Getting Started with C and C++ in Visual Studio

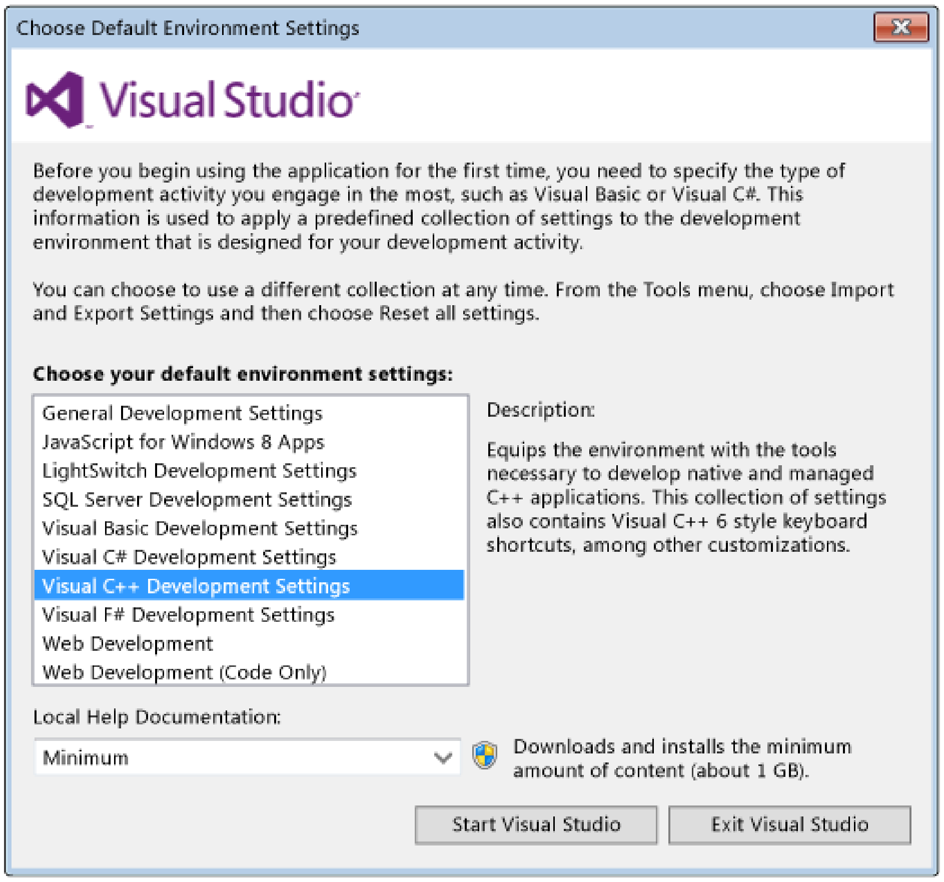
For Mac Users:
Virtualbox is a free hypervisor comparable to vmware created by Oracle. There is a version for OS-X and it will run Windows just fine. Be sure you download the Virtualbox Extension pack along with Virtualbox itself.
https://www.virtualbox.org/wiki/Downloads

You can download Windows and Visual Studio from MyASU and MyApp, or directly go to
https://webapp4.asu.edu/eacademy-sso/authn

By completing this walkthrough, you'll become familiar with many of the tools and dialog boxes that you can use when you develop apps with Visual Studio. You'll create a simple "Hello, World"-style app while you learn more about working in the integrated development environment (IDE).
This topic contains the following sections:

Configure the IDE

When you start Visual Studio for the first time, you must choose a settings combination that applies a set of pre-defined customizations to the IDE. Each combination has been designed to help you develop a certain kind of app.
Figure 1: Choose Default Environment Settings dialog box


Choose Default Environment Settings

Before you begin using the application for the first time, you need to specify the type of development activity you engage in the most, such as Visual Basic or Visual C#. This information is used to apply a predefined collection of settings to the development environment that is designed for your development activity.

You can choose to use a different collection at any time. From the Tools menu, choose Import and Export Settings and then choose Reset all settings.

Choose your default environment settings:

- General Development Settings
- JavaScript for Windows 8 Apps
- LightSwitch Development Settings
- SQL Server Development Settings
- Visual Basic Development Settings
- Visual C# Development Settings
- Visual C++ Development Settings
- Visual F# Development Settings
- Web Development
- Web Development (Code Only)

Description:

Equips the environment with the tools necessary to develop native and managed C++ applications. This collection of settings also contains Visual C++ 6 style keyboard shortcuts, among other customizations.

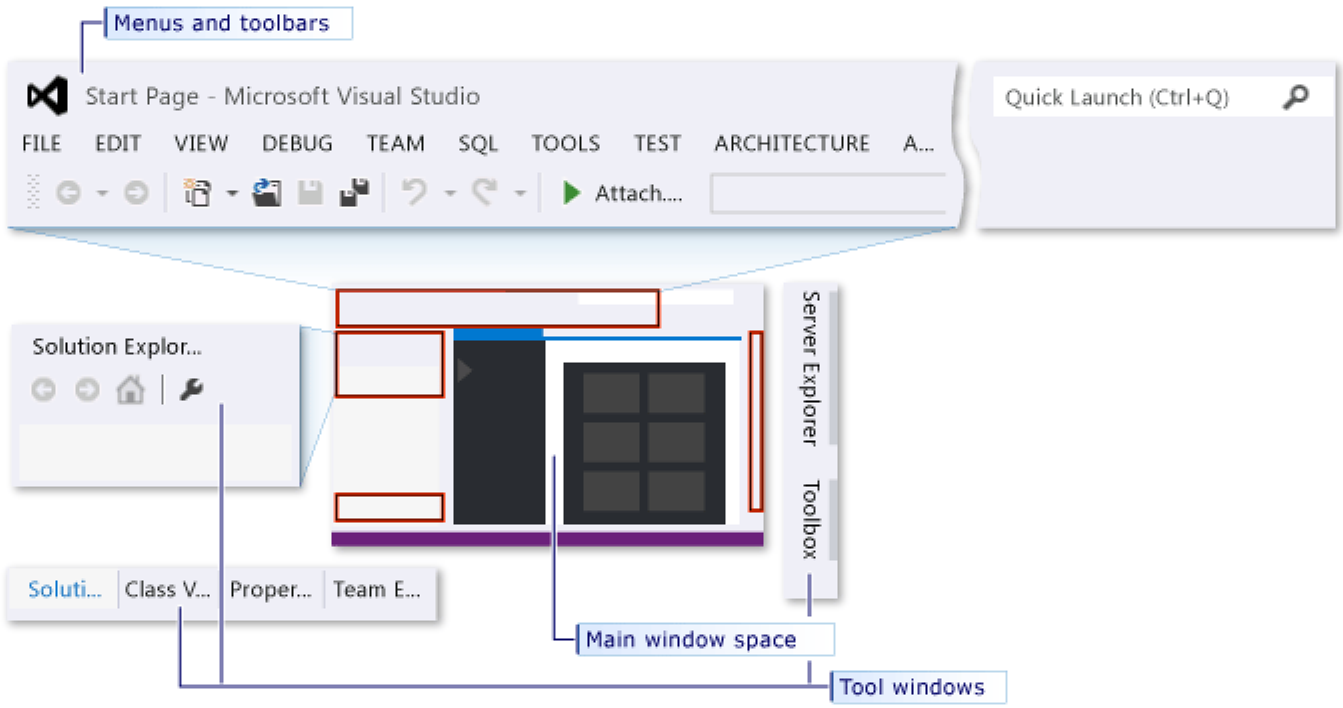
Local Help Documentation: Minimum

Downloads and installs the minimum amount of content (about 1 GB).

This walkthrough is written with **Visual C++ Development Settings** applied. You can change your settings combination by using the **Import and Export Settings Wizard**. For more information, see How to: Change Select Settings.

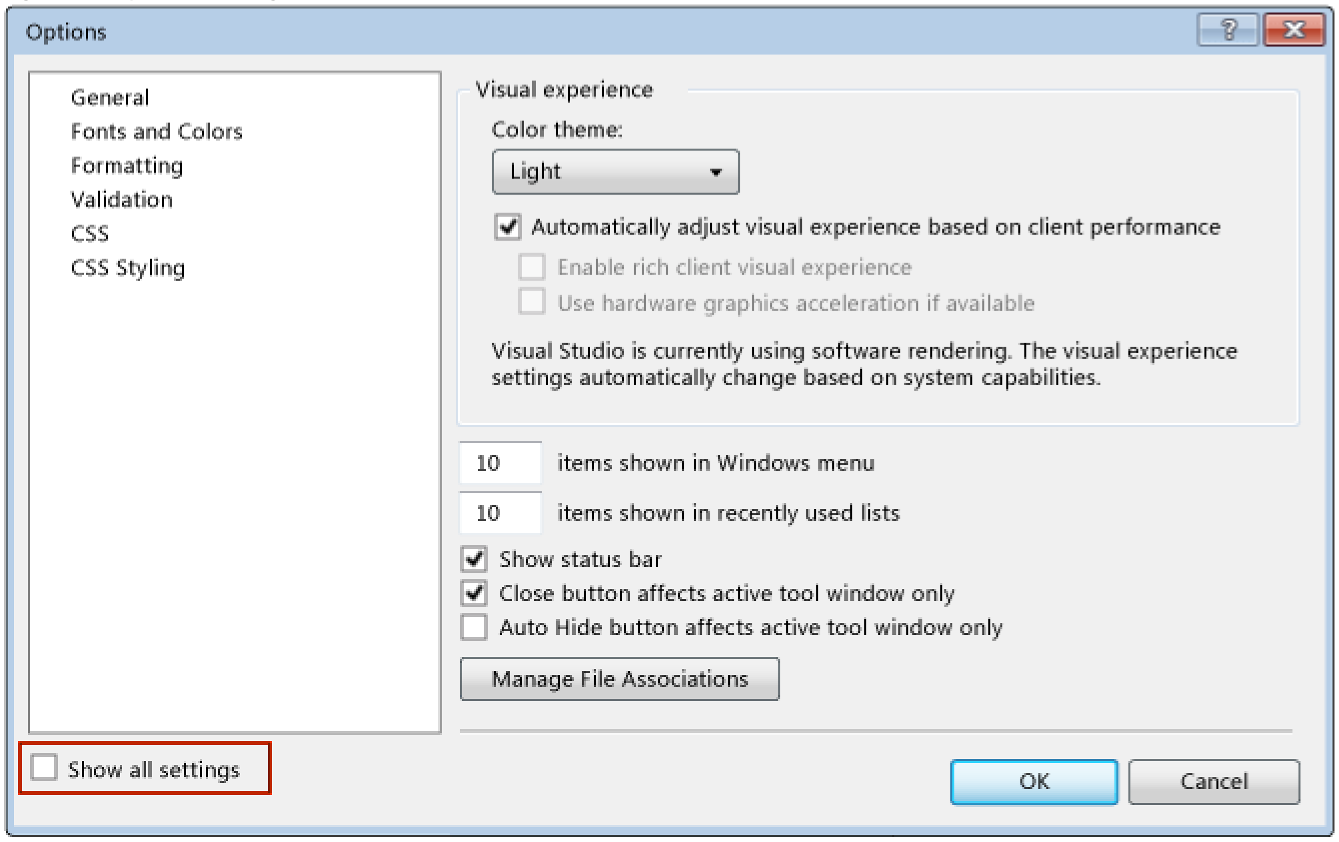
After you open Visual Studio, you can identify the three basic parts of the IDE: tool windows, menus and toolbars, and the main window space. Tool windows are docked on the left and right sides of the app window, with **Quick Launch**, the menu bar, and the standard toolbar at the top. The center of the application window contains the **Start Page**. When you open a solution or project, editors and designers appear in this space. When you develop an app, you'll spend most of your time in this central area.

Figure 2: Visual Studio IDE


Menus and toolbars
Start Page - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW DEBUG TEAM SQL TOOLS TEST ARCHITECTURE A...
Attach....
Solution Explor...
Server Explorer
Toolbox
Soluti... Class V... Proper... Team E...
Main window space
Tool windows

You can customize Visual Studio by using the **Options** dialog box. For example, you can change the typeface and size of the text that appears in the editor or the color theme of the IDE. Depending on the settings combination that you've applied, some items in that dialog box might not appear automatically. You can show all possible options by selecting the **Show all settings** check box.

Figure 3: Options dialog box



Options

- General
- Fonts and Colors
- Formatting
- Validation
- CSS
- CSS Styling

Visual experience

Color theme: Light

- [✓] Automatically adjust visual experience based on client performance
- [ ] Enable rich client visual experience
- [ ] Use hardware graphics acceleration if available

Visual Studio is currently using software rendering. The visual experience settings automatically change based on system capabilities.

10 items shown in Windows menu

10 items shown in recently used lists

- [✓] Show status bar
- [✓] Close button affects active tool window only
- [ ] Auto Hide button affects active tool window only

Manage File Associations

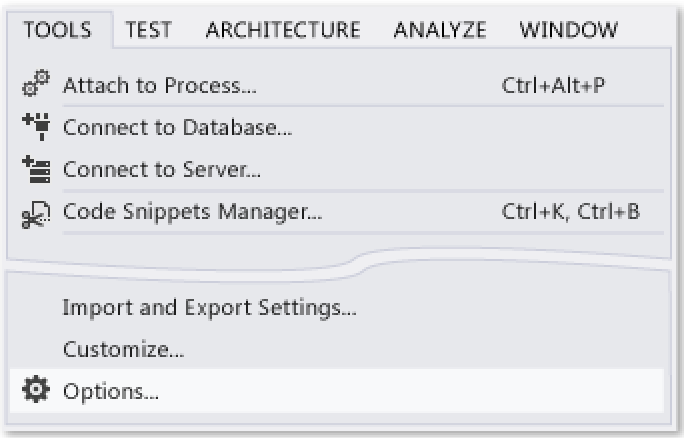
- [ ] Show all settings

OK Cancel

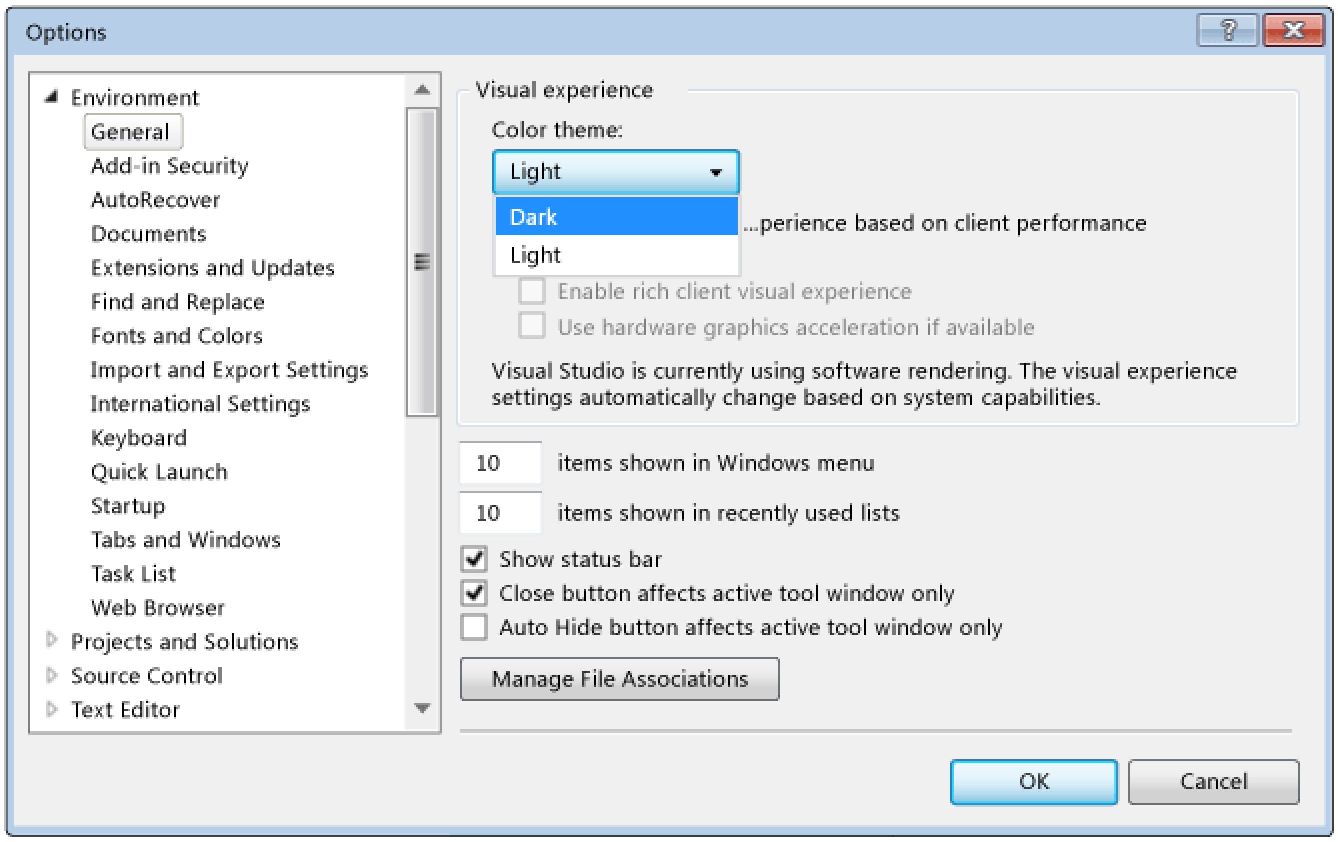
In this example, you'll change the color theme of the IDE from light to dark.

To change the color theme of the IDE

1. Open the **Options** dialog box.


TOOLS
TEST
ARCHITECTURE
ANALYZE
WINDOW
Attach to Process... Ctrl+Alt+P
Connect to Database...
Connect to Server...
Code Snippets Manager... Ctrl+K, Ctrl+B
Import and Export Settings...
Customize...
Options...

2. Change the **Color theme** to **Dark**.


Options

- Environment
  - General
  - Add-in Security
  - AutoRecover
  - Documents
  - Extensions and Updates
  - Find and Replace
  - Fonts and Colors
  - Import and Export Settings
  - International Settings
  - Keyboard
  - Quick Launch
  - Startup
  - Tabs and Windows
  - Task List
  - Web Browser
- Projects and Solutions
- Source Control
- Text Editor

Visual experience

Color theme: Light

Dark

Light

...perience based on client performance

Enable rich client visual experience

Use hardware graphics acceleration if available

Visual Studio is currently using software rendering. The visual experience settings automatically change based on system capabilities.

10 items shown in Windows menu

10 items shown in recently used lists

✓ Show status bar

✓ Close button affects active tool window only

Auto Hide button affects active tool window only

Manage File Associations

OK Cancel

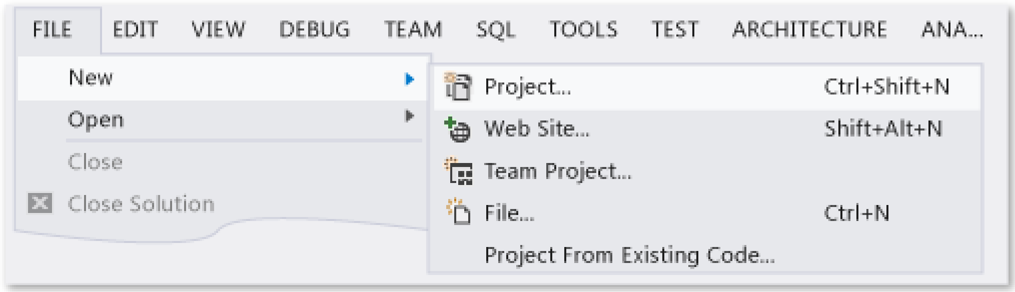
To return the IDE to the Light color theme, repeat the previous steps, except choose **Light** in step 2.

Create a simple app

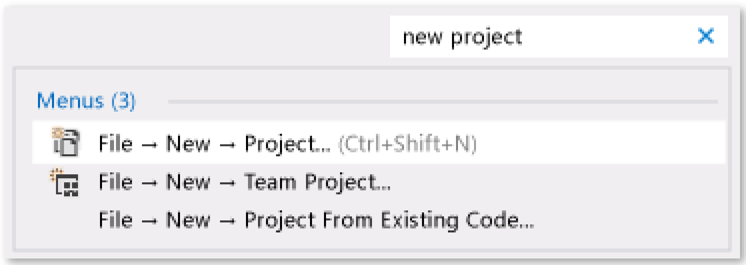
When you create an app in Visual Studio, you first create a project and a solution. For this example, you'll create a Windows console app.

To create a console app

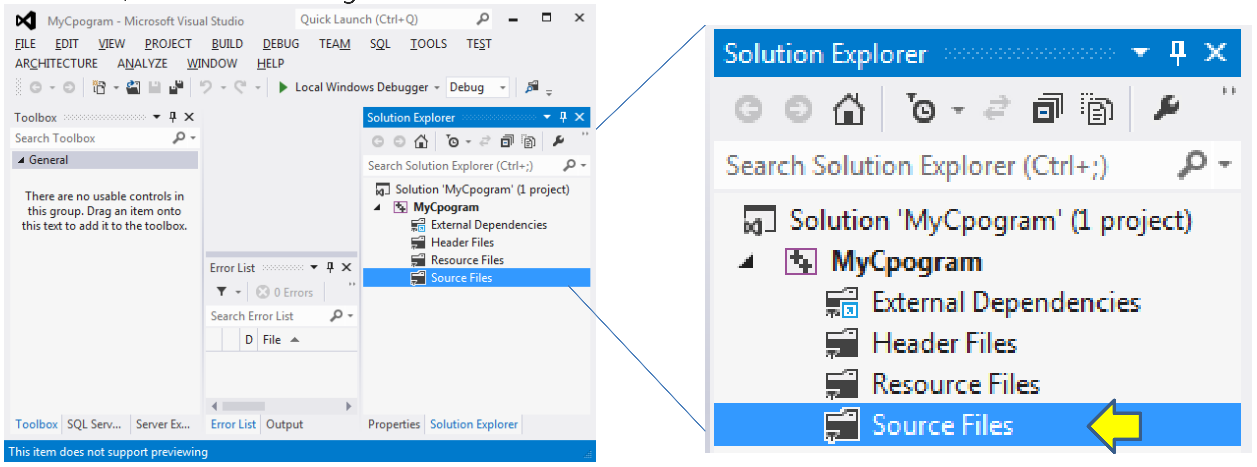
1. On the menu bar, choose **File**, **New**, **Project**.



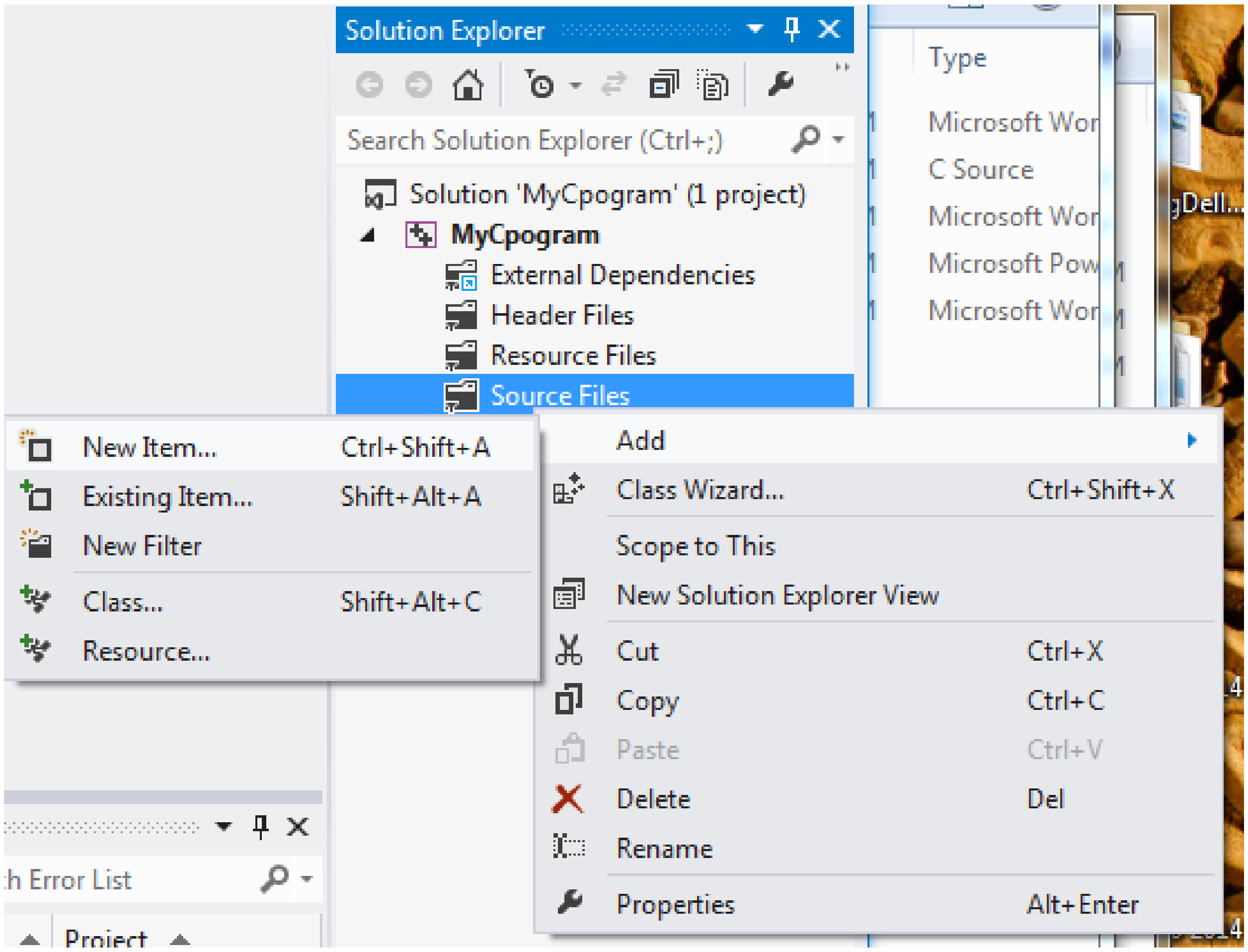
   Or use **Quick Launch** to open the **New Project** dialog box.


new project
Menus (3)
File → New → Project... (Ctrl+Shift+N)
File → New → Team Project...
File → New → Project From Existing Code...

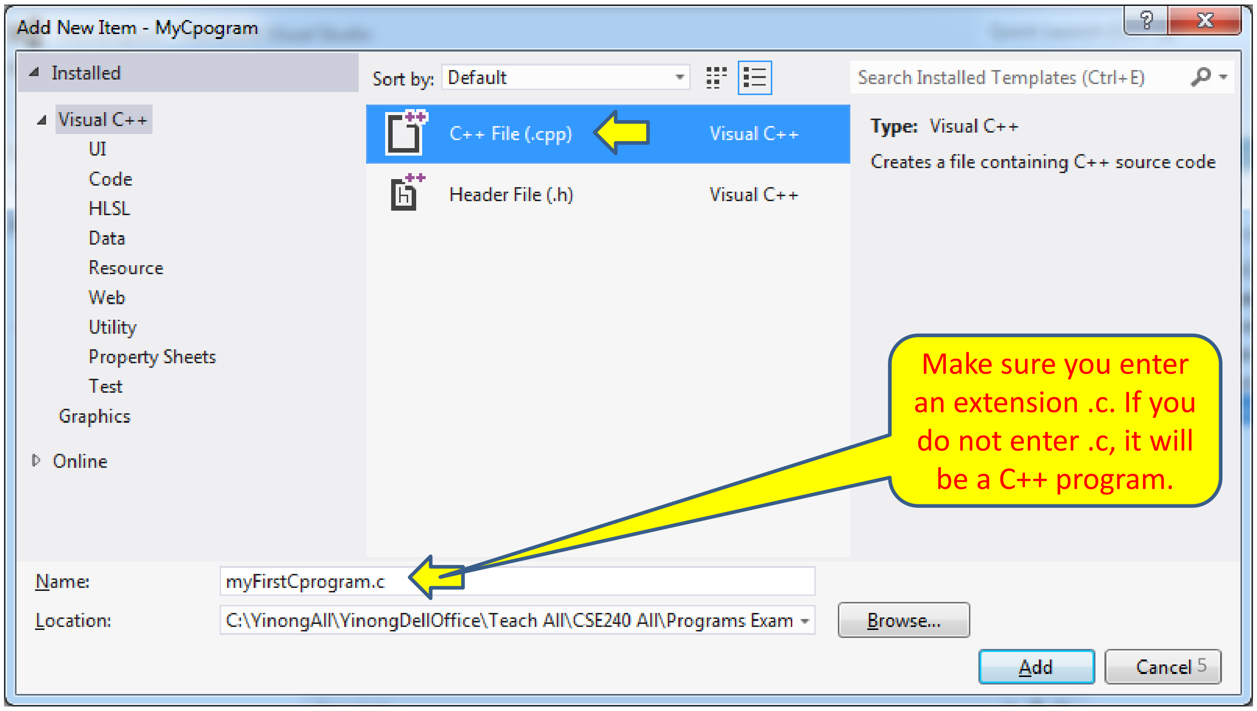
2. If you are creating a C project, you will choose "Empty Project", which will create an empty project with no source file, as shown in the figure below:



   You can add your source file to be compiled by right-clicking "Source File". And then choose the type of file to add, as shown in the figure below.



You can create a C program or a C++ program in this template. By default, it will add a C program. In order to add a C program, you must enter the program name with an extension .c, instead of .cpp. If you do not enter the extension .c, it will be considered to be C++ file too.



Type: Visual C++

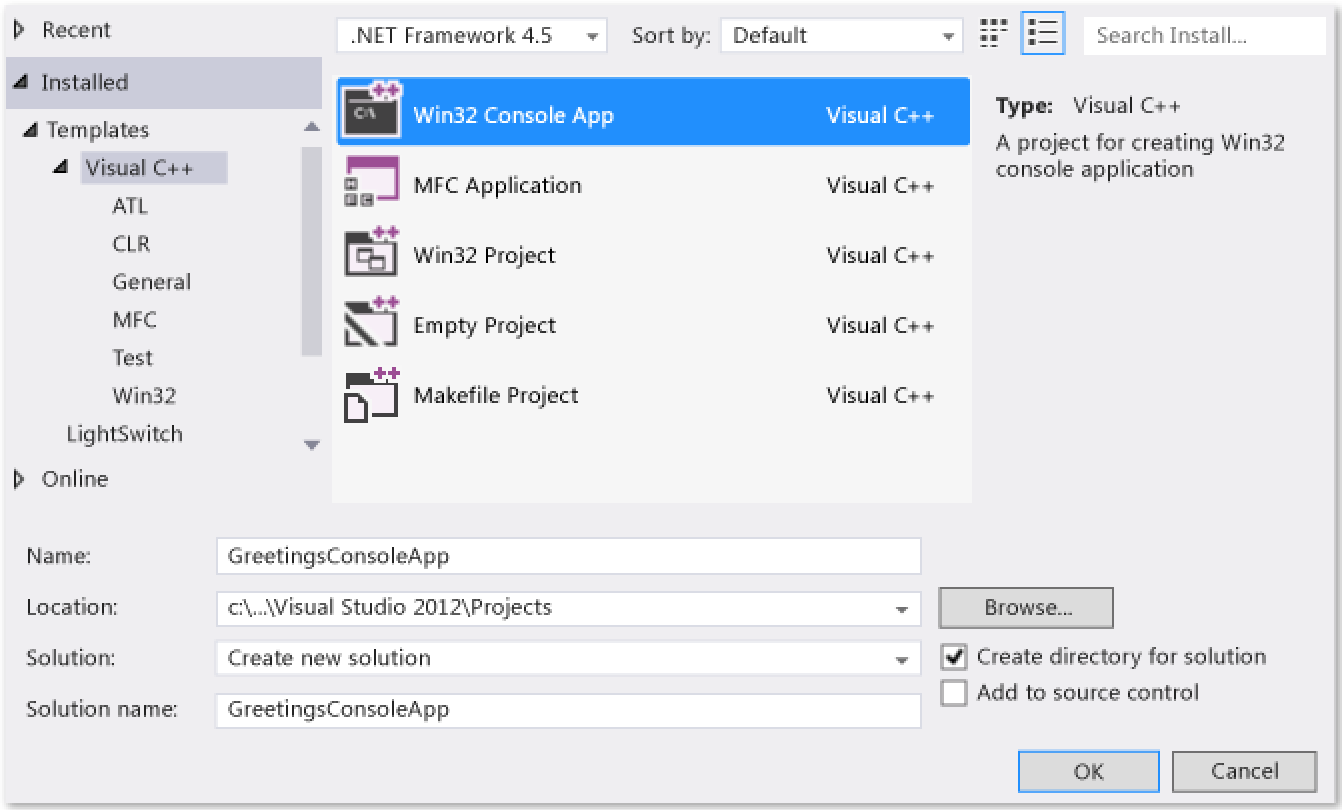
Creates a file containing C++ source code

Make sure you enter an extension .c. If you do not enter .c, it will be a C++ program.

Name: myFirstCprogram.c

Location: C:\YinongAll\YinongDellOffice\Teach All\CSE240 All\Programs Exam

3. If you are creating a C++ project, you can choose the **Win32 Console Application** template as well, and then name the project **GreetingsConsoleApp**.



Type: Visual C++

A project for creating Win32 console application

Name: GreetingsConsoleApp

Location: c:\...\Visual Studio 2012\Projects

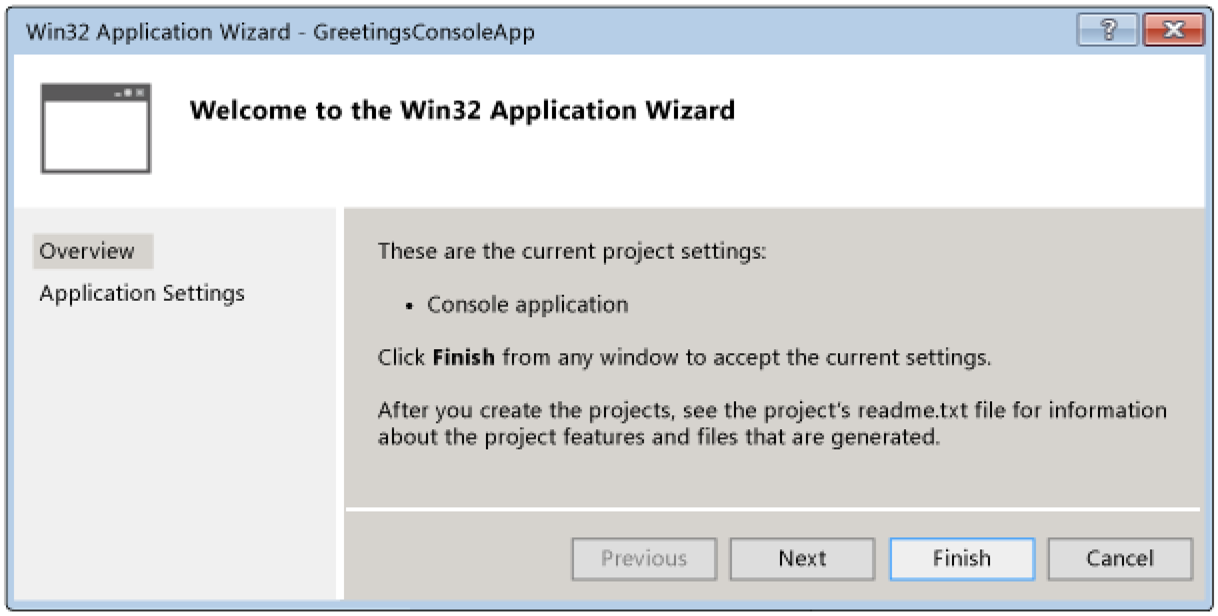
Solution: Create new solution

Solution name: GreetingsConsoleApp

[✓] Create directory for solution

[ ] Add to source control

4. When the Win32 Application Wizard appears, choose the **Finish** button.


Win32 Application Wizard - GreetingsConsoleApp

Welcome to the Win32 Application Wizard

These are the current project settings:

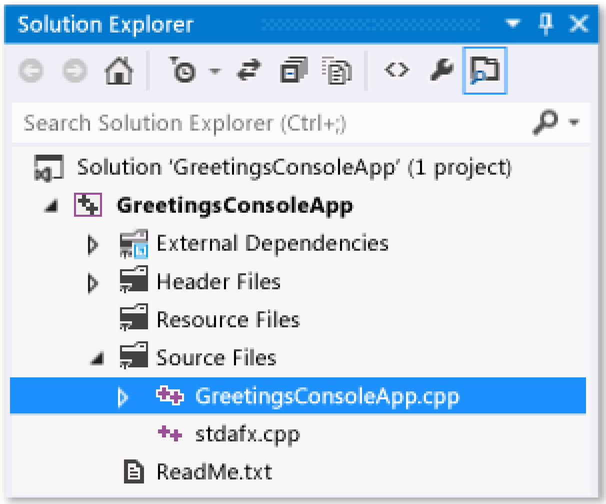
- Console application

Click **Finish** from any window to accept the current settings.

After you create the projects, see the project's readme.txt file for information about the project features and files that are generated.

The GreetingsConsoleApp project and solution, with the basic files for a Win32 console app, are created and automatically loaded into **Solution Explorer**. The GreettingsConsoleApp.cpp file is opened in the code editor. The following items appear in **Solution Explorer**:

Figure 4: Project items



```
Solution 'GreetingsConsoleApp' (1 project)
  GreetingsConsoleApp
    External Dependencies
    Header Files
    Resource Files
    Source Files
      GreetingsConsoleApp.cpp
      stdafx.cpp
    ReadMe.txt
```

Notice that the program must be added into the project. If you do not see your program file (.c or .cpp file) in the project folder “Source Files”, your program is not in the project and it will not run.

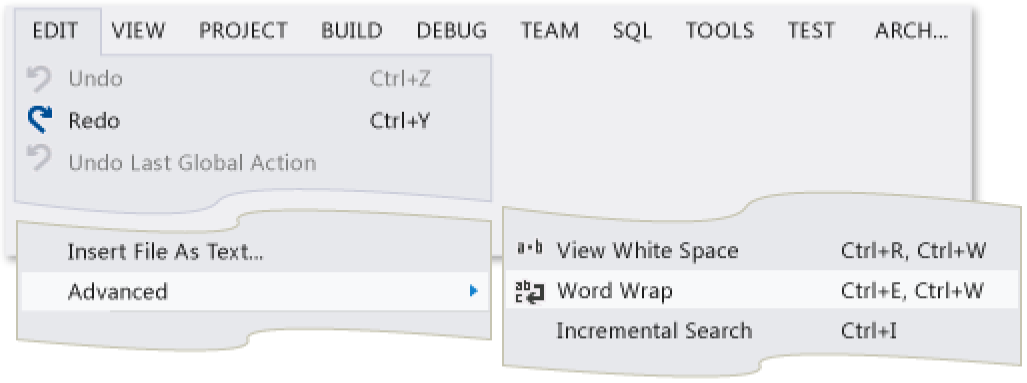
Customize the Code Editor

You can customize how code appears in the code editor. For example, you can display a dash to represent white space, or you can display line numbers to make navigation easier. You specify some customization options from the **Edit** menu, but you must open the **Options** dialog box to customize the IDE in other ways.

In the following procedures, you'll customize the code editor in some basic ways.

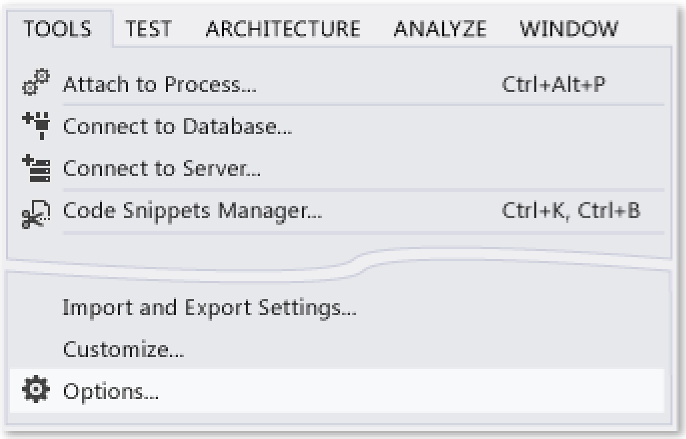
To enable word wrap

- On the menu bar, choose **Edit**, **Advanced**, **Word Wrap**.

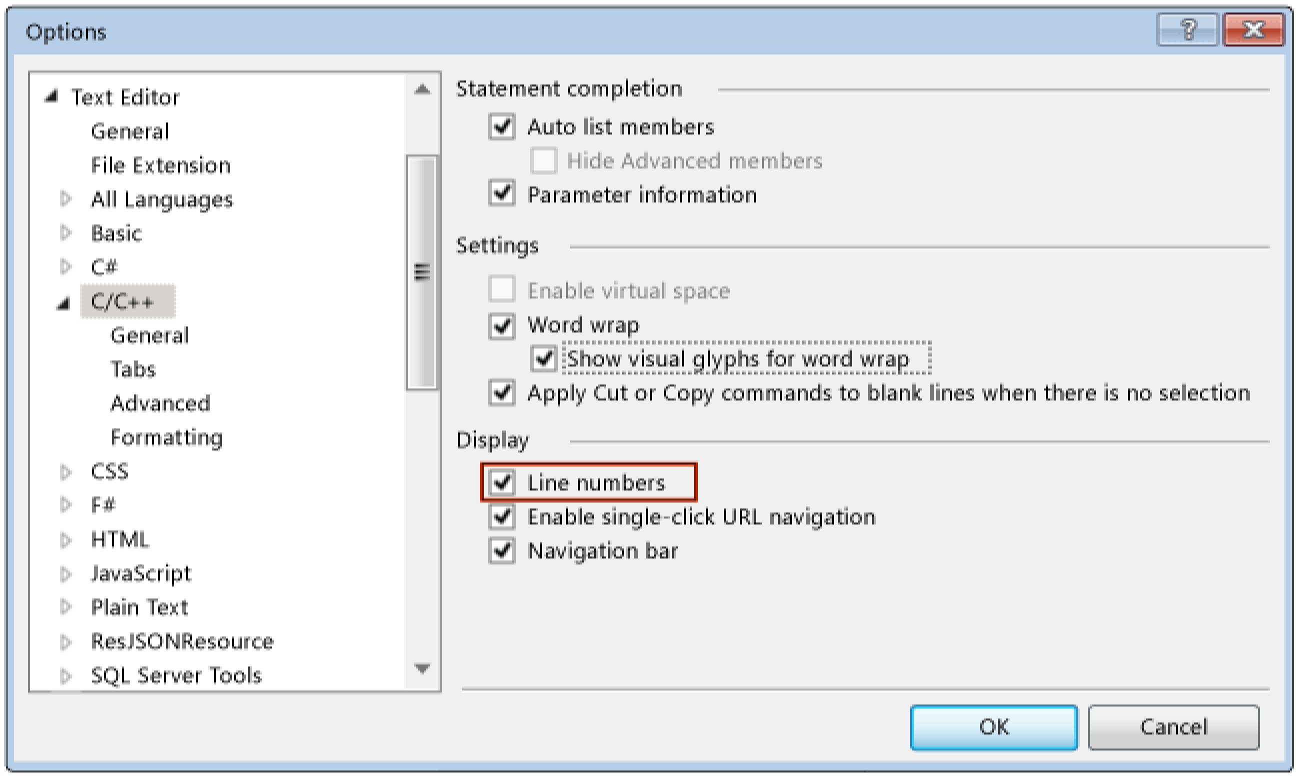


To enable line numbers

1. Open the **Options** dialog box.


TOOLS
TEST
ARCHITECTURE
ANALYZE
WINDOW
Attach to Process... Ctrl+Alt+P
Connect to Database...
Connect to Server...
Code Snippets Manager... Ctrl+K, Ctrl+B
Import and Export Settings...
Customize...
Options...

2. In the **Text Editor** category, choose the **C/C++** node, and then select the **Line numbers** check box.


Options

- Text Editor
  - General
  - File Extension
  - All Languages
  - Basic
  - C#
  - C/C++
    - General
    - Tabs
    - Advanced
    - Formatting
  - CSS
  - F#
  - HTML
  - JavaScript
  - Plain Text
  - ResJSONResource
  - SQL Server Tools

Statement completion

- [✓] Auto list members
- [ ] Hide Advanced members
- [✓] Parameter information

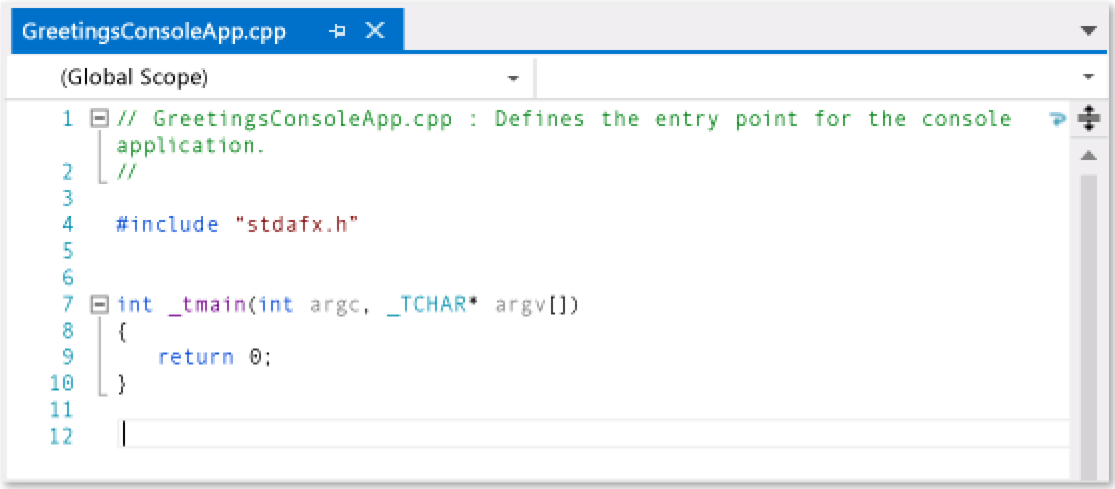
Settings

- [ ] Enable virtual space
- [✓] Word wrap
- [✓] Show visual glyphs for word wrap
- [✓] Apply Cut or Copy commands to blank lines when there is no selection

Display

- [✓] Line numbers
- [✓] Enable single-click URL navigation
- [✓] Navigation bar

The contents of the code editor should resemble the following illustration:



```
// GreetingsConsoleApp.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"


int _tmain(int argc, _TCHAR* argv[])
{
    return 0;
}

```

For more information, see Customizing the Editor.

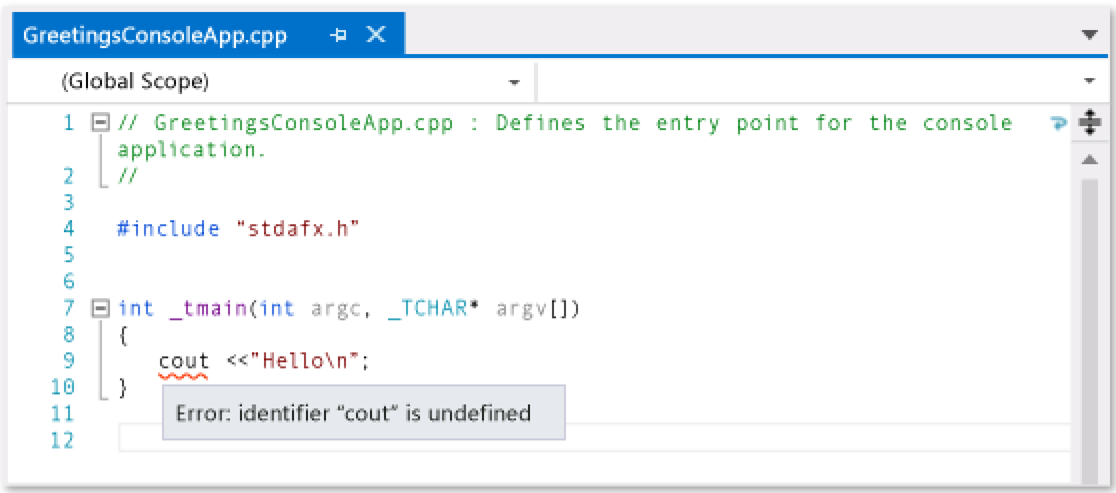
Add Code to the Application

Next, you'll add code to display the word "Hello" in the console window.

To display “Hello” in the console window

1. In the GreetingsConsoleApp.cpp file, enter a blank line before the line `return 0;`, and then enter the following code:
2. `cout <<"Hello\n";`

   A red squiggly line appears under `cout`. An error message appears if you point to it.



```
// GreetingsConsoleApp.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"


int _tmain(int argc, _TCHAR* argv[])
{
    cout <<"Hello\n";
}
```

Error: identifier "cout" is undefined

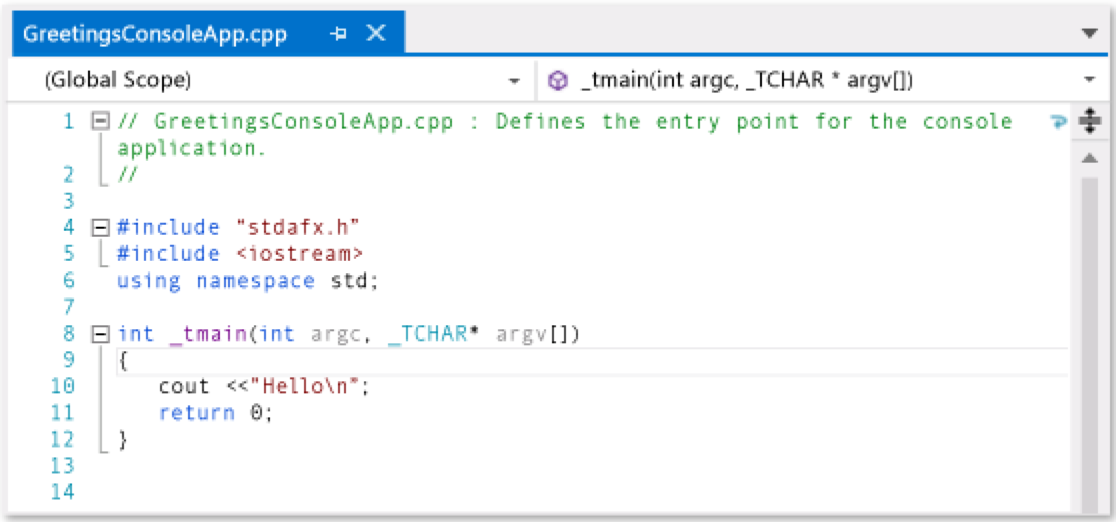
The error message also appears in the **Error List** window. You can display the window by, on the menu bar, choosing **View**, **Error List**.
cout is included in the <iostream> header file, which you've not yet declared.

3. In the line after `#include "stdafx.h"`, enter the following code:

```
4. #include <iostream>
5. using namespace std;
```

You probably noticed that a box appeared as you entered code, providing suggestions for the characters that you entered. This feature, which is named IntelliSense, provides a collection of coding prompts, such as List Members, Parameter Info, Quick Info, Signature Help, and Complete Word, to help you write code faster. In addition, you can use code snippets, which are pre-defined blocks of code. For more information, see Using IntelliSense and Code Snippets.
The red squiggly line under `cout` disappears when you fix the error.

6. Save the changes to the file.



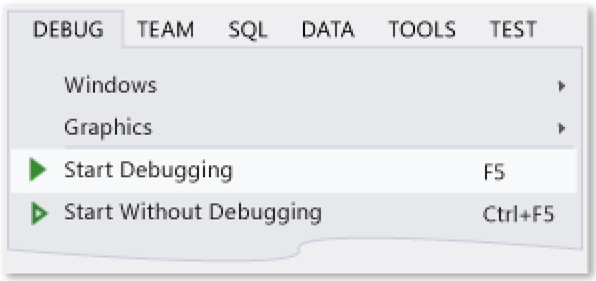
```
// GreetingsConsoleApp.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <iostream>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    cout <<"Hello\n";
    return 0;
}
```

Debug and Test the App

You should debug GreetingsConsoleApp to verify whether the word "Hello" appears in the console window as you expected.

To debug the app

- Start the debugger.

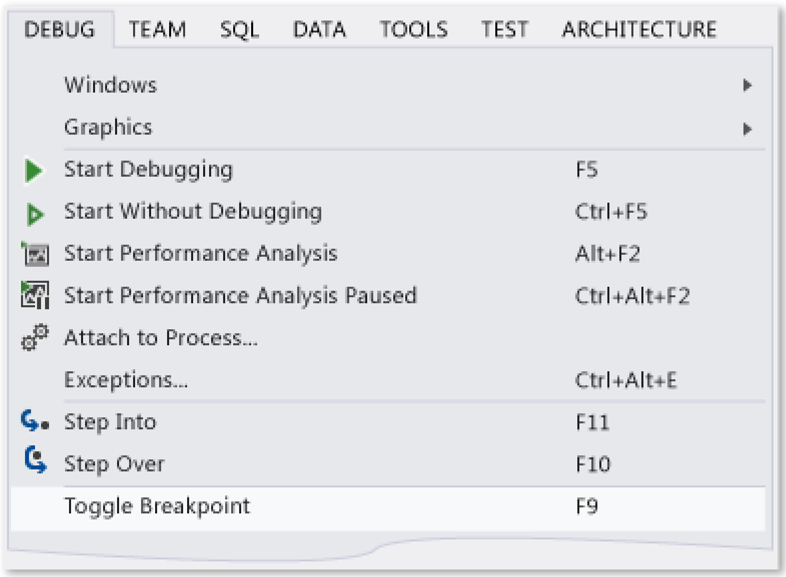


  The debugger starts and runs the code. The console window appears for a few seconds but closes quickly as the debugger stops running. The debugger completed running the app too quickly for you to verify what text appeared in the console window.

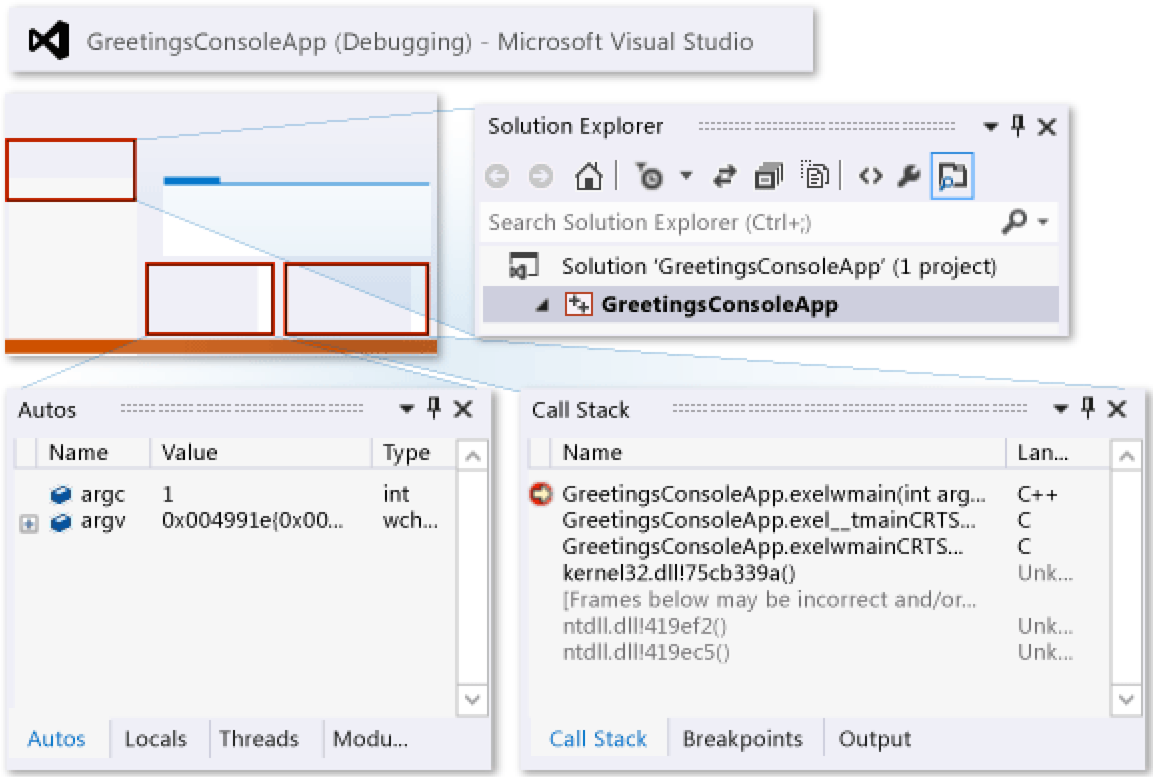
You'll use a breakpoint to force the app to pause long enough to identify what text appears in the console window.

To add a breakpoint

1. In the line `return 0;`, add a breakpoint from the menu bar.



   A red circle appears next to the line of code in the far left margin of the editor window.
2. Choose the F5 key to start debugging.
   The debugger starts, and several tool windows appear. On the bottom of the IDE, the Autos, Locals, Threads, Modules, and Watch windows are docked together on the left side, and the Call Stack, Breakpoints, and Output windows are docked together on the right side.


GreetingsConsoleApp (Debugging) - Microsoft Visual Studio

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'GreetingsConsoleApp' (1 project)
GreetingsConsoleApp

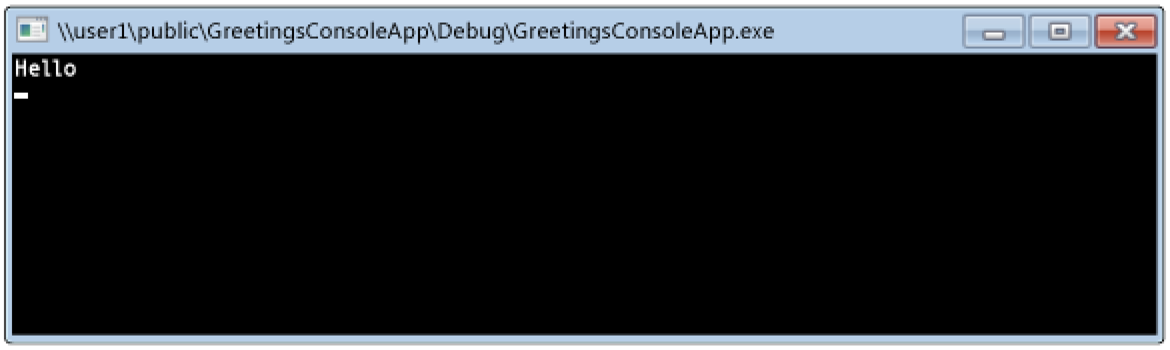
Autos

Name	Value	Type
argc	1	int
argv	0x004991e{0x00...	wch...

Call Stack

Name	Lan...
GreetingsConsoleApp.exe!wmain(int arg...	C++
GreetingsConsoleApp.exe!__tmainCRTS...	C
GreetingsConsoleApp.exe!wmainCRTS...	C
kernel32.dll!75cb339a()	Unk...
[Frames below may be incorrect and/or...	
ntdll.dll!419ef2()	Unk...
ntdll.dll!419ec5()	Unk...

3. Navigate to the console window to verify that the word Hello appears. The UI should resemble this illustration:


\\user1\public\GreetingsConsoleApp\Debug\GreetingsConsoleApp.exe

```
Hello
```

4. Choose the SHIFT + F5 keys to stop debugging.

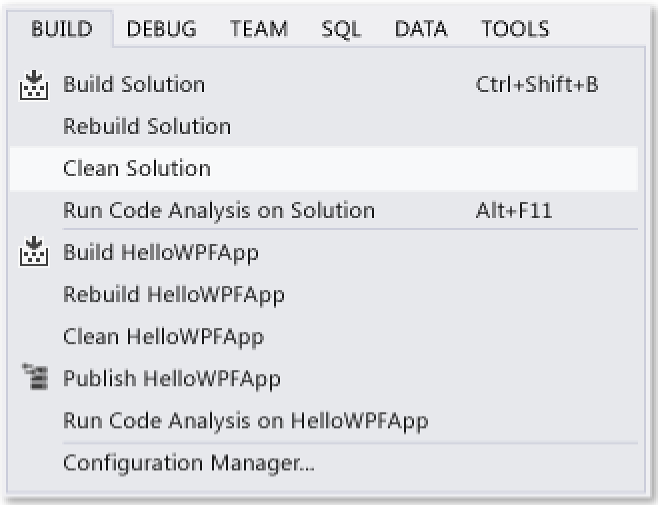
For more information, see Debugging Preparation: Console Projects.

Build a release version of the app

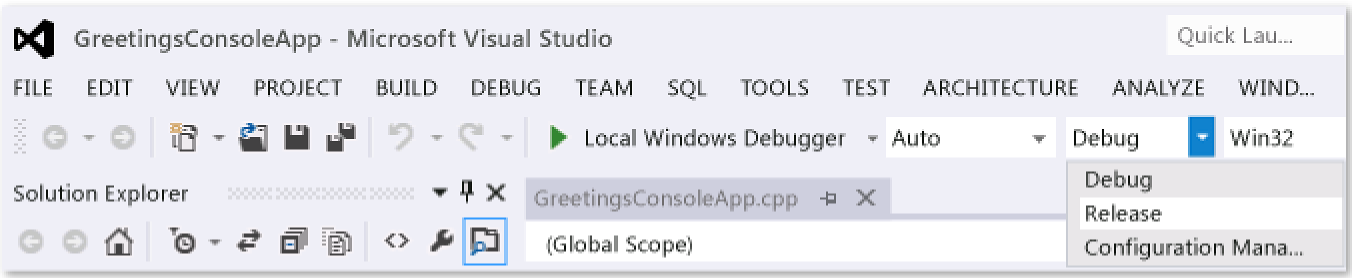
Now that you've verified that everything works, you can prepare a release build of the app.

To clean the solution files and build a release version

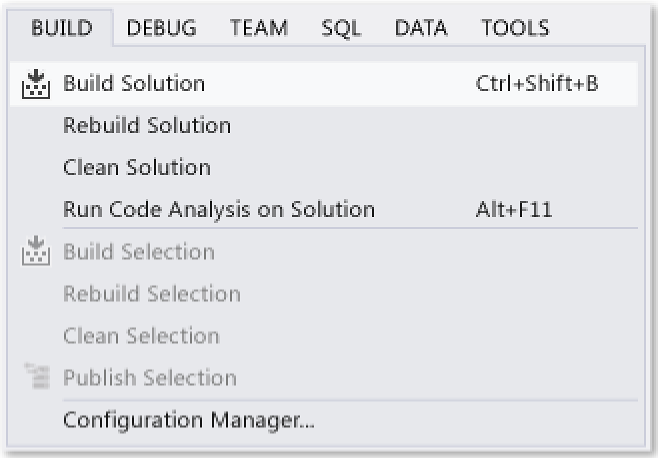
1. From the menu bar, delete intermediate files and output files that were created during previous builds.



2. Change the build configuration for GreetingsConsoleApp from **Debug** to **Release**.



3. Build the solution.


BUILD
DEBUG
TEAM
SQL
DATA
TOOLS
Build Solution Ctrl+Shift+B
Rebuild Solution
Clean Solution
Run Code Analysis on Solution Alt+F11
Build Selection
Rebuild Selection
Clean Selection
Publish Selection
Configuration Manager...