Strings

String:- **String is a sequence of characters terminated with a null character \0.**

Declaration of string:-

char string_name[size];

Example:-

char s[5];

Initialize String:-

You can initialize strings in a number of ways.

char c[] = "ajay";

char c[50] = "ajay";

char c[] = {'a', 'j', 'a', 'y', '\0'};

char c[5] = {'a', 'j', 'a', 'y', '\0'};

c[0]	c[1]	c[2]	c[3]	c[4]
a	j	a	y	\0

getchar():- **The simplest of all input/output operations is reading a character from the standard input unit(usually the keyboard) and writing it to the standard output unit(usually the screen). Reading a single character can be done by using the function getchar.**

Syntex:-

int getchar(void);

it returns the next character from the standard input (stdin). It is equivalent to getc with stdin as its argument.

- ***Parameters***

  (none)

- ***Return value:-*** The character read is returned as an int value.

Example :- write a program to use of getchar() function.

Solution:-

```
#include<stdio.h>
int main()
{
char c;
puts("Enter text. Include a dot('.') in a sentence to exit:");
do
        {
                c=getchar();
                putchar(c);
        }
        while(c!='.');
        return 0;
}
```

Output:-

```
Enter text. Include a dot('.') in a sentence to exit:      //First time run
Ram is a poor farmer.hfghhhgh
Ram is a poor farmer.
```

Putchar():- **Like getchar, there is an analogous function putchar for writing characters one at a time to the nonitor.**

Syntex:-

```
int putchar(int character);
```

This function wites character to the current position in the standard output(stdout) and advances the internal file position indicator to the next position. It is equivalent to putc(character., stdout).

- ***Parameters***

Character:- **character to be written. The character is passed as its int promotion.**

- ***Return value***

  If there are no errors, the same character that has been written is returned.

Example:- Write a program to the use of putchar() function.

Solution:-

```
#include<stdio.h>
int main()
{
    char c;
    for(c='A';c<='Z';c++)
    {
        putchar(c);
    }
```

```
    return 0;
}
```

Output:-

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

puts():-**The put() function writes the string to standard output device(monitor) and appends a newline character to the output.**

Syntax:-

```
int puts(const char *str);
```

It writes the C string pointed by str to stdout(screen) and appends a newline character("\n").

The function begins copying from the address specified (str) until it reaches the terminating null character('\n'). This final null-character is not copied to stdout.

- ***Parameters***

str

It is a C string to be written.

- ***Return Value***

On success, a non-negative value is returned.

Example:- ***Write a program to the use of puts() function.***

Solution:-

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

```
{
        char string[]="Hello sir!";
        puts(string);
        return 0;
}
```

Output:-

```
Hello sir!
```

gets():- **The gets() function reads data from the standard input device (keyboard), until a newline is read or an end-of-file condition is encountered.**

Syntax:-

```
char   *gets(char *str);
```

It reads characters from stdin(keyboard) and stores them as a string into str until a newline character('\n') is reached.

The ending newline character('\n') is not included in the string. A null character ('\0') is automatically appended after the last character copied to str to signal the end of the c string.

- ***Parameters***

str

Pointer to an array of chars where the C string is stored.

- ***Return Value***

On success, the function returns the same str parameter.

Example:- ***Write a program to the use of gets() function.***

Solution:-

```
#include<stdio.h>
int main()
{
        char string[256];
        printf("Insert your full address:\n");
        gets(string);
        printf("Your address is %s\n",string);
        return 0;
}
```

Output:-

```
Insert your full address:
Gali no 3 guna.
Your address is Gali no 3 guna.
```

String Operation

String operation:- **A large set of string handling library functions are provided with every C compiler. The following table shows the list of string handling functions with their use.**

1. ***strlen():-*** **strlen() is function which is used to find the length of a string passed as an argument. Strlen() returns the length of a string. Executing the null character.**

***Example*:- Write a program to use strlen() function.**

Solution:-

```
#include<stdio.h>
#include<string.h>
int main()
{
char s[256];
printf("Enter a sentence: ");
gets(s);
printf("The sentence entered is %u characters long. \n",strlen(s));
   return 0;
}
```

Output:-

```
Enter a sentence: Hello sir
The sentence entered is 9 characters long.
```

2. ***strcat():-*** **strcat() is used to concatenate a null-terminated string to end of another string variable this is equivalent to pasting one string onto the end of another, overwriting the null character.**

Example:- **Write a program to use strcat() function.**

Solution:-

```
#include<stdio.h>
#include<string.h>
int main()
{
        char str[80];
        strcpy(str,"These ");
        strcat(str,"strings ");
        strcat(str,"are ");
        strcat(str,"concatenated.");
        puts(str);
        return 0;
}
```

Output:-

```
These strings are concatenated.
```

3. ***strcpy():-*****strcpy() is used to copy a null-terminated string into a variable.**

Example:-**Write a program to use strcpy() function.**

Solution:-

```
#include<stdio.h>

#include<string.h>

int main()

{
```

```
        char s1[256];
        char s2[256];
        char s3[265];
        printf("Enter a sentence: ");
        gets(s1);
        strcpy(s2,s1);
        strcpy(s3,"copy successful");
        printf("s1: %s\ns2: %s\ns3: %s\n",s1,s2,s3);
        return 0;
    }
```

Output:-

```
Enter a sentence: hello sir
S1: hello sir
S2: hello sir
S3: copy successful
```

4. ***strcmp():-***strcmp() is used to compare two strings. The strings are compared character by character starting at the characters pointed at by the two pointers.
   It compares the C string str1 to the C string str2. This function starts comparing the first character of each string. If they are equal to each other, it continues with the following pairs until the characters differ or until a terminating null character is reached.

- ***Parameters***

Str1

It is a C string to be compared.

Str2

It is a C string to be compared.

- ***Return value***

Returns an integral value indicating the relationship between the strings: A zero value indicates that both strings are equal. A value greater than zero indicates that the first character that does not match has a greater value in str1 the in str2; And a value less than zero indicates the opposite.

Example:- **Write a program to use strcmp() function.**

Solution:-

```
#include<stdio.h>
#include<string.h>
int main()
{
    char skey[]="apple";
    char sipt[80];
    do
    {
        printf("Guess my favourite fruit? ");
        gets(sipt);
    }while(strcmp(skey,sipt) != 0);
    puts("Correct answer!");
```

```
        return 0;
    }
```

Output:-

```
Guess my favourite fruit? Orange
Guess my favourite fruit? Apple
Correct answer!
```

5. ***Strncmp():-*** **it is used ot compare the first given number of characters of two strings. The strings are compared character by character starting at the characters pointed at by the two pointers.**

Example:- **write a program to use strncmp() function.**

Solution:-

```
#include<stdio.h>
#include<string.h>
int main()
{
    char s1[25]="pat";
    char s2[25]="pet";
    int diff;
    diff=strncmp(s1, s2, 3);
    printf("%d",diff);
    diff=strncmp(s2, s1, 3);
```

```
    printf("\n%d",diff);
    diff=strncmp(s1,s2, 1);
    printf("\n%d",diff);
return 0;
}
```

Output:-

```
-1
1
0
```

Character handling Functions

Character handling Functions:- **The character handling function header declares asset of functions to classify and transform individual characters. All these functions take as parameter the int equivalent of one character and return an int, that can either be another character or a value representing a boolean value: an int value of 0 means false, and an int value different from 0 represents true.**

1. ***isalnum():-*** **It is used to check whether a character is alphanume ric or not.**

Example:-write a program to use isalnum() function.

Solution:-

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

```
int main()
{
        int m;
        char str[]="c3po67...";
        m=0;
        while(isalnum(str[m]))
        m++;
        printf("The first %d characters are alphanumeric.\n",m);

        return 0;
}
```

Output:-

```
The first 6 characters are alphanumeric.
```

Isalpha:- **It is used to check whether a character is alphabet or not.**

Example:- **write a program to use isalpha() function.**

Solution:-

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

```
    int i=0;
    char str[]="c++";
    while(str[i])
    {
        if(isalpha(str[i]))
        printf("Character %c is alphabetic \n",str[i]);
        else
        printf("Character %c is not alphabetic \n",str[i]);
        i++;
    }
    return 0;
}
```

Output:-

```
Character C is alphabetic
Character + is not alphabetic
Character + is not alphabetic
```

iscntrl():- **It is used to check whether a character is a control character or not.**

Example:- **write a program to use iscntrl() function.**

Solution:-

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

```
#include<ctype.h>
int main()
{
        int i=0;
        char str[]="First brother\n second line";
        while(!iscntrl(str[i]))
        {
                putchar(str[i]);
                                i++;
        }
        return 0;
}
```

Output:-

```
First brother
```

isdigit():-It is used to check whether a character is decimal digit or not.

Example:-**write a program to use isdigit() function.**

Solution:-

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

```
int main()
{
        char str[]="1234hf";
        int year;
        if(isdigit(str[0]))
        {
                year=atoi(str);
printf("The year that followed %d was%d.\n",year,year+1);

        }
        return 0;
}
```

Output:-

```
The year that followed 1234 was1235.
```

Isgraph():- **It is used to check whether a character has graphical representation or not.**

Example:- white a program to use isgraph() function.

Solution:-

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

```
int main()
{
        char x;
        printf("Enter any character: ");
        scanf("%d",&x);
        if(isgraph(x))
        {
                printf("%d",x);

        }
        else
        {
                printf("Sorry, could not print character");
        }
        return 0;
}
```

Output:-

```
Enter any character: p
Sorry, could not print character
```

islower:-It us used to check whether a character is lowercase character or not.

Example:- **write a program to use islower() function.**

Solution:-

```
#include<stdio.h>
#include<ctype.h>
int main()
{
    int i=0;
    char str[]="Test String.\n";
    char d;
    while(str[i])
    {
        d=str[i];
        if(islower(d))d=toupper(d);
        putchar(d);
        i++;
    }
    return 0;
}
```

Output:-

```
TEST STRING.
```

ispuct():- **It is used to check whether a character is punctuation character or not.**

Example:- **write a program to use ispuct() function.**

Solution:-

```
#include<stdio.h>
#include<ctype.h>
int main()
{
    int cx=0,i=0;
    char str[]="hello sir, hello mam!\n";
    while(str[i])
    {
        if(ispunct(str[i]))cx++;
        i++;
    }
    printf("Sentence contains %d punctuation characters.\n", cx);
    return 0;
}
```

Output:-

```
Sentence contains 2 punctuation characters.
```

isspace:- **it is used to check whether a character is white-space or not.**

Example:- **write a program to use isspace() function.**

Solution:-

```
#include<stdio.h>
#include<ctype.h>
int main()
{
    int c,i=0;
    char str[]="Hello Sir \n";
    while(str[i])
    {
        c=str[i];
        if(isspace(c))c='\n';
        putchar(c);
        i++;
    }
    return 0;
}
```

Output:-

```
Hello
Sir
```

isupper():- **it us used to check whether a character is uppercase character or not.**

Example:-**write a program to use isupper() function.**

Solution:-

```
#include<stdio.h>
#include<ctype.h>
int main()
{
        int i=0;
        char str[]="Test StrIng.\n";
        char d;
        while(str[i])
        {
                d=str[i];
                if(isupper(d))d=tolower(d);
                putchar(d);
                i++;
        }
        return 0;
}
```

Output:-

```
test string.
```

tolower():-it is used to convert an uppercase character into lowercase character.

Example:-**write a program to use tolower() function.**

Solution:-

```
#include<stdio.h>
#include<ctype.h>
int main()
{
    int i=0;
    char str[]="Test String.\n";
    char d;
    while(str[i])
    {
        d=str[i];
        putchar(tolower(d));
        i++;
    }
    return 0;
}
```

Output:-

```
test string.
```

toupper():-it is used to convert lowercase character into uppercase character.

Example:-**write a program to use toupper() function.**

Solution:-

```
#include<stdio.h>
#include<ctype.h>
int main()
{
    int i=0;
    char str[]="Test String.\n";
    char d;
    while(str[i])
    {
        d=str[i];
        putchar(toupper(d));
        i++;
    }
    return 0;
```

```
}
```

Output:-

```
Test string.
```

Created by ajay kumar verma