Students:
Section 5.16 is a part of 2 assignments: CSC108 CH05.8-5.16 C5B

Includes: CA
Due: 04/15/2025, 11:59 PM EDT

Please browse to this assignment through BlackboardLearn so zyBooks knows where to send your activity. Learn more

5.16 Separate files

Separating part of a program's code into a separate file can yield several benefits. One benefit is preventing a main file from becoming unmanageably large. Another benefit is that the separated part could be useful in other programs.

Suppose a program has several related functions that operate on triples of numbers, such as computing the maximum of three numbers or computing the average of three numbers. Those related functions' definitions can be placed in their own file as shown below in the file `threeintsfcts.cpp`.

Figure 5.16.1: Putting related functions in their own file.

main.cpp

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

// Normally lots of other code here

int main() {

   cout << ThreeIntsSum(5, 10, 20) << endl;
   cout << ThreeIntsAvg(5, 10, 20) << endl;

   return 0;
}

// Normally lots of other code here
```

threeintsfcts.cpp

```
int ThreeIntsSum(int num1, int num2, int num3) {
   return (num1 + num2 + num3);
}

int ThreeIntsAvg(int num1, int num2, int num3) {
   int sum;
   sum = num1 + num2 + num3;
   return (sum / 3);
}
```

threeintsfcts.h

```
int ThreeIntsSum(int num1, int num2, int num3);
int ThreeIntsAvg(int num1, int num2, int num3);
```

```
> a.out
35
11
>
```

Feedback?

One could then compile the main.cpp and threeintsfcts.cpp files together as shown below.

Figure 5.16.2: Compiling multiple files together.

Without `#include "threeintsfcts.h"` in main.cpp

```
> g++ -Wall main.cpp threeintsfcts.cpp
main.cpp: In function  int main():
main.cpp:8: error: ThreeIntsSum was not declared in this scope
main.cpp:9: error: ThreeIntsAvg was not declared in this scope
```

With `#include "threeintsfcts.h"` in main.cpp

```
> g++ -Wall main.cpp threeintsfcts.cpp
>
```

Feedback?

Just compiling those two files (without the `#include "threeintsfcts.h"` line in the main file) would yield an error, as shown above on the left. The problem is that the compiler does not see the function definitions while processing the main file because those definitions are in another file, which is similar to what occurs when defining functions after main(). The solution for both situations is to provide function declarations before main() so the compiler knows enough about the functions to compile calls to those functions. Instead of typing the declarations directly above main(), a programmer can provide the function declarations in a header file, such as the threeintsfcts.h file provided in the figure above. The programmer then includes the contents of that file into a source file via the line: `#include "threeintsfcts.h"`.

The reader may note that the .h file could have contained function definitions rather than just function declarations, eliminating the need for two files (one for declarations, one for definitions). However, the two file approach has two key advantages. One advantage is that with the two file approach, the .h file serves as a brief summary of all functions available. A second advantage is that the main file's copy does not become exceedingly large during compilation, which can lead to slow compilation.

One last consideration that must be dealt with is that a header file could get included multiple times, causing the compiler to generate errors indicating an item defined in that header file is defined multiple times (the above header files only declared functions and didn't define them, but other header files may define functions, types, constants, and other items). Multiple inclusion commonly can occur when one header file includes another header file, e.g., the main file includes file1.h and file2.h, and file1.h also includes file2.h – thus, file2.h would get included twice into the main file.

The solution is to add some additional preprocessor directives, known as header file guards, to the .h file as follows.

Construct 5.16.1: Header file guards.

```
#ifndef FILENAME_H
#define FILENAME_H

// Header file contents

#endif
```

Feedback?

Header file guards are preprocessor directives, which cause the compiler to only include the contents of the header file once. `#define FILENAME_H` defines the symbol FILENAME_H to the preprocessor. The `#ifndef FILENAME_H` and `#endif` form a pair that instructs the preprocessor to process the code between the pair only if FILENAME_H is not defined ("ifndef" is short for "if not defined"). Thus, if the preprocessor includes encounter the header more than once, the code in the file during the second and any subsequent encounters will be skipped because FILENAME_H was already defined.

Good practice is to guard every header file. The following shows the threeintsfcts.h file with the guarding code added.

Figure 5.16.3: All header files should be guarded.

```
#ifndef THREEINTSFCTS_H
#define THREEINTSFCTS_H

int ThreeIntsSum(int num1, int num2, int num3);
int ThreeIntsAvg(int num1, int num2, int num3);

#endif
```

Feedback?

PARTICIPATION ACTIVITY 5.16.1: Header files.

1) Header files must end with .h.
   - True
   - False

2) Header files should contain function definitions for functions declared in another file.
   - True
   - False

3) Guarding a header file prevents multiple inclusion of that file by the preprocessor.
   - True
   - False

4) Is the following the correct two-line sequence to guard a file named myfile.h?

```
#ifdef MYFILE_H
#define MYFILE_H
```

   - True
   - False

Feedback?

CHALLENGE ACTIVITY 5.16.1: Separate files.

620890.3010016.qx3zqy7

Start

Files main.cpp, basicfunctions.h, and basicfunctions.cpp files are provided. Add the missing function declarations to the basicfunctions.h header file.

Ex: If the input is `6 5 11`, then the output is:

```
Minimum: 5
```

basicfunctions.h | basicfunctions.cpp | main.cpp

```
1 #ifndef BASICFUNCTIONS_H
2 #define BASICFUNCTIONS_H
3
4 /* Your code goes here */
5
6 #endif
```

1 2

Check Next level

Feedback?

Exploring further:

- Preprocessor tutorial on cplusplus.com
- Preprocessor directives on MSDN

How was this section? Provide section feedback

Activity summary for assignment: CSC108 CH05.8-5.16 C5B

Due: 04/15/2025, 11:59 PM EDT

0 / 19 points

Please browse to this assignment through BlackboardLearn so zyBooks knows where to send your activity. Learn more

Completion details