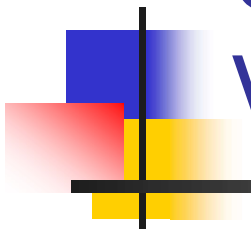
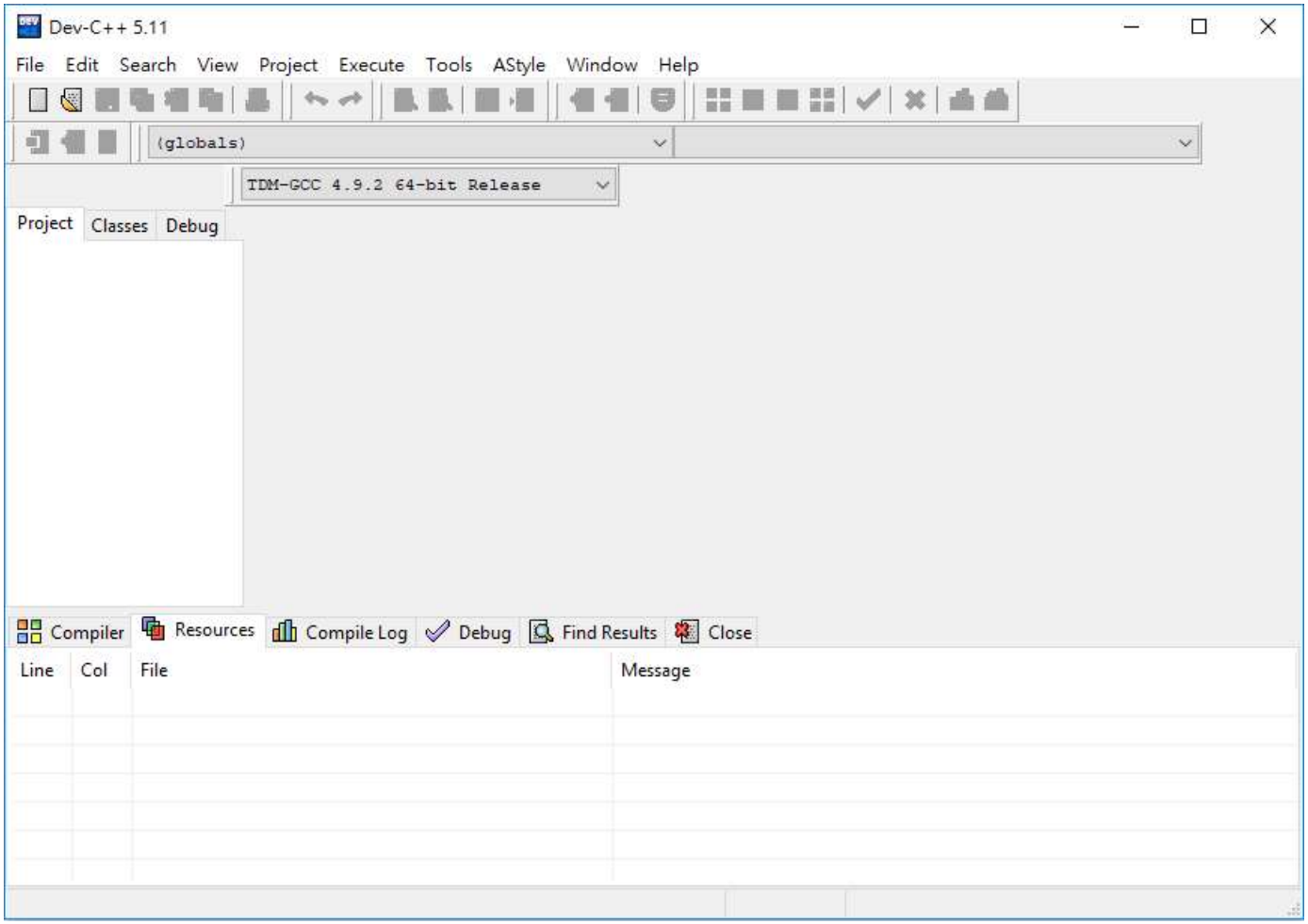


Introduction to Computer Programing C++ Ch1 Introduction and Variables and Assignment

National Taiwan University
Dept. of Chemical Engineering
Prof. Chengche Hsu

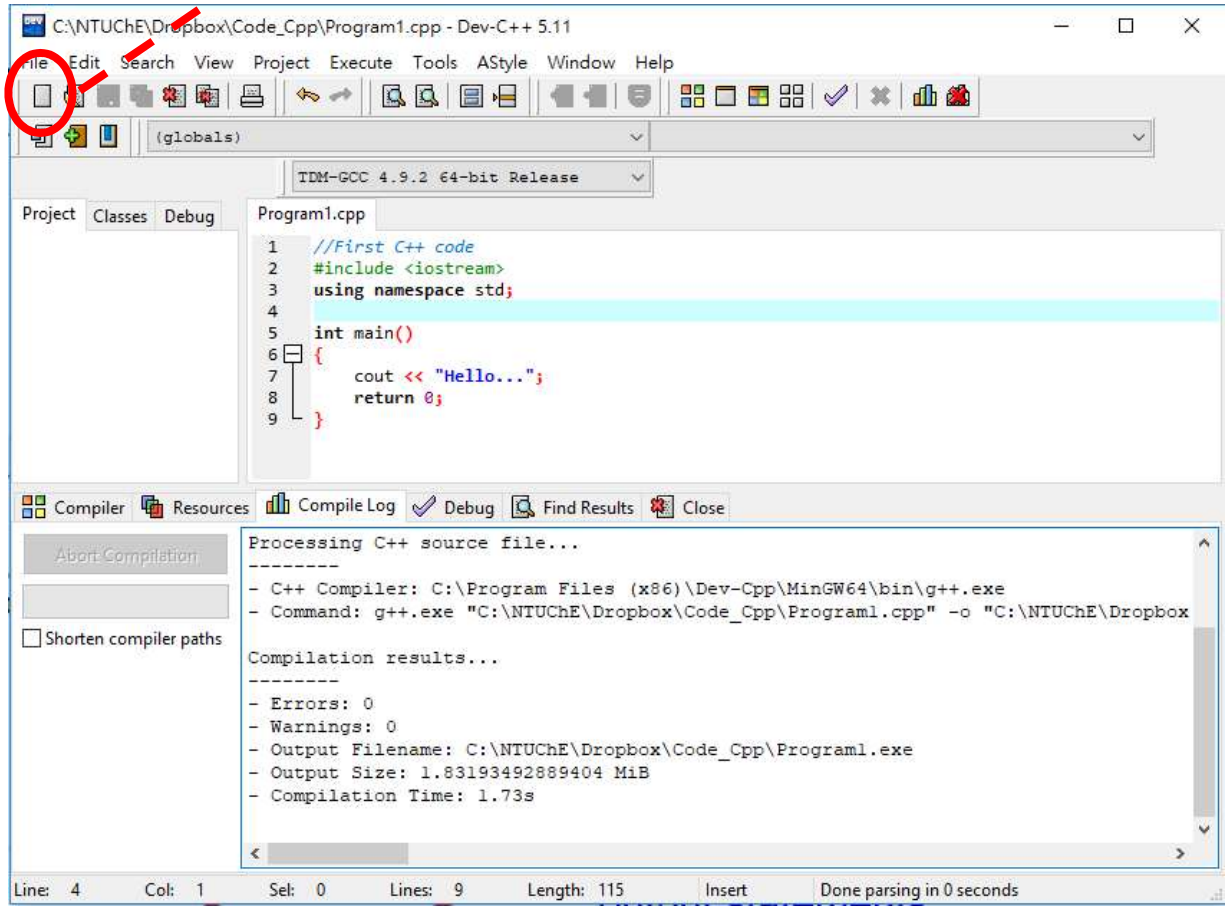
Dev C++ Download and Installation

https://sourceforge.net/projects/orwelldevcpp/



First C++ Program

New → Source file

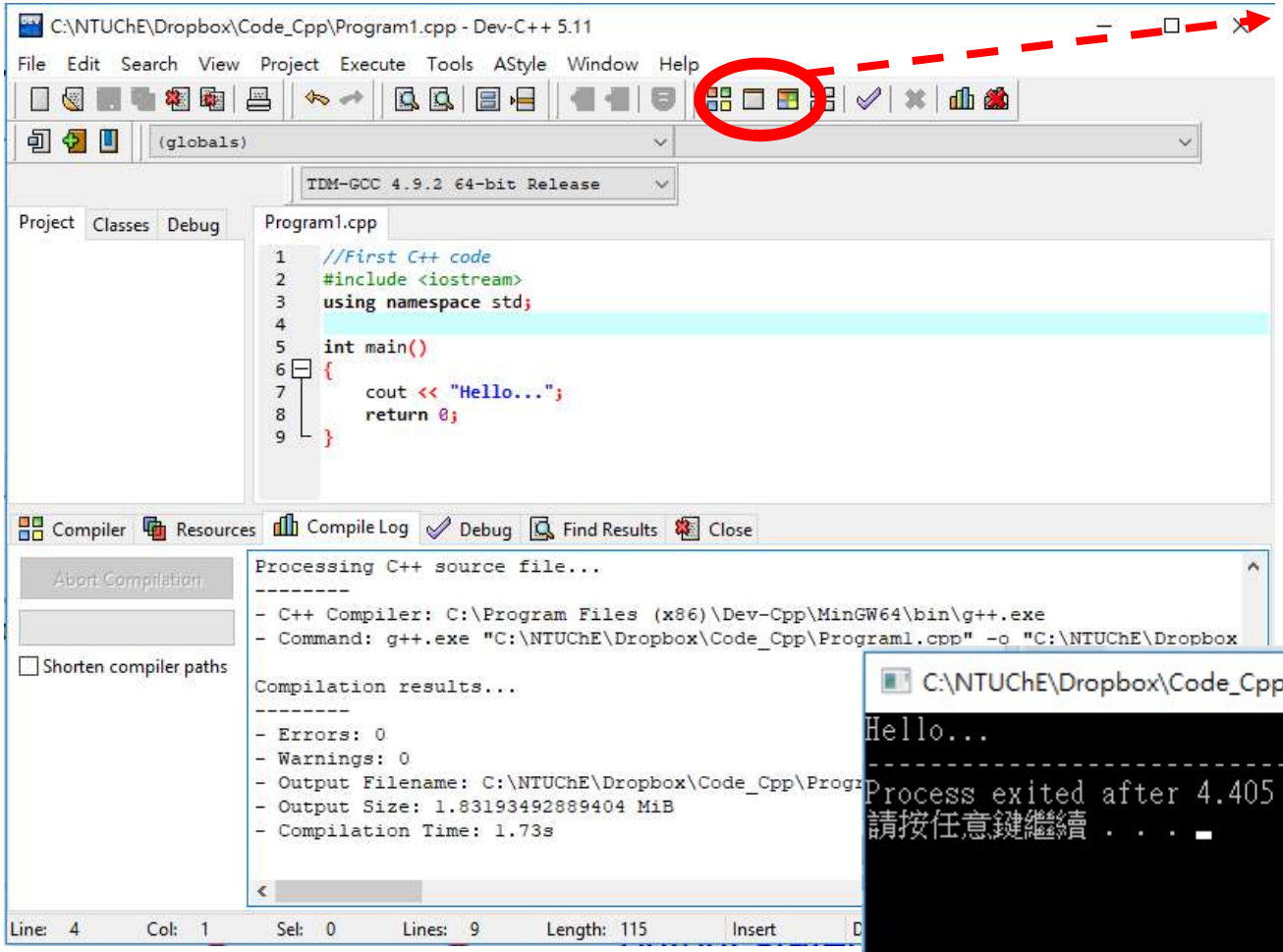


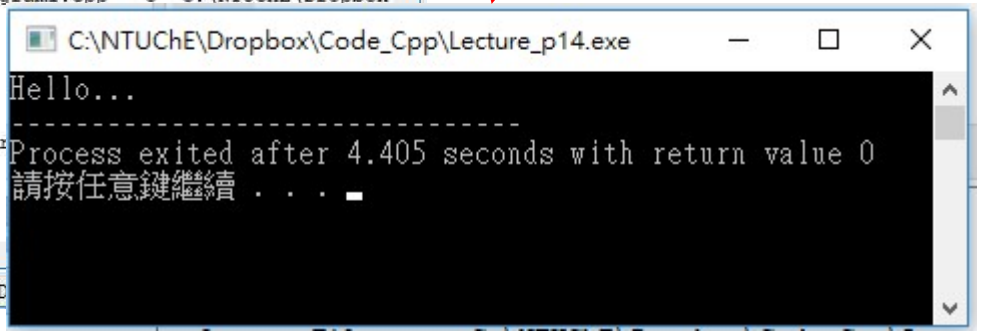

```
//First C++ code
#include <iostream>
using namespace std;

int main()
{
        cout << "Hello...";
        return 0;
}
```

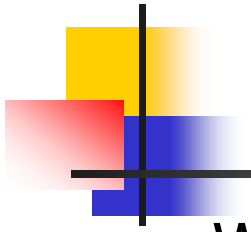
Compile and Run

- Compile (編譯)
- Run
- Compile & Run

Getting output



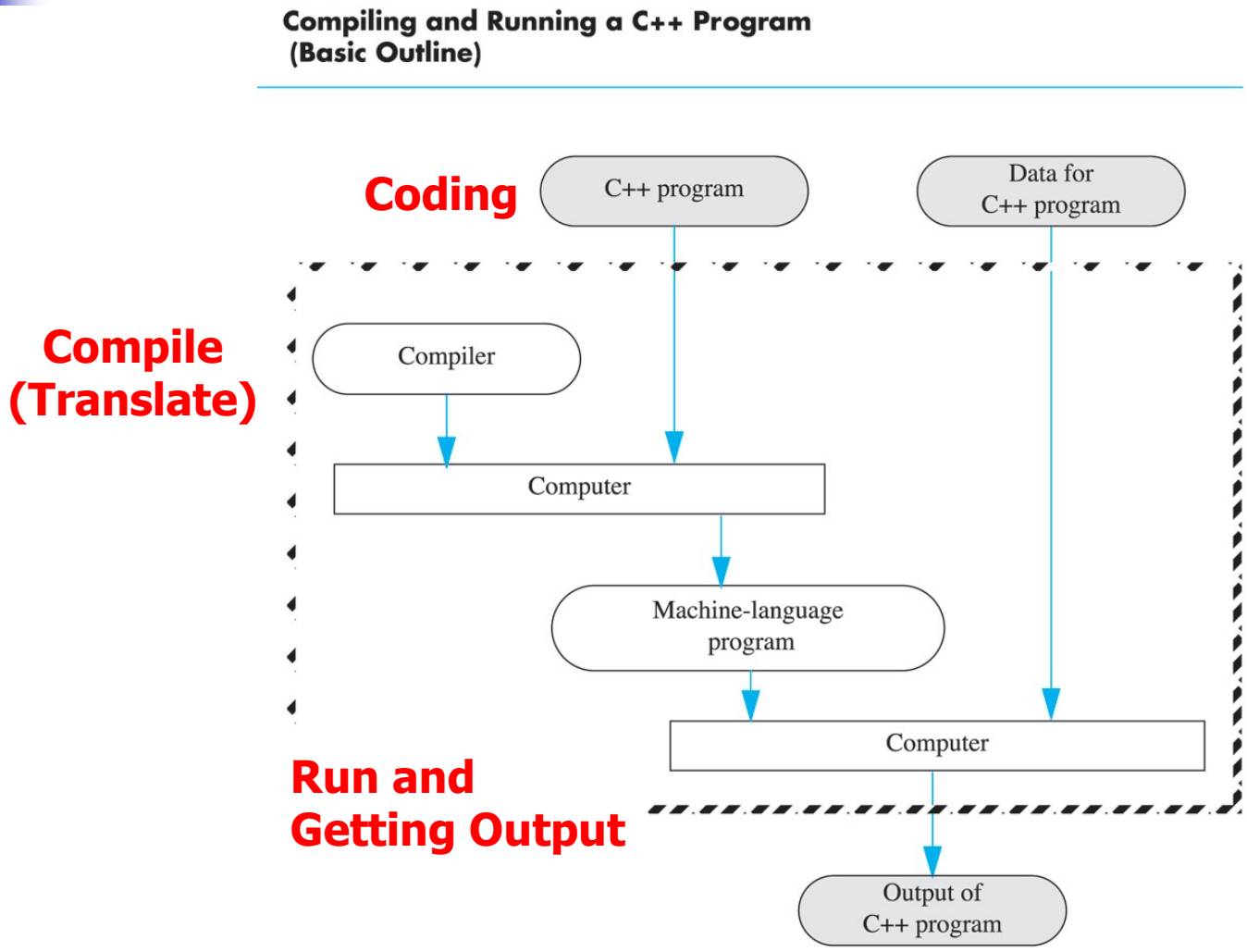




Error Messages

- What error messages would you get if you:
  - Remove ";" somewhere
  - Typo….
  - Remove {
  - Replace { } by ( )
  - Add one space before "iostream"
  - ….
- Pay attention to error messages from NOW on.

What Did We Just Do?



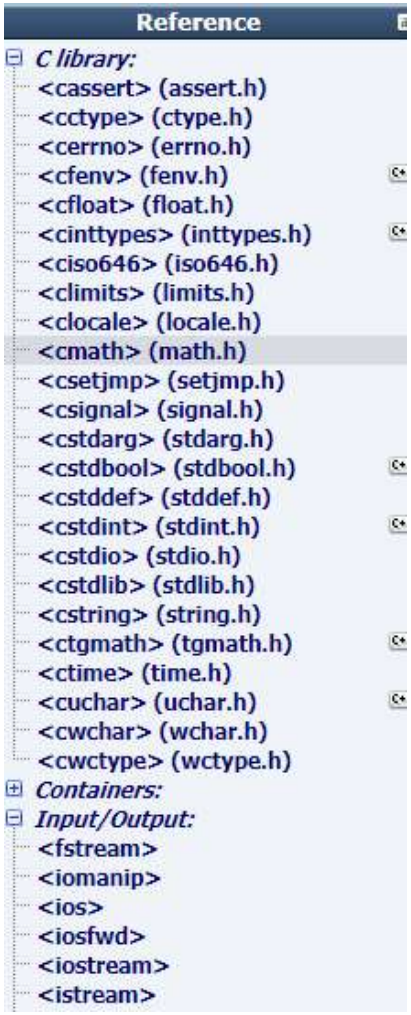
Structure of Code

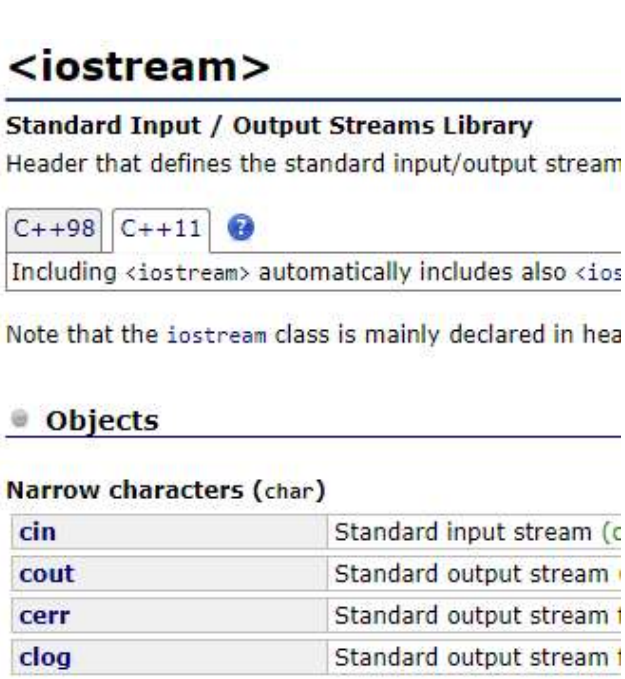
```
//First C++ code

#include <iostream>

using namespace std;

int main()
{
        cout << "Hello...";
        return 0;
}
```

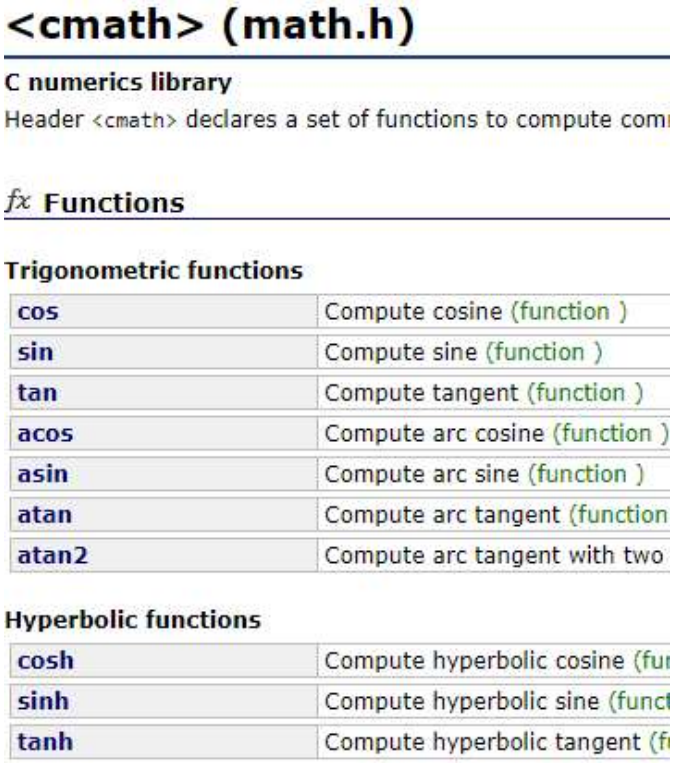
- Comment: For programmers, C++ does not read it.
- Include directive: Inclusion of the standard library "iostream", for input and out using keyboard and screen, respectively.
- Using directive: inclusion of several "standard" names (e.g. cout).
- The main() function: every C++ program has ONE and ONLY ONE main() function.
  - Output statement
  - Return value for main() function. Every function must have a return value.
- Note
  - ; at the end of each line.
  - { }

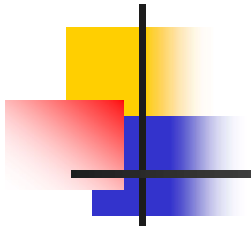
More about Inclusion Directive….

http://www.cplusplus.com/reference/









Key Component of a Program

- **Data**
  - **Basic (int,double,char,…)**
  - **Pointer**
  - Composites (array, struct, class)
- **Data (numeric) Operation**
  - **+, -, *, /, etc.**
- **Process flow control**
  - **Logical Operation**
  - **If-else**
  - **for-loop**
  - **While-loop**
- **Input/Output**
  - **Screen**
  - **Keyboard**
  - Disk (to File)
- **Functions**

Variable (Data) Type

DISPLAY 2.2 Some Number Types

Type Name	Memory Used	Size Range	Precision
short (also called *short int*)	2 bytes	-32,768 to 32,767	(not applicable)
int	4 bytes	-2,147,483,648 to 2,147,483,647	(not applicable)
long (also called *long int*)	4 bytes	-2,147,483,648 to 2,147,483,647	(not applicable)
float	4 bytes	approximately 10^{-38} to 10^{38}	7 digits
double	8 bytes	approximately 10^{-308} to 10^{308}	15 digits
long double	10 bytes	approximately 10^{-4932} to 10^{4932}	19 digits

These are only sample values to give you a general idea of how the types differ. The values for any of these entries may be different on your system. **Precision** *refers to the number of meaningful digits, including digits in front of the decimal point. The ranges for the types* **float**, **double**, *and* **long double** *are the ranges for positive numbers. Negative numbers have a similar range, but with a negative sign in front of each number.*

p.95 Textbook

More about Variable (Data) Type

1 byte=8 bits

0	1	1	0	1	1	1	0

8 bits: 11111111 = $2^7+2^6+\ldots+2^1+2^0 = 2^8-1 = 255$

Type Name	Memory Used	Size Range	Precision
short (also called *short int*)	2 bytes	-32,768 to 32,767	(not applicable)
int	4 bytes	-2,147,483,648 to 2,147,483,647	(not applicable)

2 bytes = 16 bits → $2^{16}-1 = 65536-1$
4 bytes = 32 bits → $2^{32}-1$

Variable Declaration

Syntax 1:

Type_name variable1, variable2, …;

variable1=XXX;

variable2=YYY;

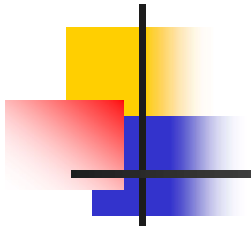
Syntax 2:

Type_name var3=XXX, var4=YYY;

Notes

- Variable name:
  - Start with a letter,
  - Letters, _, and numbers are allowed.
- Case sensitive: VarTest vs. vartest.
- "=": assignment!!
- using those have been used.

```
//Variable Declaration
#include <iostream>
using namespace std;

int main()
{
        int a=2;
        int b=3, test;
        cout << "a=" << a;
        cout << "b=" << b;
        cout << "test=" << test;
        return 0;
}
```



Output

Syntax for cout (on the screen):

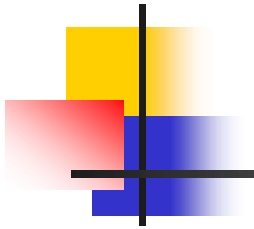
```
cout << "text" << variable1;
cout << "text" << variable2 <<endl;
cout << "text" << variable3 << "\n";
```

```
//Input and Output
#include <iostream>
using namespace std;

int main(){
        int a=3, b=4;
        cout << "a=" << a << endl;
        cout << "the number is " << a << "\n";
        }
```

Notes:

- All the variables have to declare before using.
- <<: insertion operator
- Create a new line: <<endl is the same as <<"\n"
- Test for "=", add a=a+2, then output a before and after this line.

(Note: We skip output to files)



Move about Output

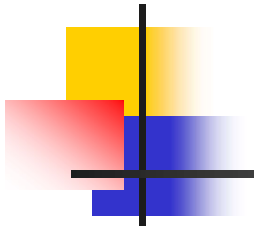
"\": escape sequence

"tells the compiler that the character following the \ does not have the same meaning as that appearing by itself.

Try the following and see what do they display:

```
cout << "I am "the king of C++."" <<endl;

cout << "I am \"the king of C++.\"" <<endl;

cout << "Read the file C:\windows" <<endl;
```

(how to fix it?)



Test Your Understanding

- Display the following on the screen (XXX is your name)

  Hello!

  I am XXX.

  This is my first C++ code.

- Declare a=3, b=4, c=5, display

  a=…, b=…, c=…

  a+b+c=….

- Declare a=30000 (short), b=30000 (int), c=3,000,000,000 (int) , output the following on the screen:

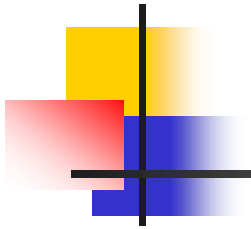
  a=…, b=…, c=…

  a*2=…, b*3=…, c*4=…

- Display the following on the screen

  Please read the file c:\windows\read.txt

  He is called “superman.”



Input

Syntax for cin (from the keyboard):

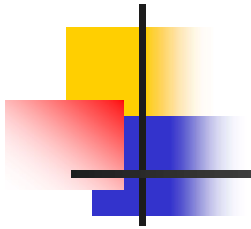
cin >> input1;

cin >> input2 >> input3….;

```
//Input and Output
#include <iostream>
using namespace std;

int main(){
        int a=3, b=4;
        cout << "a=" << a << endl;
        cout << "the number is "
        << a << "\n";
        cout << "input a number\n";
        cin >> c >>d;
        cout << "c=" << c << "  d=" <<d << endl;
        return 0;
}
```

Notes:

- All the variables have to declare before using.
- Similar to cout, using >> instead.
- Need to hit enter after input.
- Can you use space between inputs?
- What is the problem with the code?



Math Operators

Notes:

- +, -, *, / are straightforward.
- %: a%b, the remainder of a/b.
- What does x+y*z do?
- Using ( ) whenever needed.

More assignment operation:

Example	**Equivalent to**
count +=2;	count = count +2;
total -= discount;	total = total – discount;
bonus *=2;	bonus = bonus *2
amount *= cnt1+cnt2	amount = amount * (cnt1+cnt2)

Test Your Understanding

- Ask user to input an integer bb, and display:

  Your input is XXX.

  Your input +2 is YYY.

- Ask user to input a positive integer (in min), then output " your input, XX min, is equivalent to YY hr and XX min."
- Ask user to input an integer "aa", and try to see if following can be **correctly calculated and displayed**

  aa=…

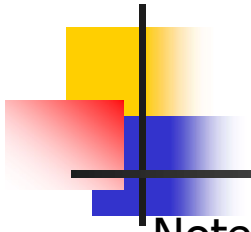
  sin(aa)=

  log(aa)=…

  ln(aa)=…

  aa^4=….

Bug / Error Messages



Note for error messages

- What error messages would you get if you:
  - Remove ";"
  - Typo….
  - Remove {
  - Add one space before "iostream"
- Pay attention to error messages FROM NOW.
- Be patient and confident – One day you will fix the code….

Types of 'bugs' or errors

- Syntax errors: Wrong 'grammar'. Simple to find, and can be found by C++ or Matlab or the Programmer
- Runtime errors: do not necessarily show error message every time and every condition. (May depend on the input)
  - Test a wider range of inputs.
  - Test using a simple version program (that can be checked by hand)
  - ….

- The worst bug does not show error messages, but gives wrong results.

Code Bugs…. Ariane 5 Launch Failure

Source: Youtube

The Ariane 5 reused the inertial reference platform from the Ariane 4, but the Ariane 5's flight path differed considerably from the previous models. The Ariane 5's greater horizontal acceleration caused the computers in both the back-up and primary platforms to crash and emit diagnostic data misinterpreted by the autopilot as spurious position and velocity data.[*citation needed*] Pre-flight tests had never been performed on the inertial platform under simulated Ariane 5 flight conditions so the error was not discovered before launch. During the investigation, a simulated Ariane 5 flight was conducted on another inertial platform. It failed in exactly the same way as the actual flight units.[*citation needed*]

The greater horizontal acceleration caused a data conversion from a 64-bit floating point number to a 16-bit signed integer value to overflow and cause a hardware exception. Efficiency considerations had omitted range checks for this particular variable, though conversions of other variables in the code were protected. The exception halted the reference platforms, resulting in the destruction of the flight.[4]

Source: Wikipedia