Bölüm 6 – Diziler (Arrays)

Outline

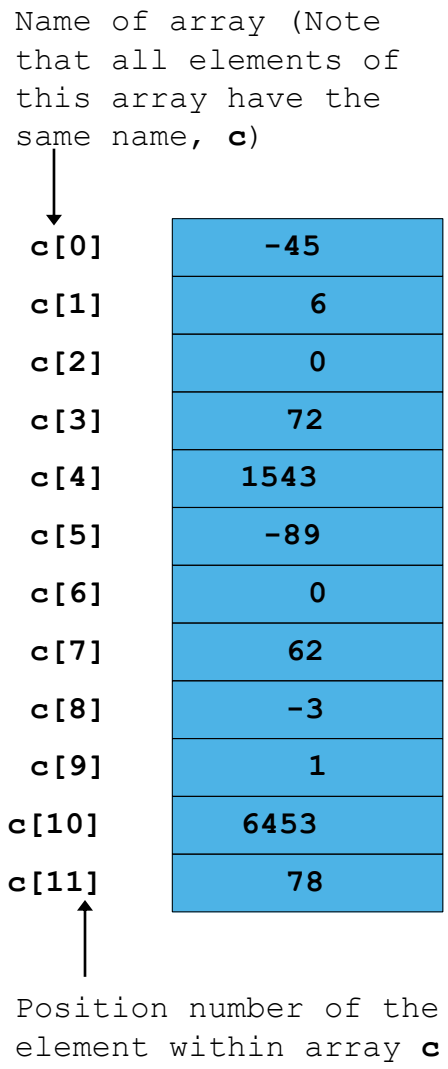
6.1 Giriş
6.2 Diziler
6.3 Dizileri Tanımlama
6.4 Dizi kullanımına örnekler
6.5 Fonksiyonlara Dizi göndermek
6.6 Dizileri sıralamak
6.7 Çalışmalar: Ortalama, Medyan ve mod hesabını dizilerle bulmak
6.8 Dizilerde arama yapmak
6.9 Çok boyutlu diziler

6.1 GİRİŞ

- *Diziler*
  - İlgili veri maddelerinin yapılarıdır.
  - Sabit girişli – tüm program boyunca aynı boyda
  - Dinamik veri yapıları 12. bölümde anlatıalcaktır.

6.2 Arrays

- Diziler
  - Birbirini takip eden grup hafıza yeridir
  - Aynı ad ve tip
- Bir elemanı belirtmek için
  - Dizi adı
  - ve pozisyon numarası kullanılır
- Biçim: *diziadı[pozisyon numarası]*
  - İlk elemanın pozisyonu 0 (sıfir)
  - n elemanlı **c** dizisi: **c[0], c[1]...c[n-1]**



6.2 Arrays (II)

- Dizi elemanları normal değişkenler gibidir.

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

  - Index numarasında işlem gerçekleşebilir . If **`x = 3`**,

```
c[5-2] == c[3] == c[x]
```

6.3 Dizileri Tanımlama

- Dizileri tanımlarken, şunlar belirtilir
  - Ad
  - Dizinin Tipi
  - Elementlerin sayısı

```
dizitipi diziadı[ elemansyısı ];
int c[ 10 ];
float myArray[ 3284 ];
```

- Aynı tipte çok dizi tanımlama
  - Normal değişkenler gibi tanımlanabilir

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

6.4 Dizilerde Örnekler

- Başlangıç tanımlarken verileri girmek

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

  - Eğer yeterli eleman yoksa en sağdaki eleman değeri 0 olur
  - Eğer fazla ise bir yazım hatası meydana gelir

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

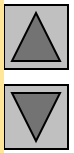
    - Tüm elemanlar sıfır 0
  - C dilinde dizilerin sınır kontrolü yoktur

- Eğer boyut belirtilmemişse başlangıçta atanan değer boyutu belirler

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

  - 5 başlangıç eğeri, bundan dolayı 5 dizi elamanı

```c
/* Fig. 6.8: fig06_08.c
   Histogram printing program */
#include <stdio.h>
#define SIZE 10

int main()
{
   int n[ SIZE ] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
   int i, j;

   printf( "%s%13s%17s\n", "Element", "Value", "Histogram" );

   for ( i = 0; i <= SIZE - 1; i++ ) {
      printf( "%7d%13d        ", i, n[ i ]) ;

      for ( j = 1; j <= n[ i ]; j++ )   /* print one bar */
         printf( "%c", '*' );

      printf( "\n" );
   }

   return 0;
}
```

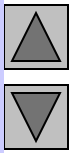


Outline

1. Diziyi başlat

2. Dön

3. Yazdır



Outline

Program Output

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

6.4 Dizilerde Örnekler (II)

- Karakter dizileri
  - String **"hello"** zaten **static** bir karakter dizisidir.
  - Karakter dizileri string değerler ile başlatılabilir.

    char string1[] = "first";
    - null karakter **'\0'** dizinin bitişini belirler
    - **string1** aslında 6 elemanı vardır.

```
char string1[] = { 'f', 'i', 'r', 's', 't', '\0' };
```

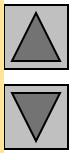
6.4 Dizilerde Örnekler (III)

- Karakter dizileri (devam)
  - Tek bir karaktere ulaşmak için
    - **`string1[ 3 ], 's'`** karakteridir
  - Dizi adları, dizilerin adresleridir. Bundan dolayı & **`scanf`** kullanırken gerekli değildir.

    `scanf( "%s", string2 ) ;`
    - Karakterleri boşluğa rastlayana kadar okur
    - Bir dizinin sonundan fazla yazabilir, dikkatli olunmalıdır.

```c
/* Fig. 6.10: fig06_10.c
   Treating character arrays as strings */
#include <stdio.h>

int main()
{
   char string1[ 20 ], string2[] = "string literal";
   int i;

   printf(" Enter a string: ");
   scanf( "%s", string1 );
   printf( "string1 is: %s\nstring2: is %s\n"
           "string1 with spaces between characters is:\n",
           string1, string2 );

   for ( i = 0; string1[ i ] != '\0'; i++ )
      printf( "%c ", string1[ i ] );

   printf( "\n" );
   return 0;
}
```

```
Enter a string: Hello there
string1 is: Hello
string2 is: string literal
string1 with spaces between characters is:
H e l l o
```



Outline

1. Stringleri başlat

2. Stringleri yazdır

2.1 döngü

2.2 karakterleri tek tek yazdır

2.3 string girdisi

3. string yazdır

Program Çıkışı

6.5 Fonksiyonlara Dizi Göndermek

- Dizileri Göndermek
  - Dizi adını parantezsiz belirt

```
int myArray[ 24 ];
myFunction( myArray, 24 );
```

    - Dizi büyüklüğü genellikle fonksiyona gönderilir.
  - Diziler referansları ile gönderilir
  - Dizinin adı ilk elemanın adresini içerir
  - Fonksiyon dizinin nerede saklandığını bilir.
    - Orijinal hafıza yerinde düzenleme yapar
- Dizi elemanlarını göndermek
  - Değer olarak gönderilir.
  - İndeks numarasını gönder (i.e., **`myArray[3]`**)

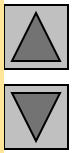
6.5 Dizileri fonksiyona göndermek (II)

- Prototip fonksiyon

```
void modifyArray( int b[], int arraySize );
```

  - `int b[]` sadece `int []` olabilir
  - `int arraySize` sadece `int` olabilir

```c
/* Fig. 6.13: fig06_13.c
   Passing arrays and individual array elements to functions */
#include <stdio.h>
#define SIZE 5

void modifyArray( int [], int );  /* appears strange */
void modifyElement( int );

int main()
{
   int a[ SIZE ] = { 0, 1, 2, 3, 4 }, i;

   printf( "Effects of passing entire array call "
           "by reference:\n\nThe values of the "
           "original array are:\n" );

   for ( i = 0; i <= SIZE - 1; i++ )
      printf( "%3d", a[ i ] );

   printf( "\n" );
   modifyArray( a, SIZE );  /* passed call by reference */
   printf( "The values of the modified array are:\n" );

   for ( i = 0; i <= SIZE - 1; i++ )
      printf( "%3d", a[ i ] );

   printf( "\n\n\nEffects of passing array element call "
           "by value:\n\nThe value of a[3] is %d\n", a[ 3 ] );
   modifyElement( a[ 3 ] );
   printf( "The value of a[ 3 ] is %d\n", a[ 3 ] );
   return 0;
}
```



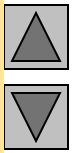
Outline

Entire arrays passed call-by-reference, and can be modified

Array elements passed call-by-value, and cannot be modified

```
33
34 void modifyArray( int b[], int size )
35 {
36    int j;
37
38    for ( j = 0; j <= size - 1; j++ )
39       b[ j ] *= 2;
40 }
41
42 void modifyElement( int e )
43 {
44    printf( "Value in modifyElement is %d\n", e *= 2 );
45 }
```

```
Effects of passing entire array call by reference:

The values of the original array are:
  0  1  2  3  4
The values of the modified array are:
  0  2  4  6  8

Effects of passing array element call by value:


The value of a[3] is 6
Value in modifyElement is 12
The value of a[3] is 6
```



Program Output

6.6 Dizileri Sıralamak

- Verileri Sıralamak
  - Önemli bir hesaplama uygulaması
  - Neredeyse her bir organizasyon verileri sıralamak zorunda
    - Çok büyük miktarlar mutlaka sıralanmalı
- Bubble sort (sinking sort)
  - Dizi boyunca bir çok geçiş var
  - Taranan çiftler arasında kıyaslama oluşur
    - Eğer artan sıralama ise, değişiklik yok
    - Eğer azalan sırada ise , elemanlar yer değiştirilir
  - Tekrar edilir
- Example:

  original: **3 4 2 6 7**

  pass 1: **3 2 4 6 7**

  pass 2: **2 3 4 6 7**
  - Small elements "bubble" to the top

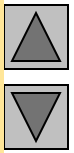
6.7 Ortalama Medyan ve mod hesaplamak

- Ortalama -
- Medyan – Sıralı bir dizinin en ortasındaki eleman Dizi eleman sayısı çift ise ortadaki iki elemanın aritmetik ortalamasıdır
  - 1, 2, 3, 4, 5

    3 is the median
- Mod – en sıklıkla karşılaşılan eleman
  - 1, 1, 1, 2, 3, 3, 4, 5

    1 is the mode

```c
/* Fig. 6.16: fig06_16.c
   This program introduces the topic of survey data analysis.
   It computes the mean, median, and  mode of the data */
#include <stdio.h>
#define SIZE 99

void mean( const int [] );
void median( int [] );
void mode( int [], const int [] ) ;
void bubbleSort( int [] );
void printArray( const int [] );

int main()
{
   int frequency[ 10 ] = { 0 };
   int response[ SIZE ] =
      { 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 };

   mean( response );
   median( response );
   mode( frequency, response );
   return 0;
}
```



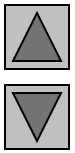
Outline

1. Function prototypes

1.1 Initialize array

2. Call functions `mean`, `median`, and `mode`

```
33
34 void mean( const int answer[] )
35 {
36    int j, total = 0;
37
38    printf( "%s\n%s\n%s\n", "********", "  Mean", "********" );
39
40    for ( j = 0; j <= SIZE - 1; j++ )
41       total += answer[  j  ];
42
43    printf( "The mean is the average value of the data\n"
44            "items. The mean is equal to the total of\n"
45            "all the data items divided by the number\n"
46            "of data items ( %d ). The mean value for\n"
47            "this run is: %d / %d = %.4f\n\n",
48            SIZE, total, SIZE, (  double  ) total / SIZE );
49 }
50
51 void median( int answer[] )
52 {
53    printf( "\n%s\n%s\n%s\n%s",
54            "********", " Median", "********",
55            "The unsorted array of responses is" );
56
57    printArray( answer );
58    bubbleSort( answer );
59    printf( "\n\nThe sorted array is" );
60    printArray( answer );
61    printf( "\n\nThe median is element %d of\n"
62            "the sorted %d element array.\n"
63            "For this run the median is %d\n\n",
64            SIZE / 2, SIZE, answer[ SIZE / 2 ] );
```



Outline

3. Define function `mean`

3.1 Define function `median`

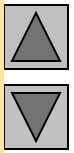
3.1.1 Sort Array

3.1.2 Print middle element

```
65 }
66
67 void mode( int freq[], const int answer[] )
68 {
69    int rating, j, h, largest = 0, modeValue = 0;
70
71    printf( "\n%s\n%s\n%s\n",
72            "********", "  Mode", "******** );
73
74    for ( rating = 1; rating <= 9; rating++ )
75       freq[ rating ] = 0;
76
77    for ( j = 0; j <= SIZE - 1; j++ )
78       ++freq[ answer[ j ] ];
79
80    printf( "%s%11s%19s\n\n%54s\n%54s\n\n",
81            "Response", "Frequency", "Histogram",
82            "1    1    2    2", "5    0    5    0    5" );
83
84    for ( rating = 1; rating <= 9; rating++ ) {
85       printf( "%8d%11d          ", rating, freq[ rating ] );
86
87       if ( freq[ rating ] > largest ) {
88          largest = freq[ rating ];
89          modeValue = rating;
90       }
91
92       for ( h = 1; h <= freq[ rating ]; h++ )
93          printf( "*" );
94
```

Notice how the subscript in **`frequency[]`** is the value of an element in **`response[]`** (**`answer[]`**)

Print stars depending on value of **`frequency[]`**



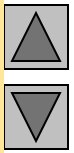
Outline

3.2 Define function `mode`

3.2.1 Increase `frequency[]` depending on `response[]`

```
95       printf( "\n" );
96    }
97
98    printf( "The mode is the most frequent value.\n"
99            "For this run the mode is %d which occurred"
100           " %d times.\n", modeValue, largest );
101}
102
103void bubbleSort( int a[] )
104{
105   int pass, j, hold;
106
107   for ( pass = 1; pass <= SIZE - 1; pass++ )
108
109      for ( j = 0; j <= SIZE - 2; j++ )
110
111         if ( a[ j ] > a[ j + 1 ] ) {
112            hold = a[ j ];
113            a[ j ] = a[ j + 1 ];
114            a[ j + 1 ] = hold;
115         }
116}
117
118void printArray( const int a[] )
119{
120   int j;
121
122   for ( j = 0; j <= SIZE - 1; j++ ) {
123
124      if ( j % 20 == 0 )
125         printf( "\n" );
```

Bubble sort: if elements out of order, swap them.

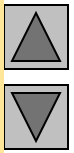


Outline

3.3 Define `bubbleSort`

3.3 Define `printArray`

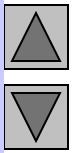
```
126
127       printf( "%2d", a[ j ] );
128    }
129}
```



Outline

Program Output

```
********
 Mean
********
The mean is the average value of the data
items. The mean is equal to the total of
all the data items divided by the number
of data items (99). The mean value for
this run is: 681 / 99 = 6.8788

********
 Median
********
The unsorted array of responses is
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

The sorted array is
 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 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

The median is element 49 of
the sorted 99 element array.
For this run the median is 7
```



Outline

Program Output

```
********
  Mode
********
Response   Frequency         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           *******************
The mode is the most frequent value.
For this run the mode is 8 which occurred 27 times.
```

6.8 Dizilerde Arama : Doğrusal Arama Ve İkili Arama

- Bir dizide anahtar aramak

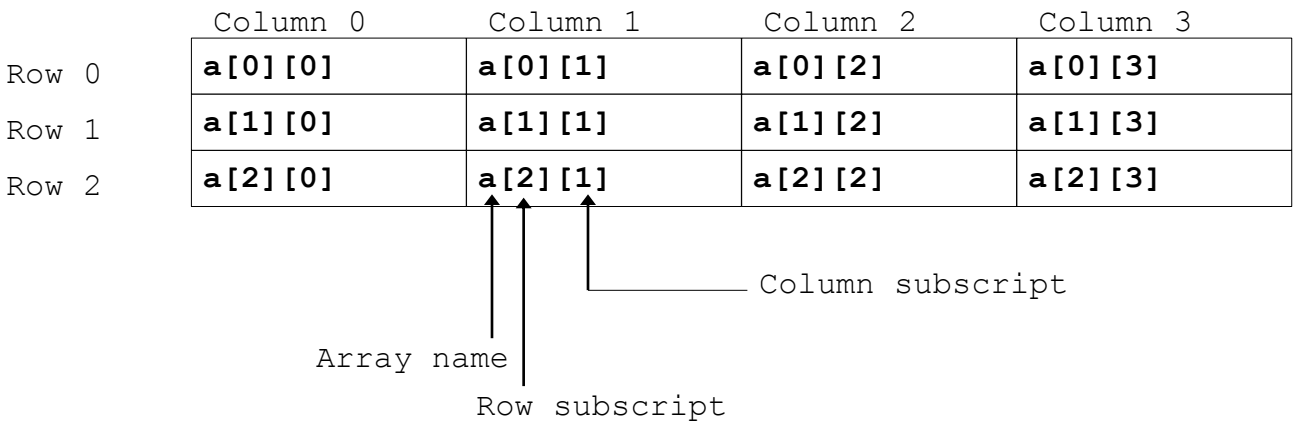
- Lineer arama
  - Basit
  - Her elemanı anahtar değer ile kıyasla
  - Küçük ve sırasız diziler için kullanışlıdır.

6.8 Dizilerde Arama : Doğrusal Arama Ve İkili Arama(II)

- İkili arama
  - Sıralı diziler için
  - En ortadaki elemanı anahtar ile kıyaslar
    - Eşitse , aranan bulundu
    - Eğer Anahtar < orta değer , ilk yarıya bakar
    - Eğer Anahtara > orta değer, diğer yarıya bakar
    - Tekrar eder
  - Çok hızlı ; en fazla n adımda, $2^n \geq$ elaman sayısı
    - 30 elamanlı bir dizi en fazla 5 adım alır
      - $2^5 > 30$

6.9 Çok boyutlu diziler

- Çok boyutlu diziler
  - Satır ve sütunlardan oluşan tablolar (*mX* y *n* dizisi)
  - Like matrices: specify row, then column



6.9 Multiple-Subscripted Arrays (II)

- Initialization

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

1	2
3	4

  - Initializers grouped by row in braces
  - If not enough, unspecified elements set to zero

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

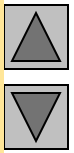
1	0
3	4

- Referencing elements
  - Specify row, then column

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

```
1  /* Fig. 6.22: fig06_22.c
2     Double-subscripted array example */
3  #include <stdio.h>
4  #define STUDENTS 3
5  #define EXAMS 4
6
7  int minimum( const int [][ EXAMS ], int, int );
8  int maximum( const int [][ EXAMS ], int, int );
9  double average( const int [], int );
10 void printArray( const int [][ EXAMS ], int, int );
11
12 int main()
13 {
14    int student;
15    const int studentGrades[ STUDENTS ][ EXAMS ] =
16       { { 77, 68, 86, 73 },
17         { 96, 87, 89, 78 },
18         { 70, 90, 86, 81 } };
19
20    printf( "The array is:\n" );
21    printArray( studentGrades, STUDENTS, EXAMS );
22    printf( "\n\nLowest grade: %d\nHighest grade: %d\n",
23            minimum( studentGrades, STUDENTS, EXAMS ),
24            maximum( studentGrades, STUDENTS, EXAMS ) );
25
26    for ( student = 0; student <= STUDENTS - 1; student++ )
27       printf( "The average grade for student %d is %.2f\n",
28               student,
29               average( studentGrades[ student ], EXAMS ) );
30
31    return 0;
32 }
```

Each row is a particular student, each column is the grades on the exam.



Outline

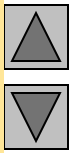
1. Initialize variables

1.1 Define functions to take double scripted

`studentgrades[][]`

2. Call functions `minimum`, `maximum`, and `average`

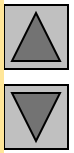
```
33
34 /* Find the minimum grade */
35 int minimum( const int grades[][ EXAMS ],
36              int pupils, int tests )
37 {
38    int i, j, lowGrade = 100;
39
40    for ( i = 0; i <= pupils - 1; i++ )
41       for ( j = 0; j <= tests - 1; j++ )
42          if ( grades[ i ][ j ] < lowGrade )
43             lowGrade = grades[ i ][ j ];
44
45    return lowGrade;
46 }
47
48 /* Find the maximum grade */
49 int maximum( const int grades[][ EXAMS ],
50              int pupils, int tests )
51 {
52    int i, j, highGrade = 0;
53
54    for ( i = 0; i <= pupils - 1; i++ )
55       for ( j = 0; j <= tests - 1; j++ )
56          if ( grades[ i ][ j ] > highGrade )
57             highGrade = grades[ i ][ j ];
58
59    return highGrade;
60 }
61
62 /* Determine the average grade for a particular exam */
63 double average( const int setOfGrades[], int tests )
64 {
```



Outline

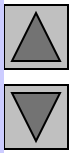
3. Define functions

```
65    int i, total = 0;
66
67    for ( i = 0; i <= tests - 1; i++ )
68       total += setOfGrades[ i ];
69
70    return ( double ) total / tests;
71 }
72
73 /* Print the array */
74 void printArray( const int grades[][ EXAMS ],
75                  int pupils, int tests )
76 {
77    int i, j;
78
79    printf( "                 [0]  [1]  [2]  [3]" );
80
81    for ( i = 0; i <= pupils - 1; i++ ) {
82       printf( "\nstudentGrades[%d] ", i );
83
84       for ( j = 0; j <= tests - 1; j++ )
85          printf( "%-5d", grades[ i ][ j ] );
86    }
87 }
```



Outline

3. Define functions



Outline

Program Output

```
The array is:
                 [0]   [1]   [2]   [3]
studentGrades[0] 77    68    86    73
studentGrades[1] 96    87    89    78
studentGrades[2] 70    90    86    81

Lowest grade: 68
Highest grade: 96
The average grade for student 0 is 76.00
The average grade for student 1 is 87.50
The average grade for student 2 is 81.75
```