

# Lecture 1: Introduction to C++ Programming

Curtin FIRST Robotics Club (FRC) Pre-season Training

---

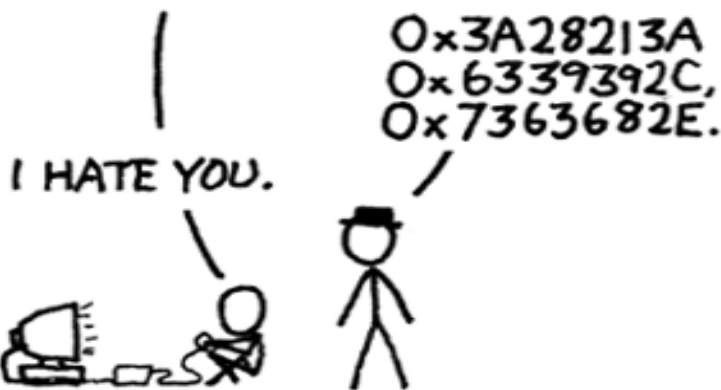
Scott Day

265815F@curtin.edu.au

October 15, 2016

Curtin University

MAN, I SUCK AT THIS GAME.  
CAN YOU GIVE ME  
A FEW POINTERS?



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

## Programming

---

# What is a Computer Program?

**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:



Recipe to writing programs:

1. Understand the problem.

Recipe to writing programs:

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

Recipe to writing programs:

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.

Recipe to writing programs:

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.

Recipe to writing programs:

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.

Recipe to writing programs:

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!

Recipe to writing programs:

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

Think like a computer!

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





## Example Code Appreciation

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



```
1 {center}
2 \begin{tikzpicture}
3 \path[fill=yellow,
4       draw=yellow!75!red]
5       (0,0) circle (1cm);
6 \fill[red] (45:5mm) circle (1mm);
7 \fill[red] (135:5mm) circle
8   \> (1mm);
9 \draw[line width=1mm,red]
10  \> (215:5mm) arc
11  \> (215:325:5mm);
12 \end{tikzpicture}
13 \end{center}
```

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



**99 little bugs in the code.**

**99 little bugs in the code.**

**Take one down, patch it around.**

**127 little bugs in the code...**

C++

---

If you visit [stroustrup.com/C++](http://stroustrup.com/C++), you will come across a plethora of information about the C++ programming language, direct from the designer of the language, Bjarne Stroustrup.

Bjarne lists a definition of C++ as:

"... 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."

- The language started in 1979 and was originally known as C with Classes.

- The language started in 1979 and was originally known as C with Classes.
- Essentially it meant that class files (used in object-oriented programming), were added to the C language.

- The language started in 1979 and was originally known as C with Classes.
- Essentially it meant that class files (used in object-oriented programming), were added to the C language.
- In 1983 it was renamed to C++.



- The language started in 1979 and was originally known as C with Classes.
- Essentially it meant that class files (used in object-oriented programming), were added to the C language.
- In 1983 it was renamed to C++.
- C++ exists under the stewardship of a standards committee and became an ISO standard in 1998 with a revision in 2011 and a minor revision in 2014.

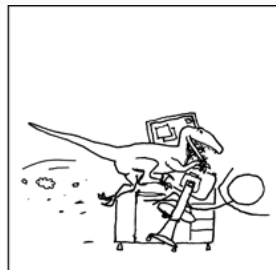
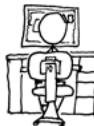
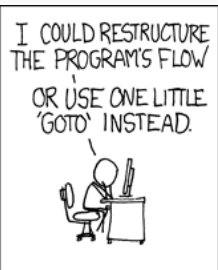
- The language started in 1979 and was originally known as C with Classes.
- Essentially it meant that class files (used in object-oriented programming), were added to the C language.
- In 1983 it was renamed to C++.
- C++ exists under the stewardship of a standards committee and became an ISO standard in 1998 with a revision in 2011 and a minor revision in 2014.

## Example C++ Program (Hello World)

```
1 #include <iostream>
2
3 int main()
4 {
5     cout << "Hello World!";
6     return 0;
7 }
8
```

```
[user@pc]$ g++ -c helloworld.cpp
[user@pc]$ g++ helloworld.o -o helloworld
[user@pc]$ ./helloworld
[user@pc]$ Hello World!
```

# goto, Your Worst Enemy



## Environment Setup

---

The easiest way to compile console programs depends on the particular tool you are using.

The easiest way for beginners to compile C++ programs is by using an Integrated Development Environment (IDE).

An IDE generally integrates several development tools, including a text editor and tools to compile programs directly from it.

Some IDE's:

- Visual Studio,
- CLion

Or you can use a text editor:

- Atom (pretty baller),
- Sublime Text,
- Notepad++

Sample compilers:

- GCC (use MinGW for windows),
- Clang

Refer to the relevant documentation of whatever tool/compiler you choose to use.



# Compiling

The typical filename extensions are:

- ".cpp" for a C++ source file.
- ".hpp" for a C++ header file.

Gcc can compile C++ as well as C:

```
[user@pc]$ gcc -c file1.cpp
[user@pc]$ gcc -c file2.cpp
[user@pc]$ gcc file1.o file2.o -o prog -lstdc++
```

- The ".cpp" extension tells gcc that it's dealing with C++ code.
- "-o name" gives the output filename. Without it the executable will be called "a.out", which is silly.
- "-lstdc++" tells gcc to link against the C++ library.

Alternatively, you can use "g++":

```
[user@pc]$ g++ file1.o file2.o -o prog
```

You can use Makefiles to simplify the process.

```
1 target: dependencies
2 [tab] system command
3
```

This syntax applied to our example would look like:

```
1 all:
2     g++ file1.cpp file2.cpp -o hello
3
```

To run this makefile, type:

```
[user@pc]$ make
```

## Lets Just Remember the Cats



## Editing, Compiling, and Execution

---

# A Simple Program to Add Two Numbers

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

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int num1, num2, total;
7
8     cout << "Enter integers to be added:" << endl;
9     cin >> num1 >> num2;
10    total = num1 + num2;
11    cout << "The sum is " << total << endl;
12
13    return 0;
14 }
```

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++.

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int num1, num2, total;
7
8     cout << "Enter integers to be added:" << endl;
9     cin >> num1 >> num2;
10    total = num1 + num2;
11    cout << "The sum is " << total << endl;
12
13    return 0;
14 }
```

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++.

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int num1, num2, total;
7
8     cout << "Enter integers to be added:" << endl;
9     cin >> num1 >> num2;
10    total = num1 + num2;
11    cout << "The sum is " << total << endl;
12
13    return 0;
14 }
```

What could we do to make understanding this program easier?

# A Simple Program with Comments

Add comments!

```
1 // Program to add two integers typed by user at keyboard
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int num1, num2, total;
8
9     cout << "Enter integers to be added:" << endl;
10    cin >> num1 >> num2;
11    total = num1 + num2;
12    cout << "The sum is " << total << endl;
13
14    return 0;
15 }
```



Or hey, if you want to guarantee yourself a job

```
1 #include <iostream>
2 using namespace std;
3 int main(){
4     int n,b,memes = 42;
5     cout<<"gimmie:" <<endl;
6     cin>>n>>b;
7     memes=n+b;
8     cout<<"got em" <<memes+1-1<<endl;
9     return (pow(meme, 0) - 1);}
```

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.

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.

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

```
1 // Comment statements which are ignored by computer
2 /* Also a comment */
3 #include < header file name >
4
5 int main()
6 {
7     declaration of variables;
8     statements;
9
10    return 0;
11 }
```

# Program Structure and Syntax

```
1 // Program to add two integers typed by user at keyboard
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int num1, num2, total;
8
9     cout << "Enter integers to be added:" << endl;
10    cin >> num1 >> num2;
11    total = num1 + num2;
12    cout << "The sum is " << total << endl;
13
14    return 0;
15 }
```

## Line 1

- 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.

# Program Structure and Syntax

```
1 // Program to add two integers typed by user at keyboard
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int num1, num2, total;
8
9     cout << "Enter integers to be added:" << endl;
10    cin >> num1 >> num2;
11    total = num1 + num2;
12    cout << "The sum is " << total << endl;
13
14    return 0;
15 }
```

## Line 2

- 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**.

# Program Structure and Syntax

```
1 // Program to add two integers typed by user at keyboard
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int num1, num2, total;
8
9     cout << "Enter integers to be added:" << endl;
10    cin >> num1 >> num2;
11    total = num1 + num2;
12    cout << "The sum is " << total << endl;
13
14    return 0;
15 }
```

## Line 3

- 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.

# Program Structure and Syntax

```
1 // Program to add two integers typed by user at keyboard
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int num1, num2, total;
8
9     cout << "Enter integers to be added:" << endl;
10    cin >> num1 >> num2;
11    total = num1 + num2;
12    cout << "The sum is " << total << endl;
13
14    return 0;
15 }
```

## Line 5

- When the program is executed the instructions will be executed in the order 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).



# Program Structure and Syntax

```
1 // Program to add two integers typed by user at keyboard
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int num1, num2, total;
8
9     cout << "Enter integers to be added:" << endl;
10    cin >> num1 >> num2;
11    total = num1 + num2;
12    cout << "The sum is " << total << endl;
13
14    return 0;
15 }
```

Line 6

- 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;

# Program Structure and Syntax

```
1 // Program to add two integers typed by user at keyboard
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int num1, num2, total;
8
9     cout << "Enter integers to be added:" << endl;
10    cin >> num1 >> num2;
11    total = num1 + num2;
12    cout << "The sum is " << total << endl;
13
14    return 0;
15 }
```

Line 7

- 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**

# Program Structure and Syntax

```
1 // Program to add two integers typed by user at keyboard
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int num1, num2, total;
8
9     cout << "Enter integers to be added:" << endl;
10    cin >> num1 >> num2;
11    total = num1 + num2;
12    cout << "The sum is " << total << endl;
13
14    return 0;
15 }
```

## Line 9

- 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.

# Program Structure and Syntax

```
1 // Program to add two integers typed by user at keyboard
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int num1, num2, total;
8
9     cout << "Enter integers to be added:" << endl;
10    cin >> num1 >> num2;
11    total = num1 + num2;
12    cout << "The sum is " << total << endl;
13
14    return 0;
15 }
```

Line 10

- 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`.

# Program Structure and Syntax

```
1 // Program to add two integers typed by user at keyboard
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int num1, num2, total;
8
9     cout << "Enter integers to be added:" << endl;
10    cin >> num1 >> num2;
11    total = num1 + num2;
12    cout << "The sum is " << total << endl;
13
14    return 0;
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`.

# Program Structure and Syntax

```
1 // Program to add two integers typed by user at keyboard
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int num1, num2, total;
8
9     cout << "Enter integers to be added:" << endl;
10    cin >> num1 >> num2;
11    total = num1 + num2;
12    cout << "The sum is " << total << endl;
13
14    return 0;
15 }
```

Line 12

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

# Program Structure and Syntax

```
1 // Program to add two integers typed by user at keyboard
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int num1, num2, total;
8
9     cout << "Enter integers to be added:" << endl;
10    cin >> num1 >> num2;
11    total = num1 + num2;
12    cout << "The sum is " << total << endl;
13
14    return 0;
15 }
```

Line 14

- 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.

# Program Structure and Syntax

```
1 // Program to add two integers typed by user at keyboard
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int num1, num2, total;
8
9     cout << "Enter integers to be added:" << endl;
10    cin >> num1 >> num2;
11    total = num1 + num2;
12    cout << "The sum is " << total << endl;
13
14    return 0;
15 }
```

Line 15

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



# Program Structure and Syntax

```
1 // Program to add two integers typed by user at keyboard
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int num1, num2, total;
8
9     cout << "Enter integers to be added:" << endl;
10    cin >> num1 >> num2;
11    total = num1 + num2;
12    cout << "The sum is " << total << endl;
13
14    return 0;
15 }
```

## Blank lines

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

# Program Structure and Syntax

```
1 // Program to add two integers typed by user at keyboard
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int num1, num2, total;
8
9     cout << "Enter integers to be added:" << endl;
10    cin >> num1 >> num2;
11    total = num1 + num2;
12    cout << "The sum is " << total << endl;
13
14    return 0;
15 }
```

## Indentation

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

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

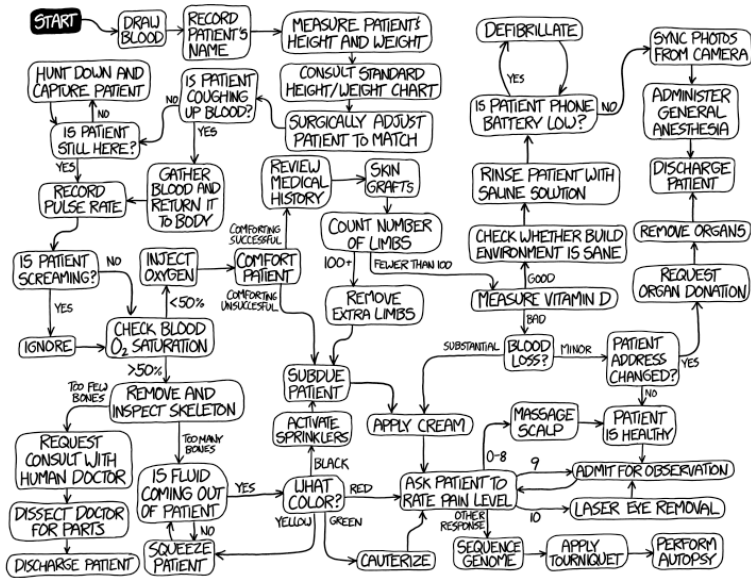
**Editing** Writing the program,

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

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

# Check this out

## A GUIDE TO THE MEDICAL DIAGNOSTIC AND TREATMENT ALGORITHM USED BY IBM'S WATSON COMPUTER SYSTEM



I should fill this out