ARRAYS

Sevil ŞEN
Hacettepe University
November 2010

Content

In this chapter, you will learn:

- To introduce the array data structure.
- To understand the use of arrays.
- 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 be able to define and manipulate multi-dimensional arrays.

Introduction

Arrays

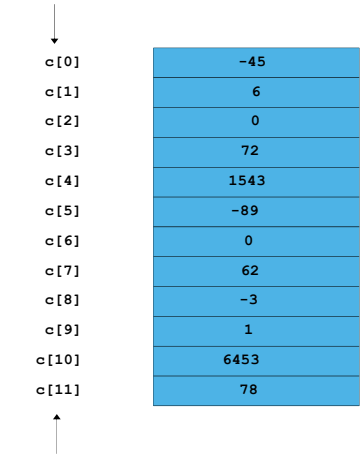
- Structures of related data items
- Static entity – same size throughout program
- Dynamic data structures will be discussed later

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 ]`

Name of array (Note that all elements of this array have the same name, **c**)



Position number of the element within array **c**

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 ]
c[x+1] == c[4]
c[x-1] == c[2]
```

Arrays

Operators						Associativity	Type
[]	()					left to right	highest
++	--	!	(*type*)			right to left	unary
*	/	%				left to right	multiplicative
+	-					left to right	additive
<	<=	>	>=			left to right	relational
==	!=					left to right	equality
&&						left to right	logical and
||						left to right	logical or
?:						right to left	conditional
=	+=	-=	*=	/=	%=	right to left	assignment
,						left to right	comma

Fig. 6.2 Operator precedence.

double x[8];

x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]
16.0	12.0	6.0	8.0	2.5	12.0	14.0	-54.5

i=5

printf("%d %.1f", 4, x[4]);	4 2.5
printf("%d %.1f", i, x[i]);	5 12.0
printf("%.1f", x[i]+1);	13.0
printf("%.1f", x[i]+i);	17.0
printf("%.1f", x[i+1]);	14.0
printf("%.1f", x[i+i]);	invalid
printf("%.1f", x[2*i]);	invalid
printf("%.1f", x[2*i-3]);	-54.5
printf("%.1f", x[(int)x[4]]);	6.0
printf("%.1f", x[i++]);	12.0
printf("%.1f", x[--i]);	12.0

May result in a run-time error
Display incorrect results

double x[8];

x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]
16.0	12.0	6.0	8.0	2.5	12.0	14.0	-54.5

i=5

x[i-1] = x[i]

x[i] = x[i+1]

x[i]-1 = x[i] Illegal assignment statement!

Defining Arrays

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

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

Examples:

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

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

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

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
  - C arrays have no bounds checking

- If size omitted, initializers determine it

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

  - 5 initializers, therefore 5 element array

Initializing an Array

```
1  /* Fig. 6.3: fig06_03.c
2     initializing an array */
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; /* indicates successful termination */
24
25 } /* end main */
```

Program Output

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

Examples

- Reading values into an array

```
int i, x[100];
```

```
for (i=0; i < 100; i=i+1)
{
  printf("Enter an integer: ");
  scanf("%d",&x[i]);
}
```

- Summing up all elements in an array

```
int sum = 0;
for (i=0; i<=99; i=i+1)
   sum = sum + x[i];
```

Examples

- Finding the location of a given value (`item`) in an array.

```
i = 0;
while ((i<100) && (x[i] != item))
   i = i + 1;

if (i == 100)
   loc = -1; // not found
else
   loc = i;  // found in location i
```

Examples

- Shifting the elements of an array to the left.

```
/* store the value of the first element in a
 * temporary variable
 */
temp = x[0];

for (i=0; i < 99; i=i+1)
   x[i] = x[i+1];

//The value stored in temp is going to be
the value of the last element:
x[99] = temp;
```

Examples Using Arrays

- Character arrays
  - String "`first`" is really a static array of characters
  - Character arrays can be initialized using string literals
    - char string1[] = "first";
    - Null character `'\0'` terminates strings
    - `string1` actually has 6 elements
      - equivalent to `char string1[] = { 'f', 'i', 'r', 's', 't', '\0' };`
  - Can access individual characters
    `string1[ 3 ] is character 's'`
  - Array name is address of array, so & not needed for scanf
    `scanf( "%s", string2 );`
    - Reads characters until whitespace encountered
    - Can write beyond end of array, be careful

```c
/* Fig. 6.4: fig06_04.c
   Initializing an array with an initializer list */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   /* use initializer list to initialize array n */
   int n[ 10 ] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };
   int i; /* counter */

   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 ] );
   } /* end for */

   return 0; /* indicates successful termination */

} /* end main */
```

Program Output

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

```c
/* Fig. 6.5: fig06_05.c
   Initialize the elements of array s to the even integers from 2 to 20 */
#include <stdio.h>
#define SIZE 10

/* function main begins program execution */
int main()
{
   /* symbolic constant SIZE can be used to specify array size */
   int s[ SIZE ]; /* array s has 10 elements */
   int j;         /* counter */

   for ( j = 0; j < SIZE; j++ ) { /* set the values */
      s[ j ] = 2 + 2 * j;
   } /* end for */

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

   /* output contents of array s in tabular format */
   for ( j = 0; j < SIZE; j++ ) {
      printf( "%7d%13d\n", j, s[ j ] );
   } /* end for */

   return 0; /* indicates successful termination */

} /* end main */
```

Program Output

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

```c
/* Fig. 6.6: fig06_06.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 */
```

Total of array element values is 383

```c
/* Fig. 6.7: fig06_07.c
   Student poll program */
#include <stdio.h>
#define RESPONSE_SIZE 40 /* define array sizes */
#define FREQUENCY_SIZE 11

/* function main begins program execution */
int main()
{
   int answer; /* counter */
   int rating; /* counter */

   /* initialize frequency counters to 0 */
   int frequency[ FREQUENCY_SIZE ] = { 0 };

   /* place survey responses in array responses */
   int responses[ RESPONSE_SIZE ] = { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
        1, 6, 3, 8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6,
        5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };

   /* for each answer, select value of an element of array responses
      and use that value as subscript in array frequency to
      determine element to increment */
   for ( answer = 0; answer < RESPONSE_SIZE; answer++ ) {
      ++frequency[ responses [ answer ] ];
   } /* end for */

   /* display results */
   printf( "%s%17s\n", "Rating", "Frequency" );

   /* output frequencies in tabular format */
   for ( rating = 1; rating < FREQUENCY_SIZE; rating++ ) {
      printf( "%6d%17d\n", rating, frequency[ rating ] );
   } /* end for */

   return 0; /* indicates successful termination */

} /* end main */
```

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

```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;
}
```

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        *
```

```c
/* Fig. 6.9: fig06_09.c
   Roll a six-sided die 6000 times */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define SIZE 7

/* function main begins program execution */
int main()
{
   int face;                    /* random number with value 1 - 6 */
   int roll;                    /* roll counter */
   int frequency[ SIZE ] = { 0 }; /* initialize array to 0 */

   srand( time( NULL ) ); /* seed random-number generator */

   /* roll die 6000 times */
   for ( roll = 1; roll <= 6000; roll++ ) {
      face = rand() % 6 + 1;
      ++frequency[ face ]; /* replaces 26-line switch of Fig. 5.8 */
   } /* end for */

   printf( "%s%17s\n", "Face", "Frequency" );

   /* output frequency elements 1-6 in tabular format */
   for ( face = 1; face < SIZE; face++ ) {
      printf( "%4d%17d\n", face, frequency[ face ] );
   } /* end for */

   return 0; /* indicates successful termination */

} /* end main */
```

Program Output

```
Face        Frequency
   1             1029
   2              951
   3              987
   4             1033
   5             1010
   6              990
```

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

/* function main begins program execution */
int main()
{
   char string1[ 20 ];               /* reserves 20 characters */
   char string2[] = "string literal"; /* reserves 15 characters */
   int i;                            /* counter */

   /* read string from user into array string2 */
   printf("Enter a string: ");
   scanf( "%s", string1 );

   /* output strings */
   printf( "string1 is: %s\nstring2 is: %s\n"
           "string1 with spaces between characters is:\n",
           string1, string2 );

   /* output characters until null character is reached */
   for ( i = 0; string1[ i ] != '\0'; i++ ) {
      printf( "%c ", string1[ i ] );
   } /* end for */
   printf( "\n" );

   return 0; /* indicates successful termination */

} /* end main */
```

Program Output

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

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

```c
/* Fig. 6.15: fig06_15.c
   This program sorts an array's values into ascending order */
#include <stdio.h>
#define SIZE 10

/* function main begins program execution */
int main()
{
   /* initialize a */
   int a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
   int i;    /* inner counter */
   int pass; /* outer counter */
   int hold; /* temporary location used to swap array elements */

   printf( "Data items in original order\n" );

   /* output original array */
   for ( i = 0; i < SIZE; i++ ) {
      printf( "%4d", a[ i ] );
   } /* end for */

```

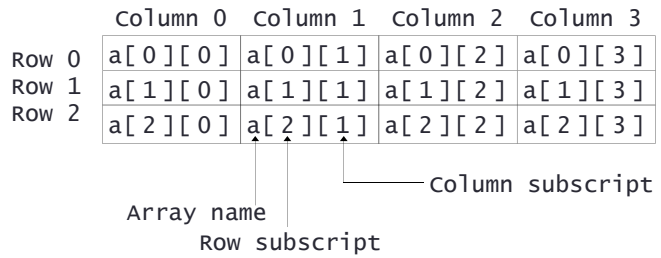
```
22    /* bubble sort */
23    /* loop to control number of passes */
24    for ( pass = 1; pass < SIZE; pass++ ) {
25
26       /* loop to control number of comparisons per pass */
27       for ( i = 0; i < SIZE - 1; i++ ) {
28
29          /* compare adjacent elements and swap them if first
30          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
```

```
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
```

```
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
```

Multi-Dimensional Arrays

- Multiple subscripted arrays
  - Tables with rows and columns (m by n array)
  - Like matrices: specify row, then column



Multi-Dimensional Arrays

- Initialization
  - `int b[2][2] = { { 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	2
3	4

1	0
3	4

- Referencing elements
  - Specify row, then column
    `printf( "%d", b[0][1] );`

Example:Multi-Dimensional Array

```
#include <stdio.h>
int main()
{
   int i; /* counter */
   int j; /* counter */

   /* initialize array1, array2, array3 */
   int array1[ 2 ][ 3 ] = { { 1, 2, 3 }, { 4, 5, 6 } };
   int array2[ 2 ][ 3 ] = { 1, 2, 3, 4, 5 };
   int array3[ 2 ][ 3 ] = { { 1, 2 }, { 4 } };

   printf( "Values in array1 by row are:\n" );

   for ( i = 0; i <= 1; i++ ) {                    /* loop through rows */
            for ( j = 0; j <= 2; j++ )
                     printf( "%d ", array1[ i ][ j ] );    /* output column values */
            printf( "\n" );
   }
```

Example:Multi-Dimensional Array

```
   printf( "Values in array2 by row are:\n" );
   for ( i = 0; i <= 1; i++ ) {                    /* loop through rows */
            for ( j = 0; j <= 2; j++ )
                     printf( "%d ", array2[ i ][ j ] );    /* output column values */
            printf( "\n" );
   }

   printf( "Values in array3 by row are:\n" );
   for ( i = 0; i <= 1; i++ ) {                    /* loop through rows */
            for ( j = 0; j <= 2; j++ )
                     printf( "%d ", array3[ i ][ j ] );
            printf( "\n" );
   }

   return 0;
}
```

```
Values in array1 by row are:
1 2 3
4 5 6
Values in array2 by row are:
1 2 3
4 5 0
Values in array3 by row are:
1 2 0
4 0 0
```