Dev-C++ Tutorial for CSC 161 Students (Maintained by Michael Serrano)

Update (9/12/2006):

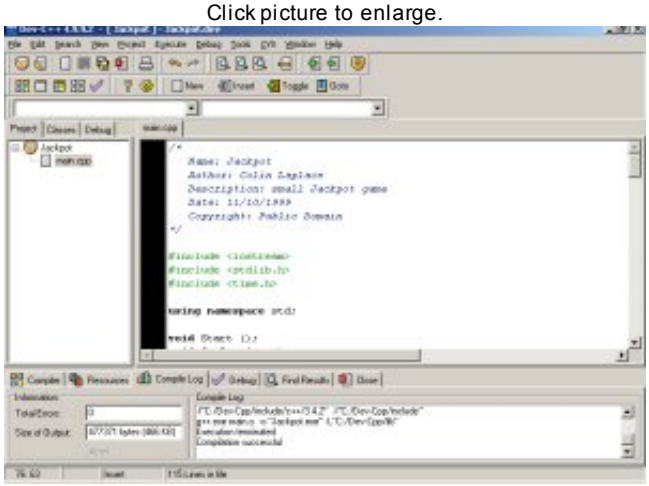
It's been quite some time since I've updated this page; in fact, I lost track of it for awhile. In the hopes that it may still be useful, I've updated this tutorial to be current with the latest version of Dev-C++ (as of this writing, Version 5 Beta 9--a.k.a. Version 4.9.9.2). ‹‹ Mike ››

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.*

Click picture to enlarge.



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-Cpp, as it will make it easier to later install add-ons or upgrades.

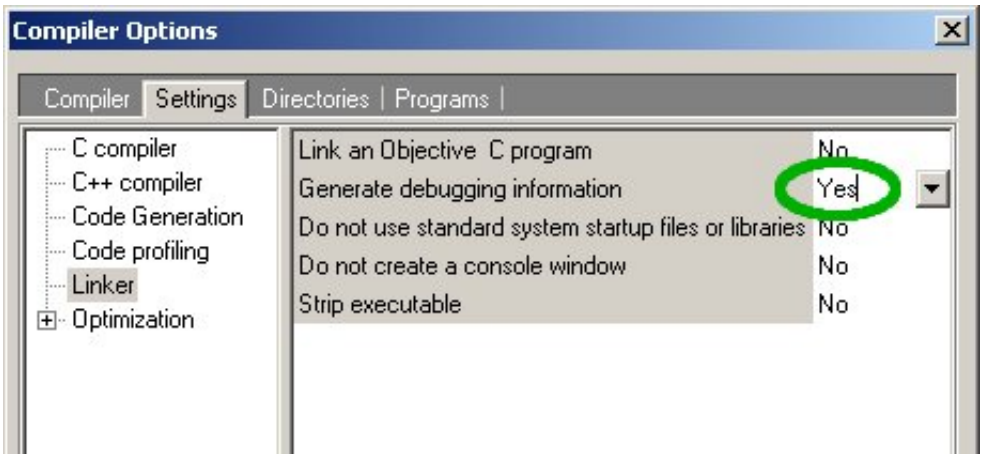
Using Dev-C++

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 modify one of the default settings to allow you to use the debugger with your programs.

- Go to the "`Tools`" menu and select "`Compiler Options`".
- In the "`Settings`" tab, click on "Linker" in the left panel, and change "`Generate debugging information`" to "`Yes`":



- Click "`OK`".

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...`".
- Choose "`Empty Project`" and make sure "`C++ project`" is selected.
  Here you will also 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 final executable.
- Once you have entered a name for your project, click "`OK`".
- Dev-C++ will now ask you where to save your project.

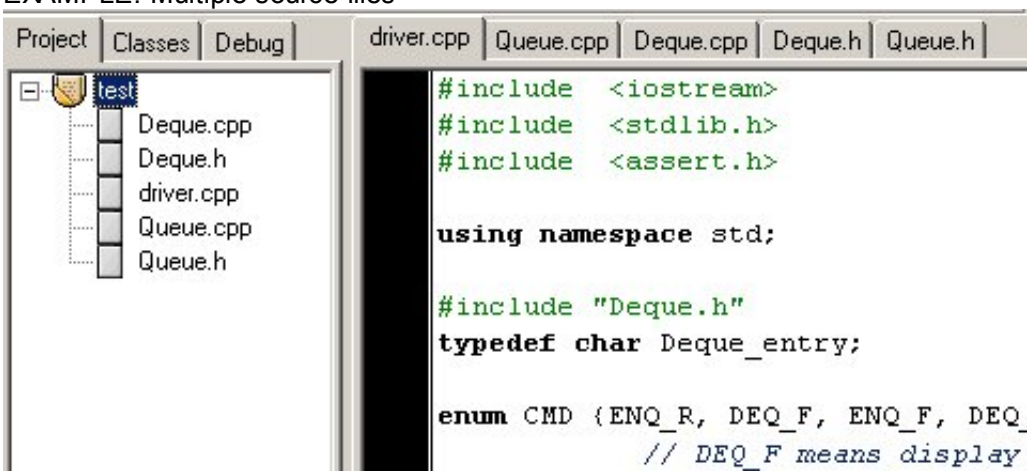
Step 3: Create/add source file(s).
You can add empty source files one of two ways:

- Go to the "`File`" menu and select "`New Source File`" (or just press CTRL+N) OR
- Go to the "`Project`" menu and select "`New File`".
  Note that Dev-C++ will not ask for a filename for any new 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
- Right-click on the project name in the left-hand panel and select "`Add to Project`".

EXAMPLE: Multiple source files



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

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.

Once your project successfully compiles, the "`Compile Progress`" dialog box will have a status of "`Done`". At this point, you may click "`Close`".

Step 5: Execute.

You can now run your program.

- Go to the "`Execute`" menu, choose "`Run`".

Note: to pass command-line parameters to your program, go to the "`Execute`" menu, choose "`Parameters`" and type in any paramaters you wish to pass.

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. Dev-C++'s basic debugger functions are controlled via the "`Debug`" tab at the bottom of the screen; more advanced functions are available in the "`Debug`" menu.

> *Using the debugger:*
> The various features of the debugger are pretty obvious. Click the "`Run to cursor`" icon to run your program and pause at the current source code cursor location; Click "`Next Step`" to step through the code; Click "`Add Watch`" to monitor variables.
> Setting breakpoints is as easy as clicking in the black space next to the line in the source code.
> See the Dev-C++ help topic "Debugging Your Program" for more information.

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

Why do I keep getting errors about "cout", "cin", and "endl" being undeclared?

It has to do with namespaces. You need to add the following line after the includes of your implementation (.cpp) files:

```
using namespace std;
```

How do I use the C++ string class?
Again, it probably has to do with namespaces. First of all, make sure you "#include <string>" (not string.h). Next, make sure you add "using namespace std;" after your includes.

Example:

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

using namespace std;

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

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 to be used under MinGW (and therefore Dev-C++) which he has aptly named WinBGIm.

The files we need are:

graphics.h (download to C:\Dev-Cpp\include)
libbgi.a (download to C:\Dev-Cpp\lib)

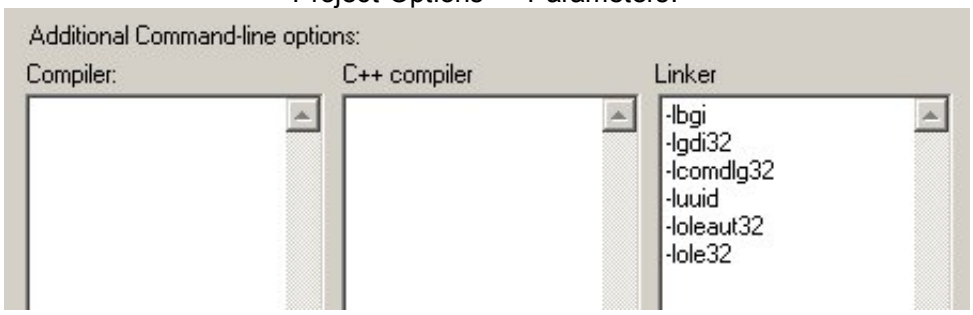
After you have downloaded the files to the correct locations, you can now use WinBGIm's graphic.h as you would Borland's graphics.h with a few caveats.

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).
- Go to the "Parameters" tab
- In the "Linker" field, enter the following text:

```
-lbgi
-lgdi32
-lcomdlg32
-luuid
-loleaut32
-lole32
```

Project Options -> Parameters:



- 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 <graphics.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)?
In previous versions of Dev-C++, the included compiler did not implement sstream. Fortunately, new versions of Dev-C++ do not have this problem. Upgrade to the latest version of Dev-C++.

Page change log:

9/12/2006:
Overhauled page to be current with Dev-C++ 5 Beta 9 (4.9.9.2).

10/10/2001:
Added sstream info in FAQ.

4/16/2001:
Fixed error in winbgim.h (conio.h confict).

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 Dev-C++ expert by any means (in fact, I now use Apple's Xcode), but if you have any questions, feel free to email me at webmaster@uniqueness-template.com

Happy coding!

Last Modified 3:13 PM 9/12/2006