|  |
| --- |
| **Functions**  Functions – declaring, defining, and accessing functions –parameter passing methods – – passing arrays to functions , Recursion . Storage classes – extern, auto, register and static. Example programs. |



**Function**

A function is a set of instructions that works together to perform a task.every program consist of at least one function,which is the main() function.we can use multiple no.of functions in a single program based on our need.

**Advantages of using function**

* Functions are reusable **:** we can call and execute a function many times from any where in the program
* Reduce repeating of instructions
* Reduce complexity of program
* Reduce size of program.
* It allows division of labour while program development

Functions are mainly classified into two

1. Library or standard functions
2. User defined functions

**Library functions**

Functions that are already defined in c language compiler is known asLibrary or standard functions .All these Library functions are stored in library files of c language.programmer can use this functions anywhere in program by using function call.but programmer must include the corresponding header file in program that contain function need to be used.

Some header files are ,Stdio.h ,Conio.h ,String.h etc

In order to access a library function,programmer must use instruction “#include<header file name>” Before the main function.

Eg: #include<stdio.h>

Some of the standard or library functions are

|  |  |  |  |
| --- | --- | --- | --- |
| **function** | **type** | **purpose** | **Header file** |
| Scanf() | int | Allow input of data from standard input device | Stdio.h |
| Printf() | int | Send data to standard output device | Stdio.h |
| Strcmp(s1,s2) | int | Compare two strings and return 0 if s1=s2,+ve value if s1>s2,-ve value if s1<s2. | String.h |
| Strcpy(s1,s2) | int | Compare strings s1 and s2 | String.h |
| Clrscr() |  | Clear screen after each execution of program | Conio.h |
| getch() |  | Get output on screen | Conio.h |
| Fclose(f) |  | Close a file | Stdio.h |

**(Refer tp for more examples )**

**User defined functions**

User defined functions are functions created by programmer based on their needs.one can use user defined functions as library functions where ever they want.

User defined functions can be modified based on our need.user defined functions are used when we have to use same set of instructions again and again.

This will give advantages like

* Reduce repeating of instructions
* Reduce complexity of program
* Reduce size of program.
* It allows division of labour while program development

Eg:

**Int max(int x,int y)**

**{**

**Int sum;**

**Sum=x+y;**

**Return sum;**

**}**

This is user defined function to add two integers.

|  |  |
| --- | --- |
| **User defined functions** | **Library functions** |
| Functions created by user | Pre written function present in c library |
| User can modify user defined function | User can’t modify library functions |
| User defined functions are part of program. | library functions are part of header files like  Stdio.h,conio.h etc |
| User defined functions Compiled run time | library functions Called run time |

## How user-defined function works?

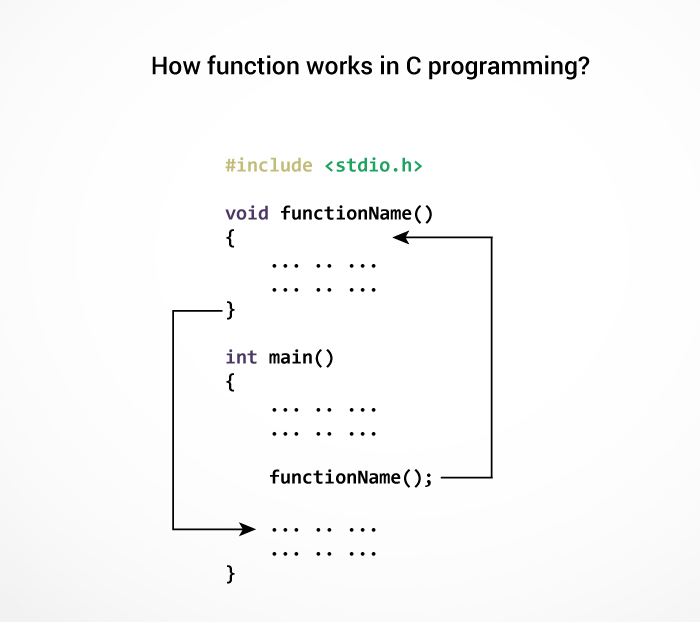
The execution of a C program begins from the main() function.

When the compiler encounters function Name();, control of the program jumps to

void function Name()

And, the compiler starts executing the codes inside function Name().

The control of the program jumps back to the main() function once code inside the function definition is executed.



## Basics of function

## Defining a Function (ie,we are creating a function)

We can a define a function in C programming language using the following syntax

**Syntax:**

**Return type function name (list of parameters)**

**{**

**Body of function**

**}**

* A function definition in C programming consists of a ***function header*** and **a *function body***.
* **“Return type function name (list of parameters)”** is the function header.
* Body of the function consist of no.of instructions need to be executed.

**Here are all the parts of a function −**

**Return Type** − A function generally return a value to main function (which is using to inform that the task given to that function is completed successfully). The **return\_type** is the data type of the value the function returns. It may be int,char etc.(in this case main() function should consist of a variable to store the returned value).Some functions perform the desired operations without returning a value. In this case, the return\_type is the keyword **void**.

Eg:.

Consider some function declarations below

* **int function name (parameter list)**

**{**

**Body of function**

**}**

This function will return an integer value to main function.

* **char function name (parameter list)**

**{**

**Body of function**

**}**

This function will return a character value to main function.

* **void function name (parameter list)**

**{**

**Body of function**

**}**

This function won’t return anything to main function.

**Function Name** − This is the actual name of the function.this can be anything The function name and the parameter list together constitute the function signature.

**Parameters or arguments** − Parameters or arguments are the special kind variables using in a function in which the data need to be processed is stored. There are two type of arguments in c language.they are **actual parameter or argument and formal parameters or arguments.**

The parameter list refers to the date type, order, and number of the parameters of a function. Parameters are optional; that is, a **function may contain no parameters**.

**Function Body** − The function body contains a collection of statements that define what the function does.

**Eg:**

Given below is the source code for a function called **max()**. This function takes two parameters “**x**” and “**y**” and add them .then it will returns an integer value to main function.

headerr

**Int max(int x,int y)**

**{**

Local variable declaration

**Int sum;**

**Sum=x+y;**

**Return sum;**

**}**

**Note**

**A user defined function can be defined after main() function or before main function**

## Function Declarations / function prototyping

A function **declaration** **or prototyping** specifies the data type and list parameters using in a function.(it does not contain any kind of codes,there is only a header of a function).

compiler use the information from function declaration to ensure that return type ,data type and list of parameters are same as in function definition.other wise compiler will show error.

**Syntax:**

**Return type function name (list of parameters);**

**Note**

**Function declaration is required when you define a function in one source file and you call that function in another file. In such case, you should declare the function at the top of the file calling the function**.

**Eg:**

For the above defined function max(), the function declaration is as follows

**Int max(int x,int y);**

Parameter names are not important in function declaration only their type is required, so the following is also a valid declaration

**Int max(int,int);**

## Function Arguments or parameter

Parameters or arguments are the special kind variables using in a function in which the data need to be processed is stored.

In c language,there is two type of arguments

1. **Actual parameters or actual arguments**

If any portion of a program(it can be main function or others) calls a function,then parameters present in that calling portion are called actual Parameters.the actual data is need to be processed is present in this parameters.

1. **formal parameters or formal arguments**

The parameters written in function definition are known as formal parameters or formal arguments.it is also known as dummy parameters.

Formal parameters behave like other local variables inside the function and are created upon entry into the function and destroyed upon exit.

Eg:

**#include <stdio.h>**

**#include <conio.h>**

**int max(int a,int b)**

**{**

**Int s;**

**If (a>b)**

**{**

**Printf(“hello\n”);**

**}**

**Else**

**{**

**Printf(“hai\n”);**

**}**

**s=a+b;**

**Return s;**

**}**

**Void main()**

**{**

**Clrscr();**

**Int x,y,r;**

**Printf(“enter numbers\n”);**

**Scanf(“%d%d”,&x,&y);**

**r=Max(x,y);**

**Printf(“%d\n”,r);**

**getch();**

**}**

Here main function consist of parameters,they are “x” and “y”.these parameters are called as actual parameters.

Parameters present in function max( ) is “a” and “b” these are known as formal parameters.

## parameter passing methods

Consider a situation In which we have some data need to be processed in our actual parameters and we want to process this data using a function.

In order to do this we have to pass data from actual parameters to formal parameters ,this is known as parameter passing.this should done along with function call.we can do this using two methods.

1. **Call by value**
2. **Call by reference**

**Note**

**In order to pass pass data actual argument must be match with formal arguments,in terms of data type,order and number**

While calling a function, there are two ways in which arguments can be passed to a function

1. **Call by value**

In this method of parameter passing,copies of the actual parameter values are transferred into the formal parameter of the function. Data of actual and formal parameters are stored in different location.In this case, manipulation of the formal parameter inside the function have no effect on the actual parameters as it the copy of actual parameter.

Eg:

**#include <stdio.h>**

**#include <conio.h>**

**int max(int a,int b)**

**{**

**Int s;**

**If (a>b)**

**{**

**Printf(“hello\n”);**

**}**

**Else**

**{**

**Printf(“hai\n”);**

**}**

**s=a+b;**

**Return s;**

**}**

**Void main()**

**{**

**Clrscr();**

**Int x,y,r;**

**Printf(“enter numbers\n”);**

**Scanf(“%d%d”,&x,&y);**

**r=Max(x,y);**

**Printf(“%d\n”,r);**

**getch();**

**}**

Here “x” and “y” are actual parameters,copy of actual parameter value is transferred into formal parameters “a” and “b”. (in this example all parameters are integer type)

|(By default, C uses **call by value** to pass arguments. In general, it means the code within a function cannot alter the arguments used to call the function).

**Call by reference**

in this method of parameter passing,the address of actual parameter is passed to formal parameters.so the function can directly access actual parameter data(note: in this case actual and formal parameters refers the same memory location).hence manipulations done on formal parameter values will effect actual parameters.(in this formal arguments are pointers.Pointers are those variables that can store memory addresses.)

Eg:

#include <stdio.h>

#include <conio.h>

This is Function definition,we don’t need function declaration here . bcz,function is defined in the same program itself.

This is “int” return type function

int max(int \*a,int \*b)

{

Int sum;

Sum=\*a+\*b;

Return sum;

}

Void main()

{

Clrscr();

Int x,y,r;

This is Function call,and return value of function is assigned to variable**” r ”**.so we return value from function is available in main function.**(ie, r=sum)**

Printf(“enter numbers\n”);

Scanf(“%d%d”,&x,&y);

r=Max(&x,&y);

Printf(“%d\n”,r);

getch();

}

**Explanation**

* Consider data present in variables “x” and ‘y’ (actual variables) are stored in memory locations 1000 and 1300 respectively.
* Now Consider our function call

**Max=(&x,&y);**

We can see that in function call we used a symbol “&” (called as ambrescent).

* This symbol is using to indicate memory address.

“&x” means address of variable x(ie,1000).”&y” means address of variable y.

* So here we are transferring address of variables “x”and “y” (ie,1000 and 1300) to a and b respectively.(a and b are formal parameters,they are pointers).
* By using “ \* ” we are accessing data in the memory location stored in pointers.and then we are doing an addition of values present in memory location 1000 and 1300.
* Sum is then returned to main function.

## Calling a Function

While creating a C function, you give a definition of what the function has to do. To use a function, you will have to call that function to perform the defined task.

When a program calls a function, the program control is transferred to the called function. A called function performs a defined task and when its return statement is executed or when its function-ending closing brace is reached, it returns the program control back to the main program.

**Syntax:**

**Function name(parameter 1,parameter 2….);**

To call a function, you simply need to pass the required parameters along with the function name.no need of parameter data type.

Eg:

consider the statement

**Max(x,y);**

is the true statement.

Consider another statement

**Max(int x,int y);**

is false statement.If we call the function like this,compiler will show a syntax error.

if the function returns a value, then you can store the returned value. For example −

[Live Demo](http://tpcg.io/T4MSFr" \t "https://www.tutorialspoint.com/cprogramming/_blank)

#include <stdio.h>

#include <conio.h>

int max(int num1, int num2); /\* function declaration \*/

int main ()

{

int a = 100; /\* local variable definition and declaration \*/

int b = 200;

int ret;

ret = max(a, b); /\* calling a function to get max value \*/

printf( "Max value is : %d\n", ret );

return 0;

}

int max(int num1, int num2)

{

int result; /\* local variable declaration \*/

if (num1 > num2)

result = num1;

else

result = num2;

return result;

}

We have kept max() along with main() and compiled the source code. While running the final executable, it would produce the following result −

Output:

Max value is : 200

**Recursion**

A function call it self again and again until some objectives are achieved.this process is called as recursion.

Recursion is used when we have to use same sets of codes again and again in a program.

Eg: program to find factorial using recursion

**#include <stdio.h>**

**#include <conio.h>**

**int fact(int n)**

**{**

**If (n==1)**

**{**

**Return 1;**

**}**

**Else**

**{**

**Return n\*fact(n-1);**

**}**

**}**

**Void main()**

**{**

**Clrscr();**

**Int n,s;**

**Printf(“enter number\n”);**

**Scanf(“%d”,&n);**

**S=fact(n);**

**Printf(“factorial of %d is %d\n”,n,s);**

**getch();**

**}**

**Example program for void return type function**

#include <stdio.h>

#include <conio.h>

Void max(int x,int y)

{

This is Function definition,we don’t need function declaration here . bcz,function is defined in the same program itself.

This is void return type function

If (x>y)

{

Printf(“hello\n”);

}

Else

Function body

{

Printf(“hai\n”);

}

}

Void main()

{

Clrscr();

Int x,y,s;

Printf(“enter numbers\n”);

Scanf(“%d%d”,&x,&y);

This is Function call

Max(x,y);

S=x+y;

Printf(“%d\n”,s);

getch();

}

**Output**

(if x=4 and y=6)

Hai

10

**Output**

(if x=4 and y=6)

Hai

10

**Example program for integer return type function**

#include <stdio.h>

#include <conio.h>

int max(int x,int y)

{

Int s;

If (x>y)

This is Function definition,we don’t need function declaration here . bcz,function is defined in the same program itself.

This is int return type function

{

Printf(“hello\n”);

}

Else

{

Printf(“hai\n”);

}

S=x+y;

Return s;

}

Void main()

{

Clrscr();

Int x,y,r;

This is Function call,and return value of function is assigned to variable**” r ”**.so we return value from function is available in main function.**(ie, r=s)**

Printf(“enter numbers\n”);

Scanf(“%d%d”,&x,&y);

r=Max(x,y);

Printf(“%d\n”,r);

getch();

}

**Output**

(if x=5 and y=4)

Hello

9

**Output**

(if x=4 and y=6)

Hai

10