Arrays and Strings

Unit 6

INTRODUCTION

In the previous lessons you have learned about the programming structure, decision making procedure, how to write statements, as well as different types of conditional statements, branching statements and various types of loops. In this unit you will learn about definition of an array, declaration and initialization of an array, various types of arrays and its applications and also how to handle arrays. On the other hand you also understand how strings are declared, initialized as well as how strings are handled using C library functions. In C programming language the array is very useful data structure.

Timeframe 	We expect that this unit will take maximum 10 hours to complete.

Unit Structure	
Lesson- 1 : Familiar with Arrays Lesson- 2 : Familiar with One Dimensional Array Lesson- 3 : Familiar with Two Dimensional Arrays Lesson- 4 : Introduction to Strings Lesson- 5 : Understanding String Handling Functions	

Lesson-1 Familiar With Arrays

Learning Outcomes

Outcomes

Upon completion of this lesson you will be able to

- Define, describe, and explain the array data structure.
- Understand the declaration of an array.
- Be aware of initialization of an array.

	**Keywords**	**Array, Declaration, Initialization, Pictorial Representation**

INTRODUCTION TO AN ARRAY

An array is a one kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of same type of data. In general it is often more useful to think of an array as a collection of variables of the same type. On the other hand we also define that an array is a sequence of data item of homogeneous value.


Note it!

An array is used to store a collection of same type of data.

WHY DO WE USE ARRAY?

Practically, in C programming language, one of the frequently rising problems is to handle similar types of data. For instance, if the user or programmer want to store marks of 200 students. Generally this can be done by creating 200 variables individually but, this process is rather tedious and impracticable. This type of problem can be handled using arrays. All elements of any given array must be of the same type i.e., we can't have an array of 10 numbers, of which 5 are integer type (int data type) and 5 are float type. Instead of declaring individual variables, such as marks0, marks1, marks2, marks3 ….. and marks199, you can declare one array variable such as marks and use marks[0], marks[1],marks[2] ,marks[3] and ..., marks[199] to represent individual variable. A particular component or element in an array is accessed by an index.


Activity

1. Mention the significance of using an array in C language by your own concept.
   ……..……..
   ……...
   ……..…….

ARRAY DECLARATION PROCEDURE

An array must be declared before it can be used in the program. In general, in C programming language an array declaration procedure is as follows:

data- type variable_name[array_size];

After array declaration you must put semicolon (;).

SOME DECLARATION EXAMPLE

1. If we want to store 6 integer type values in an array, we can write in C:
   int numbers[6];
2. If we want to store 100 float type values in an array, we can write in C:
   float marks[100];
3. If want to store 10 double type values in an array, we can write in C:
   double marks[10];
4. If we want to store 20 char type values in an array, we can write in C:
   char name[20];


Study skills

Which of the following is true?

A. An array is stored various types of data at a time.
B. A particular component or element in an array is accessed by an index.

Solution:

B. We know that an array is stored only same type of data and every element is accessed by array index.

ARRAY INITIALIZATION PROCEDURE

Initializing of an array is very simple in C programming. The initializing values are enclosed within the curly { } braces in the declaration and placed following an equal sign after the array name. In C Language, an array starts at position 0. The elements of the array occupy adjacent locations in memory. C Language treats the name of the array as if it were a pointer to the first element. Any item in the array can be accessed through its index, and it can be accessed anywhere using program.

INITIALIZATION EXAMPLE

Here is an illustration which declares and initializes an array of five student's marks of type integer (int).

```
int marks[5]={50,80,30,40,70};
```

Array can also be initialized after declaration. In the following example, that demonstrates the declaration and initialization of an array.

```
int age[5];
age[0]=50;
age[1]=70;
age[2]=60;
age[3]=40;
age[4]=30;
```

All arrays have 0 as the index of their first element which is also called the base index and the last index of an array will be total size of the array minus 1. The pictorial representation of the array we discussed above is shown in below

age[0]	age[1]	age[2]	age[3]	age[4]
50	70	60	40	30

Figure 6.1.1: Pictorial representation of an array

Study skills

1. Suppose we want to store 10 students CGPA number as grade. Which declaration is correct?

   A. int number[10];
   B. int CGPA[10];
   C. float grade[10];
   D. Char grade[20];

Solution:

C. As per rule of array we know that array is stored same type of data. Students CGPA is float type number that is float type data. So if we store float type value of an array we required float type of array declaration.

Summary **Summary**	
In this lesson ▪ We have covered about C language array and its declaration. ▪ Moreover we have learned some example of initialization procedure of an array. ▪ Wealso have understood that how an array is presented graphically.	

ASSIGNMENT

Assignment

1. Suppose you are a programmer of a firm. Your firm has 10 employees. Now declare and initialization an array named "Salary" to store employee's salary.
   …….
2. Represent the array as graphically considering the activity 1
   ……

Assessment

Assessment

Fill in the blanks

1. An array is a one kind of …….. that can store a ……. sequential collection of elements of the ……..
2. An array must be declared ……. it can be used in the program.

Multiple Choice Questions (MCQ)

1. After array declaration you must put—
   a) Comma (,) b) Semicolon (;) c) Desh (-) d) Hash symbol (#)
2. The elements of the array occupy-
   a) adjacent locations in memory b) Separate locations in memory
   c) No locations in memory d) None of these
3. What is the correct declaration of float type array?
   a) int num[10]; b) double test[10]; c) float value[10]; d) None of these

Exercises

1. What is an array? How do you declare an array in C language?
2. Explain the declaration and initialization procedure of an array with suitable example.
3. Discuss the pictorial presentation of an array with an example.

Lesson-2 Familiar With One Dimensional Array

Learning Outcomes

Outcomes

Upon completion of this lesson you will be able to

- Understand the various types of arrays
- Understand about one dimensional array
- Understand about two dimensional array
- Understand how to access the various arrays

	**Keywords**	**Classification, 1D array, Pictorial Representation**

CLASSIFICATION OF ARRAY

We know that Array is a kind of data structure that can store a fixed-size sequential collection of elements of the same type. It consists of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.
Generally array is two types such as:

i. One dimensional *(1D)* Array
ii. Two dimensional *(2D)* Array


Note it!

Two types of arrays: One dimensional (1D) and Two dimensional (2D).

ONE DIMENSIONAL *(1D)* ARRAY

One dimensional array is defined with a single subscript. In the one dimensional array, the size is specified by a positive integer (greater than 0) enclosed in single square [ ] brackets.
In general, the declaration form of one-dimensional array is expressed as

data-type array-name[array-size];

Where ***data-type*** is the valid data type in C and ***array-name*** is the user defined array name and ***array-size*** is a positive valued integer expression, which indicates the number of array elements.

EXAMPLE

The following declares an array called 'numbers' to hold 5 integers value:

int numbers[5];

SOME *1D* ARRAY DECLARATION EXAMPLE

1. If we want to store 100 integer type values in an *1D* array, we can write in C:
   int numbers[100];

2. If we want to store 50 float type values in an *1D* array, we can write in C:
   float marks[50];
3. If want to store 50 double type values in an *1D* array, we can write in C:
   double CGPA[50];
4. If we want to store 20 char type values in an *1D* array, we can write in C:
   char name[20];


Note it!

One dimensional array is defined with a single subscript.


Activity

1. Mention the significance of using a *1D* array in C language by your own concept.
   ……..
   ……..

_1D_ ARRAY INITIALIZATION PROCEDURE

One dimensional *(1D)* array is also known as simple array. The initialization procedure is also very easy. Whenever we declare an array, we initialize that array directly at compile time. Initializing *1D* array is called as compiler time initialization if and only if we assign certain set of values to array element before executing program.

There are two ways of *1D* array initialization such as:

- Array Size is Specified Directly
- Array Size is Specified Indirectly

In the Specified Directly method, we try to specify the array Size directly as follows:

```
int numbers[10]={20,40,100,50,30,250,450,80,70,300};

float numbers[5]={30.00,40.00,50.00,60.00,90.00};
```

In the Specified Indirectly method, we do not provide size to an array but instead we provide set of values to the array. The example of this method is as follows:

```
int number[ ]={20,40,100,50,30,250};
```

In this cases, compiler counts the number of elements written inside pair of braces { } and determines the size of an array. In this example, after counting the number of elements inside the braces { }, the size of array is considered as 6 during complete execution. This type of initialization scheme is also called as “Compile Time Initialization“.


Note it!

In Specified Directly method, specify the array Size directly. In Specified Indirectly method do not provide size to an array.

MEMORY REPRESENTATION OF *1D* ARRAY

Which are the following is true?

A. float CGPA[5]={2.50,3.30,3,40,4.00,3.75}
B. integer values[5]={20.00,30,50.00,3.45,100}
C. int NUM[ ]={30,30,40,50,60,70,80,100}

Solution:

Both A and C

In an *n*-element one dimensional array, the array elements are a[0],a [1],a[2],……..,a[n-1], as illustrated in figure 4.2.1.

Figure 6.2.1: Memory representation of one dimensional array with n-element

Here ***a*** is the array name and 0, 1, 2……..n-1 is the memory location number. In an one dimensional array the first location is 0, second location is 1, third location is 2 and so on and last location is n-1. This means that the first value is stored in location 0, second value is stored in location 1 and third value is stored in location 2 and so on and last value is stored in location n-1.

Now if we declare and initialize the following array with the name "numbers" with five integer values then memory representation in figure 4.2.2 would be as follows:

int numbers[5];
numbers[0]=100;
numbers[1]=200;
numbers[2]=400;
numbers[3]=250;
numbers[4]=500;

numbers [0]	numbers[1]	numbers[2]	numbers[3]	numbers[4]
100	200	400	250	500

Figure 6.2.2: Memory representation of one dimensional array with five (5) elements

Which are the following is true?

A. The last index of 1D array is *n-1*.
B. A 1D array consists of different types of values.
C. In Specified Indirectly method array size declaration is not mandatory.

Solution:

Both A and C

ELEMENT INSERTION PROCEDURE TO *1D* ARRAY

Programmatically, one of the nice things is that you can use a loop to manipulate or insert each element to one dimensional array. When an array is declared, the values of each element are not set to zero automatically. In some cases you want to initialize the array.

The following example (**Program-6.2.1**) illustrates insertion procedure of 10 numbers from the user in to the one dimensional array.

Program 6.2.1: Write a program that insert elements insertion to an array.

```
#include <stdio.h>
#include<conio.h>
 void main ()
  {
     clrscr();
     int numbers[ 10 ];                    /* numbers is an array of 10 integers */
     int i;
    printf("Enter the ten numbers:\n");
    for ( i = 0; i < 10; i++ )
      {
        scanf("%d",& numbers [ i ]);       /* insert elements in the  numbers array */
      }
   getch();
   }
```

……..

Output

```
Enter the ten numbers:
10
50
48
30
400
33
456
678
100
500
```

ELEMENTS DISPLAY OR PRINT FROM *1D* ARRAY

Programmatically, one of the nice things is that you can use a loop to print or display each element from one dimensional array.

The following example (**Program-6.2.2**) illustrates insertion procedure of 10 numbers from the user in to the one dimensional array as well as display or print stored numbers from array.

Program 6.2.2: Write a program that display or print elements from an array.

```
#include <stdio.h>
#include<conio.h>

void main ()
  {
     clrscr();
     int numbers[ 10 ];                    /* numbers is an array of 10 integers */
     int i,j;
    printf("Enter the ten numbers:\n");
    for ( i = 0; i < 10; i++ )
      {
```

```
        scanf("%d",& numbers [ i ]);       /* insert elements in the  numbers array */
      }
   printf("Array Values are:\n");
    for (j = 0; j < 10; j++ )
      {
        printf("%d    ", numbers [ j ]);       /* display or print elements from  array */
      }
  getch();
   }
```

……..

Output

```
Enter the ten numbers:
10
50
48
30
400
33
456
678
100
500
Array Values are:
10   50   48  30   400   33   456   678  100   500
```


Activity

1. Which lines are used to insert elements to *1D* array in program-1?
   ……. ……..
   ..……..
2. Which lines are used to print elements from *1D* array in program-2?
   ……..
   ……..

SOME EXAMPLE OF *1D* ARRAY

Program 6.2.3 Write a program to calculate summation of 10 integer's numbers.

```
#include <stdio.h>
#include<conio.h>

void main ()
  {
     clrscr();
     int numbers[ 10];
     int i,j,sum=0;
     printf("Enter the 10 numbers:\n");
    for ( i = 0; i < 10; i++ ) {
        scanf("%d",& numbers [ i ]);
      }
   for (i = 0; i < 10; i++ )
      {
        sum=sum+ numbers[i];
      }
    printf("The summation of ten numbers is: ");
```

```
    printf("%d", sum);
     getch();
 }
```

Output

```
Enter the 10 numbers:
10
20
30
40
50
60
70
80
90
100
The summation of ten numbers is: 550
```

Program 6.2.4 Write a program to calculate average of *n* integer numbers, which are given by user.

```
#include <stdio.h>
#include<conio.h>
void main ()
  {
    clrscr();
    int numbers[ 100];
     int i,j,n,sum=0,
    float average=0.0;
    printf("Enter the value of n:");
   scanf("%d",&n);
    printf("Enter the n numbers:\n");
    for ( i = 0; i < n; i++ )
       {
         printf("Enter the %d number:",i+1);
         scanf("%d",& numbers [ i ]);
        printf("\n");
       }
   for (j = 0; j < n;j++ )
       {
         sum=sum+ numbers[j];
       }
   average=sum/n;
    printf("\nThe average value is:%f ", average);
  getch();
 }
```

Output

```
Enter the value of n: 5
Enter the n numbers:
Enter the 1 number:10
Enter the 2 number:20
Enter the 3 number:30
Enter the 4 number:40
```

```
Enter the 5 number:50
The average value is: 30.00000
```

Program 6.2.5 Write a program to calculate the summation of following series.

$2^2+4^2+6^2+\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots+100^2$

```
#include <stdio.h>
#include<math.h>
#include<conio.h>
void main ()
  {
    clrscr();
    long int a[ 100];
    long int i,sum=0;
    for ( i = 2; i <= 100; i=i+2){
          sum=sum+pow(i,2);
      }
     printf("Summation of the series is:%ld",sum);
    getch();
  }
……
Output
Summation of the series is: 171700
```

Program 6.2.6 Write a program to find out the greatest number from n numbers.

```
#include <stdio.h>
#include<conio.h>
void main ()
  {
       clrscr();
       int a[ 100],i,n,max=0;
       printf("Enter the value of n\n:");
       scanf("%d",&n);
       max=a[0];
  for(i=0;i<n;i=i+1){
        printf("Enter the %d number:",i+1);
        scanf("%d",&a[i]);
        printf("\n")
       }
    for ( i = 1; i <=n; i=i+1){
          if(max<a[i]){
            max=a[i];
            }
       }
     printf("Greatest Number is:%d",max);
   getch();
  }
……..……..
Output
Enter the value of n: 5
Enter the 1 number:20
Enter the 2 number:400
```

```
Enter the 3 number:60
Enter the 4 number:600
Enter the 5 number:40
Greatest Number is:600
```

Summary  Summary	
In this lesson ▪ We have covered about *1D* array and its declaration and initialization. ▪ We also have understood that how *1D* array is presented graphically as well as we have seen that 1D array memory representation and some 1D example	

ASSIGNMENT

Assignment

1. Write a program in C, which finds out the minimum number from 200 integer numbers.
2. Write a program in C, that calculates the summation of all even numbers from 100 integer numbers.
3. Write a program in C that determines the summation of following series $1^1+2^2+3^3+4^4+\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots n^n$

Assessment

Assessment

Fill in the blanks

1. The lowest address corresponds to the …….. and the ……to the…… element.
2. In the ……array, the size is specified by a ……..
3. When an array is declared, the values of each element are not set to……..

Multiple Choice Questions (MCQ)

1. In the Specified Indirectly method, we do not provide …
   a) size to an array b) size to an data type
   c) character type values d) none of these
2. One dimensional *(1D)* array is also known.
   a) As data structure b) As simple array
   c) As OOP d) None of these
3. When an array is declared, the values of each element are not set to—
   a) One by default b) Zero by default c) Zero automatically d) None of these
4. Which is used to manipulate *1D* array?
   c) Loop d) Data type e) Pointer f) None of these

Exercises

1. How do you define 1D array? Mention the benefits of 1D array.
2. Explain the declaration and initialization procedure of 1D array with suitable example.
3. Explain the memory representation of one dimensional array with proper example.
4. Write a program in C to sort a list and find its median

Lesson-3 Familiar With Two Dimensional Arrays

Learning Outcomes

Outcomes

Upon completion of this lesson you will be able to

- Understand about two dimensional array.
- Understand how *2D* array is declared in program.
- Be aware of memory representation of *2D* array.

	Keywords	2D array, Declaration , Initialization, Example

TWO DIMENSIONAL *(2D)* ARRAY

Two dimensional *(2D)* arrays are defined with a double subscript. In the two dimensional arrays, the size is specified by two positive integer (greater than 0) enclosed in double square [ ][ ] brackets. Two Dimensional array stores the values in the form of matrix. Generally in *2D*, one subscript denotes the "**Row**" and another subscript denotes the "**Column**". So, *2D* arrays are generally known as matrix.

***2D*ARRAY DECLARATION**

In general, the declaration form of two-dimensional (*2D*) array is expressed as

data-type array-name[row-size][column-size];

Where ***data-type*** is the valid data type in C and ***array-name*** is the user defined array name and ***row size and column size*** is a positive valued integer expression, which indicates the row number and column number of array respectively.

Note it!

Two dimensional (2D) arrays are defined with a double subscript

EXAMPLE

The following declares two dimensional array called 'matrix' to hold total 12 integers values consisting four *rows* and three *columns*:

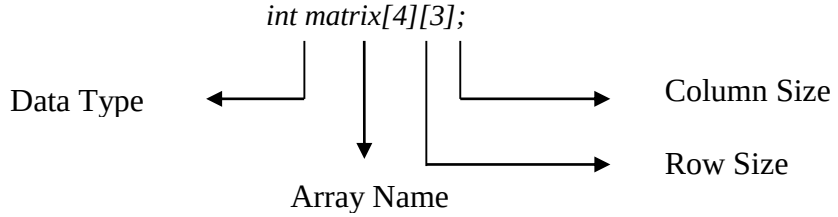

Figure 6.3.1 Array declaration procedure

Consider the following data table, which shows the value of sales of three items by three salesmans:

	ItemNo 1	ItemNo 2	ItemNo 3
Salesmans#1	**300**	**200**	**365**
salesmans#2	**200**	**100**	**150**
salesmans#3	**260**	**500**	**280**

Table 1: Sales table of items

The table contains a total 9 values, three in each line. We can think of this table as a matrix consisting of three (03) rows and three (03) columns. Here, each row represents the values of sales by a particular salesmans and each column represents the values of sales of a particular item. So we can represent the above concepts as a two dimensional matrix as follows:

int salesmans[3][3];

SOME *2D* ARRAY DECLARATION EXAMPLE

1. If we want to store five subjects numbers information's of 5 students, then 2D array is used as follows:

   int stumarks[5][5];

2. If we want to store three items by four sales mans of a inventory management system we declare 2D array as follows:

   int slalesman[4][3];


Activity

1. Mention the significance of using a *2D* array in C language by your own concept.
   ……..
   ……..
   ……
2. Suppose you are an assistant programmer of a software firm. Your firm has fifteen employees. Now you need to create a program that gives the monthly salary of all employees. For this purpose which type of array is required? And also show the declaration procedure of array.
   ……
   ……

***2D* ARRAY INITIALIZATION PROCEDURE**

There are many ways to initialize *2D* arrays. We have already described in previous section that when we give values during one dimensional array declaration, we don't need to mention dimension. But that's not the case with *2D* array; you must specify the second dimension even if you are giving values during the declaration. Some illustrations help you to initialization *2D* in many ways as follows:

int totalvals[3][3] = {1, 2, 3 ,4,5,6,7,8,9 } / Valid declaration*/*

int totalvals[4][3] = {1, 2, 3 ,4,5,6,7,8,9,10,11,12} / Valid declaration*/*

int totalvals [ ][2] = {1, 2, 3 ,4 } / Valid declaration*/*

int totalvals [ ][ ] = {1, 2, 3 ,4,5,6,7 } / Invalid declaration, you must specify second dimension*/*
int totalvals [2][ ] = {1, 2, 3 ,4 } /* Invalid declaration ,you must specify second dimension */
In 2D arrays for initializing, we can also need to assign values to each element of an array using the following syntax. In these initializations, comma (,) must be put after every initialization except last initialization which is shown in bellow:

1. *int array[3][2] = {*
   {1, 4},
   {5, 2},
   {6, 5}
   }

2. *float interest[4][3]={*
   {1.50, 1.20, 0.02},
   {0.50, 2.50, 4.00},
   {3.00, 2.50, 1.10},
   {1.10, 1.40, 2.60}
   };

MEMORY REPRESENTATION OF 2D ARRAYS

2D arrays are stored in memory as shown in Fig. 2. A two-dimensional array can be considered as a table which will have x number of rows and y number of columns. The memory representation of a 2D array ***a***, which contains four rows and three columns can be shown as follows –

int a[4][3];

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

Figure 6.3.2: *2D* array memory representation

Here, every element in the array **a** is identified by an element name of the form **a[i][j]**, where '**a**' is the name of the array, and '**i**' and '**j**' are the subscripts that uniquely identify each element in 'a'. In *2D*, the first index selects the row and second index selects the column within that row. Here, a[0][0] means 0^{th} row and 0^{th} column, a[0][1] means 0^{th} row and 1th column, a[2][1] means 2^{nd} row and 1^{st} column and so on.


Note it!

A two-dimensional array can be considered as a table which will have x number of rows and y number of columns

In *2D* arrays are stored in contiguous memory location row wise that is, array element of first location a[0][0] can be stored any memory location (i.e., address), then next element a[0][1] will be stored next memory location and so on. Since elements are stored row wise, so after elements of first row are stored in appropriate memory location, elements of next row get their corresponding memory locations. For instance, if array element of first location a[0][0] is stored in memory address 5000, then next array element a[0][1] is stored next memory location like 5002. Here next memory address is calculated as follows:

a[0][1]=a[0][0]+ size of data type
a[0][2]=a[0][1]+ size of data type
a[1][0]=a[0][2]+ size of data type

Here, we declare integer array *a* so each element requires 2 bytes of memory, for float data type it is required 4 bytes of memory and so on. So ultimate calculation is as follows:

a[0][1]=a[0][0]+ size of data type
=5000+ 2
=5002
a[0][2]=a[0][1]+ size of data type
=5002+ 2
=5004
a[1][0]=a[0][2]+ size of data type
=5004+ 2
=5006

For figure 6.3.2, the full memory location or address calculation is presented in the following table 2:

Elements	Memory location/Address	Elements	Memory location/Address
a[0][0]	5000	a[2][0]	5012
a[0][1]	5002	a[2][1]	5014
a[0][2]	5004	a[2][2]	5016
a[1][0]	5006	a[3][0]	5018
a[1][1]	5008	a[3][1]	5020
a[1][2]	5010	a[3][2]	5022

Table 2: Memory location for array *int a[4][3]*


Study skills

1. Which are the following is true?
   A. float CGPA[5][2]={2.50,3.30,3,40,4.00,3.75}
   B. int matrix[2][3]={20,30,50,45,100,90}
   C. int NUM[ ][ ]={30,30,40,50,60,70,80,100,70,40,44,23}

Solution:
Both B and C because, in B, the matrix size is identified, the row size is 2 and column size is 3, so the stored values are total is 6 on the other hand, in C array sizes are not identified, but it is identified by initialization values by compiler.

2. Which are the correct statements?
   A. In 2D array, array size must be declared before access.
   B. In 2D array, initialization is not mandatory if it is not declared array size.
   C. A 2D array can be considered as a table, which has x number of rows and y number of columns

Solution:
Both A and C

ELEMENT INSERTION PROCEDURE TO *2D* ARRAY

Programmatically, one of the nice things is that you can use two loops to manipulate or insert elements to *2D* array. When an array is declared, the values of each element are not set to zero automatically. In some cases you want to initialize the array. The following example **Program 6.3.1** illustrates insertion procedure of 3 subject's marks of 5 students in to the *2D* array. The marks are given by user.

Program 6.3.1Write a program to insert elements to 2D array.

```
#include <stdio.h>
#include<conio.h>
void main ()
```

```
  {
     clrscr();
     int stumarks[3][5];
     int row,col;
     printf("Enter the  3 subjects mark of 5 students:\n");
   for ( row = 0; row < 3; row++ )
     {
          for(col=0;col<5;col++)          /* insert marks in to the 2D array */
          {
           printf("Enetr the marks of %d subject of %d student: ",row+1, col+1);
           scanf("%d",&stumarks[row][col]);
           }
     }
  getch();
   }
```

……..

Output

```
Enter the  3 subjects mark of 5 students:
Enetr the marks of 1 subject of 1 student: 50
Enetr the marks of 1 subject of 2 student: 60
Enetr the marks of 1 subject of 3 student: 70
Enetr the marks of 1 subject of 4 student: 44
Enetr the marks of 1 subject of 5 student: 56
Enetr the marks of 2 subject of 1 student: 78
Enetr the marks of 2 subject of 2 student: 56
Enetr the marks of 2 subject of 3 student: 55
Enetr the marks of 2 subject of 4 student: 34
Enetr the marks of 2 subject of 5 student: 78
Enetr the marks of 3 subject of 1 student: 88
Enetr the marks of 3 subject of 2 student: 90
Enetr the marks of 3 subject of 3 student: 54
Enetr the marks of 3 subject of 4 student: 55
Enetr the marks of 3 subject of 5 student: 76
```

ELEMENTS DISPLAY OR PRINT OR ACCESS FROM *2D* ARRAY

Programmatically, one of the nice things is that you can use two loops to print or display or access each element from *2D* array. The following example **Program 6.3.2** illustrates display or print stored numbers from *2D* array.

Program 6.3.2 Write a program to display or print store numbers from 2D array

```
#include <stdio.h>
#include<conio.h>
void main ()
  {
     clrscr();
     int stumarks[2][3];
     int row,col;
    printf("Enter the  2 subjects marks of 3 students:\n");
    for ( row = 0; row < 2; row++ )
      {
         for(col=0;col<3;col++)
           {                                                    /* insert marks in to the 2D array */
```

```
            printf("Enetr the marks of %d subject of %d student: ",row+1, col+1);
            scanf("%d",&stumarks[row][col]);
          }
      }
      printf("The marks of students are as follows:\n");
     for ( row = 0; row < 2; row++ )
        {
         for(col=0;col<3;col++)
          {                                          /* Display or print marks from the 2D array */
          printf(" Marks of  %d subject of %d student is: %d ",row+1,col+1,
                  stumarks[row][col]);
          printf("\n");
         }
       printf("\n");
      }
   getch();
   }
```

……

Output

```
Enter the  2 subjects marks of 3 students:
Enter the marks of 1 subject of 1 student: 30
Enter the marks of 1 subject of 2 student: 50
Enter the marks of 1 subject of 3 student: 70
Enter the marks of 2 subject of 1 student: 55
Enter the marks of 2 subject of 2 student: 60
Enter the marks of 2 subject of 3 student: 55
The marks of students are as follows:
 Marks of  1 subject of 1 student is: 30
 Marks of  1 subject of 2 student is: 50
 Marks of  1 subject of 3 student is: 70

 Marks of  2 subject of 1 student is: 55
 Marks of  2 subject of 2 student is: 60
 Marks of  2 subject of 3 student is: 55
```


Activity

1. Which lines are used to insert elements to *2D* array in program-3?
   ……..……..……..……..
   ……..……..……...……..
2. Which lines are used to print elements from *2D* array in program-4?
   …..…..……..……..
   …..……. …….. ……..……..
3. Suppose you have a *2D* array with the named *" float CGPA[4][4]"* and its first element is stored at memory location 1004, what will be the addresses of remaining elements?
   ……..…….

SOME EXAMPLE OF *2D* ARRAY

Program 6.3.3 Write a program in C that takes 12 integer numbers from user and display them as 4X3 matrix.

```
#include<stdio.h>
#include<conio.h>
```

```
void main()
{
    clrscr();
    int numbers[4][3];
    int row,col;
    printf("Enter the  12 values:\n");
   for(row=0;row<4;row++)
    {
       for(col=0;col<3;col++)
        {
          scanf("%d",&numbers[row][col]);
        }
    }
  printf(" Matrix as follows:\n ");
  for(row=0;row<4;row++)
   {
      for(col=0;col<3;col++)
       {
           printf("%d  ",numbers[row][col]);
       }
     printf("\n");
   }
getch();
}
```

…….

Output

```
Enter the  12 values:
10
20
30
40
50
60
70
80
90
45
67
88
Matrix as follows:
10  20  30
40 50  60
70  80  90
45 67  88
```

Program 6.3.4 Write a program in C, that calculates the summation of all values of a 3X3 matrix.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    int numbers[3][3];
```

```
        int row,col,sum=0;
        printf("Enter the  9 values:\n");
    for(row=0;row<3;row++)
     {
        for(col=0;col<3;col++)
        {
          scanf("%d",&numbers[row][col]);

        }
     }
     printf(" Summation of all values of 3X3 Matrix is: ");
     for(row=0;row<3;row++)
       {
          for(col=0;col<3;col++)
            {
              sum=sum+ numbers[row][col];
            }
       }
    printf("%d",sum);
    getch();
    }
```

……..

Output

```
Enter the  9 values:
10
20
30
40
50
60
70
80
90
Summation of all values of 3X3 Matrix is: 450
```

Program 6.3.5 Write a program in C that calculates of total items of individual salesmans according to following table:

	Item No 1	Item No 2	Item No 3
Salesmans#1	300	200	365
salesmans#2	200	100	150
salesmans#3	260	500	280

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
int salsmans[3][3];
int row,col,sum=0;
for(row=0;row<3;row++)
 {
    for(col=0;col<3;col++)
    {
```

```
            printf("Enter the value of  item number %d : of sales man %d ",col+1,row+1);
            scanf("%d",&salsmans[row][col]);
            printf("\n");
           }
      }
     for(row=0;row<3;row++)
        {
          printf("\n");
          printf(" The total items of sales man %d is: ",row+1);
          for(col=0;col<3;col++)
            {
              sum=sum+salsmans[row][col];
          }
          printf("%d\n",sum);
          sum=0;
        }getch();
     }
```

……

Output

```
Enter the value of  item number 1 : of sales man 1 300
Enter the value of  item number 2 : of sales man 1 200
Enter the value of  item number 3 : of sales man 1 365
Enter the value of  item number 1 : of sales man 2 200
Enter the value of  item number 2 : of sales man 2 100
Enter the value of  item number 3 : of sales man 2 150
Enter the value of  item number 1 : of sales man 3 260
Enter the value of  item number 2 : of sales man 3 500
Enter the value of  item number 3 : of sales man 3 280
The total items of sales man 1 is: 865
The total items of sales man 2 is: 450
The total items of sales man 3 is: 1040
```

Summary

Summary

In this lesson

- We have covered about C language *2D* array and its declaration.
- Moreover we have learned how *2D* is initialized.
- We also have understood that how *2D* array is presented graphically as well as we have seen that *2D* array memory representation.
- We also have seen some *2D* example, which knowledge is helped to implement another such problems.

ASSIGNMENT

Assignment

1. Write a program in C, that calculates summation of diagonal elements of a 3X3 matrix
2. Write a program in C, that calculates summation of non-diagonal elements of a 3X3 matrix
3. Suppose you are a programmer of a software firm. Your firm has fifteen employees. Now you need to write a program that gives the total salary of four month of all employees.

4. The daily maximum temperature recorded in 5 cities during the month of December have been tabulated as follows:

City/day → ↓	1	2	3	4	……..	30	31
1	30	33	32	25	……..	27	29
2	28	29	30	31	……..	30	29
3	25	25	27	29	……..	29	29
. . .					……..		
9							
10	25	26	27	27	……..	28	28

Now write a program in C, to read the table elements into 2D array named "Temperature", and to find the city and day corresponding to

a) The highest temperature.
b) The lowest temperature.

5. The annual examination results of 5 students are listed as follows:

Student Id	Subject-1	Subject-2	Subject-3	Subject-4
1	45	50	70	60
2	47	57	69	70
3	78	67	89	77
4	66	55	45	40
5	67	68	69	70

Write a program to read the data from the above table and determine the following:

a) Total marks obtained by each student
b) The highest marks in each subject and the student id of the student who scored it.
c) The student who obtained the highest total marks.

6. Given the two matrix A and B as follows:

$$A=\begin{pmatrix}5 & 4 & 5\\ 3 & 2 & 6\\ 2 & 3 & 7\end{pmatrix} \text{ and } B=\begin{pmatrix}3 & 4 & 2\\ 3 & 1 & 3\\ 4 & 5 & 2\end{pmatrix}$$

Now write a program in C, that will read the values of elements of A and B and produce the product matrix C.

Assessment

Assessment

Write "T" for true and "F" for false for the following sentences

1. Two dimensional *(2D)* arrays are defined with a double subscript
2. *2D* arrays are generally known as matrix.
3. In *2D* initialization, array size must be initialized before used.

1. Identify errors, if any, in each of the following array declaration statements:
   i. int score(100);
   ii. float values[10,15];
   iii. float average[50],[30];
   iv. char names[15]
   v. int summation[ ];
2. Identify errors, if any, in each of the following array initialization statements:
   i. static int num[ ]={0,0,0,0,0};
   ii. float items[2][3]={2,3,4,5,6,7};
   iii. static char test[ ]={'N', 'A', 'M', 'E'};
   iv. double salary[2,4]={(30.98,45.00,4.00,50.00),{100.00,3.00,50.89,40.00}};
   v. float result[10]=0;
   vi.
```
for(i=1;i<=5;j++)
{
  for(k=0;k<=5;i++)
    a[i][j]=0;
}
```

Multiple Choice Questions (MCQ)

1. Two Dimensional array stores the values in the form of-
   a) Pointer
   b) Matrix
   c) Colum
   d) None of these
2. Which is the valid initialization in 2D?
   a) Matrix[2][2]={4 4 5 6}
   b) Matrix[3][3]={5,6,7,3,8,9,10,11,12};
   c) float salary[3][2]={{1000.00,2000.50},{500.34,800.25},{3000.90,4000.00}};
   d) float salary[3][2]={{1000.00,2000.50},{500.34,800.25},{3000.90,4000.00},};

Exercises

1. How do you define *2D* array? Mention the benefits of *2D* array.
2. Distinguish between *1D* and *2D* array with proper example.
3. Explain the declaration and initialization procedure of *2D* array with suitable example.
4. Explain the memory representation of two dimensional arrays with proper example.

Lesson-4 Introduction to Strings

Learning Outcomes

Outcomes

Upon completion of this lesson you will be able to

- Understand about basic concept on Strings.
- Understand how strings are declared in C language.
- Understand how strings are initialized and accesses in C language.

	**Keywords**	**Strings, Declaration , Initialization, Example**

STRINGS

Usually, a string is a sequence of characters. In C language, we can define that, "***Strings are array of characters terminated by a null character '\0'*** ". Actually, strings are one dimensional array of characters. In such cases, strings are operated by programmers according to the requirement of problems. Any group of characters defined between double quotation marks (" ") is a constant string. A constant string example is given below:

"Bangladesh Open University"

If you need to include a double quote in the string, then you must be included a back slash as follows:

"\ "Bangladesh Open University\" "

If you want to print or display a string with double quote then you should write the following line:

printf("\"Hello ! Bangladesh Open University!!\" ");

The output of this line is:

"Hello ! Bangladesh Open University!!"

But if you write the following line:

printf("Hello ! Bangladesh Open University!!");

Then the output will be:

Hello ! Bangladesh Open University!!

Generally, character type strings are frequently used to construct significant and understandable programs. The common functions executed on character strings are as follows:

i. Reading and writing strings
ii. Joining strings together
iii. Copying one string to another string
iv. Comparing strings for equality
v. Pull out or extracting a portion of a string

DECLARING STRING VARIABLES

A string variable is any suitable C variable name and is all the time declared as an array. We have already known about data types in C. In C language string data type is not supported. For this reason, we cannot declare string using string data type. So instead of we use array of type character to create string. The general structure or form of declaration of a string variable is as follows:

char String_Variable_name[size];

Here, ***char*** is the data type, which is only used for string declaration, ***String_Variable_name*** is the user defined array name and ***size*** determines the number of characters in the ***String_Variable_name.***


Note it!

Strings are array of characters terminated by a null character '\0'.

DECLARATION EXAMPLE

Some examples are as follows:

```
char country[20];
char city[10];
char uniname[25];
```

Here, *country, city and uniname* are the string array variables name respectively. When the compiler allocates a character string to a character array, it automatically supplies a *null character ('\0')* at the end of the string. As a result, the ***size*** should be equal to the highest number of characters in the string ***plus*** one (1).


Note it!

When the compiler allocates a character string to a character array, it automatically supplies a null character ('\0') at the end of the string.

INITIALIZING STRING VARIABLES

Character arrays are initialized when they are declared. In C programming language a character array to be initialized in either of the following two structures:

```
char country[11]= "BANGLADESH";
char country[11]={'B', 'A', 'N', 'G', 'L', 'A', 'D', 'E', 'S', 'H', '\0'};
```

The reason here that, the variable *country* had to be 11 elements long are that the string BANGLADESH contains 10 characters and one element space is supplied for the *null* terminator ('\0'). It is noted that, when we initialize a character array by listing its elements, we must supply explicitly the *null* terminator ('\0'). In C programming language, it permits us to initialize a character array without identified the number of elements. In such cases, the size of the array will be decided automatically, based on the number of elements initialized. Consider the following example:

```
char department[ ]={'S', 'S', 'T', '\0'};
```

This example statement defines the array *department* as a three (3) element array.


Study skills

Which are the correct statements of the following?

A. Strings are summation of characters
B. '\o' is the null terminator at the end of the strings
C. Strings are one dimensional array of characters
D. String data type is available in C language
E. When the compiler allocates a character string to a character array, it automatically supplies a null character at the end of the string.
F. int name[8]= "ENGLAND";
G. char city[9]= "NEW YORK";
H. char department[11]= {'O', 'P', 'E', 'N', ' ', 'S', 'C', 'H', 'O', 'O', 'L'};

Solution:

C, E, and G are the correct statements (Remember your study)


Activity

1. Mention the significance of strings in C language by your own concept.
……..
……..
……

2. Suppose you are an assistant programmer of a software firm. Your firm has fifteen employees. Now you need to create a program that stores all employee names in *Empname* character type array. Mention the array declaration for this program.
……
……

READING STRINGS FROM USER

In C programming language, two concepts are imagined to read strings from user or terminal such as *(i) Reading words and (ii) Reading a line of text*

Reading Words

The well-known input function ***scanf( )*** is used with ***%s*** format specification to read in a string of characters from user or terminal.

Consider following example to read city name from user:

```
char cityname[12];
scanf("%s",cityname);
```

However, the problem with this ***scanf( )*** is that, it terminates its input on first white space it finds. Here white space means that, blanks, tabs, carriage return (Enter), form feeds, and new line. For that reason, if the following line of text is typed by user at the terminal,

DHAKA CITY

Then only the string **DHAKA** will be read in to the array ***cityname*.** As a result, the string would get cut off. The blank space after the word **DHAKA** will terminate the string.

In this situation, if you want to read the complete line using ***scanf( )*** function, then you may use more than one character arrays as you need.

It is noted that, in the case of character arrays, the **ampersand sign (&)** is not used before the variable name. The ***scanf( )*** function automatically terminates the string that is read with a *null* character.


Note it!

In the case of character arrays, the ampersand sign (&) is not used before the variable name.

Consider the following example:

Program 6.4.1 Write a program to read a series of words from user at terminal using scanf( ) function.

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
        char uniname1[40];
        char uniname2[40];
        char uniname3[40];
        printf("Enetr your Text: \n");
        scanf("%s",uniname1);
        scanf("%s",uniname2);
        scanf("%s",uniname3);
        printf("\n");
        printf("After Insertion text: \n");
        printf("%s",uniname1);
        printf(" ");
        printf("%s",uniname2);
       printf(" ");
        printf("%s",uniname3);
        getch();
}
```

……

Output

```
Enetr your Text:
Bangladesh Open University
After Insertion text:
Bangladesh Open University
```

Or

If you enter the input as follows:

```
Enetr your Text:
Bangladesh
Open
University
After Insertion text:
Bangladesh Open University
```

Reading a Line of Text

Practically, many programming applications, we need to read in an entire line of text from the user. It is not possible to use ***scanf( )*** function to read a line containing more than one word. We have already known about ***getchar( )*** function, that reads single character from the user. So by using this ***getchar( )*** function repeatedly to read successive single characters from the input and place them into a character array. Therefore, a entire line of text can be read and stored in an array. In this case, when the new line character (‘\n’) is entered, then the reading is terminated and the *null* character (‘\0’) is inserted at the end of the string. Consider the following program:

Program 6.4.2 Write a program to read a line of text containing a series of words from the user.

```
#include<stdio.h>
#include<conio.h>

void main()
{
      clrscr();
      char subject[81];
      char character;
      int count=0;
      printf("Enter your Text: Press<Return/Enter> at end \n");
do{
      character=getchar();
      subject[count]=character;
      count++;
   }while(character!='\n');
   count=count-1;
    subject[count]='\0';
    printf("After Inserting line of Text: \n");
    printf("\n%s\n",subject);
getch();
}
```

……

Output

```
Enter your Text: Press<Return/Enter> at end
Programming C is interesting subject
After Inserting line of Text:
Programming C is interesting subject
Enter your Text: Press<Return/Enter> at end
I am a man, I am a student of BOU. My subject is DSCA-1302.
After Inserting line of Text:
I am a man, I am a student of BOU. My subject is DSCA-1302
```

gets( ) and puts( ) Functions

We have already seen the problems of ***scanf( )*** function in previous section. To solve this ***scanf()*** function problem, we use the function ***gets().*** This is the built-in standard library function, which is belongs to ***stdio.h*** header file. This function receives series of characters words that is this function waits until the user hits the enter key before it cuts off the input. This function writes a string to ***stdout*** up to but not including the null character. Normally a newline character is appended to the output. The general syntax of ***gets()*** is as follows:

gets(char_array_variable);

gets() function takes just one argument such as a char pointer, or the name of a char array, it automatically prints out a newline character, making the output a little neater.

puts() function is similar to *gets()* function. It is also a built-in function and belongs to *stdio.h* header file. It has also one argument such as a char pointer or the name of a char array. This also automatically adds a newline character after printing out the string. This is the alternative of *printf()* function. Sometimes this can be a disadvantage, so *printf()* could be used instead. The general syntax of ***puts()*** is as follows:

puts(char_array_variable);

The following is shown as the *gets()* and *puts()* example

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    char names[100];
     printf("Type your full name: ");
     gets(names);
     printf("\n\n");
     puts(names);
     getch();
}
```

Summary  Summary	
In this lesson we have ▪ Learned about Strings variable and its declaration and its initialization. ▪ Also understood how strings are read from user and disply to screen. ▪ Uunderstood significance of *gets( )* and *puts( )* library functions.	

ASSIGNMENT

Assignment

1. Find out the errors and mention why errors are occurred for the following:
   A. c=c-1;
      line[c]= '\o'
      printf("%d",line);
   B. string names[20];
      scanf("%s",&name[i]); printf("%s",&names);
   C. char dept[5]={"'D', 'C' 'S' 'A'"};
      while(1)
      {
       char val=puts(dept);
      }
   D. char uniname[10]= "C PROGRAMMING";
      printf(%s,uniname);
2. What will be the output for the following program segments:
   A. char city[9]= "NEW YOURK";
      printf("%s",city);
   B. void main()
      {
      char word1[40]= "Oxford", word2[40]= "University",word3[40]= "London";
      printf("\n"); printf("output will be:\n");
      printf("%s",word1);
      printf(" ");printf("%s",word2);
      printf(" ");
      printf("%s",word3);
      }

Assessment

Assessment

Write "T" for true and "F" for false for the following sentences

1. Strings are one dimensional array of characters.
2. In C language string data type is supported.
3. In C programming language, it permits us to initialize a character array without identified the number of elements.
4. When the new line character ('\n') is entered, then the reading is terminated and the *null* character ('\0') is inserted at the end of the string.

Multiple Choice Questions (MCQ)

1. Strings are array of characters terminated by
   a) a null character '\0' b) a '\o' character c) a integer value d) any character value

2. If you want to print a string with double quote, what is the correct sysntax?
   a) printf("open university") b) printf("\n open university");
   c) printf("\ "open university\" "); d) printf("\open university/" ");
3. gets() and puts() function are belong to
   a) ctype.h b) string.h c) stdio.h d) math.h
4. scanf( ) function is used to read—
   a) A line containing more than one word.
   b) A first single word from a line
   c) Two word from a line
   d) None of these
5. gets() and puts() functions have—
   a) Only one argument b) Two arguments
   c) No arguments d) None of these
6. Which initialization is valid?
   a) int name[ ]= "DCSA";
   b) char book[5]= {"D,C,S,A"};
   c) char book[4]= "DCSA";
   d) char book[5]={'D', 'C', 'S', 'A', '\0'};
7. The ampersand sign (&) is not used before
   a) integer variable name b) the string variable name
   c) float type variable name d) char type variable name

Exercises

1. What is string? How do you declare and initialize strings in C programming language?
2. Mention the problems of *scanf( )* function with suitable example.
3. Explain the procedure of reading strings from user with proper example.
4. Discuss the significance *of gets( ) and puts( )* functions with an example.
5. Write a program to read a line of text containing a series of words from the user.

Lesson-5 Understanding String Handling Functions

Learning Outcomes

Outcomes

Upon completion of this lesson you will be able to

- Learn about syntaxes of some string handling functions.
- Learn how string handling functions are used in program with example.

	**Keywords**	**Strings, Library functions , Concatenation, String compare, String copy, String length**

STANDARD LIBRARY FUNCTIONS

Basically a standard library in computer programming is the library that made obtainable across implementations of a programming language. These libraries are traditionally expressed in programming language specifications. Standard library functions in C programming are built-in functions. Many data definitions and function prototypes of these functions are written in their respective header file. However, some of the commands in C language are not really commands at all, but these are functions, is called library functions. For instance, we already know some library functions such as *printf( ), scanf( ), rand( ), gets( ), puts( )* etc. These standard functions are included with C compilers and while these are not really part of the language, but these are not rewritable.

STRING HANDLING FUNCTIONS

Naturally, strings are habitually needed to be operated or manipulated by programmer or user according to the need of problems. All strings manipulation process can be done manually by the programmer but, this makes programming complex and large. To solve this problem, the C library sustains a large number of string handling functions. Sting manipulation library functions are belongs to ***"string.h"*** header file. So when you use string library functions in your program, you must be included ***"string.h"*** header file.

Following are the most commonly used string handling functions:

Function Name	Description
strcat ()	This is used to concatenate two strings
strcmp( )	This is used to compares two strings
strcpy()	This is used to copies one string over another string
strlen()	This is used to determine the length of a string

We will describe how each of these functions can be used in the processing of strings.

Note it!

Standard library functions in C programming are built-in functions

strcat( ) Function

strcat() means string concatenation. This function is used to concatenate (Join) two strings together. This function takes two arguments, i.e., two strings and resultant string is stored in the first string specified in the argument.

The general form of *strcat( )* function is as follows:

strcat(string1, string2);

Here, string1 and string2 are character arrays which are already described in previous lesson. After executed *strcat( )* function , string2 is appended to string1. So the resultant value is stored in string1. In this case removing the null character at the end of string1 and placing string2 from there. C language also permits nesting of *strcat()* functions which is shown in example-2. Nesting function syntax is as follows:

strcat(strcat(string1,string2),string3);

The nesting function is allowed and concatenates all the three strings together and the resultant string is stored in string1.

Some *strcat()* function example are given below:

Program 6.5.1 Write a C program to concatenate two strings

```
#include <stdio.h>
#include <string.h>
#include<conio.h>
void main( )
{
    clrscr();
   char string1[ ]="Welcome Bangladesh Open ";
   char string2[ ]=" University";
   strcat(string1,string2);
   puts(string1);
   getch();

}
……..
Output:
Welcome Bangladesh Open University
```

Program 6.5.2 Write a C program to concatenate three strings

```
#include <stdio.h>
#include <string.h>
#include<conio.h>
void main( )
{
   clrscr();
   char string1[ ]="Welcome Bangladesh Open ";
   char string2[ ]="University";
   char string3[ ]=" DCSA-1301";
   strcat(strcat(string1,string2),string3);
   printf("%s",string1);
  getch();
}
```

……

Output:

Welcome Bangladesh Open University DCSA-1301

strcmp() Function

strcmp( ) means string compare. This function is used to compare two stings, recognized by the function arguments and has two arguments string1 and string2 and the *strcmp()* function returns an integer value as follows:

- if Return value < 0 then it indicates that string1 is less than string2.
- if Return value > 0 then it indicates that string1 is greater than string2.
- if Return value = 0 then it indicates that string1 is equal to string2.

The general syntax of *strcmp( )* function is:

strcmp(string1,string2);

here, string1 and string2 may be character type string variables or string constant. Some examples are as follows:

```
strcmp(name1,name2);
strcmp(name1, “AMRAN”);
strcmp(“RAM”, “ROM”);
```

Some examples are given below:

Program 6.5.3 Write a C program to compare two strings

```
#include <stdio.h>
#include <string.h>
#include<conio.h>

void main ()
{
  clrscr();
  char string1[15]= “abcdef”
  char string2[15]= “ABCDEF”;
  int ret;
  ret = strcmp(string1, string2);
  if(ret < 0)
```

```
  {
    printf("string1 is less than string2");
  }
  else if(ret > 0)
  {
    printf("string1 is greater than string2");
  }
  else
  {
    printf("string1 is equal to string2");
  }
  getch();
}
```

Output:

string1 is greater than string2

Program 6.5.4 Write a program to compare two names either equal or not

```
#include <stdio.h>
#include <string.h>
#include<conio.h>
void main()
{
      clrscr();
      char name1[30],name2[30];
      printf("Enter first name as string: ");
      gets(name1);
      printf("\nEnter second name as string: ");
     gets(name2);

    if(strcmp(name1,name2)==0)
         printf("Both names are equal !!! ");
   else
      printf("Name  are unequal !! ");
   getch();
}
```

Output:

Enter first name as string: MAMUN
Enter second name as string: MAMUN
Both names are equal !!!

strcpy() Function

strcpy() means string copy. This function is used to copies one string over another string. It works like assignment operator. It has two parameters. One is destination and another is source. Destination may be character type array variable and source is character type array variable or c string constant. The content of source variable or string constant is copied to the destination array variable. The general syntax of *strcpy( )* function is as follows:

strcpy(destination array, source arry/source_string);

Consider the following examples:

strcpy(city, "Gazipur");
strcpy(city1,city2);

from the example, the statement-1 will assign the string "**Gazipur**" to the string variable ***city*** and the statement-2 will assign the contents of the string variable ***city2*** to the string variable ***city1***. Here, the size of the array ***city1*** should be large enough to receive the contents of ***city2.***

Program 6.5.5: Write a program to copy the contents of source to destination

```
#include <stdio.h>
#include <string.h>
#include<conio.h>

void main()
{
  clrscr();
  char city1[40];
  char city2[100];
  strcpy(city1, "Gazipur");
  strcpy(city2, city1);
  printf("Final copied string to city2 is  : %s\n", city2);
}
```

Output:
Final copied string to city2 is: Gazipur

Program 6.5.6: Write a program to read the string from user and copy the contents to another variable

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
 void main()
{
   clrscr();
   char str1[50],str2[50];
   printf("Enter your string: ");
   gets(str1);
   strcpy(str2,str1);
   printf("Copied string is : ");
   puts(str2);

}
```

Output:
Enter your string: C programming Subject
Copied string is : C programming Subject

strlen() Function

strlen( ) means string length. This is used to determine or count and return the number of characters in a string. The return value is received by integer type variable. The general structure of ***strlen( )*** is as follows:

count = strlen(string);

Here, ***string*** is the user provided string whose length is to be computed and ***count*** is integer type variable which receives the value of the length of ***string and*** counting is halt when first null character ('\0') is found.

Program 6.5.7 Write a program to count the length of string

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main ()
{
  clrscr();
  char string[50];
  int count;
  strcpy(string, "School of Science and Technology");

  count = strlen(string);
  printf("\n Length of string is:  %d ", count);

}
```

Output:
Length of string is: 32


Study skills

Which are the correct statements of the following?

A. rand() is a library function
B. '\o' is the null terminator at the end of the strings
C. *strcpy()* is used to copy one string to another string
D. String compare syntax is as follows:
   strcmp(string1,string2,string3);
E. String length function has two parameters.
F. String concatenation function has two parameters.
G. In *strlen()* function returns integer number.

Solution:
A, F, and G are the correct statements (Remember your study)


Activity

1. Mention the significance of strings library function in C language by your own concept.
2. Write a program to identify the length of the following string
   "Thank you!! You are welcome"

Summary


Summary

In this lesson we have

- Learned about Strings library function.
- Learned the syntax of string library functions.
- Also understood how strings are read from user and disply to screen.
- Mentioned significance of *strcat( ), strcpy( ),strcmp( ) and strlen( )* functions with examples.

ASSIGNMENT

Assignment

1. Write a program that reads the 10 students name and their phone number, the program counts the character of every students name and phone number.
2. Write a program that read your first name, middle name and last name and store them to separate three variables and join them with space after you first name, middle name and last name and display your full name.
3. Write a program which will read a text and count total number of characters of the text.

Assessment

Assessment

Write "T" for true and "F" for false for the following sentences

1. Rands() is a string library function
2. The *strcmp()* function returns an integer value.
3. In C *strcpy()* function works like assignment operator.
4. When the new line character ('\n') is entered, then the reading is terminated and the *null* character ('\0') is inserted at the end of the string.

Multiple Choice Questions (MCQ)

1. Which one is the header file for *strcmp()* function?
   a) sting.h b) strlib.h c) stdio.h d) none of these
2. *strlen()* function is used to
   a) Determine the number of character in a line
   b) Identify the value of character
   c) Compare two strings
   d) None of these
3. Standard functions are included with C compilers, which are –
   a) modifiable in program b) rewriteable c) not rewritable d) none of these
4. *strcat()* function means
   a) comparing two strings
   b) joining two or more strings
   c) copy source strings to destination
   d) none of these

Exercises

1. What do you mean by library function? Mention the significance of string library functions.
2. Distinguish between *strcmp()* and *strcpy()* functions with their syntax.
3. Assume str1, str2, and str3 are three string variables. Write a program to read two strings and store these into str1 and str2 and compare whether they are equal or not. If they are not, join them together. Then copy the contents of str1 to the variable str3. At the end, the program should print the contents of all the three variables and their lengths.