Installing and Using
The Visual Studio IDE

You can get the "Community" version of Visual Studio free. The community version is very similar to the Enterprise edition; differences are listed here: https://www.visualstudio.com/vs/compare/. Note that in the rest of this document there are screenshots to help you. Sometimes these are just a portion of the entire application, in order to save space.

Downloading and Installing

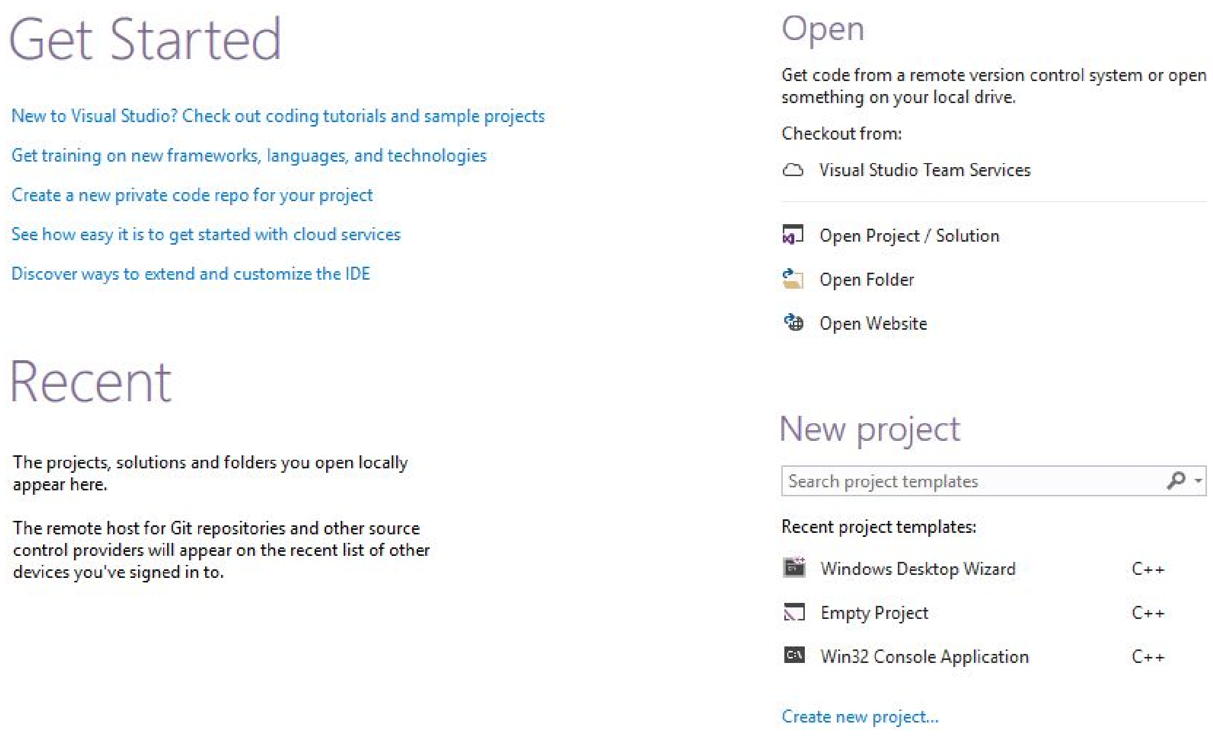
If you're using Windows 8.1 or Windows 10, go to https://www.visualstudio.com/vs/community/ to find the Visual Studio Community 2017 page. Click the "Download Community 2017" button. OR you can download the full Enterprise version of Visual Studio from this ugly link.

If you're still using Windows 7 or 8, I would suggest upgrading. If that's not possible, go to https://www.visualstudio.com/en-us/news/releasenotes/vs2013-archive to find the Visual Studio Community 2013 page. Click the "Download Community 2013" button.

You'll be downloading a web installer that's around 1 MB. Run the application named "vs_Community" (there may be a long version number also) to start the installation process. This will in turn download somewhere around 5 GB to install the entire program, so make sure you have sufficient time, battery, etc. Choose the installation option of "Desktop development with C++". After launching, you will probably have to log in to a hotmail or outlook account (my copy is already set up and no longer forces me to perform this step). If you need to find the menu option for this, it is under Help -> Register Product. After installing, you have to perform a one-time setup. Choose "Visual C++" as your default settings.

Creating Projects

The primary screen you see after starting Visual Studio should look similar to that shown below. The second time you open it, some of your previous projects will appear in the "Recent" area. If the project you want is not there, click on "Open Project" (more on this later).



Get Started

New to Visual Studio? Check out coding tutorials and sample projects

Get training on new frameworks, languages, and technologies

Create a new private code repo for your project

See how easy it is to get started with cloud services

Discover ways to extend and customize the IDE

Recent

The projects, solutions and folders you open locally appear here.

The remote host for Git repositories and other source control providers will appear on the recent list of other devices you've signed in to.

Open

Get code from a remote version control system or open something on your local drive.

Checkout from:

Visual Studio Team Services

Open Project / Solution

Open Folder

Open Website

New project

Search project templates

Recent project templates:

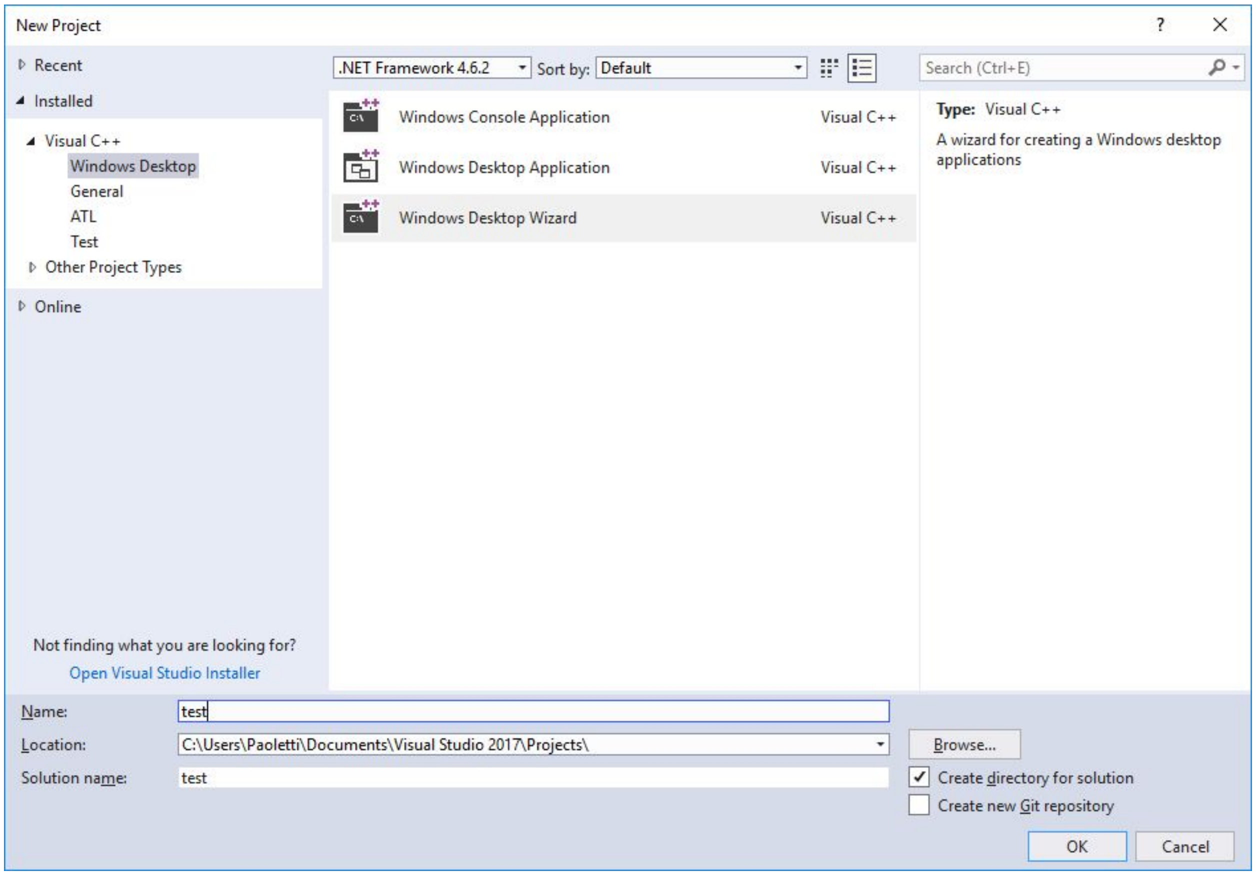
Windows Desktop Wizard	C++
Empty Project	C++
Win32 Console Application	C++

Create new project...

You will probably not have a “Recent project templates” section or it will be empty. I’ve been practicing with different project types.

Click on “Create new project” (in the bottom right) and you will see a window pop up. The default selections that we’re going to make will very likely not be selected the first time that you use it, but they will become the defaults afterward. For Visual Studio Community, choose the project type “Windows Desktop” / “Windows Desktop Wizard”. If you happen to have the Enterprise (or 2015 or older) version, choose “Win32” / “Win32 Console Application”.

To keep project folders consistent between Community and Enterprise editions, we’re going to suggest that you leave the box CHECKED that is labeled “Create directory for solution”. Once you do this, it also becomes the default setting. Type in a name for the new project, such as test, Hello, Project 1, etc.



Windows Console Application	Visual C++
Windows Desktop Application	Visual C++
Windows Desktop Wizard	Visual C++

Type: Visual C++

A wizard for creating a Windows desktop applications

Name: test

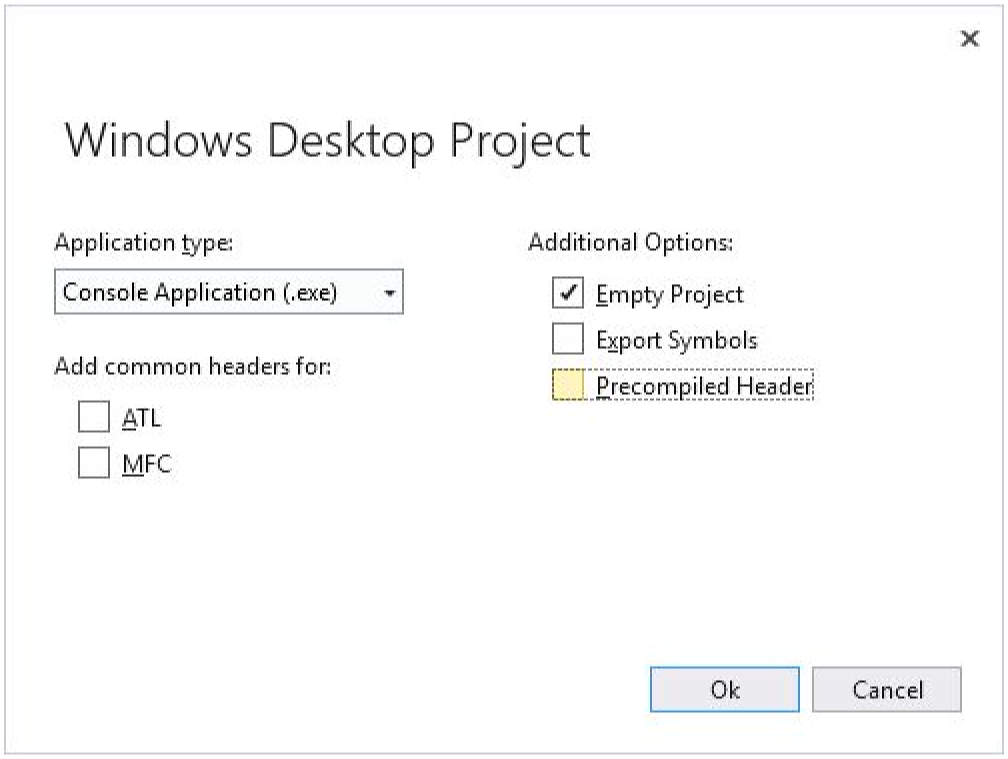
Location: C:\Users\Paoletti\Documents\Visual Studio 2017\Projects\

Solution name: test

[✓] Create directory for solution

[ ] Create new Git repository

After you select “OK”, you will see a second screen very similar to the first. DO NOT click “Ok” yet! First check the box labeled “Empty project”. This means that Visual Studio will not try to create any files for you. Also be sure to UNCHECK the box that says “Precompiled Header”. These do not become default settings, and must be selected every time you create a project. If you create a project and it already contains files named stdafx, you’ve forgotten this step. Close the project and create a new one.


Windows Desktop Project

Application type: Console Application (.exe)

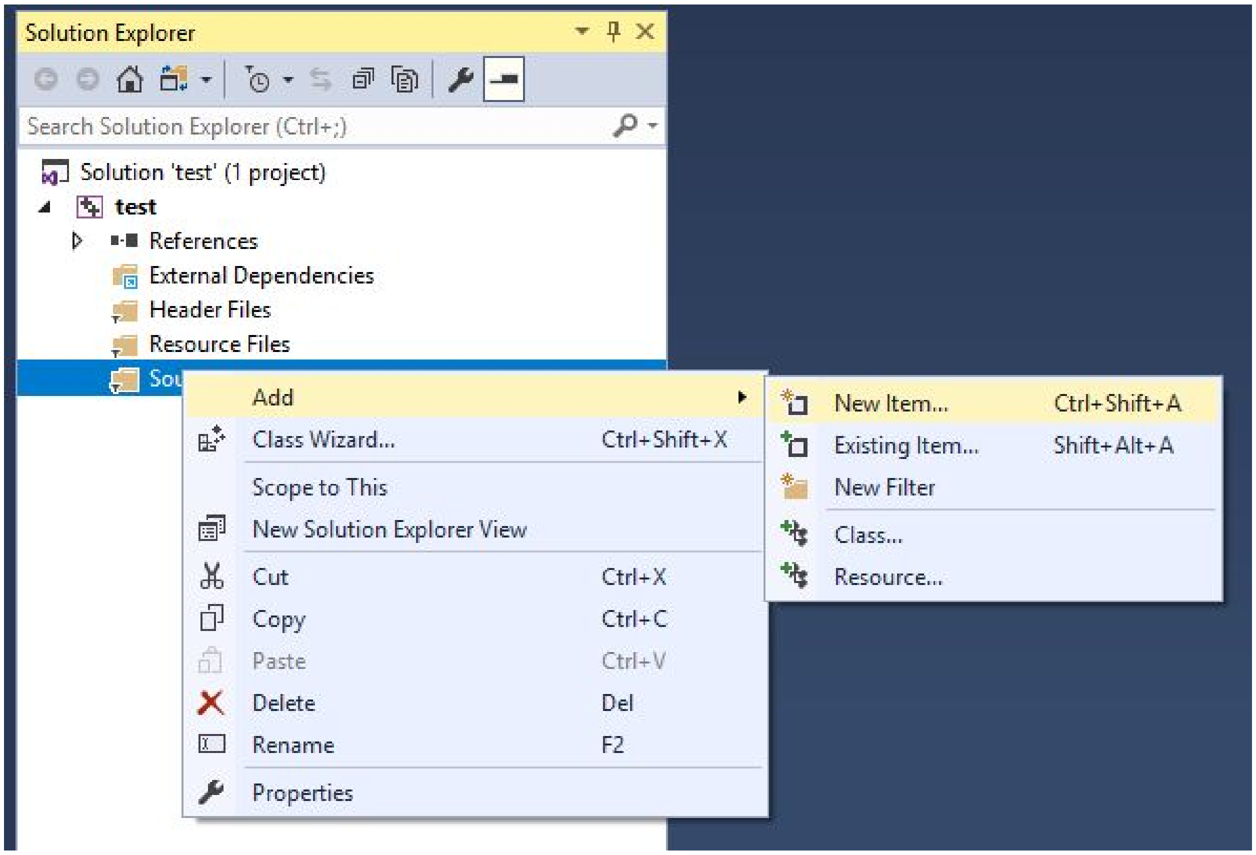
Add common headers for:

- [ ] ATL
- [ ] MFC

Additional Options:

- [x] Empty Project
- [ ] Export Symbols
- [ ] Precompiled Header

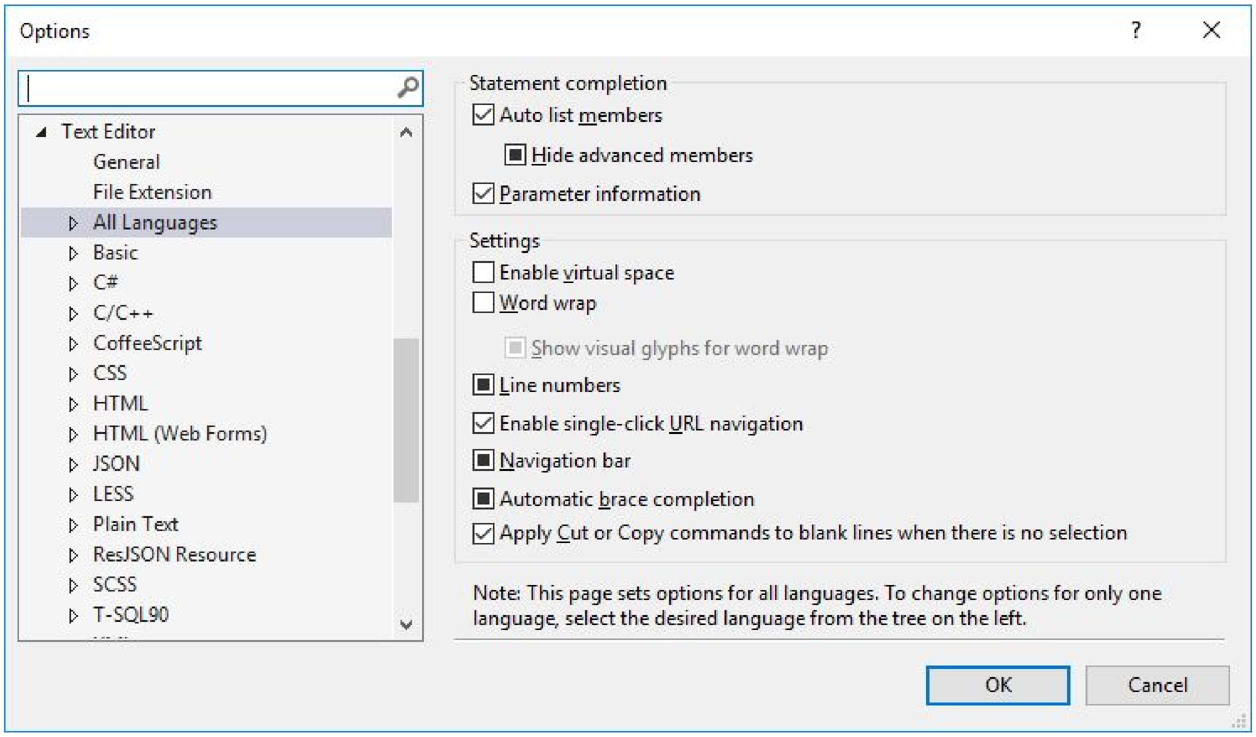
After you create the project, you need a place to put your source code! To do this, right-click on "Source Files" and you'll get a popup menu opened from it. Hover over "Add" and another new menu opens. In this menu, click on "New Item" since we're going to create this program completely from scratch.



To specify what type of file to add, under Visual C++, select the "Code" entry, then select "C++ File (.cpp)". Type in the name of the source code file below. For instance, for this test project, you could type the filename Hello; the .cpp will be added automatically. One thing to be careful of is not to put any periods in the filename. If you do that, Visual Studio will not automatically append the .cpp file type, and then you won't be able to compile.

If you already had source code files, you would copy them to the project folder before this, then choose "Add / Existing Item". When you do this, make sure that your source code files are inside your project folder! If you have to browse around for them, say in a Downloads folder, you should first copy or move them to the project folder (outside of Visual Studio, such as Windows Explorer).

To make editing easier, you should turn on line number display (if it's not already on by default). To do that, choose the menu Tools -> Options, expand the Text Editor option (click the triangle to the left of it), then choose "All Languages". Under Settings, select "Line numbers" and click OK.



Statement completion

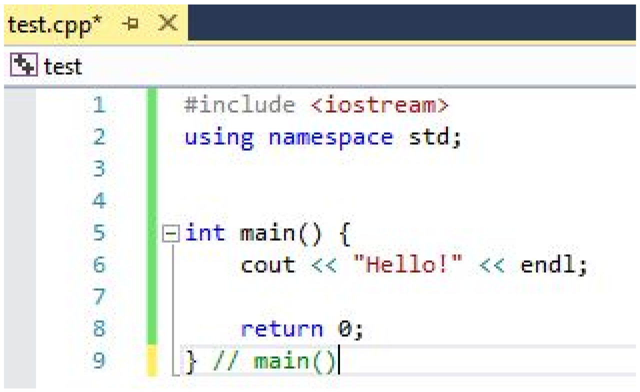
- [✓] Auto list members
- [■] Hide advanced members
- [✓] Parameter information

Settings

- [ ] Enable virtual space
- [ ] Word wrap
- [ ] Show visual glyphs for word wrap
- [■] Line numbers
- [✓] Enable single-click URL navigation
- [■] Navigation bar
- [■] Automatic brace completion
- [✓] Apply Cut or Copy commands to blank lines when there is no selection

Note: This page sets options for all languages. To change options for only one language, select the desired language from the tree on the left.

To make sure that things are working, type in a simple "Hello, world" type program. That will let you get a little practice with the editor, make sure that you can build and run a project, etc. For example:



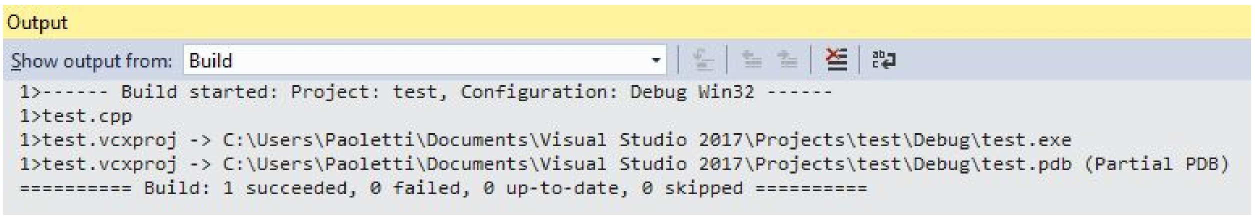
```
#include <iostream>
using namespace std;


int main() {
    cout << "Hello!" << endl;

    return 0;
} // main()
```

Notice the "test.cpp*" tab above; that shows the name of the file you're currently editing, and the asterisk means it has not been saved. You don't need to save every file separately though!

When you're editing a large project with multiple files, every time you Build (compile), every file is saved automatically.

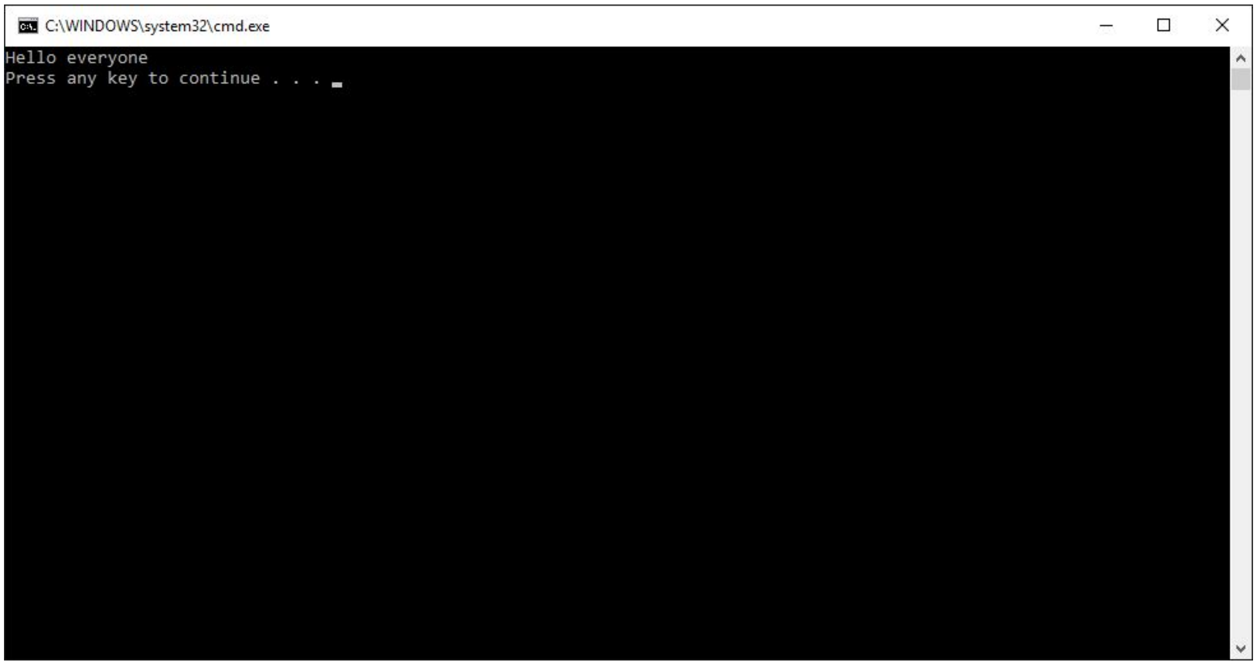
Building

Choose Build -> Build Solution from the menu, and keep an eye on the shortcut keys that are displayed, getting used to using those will save you time in the long run. After choosing this, look at the output window near the bottom; if it compiles correctly you'll see output like that shown below. If not, you can scroll through any error messages and, if they're from the compiler, double-click on them to take you directly to the offending line in the source code!



```
1>------ Build started: Project: test, Configuration: Debug Win32 ------
1>test.cpp
1>test.vcxproj -> C:\Users\Paoletti\Documents\Visual Studio 2017\Projects\test\Debug\test.exe
1>test.vcxproj -> C:\Users\Paoletti\Documents\Visual Studio 2017\Projects\test\Debug\test.pdb (Partial PDB)
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Even if it builds correctly, get in the habit of scrolling up to check for warnings (things like a type mismatch). When you're ready, choose Debug -> Start Without Debugging to run the program. You should see a console window open up similar to the following.



```
Hello everyone
Press any key to continue . . .
```

If the font size is too small, right-click on the "C:\" box at the upper left, choose Properties -> Font, and change the size. When you're done, DO NOT click on the "X" at the upper right, instead do what it says and press any key. If you do that, Visual Studio performs some memory checks (such as invalid pointers) that get skipped if you close it via the “X”. Your hello program probably doesn’t use pointers, but it’s best to get in the habit of pressing any key early.

What happens if your output window opens and then immediately closes before you can read it? I’ve had students who this always happens to, and I’m not sure why, since mine works. There is a way to prevent this. Set a breakpoint on the “return 0” line of main() by either pressing F9 when you’re on that line, or left-clicking in the gray margin to the left of the line numbers. Then use Debug -> Start Debugging to start your project execution.

Source Code Files

If you need to upload your files to CAEN, or download files to Visual Studio, you can look in the project folder on your computer. By default, these are created in the following location:

> Community: C:\Users\<your-name>\source\repos\
> Enterprise: C:\Users\<your-name>\Documents\Visual Studio 2017\Projects\

Rather than having multiple copies of your files, on your PC, on CAEN for compiling there, on a flash drive, etc., it would be better to keep them in one place. Consider using the EECS gitlab server: https://gitlab.eecs.umich.edu. Visual Studio is compatible with it! If you look back a few pages to when we created the project, there is an option on the bottom to “Add to source control”.

Visual Studio and getopt

One problem with using Visual Studio with EECS 281 is that the IDE/compiler does not have the file getopt.h in its include folder, nor does a compiled version of the functions exist. You can overcome this by using two files supplied on the course webpage (Files/Resources/Visual Studio Files): getopt.h and getopt.c.

The getopt.c file is easier to use: simply put a copy in each project that needs it. When you turn in your project, do NOT make getopt.c part of your tarball.

The getopt.h file is a little more problematic. There are two approaches to solving this problem.

1) Easier but poor solution: Put getopt.h inside your project folder, and #include "getopt.h" instead of <getopt.h>. When you pack up your project to test on CAEN or submit, you have to change your include statement to <getopt.h>. You should also NOT include getopt.h as part of your submission tarball. The big problem with this solution is that if you forget to change from double quotes to angle quotes, your project will fail to compile when you submit, and waste time.

2) More difficult but better solution: Put the getopt.h file inside the Visual Studio include folder. For Visual Studio 2015, this folder is usually located in:

c:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\include

However this location is usually locked and can only be accessed if you are “Administrator”. See below for details that depend on your Operating System.

Windows 7/8 method

Click Start -> Run, type in cmd, and instead of pressing Enter, press Shift-Control-Enter. You can then use the copy command to put the file in the correct location by hand, or type explorer to use the Windows Explorer to copy the file to the target folder.

Windows 10 method

Type Windows-X (using the Windows key like a shift key, hold it down and type X, then release the Windows key). From the menu that appears, choose “Command Prompt (Admin)” or “Windows PowerShell (Admin)”, whichever is present. From here, you have to copy the file yourself using commands. Here’s an example of two commands that worked for me, you will have to at least change the user name, if not the folder containing the getopt.h file:

```
cd "\Program Files (x86)\Microsoft Visual Studio 14.0\VC\include"
copy \Users\Paoletti\Downloads\getopt.h
```

Update: I had to look at a different place to put this file on one of my computers, the cd command (all on one line) turned out to be:

```
cd "\Program Files (x86)\Microsoft Visual
Studio\2017\Community\VC\Tools\MSVC\14.11.25503\include"
```

You might need to change the 14.11.25503 to something else. An easier way is to issue multiple cd commands and hit <Enter> after each one, using the <Tab> key to help auto-complete. For example:

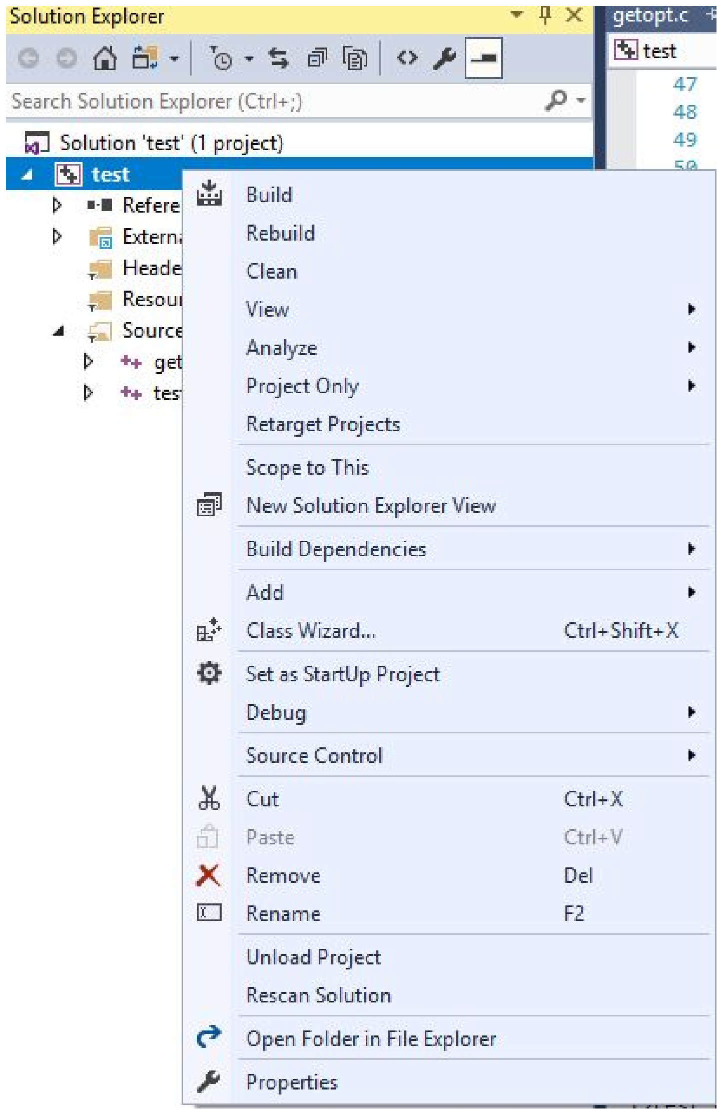
```
cd \progr<Tab><Tab><Enter>
cd microso<Tab><Tab><Tab> ….  Hit <Tab> until it says Microsoft Visual Studio, then <Enter>
cd 20<Tab><Enter>
cd comm<Tab><Enter>
cd vc<Enter>
cd too<Tab<Enter>
cd ms<Tab><Enter>
cd <Tab><Enter>
cd inc<Tab><Enter>
```

If you’re still having trouble

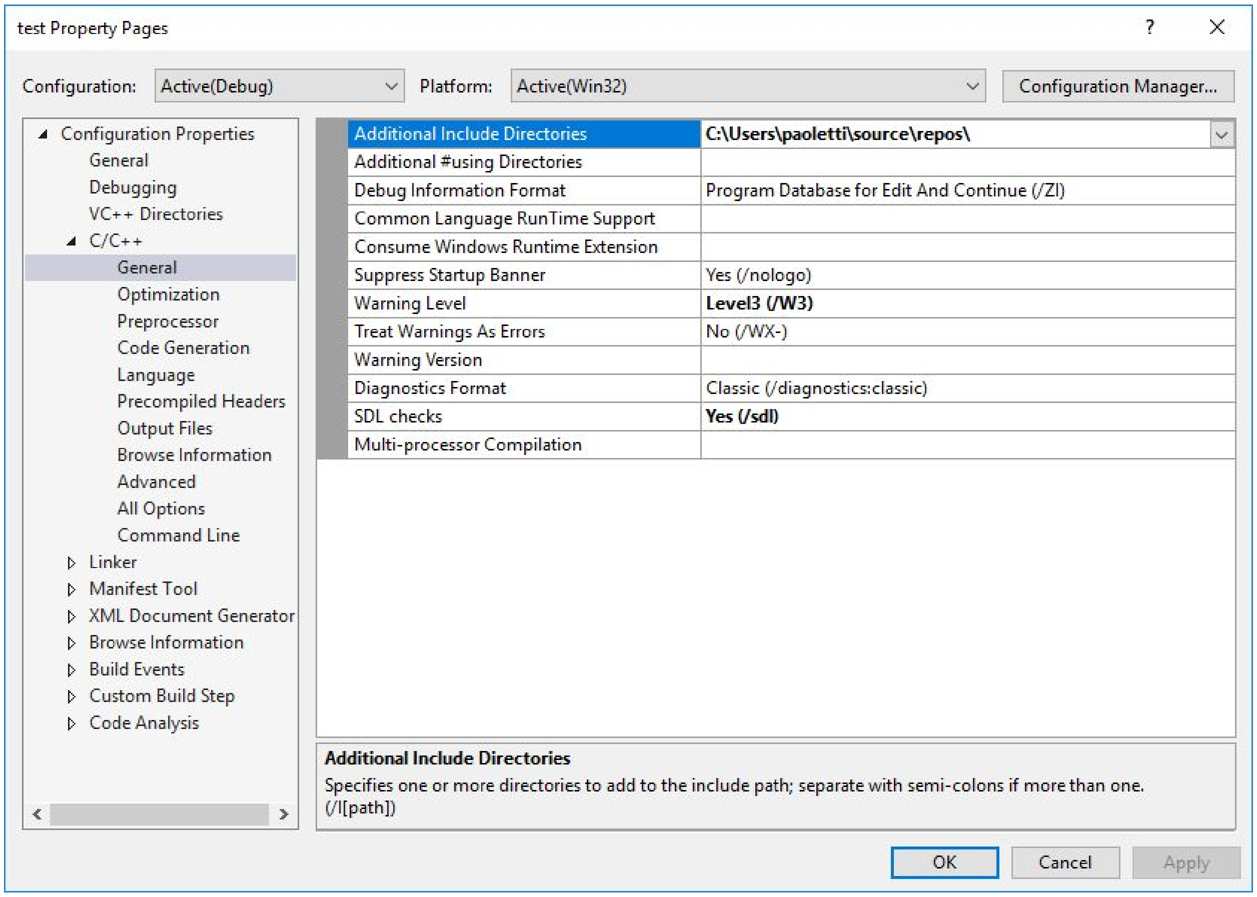
Instead of trying to put the getopt.h file in a folder that requires admin access, you could instead put it in your repos folder. By default, this location is:

```
C:\Users\<your-name>\source\repos\
```

Just put the getopt.h in that folder, then modify EACH project that you create to tell Visual Studio to look in that location. The way to do is this to right-click on the project name (not the solution name, below I right-clicked on **test** which is shown in bold), and choose Properties (last line at the bottom):



After the Properties dialog box is visible, open up the C/C++ menu option (click on the arrow next to it), and edit the “Additional Include Directories” as shown below:



Additional Include Directories	**C:\Users\paoletti\source\repos**
Additional #using Directories	
Debug Information Format	Program Database for Edit And Continue (/ZI)
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)
Warning Level	**Level3 (/W3)**
Treat Warnings As Errors	No (/WX-)
Warning Version	
Diagnostics Format	Classic (/diagnostics:classic)
SDL checks	**Yes (/sdl)**
Multi-processor Compilation	

Additional Include Directories
Specifies one or more directories to add to the include path; separate with semi-colons if more than one. (/I[path])

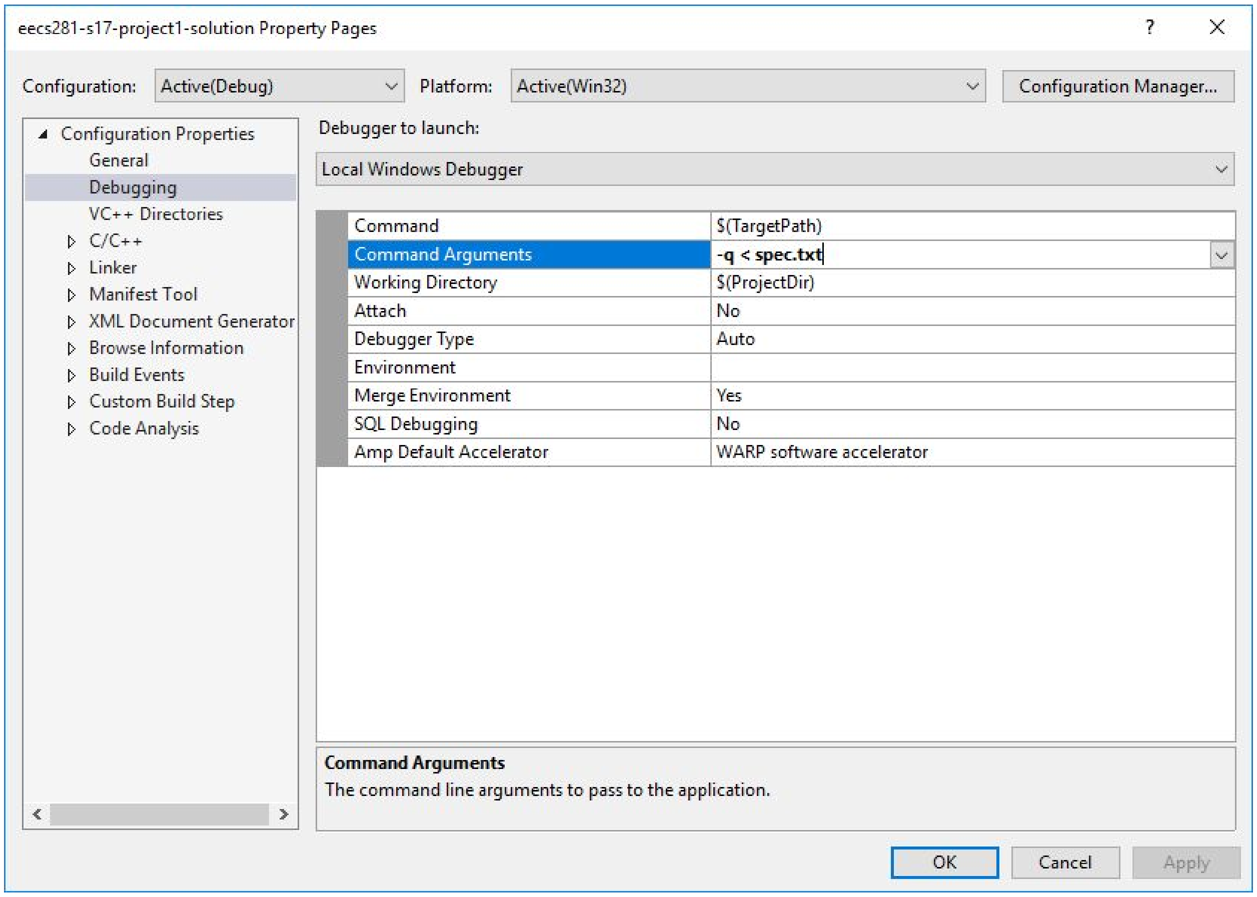
Using Command-Line Parameters

After creating a project, you may want to pass command-line parameters when you run the program, such as flags, filenames, etc. To do this, right-click on the project name (in bold) inside of the Solution Explorer pane, choose Properties (the last option in the popup menu), then select Configuration Properties -> General -> Debugging. The second option on the right should be “Command Arguments”; left-click on the empty place to the right and type in whatever you need. For example, for Project 0 you could type in:

```
    -m nosize < sample-n.txt
```

Where `sample-n.txt` is the name of a text file inside of your project folder, that will be used as input. To create the input file, either use Notepad and Save As into the project folder, or use Windows Explorer to navigate to the project folder and right-click where there are no filenames to create a New -> Text

Document. Put the input file(s) in the same folder as your .cpp files. For Visual Studio to redirect input files to cin, the file must be located inside of the project folder.


eecs281-s17-project1-solution Property Pages

Configuration: Active(Debug) Platform: Active(Win32)

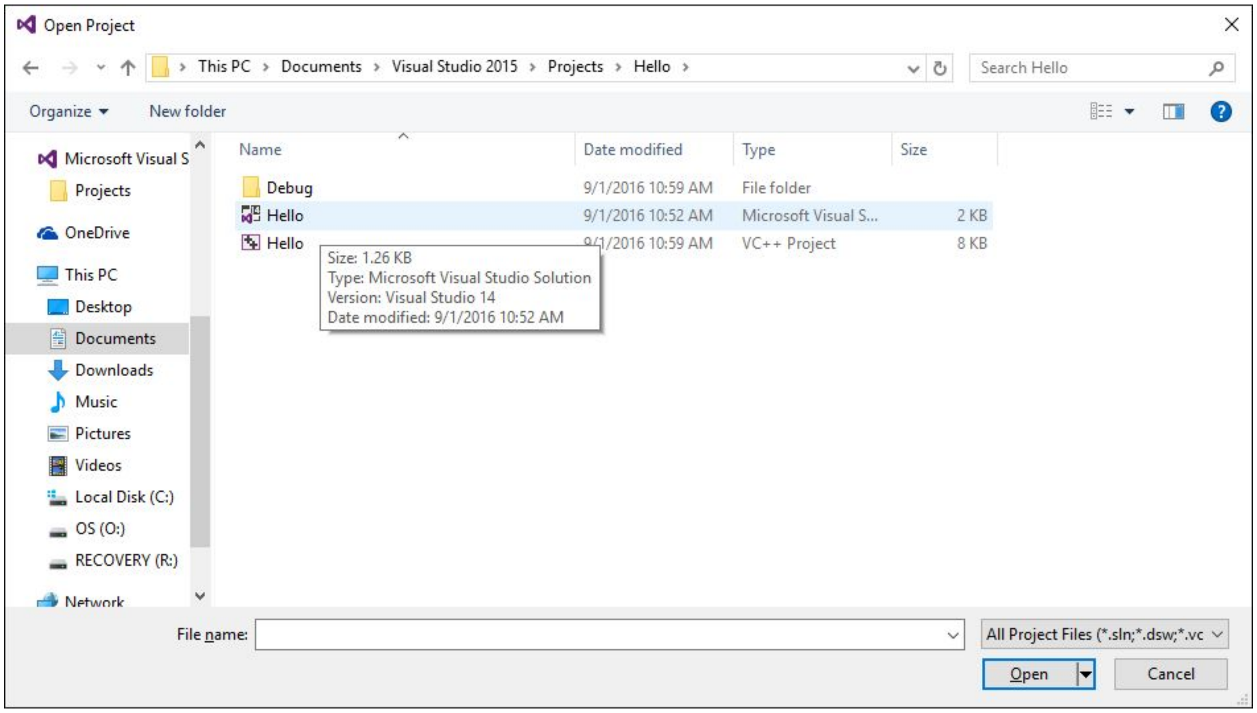
Debugger to launch: Local Windows Debugger

Command	$(TargetPath)
Command Arguments	-q < spec.txt
Working Directory	$(ProjectDir)
Attach	No
Debugger Type	Auto
Environment	
Merge Environment	Yes
SQL Debugging	No
Amp Default Accelerator	WARP software accelerator

Command Arguments
The command line arguments to pass to the application.

Opening a Project

If the project that you want to open has scrolled off your list of recently opened projects, select "Open Project". This will open a file viewer, which forces you to know what to look for. Click on a folder name, then look for a file of type "Microsoft Visual Studio Solution" and select that file:



Name	Date modified	Type	Size
Debug	9/1/2016 10:59 AM	File folder	
Hello	9/1/2016 10:52 AM	Microsoft Visual S...	2 KB
Hello	9/1/2016 10:59 AM	VC++ Project	8 KB

Size: 1.26 KB
Type: Microsoft Visual Studio Solution
Version: Visual Studio 14
Date modified: 9/1/2016 10:52 AM

Resetting Options

Sometimes it is possible to mistakenly click on things and lose access to certain portions of the IDE. For example, if you accidentally close the Solution Explorer that becomes the default setting for all projects. To get it back, click on View -> Solution Explorer. If you make too many changes that you don't like, it is possible to reset the entire user interface to the default settings. To do this, select Tools -> Import and Export Settings -> Reset all settings.