

Chapter 11 – Header Files

1. What are Header Files?

Files named so that they end in “.h” are referred to as header files. Header files contain code that can be included in C++ programs via the preprocessor. We already have experience using code found in *iostream*, *string* and *cmath*. In these files, there are class definitions, function headers, and other definitions, but little working code. That is because the bodies of the classes and functions have been compiled to binary and placed in a library of binary code. Since these standard libraries do not have to be compiled before linking into your program, this is a time saver. Compiling this type of code into a library is reserved for code that has been completely debugged and will be used over and over again.

Student code, or learning code, is usually not reused. Once the student has debugged and verified the accuracy of the code, it is usually abandoned. Because students usually do not reuse code and creating a program across multiple files can be an additional point of confusion, students usually create their first header files with the working code inside.

2. Using the Preprocessor with Header Files

To create a header file, open a file and save it with the “.h” ending. This chapter will assume the existence of a header file named *mymath.h*.

Because a header file may be included into a large program in several places, it is important to always wrap the contents of a header file in preprocessor instructions that will prevent the compiler from wasting time by re-compiling already compiled code. This is done with the preprocessor instructions *ifndef* (if not defined), *define*, and *endif*.

The first line in a header file should be a the preprocessor instruction *ifndef*, which checks to see if a name has been defined. If the name has not been defined, everything between the *ifndef* and the *endif* instructions will be available for compiling. If the name has previously been defined, nothing between the *ifndef* and *endif* will be used by the compiler. For example:

```
#ifndef _MYMATH
:
:
... code goes here ...
:
:
#endif
```

In this example, if *_MYMATH* has not been defined, the code between the *ifndef* and *endif* will be compiled. *_MYMATH* is not a variable. It is just a name and contains no value. To avoid confusion with other named objects within the program, the example follows generally accepted practice and starts the name with an underscore, is in all caps, and is otherwise the same name as the header file.

To prevent the code between the *ifndef* and *endif* from being compiled more than once, all that needs to be added to the header file is a define instruction to create `_MYMATH`.

```
#ifndef _MYMATH
#define _MYMATH
:
:
... code goes here ...
:
:
#endif
```

In this example, if `_MYMATH` does not exist, the code will be compiled and `_MYMATH` will be created. This insures that the code in `mymath.h` will not be compiled more than once if `mymath.h` is included into the program twice.

3. Header File Example

Here is an example header file named `mymath.h`. It will contain a function named `mysine` that will accept degrees instead of radians as an argument.

```
#ifndef _MYMATH
#define _MYMATH
#include <cmath>
using namespace std;

double MySine(double degrees)
{
    double radians = degrees * 3.1416 / 180.0;
    return sin(radians);
}

#endif
```

The following is an example program that will make use of *mymath.h*

```
#include <iostream>
#include "mymath.h" // full path to this file may be need inside the quotes
using namespace std;

void main() {
    cout << "Enter Degrees: ";
    double degrees;
    cin >> degrees;
    cout << "The sine of " << degrees << " is "
        << MySine(degrees) << endl;
}
```

If the user enters 90 when prompted, the program run looks like the following:

```
Enter Degrees: 90
The sine of 90 degrees is 1
```

4. Example Using Separate Header and Code Files

Here is the header file example of section 3, but written in 3 files rather than 2. Notice the *include* at the end of the header file which loads the ".cpp." file. This is not standard practice, but is often used with student code to allow students to build programs in multiple files, separating implementation and prototypes into separate files without having to create a project for each program and without having to issue complex compiler directives from the command line.

File with function main in it

```
#include <iostream>
#include "mymath.h" // full path to this file may be needed
//inside the quotes

using namespace std;

void main() {
    cout << "Enter Degrees: ";
    double degrees;
    cin >> degrees;
    cout << "The sine of " << degrees << " is "
        << MySine(degrees) << endl;
}
```

File *mymath.h*

```

#ifndef _MYMATH
#define _MYMATH
using namespace std;

double MySine(double degrees);    // prototype of function

#include "mymath.cpp"             // full path to this file may be needed
                                  // inside the quotes

#endif

```

File *mymath.cpp*

```

#include <cmath>
using namespace std;

double MySine(double degrees) {
    double radians = degrees * 3.1416 / 180.0;
    return sin(radians);
}

```

Exercises

1. What is the purpose of header files?
2. What is the output of the following program?

```

#include <iostream>
using namespace std;

void main() {
#ifndef _POTATOE
#define _POTATOE
    cout << "One Potatoe" << endl;
#endif
#ifndef _POTATOE
    cout << "Two Potatoe" << endl;
#endif
}

```

Programming Assignment 11.1

Put the volume functions of programming assignment 10.1 in a header file. Include the header file in a program that uses the functions found in `mymath.h`.

Programming Assignment 11.2

Create the file `mymath.h` from page 2 of this chapter. Expand it to include two more functions, `MyCosine(double degrees)` and `MyTangent(double degrees)`. These functions should accept degrees rather than radians as parameters and return the cosine of the calling parameter and the tangent of the calling parameter, respectively. Include `mymath.h` in a program that uses the functions found in `mymath.h`.