Functions in C++

- Modular programming: breaking a program up into smaller, manageable functions or modules
- Function: a collection of statements to perform a task
- Motivation for modular programming:
  - Improves maintainability of programs
  - Simplifies the process of writing programs

This program has one long, complex function containing all of the statements necessary to solve a problem.

```
int main()
{
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
    statement;
}
```

In this program the problem has been divided into smaller problems, each of which is handled by a separate function.

```
int main()
{
    statement;
    statement;
    statement;
}
```

main function

```
void function2()
{
    statement;
    statement;
    statement;
}
```

function 2

```
void function3()
{
    statement;
    statement;
    statement;
}
```

function 3

```
void function4()
{
    statement;
    statement;
    statement;
}
```

function 4

Defining and Calling Functions

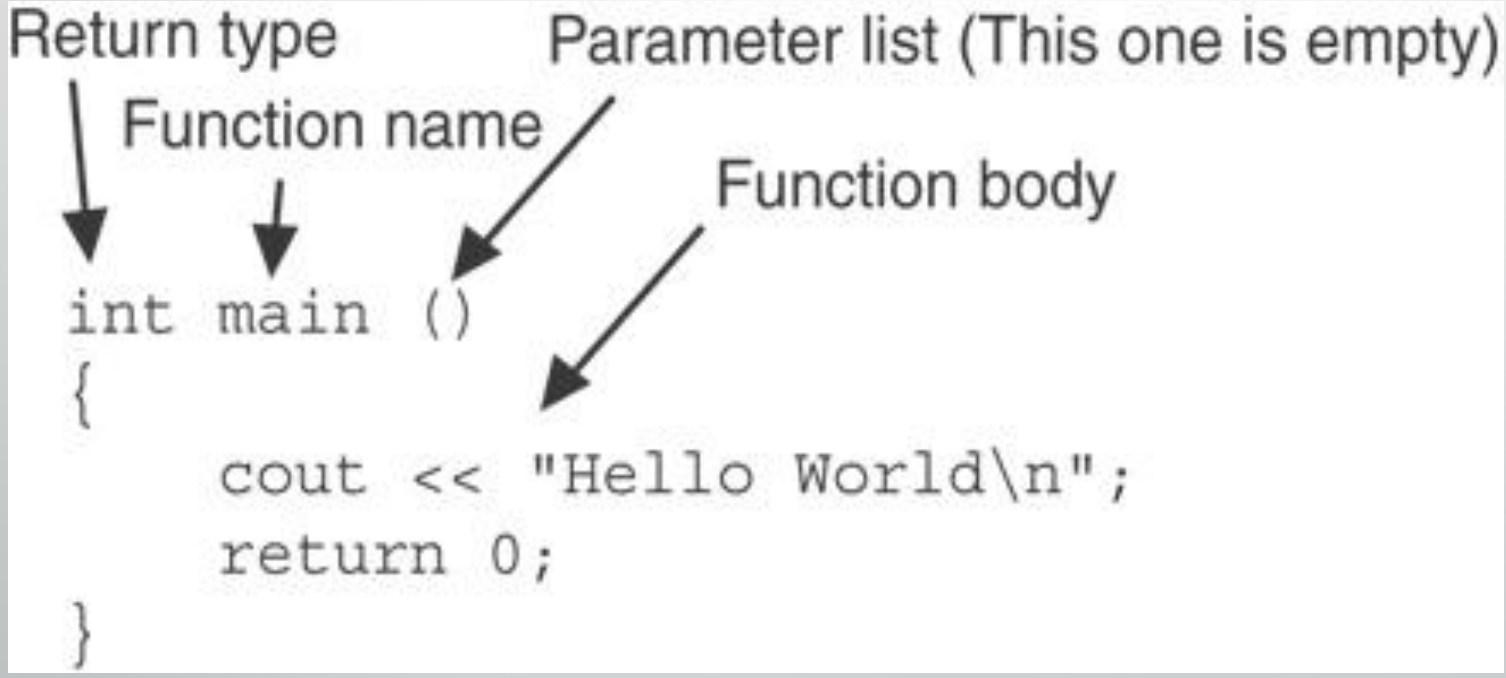
- Function call: statement causes a function to execute
- Function definition: statements that make up a function

Function Definition

Definition includes:

- <u>return type:</u> data type of the value that function returns to the part of the program that called it
- <u>name:</u> name of the function. Function names follow same rules as variables
- <u>parameter list:</u> variables containing values passed to the function
- <u>body:</u> statements that perform the function's task, enclosed in { }

Function Definition



Function Return Type

- If a function returns a value, the type of the value must be indicated:

```
int main()
```

- If a function does not return a value, its return type is `void`:

```
void printHeading()
{
   cout << "Monthly Sales\n";
}
```

Calling a Function

- To call a function, use the function name followed by `()` and `;`

```
printHeading();
```

- When called, program executes the body of the called function

- After the function terminates, execution resumes in the calling function at point of call.

Program 6-1

```
1   // This program has two functions: main and displayMessage
2   #include <iostream>
3   using namespace std;
4
5   //*****************************************
6   // Definition of function displayMessage  *
7   // This function displays a greeting.     *
8   //*****************************************
9
10  void displayMessage()
11  {
12     cout << "Hello from the function displayMessage.\n";
13  }
14
15  //*****************************************
16  // Function main                          *
17  //*****************************************
18
19  int main()
20  {
21     cout << "Hello from main.\n";
22     displayMessage();
23     cout << "Back in function main again.\n";
24     return 0;
25  }
```

Program Output

```
Hello from main.
Hello from the function displayMessage.
Back in function main again.
```

Flow of Control in Previous Program

```
void displayMessage()
{
   cout << "Hello from the function displayMessage.\n";
}
```

```
int main()
{
   cout << "Hello from main.\n"
   displayMessage();
   cout << "Back in function main again.\n";
   return 0;
}
```

Calling Functions

- `main` can call any number of functions
- Functions can call other functions
- Compiler must know the following about a function before it is called:
  - name
  - return type
  - number of parameters
  - data type of each parameter