UNIT-1

ARRAYS

INTRODUCTION

An array is a collection of similar data elements.

These data elements have the same data type.

The elements of the array are stored in consecutive memory locations and are referenced by an index (also known as the subscript).

Declaring an array means specifying three things:

The data type- what kind of values it can store ex, int, char, float

Name- to identify the array

The size- the maximum number of values that the array can hold

INTRODUCTION

Arrays are declared using the following syntax.

```
type name[size];
int marks[10];
```

1st element	2nd element	3rd element	4th element	5th element	6th element	7th element	8th element	9th element	10th element
marks[0]	marks[1]	marks[2]	marks[3]	marks[4]	marks[5]	marks[6]	marks[7]	marks[8]	marks[9]

TYPES

1. ONE DIMENSIONAL ARRAY(1D)
2. TWO DIMENSIONAL ARRAY(2D)
3. MULTI DIMENSIONAL ARRAY

1D ARRAY-ACCESSING ELEMENTS OF THE ARRAY

To access all the elements of the array, you need to use a loop. That is, we can access all the elements of the array by varying the value of the subscript into the array. But note that the subscript must be an integral value or an expression that evaluates to an integral value.

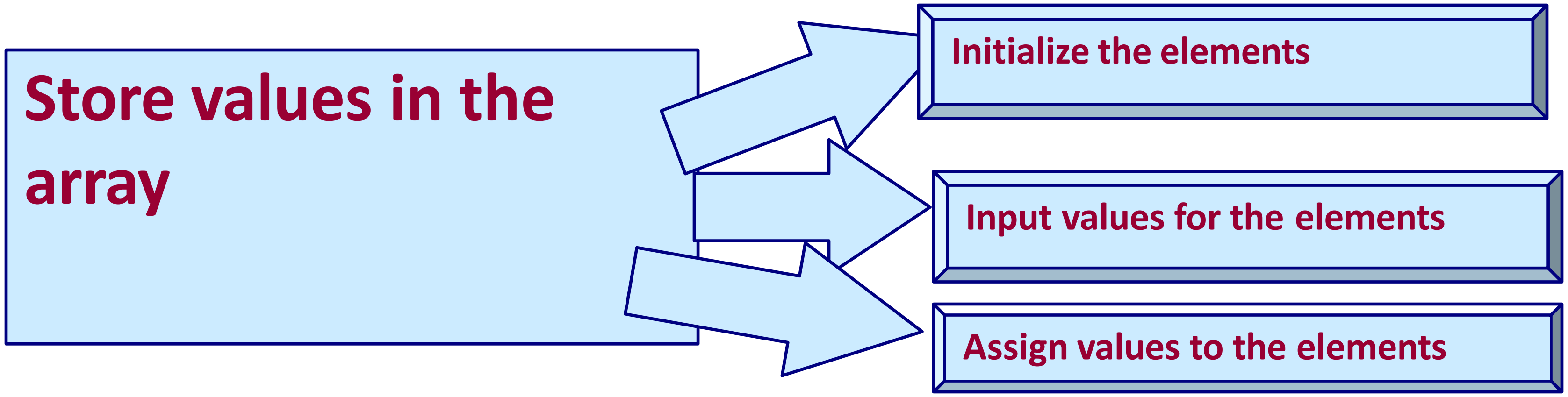
1-D ARRAY STORING VALES IN ARRAYS

Initialization of Arrays

Arrays are initialized by writing,

type array_name[size]={list of values};

int marks[5]={90, 82, 78, 95, 88};


Store values in the array
Initialize the elements
Input values for the elements
Assign values to the elements

Initialize the elements

```
#include<stdio.h>
#include<conio.h>
void main()
{
		int a[5]={10,20,30,40,50},i;
		clrscr();
		for(i=0;i<5;i++)
		{
			printf("\n%d",a[i]);
		}
		getch();
}
```

1-D ARRAY INPUTING VALUES

```
int i, marks[10];
for(i=0;i<10;i++)
{
scanf("%d", &marks[i]);
}
```

1-D ARRAY INPUTING VALUES

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

void main()
{
        int a[5],i;
        clrscr();
        for(i=0;i<5;i++)
        {
                printf("\n Enter Value %d : ",i+1);
                scanf("%d",&a[i]);
        }
        printf("\n Entered Values are as follows \n");

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

ASSIGNING VALUES

```
int i, arr1[10], arr2[10];
for(i=0;i<10;i++)
{
      arr2[i] = arr1[i];
}
```

ASSIGNING VALUES

```
int i, arr1[10], arr2[10];
arr2[0]=10;
arr2[1]=20;
arr2[2]=30;
arr2[3]=40;
arr2[4]=50;
.
.
```

ASSIGNING VALUES

```
#include<stdio.h>
#include<conio.h>
void main()
{
		int a[5],b[5],i;
		clrscr();
		a[0]=10;
		a[1]=20;
		a[2]=30;
		a[3]=40;
		a[4]=50;
		for(i=0;i<5;i++)
		{
				b[i]=a[i];
		}
		for(i=0;i<5;i++)
		{
				printf("\n%d",a[i]);
		}
		getch();
}
```

CALCULATING THE LENGTH OF THE ARRAY

Length = upper_bound – lower_bound + 1

Where, upper_bound is the index of the last element and lower_bound is the index of the first element in the array

99	67	78	56	88	90	34	85
Marks[0]	marks[1]	marks[2]	marks[3]	marks[4]	marks[5]	marks[6	marks[7]]

Here, lower_bound = 0, upper_bound = 7

Therefore, length = 7 – 0 + 1 = 8

Advantage of C Array

1) Code Optimization: Less code to the access the data.

2) Ease of traversing: By using the for loop, we can retrieve the elements of an array easily.

3) Ease of sorting: To sort the elements of the array, we need a few lines of code only.

4) Random Access: We can access any element randomly using the array.

Disadvantage of C Array

Fixed Size: Whatever size, we define at the time of declaration of the array, we can't exceed the limit. So, it doesn't grow the size dynamically.

WRITE A PROGRAM TO READ AND DISPLAY N NUMBERS USING ONE DIMENSIONAL ARRAY

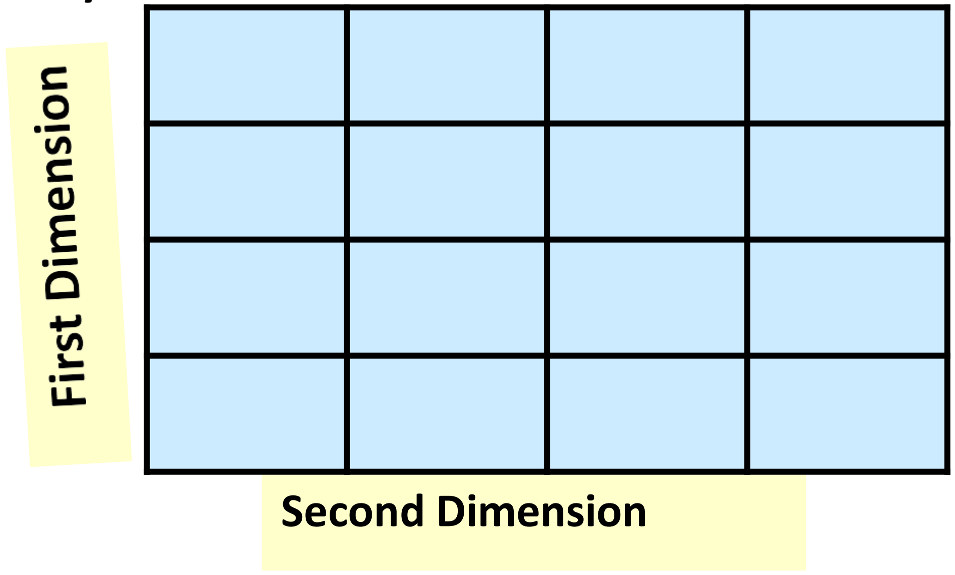
```
#include<stdio.h>
#include<conio.h>
void main()
{
        int i=0, n, arr[20];
        clrscr();
        printf(“\n Enter the number of elements : “);
        scanf(“%d”, &n);

        for(i=0;i<n;i++)
        {
                printf(“\n Arr[%d] = “, i);
                scanf(“%d”,&num[i]);
        }
        printf(“\n The array elements are “);
        for(i=0;i<n;i++)
                printf(“Arr[%d] = %d\t”, i, arr[i]);
        getch();
}
```

TWO DIMENSIONAL ARRAYS

A two dimensional array is specified using two subscripts where one subscript denotes row and the other denotes column.

C looks a two dimensional array as an array of a one dimensional array.



TWO DIMENSIONAL ARRAYS

A two dimensional array is declared as:

data_type array_name[row_size][column_size];

Therefore, a two dimensional mXn array is an array that contains m*n data elements and each element is accessed using two subscripts, i and j where i<=m and j<=n

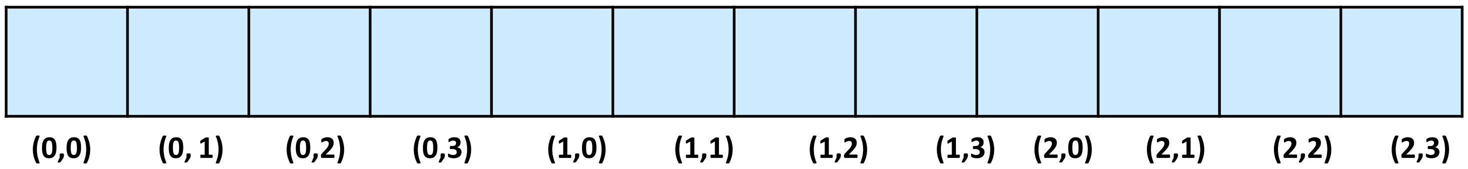
TWO DIMENSIONAL ARRAYS

int marks[3][5]

Rows/Columns	Col 0	Col 1	Col2	Col 3	Col 4
Row 0	Marks[0][0]	Marks[0][1]	Marks[0][2]	Marks[0][3]	Marks[0][4]
Row 1	Marks[1][0]	Marks[1][1]	Marks[1][2]	Marks[1][3]	Marks[1][4]
Row 2	Marks[2][0]	Marks[2][1]	Marks[2][2]	Marks[2][3]	Marks[2][4]

MEMORY REPRESENTATION OF A TWO DIMENSIONAL ARRAY

In the row major order the elements of the first row are stored before the elements of the second and third row. That is, the elements of the array are stored row by row where n elements of the first row will occupy the first nth locations.



Example of 2-D Array

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a[5][5],b[5][5],c[5][5],i,j;
        clrscr();
        printf("Enter the values of matrix A\n\n");
        for(i=0;i<3;i++)
        {
                for(j=0;j<3;j++)
                {
                        scanf("%d",&a[i][j]);
                }
                printf("\n");
        }
        printf("Enter the values of matrix B\n\n");
        for(i=0;i<3;i++)
        {
                for(j=0;j<3;j++)
                {
                        scanf("%d",&b[i][j]);
                }
                printf("\n");
        }
```

Example of 2-D Array

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

Example of 2-D Array

```
        printf("\nMatrix B\n\n");
        for(i=0;i<3;i++)
        {
                for(j=0;j<3;j++)
                {
                        printf("%d\t",b[i][j]);
                }
                printf("\n");
        }
        printf("\nMatrix C\n\n");
        for(i=0;i<3;i++)
        {
                for(j=0;j<3;j++)
                {
                        printf("%d\t",c[i][j]);
                }
                printf("\n");
        }
        getch();
}
```

MULTI DIMENSIONAL ARRAYS

A multi dimensional array is an array of arrays.

Like we have one index in a single dimensional array, two indices in a two dimensional array, in the same way we have n indices in a n-dimensional array or multi dimensional array.

Conversely, an n dimensional array is specified using n indices.

An n dimensional m1 x m2 x m3 x ….. mn array is a collection m1*m2*m3* ….. *mn elements.

PROGRAM TO READ AND DISPLAY A 2X2X2 ARRAY

```
#include<stdio.h>
void main()
{                       int array1[3][3][3], i, j, k;
                        clrscr();
                        printf("\n Enter the elements of the matrix");
                        printf("\n ******************************");
                        for(i=0;i<=2;i++)
                        {   for(j=0;j<=2;j++)
                            {
                                  for(k=0;k<=2;k++)
                                  {
                                            printf("\n array[%d][ %d][ %d] = ", i, j, k);
                                            scanf("%d", &array1[i][j][k]);
                                  }
                            }
                        }
                        for(i=0;i<=2;i++)
                        {         printf("\n\n");
                                  for(j=0;j<=2;j++)
                                  {
                                            printf("\n");
                                            for(k=0;k<=2;k++)
                                            printf("\t   array[%d][   %d][   %d]   =   %d",   i,   j,   k,
array1[i][j][k]);
                                  }
                        }
                        getch();
}
```

<u>STRINGS</u>

INTRODUCTION

A string is a null-terminated character array. This means that after the last character, a null character (‘\0’) is stored to signify the end of the character array.

The general form of declaring a string is

```
char str[size];
```

For example if we write,

```
char str[] = “HELLO”;
```

INTRODUCTION

We are declaring a character array with 5 characters namely, H, E, L, L and O. Besides, a null character (‘\0’) is stored at the end of the string. So, the internal representation of the string becomes- HELLO’\0’. Note that to store a string of length 5, we need 5 + 1 locations (1 extra for the null character).

The name of the character array (or the string) is a pointer to the beginning of the string.

INTRODUCTION

str[0]	1000	H
str[1]	1001	E
str[2]	1002	L
str[3]	1003	L
str[4]	1004	O
str[5]	1005	\0

READING STRINGS

If we declare a string by writing

```
char str[100];
```

Then str can be read from the user by using two ways

1.use scanf function

2.using gets() function

1. The string can be read using scanf() by writing

```
scanf(“%s”, str);
```

READING STRINGS

2. The string can be read by writing

```
gets(str);
```

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

WRITING STRINGS

The string can be displayed on screen using two ways

1.use printf() function

2.using puts() function

1.The string can be displayed using printf() by writing

```
printf(“%s”, str);
```

2. Using puts()

```
puts(str);
```

USING SCANSET

The ANSI standard added the new **scanset** feature to the C language. A scanset is used to define a set of characters which may be read and assigned to the corresponding string. A scanset is defined by placing the characters inside square brackets prefixed with a %

```
void main()
{
  char str[10];
  printf("\n Enter string: " );
  scanf("%[aeiou]", str );
  printf( "The string is : %s", str);
  getch();
}
```

The code will stop accepting character as soon as the user will enter a character that is not a vowel.

LENGTH

The number of characters in the string constitutes the length of the string.

For example, LENGTH(“C PROGRAMMING IS FUN”) will return 20. Note that even blank spaces are counted as characters in the string.

LENGTH

ALGORITHM TO CALCULATE THE LENGTH OF A STRING

```
Step 1: [INITIALIZE] SET I = 0
Step 2: Repeat Step 3 while STR[I] != '\0'
Step 3:           SET I = I + 1
      [END OF LOOP]
Step 4:     SET LENGTH = I
Step 5: END
```

LENGTH

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

void main()
{
        char str[100];
        int i=0,length;
        clrscr();
        printf("\n Enter The String : ");
        fflush(stdin);
        gets(str);
```

LENGTH

```
        while(str[i]!='\0')
        {
                i++;
        }
        length=i;
        printf("\n The length of the String is : %d",length);
        getch();
}
```

CONVERTING CHARACTERS OF A STRING INTO UPPER CASE

In memory the ASCII code of a character is stored instead of its real value. The ASCII code for A-Z varies from 65 to 90 and the ASCII code for a-z ranges from 97 to 122. So if we have to convert a lower case character into upper case, then we just need to subtract 32 from the ASCII value of the character.

CONVERTING CHARACTERS OF A STRING INTO UPPER CASE

```
ALGORITHM TO CONVERT THE CHARACTERS OF STRING
INTO UPPER CASE

Step1: [Initialize] SET I=0
Step 2: Repeat Step 3 while STR[I] != '\0'
Step 3:       IF STR[I] > 'a' AND STR[I] < 'z'
                   SET Upperstr[I] = STR[I] - 32
              ELSE
                   SET Upperstr[I] = STR[I]
                [END OF IF]
         [END OF LOOP]
Step 4: SET Upperstr[I] = '\0'
Step 5: EXIT
```

CONVERTING CHARACTERS OF A STRING INTO UPPER CASE

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char str[100],upper_str[100];
        int i=0;
        clrscr();
        printf("\n Enter String : ");
        gets(str);
```

CONVERTING CHARACTERS OF A STRING INTO UPPER CASE

```
        while(str[i]!='\0')
        {
                if(str[i]>='a' && str[i]<='z')
                        upper_str[i]=str[i]-32;
                else
                        upper_str[i]=str[i];
                i++;
        }
        upper_str[i]='\0';
        printf("\n String Converted int Upper Case is : ");
        puts(upper_str);
        getch();
}
```

CONVERTING CHARACTERS OF A STRING INTO LOWER CASE

In memory the ASCII code of a character is stored instead of its real value. The ASCII code for A-Z varies from 65 to 90 and the ASCII code for a-z ranges from 97 to 122. So if we have to convert a upper case character into lower case, then we just need to ADD 32 from the ASCII value of the character.

CONVERTING CHARACTERS OF A STRING INTO LOWER CASE

```
ALGORITHM TO CONVERT THE CHARACTERS OF STRING
INTO LOWER CASE

Step1: [Initialize] SET I=0
Step 2: Repeat Step 3 while STR[I] != '\0'
Step 3:       IF STR[1] > 'A' AND STR[I] < 'Z'
                    SET lower[I] = STR[I] + 32
              ELSE
                    SET lower[I] = STR[I]
                [END OF IF]
       [END OF LOOP]
Step 4: SET lower[I] = '\0'
Step 5: EXIT
```

CONVERTING CHARACTERS OF A STRING INTO LOWER CASE

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char str[100],lwr_str[100];
        int i=0;
        clrscr();
        printf("\nEnter string:");
        gets(str);
        while(str[i]!='\0')
```

CONVERTING CHARACTERS OF A STRING INTO LOWER CASE

```
    while(str[i]!='\0')
    {
            if(str[i]>='A' && str[i]<='Z')
            {
                    lwr_str[i]=str[i]+32;
            }
            else
            {
                    lwr_str[i]=str[i];
            }
            i++;
    }
    lwr_str[i]='\0';
    printf("\n\nString converted into lowercase is:");
    puts(lwr_str);
    getch();
}
```

CONCATENATING TWO STRINGS TO FORM A NEW STRING

IF S1 and S2 are two strings, then concatenation operation produces a string which contains characters of S1 followed by the characters of S2.

CONCATENATING TWO STRINGS TO FORM A NEW STRING

```
ALGORITHM TO CONCATENATE TWO STRINGS

1. Initialize I =0 and J=0
2. Repeat step 3 to 4 while I <= LENGTH(str1)
3       SET new_str[J] = str1[I]
4       Set I =I+1 and J=J+1
        [END of step2]
 5. SET I=0
6  Repeat step 6 to 7 while I <= LENGTH(str2)
7       SET new_str[J] = str1[I]
8       Set I =I+1 and J=J+1
        [END of step5]
 9.  SET new_str[J] = '\0'
 10. EXIT
```

CONCATENATING TWO STRINGS TO FORM A NEW STRING

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

void main()
{
        char str1[100],str2[100],str3[100];
        int i=0,j=0;
        clrscr();
        printf("\n Enter First String : ");
        gets(str1);
        printf("\n Enter Second String : ");
        gets(str2);
```

CONCATENATING TWO STRINGS TO FORM A NEW STRING

```
while(str1[i]!='\0')
{
        str3[j]=str1[i];
        i++;
        j++;
}
i=0;
while(str2[i]!='\0')
{
        str3[j]=str2[i];
        i++;
        j++;
}
```

CONCATENATING TWO STRINGS TO FORM A NEW STRING

```
        str3[j]='\0';
        printf("\n The Concatenated String is : ");
        puts(str3);
        getch();

}
```

ARITHMETIC OPERATION ON STRING

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

void main()
{
  char s[] = "Hello";
  clrscr();

  printf("%s\n", s);
  printf("%s",(s+1));

  getch();
}
```

getch()

During the program execution, a single character is get or read through the **getch()**. The given value is not displayed on the screen and the compiler does not wait for another character to be typed.And then,the given character is printed through the **printf** function.

```
void main()
{
char ch;
ch = getch();
printf("Input Char Is :%c",ch);
}
```

getche()

During the program execution, a single character is get or read through the **getche()**. The given value is displayed on the screen and the compiler does not wait for another character to be typed. Then,after wards the character is printed through the **printf** function.

```
void main()
{
char ch;
ch = getche();
printf("Input Char Is :%c",ch);
}
```

getchar()

During the program execution, a single character is get or read through the **getchar()**. The given value is displayed on the screen and the compiler wait for another character to be typed. If you press the enter key then only the given character is printed through the **printf** function.

```
void main()
{
char ch;
ch = getchar();
printf("Input Char Is :%c",ch);
}
```

ARRAY OF STRINGS

Now suppose that there are 20 students in a class and we need a string that stores names of all the 20 students. How can this be done? Here, we need a string of strings or an array of strings. Such an array of strings would store 20 individual strings. An array of string is declared as,

```
char names[20][30];
```

ARRAY OF STRINGS

Here, the first index will specify how many strings are needed and the second index specifies the length of every individual string. So here, we allocate space for 20 names where each name can be maximum 30 characters long.

Let us see the memory representation of an array of strings. If we have an array declared as,

char name[5][10] = {“Ram”, “Mohan”, “Shyam”, “Hari”, “Gopal”};

ARRAY OF STRINGS

Name[0]	R	A	M	‘\0’						
Name[1]	M	O	H	A	N	‘\0’				
Name[2]	S	H	Y	A	M	‘\0’				
Name[3]	H	A	R	I	‘\0’					
Name[4]	G	O	P	A	L	‘\0’				

STRING MANIPULATION FUNCTIONS

Function	Usage
Char * strcat(char *str1, char *str2);	This function appends the string pointed to by str2 to the end of the string pointed to by str1.
Char *strncat(char *str1, const char *str2,size_t n);	This function appends strings pointed to by str2 to end of string pointed to by str1 up to n characters long.
Char *strchr(const char *str,int c);	This function searches for the first occurrence of character c in string pointed to by the argument str.
Int strcmp(const char *str1,const char *str2)	This function compares string pointed to by str1 to the string pointed to by str2. function returns zero if strings are equals.Otherwise, it returns value less than zero or greater than zero if str1 is less than or greater than str2 respectively.

STRING MANIPULATION FUNCTIONS

Function	Usage
Char *strcpy(char *str1,const char *str2);	This function copies string pointed to by str2 to str1 including null character of str2.
Char strncpy(char *st r1,const char *str2,size_t n);	This function copies up to n characters from the string pointed to by str2 to str1. copying stops when n characters are copied.
Size_t strlen(const char *str)	This function calculates the length of the string str up to but not including the null character.

STRING MANIPULATION FUNCTIONS

Function	Usage
Char *strupr(char *str1)	This function convert String str1 in upper case
Char *strlwr(char *str1)	This function convert string str1 in lower case.
Int strcmpi(const char *str1,const char *str2)	This function compares string pointed to by str1 to the string pointed to by str2. but it ignores the case. function returns zero if strings are equals.Otherwise, it returns value less than zero or greater than zero if str1 is less than or greater than str2 respectively.
Char strrev(char *str1)	This function convert string str1 in reverse order.

STRLEN

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        int l;
        char str[25];
        clrscr();
        printf("\nEnter a string:");
        gets(str);
        l=strlen(str);
        printf("\n length of the string is %d",l);
        getch();
}
```

STRCAT

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        int l;
        char str[25],str2[25];
        clrscr();
        printf("\nEnter string 1:");
        gets(str);
        printf("\nEnter string 2:");
        gets(str2);
        printf("\nConcatenated string= “,strcat(str,str2);
        getch();
}
```

STRCHR

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

void main()
{
		char str[20],ch;
		clrscr();
		printf("\n Enter String : ");
		gets(str);
		printf("\n Enter Character for Search : ");
		scanf("%c",&ch);

		printf("\n Search Character is %s",strchr(str,ch));

		getch();
}
```

STRNCAT

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
		int n;
		char str[25],str2[25];
		clrscr();
		printf("\nEnter string 1:");
		gets(str);
		printf("\nEnter string 2:");
		gets(str2);
		printf("\nEnter number of characters:");
		scanf("%d",&n);
		printf("\nConcatenated string=%s", strncat(str,str2,n));
		getch();
}
```

STRCMP

```
include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
		int n;
		char str[25],str2[25];
		clrscr();
		printf("\nEnter string 1:");
		gets(str);
		printf("\nEnter string 2:");
		gets(str2);
		if(strcmp(str,str2)==0)
				printf("\nBoth strings are equal");
		else
				printf("\nBoth strings are not equal");
		getch();
}
```

STRCMPI

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        int n;
        char str[25],str2[25];
        clrscr();
        printf("\nEnter string 1:");
        gets(str);
        printf("\nEnter string 2:");
        gets(str2);
        if(strcmpi(str,str2)==0)
                printf("\nBoth strings are equal");
        else
                printf("\nBoth strings are  not equal");
        getch();
}
```

STRCPY

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
		int n;
		char str[25],str2[25];
		clrscr();
		printf("\nEnter string 1:");
		gets(str);
		printf("\nEnter string 2:");
		gets(str2);
		strcpy(str,str2);
		printf("\nCopied string=%s", str);
		getch();
}
```

STRNCPY

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        int n;
        char str[25],str2[25];
        clrscr();
        printf("\nEnter string 1:");
        gets(str);
        printf("\nEnter string 2:");
        gets(str2);
        printf("\nEnter number of characters to be copied:");
        scanf("%d",&n);
        strncpy(str,str2,n);
        printf("\nCopied string=%s", str);
        getch();
}
```

STRREV

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
		int n;
		char str[25],str2[25];
		clrscr();
		printf("\nEnter string:");
		gets(str);
		printf("\nReversed string is=%s",strrev(str));
		getch();
}
```

STRUPR

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        int l;
        char str[25];
        clrscr();
        printf("\nEnter a string:");
        gets(str);
        printf("\n uppercase string= %s",strupr(str));
        getch();
}
```

STRLWR

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
		int l;
		char str[25];
		clrscr();
		printf("\nEnter a string:");
		gets(str);
		printf("\n lowercase string= %s",strlwr(str));
		getch();
}
```

Thank You