UNIT-III Arrays and Strings

Arrays indexing ,memory model, Programs with array of integers, two dimensional arrays, Introduction to strings.

Concept of Arrays:

When we work with a large number of data values we need that any number of different variables. As the number of variables increases, the complexity of the program also increases and so the programmers get confused with the variable names. There may be situations where we need to work with a large number of similar data values. To make this work easier, C programming language provides a concept called "Array".

Arrays:

- An array is defined as the collection of elements of similar data type which are stored at contiguousmemory locations.
- Arrays can store the primitive type of data such as int, char, double, float, etc. It alsohas the capability to store the derived data types, such as pointers, structure, etc.
  Syntax for declaring array:

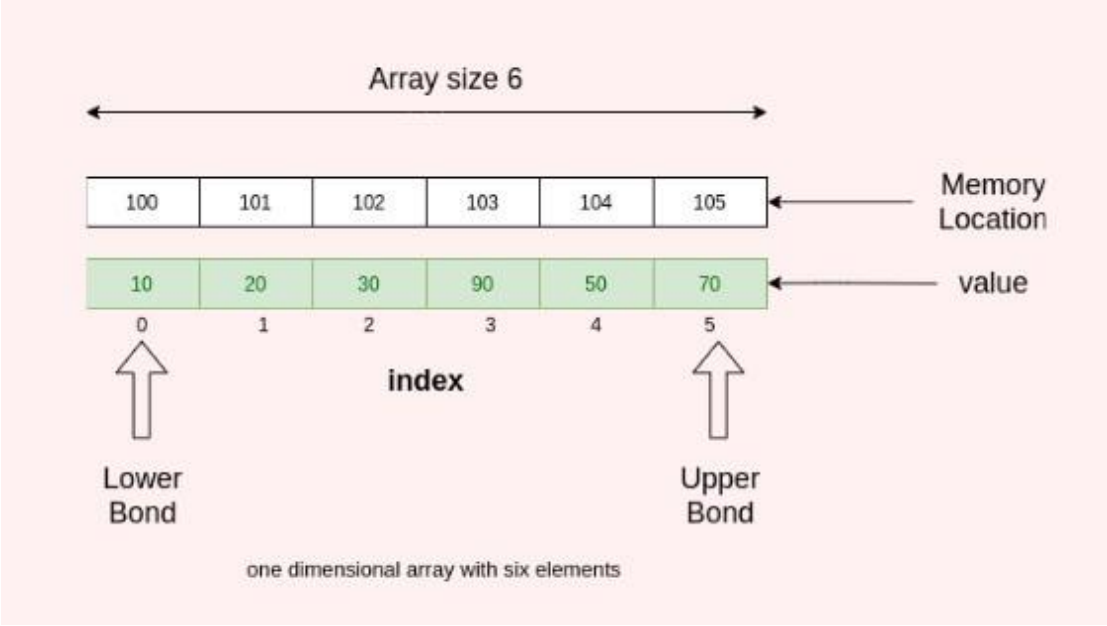
```
datatype   arrayName [ size ] ;
```

Array Indexing:

Arrays in C are indexed starting at 0 to size-1.The index of the first element of the array is starts with 0 , the second element is 1, the third element is 2 , and The index to the last value in the array is the array size minus one.

In the example below the subscripts run from 0 through 5,

the size of the array=upperbound - lower bound+1



Memory Allocation:

- With the contiguous memory allocation, it becomes easier to find out the memory location of any element in the array by just knowing the first memory location and adding the offset.
- For example:

  int a[8];

Memory Location	4000	4004	4008	4012	4016	4020	4024	4028
	1	2	3	4	5	6	7	8

- In the above example, we have an integer array, and the memory locations are contiguous, where each integer takes 4 bytes, and the starting address is 4000.

To calculate the address of a particular element in an array having base address(starting address) the following formula is used.

Address of n element= base address+ index of n element* size of datatype(of array)

Eg: find the address of 5th element of an integer array int a[8]={1,2,3,4,5,6,7,8}, having base address 4000?

Answer: the address of the fifth element is

4000+ 4*4=4016. (here 5th element index is 4, size of integer is 4)

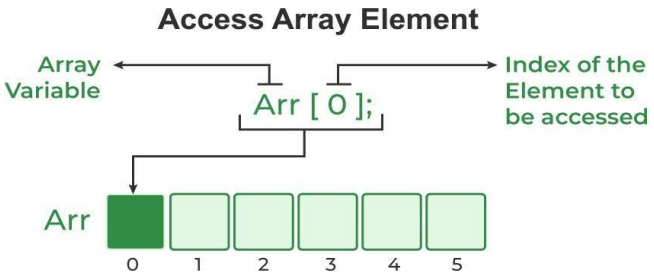
Accessing Array elements :

We can access any element of an array in C using the array subscript operator **[ ]** and the index value ***i*** of the element.

syntax to access individual elements of an array..

```
arrayName [ indexValue ] ;
```

One thing to note is that the indexing in the array always starts with 0, i.e., the **first element** is at index **0** and the **last element** is at **N – 1** where **N** is the number of elements in the array.



Example:

int arr[5] = { 15, 25, 35, 45, 55 };

accessing element at index 0 i.e first element: arr[0]
accessing element at index 1 i.e Second element: arr[1]
accessing element at index 2 i.e 3rd element: arr[2]
accessing element at index 4 i.e last element: arr[4]

Types of Arrays:

One-dimensional array – array having only one subscript variable is called one-dimensional array.

Multi-dimensional array (2D,3D..etc) - array having more than one subscript variable is called multi- dimensional arrays.

1. One Dimensional Array:

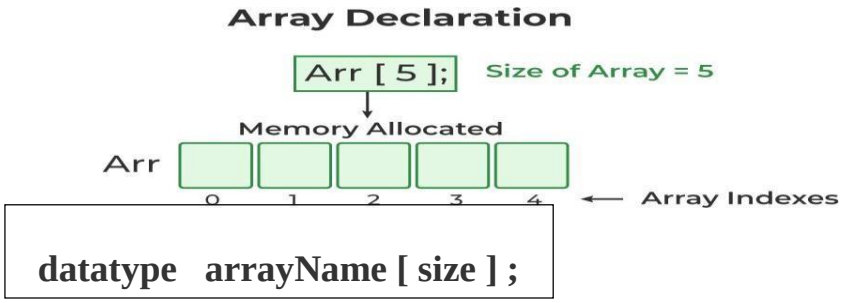
A list of items can be given one variable name using only one subscript and such a variable is called a single subscripted variable of a One Dimensional Array.

a) Declaration of One Dimensional Array:

Like any other variable, arrays must be declared before they are used so that the compiler can allocate space for them in memory.

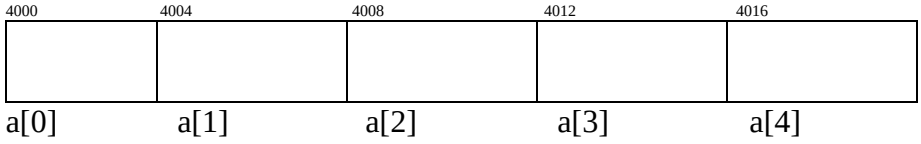
when we want to create an array we must know the data type of values to be stored in that array and also the number of values to be stored in that array.

syntax to create an array...



Ex: int a[5];

It represented in memory location as:



b) Initialization of One Dimensional Array:

Initialization in C is the process to assign some initial value to the variable. When the array is declared or allocated memory, the elements of the array contain some garbage value. So, we need to initialize the array to some meaningful value.

There are several ways in which we can initialize an array in C.

1. Array Initialization with Declaration(compile time):

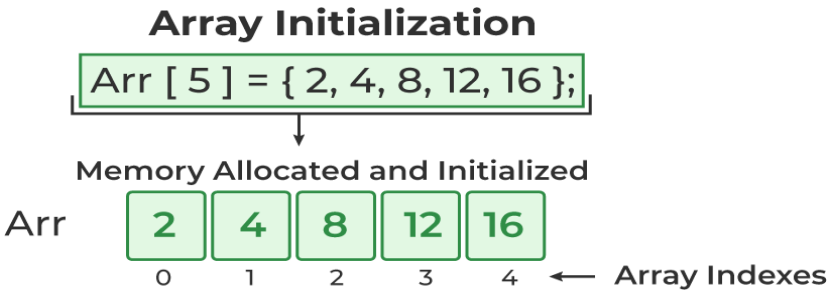
In compile time initialisation, We can initialize the elements of arrays in the same way as the ordinary variables when they are declared.

In this method, **we initialize the array along with its declaration**.

We use an initializer list to initialize multiple elements of the array. An initializer list is the list of values enclosed within braces **{ }** separated by a comma.

Syntax for creating an array with size and initial values

datatype arrayName [size] = {value1,value2,... valuen} ;



- Example:

  int a[5] = { 20, 40, 30, 60, 70 }

  It represented in memory location as:

v

20	10	30	50	40
a[0]	a[1]	a[2]	a[3]	a[4]

Array Initialization with Declaration without Size:

If we initialize an array using an initializer list, we can skip declaring the size of the array as the compiler can automatically deduce the size of the array in these cases. The size of the array in these cases is equal to the number of elements present in the initializer list as the compiler can automatically deduce the size of the array.

datatype arrayname[ ] = {1,2,3,4,5};

The size of the above arrays is 5 which is automatically deduced by the compiler.

2.Array Initialization (at Runtime) after Declaration (Using Loops):

We initialize the array after the declaration by assigning the initial value to each element individually. We can use for loop, while loop, or do-while loop to assign the value to each element of the array.

Ex: int a[100]; n-no. of elements in an array

```
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
```

//Example program1 : Array Input/Output

```
#include<stdio.h>
int main( )
{
   int i =0, n , arr[20];
   printf("enter the numberof elements:");
     scanf("%d", &n);
   printf("enter the %d elements of array\n",n);
   for(i=0;i<n;i++)
   {
    scanf("%d",&arr[i]);
   }
    printf("the array elements are:\n");
   for(i=0;i<n;i++)
   printf("arr[%d]=%d\t" ,i,arr[i]);
   return 0;
}
```

Output:

```
enter the numberof elements:5
enter the 5 elements of array
10
20
30
40
50
the array elements are:
arr[0]=10      arr[1]=20      arr[2]=30      arr[3]=40      arr[4]=50
```

Ex2:

// Program to find the average of n numbers using arrays

```
#include <stdio.h>
int main()
{
  int marks[10], i, n, sum = 0;
  double average;
  printf("Enter number of elements: ");
  scanf("%d", &n);
  for(i=0; i < n; ++i)
{
   printf("Enter number%d: ",i+1);
    scanf("%d", &marks[i]);
    sum += marks[i];
 }
  average = (double) sum / n;
  printf("Average = %.2lf", average);
  return 0;
}
```

Output:

```
Enter number of elements: 5
Enter number1: 45
Enter number2: 35
Enter number3: 38
Enter number4: 31
Enter number5: 49
Average = 39.60
```

Ex:3//perform linear search in an array

```
#include <stdio.h>
int main()
{
int array[100], key, i, n;
printf("Enter number of elements in array\n");
scanf("%d", &n);
printf("Enter %d integer(s)\n", n);
for (i = 0; i < n; i++)
scanf("%d",  &array[i]);
printf("Enter a number to search\n");
scanf("%d", &key);
for (i= 0; i < n; i++)
{
   if (array[i] == key)
   {
   printf("%d is present at location %d.\n", key, i+1);
   break;
   }
}
if (i == n)
printf("%d isn't present in the array.\n", key);
return 0;
}
```

Output:

```
Enter number of elements in array
5
Enter 5 integer(s)
10 20 30 40 50
Enter a number to search
30
30 is present at location 3.
```

Output2:

```
Enter number of elements in array
5
Enter 5 integer(s)
1 2 8 9 7
Enter a number to search
10
10 isn't present in the array.
```

//Ex4: Sort the elements of an array in ascending order

```
#include <stdio.h>
   int main()
   {
       int i, j, n, a[30],temp;
      printf("Enter the number of elements in an array \n");
      scanf("%d", &n);
      printf("Enter the %d elements \n",n);
      for (i = 0; i < n; ++i)
         scanf("%d", &a[i]);
      for (i = 0; i < n; ++i)
      {
         for (j = i + 1; j < n; ++j)
         {
            if (a[i] > a[j])
            {
               temp = a[i];
               a[i] = a[j];
               a[j] =temp;
            }

         }

      }
      printf("The numbers arranged in ascending order are given below \n");
      for (i = 0; i < n; ++i)
         printf("%d\t", a[i]);
      return 0;
   }
```

Output:

```
Enter the number of elements in an array
5
Enter the 5 elements
80 30 20 10 90
The numbers arranged in ascending order are given below
10      20      30      80      90
```

2. Two Dimensional Array in C:

The two-dimensional array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns. However, 2D arrays are created to implement a relational database lookalike data structure. It provides ease of holding the bulk of data at once which can be passed to any number of functions wherever required.

Declaration of two dimensional Array in C:

The syntax to declare the 2D array is given below.

datatype arrayname[rows][columns];

Consider the following example.

int A[3][4];

Here, 3 is the number of rows, and 4 is the number of columns

Example: int X[3][3];

- Here, 3 is the number of rows, and 3 is the number of columns.
- Elements in two-dimensional arrays are commonly referred by x[i][j] where i is the row number and 'j' is the column number.
- A two – dimensional array can be seen as a table with 'x' rows and 'y' columns where the row number ranges from 0 to (i-1) and column number ranges from 0 to (j-1). A two – dimensional array 'x' with 3 rows and 3 columns is shown below:

	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

Initialization of 2D Array in C

- There are two ways in which a Two-Dimensional array can be initialized.

First Method:

int x[3][4] = {0, 1 ,2 ,3 ,4 , 5 , 6 , 7 , 8 , 9 , 10 , 11};

- The above array has 3 rows and 4 columns. The elements in the braces from left to right are stored in the table also from left to right.
- The elements will be filled in the array in the order, first 4 elements from the left in first row, next 4 elements in second row and so on.

Better Method:

int x[3][4] = {{0,1,2,3}, {4,5,6,7}, {8,9,10,11}};

- This type of initialization makes use of nested braces. Each set of inner braces represents one row.
- In the above example there are total three rows so there are three sets of inner braces.

Accessing Elements of Two-Dimensional Arrays

- Elements in Two-Dimensional arrays are accessed using the row indexes and column indexes.

Example: int value=x[2][1];

- The above example represents the element present in third row and second column.

Program using nested loop to handle 2-D Array

```
#include <stdio.h>
int main ()
{
        int a[3][3] = { 1,2,3,4,5,6,7,8,9};
        int i, j;
        for ( i = 0; i< 3; i++ )
        {
                for ( j = 0; j < 3; j++ )
                {
                        printf(" %d\t", a[i][j] );
                }
                printf("\n");
        }
        return 0;
}
```

Output:

```
 1      2      3
4       5      6
7       8      9
```

Ex2:

/*Program to read the elements of matrix from the user*/

```
#include <stdio.h>
int main ()
{

int arr[3][3],i,j;
for (i=0;i<3;i++)
        {
           for (j=0;j<3;j++)
         {
                printf("Enter a[%d][%d]: ",i,j);
                scanf("%d",&arr[i][j]);
         }
        }

printf("\n printing the elements\n");
```

```
for(i=0;i<3;i++)
{
        for (j=0;j<3;j++)
        {
                printf("%d\t",arr[i][j]);
        }
        printf("\n");
}

return 0;
}
```

Output:

Enter a[0][0]: 1

Enter a[0][1]: 2

Enter a[0][2]: 3

Enter a[1][0]: 4

Enter a[1][1]: 5

Enter a[1][2]: 6

Enter a[2][0]: 7

Enter a[2][1]: 8

Enter a[2][2]: 9

printing the elements

1 2 3

4 5 6

7 8 9

Ex3:

// C program to find the sum of two matrices of order 2*2

```
#include <stdio.h>
int main()
{
  float a[2][2], b[2][2], result[2][2];
  int i,j;
  printf("Enter elements of 1st matrix\n");
  for ( i = 0; i < 2; ++i)
    for ( j = 0; j < 2; ++j)
    {
      printf("Enter a[%d][%d]: ", i, j);
      scanf("%f", &a[i][j]);
    }
```

```
printf("Enter elements of 2nd matrix\n");
 for (i = 0; i < 2; ++i)
    for ( j = 0; j < 2; ++j)
    {
      printf("Enter b[%d][%d]: ", i, j);
      scanf("%f", &b[i][j]);
    }
  for ( i = 0; i < 2; ++i)
    for ( j = 0; j < 2; ++j)
    {
      result[i][j] = a[i][j] + b[i][j];
    }

  printf("\nSum Of Matrix:\n");

  for ( i = 0; i < 2; ++i)
  {
    for ( j = 0; j < 2; ++j)
    {
      printf("%.1f\t", result[i][j]);
    }
        printf("\n");
    }
  return 0;
}
```

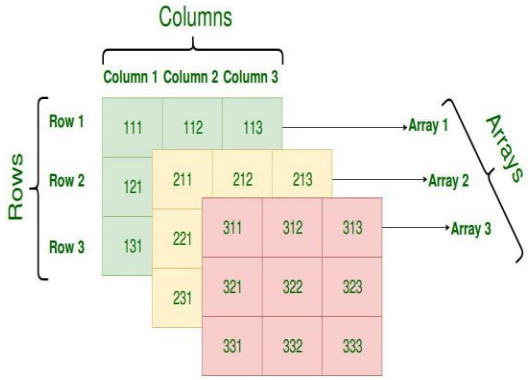
Output:

```
Enter elements of 1st matrix
Enter a[0][0]: 1
Enter a[0][1]: 2
Enter a[1][0]: 3
Enter a[1][1]: 4
Enter elements of 2nd matrix
Enter b[0][0]: 1
Enter b[0][1]: 5
Enter b[1][0]: 6
Enter b[1][1]: 8

Sum Of Matrix:
2.0     7.0
9.0     12.0
```

3. Multidimensional Arrays:

- Multidimensional Arrays can have three, four or more dimensions.
- The first dimension is called a plane, which consists of rows and columns. In C, the three dimensional array is taken as an array of two dimensional arrays. It is an array of arrays of arrays.
- The following figure shows an array of three dimensions.



Declaring Multidimensional Arrays

- These arrays must be declared before being used. It can be declared as shown below.

For example: int A[3][4][5];

- One dimension for rows, One for columns and One for Planes.

int A[planes][rows][columns];

Initialization

- Declaration only reserves space for the elements in the array. No values will be stored in the array.
- To store the values, we must either initialize the elements or read from the keyboard or assign the values to each element.

```
intA[2][3][4] = {
                    // Plane 0
{ {1,2,3,4 }, {5,6,7,8}, {9,0,1,2} } ,
                    // Plane 1
{ {8,9,7,6}, {5,4,3,2}, {1,0,2,3} }
};
```

- All elements can be initialized to zero as intA[2][3][4] = {0};

Applications of Arrays in C

In c programming language, arrays are used in wide range of applications. Few of them are as follows...

● Arrays are used to Store List of values

In c programming language, single dimensional arrays are used to store list of values of same datatype. In other words, single dimensional arrays are used to store a row of values. In single dimensional array data is stored in linear form.

● Arrays are used to Perform Matrix Operations

We use two dimensional arrays to create matrix. We can perform various operations on matrices using two dimensional arrays.

● Arrays are used to implement Search Algorithms

We use single dimensional arrays to implement search algorihtms like ...

1. **Linear Search**
2. **Binary Search**

- **Arrays are used to implement Sorting Algorithms**

We use single dimensional arrays to implement sorting algorihtms like ...

1. **Insertion Sort**
2. Bubble Sort
3. **Selection Sort**
4. **Quick Sort**
5. Merge Sort, etc.,

- **Arrays are used to implement Data structures**

We use single dimensional arrays to implement datastructures like...

1. **Stack Using Arrays**
2. **Queue Using Arrays**

- **Arrays are also used to implement CPU Scheduling Algorithms**

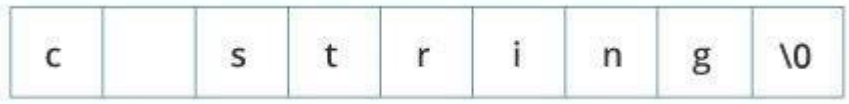
Introduction to Strings:

String: A String is a sequence of characters terminated with a null character '\0'.
String is enclosed in double quotation marks(" ").
The C String is stored as an array of characters. The difference between a character array and a C string is that the string in C is terminated with a unique character '\0.

For example,

```
char c[ ] = "c string";
char c[5]= "123";
```

When the compiler encounters a sequence of characters enclosed in the double quotation marks, it appends a null character \0 at the end by default.



Memory Diagram

Declaration of a string:

Declaring a string in C is as simple as declaring a one-dimensional array.
Basic syntax for declaring a string:

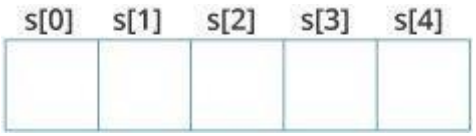
char stringName[size];

here, stringName is any name given to the string variable and size is used to define the length of the string, i.e the number of characters strings will store.

There is an extra terminating character which is the Null character ('\0') used to indicate the termination of a string that differs strings from normal character arrays.

Example: char s[5];

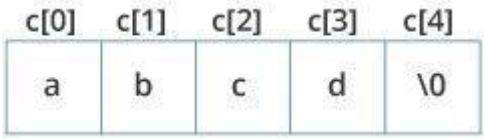
Here, we have declared a string of 5 characters.



String Initialization:

You can initialize strings in a number of ways.

1.char c[ ] = "abcd";
String literals can be assigned without size. Here, the name of the string c acts as a pointer because it is an array.
2.char c[50] = "abcd";
String literals can be assigned with a predefined size. But we should always account for one extra space which will be assigned to the null character. If we want to store a string of size n then we should always declare a string with a size equal to or greater than n+1.
3.char c[ ] = {'a', 'b', 'c', 'd', '\0'};
We can assign character by character without size with the NULL character at the end. The size of the string is determined by the compiler automatically.
4.char c[5] = {'a', 'b', 'c', 'd', '\0'};
We can also assign a string character by character. But we should remember to set the end character as '\0' which is a null character.



Let's take another example:

```
char c[5] = "abcde";
```

Here, we are trying to assign 6 characters (the last character is '\0') to a char array having 5 characters. This is bad and you should never do this.

Assigning Values to Strings:

Arrays and strings are second-class citizens in C; they do not support the assignment operator once it is declared.
For example,
char c[100];
c = "C programming"; // Error! array type is not assignable.
Note: Use the strcpy() function to copy the string instead

Read String from the user:

We can read a string value from the user during the program execution. We use the following two methods...

1. Using **scanf() method - reads single word**
2. Using **gets() method - reads a line of text**

Using scanf() function to read a string:
The scanf() function reads the sequence of characters until it encounters whitespace (space, newline, tab, etc.).

Example 1: scanf() to read a string:

```
#include <stdio.h>
int main()
{
   char name[20];
   printf("Enter name: ");
   scanf("%s", name);
   printf("Your name is %s.", name);
   return 0;
}
```

Output:
Enter name: Dennis Ritchie
Your name is Dennis.

Even though Dennis Ritchie was entered in the above program, only "Dennis" was stored in the name string. It's because there was a space after Dennis.
Also notice that we have used the code name instead of &name with scanf().
scanf("%s", name);
This is because name is a char array, and we know that array names declare to pointers in C. Thus, the name in scanf() already points to the address of the first element in the string, which is why we don't need to use &.

Using gets() method to read a string:
When we want to read multiple words or a line of text, we use a pre-defined method **gets()**.
The **gets()** method terminates the reading of text with **Enter** character.
Example:

```
#include<stdio.h>
int main()
{
  char name[50];
  printf("Please enter your name : ");
```

```
  gets(name);
  printf("Your name is %s ", name);
  return 0;
}
```

Output:
Please enter your name : Durga prasanna
Your name is Durga prasanna

String Handling Functions in C/Library functions:

C Programming language provides a set of pre-definied functions called **String Handling Functions** to work with string values. All the string handling functions are defined in a header file called **string.h**.
There are many important string functions defined in "string.h" library.

No.	Function	Description
1)	strlen(string_name)	returns the length of string name.
2)	strcpy(destination, source)	copies the contents of source string to destination string.
3)	strcat(first_string, second_string)	concats or joins first string with second string. The result of the string is stored in first string.
4)	strcmp(first_string, second_string)	compares the first string with second string. If both strings are same, it returns 0.
5)	strrev(string)	returns reverse string.
6)	strlwr(string)	returns string characters in lowercase.
7)	strupr(string)	returns string characters in uppercase.

1. String Length: strlen( ) function:

The strlen( ) function returns the length of the given string. It doesn't count null character '\0'.
Syntax: strlen(StringName);

Example:

```
#include<stdio.h>
#include <string.h>
int main()
{
  char ch[20]={'p', 'r', 'o', 'g', 'r', 'a', 'm', 'm', 'i', 'n', 'g', '\0'};
  printf("Length of string is: %d", strlen(ch));
  return 0;
}
```

Output:

Length of string is 11

2. Copy String: strcpy( ):

It is for copying source string into destination string.
The length of the destination string >= source string.
Syntax :

strcpy (Destination string, Source String);

The strcpy(destination, source) function copies the source string in destination string.

strncpy (Destination string, Source String, n);

It copy's 'n' characters of source string into destination string.

1) char a[50];
strcpy ("Hello",a);
o/p: error
2) char a[50];
strcpy (a,"hello");
o/p: a= "Hello"

Example:

```
#include<stdio.h>
#include <string.h>
int main()
{
  char s1[20]={'h', 'e', 'l', 'l', 'o', 'w', 'o', 'r', 'l', 'd', '\0'};
  char s2[20];
  strcpy(s2,s1);
  printf("Value of second string is: %s",s2);
 return 0;
}
```

Output:

Value of second string is: helloworld

3. String Concatenation: strcat()

This is used for combining or concatenating n characters of one string into another.
The length of the destination string must be greater than the source string.
The resultant concatenated string will be in the destination string
Syntax:. strcat (Destination String, Source string, n);
The strcat(first_string, second_string) function concatenates two strings and result is returned to first_string.

Example1:

```
#include<stdio.h>
#include <string.h>
int main()
{
  char ch[10]={'h', 'e', 'l', 'l', 'o', '\0'};
   char ch2[10]={'c', '\0'};
   strcat(ch,ch2);
   printf("Concatenated  string is: %s",ch);
 return 0;
}
```

Output:

Concatenated string is: helloc

Example2:

```
#include<stdio.h>
#include <string.h>
int main()
{
 char a [30] = "Hello";
 char b [20] = "Good Morning";
 strcat (a,b,4);
 a [9] = '\0';
printf("concatenated string = %s", a);
}
```

Output :

Concatenated string = Hello Good.

4. Compare String: strcmp():

This function compares 2 strings.

It returns the ASCII difference of the first two non – matching characters in both thestrings

The strcmp(first_string, second_string) function compares two string and returns 0 if both strings are equal.

//If the difference is equal to zero, then string1 = string2

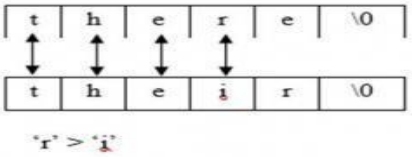
//If the difference is positive, then string1 > string2

//If the difference is negative, then string1 < string2

eg:

1) char a[10]= "there"
   char b[10] = "their"
   strcmp (a,b);

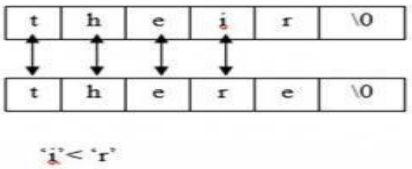
   Output: string1 >string2



'r' > 'i'

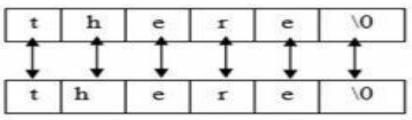
2) char a[10]= "their"
   char b[10] = "there"
   strcmp (a,b);

   Output: string1 <string2



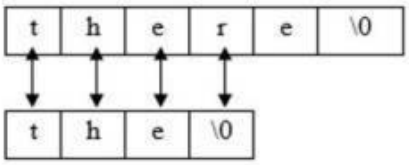
'i' < 'r'

3) char a[10]= "there"
   char b[10] = "there"
   strcmp (a,b);
   Output: string1 =string2



4) char a[10]= "there"
char b[10] = "the"
strcmp (a,b)

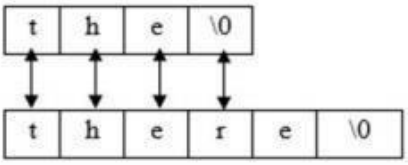
Output: string1 >string2



'r' > '\0'

5) char a[10]= "the"
char b[10] = "there"
strcmp (a,b);

Output: string1 <string2



'\0' < 'r'

Example1:

```
#include<stdio.h>
#include <string.h>
int main()
{
  char str1[20],str2[20];
  printf("Enter 1st string: ");
  gets(str1);  //reads string from console
  printf("Enter 2nd string:  ");
  gets(str2);
  if(strcmp(str1,str2)==0)
     printf("Strings are equal");
  else
     printf("Strings are not equal");
 return 0;
}
```

Output:

```
Enter 1st string: hello
Enter 2nd string: hello
Strings are equal
```

Example2:

```
#include<stdio.h>
#include<string.h>
int main ()
{
char a[50], b [50];
int d;
```

```
    printf ("Enter 2 strings:");
    scanf ("%s %s", a,b);
    d = strcmp(a,b);
    if (d==0)
    {
printf("%s is (alphabetically) equal to %s", a,b);
    }
else if (d>0)
    {
printf("%s is (alphabetically) greater than %s",a,b);
    }
else if (d<0)
    {
printf("%s is (alphabetically) less than %s", a,b);
    }
return 0;
    }
```

Output:

```
Enter 2 strings:hello
world
hello is (alphabetically) less than world
```

5. Reverse String: strrev():

The strrev(string) function returns reverse of the given string.

The reversed string will be stored in the same string.

Let's see a simple example of strrev() function.

Example:

```
#include<stdio.h>
#include <string.h>
int main()
{
  char str[20];
  printf("Enter string: ");
  gets(str);  //reads string from console
  printf("String is: %s",str);
  printf("\nReverse String is: %s",strrev(str));
 return 0;
}
```

Output:

```
Enter string: helloworld
String is: helloworld
Reverse String is: dlrowolleh
```

****String Input and Output Functions:** gets( ),puts( ),getchar( ),putchar( )--→Refer Unit-I material.

6. Lowercase String: strlwr()

This function converts a string to lowercase. Note, this function is not included in the Standard Library.

Syntax:

```
char *strlwr(char *str);
```

Example:

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

int main()
{
   char str[ ] = "GEEKSFORGEEKS IS THE BEST";

   // converting the given string into lowercase.
   printf("%s\n",strlwr (str));

   return  0;
}
```

Output:

```
geeksforgeeks is the best
```

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

int main()
{
  char str[] = "CompuTer ScienCe PoRTAl fOr geeKS";

  printf("Given string is: %s\n",str);

  printf("\nString after converting to the "
          "lowercase is: %s",strlwr(str));

  return 0;
}
```

Output:

```
Given string is: CompuTer ScienCe PoRTAl fOr geeKS


String after converting to the lowercase is: computer science portal for geeks
```

7. Uppercase String: strupr()

This function converts a string to uppercase. Note, this function is not included in the Standard Library.
The **strupr( )** function is used to converts a given string to uppercase.

Syntax:

```
char *strupr(char *str);
```

Example:

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

```
int main()
{
   char str[ ] = "geeksforgeeks is the best";
   //converting the given string into uppercase.
   printf("%s\n", strupr (str));
   return 0;
}
```

Output:

```
GEEKSFORGEEKS IS THE BEST
```

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

int main()
{
char str[] = "CompuTer ScienCe PoRTAl fOr geeKS";

printf("Given string is: %s\n", str);

printf("\nstring after converting to the uppercase is: %s", strupr(str));
return 0;
}
```

```
Given string is: CompuTer ScienCe PoRTAl fOr geeKS


string after converting to the uppercase is: COMPUTER SCIENCE PORTAL FOR GEEKS
```