
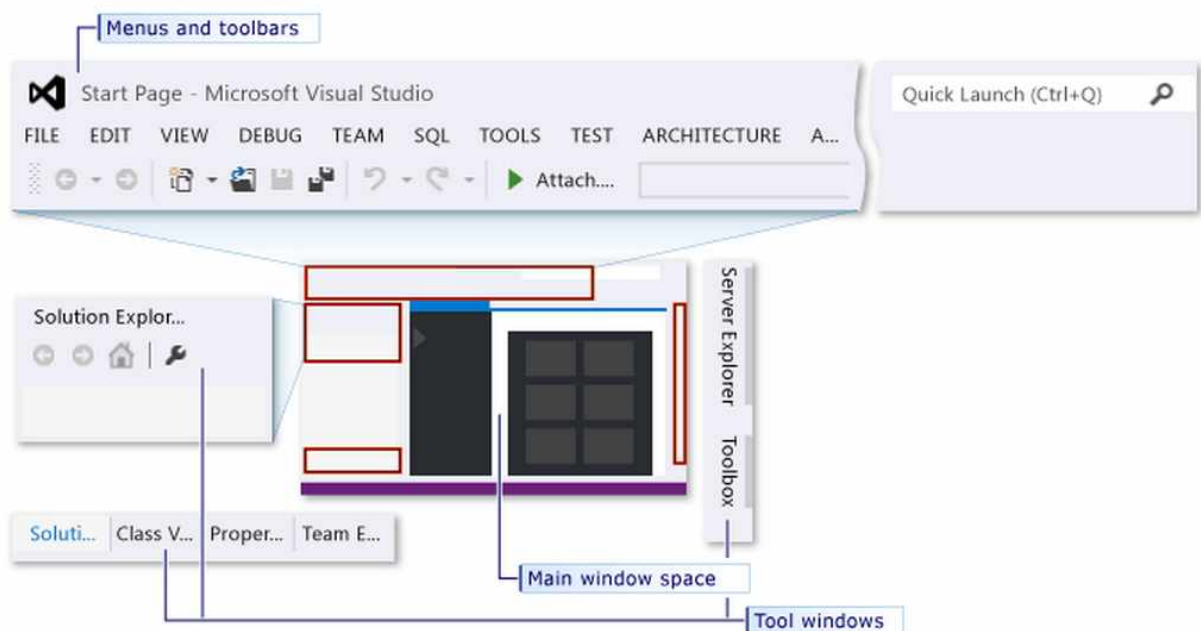


# Getting Started with C++ in Visual Studio

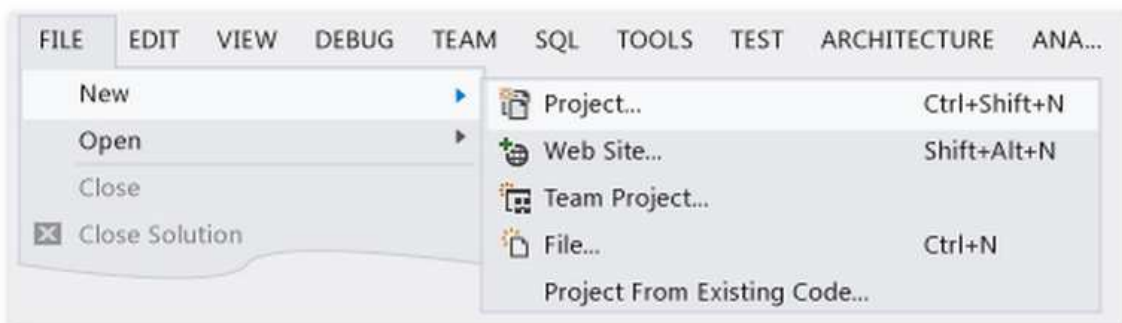
## Step 1: Open Visual Studio

1. **Double click** on the Visual Studio icon .
2. After you open Visual Studio, you can see 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.

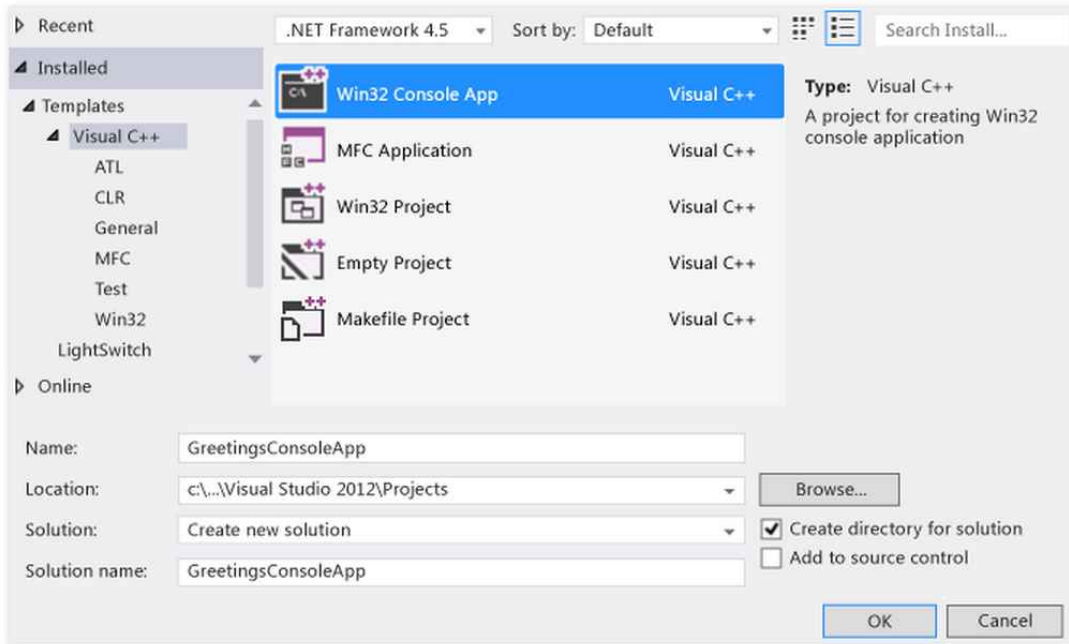


## Step 2: Create a console app.

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



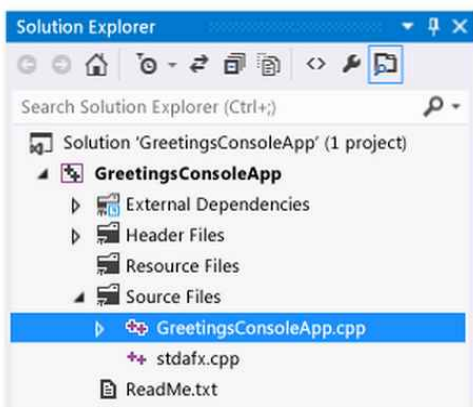
2. In the **Visual C++** category, choose the **Win32 Console Application** template, and then name the project **GreetingsConsoleApp**.



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



The GreetingsConsoleApp project and solution, with the basic files for a Win32 console app, are created and automatically loaded into **Solution Explorer**. The GreetingsConsoleApp.cpp file is opened in the code editor. The following items appear in **Solution Explorer**: To get to the file double click on the file name in the workspace window.



## Step 3: 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:

```
cout <<"Hello\n";
```

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



The error message also appears in the **Error List** window. You can display the window by, on the menu bar, choosing **View, Error List**.

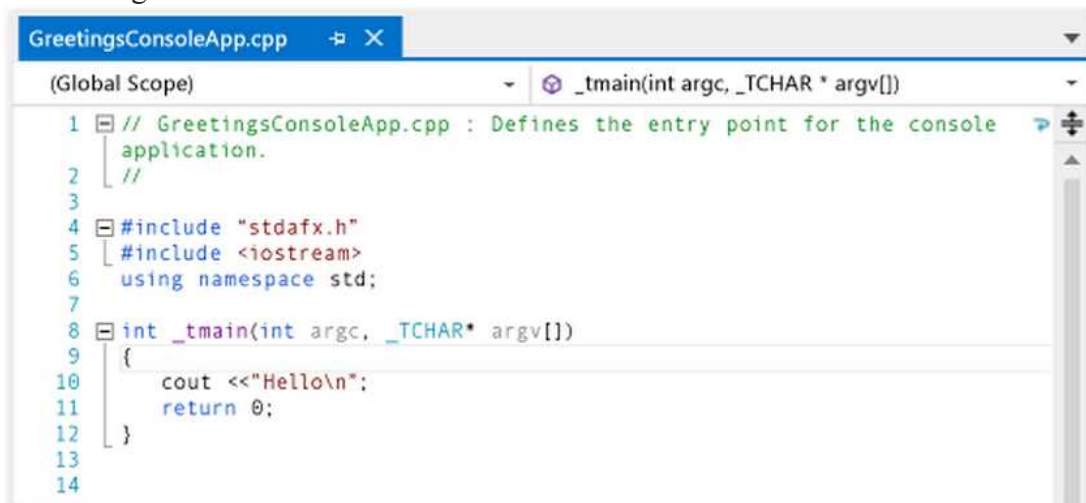
`cout` is included in the `<iostream>` header file.

2. To include the `iostream` header, enter the following code after `#include "stdafx.h"`:

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

The red squiggly line under `cout` disappears when you fix the error.

3. Save the changes to the file.



*Better remember the shortcut commands, will save a lot of time you waste in positioning the mouse.*

4. You can compile your code by **pressing F7** or by simply **clicking the Execute Program icon** (a red exclamation symbol) in the tool bar as shown below. Click on it and then click on Yes in the dialog box. If there is no warning or error in your project, the system displays the output window

### **Shortcut commands:**

Save Files	Cntrl + S	Compile the file	F7
Execute the program	Cntrl + F5		

# Step 4: Debugging with Visual C++ Debugger

1. Key in the following buggy program in order to learn how to use debugger in Visual C++ IDE.

## The Buggy Code

Copy the following code into Visual Studio to follow the tutorial:

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

int toPercent (float decimal);

void main() {
    int a, b;
    float c;
    int cAsPercent;

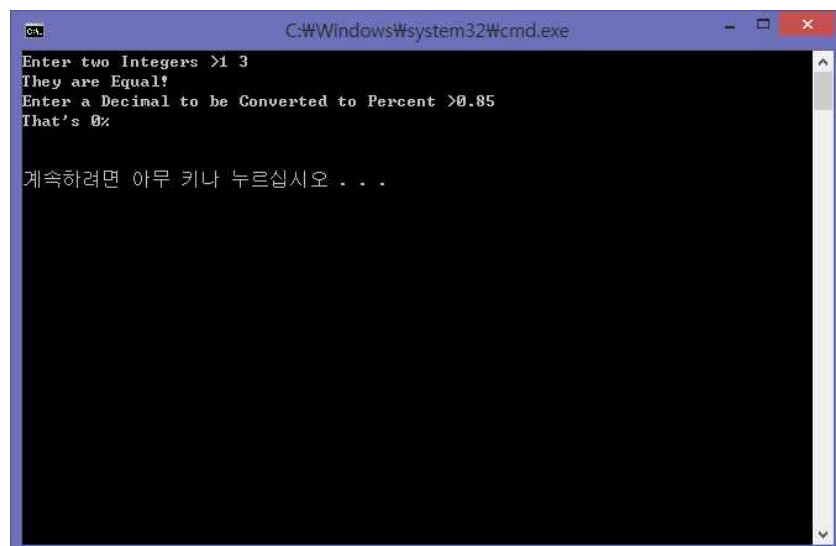
    cout << "Enter two Integers >";
    cin >> a >> b;

    if (a = b) cout << "They are Equal!\n";
    else if (a > b) cout << "The first one is bigger!\n";
    else cout << "The second one is bigger!\n";

    cout << "Enter a Decimal to be Converted to Percent >";
    cin >> c;
    cAsPercent = toPercent(c);
    cout << "That's " << cAsPercent << "%\n";
    cout << endl << endl;
    system("pause");
}

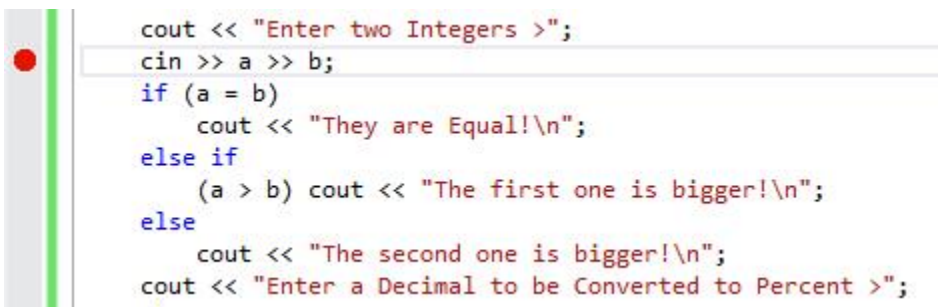
/* ToPercent():
Converts a given float (eg 0.9) to a percentage (90).
*/
int toPercent (float decimal) {
    int result;
    result = int(decimal) * 100;
    return result;
}
```

Trying to debug a program that's working perfectly is rather pointless, so we'll begin with a program that has some obvious problems:



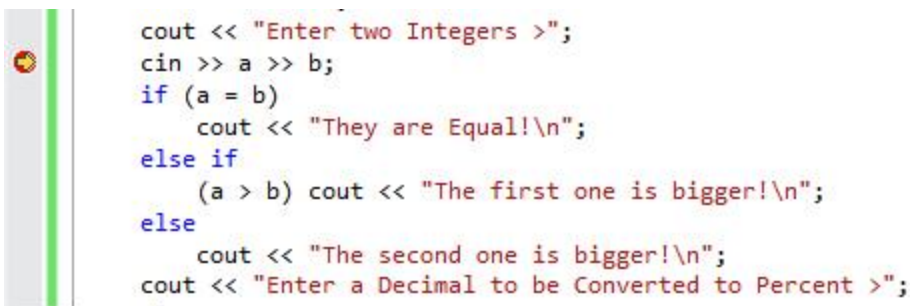
This program compiled cleanly (0 warnings, 0 errors), but obviously some things aren't working right. One and three AREN'T equal, and 0.85 is NOT 0%. So we'll examine the code and determine where the problems are.

2. None of the debugging tools available are useful if the program isn't being run in debug mode. Rather than just clicking the Exclamation Point icon or pressing Ctrl+F5 to run your program, the debugger must be run from the "Build" menu by clicking "Start Debug" then "Go". **The F5 key alone will also run in debug mode.**
3. In the Buggy Program example, let's set a breakpoint just before the user enters the two numbers that the program thinks are equal. **Move to that line (cin >> a >> b;)** and either **right-click** and **select "Insert/Remove Breakpoint"** or press the F9 key. You can also just click in the gray bar at the left to set a breakpoint. Right click on an existing breakpoint glyph to set properties, disable, or delete the breakpoint. A red glyph will appear next to the line, indicating a breakpoint is set:



```
cout << "Enter two Integers >";
cin >> a >> b;
if (a = b)
    cout << "They are Equal!\n";
else if
    (a > b) cout << "The first one is bigger!\n";
else
    cout << "The second one is bigger!\n";
cout << "Enter a Decimal to be Converted to Percent >";
```

4. Run the program in Debug mode(**F5**), and the output window will appear, but the compiler window will 'jump' to the foreground and a yellow arrow will be pointing to the line with your breakpoint.
5. The yellow arrow means that statement will execute next! That is, it has not executed yet.



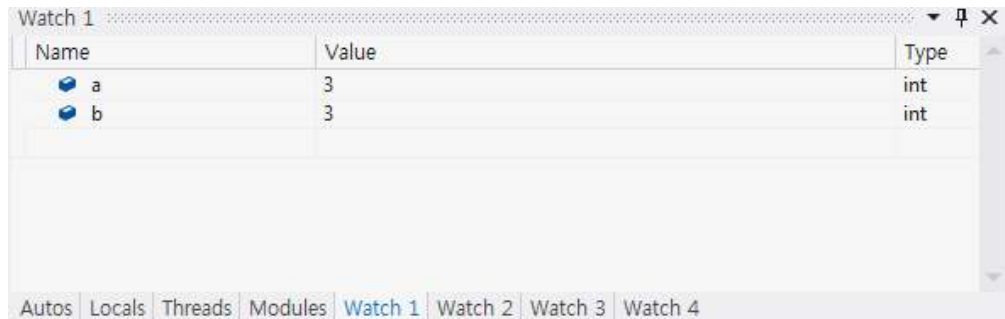
```
cout << "Enter two Integers >";
cin >> a >> b;
if (a = b)
    cout << "They are Equal!\n";
else if
    (a > b) cout << "The first one is bigger!\n";
else
    cout << "The second one is bigger!\n";
cout << "Enter a Decimal to be Converted to Percent >";
```

6. You've setup a starting breakpoint, you can start checking for data problems. The "Watch" window lets you watch the contents of any variables you select as your program executes. Thus, if the value changes in an unexpected way you'll see it as it happens. If the Watch window isn't already open, open it from the View menu (**Debug Windows > Watch**), or by clicking the "Watch" icon in the toolbar, or by pressing Ctrl+Alt+W, 1 or 2 or 3 or 4



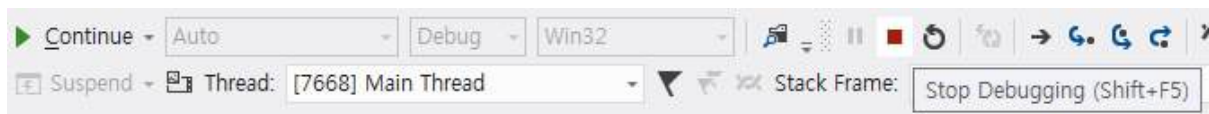


4. "Step Over" the next line (which won't require any user input) and again examine the Watch window.



Examine the line you just executed, and the error should be fairly obvious — the condition of the 'if' is (a = b) rather than (a == b).

5. "Stop Debugging" from the Debug menu or on the toolbar or press **Shift+F5**.



### For Debugging:

Start Debugging	F5	Introduce/ Remove Break point	F9
Step Over	F10	Step into	F11
Stop Debugger	Shift + F5		

## Step 6: Step into toPercent()

1. In the "Buggy Program" example, let's assume again that the value for 'c' is stored correctly and set a breakpoint on the next line — the one that calls the ToPercent() function. Then run the program until that point is reached.



2. If you step *over* the function call, the value of 'cAsPercent' changes:





Looking closely at the line, 'decimal' is being truncated to an int *before* it's multiplied.

## Step 7: Viewing state

1. Enter the code below to see the contents of the **all** array.

### Debugtest2.cpp

```
#include <iostream>
#include <cstdlib>
using namespace std;
class Pair
{
    public:
    int x,y;
    Pair(int x1=0, int y1=0){x = x1; y = y1;}
};
int main() {
    int const SIZE=10;
    int a, b;
    Pair all[SIZE];
    int tot=0;
    for (int i=0; i < SIZE; i++)
    {
        cout << "Enter two Integers >";
        cin >> a >> b;
        all[i] = Pair(a,b);
        tot += all[i].x;
    }
    return 0;
}
```

2. Place the mouse cursor over the variable in code and allow Visual Studio to display a DataTip. DataTips are only available when the program is in break mode.



### 3. Variable Windows

There are other options besides the watch window. You can also open a locals window from the **Debug menu (Windows -> Locals)**. The locals window will automatically display all local variables in the current block of code. If you are inside of a method, the locals window will display the method parameters and locally defined variables. The **Autos window (Debug -> Windows -> Autos)** will display variables *and expressions* from the current line of code, and the preceding line of code.