Lab2

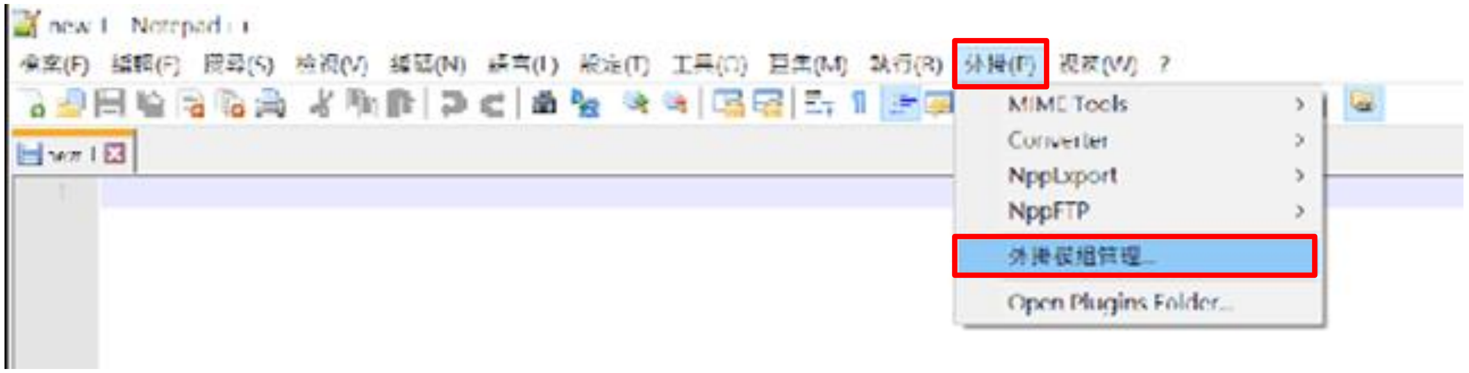
Stream & File I/O

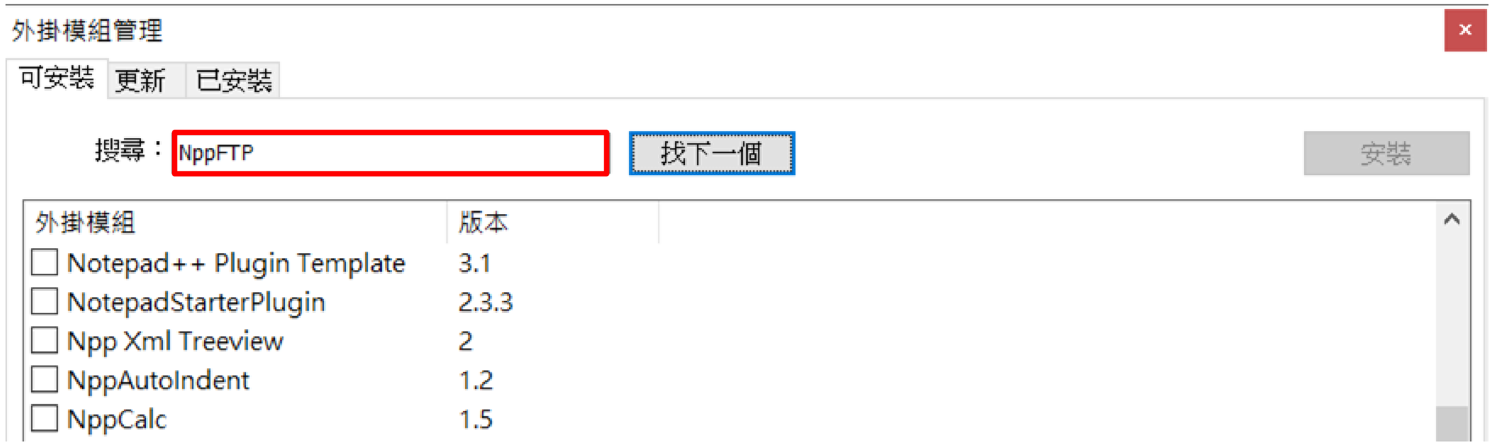
Notepad++

- https://notepad-plus-plus.org/downloads/
- Choose the edition you desired
- Plugins→Plugin Manager → NppFTP → Download
- NppFTP → Show NppFTP Window
- Settings → Profile Settings
- Hostname: 140.113.212.154
- Connection type: SFTP

Notepad++ (Cont.)

- 外掛 → 外掛模組管理 → 搜尋NppFTP並下載



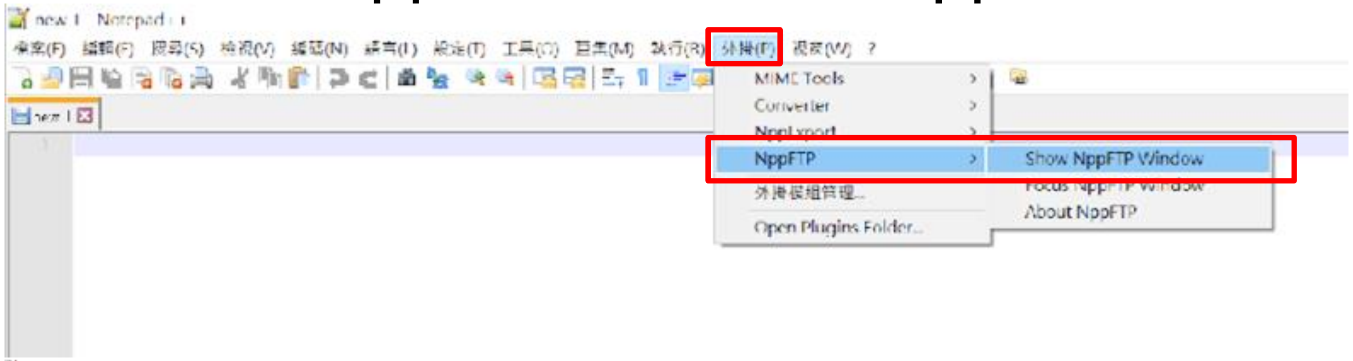


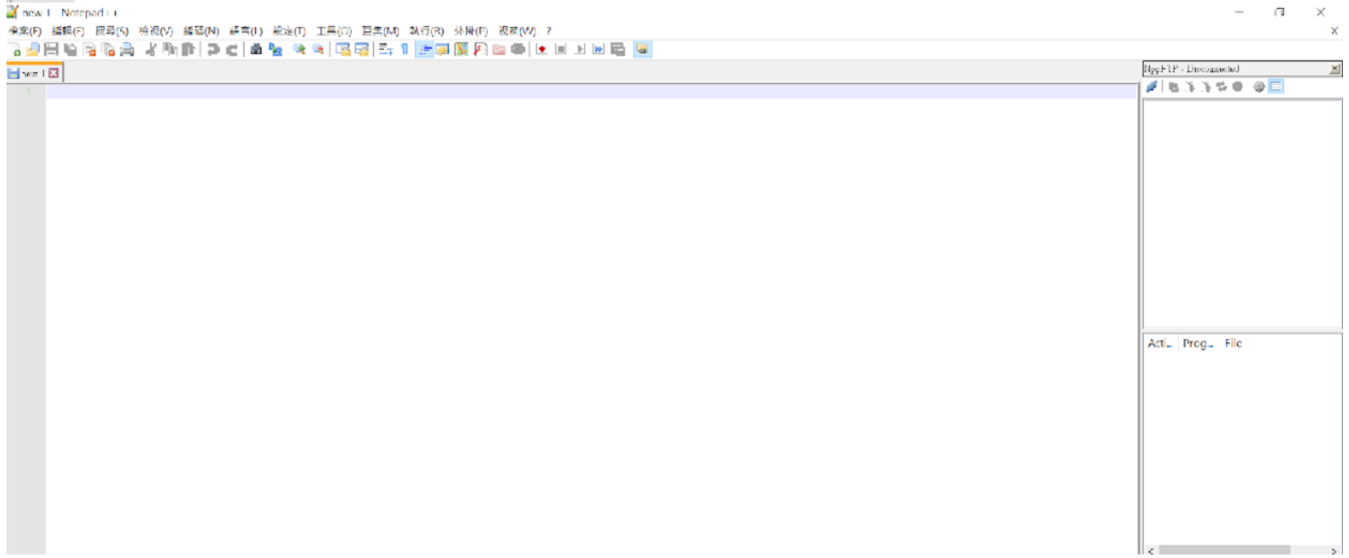
搜尋：NppFTP

外掛模組	版本
☐ Notepad++ Plugin Template	3.1
☐ NotepadStarterPlugin	2.3.3
☐ Npp Xml Treeview	2
☐ NppAutoIndent	1.2
☐ NppCalc	1.5

Notepad++ (Cont.)

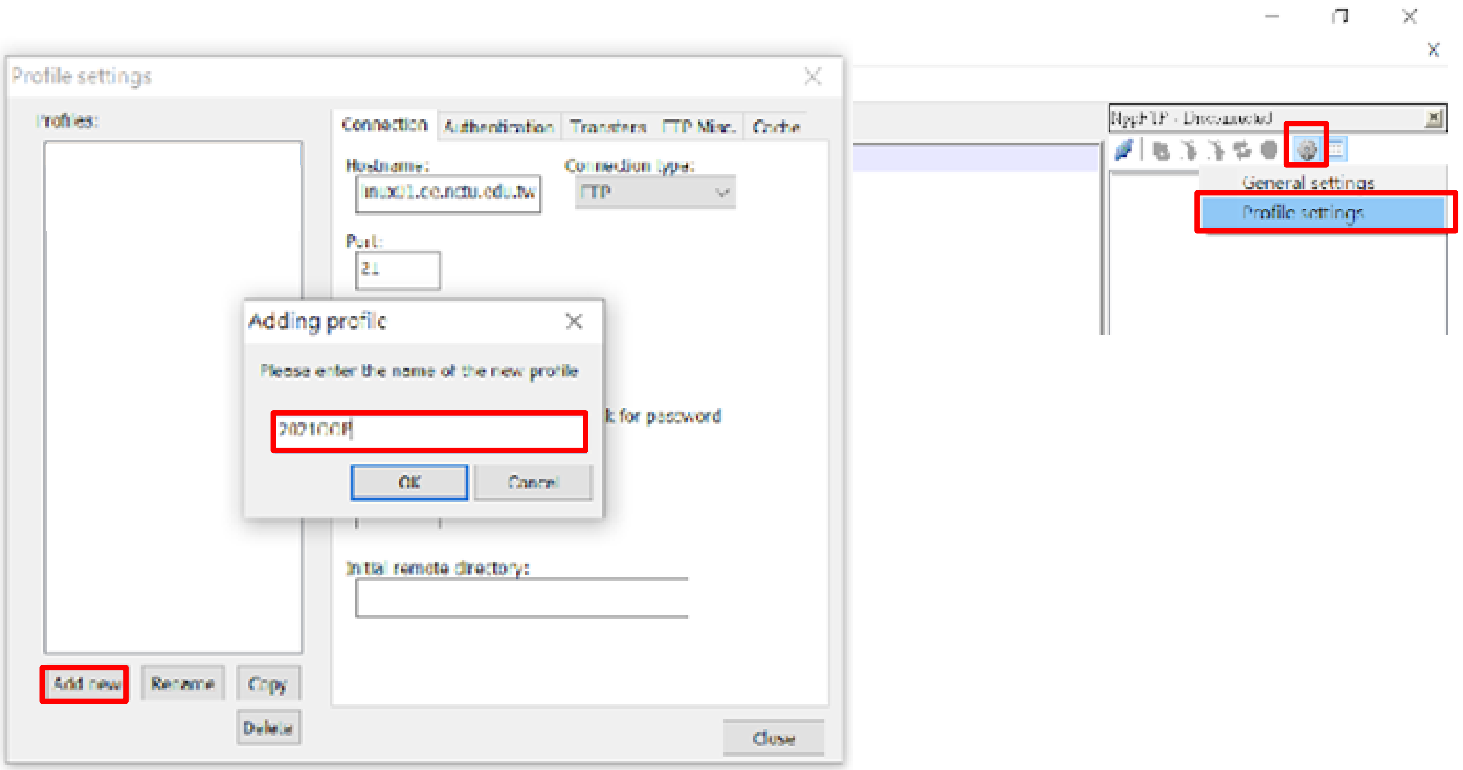
- 外掛 → NppFTP → Show NppFTP Window





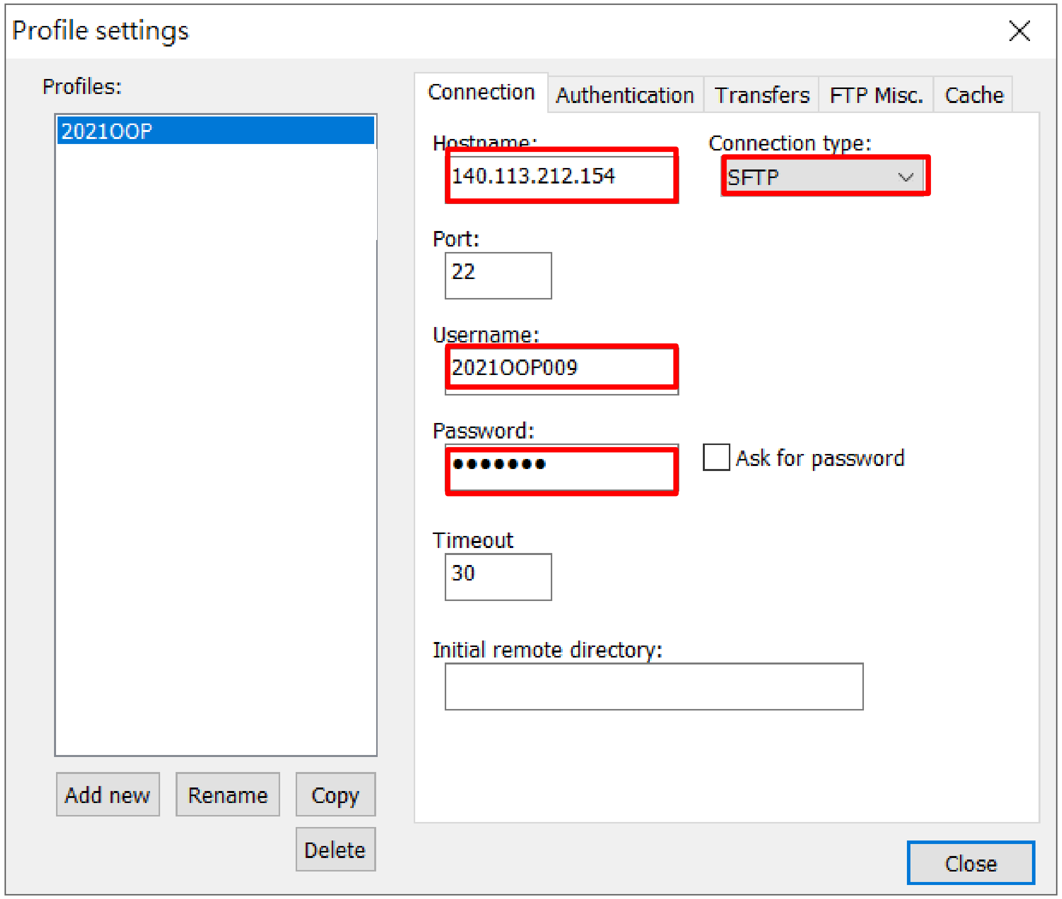
Notepad++ (Cont.)

- Settings → Profile Settings



Hostname: linux01.cc.nctu.edu.tw

Notepad++ (Cont.)



Profile settings

Profiles:

2021OOP

Connection | Authentication | Transfers | FTP Misc. | Cache

Hostname: 140.113.212.154

Connection type: SFTP

Port: 22

Username: 2021OOP009

Password: ●●●●●●●

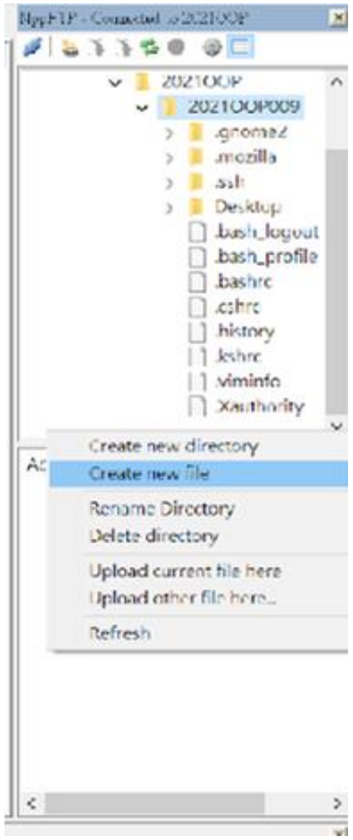
[ ] Ask for password

Timeout 30

Initial remote directory:

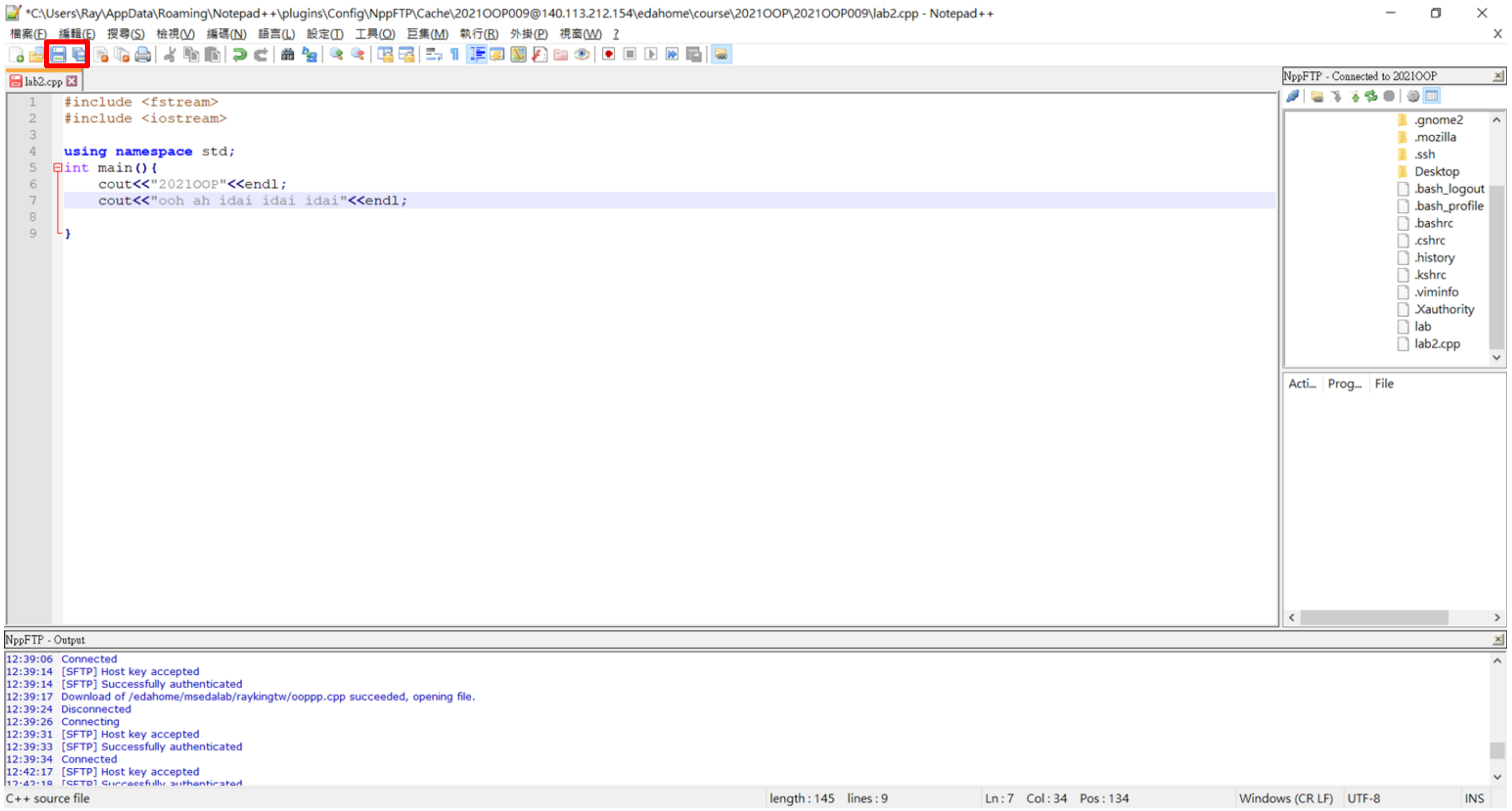
Notepad++ (Cont.)

- 在工作站上開啟cpp檔案，在右邊空白處點右鍵Creat new file



Coding

- Coding → Ctrl+s存檔(or左上角存檔)



*C:\Users\Ray\AppData\Roaming\Notepad++\plugins\Config\NppFTP\Cache\2021OOP009@140.113.212.154\edahome\course\2021OOP\2021OOP009\lab2.cpp - Notepad++

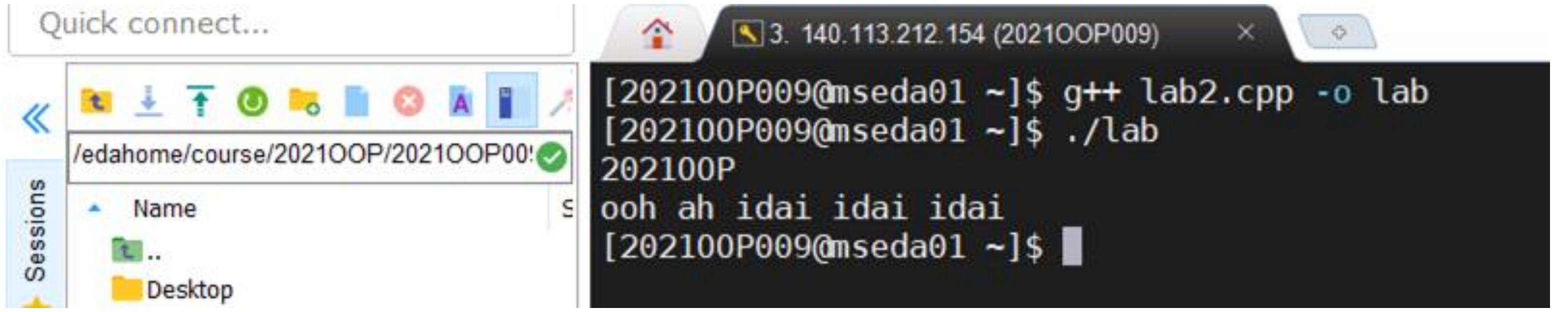
```cpp
#include <fstream>
#include <iostream>

using namespace std;
int main(){
    cout<<"2021OOP"<<endl;
    cout<<"ooh ah idai idai idai"<<endl;

}
```

```
12:39:06  Connected
12:39:14  [SFTP] Host key accepted
12:39:14  [SFTP] Successfully authenticated
12:39:17  Download of /edahome/msedalab/raykingtw/ooppp.cpp succeeded, opening file.
12:39:24  Disconnected
12:39:26  Connecting
12:39:31  [SFTP] Host key accepted
12:39:33  [SFTP] Successfully authenticated
12:39:34  Connected
12:42:17  [SFTP] Host key accepted
12:42:18  [SFTP] Successfully authenticated
```

C++ source file   length : 145   lines : 9   Ln : 7   Col : 34   Pos : 134   Windows (CR LF)   UTF-8   INS

Compile & Run

- Compile:
  - g++ <.cpp檔> -o <執行檔名>
  - EX: g++ lab2.cpp -o lab
- Run:
  - ./<執行檔名>
  - EX: ./lab



Quick connect...
/edahome/course/2021OOP/2021OOP00
Name
Desktop
Sessions
3. 140.113.212.154 (2021OOP009)

```
[2021OOP009@mseda01 ~]$ g++ lab2.cpp -o lab
[2021OOP009@mseda01 ~]$ ./lab
2021OOP
ooh ah idai idai idai
[2021OOP009@mseda01 ~]$
```

File I/O

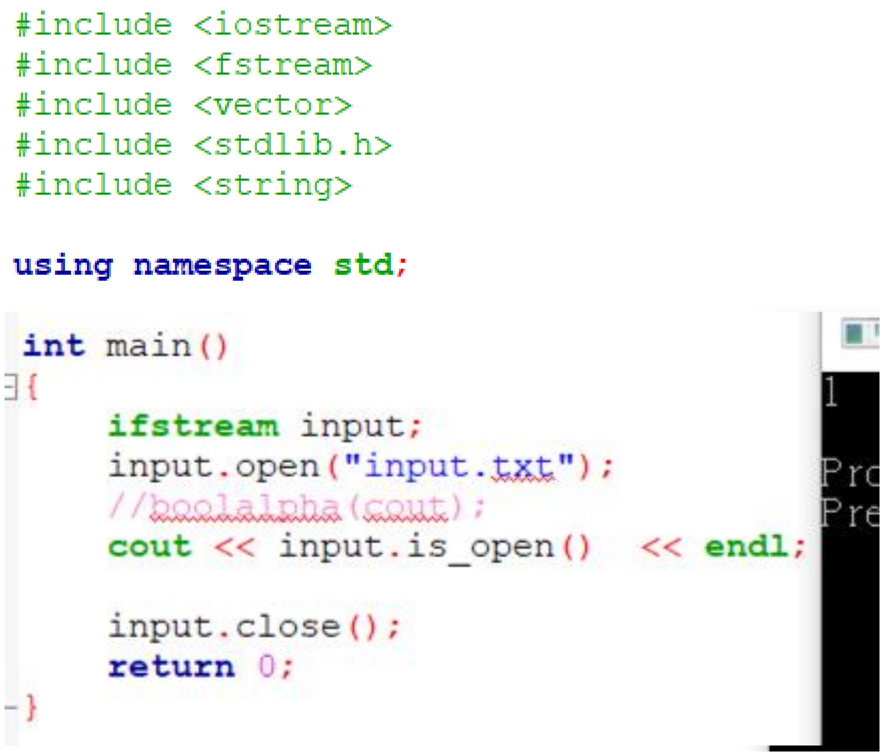
- Declare a file name variable

```
#include <fstream>
ifstream input_filename_var ;       // input file
ofstream output_filename_var ;  // output file
```

- Open the file

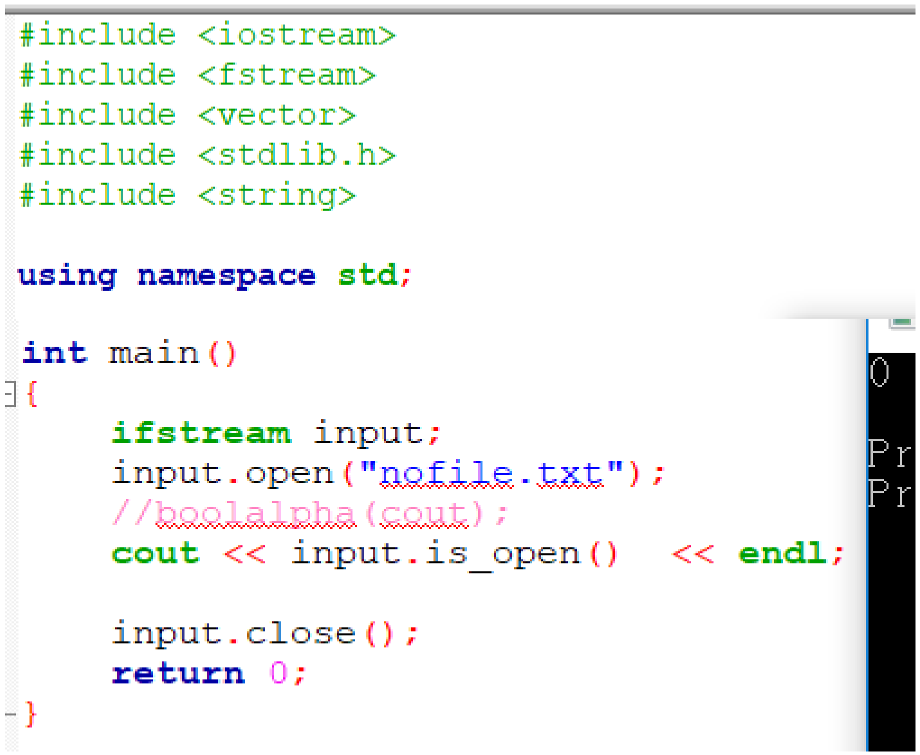
```
input_filename_var.open( “…/file_path/filename” );
output_filename_var.open( “…/file_path/filename” );
```

Check file opened successfully



```
#include <iostream>
#include <fstream>
#include <vector>
#include <stdlib.h>
#include <string>

using namespace std;

int main()
{
    ifstream input;
    input.open("input.txt");
    //boolalpha(cout);
    cout << input.is_open()  << endl;

    input.close();
    return 0;
}
```

```
1
Pro
Pre
```



```
#include <iostream>
#include <fstream>
#include <vector>
#include <stdlib.h>
#include <string>

using namespace std;

int main()
{
    ifstream input;
    input.open("nofile.txt");
    //boolalpha(cout);
    cout << input.is_open()  << endl;

    input.close();
    return 0;
}
```

```
0

Pr
Pr
```

Read file

- Input file syntax just like input stream “cin”.
  - ? Example:

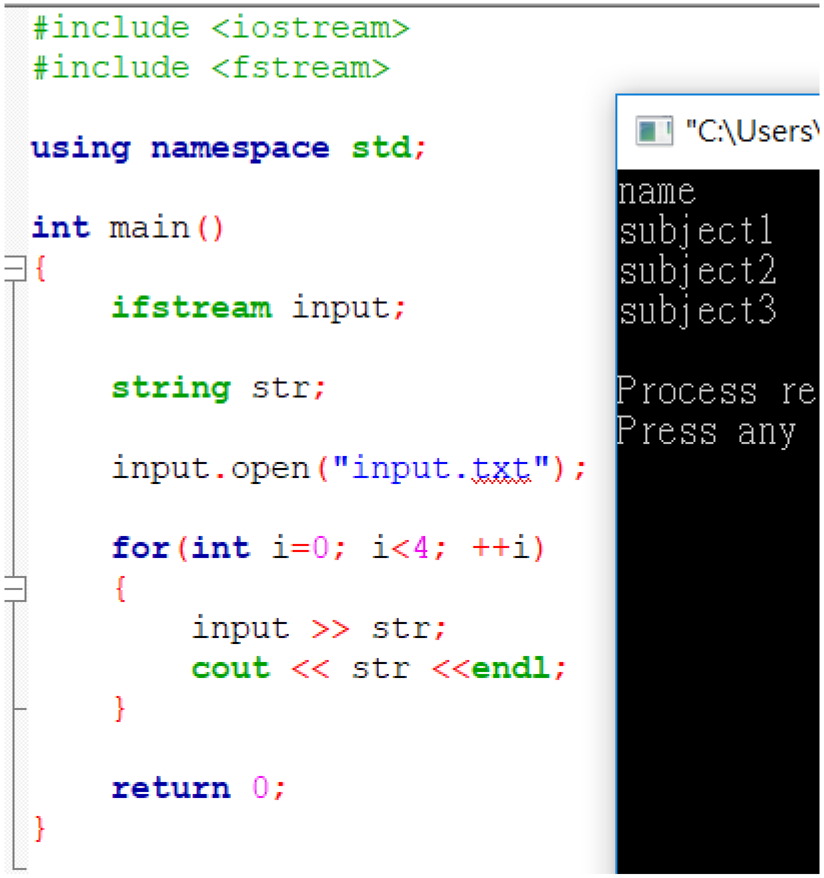
1. input_filename_var >> x >> y;   // x and y are integers
2. input_filename_var >> ch;   // ch is a char
3. ch = input_filename_var.get();   // ch is a char
4. input_filename_var.getline(ch, ch_num); //ch is char*, ch_num is streamsize
5. getline(input_filename_var, str_var); // str_var is string

Output file

- Treat ***ofstream*** (output file stream) just as ***cout.***
  - Example:

1. *output_filename_var* << x << y;   // x and y are integers
2. *output_filename_var* << ch;   // ch is a char
3. *output_filename_var* << “Hello World!” <<‘\n’;   // literal string
4. *output_filename_var* << str;   // str is a char* or string

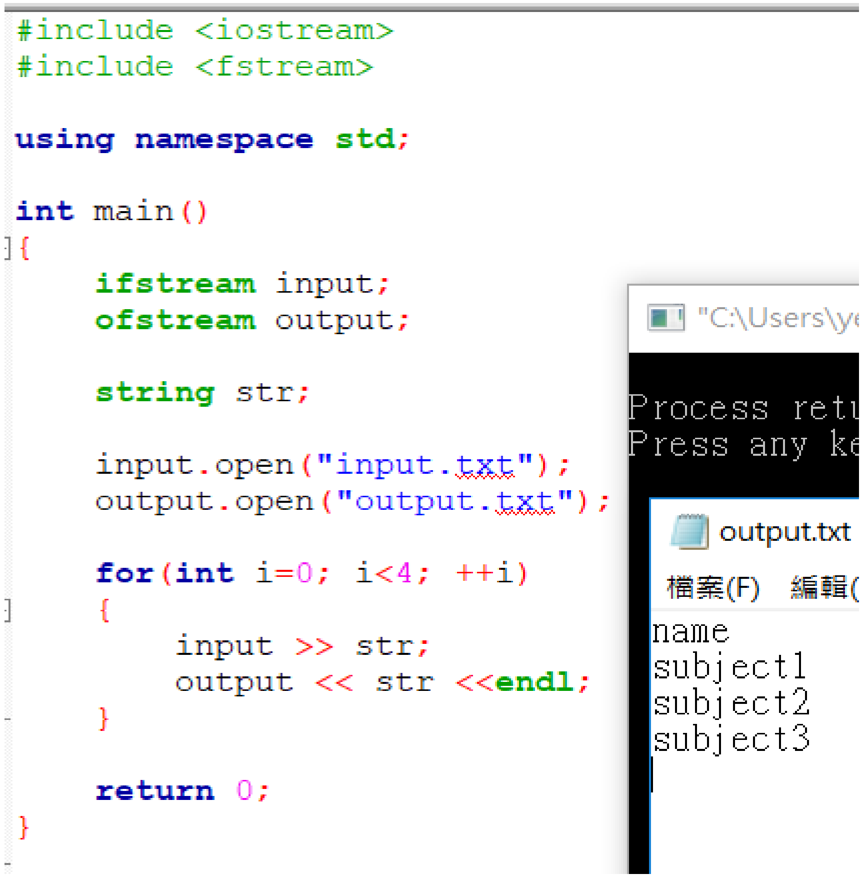
Output file (Cont.)



```
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    ifstream input;

    string str;

    input.open("input.txt");

    for(int i=0; i<4; ++i)
    {
        input >> str;
        cout << str <<endl;
    }

    return 0;
}
```

```
"C:\Users\
name
subject1
subject2
subject3

Process re
Press any
```

Using ***cout***



```
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    ifstream input;
    ofstream output;

    string str;

    input.open("input.txt");
    output.open("output.txt");

    for(int i=0; i<4; ++i)
    {
        input >> str;
        output << str <<endl;
    }

    return 0;
}
```

```
"C:\Users\y

Process retu
Press any ke
```

```
name
subject1
subject2
subject3
```

Using ***ofstream***

Close file

- All opened files will be closed automatically after the execution of the program.
- Always close the open files explicitly if they are no longer being used.

    input_filename_var.**close()**;

    output_filename_var.**close()**;

End of file (EOF)

string a;

1. while(input >> a){

   …

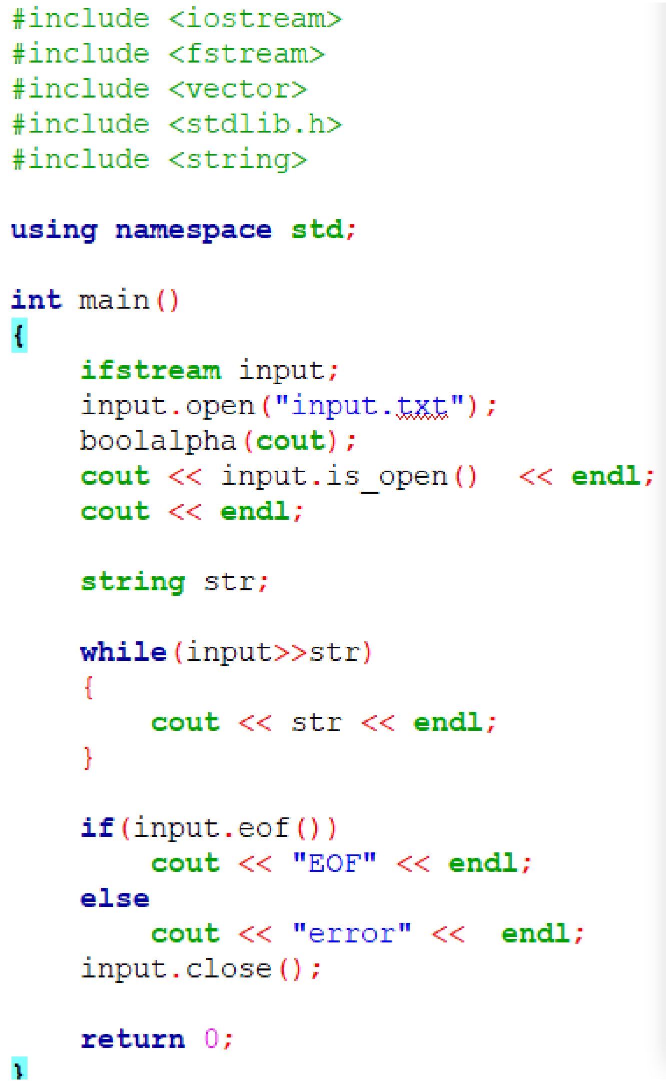
   }

2. while(!input.eof()){

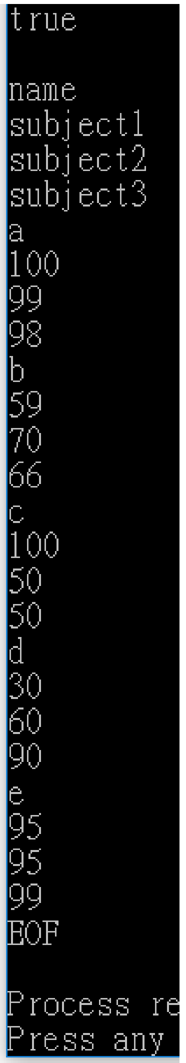
   input >> a;

   …

   }



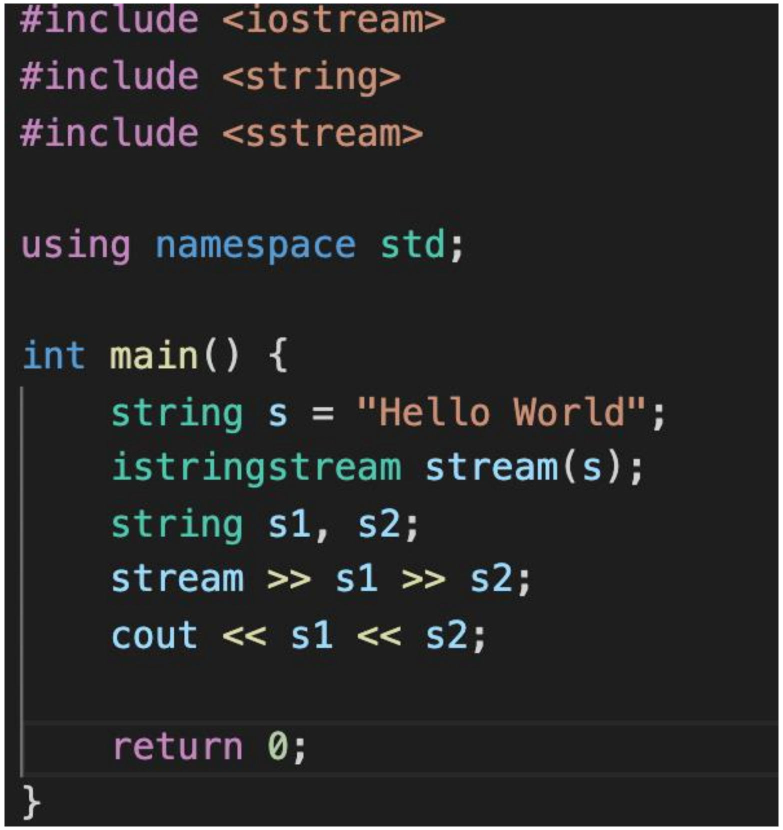
```
#include <iostream>
#include <fstream>
#include <vector>
#include <stdlib.h>
#include <string>

using namespace std;

int main()
{
    ifstream input;
    input.open("input.txt");
    boolalpha(cout);
    cout << input.is_open()  << endl;
    cout << endl;

    string str;

    while(input>>str)
    {
        cout << str << endl;
    }

    if(input.eof())
        cout << "EOF" << endl;
    else
        cout << "error" <<  endl;
    input.close();

    return 0;
}
```



```
true

name
subject1
subject2
subject3
a
100
99
98
b
59
70
66
c
100
50
50
d
30
60
90
e
95
95
99
EOF

Process re
Press any
```

From string to stream

- #include <sstream>
- istringstream stream_name(string)

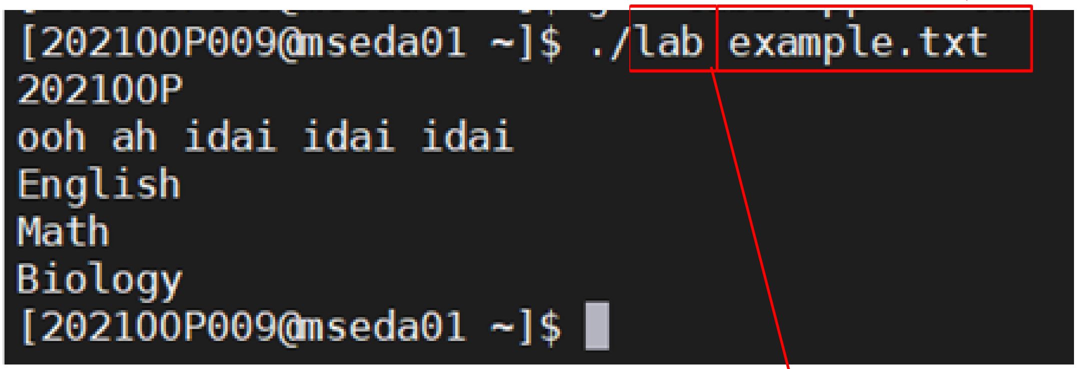


```
#include <iostream>
#include <string>
#include <sstream>

using namespace std;

int main() {
    string s = "Hello World";
    istringstream stream(s);
    string s1, s2;
    stream >> s1 >> s2;
    cout << s1 << s2;

    return 0;
}
```

Information from terminal

- int main(int argc, char **argv)
- int main(int argc, char* argv[])
- ifstream input
- input.open(argv[1], ios::in)

Other arguments start from argv[1]

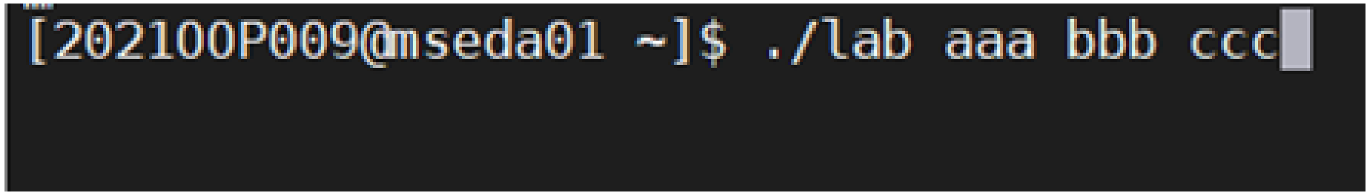


```
[202100P009@mseda01 ~]$ ./lab example.txt
202100P
ooh ah idai idai idai
English
Math
Biology
[202100P009@mseda01 ~]$ 
```

argv[0] is the name of your execution file

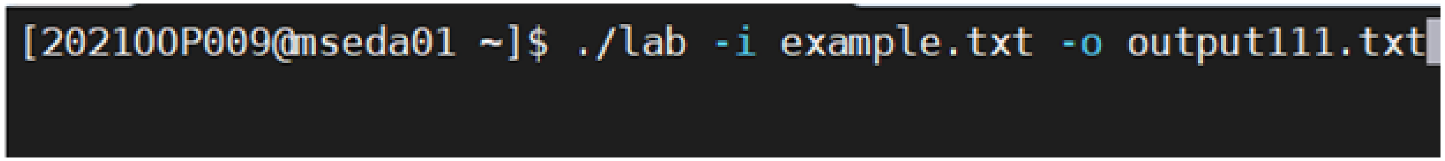
Information from terminal (Cont.)

- EX: aaa → argv[1], bbb → argv[2], ccc → argv[3]



```
[202100P009@mseda01 ~]$ ./lab aaa bbb ccc
```

- EX: -i → argv[1], example.txt → argv[2], …



```
[202100P009@mseda01 ~]$ ./lab -i example.txt -o output111.txt
```

Exercise – Interesting blocks

- Download files “sample_input.txt” from newE3
- Then you receive lots of special blocks, some of which can even be squeezed as you please… 😳
- Read input file by using ***argv[]***

Exercise – Interesting blocks (Cont.)

Input explanation:

Input format	sample_input.txt
<$softBlockRatio_{lowerBound}$> <$softBlockRatio_{upperBound}$> *<blockName> <blockWidth> <blockHeight>* *// It's a hard block* … *<blockName> <blockWidth> <blockHeight>* S *// It's a soft block* …	0.5 4.0 b3 10 20 b1 25 50 S b5 50 80 b2 40 10 S b4 10 10

- First line contains 2 ***double***, they indicates the lower bound and upper bound of soft blocks.
- The line without an ***S*** append at the end implies that it's a ***hard block***.
- The line with an ***S*** append at the end implies that it's a ***soft block***.
- The ID of the blocks always start from 1, and its form is bn, where $n \leq 9$

Exercise – Interesting blocks (Cont.)

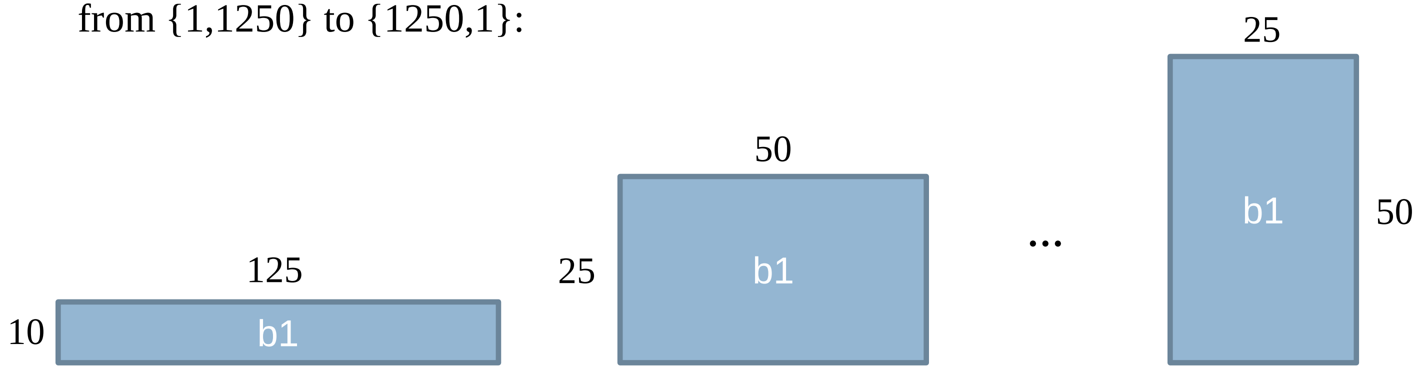
Output explanation:

- Output format of an ***soft block***:

Ex: If the description of soft block is b1 10 125 S, that means the area of the block is 10 × 125.

Such block may have multiple possible ***integer*** width-height pairs range

from {1,1250} to {1250,1}:



Some possible width-height pairs of soft block

However, there's a bounding ratio range from 0.5 to 4.0; therefore, only {25,50} and {50,25} are available.

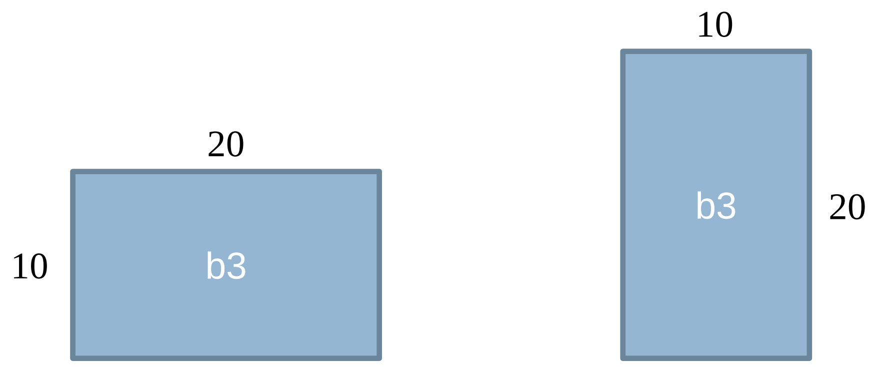
Finally, the output result is b1 {25,50} {50,25}.

Exercise – Interesting blocks (Cont.)

Output explanation:

- Output format of an ***hard block***:

Ex: If the line of the hard block is b3 10 20.



All possible width-height pairs of hard block

Directly output the only 2 width-height pairs: b1 {10,20} {20,10}.

Exercise – Interesting blocks (Cont.)

Output explanation:

- The output width-height pairs should be sort in ***increasing width (decreasing height)***.
- ***Don't*** output the same width-height pairs.

  Ex1: {30,40} and {40,30} are different width-height pairs.

  Ex2: {20,20} and {20,20} are same width-height pairs.
- The restriction of ***softBlockRatio*** is

$$softBlockRatio_{lowerBound} \leq \frac{width}{height} \leq softBlockRatio_{upperBound}$$

- You only need to find ***integer*** width-height pairs!
- Please output the blocks in ***increasing*** order.

Exercise – Interesting blocks (Cont.)

Sample output:

```
b1 {25,50} {50,25}
b2 {16,25} {20,20} {25,16} {40,10}
b3 {10,20} {20,10}
b4 {10,10}
b5 {50,80} {80,50}
```

You can check ***sample_output.txt*** to confirm the output format.

Exercise – Interesting blocks (Cont.)

- Create a directory “OOP112” (mkdir OOP112)
- Change your working directory to “OOP112” (cd OOP112)
- Create a cpp file “Lab-02.cpp” (touch Lab-02.cpp)
- Write your code in Lab-02.cpp
- Use following command to demo:

```
/home/share/demo_OOP112 Lab 02
```

Submission

- Ask TAs for demo
- Try your best to debug your code by yourself
- Upload all your cpp to new E3
- Naming rule : studentID_lab2.cpp