In programming, a function is a segment that groups code to perform a specific task.

A C program has at least one function main( ). Without main() function, there is technically no C program.

Types of C functions

Basically, there are two types of functions in C on basis of whether it is defined by user or not.

* Library function
* User defined function

Library function

Library functions are the in-built function in C programming system. For example:

Main() - The execution of every C program starts from this main() function.

printf() - is used for displaying output in C.

scanf() – is used for taking the input from user and store it computer memory.

User defined function

C allows programmer to define their own function according to their requirement. These types of functions are known as user-defined functions. Suppose, a programmer wants to find factorial of a number and check whether it is prime or not in same program. Then, he/she can create two separate user-defined functions in that program: one for finding factorial and other for checking whether it is prime or not.

How user-defined function works in C Programming?

#include <stdio.h>

void function\_name(){

................

................

}

int main(){

...........

...........

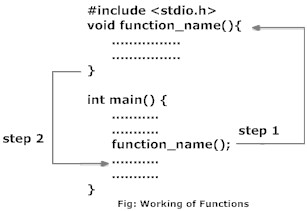
function\_name();

...........

...........

}

As mentioned earlier, every C program begins from main() and program starts executing the codes inside main() function. When the control of program reaches to function\_name() inside main()function. The control of program jumps to void function\_name() and executes the codes inside it. When all the codes inside that user-defined function are executed, control of the program jumps to the statement just after function\_name() from where it is called. Analyze the figure below for understanding the concept of function in C programming.



Remember, the function name is an identifier and should be unique.

Advantages of user defined functions

1. User defined functions helps to decompose the large program into small segments which makes programmar easy to understand, maintain and debug.
2. If repeated code occurs in a program. Function can be used to include those codes and execute when needed by calling that function.
3. Programmer working on large project can divide the workload by making different functions.

## Example of user-defined function

**Write a C program to add two integers. Make a function add to add integers and display sum inmain() function**

/\*Program to demonstrate the working of user defined function\*/

#include <stdio.h>

int add(int a, int b); //function prototype(declaration)

int main(){

int num1,num2,sum;

printf("Enters two number to add\n");

scanf("%d %d",&num1,&num2);

sum=add(num1,num2); //function call

printf("sum=%d",sum);

return 0;

}

int add(int a,int b) //function definition

{

/\* Start of function definition. \*/

int add;

add=a+b;

return add; //return statement of function

/\* End of function definition. \*/

}

## Function prototype(declaration):

Every function in C programming should be declared before they are used. These type of declaration are also called function prototype. Function prototype gives compiler information about function name, type of arguments to be passed and return type.

### Syntax of function prototype

return\_type function\_name(type(1) argument(1),....,type(n) argument(n))

In the above example,int add(int a, int b); is a function prototype which provides following information to the compiler:

1. name of the function is add()
2. return type of the function is int.
3. two arguments of type int are passed to function.

Function prototype are not needed if user-definition function is written before main() function.

## Function call

Control of the program cannot be transferred to user-defined function unless it is called invoked.

### Syntax of function call

function\_name(argument(1),....argument(n));

In the above example, function call is made using statement add(num1,num2); from main(). This make the control of program jump from that statement to function definition and executes the codes inside that function.

## Function definition

Function definition contains programming codes to perform specific task.

#### Syntax of function definition

return\_type function\_name(type(1) argument(1),..,type(n) argument(n))

{

//body of function

}

Function definition has two major components:

### 1. Function declarator

Function declarator is the first line of function definition. When a function is called, control of the program is transferred to function declarator.

#### Syntax of function declaratory

return\_type function\_name(type(1) argument(1),....,type(n) argument(n))

Syntax of function declaration and declarator are almost same except, there is no semicolon at the end of declarator and function declarator is followed by function body.

In above example, int add(int a,int b) in line 12 is a function declarator.

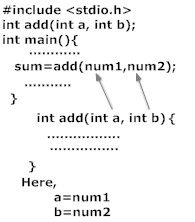
### 2. Function body

Function declarator is followed by body of function inside braces.

## Passing arguments to functions

In programming, argument(parameter) refers to data this is passed to function(function definition) while calling function.

In above example two variable, num1 and num2 are passed to function during function call and these arguments are accepted by arguments a and b in function definition.



Arguments that are passed in function call and arguments that are accepted in function definition should have same data type. For example:

If argument num1 was of int type and num2 was of float type then, argument variable a should be of type int and b should be of type float,i.e., type of argument during function call and function definition should be same.

A function can be called with or without an argument.

## Return Statement

Return statement is used for returning a value from function definition to calling function.

### Syntax of return statement

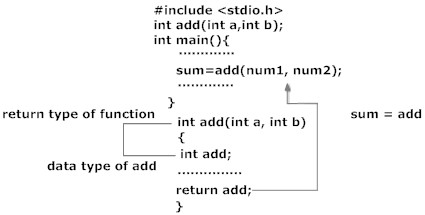
return (expression);

For example:

return a;

return (a+b);

In above example, value of variable add in add() function is returned and that value is stored in variable *sum* in main() function. The data type of expression in return statement should also match the return type of function.



## Types of User-defined functions

For better understanding of arguments and return type in functions, user-defined functions can be categorised as:

1. [Function with no arguments and no return value](http://www.programiz.com/c-programming/types-user-defined-functions#no_no)
2. [Function with no arguments and return value](http://www.programiz.com/c-programming/types-user-defined-functions#no_yes)
3. [Function with arguments but no return value](http://www.programiz.com/c-programming/types-user-defined-functions#yes_no)
4. [Function with arguments and return value](http://www.programiz.com/c-programming/types-user-defined-functions#yes_yes).

Let's take an example to find whether a number is prime or not using above 4 cateogories of user defined functions.

## Function with no arguments and no return value.

/\*C program to check whether a number entered by user is prime or not using function with no arguments and no return value\*/

#include <stdio.h>

void prime();

int main(){

prime(); //No argument is passed to prime().

return 0;

}

void prime(){

/\* There is no return value to calling function main(). Hence, return type of prime() is void \*/

int num,i,flag=0;

printf("Enter positive integer enter to check:\n");

scanf("%d",&num);

for(i=2;i<=num/2;++i){

if(num%i==0){

flag=1;

}

}

if (flag==1)

printf("%d is not prime",num);

else

printf("%d is prime",num);

}

Function prime() is used for asking user a input, check for whether it is prime of not and display it accordingly. No argument is passed and returned form prime() function.

## Function with no arguments but return value

/\*C program to check whether a number entered by user is prime or not using function with no arguments but having return value \*/

#include <stdio.h>

int input();

int main(){

int num,i,flag = 0;

num=input(); /\* No argument is passed to input() \*/

for(i=2; i<=num/2; ++i){

if(num%i==0){

flag = 1;

break;

}

}

if(flag == 1)

printf("%d is not prime",num);

else

printf("%d is prime", num);

return 0;

}

int input(){ /\* Integer value is returned from input() to calling function \*/

int n;

printf("Enter positive integer to check:\n");

scanf("%d",&n);

return n;

}

There is no argument passed to input() function But, the value of n is returned from input() tomain() function.

## Function with arguments and no return value

/\*Program to check whether a number entered by user is prime or not using function with arguments and no return value \*/

#include <stdio.h>

void check\_display(int n);

int main(){

int num;

printf("Enter positive enter to check:\n");

scanf("%d",&num);

check\_display(num); /\* Argument num is passed to function. \*/

return 0;

}

void check\_display(int n){

/\* There is no return value to calling function. Hence, return type of function is void. \*/

int i, flag = 0;

for(i=2; i<=n/2; ++i){

if(n%i==0){

flag = 1;

break;

}

}

if(flag == 1)

printf("%d is not prime",n);

else

printf("%d is prime", n);

}

Here, check\_display() function is used for check whether it is prime or not and display it accordingly. Here, argument is passed to user-defined function but, value is not returned from it to calling function.

## Function with argument and a return value

/\* Program to check whether a number entered by user is prime or not using function with argument and return value \*/

#include <stdio.h>

int check(int n);

int main(){

int num,num\_check=0;

printf("Enter positive enter to check:\n");

scanf("%d",&num);

num\_check=check(num); /\* Argument num is passed to check() function. \*/

if(num\_check==1)

printf("%d is not prime",num);

else

printf("%d is prime",num);

return 0;

}

int check(int n){

/\* Integer value is returned from function check() \*/

int i;

for(i=2;i<=n/2;++i){

if(n%i==0)

return 1;

}

return 0;

}

Here, check() function is used for checking whether a number is prime or not. In this program, input from user is passed to function check() and integer value is returned from it. If input the number is prime, 0 is returned and if number is not prime, 1 is returned.

Recurison

A function that calls itself is known as recursive function and this technique is known as recursion in C programming.

### Example of recursion in C programming

**Write a C program to find sum of first n natural numbers using recursion. Note: Positive integers are known as natural number i.e. 1, 2, 3....n**

#include <stdio.h>

int sum(int n);

int main(){

int num,add;

printf("Enter a positive integer:\n");

scanf("%d",&num);

add=sum(num);

printf("sum=%d",add);

}

int sum(int n){

if(n==0)

return n;

else

return n+sum(n-1); /\*self call to function sum() \*/

}

In, this simple C program, sum() function is invoked from the same function. If n is not equal to 0 then, the function calls itself passing argument 1 less than the previous argument it was called with. Suppose, n is 5 initially. Then, during next function calls, 4 is passed to function and the value of argument decreases by 1 in each recursive call. When, n becomes equal to 0, the value of n is returned which is the sum numbers from 5 to 1

For better visualization of recursion in this example:

sum(5)

=5+sum(4)

=5+4+sum(3)

=5+4+3+sum(2)

=5+4+3+2+sum(1)

=5+4+3+2+1+sum(0)

=5+4+3+2+1+0

=5+4+3+2+1

=5+4+3+3

=5+4+6

=5 +10

=15

Every recursive function must be provided with a way to end the recursion. In this example when, nis equal to 0, there is no recursive call and recursion ends.

### Advantages and Disadvantages of Recursion

Recursion is more elegant and requires few variables which make program clean. Recursion can be used to replace complex nesting code by dividing the problem into same problem of its sub-type.

In other hand, it is hard to think the logic of a recursive function. It is also difficult to debug the code containing recursion.

### More Examples on Recursive Function

* [Program to calculate factorial using recursion](http://www.programiz.com/c-programming/examples/factorial-recursion)
* This program takes a positive integer from user and calculates the factorial of that number. Instead of [loops to calculate factorial](http://www.programiz.com/c-programming/examples/factorial), this program uses recursive function to calculate the factorial of a number.

## Source Code to Calculate Factorial Using Recursion

/\* Source code to find factorial of a number. \*/

#include<stdio.h>

int factorial(int n);

int main()

{

int n;

printf("Enter an positive integer: ");

scanf("%d",&n);

printf("Factorial of %d = %ld", n, factorial(n));

return 0;

}

int factorial(int n)

{

if(n!=1)

return n\*factorial(n-1);

}

* [Program to reverse sentence using recursion](http://www.programiz.com/c-programming/examples/reverse-sentence-recursion)

This program takes a sentence from user and reverses that sentence using recursion. This program does not use string to reverse the sentence or store the sentence.

/\* Example to reverse a sentence entered by user without using strings. \*/

#include <stdio.h>

void Reverse();

int main()

{

printf("Enter a sentence: ");

Reverse();

return 0;

}

void Reverse()

{

char c;

scanf("%c",&c);

if( c != '\n')

{

Reverse();

printf("%c",c);

}

}

**Output**

Enter a sentence: margorp emosewa

awesome program

**Explanation**

This program prints "Enter a sentence: " then, Reverse() function is called. This function stores the first letter entered by user and stores in variable c. If that variable is other than '\n' [ enter character] then, again Reverse() function is called. Don't assume this Reverse() function and the Reverse()function before is same although they both have same name. Also, the variables are also different, i.e., c variable in both functions are also different. Then, the second character is stored in variable c of second Reverse function. This process goes on until user enters '\n'. When, user enters '\n', the last function Reverse() function returns to second last Reverse() function and prints the last character. Second last Reverse() function returns to the third last Reverse() function and prints second last character. This process goes on and the final output will be the reversed sentence.

* [Program to find H.C.F using recursive function](http://www.programiz.com/c-programming/examples/hcf-recursion)

This program takes two positive integers from user and calculates HCF or GCD using recursion.

/\* Example to calculate GCD or HCF using recursive function. \*/

#include <stdio.h>

int hcf(int n1, int n2);

int main()

{

int n1, n2;

printf("Enter two positive integers: ");

scanf("%d%d", &n1, &n2);

printf("H.C.F of %d and %d = %d", n1, n2, hcf(n1,n2));

return 0;

}

int hcf(int n1, int n2)

{

if (n2!=0)

return hcf(n2, n1%n2);

else

return n1;

}

C programming storage class

Every variable in C programming has two properties: type and storage class. Type refers to the data type of variable whether it is character or integer or floating-point value etc. And storage class determines how long it stays in existence.

There are 4 types of storage class:

1. automatic
2. external
3. static
4. register

## Automatic storage class

### Keyword for automatic variable

**auto**

Variables declared inside the function body are automatic by default. These variable are also known as local variables as they are local to the function and doesn't have meaning outside that function

Since, variable inside a function is automatic by default, keyword auto are rarely used.

## External storage class

External variable can be accessed by any function. They are also known as global variables. Variables declared outside every function are external variables.

In case of large program, containing more than one file, if the global variable is declared in file 1 and that variable is used in file 2 then, compiler will show error. To solve this problem, keyword extern is used in file 2 to indicate that, the variable specified is global variable and declared in another file.

### Example to demonstrate working of external variable

#include

void Check();

int a=5;

/\* a is global variable because it is outside every function \*/

int main(){

a+=4;

Check();

return 0;

}

void Check(){

++a;

/\* ----- Variable a is not declared in this function but, works in any function as they are global variable ------- \*/

printf("a=%d\n",a);

}

**Output**

a=10

## Register Storage Class

### Keyword to declare register variable

Register

**Example of register variable**

register int a;

Register variables are similar to automatic variable and exists inside that particular function only.

If the compiler encounters register variable, it tries to store variable in microprocessor's register rather than memory. Value stored in register are much faster than that of memory.

In case of larger program, variables that are used in loops and function parameters are declared register variables.

Since, there are limited number of register in processor and if it couldn't store the variable in register, it will automatically store it in memory.

## Static Storage Class

The value of static variable persists until the end of the program. A variable can be declared static using keyword: static. For example:

static int i;

Here, i is a static variable.

### Example to demonstrate the static variable

#include <stdio.h>

void Check();

int main(){

Check();

Check();

Check();

}

void Check(){

static int c=0;

printf("%d\t",c);

c+=5;

}

0 5 10

During first function call, it will display 0. Then, during second function call, variable c will not be initialized to 0 again, as it is static variable. So, 5 is displayed in second function call and 10 in third call.

If variable c had been automatic variable, the output would have been:

0 0 0