Ders 10: *Diziler*

```
###################- (%95)
```

En son güncelleme: Wed, 30 Nov 2011 13:22:02 +0200

Giriş

Dizi, aynı tipteki verilere tek bir isimle erişmek için kullanılan bir kümedir. Bu küme matematikteki küme kavramından biraz farklıdır. Bir dizi bildirildikten sonra, dizinin bütün elemanları bellekte peşpeşe saklanır [1]. Bu yüzden dizilere tek bir isim altında çok sayıda değişken içeren bellek bölgesi de denir. Buna göre, bir diziyi dizi yapan iki temel özellik vardır [2]:

- dizi elemanların bellekte (program çalıştığı sürece) sürekli biçimde bulunması
- dizi elemanların aynı türden değişkenler olması

10.1 Dizilerin Bildirimi

Bir dizi çok sayıda değişken barındırdığından, bunları birbirinden ayırdetmek için *indis* adı verilen bir bilgiye ihtiyaç vardır. C Programlama Dili'nde, bir dizi hangi tipte tanımlanmış olursa olsun başlangıç indisi her zaman 0'dır.

Bir dizinin bildirim işleminin genel biçimi söyledir:

```
veriTipi dizi_adı[eleman_sayısı];
```

Örneğin, 5 elemanlı, kütle verilerini bellekte tutmak için, `kutle` dizisi şöyle tanımlanabilir:

```
float kutle[5];
```

Bu dizinin elemanlarına bir değer atama işlemi şöyle yapılabilir:

```
kutle[0] = 8.471
kutle[1] = 3.683
kutle[2] = 9.107
kutle[3] = 4.739
kutle[4] = 3.918
```

NOT

1. elemanın indisi 0,
5. elemanın indisinin 4 olduğuna dikkat edin.

Bildirim sırasında dizilerin eleman sayısı tamsayı türünden bir sabit ifadesiyle belirtilmesi zorunludur. Örneğin:

```
int n = 100;
int a[n];
```

şeklindeki tanımlama, dizi uzunluğunun değişken (`n`) ile belirtilmesi nedeniyle geçersizdir. Bunun yerine, dizilerin eleman sayısı aşağıdaki gibi sembolik sabitlerle belirtmek mümkündür.

```
#define  n  100
...
int a[n];
```

Bir dizinin bellekte kapladığı alanın bayt cinsinden karşılığı `sizeof` operatörü ile öğrenilebilir.

```
int   a[5], b, c;
...
b = sizeof(a);                /* bellekte kapladığı alan: b = 4*5  = 20 bayt */
c = sizeof(a) / sizeof(int);  /* Dizinin boyutu         : c = 20/4 = 5       */
```

10.2 Dizilere Başlangıç Değeri Verme

Bir diziye başlangıç değerleri aşağıdaki gibi kısa formda atanabilir:

```
float  kutle[5]= { 8.471, 3.683, 9.107, 4.739, 3.918 };
int    maliyet[3] = { 25, 72, 94 };
double a[4] = { 10.0, 5.2, 7.5, 0.0};
```

Küme parantezlerinin sonlandırıcı **;** karakteri ile bittiğine dikkat ediniz.

Bir dizinin uzunluğu belirtilmeden de başlangıç değeri atamak mümkündür.

```
int   a[] = { 100, 200, 300, 400 };
float v[] = { 9.8, 11.0, 7.5, 0.0, 12.5};
```

Derleyici bu şekilde bir atama ile karşılaştığında, küme parantezi içindeki eleman sayısını hesaplar ve dizinin o uzunlukta açıldığını varsayar. Yukarıdaki örnekte, `a` dizisinin 4, `v` dizisinin 5 elemanlı olduğu varsayılır.

10.3 Dizileri Yazdırma/Okuma

`printf` ve `scanf` fonksiyonları bir dizinin okunması ve yazdırılması için de kullanılır. Örneğin bir `A` dizisinin aşağıdaki gibi bildirildiğini varsayalım:

```
int A[10];
```

Bu dizinin elemanlarını klavyeden okumak için:

```
for(i=0; i<10; i++)
   scanf("%d",&A[i]);
```

daha sonra bu değerlerini ekrana yazmak için:

```
for(i=0;i<10;i++)
   printf("%d\n",A[i]);
```

Program 10.1, klavyeden girilen $N = 10$ adet sayının ortalamasını hesaplar. Ortalama $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ formülü ile hesaplanabilir.

Program 10.1: *10 sayının ortalamasını hesaplar*

```
01: /* 10prg01.c: 10 tamsayının aritmetik ortalamasını hesaplar */
02:
03: #include <stdio.h>
04:
05: #define N 10
06:
07: int main()
08: {
09:    int   i;
```

```
10:    float x[N], ort, toplam = 0.0;
11:
12:    for(i=0; i<N; i++)
13:    {
14:       /* i. eleman okunuyor ... */
15:       printf("%d. sayi : ",i+1);
16:       scanf("%f",&x[i]);
17:
18:       toplam += x[i];
19:    }
20:
21:    ort = toplam/N;
22:
23:    printf("Sayilarin ortalamasi = %f\n",ort);
24:
25:  return 0;
26: }
```

ÇIKTI

```
1. sayi : 1
2. sayi : 0
3. sayi : 9
4. sayi : 7
5. sayi : 2
6. sayi : 10
7. sayi : 11
8. sayi : 4
9. sayi : 6
10. sayi : 5
Sayilarin ortalamasi = 5.500000
```

Bu programda, ortalaması alınacak sayılar adı x olan 10 elemanlı tamsayı tipindeki bir dizide saklanmıştır. Bu şekilde saklanan sayıların hepsi program çalıştığı sürece bellekte kalacaktır. Bu sayede, program içinde daha sonra (gerektiğinde) aynı sayılar tekrar kullanılabilir. Bu program, dizi kullanmadan da yazılabilirdi. Fakat, bazı hallerde dizi kullanmak kaçınılmaz olur.

Porgram 10.2, n = 10 tane sayının ortalamasını ve standart sapmasını hesaplar. Standart sapma,

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

formülü ile hesaplanabilir. Burada,

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Program 10.2: *10 sayının ortalamasını ve standart sapmasını hesaplar*

```
01: /* 10prg02.c
02:    10 tane sayının aritmetik ortlamasını ve standart sapmasını hespalar. */
03:
04: #include <stdio.h>
05: #include <math.h>
06:
07: #define N 10
08:
09: int main(void)
10: {
11:    int   i;
12:    float x[N], toplam = 0.0, ort, std_sap = 0.0;
13:
14:    /* ortalama hesabı */
15:    for(i=0; i<N; i++)
16:    {
17:       printf("%d. sayi : ",i+1);
18:       scanf("%f",&x[i]);
19:
```

```
20:        toplam += x[i];
21:     }
22:
23:     ort = toplam/N;
24:
25:     /* standart sapma hesabı */
26:     for(toplam = 0.0, i=0; i<N; i++)
27:         toplam += pow(x[i]-ort, 2.0);
28:
29:     std_sap = sqrt( toplam/(N-1) );
30:
31:     printf("Ortalama       = %f\n",ort);
32:     printf("Standart sapma = %f\n",std_sap);
33:
34:    return 0;
35: }
```

ÇIKTI

```
1. sayi : 7
2. sayi : 8
3. sayi : 9
4. sayi : 6
5. sayi : 5
6. sayi : 8
7. sayi : 8
8. sayi : 10
9. sayi : 11
10. sayi : 6
Ortalama       = 7.000000
Standart sapma = 2.054805
```

10.4 Sıralama (Sorting)

Bazı uygulamalarda bir grup sayının büyükten küçüğe, veya küçükten büyüğe, doğru sıralanması gerekebilir. Bu tip sıralama problemleri için çeşitli algoritmalar geliştirilmiştir. Sıralama mantığını anlamadan önce bir dizinin en büyük (veya en küçük) elemanının nasıl bulunduğunu inceleyelim. Program 10.3, bir dizinin en büyük elemanını bulup ekrana yazar.

Program 10.3: *Bir dizinin en büyük elemanının bulunuşu*

```
01: /* 10prg03.c
02:    Bir dizinin en büyük elemanını bulup ekrana yazar */
03:
04: #include <stdio.h>
05:
06: int main(void)
07: {
08:    int   a[10] = {100, -250, 400, 125 ,550, 900, 689, 450, 347, 700};
09:    int   k, eb;
10:
11:    /* ilk eleman en büyük kabul ediliyor */
12:    eb = a[0];
13:
14:    for(k=1; k<10; k++)
15:       if( a[k]>eb ) eb = a[k];
16:
17:    printf("En buyuk eleman = %d\n",eb);
18:
19:  return 0;
20: }
21:
```

ÇIKTI

```
En buyuk eleman = 900
```

En büyük sayıyı bulan bu algoritma oldukça kolaydır. 12. satırda `eb = a[0];` ataması ile dizinin ilk elemanının en büyük olduğu varsayılır. Daha sonra büyüğe rastladıkça (15. satır) `eb = a[k];` ile `eb` değişmektedir.

Program 10.3 bir dizinin en büyük elemanını bulur. En büyük elemanın kaçıncı indis (eleman) olduğu sorgulanmak istendiğinde: programdaki koşul yapısını aşağıdaki gibi değiştirmek yeterlidir. `eb` değiştikçe, `i` değişkeni en büyük elemanın indisini tutar.

```
for(k=0; k<10; k++){
   if( a[k] > eb ){
     eb = a[k];
     i = k;
   }
}
```

n elemanlı bir dizinin, elemanlarını büyükten küçüğe doğru sıralamak için çok popüler iki algoritma aşağıda verilmiştir[2].

Seçerek Sıralama (Selection Sort):
En büyük elemanı bul başa koy biçimindeki sıramadır. Algoritmanın uygulaması Program 10.4'de gösterilmiştir.
Bu algoritmada kullanılan kombinasyon sayısı (algoritmanın karmaşıklığı): `n*(n-1)/2` dir.

Program 10.4: *Seçerek Sıralama (Selection Sort) Algoritması*

```
01: /* 09prg04.c
02:    Seçerek Sıralama (Selection Sort) Algoritması ile bir
03:    dizinin elemanlarını büyükten küçüğe dogru sıralar */
04:
05: #include <stdio.h>
06:
07: #define n 10
08:
09: int main(void)
10: {
11:    int   a[n] = {100, -250, 400, 125 ,550, 900, 689, 450, 347, 700};
12:    int   i, j, k, eb;
13:
14:    /* Dizinin kendisi */
15:    printf("Once : ");
16:    for(k=0;k<n;k++)
17:       printf("%5d ",a[k]);
18:
19:    /* Sırala */
20:    for(k=0; k<n; k++){
21:
22:       eb = a[k];
23:        i = k;
24:
25:        for(j=k+1; j<n; j++)
26:           if( a[j]>eb ){
27:             eb = a[j];
28:              i = j;
29:           }
30:
31:        a[i] = a[k];
32:        a[k] = eb;
33:    }
34:
35:    /* Sıralama bitti */
36:    printf("\nSonra: ");
37:    for(k=0; k<n; k++)
38:       printf("%5d ",a[k]);
39:
40:    printf("\n");
41:
42:  return 0;
43: }
44:
```

ÇIKTI

```
Once :   100  -250   400   125   550   900   689   450   347   700
Sonra:   900   700   689   550   450   400   347   125   100  -250
```

Kabarcık Sıralama (Bubble Sort):
Yanyana elemanları karşılaştırarak yer değiştir biçimde sıralamadır. Algoritmanın uygulaması Program 10.5'de gösterilmiştir.
Bu algoritmanın karmaşıklığı: $(n-1)^2$ dir.

Program 10.5: *Kabarcık Sıralama (Bubble Sort) Algoritması*

```
01: /* 09prg05.c
02:    Kabarcık Sıralama (Bubble Sort) Algoritması ile bir
03:    dizinin elemanlarını büyükten küçüğe dogru sıralar */
04:
05: #include <stdio.h>
06:
07: #define n 10
08:
09: int main(void)
10: {
11:    int   a[n] = {100, -250, 400, 125 ,550, 900, 689, 450, 347, 700};
12:    int   j,k,gecici;
13:
14:    /* Dizinin kendisi */
15:    printf("Once : ");
16:    for(k=0; k<n; k++)
17:       printf("%5d ",a[k]);
18:
19:    /* Sırala */
20:    for(k=0; k<n-1; k++)
21:    for(j=0; j<n-1; j++)
22:       if( a[j]<a[j+1] ){
23:          gecici = a[j];
24:            a[j] = a[j+1];
25:          a[j+1] = gecici;
26:       }
27:
28:    /* Sıralama bitti */
29:    printf("\nSonra: ");
30:    for(k=0; k<n; k++)
31:       printf("%5d ",a[k]);
32:
33:    printf("\n");
34:
35:   return 0;
36: }
```

ÇIKTI

```
Once :   100  -250   400   125   550   900   689   450   347   700
Sonra:   900   700   689   550   450   400   347   125   100  -250
```

10.5 Karakter Dizileri (Strings)

C dilinde, karakter dizileri oldukça sık kullanılır. Sadece karakter dizilerine özel olarak, karakter dizilerinin sonuna sonlandırıcı karakter olarak adlandırılan bir simge eklenir. Sonlandırıcı karakter, işlemlerin hızlı ve etkin bir biçimde yapılabilmesine olanak sağlar[2].

Sonlandırıcı karakter:

- dizinin bittiği yeri gösterir,
- ASCII tablosunun sıfır numaralı (`'\0'`) karakteridir.

Karakter dizilerine iki şekilde başlangıç değeri verilebilir.

```
char s[7] = {'d','e','n','e','m','e','\0'};
```

yada

```
char s[7] = "deneme";
```

Birinci tanımlamada sonlandırıcı karakter programcı tarafından konmalıdır. Ikinci tanımlamada ise buna gerek yoktur. Çünkü, sonlandırıcı karakter bu atamayla, derleyici tarafından eklenir.

NOT

```
char s[7] = "deneme";
```

ataması geçeli olmasına rağmen, aşağıdaki atama geçersizdir:

```
char s[7];
char s = "deneme";
```

Karakter dizileri `gets()` fonksiyonu ile klavyeden okunabilir.

```
char ad[20];
...
gets(ad);
```

Karakter dizleri veya katarlar Bölüm 12'de daha ayrıntılı işlenecektir. Burada sadece iki basit örnek sunulmuştur. Program 10.6'da bir karakter dizisinin uzunluğunun nasıl bulunduğu, Program 10.7'de ise bir karakter dizisinin tersyüz edilişi gösterilmiştir. İnceleyiniz.

Program 10.6: *Bir karakter dizisinin uzunluğunu bulur*

```
01: /* 09prg06.c: Bir karakter dizisinin uzunluğunu bulur */
02:
03: #include <stdio.h>
04:
05: int main(void)
06: {
07:    char s[40];
08:    int  k = 0;
09:
10:    /* diziyi oku */
11:    printf("Bir seyler yazin : ");
12:    gets(s);
13:
14:    /* sonlandırıcı karaktere kadar karakterleri say */
15:    while( s[k]!='\0' )
16:       k++;
17:
18:    printf("Dizinin uzunlugu : %d\n",k);
19:
20:  return 0;
21: }
```

ÇIKTI

```
Birseyler yazin  : Gazintep Universitesi
Dizinin uzunlugu : 21
```

Program 10.7: *Bir karakter dizisinin tersini bulur*

```
01: /* 09prg07.c: Bir karakter dizisini tersyüz eder */
02:
03: #include <stdio.h>
04:
05: int main(void)
06: {
07:    char s[40], gecici;
08:    int  i, n;
09:
10:    /* diziyi oku */
11:    printf("Bir seyler yazin : ");
12:    gets(s);
13:
14:    /* sonlandırıcı karaktere kadar */
15:    for(n=0; s[n] != '\0'; n++)
16:       ;
17:
```

```
18:     for(i=0; i<n/2; i++){
19:        gecici   = s[n-i-1];
20:        s[n-i-1] = s[i];
21:        s[i]     = gecici;
22:     }
23:
24:     printf("Tersi            : %s\n",s);
25:
26:   return 0;
27:  }
```

ÇIKTI

```
Bir seyler yazin : Deneme
Tersi            : emeneD
```

10.6 Çok Boyutlu Diziler

Bir dizi aşağıdaki gibi bildirildiğinde bir boyutlu (tek indisli) dizi olarak adlandırılır. Bu tip dizilere *vektör* denir.

```
float a[9];
```

Bir dizi birden çok boyuta sahip olabilir. Örneğin iki boyutlu `b` dizisi şöyle tanımlanabilir:

```
float b[9][4];
```

İki boyutlu diziler *matris* olarak adlandırılır. ilk boyuta *satır*, ikinci boyuta *sütün* denir. Yukarıda `b` matrisinin eleman sayısı 9x4=36 dır. Bu durumda, genel olarak bir dizi şöyle gösterilir:

Tablo 10.1: *Dizlerin Bildirimi*

Dizi Çeşiti	Genel Bildirimi	Örnek
Tek boyutlu diziler (Vektörler)	*tip dizi_adı*[*eleman_sayısı*]	`int veri[10];`
İki boyutlu diziler (Matrisler)	*tip dizi_adı*[*satır_sayısı*][*sutun_sayısı*]	`float mat[5][4];`
Çok boyutlu diziler	*tip dizi_adı*[*boyut_1*][*boyut_2*]...[*boyut_n*];	`double x[2][4][2];`

Çok boyutlu diziler tek boyuta indir generek bellekte tutulurlar. Tek indisli dizilerde olduğu gibi, çok indisli dizilere de başlangıç değeri vermek mümkün. Örneğin 3 satır ve 4 sütünlu (3x4=12 elemanlı) bir `x` matrisinin elemanları şöyle tanımlanabilir:

```
int x[3][4] = {11,34,42,60, 72,99,10,50,  80,66,21,38};
```

yada

```
int x[3][4] = {11,34,42,60,  /* 1. satır elemanları */
               72,99,10,50,  /* 2. satır elemanları */
               80,66,21,38}; /* 3. satır elemanları */
```

Bu matris ekrana matris formunda yazılmak istendiğinde:

```
 for(i=0; i<3; i++)
 {
   for(j=0; j<4; j++)
      printf("%4d",x[i][j]);

   printf("\n");
 }
```

çıktısı:

```
  11   34   42   60
  72   99   10   50
  80   66   21   38
```

şeklinde olacaktır.

Program 10.8, iki matrisin toplamını başka bir matrise aktarır. Matris toplamı $c_{ij} = a_{ij} + b_{ij}$ formülü ile tanımlıdır. İnceleyiniz.

Program 10.8: *İki matrisin toplamı*

```
01: /* 09prg08.c: iki matrisin toplamı */
02:
03: #include <stdio.h>
04:
05: #define SAT 2
06: #define SUT 3
07:
08: int main()
09: {
10:     int a[SAT][SUT] = {5, 3, 7,  0, 1, 2};
11:
12:     int b[SAT][SUT] = {1, 2, 3,  4, 5, 6};
13:     int c[SAT][SUT];
14:     int i, j;
15:
16:     puts("A Matrisi:");
17:     for(i=0; i<SAT; i++){
18:       for(j=0; j<SUT; j++)
19:          printf("%4d",a[i][j]);
20:       printf("\n");
21:     }
22:
23:     puts("B Matrisi:");
24:     for(i=0; i<SAT; i++){
25:       for(j=0; j<SUT; j++)
26:          printf("%4d",b[i][j]);
27:       printf("\n");
28:     }
29:
30:     puts("\nC Matrisi:");
31:     for(i=0; i<SAT; i++){
32:        for(j=0; j<SUT; j++){
33:          c[i][j] = a[i][j] + b[i][j];
34:          printf("%4d",c[i][j]);
35:        }
36:        printf("\n");
37:     }
38:
39:   return 0;
40: }
```

ÇIKTI

```
A Matrisi:
   5   3   7
   0   1   2
B Matrisi:
   1   2   3
   4   5   6

C Matrisi:
   6   5  10
   4   6   8
```

Program 10.9, iki kare matrisin çarpımı başka bir matrise aktarır. Matris çarpımı $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$ formülü ile tanımlıdır.

Program 10.9: *İki matrisin çarpımı*

```
01: /* 10prg09.c: 3x3 boyutundaki iki kare matrisin çarpımı */
02:
03: #include <stdio.h>
04:
05: #define N  3
06:
```

```
07:  int main()
08:  {
09:      int a[N][N], b[N][N], c[N][N];
10:      int i,j,k,toplam;
11:
12:      puts("A Matrisini girin:");
13:      for(i=0; i<N; i++)
14:         for(j=0; j<N; j++)
15:            scanf("%d",&a[i][j]);
16:
17:      puts("B Matrisini girin:");
18:      for(i=0; i<N; i++)
19:         for(j=0; j<N; j++)
20:            scanf("%d",&b[i][j]);
21:
22:
23:      puts("\nC Matrisi:");
24:      for(i=0; i<N; i++){
25:         for(j=0; j<N; j++){
26:
27:            for(toplam=0, k=0; k<N; k++)
28:               toplam += a[i][k]*b[k][j];
29:
30:            c[i][j] = toplam;
31:            printf("%4d",c[i][j]);
32:         }
33:         printf("\n");
34:      }
35:
36:   return 0;
37:  }
```

ÇIKTI

```
A Matrisini girin:
5   3   7
0   1   2
9   0   4
B Matrisini girin:
1   2   3
4   5   6
7   8   9

C Matrisi:
  66  81  96
  18  21  24
  37  50  63
```

10.7 Dizilerin Fonksiyonlarda Kullanılması

Diziler de sıradan değişkenler gibi bir fonksiyona parametere olarak akratılabilir. Fakat, aktarma kuralı biraz farklıdır. Her zaman dizinin yanında boyutunun da bilinmesi gerekir.

Program 10.10'da, bir dizinin elemanlarının yazdırılması işi bir fonksiyona yaptırılmıştır. Fonkisyona parametre olarak dizinin yanında boyutu da ilave edilmiştir. İnceleyiniz.

Program 10.10: *Bir dizinin yazdırılması*

```
01:  /* 10prg10.c: bir dizinin yazdırılması */
02:
03:  #include <stdio.h>
04:
05:  void dizi_yaz(float x[], int n);
06:
07:  int main(){
08:
09:     float kutle[5]= { 8.471, 3.683, 9.107, 4.739, 3.918 };
10:
11:     dizi_yaz(kutle, 5);
```

```
12:
13:     return 0;
14:  }
15:
16:  void dizi_yaz(float x[], int n)
17:  {
18:     int i;
19:
20:     for(i=0; i<n; i++)
21:        printf("%7.3f", x[i]);
22:
23:     printf("\n");
24:  }
```

ÇIKTI

```
 8.471  3.683  9.107  4.739  3.918
```

Eğer dizi boyutu `#define` önişlemcisi ile belirtilirse boyutun ayrıca parametre olarak kullanılmasına gerek youktur. Bu durumda Program 10.10 şöyle değiştirlebilir:

```
...
#define BOYUT 5
void dizi_yaz(float x[]);

void main(void)
{
  float kutle[BOYUT]= { 8.471, 3.683, 9.107, 4.739, 3.918 };
  dizi_yaz(kutle);
}
...
```

Program 10.3'de bir dizinin en büyük elemanının nasıl bulunduğu gösterilmişti. En büyük elemanı bulma işlemi bir fonksiyona nasıl yaptırıldığı Program 10.11'de gösterilmiştir.

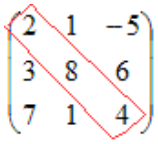
Program 10.11: *Bir dizinin en büyük elemanının bulunması*

```
01: /* 10prg11.c
02:    Bir dizinin en büyük elemanının fonksiyonla bulunması */
03:
04: #include <stdio.h>
05:
06: /* n elemanlı bir dizinin enbüyük elemanını gönderir */
07: int enBuyuk(int a[], int n)
08: {
09:    int k, en_buyuk_eleman;
10:
11:    /* ilk eleman en büyük kabul ediliyor */
12:    en_buyuk_eleman = a[0];
13:
14:    for(k=1; k<n; k++)
15:       if( a[k]>en_buyuk_eleman )
16:           en_buyuk_eleman = a[k];
17:
18:    return en_buyuk_eleman;
19: }
20:
21: int main()
22: {
23:    int   a[10] = {100, -250, 400, 125 ,550, 900, 689, 450, 347, 700};
24:    int   eb;
25:
26:    eb = enBuyuk(a,10);
27:
28:    printf("En buyuk eleman = %d\n",eb);
29:
30:  return 0;
31: }
```

ÇIKTI

```
En buyuk eleman = 900
```

Son olarak, bir kare matrisin iz (trace) değerini bulup ekrana yazan bir fonksiyon Program 10.12'de verilmişitir. Bir kare matrisin izi, matrisin asal köşegen üzerinde bulunan elemanların toplamı olarak tanımlıdır. Bu tanıma göre, aşağıdaki matrisin izi 2 + 8 + 4 = 14 tür..



İz matematiksel olarak şöyle gösterilir:

$$\text{İz}(\mathbf{A}) = a_{11} + a_{22} + \cdots + a_{nn} = \sum_{k=1}^{n} a_{ii}$$

Program 10.12: *Bir matrisin izi*

```
01: /* 10prg12.c
02:    Bir 3x3 bir matrisin izinin fonksiyonla bulunması */
03:
04: #include <stdio.h>
05:
06: double iz(double a[][3], int);
07:
08: int main()
09: {
10:    double a[3][3], izA;
11:    int i,j;
12:
13:    puts("matrisi girin:");
14:    for(i=0; i<3; i++)
15:    for(j=0; j<3; j++)
16:       scanf("%lf",&a[i][j]);
17:
18:    izA = iz(a,3);
19:
20:    printf("matrisin izi = %lf\n",izA);
21:
22:  return 0;
23: }
24:
25: double iz(double a[][3], int n)
26: {
27:    int i;
28:    double toplam = 0.0;
29:
30:
31:    for(i=0; i<n; i++)
32:        toplam += a[i][i];
33:
34:    return toplam;
35: }
36:
37:
```

ÇIKTI

```
matrisi girin:
2  1  -5
3  8  6
7  1  4
matrisin izi = 14.000000
```

Matrisler, fonksiyonlara parametre olarak geçirilirken ikinci boyununda verildiğine dikkat edin.

Powered by PHP