Unit 6

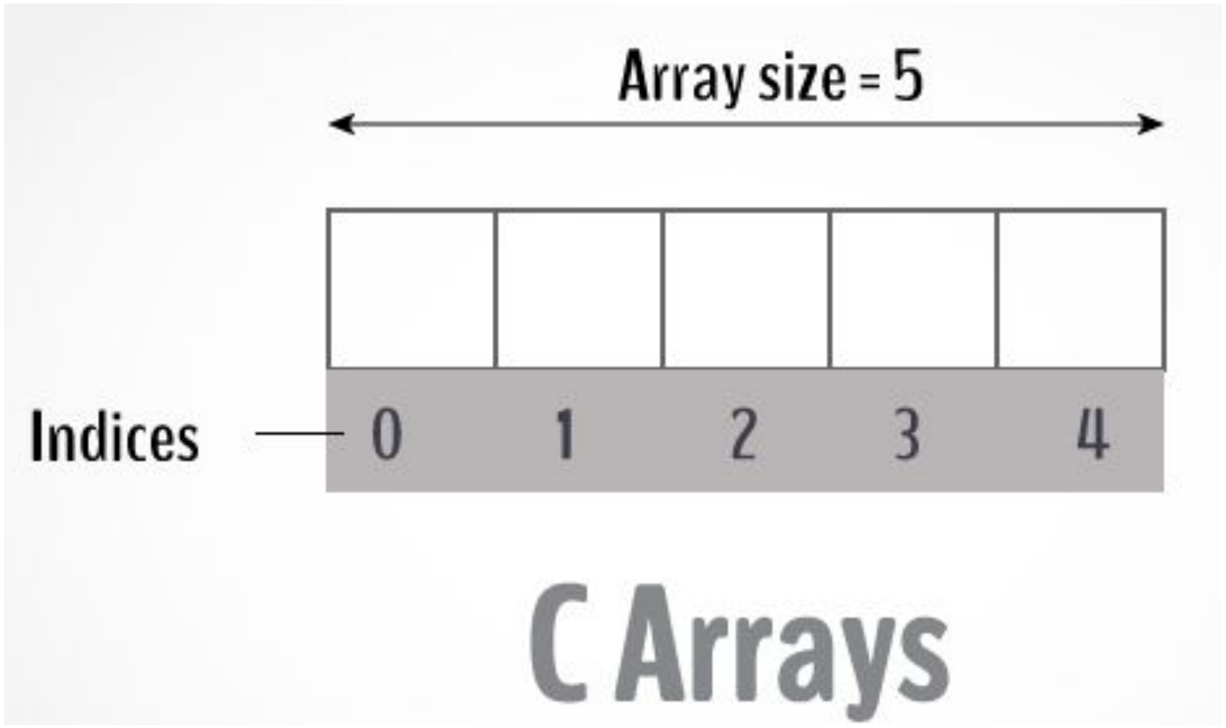
Array and String

Introduction

- An array is a collection of a fixed number of values of a single type.
- For example: if you want to store 100 integers in sequence, you can create an array for it.

```
int data[100];
```

- The size and type of arrays cannot be changed after its declaration.
- Arrays are of two types:
  - One-dimensional arrays
  - Multidimensional arrays


Array size = 5
Indices
0
1
2
3
4
C Arrays

Single Dimensional Array

How to declare arrays?

- Syntax:

```
data_type  array_name[array_size];
```

For example,

```
float mark[5];
```

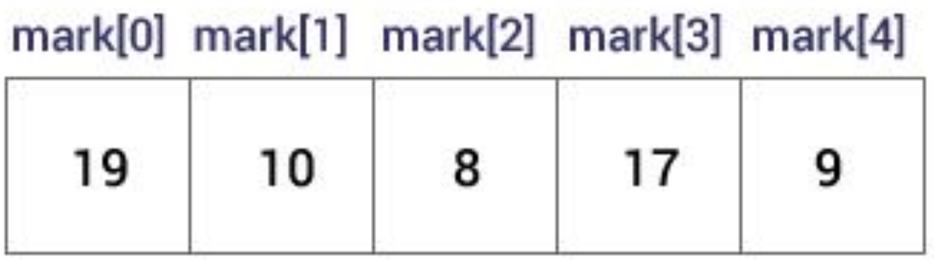
- Here, we declared an array, mark, of floating-point type and size 5. Meaning, it can hold 5 floating-point values.

How to initialize an array?

- It's possible to initialize an array during declaration. For example,

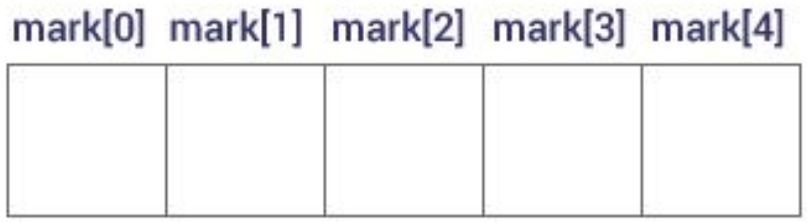
  int mark[5] = {19, 10, 8, 17, 9};
- Another method to initialize array during declaration:

  int mark[] = {19, 10, 8, 17, 9};



Elements of an Array and How to access them?

- You can access elements of an array by indices.
- Suppose you declared an array mark as above. The first element is mark[0], second element is mark[1] and so on.
- **Few key notes:**
- Arrays have 0 as the first index not 1. In this example, mark[0]
- If the size of an array is n, to access the last element, (n-1) index is used. In this example, mark[4]
- Suppose the starting address of mark[0] is 2120. Then, the next address, a[1], will be 2124, address of a[2] will be 2128 and so on. It's because the size of a float is 4 bytes.



Loading Data into an Array

```
void main()
{
    int a[5],i;
    printf("enter age of five persons");
    for (i=0;i<5;i++)
    {
        scanf("%d",&a[i]);
    }
}
```

Example

```
#include<stdio.h>
void main()
{
	int a[5],sum=0,avg,i;
	printf("enter 5 numbers");

	for(i=0;i<5;i++){
		scanf("%d",&a[i]);
	}

	for(i=0;i<5;i++){
		sum+=a[i];
	}
	avg=sum/5;
	printf("\nsum is %d",sum);
	printf("\naverage is %d",avg);
}
```

- Write a program in C to store elements in an array and print it.
- Write a program in C to find the sum of all elements of the array
- Write a program in C to read n number of values in an array and display it in reverse order
- Write a program in C to copy the elements of one array into another array

- WAP to find the smallest and the largest element in the array of size 10.
- WAP to find the second largest element in the array.
- WAP to search the number inputted by user in an array.
- WAP to count the even numbers present in an array of size n.
- WAP to sort the array in ascending and descending order.
- Write a program in C to separate odd and even integers in separate arrays

Sorting array in ascending order

```
#include<stdio.h>
#include<conio.h>
void main()
{
	int a[10],i,j,temp=0;
	printf("enter 5 numbers");
	for(i=0;i<5;i++)
	{
		scanf("%d",&a[i]);
	}
	printf("\nbefore sorting \n");
	for(i=0;i<5;i++)
	{
		printf("%d\t",a[i]);
	}
for(i=0;i<5-1;i++){
        for(j=i+1;j<5;j++){
                if(a[i]>a[j]){
                        temp=a[i];
                        a[i]=a[j];
                        a[j]=temp;
                        }
                }
        }
        printf("\nafter sorting ----");
        for(i=0;i<5;i++){
                printf("%d\t",a[i]);
        }
        getch();
}
```

Second largest element in an array

```
#include<stdio.h>
void main()
{
    int a[5];
    int i,max1,max2;

    printf("enter five numbers");
    for(i=0;i<5;i++)
    {
        scanf(" %d",&a[i]);
    }
    max1=max2=0;
```

```
	for(i=0;i<5;i++)
	{
		if(a[i]>max1){
			max2=max1;
			max1=a[i];
		}
		else if(a[i] > max2 && a[i] < max1)
			{
				max2 = a[i];
			}
	}
	printf("second largest element is %d",max2);
}
```

Join two array in 3rd array

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[10],b[10],c[10],i,j,temp=0;
    printf("enter 5 numbers for 1st array");
    for(i=0;i<5;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("enter 5 numbers for 2nd array");
    for(i=0;i<5;i++)
    {
        scanf("%d",&b[i]);
    }
```

```
        for(i=0;i<5;i++){
         c[i]=a[i];
         temp++;
     }
     j=temp;
     for(i=0;i<5;i++){
         c[j]=b[i];
         j++;
     }
     printf("\nafter concatination ----");
     for(i=0;i<10;i++){
         printf("%d\t",c[i]);
     }
     getch();
}
```

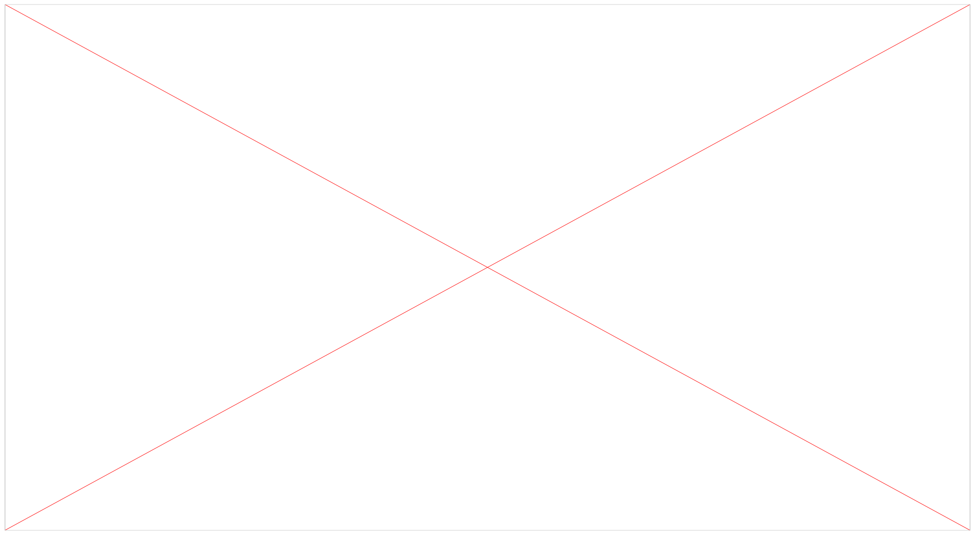
To separate odd and even integers in separate arrays

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[10],o[10],e[10],i,c1=0,c2=0;
    printf("enter 10 numbers");
    for(i=0;i<10;i++){
        scanf("%d",&a[i]);
    }
    for(i=0;i<10;i++){
        if(a[i]%2==0){
            e[c1]=a[i];
            c1++;
        }
else
      {
            o[c2]=a[i];
            c2++;
      }
}
printf("\neven items in the array are");
      for(i=0;i<c1;i++)
      {
            printf("\n%d",e[i]);
      }
```

```
	printf("\nodd items in the array are");
	for(i=0;i<c2;i++)
	{
		printf("\n%d",o[i]);
	}
	getch();
}
```

Multidimensional Array

- In C programming, you can create array of an array known as multidimensional array. For example,
- float x[3][4];
- Here, x is a two-dimensional (2d) array. The array can hold 12 elements. You can think the array as table with 3 row and each row has 4 column.



How to initialize a multidimensional array?

```
int c[2][3] = {{1, 3, 0}, {-1, 5, 9}};

int c[][3] = {{1, 3, 0}, {-1, 5, 9}};

int c[2][3] = {1, 3, 0, -1, 5, 9};
```

Wrong

```
int c[2][ ] = {1, 3, 0, -1, 5, 9};

int c[ ][ ] = {1, 3, 0, -1, 5, 9};
```

Matrix Multiplication in c

```
#include<stdio.h>
#include<conio.h>
void main()
{
      int a[10][10],b[10][10],c[10][10];
      int r1,c1,i,j,k,r2,c2,sum=0;
      printf("Enter number of rows and columns of first matrix\n");
      scanf("%d%d", &r1, &c1);
      printf("Enter elements of first matrix\n");

      for (i = 0; i < r1; i++)
      for (j = 0; j < c1; j++)
            scanf("%d", &a[i][j]);
```

```
printf("Enter number of rows and columns of second matrix\n");
scanf("%d%d", &r2, &c2);

if (c1 != r2){
  printf("The matrices can't be multiplied with each other.\n");
}
else
  {
    printf("Enter elements of second matrix\n");
    for (i = 0; i < r2; i++)
      for (j = 0; j < c2; j++)
         scanf("%d", &b[i][j]);
```

```
for (i = 0; i < r1; i++) {
    for (j = 0; j < c2; j++) {
      for (k = 0; k < r2; k++) {
        sum = sum + a[i][k]*b[k][j];
      }

      c[i][j] = sum;
      sum = 0;
    }
  }
```

```
printf("Product of the matrices:\n");

    for (i = 0; i < r1; i++) {
      for (j = 0; j < c2; j++)
        printf("%d\t", c[i][j]);

      printf("\n");
    }
  }

}
```

Exercises

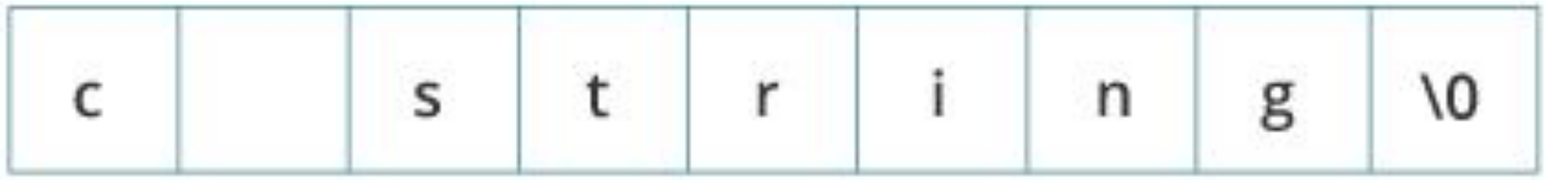
- Write a program in C to input one matrix and find sum of diagonals elements of a matrix
- Write a program in C to find transpose of a given matrix and display.
- Write a program in C to accept two matrices and check whether they are equal
- Write a program in C to find number of times odd number occured in an matrix
- Write a program in C to find the largest number present in a matrix.
- Write a program in C to check whether the number given by user is present in 3*3 matrix or not.

Character Array

String Handling in C

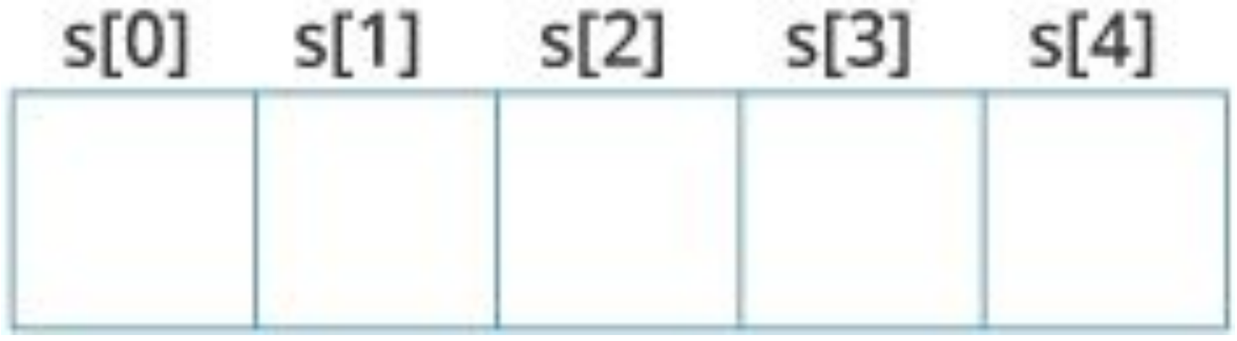
Introduction

- In C programming, a string is an array of characters terminated with a null character \0. For example:
- "c string“
- When compiler encounters a sequence of characters enclosed in the double quotation marks, it appends a null character \0 at the end.



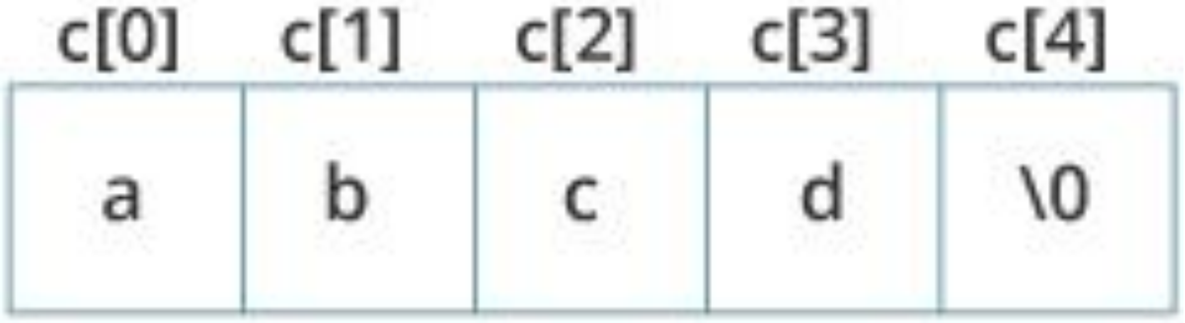
How to declare a string?

- Before you can work with strings, you need to declare them first. Since string is an array of characters. You declare strings in a similar way like you do with arrays.
- Here's how you declare a string:
  - char s[5];



How to initialize strings?

- You can initialize strings in a number of ways.
  - char c[] = "abcd";
  - char c[50] = "abcd";
  - char c[] = {'a', 'b', 'c', 'd', '\0'};
  - char c[5] = {'a', 'b', 'c', 'd', '\0'};



Read and write String from/to the user

```
#include<stdio.h>
void main()
{
    char name[50];
    printf("enter your name");
    //scanf("%s",name);
    //scanf("%[^\n]",name);
    gets(name);

    //printf("you have entered %s",name);
    puts(name);
}
```

Manipulating string without using string library function

1. Finding the length of a string
2. Reversing the string
3. Converting lowercase to uppercase
4. Converting uppercase to lowercase
5. Copying one string to another
6. Compare two strings
7. Concatenating two strings

Find the length of a string

```
#include<stdio.h>
void main()
{
    char name[50];
    int i,len=0;
    printf("enter your name");
    gets(name);
    for(i=0;name[i]!='\0';i++){
        len++;
    }
    printf("\nthe length of your name is %d",len);
}
```

Copy one string to another

```
#include<stdio.h>
void main()
{
    char a[50],b[50];
    int i=0,len=0,temp;
    printf("enter your name");
    gets(a);
    for(i=0;a[i]!='\0';i++){
        b[i]=a[i];
    }
    b[i]='\0';
    printf("\nthe 1st array value is%s",a);
    printf("\nthe 2nd array value is %s",b);
}
```

Reverse the given string

```
#include<stdio.h>
void main()
{
    char name[50];
    int i=0,len=0;
    int temp;
    printf("enter your name");
    gets(name);
    for(i=0;name[i]!='\0';i++){
        len++;
    }
len=len-1;
i=0;
while(i<len)
    {
        temp=name[i];
        name[i]=name[len];
        name[len]=temp;
        i++;
        len--;
    }
    printf("\nthe reverse of your name is %s",name);
}
```

Find word in sentence

```
#include<stdio.h>
#include<string.h>
void main()
{
    int i=0, j=0, flag=0, n, m;
    char sen[50], word[20];
    printf("Enter a sentence");
    gets(sen);
    printf("Enter a word");
    gets(word);
    n = strlen(sen);
    m = strlen(word);
```

```
if (m <= n)
        {
                while (i < n)
                {
                        j = 0;
                        while (i < n && j < m && sen[i] == word[j]) {
                                i++;
                                j++;
                        }
                        if (( i == n || sen[i] == ' ') && j == m){
                                flag=1;
                                break;
                        }
```

```
            ++i;
            }
        }
        else
        {
            printf("Not found");
        }
        if(flag==1){
            printf("%s is present",word);
        }
        else{
            printf("%s is not present",word);
        }
}
```

String Manipulations In C Programming Using Library Functions

Function	Work of Function
strlen()	Calculates the length of string
strcpy()	Copies a string to another string
strncpy()	Copies first n characters of string2 to string1
strcat()	Concatenates(joins) two strings
strlwr()	Converts string to lowercase
strupr()	Converts string to uppercase
strrev()	Reverse the given string
strcmp()	Compares two string
strncmp()	Compares first n characters of two strings
strcasecmp()	Case insensitive version of strcmp()
strncasecmp()	Case insensitive version of strncmp()

- Strings handling functions are defined under "string.h" header file.

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

C strlen()

- **The strlen() function calculates the length of a given string.**
- **C strlen() Prototype**
  - size_t strlen(const char *str);
- The function takes a single argument, i.e, the string variable whose length is to be found, and returns the length of the string passed.
- The strlen() function is defined in <string.h> header file.

Example: C strlen() function

```
#include <stdio.h>
#include <string.h>
void main()
{
   char a[20]="Program";
   char b[20]={'P','r','o','g','r','a','m','\0'};
   char c[20];
   printf("Enter string: ");
   gets(c);
   printf("Length of string a = %d \n",strlen(a));
   //calculates the length of string before null charcter.
   printf("Length of string b = %d \n",strlen(b));
   printf("Length of string c = %d \n",strlen(c));
}
```

C strcpy()

- **The strcpy() function copies the string to the another character array.**
- **strcpy() Function prototype**
- char* strcpy(char* destination, const char* source);
- The strcpy() function copies the string pointed by source (including the null character) to the character array destination.
- This function returns character array destination.

Example: C strcpy()

```
#include <stdio.h>
#include <string.h>
int main()
{
   char str1[10]= "awesome";
   char str2[10];
   char str3[10];
   strcpy(str2, str1);
   strcpy(str3, "well");
   puts(str2);
   puts(str3);
   return 0;
}
```

C strcat()

- **The function strcat() concatenates two strings.**
- In C programming, strcat() concatenates (joins) two strings.
- The strcat() function is defined in <string.h> header file.
- **C strcat() Prototype**
- char *strcat(char *dest, const char *src)
- It takes two arguments, i.e, two strings or character arrays, and stores the resultant concatenated string in the first string specified in the argument.

Example: C strcat() function

```
#include <stdio.h>
#include <string.h>
void main()
{
   char str1[20] , str2[20] ;
   printf("enter your firstname");
   gets(str1);
   printf("enter your lastname");
   gets(str2);
   //concatenates str1 and str2 and resultant string is stored in str1.
   strcat(str1,str2);
   puts(str1);
   puts(str2);
}
```

C strcmp()

- **The strcmp() function compares two strings and returns 0 if both strings are identical.**
- **C strcmp() Prototype**
- int strcmp (const char* str1, const char* str2);
- The strcmp() function takes two strings and return an integer.
- The strcmp() compares two strings character by character. If the first character of two strings are equal, next character of two strings are compared. This continues until the corresponding characters of two strings are different or a null character '\0' is reached.

Return Value from strcmp()

Return Value	**Remarks**
0	if both strings are identical (equal)
negative	if the ASCII value of first unmatched character is less than second.
positive integer	if the ASCII value of first unmatched character is greater than second.

Example: C strcmp() function

```
#include <stdio.h>
#include <string.h>
int main()
{
    char str1[] = "abcd", str2[] = "abCd", str3[] = "abcd";
    int result;
    // comparing strings str1 and str2
    result = strcmp(str1, str2);
    printf("strcmp(str1, str2) = %d\n", result);
    result = strcmp(str2, str1);
    printf("strcmp(str2, str1) = %d\n", result);
    // comparing strings str1 and str3
    result = strcmp(str1, str3);
    printf("strcmp(str1, str3) = %d\n", result);
    return 0;
}
```

Example: C strrev() function

```
#include <stdio.h>
#include <string.h>
int main()
{
   char arr[100];
   printf("Enter a string to reverse\n");
   gets(arr);
   strrev(arr);
   printf("Reverse of the string is \n%s\n", arr);
   return 0;
}
```

Example: Program to Sort Strings in Dictionary Order

```
#include<stdio.h>
#include <string.h>
int main()
{
   int i, j;
   char str[10][50], temp[50];
   printf("Enter 10 words:\n");
   for(i=0; i<10; ++i){
     gets(str[i]);
   }
```

```
for(i=0; i<9; ++i)
       for(j=i+1; j<10 ; ++j)
       {
           if(strcmp(str[i], str[j])>0)
           {
               strcpy(temp, str[i]);
               strcpy(str[i], str[j]);
               strcpy(str[j], temp);
           }
       }
```

```
printf("\nIn lexicographical order: \n");
    for(i=0; i<10; ++i)
    {
        puts(str[i]);
    }
    return 0;
}
```

Character Handling Library Function

- The **ctype.h** header file of the C Standard Library declares several functions that are useful for testing and mapping characters.
- All the functions accepts **int** as a parameter
- All the functions return non-zero (true) if the argument c satisfies the condition described, and zero(false) if not.

S.No	Function & Description
1	**int isalnum(int c)**This function checks whether the passed character is alphanumeric.
2	**int isalpha(int c)**This function checks whether the passed character is alphabetic.
3	**int iscntrl(int c)**This function checks whether the passed character is control character. Characters that cannot be printed on the screen are known as control characters. For example, backspace, Escape, newline etc.
4	**int isdigit(int c)**This function checks whether the passed character is decimal digit.
5	**int isgraph(int c)**This function checks whether the passed character has graphical representation using locale. The characters with graphical representations are all those characters that can be printed except for whitespace characters (like ' '), which is not considered as isgraph characters.
6	**int islower(int c)**This function checks whether the passed character is lowercase letter.

7	**int isprint(int c)**This function checks whether the passed character is printable.
8	**int ispunct(int c)**This function checks whether the passed character is a punctuation character.
9	**int isspace(int c)**This function checks whether the passed character is white-space.
10	**int isupper(int c)**This function checks whether the passed character is an uppercase letter.
11	**int isxdigit(int c)**This function checks whether the passed character is a hexadecimal digit.

1	int tolower(int c)This function converts uppercase letters to lowercase.
2	int toupper(int c)This function converts lowercase letters to uppercase.

String Conversion Functions

Typecast function	**Description**
ATOF()	atof( ) function converts string to float
ATOI()	atoi( ) function converts string to int
ATOL()	atol( ) function converts string to long
ITOA()	itoa( ) function converts int to string
LTOA()	ltoa( ) function converts long to string

EXAMPLE PROGRAM FOR ATOI() FUNCTION IN C:

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

int main()
{
   char a[10] = "100";
   int value = atoi(a);
   printf("Value = %d\n", value);
   return 0;
}
```

EXAMPLE PROGRAM FOR ATOF() FUNCTION IN C:

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

int main()
{
   char a[10] = "3.14";
   float pi = atof(a);
   printf("Value of pi = %f\n", pi);
   return 0;
}
```

EXAMPLE PROGRAM FOR ATOL() FUNCTION IN C:

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

int main()
{
   char a[20] = "100000000000";
   long value = atol(a);
   printf("Value = %ld\n", value);
   return 0;
}
```

EXAMPLE PROGRAM FOR ITOA () FUNCTION IN C:

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

int main()
{
   int a=54325;
   char buffer[20];
   itoa(a,buffer,2);   // here 2 means binary
   printf("Binary value = %s\n", buffer);
```

`

```
  itoa(a,buffer,10);   // here 10 means decimal
   printf("Decimal value = %s\n", buffer);

   itoa(a,buffer,16);   // here 16 means Hexadecimal
   printf("Hexadecimal value = %s\n", buffer);
   return 0;
}
```