# Lecture 1: Introduction to C++ Programming

Curtin FIRST Robotics Club (FRC) Pre-season Training

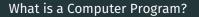
Scott Day 265815F@curtin.edu.au October 14, 2016

Curtin University

#### Table of contents

- 1. Programming
- 2. C++
- 3. Environment Setup
- 4. Editing, Compiling, and Execution

Programming



**Computer Program** A collection of instructions that performs a specific task when executed by a computer.

### What is a Computer Program?

**Computer Program** A collection of instructions that performs a specific task when executed by a computer.

**Algorithm** A part of a computer program that performs a well-defined task.

### What is a Computer Program?

- **Computer Program** A collection of instructions that performs a specific task when executed by a computer.
  - **Algorithm** A part of a computer program that performs a well-defined task.
    - **Software** A collection of computer programs, libraries and related data.

Recipe to writing programs:

1. Understand the problem.

- 1. Understand the problem.
- 2. Think of a solution.

- 1. Understand the problem.
- 2. Think of a solution.
- 3. Describe the solution in as much detail as possible. You may use diagrams or plain English to do this.

- 1. Understand the problem.
- 2. Think of a solution.
- 3. Describe the solution in as much detail as possible. You may use diagrams or plain English to do this.
- 4. Translate your solution into a program.

- 1. Understand the problem.
- 2. Think of a solution.
- 3. Describe the solution in as much detail as possible. You may use diagrams or plain English to do this.
- 4. Translate your solution into a program.
- 5. Run your program and see if it works.

- 1. Understand the problem.
- 2. Think of a solution.
- 3. Describe the solution in as much detail as possible. You may use diagrams or plain English to do this.
- 4. Translate your solution into a program.
- 5. Run your program and see if it works.
  - · Yes? Hurray! Victory!

- 1. Understand the problem.
- 2. Think of a solution.
- 3. Describe the solution in as much detail as possible. You may use diagrams or plain English to do this.
- 4. Translate your solution into a program.
- 5. Run your program and see if it works.
  - · Yes? Hurray! Victory!
  - No? Go back to 1

### Example

Think like a computer!

What steps do you need to take to draw a smiley face?

# Example



### **Example Code Appreciation**

Lets just take a minute to appreciate what it took to make that smiley face



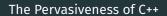
And act like it totally didn't take me like ... 2 hours to figure out how to do it.

C++

# The C++ Programming Language

C++ is a general-purpose programming language with a bias towards systems programming that:

- · Is a better C,
- · Supports data abstraction,
- · Supports object-oriented programming, and
- Supports generic programming.



# Why choose C++

### Brief History of C++

The C++ programming language is an extension of C that was developed by (the god himself) Bjarne Stroustrup in the early 1980s at Bell Laboratories.

# Example C++ Program (Hello World)

```
#include <iostream>

int main()

{
    cout << "Hello World!";
    return 0;
}</pre>
```

```
[user@pc]$ Hello World!
```

Environment Setup

# Setup and Installation

#### C++ Editors

# Compiling

Editing, Compiling, and Execution

### A Simple Program to Add Two Numbers

The following is an example of a simple program written in C++.

```
#include <iostream>
   using namespace std;
3
   int main()
 5
       int num1, num2, total;
 6
 7
       cout << "Enter integers to be added:" << endl;</pre>
       cin >> num1 >> num2;
       total = num1 + num2;
10
       cout << "The sum is " << total << endl;</pre>
11
12
13
```

I double dare you to guess what this program does.

#### A Simple Program to Add Two Numbers

The following is an example of a simple program written in C++.

```
#include <iostream>
using namespace std;

int main()

int num1, num2, total;

cout << "Enter integers to be added:" << endl;
cin >> num1 >> num2;
total = num1 + num2;
cout << "The sum is " << total << endl;

return 0;
}</pre>
```

This program is designed to read two numbers typed by the user at the keyboard; Compute their sum and display the result on the screen.

### A Simple Program to Add Two Numbers

The following is an example of a simple program written in C++.

```
#include <iostream>
   using namespace std;
3
   int main()
 5
       int num1, num2, total;
 6
 7
       cout << "Enter integers to be added:" << endl;</pre>
       cin >> num1 >> num2;
       total = num1 + num2;
10
       cout << "The sum is " << total << endl;</pre>
11
12
13
```

What could we do to make understanding this program easier?

#### A Simple Program with Comments

#### Add comments!

```
#include <iostream>
   using namespace std;
 4
   int main()
6
       int num1, num2, total;
 7
 8
       cout << "Enter integers to be added:" << endl;</pre>
       cin >> num1 >> num2;
10
       total = num1 + num2;
11
       cout << "The sum is " << total << endl;</pre>
12
13
14
15
```

#### A Not So Simple Program Because OMG whoever wrote this is an ass

#### Or hey, if you want to guarantee yourself a job

```
#include <iostream>
using namespace std;
int main(){
   int n,b,memes = 42;
cout<<"gimmie:" <<endl;
   cin>>n>>b;
memes=n+b;
cout<<"got em" <<memes+1-1<<endl;
return (pow(meme, 0) - 1);}</pre>
```

#### Program Structure and Syntax

C++ uses notation that may appear strange to non-programmers (and me). The notation is part of the programming language syntax.

**Syntax** Formal rules that specify the structure of a legal program.

### Program Structure and Syntax

The notation and explanations which follow will appear strange if you have never written a computer program.

Don't worry about them or how the program works. This will be explained in more detail later.

The following is an overview.

#### **Program Structure and Syntax**

Every C++ program consists of a header and a main body and has the following structure:

```
// Comment statements which are ignored by computer
/* Also a comment */
#include < header file name >
int main()
{
    declaration of variables;
    statements;
}
return 0;
```

```
using namespace std;
 4
   int main()
6
       int num1, num2, total;
       cout << "Enter integers to be added:" << endl;</pre>
9
       cin >> num1 >> num2:
10
       total = num1 + num2;
11
       cout << "The sum is " << total << endl;</pre>
12
13
14
15
```

- Lines beginning with // indicate that the rest of the line is a **comment**.
- Comments are inserted by programmers to help people read and understand the program.
- · Can be placed anywhere in a program.

```
using namespace std;
 4
   int main()
6
       int num1, num2, total;
       cout << "Enter integers to be added:" << endl;</pre>
9
       cin >> num1 >> num2:
10
       total = num1 + num2;
11
       cout << "The sum is " << total << endl;</pre>
12
13
14
15
```

- Lines beginning with # are instructions to the compiler's preprocessor.
- The **include** instruction says "what follows is a file name, find that file and insert its contents right here".
- Here the file iostream contains the definitions of cin, cout.

```
using namespace std;
 4
   int main()
6
       int num1, num2, total;
       cout << "Enter integers to be added:" << endl;</pre>
       cin >> num1 >> num2;
10
       total = num1 + num2;
11
       cout << "The sum is " << total << endl;</pre>
12
13
14
15
```

- Specifies that names used in the program (ie. cin and cout) are defined in the standard libraries.
- This is used to avoid problems with other libraries which may also use these names.

```
using namespace std;
 4
   int main()
6
       int num1, num2, total;
       cout << "Enter integers to be added:" << endl;</pre>
9
       cin >> num1 >> num2:
10
       total = num1 + num2;
11
       cout << "The sum is " << total << endl;</pre>
12
13
14
15
```

- When the program is executed the instructions will be executed in the oder they appear in the main body of the program.
- The main body is delimited by main() and the curly braces { }.
- This line also specifies that main() will return a value of type integer (int) on its completion (see line 14).

```
using namespace std;
 4
   int main()
6
       int num1, num2, total;
       cout << "Enter integers to be added:" << endl;</pre>
9
       cin >> num1 >> num2:
10
       total = num1 + num2;
11
       cout << "The sum is " << total << endl;</pre>
12
13
14
15
```

- The opening (left) brace marks the beginning of the main body of the program.
- The main body consists of instructions which are declarations defining the data or statements on how the data should be processed.
- · All C++ declarations and statements must end with a semicolon;

```
using namespace std;
 4
   int main()
6
       int num1, num2, total;
       cout << "Enter integers to be added:" << endl;</pre>
9
       cin >> num1 >> num2:
10
       total = num1 + num2;
11
       cout << "The sum is " << total << endl;</pre>
12
13
14
15
```

- This is a declaration. The words num1 and num2 are the names of variables.
- A variable is a location in the computer's memory where a value can be stored for use by a program.
- The declaration also specifies the variable type

```
using namespace std;
 4
   int main()
6
       int num1, num2, total;
       cout << "Enter integers to be added:" << endl;</pre>
9
       cin >> num1 >> num2:
10
       total = num1 + num2;
11
       cout << "The sum is " << total << endl;</pre>
12
13
14
15
```

- This statement instructs the computer to output the string of characters contained between the quotation marks, followed by a new line endl.
- The location of the output is denoted by cout which in this case will be the terminal screen.

```
using namespace std;
 4
   int main()
6
       int num1. num2. total:
7
       cout << "Enter integers to be added:" << endl;</pre>
9
       cin >> num1 >> num2;
10
       total = num1 + num2:
11
       cout << "The sum is " << total << endl;</pre>
12
13
14
15
```

- This statement instructs the computer to read data typed in at the keyboard (standard input), denoted by **cin**.
- These values are **assigned to** (stored in) variables num1 and num2.

```
using namespace std;
 4
   int main()
 6
       int num1, num2, total;
       cout << "Enter integers to be added:" << endl;</pre>
9
       cin >> num1 >> num2:
10
       total = num1 + num2;
11
       cout << "The sum is " << total << endl;</pre>
12
13
14
15
```

#### Line 11

 This statement is an arithmetic expression which assigns the value of the expression num1 + num2 (sum of integer values stored at num1 and num2) to the variable total.

```
using namespace std;
   int main()
6
       int num1, num2, total;
       cout << "Enter integers to be added:" << endl;</pre>
9
       cin >> num1 >> num2;
10
       total = num1 + num2;
11
       cout << "The sum is " << total << endl;</pre>
12
13
14
15
```

#### Line 12

• Instructs the computer to display the value of the variable total.

```
using namespace std;
 4
   int main()
6
       int num1. num2. total:
 7
       cout << "Enter integers to be added:" << endl;</pre>
9
       cin >> num1 >> num2;
10
       total = num1 + num2:
11
       cout << "The sum is " << total << endl;</pre>
12
13
14
15
```

- The last instruction of every program is the return statement.
- The return statement with the int value 0 (zero) indicates to the operating system that the program has terminated successfully.

```
using namespace std;
 4
   int main()
6
       int num1, num2, total;
       cout << "Enter integers to be added:" << endl;</pre>
       cin >> num1 >> num2;
10
       total = num1 + num2;
11
       cout << "The sum is " << total << endl;</pre>
12
13
14
15
```

#### Line 15

 The closing (right) brace marks the end of the main body of the program.

```
using namespace std;
 4
   int main()
6
       int num1, num2, total;
       cout << "Enter integers to be added:" << endl;</pre>
9
       cin >> num1 >> num2:
10
       total = num1 + num2;
11
       cout << "The sum is " << total << endl;</pre>
12
13
14
15
```

#### Blank lines

- Lines 4, 8 and 13 are used to make the program more readable.
- · They will be ignore by the compiler.
- Whitespace (spaces, tabs and newlines) are also ignored (unless within quotation marks).

```
using namespace std;
 4
   int main()
6
       int num1, num2, total;
       cout << "Enter integers to be added:" << endl;</pre>
       cin >> num1 >> num2;
10
       total = num1 + num2;
11
       cout << "The sum is " << total << endl;</pre>
12
13
14
15
```

### Indentation

• It does not matter where you place statements, either on the same line or on separate lines.

## Development Environment & Development Cycle

C++ programs go through 3 main phases during development:

Editing Writing the program,

**Compiling** Translating the program to executable code and detecting syntax errors, and

**Debugging** Running the program and checking for logical errors.

## References I