Bölüm 6 - Diziler

İçerik

6.1 Giriş

- Diziler
  - Aynı türden veri yapıları
  - Static değer – program boyunca aynı boyut
  - Dinamik veri yapısı (Bölüm 13)
- Dizi
  - Ardarda bellek yerleşimli grup
  - Aynı isim ve tip
- Bir elemanı belirlemek için
  - Dizi adı
  - Yer numarası (indis)
- Format:

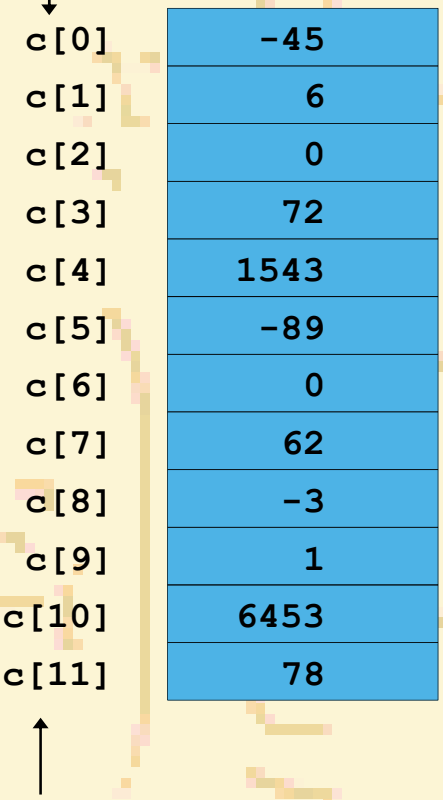
  diziadı **[** *yer numarası* **]**
  - İlk elemanın yeri (indisi) **0**
  - **n** elemanlı **c** adlı dizi:
    - **c[0], c[1]...c[n - 1]**
- Dizi elemanları normal değişkenler gibidir

```
c[ 0 ] =  3;
printf( "%d", c[ 0 ] );
```

  - İndislerle işlem yapılabilir. Eğer **x** eşit **3 ise**

    c[5 - 2], c[3], c[x] aynı değerdir

Dizi adı (dizinin her elemanı aynı adı taşıyor, **c**)



C dizisinin indisleri

6.2 Dizi Deklarasyonu

- Diziyi tanımlarken belirtilecekler:
  - Adı
  - Dizinin tipi
  - Eleman sayısı (boyutu)

```
diziTipi diziAdı[ elemanSayısı ];
```

  - Örnek:

```
int c[ 10 ];
float xdizisi[ 3284 ];
```

- Aynı tipten birden çok dizi
  - Normal değişkenlere benzer format
  - Örnek:

```
int b[ 100 ], x[ 27 ];
```

6.3 Dizi Örnekleri

- Tanımlama ve ilk atama

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

  - Boyuttan az eleman verilmişse geriye kalanlar **0** alınır

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

    - Tüm elemanlar 0
  - Boyuttan fazla eleman atanırsa yazılım hatası üretilir
  - C dizileri sınır kontrolüne sahip değildir
- Eğer boyut verilmezse, atamada belirlenir

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

  - n, 5 elemanlı bir dizidir

6.3 Dizi Örnekleri

```
1  /* Örnek 6.8.
2     Histogram yazma programı */
3  #include <stdio.h>
4  #define BOYUT 10
5
6  int main()
7  {
8     int n[ BOYUT ] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
9     int i, j;
10
11    printf( "%s%13s%17s\n", "Eleman", "Değer", "Histogram" );
12
13    for ( i = 0; i <= BOYUT - 1; i++ ) {
14       printf( "%7d%13d          ", i, n[ i ]) ;
15
16       for ( j = 1; j <= n[ i ]; j++ )   /* Bir bar yaz */
17          printf( "%c", '*' );
18
19       printf( "\n" );
20    }
21
22    return 0;
23 }
```

6.3 Dizi Örnekleri

Çıktı:

```
Eleman        Değer         Histogram
  0             19          ********************
  1              3          ***
  2             15          ***************
  3              7          *******
  4             11          ***********
  5              9          *********
  6             13          *************
  7              5          *****
  8             17          *****************
  9              1          *
```

6.3 Dizi Örnekleri

- Karakter dizileri (string-katar)
  - "**fizan**" kelimesi bir statik karakter dizisidir
  - Karakter dizileri aslında string harfleri kullanılarak belirlenebilir

```
char katar1[] = "fizan";
```

    - Null karakteri '\0' katarı sonlandırır
    - **katar1** aslında 6 elemanlıdır
      - Denk olarak;

```
char katar1[] = { 'f', 'i', 'z', 'a', 'n', '\0' };
```

  - Bireysel karakterlere erişilebilir

```
katar1[2], 'z' karakterini gösterir
```

  - Dizi adı, dizinin adresidir, bu nedenle **scanf** de & işaretine gerek yok

```
scanf("%s", katar2);
```

    - Whitespace(boşluk karakteri) gelinceye kadar karakter okur
    - Dizinin sonrasını da yazabileceğinden dikkatli olunmalıdır

6.3 Dizi Örnekleri

```
1  /* Fig. 6.10: fig06_10.c
2     Karakter dizilerini katar olarak işlemek */
3  #include <stdio.h>
4
5  int main()
6  {
7     char katar1[ 20 ], katar2[] = "katar harfleri";
8     int i;
9
10    printf(" Bir katar gir: ");
11    scanf( "%s", katar1 );
12    printf( "katar1 : %s\nkatar2 : %s\n"
13            "Karakterler arasında boşluklu katar1:\n",
14            katar1, katar2 );
15
16    for ( i = 0; katar1[ i ] != '\0'; i++ )
17       printf( "%c ", katar1[ i ] );
18
19    printf( "\n" );
20    return 0;
21 }
```

Çıktı:

```
Bir katar gir: Herkese merhaba
katar1 : Herkese
katar2 : katar harfleri
Karakterler arasında boşluklu
katar1:
H e r k e s e
```

6.4 Diziyi Fonksiyona Aktarma

- Diziyi aktarma
  - Bir dizi argümentini bir fonksiyona aktarmak için, dizi adını parantezsiz olarak gir

```
int dizi[ 24 ];
fonksiyon( dizi, 24 );
```

    - Dizi boyutu genellikle fonksiyona geçer
  - Diziler, refereansla-çağırma ile geçer
  - Dizinin adı, ilk elemanın adresidir
  - Fonksiyon dizinin bellekte nereye yerleştiğini bilir
    - Orjinal bellek yerini değiştirir
- Dizi elemanlarını aktarma
  - Değer-Çağırma ile geçer
  - İndis adı (**`dizi[ 3 ] gibi`**) fonksiyona aktarılır

Fonksiyon prototipi	• **void dizi(int b[], int diziBoyutu);**
Prototipte parametre adları opsiyoneldir	• **int b[]** yerine **int []** yazılabilir • **int diziBoyutu** yerine basitçe **int** yazılabilir

6.4 Diziyi Fonksiyona Aktarma

```
1  /* Örnek 6.13
2     Dizivi ve dizi elemanlarını fonksivona aktarma */
3  #include <stdio.h>
4  #define BOYUT 5
5
6  void dizi( int [], int );  /* tuaf görünüvor */
7  void eleman( int );
8
9  int main()
10 {
11    int a[ BOYUT ] = { 0, 1, 2, 3, 4 }, i;
12
13    printf( "Tüm dizivi referansla-cağırma "
14            "ile aktarma etkisi:\n\n Orjinal dizinin "
15            "değerleri:\n" );
16
17    for ( i = 0; i <= BOYUT - 1; i++ )
18       printf( "%3d", a[ i ] );
19
20    printf( "\n" );
21    dizi( a, BOYUT );  /* ref-cağrı ile aktarıldı */
22    printf( "Değistirilmis dizinin değerleri:\n" );
23
24    for ( i = 0; i <= BOYUT - 1; i++ )
25       printf( "%3d", a[ i ] );
26
27    printf( "\n\n\nDizi elemanını değerle-cağırma ile aktarma"
28            "etkisi:\n\n a[3]'ün değeri %d\n", a[ 3 ] );
29    eleman( a[ 3 ] );
30    printf( "a[ 3 ]'ün değeri%d\n", a[ 3 ] );
31    return 0;
32 }
```

Tüm dizi referansla-çağırma ile geçer ve değiştirilebilir

Elemanlar değerle-çağırma ile geçer ve değiştirilemez

6.4 Diziyi Fonksiyona Aktarma

```
33
34 void dizi( int b[], int boyut )
35 {
36    int j;
37
38    for ( j = 0; j <= boyut - 1; j++ )
39       b[ j ] *= 2;
40 }
41
42 void eleman( int e )
43 {
44    printf( "Eleman fonksiyonundaki değer: %d\n", e *= 2 );
45 }
```

```
Tüm diziyi referansla-çağırma ile aktarma etkisi:

Orjinal dizinin değerleri:
  0  1  2  3  4
Değiştirimiş dizinin değerleri:
  0  2  4  6  8

Dizi elemanını değerle-çağırma ile aktarma etkisi:

a[3] ün değeri 6
Eleman fonksiyonundaki değer: 12
a[3] ün değeri 6
```

6.5 Sıralama

- Verileri sıralama
  - Önemli bir uygulama
  - Hemen hemen her yerde veriler sıralanmalıdır
- Balon sıralaması
  - Diziyi birçok kez tara
  - Ardışık elemanları karşılaştır
    - Artan (veya azalan) sıra ise, değişiklik yapma
    - Azalan (veya artan) sıra ise, yer değiştir
  - Tekrar et
- Örnek:
  - orjinal: 3 4 2 6 7
  - 1. tarama: 3 2 4 6 7
  - 2. tarama: 2 3 4 6 7
  - Küçük elemanlar öne geçirilir (balonlanır)

6.6 Örnek Çalışma: Ortalama, Medyan ve Mod Hesabı

- Ortalama
- Medyan – sıralı listenin orta değeri
  - 1, 2, 3, 4, 5
  - 3 medyandır
- Mod – en çok görünen sayı
  - 1, 1, 1, 2, 3, 3, 4, 5
  - 1 moddur

6.6 Örnek Çalışma: Ortalama, Medyan ve Mod Hesabı

```
/* Örnek 6.16
   Veri analizi.
   verilerin ortalaması, medyanı ve modu */
#include <stdio.h>
#define BOYUT 99

void ort( const int [] );
void medyan( int [] );
void mod( int [], const int [] ) ;
void balon( int [] );
void dizivaz( const int [] );

int main()
{
   int tekrar[ 10 ] = { 0 };
   int cevap[ BOYUT ] =
      { 6, 7, 8, 9, 8, 7, 8, 9, 8, 9,
        7, 8, 9, 5, 9, 8, 7, 8, 7, 8,
        6, 7, 8, 9, 3, 9, 8, 7, 8, 7,
        7, 8, 9, 8, 9, 8, 9, 7, 8, 9,
        6, 7, 8, 7, 8, 7, 9, 8, 9, 2,
        7, 8, 9, 8, 9, 8, 9, 7, 5, 3,
        5, 6, 7, 2, 5, 3, 9, 4, 6, 4,
        7, 8, 9, 6, 8, 7, 8, 9, 7, 8,
        7, 4, 4, 2, 5, 3, 8, 7, 5, 6,
        4, 5, 6, 1, 6, 5, 7, 8, 7 };

   ort( cevap );
   medyan( cevap );
   mod( tekrar, cevap );
   return 0;
}

void ort( const int yanit[] )
{
   int j, toplam = 0;
```

6.6 Örnek Çalışma: Ortalama, Medyan ve Mod Hesabı

```
38    printf( "%s\n%s\n%s\n", "********", "  Ortalama", "********" );
39
40    for ( i = 0; i <= BOYUT - 1; i++ )
41       toplam += yanit[  i  ];
42
43    printf( "Oralama, bir veri grubunun toplamının \n"
44
45            "veri savısına ( %d ) bölümüdür\n"
46            "Bu örneklemin \n"
47            "ortalaması: %d / %d = %.4f\n\n",
48            BOYUT, toplam, BOYUT, (  double  ) toplam / BOYUT );
49 }
50
51 void medyan( int yanit[] )
52 {
53    printf( "\n%s\n%s\n%s\n%s",
54            "********", " Medyan", "********",
55            "Cevapların sıralanmamıs dizisi:" );
56
57    dizivaz( yanit );
58    balon( yanit );
59    printf( "\n\nSıralı dizi:" );
60    dizivaz( yanit );
61    printf( "\n\n Medyan, %d elemanlı \n"
62            "sıralı dizinin %d.elemanıdır.\n"
63            "Bu örneklemin medyanı: %d\n\n",
64             BOYUT, BOYUT / 2, yanit[ BOYUT / 2 ] );
```

6.6 Örnek Çalışma: Ortalama, Medyan ve Mod Hesabı

```
65 void mod( int tek[], const int yanit[] )
66 {
67    int oran, j, h, enbuyuk = 0, moddeger = 0;
68
69    printf( "\n%s\n%s\n%s\n",
70            "*********", "  Mod", "*********" );
71
72    for ( oran = 1; oran <= 9; oran++ )
73       tek[ oran ] = 0;
74
75    for ( j = 0; j <= BOYUT - 1; j++ )
76       ++tek[ yanit[ j ] ];
77
78    printf( "%s%11s%19s\n\n%54s\n%54s\n\n",
79            "Cevap", "Tekrar", "Histogram",
80            "1    1    2    2", "5    0    5    0    5" );
81
82    for ( oran = 1; oran <= 9; oran++ ) {
83       printf( "%8d%11d          ", oran, tek[ oran] );
84
85       if ( tek[ oran ] > enbuyuk ) {
86          enbuyuk = tek[ oran ];
87          moddeger = oran;
88       }
89       for ( h = 1; h <= tek[ oran ]; h++ )
90          printf( "*" );
91       printf( "\n" );
92    }
93    printf( "Mod,
94     en çok tekrarlanan
95  "Bu örneklem için mod %d olup"
96  " %d kez tekrarlanmıştır.\n",
97    moddeger, enbuyuk );
99 }
```

tekrar[] daki indisin nasıl **cevap[]** (**yanit[]**) daki eleman olduğuna dikkat

tekrar[] ın değerine göre yıldız yaz

6.6 Örnek Çalışma: Ortalama, Medyan ve Mod Hesabı

```
100 void balon( int a[] )
101 {
102   int gec, j, tut;
103
104   for ( gec = 1; gec <= BOYUT - 1; gec++ )
105
106      for ( j = 0; j <= BOYUT - 2; j++ )
107
108         if ( a[ j ] > a[ j + 1 ] ) {
109            tut = a[ j ];
110            a[ j ] = a[ j + 1 ];
111            a[ j + 1 ] = tut;
112         }
113 }
114
115 void diziyaz( const int a[] )
116 {
117   int j;
118
119   for ( j = 0; j <= BOYUT - 1; j++ ) {
120
121      if ( j % 20 == 0 )
122         printf( "\n" );
123      printf( "%2d", a[ j ] );
124   }
125 }
```

Balon sıralaması

6.6 Örnek Çalışma: Ortalama, Medyan ve Mod Hesabı

```
********
Ortalama
********
Ortalama, bir veri grubunun toplamının
veri sayısına (99) bölümüdür
Bu örneklemin
ortalaması: 681 / 99 = 6.8788

********
 Medyan
********
Cevapların sıralanmamış dizisi:
7 8 9 8 7 8 9 8 9 7 8 9 5 9 8 7 8 7 8
 6 7 8 9 3 9 8 7 8 7 7 8 9 8 9 8 9 7 8 9
 6 7 8 7 8 7 9 8 9 2 7 8 9 8 9 8 9 7 5 3
 5 6 7 2 5 3 9 4 6 4 7 8 9 6 8 7 8 9 7 8
 7 4 4 2 5 3 8 7 5 6 4 5 6 1 6 5 7 8 7

Sıralı dizi:
 1 2 2 2 3 3 3 3 4 4 4 4 4 5 5 5 5 5 5 5
 5 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9

Medyan 99 elemanlı
Sıralı dizinin 49. elemanıdır.
Bu örneklem için medyan: 7
```

6.6 Örnek Çalışma: Ortalama, Medyan ve Mod Hesabı

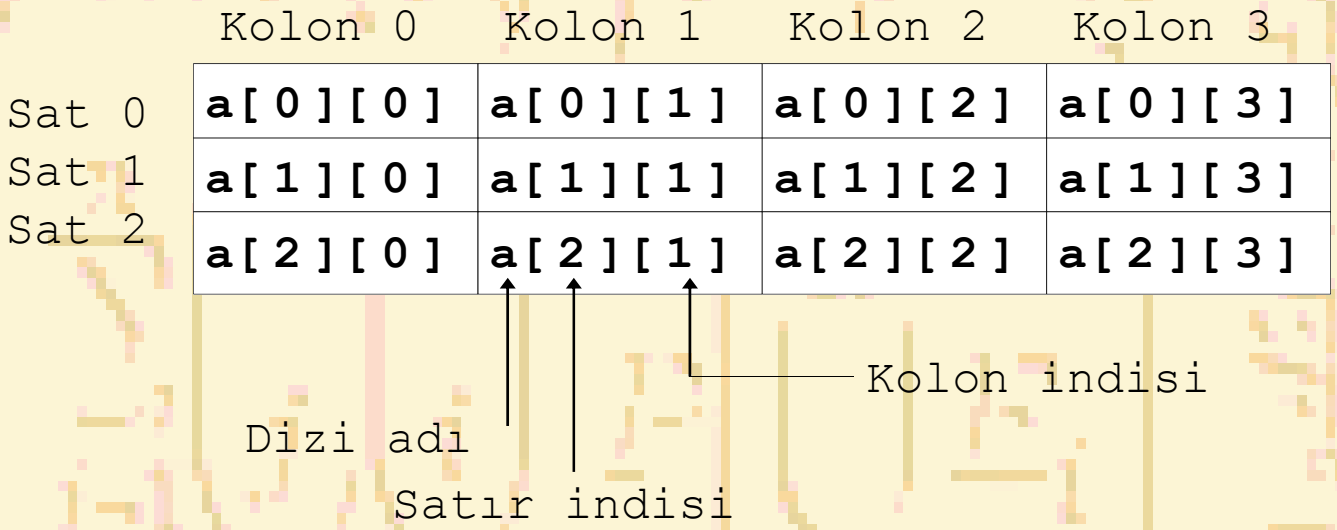
```
********
  Mode
********
   Cevap      Tekrar         Histogram

                                    1    1    2    2
                               5    0    5    0    5

      1           1           *
      2           3           ***
      3           4           ****
      4           5           *****
      5           8           ********
      6           9           *********
      7          23           ***********************
      8          27           ***************************
      9          19           *******************
Mod, en çok tekrarlanan değerdir.
Bu örneklem için mod 8 olup 27 kez tekrarlanmıştır.
```

6.7 Dizi Tarama: Lineer ve İkili Tarama

- *anahtar değer* ile dizi tarama
- Lineer tarama
  - Basit
  - Dizinin her elemanını bir anahtar değer ile tara
  - Küçük ve sıralı olmayan dizilerde kullanışlı

- İkili tarama
  - Sıralı diziler için
  - Bir anahtar ile orta elemanı karşılaştırır
    - eşitse, bulunmuş olur
    - Eğer **`anahtar < orta`**
    - Eğer **`anahtar > orta`** ise dizinin diğer yarısına bak
    - Tekrarla
  - Çok hızlı; 2^n > eleman sayısı için, en fazla n adım
    - 30 elemanlı bir dizi için en fazla 5 adım
      - $2^5 > 30$ olduğundan en fazla 5 adım

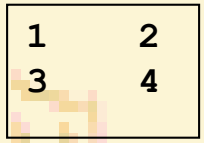
- Çoklu indisli dizi
  - Satır ve kolonlu tablolar (**m** x **n** dizisi)
  - Matrisler gibi: Önce satır, sonra kolon belirlenir



6.8 Çoklu-İndisli Diziler

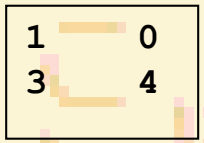
- İlk atama
  - `int b[ 2 ][ 2 ] = { { 1, 2 }, { 3, 4 } };`
  - Satır satır gruplanır
  - Yeterli eleman verilmemişse, sıfır kabul edilir

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



- Elemanı referans verme
  - Önce satırı sonra kolonu belirle

```
printf( "%d", b[ 0 ][ 1 ] );
```



6.8 Çoklu-İndisli Diziler

```
// Örnek 6.22   İki indisli dizi
   #include <stdio.h>
   #define OGRENCI 3
   #define SINAVLAR 4
   int minimum( const int [][ SINAVLAR ], int, int );
   int maximum( const int [][ SINAVLAR], int, int );
   double ortalama( const int [], int );
void diziyaz( const int [][ SINAVLAR ], int, int );

   int main() {
      int ogr;
   const int notlar[ OGRENCI ][ SINAVLAR ] =
      { { 77, 68, 86, 73 },
        { 96, 87, 89, 78 },
        { 70, 90, 86, 81 } };
      printf( "Dizi:\n" );

      diziyaz( notlar, OGRENCI, SINAVLAR );
      printf( "\n\nEn düşük not: %d\nEn yüksek not: %d\n",
              minimum( notlar, OGRENCI, SINAVLAR ),
           maximum( notlar, OGRENCI, SINAVLAR ) );

      for ( ogr = 0; ogr <= OGRENCI - 1; ogr++ )
         printf( "%d. Öğrencinin not ortalaması: %.2f\n",
                 ogr,
              ortalama( notlar[ ogr ], SINAVLAR ) );

   return 0;
   }
```

Her bir satır öğrenci nosunu ve kolon da notu belirtir

```
33
34 /* En düşük notu bul */
35 int minimum( const int not[][ SINAVLAR ],
36              int P, int test )
37 {
38    int i, j, dnot = 100;
39
40    for ( i = 0; i <= p - 1; i++ )
41       for ( j = 0; j <= test - 1; j++ )
42          if ( not[ i ][ j ] < dnot )
43             dnot = not[ i ][ j ];
44
45    return dnot;
46 }
47
48 /* En yüksek notu bul */
49 int maximum( const int not[][ SINAVLAR ],
50              int p, int test )
51 {
52    int i, j, ynot = 0;
53
54    for ( i = 0; i <= p - 1; i++ )
55       for ( j = 0; j <= test - 1; j++ )
56          if ( not[ i ][ j ] > dnot )
57             dnot = not[ i ][ j ];
58
59    return dnote;
60 }
61
62 /* Bir sınavın ortalaması */
63 double ortalama( const int snot[], int test )
64 {
```

6.8 Çoklu-İndisli Diziler

```
65    int i, top = 0;
66
67    for ( i = 0; i <= test - 1; i++ )
68       top += snot[ i ];
69
70    return ( double ) top / test;
71 }
72
73 /* Diziyi yaz */
74 void diziyaz( const int not[][ SINAVLAR ],
75                  int p, int test )
76 {
77    int i, j;
78
79    printf( "                [0]  [1]  [2]  [3]" );
80
81    for ( i = 0; i <= p - 1; i++ ) {
82       printf( "\nNOTLAR[%d] ", i );
83
84       for ( j = 0; j <= test - 1; j++ )
85          printf( "%-5d", not[ i ][ j ] );
86    }
87 }
```

6.8 Çoklu-İndisli Diziler

```
Dizi:
          [0]   [1]   [2]   [3]
NOTLAR[0] 77    68    86    73
NOTLAR[1] 96    87    89    78
NOTLAR[2] 70    90    86    81

En düşük not: 68
En yüksek not: 96
0. öğrencinin en düşük notu: 76.00
1. öğrencinin en düşük notu: 87.50
2. öğrencinin en düşük notu: 81.75
```