C Programming Functions

**(both user-defined and standard library functions) in C programming. Also, you will learn why functions are used in programming.**



A function is a block of code that performs a specific task.

Suppose, a program related to graphics needs to create a circle and color it depending upon the radius and color from the user. You can create two functions to solve this problem:

* create a circle function
* color function

Dividing complex problem into small components makes program easy to understand and use.

## Types of functions in C programming

Depending on whether a function is defined by the user or already included in C compilers, there are two types of functions in C programming

There are two types of functions in C programming:

* [Standard library functions](https://www.programiz.com/c-programming/library-function)
* [User defined functions](https://www.programiz.com/c-programming/c-user-defined-functions)

### Standard library functions

The standard library functions are built-in functions in C programming to handle tasks such as mathematical computations, I/O processing, string handling etc.

These functions are defined in the header file. When you include the header file, these functions are available for use. For example:

The printf() is a standard library function to send formatted output to the screen (display output on the screen). This function is defined in "stdio.h" header file.

There are other numerous library functions defined under "stdio.h", such as scanf(), fprintf(), getchar() etc. Once you include "stdio.h"in your program, all these functions are available for use.

Visit this page to learn more about [standard library functions in C programming](https://www.programiz.com/c-programming/library-function).

### User-defined functions

As mentioned earlier, C allow programmers to define functions. Such functions created by the user are called user-defined functions.

Depending upon the complexity and requirement of the program, you can create as many user-defined functions as you want.

## How user-defined function works?

#include <stdio.h>

void functionName()

{

... .. ...

... .. ...

}

int main()

{

... .. ...

... .. ...

functionName();

... .. ...

... .. ...

}

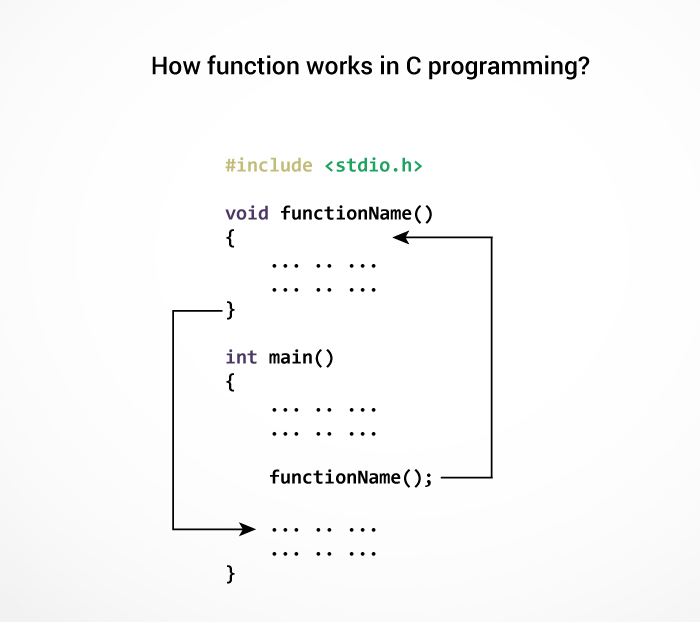
The execution of a C program begins from the main() function.

When the compiler encounters functionName(); inside the main function, control of the program jumps to

void functionName()

And, the compiler starts executing the codes inside the user-defined function.

The control of the program jumps to statement next to functionName();once all the codes inside the function definition are executed.



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

This is just an overview on user-defined function. Visit these pages to learn more on:

* [User-defined Function in C programming](https://www.programiz.com/c-programming/c-user-defined-functions)
* [Types of user-defined Functions](https://www.programiz.com/c-programming/types-user-defined-functions)

### Advantages of user-defined function

1. The program will be easier to understand, maintain and debug.
2. Reusable codes that can be used in other programs
3. A large program can be divided into smaller modules. Hence, a large project can be divided among many programmers.

Check out these examples to learn more:

* [Display Prime Numbers Between Intervals Using Function](https://www.programiz.com/c-programming/examples/prime-interval-function)
* [Check Prime or Armstrong Number Using User-defined Function](https://www.programiz.com/c-programming/examples/prime-armstrong-function)
* [Check Whether a Number can be Expressed as Sum of Two Prime Numbers](https://www.programiz.com/c-programming/examples/sum-prime-numbers)

# C Programming User-defined functions

A function is a block of code that performs a specific task.

C allows you to define [functions](https://www.programiz.com/c-programming/c-functions) according to your need. These functions are known as user-defined functions. For example:

Suppose, you need to create a circle and color it depending upon the radius and color. You can create two functions to solve this problem:

* createCircle() function
* color() function

## Example: User-defined function

Here is a example to add two integers. To perform this task, a user-defined function addNumbers() is defined.

#include <stdio.h>

int addNumbers(int a, int b); // function prototype

int main()

{

int n1,n2,sum;

printf("Enters two numbers: ");

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

sum = addNumbers(n1, n2); // function call

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

return 0;

}

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

{

int result;

result = a+b;

return result; // return statement

}

## Function prototype

A function prototype is simply the declaration of a function that specifies function's name, parameters and return type. It doesn't contain function body.

A function prototype gives information to the compiler that the function may later be used in the program.

### Syntax of function prototype

returnType functionName(type1 argument1, type2 argument2,...);

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

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

The function prototype is not needed if the user-defined function is defined before the main() function.

## Calling a function

Control of the program is transferred to the user-defined function by calling it.

### Syntax of function call

functionName(argument1, argument2, ...);

In the above example, function call is made using addNumbers(n1,n2); statement inside the main().

## Function definition

Function definition contains the block of code to perform a specific task i.e. in this case, adding two numbers and returning it.

#### Syntax of function definition

returnType functionName(type1 argument1, type2 argument2, ...)

{

//body of the function

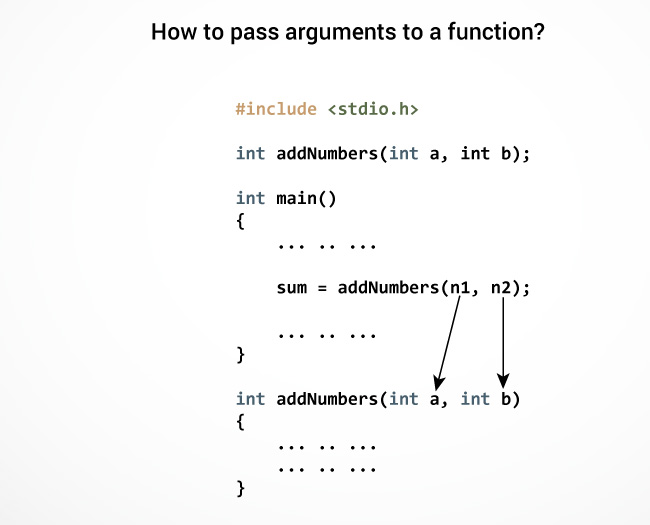
}

When a function is called, the control of the program is transferred to the function definition. And, the compiler starts executing the codes inside the body of a function.

## Passing arguments to a function

In programming, argument refers to the variable passed to the function. In the above example, two variables n1 and n2 are passed during function call.

The parameters a and b accepts the passed arguments in the function definition. These arguments are called formal parameters of the function.



The type of arguments passed to a function and the formal parameters must match, otherwise the compiler throws error.

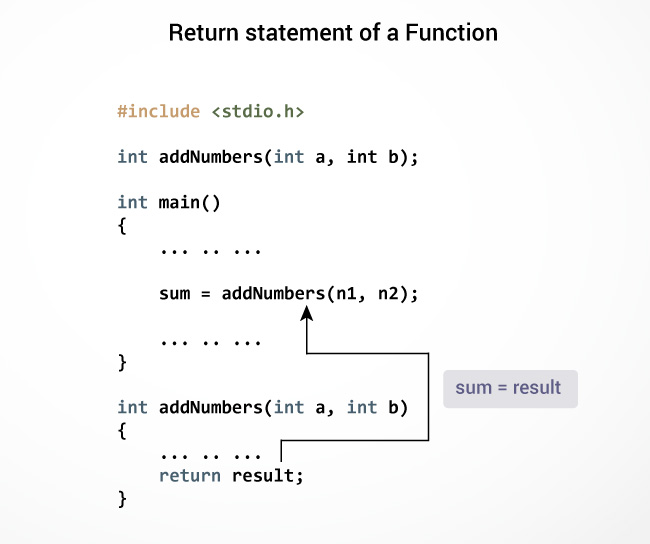
If n1 is of char type, a also should be of char type. If n2 is of float type, variable b also should be of float type.

A function can also be called without passing an argument.

## Return Statement

The return statement terminates the execution of a function and returns a value to the calling function. The program control is transferred to the calling function after return statement.

In the above example, the value of variable result is returned to the variable sum in the main() function.



### Syntax of return statement

return (expression);

For example,

return a;

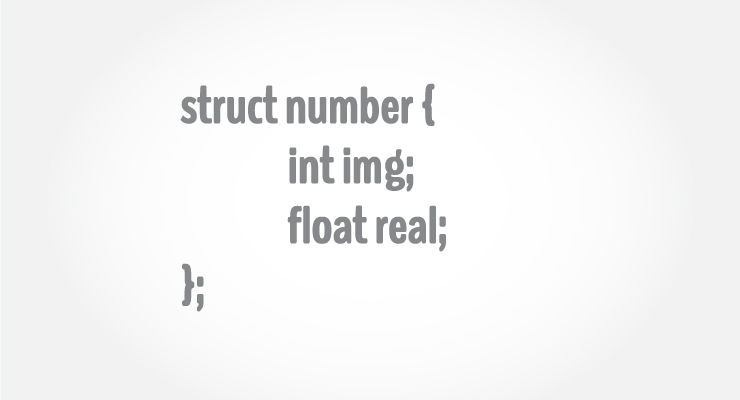
return (a+b);

The type of value returned from the function and the return type specified in function prototype and function definition must match.

Check out these examples to learn more:

* [Display Prime Numbers Between Intervals Using Function](https://www.programiz.com/c-programming/examples/prime-interval-function)
* [Check Whether a Number can be Expressed as Sum of Two Prime Numbers](https://www.programiz.com/c-programming/examples/sum-prime-numbers)
* [Find the Sum of Natural Numbers using Recursion](https://www.programiz.com/c-programming/examples/natural-number-sum-recursion)

# C Programming Structure



## Structure Definition in C

Keyword struct is used for creating a structure.

### Syntax of structure

struct structure\_name

{

data\_type member1;

data\_type member2;

.

.

data\_type memeber;

};

**Note**: Don't forget the semicolon }; in the ending line.

We can create the structure for a person as mentioned above as:

struct person

{

char name[50];

int citNo;

float salary;

};

# Structures in C

**What is a structure?**  
A structure is a user defined data type in C/C++. A structure creates a data type that can be used to group items of possibly different types into a single type.

**How to create a structure?**  
‘struct’ keyword is used to create a structure. Following is an example.

|  |
| --- |
| struct addrress  {     char name[50];     char street[100];     char city[50];     char state[20]     int pin;  }; |

Run on IDE

**How to declare structure variables?**  
A structure variable can either be declared with structure declaration or as a separate declaration like basic types.

|  |
| --- |
| // A variable declaration with structure declaration.  struct Point  {     int x, y;  } p1;  // The variable p1 is declared with 'Point'  // A variable declaration like basic data types  struct Point  {     int x, y;  };    int main()  {     struct Point p1;  // The variable p1 is declared like a normal variable  } |

Run on IDE

Note: In C++, the struct keyword is optional before in declaration of variable. In C, it is mandatory.

***How to initialize structure members?***  
Structure members **cannot be** initialized with declaration. For example the following C program fails in compilation.

struct Point

{

   int x = 0;  // COMPILER ERROR:  cannot initialize members here

   int y = 0;  // COMPILER ERROR:  cannot initialize members here

};

#include <stdio.h>

#include <string.h>

struct student

{

           int id;

           char name[20];

           float percentage;

};

int main()

{

           struct student record = {0}; //Initializing to null

           record.id=1;

           strcpy(record.name, "Raju");

           record.percentage = 86.5;

           printf(" Id is: %d \n", record.id);

           printf(" Name is: %s \n", record.name);

           printf(" Percentage is: %f \n", record.percentage);

           return 0;

}