FUNCTION

Defination: A function is a self contained block of statements written in C language.

Syntax

```
return_type function_name( parameter list )
{
  body of the function
}
```

- **Return Type** − A function may return a value. The **return_type** is the data type of the value the function returns. When function does not return any thing **void**.
- **Function Name** − This is the actual name of the function.
- **Parameters** − When a function is called, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function.
- **Function Body** − The function body contains a collection of statements that define what the function does.

Example

Function without parameter

```
void  show ()
{
   printf("\n This is show Function");
}
```

Function with parameter

```
void add(int a, int b)
{
   int c;
  c=a+b;
}
```

Calling a Function

To use a function, you will have to call that function to perform the defined task. To call a function, you simply need to pass the required parameters along with the function name, and if the function returns a value, then you can store the returned value.

Above functions can be called as

```
show();
add(10.20);
```

There are two types of calling a function

1) Call by Value
2) Call by Reference

- **Call by Value**

  This method copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument.

```
void change (int a)
{
   a=a+10;
}
int a=100;
change(a);   //  call by value
```

here the value of a=100 only after calling the function change.

- **Call by Reference**

  This method copies the address of an argument into the formal parameter of the function.

```
void change (int *a)
{
    *a=*a+10;
}

int a=100;
change(&a);  // call by Reference
```

here the value of a=110 only calling the function change.

Types of Function

1) Function with no return value and no parameters.

```
void show()
{
}
```

2) Function with no return value and with one or more arguments.

```
void show (int a)
{
}
```

3) Function with return value and no parameters.

```
int show()
{
}
```

4) Function with return value and with one or more parameters

```
int show( int a, char ch)
{
}
```

Local Variables

Variables that are declared inside a function or block are called local variables. They can be used only by statements that are inside that function or block of code.

Global Variables

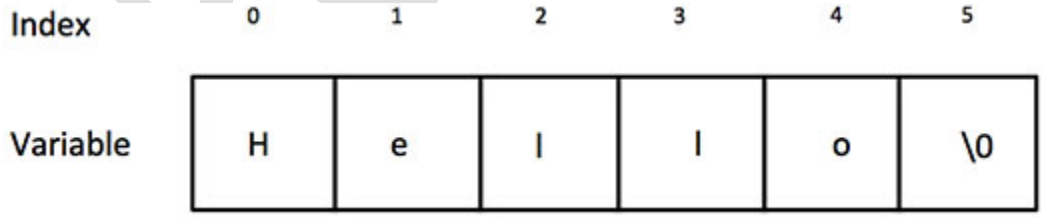
Global variables are defined outside a function, usually on top of the program. Global variables hold their values throughout the lifetime of your program and they can be accessed inside any of the functions defined for the program.

```
#include <stdio.h>
 /* global variable declaration */
int g;

void main ()
{
 /* local variable declaration */
 int a, b;

 /* actual initialization */
 a = 10;
 b = 20;
}
```

Strings

Def : A string is collection of characters included in double quotes.

Strings are actually one-dimensional array of characters terminated by a **null** character '\0'.




```
#include <stdio.h>

 void main ()
 {

   char msg[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
   printf("%s\n", msg );
```

Function

}

Standard String Functions

No.	Function	Purpose
1	strcpy(s1, s2);	Copies string s2 into string s1.
2	strcat(s1, s2);	Concatenates string s2 onto the end of string s1.
3	strlen(s1);	Returns the length of string s1.
4	strcmp(s1, s2);	Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.

Storage Classes

A storage class defines the scope and life-time of variables or functions within a C Program. 'auto' can only be used within functions, i.e., local variables.

Four types of storage classes in C

- auto
- register
- static
- extern

1) **auto storage class**

It is a default storage class for all variable.

2) **register storage class**

The **register** storage class is used to define local variables that should be stored in a register instead of RAM.T he register should only be used for variables that require quick access such as counters.

3) **static storage class**

A `static` variable tells the compiler to persist/save the variable until the end of program. Its default initial value is zero.

static int count = 5;

4)**extern storage class**

The **extern** storage class is used to give a reference of a global variable that is visible to ALL the program files.

extern int count;

Function