STRINGS

Input / Output Functions:

- ✓ Based on kind of data processed, the input output functions classified as.
  - ➢ Token oriented input/output functions
  - ➢ Line oriented input/output functions
  - ➢ Character oriented input/output functions

Token oriented input/output functions

- ✓ The input/output functions which processes individual units - characters, integers, strings separated by white spaces is termed as token oriented input/output functions

  Ex:- scanf() and printf()

Line oriented input/output functions

- ✓ The input/output functions which processes entire line is termed as line oriented input/output functions

  Ex:- gets() and puts() functions

Reading a string: gets () -

- ✓ This function allows reading entire line of input including white spaces characters.

Syntax: **Result=gets(input_string); or gets(input_string);**

- ➢ The input_string is the pointer to character array.
- ➢ The gets returns a value of type char.

Ex:-
```
char name[20];
gets(name);
```

The function accepts the characters including white spaces or tabs till the entire key is pressed. Once the entire key is pressed the entered character is stored in the string name.

Putting a string: puts ( ):

- ✓ This function displays string of characters on screen

Syntax: **puts(string_name);**

- ➢ Where, string_name is the string variable which is passed as parameter

- ✓ This displays all the characters till the null (\0) is encountered.

Ex:
```
char name[20]="This is GITAM";
Puts(name);
```
Output: This is GITAM.

Write a C program to read and display string using gets and puts.

```
#include<stdio.h>
        void main()
        {
               char name[20];
               gets(name);
               puts(name);
         }
```

Input:

This is GITAM

Output:

This is GITAM

Write a C program to read and display string using pointer to a string

```
#include<stdio.h>
void main()
 {
        char name[20],*p;
        printf("enter name");
        p=gets(name);
        printf("\n string:");
        puts(p);
 }
```

Output:

```
Enter name
Mother
String: Mother
```

- ✓ The main advantage of using gets function is, the function doesn't check to see whether there is memory for the string which is being reading.

Character oriented input/output functions

- ✓ These functions processes the characters

Using getchar() function and putchar() function:

- ➢ There are single character input-output functions.

getchar():

- ✓ Reads input character from keyboard.
- ✓ No arguments are passed to getchar function

Syntax: **ch = getchar( );**

Where ch is a variable of char data type / integer.

- ➢ This function accepts one character i.e, first character from keyboard buffer and remaining characters will be left in buffer.

Ex:

```
#include<stdionh>
void main()
{
char ch;
ch=getchar();
}
```

Output:

H
ch contains H

putchar():

- ➢ Displays character stored in memory location on the screen.

Syntax: **putchar();**

Note:

- ✓ An EOF (end of file) indicates when no more data can be read. If EOF is encountered it returns -1. We can use ctrl+z to encounter EOF.

Write a C program to read and display a character.

```
#include<stdio.h>
void main()
{
    char ch;   //int ch;
    ch=getchar();
    putchar(ch);
}
```

Output:

Input: a   *Output:* a

Note:

- ✓ Generally the data type in getchar/putchar function, the variable is declared as integer because, if EOF is encountered it returns -1 which is not a character.

Other functions: getch(), getche() and putch().

- ✓ These functions are available in conio.h header file

getch() / getche():

- ✓ Reads character from keyboard without waiting for user to press "enter key"

Syntax : **ch=getch();**

// Typed character will not be displayed

Syntax : **ch=getche();**

// Typed character will be displayed on screen

- ➢ No arguments required for these functions
- ➢ It doesn't waits for user to press "enter key"

putch():

- ✓ Displays a character stored in memory location identified by variable ch on screen.

Syntax: **putch();**

Write a C program to read and display a character.

```
#include<conio.h>
 void main()
   {
         char ch;
         ch=getche();
         putch(ch);
   }
```

Testing the type of character & Built in Character functions-

- ✓ There are built in functions in C to check whether a character in digit, alphabet etc...
- ✓ CTYPE functions fit into two categories:
  - ➢ Testing
  - ➢ Manipulation.
- ✓ All character oriented functions were defined under ctype.h header file.
- ✓ Some of the function with description was listed in the table.

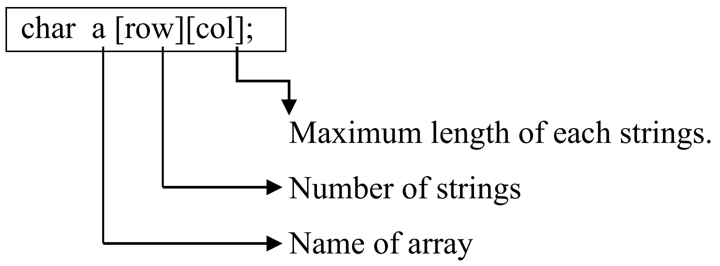
Character Testing Functions	
Function	**Returns TRUE When *ch* is**
isalnum(*ch*)	A letter of the alphabet (upper- or lowercase) or a number
isalpha(*ch*)	An upper- or lowercase letter of the alphabet
isascii(*ch*)	An ASCII value in the range of 0 through 127
isblank(*ch*)	A tab or space or another blank character
iscntrl(*ch*)	A control code character, values 0 through 31 and 127
isdigit(*ch*)	A character 0 through 9
isgraph(*ch*)	Any printable character except for the space
ishexnumber(*ch*)	Any hexadecimal digit, 0 through 9 or A through F (upper- or lowercase)
islower(*ch*)	A lowercase letter of the alphabet, *a* to *z*
isnumber(*ch*)	*See* isdigit()
isprint(*ch*)	Any character that can be displayed, including the space
ispunct(*ch*)	A punctuation symbol
isspace(*ch*)	A white-space character, space, tab, form feed, or an Enter, for example
isupper(*ch*)	An uppercase letter of the alphabet, *A* to *Z*
isxdigit(*ch*)	*See* ishexnumber()
Character Manipulation Functions	
Function	**Returns**
toascii(*ch*)	The ASCII code value of ch, in the range of 0 through 127
tolower(*ch*)	The lowercase of character *ch*
toupper(*ch*)	The uppercase of character *ch*

Write a C program to count number of alphabets, digits, spaces and punctuations.

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

         void main ()
         {
             int ch;

             int nalpha=0,ndigits=0,nspaces=0,npunct=0;

             clrscr();

                   while(ch=getchar()!=EOF)

                   {

                        getchar();


                        if(isalpha(ch))

                        nalpha++;


                        else if(isdigit(ch))

                        ndigits++;


                        else if(isspace(ch))

                        nspaces++;


                        else if(ispunct(ch))

                        npunct++;

                   }

              printf("\n Number of digits = %d",ndigits);

              printf("\n Number of spaces = %d",nspaces);

              printf("\n Number of alphabets = %d",nalpha);

              printf("\n Number of punctuations = %d",npunct);

             getch();

         }
```

Array of String:

- ✓ To declare array of strings we should use two dimensional arrays as follows-



Ex: char name[5][10]= {
"satya",
"dharma",
"kartavya",
"seva"
"bhakthi"
};

This is stored in memory as

name	0	1	2	3	4	5	6	7	8	9
0	s	a	t	y	a	\0				
1	d	h	a	r	m	a	\0			
2	k	a	r	t	a	v	y	a	\0	
3	s	e	v	a	\0					
4	b	h	a	k	t	h	i	\0		

We can access the individual name as name[0],name[1],name[2],name[3],name[4].

Write a program to read and print array of name.

```
#include<stdio.h>
   void main()
      {
          char name[10][10];
          int i,n;
          printf("\n enter the number of names");
          scanf("%d",&n);
          printf("enter the names");
          for(i=0;i<n;i++)
          scanf("%s",name[i]);
          printf("entered names are:\n");
          for(i=0;i<n;i++
          printf("%s",name[i]);
         getch();
      }
```