**Name: Mukul Dev**

**REG NO.:- 13BIT0269**

**DATE:-27/08/15**

# INSTRUCTIONS

## 1. How do we 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:

Step1: 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++".

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").

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++".***

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 parameters’ you wish to pass.

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.