Lab: Multi-File Projects in Code::Blocks

May 25, 2013

Contents

In an earlier lab, we looked at using Code::Blocks to create simple projects with only a single .cpp file. In the recent lectures, you have learned about the importance of dividing programs into modules. In this lab, we will look at the support Code::Blocks provides for such projects and at some of the common things that can go wrong.

1 Projects with Multiple Files

Run Code::Blocks and create a new project. Edit `main.cpp` to invoke a "greeting" function:

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

#include "greet.h"

using namespace std;

int main()
{
    cout << greet() << endl;
    return 0;
}
```

You can try compiling this if you like. It will fail because we don't have a file named "greet.h" yet. So let's create one. From the "File" menu, select New->File..., choose "C/C++ header", use the "..." button to name your new file "greet.h", click the "All" button, and then "finish". You now have a basic skeleton for a header file. Change it to look like this:

```
#ifndef GREET_H_INCLUDED
#define GREET_H_INCLUDED

#include <string>

std::string greet();

#endif // GREET_H_INCLUDED
```


and save the file. If you open up the "Headers" folder on the left, you can see that your project now has two files.

Again, you can try compiling with the Build button. The build will fail in the link stage because we have not provided a body for the `greet()` function.

From the "File" menu, select New->File..., choose "C/C++ source", select "C++" for the language, use the "..." button to name your new file "greet.cpp", click the "All" button, and then "finish". You now have an empty greet.cpp file. Add a few lines of code:

```
#include "greet.h"

using namespace std;

string greet()
{
  string result = "Hello ";
  result = result + "again";
  return result;
}
```

Save and Build – things should work. Run it and see.

2 Declarations and Definitions

Next you will experiment with declarations and definitions. (Review the prior lectures, if you don't remember the difference between these.) Save a copy of all your `.h` and `.cpp` files.

1. Now make as simple a change as you can devise that causes your program to fail compilation with an error message

   ```
   error: ... undeclared (first use this function)
   ```

   Use your saved copies to restore your program so that it compiles again.

2. Now make as simple a change as you can devise that causes your program to fail compilation with an error message "Undefined symbol"

   Use your saved copies to restore your program so that it compiles again.

3. Now make as simple a change as you can devise that causes your program to fail compilation with an error message

```
fatal: symbol '...' is multiply-defined:
```

   Use your saved copies to restore your program so that it compiles again.

These three errors are very common, even for experienced programmers, and the error messages tend to look similar (e.g., "undeclared" versus "undefined"). But, hopefully, you can see how each one stems from a very different kind of mistake so that, as you encounter these messages in the future, you will know just what to look for,