
        {
            printf( "%d Mukkemeldir.\n", j );
        }
    } //for
    getch();
    return 0;
}
```

```
int perfect( int value )
{
    int factorSum = 1;
    int i;
    for ( i = 2; i <= value-1; i++ )
    {
        if ( value % i == 0 )
        {
            factorSum += i;
        }
    }
    if ( factorSum == value )
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
```



Soru 6 U01.cpp

- ▶ Girilen **10 adet** değerden oluşan diziyi:
 - ▶ *ortalama isimli* bir fonksiyona gönderip, orada ortalamalarını hesaplayan ve sonucu ana programda yazdıran,
 - ▶ Hesaplanan ortalamayı, *ortalama isimli fonksiyondan mükemmel isimli bir fonksiyona gönderip*, bu ortalama değerın tam kısmının bir mükemmel sayı olup olmadığını bulup, *sonucu ortalama isimli fonksiyonda* yazdıran bir program yazınız.

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

```
int mukemmel(float ort1)
```

```
{
```

```
    int i,c,top1=0;
```

```
    c=(int)ort1;
```

```
    for(i=1;i<c;i++)
```

```
    {
```

```
        if(c%i==0) top1+=i;
```

```
    }
```

```
    if(top1==c) return 1;
```

```
    else return 0;
```

```
}
```

mukemmel isimli fonksiyonun tanımlanması

ort değerinin tam kısmının alınması

ort değerinin mükemmel sayı olup-olmadığının test edilmesi

Mükemmel sayı ise **1**, değilse **0** değerinin döndürülmesi

```
float ortalama(int* y, int n)
```

```
{
```

```
    int top=0,i;
```

```
    float ort;
```

```
    for(i=0;i<n;i++)
```

```
        top=top+y[i];
```

```
    ort=(float)top/n;
```

```
    m=mukemmel(ort);
```

```
    if(m==1)
```

```
        printf("%d mukemmel sayi",(int)ort);
```

```
    else printf("%d mukemmel sayi degil",(int)ort);
```

```
    return ort;
```

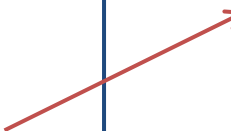
```
}
```

ortalama
isimli
fonksiyonun
tanımlanması


n adet sayının
ortalamanın
bulunması

mukemmel
isimli
fonksiyonun
çağırılması

```
int main()
{
    int a[10],i;
    float ort2;
    for(i=0;i<10;i++)
    {
        printf("\n a[%d] =>", i);
        scanf("%d",&a[i]);
    }
    ort2=ortalama(a,10);
    printf("\nortalama=%f",ort2);
    getch();
    return 0;
}
```



a,b,c, değerlerinin
klavyeden girilerek
okutulması



Y değerinin
hesaplanması,
ortalama
fonksiyonunun
çağırılması

Soru 7

- ▶ **floor** fonksiyonu bir ondalıklı sayıyı en yakın tamsayıya yuvarlar.
 - ▶ $y = \text{floor}(x + .5);$
- ▶ ifadesi x 'i en yakın tam sayıya yuvarlar ve y 'ye atar.
- ▶ Kullanıcıdan bir kaç sayı alan ve yukarıdaki ifadeyle bu sayıları yuvarlayan hem orijinal sayıyı hem de yuvarlanmış sayıyı ekrana yazdıran bir program yazınız.

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

```
void FloorHesapla( void);
```

```
main()
```

```
{
    FloorHesapla();
    getch();
}
```

```
void FloorHesapla( void )
```

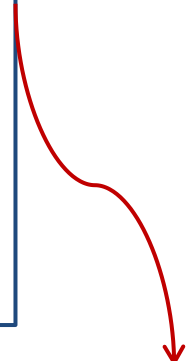
```
{
    double x;
    double y;
    int i;
    for(i=1;i<=5;i++){
        printf("Bir ondalikli sayi giriniz: ");
        scanf("%lf",&x);
        y=floor(x+.5);
        printf("%f sayisinin yuvarlanmisi: %.f\n",x,y);
    }
}
```

```
Bir ondalikli sayi giriniz: 10.2
10.200000 sayisinin yuvarlanmisi: 10
Bir ondalikli sayi giriniz: 9.3
9.300000 sayisinin yuvarlanmisi: 9
Bir ondalikli sayi giriniz: 7.7
7.700000 sayisinin yuvarlanmisi: 8
Bir ondalikli sayi giriniz: 15.4
15.400000 sayisinin yuvarlanmisi: 15
Bir ondalikli sayi giriniz: 16.7
16.700000 sayisinin yuvarlanmisi: 17
```

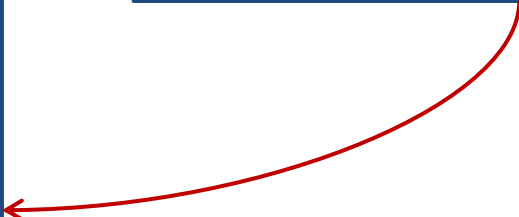

Soru 8

- ▶ Saati, üç argümanla (**saat, dakika ve saniye**) alan ve saat 12'den, girilen saate kadar olan zamanı saniye cinsinden hesaplayıp döndüren bir program yazınız.
- ▶ Daha sonra bu fonksiyonu kullanarak girilen iki saat arasındaki farkı 12'lik saat dilimine göre hesaplayan bir program yazınız.

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
unsigned saniye(unsigned h,unsigned m,unsigned s)
{
    return 3600 * h + 60 * m + s;
}
```



```
int main()
{
    int st,dk,sn,ilk,ikinci,fark;
    printf("Ilk (saat dak san) :");
    scanf("%d%d%d",&st,&dk,&sn);
    ilk=saniye(st,dk,sn);
```



```
    printf("Ikinci (saat dak san) :");
    scanf("%d%d%d",&st,&dk,&sn);
    ikinci=saniye(st,dk,sn);

    fark=fabs(ilk-ikinci);
    printf("Fark %d saniye\n",fark);
    getch();
    return 0;
}
```

Soru 9: Bir matrisin iz değerini bulup sonucu main()'de yazan bir fonksiyon tanımlayınız.

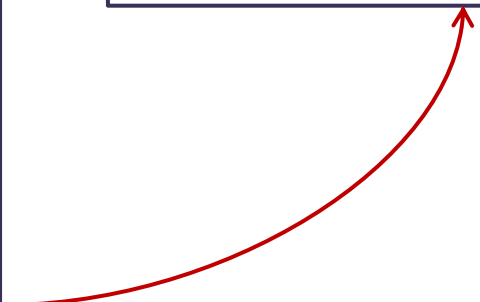
```
#include <stdio.h>
#include <conio.h>
double iz(double a[][3], int);
int main()
{
    double a[3][3], izA;
    int i,j;
    printf("matrisi girin:");
    for(i=0; i<3; i++)
        for(j=0; j<3; j++)
            scanf("%lf",&a[i][j]);

    izA = iz(a,3);
    printf("matrisin izi = %lf\n",izA);
    return 0;
}
```

```
double iz(double a[][3], int n)
{
    int i;
    double toplam = 0.0;

    for(i=0; i<n; i++)
        toplam += a[i][i];

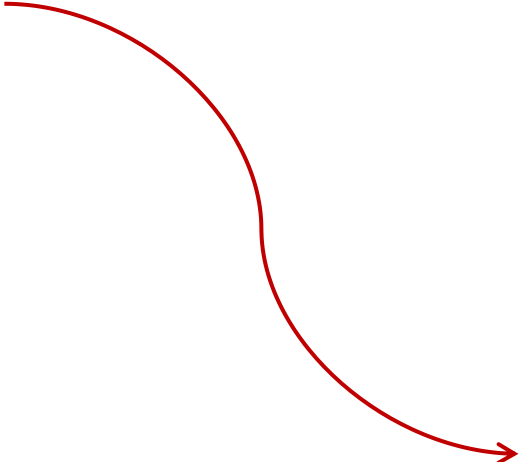
    return toplam;
}
```



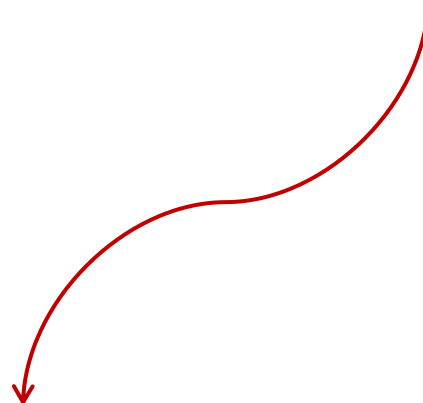
Soru 10: Şans Oyunu

- ▶ Oyuncu iki zarı aynı anda atar. İki zarında altı yüzü vardır. Bu yüzlerde 1,2,3,4,5 ve 6 adet nokta bulunur. Zarlar durduktan sonra her iki zarında üste gelen yüzleri toplanır.
- ▶ Eğer toplam ilk atışta 7 ya da 11 ise oyuncu *kazanır*.
- ▶ Eğer toplam ilk atışta 2,3 ya da 12 gelirse (buna barbut denir) oyuncu *kaybeder*.
- ▶ Eğer ilk atışta toplam 4,5,6,8,9,10 ise bu toplam oyuncunun sayısı haline gelir.
- ▶ Kazanmak için oyuncu sayısını bulana kadar zarları atmaya devam eder. Zarları atmaya devam ederken kendi sayısı yerine 7 atarsa kaybeder.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <conio.h>
int rollDice(void);
int main()
{
    int gameStatus, sum, myPoint;
    srand( time( NULL ) );
    sum = rollDice( );
```




```
switch( sum ) {
    case 7: case 11:
        gameStatus = 1;
        break;
    case 2: case 3: case 12:
        gameStatus = 2;
        break;
    default:
        gameStatus = 0;
        myPoint = sum;
        printf( "Oyuncunun kazanacagi zar: %d\n", myPoint );
        break;
}
```



```
while ( gameStatus == 0 ) {  
    sum = rollDice( );  
  
    if ( sum == myPoint )  
        gameStatus = 1;  
    else  
        if ( sum == 7 )  
            gameStatus = 2;  
}  
if ( gameStatus == 1 )  
    printf( "Oyuncu kazanadi\n" );  
else  
    printf( "Oyuncu kaybettti\n" );  
getch();  
return 0;  
}
```

int rollDice(void) //rollDice fonksiyonu
argüman almamaktadır.
{
 //Bu sebepten, fonksiyon
parametresi **void** kullanılmıştır.

```
int die1, die2, workSum;  
die1 = 1 + ( rand() % 6 );  
die2 = 1 + ( rand() % 6 );  
workSum = die1 + die2;  
printf( "Oyuncunun attigi zar: %d +  
%d = %d\n", die1, die2, workSum );  
return workSum;  
}
```



```
Oyuncunun attigi zar: 1 + 3 = 4  
Oyuncunun kazanacagi zar: 4  
Oyuncunun attigi zar: 4 + 4 = 8  
Oyuncunun attigi zar: 1 + 4 = 5  
Oyuncunun attigi zar: 5 + 3 = 8  
Oyuncunun attigi zar: 3 + 6 = 9  
Oyuncunun attigi zar: 2 + 2 = 4  
Oyuncu kazanadi
```

SON