GETTING STARTED WITH

Steve Jones
www.gameinstitute.com

TABLE OF CONTENTS

TABLE OF FIGURES

INTRODUCTION

This document describes, step-by-step, how to install and setup Visual C++ Community 2017, which is one of the free versions of Microsoft Visual Studio. After showing how to install the software you will then learn how to create your first C++ program. The sample program is also found in the *C++ Programming for Games Module I* course textbook or you can type or copy it directly from this document. Let us get started first by downloading and installing Microsoft Visual Studio 2017 Community.

INSTALLING VISUAL STUDIO COMMUNITY 2017

Download your copy of Visual Studio 2017 Community from Microsoft's website.
https://www.visualstudio.com/downloads/

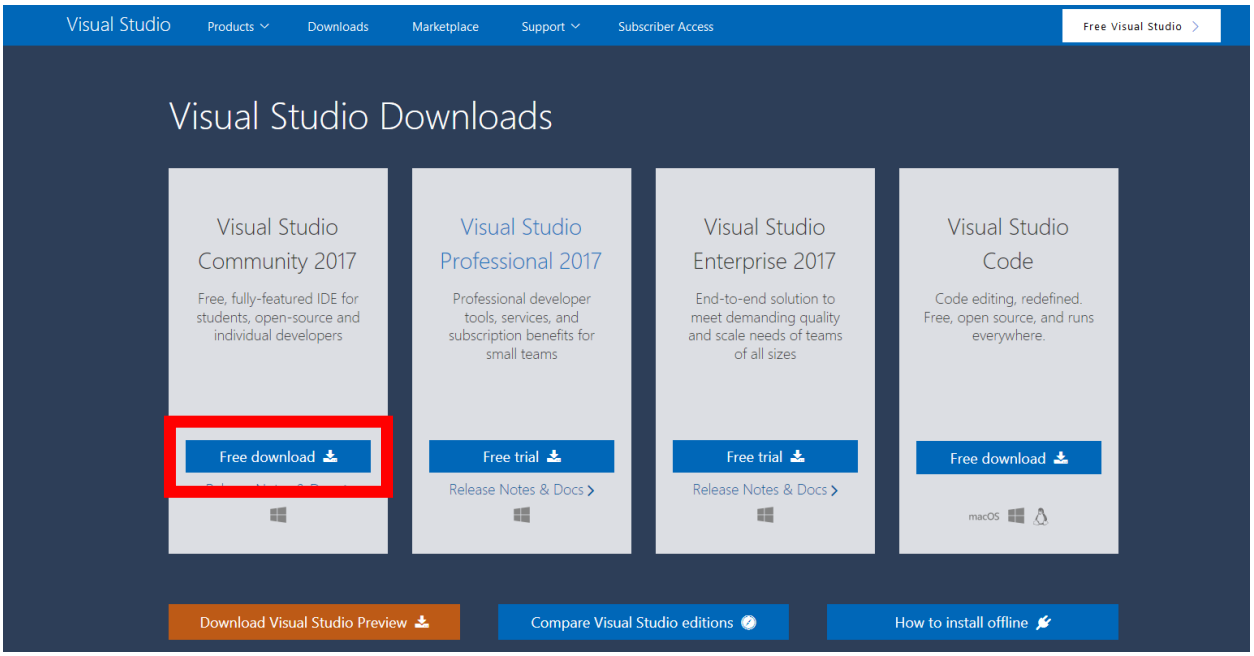


Figure 1. Visual Studio Community 2017 download web page.

Let's get started installing Visual Studio Community 2017!

INSTALLATION PROCEDURES

Launch the executable **vs_community__<big number>.exe** that was downloaded from the Microsoft website. The first screen you will see is the installer loading dialog. Click **Continue**.

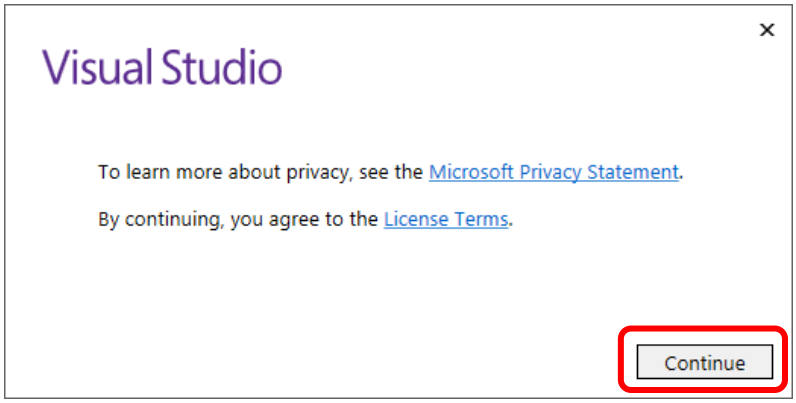


Figure 2. Visual Studio Community 2017 installer loading dialog

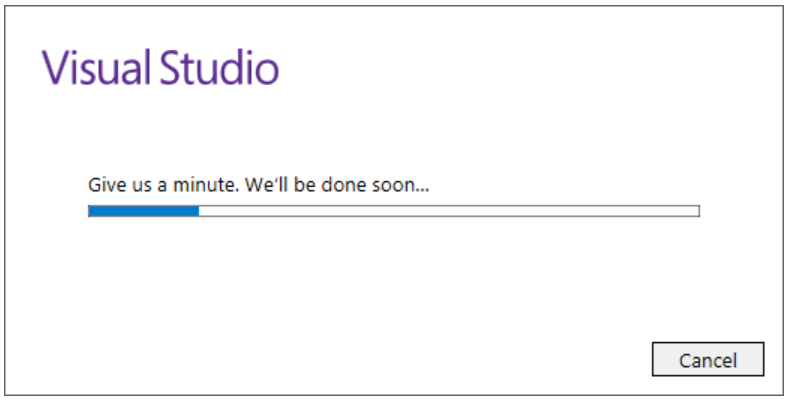


Figure 3. Installation process loading screen.

After the installer window loads, there will be many types of Visual Studio installations and options to select. The type of installation or *Workload* depends on the type of application you will be creating such as Web, Mobile and Desktop Windows applications. For this course you will need the *Desktop development with C++* installation type.

Select **Desktop development with C++**.

There are more options to select for the type Workload that was chosen. The scroll bar can show more options that you may want to install, however this course will only require the default options. Note that you can always run the installer to modify your installation later if you choose.

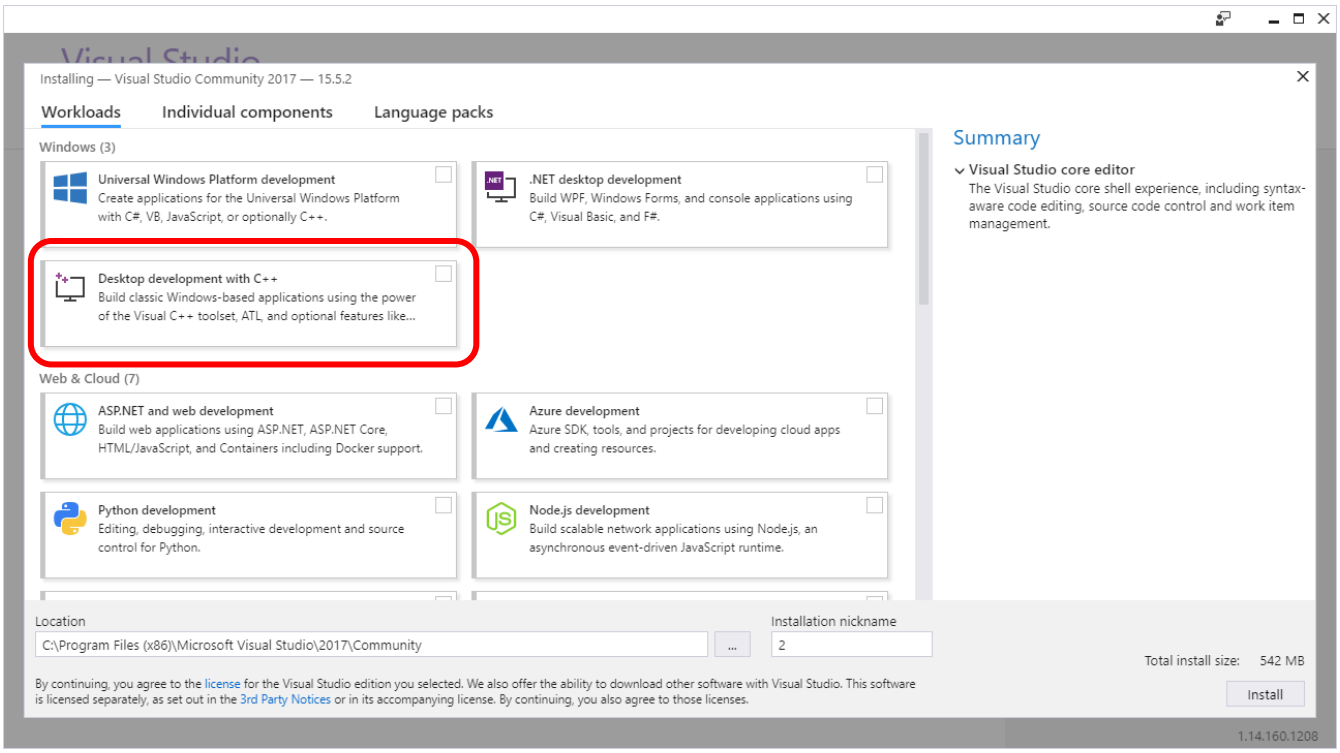


Figure 4. Select Desktop development with C++ for this course.

A summary panel on the left will show the options selected for installation. This installation guide only shows installing the defaults. Other options can be selected to install; however, they are not required for this course.

You need to select option **VC++ 2015.3 v140 toolset for desktop (x86,x64)** option for building the *C++ Module II* Windows applications. This option is not required for *C++ Module I* Console-based applications, however, but it is fine to install it anyway even if you only plan to take *C++ Module I*.

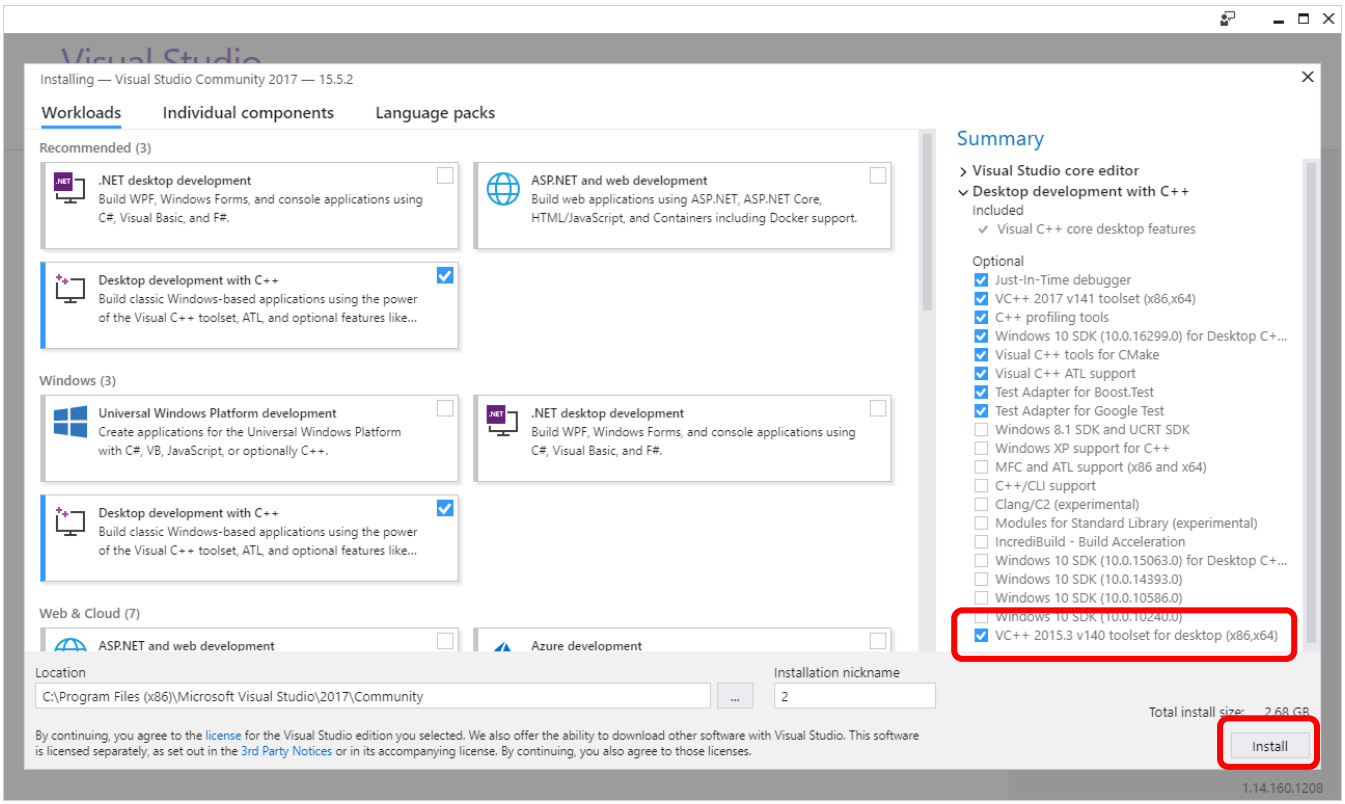


Figure 5. Select options included in the installation.

The software that will be installed is shown in the Summary. Click **Install** to start the installation.

Figure 6 shows the progress of the installation. The installation can take many minutes, so be patient!

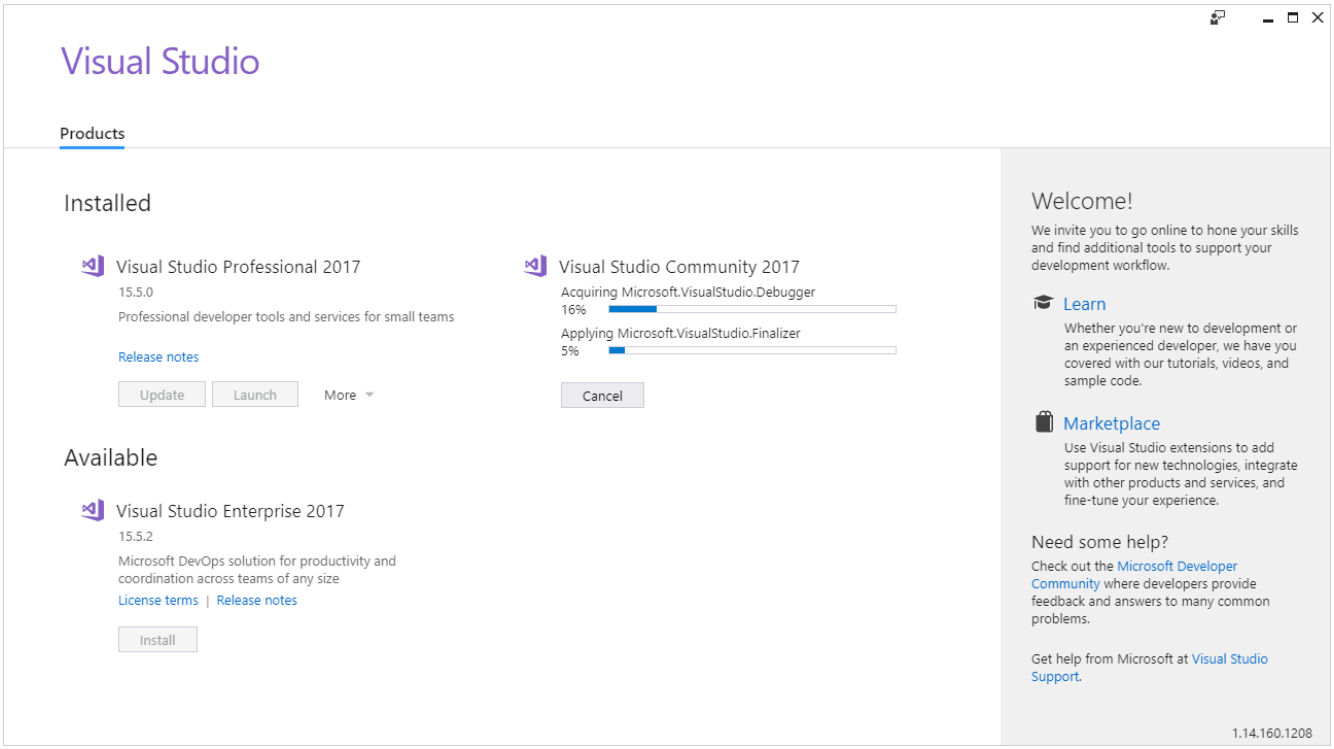


Figure 6. Installing Visual Studio Community 2017.

That's it! You've just installed Visual Studio 2017 Community Edition and have all that you need for the *C++ Module I* and *Module II* courses. There will be an application menu item added. Click **LAUNCH** to start Visual Studio 2017. There are a couple more dialogs that will be presented the first time you run Visual Studio.

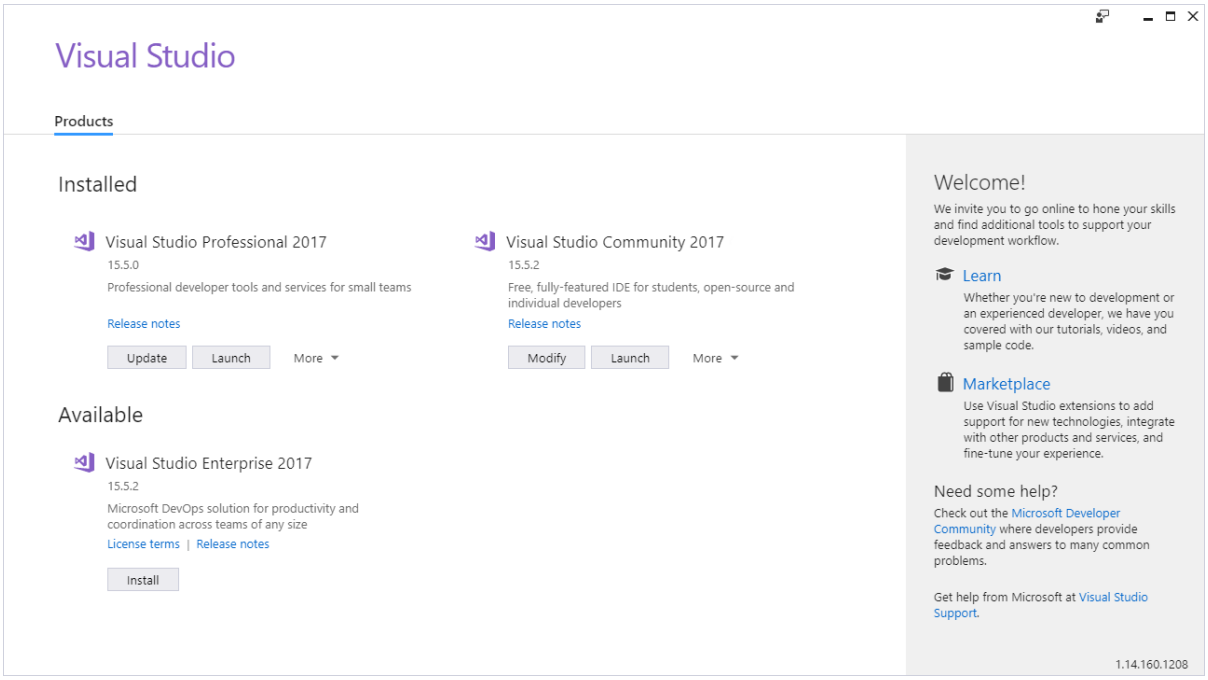


Figure 7. Visual Studio Community 2017 installation complete.

You will be presented with a Welcome login screen during the installation (Figure 8). If you already have a Microsoft account, you can sign in using that account or you can click Sign up to create an account. You will need an account for the free Community license. For now, we will assume you have an account already so click the **"Not now, maybe later"** link to continue.

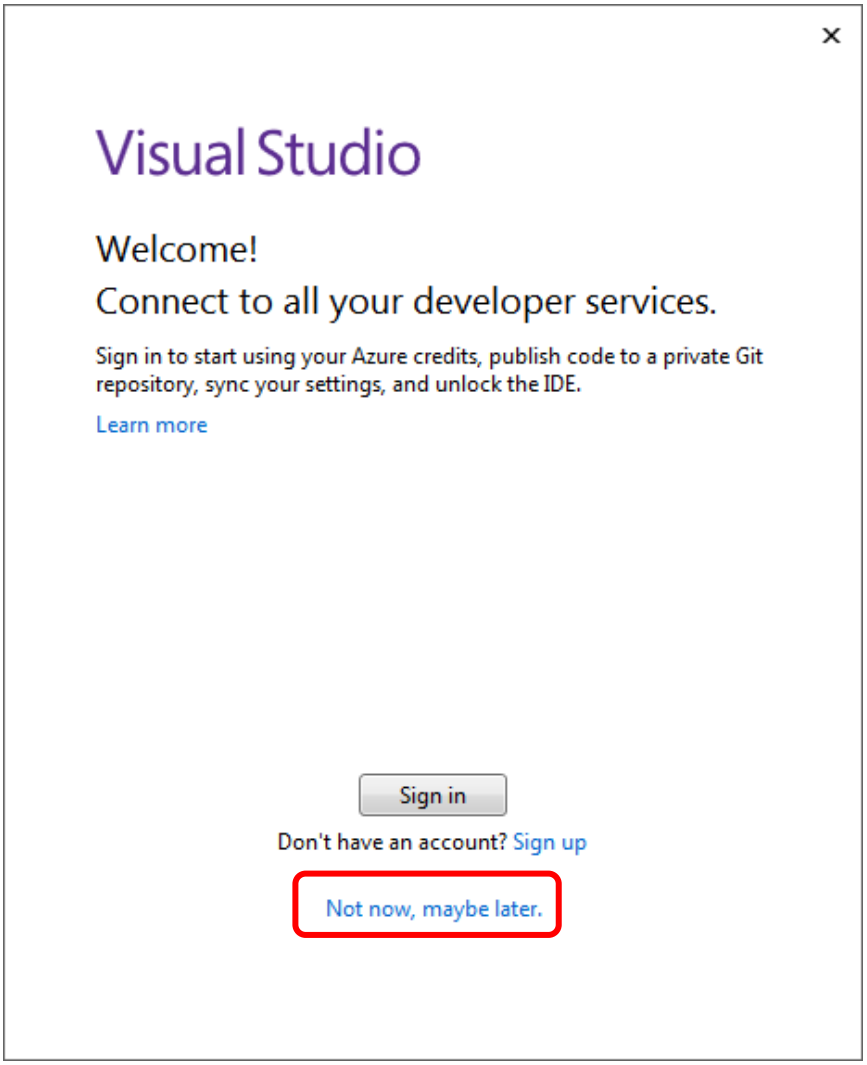


Figure 8. Login screen.

The next screen shown (Figure 9) will allow you to select environment settings such as the color theme and development settings. The Dark theme is particularly easy on the eyes, so that is what was chosen for this tutorial, but you should choose according to your personal preference.

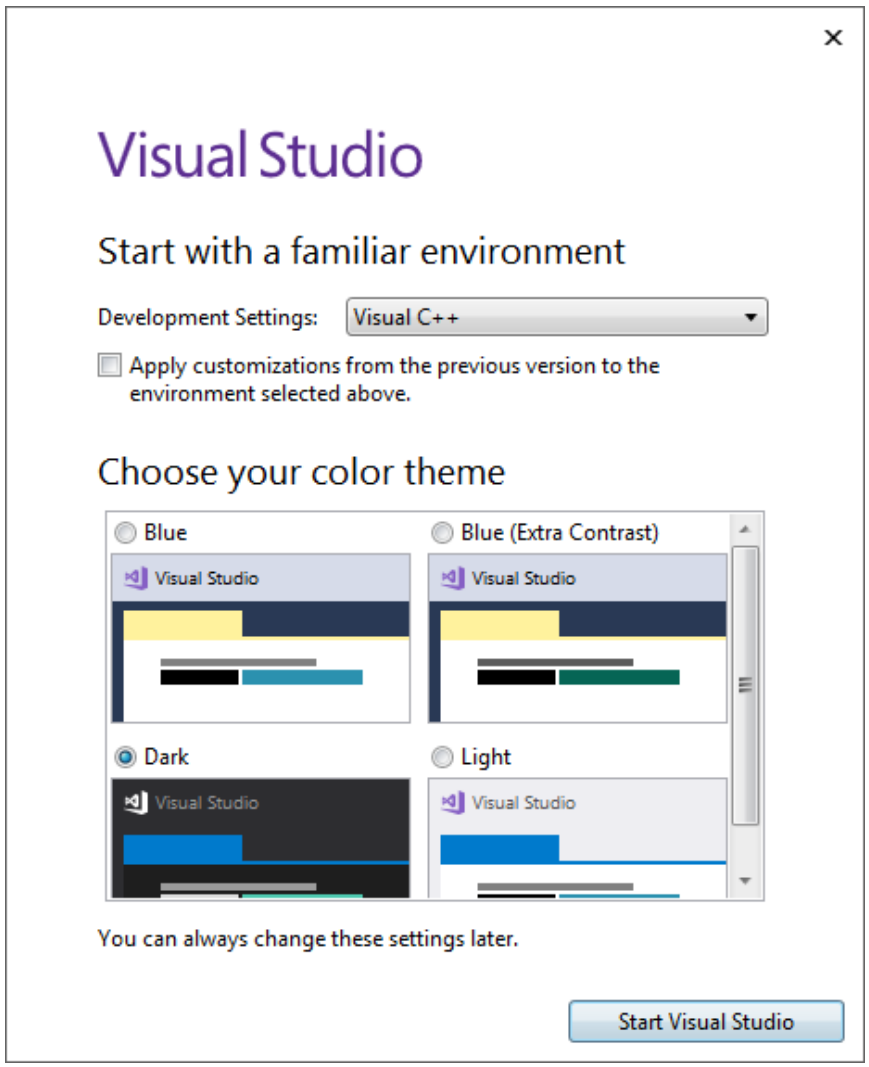


Figure 9. Select Visual C++ for the Development Settings.

Select Visual C++ from the Development Settings drop down list. This will set the common keyboard keys for building and other common settings used for C++ development. Note that both the color theme and **Development Settings** can be changed at any time later under **Tools - Options**.

You do not need to check **Apply customizations from the previous version to the environment selected above** unless you have installed another version of Visual Studio already and want any custom settings to be applied to this installation.

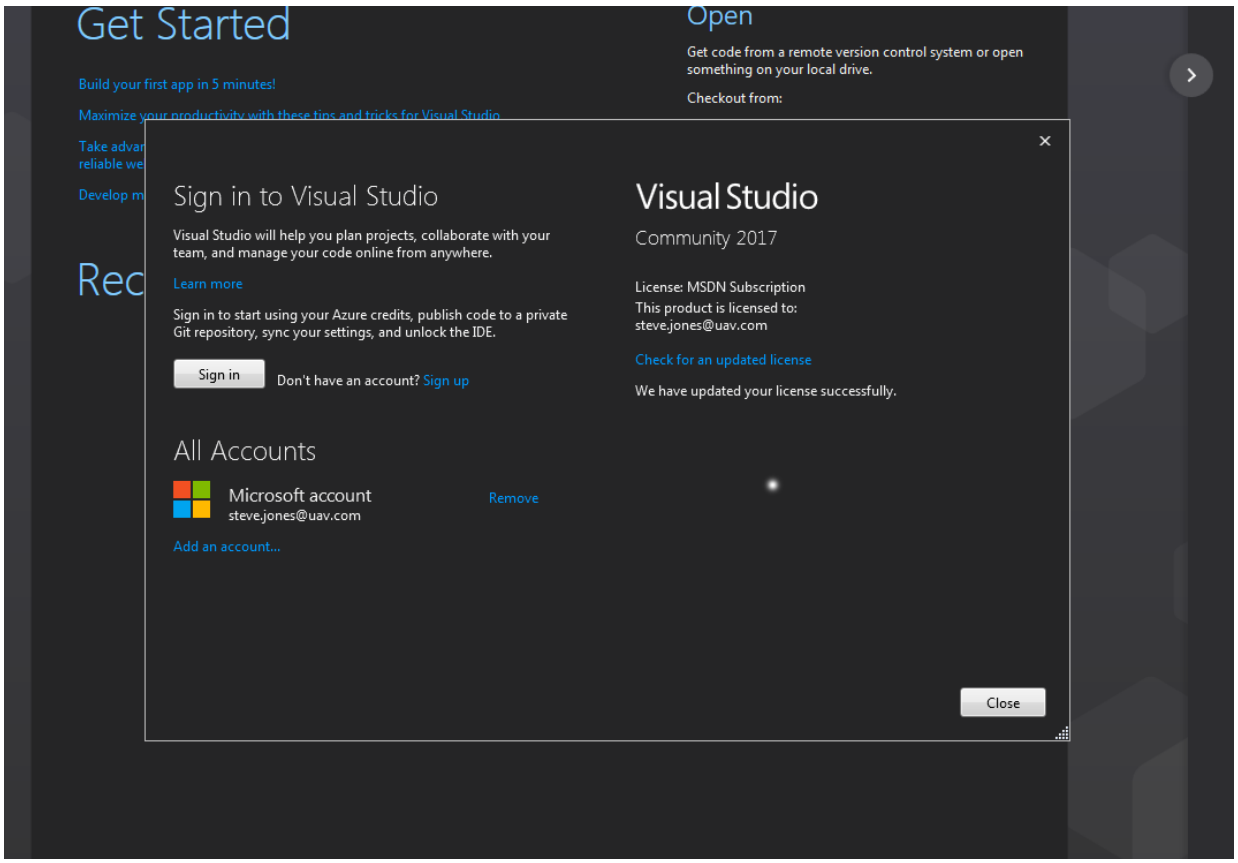


Figure 10. Sign in dialog.

When you launch the app for the first time, you may get this sign in window shown in Figure 10. It may also request a license to continue using Visual Studio Community 2017. The license is free for the Community version so all you need to do is create a login to Microsoft and log in when prompted after starting Visual Studio. If you do not have a Microsoft account, you will need to create one to use Visual Studio Community 2017. After logging in you will not need to log in again to use Visual Studio.

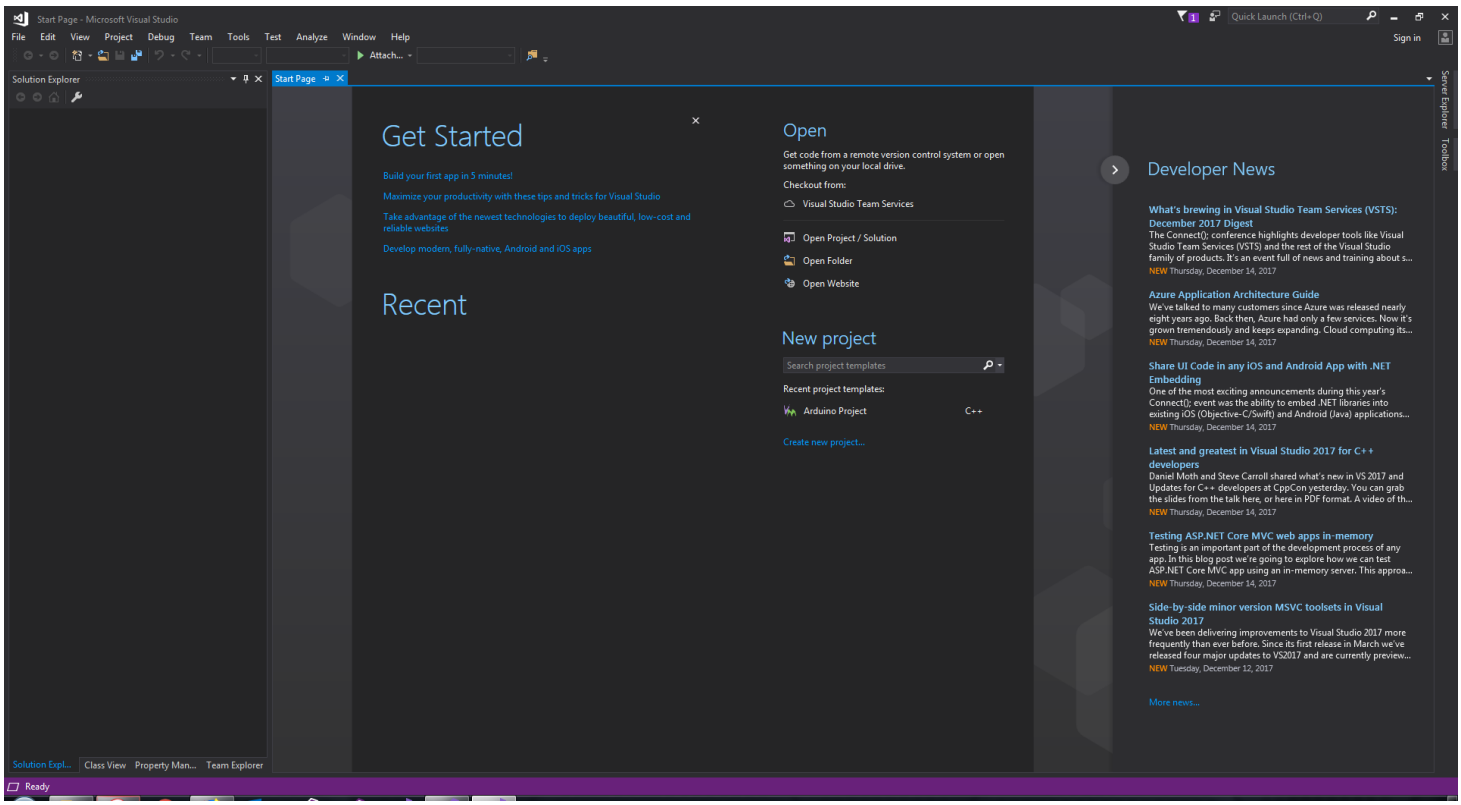


Figure 11. Visual Studio Community 2017 main screen.

GETTING STARTED - CREATING YOUR FIRST C++ PROJECT

ADDING A NEW PROJECT

After launching Visual Studio 2017 Community Edition we will create a new Project. This project, along with all projects for the *C++ Module I* course, will be Windows Console Application types. This type of application does not have the typical Windows UI that we are used to seeing on Windows applications. There are a couple of ways to do this. From the menu, go to **File – New – Project**. Or you can select the "**Create new project…**" link from the Start Page. By default, you will get the Start Page when Visual Studio starts up.

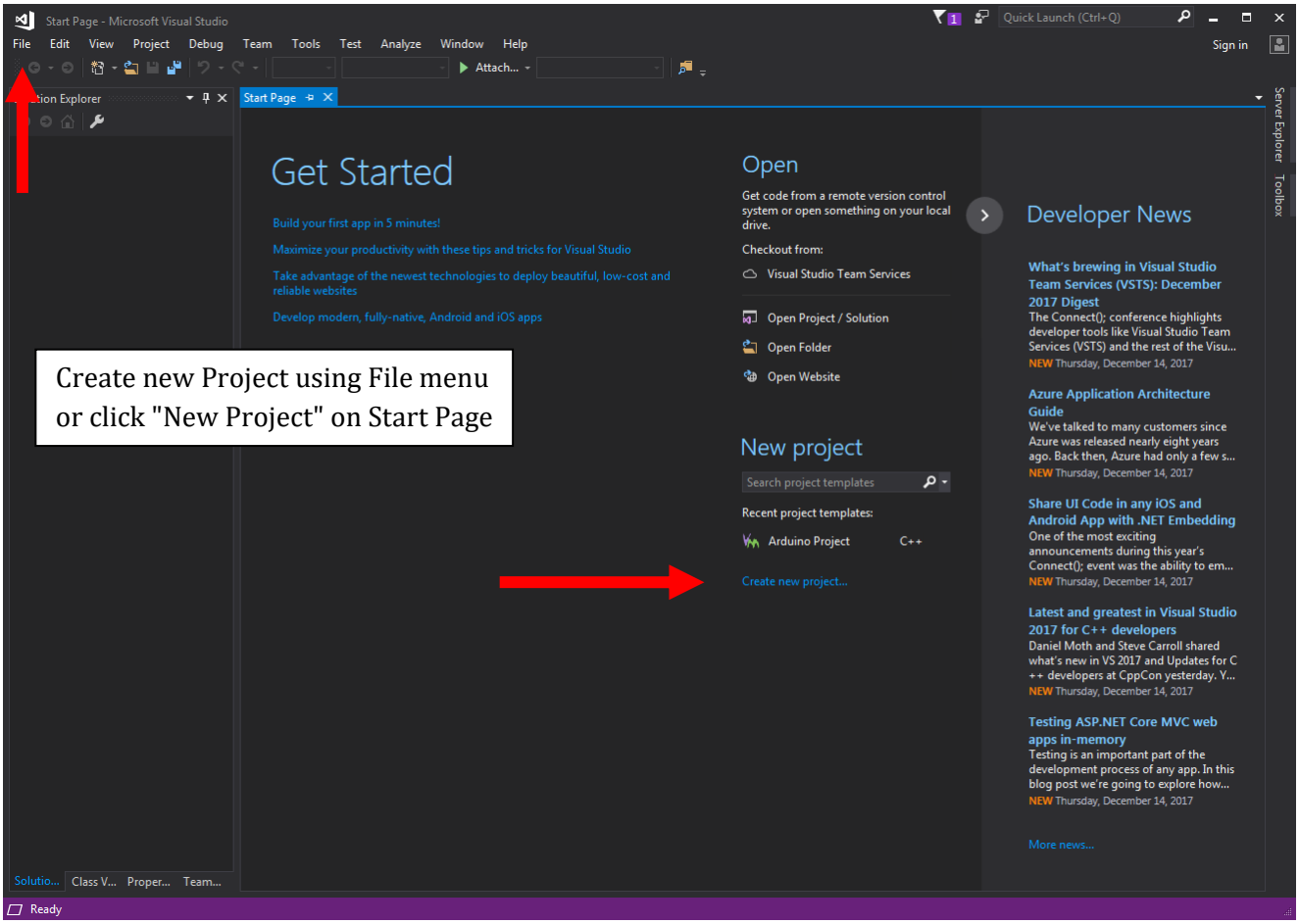


Figure 12. Starting a new project using menus

The following dialog should appear. Under **Installed** in the left pane, click **Windows Desktop** to highlight the available projects. The projects will show up on the right. Click on **Windows Desktop Wizard**. We will be able to choose other options including the application type.

Enter a name for the project in the **Name** field. Enter or browse to the location where you want to save it on your hard drive. We are not going to create a separate directory for the Solution, so uncheck that box. Creating a new Solution directory is most beneficial when you have multiple Projects in the Solution. The Solution becomes the parent directory with all Projects under it in subdirectories. This adds more complexity than what we need. Un-checking this box will create the Solution in the same directory as the Project. Click **OK**.

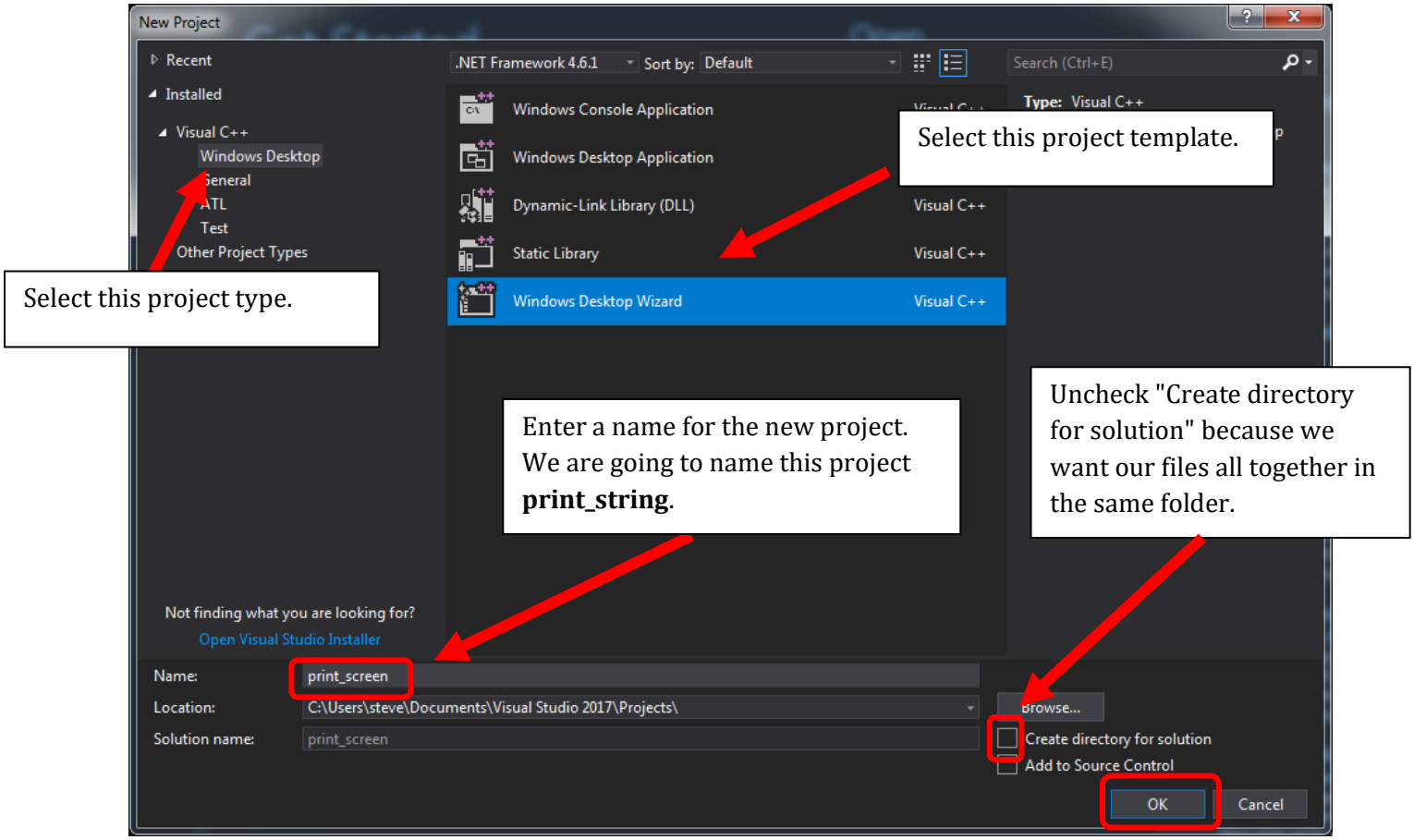


Figure 13. Creating the project window

A Windows Desktop Project options dialog box now appears as shown in Figure 14. We are going to make a few changes to the project, including picking the type of application we want. Drop down Application type to select **Console Application (.exe**). Select **Empty Project.** When you do that you will get a Solution with a Project named **print_string** but it will contain no source files. We will add them manually in a minute. Next, uncheck **Precompiled Header** and **Security Development Lifecycle (SDL) checks** as shown in Figure 14 as we will not need those options in our application at this time.

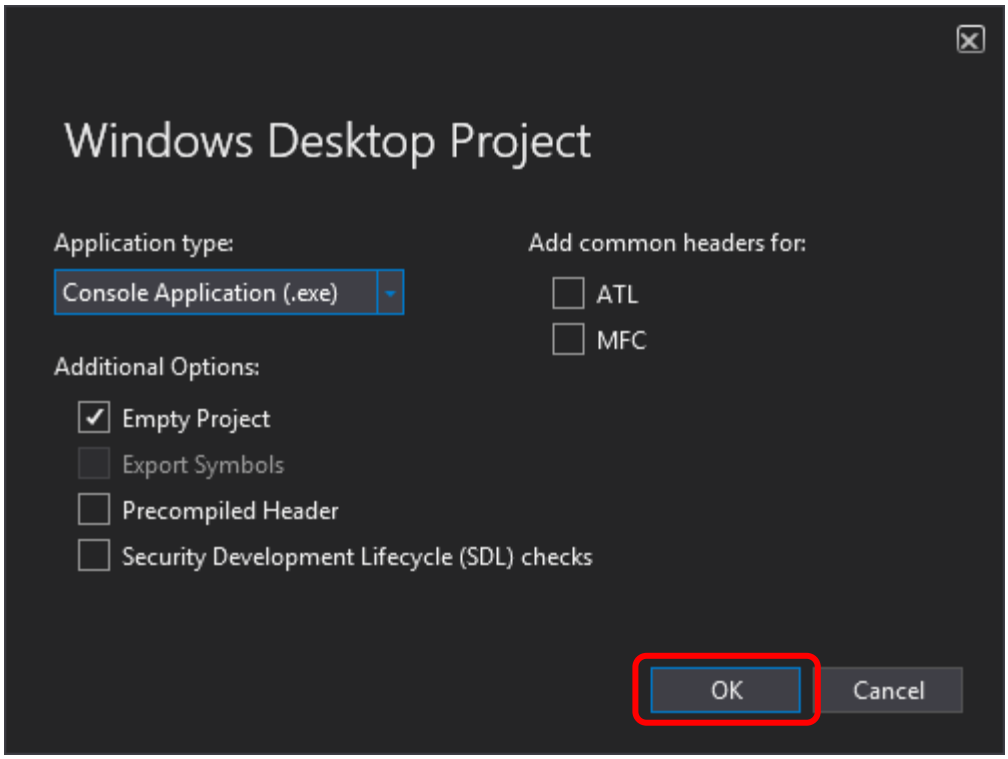


Figure 14. Windows Desktop Project options dialog.

At this point, we have successfully created a C++ project. The next step is to add some source code files.

As a side note, you might have noticed in the project selection dialog shown in Figure 13 that there was a project type called ***Windows Console Application***. We did not choose that type because we would not have been given any options to allow us to make an empty project.

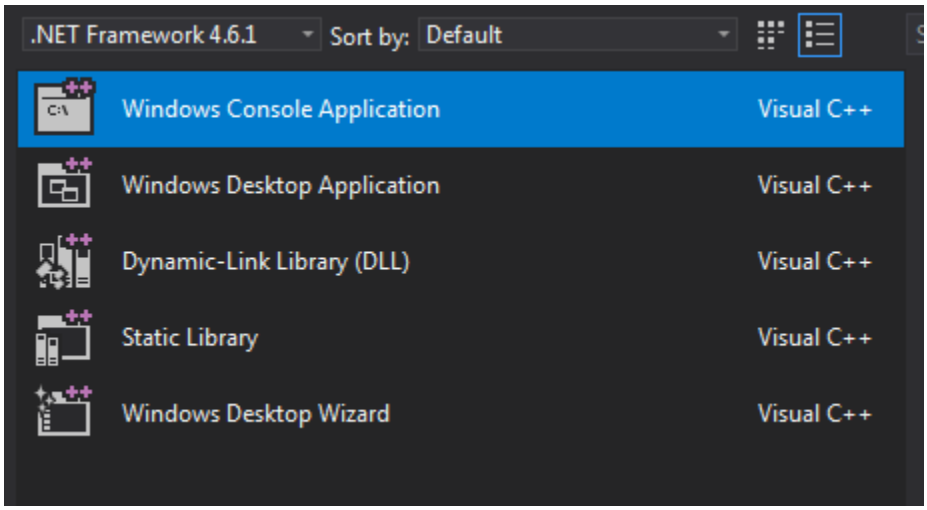


Figure 15. Alternative Console project selection.

Click **OK** to create the project.

Let's take a look at what we have so far. In Figure 16 the window on the left side is called the Solution Explorer. It shows the layout of the files for the Project. It is important to note that the structure shown here *does not necessarily match* the physical directory structure as found on your hard drive. There are currently no files in the Project called **print_string**. That was intentional, in this case, since we chose an **Empty project**. You will not necessary always want to create empty projects but in this case, we chose to do it so that you can learn how to manually add you own files to the project.

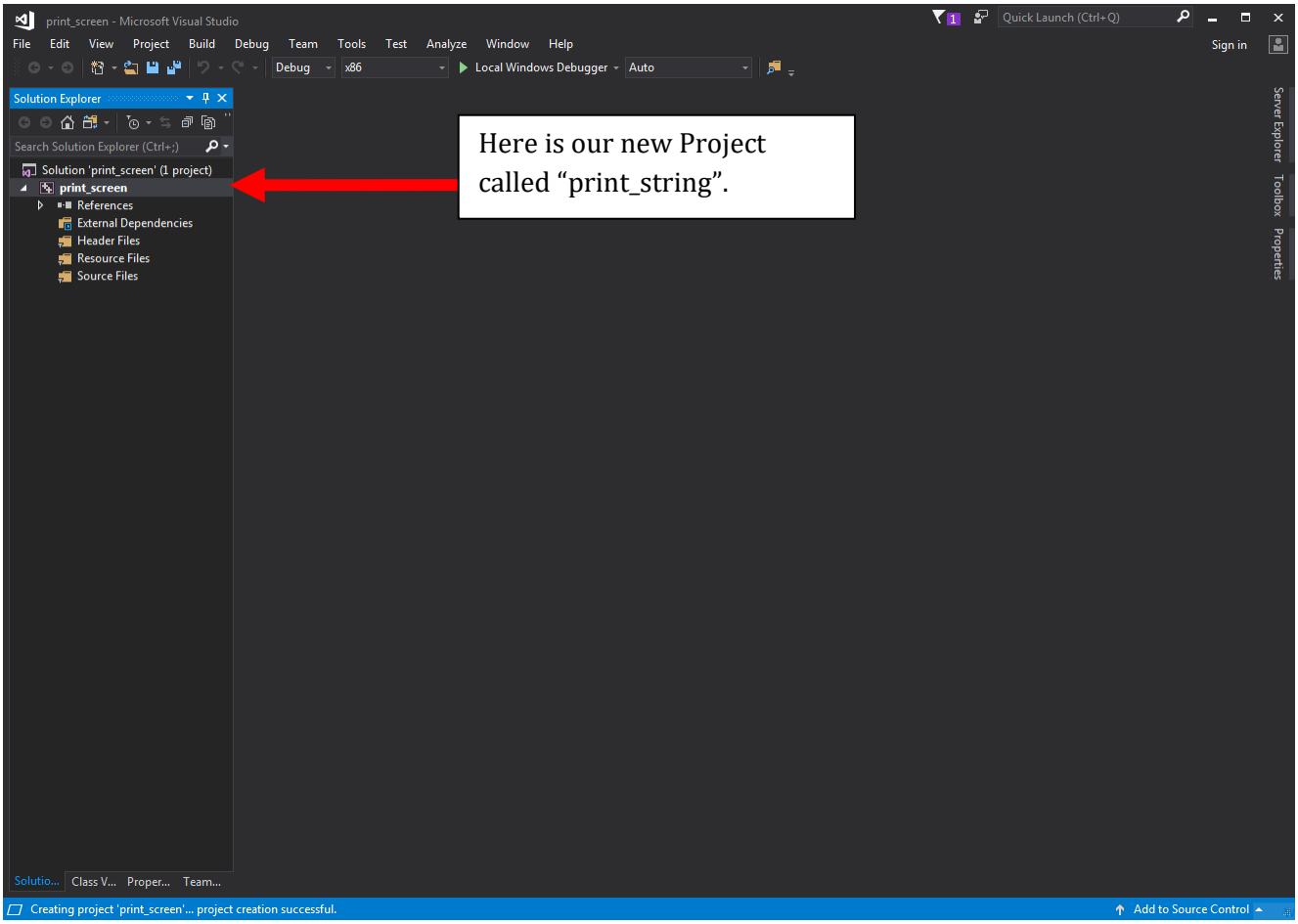


Figure 16. Visual Studio 2017 Community Edition - Highlighting the Solution Explorer

ADDING A .CPP FILE TO THE PROJECT

Now let's add some code. We will start by adding a .CPP file to the project. From the top menu select **Project – Add new item**.

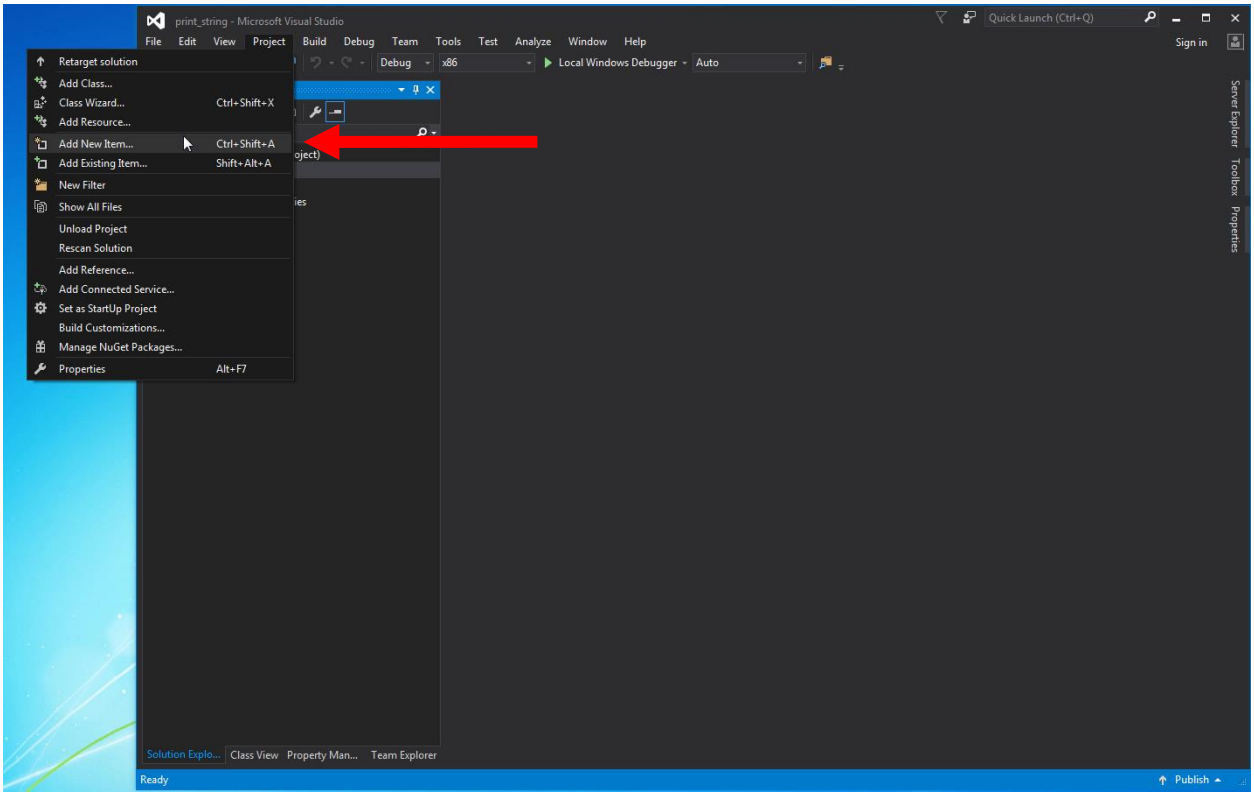


Figure 17. Using the menus to create a new Item that will be added to the Project.

You can also right mouse click on the Project name in the Solution Explore and select **Add – New Item** from the context menu. Either way it is important that you do not select menu **File – New – File** otherwise the newly created file will not be part of the Project, therefore it will not get compiled into your final executable.

You will be presented with a dialog as shown in Figure 18.

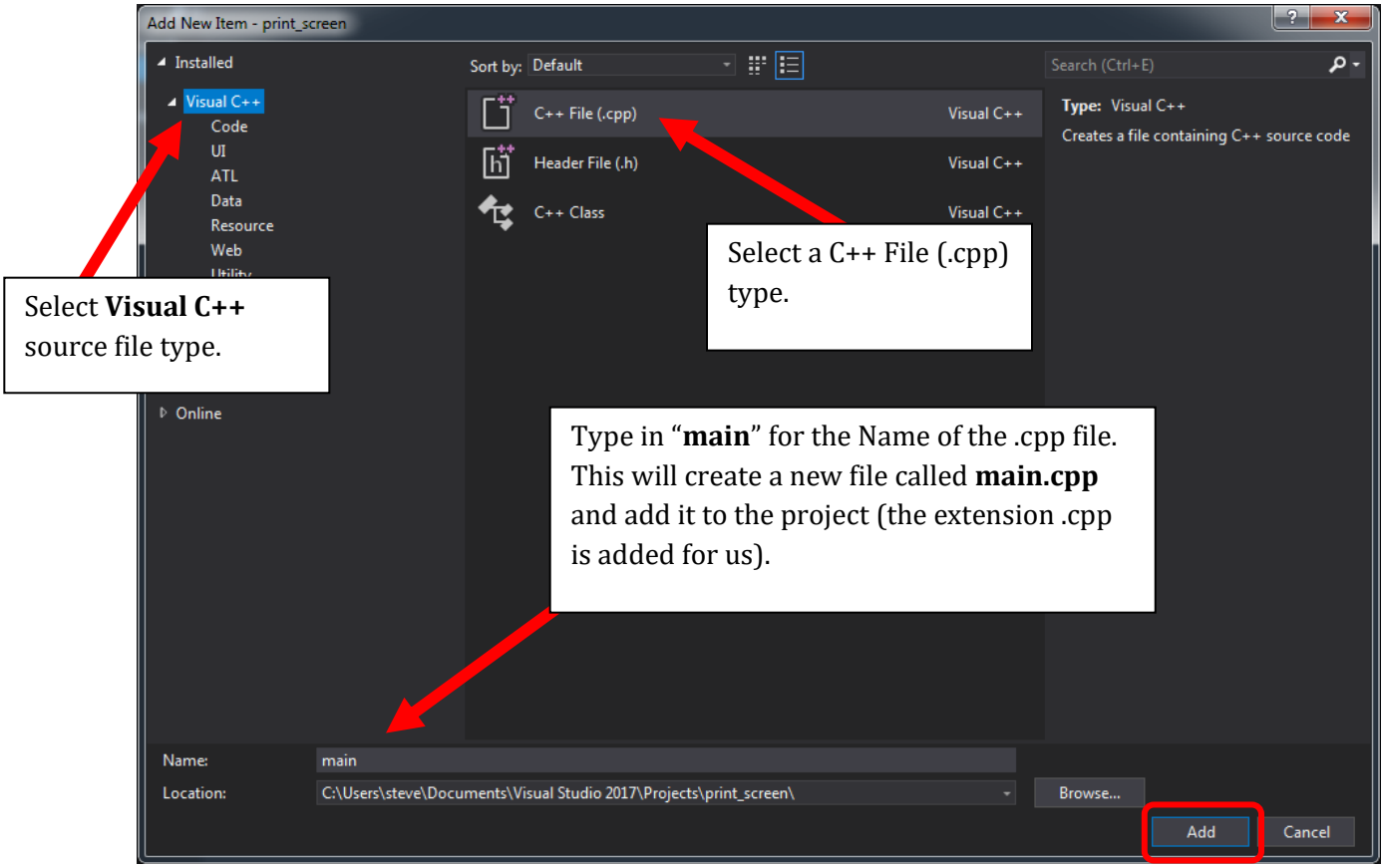
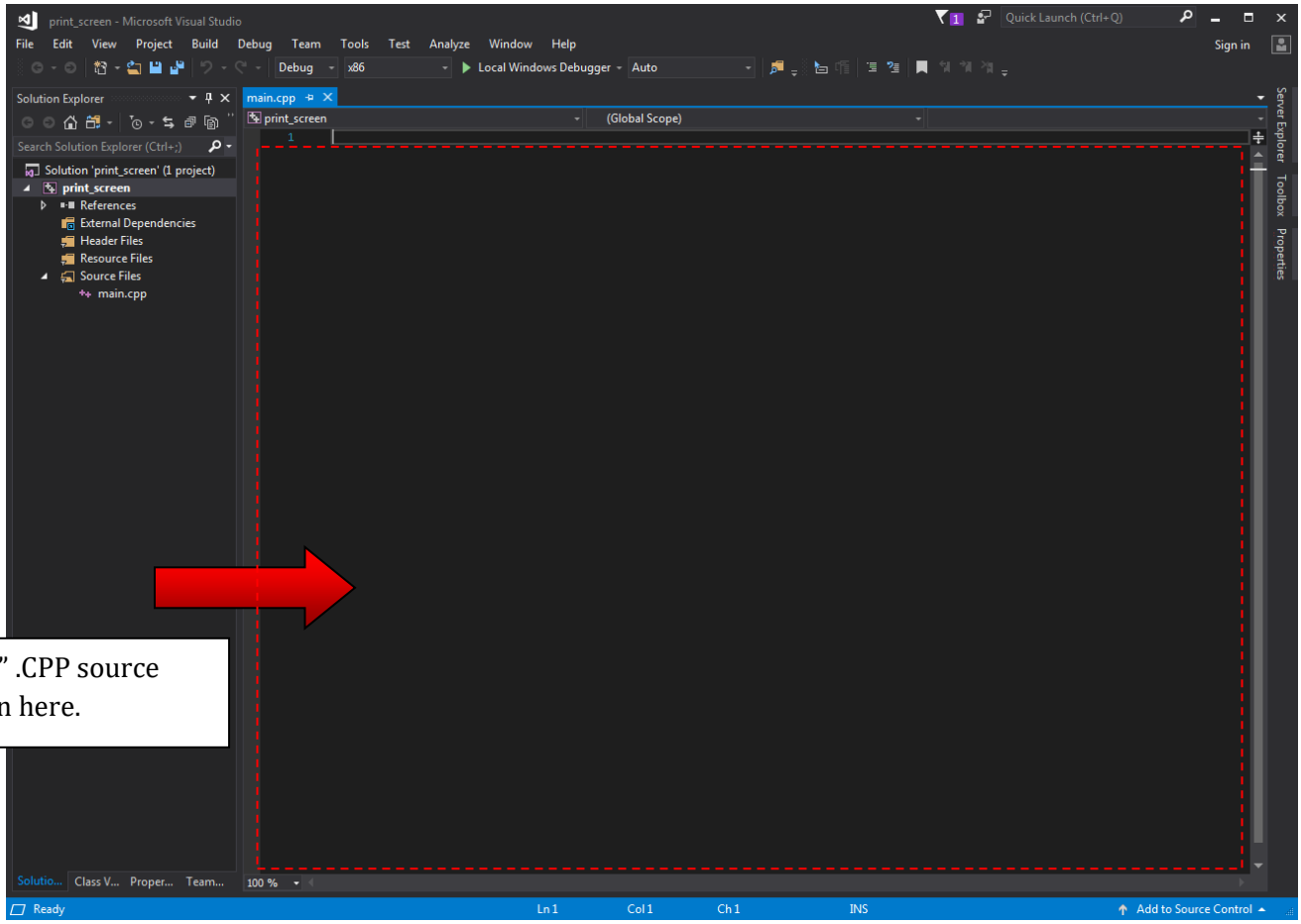


Figure 18. Adding a new .CPP file

Click on **Visual C++** in the left pane box. In the middle pane select a **C++ file (.cpp)**. Give the .CPP file a name and leave the Location as it shows. For console applications it is typical to call the source file "**main**" in a single file console application. Click **Add** to create the new file. You should see a blank .CPP source file showing in the main window of the IDE (Integrated Development Environment) as well as the .CPP file under the source folder in the Solution Explorer.



"main.cpp" .CPP source file. Edit in here.

Figure 19. Visual Studio 2017 Community Edition window showing the newly added file.

WRITING THE CODE

In the blank .CPP file type the following C++ code (outlined by the red dashed line in the code box below), exactly as it is, into your .CPP file. You may have trouble copy-pasting the code from this file as it may change the double quotes to a different character that will cause the code to fail to compile. It is good practice anyway to just type in the code.

```
//====================================================================
// print_string.cpp
//====================================================================

#include <iostream>
#include <string>

int main()
{
	std::string firstName = "";

	std::cout << "Enter your first name and press Enter: ";

	std::cin >> firstName;

	std::cout << std::endl;

	std::cout << "Hello, " << firstName << std::endl << std::endl;
}
```

We will not discuss the source code and what it means here because this has already been done in the *C++ Programming for Games Module I* course textbook. Refer to Section 1.2 **The "Print String" program Explained** to have the specifics of the source code explained.

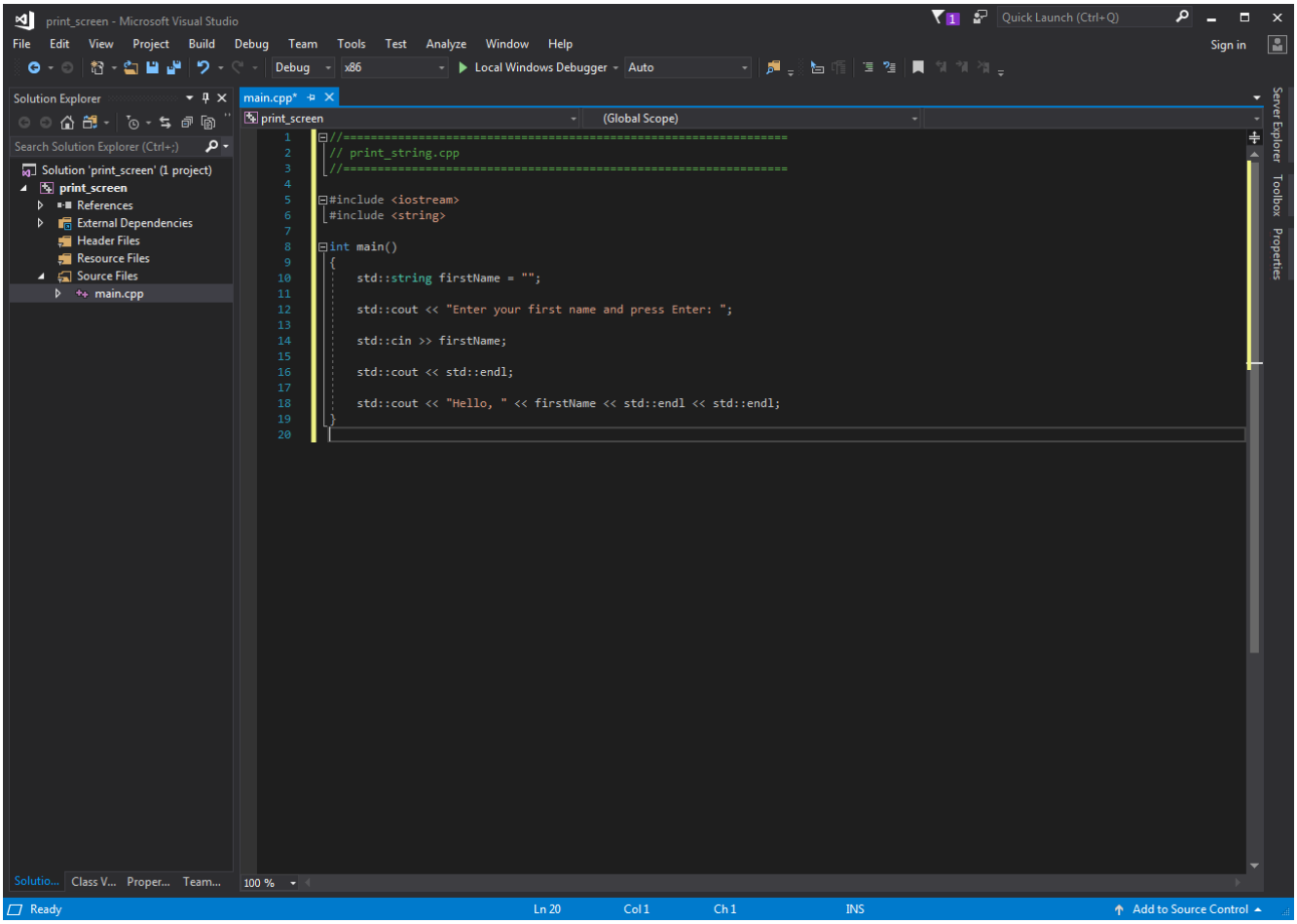


Figure 20. Showing the final code edited in the source file "main.cpp".

BUILDING THE PROGRAM

After the C++ source code has been entered, it must be translated into a language that the computer understands. There are two steps in that translation process.

1. Compilation
2. Linking

Solutions in Visual Studio Community 2017, as in previous editions, contain one or more Projects. Throughout the *C++ Programming for Games Modules I and II*, only a single project is ever discussed and used in a Solution.

These steps are explained in more detail in the course. Now let's build our project.

From the main menu select **Build – Rebuild Solution** (see Figure 21) or Press **F7** which is the short cut key. Rebuild Solution vs Build Solution will perform a Clean Solution first before building. A Clean deletes any existing files that were created during a prior build before proceeding with the build. It does not erase source files, of course.

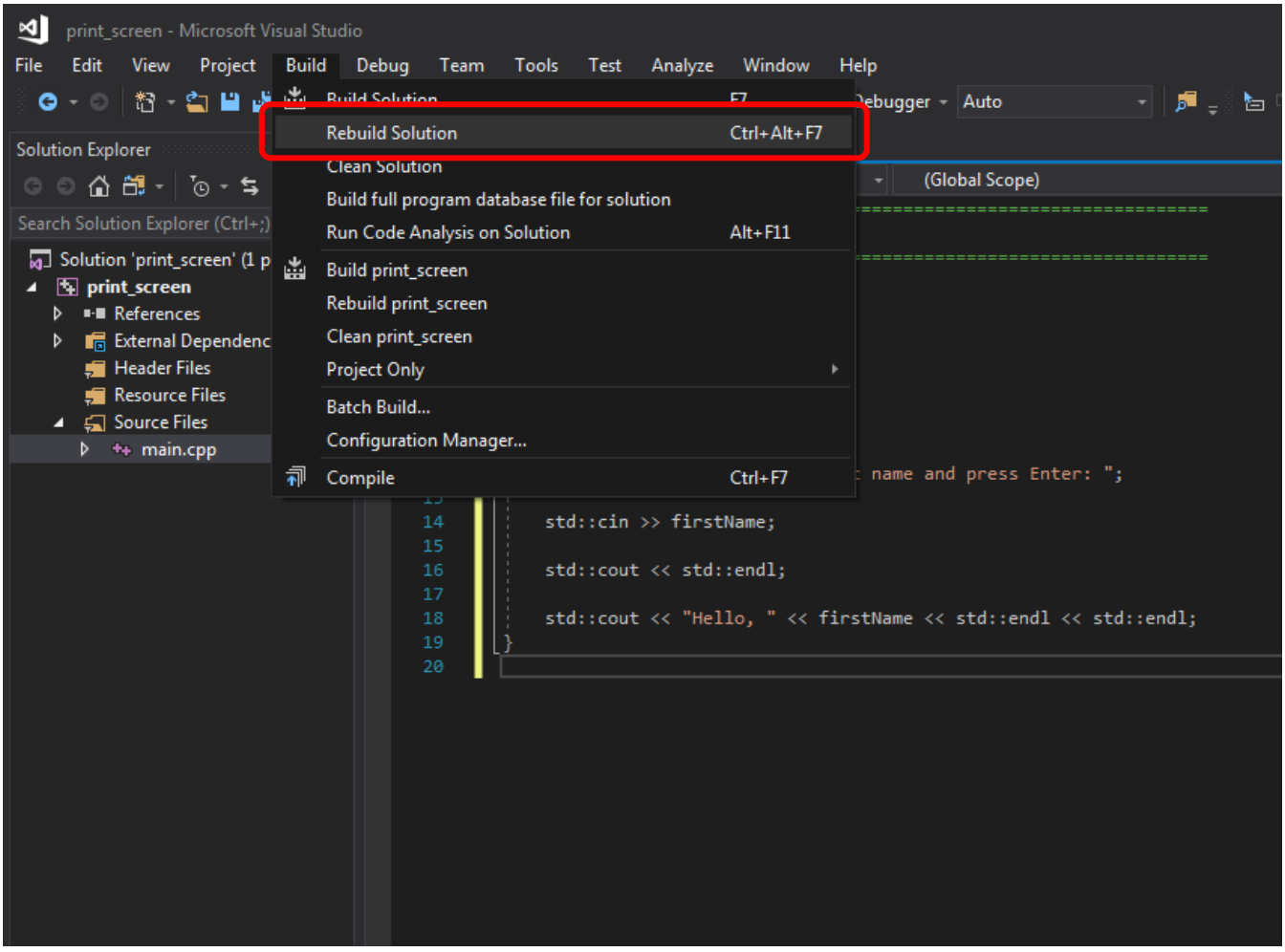


Figure 21. Build the Solution

During compilation and linking, you will see status activity in the Output window (Figure 22). If you do not see the Output window, or any other window for that matter, you can always go to the View menu and show and/or hide them. You will see all of the compiler complaints and other information in this window when compiling and linking projects.

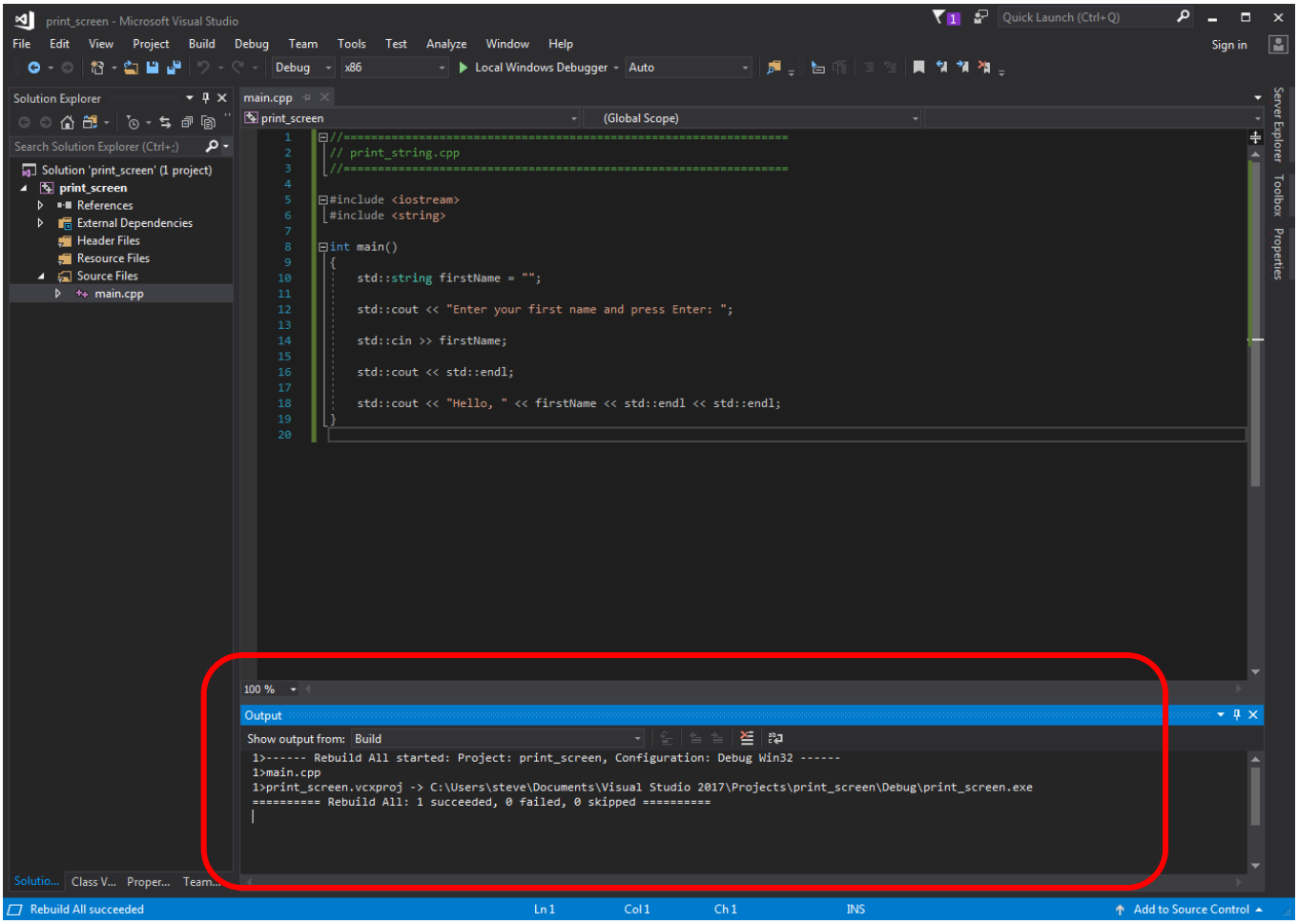


Figure 22. The Output Window after compiling and linking

Look at the last two lines in Figure 22. This compilation had 0 errors and 0 warnings which is the best we can ever hope to achieve. That means the code was written with legal C++ code and therefore the program has linked successfully. The textbook goes into more detail on what exactly it means to compile and link a program.

RUNNING THE PROGRAM

At this point an executable has been created for our program. We can now run the executable and make sure it works. From the Debug menu, select **Start without Debugging** or use CTRL+F5 keyboard keys. This will launch the console application from within the Visual C++ IDE.

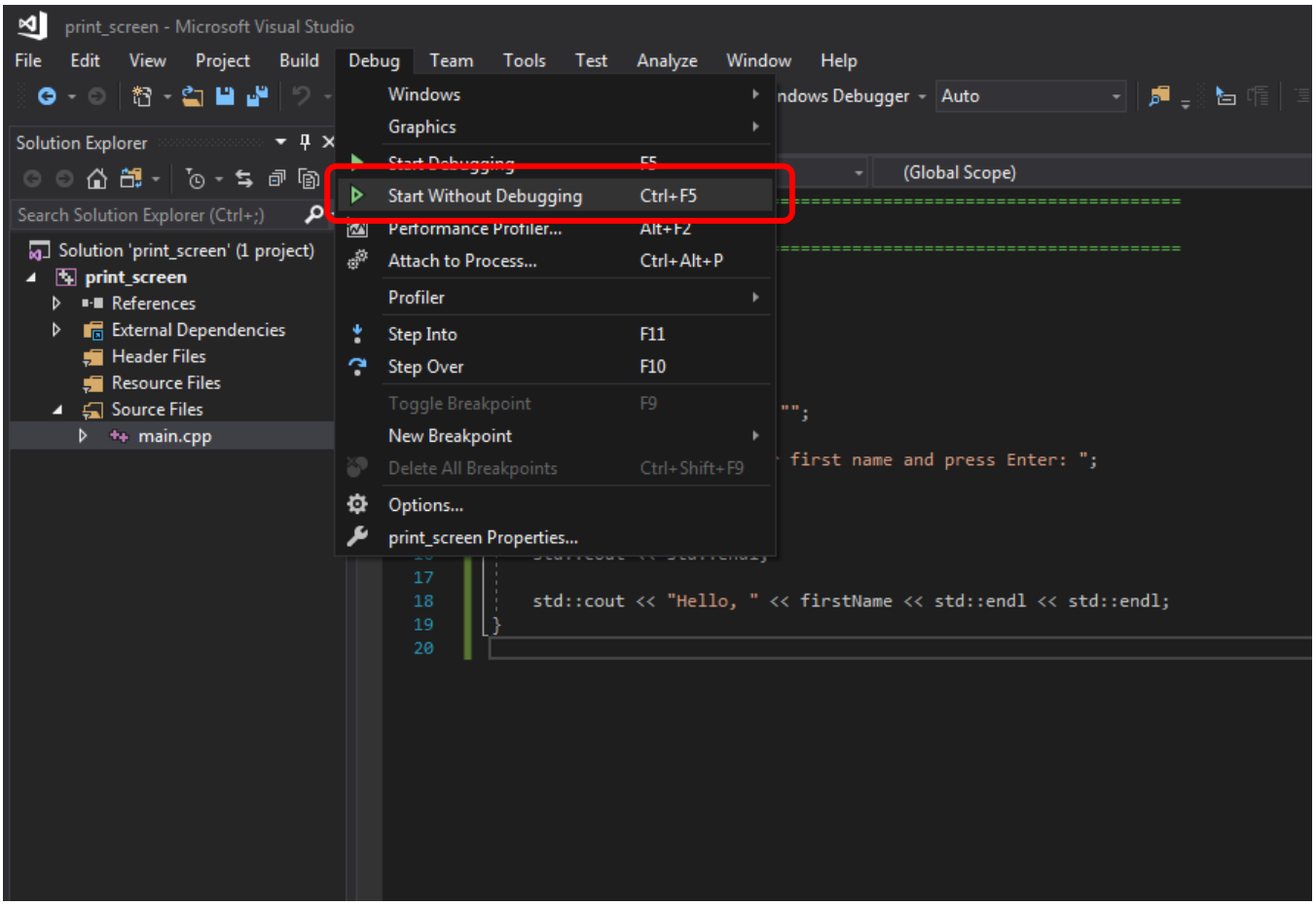


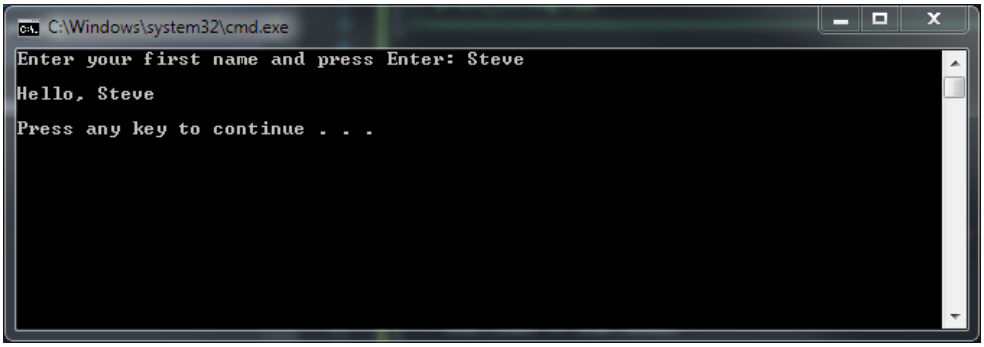
Figure 23. Running your program!

You should first see the following output:

```
Enter your first name and press Enter:
```

In this example the text “Steve” was typed and the Enter key was pressed. The program then displays the following:

Program 1: Output



Congratulations!! You have just created and executed your first C++ program from scratch! Note that the textbook for *C++ Programming for Games Module I* course goes through each line in the source code explaining exactly what is going on in the code.

CONCLUSION

We began by installing the Visual Studio 2017 Community Edition software, then we moved on to writing our first simple C++ program. Finally, we built and then ran the program, resulting in some output in a console window. The basic steps to creating a console application written in C++ have been demonstrated. With this knowledge you can develop other types as well, such as Win32 Windows applications. The line by line detailed explanation of the source code we used to build the executable is found in the textbook for the first course on C++ at the Game Institute, *C++ Programming for Games Module I*. Have fun coding!