

# 11- FONKSİYONLAR (FUNCTIONS)

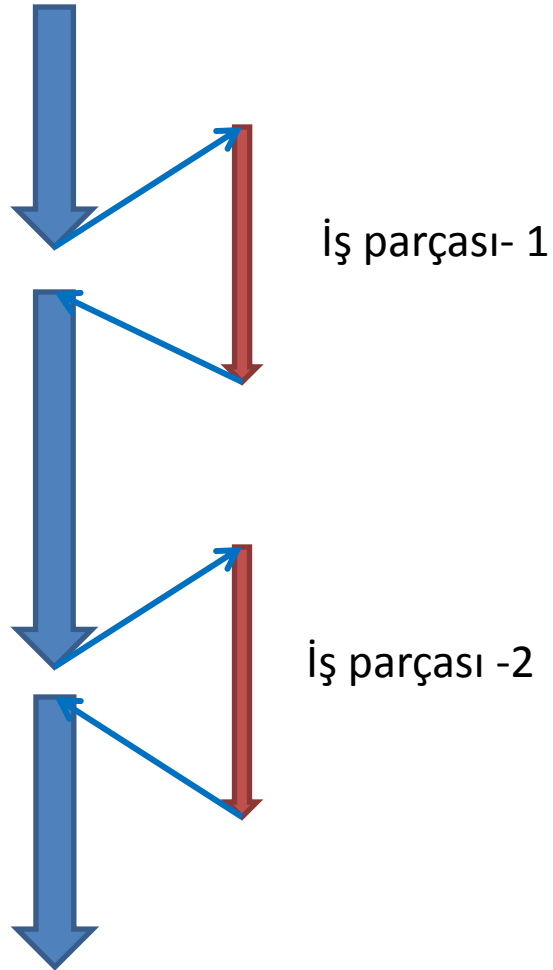
### Fonksiyon :

Belirli bir işi yapan programın bir isim altına tanımlanarak, o isimle çağrılarak kullanılması.

### Fonksiyonun faydaları :

- Programın modülerliğini arttırır.
- Aynı işi yapan program parçası tekrar tekrar yazılmaz.
- Fonksiyon farklı programlama dillerinde yazılabilir. Farklı dillerde kullanılabilir.
- Program iş parçalarını bölünerek birden çok kişi aynı program üzerinde çalışabilir.
- Fonksiyon program içerisinde gerektiği zaman kullanılır, gerekmedikçe kullanılmaz.

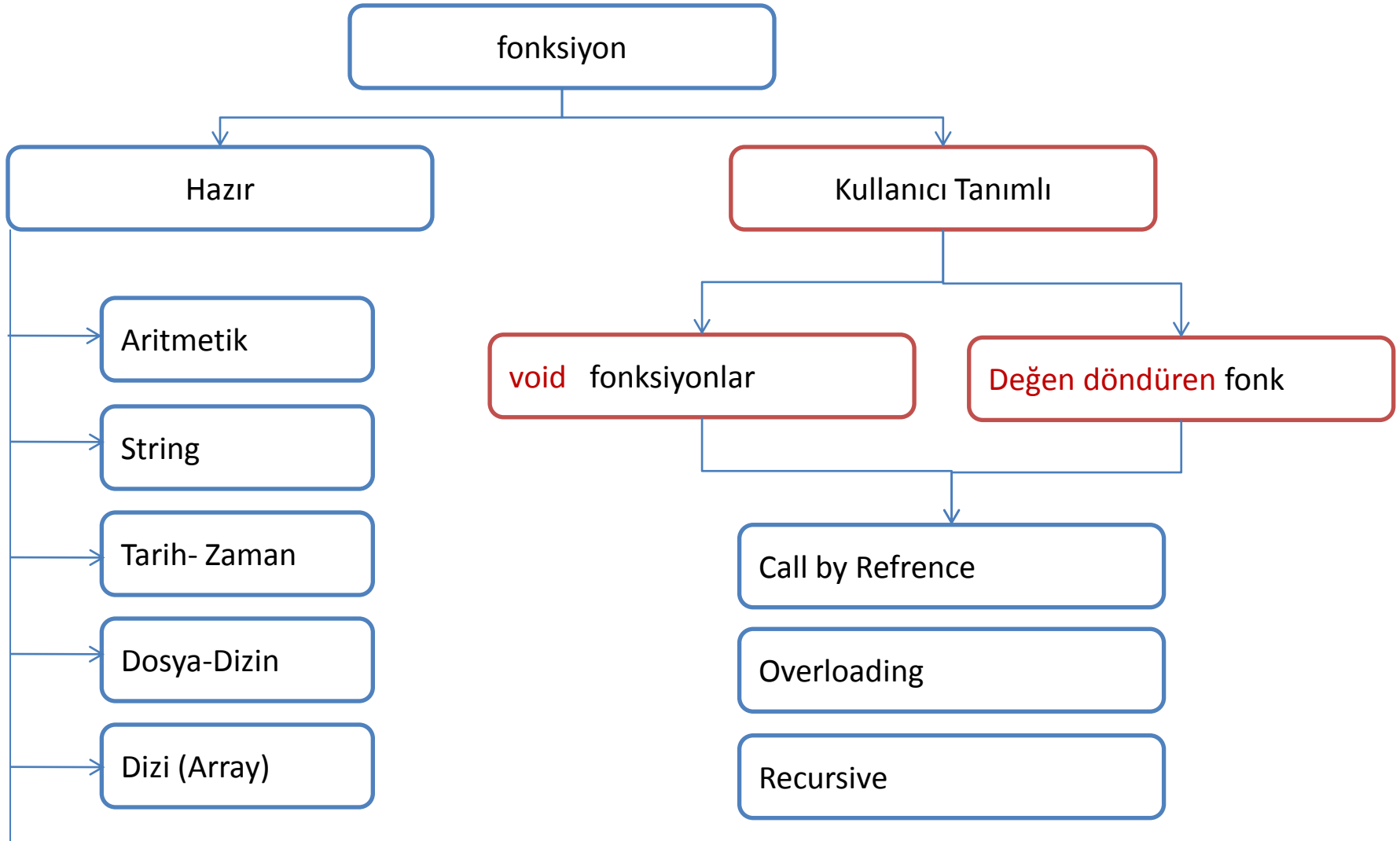
Main( ) Ana fonksiyon  
{



} Ana fonksiyon sonu

```
void Main ( )  
{  
    .....  
    İş parçası-1 ( ) ;  
    .....  
    İş parçası-2 ( ) ;  
    .....  
}  
  
void İş parçası-1 ( )  
{  
    .....  
}  
  
void İş parçası-2( )  
{  
    .....  
}
```

# 11- FONKSİYONLAR (FUNCTIONS)



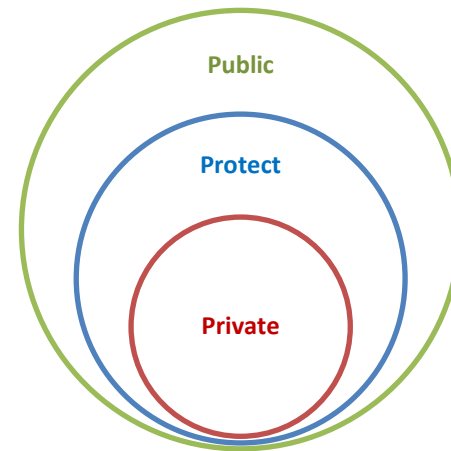
## 11- FONKSİYONLAR (FUNCTIONS)

Bir fonksiyonun genel yapısı :

```
Erişim_izni static veri_tipi fonksiyon_adi ( parametreler )  
{  
    .....  
}
```

} Fonksiyon gövdesi

Erişim izinleri : *public* , *protect* , *private*



**Not:** Erişim izini olarak hiçbir şey yazılmaz ise default(varsayılan) değeri *protect*

## 11- FONKSİYONLAR (FUNCTIONS)

Fonksiyon nereye yazılmalıdır ?

```
using System;  
namespace ORNEK  
{
```

```
    class Program  
    {
```

Fonksiyonlar, bu alana yazılabilir

```
        static void Main ( string[ ] args) // Temel (ana ) fonksiyon  
        {  
  
        }  
  
        Fonksiyonlar, bu alana da yazılabilir
```

```
    }
```

```
}
```

# **void** (tipi olmayan) fonksiyonlar

```
static void fonksiyon_adi ( parametreler)
{
    ...
    ...
    ...
}
```

## 11- FONKSİYONLAR (FUNCTIONS)

```
static void MesajYaz ( )  
{  
    Console.Write ( «Çıkmak için Bir tuşa basınız» );  
}
```

Kullanımı:

```
MesajYaz( );
```

```
static void MesajYaz ( string msg)  
{  
    Console.Write ( msg );  
}
```

Kullanımı:

```
MesajYaz( « merhaba»);
```

```
string s=«merhaba» ;  
MesajYaz( s );
```



## 11- FONKSİYONLAR (FUNCTIONS)

```
using System;
namespace ORNEK
{
    class Program
    {
        // kullanıcı tanımlı fonksiyon
        static void MesajYaz ( string msg)
        {
            Console.Write( msg );
        }

        // Temel (ana ) fonksiyon
        static void Main ( string[ ] args)
        {
            MesajYaz ( " Bugün Hava Çok Güzel ");
        }
    }
}
```

## 11- FONKSİYONLAR (FUNCTIONS)

### 10.3 - **void** tipli parametresiz fonksiyonlar.

namespace ORNEK

```
{
    class Program
    {
        static void topla ( int x , int y)
        {
            int z = x + y;
            Console.Write ( " Toplam : " + z );
        }

        static void Main ( string[ ] args)
        {
            topla ( 5, 7 ); // void fonksiyonun çağırılması

            int a= 4 ;
            int b= 7;
            topla ( a, b ); // void fonksiyonun çağırılması
        }
    }
}
```

## 11- FONKSİYONLAR (FUNCTIONS)

```
using System;
namespace ORNEK
{
    class Program
    {
        // kullanıcı tanımlı fonksiyon
        static void Sayac ( )
        {
            for( int x=1 ; x<= 5 ; x++)
                Console.WriteLine( x );
        }

        // Temel (ana ) fonksiyon
        static void Main ( string[ ] args)
        {
            Sayac( );
        }
    }
}
```

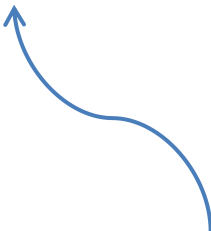
## 11- FONKSİYONLAR (FUNCTIONS)

```
using System;
namespace ORNEK
{
    class Program
    {
        // kullanıcı tanımlı fonksiyon
        static void Sayac ( int k )
        {
            for( int x=1 ; x<= k ; x++) Console.WriteLine( x );
        }

        // Temel (ana ) fonksiyon
        static void Main ( string[ ] args)
        {
            Sayac( 5 );
            int a=6;
            Sayac( a );
        }
    }
}
```

# Değer döndüren fonksiyonlar

```
static veritipi FonksiyonAdı ( parametreler)
{
    ...
    ...
    return (döndürelecek değer) ;
}
```



## 11- FONKSİYONLAR (FUNCTIONS)

### 10.3 - değer döndüren fonksiyonlar.

namespace ORNEK

```
{
    class Program
    {
        static int Ekle ( int x)
        {
            return ( x+6 );
        }

        static void Main(string[] args)
        {
            int sonuc = Ekle( 7 );    // 13
            Console.Write ( sonuc );  // 13
            Console.Write ( Ekle( 3 ) ); // 9
        }
    }
}
```

## 11- FONKSİYONLAR (FUNCTIONS)

### 10.3 - değer döndüren fonksiyonlar.

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

```
static void Main ( string[] args)
{
    int a= 5, b= 9;
    int sonuc = topla (3 , 5);
    Console.Write ( topla (a,b) );
    if ( topla (4,9) > 13 ) Console.Write ("A");
    Console.Write ( topla (3.15 , 9) ); // YANLIŞ KULLANIM parametre veri tipleri uyuşmuyor
    topla (7,9) ; // YANLIŞ KULLANIM hata vermez ama sonuç bir yerde kullanılmıyor
}
```

## 11- FONKSİYONLAR (FUNCTIONS)

### 10.5 - değer döndüren fonksiyonlar.

```
static int fakt ( int x )  
{  
    int f=1;  
    if( x==0 || x==1 ) return 1;  
    for ( int m=1 ; m<= x ; m++ ) f = f * m ;  
    return f;  
}
```

```
static void Main (string [] args)  
{  
    int a= 4;  
    int sonuc = fakt(a);  
    Console.Write ( fakt (5) );  
}
```



# Referans ile çağırma (call by reference)

```
FonksiyonAdi ( ref parametreler)  
{  
    ...  
    ...  
}
```

## 11- FONKSİYONLAR (FUNCTIONS)

### 10.6 - Referans ile çağırma ( **call by refrence** ) - **ref** parametre bildirimli

```
static void Ekle (ref int x)
{
    x = x + 1;
}

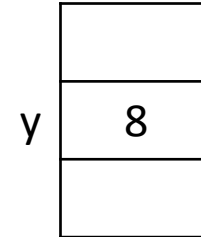
static void Main ( )
{
    int y = 8;
    Console.WriteLine( y ); // 8 yazar
    Ekle (ref y);           // y değişkeni fonksiyonda değişir
    Console.WriteLine ( y ); // 9 yazar
}
```

Çıktı:

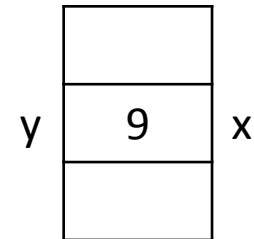
8  
9

```
Ekle ( y      );    //call by value
Ekle (ref y );    //call by reference
```

Önce



Sonra



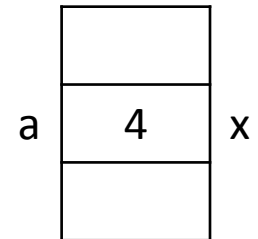
## 11- FONKSİYONLAR (FUNCTIONS)

### 10.6 - Referans ile çağırma ( **call by reference** ) - **ref** parametre bildirimli

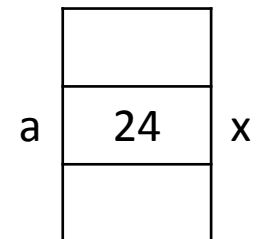
```
static int fakt ( ref int x)
{
    int z=1;
    if( x==0 || x==1 ) return 1;
    for ( int m=1 ; m<= x ; m++ ) z = z* m ;
    x = z;
    return z;
}

static void Main (string [] args)
{
    int a= 4;
    Console.Write ( a );      // a' nın değeri 4
    int sonuc = fakt (ref a) ;
    Console.Write ( a );      // a' nın değeri 24 olmuştur
}
```

Önce



Sonra



## 11- FONKSİYONLAR (FUNCTIONS)

### 10.6 - Referans ile çağırma ( **call by refrence** ) **ref &out** parametre bildirimli

**ref** : Başlangıç değeri gerektirir.

**Out** : Başlangıç değeri gerek yoktur.

---

```
static void FunctionOutRef(ref int c, out int d)
{
    d = c + 1;
    c = c + 1;
}
static void Main(string[] args)
{
    int x = 5; // başlangıç değeri var
    int y;      // başlangıç değeri yok
    FunctionOutRef (ref x, out y);
}
```

## 11- FONKSİYONLAR (FUNCTIONS)

### 10.6 - Referans ile çağırma ( **call by refrence** ) **out** parametre bildirimli

```
static int EkleBir( out int x)
{
    int y = 90;
    x = 19;           // out değişken
    x = x + 1;        // 20
    return (y + 1);   // 90+1
}

static void Main()
{
    int z;             // out değişken tanımlaması başlangıç değeri yok
    Console.WriteLine( EkleBir ( out z ) ); // return ile gelen değer 91
    Console.WriteLine( z );                 // out değişken yerine geçen z->x deki değer 20
    Console.Read();
}
```

**Çıktı:**

91

20

## 11- FONKSİYONLAR (FUNCTIONS)

### 10.6 - Referans ile çağırma ( **call by refrence** ) **ref &out** parametre bildirimli

Örnek:

```
void swap( ref int a, ref int b)
```

```
{
```

```
    int temp;
```

```
    temp = a;
```

```
    a = b;
```

```
    b = temp;
```

```
}
```

```
void Main()
```

```
{
```

```
    int a=5, b=8;
```

```
    swap(ref a, ref b );
```

```
    swap(ref a, ref 5 ); // HATA çünkü sabit değer verilmiş. Değişken olmalıdır.
```

```
}
```

# Overloading Functions

## Aşırı yüklenmiş Fonksiyon

### Overloading Functions ( Aşırı yüklenmiş Fonksiyon) :

İsimleri aynı, parametrelerinin sayısı veya veri tipi farklı olan fonksiyonlar

```
Console.Write ( );  
Console.Write( 2 );  
Console.Write( 3.14 );  
Console.Write(" MAKÜ meslek yüksekokulu " );  
Console.Write( 'A' );  
Console.Write(" Sonuç: " + (6+8) );
```



## 11- FONKSİYONLAR (FUNCTIONS)

10.7- **Overloading** – **Aşırı yüklenmiş Fonksiyon**: İsimleri aynı, parametrelerinin sayısı veya veri tipi farklı olan fonksiyonlar

```
static float carp ( int x , int y)
{
    return ( x * y);
}
```

```
static float carp ( int x , float y)
{
    return ( x * y);
}
```

```
static float carp ( float x , int y)
{
    return ( x * y);
}
```

```
static float carp ( float x , float y)
{
    return ( x * y);
}
```

```
Console.Write ( carp (4 , 6) );
Console.Write ( carp (3.5 , 2) );
Console.Write ( carp (4 , 2.5) );
Console.Write ( carp (2.5 , 3.14) );
```

## 10.7- **Overloading** – Aşırı yüklenmiş Fonksiyon: isimleri aynı parametreleri farklı olan fonksiyonlar

```
using System;
class Program
{
    static void F1(int x)    // Tek parametrelili
    {
        x = 12;
        Console.WriteLine(x + " int dir");
    }

    static void F1(int x, int y) //İki parametrelili
    {
        Console.WriteLine(x + " ve " + y + " int dir");
    }

    //temel fonksiyon
    static void Main()
    {
        int a = 12;
        int b = 12;
        int c = 12;
        F1 (a);      // tek parametrelili
        F1 (b, c);   // iki parametrelili
    }
}
```

### Rekursiv Functions

### Öz Yinelemeli Fonksiyon

(Kendi kendini çağıran fonksiyonlar)

## 10.8- Recursive (öz yinelemeli) Kendi kendini çağıran

```
using System;
class Program
{
    static void Yaz ( int x )
    {
        if (x > 5) return;           // Base şarta bağlı fonksiyondan çıkar
        else
        {
            Console.WriteLine ( x );
            Yaz (x + 1);              // Call myself (kendini çağırır)
        }
    }

    static void Main()
    {
        Yaz (1) ; // fonksiyonun çağırılması
        Console.ReadKey( );
    }
}
```

Çıktı:

1  
2  
3  
4  
5

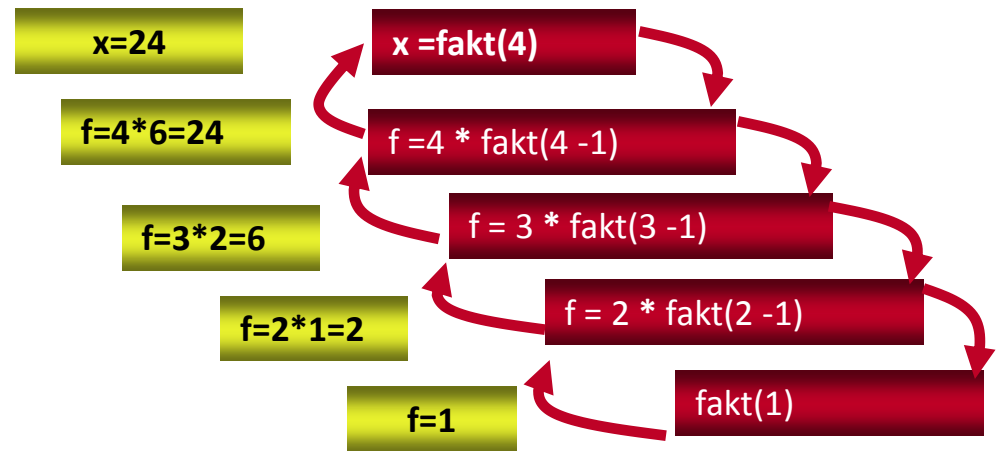
# 11- FONKSİYONLAR (FUNCTIONS)

## 10.8 - Recursive – Özyinelemeli Fonksiyon

```
static float fakt ( int x)
{
    float f = 1;
    if (x>1) f = x * fakt ( x - 1);

    return f ;
}

static void Main ( )
{
    float x = fakt( 4 );
    Consoler.Write (x);
}
```



# Global ve Local Variables

## Genel ve Yerel Değişkenler

**Global (Genel)** : Programın tümünde geçerli olan değişkenlerdir  
**Local(Yerel)** : Sadece tanımlandığı fonksiyon içerisinde geçerli olan değişkenlerdir.

## 11- FONKSİYONLAR (FUNCTIONS)

### 10.9 - Genel (Global) ve yerel (local) değişkenler.

```
using System;

class Program
{
    static int a = 2; // GENEL programın tümünde kullanılabilir

    static void test( )
    {
        int x=5;      // yerel sadece bu fonksiyon içerisinde geçerli
        Console.WriteLine(a);
    }

    static void Main()
    {
        int x=5; // yerel sadece bu fonksiyon içerisinde geçerli
        test();
        Console.Write (a);
        Console.Read();
    }
}
```

## 11- FONKSİYONLAR (FUNCTIONS)

### 10.9 - Genel (Global) ve yerel (local) değişkenler.

namespace ORNEK

```
{    class Program
    {
        static int g; // genel

        static int topla ( int x , int y)
        {   int z = 1; // yerel
            z = x+ y;
            g = z;
            return z;
        }

        static void Main(string[] args)
        {
            int a=4, b=6;           // yerel
            Console.Write ( topla ( a, b) );
            Console.Write ( g );
        }
    }
}
```

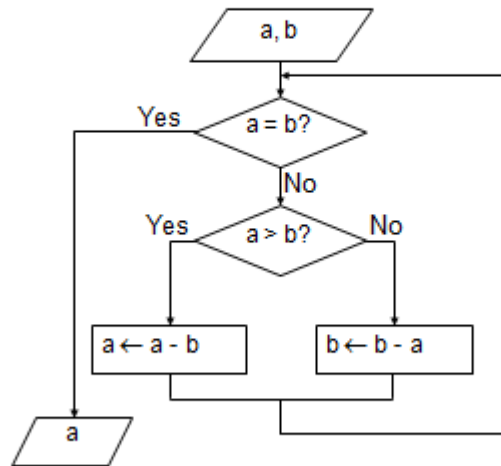


## Çeşitli Örnekler

## 11- FONKSİYONLAR (FUNCTIONS)

### Euclid's Algorithm (cont'd)

(continued) önceki sayfanın devamı demek)



### Euclid's Algorithm (cont'd)

- With iterations

```
public static int gcf (int a, int b)
{
    while (a != b) // a not equal to b
    {
        if (a > b)
            a -= b; // subtract b from a
        else
            b -= a; // subtract a from b
    }
    return a;
}
```

- With recursion

```
public static int gcf (int a, int b)
{
    if (a == b) // if a equals b
        return a;
    if (a > b)
        return gcf( a - b, b);
    else // if a < b
        return gcf(a, b - a);
}
```

Base case

## 11- FONKSİYONLAR (FUNCTIONS)

### İterasyon yöntemiyle

```
static float oklit ( int x, int y)
{
    while (x != y )
    {
        if(x>y) x = x-y;
        else y = y-x;
    }
    return x;
}

static void Main ( )
{
    float sonuc = oklit( 7,2 );
    Console.Write (oklit( 7,2) );
}
```

### recursive yöntemiyle

```
static float oklit ( int x, int y)
{
    if (x == y ) return x; // base
    if(x>y) return oklit (x-y , y);
    else return oklit (x , y-x);
}

static void Main ( )
{
    float sonuc = oklit ( 7 , 2 );
    Console.Write ( oklit ( 7 , 2 ) );
}
```

Sonuç: 1.0 dır

## 11- FONKSİYONLAR (FUNCTIONS)

### 10.10 – Parametresi **Dizi(array)** olan fonksiyonlar :

```
using System;
class Program
{
    static void DiziYaz(int[] d)
    {
        foreach (int tmp in d) Console.WriteLine(tmp);
    }

    static void Main()
    {
        int[] z = { 4, 7, 3, 6 };
        DiziYaz(z);
        Console.ReadKey();
    }
}
```

## 11- FONKSİYONLAR (FUNCTIONS)

### 10.10 – Parametresi **Dizi(array)** olan fonksiyonlar :

```
static double toplama( int [] d )
{
    double top = 0;
    for(int i = 0; i < d.Length; i++) top += d [ i ];

    return top ;
}

static void Main(string [] args)
{
    int [] x = {3,9,4,2,7};
    Console.Write( toplama( x ) );
}
```

## 10.10 – Parametresi **Dizi(array)** olan fonksiyonlar :

```
using System;
class Program
{
    static void DiziYaz( int[,] D )
    {
        for (int sat=0; sat<D.GetLength(0); sat++)
        {
            for(int sut=0; sut <D.GetLength(1) ; sut ++ )
            {
                Console.Write(" " + D[ sat, sut] );
            }
            Console.WriteLine();
        }
    }

    static void Main()
    {
        int[,] z = { {4, 5}, {2, 6}, {9, 3}, {7, 1} };
        DiziYaz(z);

        Console.Read();
    }
}
```

**Çıktı:**

4	5
2	6
9	3
7	1

## 11- FONKSİYONLAR (FUNCTIONS)

### 10.11 – Geri dönüş değeri **Dizi(Array)** olan fonksiyon :

```
static int [ ] dizi_olustur ( int adet )
{
    Random r = new Random();
    int [ ] d= new int [adet];
    for ( int i=0 ; i< adet ; i ++ )
        d [i] = r.Next(1,100);
    return d ;
}

static void Main(string[] args)
{
    int say = 5 );
    int [ ] D = dizi_olustur ( say ) ;
    foreach ( int k in D ) Console.Write ( " " + k );
    Console.ReadKey();
}
```

## 11- FONKSİYONLAR (FUNCTIONS)

Örn:  $y = 1 + x + x^2 + x^3 + \dots + x^n$  denkleminde klavyeden girilecek  $x$  ve  $n$  değeri için  $y$  değerini hesaplayıp yazdıran programı yazınız ?

```
static int usal ( int x, int y )
{
    int top =1;
    for ( int i=1 ; i<= y ; i ++ ) top *= x; ;
    return top ;
}

static void Main(string[] args)
{
    int x = int.Parse( Console.ReadLine() );
    int n = int.Parse( Console.ReadLine() );
    int y=0;
    for (int i = 0; i <= n; i++)
        y += usal (x, i);
    Console.WriteLine("\n x:{0} , n:{1} , y:{2}", x, n, y );
    Console.ReadKey();
}
```



## 11- FONKSİYONLAR (FUNCTIONS)

Örn:  $y = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \dots + \frac{x^n}{n!}$  klavyeden girilecek  $x$  ve  $n$  değeri için  $y$  değerini hesaplayıp yazdıran programı yazınız ?

```
static int fakt ( int x )
{
    int f =1;
    for ( int i=1 ; i<= x ; i ++ ) f *= i ;
    return f ;
}

static int usal ( int x, int y )
{
    int top =1;
    for ( int i=1 ; i<= y ; i ++ ) top *= x; ;
    return top ;
}

static void Main(string[] args)
{
    int x = int.Parse( Console.ReadLine() );
    int n = int.Parse( Console.ReadLine() );
    int y=0;
    for (int i = 0; i <= n; i++)
    {
        if( i%2== 0) y += (usal (x, i) / fakt(i) );
        else y -= (usal (x, i) / fakt(i) );
    }
    Console.Write("\n x:{0} , n:{1} , y:{2}", x, n, y );
}
```

## 11- FONKSİYONLAR (FUNCTIONS)

10.11 – Dizinin, bir fonksiyonun geri dönüş tipi olarak kullanılması :

```
static double [ ] sirala ( double [ ] d )
{
    for(int i = 0; i < d.Length-1; i++)
    {
        for(int k = i+1; k < d.Length; k++)
        {
            if(d[i] > d[k] )
            {
                int temp= d[i];
                d [i]= d[k];
                d[k]=temp;
            }
        }
    }
    t += d [ i ];
    return t ;
}

void Main (string [ ] s)
{
    int [ ] x = { 3,9,4,2,7 } ;
    yazdir ( x ) ;
    x= sirala ( x);
    yazdir ( x ) ;
}
```