© JanPietruszka/iStockphoto.

Chapter One: Introduction

Topic 5

1. What is programming?
2. Anatomy of a computer
3. Machine code and programming
4. Becoming familiar with your programming environment
5. Analyzing your first program
6. Errors
7. Problem solving: algorithm design
8. Chapter Summary

Your First Program

- At this point we will analyze the classic first program that everyone writes: Hello World!
- Write the words Hello World! on the screen.

ch01/hello.cpp

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

First Program, the `#include`

- The first line tells the compiler to include a service for "stream input/output". Later you will learn more about this but, for now, just know it is needed to write on the screen.

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

First Program, `using namespace std`

- The second line tells the compiler to use the “standard namespace”. This is used in conjunction with the `<iostream>` first line for controlling input and output.

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

First Program: `int main()`

- The next set of code *defines* a ***function,*** named **`main`.**
  - Every C++ program must contain its one `main` function.
  - All function names must be followed by parentheses. In `main`'s case, the parentheses are empty.
- Braces `{ }` must enclose all the code that belongs to `main`. The braces tell the compiler where to start reading the `main` code, and where to finish.

- ***A project is a company; method main is its CEO.***
- ***Codes in main are TODO list of the CEO.***
- ***Method main is entry point of a project.***

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

First Program: `cout` statement

- To show output on the screen, we use **`cout`.**
- What you want seen on the screen is "sent" to the **`cout`** entity using the **`<<`** operator (sometimes called the insertion operator): **`<< "Hello, World!"`**
- **The curious non-word `endl` means end-of-line, which tells the display to move the cursor down to the start of the next line.**

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

One `cout` can print multiple items

- You can display more than one thing by chaining or "streaming" multiple copies of the << operator into the same statement:

```
   << "A big "<< "Hello, World!" << endl;
```

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

First Program: `return` statement

- The **`main`** function "returns" an "integer" (that is, a whole number without a fractional part, called **`int`** in C++) with value 0.
- This value indicates that the program finished successfully.

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

Output Statements and Streaming Operator <<

The statement

```
cout << "Hello, World!" << endl;
```

is an *output statement*.

- To display values on the screen, you send them to an entity called `cout`.
  - Which stands for "character output" or "console output".
- The `<<` operator denotes the “send to” command.

"Strings" and `endl`

```
cout << "Hello World!" << endl;
```

- **`"Hello World!"`** is called a *`string`*.
- You must put those double-quotes around strings.

- The **`endl`** symbol denotes an *end of line* marker which causes the cursor to move down to the next screen line.

Semicolons are Required after Statements

- Each statement in C++ ends in a semicolon;
  - Note that not every line in a program is a statement, so there are no semicolons after the `<iostream>` line and the `main()` line
  - It is a strange idiosyncrasy, but you will get used to it.

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