# Lecture 1 - C++ Basics

Meng-Hsun Tsai
CSIE, NCKU

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello NCKU!" << endl;
    return 0;
}
```

# First Program in C++: Printing a Line of Text

```
1  #include <iostream>    // for using cout
2  using namespace std;   // avoid repeating "std::"
3
4  int main()
5  {
6      std::cout << "Hello ";
7      cout << "NC" << "KU!!" << endl;
8      return 0;
9  }
```

```
> g++ -o hello hello.cpp
> ./hello
Hello NCKU!!
>
```

# Where is *iostream*?

- In Cygwin

> find /usr -name iostream

/usr/lib/gcc/i686-pc-cygwin/3.4.4/include/c++/iostream

/usr/lib/gcc/i686-pc-mingw32/3.4.4/include/c++/iostream

- In FreeBSD

> find /usr -name iostream

/usr/include/c++/4.2/iostream

# What's Inside *iostream*?

```
> cat /usr/include/c++/4.2/iostream
// Standard iostream objects -*- C++ -*-
…
#include <ostream>
#include <istream>
namespace std
{
    extern istream cin;     ///< Linked to standard input
    extern ostream cout;    ///< Linked to standard output
    extern ostream cerr;    ///< Linked to standard error (unbuffered)
    extern ostream clog;    ///< Linked to standard error (buffered)
}  // namespace std
…
```

# A Simple Example using *#include*

*included_file.h*

```
1   std::cout << "included_file!\n" ;
```

*including_file.cpp*

```
1  #include <iostream>
2  int main()
3  {
4  #include "included_file.h"
5        std::cout << "including_file!\n" ;
6        return 0;
7  }
```

```
> g++ -o including_file including_file.cpp
> ./including_file
included_file!
including_file!
```

# Output of Preprocessor

```
$ g++ -E  including_file.cpp
namespace std
{
# 63 "/usr/lib/gcc/i686-pc-cygwin/3.4.4/include/c++/iostream" 3
  extern istream cin;
  extern ostream cout;
…
# 2 "including_file.cpp" 2
int main()
{
# 1 "included_file.h" 1
 std::cout << "included_file !\n" ;
# 5 "including_file.cpp" 2
 std::cout << "including_file !\n" ;
 return 0;
}
```

From g++'s man page:

-E Stop after the preprocessing stage; do not run the compiler.

# Using #ifdef to Turn on/off Debugging Messages

```
 1  #include <iostream>
 2  #include <cstring>
 3  int main(int argc, char ** argv)
 4  {
 5  #ifdef DEBUG
 6          std::cout << argv[1] << "\n";
 7  #endif
 8          std::cout << strlen(argv[1]) << "\n";
 9          return 0;
10  }
```

```
> g++ -o str_len str_len.cpp
> ./str_len NCKU
4
> g++ -DDEBUG -o str_len str_len.cpp
> ./str_len NCKU
NCKU
4
```

也可在前面加
# define DEBUG來定義DEBUG
# undef  DEBUG則是取消定義

# Preprocessor Wrapper

"Preprocessor wrappers" in header files to prevent the code in the header from being included into the same source code file more than once.

*Sudoku.h*

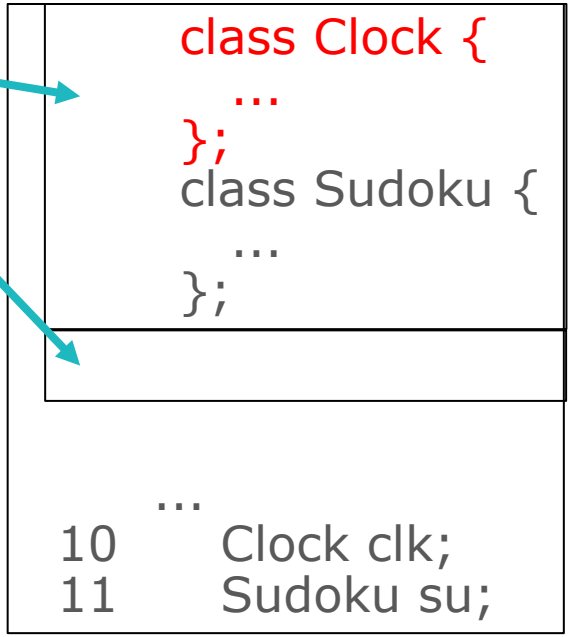
```
1  #ifndef SUDOKU_H
2  #define SUDOKU_H
3  #include "Clock.h"
4  class Sudoku {
     ...
20 };
21 #endif
```

*main.cpp*

```
1  #include "Sudoku.h"
2  #include "Clock.h "
     ...
10     Clock clk;
11     Sudoku su;
     ...
```

*Clock.h*

```
1  #ifndef CLOCK_H
2  #define CLOCK_H
3  class Clock {
     ...
15 };
16 #endif
```



# Preprocessor Wrapper (cont.)

- The clock class definition is enclosed in the following preprocessor wrapper:

```
#ifndef CLOCK_H
#define CLOCK_H
  ...
#endif
```

- This prevents the code between #ifndef and #endif from being included if the name CLOCK_H has been defined.
- If the header has not been included previously in a file, the name CLOCK_H is defined by the #define directive and the header file statements are included.
- If the header has been included previously, CLOCK_H is defined already and the header file is not included again.

# Preprocessor Directive

- A preprocessor directive is a message to the C++ preprocessor.
- Lines that begin with # are processed by the preprocessor before the program is compiled.
- #include <iostream> notifies the preprocessor to include in the program the contents of the input/output stream header file <iostream>.
  - Must be included for any program that outputs data to the screen or inputs data from the keyboard using C++-style stream input/output.

# Comments and *using* Declaration

- // indicates that the remainder of each line is a comment.
  - You insert comments to document your programs and to help other people read and understand them.
  - Comments are ignored by the C++ compiler and do not cause any machine-language object code to be generated.
- You also may use C's style in which a comment—possibly containing many lines—begins with /* and ends with */.
- using declaration eliminates the need to repeat the std:: prefix.

# Getting Return Value in Unix

```
> cat return_minus1.cpp
int main()
{
        return -1;
}
> g++ -o return_minus1 return_minus1.cpp
> echo $?
0
> ./return_minus1
> echo $?
255
> echo $?
0
```

# The *cout* Object

- When a cout statement executes, it sends a stream of characters to the standard output stream object—std::cout—which is normally "connected" to the screen.
- The notation std::cout specifies that we are using a name, in this case cout, that belongs to "namespace" std.
- The << operator is referred to as the stream insertion operator. The value to the operator's right, the right operand, is inserted in the output stream.

# The *endl* Stream Manipulator

- std::endl is a so-called stream manipulator.
- The name endl is an abbreviation for "end line" and belongs to namespace std.
- The std::endl stream manipulator outputs a newline, then "flushes the output buffer."
  - This simply means that, on some systems where outputs accumulate in the machine until there are enough to "make it worthwhile" to display them on the screen, std::endl forces any accumulated outputs to be displayed at that moment.
  - This can be important when the outputs are prompting the user for an action, such as entering data.

# Adding Two Integers

```
 1  #include <iostream>
 2  using namespace std;
 3  int main()
 4  {
 5      int num1, num2;
 6      cout << "Please enter the first number: ";
 7      cin >> num1;
 8      cout << "Please enter the second number: ";
 9      cin >> num2;
10      cout << "Sum of the two numbers are: " << num1 + num2 << endl;
11      return 0;
12  }
```

```
> ./add
Please enter the first number: 3
Please enter the second number: 5
Sum of the two numbers are: 8
```

# The *cin* Object

- A cin statement uses the input stream object cin (of namespace std) and the stream extraction operator, >>, to obtain a value from the keyboard.
- When the computer executes an input statement that places a value in an int variable, it waits for the user to enter a value for variable num1.
- The computer converts the character representation of the number to an integer and assigns this value) to the variable num1.