Programming Fundamentals

Muhammad Ateeq

[Updated: 15-02-2023]

Program Skeleton, Printing, Variables

C++ Program Skeleton

What is a computer program?
A computer program, also known as software or an application, is a set of instructions that tells a computer what to do. Programs are written by programmers in various programming languages, such as C++, Java, Python, and many others. Programs can perform a wide variety of tasks, such as processing data, performing calculations, displaying graphics, playing audio or video, communicating over networks, and interacting with users through graphical user interfaces.

Computer programs can be classified into different categories based on their purpose, such as operating systems, productivity software, entertainment software, scientific and engineering software, and many others. Programs can also be designed to run on different types of hardware platforms, such as desktop computers, mobile devices, and servers.

Typically, computer programs take some input, process it, and give some output as shown in figure below.

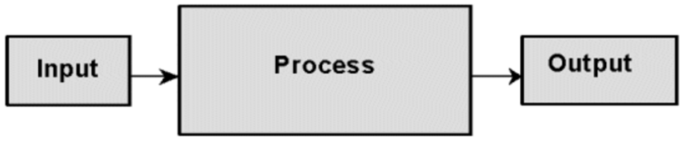

Figure 1. Input output in a computer program

What do I need to know to write computer programs?
To write computer programs, you will need to learn a programming language and become familiar with programming concepts and tools. Here are some key things you will need to know:

<u>Important Currently</u>

Programming languages: A programming language is a formal language used to write computer programs. There are many programming languages available, such as C++, Java, Python, and many others. Each language has its own syntax and features, and is suited for different types of programs.
Programming concepts: You will need to learn programming concepts, such as variables, data types, control structures, functions, and object-oriented programming. These concepts form the building blocks of programs and help you to structure your code and solve problems.
Integrated development environment (IDE): An IDE is a software application that provides a comprehensive environment for writing, testing, and debugging code. Some popular IDEs for programming include Visual Studio, Eclipse, and Code::Blocks.
Debugging tools: When writing code, you will inevitably encounter bugs or errors in your program. Debugging tools help you to identify and fix these issues. IDEs often provide built-in debugging tools, but there are also standalone tools available, such as Valgrind and GDB.
Problem-solving skills: Writing programs requires you to think logically and creatively to solve problems. Developing strong problem-solving skills can help you to approach programming challenges with confidence and find effective solutions.

Not Important Currently

Version control: Version control is a system that tracks changes to your code over time and allows you to collaborate with other programmers. Git is a popular version control system that is widely used in the programming community.

Data structures and algorithms: Data structures are ways of organizing and storing data in a program, while algorithms are step-by-step procedures for solving problems. Learning about data structures and algorithms can help you to write efficient and optimized programs.

Overall, learning to write computer programs requires a combination of technical skills and problem-solving abilities. With practice and persistence, you can develop the skills needed to become a proficient programmer.

Ok, since we are to write programs in C++, how does a C++ program like?

The skeleton of a C++ program is the basic structure that all C++ programs follow. It is a set of statements that must be included in any C++ program in order for it to be a valid and executable program.

Here is a simple skeleton for a C++ program:

```
// include necessary libraries and namespaces
#include <iostream>
using namespace std;

// define the main function
int main() {
    // program statements go here
    return 0; // indicate successful program completion
}
```

This skeleton can be broken down into the following components:

Libraries and namespaces: This is the first part of the skeleton, which includes the libraries and namespaces that the program will use. Libraries provide functions and classes that can be used in the program, while namespaces help to organize and avoid naming conflicts between the different elements of the program. The most common library used in C++ programs is iostream, which provides input and output functionality. The using namespace std; statement is used to avoid having to write std:: before every use of cout or cin (which are part of the iostream library).

The main function: This is the heart of the program, where all of the program statements go. The main function is required in all C++ programs, and is the first function that is executed when the program runs. In this skeleton, the main function is defined with the int return type, which indicates that the function returns an integer value. The main function is also enclosed in curly braces {}.

Program statements: These are the statements that make up the body of the program, which go inside the main function. Program statements can be any valid C++ statement, such as variable declarations, input/output statements, conditional statements, loops, and function calls. In this skeleton, there are no program statements inside the main function.

Return statement: This is the last statement in the main function, which indicates the end of the program. The return 0; statement is used to indicate successful program completion, and can be replaced with any integer value to indicate different types of program completion (such as error codes).

Overall, the skeleton of a C++ program provides a basic framework for building more complex programs, by including the necessary libraries, defining the main function, and providing a structure for adding program statements.

Output with cout

Ok, so what is the first thing I should do as a computer programmer?
Since computers computer programs typically take some input, process it, and give some output. Let's start by learning how a C++ program can output something. Let's do so by trying to print something on the computer screen (aka standard out).

So how does output or printing work in C++?
In C++, printing is typically accomplished using the "stream insertion" operator, which is represented by the two less-than signs (<<). The operator is used to insert data into an output stream, which is usually directed to the console or a file.

To print a value or a string to the console, you would use the cout object, which is part of the standard library's iostream header. Here's an example:

```
#include <iostream>
Using namespace std;
int main() {
  cout << "Hello, world!" << endl;
  return 0;
}
```

In this code, the << operator is used to insert the string "Hello, world!" into the output stream represented by cout. The endl command is used to insert a newline character and flush the output stream, causing the text to be printed to the console.

Comments

Ok, what next thing are you going to tell us?
Everything you write as part of C++ code must follow the defined rules of the language. Sometime, however, you may need to leave out some notes explaining certain things that may help you or other readers/editors in understanding. These are called comments. Comments in C++ are used to annotate the code with text that is ignored by the compiler. They are used to explain the purpose of the code, to provide context to other developers, or to temporarily disable a section of code during debugging. There are two types of comments in C++:

Single-line comments: Single-line comments start with // and continue until the end of the line. They are commonly used for short, descriptive comments. Here's an example:

```
int main() {
  // This is a single-line comment
  int x = 10; // This line initializes the variable x to 10
  return 0;
}
```

Multi-line comments: Multi-line comments start with /* and end with */. They can span multiple lines and are commonly used for longer comments, such as function or file headers. Here's an example:

```
int main() {
  /* This block of code is temporarily disabled
     for debugging purposes.

     cout << z << endl;
  */

  return 0;
}
```

Note that comments are not compiled or executed by the program, and therefore do not affect the program's behavior or performance. They are only used for human consumption and can be added or removed as needed without affecting the program's functionality.

Variables

What is a variable? Why do we need a variable? How do variables work in C++?
In computer programming, a variable is a named storage location in memory that can hold a value. The value stored in a variable can be changed during the execution of a program. Variables are used to store data that is used by the program, such as user input, intermediate calculations, or final output.

We need variables in programming because they allow us to store and manipulate data. By storing data in a variable, we can refer to that data by its name rather than its memory location. This makes the code easier to read and understand, and makes it easier to modify the data if necessary.

In C++, variables are declared using a data type and a name, and optionally initialized with a value. Here's an example:

```
int x;            // declare an integer variable named x
double pi = 3.14; // declare and initialize a double variable named pi
char ch = 'A';  // declare and initialize a character variable named ch
```

In this code, int, double, and char are data types that specify the kind of data that can be stored in the variable. The names x, pi, and ch are the identifiers that are used to refer to the variables. The = operator is used to initialize the variables with a value.

Once a variable is declared and initialized, its value can be accessed and modified using its name. Here's an example:

```
int x = 10;      // declare and initialize an integer variable named x
x = x + 5;       // add 5 to the value of x
cout << x << endl; // output the value of x (should be 15)
```

In this code, the value of x is first set to 10. Then, the value of x is modified by adding 5 to it. Finally, the value of x is printed to the console using the cout statement.

Basic Actions on Variables:

Declaring, Assigning, and Accessing Variables

How do we declare, assign values, and access variables in C++?
Declaring, assigning, and accessing variables are some of the basic actions performed on variables in C++. Here is an explanation of each operation:

Declaring variables: Before using a variable in C++, it must be declared, which tells the compiler the name and data type of the variable. The syntax for declaring a variable is:

```
data_type variable_name;
```

Here, data_type is the type of data that the variable can hold, and variable_name is the name given to the variable. For example:

```
int x;          // declares an integer variable named x
float y;        // declares a floating-point variable named y
bool flag;      // declares a boolean variable named flag
```

Assigning values to variables: Once a variable is declared, a value can be assigned to it using the assignment operator (=). The syntax for assigning a value to a variable is:

```
variable_name = value;
```

Here, variable_name is the name of the variable, and value is the value to be assigned to the variable. For example:

```
int x;
x = 5;          // assigns the value 5 to the variable x
float y;
y = 3.14;       // assigns the value 3.14 to the variable y
bool flag;
flag = true;    // assigns the value true to the variable flag
```

Accessing variables: Once a variable is declared and assigned a value, its value can be accessed using its name. The syntax for accessing the value of a variable is:

```
variable_name
```

Here, variable_name is the name of the variable whose value is to be accessed. For example:

```
int x = 5;
cout << "The value of x is " << x << endl; // prints "The value of x is 5"

float y = 3.14;
```

```
cout << "The value of y is " << y << endl; // prints "The value of y is 3.14"

bool flag = true;
cout << "The value of flag is " << flag << endl; // prints "The value of flag is 1"
```

These are the basic operations that can be performed on variables in C++. By declaring, assigning, and accessing variables, you can store and manipulate data in your program.