Classes (Part 2)
Separate Files

Introduction

Different applications might want to use your Height class object. For example, a medical records program might want to store the height of a patient; an architectural program might want to store the height of a building; and a football team might want to record the heights of its players. In order to do that, we need to separate the Height class object declaration and definition from the main program so that it can be reused by different programs.

Recall that in the previous Classes document, we have put the entire code for the Height class declaration, definition and the main program in the same main.cpp file. It is listed below for convenience.

```
#include <iostream>
using namespace std;

////////////////////////////////////////////////////////////
// Class declaration

class Height {
private:
   int feet;
   int inches;
public:
   void InputHeight();
   void OutputHeight();
   bool EqualHeight(Height ht);
};

////////////////////////////////////////////////////////////
// Class definition

void Height::InputHeight() {
   cout << "Enter feet and inches? ";
   cin >> feet >> inches;
}

void Height::OutputHeight() {
   cout << feet << " feet " << inches << " inches" << endl;
}

bool Height::EqualHeight(Height ht) {
   if (feet == ht.feet && inches == ht.inches) {
      return true;
   }
   else {
      return false;
   }
}
```

```
////////////////////////////////////////////////////////////
// main

int main() {
   Height myHeight, yourHeight;

   cout << "My height " << endl;
   myHeight.InputHeight();
   myHeight.OutputHeight();

   cout << "Your height " << endl;
   yourHeight.InputHeight();
   yourHeight.OutputHeight();

   if (myHeight.EqualHeight(yourHeight)) {
      cout << "We have the same height!" << endl;
   } else {
      cout << "We have different heights" << endl;
   }
}
```

1. Separate Files

To separate the class declaration and definition from the main program, we need to create two additional files, one for the class declaration and one for the class definition. The main program remains in the original main.cpp file. The file for the class declaration will have for its name the same as the class name, and with the file extension .h which stands for header. This is also called a header file. The file for the class definition will have for its name the same as the class name, but with the file extension .cpp which stands for C++ (but since we cannot use the + symbol in the name therefore we use p for plus. So we create the Height.h file for the Height declaration code, and the Height.cpp file for the Height definition code.

Here are the three separate files.

Height.h file

```
#ifndef  __HEIGHT__
#define __HEIGHT__

////////////////////////////////////////////////////////////
// Class declaration

class Height {
private:
   int feet;
   int inches;
public:
   void InputHeight();
   void OutputHeight();
   bool EqualHeight(Height ht);
};

#endif
```


Noticed that there are three extra lines with a # sign; they are called compiler directives. These three compiler directives are commands to tell the compiler whether or not to include and use the code in between the `#ifndef __HEIGHT__` and `#endif`.

```
#ifndef  __HEIGHT__
#define __HEIGHT__

...

#endif
```

This is needed in order to prevent re-declaring the same class object in case your main code has two or more `#include "Height.h"` lines.

Height.cpp file

```
#include <iostream>
using namespace std;
#include "Height.h"

////////////////////////////////////////////////////////////
// Class definition

void Height::InputHeight() {
   cout << "Enter feet and inches? ";
   cin >> feet >> inches;
}

void Height::OutputHeight() {
   cout << feet << " feet " << inches << " inches" << endl;
}

bool Height::EqualHeight(Height ht) {
   if (feet == ht.feet && inches == ht.inches) {
      return true;
   } else {
      return false;
   }
}
```

The Height class *definition* in Height.cpp needs to know the Height class *declaration* information in the Height.h file, therefore, at the beginning it needs to have the line

```
#include "Height.h"
```

to tell the compiler to include the contents of the Height.h file.

main.cpp file

```
#include <iostream>
using namespace std;
#include "Height.h"

////////////////////////////////////////////////////////
// main

int main() {
   Height myHeight, yourHeight;

   cout << "My height " << endl;
   myHeight.InputHeight();
   myHeight.OutputHeight();

   cout << "Your height " << endl;
   yourHeight.InputHeight();
   yourHeight.OutputHeight();

   if (myHeight.EqualHeight(yourHeight)) {
      cout << "We have the same height!" << endl;
   } else {
      cout << "We have different heights." << endl;
   }
}
```

The main.cpp file is now your main application program. Since it uses the Height data type, therefore, it needs to include the Height.h declaration file.

```
#include "Height.h"
```

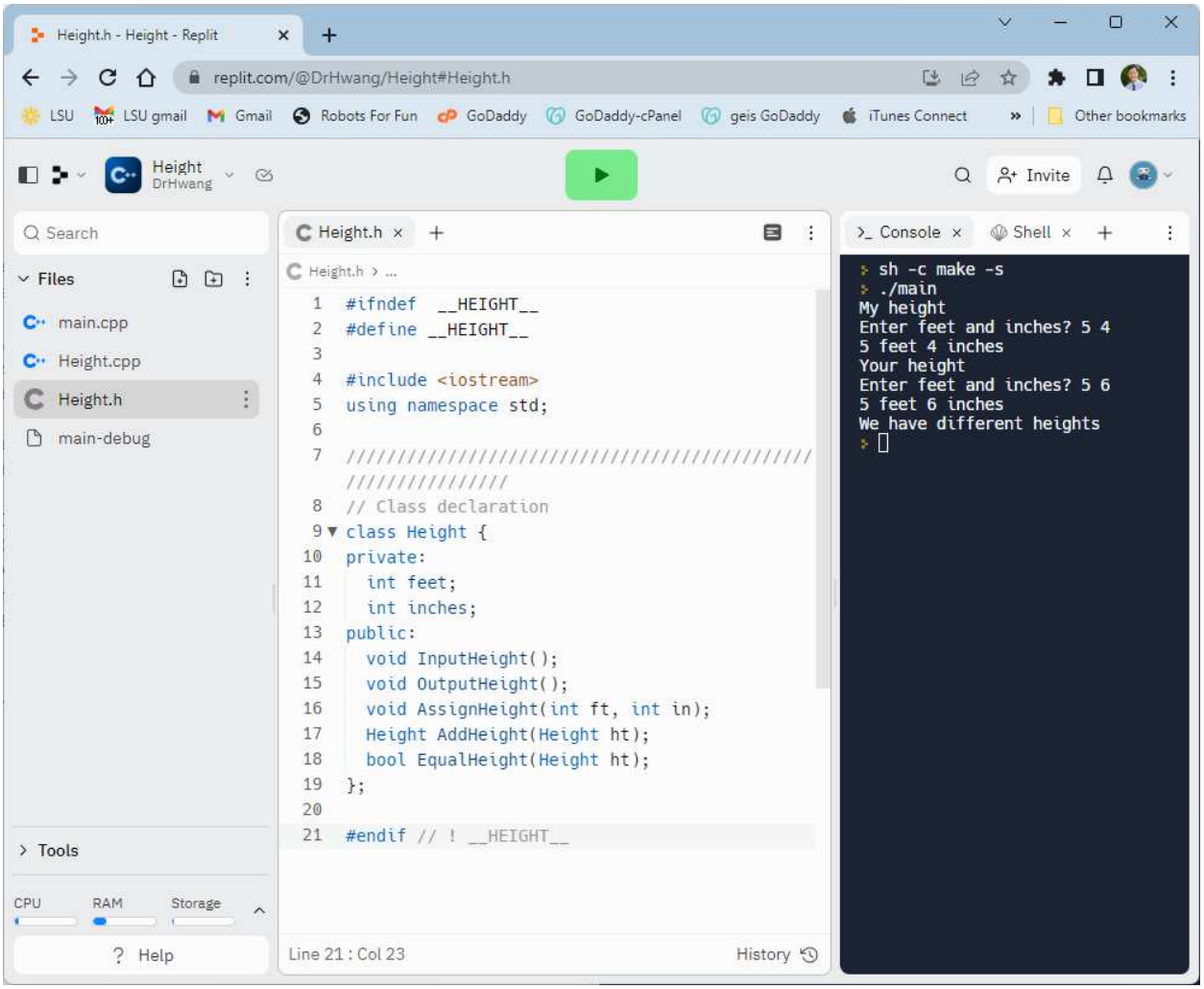
Here we use the double quote instead of the angle bracket as in when you include the iostream library

```
#include <iostream>
```

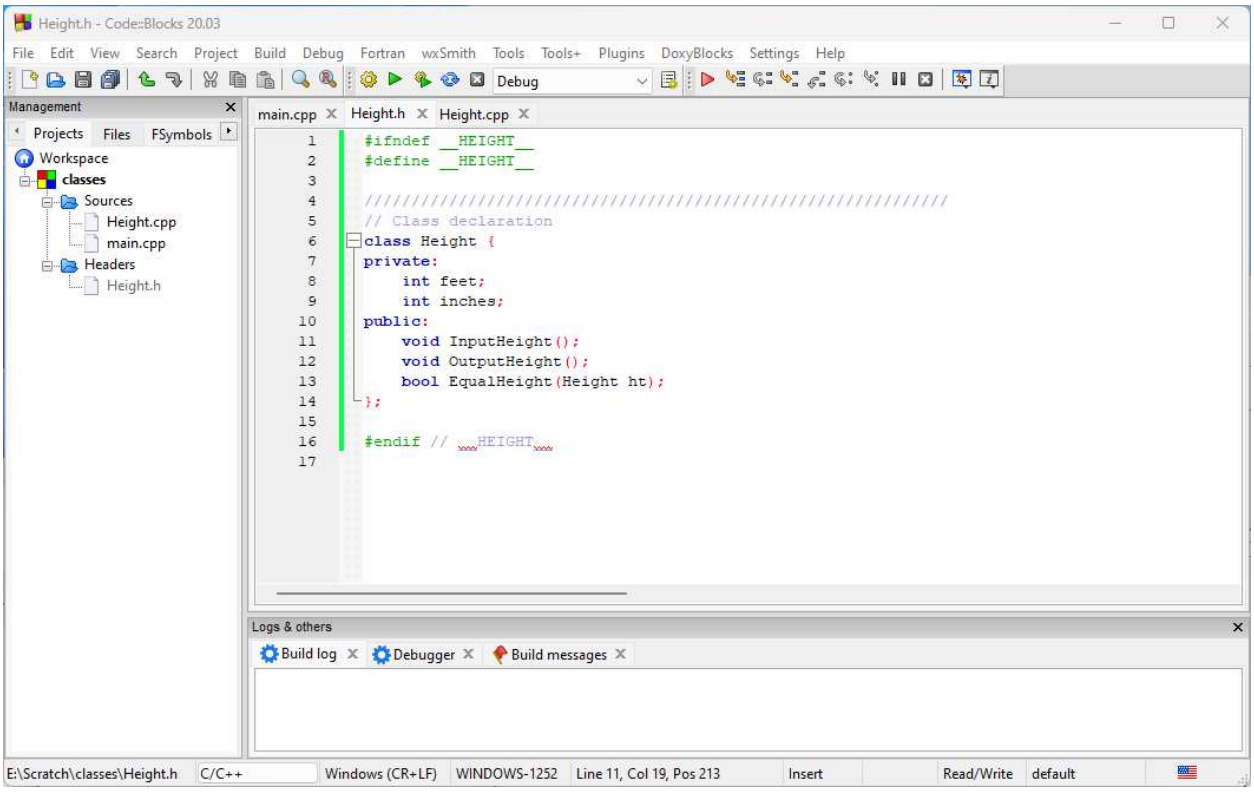
This is to distinguish the location of these files. The angle bracket means the file is in the system library, whereas, the double quote is a file in your local project folder.

2. Screen shots

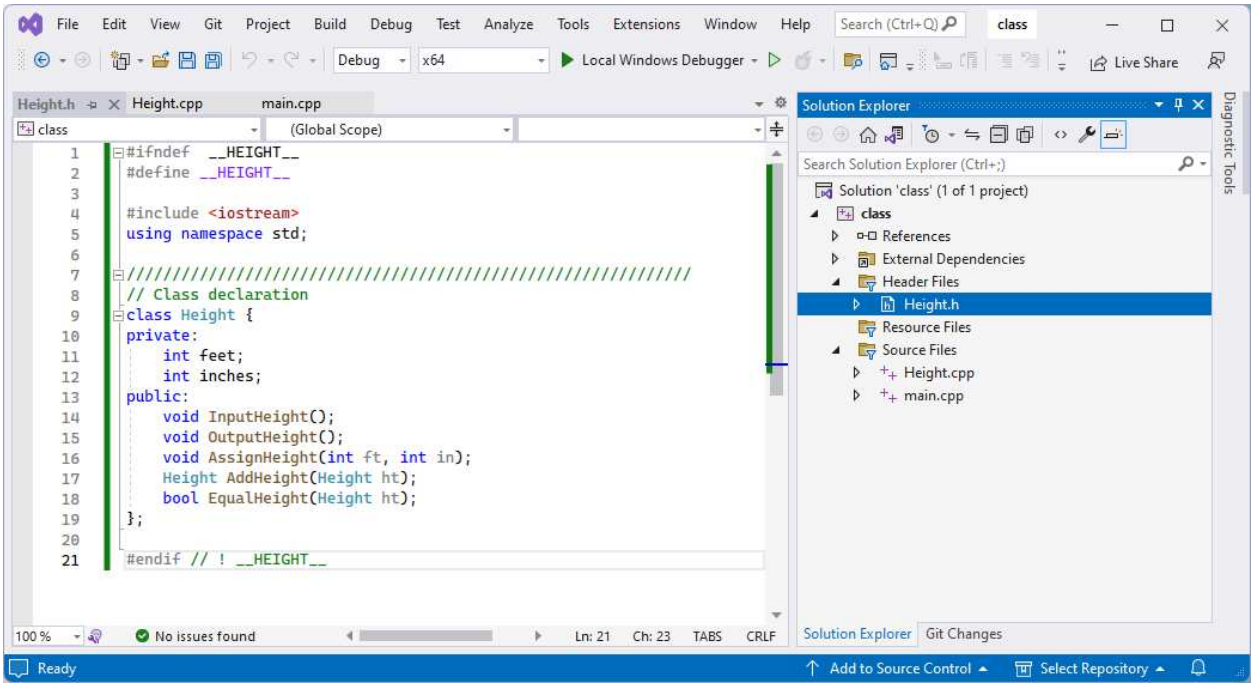
Below is a sample screenshot for the project in replit.com.



Below is a sample screenshot for the project in Code::Blocks.



Below is a sample screenshot for the project in Visual Studio.



3. Exercises (Problems with an asterisk are more difficult)

1. For the Circle class program that you created earlier, separate it out into the three different files.

2. For the Temperature class program that you created earlier, separate it out into the three different files.

3. Create a Date class with the following members:
   - Data members for storing the year, month, day and day of the week.
   - Create several appropriate member functions.

