Arrays and Strings

Introduction to Arrays:

Sometimes, it is necessary for the programmer's to store and process large volume of similar type of data elements in in computer's memory while solving complex problems using a computer system and is possible by using derived data type array.

For examples:

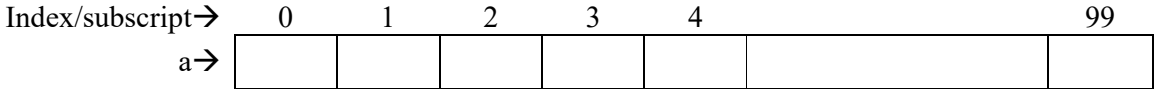
1. To store roll number of 100 students in computer's memory, we define only one array i.e. **int rno[100];** instead of defining 100 separate variables.
2. To store average marks of 100 students in computer's memory, we define only one array i.e. **float avg[100];** instead of defining 100 separate variables.
3. To store name of one student i.e. **char name[25];**

What is an array?

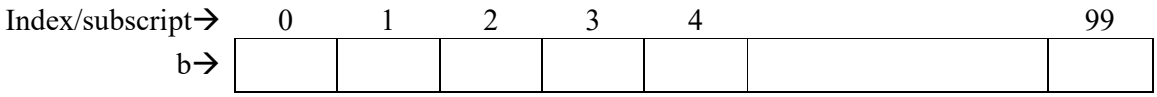
"An array can be defined as a collection of related data elements of **similar data type** stored in computer's continuous memory. The data elements can be processed by using a common name and different index value that starts from 0 and ends with ArraySize-1"

Examples:

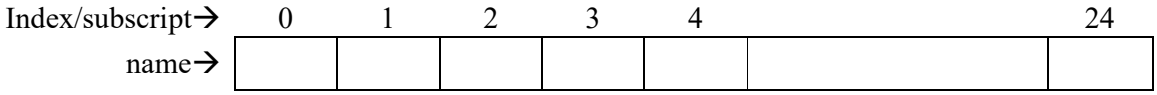
1. An array to store 100 integer values. **int a[100];**



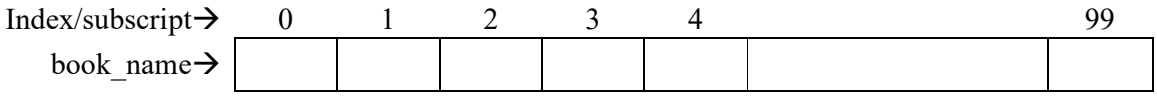
2. An arrays to store 100 floating point numbers. **float b[100];**



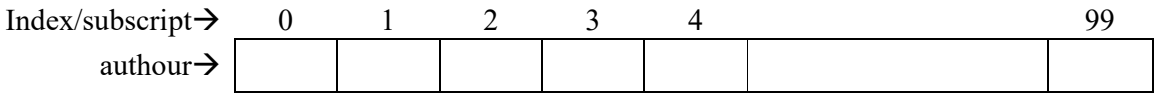
3. An array to store name of a single student. **char name[25];**



4. An array to store book name. **char book_name[100];**



4. An array to store author name. **char authour[100];**



Types of arrays:

The arrays can be classified according to the volume of data to be stored in computer's memory.

1. One dimensional array e.g. int a[100];
2. Two dimensional array e.g. int a[100][100];
3. Multi dimensional array e.g. int a[100][100][100];

1. One dimensional array:

One dimensional array can be defined as a collection of related data elements of the same data type stored in computer's consecutive memory. The array elements can be accessed by using a common name with different index that starts with 0 and ends with Array Size – 1 . It helps to store and process linear list of data elements.

Declaration of one-dimensional array:

The one dimensional array must be declared/defined in the declaration part of the main() before the array elements are used in the executable part of main() by using following syntax.

Syntax:

data_type array_name[SIZE];

where,

data_type → the data type of data to be stored and processed in computer's memory like int, float, char, double, long int, etc.

array_name → valid identifier

SIZE→ the integer constant indicating maximum number of data elements can be stored.

Examples:

int a[5];

Index→	0	1	2	3	4
a					

float price[5];

Index→	0	1	2	3	4
price					

double result[5];

Index→	0	1	2	3	4
result					

char name[5];

Index→	0	1	2	3	4
name					

char city[5];

Index→	0	1	2	3	4
city					

Note: Computer allocates 5 memory locations with garbage (unknown) values

Initialization of one dimensional array:

The process of assigning the values to the defined array elements is called initialization of array. The arrays can be initialized with the values in the following two ways.

1. Compile Time Array Initialization
2. Run Time Array Initialization

1. Compile Time Array Initialization: If the values are well known by programmers, then the programmer makes the use of compile time initialization. The process of assigning the values during the declaration of arrays in the declaration part of main() is called compile time initialization. The syntax is as follow.

Syntax:

data_type array_name[SIZE]={list of values separated by commas};

Examples:

1. int a[5]={10,20,30,40,50};

   or

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

Index→	0	1	2	3	4
A	10	20	30	40	50

2. float price[5]={55.50,99.50,100.50,75.60,50.60};

Index→	0	1	2	3	4
price	55.50	99.50	100.50	75.60	50.60

3. char name[5]="John";or char name[5]={'J','o','h','n'};

   or char name[ ]="John"

Index→	0	1	2	3	4
Name	J	O	h	n	'\0'

   char name[5]={'J'};

Index→	0	1	2	3	4
Name	J	'\0'	'\0'	'\0'	'\0'

4. int a[5]={0,0,0,0,0}; or int a[5]={0};

Index→	0	1	2	3	4
a	0	0	0	0	0

5. int a[5]={10, 20};

Index→	0	1	2	3	4
a	10	20	0	0	0

6. int a[3]={1,2,3,4,5}; /* invalid i.e. to many initialization*/

1. Run Time Array Initialization: If the values are not known by programmers in advance then the programmer makes the use of run time initialization. It helps the programmers to read unknown values from the end users of a program by the keyboard by using input functions like scanf() getch() and gets(),etc.

Examples:

1. To read and display 5 integer values from a keyboard by using an array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5],i;
clrscr();
printf("\n Enter any 5 integer values");
for(i=0;i<5;i++)
{
scanf("%d",&a[i]);
}
for(i=0;i<5;i++)
{
printf("\n%d",a[i]);
}
getch();
}
```

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

o/p:

Enter any 5 integer values

10 20 30 40 50

2. To read and display ‘n’ integer values from a keyboard by using an array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[50],i,n;
clrscr();
printf("\n Enter n: ");
scanf("%d",&n);
printf("\n Enter %d integer values: ",n);
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("\n Array Elements ");
for(i=0;i<n;i++)
{
printf("\n%d",a[i]);
}
getch();
}
```

o/p:
Enter n: 7
Enter 7 integer values
10 20 30 40 50 60 70
Array Elements
10
20
…
70

Assignment Questions:

1. What is one dimensional array? Explain its declaration and initialization with syntax and examples. (2+2+2=6 marks) (Model QP)
2. What is two dimensional array? Explain its declaration and initialization with syntax and examples. (2+2+2=6 marks)
3. Write a c program to implement bubble sort technique.
4. Write a c program to find largest of ‘n’ numbers.
5. Write a c program to read N integers into an array A and to
   i. Find the sum of odd numbers
   ii. Find the sum of even numbers
   iii. Find the average of all numbers

   Output the results computed with appropriate headings. (06 marks June/July 2015)
6. Write a program to multiple of two arrays of given order a[m x n] and b[p x q] (09 marks Jan. 2015)

Anyone who stops learning is old, whether at twenty or eighty~ Henry Ford

“When Henry Ford decided to produce his famous V-8 motor, he chose to build an engine with the entire eight cylinders cast in one block, and instructed his engineers to produce a design for the engine. The design was placed on paper, but the engineers agreed, to a man, that it was simply impossible to cast an eight-cylinder engine-block in one piece.

Ford replied,"Produce it anyway.”~ Henry Ford

What we usually consider as impossible are simply engineering problems ~ Michio Kaku

***

Programming Examples on one dimensional arrays

```
/* To find largest and lowest of n numbers*/
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5],i,n,max,min;
clrscr();
printf("\n Enter n: ");
scanf("%d",&n);
printf("\n Enter %d values: ",n);
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
max=min=a[0];
for(i=1;i<n;i++)
{
if (a[i]>max)
max=a[i];
if (a[i]<min)
min=a[i];
}
printf("\n Largest number is %d",max);
printf("\n Lowest number is %d",min);
getch();
}
```

o/p:

Enter n: 5

Enter 5 values: 10 2 45 67 12

Largest number is 67

Lowest number is 2

```
/* To find sum of odd,even,all and avg of N numbers by using array */
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5],i,n,sum=0,esum=0,osum=0;
float avg;
clrscr();
printf("\n Enter n: ");
scanf("%d",&n);
printf("\n Enter %d values: ",n);
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
for(i=0;i<n;i++)
{
        if (a[i]%2==0)
        esum=sum+i;
        else
        osum=osum+i;
sum=sum+a[i];
}
avg=(float)sum/n;
printf("\n Sum of even numbers=%d",esum);
printf("\n Sum of odd numbers=%d",osum);
printf("\n Sum of all numbers=%d",sum);
printf("\n Average of all numbers=%f",avg);
getch();
}
```

o/p:

```
Enter n: 5
Enter 5 values: 1 2 3 4 5
Sum of even numbers = 9
Sum of even numbers = 6
Sum of all numbers = 15
Average of all numbers =3.000000
```

```
/* Program to sort n numbers using Bubble sort technique */
#include<stdio.h>
#include<conio.h>
void main()
{
int n, a[25], i, j, temp;
clrscr();
printf("Enter n:");
scanf("%d",&n);
printf("\n Enter any %d values",n);
for(i=0;i<n;i++)
        scanf("%d",&a[i]);
printf("\nBefore sorting\n");
for(i=0;i<n;i++)
        printf("\t%d",a[i]);
for(i=0;i<n;i++)
{
        for(j=0; j<n-i-1 ;j++)
        {
         if (a[j]> a[j+1])
         {
         temp=a[j];
         a[j]=a[j+1];
         a[j+1]= temp;
         }
        }
}
printf("\nAfter sorting\n");
for(i=0;i<n;i++)
        printf("\t%d",a[i]);
getch();
}
```

o/p:
Enter n: 5
Enter any 5 values: 5 4 3 2 1
Before Sorting
5 4 3 2 1
After Sorting
1 2 3 4 5

```
/* Program to find sum and average of N numbers using array */
#include<stdio.h>
#include<conio.h>
void main()
{
int a[25],n,i,sum=0;
float avg;
clrscr();
printf("Enter n:");
scanf("%d",&n);
printf("\nEnter any %d values",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<n;i++)
{
sum=sum+a[i];
}
avg=(float)sum/n;
printf("\n Sum of all numbers=%d",sum);
printf("\n Average of all numbers=%f",avg);
getch();
}
```

o/p:
Enter n: 5
Enter any 5 values: 1 2 3 4 5
Sum of all numbers = 15
Average of all numbers = 3.000000

Bell Teacher: **A new student approached the Zen master and asked how he should prepare himself for his training. "Think of me a bell," the master explained. "Give me a soft tap, and you will get a tiny ping. Strike hard, and you'll receive a loud, resounding peal."**

People's reactions to this story:

"You get out of something what you put into it."
"The more you try, the more a good teacher will help."
"Be careful what you ask for. The universe may just provide you with what you seek."
"Sounds like the master is saying pay me a lot, and I will help you a lot; pay me little, and that's what I'll give you in return."
"I think the teacher was warning the student that if he is struck he will strike back with equal force."

```
/* Program to implement linear search technique by using an array*/
#include<stdio.h>
#include<conio.h>
void main()
{
int a[25],n,i,key ;
clrscr();
printf("Enter n:");
scanf("%d",&n);
printf("\n Enter any %d values",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\n Enter item to be search:");
scanf("%d",&key);
for(i=0;i<n;i++)
{
        if (key==a[i])
        {
        printf("\n Item found at %d position",i+1);
        getch();
        exit();
        }
}
printf("\n Item not found!");
getch();
}
```

```
o/p:
Enter n: 5
Enter any 5 values: 15  22  31  88  55
Enter item to be search: 88
Item found at 4 position

o/p:
Enter n: 5
Enter any 5 values: 15  22  31  88  55
Enter item to be search: 100
Item not found !
```

```
/* Program to implement binary search technique by using an array*/
#include<stdio.h>
#include<conio.h>
void main()
{
int a[25], n, i, key, low, high, mid;
float avg;
clrscr();
printf("Enter n:");
scanf("%d",&n);
printf("\n Enter any %d values in ascending order: ",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\n Enter item to be search:");
scanf("%d",&key);
low=0;
high=n-1;
while(low<=high)
{
mid=(low+high)/2;
        if (key==a[mid])
        {
        printf("\n Item found at %d position",mid+1);
        getch();
        exit();
        }
        if (key>a[mid])
        low=mid+1;
        if (key<a[mid])
        high=mid-1;
}
printf("\n Item not found!");
getch();
}
```

```
o/p:
Enter n: 5
Enter any 5 values in ascending order: 1  2  3  4  5
Enter item to be search: 3
Item found at 3 position
```

```
o/p:
Enter n: 5
Enter any 5 values in ascending order: 1  2  3  4  5
Enter item to be search: 8
Item not found !
```

Two dimensional arrays:

A two dimensional array helps the programmers *to store and process table of data elements of similar type* in computer's memory. It is similar to a matrix containing numbers of rows and columns. The rows and columns of 2D array are indexed by starting from 0(zero). i.e row index starts with 0 and end with ROW_SIZE-1 and column index starts with 0 and end with COLUMN_SIZE-1.

The 2D arrays elements can be processed by using common array name with two index representing row and column.

Examples:

1. To store and process 6 subject marks of 5 students.

int a[5][6];

	Col-0	Col-1	Col-2	Col-3	Col-4	Col-5
Row-0	65	75	90	80	66	68
Row-1						
Row-2						
Row-3						
Row-4	50	60	70	80	90	99

In each row, programmer can store 6 subject marks of each student.

2. To store and process *table of integer values* containing 2 rows & 3 columns.

int a[2][3];

	0	1	2
0			
1			

3. To store and process *table of floating point numbers* of the size 3 x 2.

float b[3][2];

	0	1
0		
1		
2		

Declaration of two-dimensional array:

The two dimensional array must be declared/defined in the declaration part of the main() before the array elements are used in the executable part of main() by giving common name with two indexes representing no. of rows and columns. The syntax is as follow.

Syntax:

data_type array_name[ROW_SIZE] [COLUMN_SIZE];

where,

data_type → the data type of data to be stored and processed in computer's memory like int, float, char, double, long int, etc.

array_name → valid identifier

ROW_SIZE & COLUMN_SIZE→ the maximum number of row and column elements can be stored.

Examples:

1. To store the integer values for 5 x 5 matrix.

int a[5][5];

	Col-0	Col-1	Col-2	Col-3	Col-4
Row-0					
Row-1					
Row-2					
Row-3					
Row-4					

Total memory occupied by this array is (5 x 5 x 2 bytes=50 bytes)

2. To store and process *table of integer values* containing 2 rows & 3 columns.

int a[2][3];

	0	1	2
0			
1			

3. To store the names of 100 students

char name[100][25];

Total memory occupied by this array is (100 x 25 x 1 byte=25000 bytes)

Initialization of two dimensional array:

The process of assigning the values to the defined array elements of 2D array is called initialization of 2D array. The array elements can be initialized with different in the following two ways.

1. Compile Time Array Initialization
2. Run Time Array Initialization

1. Compile Time Array Initialization: If the values are well known by programmers in advance, then the programmer makes the use of compile time initialization. i.e the process of assigning the values during the declaration of arrays in the declaration part of main() is called compile time initialization. The syntax is as follow.

Syntax:

```
data_type    array_name[ROW_SIZE] [COLUMN_SIZE]=
        {
        {list of values },
        {list of values },
        {list of values }
        };
```

Examples:

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

or

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

	0	1	2
0	10	20	30
1	40	50	60

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

	0	1
0	10	20
1	30	40
2	50	60

```
int a[ ][2]={0,0,0,0};
```

	0	1
0	0	0
1	0	0

1. Run Time Array Initialization: If the values are not known by programmers in advance then the programmer makes the use of run time initialization. i.e. programmer reads values from users through the keyboard by using input functions like scanf() getch() and gets(),etc.

Examples:

1. To read and display integer values for m x n matrix by using 2D array.

```
/* To read and display values for (m x n) matrix*/
#include<stdio.h>
#include<conio.h>
void main()
{
int a[25][25],m,n,i,j;
clrscr();
printf("Enter the order of matrix A:");
scanf("%d%d",&m,&n);
printf("\n Enter %d values in row order: ",m*n);
for(i=0;i<m;i++)
{
      for(j=0;j<n;j++)
      {
      scanf("%d",&a[i][j]);
      }
}
printf("\n Array Elements \n"
for(i=0;i<m;i++)
{
      for(j=0;j<n;j++)
      {
      printf("\t%d",a[i][j]);
      }
      printf("\n");
}
getch();
}
```

```
o/p:

Enter the order of matrix A: 2 3

Enter 6 values in row order: 1 2 3 4 5 6

Array Elements

      1     2     3

      4     5     6
```

Assignments:

1. Write a c program to find addition of two matrices A and B of the same size.
2. Write a c program to find multiplication two matrices A(m,n) and B(p,q).

```
/* Addition of two matrices*/
#include<stdio.h>
#include<conio.h>
void main()
{
int a[25][25],b[25][25],c[25][25],m,n,i,j;
clrscr();
printf("Enter the order of both the matrix A and B:");
scanf("%d%d",&m,&n);
printf("\nEnter %d values of A matrix: ",m*n);
for(i=0;i<m;i++)
{
     for(j=0;j<n;j++)
     {
     scanf("%d",&a[i][j]);
     }
}
printf("\nEnter %d values of B matrix: ",m*n);
for(i=0;i<m;i++)
{
     for(j=0;j<n;j++)
     {
     scanf("%d",&b[i][j]);
     }
}
/* code to find addition of matrix*/

for(i=0;i<m;i++)
{
     for(j=0;j<n;j++)
     {
     c[i][j]=a[i][j]+b[i][j];
     }
}

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

```
o/p:
Enter the order of both matrix A & B : 2 3
Enter 6 values of A matrix : 1 2 3 4 5 6
Enter 6 values of B matrix : 1 2 3 4 5 6
Resultant Matrix
     2     4     6
     8     10    12
```

```
/* Multiplication of two matrices*/
#include<stdio.h>
#include<conio.h>
void main()
{
int a[25][25],b[25][25],c[25][25],m,n,p,q,i,j,k;
clrscr();
printf("Enter the order of matrix A:");
scanf("%d%d",&m,&n);
printf("Enter the order of matrix B:");
scanf("%d%d",&p,&q);
if (n==p)
{
printf("\n Multiplication of A and B is possible\n");
printf("\nEnter values of A matrix: ");
for(i=0;i<m;i++)
{
     for(j=0;j<n;j++)
     {
     scanf("%d",&a[i][j]);
     }
}
printf("\nEnter values of B matrix: ");
for(i=0;i<p;i++)
{
     for(j=0;j<q;j++)
     {
     scanf("%d",&b[i][j]);
     }
}
/* code to find multiplication of matrix*/
for(i=0;i<m;i++)
{
     for(j=0;j<q;j++)
     {
     c[i][j]=0;
          for(k=0;k<n;k++)
          c[i][j]=c[i][j]+a[i][k]*b[k][j];
     }
}
printf("\n Resultant Matrix \n");
for(i=0;i<m;i++)
{
     for(j=0;j<q;j++)
     {
     printf("\t%d",c[i][j]);
     }
     printf("\n");
}
}
else
     printf("\n Multiplication is not possible!");
getch();
}
```

```
o/p:
Enter the order matrix A & B : 2 3
Enter the order matrix A & B : 3 2
Multiplication is possible
Enter values of A matrix : 1 1 1 1 1 1
Enter values of B matrix : 1 1 1 1 1 1
Resultant Matrix
        3       3       3
        3       3       3

o/p:
Enter the order matrix A & B : 2 3
Enter the order matrix A & B : 2 3
Multiplication is not possible
```

Introduction to Strings

In order to design and develop user friendly software, the programmer's need to make the use of strings.

A string can be defined as a sequence of enclosed in double quotes and that ends with the NULL value (\0) by default.

Programmers will make the use of input and output functions like scanf(), printf(), gets() and puts() functions to read and write strings.

Examples for strings:

1) "Brain W. Kernighan and Dennis M. Ritchie"
2) "Computer Programming in C"
3) "VTU"

V	T	U	\0

4) "A"
5) "2015" and
6) " " etc.

Declaration and Initialization of string variables:

The string variable(s) must be defined in the declaration part of main() function before they are used in executable part by using following syntax.

Syntax:

char string_variable[size];

string_variable: Valid identifier

Example:

1) String varibales to store student name , city and USN.

```
char name[25]="Thomas";
```

	0	1	2	3	4	5	6		24
Name→	T	h	o	m	a	s	\0		\0

```
char city[25]="Vijayapur";
```

Name→	V	i	j	a	y	a	p	u	r	\0

```
char USN[25]="2BL18CV001";
```

Name→	2	B	L	1	8	C	V	0	0	1	\0

Reading and Writing Strings (Input / Output Functions used with Strings):

The strings can be read and write by using following I/O functions.

1. scanf() and printf()
2. gets() and puts()

The scanf( ) reads a string until white space found from a keyboard, whereas gets( ) reads a string until user press ENTER KEY.

The printf( ) helps to write (display) strings on the monitor screen in different formats with appropriate message, whereas puts () helps to display only the string on monitor screen without format.

```
/* To read and siplay strings by using scanf() and printf() */
#include<stdio.h>
void main()
{
char name[5][25];
clrscr();
printf("Enter your name: ");
scanf("%s",name);
printf("Hello! %s",name);
getch();
}
```

```
o/p:
Enter your name: Alice Bob
Hello ! Alice
```

```
/* To read and siplay strings by using gets() and puts() */
#include<stdio.h>
void main()
{
char name[25];
clrscr();
printf("Enter your name");
gets(name);
printf("Hello! ");
puts(name);
getch();
}
```

```
o/p:
Enter your name: Alice Bob
Hello ! Alice Bob
```

```
/* Program to find length of a string*/
#include<stdio.h>
void main()
{
char str[25];
int i,counter=0;
clrscr();
printf("Enter a string");
gets(str);
for(i=0;str[i]!='\0';i++)
{
counter++;
}
printf("\n Length of a string is %d",counter);
getch();
}
```

```
o/p:
Enter a string: Alice
Length of a string is 5
```

```
/* Program to copy one string1 into string2*/
#include<stdio.h>
void main()
{
char str1[25],str2[25];
int i;
clrscr();
printf("Enter string1 to be copy:
gets(str1);
for(i=0;str1[i]!='\0';i++)
{
str2[i]=str1[i];
}
str2[i]='\0';
printf("\nString Copied\n");
printf("\nString1=%s  String2=%s",str1,str2);
getch();
}
```

```
o/p:
Enter string1 to be copy: Alice
String Copied
String1= Alice String2=Alice
```

```
/* To reverse a string*/
#include<stdio.h>
#include<string.h>
void main()
{
char str[25];
int i,len,temp;
clrscr();
printf("Enter string to be reverse: ");
gets(str);
len=strlen(str)-1;
for(i=0;i<strlen(str)/2;i++)
{
temp=str[i];
str[i]=str[len];
str[len--]=temp;
}
printf("\n The reversed string is %s",str);
getch();
}
```

```
o/p:
Enter string to be reverse: tech
The reversed string is hcet
```

String Manipulating Functions:

C supports different string handling functions to perform different operations on strings. All the string handling functions are stored in *string.h* file. Some of the most common used string functions are as follow.

Sl.No.	Function	Description
1	strlen()	Returns number of characters in the given string
2	strcpy()	Copies the given source string to another destination string variable
3	strncpy()	Copies first n characters from source to destination
4	strcmp()	Compares two strings for their similarity
5	strncmp()	Compares first n characters from two strings for their similarity
6	strcat()	Concatenates (joins) two strings into single string
7	strncat()	Concatenates first n characters from source string to destination
8	strupr()	Converts characters into uppercase
9	strlwr()	Converts characters into lowercase
10	strrev()	Reverses a given string

1. strlen(): It returns the number of characters in the given string.

Syntax:

strlen(string);

Examples:

```
int len;
len=strlen("vtu");
∴ len=3
```

```
int len;
char str[]="vtu";
len=strlen(str);
∴ len=3
```

Programming Example:

```
/* To find length of a string */
#include<stdio.h>
#include<string.h>
void main()
{
char str[50];
int len;
clrscr();
printf("Enter a string");
gets(str);
len=strlen(str);
printf("\n Length of the string is %d",len);
getch();
}
```

```
o/p:
Enter a string : vtu
Length of the string is 3
```

2. strcpy(): It copies the contents of source string to another destination string variable.

Syntax:

strcpy(destination, source string);

Examples:
char str1[25]=”vtu”,str2[25];
strcpy(str2,str1);
printf(“copied string is %s”,str2);
o/p: Copied string is vtu

Programming Example:

```
/* To copy a string */
#include<stdio.h>
#include<string.h>
void main()
{
char str1[50],str2[50];
clrscr();
printf("Enter string1 to be copy: ");
gets(str1);
strcpy(str2,str1);
printf("\n Copied string is %s",str2);
getch();
}
```

```
o/p:
Enter string1 to be copy : vtu
Copied string is vtu
```

3. strncpy(): It copies the number of characters from source string to another destination string variable.

Syntax:

strncpy(destination, source string, n);

where,

n → number of characters to be copy

Examples:

char str1[25]=”Technology”,str2[25];
strncpy(str2,str1,4);
printf(“copied string is %s”,str2);
o/p: Copied string is Tech

4. strcmp(): It compares given two strings for their similarity character by character. If, the given strings are same then it returns zero; otherwise, non-zero (ASCII difference of dissimilar characters).

Syntax:

strcmp(string1,string2);

Examples:

char str1[25]=”VTU”,str2[25]=”VTU”;
int temp;
temp=strcmp(str1,str2);

∴ temp=0 /* because both are same */

```
char str1[25]="Ant",str2[25]="Cat";
int temp;
temp=strcmp(str1,str2);
```
∴ temp=-2 /* ASCII of 'A' – ASCII of 'C' i.e. 65-67=-2 */

5. strncmp(): It compares given two strings for their similarity for the specified number of characters from the beginning. If, the given strings are same then it returns zero; otherwise, non-zero (ASCII difference of dissimilar characters).

Syntax:

strncmp(string1,string2,n);

where,

n→ number of characters to be compare.

Examples:

```
char str1[25]="THERE",str2[25]="THEIR";
int temp;
temp=strncmp(str1,str2,3);
```
∴ temp=0 /* because first 3 characters are same */

6. strcat(): It concatenates (joins) two strings into a single string.

Syntax:

strcat(string1,string2);

Example:

```
char str1[25]="Honey", str2[25]="well", str3[25];
strcpy(str3,strcat(str1,str2));
printf("%s",str3);
```
o/p: Honeywell

7. strncat(): It concatenates (joins) only specified number of characters from source to destination string.

Syntax:

strncat(string1,string2,n);

where,

n→ number of characters to be joined.

Example:

```
char str1[25]="Honey", str2[25]="well", str3[25];
strcpy(str3,strcat(str1,str2,2));
printf("%s",str3);
```
o/p: Honeywe

8. strrev(): It reverses a string.

Syntax:

strrev(string);

Example:

```
char str1[25]=”computer”;
strrev(str1);
printf(“%s”,str1);
o/p: retupmoc
```

```
/* check string for palindrome */
#include<stdio.h>
#include<string.h>
void main()
{
char str1[25],str2[25];
clrscr();
printf("Enter a string:");
gets(str1);
strcpy(str2,str1);
strrev(str1);
if(strcmp(str1,str2)==0)
printf("\n Palindrome");
else
printf("\n Not palindrome");
getch();
}
```

```
o/p:
Enter a string : madam
Palindrome
o/p:
Enter a string : teacher
Not Palindrome
```

9. strupr(): It converts the characters of a given string into uppercase.

Syntax:

strupr(string);

Example:

```
char str1[25]=”computer”;
strupr(str1);
printf(“%s”,str1);
o/p: COMPUTER
```

10. strlwr(): It converts the characters of a given string into lowercase.

Syntax:

strlwr(string);

Example:

```
char str1[25]=”COMPUTER”;
strlwr(str1);
printf(“%s”,str1);
o/p: computer
```

***

…