FUNCTIONS

A function is a named unit of a group of statement designed to perform a specific task and return a single value. Or we can also define as a named group of statement developed to solve a sub problem and is return a value to other function when it is called We have two types of function and they are

- Library function
- User defined function

Library function: it is the collection of predefined functions supplied along with the compiler. We can access it by using proper header file.

Header file : A file contain all definition required for the library functions used in the program . Example iomanip.h , iostream.h
We can the header file by using #include directive and the name of the file to be included inside the angle bracket.

User-defined function: A function defined by the user to perform a specific task. This function is invoked by using the function call.

The different types of header file

1. stdio.h : This header file contain function and macros to perform standard I/O operation. Under this header file we have the following functions

   Functions

clearerr	fclose	fcloseall	fdopen	feof	ferror
fflush	fgetc	fgetchar	fgetpos	fgets	fileno
flushall	fopen	fprintf	fputc	fputchar	fputs
fread	freopen	fscanf	fseek	fsetpos	ftell
fwrite	getc	getchar	gets	getw	perror
printf	putc	putchar	puts	putw	remove
rename	rewind	rmtmp	scanf	setbuf	setvbuf
sprintf	sscanf	strerror	_strerror	tempnam	tmpfile
tmpnam	ungetc	unlink	vfprintf	vfscanf	vprintf
vscanf	vsprintf	vsscanf			

2. stdlib.h This header file is used to declare conversion routines search and sort routines. Under this header file we have the following functions

abort	abs	atexit	atof	atoi
atol	bsearch	calloc	div	ecvt
exit	_exit	fcvt	free	_fullpath
gcvt	getenv	itoa	labs	ldiv
lfind	_lrotl	_lrotr	lsearch	ltoa
_makepath	malloc	max	mblen	mbtowc
mbstowcs	min	putenv	qsort	rand
random	randomize	realloc	_rotl	_rotr

_searchenv	_splitpath	srand	strtod	strtol
_strtold	strtoul	swab	system	time
ultoa	wctomb	wcstombs		

Explanation of some functions

abort	terminates program execution abnormally
abs	returns absolute value
exit	terminate the program execution
rend()	generate the random number between 0 to RAND_MAX (it is a constant)

3. iostream.h this header file contains c+ streams and I/O routines. Under this header file we have the following functions

open	close	get	getline	read	write	put	seekg	tellg	seekp
bad	eof	fail	rdstate	good	clear				

Explanation of some function

bad	returns true if an invalid or unrecoverable error is occurred
clear	clears the error state
good	returns true if no error is occurred
eof	returns true if end of file is encountered
seekp	moves output pointer to a specific location
write	writing binary data into a disk file
open	create a new file and open existing file

4. iomanip.h : This header file contain functions and macros for I/O manipulation. Under this header file we have the following functions

setw(n)	sets the field width to integer n
setfill(f)	sets fill character to f
ws	extracts white space (blank) characters on input stream
dec	tells the subsequent operation to use decimal representation
hex	tells subsequent I/O operation to use hexadecimal representation
oct	tells subsequent I/O operation to use octal representation
flush	flushes (clears) an output stream
endl	insert new line character '\n' and flushes output stream
ends	insert null character('\0') in an output stream

Note: when we include iostream.h header file the stdio.h is automatically included

5. math.h : This header file declares prototype of the mathematical function and error handler . The some functions are
   log   sqrt   pow   tan   sin   cos   fabs   exp divmodf

6. ctype.h: This header file contain the character related function. The character is any single value enclosed in single quote The character related function will have the form
   int function_name (int character);

The some functions are

The character manipulation function are of two types and are classification and conversion. The **classification** function is always begins with keyword **is** and the **conversion** function always begins with keyword **to**

Classification function: These functions are used to identify and it return the values as 1 for true and 0 for false

1) isupper() This function return 1(true) if the enter character is upper case otherwise give 0(false)
   Syntax: isupper(character);
   Ex: isupper(‘A’);   gives 1
   isupper(‘a’); gives 0
2) islower() : This function return 1(true) if the enter character is lower case otherwise give 0(false)
   Syntax: islower(character);
   Ex: islower(‘a’); gives 1
3) isspace(): This function gives 1 if the entered character is space, carriage return, form feed , vertical tab, tab, new-line otherwise return 0
   syntax: isspace(character);
   isspace(‘a’); will gives 0
4) isprint(): This function will return 1 if the character is printable character else return false
   syntax: isprint(character);
   Ex: isprint( ); gives true
5) isdigit(): This function return nonzero if character is a digit else return zero
   syntax: isdigt(character);
   Ex: isdigit(‘6’); gives 1(true)
6) isascii(): Identify whether the character is an ASCII (0-127)character or not
   syntax: isascii(character);
   Ex:isascii(‘!’); will gives 0   isascii(‘f’) will gives1
7) isalpha(): return non zero(1) if character is alphabet
   syntax: isalpha(character);
   isalpha(‘3’); will gives 0
8) iscntrl(): used to test whether the given character is control character or not
   syntax: iscntrl(character);
9) isalnum(): Test whether a character is an alphabet or a number. If the character is number or alphabet then gives 1 else 0
   syntax: isalnum(character);
   Ex: isalnum(‘8’); gives 1
   isalnum(‘w’); gives1
   isalnum(‘*’); gives 0
10) isxdigit(): this function gives 1 if the character is hexadecimal digit
   syntax: isxdigit(character);

Conversion Function: It is used to convert from one type to other

1) toascii() : translate character to ASCII
   syntax: toascii(character);
   Ex: toascii(‘A’); gives 65

2) tolower(): translate the character to lower case
   syntax: tolower(character);
   Ex: tolower('A') will gives a
   tolower('d'); will gives d
3) toupper(): convert the character from lowercase to uppercase
   syntax: toupper(character);
   Ex: toupper('f'); gives F

7. string.h : This header file contain the some of the string related functions are

a) strlen() function

This function counts and returns the number of characters in a string. This functions does not include the NULL character. This function return the integer value

Syntax: n= strlen(str);

Where n is integer variable which receives the length of string variable str

Ex: length=strlen("sms");

If we print length we get the output as 3

b)strcpy() function

This function is used to copy one string to other with following syntax

Syntax: strcpy(string1, string2);

Copy the content of string2 to string1

Ex: strcpy(college, "sms");

In the above example the college will get the value sms

c)strcat() function

This function add the 2 strings this process can also be called as concatenation.

Syntax: strcat(string1,string2);

The string2 will add to the end of the string1. The string2 will remains unchanged but the string1 will modifies

d) strcmp() function

This function is used to compare the 2 strings with following syntax

Syntax: strcmp(string1, string2);

The string1 will compare with string2, this comparison will perform based on ASCII value and it return the following 3 possible values

Return positive if first string (string1) is alphabetically greater than second (string2)

Return negative value if first string is (string1) is alphabetically less than second (string2)

Return zero if both strings are equal

Ex: strcmp("sms", "SMS"); will return positive value

e)strlwr() function

This function converts all upper case character into an equivalent lower case character

Syntax: strlwr(string);

Ex: strlwr("Abc"); will give the output as abc

f) strupr() function :

This function converts all lower case character into an equivalent upper case character

Syntax: strupr(string);

Ex: strupr("Abc"); will give the output as ABC

g)strrev() function:

This function reverse the character in the string
Syntax: strrev(string);
Ex strrev("PUC"); will give the output as CUP
h)strncmp() function
This function compares the first n character of the 2 input strings
Syntax: strncmp(string1, string2,n);
Where n is integer
I) strcmpi() function
This function is same as strcmp() but it ignore the case i.e. compare the 2 string but is not case sensitive
Ex: strcmpi("HAI", "hai"); will return zero
j) strncpy() function
This function copies first n characters of the second string to the first string its syntax is
Syntax: strncpy(string1,string2,n);
Ex: strncpy(string1, "sms",1);
If we print string1 we get output as s
k)strncat() function
This function appends first n characters of second string to the end of the first string
Syntax: strncat(string1, string2, n);
l) strchr() function
This function search for a specified character in the string. It returns NULL if the desired character is not found in the string
Syntax: strchr(string, char-to-search);
Ex: strchr("sms", s);
Since s is present in sms . The search for the character s is successful

Strings
The definition of string would be anything that contains more than one character in side of double quote.
OR
We can also define the string as group of character enclosed within side of double quote and always terminated by null character.
Example: "This is C plus program"
Declaration
Syntax : Char string-name[max-size-of char];
Ex: char name[20];
This would declare a string variable name with length 20 characters. Do not forget that the array index begins with zero not from one. In addition the string ends with the null ('\0) character. Remember that it is a extra character added at the end, how we add the full stop at the end of each sentence.. In 20 character the name will hold only 19 character and one will be used to store the null character at the end which is needed to terminate the string.
Note: Null character represented by back slash zero
Initializing a string: If we assign the value to the string variable at the time of declaration is called initialization of string. We can achieve this any of the following

ways
1) char str[10]=”Karnataka”;
In the above string initialization the null character is automatically appends to the end of the string
2) char str[]=”Karnataka”;
In the above string initialization the null character is automatically appends to the end of the string and the maximum size is automatically determine by the compiler
3) We can also initialize the string character by character by the following ways and we have to add the null character at the end manually.
char str[10]={ ‘k’, ‘a’ , ‘r’ , ‘n’ , ‘a’, ‘t’ ,’a’ ,‘k’ ,’a’,’\0’};
4) char str[7]= “book”;
In the above string initialization string is initialized with 4 character and remaining 3 location initialized with null character automatically.
Inputting a string: We can read the value to the string by using two method.
If we use **cin>>** to input a string it works but it will terminate the string after it reads the first space.
The best way to handle this situation is by using the function **cin.getline**
The syntax of the getline function is **cin.getline(str , size);**
where str is the string to be read.
the size is the number of character that we want to read
The getline is a function is used to read to one entire line of character or the number of character is equal to zero or till the newline character is read
example : cin.getline(str,20);
or
cin.getline(string,200,’\n’);

Outputting a string: we can print the value of sting by two method. One method is like printing of a normal variable and other one is by using write function with following syntax
cout.write(str , size);
where str is a string variable to print
size is the number of the character to be print .
This function will not stop displaying character when encountering the null character. If the specified size is greater than length of the string it is displayed beyond the limit
Note: the write() function does not stop displaying the character when encountering the null character

Example : program to read a string and print that string along with the length
```
#include <iostream.h>
#include <conio.h>
#include <string.h>

void main()
{
clrscr();
int slength;
char x[81]; //Allowing the user to input a maximum of 80 characters.
cout << "Enter the string : " << endl;
cin>>x;
slength=strlen(x);
cout << "The length of the string " << x << " is " << slength << "."
<< endl;
getch();
}
```

Sample output

```
Enter the string :
santhosh
The length of the string santhosh is 8.
```

Note: Using the above input statement it is not possible to read blank space given while entering the string. In the below example the Neelavar is not considered

Sample output

```
Enter the string :
santhosh Neelavar
The length of the string santhosh is 8.
```

Example 2: The program to input a sting and print its length

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

void main()
{
clrscr();
int slength;
char x[81]; //Allowing the user to input a maximum of 80 characters.
cout << "Enter the string : " << endl;
cin.getline(x,80);
slength=strlen(x);
cout << "The length of the string " << x << " is " << slength << "."
<< endl;
getch();
}
```

sample output

```
Enter the string :
santhosh neelavar
The length of the string santhosh neelavar is 17.
```

Note: The blank space and special character is also considered while finding length

Inputting single character: we can input a character using the function get()
The general form is

```
char ch;
Ch=cin.get(ch);
```

Outputting single character: The put()function is used to display a single character. The general format is **cout.put(ch);**
Example:

```
char ch;
ch=cin.get();
cout.put(ch);
```