



SLIIT

Discover Your Future

IT1010 – Introduction to Programming

Lecture 7 - Arrays



Objectives

- At the end of the Lecture students should be able to
 - To define an array, initialize an array and refer to individual elements of an array.
 - To pass array to functions.
 - To define and manipulate multidimensional arrays.
 - To use string functions to handle character arrays.

Introduction

- Array is a data structure which store the data items of the same data type.
- Array store all the data items in continuous memory locations.

c[0]	-45
c[1]	6
c[2]	0
c[3]	72
c[4]	1543
c[5]	-89
c[6]	0
c[7]	62

Defining Arrays

- To define an array, we need to specify the type of data elements, name and the number of elements (size).

```
int c[8]
```

- The above definition reserves 8 elements for integer array c.
- Array name, like other variables can contain only letters, digits and underscore and cannot begin with a digit.

Using Arrays

- To refer to a particular location or element in the array, we need to specify array's name followed by the position number (index, subscript) of the particular element in square brackets.
- First element in the array is the zeroth element. Last element is size – 1.

`c[0], c[1], c[2],c[3].....c[7]`

Using Arrays

- Print the first element in the array.

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

- Print the sum of first three elements in the array.

```
printf("%d", c[0] + c[1] + c[2]);
```

- Add 2 to the fifth element

```
c[4] += 2;
```

Defining and initializing an array

```
//initializing the elements of an array to zeros
#include <stdio.h>
int main(void)
{
    int n[ 5 ]; // n is an array of 5 integers
    int i; // counter

    //initialize elements of array n to 0
    for( i = 0; i < 5; ++i)
        n[ i ] = 0;

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

    //output contents of array n in a tabular format
    for( i = 0; i < 5; ++i)
        printf("%7d %13d\n", i , n[ i ]);
    return 0;
}
```

Initializing array using an initializer list

```
/*initializing the elements of an array using an initializer list */  
# include <stdio.h>  
int main(void)  
{  
    int n[5] = {5, 12, 34, 56, 23};  
    int i;  
    printf("%s%13s\n", "Element", " Value");  
    //output contents of array n in a tabular format  
    for( i = 0; i < 5; ++i)  
        printf("%7d %13d\n", i , n[ i ]);  
    return 0;  
}
```


Initializing array using an initializer list

```
/*initializing the elements of an to zero
# include <stdio.h>
int main(void)
{
    int n[5] = { 0 };
    int i;

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

    //output contents of array n in a tabular format
    for( i = 0; i < 5; ++i)
        printf("%7d %13d\n", i , n[ i ]);
}
```

Specifying an array's size with a symbolic constant

```
#include <stdio.h>
#define SIZE 10
int main(void)
{
    int a[ SIZE ];
    int j; // counter

    for( j = 0; j < SIZE; ++j)
        a[ j ] = 2 + 2 * j;

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

    for( j = 0; j < SIZE; ++j)
        printf("%7d %13d\n", j , a[ j ]);
}
```

Summing the Elements of an Array

```
#include <stdio.h>
#define SIZE 12
int main(void)
{
    int a[ SIZE ];
    int i;
    int total = 0; // sum of array

    for( i = 0; i < SIZE; ++i)
    {
        printf("\na[ i ] = ");
        scanf("%d", &a[ i ]);
    }
    for( j = 0; j < SIZE; ++j)
        total += a[ j ];
    printf("Total of array elements is %d \n", total);
}
```

Exercise 1

- Write a C program to the following.
 - Define an integer array *counts* with 10 elements.
 - Initialize all elements to zeros.
 - Read and store 10 numbers each of which is between 10 to 100.
 - Find the maximum number from the stored numbers.

Storing *strings* in character arrays

- A string can be stored in a character array as follows:

```
char string1 [ ] = "first";
```

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

```
scanf( "%19s", string1);
```

- Function scanf will read characters until space, tab, newline or end-of-file indicator is encountered.

Display character strings

- A character array representing a string can be printed as follows:

```
printf("string1 is : %s\n", string1);
```

```
for ( i= 0; i < SIZE && string1 [ i ] != '\0'; ++i){  
    printf("%c", string1[ i ] );  
}
```

Function strcpy

- strcpy copy the entire string in array x into y

```
# include <stdio.h>
# include <string.h>
# define SIZE1 25
# define SIZE2 15
int main ( void )
{
    char x[ ]= 'Happy Birthday to You';
    char y[ SIZE1];

    strcpy( y , x );
    printf("The string in array y is : %s\n", y);
    return 0;
}
```

Output: The string in array y is : Happy Birthday to You

Function strlen

- strlen takes a string as an argument and return the number of characters in the string.

```
#include <stdio.h>
#include <string.h>
int main ( void )
{
    char string1[ ]= 'I love C programming';
    printf("The length of string1 is %d", strlen(string1));
    return 0;
}
```

Output: The length of string1 is 20

Passing Arrays to Functions

- To pass an array argument to a function, specify the array's name without any brackets.

```
#include <stdio.h>
#define SIZE 5
void modifyArray(int b[ ], size_t size);

int main ( void )
{
    int a[ SIZE] = { 0, 1, 2, 3, 4};
    modifyArray(a, SIZE);
    return 0;
}
```

```
void modifyArray( int b [ ], size_t size)
{
    size_t j;
    // multiply each array element by 2
    for( j = 0; j < size; ++j)
        b[j] *= 2;
}
```

Passing Arrays to Functions

```
# include <stdio.h>
# define SIZE 5
void modifyArray( int b[ ], size_t size);
int main(void)
{
    int a[ SIZE ] = {0, 1, 2, 3, 4};
    size_t i; // counter
    //output original array
    for( i = 0; i < SIZE; ++i)
        printf("%3d", a[ i ]);

    puts(" ");
    modifyArray( a , SIZE);
    // output modified array
    for( i = 0; i < SIZE; ++i)
        printf("%3d ", a[i ]);
}
```

Output

Original Array : 0 1 2 3 4

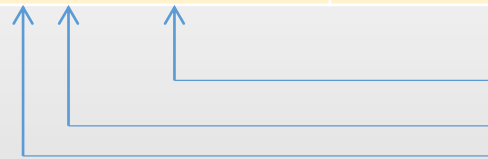
Modified Array : 0 2 4 6 8

Multidimensional Arrays

- C language have arrays with multiple subscripts.
- These arrays are refers to as multidimensional arrays.
- Multidimensional arrays are used to represent table of values consisting of information arranged in rows and columns.
- A array with two subscripts is called **double-subscripted or Two-Dimensional** array.

Two-Dimensional Array

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]



Column index
Row index
Array name

Define and initialize 2D array

```
//initializing multidimensional arrays
#include <stdio.h>

int main(void)
{
    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}};

    for( i = 0; i <= 1; ++i){
        for( j = 0; j <= 2; ++j)
            printf("%d\n", array1[i][j]);
        printf("\n");
    }
```

```
for( i = 0; i <= 1; ++i){
    for( j = 0; j <= 2; ++j)
        printf("%d\n", array2[i][j]);
    printf("\n");
}

for( i = 0; i <= 1; ++i){
    for( j = 0; j <= 2; ++j)
        printf("%d\n", array3[i][j]);
    printf("\n");
}

return 0;
}
```

Define and initialize 2D array

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

Summing the Elements of a 2D Array

```
# include <stdio.h>
# define SIZE 12
int main(void)
{
    int row, column;
    int a[ 2][3];
    int total = 0;
    for( row = 0; row <=1; ++row){
        for( column = 0; column <= 2; ++ column)
        {
            printf("\na[ row][column ] = ", row, column);
            scanf("%d", &a[ row ][column ]);
        }
        for( row = 0; row <=1; ++row)
            for( column = 0; column <= 2; ++ column)
                total += a [row] [column];
    }
    printf("The total of the elements of the array : %d", total);
    return 0;
}
```

Summary

- Handling 1D arrays
- String manipulation
- Passing arrays to functions
- Handling 2D arrays