

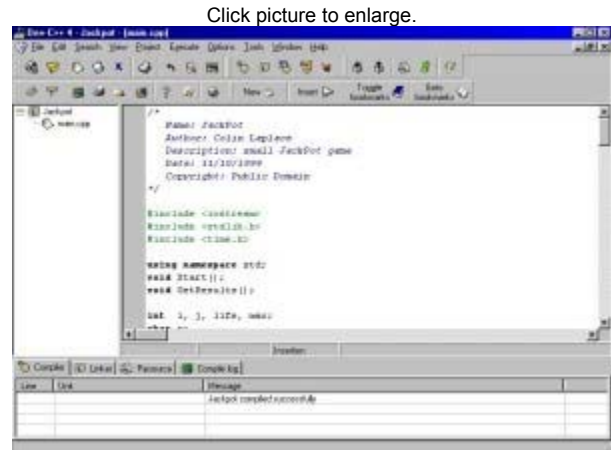
# Dev-C++ Tutorial for CSC 161 Students (Maintained by

**Mike Serrano)**

## What is Dev-C++?

Dev-C++, developed by [Bloodshed Software](#), is a fully featured graphical IDE (Integrated Development Environment), which is able to create Windows or console-based C/C++ programs using the MinGW compiler system. MinGW (Minimalist GNU\* for Windows) uses GCC (the GNU g++ compiler collection), which is essentially the same compiler system that is in Cygwin (the unix environment program for Windows) and most versions of Linux. There *are*, however, differences between Cygwin and MinGW; link to [Differences between Cygwin and MinGW](#) for more information.

*\*GNU is a recursive acronym for "GNU's Not Unix"; it is pronounced "guh-NEW". The [GNU Project](#) was launched in 1984 to develop a [free](#) and complete Unix-like operating system.*



## Bloodshed!?

I'll be the first to say that the name Bloodshed won't give you warm and fuzzies, but I think it's best if the creator of Bloodshed explains:

*First I would like to say that I am not a satanist, that I hate violence/war and that I don't like heavy metal / hard-rock music. I am french, but I do know the meaning of the "Bloodshed" word, and I use this name because I think it sounds well. If you are offended by the name, I am very sorry but it would be a big mess to change the name now.*

*There's also a reason why I keep the Bloodshed name. I don't want people to think Bloodshed is a company, because it isn't. I'm just doing this to help people.*

*Here is a good remark on the Bloodshed name I received from JohnS:  
I assumed that this was a reference to the time and effort it requires of you to make these nice software programs, a la "Blood, Sweat and Tears".*

*Peace and freedom,*

*Colin Laplace*

## Getting Dev-C++

The author has released Dev-C++ as free software (under GPL) but also offers a [CD for purchase](#) which can contain all Bloodshed software (it's customizable), including Dev-C++ with all updates/patches.

Link to Bloodshed Dev-C++ for a list of [Dev-C++ download sites](#).

You should let the installer put Dev-C++ in the default directory of C:\Dev-C++, as it will make it easier to later install add-ons or upgrades.

## The Insight Debugger

The download version of Dev-C++ comes with GDB, the GNU console-mode debugger. If you are extremely comfortable with the command line, you should be able to use GDB, but I highly recommend installing the Insight Debugger. Insight is the graphical counter-part of GDB (in fact, Insight is based on the GDB source code) originally written for Cygwin. You can, however, use Insight with Dev-C++ if you know where to put the files. Fortunately, the hard work of figuring out what files go where has already been done for you. I have put together a self-extracting ZIP file that should put everything where it needs to go.

Download the Dev-C++ customized distribution of Insight v5.0 here: [insight5\\_win32\\_exe.zip](#) (2.76 MB)

[It's wrapped inside of a ZIP file because Homestead.com does not allow EXE files]

If you having trouble downloading the above file, the non-self-extracting version is available via the [Dev-C++ development and resources page](#).

## Using Dev-C++

This section is probably why you are here.

All programming done for CSC161 will require separate compilation projects (i.e. class header file(s), class implementation file(s) and a main/application/client/driver file). This process is relatively easy as long as you know what Dev-C++ requires to do this.

### **Step 1: Configure Dev-C++.**

We need to tell Dev-C++ where we plan to save header files and modify one of the default settings.

- Go to the "Options" menu and select "Compiler Options".
- In the "Directories" tab, put check in the "Add the directory below to be searched for include files:" option, and enter the location(s) of where you plan to save your files. *TIP: Do not use directory names with spaces and make sure to separate multiple entries with semicolons.* As an example, mine is set to: C:\PRG;C:\PRG\CSC161.
- In the "Linker" tab, put check in the "Generate debugging information" option. This will allow you to use the debugger with your programs.

### **Step 2: Create a new project.**

A "project" can be considered as a container that is used to store all the elements that are required to compile a program.

- Go to the "File" menu and select "New Project..." (or just press CTRL+N).
- Choose "Empty Project", make sure "C++ project" is selected, and click "OK".
- At this point, Dev-C++ will ask you to give your project a name. You can give your project any valid filename, but keep in mind that the name of your project will also be the name of your executable.

### **Step 3: Create/add source file(s).**

You should now have a project with one empty and untitled source file. This would be sufficient if we were writing simple programs that relied exclusively on the standard library and/or precompiled object code, but we're not.

You can create additional empty source files one of two ways,

- Go to the "File" menu and select "New Source File" (or just press CTRL+U) OR
- Go to the "Project" menu and select "New Unit in Project" (or just press CTRL+F1).

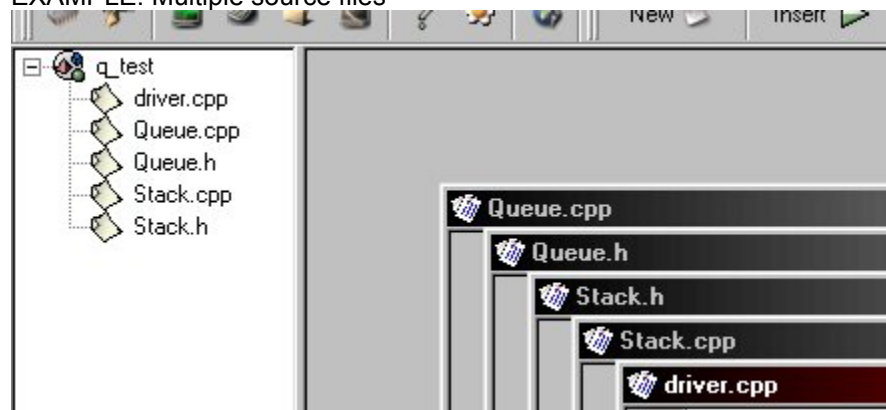
Note that Dev-C++ will not ask for a filename for any source file until you attempt to:

1. Compile
2. Save the project
3. Save the source file
4. Exit Dev-C++

You can add pre-existing source files one of two ways,

- Go to the "Project" menu and select "Add to project..." (or just press CTRL+F2) OR
- Right-click on the project name in the left-hand panel and select "Add to project..." (or just press CTRL+F2).

EXAMPLE: Multiple source files



In this example, more than 3 files are required to compile the program; The "driver.cpp" file references "Queue.h" (which requires "Queue.cpp") and "Queue.cpp" references "Stack.h" (which requires "Stack.cpp").

Note that Dev-C++ uses a ".cpp" filename extension for source files (rather than ".cc").

#### Step 4: Compile.

Once you have entered all of your source code, you are ready to compile.

- Go to the "Execute" menu and select "Compile" (or just press CTRL+F9).

It is likely that you will get some kind of compiler or linker error the first time you attempt to compile a project. Syntax errors will be displayed in the "Compiler" tab at the bottom of the screen. You can double-click on any error to take you to the place in the source code where it occurred. The "Linker" tab will flash if there are any linker errors. Linker errors are generally the result of syntax errors not allowing one of the files to compile.

#### Step 5: Execute.

Once your project successfully compiles, you will get a dialog box with several options:

- "Continue", which will just take you back to Dev-C++.
- "Parameters", which will allow you to pass command-line parameters to your program.
- "Execute", which will execute your program.

#### *Disappearing windows*

If you execute your program (with or without parameters), you may notice something peculiar; a console

window will pop up, flash some text and disappear. The problem is that, if directly executed, console program windows close after the program exits.

You can solve this problem one of two ways:

- *Method 1 - Scaffolding:*

Add the following code before any `return` statement in `main()` or any `exit()` or `abort()` statement (in any function):

```
/* Scaffolding code for testing purposes */
cin.ignore(256, '\n');
cout << "Press ENTER to continue..." << endl;
cin.get();
/* End Scaffolding */
```

This will give you a chance to view any output before the program terminates and the window closes.

- *Method 2 - Command-prompt:*

Alternatively, instead of using Dev-C++ to invoke your program, you can just open an MS-DOS Prompt, go to the directory where your program was compiled (i.e. where you saved the project) and enter the program name (along with any parameters). The command-prompt window will not close when the program terminates.

For what it's worth, I use the command-line method.

### Step 6: Debug.

When things aren't happening the way you planned, a source-level debugger can be a great tool in determining what really is going on. From this point forward, I will assume that you have downloaded and installed the Insight debugger (see [above](#)).

- Go to the "Execute" menu and select "Debug" (or just press F8).  
After a console window pops up and disappears, the debugger Source Window should appear. If the Source Window is blank, it is likely that you compiled your project without the necessary debugging information (see [above](#)).

*Passing command-line arguments:*

- Go to the "File" menu of the Source Window and select "Target Settings..."; select "Exec" from the "Target" drop-down list and enter any command-line arguments (separated by spaces) in the "Arguments:" field. If using filenames as arguments, be sure to include the path. (For example, instead of entering just "IN.TXT", enter "C:\PRG\CSC161\IN.TXT").

*Using the debugger:*

The various features of the debugger are pretty obvious. Click the "Run" icon to start your program; "Step" to step through every command; "Next" to execute the next block of code. You can view memory/variables/expressions using the various tools such as Registers, Memory, Watch Expressions and Local Variables.

Note that Insight was not originally designed to work with Dev-C++ and may be a bit flaky under certain situations. If you are having too much trouble with Insight and are feeling adventurous, you can reinstall Dev-C++ and use the console-mode GDB debugger that comes with the program. For more

information about using GDB, start Dev-C++ and go to "Help", "GNU Debugger help".

---

## **Dev-C++ User F.A.Q.**

This is the work-in-progress Dev-C++ User Frequently Asked Questions (with answers).

### **How do I use the C++ string class?**

Dev-C++ *does* come with support for the C++ string class template--the problem some people are having is getting it to work. I have no definitive answer to what is causing this. First of all, make sure you "#include <string>" (not string.h).

Example:

```
#include <iostream>
#include <string>

int main()
{
    string s;
    s = "This is a test";
    cout << s << endl;
    system("PAUSE");
    return 0;
}
```

If this example program does not compile, first try reinstalling Dev-C++. If that doesn't work, download this file: [gcc-2.95.2-3.zip](#) (2.18 MB)

The ZIP file contains the latest stable version (2.95.2-3) of GCC, which is the underlying compiler (and libraries) which Dev-C++ uses. Just unzip in the directory where you installed Dev-C++.

If you are still having problems getting strings to work, let me know--any and all information on this issue is welcome.

### **How do I use Borland Graphics Interface (graphics.h)?**

For those of you migrating from Borland, you may be wondering where graphics.h is. Unfortunately, graphics.h is a Borland specific library and cannot be used with Dev-C++. Fortunately, a benevolent soul by the name of Michael Main has modified a BGI emulation library for Windows applications (originally written by [Konstantin Knizhnik](#)) to be used under MinGW (and Dev-C++) which he has aptly named [WinBGIm](#).

The files we need are:

[winbgim.h](#) (download to /Dev-C++/include)

[winbgim.cpp](#) (download to /Dev-C++/include)

[libbgi.a](#) (download to /Dev-C++/lib)

After you have downloaded the files to the correct locations, you can now use winbgim.h as you would graphics.h with a few caveats.

*Using library files:*

First, you have to tell Dev-C++ where to find the library functions that WinBGIm references--this is

done in the "Project Options" dialog box.

Here are instructions on how to do this with a new project:

- Follow [step 2](#) and [step 3](#) of "Using Dev-C++".
- Go to "Project" menu and choose "Project Options" (or just press ALT+P).
- In the "Further object files or linker options" field, enter: `-lbgi -lgdi32`
- Click "OK".
- Follow [step 4](#), [step 5](#) and [step 6](#) of "Using Dev-C++".

*BGI and WinBGIm differences:*

WinBGIm is a superset of BGI and as such may have functions and features with which you are unfamiliar. See Michael Main's [BGI Documentation](#) for more information.

*Test code:*

Just to make sure you've got everything set up correctly, try this test code in a new Dev-C++ WinBGIm project:

```
#include <winbgim.h>

int main()
{
    initwindow(400,300); //open a 400x300 graphics window
    moveto(0,0);
    lineto(50,50);
    while(!kbhit());    //wait for user to press a key
    closegraph();      //close graphics window
    return 0;
}
```

If you've done everything correctly, you should get a simple graphic that closes when the user presses a key.

### Where is <sstream> (string streams)?

The compiler included with Dev-C++ was the latest stable version of the MinGW implementation of GCC (2.95.2). Unfortunately, sstream was not yet implemented in that version. Fortunately, the latest MinGW GCC snapshots (test versions) starting with gcc-2.95.3-20010723 do implement sstream. Source and binaries can be found at [SourceForge](#). **NOTE:** You cannot directly extract the SourceForge archives into the directory where you installed Dev-C++ due to the differences in where Dev-C++ stores its files. I have created some ZIP files provided below containing the files in latest snapshots, but in the directory structure expected by Dev-C++. To install, download the version that you want and extract into the directory where you installed Dev-C++ (usually, C:\Dev-C++).

You do not need to download all of the ZIP files below, just the version you want to install. I have also provided the latest stable version in case you have any problems with the test "snapshot" versions.

[gcc-2.95.2-3.zip](#) (latest stable version [2.95.2-3]--does NOT include sstream)

[gcc-2.95.3-20010723.zip](#) (2.95.3 07/23/2001 snapshot)

[gcc-2.95.3-20010828.zip](#) (2.95.3 08/28/2001 snapshot)

### Page change log:

*10/10/2001:*

Added sstream info in FAQ.

*4/16/2001:*

Fixed error in winbgim.h (conio.h conflict).

*3/23/2001:*

Updated WinBGIm procedures.

*3/21/2001:*

Started FAQ section with string and BGI info.

*3/19/2001:*

Updated string info.

*3/16/2001:*

Added string info.

---

That's it for now.

I am not a C++ expert nor do I know every feature or workaround of Dev-C++, but if you have any questions, feel free to email me at [uniqueness\\_template@hotmail.com](mailto:uniqueness_template@hotmail.com)

Happy coding!

---



Copyright ©2001 [Mike Serrano](#).  
Last Modified 11:43 AM 10/10/2001