Lecture 1: Introduction to C++ Programming

Curtin FIRST Robotics Club (FRC) Pre-season Training

Scott Day
265815F@curtin.edu.au
October 19, 2016

Curtin University

Insert Mandatory Programming Joke

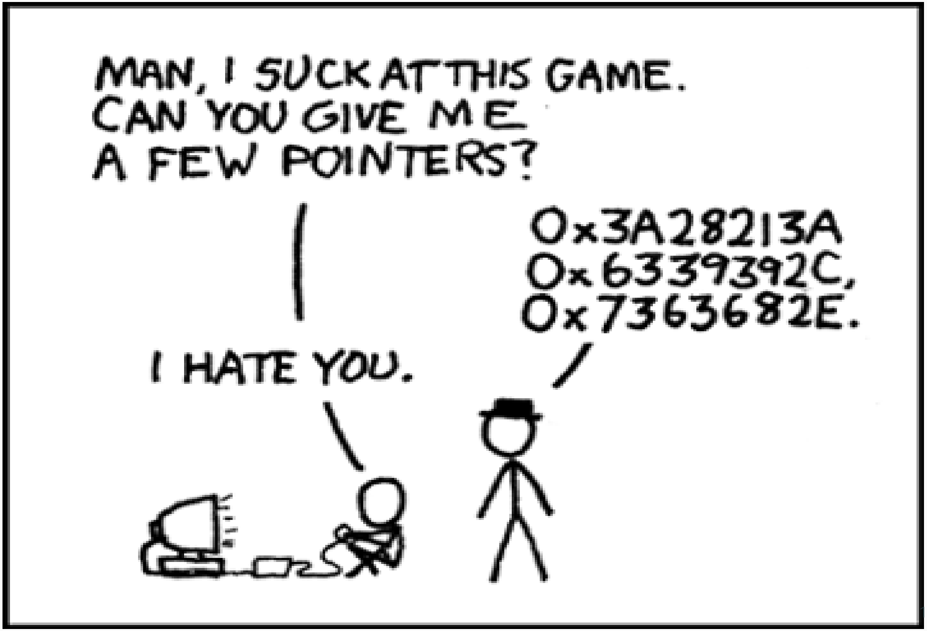

MAN, I SUCK AT THIS GAME. CAN YOU GIVE ME A FEW POINTERS?
0x3A28213A 0x6339392C, 0x7363682E.
I HATE YOU.

Table of contents

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.

Problem Solving

Recipe to writing programs:

Problem Solving

Recipe to writing programs:

1. Understand the problem.

Problem Solving

Recipe to writing programs:

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

Problem Solving

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.

Problem Solving

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.

Problem Solving

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.

Problem Solving

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!

Problem Solving

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

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

```
{center}
\begin{tikzpicture}
\path[fill=yellow,
        draw=yellow!75!red]
        (0,0) circle (1cm);
\fill[red] (45:5mm) circle (1mm);
\fill[red] (135:5mm) circle
↪   (1mm);
\draw[line width=1mm,red]
↪   (215:5mm) arc
↪   (215:325:5mm);
\end{tikzpicture}
\end{center}
```

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

Lets Just Remember the Cats

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++

The C++ Programming Language

If you visit 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.”

Brief History of C++

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

Brief History of 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.

Brief History of 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++.

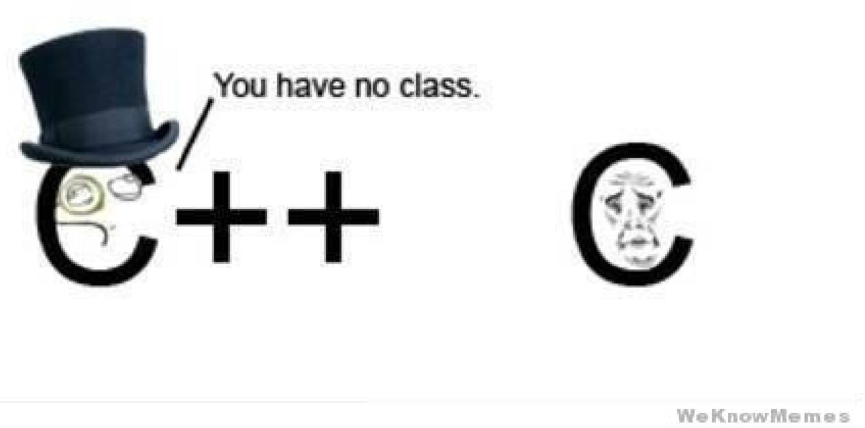
Brief History of C++

- 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.

Brief History of 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.

If You Understand This, I Have Nothing Left To Teach You

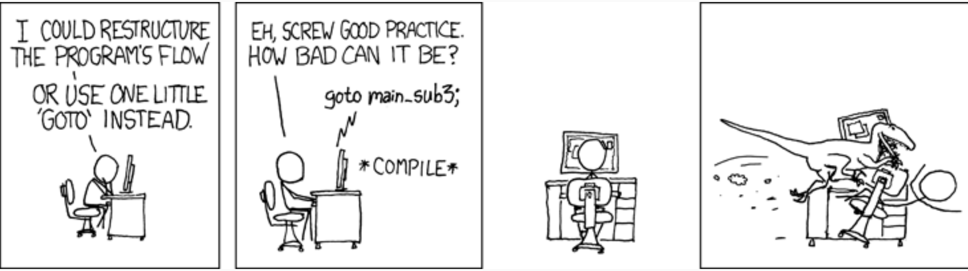


Example C++ Program (Hello World)

```cpp
#include <iostream>

int main()
{
    cout << "Hello World!";
    return 0;
}

```

```
[user@pc]$ g++ -std=c++11 -o helloworld *.cpp
[user@pc]$ ./helloworld
[user@pc]$ Hello World!
```

goto, Your Worst Enemy


I COULD RESTRUCTURE THE PROGRAM'S FLOW
OR USE ONE LITTLE 'GOTO' INSTEAD.
EH, SCREW GOOD PRACTICE. HOW BAD CAN IT BE?
goto main_sub3;
COMPILE

Environment Setup

C++ Editors

The easiest way to compile console programs depeds 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.

Samples

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

Setup and Installation

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.

Makefiles

```
target: dependencies
[tab] system command

```

This syntax applied to our example would look like:

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

```

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++.

```cpp
#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;
}
```

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++.

```cpp
#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;
}
```

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;

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;
}
```

What could we do to make understanding this program easier?

A Simple Program with Comments

Add comments!

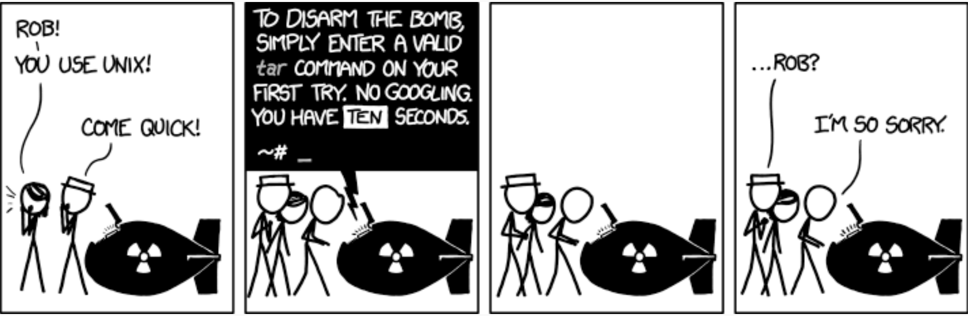
```cpp
// Program to add two integers typed by user at keyboard
#include <iostream>
using namespace std;

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;
}
```

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);}
```

Literally My Group Project This Semester


ROB!
YOU USE UNIX!
COME QUICK!
TO DISARM THE BOMB, SIMPLY ENTER A VALID tar COMMAND ON YOUR FIRST TRY. NO GOOGLING. YOU HAVE TEN SECONDS.
~# _
...ROB?
I'M SO SORRY.

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:

```cpp
// Comment statements which are ignored by computer
/* Also a comment */
#include < header file name >

int main()
{
    declaration of variables;
    statements;

    return 0;
}
```

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 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).

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 ignore by the compiler.
- **Whitespace** (spaces, tabs and newlines) are also ignored (unless within quotation marks).

Program Structure and Syntax

```cpp
// Program to add two integers typed by user at keyboard
#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;
}
```

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.

Programming Challenge

Calculator Challenge

The following program should function as a basic calculator; it should ask the user to input what type of arithmetic operation he would like, and then ask for the numbers on which the operation should be performed. The calculator should then give the output of the operation.

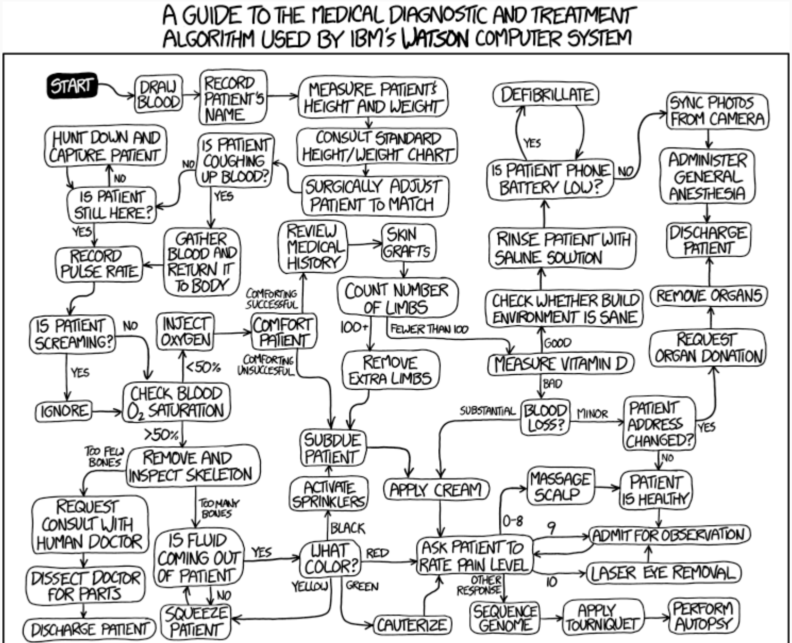
Calculator Challenge Problem Code

```
         <iostream>

___ multiply(int x, int y)
{
    ______ x _ y;
}

____ divide(int x, int y)
{
    _____ x _ y;
}

_____ add(int x, int y)
{
    ______ x _ y;
}

______ subtract(int x, int y)
{
    _____ x _ y;
}

using namespace std;

___ _____()
{
         op = 'c';
```

Calculator Challenge Solution Code

```cpp
#include <iostream>

int multiply(int x, int y)
{
    return x * y;
}

int divide(int x, int y)
{
    return x / y;
}

int add(int x, int y)
{
    return x + y;
}

int subtract(int x, int y)
{
    return x - y;
}

using namespace std;

int main()
{
    char op = 'c';
```

Check this out



References I

I should fill this out