Function in programming C

A function is a group of commands that go together to perform a task. Each C program has at least one function, main (), and all most normal programs define additional functions.

A function is a group of commands that go together to perform a task. Each C program has at least one function, **main ()** , and all most normal programs define additional functions.

You can break your code into separate functions. The way you divide your code into different functions depends on you, but logically, a function usually has a certain task.

A function **declaration informs** the compiler about the name of the function, the return type and the parameter. A function **definition** provides the body of a function.

The standard C language libraries offer many functions available for your program to call. For example, **strcat ()** can concatenate two strings, **memcpy ()** uses to copy a memory to another memory area and many other functions.

A function is known for different names such as a method, a sub-route, or a procedure.

Define a function in C

The general pattern of the function definition in Language C is as follows:

```
kieu_tra_ve ten_ham (  danh_sach_tham_so  ) {  than_cua_ham  }
```

A function definition in C includes *a function head* and a *function body* . Here are the parts of a function:

Return type : A function can return a value. **Kieu_tra_ve** is the data form of the value returned by the function. Some functions provide operations and do not return any values. That is the **void** function.

Function name : This is the actual name of the function. Function name and parameter list make up the function sign.

Parameter : When the function is called, you must pass the parameter list. A value towards an actual parameter. The parameter list has the type, order, and number of function parameters. The parameters in the function are optional, meaning a function may have no parameters.

Function body : The body of a function includes a set of commands that determine what the function does.

For example:

The following is the source code for a function called **timGTLN ()** . This function has 2 parameters: so1 and so2 and returns the largest value between functions:

```
/* ham tra ve gia tri lon nhat giua hai so */ int  timGTLN ( int  so1 , int  so2
```

Declaring the function in C

A function **declaration informs** the compiler of the function name and function call. The body function can be defined in a discrete way.

A function declaration has the following sections:

```
kieu_tra_ve ten_ham (  danh sach tham so  );
```

For example, when defining the timGTLN () function, here is the function declaration:

```
int  timGTLN ( int  so1 , int  so2 );
```

The parameter names are not important in the function declaration, the following types are valid declarations:

```
int  timGTLN ( int , int );
```

A function declaration is required when you define a function and source and when calling a function from another source file. In this case, you should declare the function before calling that function.

Call the function in C

While creating a function, you define what the function must do. To use a function, you must call that function to perform a specific task.

When a program calls a function, the control part is passed to the called function. A called function performs defined tasks and returns the value after executing the program.

To call a function, you simply need to pass the required parameters along with the function's name and if the function returns the values, you can reserve these return values, for example:

```
#include /* khai bao ham */ int  timGTLN ( int  so1 , int  so2 ); int  main  ()
```

I keep the timGTLN () function value in the main function into the kq variable. Compile and run the above C program to see the results:

Parameters of functions in C

A function that uses parameter lists, it must declare variables and accept the values ??of these variables. These variables are called **official variables** .

Official variables are like other local variables inside the function.

When you call the function, there are 2 ways to pass the values ??to the function:

Call type Description
Call by value

This method copies the actual value of the parameter into the official parameter of a function. In this case, the changes of the parameters themselves within the function do not affect the parameters.
Called by reference

This method copies the address of the parameter into the official parameter. Inside the function, the address used to access the parameter is actually used when calling the function. This means that changes to parameters make the parameter change.
By default, C uses **calling by value** to pass parameters. In general, that code in a function cannot change the parameters used to call that function and in the example above, when calling the timGTLN () function is using the same method.

According to Tutorialspoint

Previous article: Loop in programming C

Next lesson: Scope rules in programming C