



3 Array

Variable can hold only one value at a time where as an array is a collection of variables referenced under a common name hence can be used to store more than one value under a common name. In other words, an array is a homogeneous data structure (elements having same data type) that stores a sequence of consecutively numbered objects allocated in contiguous memory. It contains multiple values of same data type.

NEED OF AN ARRAY

If we want to add two variables say a and b we need to declare them as
`int a,b;`

But suppose if we need sum of 100 variables, it becomes a difficult task for us to declare 100 variables and even more difficult task to remember all the 100 variable names. So to make it simple we can use array.

Like an Excel spreadsheet, arrays have a position number for each row.

Each object of the array can be accessed by using its positions number - index. The positions-index in an array start at 0 and go up sequentially. Each position in the array can hold a value. When we declare an array, we need to set its size.

	A	B	C
1	Position	Value	
2	0	11	
3	1	22	
4	2	33	
5	3	44	
6	4	55	
7			

For example, we want to declare 10 variables of int data type, instead of declaring individual variables, such as no0, no1, ..., and no9, we can declare one array variable such as no and use no[0], no[1], and ..., no[9] to represent individual variables.

Once an array has been created its size cannot be changed.

In C values of the array are placed in contiguous memory location. Each element of the array is identified by its index number. Index number always starts with 0.





There are two types of Array






Properties of Array








-  An array holds elements that have the same data type.
-  Array size should be mentioned in the declaration. Array size must be a constant expression and not a variable.
-  Array have number of memory cell which is called elements.
-  All the elements of array share the common name, and they are distinguished from one another with help of the index.





 ArrayName indicates the base address or address of the first element in the Array.

 Array start with 0 index number & end with one less than size of array.

element	arr[0]	arr[1]	arr[2]	arr[3]	arr[4]
Address	1000	1002	1004	1006	1008

-  The array element is always stored sequential or in linear fashion.
-  When an array is declared & not initialized, it contains garbage values.
-  Two-dimensional array elements are stored row by row in subsequent memory locations.
-  2D arrays are used to represent matrices.
-  An Array cannot be copied/assigned/compared from another Array directly, these operations have to be done element by element basis.



Disadvantages

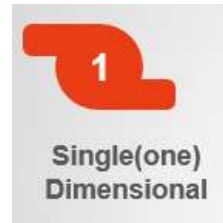
-  We must know in advance that how many elements are to be stored in array.
-  The size of an array is of fixed. The memory which is allocated to array cannot be increased or reduced.
-  Insertions and deletions is time consuming process.
-  Only elements of same data types can be stored in an array. We cannot store elements of multiple data types in a single array.





3.1 One Dimensional Array

-  An array having only one subscript (index) is called one dimensional array.
-  In one dimensional array values are stored in sequential manner which is starting with 0 index.



Syntax#

Data-type array-name[size];

Size is the maximum number of elements stored by the array.

Example#

```
int arrayNo[3];
```

The above example declares a integer array arrayNo of size 3.

In C values of the array are placed in contiguous memory location. Each element of the array is identified by its index number. Array always starts with 0 index which is actually the first element of the array.

By giving the index number with the array name, elements of the array can be accessed.

For example, for the above array declared, we can access its else as shown

arrayNo[0] → it referred to the first element of the array, n.
 arrayNo[1] → it referred to the second element of the array, n.
 arrayNo[2] → it referred to the third element of the array, n.

In the memory, all the array elements are stored in contiguous memory location.

Assume that the values in the array are 10, 4, 6 in the 1st, 2nd and 3rd elements respectively. Pictorially we can draw as:

	arrayNo[0]	arrayNo[1]	arrayNo[2]	
Memory	10	4	6	→

In the memory, values 10, 4 and 6 stored in contiguous memory location.

3.1.1 Initialization of the array:

There are two ways to initialize an array

1. Static – at the time of declaration
2. Dynamic – at runtime from the user or on the basis of existing values

3.1.1.1 Static initialization of One Dimensional array





At the time of array declaration, array can have initialized.

3.1.1.1.1 Initialize all the values as 0

Syntax#

```
data-type array-name[size] = { };
```

We can initialize all the elements of an integer Array with 0 value by assigning with empty { } as below,

Example#

```
int a[5] = { };
```

3.1.1.1.2 Initialize list of values

Syntax#

```
data-type array-name[size] = { list of values };
```

Size of the array is optional. If you do not specify the size, size will be set according to the number of values assigned.

List of values are the value assigned to each elements of the array and are separated by comma.

First value in the list will be assign to first element, second value to second element so on.

Example#

```
int a[3] = { 5, 7, 20 };
```

Three integer values are assigned to array n. we can indicate each element values as:

```
a[0] = 5
```

```
a[1] = 7;
```

```
a[2] = 20;
```

Now we can access each element of the array by giving the index number.

Suppose, to print the value of the array element at index 0, we can write as

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

it will print 5.

Example# To declare and initialize one dimensional array and print its value

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[3]={12, 6, 15};
    clrscr();
    printf("\n %d %d %d", a[0], a[1], a[2]);
    getch();
}
```





```
}  
Output:  
12 6 15
```

We can also assign a value to each array elements after declaration.

```
Example#   int a[3];  
           a[0] = 12;  
           a[1] = 6;  
           a[2] = 15;
```

Example# To print array element using for loop

```
#include<stdio.h>  
#include<conio.h>  
void main()  
{  
    int a[3]={12, 6, 15};  
    int i;  
    clrscr();  
    for(i=0; i<3; i++)  
    {  
        printf("\n Array element n[%d] =%d ", i, a[i]);  
    }  
    getch();  
}
```

Output:
Array element a[0] =12
Array element a[1] =6
Array element a[2] =15

3.1.1.2 Dynamic initialization of array

Instead of providing the value to the array in the program, we can also assign values to array during runtime.

Example#

```
int n[3];  
scanf ("%d %d %d", &n[0], &n[1], &n[2]);
```

Or

```
Using loop,  
for(i=0; i<3; i++)  
{  
    scanf("%d", &n[i]);  
}
```

Assigning value to array during run time is useful when you want to read big array say of size 1000.





Dr. Shyam N. Chawda, C Language Tutorial , 78 74 39 11 91

Suppose you have declared an array of 1000 elements. Declaration of array is not difficulty but to initialize or assign the value of the array of 1000 elements is cumbersome.

Better approach is to apply the values during program execution.

Note: To read and print one dimensional numeric array, one loop statement is used. This loop is used to change the index number of the array.

Example# To read and print one dimensional array using for loop

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

void main()
{
    int a[3];
    int i;
    clrscr();
    for(i=0; i<3; i++)
    {
        printf("\n Enter array value for element n[%d]" , i);
        scanf("%d", &a[i]); /* read array */
    }
    printf("Array contains \n");
    for(i=0; i<3; i++)
    {
        printf(" %d ", a[i]);
    }
    getch();
}
```

Output:

```
Enter array value for element a[0] = 2
Enter array value for element a[1] = 12
Enter array value for element a[2] = 42
Array contains
2 12 42
```

3.1.2 Accessing the array

Processing the array is not as simple as processing the simple variable.
For example,

```
int a, b, sum ;
a=2; b=4;
sum= a + b;
```

} Two integer numbers can be easily added by writing the formulas
sum= a+b;
but this is no true for array variables.

Consider the same example using array variables.





```
int a[3], b[3], sum;
```

```
.....
```

```
.....
```

sum= a +b; → this is invalid because for array each element of the array must be added separately by the use of index number.

Following program shows how to access each array element for finding out the sum of all the elements of the array.

Example# To read and find the sum of all the elements of an array.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int num[3], sum;
    int i;
    sum=0;
    clrscr();
    for(i=0; i<3; i++)
    {
        printf("\n Enter array value for element n[%d]" , i);
        scanf("%d", &num[i]);    //read array
    }
    for(i=0; i<3; i++)
    {
        sum = sum + num[i] ; //add each element value to the sum variable
    }
    printf("\n The sum of integer array : %d", sum);
}
```

Output:

```
Enter array value for element n[0] = 5
Enter array value for element n[1] = 7
Enter array value for element n[2] = 8
The sum of integer array: 20
```





Following program shows how to access the each array element for finding out the maximum value from the array.

Example# To find the maximum value from an array

```
#include<stdio.h>
int main()
{
    int num[3], largest;
    int i;
    clrscr();
    for(i=0; i<3; i++)
    {
        printf("\n Enter array value for element n[%d]" , i);
        scanf("%d", &n[i]); /* read array */
    }
    largest=num[0];
    for(i=0; i<3; i++) /* this loop find out largest element */
    {
        if(num[i] > largest)
            largest = num[i];
    }
    printf("\n The largest element in the array is %d", largest);
    getch();
}
```

Output:

```
Enter array value for element n[0] = 5
Enter array value for element n[1] = 7
Enter array value for element n[2] = 8
The largest element in the array is 8
```

3.1.3 Memory allocation for the array:

In C, memory is allocated for each data type, for example, integer type has 2 bytes, long type has 4 byte, float type has 4 byte etc.

Because array is a collection of elements of similar data type, memory is allocated to each element of the array.

Example#

```
int n[3];
```



integer data type occupies 2 bytes memory, 2 bytes is occupied by each element of the array, n.

Therefore, total byte allocated to array n is $2 \times 3 = 6$ bytes.





Example# Program to demonstrate use of sizeof() operator

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int n[3];
    clrscr();
    printf ("\n length of integer is %d bytes", sizeof(int));
    printf ("\n length of array is %d bytes", sizeof(n) );
    getch();
}
```

Output:

length of integer is 2 bytes
length of array is 6 bytes

Example# Program to demonstrate use of multiple array.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int rollno[5], maths[5], english[5], science[5], total[5];
    float percentage[5];
    int i;
    clrscr(); /* to clear the screen before display any output */
    printf (" Enter student's result data for five students \n");
    for(i=0; i<5; i++)
    {
        printf ("\n Enter rollno, three subject marks for Maths, English,
        Science: ");
        scanf ("%d %d %d %d", &rollno[i], &maths[i], &english[i],
        &science[i]);
    }
    for(i=0; i<5; i++)
    {
        total[i] = maths[i] + english[i] + science[i];
        percentage[i] = float (( total[i] * 100)/ 3 );
    }
    printf ("\n Result of the students \n");
    printf ("Rollno maths English science total percentage");
    for(i=0; i<5; i++)
    {
        printf ("\n %d/t %d /t %d /t %d /t %d /t %f", rollno[i], maths[i],
        english[i], science[i], total[i], percentage[i]);
    }
    getch();
}
```





```
}
```

Output:

Enter student's result data for five students

Enter rollno, three subject marks for maths, English, science: 1 50 60 70

Enter rollno, three subject marks for maths, English, science: 2 45 55 66

Enter rollno, three subject marks for maths, English, science: 3 67 78 89

Enter rollno, three subject marks for maths, English, science: 4 71 81 91

Enter rollno, three subject marks for maths, English, science: 5 80 90 95

Result of the students

Rollno	maths	English	science	total	percentage
--------	-------	---------	---------	-------	------------

1	50	60	70	180	60.000000
---	----	----	----	-----	-----------

2	45	55	66	166	55.333332
---	----	----	----	-----	-----------

3	67	78	89	234	78.000000
---	----	----	----	-----	-----------

4	71	81	91	243	81.000000
---	----	----	----	-----	-----------

5	80	90	95	265	88.333336
---	----	----	----	-----	-----------

Example# Program to arrange an integer array in ascending order

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n[5], i, j, t;
    i=0;
    clrscr();
    while(i<5)
    {
        printf("Enter integer value of n[%d] element: ",i);
        scanf("%d", &n[i]);
        i++;
    }
    i=0;
    j=0;
    for(j=0; j<5; j++) /* these nested loop interchange the element value if
                        previous element is greater than next element */
    {
        for(i=j+1; i<5; i++)
        {
            if(n[j]>n[i])
            {
                t=n[j];
                n[j]=n[i];
                n[i]=t;
            }
        }
    }
    i=0;
```





```
printf("\n array in ascending order \n");
while(i<5)
{
    printf(" %d ", n[i]);
    i++;
}
}
```

Output:

Enter integer value of n[0] element: 3
 Enter integer value of n[1] element: 13
 Enter integer value of n[2] element: 14
 Enter integer value of n[3] element: 2
 Enter integer value of n[4] element: 5
 array in ascending order
 2 3 5 13 14

3.2 Two-dimensional integer array (Matrix)

In 2-dimentional array elements are arranged in row and column format. When we are working with 2-dimentional array we require to refer 2-subscript operator which indicates row and column sizes. The main memory of 2-dimentional array is rows and sub-memory is columns.

Syntax#

int ARRAY-NAME [ROWS][COLUMNS];

ROWS and COLUMNS are size of array which sets the number of rows and columns respectively.

3.2.1 Initialization of two-dimensional Array at Compile time / Static

int table[2][3] = { 10,20,30,40,50,60};

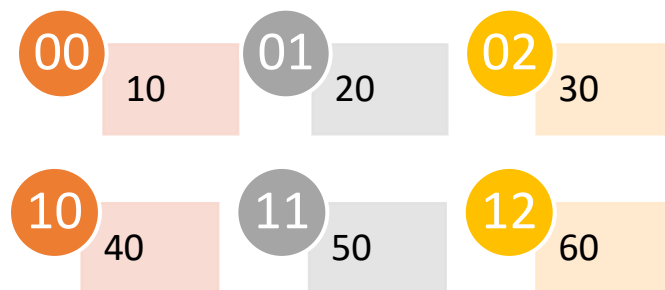
int table[2][3] = { { 10,20,30},{ 40,50,60} };

If the values are missing in an initialize, they are automatically set to zero.

int table[2][3] = { { 1,1},{ 2} };

When all the elements are to be initialized to zero following method is used:

int table[2][3] = { {0},{0} };



3.2.2 Initialization two-dimensional array at Runtime

We can also assign the value at run time from users with the help of scanf(), see the following example.





Example# To take input and then display two-dimensional array(matrix)

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

void main()
{
    int a[3][3],i,j;
    clrscr();
    //loop for taking input
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("Enter value of a[%d][%d]: ",i,j);
            scanf("%d",&a[i][j]);
        }
    }
    //loop to display matrix
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf(" %d ",a[i][j]);
        }
        printf("\n");
    }
    getch();
}
```

Output:

```
Enter value of a[0][0]:1
Enter value of a[0][1]:2
Enter value of a[0][2]:3
Enter value of a[1][0]:4
Enter value of a[1][1]:5
Enter value of a[1][2]:6
Enter value of a[2][0]:7
Enter value of a[2][1]:8
Enter value of a[2][2]:9
1 2 3
4 5 6
6 7 8
```





Example# To add two matrix

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

void main()
{
    int a[2][2],b[2][2],c[2][2],i,j;
    clrscr();
    //loop for taking input in a
    for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
        {
            printf("Enter value of a[%d][%d]: ",i,j);
            scanf("%d",&a[i][j]);
        }
    }
    //loop for taking input in b
    for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
        {
            printf("Enter value of b[%d][%d]: ",i,j);
            scanf("%d",&b[i][j]);
        }
    }
    //loop to add two matix
    for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
        {
            c[i][j]=a[i][j]+b[i][j];
        }
    }

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





Output:

```
Enter value of a[0][0]:1
Enter value of a[0][1]:2
Enter value of a[1][0]:6
Enter value of a[1][1]:4
Enter value of b[0][0]:2
Enter value of b[0][1]:3
Enter value of b[1][0]:7
Enter value of b[1][1]:5
3 5
13 9
```

Example# To print transpose of a matrix

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

void main()
{
    int a[3][3],b[3][3],i,j;
    clrscr();
    //loop for taking input
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("Enter value of a[%d][%d]: ",i,j);
            scanf("%d",&a[i][j]);
        }
    }
    //loop for transpose
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            b[i][j]=a[j][i];
        }
    }
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf(" %d ",b[i][j]);
        }
        printf("\n");
    }
    getch();
}
```





Output:

```
Enter value of a[0][0]:1
Enter value of a[0][1]:2
Enter value of a[0][2]:3
Enter value of a[1][0]:4
Enter value of a[1][1]:5
Enter value of a[1][2]:6
Enter value of a[2][0]:7
Enter value of a[2][1]:8
Enter value of a[2][2]:9
1 4 7
2 5 8
3 6 9
```

Example# To diagonal elements of a matrix

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

void main()
{
    int a[3][3],i,j;
    clrscr();
    //loop for taking input
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("Enter value of a[%d][%d]: ",i,j);
            scanf("%d",&a[i][j]);
        }
    }
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            if(i==j)
                printf(" %d ",a[i][j]);
            else
                printf(" * ");
        }
        printf("\n");
    }
    getch();
}
```

Output:

```
Enter value of a[0][0]:1
Enter value of a[0][1]:2
Enter value of a[0][2]:3
```





```
Enter value of a[1][0]:4
Enter value of a[1][1]:5
Enter value of a[1][2]:6
Enter value of a[2][0]:7
Enter value of a[2][1]:8
Enter value of a[2][2]:9
1 **
* 5 *
** 9
```

3.3 One dimensional character array (string)

Character arrays are known as a String.

's','h','y','a','m'

A string is nothing but an array of characters terminated by '\0'.

"Shyam"

The way a group of integers can be stored in an integer array, similarly a group of characters can be stored in a character array.

Character array is terminated by null character '\0'.

When we define a character array as 'char name[10]', this indicate that the array 'name' can hold a string of length ten character.

3.3.1 Array initialization

3.3.1.1 Static initialization

Character array can be initialized at the time of declaration.

Syntax#
char arrayName[]={""}

or

char arrayName[]={ ' ',' ',' ',' ',' ' }

Example#

char name[] = {"jayul"};
Or
char name[]={ 'j', 'a', 'y', 'u', 'l' };

Each character is treated as element of the array 'name'. Internal representation in memory is:

'j'	'a'	'y'	'u'	'l'	'\0'
-----	-----	-----	-----	-----	------

Each string is terminated by null character ('\0'). Null character indicates the end of string.





One extra byte must be considered for null character while deciding the length of the string.

3.3.2 Reading and Writing Strings

3.3.2.1 For Reading Strings

To read from the user

- scanf()
- [^\n]
- gets () function reads line from keyboard.
- getchar () function reads character from keyboard.

3.3.2.1.1 Reading String using scanf() function

Syntax#

scanf("%s",&VariableName);

Example# problem with simple scanf

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

main()
{
char name[25];
char quali[25];
clrscr();

printf("Enter name ->");
scanf("%s",name);

printf("Enter Qualification ->");
scanf("%s",quali);

printf("Name = %s and Quali = %s",name,quali);

getch();
}
```

Output:

```
Enter name ->shyam
Enter Qualification ->BCA MCA
Name = shyam and Quali = BCA *(problem)
```

scanf() function reads until a whitespace character is found in a input or the maximum number of characters have been read.

scanf() function ignores MCA which is after space, so the solution is? %[^\n]





3.3.2.1.2 Reading String using %[^\n]

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

main()
{
char name[25];
char quali[25];
clrscr();

printf("Enter name ->");
scanf("%s",name);

fflush(stdin);

printf("Enter Qualification ->");
scanf("%[^\n]",quali);

printf("Name = %s and Quali = %s",name,quali);

getch();
}
```

Output:

```
Enter name ->Shyam
Enter Qualification ->BCA MCA
Name = Shyam and Quali = BCA MCA
```

%[^\n] reads character upto Enter key

3.3.2.1.3 Reading String using gets() function

This function gets an entire string of characters from the user and stores them in an array.

To read multi word string gets function is better option.

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

main()
{
char name[25];
```





```
char quali[25];
clrscr();

printf("Enter name ->");
gets(name);

fflush(stdin);

printf("Enter Qualification ->");
gets(quali);

printf("Name = %s and Qualification = %s",name,quali);

getch();
}
```

Output:

```
Enter name ->shyam chawda
Enter Qualification ->BCA MCA
Name = shyam chawda and Qualification = BCA MCA
```

3.3.2.1.4 Reading String using getchar()

It reads single character from the input and place them into a character array.

An Entire line of text can be read and stored in an array. The reading is terminated when the newline **character('\n')** is entered and the null character is then instead at the end of the string.

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

main()
{
char name[25],quali[25],ch;
int i=0;
clrscr();

printf("Enter name ->");
gets(name);

fflush(stdin);

printf("Enter Qualification ->");

do
{
ch=getchar();
```





```
    quali[i]=ch;
    i++;
}while(ch!='\n');

    i=i-1;
    quali[i]='\0'; 'Otherwise print garbage

    printf("Name = %s and Quali = %s",name,quali);

getch();
}
```

Output:

Enter name ->shyam chawda
Enter Qualification ->BCA MCA
Name = shyam chawda and Quali = BCA MCA

3.3.2.2 For printing String,

There are three functions to print character Array

- printf("%s",stringVariableName) function is used to to writes line on standard output/screen.
- puts(stringVariableName) function is used to to writes line on standard output/screen.
- putchar(stringVariableName[pos]) function is used to write a character on standard output/screen.

Example#

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

main()
{
    char name[100];
    int i=0;
    clrscr();

    printf("\nEnter name =>");
    gets(name);

    printf("\nWelcome %s\n",name);

    printf("\nputs\n");
    puts(name);

    printf("\nputchar\n");
```





```
while(name[i]!='\0')
{
    putchar(name[i]);
    i++;
}

getch();
}
```

Output:

Enter name =>Bhavin Soni

Welcome Bhavin Soni

puts
Bhavin Soni

putchar
Bhavin Soni

Other Examples of char Array

Example# Count 'k' character from the string

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

main()
{
    char name[100];
    int i=0,cnt=0;
    clrscr();

    printf("\nEnter name =>");
    gets(name);

    while(name[i]!='\0')
    {
        if(name[i]=='k')
        {
            cnt++;
        }
        i++;
    }

    printf("\nOccurence of K is %d",cnt);
```





```
getch();  
}
```

Output

Enter name =>khdak sing ke khdak

Occurrence of K is 5

Example# Count no of Upper Characters and Lower Characters in the String

```
#include<stdio.h>  
#include<conio.h>  
  
main()  
{  
    char name[100];  
    int i=0,up=0,low=0;  
    clrscr();  
  
    printf("\nEnter name =>");  
    gets(name);  
  
    while(name[i]!='\0')  
    {  
        if(name[i]>='a' && name[i]<='z')  
        {  
            low++;  
        }  
        else if(name[i]>='A' && name[i]<='Z')  
        {  
            up++;  
        }  
        i++;  
    }  
  
    printf("\nTotal Length = %d\nUpper char = %d\nLower char = %d",i,up,low);  
  
    getch();  
}
```

Output

Enter name =>Khadak Sing KE

Total Length = 14

Upper char = 4

Lower char = 8





3.3.3 String handling using build-in functions

C supports various string handling functions such as strcpy(), strcmp(), strlen(), strcat() and these functions are available in string.h header file.

3.3.3.1 strcpy()

This function is used to copy one string to another string.

Syntax#

```
strcpy(string2, string1);
```

Copy the contents of string1 to string2. Both string1 and string2 can be string array or constant.

Example# To copy string using strcpy() function

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
    char str1[10], str2[10];
    clrscr();
    printf("\n Enter str1 :");
    scanf("%s",&str1);
    printf("\n Enter str2 :");
    scanf("%s",&str2);
    printf("\n Before coping:");
    printf("\nstr1:%s \t str2:%s",str1,str2);

    strcpy(str2, str1);

    printf("\n After coping:");
    printf("\nstr1:%s \t str2:%s",str1,str2);
    getch();
}
```

Output:

```
Enter str1 : Ram
Enter str2 : Seeta

Before coping:
str1:Ram    str2:Seeta
After coping:
```





str1:Seeta str2:Seeta

3.3.3.2 strlen()

To find out the number of characters in the string and it returns the length of the string.

Syntax#

identifier = strlen(string);

String argument can be array of string or string constant. The string length is stored in identifier which is of integer type variable.

Example# Program to find length of the string using strlen() function and print string using for loop.

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

void main()
{
    char str1[30];
    int l, k;
    clrscr();
    printf("\n Enter string :");
    gets(str1);
    l=strlen(str1);
    printf("\n The length of the string is %d \n",l);
    for(k=0; k<l-1; k++)
    {
        printf("%c", str1[k]);
    }
    getch();
}
```

Output:

Enter string: Ahmedabad is a metro city

The length of the string is 25
Ahmedabad is a metro city

Example# Print Reverse string

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

main()
```





```
{
char name[100];
int i=0,up=0,low=0;
clrscr();

printf("\nEnter name =>");
gets(name);

for(i=strlen(name)-1;i>=0;i--)
{
printf("%c",name[i]);
}

getch();
}
```

Output

Enter name =>Dell Computers
sretupmoC lleD

3.3.3.3 strcat()

To concate(join) two strings together.

Syntax#

strcat(string1, string2);

String1 and sting2 are array of string. String2 is added at the end of the string1.

Here the length of the string1 must be sufficient enough to handle both the strings.

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

void main()
{
    char name[25], sur_name[10];
    int l, k;
    clrscr();
    printf("\n Enter two strings :");
    scanf("%s %s", &name, &sur_name);
    strcat(name, " "); /* to add one space after name */
    strcat(name, sur_name); /* to add sur_name after name */
    printf("string after join operation is %s", name);
    getch();
}
```





Output:

Enter string: Varun Dhawan
String after join operation is Varun Dhawan

3.3.3.4 strcmp()

It compares two string and return integer value.

Syntax#

indetifier = strcmp(string1, string2);

String1 and sting2 are character array(string). Identifier is the name of integer variable which store the value returned by the function.

If the both the strings are equal than function return 0 value.
If not, then it return any non-zero value.

Example# To compare to string using strcmp()

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
    char str1[20], str2[10];
    int i;
    clrscr();
    printf("Enter first string: ");
    scanf("%s", &str1);
    printf("Enter second string: ");
    scanf("%s", &str1);
    i = strcmp(str1,str2);
    if(i==0)
    {
        printf("Both strings are equal %d", i);
    }
    else
    {
        printf("Both strings are not equal %d", i);
    }
    getch();
}
```

Output:

Enter first string: Baroda
Enter second string: Surat
Both strings are not equal -1.





3.3.3.5 strlwr()

strlwr() convert string to lower case

Syntax#

strlwr(string_name);

Example#

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

void main()
{
    char str[20];
    clrscr();

    printf("Enter a string: ");
    gets(str);

    strlwr(str);
    printf("Lowered string is: %s",str);
    getch();
}
```

Output:

Enter a string: I Love India

Lowered string is: i love india

3.3.3.6strupr()

strupr() convert string to upper case.

Syntax#

strupr(string_name);

Example#

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

void main()
{
    char str[20];
    clrscr();

    printf("Enter a string: ");
    gets(str);
```





```
strupr(str);  
printf("Uppered string is: %s",str);  
getch();  
}
```

Output:

Enter a string: To travel is to live

Uppered string is: TO TRAVEL IS TO LIVE

