Introduction to C++

Pengfei Zhao

Natural Language

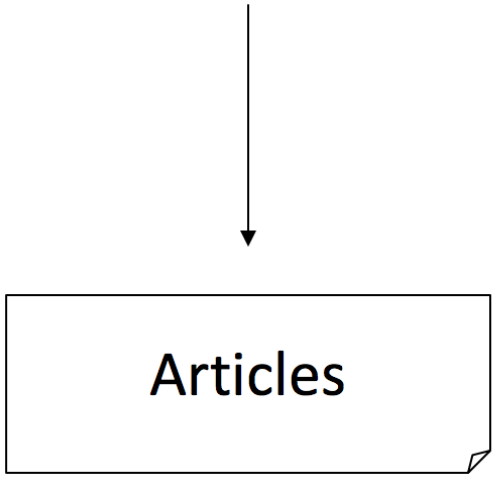
- Recognized by human beings
  - Chinese, English, German, French, …
- To define a natural language, we need to define
  - Syntax (grammar)
  - Semantics (meaning)

Programming Language

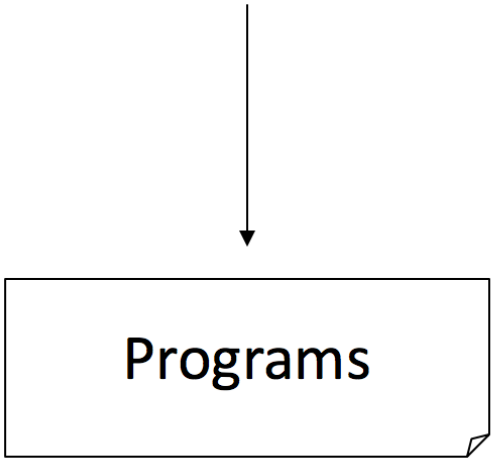
- Recognized by computers
- To define a programming language, we need also to define
  - Syntax
  - Semantic
- You can invent a programming language too!

Programming Language & Natural Language

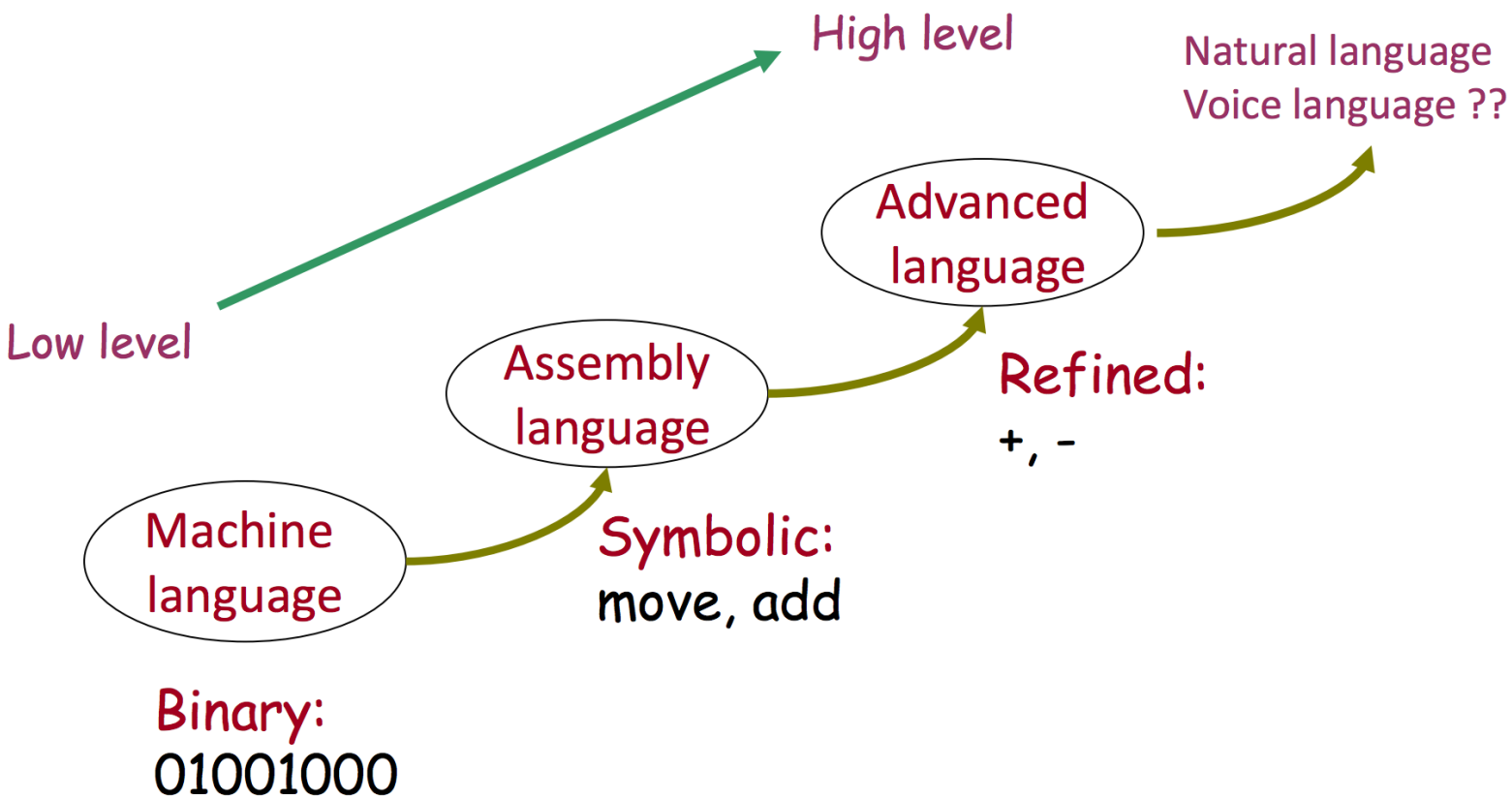
Natural language



Programming language



Evolution of Programming Language



Machine Language

- A CPU accepts instructions in machine language
- An instruction consists of 0s and 1s (binary number)
- Difficult for human to understand
- E.g.,

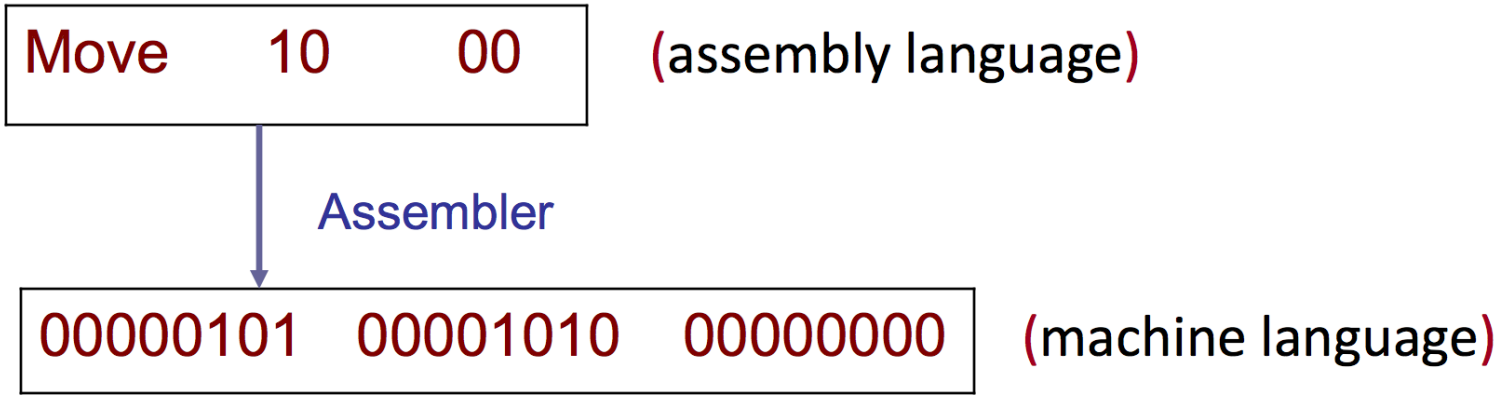
00000101	00010000	00000000
Move	**Value**	**Address**

Machine Language

```
10111001    00000000
11010010    10100001
00000100    00000000
10001001    00000000
00001110    10001011
00000000    00011110
00000000    00000010
10111001    00000000
11100001    00000011
00010000    11000011
10001001    10100011
00001110    00000100
00000010    00000000
```

Assembly Language

- An assembly language uses symbols to represent the machine language instructions.
- An assembler is needed to translates symbolic code into machine language



Assembly Language

```
mult.asm        Wed Feb 02 15:42:09 2000        1
; Impliments a simple 32 bit integer multiply by successive addition
; R. H. Klenke, Sun Jan 31 10:45:14 EST 1999
;

Multp:  .equ    32              ; multiplicand
Multd:  .equ    64              ; multiplier
        .org    4096            ; 0x1000
        lar     r30, Done       ; Load address of Done for branch
        lar     r31, Loop       ; Load address of Loop for branch
        la      r1, Multd       ; Load multiplier
        la      r2, Multp       ; Load multiplicand
        andi    r3, r3, #0      ; clear r3
        andi    r4, r4, #0      ; clear r4
        addi    r5, r3, #1      ; place 1 in r5
        neg     r3, r5          ; place -1 in r3
Loop:   add     r4, r4, r2      ; add multiplicand to running sum
        add     r1, r1, r3      ; start loop, decrement multiplier
        brzr    r30,r1          ; jump to Done if multiplier = 0
        br      r31             ; jump back to Loop
Done:   st      r4, Result      ; store result
        stop
        .org    8192            ; 0x2000
Result: .dw     1               ; storage for result
```

Advance Language

- Close to natural languages, using "if…then…else", etc.
- Make life easy for the programmers
- A compiler is needed to translate high level language programs into machine language instructions
- Examples
  - Java, **C, C++,** Pascal, Basic, Fortran,…

```
if (x > 0)
   y = y +1;
else
   y = y - 1
```

(high level language)

Compiler →

00001111	1111111	10100101
01011111	00010101	01010101
01010101	01010010	00000000
01011111	00010101	01010101
00001111	1111111	10100101
00001111	1111111	10100101

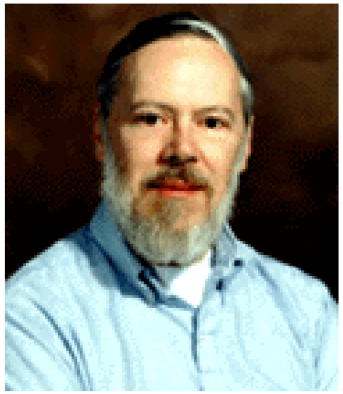
(machine language)

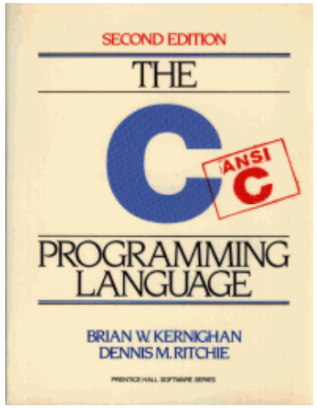
Advance Language

Language Rank	Types	Spectrum Ranking
1. Java		100.0
2. C		99.2
3. C++		95.5
4. Python		93.4
5. C#		92.2
6. PHP		84.6
7. Javascript		84.3
8. Ruby		78.6
9. R		74.0
10. MATLAB		72.6

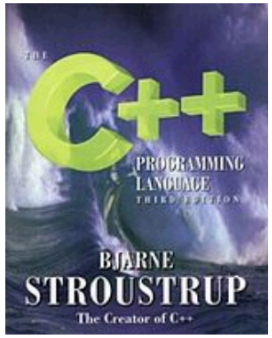
C Programming Language

- C is a high level language
- Created by Dennis M. Ritchie in 1972
  - First textbook: K&R, *The C Programming Language*.
- ANSI (American National Standards Institute)
  - First edition: ANSI C 1983.
  - Second edition: ANSI C 1999





C++ Programming Language



- C++ (see plus plus) is a combination of both high-level and low-level language features
- Developed by Bjarne Stroustrup in 1979 at Bell Labs
- Enhancements to the C programming language
  - add object-oriented (OO) features, such as classes
  - Ever named C with Classes
  - C++ (since 1983)
- C++ is one of the most popular programming languages.

Compiler

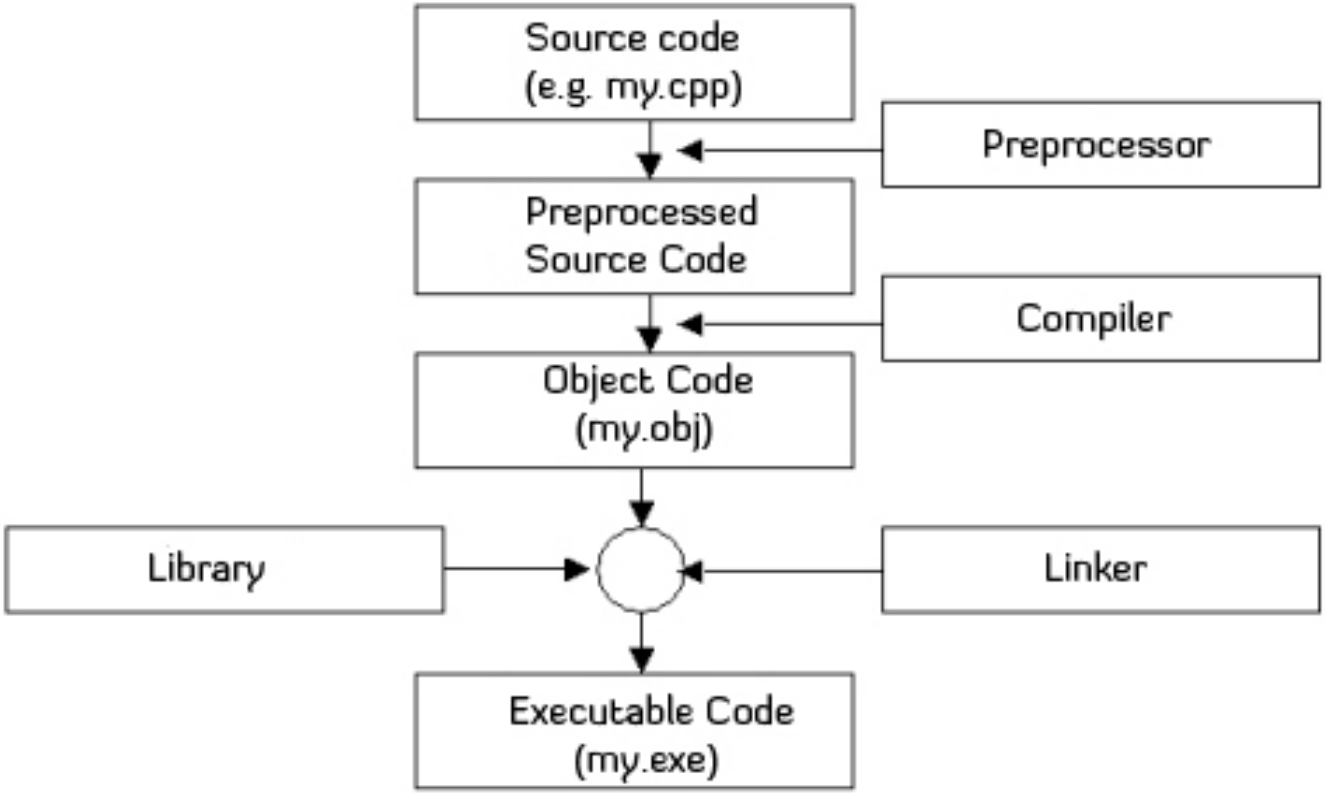
- Computers understand machine language (ones and zeros).
- Input two numbers, add the two numbers together, and displays the total:

00000	10011110
00001	11110100
00010	10011110
00011	11010100
00100	10111111
00101	00000000

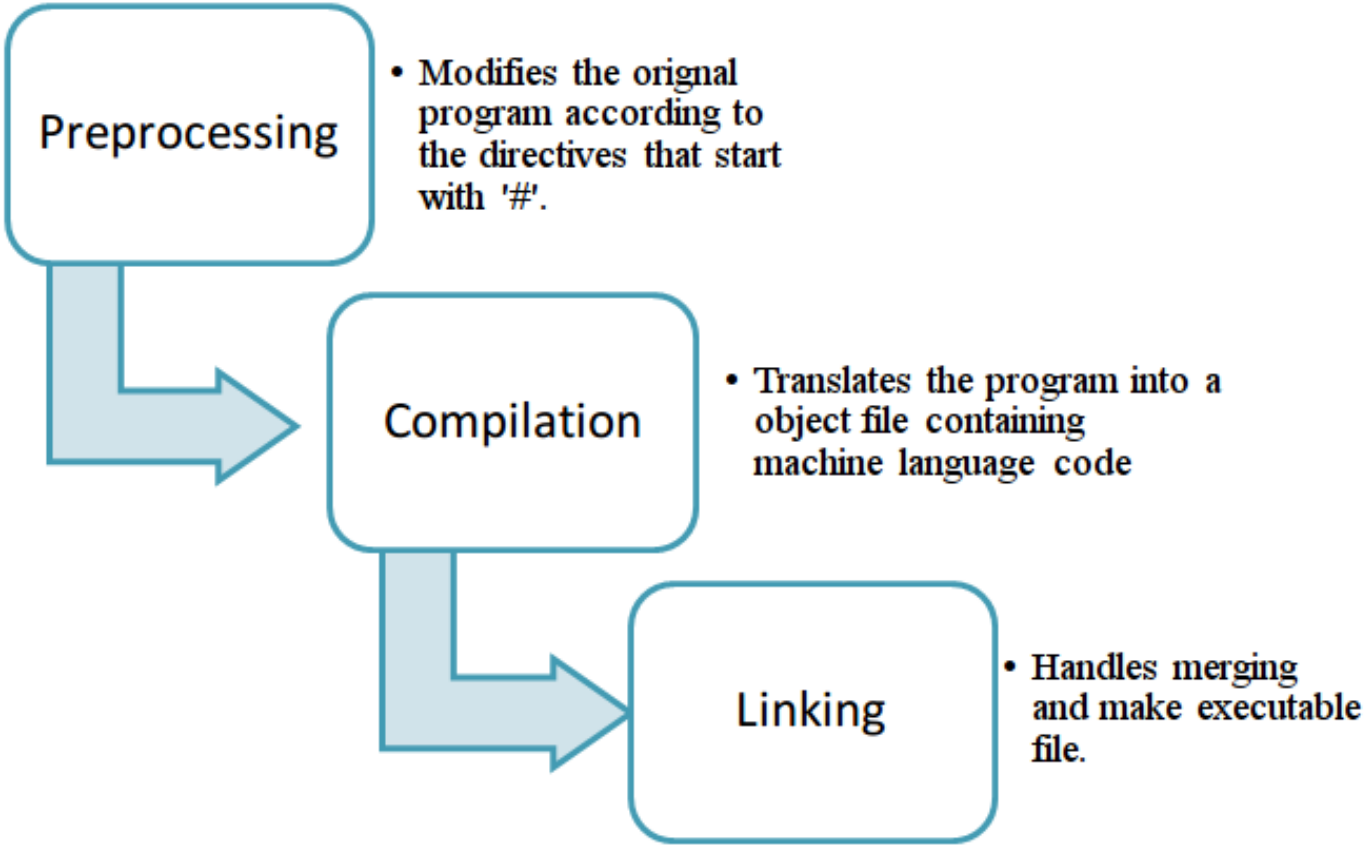
```
1 int a, b, sum;
2
3 cin >> a;
4 cin >> b;
5
6 sum = a + b;
7 cout << sum << endl;
```

- Compiler translates high level languages to machine language at some point.
- C++ is designed to be a compiled language.

Program Compilation and Execution



Program Compilation and Execution



Our First Program

```
/* Our first program */
#include<iostream>
using namespace std;
int main()
{
  cout << "Hello World!\n";
  return 0;
}
```

Guess what this program intends to do

Comments

```
/* Our first program */
#include<iostream>
using namespace std;
int main()
{
  cout << "Hello World!\n";
  return 0;
}
```

- A comment is used to explain some parts in the program.

Comments

- Two ways to insert comments into a C++ program
  - 1. // (double slash)
    - Only for one line, e.g., `//Our first program`
  - 2. /* */
    - Can be used for multiple lines
    - e.g., `/*our first program */` or

```
/*our first
  program
*/
```

- It is a good habit to insert comments into your programs
  - Programs (source code) will be more readable
  - Pay attention to CRA rubrics on how a program is assessed.

Preprocessor

```
/* Our first program */
#include<iostream>
using namespace std;
int main()
{
  cout << "Hello
World!\n";
  return 0;
}
```

Preprocessor:
tell compilers some info used in compiling.

Iostream: to use cout, the header file iostream must be included.

cout is declared in the standard name space std.

Main function

```
/* Our first program */
#include<iostream>
using namespace std;
int main()
{
  cout << "Hello World!\n";
  return 0;
}
```

- All the programs written in C must have a main function
  - `main`: function name
  - `int`: function return type
- The part between the '`{`' and the '`}`' is called body of function

Statements

Question: How many statements in this program?

```
/* Our first program */
#include<iostream>
using namespace std;
int main()
{
   cout << "Hello World!\n";
   return 0;
}
```

- A statement is an instruction telling a computer what to do
- A simple statement ends with ';'
- A statement can be a simple statement or a compound statement.

Good Habit: Indentation

```
/* Our first program */
#include<iostream>
using namespace std;
int main()
{
  cout << "Hello World!\n";
  return 0;
}
```

With indentation

```
/* Our first program */
#include<iostream>
using namespace std;
int main()
{
cout << "Hello World!\n";
return 0;
}
```

Without indentation

Questions:

- Any difference?
- Which one looks more clear?

Good Habit: Separate lines

```
1 int main ()
2 {
3   std::cout << " Hello World!";
4 }
```

```
int main () { std::cout << "Hello World!"; }
```

- In C++, the separation between statements is specified with an ending semicolon (;)

- Behavior of above two programs are the same.
- Good habit to separate different statements to different lines and make indentation.

cout

Object of ostream

Start at a new line

```
cout << "Hello World!\n";
```

Output information

Output:

```
Hello World!
```

cout

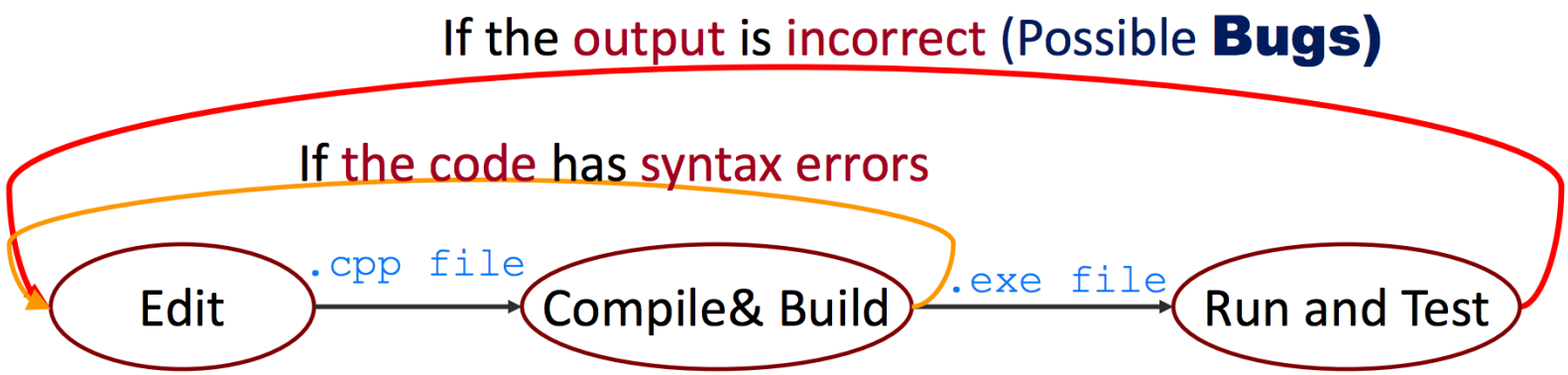
```
cout << "Hello World!\n";
```

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

Output:

```
Hello World!
```

How Does A Program Work (for a single source file)




```
/* Our first program */
#include<iostream>
using namespace std;
int main()
{
  cout << "Hello World!\n";
  return 0;
}
```

Source file
(sample.cpp)

```
00001111    1111111
01011111    00010101
01010101    01010010
01011111    00010101
```

Executable file
(sample.exe)

Class Exercises

What is the output of this program?

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

HelloWord1.cpp

Class Exercises

Guess what the output of this program is?

```
#include<iostream>
using namespace std;
int main()
{
  int value1, value2, sum;
  value1 = 50;
  value2 = 25;
  sum = value1 + value2;
  cout << value1 << '+' << value2 << '=' << sum << endl;
  return 0;
}
```

Sum.cpp