2.8 — Programs with multiple code files

BY ALEX ON JUNE 2ND, 2007 | LAST MODIFIED BY ALEX ON FEBRUARY 2ND, 2020

Adding files to your project

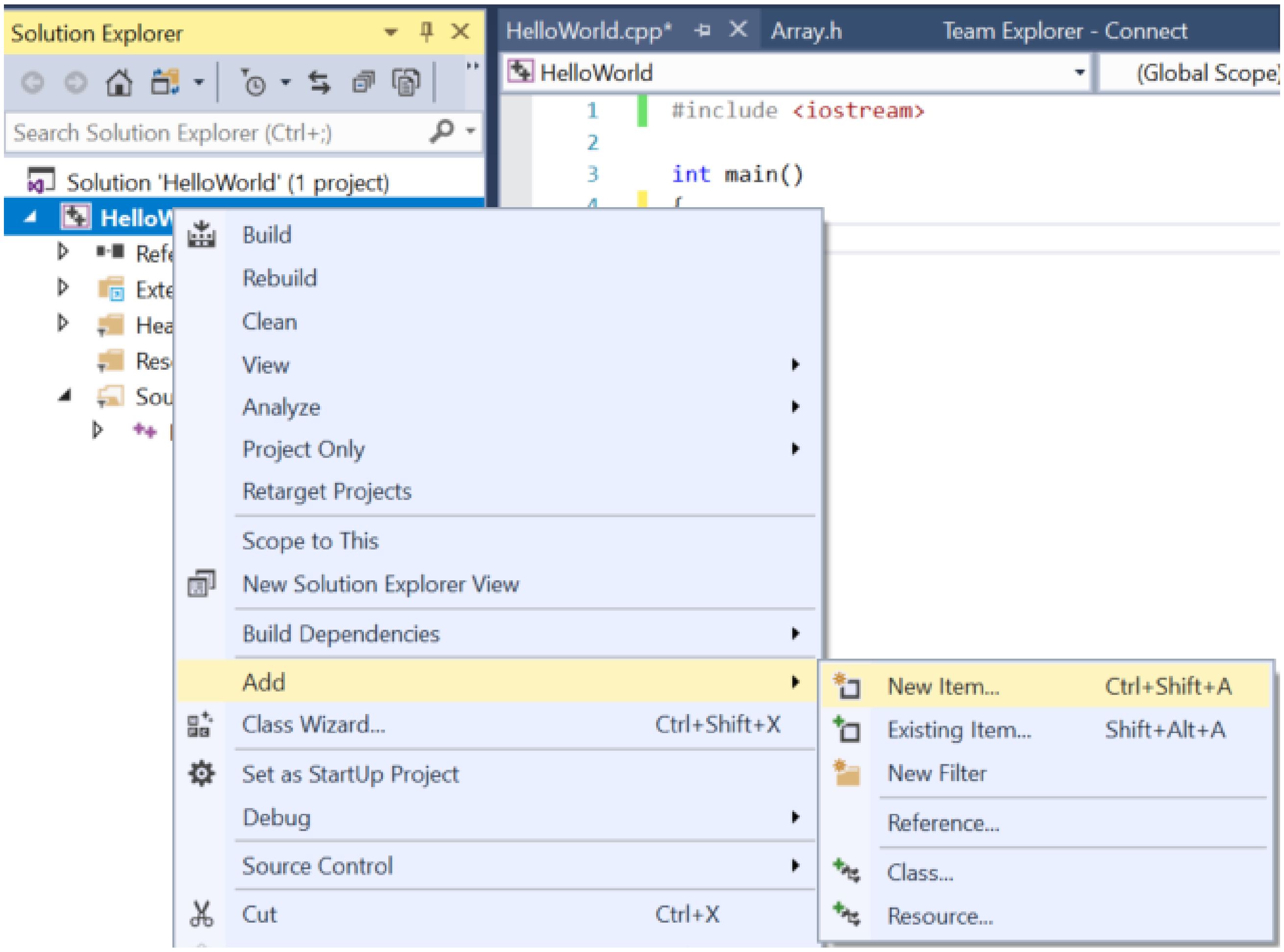
As programs get larger, it is common to split them into multiple files for organizational or reusability purposes. One advantage of working with an IDE is that they make working with multiple files much easier. You already know how to create and compile single-file projects. Adding new files to existing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

Best practice

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

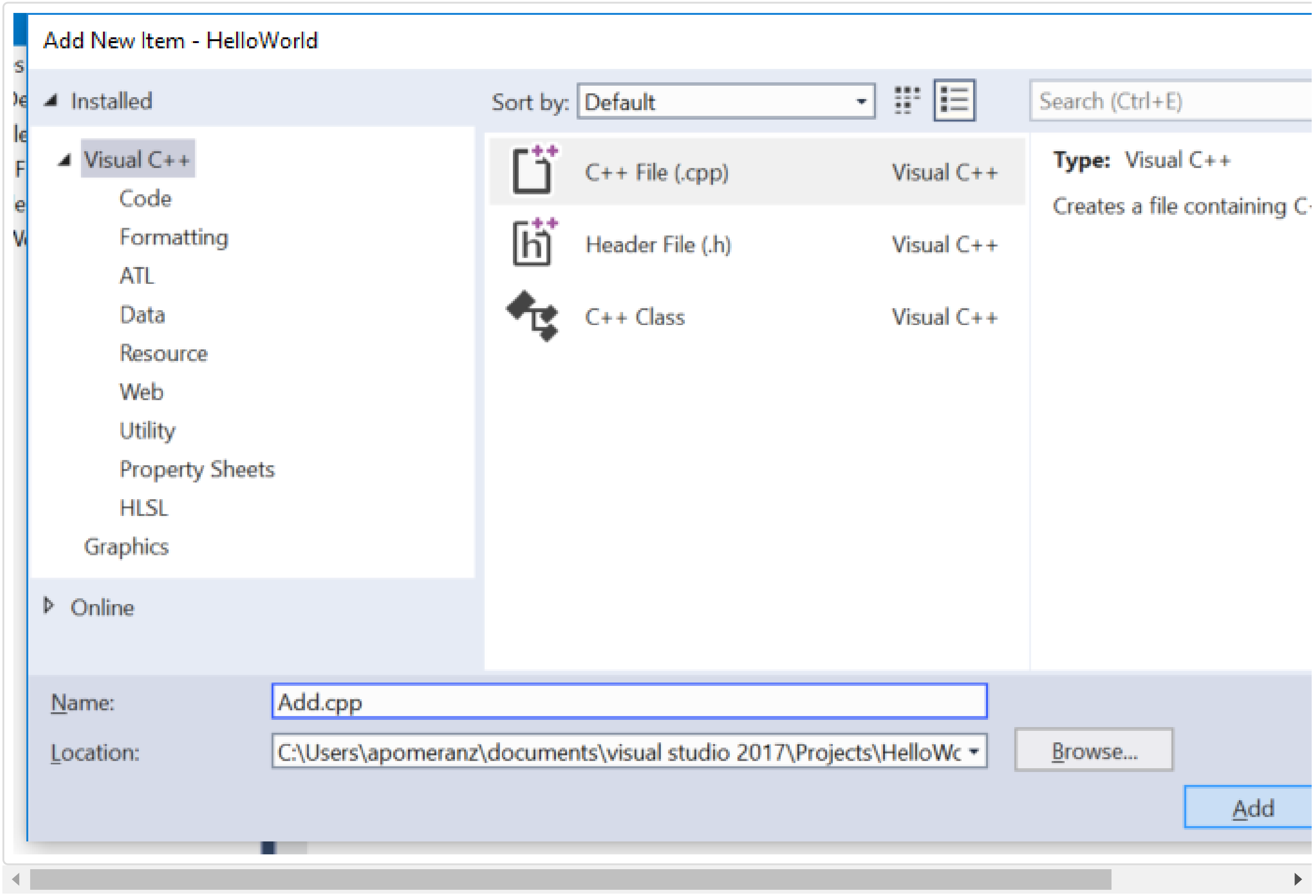
For Visual Studio users

In Visual Studio, right click on the *Source Files* folder 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.

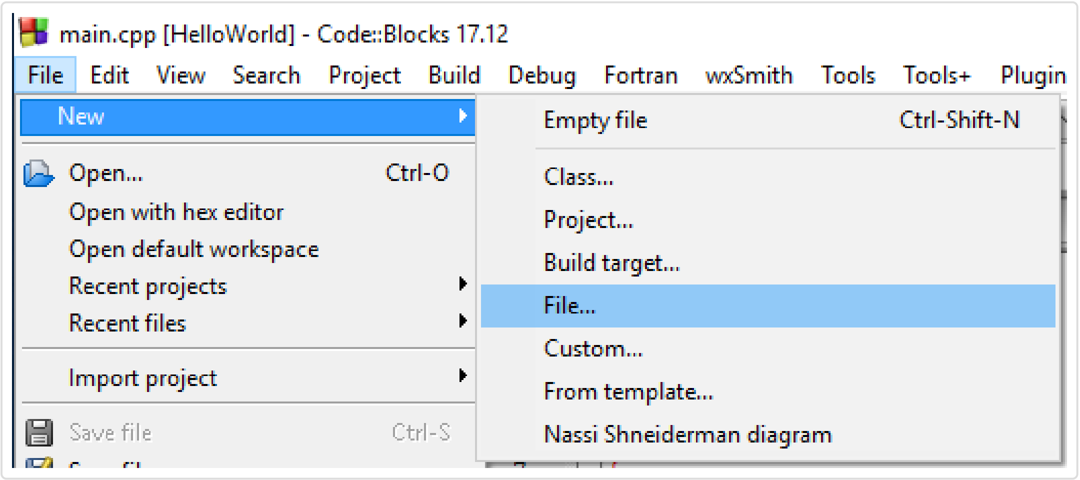


Note: If you create a new file from the *File menu* instead of from your project in the Solution Explorer, the new file won't be added to your project automatically. You'll have to add it to the project manually. To do so, right click on *Source Files* in the *Solution Explorer*, choose *Add* > *Existing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

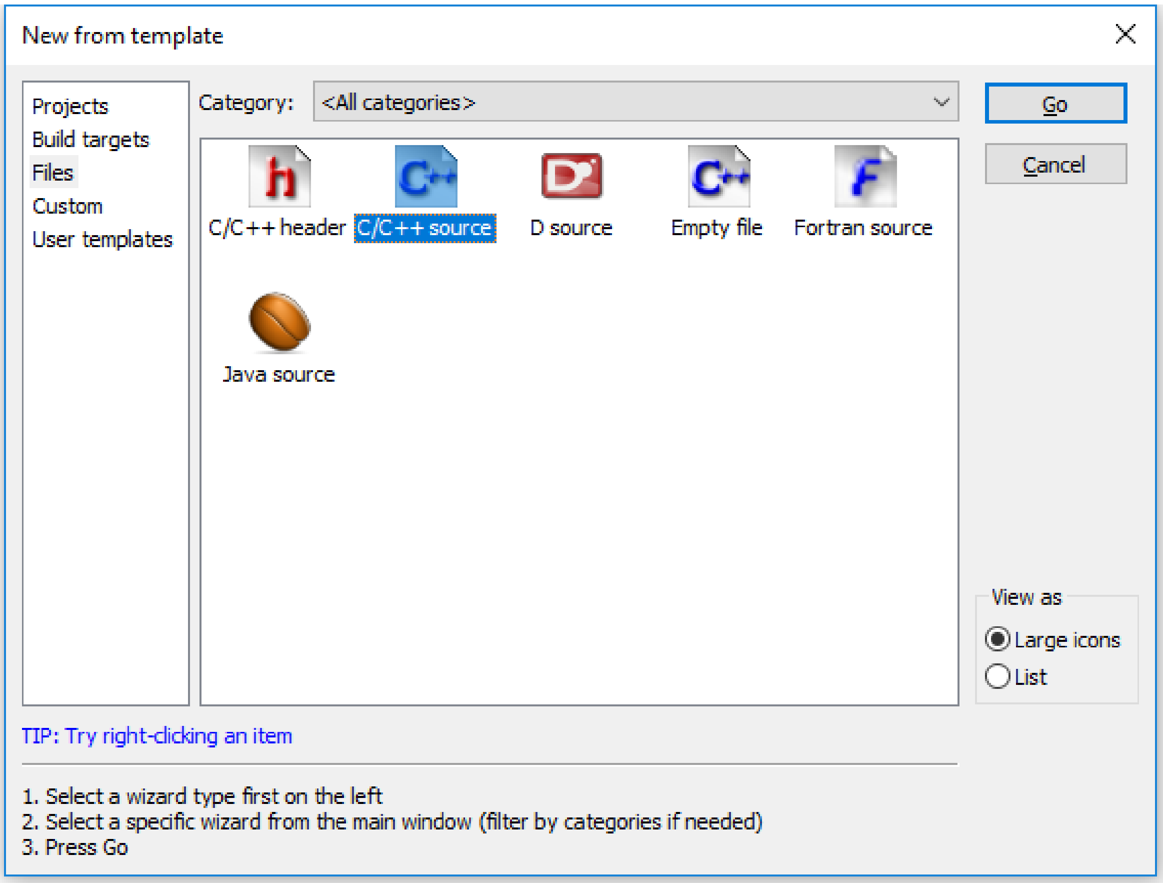
Now when you compile your program, you should see the compiler list the name of your file as it compiles it.

For Code::Blocks users

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



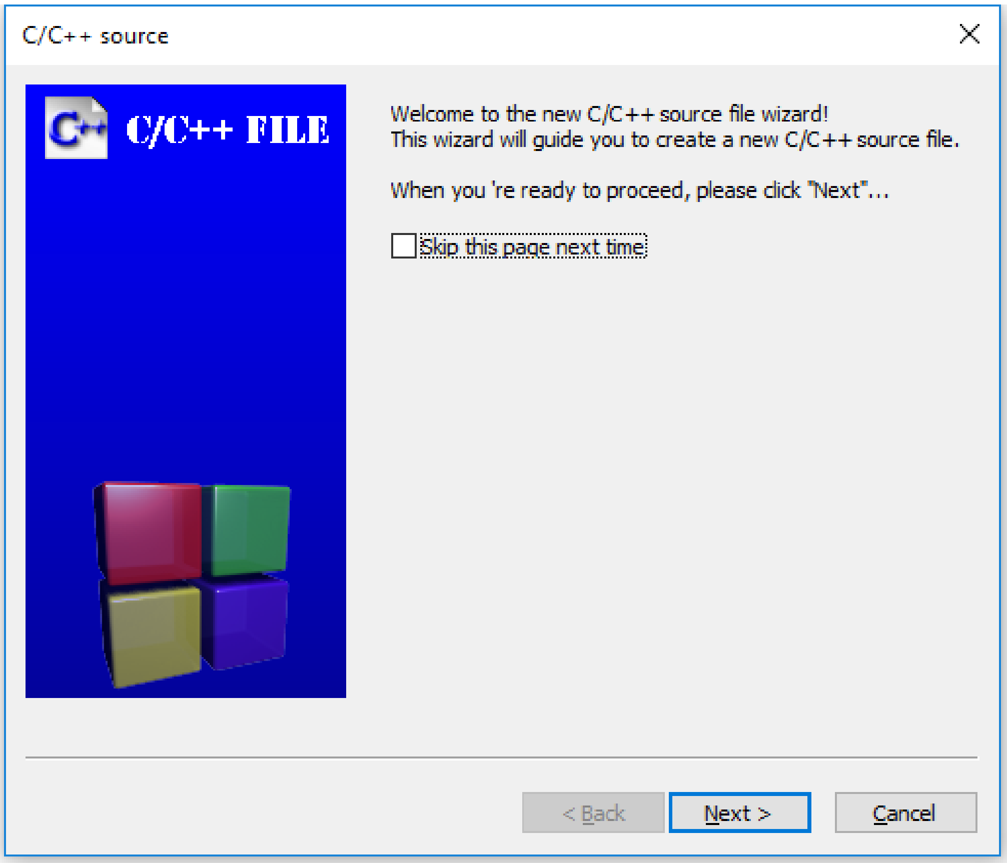
In the *New from template* dialog, select *C/C++ source* and click *Go*.



TIP: Try right-clicking an item

1. Select a wizard type first on the left
2. Select a specific wizard from the main window (filter by categories if needed)
3. Press 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*.

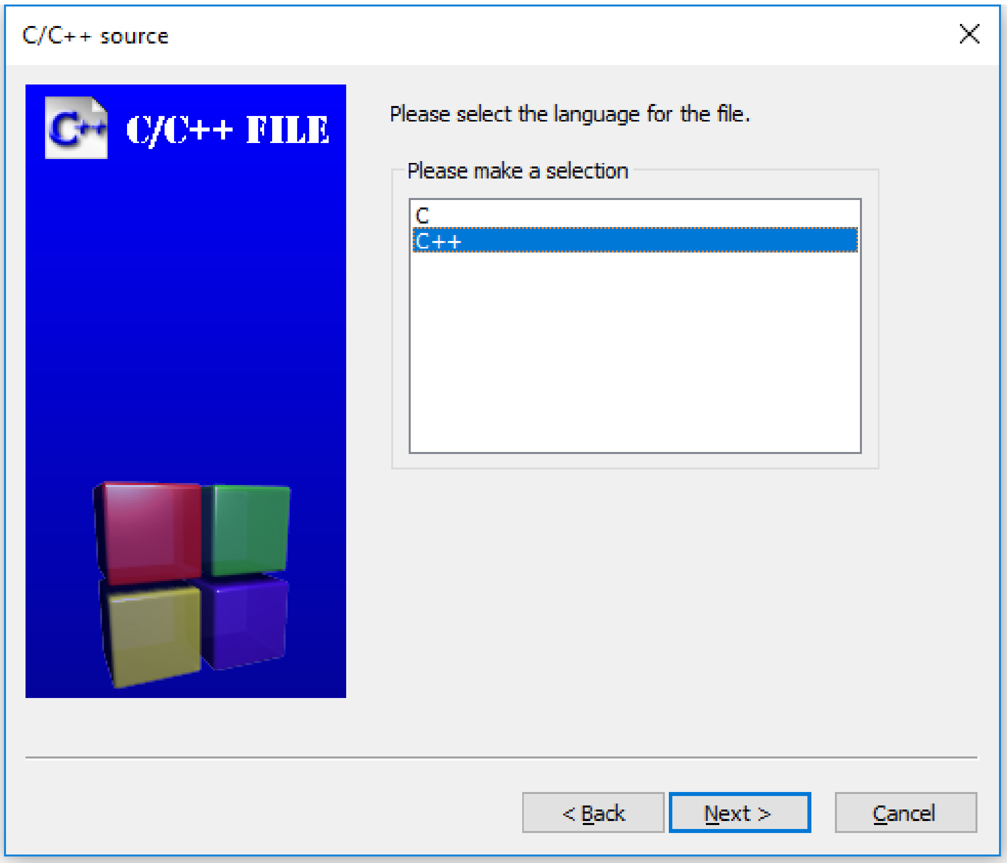


Welcome to the new C/C++ source file wizard!
This wizard will guide you to create a new C/C++ source file.

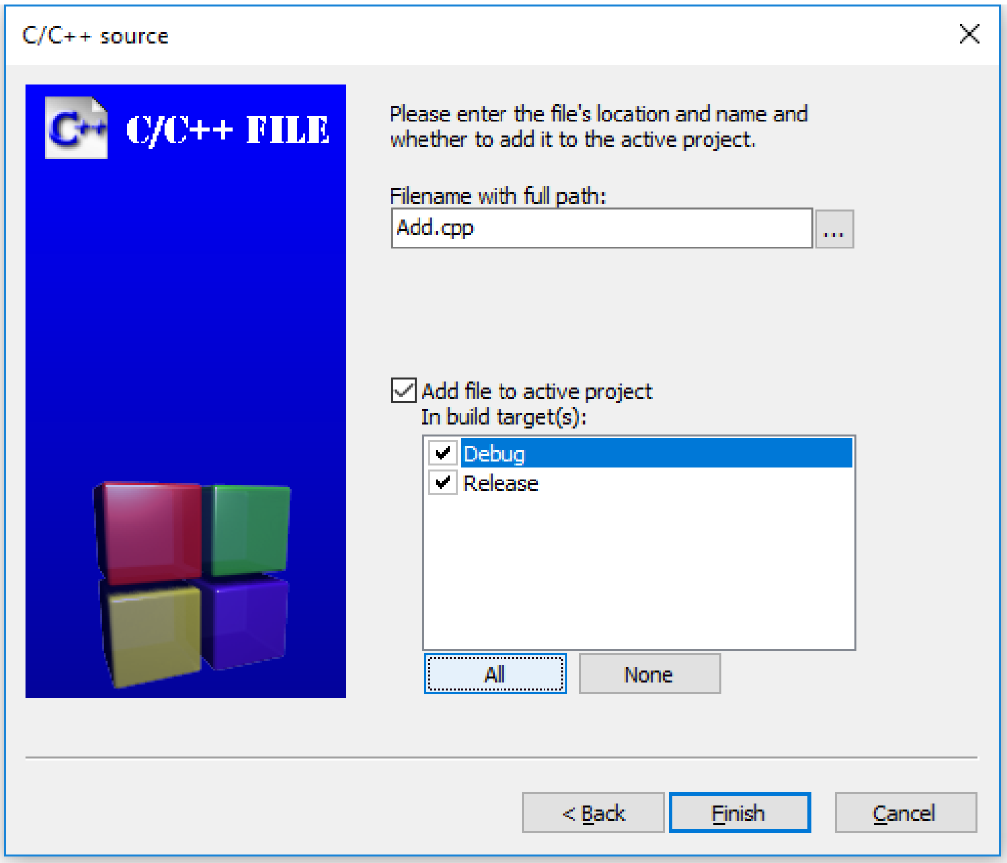
When you 're ready to proceed, please click "Next"...

☐ Skip this page next time

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*.



Please enter the file's location and name and whether to add it to the active project.

Filename with full path: Add.cpp ...

☑ Add file to active project
In build target(s):
☑ Debug
☑ Release

All None

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**, we took a look at a single-file program that wouldn't compile:

```
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:

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

main.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 for 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 a good one.

The better solution here is to use a forward declaration:

main.cpp (with forward declaration):

```
#include <iostream>

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

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

add.cpp (stays the same):

```
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.

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) referenced in funct
```

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 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 compiler to insert the contents of *add.cpp* directly into *main.cpp* instead of treating them as separate files. While it may compile and run for this simple example, you will encounter problems down the road using this method.

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

```cpp
#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

2.9 -- Naming collisions and an introduction to namespaces

Index

2.7 -- Forward declarations and definitions

C++ TUTORIAL | PRINT THIS POST

394 comments to 2.8 — Programs with multiple code files

« Older Comments 1 ... 4 5 6

Ryan
February 2, 2020 at 1:38 am · Reply

Quick typo! In "Our solution to this was to either reorder the functions (placing add first) or using a forward declaration for add", 'using' should be 'use', as in 'was to ... use a forward declaration'.

slate
January 31, 2020 at 8:36 am · Reply

Hi your site is very helpful i learn a lot of better than in my school

I think here

```
int x = getInteger();
int y = getInteger();
```

you should initialize x and y with brace initialize as you teach us in lesson 1.4

nascardriver
February 1, 2020 at 2:54 am · Reply

Absolutely! If you find any more lessons that haven't been updated to brace initialization, feel free to point them out.

Talon Reed
January 29, 2020 at 7:13 pm · Reply

Hey, Alex. I've been doing a decent job of keeping up with learncpp so far, but when it comes to adding multiple programs, I keep running into one particular issue that I can't quite seem to research away. I'm using VS 2017

The error I need help with in particular is LNK2019 {unresolved external symbol main referenced in function "int __cdecl invoke_main(void)" (?invoke_main@@YAHXZ)} I believe it might be something to do with the linker not recognizing main() as main()? I don't know.

I have 2 separate .cpp files, (getInteger.cpp, TestingThings2.cpp) and in both files, it gives me "no issues found" for both. I have no warning, yet I still end up with the error tied to getInteger.

getInteger:
```
#include <iostream>

int   getInteger()
{

    std::cout << "Enter an integer: ";
    int x;
    std::cin >> x;
    return x;
}
```

and TestingThings2:

```
#include <iostream>

int getInteger();

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

    std::cout << x << " + " << y << " is " << x + y << '\n';

    return 0;
}
```

I'm doing my best to solve this error, but no matter what I do, I can't seem to find a way out.
Any help would be incredibly appreciated.

Alex
February 2, 2020 at 12:44 pm · Reply

My best guess is that TestingThings2.cpp isn't being compiled into your program. When you recompile your program, do you see the compiler actually compiling it? If not, you need to add it to your project.

MOJI

January 27, 2020 at 5:38 am · Reply

Hi, i have a problem!! I'm using VS Code!!
I created a workspace and added add.cpp & main.cpp with the codes above.
but the files work independently and the multi-file doent work!!
whta should i do!!?

nascardriver
January 27, 2020 at 5:53 am · Reply

You need to compile both files at the same time. Assuming you're using clang++:

```
clang++ ./add.cpp ./main.cpp
```

MOJI
January 27, 2020 at 6:04 am · Reply

I'm new so i dont get the clang++. :)

I'm using g++ via Mingw

and installed a Run extension

nascardriver
January 27, 2020 at 6:07 am · Reply

You'll have to read to instructions of the extension then.

MOJI
January 27, 2020 at 5:59 am · Reply

I got the result by using the default config of VS Code! (creating task and run the program in powershell)
but the Run button in the Upper-right corner of the codin window gives me the following error and wont compile!! even deletes the already created .exe file through Build Task!!

So to sum it up:

how can i work with Run button and multi-file together!?

Tanx

Harshit Agrawal
December 26, 2019 at 9:43 am · Reply

I am getting this error.

```
1>------ Build started: Project: learn, Configuration: Debug Win32 ------
1>MSVCRTD.lib(exe_winmain.obj) : error LNK2019: unresolved external symbol _WinMain@16 referenced in function "int __cdecl invoke_main(void)" (?invoke_main@@YAHXZ)
1>C:\Users\Harshit Agrawal\source\repos\learn\Debug\learn.exe : fatal error LNK1120: 1 unresolved externals
1>Done building project "learn.vcxproj" -- FAILED.
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
```

My code:

add.cpp

int doadd(int x, int y)
{
  return x + y;
}

learn.cpp

#include <iostream>

int doadd(int x, int y);

int main()
{
  std::cout << "The sum of 3 and 4 is " << doadd(3, 4);
  return 0;
}

Tommy
December 30, 2019 at 2:11 am · Reply

You probably figured it out but I believe you forgot to add
#include <iostream> At the top of add.cpp

```
//This Would be your learn.cpp
#include <iostream>

int doadd(int x, int y);

int main()
{
    std::cout << "The sum of 3 and 4 is " << doadd(3, 4);
    return 0;
}

//And this would be your add.cpp but corrected
#include <iostream>

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

nascardriver
December 30, 2019 at 5:50 am · Reply

"add.cpp" should not include . "add.cpp" doesn't use anything from the standard library. The error is most likely caused by a misconfigured project.

David
December 20, 2019 at 5:14 pm · Reply

I tried putting everything I learned together by writing a program that asks the user for three integers, then calculates and prints the sum of the integers.

The problem is I'm getting an error around sumOf(). I know I need to have the same number of arguments but I'm not sure how to fix it.

The main.cpp file contains the following:

```
#include <iostream>

int add(int x, int y, int z);

void sumOf(int x, int y, int z)
{
    std::cout << "The sum of " << x << ", " << y << ", and " << z << " is " << add(x, y, z)
}

int main()
{

    sumOf( );

    return 0;
}
```

and the add.cpp file contains the following:

```
#include <iostream>
int add(int x, int y, int z)
{
    std::cout << "Enter the first integer: ";
    int x{};
    std::cin >> x;

    std::cout << "Enter the second integer: ";
    int y{};
    std::cin >> y;

    std::cout << "Enter the third integer: ";
    int z{};
    std::cin >> z;

    return x + y + z;
}
```

nascardriver
December 21, 2019 at 4:15 am · Reply

You need `x`, `y`, and `z` when you call `sumOf`. `main` doesn't have them, so you can't call `sumOf`.
Write a function named "getInteger" or similar that reads an int from the user and returns it. Call that function 3 times in `main` and store its return values, which you then pass to `sumOf` (Should be renamed to "printSumOf" or similar). This makes `add` obsolete.

Omran
January 24, 2020 at 6:46 am · Reply

hello i solved your problem you can write this code and then check it out

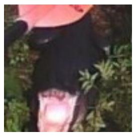
this is main.cpp file

```
#include <iostream>

int add(int x, int y, int z); //forward decleration for function add

int getValue() //this function asks the user to input an integer
```

```
{
    std::cout << "enter an integer:";
    int input;
        std::cin >> input;
    return input;

}

void sumOf(int x, int y, int z) /*reassigning the value main's variables to sumOf's p
                                and also passing the values of main's variables to ad
{
    std::cout << "The sum of " << x << ", " << y << ", and " << z << " is " << add(x,
sumOf's parameter by including the add function in add.cpp file */
}

int main()
{
    int x = getValue();// assigning the value of getValue to variable x
    int y = getValue();// assigning the value of getValue to variable y
    int z = getValue();// assigning the value of getValue to variable z
    sumOf(x , y , z );

    return 0;
}
```

this is add.cpp file

```
#include <iostream>
int add(int x, int y, int z)//adding the values of main's variables
{
    return x + y + z;
}
```



Crizonic
December 19, 2019 at 9:37 am · Reply

I had to put a #include "input.cpp" at the top of main.cpp to get mine to compile. Im using Atom IDE, thought this might help someone.

nascardriver
December 20, 2019 at 1:03 am · Reply

If you have the forward declaration in "main.cpp", the code compiles. If it doesn't for you, you didn't add "input.cpp" to the files that should be compiled. Never include "*.cpp" files, you're bound to break something.

Lord Voldemort
December 16, 2019 at 8:02 am · Reply

In the section "A multi-file example", there is a line "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." what is compilation line?

nascardriver
December 16, 2019 at 8:09 am · Reply

If you're compiling on the command line, the compilation line is the command you use to compile your program.
If you're using an IDE, this doesn't affect you.

NA
November 26, 2019 at 10:45 pm · Reply

Anyone know why syntax highlighting is only working for one source file in Visual Studio? If I open another file to do the multi-file solution, syntax highlighting only works for main.cpp

Tried googling but couldn't find anyone else with the issue. Tools>Options>Environment>Fonts & Colors looks the same on both files.

Thanks!

nascardriver
November 27, 2019 at 4:26 am · Reply

Sounds like your other file doesn't have the right extension (.cpp or .h)

Zorgot
November 30, 2019 at 1:20 am · Reply

I get the same problem, but only when I rename the file, even though it still has the .cpp extension. The only way I got it to work again is by deleting the file and making a new file with the name I want.

Diana
November 26, 2019 at 6:53 am · Reply

Why do you not need a header file such as add.h that contains the declaration of the add() function?

nascardriver
November 26, 2019 at 6:58 am · Reply

You can use a forward declaration instead, a header is just a nice way around having to type forward declarations manually.

Chayim
November 16, 2019 at 11:56 pm · Reply

I pasted the solution from the quiz in a Code Playground" and received the error listed below. Is it because the Code Playground does not have a linker?

Tried to post the error here but it's too long and it's not able to post it.

Chayim
November 16, 2019 at 11:03 pm · Reply

How is it able to understand this chapter correctly if it wasn't explained how linkers work?

Chayim
November 14, 2019 at 12:36 am · Reply

Why compile two files and not merging it in one file? Is it not more efficient and compiles and executes faster and better as one file and not to have it being linked? Is it only because of shortening the work of code not to have to merge it into one file and just linking existing files and done away with it only by linking? Or does it have any purpose?

nascardriver
November 14, 2019 at 1:56 am · Reply

Writing everything in one file gets extremely messy and you'll run into problems with circular dependencies.
The examples on learncpp are single-file for the most part because they're easier to reproduce that way. If you work on a project yourself, you should use multiple files.

fxr
October 15, 2019 at 12:05 pm · Reply

Im trying to make a program that prints and adds using functions on a separate page but I can't get the program to print the results what am I doing wrong.

this is the function page

```
[#include <iostream>

int input()
{
   std::cout << "give num ";
   int R;
   std::cin >> R;
   return R;
}
int add(int x, int y)
{
   return x + y;

}
void print(int add1, int add2, int result)
{
    std::cout << add1 << "+" << add2 << "=" << result;
}]
```

This is the main page

```
[#include <iostream>
int add(int x , int y);
int input();
void print(int f, int g, int q);
int main()
{
   int math{ input() };
   int math1{ input() };
   void print(int math, int math1, int add(int math, int math1));
      return 0;
}]
```

nascardriver
October 16, 2019 at 3:36 am · Reply

```
void print(int math, int math1, int add(int math, int math1));
```

That's a forward declaration of `print`. It doesn't do anything on its own.
When you want to call a function, don't write down the types.

```
print(math, math1, add(math, math1));
```

fxr
October 16, 2019 at 12:10 pm · Reply

Thank you so much it works perfectly now.

Mike
September 10, 2019 at 9:23 am · Reply

For me the issue was the HelloWorld.cpp file I was already using. By adding the files main.cpp and add.cpp, I then had three files, rather than just the two. Once I clicked Source Files, right clicked HelloWorld.cpp, and selected remove, I was good to go.

Hopefully this will help someone else with a similar issue.

Vulryn
August 21, 2019 at 7:54 am · Reply

I tried to make simple program that asks for 2 values and adds them. When i try to run 3 function in same page i get no problem but when i separate them into 3 file i get errors, they are all in source files of the same project. I tried to exclude forward decleration but it also didn't work. What am i doing wrong ?

```
1>main.obj : error LNK2019: unresolved external symbol "int __cdecl calculate(int,int)" (?calculate@@YAHHH@Z) referenced in function _main
1>  Hint on symbols that are defined and could potentially match:
1>    "void __cdecl calculate(int,int)" (?calculate@@YAXHH@Z)
```

My Codes :
main.cpp

```
#include <iostream>
int askvalue();
int calculate(int x, int y);

int main()
{
    int a{ askvalue() }, b{ askvalue() };
    calculate(a, b);
    return 0;
}
```

askvalue.cpp

```
#include <iostream>

```

```
int askvalue()
{
    int value;
    std::cin >> value;
    return value;

}
```

calculate.cpp

```
#include <iostream>

void calculate(int x, int y)
{
    std::cout << x + y;
}
```

nascardriver
August 21, 2019 at 8:01 am · Reply

The error means that you're using a function that has a declaration, but no definition. Your declaration and definition don't match.

```
int calculate(int x, int y)
void calculate(int x, int y)
```

Vulryn
August 21, 2019 at 8:12 am · Reply

I am really stupid. Thank you so much for correcting it ^^

David M.
July 6, 2019 at 8:57 am · Reply

Hi. Love this site. In the first examples you have main.cpp and add.cpp and the function inside of add.ccp is called "int add(int x, int y)". Further down you have input.cpp with the function inside of it called int getInteger(). Question, is that ok to have the .ccp name different than the same name of the function contained within the .ccp file? Also, could the .ccp name be anything random like fdkfdkjdkfj.ccp and have multiple functions inside instead of just one? thanks in advance for any help.

nascardriver
July 6, 2019 at 9:02 am · Reply

The cpp's name is independent of its content but should be chosen to describe its content.

fighter_fish
June 23, 2019 at 10:44 pm · Reply

I think you may have a typo in the section you're explaining how to add files in Code::Blocks

```
Now give the new file a name (don’t forget the .cpp extension), and click the Add button to
```

I think you meant this:

```
Now give the new file a name (don’t forget the .cpp extension), and click the All button to
```

Alex
June 28, 2019 at 8:40 am · Reply

Typo fixed. Thanks for pointing that out!

fighter_fish
July 25, 2019 at 10:01 pm · Reply

Welcome!

GaryTruffles
June 4, 2019 at 8:46 am · Reply

Does it matter the order in which you place cin and cin.ignore()?Thanks

nascardriver
June 4, 2019 at 8:49 am · Reply

yes

waterBottle
June 3, 2019 at 8:59 pm · Reply

If you have your main.cpp, a source.cpp, and source.h files. You include source.h into your main.cpp. This basically copies and pastes the contents of source.h in the same location of #include "source.h" in your main.cpp. Then you link main.cpp and source.cpp to make an object file. And then run it? Is this right? Thanks. Also, if both main.cpp and source.cpp both have #include <iostream> for example, won't you get an error because it is defined twice?

nascardriver
June 4, 2019 at 7:40 am · Reply

Before linking, the files get compiled.

> won't you get an error because it is defined twice?
Headers should only contain declarations (Or definitions without the one-definition-rule). Unless the author of the header violates this convention, you're fine.

mits
June 3, 2019 at 7:43 am · Reply

Question - why am I required, in app.cpp, to include iostream when it is already included in the main.cpp?

nascardriver
June 3, 2019 at 7:47 am · Reply

Source files don't know about each other.

mits
June 3, 2019 at 7:59 am · Reply

Aaah cool, thanks :)

Do Vegans Kill Cockroaches?
May 8, 2019 at 12:30 am · Reply

Have you guys considered making a night mode version of the site? Black background and white words? Y'know, just as an option for us night owls?

nascardriver
May 8, 2019 at 3:21 am · Reply

https://darkreader.org/

Is nascardriver a saint?
May 8, 2019 at 3:43 am · Reply

Wow. Thank you sooo much for this Nas.

Edit: This is amazing! Again, thank you so, so much!

Prateek
May 5, 2019 at 7:08 am · Reply

How to use multiple code files in cpp droid or cxx droid (android apps)?

« Older Comments 1 ... 4 5 6