Diziler

Dizi Tanımı

Dizi Elemanlarına Değer Atama

Diziler ve Göstergeler

2-Boyutlu Diziler

Dizi Tanımı

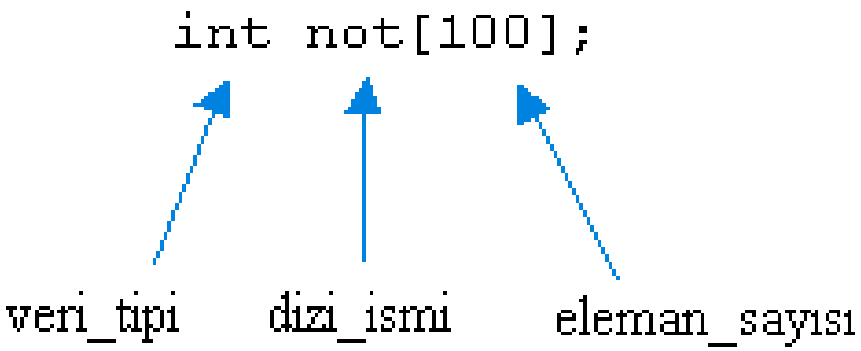
Değişkenler aynı anda tek bir değer tutabilen ***temel değişkenler*** ve birden fazla değer saklayabilen ***bileşik değişkenler*** olmak üzere ikiye ayrılır.

Temel değişkenler bellekte tek bir hücreyi tanımlayıp, içlerinde tek bir değeri tutabilirler.

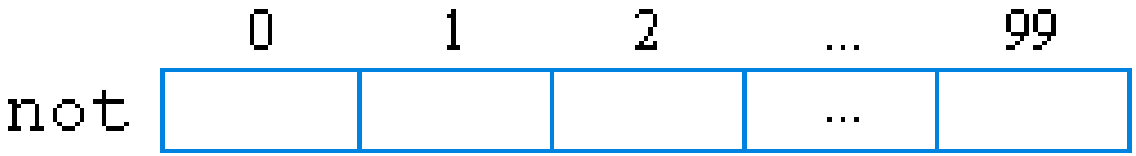
Diziler ise ardarda sıralanmış bellek hücreleridirler. Diziler bu bağlamda ***bileşik değişkenlerdir*** ve bellekte aynı anda birden fazla değerin saklamasını mümkün kılarlar.

Dizi Tanımı

veri_tipi dizi_ismi [*eleman_sayısı*];

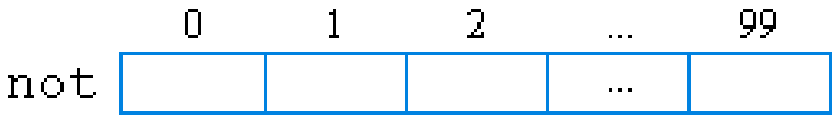


Bellek Görüntüsü:

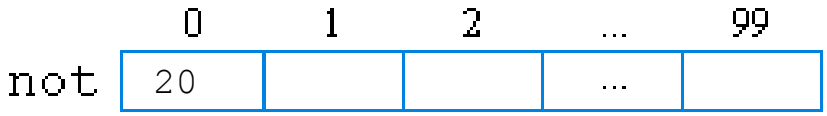


Dizi Tanımı

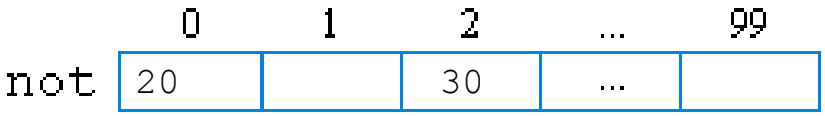
Örnek:
```
int not[4];
int i;
```




```
not[0]=20;
```




```
not[2]= not[0]+10;
```




```
i=0;
not[i]=90;
not[++i]=70;
```

Dizi Elemanlarına Değer Atama

```
int a[3],b;
scanf(“%d”, &b);
a[2]=b;
```

veya

```
int a[3];
scanf(“%d”, &a[2]);
```

Dizi Elemanlarına Değer Atama

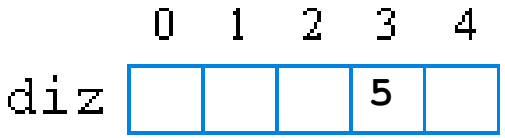
Örnek:Kullanıcıdan alınan 5 tamsayı değerini bir dizide saklayan ve bu değerlerin ortalamasını bulan bir program yazalım

```
#include <stdio.h>
int main(void)
{           int a[5];
            int i,toplam;
            double orta;
            printf("Bes tamsayi giriniz:");
            for(i=0;i<5;++i)       //Kullanici degerlerinin dizide saklanmasi
              scanf("%d",&a[i]);
            /*Toplam ve ortalamanin bulunmasi*/
            toplam=0;
            for(i=0;i<5;++i)
              toplam=toplam+a[i];
            orta= toplam/5.0;
            printf("Ortalama=%5.2f",orta);
            return (0);       }
```

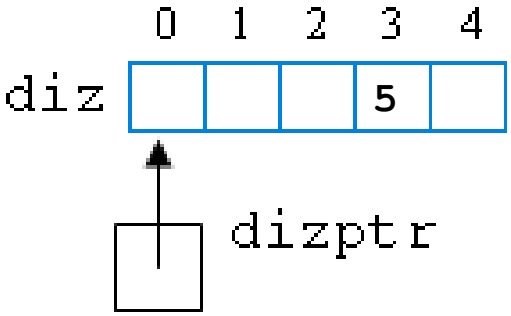
Diziler ve Göstergeler

Bir dizi ismi aslında dizinin ilk elemanını gösteren sabit bir gösterge olarak yaratılır.

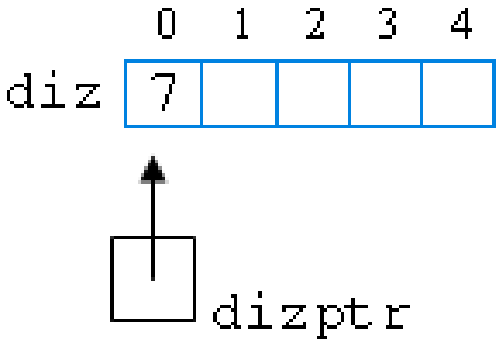
```
int diz[5];
diz[3] = 5;
```




```
int *dizptr;
dizptr = &diz[0];
```




```
*dizptr = 7;
```



Diziler ve Göstergeler

Gösterge Gösterimi

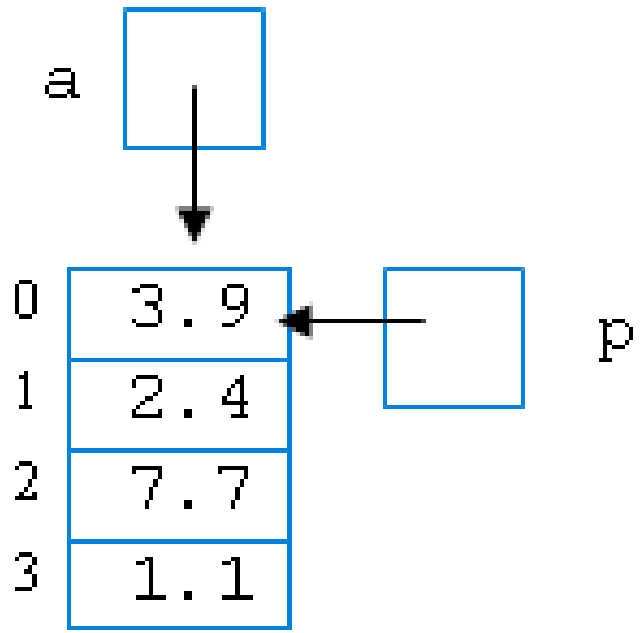
```
*dizptr = 5;
*(dizptr + 1) = 6;
*(dizptr + 2) = 7;
     .
       .
       .
*(dizptr + i) = 8;
```

Dizi Gösterimi

```
diz[0] = 5;
diz[1] = 6;
diz[2] = 7;
      .
                .
                .
diz[i] = 8;
```

Diziler ve Göstergeler

```
 1  #include <stdio.h>
 2  int main(void)
 3  {    double a[]={3.9,2.4,7.7,1.1};
 4       double *p;
 5       p=a;
 6       printf(" %f ", *a);
 7       printf(" %f ", a[0]);
 8       printf(" %f ", *p);
 9       printf(" %f \n", p[0])
10       printf(" %f ", p[1]);
11       printf(" %f ", a[1]);
12       return (0);
13  }
```



Diziler ve Fonksiyonlar

Dizi Elemanının Fonksiyona Gönderilmesi

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

```
int a[5]={1,2,3,4,5};
printf("%d", topla(a[0],a[4]));
```

Diziler ve Fonksiyonlar

Dizinin Fonksiyona Gönderilmesi

```
int a[5]={10,20,30,40,50};
```

a dizisini **f()** fonksiyonuna yollamak için aşağıdaki komutu kullanmalıyız.

```
f(...,a,...);
```

Bu fonksiyonun başlığı ise iki şekilde olabilir.

```
void f(  ,int x[5],  )
void f(  ,int x[],  )
```

Fonksiyon parametresinde tanımlanan **x** dizisi yoluyla yapılan değişiklikler, gerçek parametre olan **a** dizisini etkileyecektir. Çünkü dizi ismi dizinin ilk elemanının yerini gösteren bir göstergedir ve kaynak parametresi gibi davranacaktır.

Diziler ve Fonksiyonlar

```c
#include <stdio.h>
void f1(int b[], int n);//Kullanılacak fonksiyon

int main(void)
{       int i, a[]={1,2,3};
        f1(a,3);//Aşağıdaki fonksiyona gidilir
        printf("\nmain fonksiyonu ");

        for(i=0;i<3;++i)//Üç defa aşağıdaki işlemi tekrar et
        printf("%d", a[i]);
        return(0);
}//Ana programın sonu

void f1(int b[], int n)//Yukarıda tanımlanan fonksiyon
{       int i;
        printf("f1 fonksiyonu ");

        for(i=0;i<n;++i)//n defa aşağıdaki işlemi tekrar et
        { printf("%d", b[i]);
        b[i]=8;
        }
}
```

Çıktı:

```
f1 fonksiyonu 123
main fonksiyonu 888
```

Diziler ve Fonksiyonlar

Dizi Yerine Gösterge Kullanımı

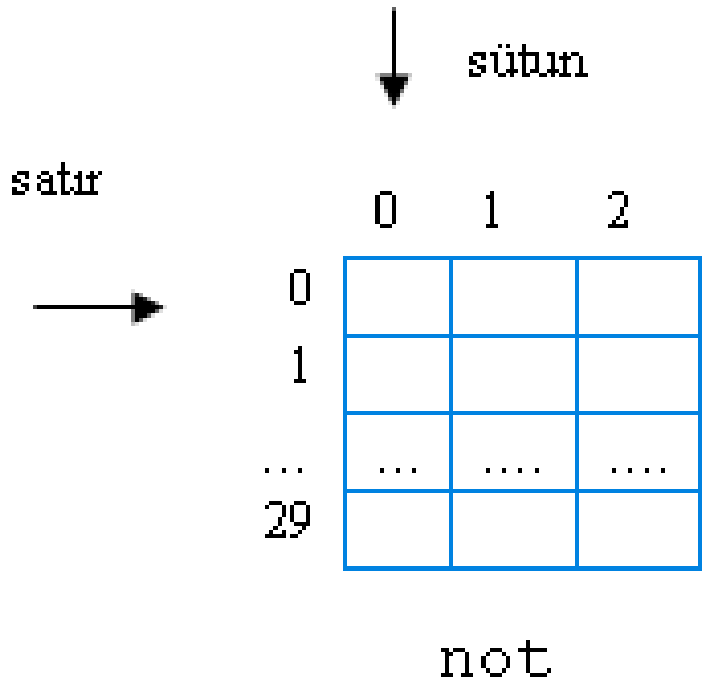
```
float topla (float apar[])
{  int i;
   float toplam=0.0;
   for (i=0;i<4; i++)
        toplam+=apar[i];
   return (toplam);
}
```

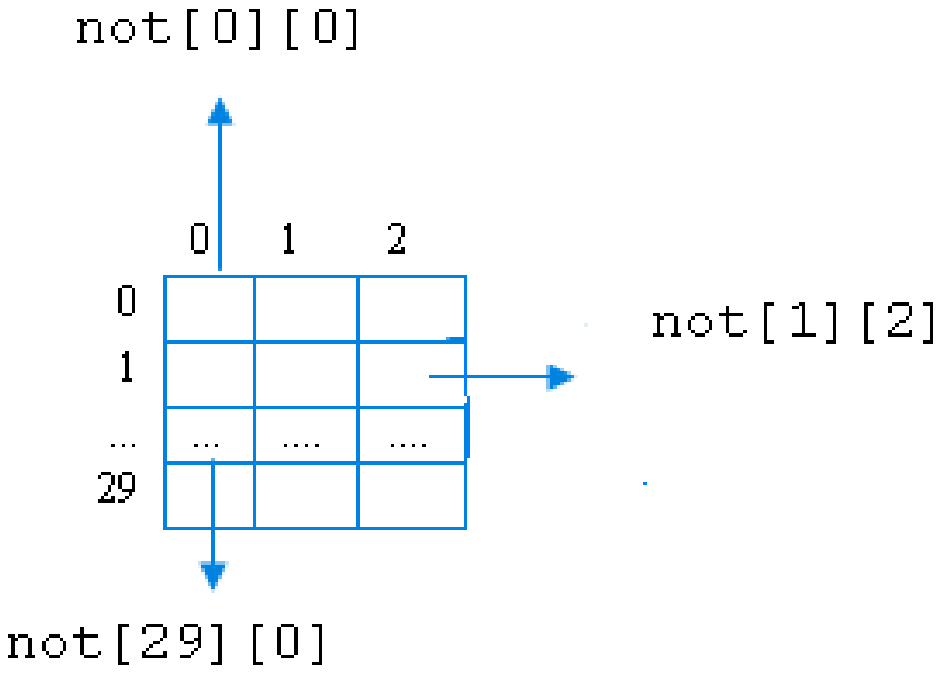
```
float topla (float *aptr)
{   int i;
    float toplam=0.0;
    for (i=0;i<4; aptr++,i++)
        toplam+=*aptr;
    return (toplam);
}
```

2-Boyutlu Diziler

Her elemanı bir boyutlu dizi olan yapılara ise ***çok boyutlu diziler*** denir. Bu bölümde ***matris*** veya ***tablo*** olarak da bilinen 2-boyutlu dizilere değineceğiz.

veri_tipi dizi_ismi[*satır_büyüklüğü*][*sütun_ büyüklüğü*];





2-Boyutlu Diziler

2-Boyutlu Dizilere Değer Atama

```
scanf("%d", &not[29][0]);
```

not[1][1]=90;

Tanımlama Sırasında Değer Atama

```
int b[4][3]={{15,30,39},{23,65,30},{32,61,12},{48,34,11}};
```

1. satır 2.satır 3.satır 4.satır

```
int b[][3]={{15,30,39},{23,65,30},{32,61,12},{48,34,11}};
```

1.satır 2.satır 3.satır 4.satır

	0	1	2
0	15	30	39
1	23	65	30
2	32	61	12
3	48	34	11

b

satır yönünde atama

2-Boyutlu Diziler

Tanımlama Sırasında Değer Atama

```
int b[4][3]= {  {15},     {23},     {32},   {48}};
```

1. satır 2.satır 3.satır 4.satır

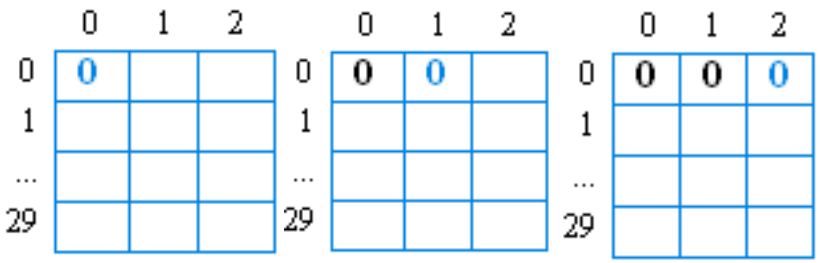
	0	1	2
0	15	30	39
1	23	0	0
2	0	0	0
3	0	0	0

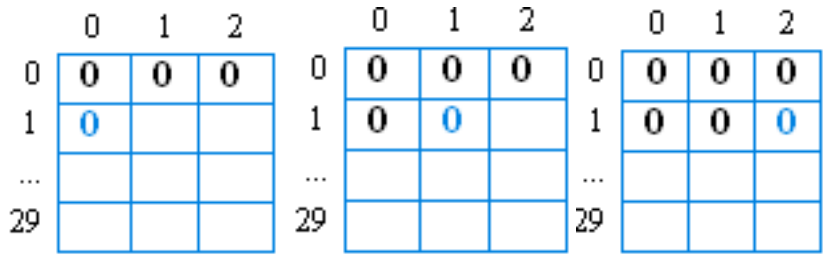
b

2-Boyutlu Diziler

Tanımlama Sonrasında Satır Yönünde Değer Atama

```
int a[30][3];
for(satir=0; satir<30; ++satir)
{ for (sutun=0;sutun<3; ++sutun)
        a[satir][sutun]=0;
}
```

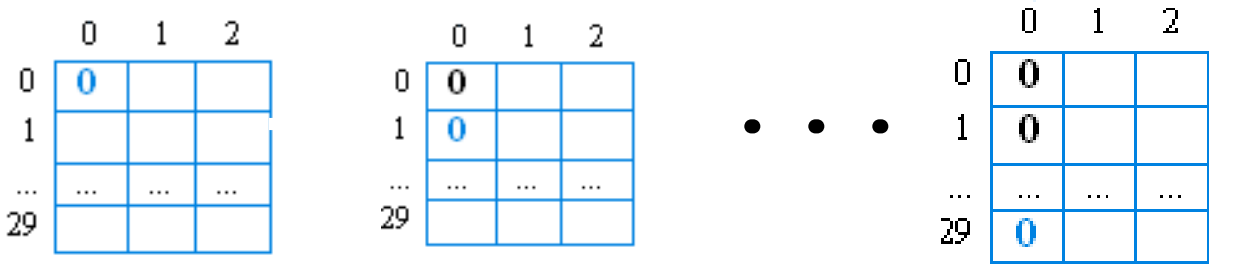


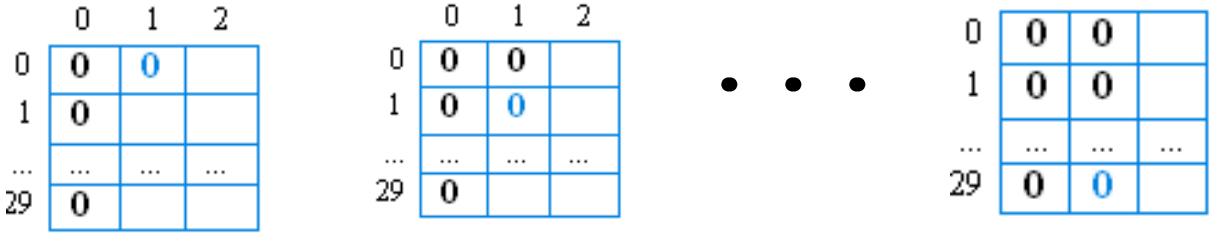


2-Boyutlu Diziler

Tanımlama Sonrasında Sütun Yönünde Değer Atama

```
int a[30][3];
for(sutun=0; sutun <3; ++sutun)
  for (satir=0;satir<30;++satir)
       a[satir][sutun]=0;
```





2-Boyutlu Diziler

Örnek:Kullanıcının, 30 kişilik bir sınıftaki her öğrenci için 3'er sınav notu gireceği ve her sınav için sınıf ortalamasının ekranda gösterileceği bir program yazınız.

```
#include <stdio.h>
int main(void)
{ int not1[5][3];
  int i,j,toplam;
  double orta;
  /* Kullanicidan notlarin alinmasi*/
  for(i=0;i<5;++i)
  {  printf("%d. ogrenci notlari:",i+1);
     for(j=0;j<3;++j)
     scanf("%d", &not1[i][j]);
  }       /*Her sinavin ortalamasinin bulunmasi*/
  for(j=0;j<3;++j)
  {  toplam=0;
     for(i=0;i<5;++i)
toplam+=not1[i][j];
     orta=toplam/5.0;
     printf("%d. sinav ortalamasi: %5.2f\n",j+1,orta);
  }
  return (0);     }
```

2-Boyutlu Diziler

2-Boyutlu Diziler ve Fonksiyonlar

Dizi tanımı

```
int a[2][3]={1,2,3,4,5,6};
```

Fonksiyon çağırma

```
f1(   ,a,   );
```

Fonksiyon başlığı

```
f1(   , int b[][3],   );
```

2-Boyutlu Diziler

Örnek: Öğrencilerin sınav ortalamalarını bulacak bir program yazalım. Bu program için `ortalama()` isimli bir fonksiyon kullanalım. `ortalama()` fonksiyonu çağırıldığı yerden 2-boyutlu bir dizi alır, dizinin her satırının ortalamasını bulup, tek boyutlu bir başka dizide saklar ve ortalamaları tutan bu tek boyutlu diziyi çağırıldığı yere geri döndürür.

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

void ortalama(int ogr_sa,int sinav_sa, int not1[][3], double orta[])
{       double toplam;
        int i,j;
        for(i=0;i<ogr_sa;++i)
        {       toplam=0;
                for(j=0;j<sinav_sa;++j)  toplam+=not1[i][j];
                orta[i]=toplam/3.0;   } }

int main(void)
{       int test[30][3];
        int i,j,toplam;
        double averaj[30];
        for(i=0;i<30;++i) /*Kullanicidan notlarin alinmasi*/
        {       printf("%d. ogrenci notlari:", i+1);
                for(j=0;j<3;++j)
                  scanf("%d",&test[i][j]);       }

ortalama(30,3,test,averaj) ; /*Her ogrencinin ortalamasinin bulunmasi*/
        for(i=0;i<30;++i)
        printf("%d.ogrenci ortalamasi:%5.2f\n",i+1, averaj[i]); }
```

Dizgiler (String)

Dizgi Tanımı

Dizgi Girdi İşlemleri

Dizgi Çıktı İşlemleri

Dizgi Fonksiyonları

Karakter Fonksiyonları

Gösterge Dizgileri

Dizgi Tanımı

Dizgi	Açıklama
“Merhaba”	7 karakter içeren bir dizgi
“Bu bir dizgi”	12 karakter içeren dizgi.
“B”	Bir karakter içeren bir dizgi
“”	Boş dizgi

Karakter dizisine değer yükleme

Üç farklı yolla yapılır:

- static char kk[]="Ankara"; /* burada program derlenirken karakter uzunluğu*/

- static char kk[]={'A', 'n', 'k', 'a' , 'r', 'a', '\0'};

- static char kk[7]="Ankara";

Dizgi Tanımı

Örnek: ileriki bölümlerde açıklanacak bir yapı

```
char cumle[] = "Merhaba Dunya";
int say = 0;
int i;
for (i = 0; cumle[i] != '\0'; i++)  say++;
printf("%s %d karakter icerir.", cumle, say);
```

Çıktı:

```
Merhaba Dunya 13 karakter icerir.
```

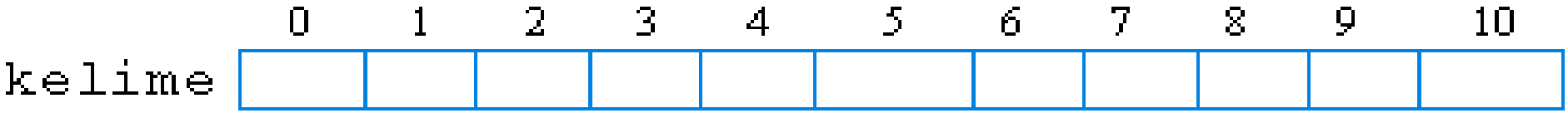
ÖRNEK: Üniversitenizin adını ekrana yazdıran kodu yazınız?

```
#include <stdio.h>
#include <conio.h>
int main()
{
char kk[]="Gazi Universitesi";
printf("Universiteniz %s'dir\n", kk);
return 0;
}
```

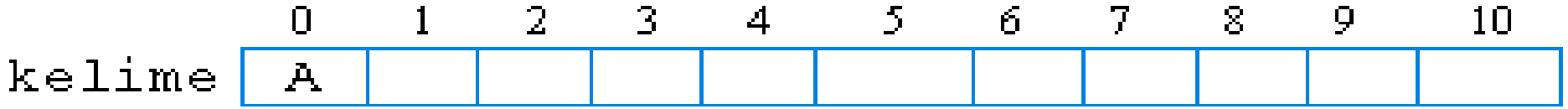
Dizgi Tanımı

char ***dizi_adı*[*uzunluk*];**

```
char kelime[11];
```

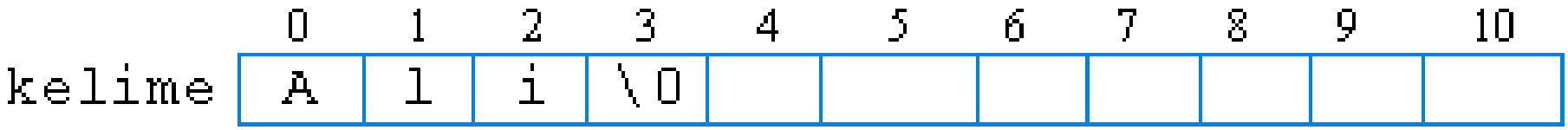



```
kelime[0] = 'A';
```



Dizgi Tanımı

```
kelime[1] = 'l';
kelime[2] = 'i';
kelime[3] = '\0';
```



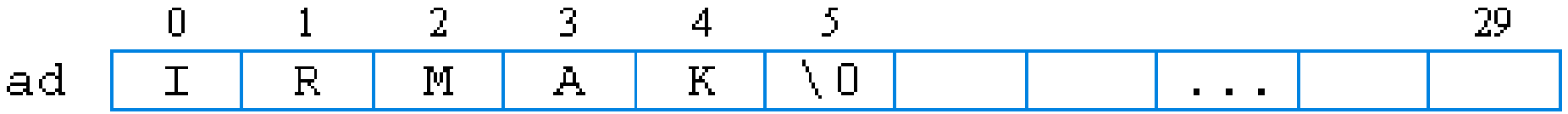
Bir dizginin sonu ***boş karakter*** (NULL character) olan `'\0'` karakteri ile biter.

Dizgi Tanımı

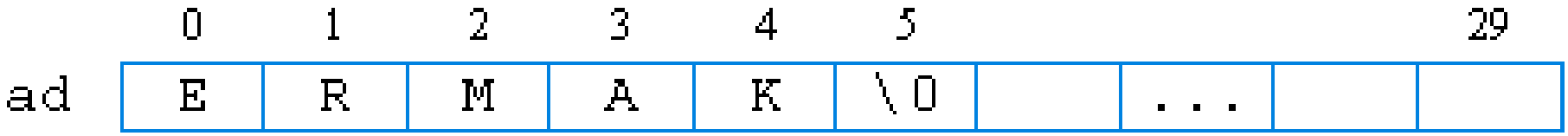
Dizgileri tanımlarken ilk değerini de atayabiliriz.

`char` *dizi_adı*[*uzunluk*] = *dizgi_sabiti*;

```
char ad[30]="IRMAK";
```




```
ad[0]='E';
```

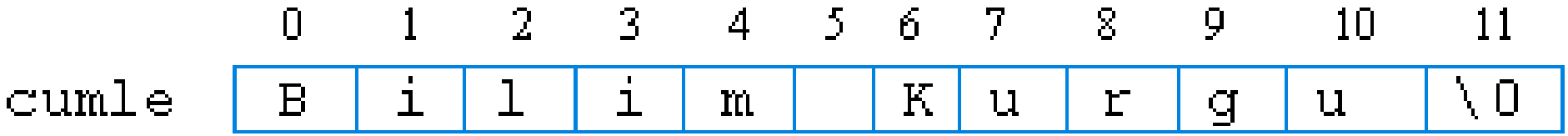


Dizgi Tanımı

Dizgi tanımlamalarını, dizgi uzunluğunu dizi tanımlaması sırasında verilmeden ve ilk değerini atayarak da yapabiliriz.

`char` *dizgi*`[]` = *dizgi_sabiti*;

```
char cumle[] = "Bilim Kurgu";
```

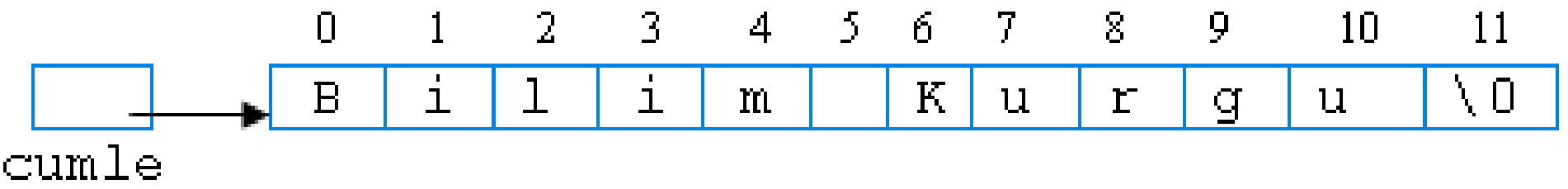


Dizgi Tanımı

Dizgiler tanımlanırken göstergeler kullanılarak da aşağıdaki gibi tanımlanabilir. Çünkü her bir dizgi aslında bir dizi ile tanımlanmıştır.

char ****dizgi_adı* **= ***dizgi_sabiti*;**

```
char *cumle = "Bilim Kurgu";
```



Dizgi Girdi İşlemleri

`scanf()` fonksiyonu girilen değerler içinde boşluk veya enter işareti (↵) görünceye kadar okuma işine devam eder

`scanf ("%s", dizgi_adı);`

Örnek:

```
char kelime[11];
scanf ("%s", kelime);
```

Girdi

```
Bilgisayar↵
Bilge Sor.↵
Bil↵
```

kelime

B	i	l	g	i	s	a	y	a	r	\0
B	i	l	g	e	\0					
B	i	l	\0							

Dizgi Girdi İşlemleri

```
char kelime[11];
scanf ("%7s", kelime);
```

Girdi	kelime									
Programlama↵	P	r	o	g	r	a	m	\0		
Prog.↵	P	r	o	g	.	\0				

Dizgi Girdi İşlemleri

`gets()` fonksiyonu enter ya da girdi sonunu belirleyen (ctrl+z) karakterini görünceye kadar girdiyi okumaya devam eder ve okuduğu değerin sonuna boş karakterini '\0' otomatik olarak ekleyerek *dizgi_adi*'na bu değerleri atar.

`gets (`*dizgi_adı*`);`

Örnek:

```
char cumle[15];
gets (cumle);
```

Girdi

␣␣iyi␣misin?↵

cumle

␣	␣	i	y	i	␣	m	i	s	i	n	?	\0		

Dizgi Girdi İşlemleri

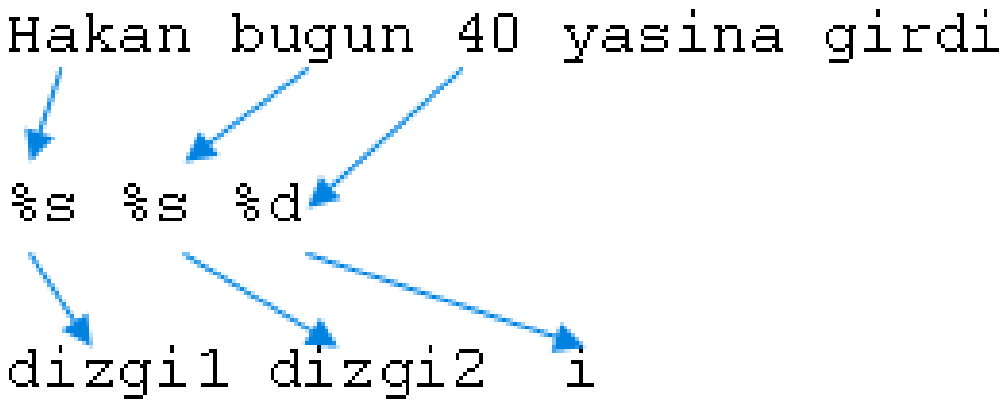
sscanf() fonksiyonu kullanıldığında girdi bilgisi klavyeden değil bir başka dizgiden alınır.

sscanf (*dizgi_adı, format_dizgisi, girdi_listesi*);

Örnek:

```
char cumle[]="Hakan bugun 40 yasina girdi";
char dizgi1[20], dizgi2[20];
int i;

sscanf (cumle,"%s %s %d",dizgi1,dizgi2,&i);
printf ("%s --> %d\n",dizgi1, i);
```



Dizgi Çıktı İşlemleri

`printf()` fonksiyonunu dizgilerin bastırılması amacıyla da kullanabiliriz.

`printf` **(“%s”, *dizgi_adı*);**

Örnek:

```
char dizgi1[15]= “merhaba”;
char dizgi2[]= “iyi”;
```

Komut	Çıktı
`printf (“%s”, dizgi1);`	`merhaba`
`printf (“%s”, “Nasilsin?”);`	`Nasilsin?`
`printf (“%5s”, dizgi2);`	`␣␣iyi`
`printf (“%-5s”, dizgi2);`	`iyi␣␣`
`printf (“%s %s”, dizgi1, dizgi2);`	`merhaba iyi`

Dizgi Çıktı İşlemleri

puts() fonksiyonu standart çıktı birimine yani ekrana dizginin değerinin bastırılmasını sağlar ve daha sonra yeni satır karakterini otomatik olarak çıktının sonuna ekler.

puts (*dizgi_adı*);

Örnek:

```
char dizgi1[15]= “merhaba”;
char dizgi2[]= “iyi”;

char dizgi1[]="merhaba";
char dizgi2[]="nasilsin?";
puts(dizgi1);
puts(dizgi2);
```

Çıktı:

```
merhaba
nasilsin?
```

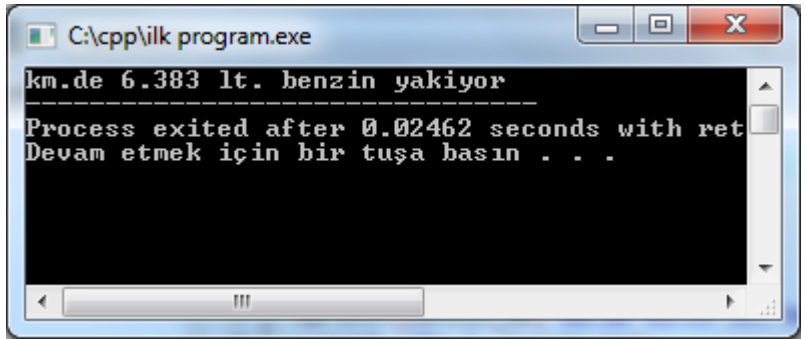
Dizgi Çıktı İşlemleri

sprintf() fonksiyonu farklı değişkenlerin değerini belirli bir format dizgisine uygun olarak yeni bir dizginin içine kopyalar.

sprintf (***dizgi_adı, format_dizgisi, liste***);

Örnek:

```
#include <stdio.h>
main(){
float benzin = 47.0;
float km = 300;
char benzin_km[80];
sprintf(benzin_km,"km.de %5.3f lt. benzin yakiyor", km/benzin);
printf ("%s ", benzin_km);
}
```



Dizgi Fonksiyonları

Dizgi işleme amacıyla hazırlanmış programlarda kolaylık sağlayabilecek bir çok fonksiyon **<string.h>** kütüphanesi içinde tanımlanmıştır. Bir dizginin içindeki karakter sayısını bulmak için **strlen()** fonksiyonu kullanılır.

strlen(*dizgi_adı*);

Örnek:

```
int uzunluk;
char dizgi[10] = "Ali"
uzunluk = strlen(dizgi);
```


3

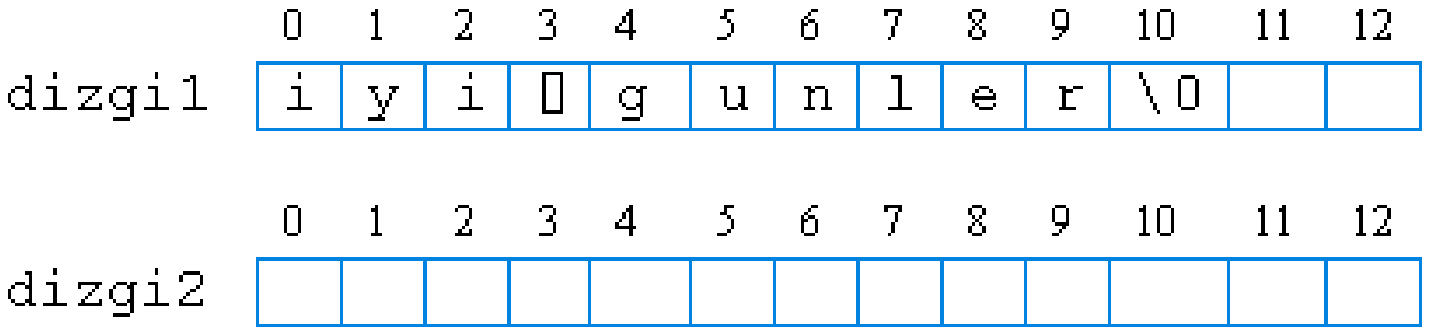
Dizgi Fonksiyonları

strcpy() fonksiyonu dizgi kopyalama fonksiyonudur.

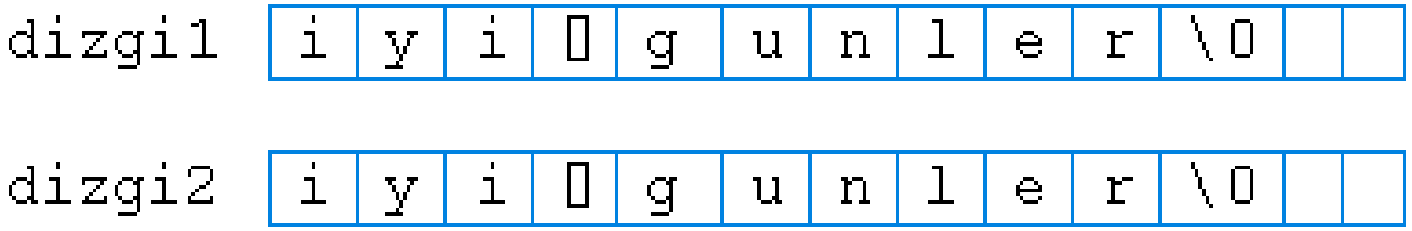
strcpy (*dizgi2_adı, dizgi1_adı*);

Örnek:

```
char dizgi1[13]="iyi gunler";
char dizgi2[13];
```




```
strcpy (dizgi2, dizgi1);
```



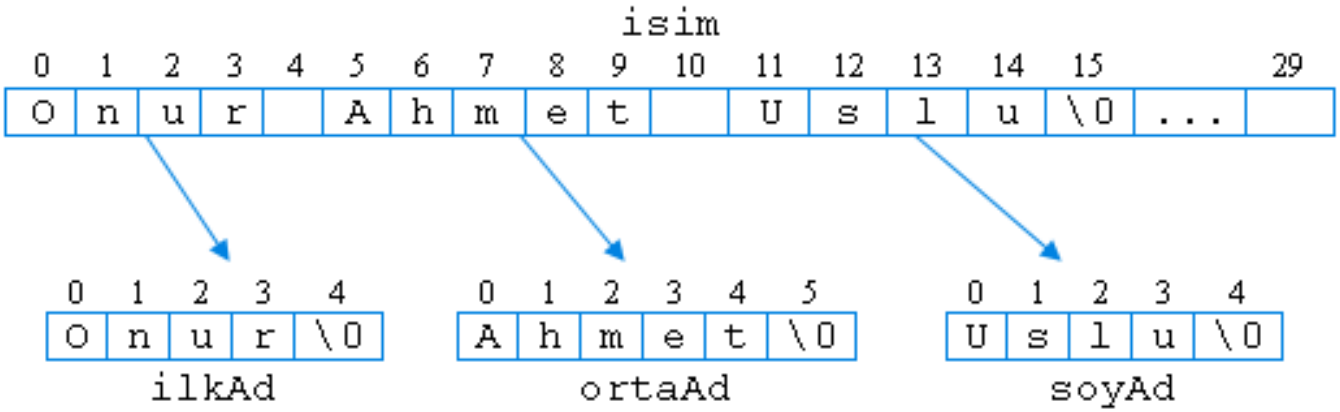
Dizgi Fonksiyonları

strncpy() fonksiyonu *dizgi1_adı*'nın içindeki ilk *n* karakterin *dizgi2_adı*'na kopyalanmasını sağlar.

strncpy (*dizgi2_adı, dizgi1_adı, n*);

Örnek:

```
char isim[30]="Onur Ahmet Uslu";
char soyAd[10], ilkAd[10], ortaAd[10];
```

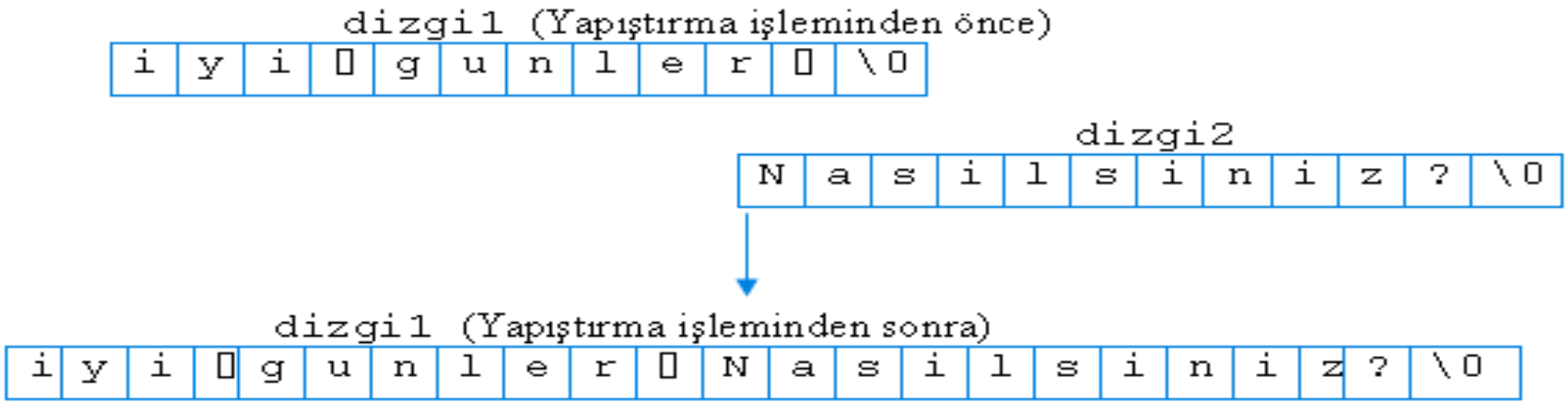



```
strncpy(ilkAd, isim,4); ilkAd[4]='\0';
strncpy(ortaAd, &isim[5], 5);
ortaAd[4]='\0'; strcpy(soyAd, &isim[11]);
```

Dizgi Fonksiyonları

`strcat()` fonksiyonu bir dizginin sonuna diğer bir dizginin yapıştırılmasını sağlar.

```
strcat (dizgi1_adı, dizgi2_adı);
Örnek:
char dizgi1[12]="iyi gunler ";
char dizgi2[12]= "Nasilsiniz?";
strcat (dizgi1, dizgi2);
printf ("\ndizgi 1: %s %d",dizgi1, strlen(dizgi1));
printf ("\ndizgi 2: %s ",dizgi2);
```



Dizgi Fonksiyonları

strncat() fonksiyonu *dizgi2_adı*'nın ilk *n* karakterinin *dizgi1_adı*'nın sonuna yapıştırılmasını sağlar.

strncat(*dizgi1_adı, dizgi2_adı, n*);

Örnek:

```
char dizgi1[15]="iyi gunler ";
char dizgi2[15]= "Nasilsiniz?";
strncat (dizgi1, dizgi2, 5);
printf ("\ndizgi 1: %s ",dizgi1);
printf ("\ndizgi 2: %s ",dizgi2);
```

Çıktı:

```
dizgi 1: iyi gunler Nasil
dizgi 2: Nasilsiniz?
```

Dizgi Fonksiyonları

`strcmp()` fonksiyonu iki dizginin karşılaştırılmasını sağlar.

`strcmp` (***dizgi1_adı, dizgi2_adı***);

dizgi1	dizgi2	strcmp(dizgi1, dizgi2)
Balik	Kapi	Negatif
Kapi	Balik	Pozitif
Kapi	Kapi	0
Kapi	Kapici	Negatif
Kapi	kapi	Pozitif

Dizgi Fonksiyonları

`strncmp()` fonksiyonu iki dizginin ilk n karakterlerinin karşılaştırılmasını sağlar.

`strncmp (`*dizgi1_adı, dizgi2_adı, n*`);`

Örnek:

```
char dizgi1[13]="iyi gunler ";
char dizgi2[13]= "iyi misiniz?";
printf ("\n%d ",strncmp (dizgi1, dizgi2, 3));
```

Çıktı:

```
0
```

Dizgi Fonksiyonları

`strstr()` fonksiyonu bir dizginin içinde diğer bir dizgiyi arar.

`strstr(`*dizgi1_adı, dizgi2_adı*`);`

```
Örnek:
char dizgi1[13]="iyi gunler ";
char dizgi2[13]= "gun";
if (strstr (dizgi1, dizgi2) == '\0')
   printf ("dizgi2 dizgi1 in icinde YOK");
else
   printf ("dizgi2 dizgi1 in icinde VAR");

Çıktı:
dizgi2 dizgi1 in icinde VAR
```

Dizgi Fonksiyonları

Örnek: Kullanıcının girdiği bir dizgiyi okuyarak, bu dizginin tersini bulan bir program yazınız.

```
#include <string.h>
 int main(void)
 {
   char str2[30], str1[30];
   int i, uzunluk;
   printf("Bir dizgi giriniz:");
   gets(str1);
   uzunluk=strlen(str1);
   for(i=0;i<=uzunluk;++i)
   strncpy(&str2[i],&str1[uzunluk-i-1],1);
   printf("%s",  str2);
   return(0);
 }
```

```
Çıktı:
Bir dizgi giriniz:kitap
patik
```

Ödev 4

1. Bir boyutlu bir dizi içine klavyeden girilmek sureti ile atanacak sayılardan sırasına bağlı olarak sorulduğunda sayıyı gösterecek programı yapınız.
2. 2x2 boyutunda bir matrisin sıra ile değerlerini soran ve sonunda matrisin elemanlarını satır ve sütunları belli olacak şekilde ekranda gösteren programı yapınız
3. Sizden sırası ile önce adınızı sonra soyadınızı yaşınızı ve telefonunuz sorup, işlem sonunda hepsini tek satırda gösteren programı yapınız
4. 2x2 boyutunda bir matrisin sıra ile değerlerini soran ve terisini başka bir matrise atayarak matrisi ekranda gösteren programı yapınız
5. Girilecek bir ismin ilk harfi büyük diğerleri küçük olacak şekilde ekranda gösteren programı yapınız.