2023

Program Design II

Kun-Ta Chuang
Department of Computer Science and Information Engineering
National Cheng Kung University

【 助教輔導時間 】

1. 星期一 14:00~17:00 地點: 4203
2. 星期三 18:00~21:00 地點: 4201
3. 星期四 18:00~21:00 地點: 4201

【 注意事項 】

1. 請自備筆電
2. Smile

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!!
>
```

Comments and *using* Declaration

- // indicates that the line is "comment".
  - You insert comments to make your programs readable and to help other people understand your logistics.
  - Comments are ignored by the C++ compiler and do not cause any machine-language object code to be generated.
- You can also 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.

What's Inside *iostream*?

```
>   cat /usr/include/c++/11/iostream                    ->Google 'cat' by yourself
// 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
    …
```

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(void)
 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.

A Simple Example using *#include*

included_file.h

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

including_file.cpp

```
1#include <iostream>
2int 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
{
….
….
# 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>
3int 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
```

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
```

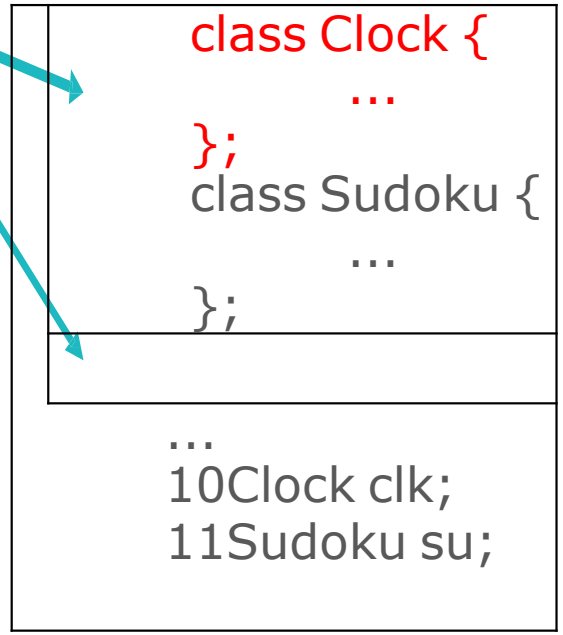
main.cpp

```
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.

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

‘Echo $?’ is commonly used in shell script. Google it.

write_file.cpp

```
1  #include <iostream>
2  #include <string>
3  #include <fstream>
4  #include <cstdlib>
5  using namespace std;
6  int main()
7 {
8 string name;
9 float proj, exam;
10 ofstream outFile("outfile",
ios::out);
11 if(!outFile) {
12 cerr << "Failed opening" << endl;
13 exit(1);
14
15 cout << “Enter NAME PROJ
EXAM each line.\n”  << "EOF to
finish.\n" << "? ";
16
17 outFile <<
"Name\tProj\tExam\tTotal\n";
18 while(cin >> name >> proj >> exam) {
19 outFile << name << "\t" << proj << "\t" <<
exam << "\t"
20      << proj*0.65 + exam*0.35 << endl;
21      cout << "? ";
22    }
23   cout << endl;
24   return 0;
25 }
```

```
> ./write_file
Enter NAME PROJ EXAM each line.
EOF to finish.
? Winner 99 100
? Loser 23 61
? ^D
> cat outfile
Name    Proj      Exam     Total
Winner  99        100      99.35
Loser   23        61       36.3
```

Creating a Sequential File

- Two arguments are passed to an ofstream object's constructor—the filename and the file-open mode (line 10).

```
10   ofstream outFile("outfile", ios::out);
```

- Existing files opened with mode ios::out are truncated—all data in the file is discarded.
- If the specified file does not yet exist, then the ofstream object creates the file, using that filename.

Creating a Sequential File (cont.)

- For an ofstream object, the file-open mode can be either ios::out to output data to a file or ios::app to append data to the end of a file.

Mode	Description
ios::app	*Append all output to the end of the file*
ios::ate	*Open a file for output and move to the end of the file (normally used to append data to a file). Data can be written anywhere in the file.*
ios::in	*Open a file for input.*
ios::out	*Open a file for output.*
ios::trunc	*Discard the file's contents (this also is the default action for ios::out).*
ios::binary	*Open a file for binary (i.e., nontext) input or output.*

Creating a Sequential File (cont.)

- An ofstream "object" can be created without opening a specific file—a file can be attached to the object later.
- For example, the statement
  - ofstream outFile;
- creates an ofstream object named outFile.
- The ofstream member function open opens a file and attaches it to an existing ofstream object as follows:
  - outFile.open("outfile", ios::out);

Creating a Sequential File (cont.)

- The if statement in lines 11–14 uses the overloaded ios member function operator! to determine whether the open operation succeeded.
- Some possible errors are
  - attempting to open a nonexistent file for reading
  - attempting to open a file for reading or writing without permission
  - opening a file for writing when no disk space is available

```
11   if(!outFile) {
12   cerr << "Failed opening" << endl;
13   exit(1);
14   }
```

Creating a Sequential File (cont.)

- When end-of-file is encountered or bad data is entered, the while statement terminates.
- Ctrl-D in Unix and Ctrl-Z in Windows represent end-of-file.

```
18    while(cin >> name >> proj >> exam) {
19        outFile << name << "\t" << proj << "\t" << exam << "\t"
20            << proj*0.65 + exam*0.35 << endl;
21        cout << "? ";
22    }
```

```
?
^D
```

Creating a Sequential File (cont.)

- Line 19 writes a set of data to the file outfile, using the stream insertion operator << and the outFile object associated with the file at the beginning of the program.

```
19      outFile << name << "\t" << proj << "\t" << exam << "\t"
20         << proj*0.65 + exam*0.35 << endl;
```

- Once the user enters the end-of-file indicator, main terminates.
- This implicitly invokes outFile's destructor, which closes the outfile file.
- You also can close the ofstream object explicitly, using member function close in the statement

Reading Data from a Sequential File

- Creating an ifstream object opens a file for input.
- The ifstream constructor can receive the filename and the file open mode as arguments.
- Create an ifstream object (called inFile) and associate it with the infile file.

```
ifstream inFile("infile", ios::in);
```

- Objects of class ifstream are opened for input by default.
- We could have used the statement

```
ifstream inFile("infile");
```

to open infile for input.

Reading Data from a Sequential File (cont.)

- Just as with an ofstream object, an ifstream object can be created without opening a specific file, because a file can be attached to it later.
- Each time line 17 executes, it reads another record from the file into the variables name, proj , exam and total.
- When the end of file has been reached, the ifstream destructor function closes the file.

```
while(inFile >> name >> proj >> exam >> total) {
  cout << name << "\t" << proj << "\t"
  << exam << "\t" << total << endl;
}
```

Questions?


consigliere
@moyodre
大家會怎麼在自己的履歷上寫「我換了一個燈泡」？
M
@MuyiwaSaka
在沒有造成任何成本超支以及安全事故的情況下，獨力成功管理了環境照明系統的升級與安裝。