Rajmani Kumar,
Lecturer, Dept. of BCA
S.U.College, Hilsa (Nalanda)
Patliputra University, Patna

BCA-2nd Year **Paper-III**

Arrays

Array is the collection of similar data types or collection of similar entity stored in contiguous memory location. Array of character is a string. Each data item of an array is called an element. And each element is unique and located in separated memory location. Each of elements of an array share a variable but each element having different index no. known as subscript.

An array can be a single dimensional or multi-dimensional and number of subscripts determines its dimension. And number of subscript is always starts with zero. One dimensional array is known as vector and two dimensional arrays are known as matrix.

ADVANTAGES: array variable can store more than one value at a time where other variable can store one value at a time.

Example:

```
int arr[100];

int mark[100];
```

DECLARATION OF AN ARRAY :

Its syntax is :

```
Data type array name [size];
  int arr[100];
    int mark[100];
      int a[5]={10,20,30,100,5}
```

The declaration of an array tells the compiler that, the data type, name of the array, size of the array and for each element it occupies memory space. Like for int data type, it occupies 2 bytes for each element and for float it occupies 4 byte for each element etc. The size of the array operates the number of elements that can be stored in an array and it may be a int constant or constant int expression.

We can represent individual array as :

```
int ar[5];
ar[0], ar[1], ar[2], ar[3], ar[4];
```

Symbolic constant can also be used to specify the size of the array as:

#define SIZE 10;

INITIALIZATION OF AN ARRAY:

After declaration element of local array has garbage value. If it is global or static array then it will be automatically initialize with zero. An explicitly it can be initialize that

Data type array name [size] = {value1, value2, value3…}

Example:

in ar[5]={20,60,90, 100,120}

Array subscript always start from zero which is known as lower bound and upper value is known as upper bound and the last subscript value is one less than the size of array. Subscript can be an expression i.e. integer value. It can be any integer, integer constant, integer variable, integer expression or return value from functional call that yield integer value.

So if i & j are not variable then the valid subscript are

ar [i*7],ar[i*i],ar[i++],ar[3];

The array elements are standing in continuous memory locations and the amount of storage required for hold the element depend in its size & type.

Total size in byte for 1D array is:

Total bytes=size of (data type) * size of array.

Example : if an array declared is:

int [20];

Total byte= 2 * 20 =40 byte.

ACCESSING OF ARRAY ELEMENT:

/*Write a program to input values into an array and display them*/

```
#include<stdio.h>
int main()
{
int arr[5],i;
for(i=0;i<5;i++)
{
printf("enter a value for arr[%d] \n",i);
scanf("%d",&arr[i]);
}
printf("the array elements are: \n");
```

```
for (i=0;i<5;i++)
{
printf("%d\t",arr[i]);
}
return 0;
}
```

OUTPUT:

```
Enter a value for arr[0] = 12
Enter a value for arr[1] =45
Enter a value for arr[2] =59
Enter a value for arr[3] =98
Enter a value for arr[4] =21
The array elements are 12 45 59 98 21
```

/* Write a program to add 10 array elements */

```
#include<stdio.h>
void main()
{
int i ;
int arr [10];
int sum=o;
for (i=0; i<=9; i++)
{
printf ("enter the %d element \n", i+1);
scanf ("%d", &arr[i]);
}
for (i=0; i<=9; i++)
{
sum = sum + a[i];
}
printf ("the sum of 10 array elements is %d", sum);
}
```

OUTPUT:

```
Enter a value for arr[0] =5
Enter a value for arr[1] =10
Enter a value for arr[2] =15
Enter a value for arr[3] =20
Enter a value for arr[4] =25
```

Enter a value for arr[5] =30
Enter a value for arr[6] =35
Enter a value for arr[7] =40
Enter a value for arr[8] =45
Enter a value for arr[9] =50
Sum = 275

Single Dimensional Array

while initializing a single dimensional array, it is optional to specify the size of array. If the size is omitted during initialization then the compiler assumes the size of array equal to the number of initializers.

For example:-

```
int marks[]={99,78,50,45,67,89};
```

If during the initialization of the number the initializers is less then size of array, then all the remaining elements of array are assigned value zero .

For example:-

```
int marks[5]={99,78};
```

Here the size of the array is 5 while there are only two initializers so After this initialization, the value of the rest elements are automatically occupied by zeros such as

Marks[0]=99 , Marks[1]=78 , Marks[2]=0, Marks[3]=0, Marks[4]=0

Again if we initialize an array like

```
int array[100]={0};
```

Then the all the element of the array will be initialized to zero. If the number of initializers is more than the size given in brackets then the compiler will show an error.

For example:-

```
int arr[5]={1,2,3,4,5,6,7,8};//error
```

we cannot copy all the elements of an array to another array by simply assigning it to the other array like, by initializing or declaring as

```
int a[5] ={1,2,3,4,5};
int b[5];
b=a;//not valid
```

(**note**:-here we will have to copy all the elements of array one by one, using for loop.)

Processing:

For processing arrays we mostly use for loop. The total no. of passes is equal to the no. of elements present in the array and in each pass one element is processed.

Example:

```
#include<stdio.h>
main()
{
int a[3],i;
for(i=0;i<=2;i++)          //Reading the array values
{
printf("enter the elements"); scanf("%d",&a[i]);
}
for(i=0;i<=2;i++)          //display the array values
{
printf("%d",a[i]); printf("\n");
}
}
```

This program reads and displays 3 elements of integer type.

Example:1

C Program to Increment every Element of the Array by one & Print Incremented Array.

```
#include <stdio.h>
void main()
{
int i;
int array[4] = {10, 20, 30, 40};
for (i = 0; i < 4; i++)
arr[i]++;
for (i = 0; i < 4; i++)
printf("%d\t", array[i]);
}
```

Example: 2

C Program to Print the Alternate Elements in an Array

```
#include <stdio.h>
void main()
{
int array[10];
int i, j, temp;
printf("enter the element of an array \n");
for (i = 0; i < 10; i++)
scanf("%d", &array[i]);
```

```
printf("Alternate elements of a given array \n");
 for (i = 0; i < 10; i += 2)
printf( "%d\n", array[i]) ;
}
```

Example-3

C program to accept N numbers and arrange them in an ascending order

```
#include <stdio.h>
void main()
{
int i, j, a, n, number[30];
printf("Enter the value of N \n");
scanf("%d", &n);
printf("Enter the numbers \n");
for (i = 0; i < n; ++i)
scanf("%d", &number[i]);
for (i = 0; i < n; ++i)
{
for (j = i + 1; j < n; ++j)
{
if (number[i] > number[j])
{
a =number[i];
number[i] = number[j];
number[j] = a;
}
}
}
printf("The numbers arranged in ascending order are given below \n");
for (i = 0; i < n; ++i)
printf("%d\n", number[i]);
}
```

TWO DIMENSIONAL ARRAYS

Arrays that we have considered up to now are one dimensional array, a single line of elements. Often data come naturally in the form of a table, e.g. spreadsheet, which need a two-dimensional array.

Declaration:

The syntax is same as for 1-D array but here 2 subscripts are used.

Syntax:

data_type array_name[rowsize][columnsize];

Rowsize specifies the no.of rows Columnsize specifies the no.of columns.

Example:

int a[4][5];

This is a 2-D array of 4 rows and 5 columns. Here the first element of the array is a[0][0] and last element of the array is a[3][4] and total no. of elements is 4*5=20.

	col 0	col 1	col 2	col 3	col 4
row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]
row 3	a[3][0]	a[3][1]	a[3][2]	a[3][3]	a[3][4]

Initialization:

2-D arrays can be initialized in a way similar to 1-D arrays.

Example:

int m[4][3]={1,2,3,4,5,6,7,8,9,10,11,12};

The values are assigned as follows:

m[0][0]:1	m[0][1]:2	m[0][2]:3
m[1][0]:4	m[1][1]:5	m[3][2]:6
m[2][0]:7	m[2][1]:8	m[3][2]:9
m[3][0]:10	m[3][1]:11	m[3][2]:12

The initialization of group of elements as follows:

int m[4][3]={{11},{12,13},{14,15,16},{17}};

The values are assigned as:

m[0][0]:1 1	m[0][1]:0	m[0][2]:0
m[1][0]:12	m[1][1]:13	m[3][2]:0
m[2][0]:14	m[2][1]:15	m[3][2]:16
m[3][0]:17	m[3][1]:0	m[3][2]:0

Note:

In 2-D arrays it is optional to specify the first dimension but the second dimension should always be present.

Example: *int*

```
m[][3]={
{1,10},
{2,20,200},
{3},
{4,40,400} };
```

Here the first dimension is taken 4 since there are 4 rows in the initialization list. A 2-D array is known as matrix.

Processing:

For processing of 2-D arrays we need two nested for loops. The outer loop indicates the rows and the inner loop indicates the columns.

Example:

```
int a[4][5];
```

a) Reading values in a

```
for(i=0;i<4;i++)
    for(j=0;j<5;j++)
        scanf("%d",&a[i][j]);
```

b) Displaying values of a

```
for(i=0;i<4;i++)
    for(j=0;j<5;j++)
        printf("%d",a[i][j]);
```

Example 1:

Write a C program to find sum of two matrices

```
#include <stdio.h> #include<conio.h>
void main()
{
float a[2][2], b[2][2], c[2][2];
int i,j;
clrscr();
printf("Enter the elements of 1st matrix\n");
/* Reading two dimensional Array with the help of two for loop. If there is an array of 'n' dimension, 'n' numbers of loops are needed for inserting data to array.*/
for(i=0;i<2;I++)
for(j=0;j<2;j++)
{
scanf("%f",&a[i][j]);
```

```
}
printf("Enter the elements of 2nd matrix\n");
for(i=0;i<2;i++)
for(j=0;j<2;j++)
{
scanf("%f",&b[i][j]);
}
/* accessing corresponding elements of two arrays. */
for(i=0;i<2;i++)
for(j=0;j<2;j++)
{
c[i][j]=a[i][j]+b[i][j]; /* Sum of corresponding elements of two arrays. */
}
/* To display matrix sum in order. */
printf("\nSum Of Matrix:"); for(i=0;i<2;++i)
{
for(j=0;j<2;++j)
printf("%f", c[i][j]);
printf("\n");
}
getch();
}
```

Example 2:

Program for multiplication of two matrices

```
#include<stdio.h>
#include<conio.h>
int main()
{
int i,j,k;
int row1,col1,row2,col2,row3,col3;
int mat1[5][5], mat2[5][5], mat3[5][5];
clrscr();
printf("\n enter the number of rows in the first matrix:");
scanf("%d", &row1);
printf("\n enter the number of columns in the first matrix:");
scanf("%d", &col1);
printf("\n enter the number of rows in the second matrix:");
scanf("%d", &row2);
```

```
printf("\n enter the number of columns in the second matrix:");
scanf("%d", &col2);
if(col1 != row2)
{
printf("\n The number of columns in the first matrix must be equal to the number of rows in the second matrix ");
getch();
 exit();
}
row3= row1; col3= col3;
printf("\n Enter the elements of the first matrix");
for(i=0;i<row1;i++)
{
for(j=0;j<col1;j++)
scanf("%d",&mat1[i][j]);
}
printf("\n Enter the elements of the second matrix");
for(i=0;i<row2;i++)
{
for(j=0;j<col2;j++)
scanf("%d",&mat2[i][j]);
}
for(i=0;i<row3;i++)
{
for(j=0;j<col3;j++)
{
mat3[i][j]=0;
for(k=0;k<col3;k++)
mat3[i][j] +=mat1[i][k]*mat2[k][j];
}
}
printf("\n The elements of the product matrix are"):
for(i=0;i<row3;i++)
{
printf("\n");
for(j=0;j<col3;j++)
printf("\t %d", mat3[i][j]);
}
return 0;
}
```

Output:

```
Enter the number of rows in the first matrix: 2
Enter the number of columns in the first matrix: 2
Enter the number of rows in the second matrix: 2
Enter the number of columns in the second matrix: 2
Enter the elements of the first matrix
1 2 3 4
Enter the elements of the second matrix
5 6 7 8
The elements of the product matrix are
19 22
43 50
```

Example 3:

Program to find transpose of a matrix.

```
#include <stdio.h>
int main()
{
int a[10][10], trans[10][10], r, c, i, j;
printf("Enter rows and column of matrix: ");
scanf("%d %d", &r, &c);
printf("\nEnter elements of matrix:\n");
for(i=0; i<r; i++)
for(j=0; j<c; j++)
{
printf("Enter elements a%d%d: ",i+1,j+1);
scanf("%d", &a[i][j]);
}
/* Displaying the matrix a[][] */
printf("\n Entered Matrix: \n"); for(i=0; i<r; i++)
for(j=0; j<c; j++)
{
printf("%d ",a[i][j]);
if(j==c-1)
printf("\n\n");
}
/* Finding transpose of matrix a[][] and storing it in array trans[][]. */
for(i=0; i<r;i++)
for(j=0; j<c; j++)
{
```

```
trans[j][i]=a[i][j];
}
/* Displaying the array trans[][]. */
printf("\nTranspose of Matrix:\n"); for(i=0; i<c;i++)
for(j=0; j<r;j++)
{
printf("%d ",trans[i][j]);
if(j==r-1)
printf("\n\n");
} return 0;
}
```

Output

```
Enter the rows and columns of matrix: 2 3
Enter the elements of matrix:
Enter elements a11: 1
Enter elements a12: 2
Enter elements a13: 9
Enter elements a21: 0
Enter elements a22: 4
Enter elements a23: 7
Entered matrix:
1 2 9
0 4 7

Transpose of matrix:
1 0
2 4
9 7
```

Multidimensional Array

More than 2-dimensional arrays are treated as multidimensional arrays.

Example:

```
int a[2][3][4];
```

Here a represents two 2-dimensional arrays and each of these 2-d arrays contains 3 rows and 4 columns.

The individual elements are:

a[0][0][0],a[0][0][1],a[0][0][2],a[0][1][0]……a[0][3][2]
a[1][0][0],a[1][0][1],a[1][0][2],a[1][1][0]……..a[1][3][2]

the total no. of elements in the above array is 2*3*4=24.

Initialization:

```
int a[2][4][3]={
{
{1,2,3},
{4,5},
{6,7,8},
{9}
},
{
{10,11},
{12,13,14},
{15,16},
{17,18,19}
}
}
```

The values of elements after this initialization are as:

a[0][0][0]:1	a[0][0][1]:2	a[0][0][2]:3
a[0][1][0]:4	a[0][1][1]:5	a[0][1][2]:0
a[0][2][0]:6	a[0][2][1]:7	a[0][2][2]:8
a[0][3][0]:9	a[0][3][1]:0	a[0][3][2]:0
a[1][0][0]:10	a[1][0][1]:11	a[1][0][2]:0
a[1][1][0]:12	a[1][1][1]:13	a[1][1][2]:14
a[1][2][0]:15	a[1][2][1]:16	a[1][2][2]:0
a[1][3][0]:17	a[1][3][1]:18	a[1][3][2]:19

Note:
The rule of initialization of multidimensional arrays is that last subscript varies most frequently and the first subscript varies least rapidly.

Example:

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

```
int d[5]; int i;
for(i=0;i<5;i++)
{
d[i]=i;
}
for(i=0;i<5;i++)
{
printf("value in array %d\n",a[i]);
}
}
```

ARRAYS USING FUNCTIONS

1-d arrays using functions

Passing individual array elements to a function

We can pass individual array elements as arguments to a function like other simple variables.

Example:

```
#include<stdio.h>
void check(int);
void main()
{
int a[10],i;
clrscr();
printf("\n enter the array elements:"); for(i=0;i<10;i++)
{
scanf("%d",&a[i]);
check(a[i]);
}
void check(int num)
{
if(num%2==0)
printf("%d is even\n",num); else
printf("%d is odd\n",num);
}
```

Output:

enter the array elements:
1 2 3 4 5 6 7 8 9 10

1 is odd

2 is even

3 is odd

4 is even

5 is odd

6 is even

7 is odd

8 is even

9 is odd

10 is even

Example:

C program to pass a single element of an array to function

```
#include <stdio.h>
void display(int a)
{
printf("%d",a);
}
int main()
{
int c[]={2,3,4};
display(c[2]); //Passing array element c[2] only.
return 0;
}
```

Output

2 3 4

Passing whole 1-D array to a function

We can pass whole array as an actual argument to a function the corresponding formal arguments should be declared as an array variable of the same type.

Example:

```
#include<stdio.h>
main()
{
int i, a[6]={1,2,3,4,5,6};
func(a);
printf("contents of array:");
for(i=0;i<6;i++)
printf("%d",a[i]);
printf("\n");
}
func(int val[])
{
int sum=0,i;
for(i=0;i<6;i++)
{
val[i]=val[i]*val[i];
sum+=val[i];
}
printf("the sum of squares:%d", sum);
}
```

Output

contents of array: 1 2 3 4 5 6 the sum of squares: 91

Example.2:

Write a C program to pass an array containing age of person to a function. This function should find average age and display the average age in main function.

```
#include <stdio.h>
float average(float a[]);
int main()
{
float avg, c[]={23.4, 55, 22.6, 3, 40.5, 18};
avg=average(c); /* Only name of array is passed as argument. */
printf("Average age=%.2f",avg);
return 0;
}
float average(float a[])
{
int i;
```

```
float avg, sum=0.0;
for(i=0;i<6;++i)
{
sum+=a[i];
}
avg =(sum/6);
return avg;
}
```

Output

Average age= 27.08

Solved Example:

1. Write a program to find the largest of n numbers and its location in an array.

```
#include <stdio.h>
#include<conio.h>
void main()
{
int array[100], maximum, size, c, location = 1;
clrscr();
printf("Enter the number of elements in array\n");
scanf("%d", &size);
printf("Enter %d integers\n", size);
for (c = 0; c < size; c++)
scanf("%d", &array[c]);
maximum = array[0];
for (c = 1; c < size; c++)
{
if (array[c] > maximum)
{
maximum = array[c];
location = c+1;
}
}
printf("Maximum element is present at location %d and it's value is %d.\n",
location, maximum);
getch();
}
```

Output:

Enter the number of elements in array
5
Enter 5 integers
2
4
7
9
1
Maximum element is present at location 4 and it's value is 9

2. Write a program to enter n number of digits. Form a number using these digits.

```
# include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int number=0,digit[10],
numofdigits,i; clrscr();
printf("\n Enter the number of digits:");
scanf("%d", &numofdigits);
 for(i=0;i<numofdigits;i++)
{
printf("\n Enter the %d th digit:", i);
scanf("%d",&digit[i]);
} i=0;
while(i<numofdigits)
{
number= number + digit[i]* pow(10,i)
i++;
}
printf("\n The number is : %d",number);
getch();
}
```

Output:

Enter the number of digits: 3
Enter the 0th digit: 5
Enter the 1th digit: 4
Enter the 2th digit: 3
The number is: 543

3. Matrix addition:

```
#include <stdio.h>
#include<conio.h>
void main()
{
int m, n, c, d, first[10][10], second[10][10], sum[10][10];
clrscr();
printf("Enter the number of rows and columns of matrix\n");
scanf("%d%d", &m, &n);
printf("Enter the elements of first matrix\n");
for ( c = 0 ; c < m ; c++ )
for ( d = 0 ; d < n ; d++ )
scanf("%d", &first[c][d]);
printf("Enter the elements of second matrix\n");
for ( c = 0 ; c < m ; c++ )
 for ( d = 0 ; d < n ; d++ )
scanf("%d", &second[c][d]);
for ( c = 0 ; c < m ; c++ )
for ( d = 0 ; d < n ; d++ )
sum[c][d] = first[c][d] + second[c][d];
printf("Sum of entered matrices:-\n");
for ( c = 0 ; c < m ; c++ )
{
for ( d = 0 ; d < n ; d++ )
printf("%d\t", sum[c][d]);
printf("\n");
}
getch();
}
```

Output:

```
Enter the number of rows and columns of matrix
2
2
Enter the elements of first matrix
1 2
3 4
Enter the elements of second matrix
5 6
```

2 1
Sum of entered matrices:- 6 8
5 5

FUNDAMENTALS OF STRINGS

A string is a series of characters treated as a single unit. A string may include letters, digits and various special characters such as +, -, *, / and $. String literals or string constants in C are written in double quotation marks as follows:

"1000 Main Street" (a street address)

"(080)329-7082" (a telephone number)

"Kalamazoo, New York" (a city)

In C language strings are stored in array of char type along with null terminating character '\0' at the end.

When sizing the string array we need to add plus one to the actual size of the string to make space for the null terminating character, '\0'.

Syntax:

char fname[4];

The above statement declares a string called fname that can take up to 3 characters. It can be indexed just as a regular array as well.

fname[]={'t','w','o'};

character	t	w	o	\0
ASCII code	116	119	41	0

Generalized syntax is:-

char str[size];

when we declare the string in this way, we can store size-1 characters in the array because the last character would be the null character. For example,

char mesg[10]; can store maximum of 9 characters.

If we want to print a string from a variable, such as four name string above we can do this.

e.g., *printf("First name:%s",fname);*

We can insert more than one variable. Conversion specification %s is used to insert a string and then go to each %s in our string, we are printing.

A string is an array of characters. Hence it can be indexed like an array.

char ourstr[6] = "EED";

– ourstr[0] is 'E'

– ourstr[1] is 'E'

– ourstr[2] is ‘D’

– ourstr[3] is ‘\0’

– ourstr[4] is ‘\0’

– ourstr[5] is ‘\0’

Reading strings:

If we declare a string by writing

char str[100];

then str can be read from the user by using three ways;

1. Using scanf() function
2. Using gets() function
3. Using getchar(), getch(), or getche() function repeatedly

The string can be read using scanf() by writing

scanf(“%s”,str);

Although the syntax of scanf() function is well known and easy to use, the main pitfall with this function is that it terminates as soon as it finds a blank space. For example, if the user enters Hello World, then str will contain only Hello. This is because the moment a blank space is encountered, the string is terminated by the scanf() function.

Example:

```
char str[10];
printf(“Enter a string\n”);
scanf(“%s”,str);
```

The next method of reading a string a string is by using gets() function. The string can be read by writing

gets(str);

gets() is a function that overcomes the drawbacks of scanf(). The gets() function takes the starting address of the string which will hold the input. The string inputted using gets() is automatically terminated with a null character.

Example:

```
char str[10];
printf(“Enter a string\n”);
gets(str);
```

The string can also be read by calling the getchar() repeatedly to read a sequence of single characters (unless a terminating character is encountered) and simultaneously storing it in a character array as follows:

```
int i=0;
char str[10],ch;
getchar(ch);
while(ch!='\0')
{
str[i]=ch; // store the read character in str
i++;
getch(ch); // get another character
}
str[i]='\0'; // terminate str with null character
```

Writing string

The string can be displayed on screen using three ways:

1. Using printf() function
2. Using puts() function
3. Using putchar() function repeatedly

The string can be displayed using pintf() by writing

```
printf("%s",str);
```

We can use width and precision specification along with %s. The width specifies the minimum output field width and the precision specifies the maximum number of characters to be displayed. Example:

```
printf("%5.3s",str);
```

this statement would print only the first three characters in a total field of five charaters; also these three characters are right justified in the allocated width.

The next method of writing a string is by using the puts() function. The string can be displayed by writing:

```
puts(str);
```

It terminates the line with a newline character ('\n'). It returns an EOF(-1) if an error occurs and returns a positive number on success.

Finally the string can be written by calling the putchar( ) function repeatedly to print a sequence of single characters.

```
int i=0;
char str[10];
while(str[i]!='\0')
{
putchar(str[i]); // print the character on the screen
```

```
i++;
}
```

Example:
Read and display a string

```
#include<stdio.h>
#include<conio.h>
void main()
{
char str[20];
clrscr();
printf("\n Enter a string:\n"); gets(str);
scanf("The string is:\n"); puts(str);
getch(); }
```

Output:
Enter a string: vssut burla
The string is: vssut burla

COMMON FUNCTIONS IN STRING

Type	*Method*	*Description*
char	*strcpy(s1, s2)*	Copy string
char	*strcat(s1, s2)*	Append string
int	*strcmp(s1, s2)*	Compare 2 strings
int	*strlen(s)*	Return string length
char	*strchr(s, int c)*	Find a character in string
char	*strstr(s1, s2)*	Find string s2 in string s1

strcpy():

It is used to copy one string to another string. The content of the second string is copied to the content of the first string.

Syntax:

strcpy (string 1, string 2);

Example:

char mystr[10];
mystr = "Hello"; // Error! Illegal !!! Because we are assigning the value to mystr which is not possible in case of an string. We can only use "=" at declarations of C-String.

strcpy(mystr, “Hello”);

It sets value of mystr equal to “Hello”.

strcmp():

It is used to compare the contents of the two strings. If any mismatch occurs then it results the difference of ASCII values between the first occurrence of 2 different characters.

Syntax:

int strcmp(string 1, string 2);

Example:

```
char mystr_a[10] = “Hello”;
char mystr_b[10] = “Goodbye”;
– mystr_a == mystr_b; // NOT allowed! The correct way is
if (strcmp(mystr_a, mystr_b ))
printf ("Strings are NOT the same.");
else
printf( "Strings are the same.");
```

Here it will check the ASCII value of H and G i.e, 72 and 71 and return the diference 1.

strcat():

It is used to concatenate i.e, combine the content of two strings.

Syntax:

strcat(string 1, string 2);

Example:

```
char fname[30]={“bob”};
char lname[]={“by”};
printf(“%s”, strcat(fname,lname));
```

Output:

bobby.

strlen():

It is used to return the length of a string.

Syntax:

int strlen(string);

Example:

char fname[30]={"bob"};
int length=strlen(fname);
It will return 3

strchr():

It is used to find a character in the string and returns the index of occurrence of the character for the first time in the string.

Syntax:

strchr(cstr);

Example:

char mystr[] = "This is a simple string";
char pch = strchr(mystr,'s');
The output of pch is mystr[3]

strstr():

It is used to return the existence of one string inside another string and it results the starting index of the string.

Syntax:

strstr(cstr1, cstr2);

Example:

Char mystr[]="This is a simple string";
char pch = strstr(mystr, "simple");

here pch will point to mystr[10]

String input/output library functions

Function prototype	Function description
int getchar(void);	Inputs the next character from the standard input and returns it as integer
int putchar(int c);	Prints the character stored in c and returns it as an integer
int puts(char s);	Prints the string s followed by new line character. Returns a non-zero integer if possible or EOF if an error occurs

int sprint(char s, char format,….)	Equivalent to printf,except the output is stored in the array s instead of printed in the screen. Returns the no.of characters written to s, or EOF if an error occurs
int sprint(char s, char format,….)	Equivalent to scanf, except the input is read from the array s rather than from the keyboard. Returns the no.of items successfully read by the function , or EOF if an error occurs