Derived Data Types :

Data types that are derived from fundamental data types are called derived data types. Some of the properties of existing primitive data types are added.

Derived data types do not create new data types. Instead, they add some functionality to the existing data types.

Derived data types are derived from the primitive data types by adding some extra relationships with the various elements of the primitive data types. The derived data type can be used to represent a single value or multiple value.

Examples :

1. Arrays
2. Pointers
3. Functions
4. Files

Arrays :

Array is derived data types : array is an indexed collection of homogeneous elements stored in contiguous memory locations.

- Arrays are derived data types in C.
- Using single variable , we can store 'n' number of elements (values).
- Array elements are accessed using index(indics).
- Array index always starts from 0 to n-1.
- All array elements must be homogeneous (Same Data types).

Array Declaration :

Syntax :

```
datatype arrayname[size/bounds];
```

datatype: it must be valid data type related keyword which represents type of elements we store in an array.

Arrayname: It must be a valid identifier: it indicates the name of the array variable.

Size: It must be integer number that specifies the number of elements that array can hold. It represents the capacity of the array.

[]-Subcript operator/Array Operator.

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.

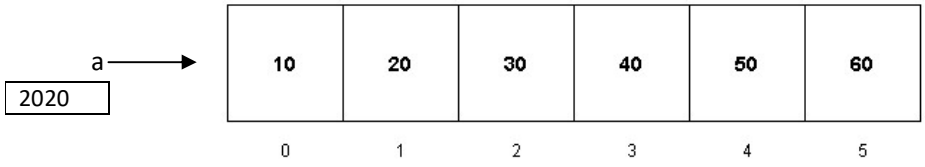
Disadvantage of C Array

1) 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 like LinkedList which we will learn later.

ex 1 :

int a[6];

“a is an array of 6 integers”



Index->	0	1	2	3	4	5
Element->	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
Address→	2010	2014	2018	2022	2026	2023
Address access-.>	&a[0]	&a[1]	&a[2]	&a[3]	&a[4]	&a[5]

Initialization of C Array

The simplest way to initialize an array is by using the index of each element. We can initialize each element of the array by using the index. Consider the following example.

```
marks[0]=80;//initialization of array
marks[1]=60;
marks[2]=70;
marks[3]=85;
marks[4]=75;
```

```
#include<stdio.h>
int main()
{
   int i=0;
   int marks[5];//declaration of array

   marks[0]=80;//initialization of array
   marks[1]=60;
   marks[2]=70;
   marks[3]=85;
   marks[4]=75;

   //traversal of array
   for(i=0;i<5;i++)
   {
      printf("%d \n",marks[i]);
   }//end of for loop

   return 0;
}
```

Output :

```
80
60
70
85
75
```

Using for loop with printf() & scanf().

-Index starts from 0 to n-1 hence we can iterate indices through loops.
ex :

```
int a[10];
int i;
for(i=0; i<10; i++)
{
     printf("\nEnter a Element : ");
      scanf("%d",&n);
}
```

Array Accessing :

- Array elements can be accessed using for loop and element wise;
- Using arrayname, subscript operator we can access array elements for processing and printing.

```
#include<stdio.h>
int main()
{
       int a[100],i,n;

      printf("\nEnter a Number of Elements : ");
      scanf("%d",&n);

      for(i=0; i<n; i++)
      {
         printf("\nEnter Element : ");
         scanf("%d",&a[i]);
       }

       printf("\nArray Elements \n");
       for(i=0; i<n; i++)
       {
          printf("%d\t",a[i]);
       }

        return 0;
}
```

Output :

```
Enter a Number of Elements : 8
Enter Element : 10

Enter Element : 20

Enter Element : 30

Enter Element : 40

Enter Element : 50

Enter Element : 60

Enter Element : 70

Enter Element : 80
 Array Elements
10     20     30     40     50     60     70     80
```

Classification of arrays:

Arrays are classified into 3 categories :

1. Single Dimensional Arrays or 1-Dimensional Arrays
2. Two –Dimensional Arrays.
3. Multi-Dimensional Arrays.

1. Single Dimensional Arrays :
   1-Dimensional array having only array size , collection of homogeneous elements stored in contiguous memory locations.

   Syntax :

   Datatype arrayname[arraysize];

   Ex : int a[5];
   Int -> type of elements to be stored in an array.
   A ->array name must be a valid identifier.
   [] -> subscribe operator.

   float salary[10];

   2.Two Dimensional Arrays :
   -two-dimensional array is array of arrays.
   -two-dimensional array is an array which contains data elements in matrix format.
   -element are stored in rows & columns format.

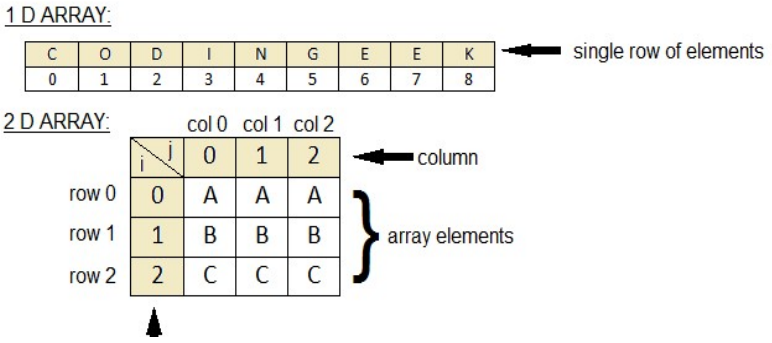
   Syntax :

   Datatype arrayname[rowsize][colsize];

   Ex :
   int a[2] [3];

   A is a 2-D array which contains 2 rows and 3 columns.
   Using below syntax, we can read and write (Access) array elements.

   An **array of arrays** is known as **2D array**. The two dimensional (2D) array in C programming is also known as matrix. A matrix can be represented as a table of rows and columns.



INITIALIZATION

- How to initialize a Two-Dimensional array?
  - Initialized directly in the declaration statement
    - int b[2][3] = {51, 52, 53, 54, 55, 56};
    - b[0][0] = 51 b[0][1] = 52 b[0][2] = 53
  - Use braces to separate rows in 2-D arrays.
    - int c[4][3] = {{1, 2, 3},
      {4, 5, 6},
      {7, 8, 9},
      {10, 11, 12}};
    - int c[ ][3] = {{1, 2, 3},
      {4, 5, 6},
      {7, 8, 9},
      {10, 11, 12}};

    Implicitly declares the number of rows to be 4.

An **array of arrays** is known as **2D array**. The two dimensional (2D) array in C programming is also known as matrix. A matrix can be represented as a table of rows and columns.

Simple Two dimensional(2D) Array Example

```
#include<stdio.h>
int main()
{
        int i=0,j=0;
        int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};

        //traversing 2D array
        for(i=0;i<4;i++)
        {
             for(j=0;j<3;j++)
             {
                 printf("arr[%d] [%d] = %d \n",i,j,arr[i][j]);
             }//end of j

         }//end of i

         return 0;
}
```

Output :

```
arr[0] [0] = 1
arr[0] [1] = 2
arr[0] [2] = 3
arr[1] [0] = 2
arr[1] [1] = 3
arr[1] [2] = 4
arr[2] [0] = 3
arr[2] [1] = 4
arr[2] [2] = 5
arr[3] [0] = 4
arr[3] [1] = 5
arr[3] [2] = 6
```

Practice Programs on Array :

1. Program to read matrix and display matrix.

```
#include<stdio.h>
int main()
{
      int row,col;
      printf("\nEnter Number of rows : ");
      scanf("%d", &row);

      printf("\nEnter Number of columns : ");
      scanf("%d", &col);

      int a[row][col],i,j;

      printf("\nEnter (%d X %d) Elements ....\n");
      for(i=0; i<row; i++)
      {
         for(j=0; j<col; j++)
         {

            printf("\nEnter (%d, %d) Element : ", i, j);
            scanf("%d", &a[i][j]);
         }
      }

      printf("\nMatrix Elements ....\n");
      for(i=0; i<row; i++)
      {
            for(j=0; j<col; j++)
            {
               printf("%d\t", a[i][j]);
            }
            printf("\n\n");
      }
  return 0;
}
```

Output :

```
Enter Number of rows : 2
Enter Number of columns : 2
Enter (2 X 2) Elements ....

Enter (0, 0) Element : 10

Enter (0, 1) Element : 20

Enter (1, 0) Element : 30

Enter (1, 1) Element : 40

Matrix Elements ....
10     20
30     40
```

2. Write a C Program to add two matrices.

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

    int row,col,i,j;
    int a[10][10],b[10][10],res[10][10];

    printf("\nEnter Number of rows : ");
    scanf("%d",&row);

    printf("\nEnter Number of columns : ");
    scanf("%d",&col);


    printf("\nEnter First Matrix Elements :\n");
    for(i=0; i<row; i++)
    {
       for(j=0; j<col; j++)
       {
          printf("\nEnter an Element : ");
          scanf("%d",&a[i][j]);
       }
    }


    printf("\nEnter First Matrix Elements :\n");
    for(i=0; i<row; i++)
    {
       for(j=0; j<col; j++)
       {
          printf("\nEnter an Element : ");
          scanf("%d",&b[i][j]);
       }
    }


   printf("\nResultant Matrix Elements \n");
   for(i=0; i<row; i++)
   {
       for(j=0; j<col; j++)
       {
           res[i][j]=a[i][j]+b[i][j];
           printf("%d\t",res[i][j]);
       }
       printf("\n");
    }

    return 0;
}
```

Output :

```
Enter Number of columns : 2

Enter First Matrix Elements :

Enter an Element : 10

Enter an Element : 10

Enter an Element : 10

Enter an Element : 10

Enter First Matrix Elements :

Enter an Element : 20

Enter an Element : 20

Enter an Element : 20

Enter an Element : 20

Resultant Matrix Elements
30      30
30      30
```

3.Write a C program to find sum of all array elements.

```
#include<stdio.h>
int main()
{
	int a[10];
	int i,n,sum=0;

	printf("\n\nEnter number of Elements :");
	scanf("%d",&n);

	for(i=0; i<n; i++)
	{
	   printf("\n\nEnter an elements :");
	   scanf("%d",&a[i]);
	}

	for(i=0; i<n; i++)
	{
	   sum=sum+a[i];
	}

	printf("\nSum of Array Elements  = %d",sum);

	return 0;
}
```

Output :

```
Enter number of Elements :5


Enter an elements :10


Enter an elements :20


Enter an elements :30


Enter an elements :40


Enter an elements :50

Sum of Array Elements  = 150
```

4. Write a C program to find sum of all even numbers from a given array.

```
#include<stdio.h>
int main()
{
        int a[10];
        int i,n,sum=0;

       printf("\n\nEnter number of Elements :");
       scanf("%d",&n);

        for(i=0; i<n; i++)
        {
           printf("\n\nEnter an elements :");
           scanf("%d",&a[i]);
        }


        printf("\nEven numbers :");
        for(i=0; i<n; i++)
        {
           if(a[i]%2==0)
           {
               printf("\n%d",a[i]);
               sum=sum+a[i];
            }
         }


         printf("\nSum of Even Numbers =%d",sum);

         return 0;
}
```

Output :

```
Enter number of Elements :8
Enter an elements :1
Enter an elements :2
Enter an elements :3
Enter an elements :4
Enter an elements :5
Enter an elements :6
Enter an elements :7
Enter an elements :8

Even numbers :
2
4
6
8
Sum of Even Numbers =20
```

5. Write a C program to find sum of all odd numbers from a given array.

```
#include<stdio.h>
int main()
{
        int a[10];
        int i,n,sum=0;

        printf("\nEnter number of Elements :");
        scanf("%d",&n);


        for(i=0; i<n; i++)
        {
           printf("\nEnter an elements :");
           scanf("%d",&a[i]);
         }


         printf("\nOdd numbers :\n");
         for(i=0; i<n; i++)
         {
              if(a[i]%2==1)
              {
                    printf("%d\n",a[i]);
                    sum=sum+a[i];
               }
          }

          printf("\nSum of digits =%d",sum);
          return 0;
}
```

Output :

```
Enter number of Elements :8
Enter an elements :1
Enter an elements :2
Enter an elements :3
Enter an elements :4
Enter an elements :5
Enter an elements :6
Enter an elements :7
Enter an elements :8

Odd numbers :
1
3
5
7

Sum of digits =16
```

6.Write a C program to find sum of all even & odd numbers from a given array.

```
#include<stdio.h>
int main()
{
   int a[10];
   int i,n,sum=0;

   printf("\nEnter number of Elements :");
   scanf("%d",&n);


   for(i=0; i<n; i++)
   {
      printf("\nEnter an elements :");
      scanf("%d",&a[i]);
   }


   for(i=0; i<n; i++)
   {
     if(a[i]%2==1 || a[i]%2==0)
     {
        sum=sum+a[i];
     }
   }

   printf("\nSum of Even&Odd Numbers =%d",sum);

   return 0;
}
```

Output :

```
Enter number of Elements :6

Enter an elements :10

Enter an elements :20

Enter an elements :5

Enter an elements :9

Enter an elements :12

Enter an elements :17

Sum of Even&Odd Numbers =73
```

7.Write a C program to find maximum of all even numbers from a given array.

```
#include <stdio.h>
int main()
{
   int a[10];
   int i,n,max;

   printf("\nEnter number of Elements :");
   scanf("%d",&n);

   for(i=0; i<n; i++)
   {
      printf("\nEnter an elements :");
      scanf("%d",&a[i]);
   }

   max=a[0];
   for(i=0; i<n; i++)
   {
      if(a[i]%2==0)
      {
         if(a[i]>max)
         {
            max=a[i];
         }

      }

   }

   printf("\nLargest Even  Number is %d\n",max);

   return 0;
}
```

Output :

```
Enter number of Elements :5

Enter an elements :10

Enter an elements :20

Enter an elements :40

Enter an elements :50

Enter an elements :60

Largest Even  Number is 60
```

8.Write a C program to find maximum of all Odd numbers from a given array.

```
#include <stdio.h>
int main()
{
        int a[10];
        int i,n,max;

        printf("\nEnter number of Elements :");
        scanf("%d",&n);

        for(i=0; i<n; i++)
        {
           printf("\nEnter an elements :");
           scanf("%d",&a[i]);
         }

        max=a[0];
        for(i=0; i<n; i++)
        {
            if(a[i]%2==1)
            {
                 if(a[i]>max)
                 {
                      max=a[i];
                 }

            }

       }

        printf("\nLargest Odd  Number is %d\n",max);

        return 0;
}
```

Output :

```
Enter number of Elements :5

Enter an elements :1

Enter an elements :3

Enter an elements :7

Enter an elements :5

Enter an elements :13

Largest Odd  Number is 13
```

9.Write a C program to find minimum of all even numbers from a given array.

```
#include <stdio.h>
int main()
{
        int a[10];
        int i,n,min;

        printf("\nEnter number of Elements :");
        scanf("%d",&n);

        for(i=0; i<n; i++)
        {
            printf("\nEnter an elements :");
            scanf("%d",&a[i]);
        }

        min=a[0];
        for(i=0; i<n; i++)
        {
             if(a[i]%2==0)
             {
                  if(a[i]<min)
                  {
                        min=a[i];
                  }

             }

         }

         printf("\nSmallest Even Number is %d\n",min);

          return 0;
}
```

Output :

```
Enter number of Elements :5

Enter an elements :2

Enter an elements :4

Enter an elements :6

Enter an elements :8

Enter an elements :10

Smallest Even Number is 2
```

10.Write a C program to find minimum of all odd numbers from a given array.

```
#include <stdio.h>
int main()
{
   int a[10];
   int i,n,min;

   printf("\nEnter number of Elements :");
   scanf("%d",&n);

   for(i=0; i<n; i++)
   {
      printf("\nEnter an elements :");
      scanf("%d",&a[i]);
   }

   min=a[0];
   for(i=0; i<n; i++)
   {
      if(a[i]%2==1)
      {
         if(a[i]<min)
         {
            min=a[i];
         }

      }

   }

   printf("\nSmallest Odd Number is %d\n",min);

   return 0;
}
```

Output :

```
Enter number of Elements :5

Enter an elements :3

Enter an elements :2

Enter an elements :6

Enter an elements :5

Enter an elements :9

Smallest Odd Number is 3
```

11.Write a function that accepts array of integers to find maximum and minimum element in an array.

```
#include<stdio.h>
int main()
{
        int a[10],i,n,min,max;

        printf("\nEnter the number of elements : ");
        scanf("%d",&n);

         for(i=0; i<n; i++)
        {
           printf("\nEnter the element :");
           scanf("%d",&a[i]);

        }

        min=max=a[0];
        for(i=0; i<n; i++)
        {
              if(a[i]<min)
              {
                  min=a[i];
              }
               if(a[i]>max)
              {
                  max=a[i];
               }
         }

         printf("\nMaximum of array element is :  %d",max);

         printf("\nMinimum of array element is :  %d",min);

         return 0;
}
```

Output ;

```
Enter the number of elements : 6
Enter the element :3
Enter the element :5
Enter the element :4
Enter the element :2
Enter the element :10
Enter the element :20

Maximum of array element is : 20
Minimum of array element is : 2
```

Strings:

String Constant/Literal: Group of characters enclosed in a double quote, ends with a

null ('\0') character.
\0 – slash zero.
- Anything that we enclose in a double quote is a string in C.

Ex:

"Welcome to c programming",
"Ravi",
"KLE BCA Athani"
"12345"
"&*&*&*tyty134343"

- In C space is a character.

String Declaration:

In C, string is declared as using char array.

Syntax:

```
char  stringname[length];
```

char – char is a keyword we must use for string declaration, which represents we are declaring a variable for characters.

stringname – stringname is the name of the string variable, which is nothing character array name.

length – length specifies number of characters an array can contains.

Ex1:

```
char str[25];
```

- here str is a string variable which can hold maximum 25 characters in the str array. char str2[50];

-here str2 is a char array which can hold up to 25 characters.

String Initialization OR String reading OR string Input:

- String initialization is nothing but reading characters into a string variable.
- Assigning character values to a string variable.
-

Ways to initialize string variable in C.

1.Using Index wise:

```
//string declaration
char name[25];

//string initialization
name[0] = 'h';
name[1] = 'e';
name[2] = 'l';
name[3] = 'l';
name[4] = 'o';
name[5] = 'w';
name[6] = '\0';
```

2.Declare and Initialize at a time.

```
char name[ 20 ] = "hello";
char name[ ] = "hello";
```

3.Using Initializer List:

```
char name[25] = {'h','e','l','l','o','\0'};
char name[ ] = {'h','e','l','l','o','\0'};
```

4.using scanf( ) & %s - Not Recommnded

```
char name[20];
printf("\nEnter Your Name : ");
fflush(stdin);
scanf("%s", name);

printf("\nYour Name : %s",name);
```

Output1:

```
Enter Your Name : Ravi
Your Name : Ravi
```

Output2:

```
Enter Your Name : Ravi Shankar
Your Name : Ravi
```

Note : %s in scanf( ) works till whitespace, it will read the data until whitespace character only. Hence it is not recommended.

1.using gets() & puts( ) function from string.h

gets( ) : To read string data from standard input device.[Input Function]
puts() : To print string data to standard output device.[Output Function]

gets ( ) :

syntax:

char *gets(char *str)

- gets (chararray) function is a standard library string input function, it is used to reada string from keyboard at runtime until Enter key.
- We need to pass a char array name to this function; it will read the string and storesit in given char array.

puts ( ) :

syntax:

int puts(char[]);

- puts(chararray) function is a standard library string unformatted output function, itis used to write/print a string data to standard output device (Console).
- We need to pass a char array name to this function; it will print the string on output screen.

Ex:

```
char name[20];
printf("\nEnter Your Name : ");
fflush(stdin);
gets(name);

printf("\nYour Name : ");
puts(name);
```

Output1:

```
Enter Your Name : Ravi
Your Name : Ravi
```

Output2:

```
Enter Your Name : Ravi Shankar
Your Name : Ravi Shankar
```

Character Handling Function :

Function Name	Syntax	Functionality
toascii()	int toascii(int ch);	The *toascii*() function shall convert itsargument into a 7-bit ASCII character
toupper()	int toupper(int ch);	The toupper() function is used toconvert lowercase alphabet to uppercase.
tolower()	int tolower(int ch);	The tolower() function is used toconvert lowercase alphabet to uppercase.
isalpha()	int isalpha(char ch)	isalpha(c) is a function in C which can be used to check if the passed character is an alphabet or not. It returns a non-zero value if it's an alphabet else it returns 0
isdigit()	int isdigit(char ch)	The isdigit(c) is a function in C which can be used to check if the passed character is a digit or not. It returns a non-zero value if it's adigit else it returns 0.
isalnum()	int isalnum(char ch)	The function isalnum() is used tocheck that the character is alphanumeric or not. It returns non-zero value, if the character is alphanumeric means letter or number otherwise, returnszero
isspace()	int isspace(char ch)	If an argument (character) passedto the isspace() function is a white-space character, it returns non-zero integer. If not, it returns0.
isupper()	int isupper(char ch)	If an argument (character) passedto the isupper() function is a Upper Case character, it returns non-zero integer. If not, it returns0.
islower()	int islower(char ch)	If an argument (character) passedto the islower() function is a Lower Case character, it returns non-zero integer. If not, it returns0.

Syntax for Unformatted output functions :

Function Name	Syntax	Functionality
gets( )	char *gets(char *str); OR char *gets(char[] str);	Is used to read string data at run time from standard input device. itreads even white space character also.
getc( )	int getc(FILE *stream);	getc() reads the next character from a file. it takes a file pointer tothe file .
getch( )	int getch();	Used to read the alphanumeric character input from the user. but that the entered character will not be displayed.
getchar( )	char getchar();	Used t reads character type datafrom the input. and reads one character at a time till the user presses the enter key.
puts( )	int puts(const char *str; OR int puts(char[] str);	Used to writes a string characteron output screen. non-negative value is returned.
putc( )	int putc (int char, FILE*stream);	Is used for writing a character intoa file.
putch( )	int putch(int ch);	It prints a specified character at standard output device(console).
putchar( )	int putchar(int char);	It is used to write a specifiedcharacter to standard output device(console).
scanf( )	Int scanf("FormatSpecifiers",&v1,&v2,);	It is used to read/scan data from standard input device at run time.
printf( )	int printf("Formatted string+format Specifiers", list of variables);	Used to print any string with format specifiers corresponds list of variable on the standard outputScreen.

String Handling Functions

Function Name	Syntax	Functionality
strlen( )	int strlen(char s1[]);	(Length of the string)
strcpy()	int strcpy(char s1[ ], char s2[ ]);	copies a string to another
strcat( )	char * strcat(char s1[ ], char s2[ ]);	concatenates(joins)two strings
strcmp( )	int strcmp(char s1[ ], char s2[ ]);	compares two strings
strlwr( )	char * strlwr(char s1[ ]);	converts string to lowercase
strupr( )	char * strupr(char s1[ ]);	converts string to uppercase
strrev( )	char * strrev(char s1[]);	reverses a given string

1.Program to convert lowercase to uppercase character.

```
#include<stdio.h>
#include<string.h>
int main()
{
	char ch1,ch2;

	printf("\nEnter a Character : ");
	scanf("%c",&ch1);

	ch2=toupper(ch1);
	printf("\nUppercase Letter : %c",ch2);

	return 0;
}
```

Output :

```
Enter a Character : d
Uppercase Letter : D
```

2.Program to convert uppercase to lowercase character.

```
#include<stdio.h>
#include<string.h>
int main()
{
	char ch1,ch2;

	printf("\nEnter a Character : ");
	scanf("%c",&ch1);

	ch2=tolower(ch1);
	printf("\nLowercase Letter : %c",ch2);

	return 0;
}
```

Output :

```
Enter a Character : D
Lowercase Letter : d
```

3.Program to check given character is alphabet or not.

```
#include<stdio.h>
#include<ctype.h>
int main()
{
      char c;

      printf("\nEnter a character :");
      scanf("%c",&c);

      if(isalpha(c)==0)
      {
          printf("%c is not an alphabet",c);
      }
      else
      {
         printf("%c is an alphabets",c);
      }

      return 0;
}
```

Output :

```
Enter a character  :5
5 is not an alphabet

Enter a character : g
g is an alphabets
```

4.Program to check the alphanumeric .

```
#include<stdio.h>
#include<ctype.h>
int main()
{
       char c;

       printf("\nEnter a character :");
       scanf("%c",&c);

       if(isalnum(c)==0)
       {
          printf("%c is not an alphanumeric ",c);
       }
       else
       {
          printf("%c is an alphanumeric",c);
       }
       return 0;
}
```

Output :

```
Enter a character :d d is an alphanumeric
Enter a character :@
@ is not an alphanumeric character
```

4.Program to check the given data is digit or not.

```
#include<stdio.h>
#include<ctype.h>
int main()
{
      char c;

      printf("\nEnter a num/char :");
      scanf("%c",&c);

      if(isdigit(c)==0)
      {
          printf("%c is not an Digit",c);
      }
      else
      {
         printf("%c is an Digit",c);
      }

       return 0;
}
```

Output :

```
Enter a num/char :5
5 is an Digit

Enter a num/char :g
g is not an Digit
```

STRING HANDLING FUNCTIONS

1.Strlen without using library functions.

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str[25]; int n;

   printf("\nEnter a string :");
   fflush(stdin);
   gets(str);

   n=strlen (str);

   printf("\nString length :%d",n);

   return 0;
}
```

Output :

```
Enter a string :school
String length :6
```

2.Strlen without using library functions.

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str[25]; int i,n;

   printf("\nEnter a string :");
   fflush(stdin);
   gets(str);

   i=0;
   while(str[i] !='\0')
   {
      i++;
   }

   printf("\nString length :%d",i);
   return 0;
}
```

Output :

```
Enter a string :sharat kle
String length :10
```

3.Strcat without using library functions.

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str1[100],str2[100]; char *str3;

   printf("\nEnter First string :");
   fflush(stdin);
   gets(str1);


   printf("\nEnter Second string :");
   fflush(stdin);
   gets(str2);

   str3=strcat(str1,str2);
   printf("\nConcatenation :");
   puts(str3);
   return 0;
}
```

Output :

```
Enter First string :sad
Enter Second string :sad
Concatenation :sadsad
```

4.Strcat without using library functions.

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str1[100],str2[100]; int i,j;

   printf("\nEnter First string :");
   fflush(stdin);
   gets(str1);

   printf("\nEnter Second string :");
   fflush(stdin);
   gets(str2);

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

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

   str1[i]='\0';
   printf("\nConcatenation :=%s",str1);

   return 0;
}
```

Output :

```
Enter First string :sharat
Enter Second string :kumat
Concatenation :=sharatkumar
```

5.Strcpy with using library functions.

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str1[100],str2[100]; int i,j;

   printf("\nEnter First string :");
   fflush(stdin);
   gets(str1);
   strcpy(str2,str1);

   printf("\nString copied :%s",str2);
   return 0;
}
```

Output :

```
Enter First string :main
String copied :main
```

6.Strcpy without using library functions.

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str1[100],str2[100]; int i,j;

   printf("\nEnter First string :");
   fflush(stdin);
   gets(str1);

   i=0;
   j=0;
   while(str1[i] !='\0')

   {
      str2[j]=str1[i];
      i++;
      j++;
   }

   str2[j]='\0';

   printf("\nString copied :%s",str2);
   return 0;
}
```

Output :

```
Enter First string :hello
String copied :hello
```

7.Strrev with using library functions.

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str1[100],str2[100]; char *temp;

   printf("\nEnter string :");
   fflush(stdin);
   gets(str1);

   temp=strrev(str1);
   printf("\nReverse string :%s",temp);

   return 0;
}
```

Output :

```
Enter string :Sample
Reverse string :elpmaS
```

8.Strrev without using library functions.

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str1[100],str2[100];
   int i,j;

   printf("\nEnter string :");
   fflush(stdin);
   gets(str1);

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

   j=0;
   i--;
   while(i>=0)
   {
      str2[j]=str1[i];
      i--;
      j++;
   }
   str2[j]='\0';

   printf("\nReverse string:%s",str2);
   return 0;
}
```

Output :

```
Enter string :Sharat
Reverse
string:tarahS
```

9. Strcmp with using library functions.

```
#include<stdio.h>
#include<string.h>
int main()
{
   char s1[50],s2[25]; int n;

   printf("\nEnter First String : ");
   fflush(stdin);
   gets(s1);

   printf("\nEnter Second String : ");
   fflush(stdin);
   gets(s2);

   n = strcmp(s1, s2);
   if(n == 0)
   {
      printf("\nStrings are equal");
   }
   else
   {
      printf("\nStrings are not equal");
   }
   return 0;
}
```

Output :

```
Enter First String : student

Enter Second String : student

Strings are equal



Enter First String : student

Enter Second String : teacher

Strings are not equal
```

10. Strcmp without using library functions.

```
#include<string.h>
#include<stdio.h>
int main()
{
   char s1[50], s2[25];
   int n1,n2,i,j,flag=0;

   printf("\nEnter First String : "); fflush(stdin);
   gets(s1);

   printf("\nEnter Second String : "); fflush(stdin);
   gets(s2);

   n1 = strlen(s1); n2 = strlen(s2);

   if(n1 == n2)
   {
      i=0; j=0;
      while(s1[i] != '\0')
      {
         if(s1[i] != s2[j])
         {
            flag = 1; break;
         }
         else
         {
            flag = 0;
         }
         i++;
         j++;
      }
      if(flag == 1)
      {
         printf("\nStrings are not equal");
      }
      else
      {
         printf("\nStrings are equal");
      }

   }

   return 0;
}
```

Output :

```
Enter First String : motive
Enter Second String : motive
Strings are equal
```

11.Program to reverse given string and check palindrome or not.

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str1[25], str2[25], *temp; int n;

   printf("\nEnter a String : ");
   fflush(stdin);
   gets(str1);

   strcpy(str2,str1);
   temp=strrev(str2);
   n=strcmp(str1,str2);

   if(n == 0)
   {
      printf("\nPalindrome");
   }
   else
   {
      printf("\nNot a Palindrome");
   }

   printf("\nReverse = %s",temp);

   return 0;
}
```

Output :

```
Enter a String : self
Not a Palindrome
Reverse = fles

Enter a String : gadag
Palindrome
Reverse = gadag
```