C PROGRAMLAMA

Özyineleme (Recursion)

- Bir fonksiyonun kendisini çağırarak çözüme gitmesine özyineleme (recursion), böyle çalışan fonksiyonlara da özyinelemeli (recursive) fonksiyonlar denilir.
- Özyineleme, iterasyon (döngüler, tekrar) yerine geçebilecek çok güçlü bir programlama tekniğidir.
- Özyinelemeli algoritmalarda, tekrarlar fonksiyonun kendi kendisini kopyalayarak çağırması ile elde edilir. Bu kopyalar işlerini bitirdikçe kaybolur.
- Bir problemi özyineleme ile çözmek için problem iki ana parçaya ayrılır.
  - 1 – cevabı kesin olarak bildiğimiz kesin durum (base case)
  - 2 – cevabı bilinmeyen ancak cevabı yine problemin kendisi kullanılarak bulunabilecek durum.

Recursive fonksiyon - Faktöriyel hesabı

- Bu yönteme en uygun örnek faktöriyel problemidir.
- Faktöriyelin tanımı şöyle yapılır; sıfır dışındaki herhangi bir doğal sayının faktöriyeli 1 den başlayarak, o sayıya kadarki tüm doğal sayıların çarpımını almak yolu ile elde edilir. 0 sayısının faktöriyeli 1 olarak tanımlanır.

Matematiksel gösterimi ise: $N!=1 \times 2 \times 3 \times \ldots\ldots \times (N-1) \times N$

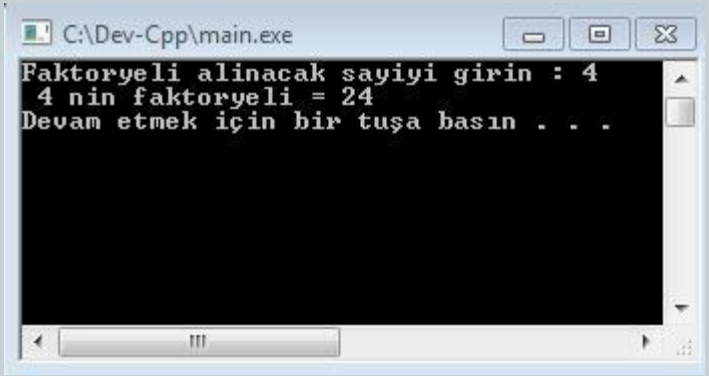
$0! = 1$

- Çarpım dizisindeki sonuncu terim olan N'i ayırdığınızda diğer sayıların çarpımı (N-1)! Olmaktadır. $N! = N \times (N-1)!$
- Yukarıdaki tanım ile son yapılan tanım arasındaki fark, açıklanan problemin tanımında yine problemin kendisinin olmasıdır.
- Burada cevabı kesin olarak bilinen 0! durumu temel durum olacaktır. Diğer durumlar ise cevabı bilinmeyen ancak problemin kendisi kullanılarak bulunabilecek durumlardır.

Recursive fonksiyon - Faktöriyel hesabı

- Şuana kadar anlatılan yöntemler ile faktöriyel hesabı aşağıdaki gibi yapılır.

```c
/* Şuana kadar anlatılan yöntemler ile
   faktöriyel hesabı yapan program
*/

#include <stdio.h>

int faktoryel(int n) {
      int i, f=1;
      for(i=1;i<=n;i++)
          f=f*i;
      return f;
}

int main()
{
     int x, fs;
     printf("Faktoryeli alinacak sayiyi girin : ");
     scanf("%d",&x);
     fs = faktoryel(x);
     printf(" %d nin faktoryeli = %d\n", x, fs);

   system("PAUSE");
   return 0;
}
```



Recursive fonksiyon - Faktöriyel hesabı

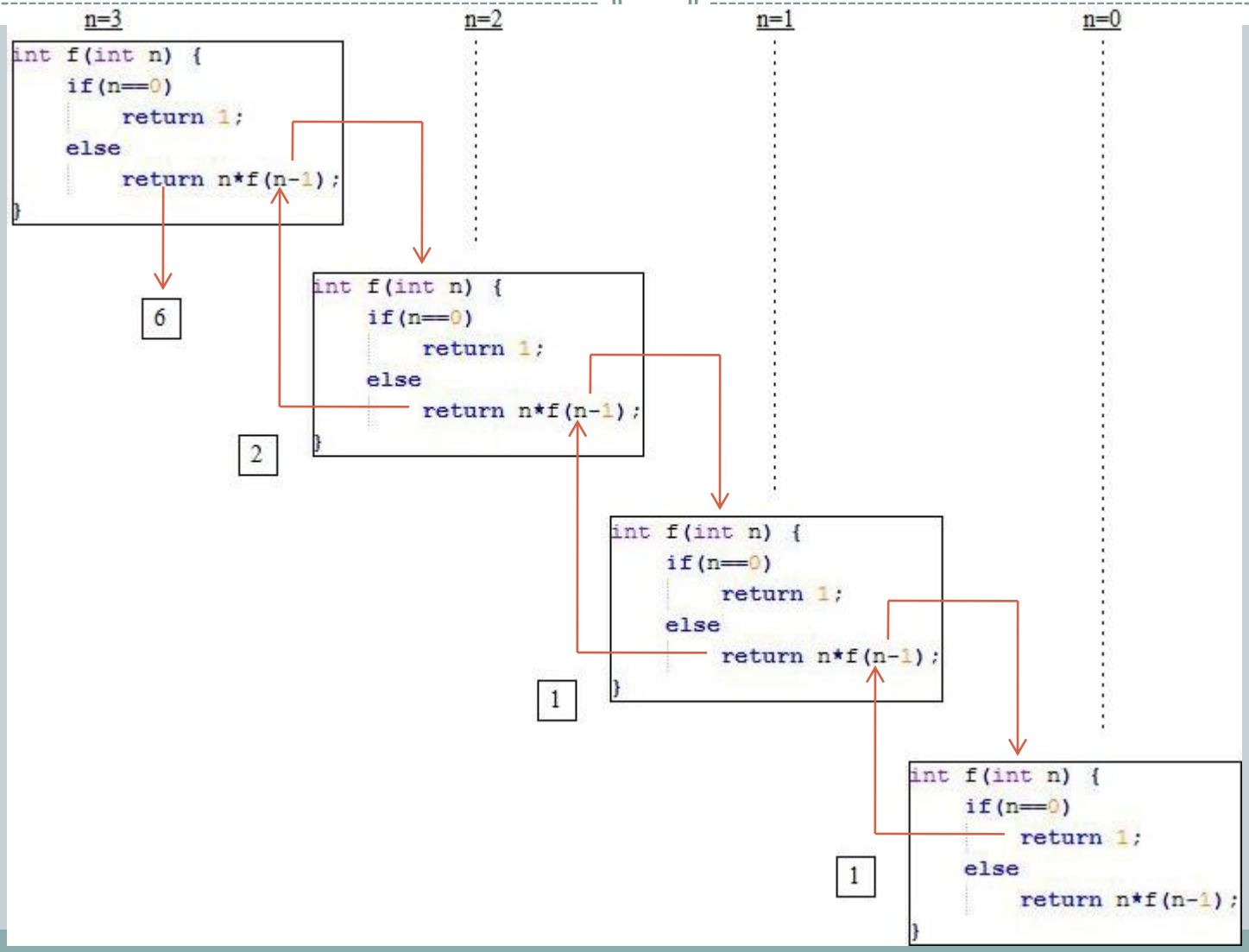
- Faktöriyel hesabı özyinelemeli fonksiyon ile aşağıdaki mantıkla çalışır.

5! = 5 x 4! Bunun cevabini bilmek icin 4 faktöriyeli bulmak gerekir
4! = 4 x 3! Bunun cevabini bilmek icin 3 faktöriyeli bulmak gerekir
3! = 3 x 2! Bunun cevabini bilmek icin 2 faktöriyeli bulmak gerekir
2! = 2 x 1! Bunun cevabini bilmek icin 1 faktöriyeli bulmak gerekir
1! = 1 x 0! Bunun cevabini bilmek icin 0 faktöriyeli bulmak gerekir
0! = 1 bu parcanin degeri tanim geregi 1 dir.

- Bu durumda faktöriyel hesabı yapan fonksiyonu recursive (özyinelemeli) olarak tekrar düşündüğümüzde aşağıdaki gibi olacaktır.

```
int faktoryel(int n) {

    if(n==0)
        return 1;
    else
        return n*faktoryel(n-1);
}
```

Recursive fonksiyon - Faktöriyel hesabı



DİZİLER (ARRAYS)

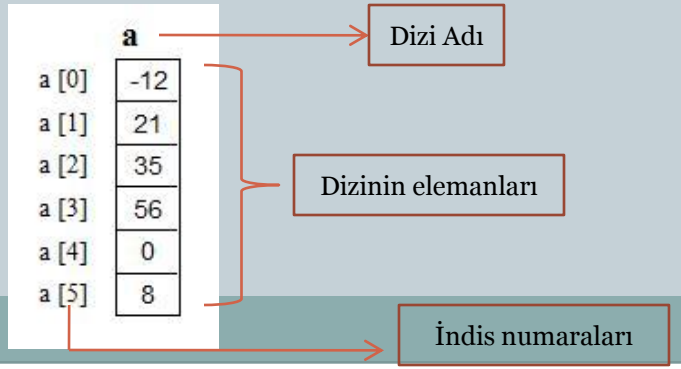
- Aynı isim altında, aynı türde birden fazla veriyi tutmak için kullanılan veri yapılarıdır (Data Structure).
- Diziler bir kümedir, aynı türde verilere tek bir isimle erişmek için kullanılır.
- Bir dizinin bütün elemanları bellekte ardışık olarak saklanır.
- Diziler bir veya daha çok boyutlu olabilirler.

DİZİLER (ARRAYS)

- Tek boyutlu diziler;
  - tek boyutlu bir dizi tanımlaması şu şekilde gerçekleştirilir :

```
dizinintürü dizininadı [elemansayısı];
int a[6];        // 6 elemanlı, her elemanı int olan dizi.
char b[15];      // 15 elemanlı, her elemanı char olan dizi.
float c[10];     // 10 elemanlı, her elemanı float olan dizi.
double d[10];    // 10 elemanlı, her elemanı double olan dizi.
```

  - Derleyici ilk örnekteki gibi bir komutla karşılaştığında 6 adet sayının saklanacağı (tutulacağı) bellek alanı ayırır. Bir tam sayı örneğin DOS işletim sisteminde 2 byte uzunluğunda ise ‘a’ dizisinin bellekte kaplayacağı toplam alan 2x6=12 byte olacaktır.
  - Dizinin adı yani ‘a’ değişkeni bu 12 byte’lık alanın başlangıç adresini tutar.
  - Dizi içerisindeki her bir elemana, dizi isminden sonra, yazılan pozisyon numarası yani indis değeri ile ulaşılır. İndis değerleri mutlaka tam sayı olmalıdır. C dilinde bütün dizilerin indis numarası 0’dan başlar.
  - Dolayısı ile dizinin son indis numarası dizinin eleman sayısından bir eksiktir.



DİZİLER (ARRAYS)

- Tek boyutlu dizilere ilk değer atama;

```
int a[5]={4,5,3,1,8};
```

a	
a [0]	4
a [1]	5
a [2]	3
a [3]	1
a [4]	8

```
int a[5]={0};
```

a	
a [0]	0
a [1]	0
a [2]	0
a [3]	0
a [4]	0

```
int a[5];
```

a	
a [0]	Rastgele
a [1]	Rastgele
a [2]	Rastgele
a [3]	Rastgele
a [4]	Rastgele

```
int a[]={4,5,3};
```

a	
a [0]	4
a [1]	5
a [2]	3

```
// döngü yardımıyla da bir dizinin elemanlarına değer atanabilir.

int i, a[6];

for(i=0;i<6;i++)
    a[i]=0; // dizinin elemanlarına 0 değerini atar.
```

DİZİLER (ARRAYS)

- Dizilerin fonksiyonlara parametre olarak gönderilmesi;
  - Bir dizi bir fonksiyona parametre olarak gönderilmek istendiğinde, fonksiyon çağrılırken dizinin sadece adı yazılır.
  - Fonksiyon başlığında diziyi karşılayan tanımlamada ise köşeli parantezler kullanılır.

```
/* kendisine gönderilen tam sayı bir diziyi
   ekrana yazdıran program
*/

#include <stdio.h>

void diziYaz(int [], int); // fonksiyon prototipi

int main()
{
    int a[5]={13,12,0,5,8};
    diziYaz(a, 5);

  system("PAUSE");
  return 0;
}

void diziYaz(int b[], int n){
    int i;
    for(i=0;i<n;i++) // döngü yardımıyla dizinin tüm elemanlarına ulaşılır.
        printf("b[%d] = %d\n",i, b[i]);
}
```

DİZİLER (ARRAYS)

- Dizilerin fonksiyonlara parametre olarak gönderilmesi;
  - Diziler fonksiyonlara parametre olarak gönderilirken bağlantılı çağırma (call by reference) yöntemiyle gönderilirler.
  - Bu yöntemde değerlerin kopyalandığı değişkenlerin değiştirilmesi orijinal değerleri etkiler.
  - Daha öncede söylediğimiz gibi dizi isminde dizinin bellekteki başlangıç adresi tutulur.
  - Bir önceki örnekte tanımladığımız a dizisinde a değişkeni dizinin bellekteki başlangıç adresini tutacaktır. İsterseniz a değişkeninin değerini (yani dizinin başlangıç adresini) aşağıdaki komutla ekrana yazdırabilirsiniz.
    - Printf("%p",a);
  - O halde fonksiyona dizi ismi ile gönderilen değer dizinin başlangıç adresidir. Fonksiyon parametresinde tanımlanan ve bu a ya karşılık gelecek olan değişkende (bir önceki örnekte b değişkeni) aynı bellek adresinden başlamış olur.
  - Bellekte aynı adrese sahip iki farklı hücre bulunamaz. Yani a ve b dizileri bellekte üst üste saklanan aslında aynı dizilerdir.
  - Birinde yapılan değişiklik aynı adresler doğrultusunda diğerini etkiler.

DİZİLER (ARRAYS)

- Çok boyutlu diziler (Multiply subscripted arrays);
  - Birden fazla indis numarası ile elemanlarına ulaşılan dizilere çok boyutlu diziler denir.
  - Bunlardan en sık kullanılanı çift boyutlu dizilerdir (double subscripted arrays).
  - Bu diziler sayesinde değerler satır ve sütunlardan oluşan (row - column) iki boyutlu bir tablodaki veriler gibi düşünebilirler (**bellekte yine ardışık olarak tutulurlar**).
  - Bu tür dizilerde her bir elemana ulaşmak için elemanın satır indisi ve sütun indisi kullanılır.
  - C dilindeki dizilerin boyutları ikiden de fazla olabilir. Bir sınır olmamakla beraber 12 boyuta kadar destekler. Ancak iki boyutlu diziler bizim için yeterli.

İki boyutlu bir dizini temsili gösterimi

	Sutun 0	Sutun 1	Sutun 2	Sutun 3
Satir 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Satir 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Satir 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

DİZİLER (ARRAYS)

- Çift (iki) boyutlu dizilere ilk değer atama;
  - Tek boyutlu dizi tanımlamaya benzer.

```
int a[3][4]={{4,5,3,1},{0,4,3,1},{1,9,7,2}};
 // altarnatif olarak aşağıdaki gibide olabilir.
int a[3][4]={{4,5,3,1},
             {0,4,3,1},
             {1,9,7,2}};
```

a [3][4]

4	5	3	1
0	4	3	1
1	9	7	2

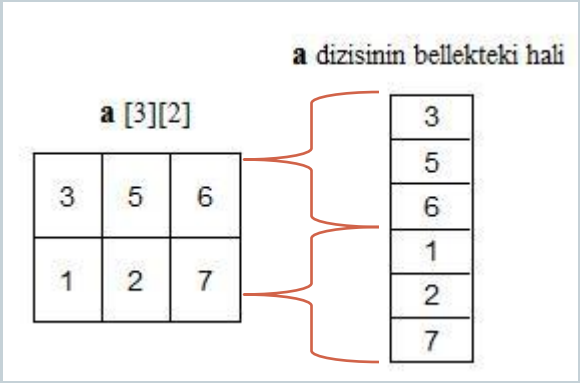
```
/* budurmda derleyici eksik kalan
değer için 0 değerini atayacaktır.*/

int a[2][3]={1,2,3,4,5};
```

a [2][3]

1	2	3
4	5	0

DİZİLER (ARRAYS)

- Çok boyutlu dizilerin fonksiyona parametre olarak gönderilmesi;
  - Çok boyutlu bir dizi fonksiyona gönderilecekse, tek boyutlularda olduğu gibi sadece adı yazılır.
  - Gönderilen dizi değişkenini karşılayan tanımlamada ise gönderilen dizinin boyutu kadar köşeli parantez (aç-kapa ikilisi) kullanılır.
  - Çift boyutlu diziler tablo halinde bellekte tutulmaz, örneğin [2][3] boyutlarında bir dizinin elemanları bellekte 6 elemanlı tek boyutlu bir diziymiş gibi tutulur.



Çift boyutlu dizide her bir satır tek boyutlu bir diziymiş gibi düşünülür ve bunlar uç uca eklenerek belleğe yerleştirilir.

Derleyicinin bellekte bir sonraki satırın başlangıçtan kaç eleman sonra başladığını bilmesi gerekir

↓

  - İlk köşeli parantez içerisine eleman sayısını ifade eden değer yazılmaz., fakat diğerlerine eleman sayıları yazılmak zorundadır.

DİZİLER (ARRAYS)

- Örnek;

```c
#include <stdio.h>

void diziYaz(int [][4]); // derleyici bellekte her 4 elemandan sonra
                         // temsili olarak yeni bir satıra gecileceğini anlıyor.

int main()
{

  int a[3][4]={{4,5,3,1},
               {0,4,3,1}, // 3 satır 4 sütundan oluşan 12 elemanlı bir dizi tanımalandı
               {1,9,7,2}}; // bu dizinin elemanları integer (tam sayı) tipindedir.

  diziYaz(a);

  system("PAUSE");
  return 0;
}

void diziYaz(int b[][4]){
     int sutun, satir;

     for(satir=0;satir<3;satir++){      // çift boyutlu dizilerin elemanlarına ulaşmak için
                                        //iç içe iki for döngüsü kullanılır.

          printf("%d. satirdaki elemanlar: ", satir);
          for(sutun=0;sutun<4;sutun++){
               printf("%d ",b[satir][sutun]);
          }
          printf("\n"); // bir alt satıra geç.
     }
}
```