Tutorial / Exercise – Introduction to C++

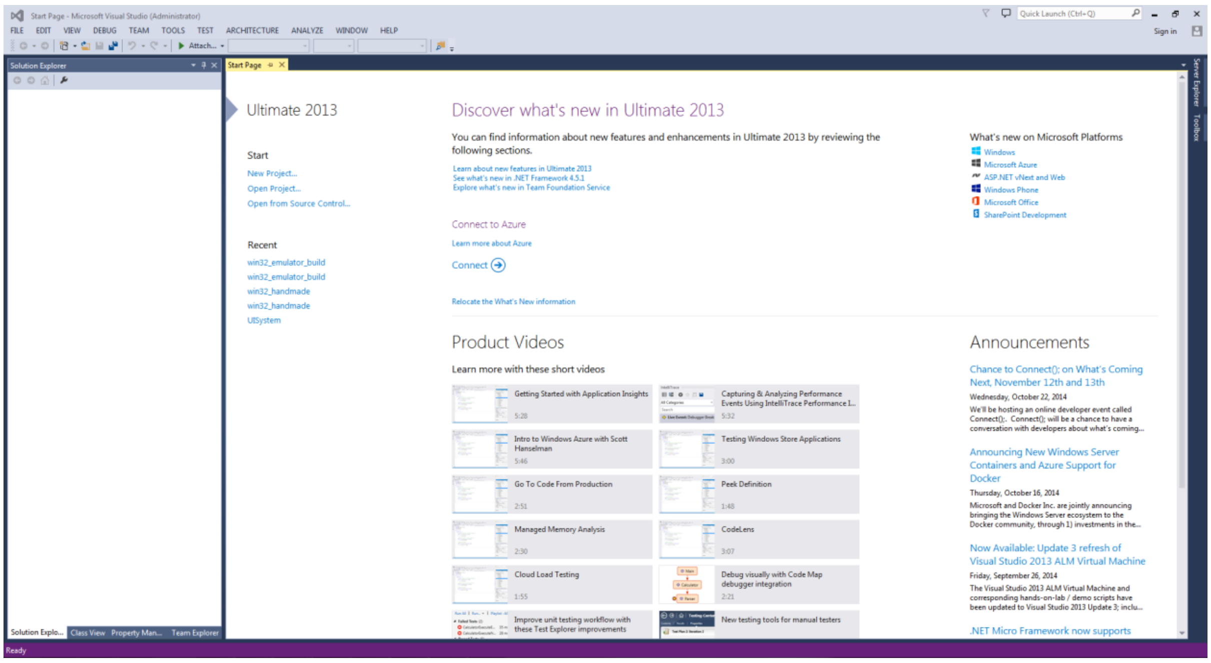
Today, we are going to make our very first C++ program!

Creating a project in Visual Studio:

In order to compile a C++ program, we need a compiler!

We're going to be using Visual Studio for almost all of our programming this year. It is an IDE (Integrated Development Environment) developed by Microsoft and is a standard throughout most of the game industry.

You should have Visual Studio 2013 installed on your computer already. Open it.


Start Page - Microsoft Visual Studio (Administrator)
FILE EDIT VIEW DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Attach...
Quick Launch (Ctrl+Q)
Sign in
Solution Explorer
Start Page
Server Explorer
Toolbox
Solution Explo... Class View Property Man... Team Explorer
Ready

Ultimate 2013

Start
New Project...
Open Project...
Open from Source Control...

Recent
win32_emulator_build
win32_emulator_build
win32_handmade
win32_handmade
UISystem

Discover what's new in Ultimate 2013

You can find information about new features and enhancements in Ultimate 2013 by reviewing the following sections.

Learn about new features in Ultimate 2013
See what's new in .NET Framework 4.5.1
Explore what's new in Team Foundation Service

Connect to Azure
Learn more about Azure
Connect

Relocate the What's New information

What's new on Microsoft Platforms
Windows
Microsoft Azure
ASP.NET vNext and Web
Windows Phone
Microsoft Office
SharePoint Development

Product Videos

Learn more with these short videos

Getting Started with Application Insights 5:28
Capturing & Analyzing Performance Events Using IntelliTrace Performance I... 5:32
Intro to Windows Azure with Scott Hanselman 5:46
Testing Windows Store Applications 3:00
Go To Code From Production 2:51
Peek Definition 1:48
Managed Memory Analysis 2:30
CodeLens 3:07
Cloud Load Testing 1:55
Debug visually with Code Map debugger integration 2:21
Improve unit testing workflow with these Test Explorer improvements
New testing tools for manual testers

Announcements

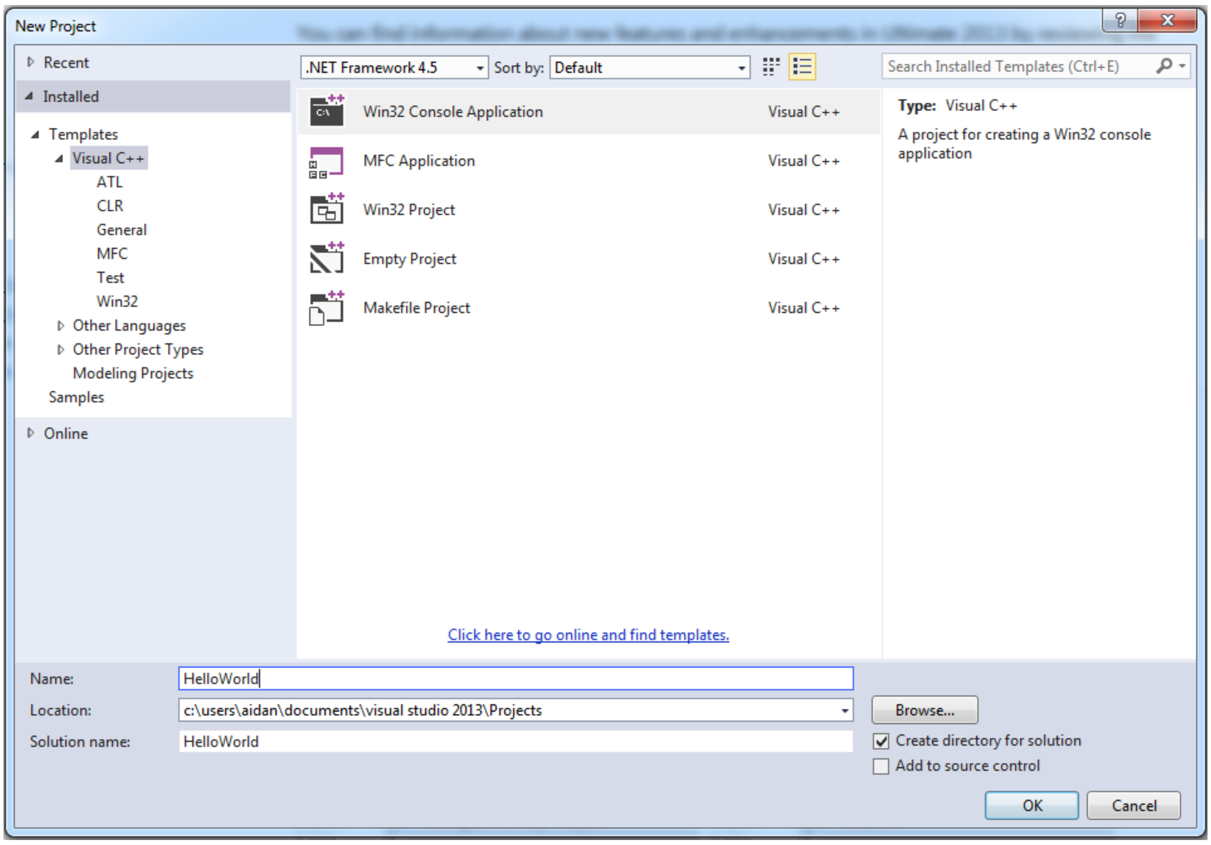
Chance to Connect(); on What's Coming Next, November 12th and 13th
Wednesday, October 22, 2014
We'll be hosting an online developer event called Connect();. Connect(); will be a chance to have a conversation with developers about what's coming...

Announcing New Windows Server Containers and Azure Support for Docker
Thursday, October 16, 2014
Microsoft and Docker Inc. are jointly announcing bringing the Windows Server ecosystem to the Docker community, through 1) investments in the...

Now Available: Update 3 refresh of Visual Studio 2013 ALM Virtual Machine
Friday, September 26, 2014
The Visual Studio 2013 ALM Virtual Machine and corresponding hands-on-lab / demo scripts have been updated to Visual Studio 2013 Update 3; inclu...

.NET Micro Framework now supports

You should see a screen that looks something like this. On the left under the start section, you should see a button that says "New Project…" Click it.



New Project

Recent
Installed
Templates
Visual C++
ATL
CLR
General
MFC
Test
Win32
Other Languages
Other Project Types
Modeling Projects
Samples
Online

.NET Framework 4.5 Sort by: Default
Search Installed Templates (Ctrl+E)

Win32 Console Application	Visual C++
MFC Application	Visual C++
Win32 Project	Visual C++
Empty Project	Visual C++
Makefile Project	Visual C++

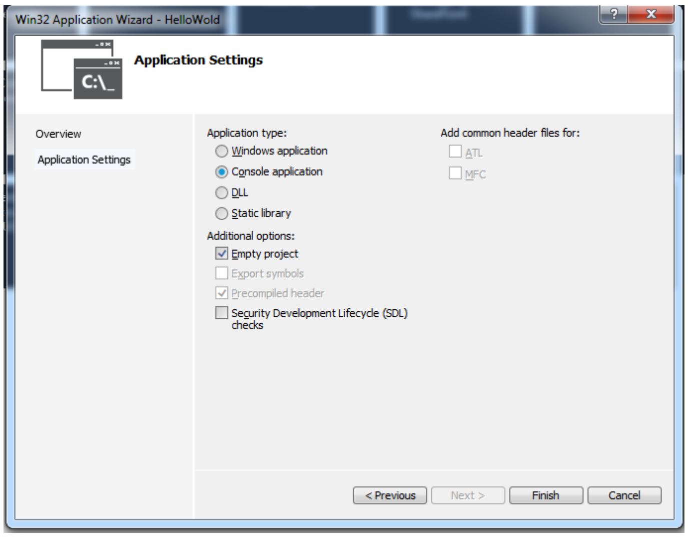
Type: Visual C++
A project for creating a Win32 console application

Click here to go online and find templates.

Name:	HelloWorld
Location:	c:\users\aidan\documents\visual studio 2013\Projects
Solution name:	HelloWorld

Browse...
[✓] Create directory for solution
[ ] Add to source control
OK Cancel

This next window is asking what kind of project you would like to make. For our first project, we're going to make a Win32 Console Application. Once you've selected Console Application, change the Name text field from ConsoleApplication1 to HelloWorld. Then click OK.



Win32 Application Wizard - HelloWold

Application Settings

Overview
Application Settings

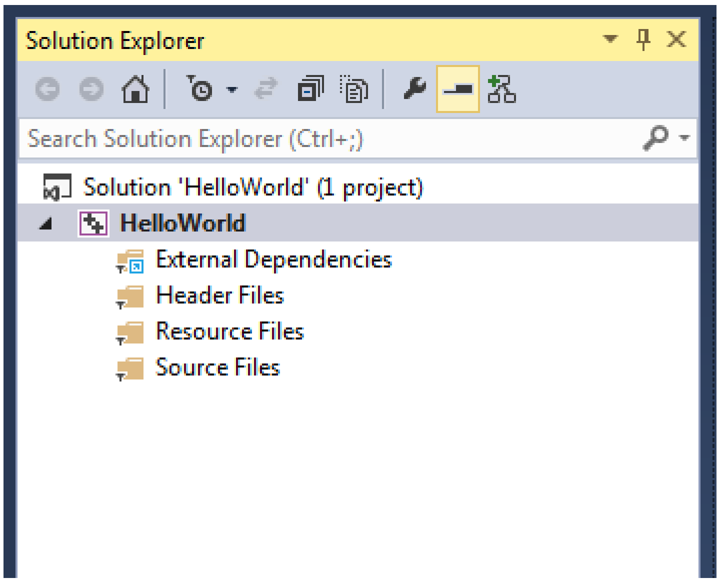
Application type:
( ) Windows application
(●) Console application
( ) DLL
( ) Static library

Additional options:
[✓] Empty project
[ ] Export symbols
[✓] Precompiled header
[ ] Security Development Lifecycle (SDL) checks

Add common header files for:
[ ] ATL
[ ] MFC

< Previous Next > Finish Cancel

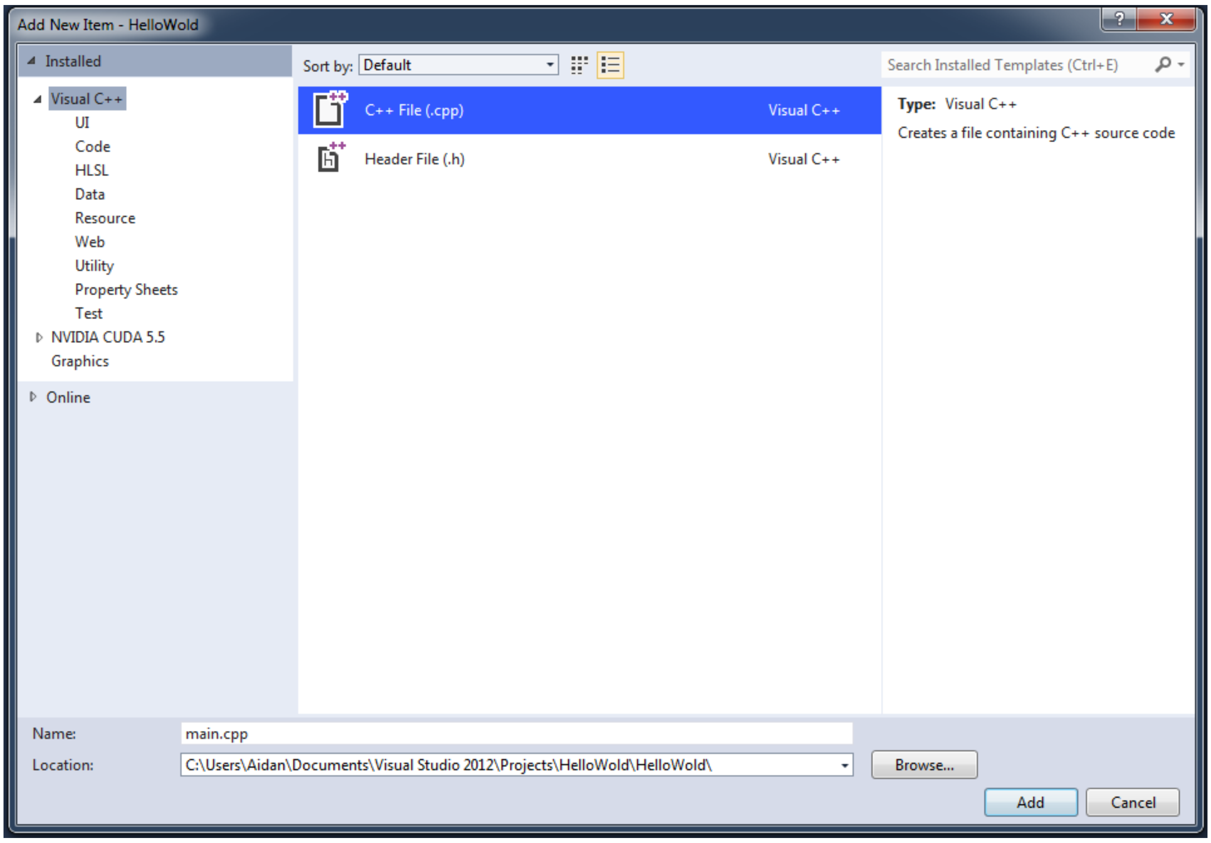
Next, the Application Settings window will open up. On the left column, click Application Settings. Now in the main window, tick Empty Project and un-tick Security Development Lifecycle checks. Then click Finish.



Solution 'HelloWorld' (1 project)
HelloWorld
External Dependencies
Header Files
Resource Files
Source Files

Your project should have now been created. Either on the left or the right of the main window, you should see a tall, thin window called the Solution Explorer. If you don't see this window, in the toolbar, click View->Solution Explorer. The Solution Explorer is where you will be managing your project. A solution in visual studio is just a container for projects. You can have as many projects in a solution as you want. A project is where you put the source code files you want Visual Studio to compile. We can see in the picture above, there is a solution called HelloWorld. Inside the solution is 1 project, also called HelloWorld. Inside the project there are 4 folders – External Dependencies, Header Files, Resource Files, and Source Files.

We want to add a new source code file. To do this, we need to right-click on Source Files and click Add->New Item.



Name: main.cpp
Location: C:\Users\Aidan\Documents\Visual Studio 2012\Projects\HelloWold\HelloWold\

The Add New Item window will now open. We want to make a C++ File, in the **Name** text field, replace **Source.cpp** with **main.cpp**.

Visual Studio will now create and open a new text file called **main.cpp** that we can now edit. You should now see main.cpp in the solution explorer, under the Source Files folder.

```
#include <iostream>

int main()
{
	std::cout << "Hello World!";
	return 0;
}
```

Now, type in the code from the lecture slides. Don't copy-paste the code, type it out yourself. Not only will typing it out help you remember the code, but the PDF formatting will sometimes break the code such that it won't compile despite looking correct – so watch out for that. In Visual Studio, press **F7** to compile your code. When you press **F7**, the output window should pop up down the bottom, which records standard logging information about the compile process. When complete, it will read `========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========`

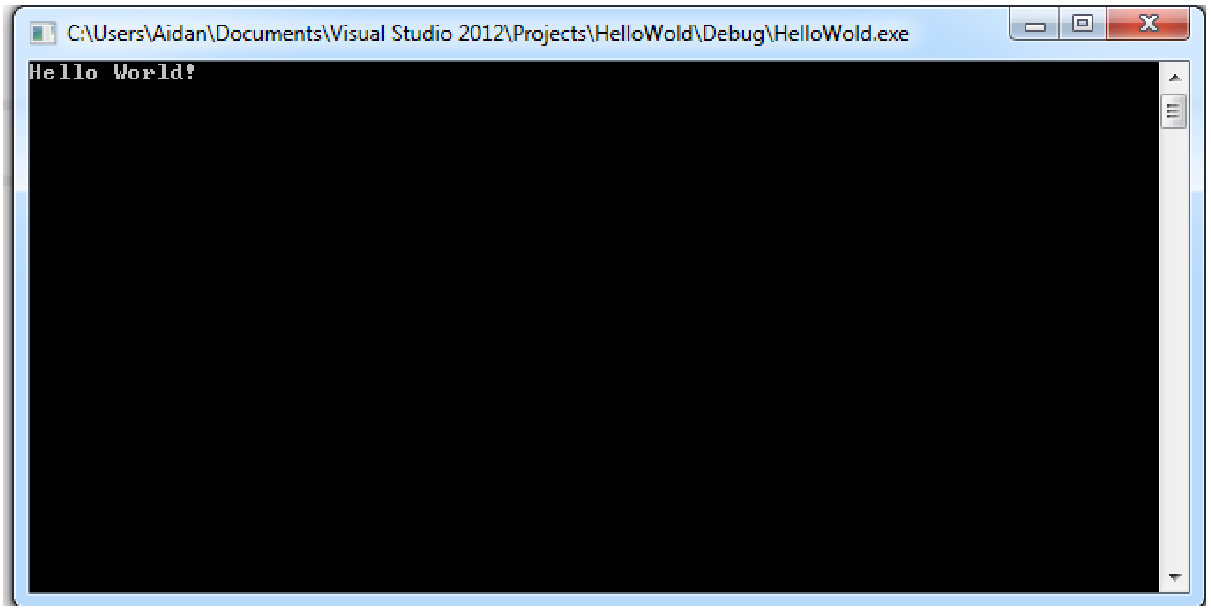
Now that we've built the project, hit **F5** to run it. You should see a window flash open and then immediately close. The reason it closed instantly is because we didn't tell it to wait. If we look back at our code, you'll notice that after printing our Hello World, the next line is the return, which tells our program to exit, which closes the window. We need a way to tell the program to wait so we can actually see what it did. We're going to add another line to our code.

```
#include <iostream>

int main()
{
	std::cout << "Hello World!";
	std::cin.get();
	return 0;
}
```

`std::cin.get();` is a standard library function. We've already seen a function declaration with `main`. A declaration is where we write our own function. Here, we are calling a function. When you call a function, all the code inside the function gets executed before the program moves on to the next line. The `std::cin.get()` function tells the program to wait until the person running the program (the user) has typed a character using the keyboard.

Now hit F5 again. F5 will check if you've made changes and ask you if you want it to rebuild before running. Click yes to build and run the program.


C:\Users\Aidan\Documents\Visual Studio 2012\Projects\HelloWold\Debug\HelloWold.exe

```
Hello World!
```

Success!

You can exit the program by hitting any key.

Exercises:

1. A C++ program is made of two things – data, and operations that do things with that data. There is only one piece of data in our program so far. What is it? What are we doing with it?

2. Add a second std::cout on the line after the first one. The text it prints out can be whatever you want. Build and run the program. Did it behave as you expected?

3. You will notice that the second line was added right after the first with no extra formatting. By default, cout won't add any formatting to the text you give it. You have to add it yourself. There's a problem if we want to have the new text on a new line, though. You aren't allowed to have strings you type in go over multiple lines. This will give you an error.

```
#include <iostream>

int main()
{
      std::cout << "Hello
                   World!";
      std::cin.get();
      return 0;
}
```

If you hit F7 to build this, you will see that the bottom line of the output now reads

```
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
```

This doesn't look very promising. The lines directly above will tell you what the error is.

```
1>main.cpp(5): error C2001: newline in constant
1>main.cpp(6): error C2146: syntax error : missing ';' before identifier 'World'
1>main.cpp(6): error C2065: 'World' : undeclared identifier
1>main.cpp(6): error C2143: syntax error : missing ';' before '!'
1>main.cpp(6): error C2001: newline in constant
1>main.cpp(7): error C2146: syntax error : missing ';' before identifier 'getchar'
```

We'll talk a lot more about syntax errors soon, but when looking at the errors that Visual Studio gives you, you should always look at the first one – one error tends to create a cascade of further errors. The first error here is "newline in constant". This is here because we're not allowed to split strings over multiple lines like this. How then, do we add new lines?
When typing a string in C++, there's a special value, called std::endl that represents a line break.

Try typing in the following code:

```
#include <iostream>

int main()
{
	std::cout << "Hello" << std::endl << "World!";
	std::cin.get();
	return 0;
}
```

A big thing to note here is that you can chain the stream-out operator to combine strings.

4. Now you know how to format text using std::cout, use the console to print out some ASCII art.