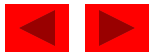
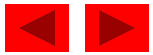
Chapter 6 - Arrays

Outline



Objectives

- In this chapter, you will learn:
  - To introduce the array data structure.
  - To understand the use of arrays to store and sort
  - To understand how to define an array, initialize an array and refer to individual elements of an array.
  - To be able to pass arrays to functions.
  - To understand basic sorting technique.

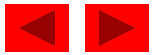


6.1 Introduction

- Arrays
  - Structures of related data items
  - Static entity – same size throughout program

  - An array is a collection of variables in same datatype.

  - we can't group different data types in array.
  - Like, combination of integer, char and float etc.
  - Hence Array is called as homogeneous data type.



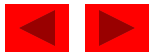
6.1 Introduction

- Why do we need an array?
  - Let's consider the situation where we need to get 10 student's age and store it for some calculation.
  - Since age is an integer type, we can store it something like below,

Example

```
int ag1, age2, age3, age4, age5, age6, age7, age8, age9, age10;
```

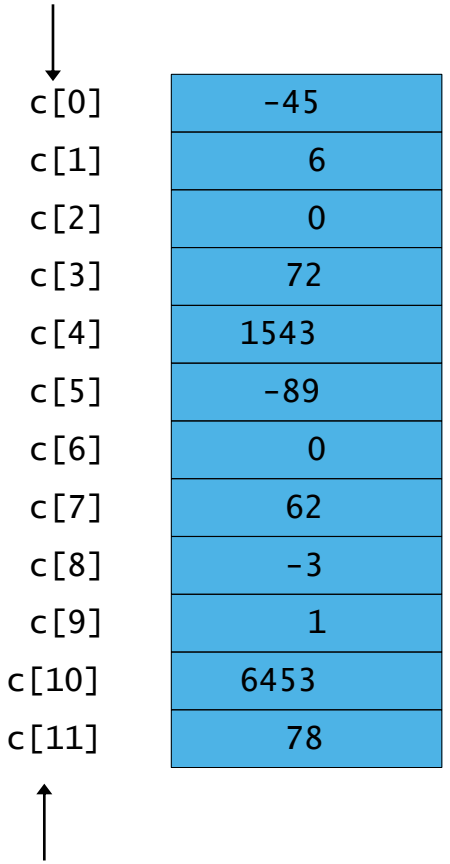
  - if we declare like above, it will be very difficult for us to manipulate the data.
  - If more number of student joins, then it is very difficult to declare a lot of variables and keep track of it.
  - To overcome this kind of situation, we should use Array data structure.



6.2 Arrays

- Array
  - Group of consecutive memory locations
  - Same name and type
- To refer to an element, specify
  - Array name
  - Position number
- Format:

  arrayname`[` *position number* `]`
  - First element at position 0
  - `n` element array named `c:`
    - `c[ 0 ], c[ 1 ]...c[ n - 1 ]`



6.2 Arrays

- Use Array to store ages
  - To store 10 students age, we can simply declare array like below,

Example

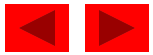
```
int age[10];
```

- Array index starts from 0 not 1.
  - To access 1st student's age, we can directly use index 0. i.e age[0]
  - To access 5th student's age, we can directly use index 4. i.e age[4]

We can manipulate Nth students age by using index N-1. Where N > 0

In general, an array of size N will have elements from index 0 to N-1.

  - Using index, we can directly access desired element.



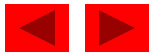
6.2 Arrays

- Array elements are like normal variables

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

  - Perform operations in subscript. If `x` equals `3`

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

6.3 Defining Arrays

- When defining arrays, specify
  - Name
  - Type of array
  - Number of elements

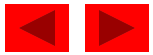
```
arrayType arrayName[ numberOfElements ];
```

  - Examples:

```
int c[ 10 ];
float myArray[3];
```

- Defining multiple arrays of same type
  - Format similar to regular variables
  - Example:

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

6.4 Examples Using Arrays

- Initializers

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

  - If not enough initializers, rightmost elements become 0

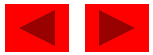
```
int n[ 5 ] = { 0 }
```

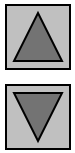
    - All elements 0
  - If too many a syntax error is produced

- If size omitted, initializers determine it

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

  - 5 initializers, therefore 5 element array

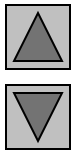




Outline

fig06_03.c

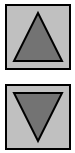
```
1  /* initializing an array */
2
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     int n[ 10 ]; /* n is an array of 10 integers */
9     int i;       /* counter */
10
11    /* initialize elements of array n to 0 */
12    for ( i = 0; i < 10; i++ ) {
13       n[ i ] = 0; /* set element at location i to 0 */
14    } /* end for */
15
16    printf( "%s%13s\n", "Element", "Value" );
17
18    /* output contents of array n in tabular format */
19    for ( i = 0; i < 10; i++ ) {
20       printf( "%7d%13d\n", i, n[ i ] );
21    } /* end for */
22
23    return 0;
24
25 }
```



Outline

Program Output

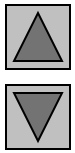
```
Element         Value
      0             0
      1             0
      2             0
      3             0
      4             0
      5             0
      6             0
      7             0
      8             0
      9             0
```



Outline

fig06_04.c

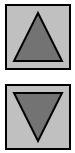
```
1  /* Initializing an array with an initializer list */
2
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     /* use initializer list to initialize array n */
9     int n[ 10 ] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };
10    int i; /* counter */
11
12    printf( "%s%13s\n", "Element", "Value" );
13
14    /* output contents of array in tabular format */
15    for ( i = 0; i < 10; i++ ) {
16       printf( "%7d%13d\n", i, n[ i ] );
17    } /* end for */
18
19    return 0; /* indicates successful termination */
20
21 } /* end main */
```



Outline

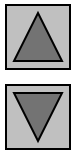
Program Output

```
Element         Value
      0            32
      1            27
      2            64
      3            18
      4            95
      5            14
      6            90
      7            70
      8            60
      9            37
```



Outline

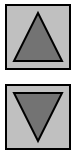
```
/*Initializing an array with a random number*/
#include "stdafx.h"
#include "time.h"
#include "stdlib.h"
void main()
{
   int n[10] ;
   int i;
   srand(time(NULL));
   printf("%s%13s\n", "Element", "Value");

   /* output contents of array in tabular format */
  for (i = 0; i < 10; i++)
  {
     n[i] = rand() % 20 + 1;
     printf("%7d%13d\n", i, n[i]);
  }

}
```



Outline

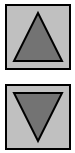
Program Output

```
Element         Value
      0            11
      1             5
      2             1
      3            10
      4            17
      5             5
      6            10
      7            16
      8            15
      9             2
```



Outline

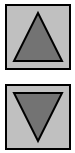
```
/*Get all elements using for loop and store it in array*/
#include "stdafx.h"
void main()
{
   int n[10] ;
   int i;

   //enter elements using a for loop
   for (i = 0; i < 10; i++)
   {
       printf("enter %d.element: ", i + 1);
       scanf_s("%d", &n[i]);
   }
   printf("%s%13s\n", "Element", "Value");
   /* output contents of array in tabular format */
   for (i = 0; i < 10; i++)
       printf("%7d%13d\n", i, n[i]);

}
```



Outline

Program Output

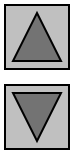
```
enter 1.element: 5
enter 2.element: 6
enter 3.element: 4
enter 4.element: 2
enter 5.element: 3
enter 6.element: 9
enter 7.element: 11
enter 8.element: 7
enter 9.element: 22
enter 10.element: 7



Element         Value
      0             5
      1             6
      2             4
      3             2
      4             3
      5             9
      6            11
      7             7
      8            22
      9             7
```



Outline

fig06_05.c

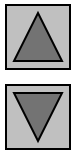
```
1  /* Initialize the elements of array s to the even integers from 2 to 20 */
2
3  #include <stdio.h>
4  #define SIZE 10
5
6  /* function main begins program execution */
7  int main()
8  {
9     /* symbolic constant SIZE can be used to specify array size */
10    int s[ SIZE ]; /* array s has 10 elements */
11    int j;         /* counter */
12
13    for ( j = 0; j < SIZE; j++ ) { /* set the values */
14       s[ j ] = 2 + 2 * j;
15    } /* end for */
16
17    printf( "%s%13s\n", "Element", "Value" );
18
19    /* output contents of array s in tabular format */
20    for ( j = 0; j < SIZE; j++ ) {
21       printf( "%7d%13d\n", j, s[ j ] );
22    } /* end for */
23
24    return 0; /* indicates successful termination */
25
26 } /* end main */
```



Outline

Program Output

```
Element         Value
      0             2
      1             4
      2             6
      3             8
      4            10
      5            12
      6            14
      7            16
      8            18
      9            20
```



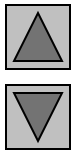
Outline

fig06_06.c

```c
/* Compute the sum of the elements of the array */

#include <stdio.h>
#define SIZE 12

/* function main begins program execution */
int main()
{
   /* use initializer list to initialize array */
   int a[ SIZE ] = { 1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45 };
   int i;         /* counter */
   int total = 0; /* sum of array */

   /* sum contents of array a */
   for ( i = 0; i < SIZE; i++ ) {
      total += a[ i ];
   } /* end for */

   printf( "Total of array element values is %d\n", total );

   return 0; /* indicates successful termination */

} /* end main */
```

Program Output

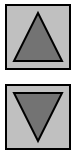
```
Total of array element values is 383
```



Outline

fig06_07.c (Part 1 of 2)

```
1  /* Student poll program - to count frequency of responses*/
2
3  #include <stdio.h>
4  #define RESPONSE_SIZE 40 /* define array sizes */
5  #define FREQUENCY_SIZE 11
6
7  /* function main begins program execution */
8  int main()
9  {
10    int answer; /* counter */
11    int rating; /* counter */
12
13    /* initialize frequency counters to 0 */
14    int frequency[ FREQUENCY_SIZE ] = { 0 };
15
16    /* place survey responses in array responses */
17    int responses[ RESPONSE_SIZE ] = { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
18         1, 6, 3, 8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6,
19         5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };
20
```



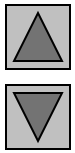
Outline

```
21    /* for each answer, select value of an element of array responses
22       and use that value as subscript in array frequency to
23       determine element to increment */
24    for ( answer = 0; answer < RESPONSE_SIZE; answer++ ) {
25       ++frequency[ responses [ answer ] ];
26    } /* end for */
27
28    /* display results */
29    printf( "%s%17s\n", "Rating", "Frequency" );
30
31    /* output frequencies in tabular format */
32    for ( rating = 1; rating < FREQUENCY_SIZE; rating++ ) {
33       printf( "%6d%17d\n", rating, frequency[ rating ] );
34    } /* end for */
35
36    return 0; /* indicates successful termination */
37
38 } /* end main */
```

fig06_07.c (Part 2 of 2)

Program Output

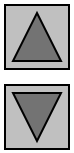
```
Rating        Frequency
     1                2
     2                2
     3                2
     4                2
     5                5
     6               11
     7                5
     8                7
     9                1
    10                3
```



Outline

fig06_08.c (Part 1 of 2)

```
1  /* Histogram printing program */
2
3  #include <stdio.h>
4  #define SIZE 10
5
6  /* function main begins program execution */
7  int main()
8  {
9     /* use initializer list to initialize array n */
10    int n[ SIZE ] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
11    int i; /* outer counter */
12    int j; /* inner counter */
13
14    printf( "%s%13s%17s\n", "Element", "Value", "Histogram" );
15
16    /* for each element of array n, output a bar in histogram */
17    for ( i = 0; i < SIZE; i++ ) {
18       printf( "%7d%13d          ", i, n[ i ]) ;
19
20       for ( j = 1; j <= n[ i ]; j++ ) { /* print one bar */
21          printf( "%c", '*' );
22        } /* end inner for */
23
```



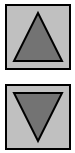
Outline

fig06_08.c (Part 2 of 2)

```
24       printf( "\n" ); /* start next line of output */
25    } /* end outer for */
26
27    return 0; /* indicates successful termination */
28
29 } /* end main */
```

Program Output

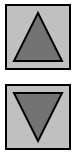
```
Element         Value         Histogram
      0            19         ********************
      1             3         ***
      2            15         ***************
      3             7         *******
      4            11         ***********
      5             9         *********
     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



Outline

fig06_09.c (Part 1 of 2)

```
1  /* Roll a six-sided die 20 times */
2
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <time.h>
6  #define SIZE 7
7
8  /* function main begins program execution */
9  int main()
10 {
11    int face;                    /* random number with value 1 - 6 */
12    int roll;                    /* roll counter */
13    int frequency[ SIZE ] = { 0 }; /* initialize array to 0 */
14
15    srand( time( NULL ) ); /* seed random-number generator */
16
17    /* roll die 20 times */
18    for ( roll = 1; roll <= 20; roll++ ) {
19       face = rand() % 6 + 1;
20       ++frequency[ face ]; /* replaces 26-line switch of Fig. 5.8 */
21    } /* end for */
22
23    printf( "%s%17s\n", "Face", "Frequency" );
24
```



Outline

```
25    /* output frequency elements 1-6 in tabular format */
26    for ( face = 1; face < SIZE; face++ ) {
27       printf( "%4d%17d\n", face, frequency[ face ] );
28    } /* end for */
29
30    return 0; /* indicates successful termination */
31
32 } /* end main */
```

```
Face        Frequency
   1                1
   2                6
   3                2
   4                6
   5                2
   6                3
```

fig06_09.c (Part 2 of 2)

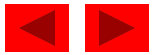
Program Output

6.5 Passing Arrays to Functions

- Passing arrays
  - To pass an array argument to a function, specify the name of the array without any brackets

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

    - Array size usually passed to function
  - Arrays passed call-by-reference
  - Name of array is address of first element
  - Function knows where the array is stored
    - Modifies original memory locations
- Passing array elements
  - Passed by call-by-value
  - Pass subscripted name (i.e., `myArray[ 3 ]`) to function

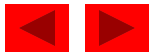


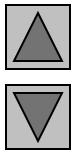
6.5 Passing Arrays to Functions

- Function prototype

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

  - Parameter names optional in prototype
    - `int b[]` could be written `int []`
    - `int arraySize` could be simply `int`

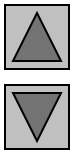




Outline

fig06_13.c (Part 1 of 3)

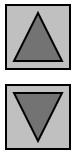
```
1  /* Passing arrays and individual array elements to functions */
2
3  #include <stdio.h>
4  #define SIZE 5
5
6  /* function prototypes */
7  void modifyArray( int b[], int size );
8  void modifyElement( int e );
9
10 /* function main begins program execution */
11 int main()
12 {
13    int a[ SIZE ] = { 0, 1, 2, 3, 4 }; /* initialize a */
14    int i; /* counter */
15
16    printf( "Effects of passing entire array by reference:\n\nThe "
17           "values of the original array are:\n" );
18
19    /* output original array */
20    for ( i = 0; i < SIZE; i++ ) {
21       printf( "%3d", a[ i ] );
22    } /* end for */
23
24    printf( "\n" );
25
```



Outline

fig06_13.c (Part 2 of 3)

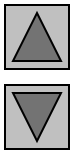
```
26    /* pass array a to modifyArray by reference */
27    modifyArray( a, SIZE );
28
29    printf( "The values of the modified array are:\n" );
30
31    /* output modified array */
32    for ( i = 0; i < SIZE; i++ ) {
33       printf( "%3d", a[ i ] );
34    } /* end for */
35
36    /* output value of a[ 3 ] */
37    printf( "\n\n\nEffects of passing array element "
38            "by value:\n\nThe value of a[3] is %d\n", a[ 3 ] );
39
40    modifyElement( a[ 3 ] ); /* pass array element a[ 3 ] by value */
41
42    /* output value of a[ 3 ] */
43    printf( "The value of a[ 3 ] is %d\n", a[ 3 ] );
44
45    return 0; /* indicates successful termination */
46
47 } /* end main */
48
```



Outline

```
49 /* in function modifyArray, "b" points to the original array "a"
50    in memory */
51 void modifyArray( int b[], int size )
52 {
53    int j; /* counter */
54
55    /* multiply each array element by 2 */
56    for ( j = 0; j < size; j++ ) {
57       b[ j ] *= 2;
58    } /* end for */
59
60 } /* end function modifyArray */
61
62 /* in function modifyElement, "e" is a local copy of array element
63    a[ 3 ] passed from main */
64 void modifyElement( int e )
65 {
66    /* multiply parameter by 2 */
67    printf( "Value in modifyElement is %d\n", e *= 2 );
68 } /* end function modifyElement */
```

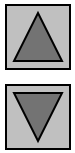
fig06_13.c (Part 3 of 3)



Outline

Program Output

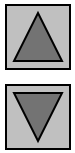
```
Effects of passing entire array by reference:

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 by value:

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



Outline

```
/*filling an array with a random number and printing in a function*/
#include "stdafx.h"
#include "stdlib.h"
#include "time.h"
#define SIZE 10
void funcArray(int[]);
void main()
{
  int i;
  int num[SIZE];
  srand(time(NULL));
  for (int i = 0; i<SIZE; i++)
     num[i] = rand() % 20 + 5;
   funcArray(num);
}
void funcArray(int num[])
{
   for (int i = 0; i<SIZE; i++)
   printf("%4d", num[i]);
}
```

```
16  24  11  24  15  19  22  23   7
```

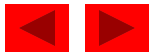



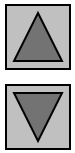
Outline

Program Output

6.6 Sorting Arrays

- Sorting data
  - Important computing application
  - Virtually every organization must sort some data
- Bubble sort (sinking sort)
  - Several passes through the array
  - Successive pairs of elements are compared
    - If increasing order (or identical), no change
    - If decreasing order, elements exchanged
  - Repeat
- 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

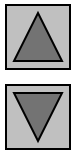




Outline

fig06_15.c (Part 1 of 3)

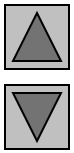
```
1  /* This program sorts an array's values into ascending order */
2
3  #include <stdio.h>
4  #define SIZE 10
5
6  /* function main begins program execution */
7  int main()
8  {
9     /* initialize a */
10    int a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
11    int i;    /* inner counter */
12    int pass; /* outer counter */
13    int hold; /* temporary location used to swap array elements */
14
15    printf( "Data items in original order\n" );
16
17    /* output original array */
18    for ( i = 0; i < SIZE; i++ ) {
19       printf( "%4d", a[ i ] );
20    } /* end for */
21
```



Outline

fig06_15.c (Part 2 of 3)

```
22    /* bubble sort */
23    /* loop to control number of passes */
24    for ( pass = 1; pass < SIZE; pass++ ) {
25
26       /* loop to control number of comparisons per pass */
27       for ( i = 0; i < SIZE - 1; i++ ) {
28
29          /* compare adjacent elements and swap them if first
30          element is greater than second element */
31          if ( a[ i ] > a[ i + 1 ] ) {
32             hold = a[ i ];
33             a[ i ] = a[ i + 1 ];
34             a[ i + 1 ] = hold;
35          } /* end if */
36
37       } /* end inner for */
38
39    } /* end outer for */
40
41    printf( "\nData items in ascending order\n" );
42
```



Outline

```
43    /* output sorted array */
44    for ( i = 0; i < SIZE; i++ ) {
45       printf( "%4d", a[ i ] );
46    } /* end for */
47
48    printf( "\n" );
49
50    return 0; /* indicates successful termination */
51
```

fig06_15.c (Part 3 of 3)

Program Output

```
Data items in original order
   2   6   4   8  10  12  89  68  45  37
Data items in ascending order
   2   4   6   8  10  12  37  45  68  89
```