2.8 — Programs with multiple code files

ALEX[1] OCTOBER 23, 2022

Adding files to your project

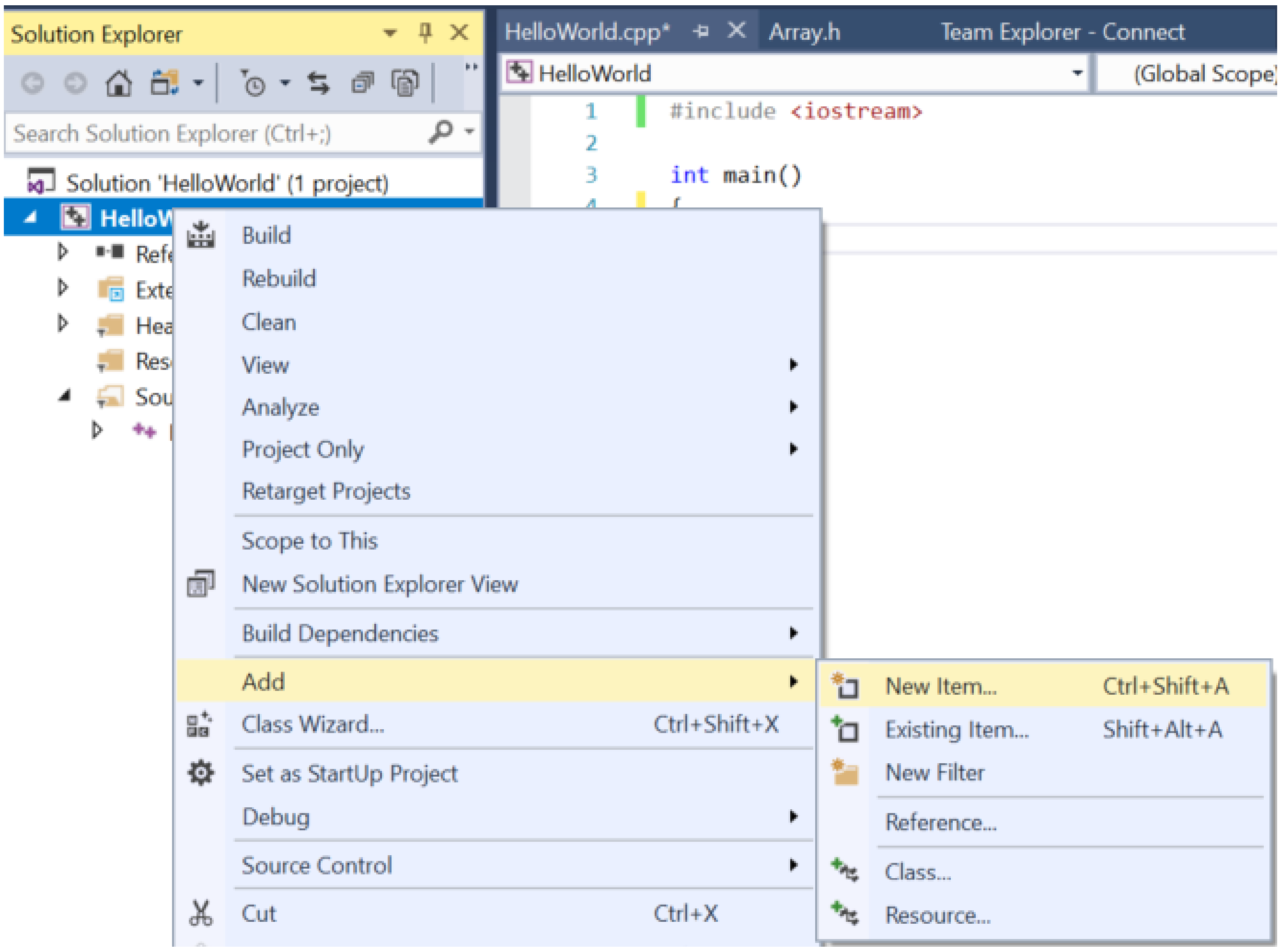
As programs get larger, it is common to split them into multiple files for organizational or reusability purposes. One advantage of working with an IDE is that they make working with multiple files much easier. You already know how to create and compile single-file projects. Adding new files to existing projects is very easy.

Best practice

When you add new code files to your project, give them a .cpp extension.

For Visual Studio users

In Visual Studio, right click on the *Source Files* folder (or the project name) in the Solution Explorer window, and choose *Add > New Item…*.



```
#include <iostream>

int main()
{
```

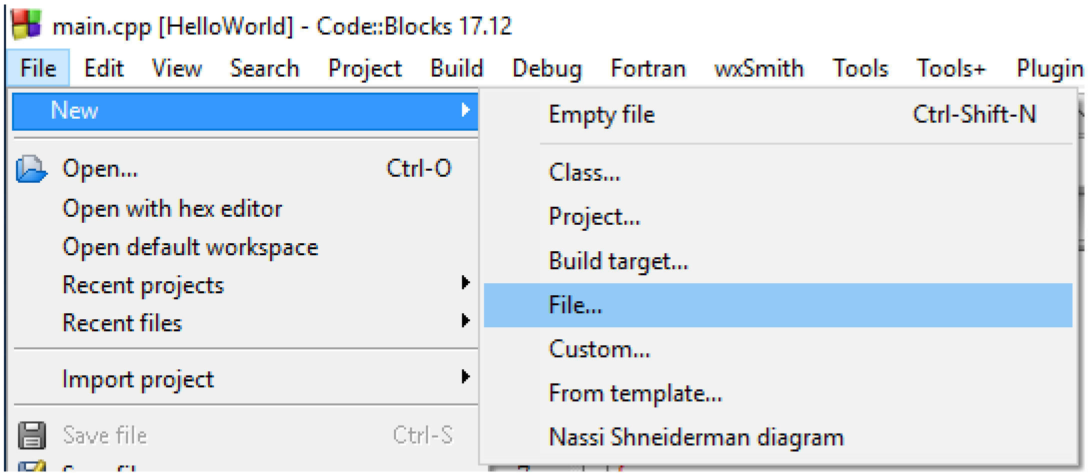
Make sure you have *C++ File (.cpp)* selected. Give the new file a name, and it will be added to your project.

Note: If you create a new file from the *File menu* instead of from your project in the Solution Explorer, the new file won't be added to your project automatically. You'll have to add it to the project manually. To do so, right click on *Source Files* in the *Solution Explorer*, choose *Add > Existing Item*, and then select your file.

Now when you compile your program, you should see the compiler list the name of your file as it compiles it.

For Code::Blocks users

In Code::Blocks, go to the *File menu* and choose *New > File...*.



In the *New from template* dialog, select *C/C++ source* and click *Go*.

You may or may not see a *welcome to the C/C++ source file wizard* dialog at this point. If you do, click *Next*.

On the next page of the wizard, select “C++” and click *Next*.

Now give the new file a name (don't forget the .cpp extension), and click the *All* button to ensure all build targets are selected. Finally, select *finish*.

Now when you compile your program, you should see the compiler list the name of your file as it compiles it.

For GCC/G++ users

From the command line, you can create the additional file yourself, using your favorite editor, and give it a name. When you compile your program, you'll need to include all of the relevant code files on the compile line. For example: *g++ main.cpp add.cpp -o main*, where *main.cpp* and *add.cpp* are the names of your code files, and *main* is the name of the output file.

A multi-file example

In lesson 2.7 -- Forward declarations and definitions (https://www.learncpp.com/cpp-tutorial/forward-declarations/)[2], we took a look at a single-file program that wouldn't compile:

```cpp
 1  #include <iostream>
 2
 3  int main()
 4  {
 5      std::cout << "The sum of 3 and 4 is: " << add(3, 4) << '\n';
 6      return 0;
 7  }
 8
 9  int add(int x, int y)
10  {
11      return x + y;
12  }
```

When the compiler reaches the function call to *add* on line 5 of *main*, it doesn't know what *add* is, because we haven't defined *add* until line 9! Our solution to this was to either reorder the functions (placing *add* first) or use a forward declaration for *add*.

Now let's take a look at a similar multi-file program:

add.cpp:

```cpp
1  int add(int x, int y)
2  {
3      return x + y;
4  }
```

main.cpp:

```cpp
1  #include <iostream>
2
3  int main()
4  {
5      std::cout << "The sum of 3 and 4 is: " << add(3, 4) << '\n'; // compile error
6      return 0;
7  }
```

Your compiler may decide to compile either *add.cpp* or *main.cpp* first. Either way, *main.cpp* will fail to compile, giving the same compiler error as the previous example:

```
main.cpp(5) : error C3861: 'add': identifier not found
```

The reason is exactly the same as well: when the compiler reaches line 5 of *main.cpp*, it doesn't know what identifier *add* is.

Remember, the compiler compiles each file individually. It does not know about the contents of other code files, or remember anything it has seen from previously compiled code files. So even though the compiler may have seen the definition of function *add* previously (if it compiled *add.cpp* first), it doesn't remember.

This limited visibility and short memory is intentional, so that files may have functions or variables that have the same names without conflicting with each other. We'll explore an example of such a conflict in the next lesson.

Our options for a solution here are the same as before: place the definition of function *add* before function *main*, or satisfy the compiler with a forward declaration. In this case, because function *add* is in another file, the reordering option isn't possible.

The solution here is to use a forward declaration:

main.cpp (with forward declaration):

```cpp
#include <iostream>

int add(int x, int y); // needed so main.cpp knows that add() is a function
defined elsewhere

int main()
{
    std::cout << "The sum of 3 and 4 is: " << add(3, 4) << '\n';
    return 0;
}
```

add.cpp (stays the same):

```cpp
int add(int x, int y)
{
    return x + y;
}
```

Now, when the compiler is compiling *main.cpp*, it will know what identifier *add* is and be satisfied. The linker will connect the function call to *add* in *main.cpp* to the definition of function *add* in *add.cpp*.

Using this method, we can give files access to functions that live in another file.

Try compiling *add.cpp* and the *main.cpp* with the forward declaration for yourself. If you get a linker error, make sure you've added *add.cpp* to your project or compilation line properly.

Tip

Because the compiler compiles each code file individually (and then forgets what it has seen), each code file that uses `std::cout` or `std::cin` needs to `#include <iostream>`.

In the above example, if `add.cpp` had used `std::cout` or `std::cin`, it would have needed to `#include .

Something went wrong!

There are plenty of things that can go wrong the first time you try to work with multiple files. If you tried the above example and ran into an error, check the following:

1. If you get a compiler error about *add* not being defined in *main*, you probably forgot the forward declaration for function *add* in *main.cpp*.
2. If you get a linker error about *add* not being defined, e.g.

```
unresolved external symbol "int __cdecl add(int,int)" (?add@@YAHHH@Z) referen
```

2a. ...the most likely reason is that *add.cpp* is not added to your project correctly. When you compile, you should see the compiler list both *main.cpp* and *add.cpp*. If you only see *main.cpp*, then *add.cpp* definitely isn't getting compiled. If you're using Visual Studio or Code::Blocks, you should see *add.cpp* listed in the Solution Explorer/project pane on the left or right side of the IDE. If you don't, right click on your project, and add the file, then try compiling again. If you're compiling on the command line, don't forget to include both *main.cpp* and *add.cpp* in your compile command.

2b. ...it's possible that you added *add.cpp* to the wrong project.

2c. ...it's possible that the file is set to not compile or link. Check the file properties and ensure the file is configured to be compiled/linked. In Code::Blocks, compile and link are separate checkboxes that should be checked. In Visual Studio, there's an "exclude from build" option that should be set to "no" or left blank.

3. Do *not #include "add.cpp"* from *main.cpp*. This will cause the preprocessor to insert the contents of *add.cpp* directly into *main.cpp* instead of treating them as separate files.

Summary

When the compiler compiles a multi-file program, it may compile the files in any order. Additionally, it compiles each file individually, with no knowledge of what is in other files.

We will begin working with multiple files a lot once we get into object-oriented programming, so now's as good a time as any to make sure you understand how to add and compile multiple file projects.

Reminder: Whenever you create a new code (.cpp) file, you will need to add it to your project so that it gets compiled.

Quiz time

Question #1

Split the following program into two files (main.cpp, and input.cpp). Main.cpp should have the main function, and input.cpp should have the getInteger function.

Show Hint (javascript:void(0))3

```
#include <iostream>

int getInteger()
{
    std::cout << "Enter an integer: ";
    int x{};
    std::cin >> x;
    return x;
}

int main()
{
    int x{ getInteger() };
    int y{ getInteger() };

    std::cout << x << " + " << y << " is " << x + y <<
'\n';
    return 0;
}
```

Show Solution (javascript:void(0))3

Next lesson
2.9 Naming collisions and an introduction to namespaces

4

Back to table of contents

5

Previous lesson
2.7 Forward declarations and definitions

2

6

B U URL INLINE CODE C++ CODE BLOCK HELP!

Leave a comment...

Name*

Email*

Find a mistake? Leave a comment!

Notify me about replies:

POST COMMENT

Avatars from https://gravatar.com/[8] are connected to your provided email address.

561 COMMENTS

Newest

padawan
October 27, 2022 6:17 am

shouldn't the file have #include "add.cpp"? How does the linker find the add function definition?

Last edited 3 days ago by padawan

0 Reply

Alex Author
Reply to padawan [9] October 27, 2022 6:41 pm

No, we never include .cpp files. Instead, .cpp files are added to our projects, and compiled as part of our projects. In this case, the compiler will compile add.cpp (into an object file), and the linker will link that object file (containing the add function definition) into our program.

1 Reply

anonyme
October 25, 2022 12:30 pm

Hi. When I compile the code from the quiz the compiler generates one error. It dosen't tell me more about what is wrong. I already checked with the solution and it's exactly the same.

0 Reply

Alex Author

Reply to anonyme 10 October 26, 2022 1:23 pm

Did you check all of the things in the "Something went wrong!" section?

0 Reply

carlos

October 19, 2022 8:31 am

I might be mistaken but I reread the instructions twice but you didn't indicate that on the input.cpp file you have to write #include <iostream> at the top. I honestly have no idea what I did different from the lesson compared to the quiz. but my quiz code wouldn't compile even though I was doing everything right. But once I looked at the answer you had. I noticed the input.cpp file also included its own #include <iostream>.

0 Reply

Alex Author

Reply to carlos 11 October 23, 2022 10:06 am

You always need to `#include <iostream>` in any file that uses std::cout or std::cin. I added a tip to the lesson to make this explicit rather than inferred.

0 Reply

carlos

Reply to carlos 11 October 19, 2022 8:32 am

I appreciate the lessons though! Ill continue learning from you.

0 Reply

Ngan

October 18, 2022 7:44 pm

What happens if i need to compile multiple files in VSCode, do i have to type all the files manually

0 Reply

ted

October 9, 2022 9:07 am

When I run the above program, I get 712142 instead of 7. The code looks right.

0 Reply

Alex Author

Reply to ted[12] October 10, 2022 8:53 pm

Make sure that your `'\n'` contains nothing else. If you put multiple characters inside single quotes, it will print extra weird numbers.

0 Reply

Ted

Reply to Alex[13] October 11, 2022 1:24 pm

Thanks Alex, I had a forward slash ('/n') instead of a back slash ('\n'), it now works

0 Reply

Marian

October 5, 2022 4:05 am

hi bro. I did as you said , made a new add.cpp file in project , put the code you wrote : int add(int x, int y)
{
return x + y;
}
and i still get this error , I don't know how to solve it, please help me.
>------ Build started: Project: Firstproject, Configuration: Debug|Win32 ------
1>C:\Users\maria\Desktop\Projects\Firstproject\Firstproject\Firstproject.obj : error LNK2019: unresolved external symbol "int __cdecl add(int,int)" (?add@@YAHHH@Z) referenced in function _main
1>C:\Users\maria\Desktop\Projects\Firstproject\Debug\Firstproject.exe : fatal error LNK1120: 1 unresolved externals
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========

0 Reply

Alex Author

Reply to Marian 14 October 6, 2022 3:56 pm

It sounds like you created add.cpp, but it was not added to your project (or it is set to not compile) so your compiler is not compiling it as part of your project. In the build summary, I only see one file (1 failed), whereas there should be 2 (whatever your main code file is named, and add.cpp)

0 Reply

Marian

Reply to Alex 15 October 10, 2022 12:44 pm

I found how to do it. Thanks Alex!

0 Reply

Ant

September 14, 2022 4:01 am

Struggled with VS Code using G++

had to change one line in tasks.json from...

"${file}",

to

"${fileDirname}/*.cpp",

thought I'd include here to help future people

0 Reply

Cat1on

Reply to Ant 16 October 14, 2022 3:08 pm

I struggled too. I had to use another editor, im on my Mac and I don't like Xcode, clion seemed to work perfectly for me.

0 Reply

KKW

September 8, 2022 5:48 am

Help! How to add files in VS Code?

0 Reply

Ant

Reply to KKW [17] September 14, 2022 4:02 am

In the explorer (default position is on the left) there is a little + icon to add a file.

0 Reply

right click on the Source Files folder

September 5, 2022 6:47 pm

>>In Visual Studio, right click on the** Source Files** folder in the Solution Explorer window, and choose Add > New Item….

This made me confused. You mentioned to right click on the 'Source Files' folder, but in the picture, you right clicked on the project name.

0 Reply

Alex Author

Reply to right click on the Source Files folder [18] September 13, 2022 5:11 pm

Either works.

2 Reply

thanks

Reply to Alex [19] September 21, 2022 8:41 pm

love you alex

2 Reply

Sarah

August 11, 2022 2:25 am

Hi, I can't seem to add files to my project. I'm working with Visual Studio Code. When I right click on "main", there is no "add" option.

2 Reply

iForgotMyName

Reply to Sarah 20 September 23, 2022 5:18 am

Hello. I struggled with this also. I am using CodeBlocks and in the tutorial it said that we should "create a new file". However if you create new file you can't work with multiple files. Instead you should "create new project", then select "empty project", and then save it as .cpp. This should work.

0 Reply

joey

Reply to Sarah 20 August 21, 2022 3:01 pm

same sarah

0 Reply

Links

1. https://www.learncpp.com/author/Alex/
2. https://www.learncpp.com/cpp-tutorial/forward-declarations/
3. javascript:void(0)
4. https://www.learncpp.com/cpp-tutorial/naming-collisions-and-an-introduction-to-namespaces/
5. https://www.learncpp.com/
6. https://www.learncpp.com/programs-with-multiple-code-files/
7. https://www.learncpp.com/cpp-tutorial/header-files/
8. https://gravatar.com/
9. https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-574262
10. https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-574229
11. https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-574089
12. https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-573905
13. https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-573935
14. https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-573795
15. https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-573841
16. https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-573113
17. https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-572867
18. https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-572746

19. https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-573084
20. https://www.learncpp.com/cpp-tutorial/programs-with-multiple-code-files/#comment-571917
21. https://www.ezoic.com/what-is-ezoic/