# Code::Blocks Student Manual

# Lawrence Goetz, Network Administrator

# Yedidyah Langsam, Professor

# and

# Theodore Raphan, Distinguished Professor

# Dept. of Computer and Information Science Brooklyn College of CUNY © 2006-2008

## Version 8.02

## **Table of Contents**

Introduction	3
Installation of the Code::Blocks IDE and the MinGW Compiler	4
Customization	
First Project	12
Adding Files To Your Project	20
Debugging a Program	
Appendix A: Installing Code::Blocks under Mac OS X and Linux	

#### Introduction

Through the aid of a **compiler**, a program written in a computer language, such as C++, is turned into machine code, which is executed on the computer. However, going from an idea to a program that works successfully takes a lot of time and effort. It may take several rewrites of code to get the program to work correctly. To accomplish this, students must learn a disciplined approach to organizing the code and learn how to trace their programs. The purpose of this manual is to help the student develop the skills to organize program coding and develop sound techniques for finding and isolating errors. Here you will learn how to trace the code step by step, so that it becomes clear where the problem is and why your program does not execute properly. This is called **debugging** the program. Hand tracing is useful in helping beginners understand where the bugs are and correct the program appropriately. Automatic tools have also been developed to help you trace programs that you have written and will be an important tool as your programs become more complex. This type of tool is called a **debugger**. A debugger lets you pause a program, while it is in the middle of running, and watch what is going on. Some debuggers work as command-line line debuggers, but newer debuggers have a nice graphical user interface, which is useful in helping you watch variables that you have defined as the program executes. The graphically-based debugger environment is part of what is called the **Integrated Development Environment (IDE)**. The purpose of these notes is to introduce you to this environment and help you learn how to use it as you develop and hone your programming skills.

A debugger cannot solve your problems for you. It is merely a tool to assist you when programming. You should first attempt to read over your code and using paper and pencil analyze the code to get an understanding of what is going on. Once you have gotten an idea of where in your code you have an error, you can then set the debugger to **watch** certain variables in your program. Watching your code will show you step by step how your program is being executed.

The debugger that you will use is part of an Open Source free IDE called **Code::Blocks**, which we have found easy to use and is described in these notes. Code::Blocks has a C++ editor and compiler. It will allow you to create and test your programs from one easy to use application. We hope these notes will assist you in making programming more enjoyable and help you develop better programming skills.

You may find additional information regarding Code::Blocks at: <u>http://www.codeblocks.org/</u> A complete manual for Code::Blocks is available here: <u>http://www.codeblocks.org/user-manual</u>

### **Installation of Code Blocks**

#### **Step 1: Download the Software**

In order to install the Code::Blocks IDE as well as the MinGW compiler, you must download it. If you are using either **Windows XP** or **Windows Vista** download the installation programs from here:

http://downloads.sourceforge.net/codeblocks/codeblocks-8.02mingw-setup.exe

Save the file to your hard disk and remember its location. Proceed to the next page in order to continue the installation.

If you are using Mac OS X or Linux, please see Appendix I for installation instructions.

#### **Step 2: Install the Software**

Next, open (click on) CodeBlocks install file and the CodeBlocks Setup will begin installing as follows:



Click Next.

CodeBlocks Setup	
-	<b>License Agreement</b> Please review the license terms before installing CodeBlocks.
Press Page Down to se	e the rest of the agreement.
	RAL PUBLIC LICENSE
Everyone is permitted	ee Software Foundation, Inc. < <u>http://fsf.org/</u> > to copy and distribute verbatim copies nt, but changing it is not allowed.
Prea	nble
The GNU General Pub software and other kir	lic License is a free, copyleft license for ds of works.
If you accept the terms agreement to install Co	of the agreement, click I Agree to continue. You must accept the deBlocks.
Nullsoft Install System v2,	
	< <u>B</u> ack I <u>A</u> gree Cancel

#### Select I Agree.

CodeBlocks Setup		×
	oose Components hoose which features of CodeBlocks you want to install.	
Check the components you war install. Click Next to continue.	nt to install and uncheck the components you don't want to	
Select the type of install:	Standard: Core plugins, core tools, and core lexers	
Or, select the optional components you wish to install:	Oefault install     Ontrib Plugins     C::B Share Config     MinGW Compiler Suite	
	Description	
Space required: 102.5MB	Position your mouse over a component to see its description,	
Nullsoft Install System v2.35		_
	< <u>B</u> ack <u>N</u> ext > Cancel	

Take the default settings by pressing Next.

🔲 CodeBlocks Setu	p	- 🗆 🗵
-	<b>Choose Install Location</b> Choose the folder in which to install CodeBlocks.	
	deBlocks in the following folder. To install in a different folder, click Bro folder. Click Install to start the installation.	wse
Destination Folde	··	]
Space required: 10: Space available: 7.		
Nullsoft Install System	v2,35	icel

Take the default folder to install CodeBlocks to and then select Install.

# **Step 3: Customization of the Code::Blocks User Interface** (**Optional**)

The following steps will enable you to customize your IDE so that it is will be consistent with what your instructor will be using in class:

- 1. Configure the editor:
  - a. Choose *Editor* from the *Settings* Menu
  - b. Under the General Setting tab
    - i. Change the font size to 10 or 12 point (Use the *Choose* button.)
    - ii. Under Other Options place a check mark the following options:
      - "Show line numbers"
      - "highlight line under caret"
      - "highlight occurrences"

Your screen should now look like the figure:

Configure editor		
	General settings	
	Font This is sample text Default encoding when opening files:	Choose
General settings	TAB options Use TAB character TAB indents TAB size in spaces:	End-of-line options Show end-of-line chars Strip trailing blanks End files with blank line Ensure consistent EOLs End-of-line mode: CR LF V
Folding Margins and caret	Indent options  Auto indent  Smart indent Backspace unindents Show indentation guides Show spaces: No	Other options Word wrap Use POSIX style for RegEx searches Use Advanced RegEx searches Show line numbers Highlight line under caret Home key always moves caret to fir Highlight occurrences
	Editor title is the file's <ul> <li>name only (no path information)</li> <li>relative filename (to the project file)</li> </ul> OK Cancel	

- c. Select the Margins and Caret tab on the right of the window.
  - i. Choose "Visible line" from the Right margin hint
  - ii. Set the *Hint Column* to 72

Your screen should now look like the figure:

Configure editor		X
	Margins and caret	
	Left margin Width for line numbers (in chars): S Dynamic : Add/remove breakpoints by left-clicking Right margin	setting
General settings	Right margin hint: Visible line   Colour: Image: Colour in the second seco	
Folding	Width: 1	
Margins and caret	Period (in milliseconds): 0 500	500
	OK Cancel	

- d. Choose the Source formatter Caret tab on the right of the window.
  - i. Select *K&R* from the style menu (Note: Your instructor may use a different style if so, examine each style and choose the style that matches your instructor's preferred indentation.)

Configure editor			×
	Source fo	ormatter	
Mouse Drag Scrolling Mouse Drag Scrolling Keyboard shortcuts Keyboard shortcuts wxSmith settings Source formatter	Style Indentation Fo	<pre>amespace foospace {     int Foo() {         if (isBar) {             bar();             return 1;         } else             return 0;     } </pre>	
	ОК	Cancel	

Your screen should now look like the figure:

e. Choose OK in order to save your customizations.

- 2. Configure the *Help* files
  - a. Download the C++ help file from

http://onnerby.se/~daniel/chm/cppreference.com/cpp.chm

- b. Save the cpp.htm file to your c:/Program Files/Codeblocks directory
- c. After saving the file, right-click on the file, choose *Properties*, and unblock the file so that it will be accessible to the Code::Blocks IDE.
- d. Choose Environment from the Settings Menu
- e. On the tab to the left of the window scroll down to the Help Files section
- f. Choose Add and enter "C++ Help"
- g. Use the file browser button in to locate the file you have just saved. (It should be at C:\Program Files\CodeBlocks\cpp.chm)
- h. Place a check mark in the this is the default help file (Shortcut: F1)

Your screen should now look like the figure:

nvironment se		
	Help files	
*	C++ Help	Add Rename Delete
HeaderFixup configuration		Up
2		
Help files		
class foo ( ii ab a; vo. ();	C:\Program Files\CodeBlocks\cpp.chm	
Thread search	TIP: \$(keyword) will be replaced by the word under the cursor	
micau scarch	✓ This is the default help file (shortcut: F1)	
	This line represents a full command to be executed	
Chiefe and Cont	Open this file with the embedded help viewer (only for HTML files)	
for Contra	Preserve keyword case	
To-do list	Default keyword value:	
	OK Cancel	

i. Choose OK in order to save your customizations.

By pressing F1 you will now be able to obtain help on the word under your cursor.

## First Project

After you have finished downloading and setting up the Code::Blocks system, you can be in to write code. Code::Blocks creates what is called a **Workspace** to keep track of the **project** you are working on. It is possible for you to be working on multiple projects within your workspace. A **project** is a collection of one or more **source** (as well as **header**) files. **Source** files are the files that contain the source code for your program. If you are developing a C++ program, you are writing C++ source code (.cpp files). **Header** files are used when you are creating **library** files (.h files). A **library** is a collection of **functions** that are called to perform specific tasks, such as doing math, etc.

Setting up a **project** allows you to keep track of all the files in an organized way. When first starting out in computer programming, generally your projects will consist of a single source file. However as you gain experience and work on more complex projects, you will have projects containing many source files and dealing with header files as well.

To create a project, click on the **File** pull-down menu, open **New** and then **Project**.



This will bring up the **New from template** window. Opening (clicking on) **Console Application** will then allow you to write a program on the console. The other application are for developing more advanced types of applications. After selecting **Console application**, click on the Go button to begin using the Console Application Wizard.



Console application	×
Console	Welcome to the new console application wizard! This wizard will guide you to create a new console application When you 're ready to proceed, please click "Next"
	< <u>B</u> ack. <u>N</u> ext > <u>Cancel</u>

Press Next to go to the next step.



Start by filling in the Project Title. You will notice that the Project Filename automatically becomes the same name. If you wish, you can change the filename, but for simplicity leave it as is. To specify the location of the folder to contain the project, click on the "…" button (selected in the picture above) and browse to a folder on your drive to store the project. Generally, you can save it in My Documents.

Press Ok after selecting My Documents

Console application	×
🐻 Console	Please select the compiler to use and which configurations you want enabled in your project.
	Compiler:
	Create "Debug" configuration: Debug
	"Debug" options Output dir.: bin\Debug
	Objects output dir.: obj\Debug
	Create "Release" configuration: Release
	"Release" options Output dir.: bin\Release
	Objects output dir.: obj\Release
	< <u>B</u> ack <u>N</u> ext > <u>C</u> ancel

When the directory for your project has been selected, the system will return to the **Console application**. As shown, Code Blocks will create a directory called First Program (Project Title) and returns your selected directory in **Folder to create project in**. Inside that directory will be the **Project filename** (First Program) and a resulting filename, which contains a Code Block Project file (.cbp) named First Program.cbp. The project title and project filename in this case are the same. However, they need not be the same and these names can be altered. Click on the Next Button when done.

The next window to pop up will be the Compiler screen. This specifies where the **Debug** and **Release** compiled versions of your program will be placed. Leave this setting alone and press Next.

Console application		×
Console	Please select the language you want to use.	
	< <u>B</u> ack <u>F</u> inish <u>C</u> ance	

The next window allows you to choose the language that you will use. Select the language as C++, then press Finish.



The system will then return to the **[First Program]** window and you are ready to write your program. It should be noted that the Build target is **Debug**, which will allow you to use the debugger to find errors. In the **Management** area of the screen (Shift-F2 toggles the Management display), you will see the files that are part of the project in the **Projects** tab. To see the source files, click on the plus [+]'s to expand the **Workspace** and its subdirectories.

📒 [First Program] - Code::Block:	s v1.0			_	
<u>File E</u> dit <u>V</u> iew Sea <u>r</u> ch <u>P</u> roject	<u>Build D</u> ebug wxSmith	<u>T</u> ools P <u>l</u> ugins	Settings E	<u>H</u> elp	
i 🗋 🖆 🖀 🚔 🔧 🖓	6 🛍 🔍 🕾				
:			<b>T</b>		
🕴 💊 🜔 🦚 🐼 🛛 Build target	Debug	•			
🛯 📲 😚 🐍 🏷 🔗 🖉	I. 1.				
Management 2	<				
Projects Symbols 4					
E-0 Workspace					
First Program					
Sources					
i main.cpp					
Open files list	< Messages				×
🔁 Opened Files	🚺 Code::Blocks	🔌 Code::Block	s Debug	·	4 ►
	Loading: AStylePlugin Au				
	ClassWizard CodeComple copystrings Debugger Fi				
	EnvVars Source Exporte	r HelpPlugin cbk	eyBinder Scr	riptedWizard	
	ToDoList wxSmith wxSmi Running startup script	ithMime w×Smith	hWizard Wind	dowsXPLookNFeel	
					-
Welcome to					

Under Sources, there is a file called main.cpp, which is automatically created for you when you build a console application.

## Adding Files To Your Project

If you have a project with additional existing files, go to the Project menu and select "Add files." This will bring in the files associated with your program. You also have the option to **Remove files**, performing **Build options** and to **Set programs' arguments...**.

<mark>-</mark> [First Program] - Coo	le::Blocks	v1.0						>	<
<u>File E</u> dit <u>V</u> iew Sea <u>r</u> ch	Project B	uild <u>D</u> ebug	wxSmith	<u>T</u> ools	Plugins	<u>S</u> ettings	Help		
Image: Second secon	Add file: Add file: Remove Project	5 s recursively. : files tree grams' argum tions	••		1	<u> </u>			
Open files list  Opened Files  Add files to t	×	Loading: AS ClassWizard copystrings EnvVars So	d CodeCom ; Debugger urce Export xSmith wxS	Autosav pletion ( FilesExt :er Helpi	ve BYOGa CodeSnipj censionHa Plugin cbł	mes CBPro pets Codes ndler DevP KeyBinder S	filer CB_Koc itatistics Co akUpdater I 5criptedWiza indowsXPLo	d ► mpiler DragScroll ard	×

Add files to proje	ect	?×
Look jn:	🔄 First Program 💽 🔶 📸 📰 -	
History Desktop My Documents My Computer	First Program.cbp	
My Network P		<u>O</u> pen Cancel

Clicking on **Add files to project,** will bring up a window so you can browse to where your files that you wish to add are. Select any additional file you want to add and press Open. The file will then be added to your project.

If you are creating a new file, you can use the pull-down **File** menu and open an **empty file**.

📒 [First Program] - Code::Blocks v1.0	
File Edit View Search Project Build Debu	g wxSmith <u>T</u> ools Plugins <u>S</u> ettings <u>H</u> elp
New 🕨	Empty file Ctrl-Shift-N
😭 Open Ctrl-O	Project
Open default workspace	Build target
Recent projects	File
Recent files	Custom
Import project	From user template
Save Ctrl-5	
😏 Save as	
Save all files Ctrl-Shift-S	
Save project	
Sa <u>v</u> e project as	
Save project as user-template	
Save all projects	×
Save <u>w</u> orkspace	te::Blocks 🔌 Code::Blocks Debug 🛛 🔹 🕨
Save wor <u>k</u> space as	StylePlugin Autosave BYOGames CBProfiler CB_Koders
Close workspace	rd CodeCompletion CodeSnippets CodeStatistics Compiler Is Debugger FilesExtensionHandler DevPakUpdater DragScroll
S glose file Ctrl-W	ource Exporter HelpPlugin cbKeyBinder ScriptedWizard
Close all files Ctrl-Shift-W	cartup script
Close project	
Close all projects	

File, New, Empty file.

You will be asked if you want to add this file to the project.



Choose Yes.

Code Blocks will ask for a file name to save the file as:

Save file					<u>? ×</u>
Save jn:	🔄 First Program		•	+ 🗈 💣 🎟+	
History History Desktop My Documents My Computer	C main.cpp				
My Network P	File <u>n</u> ame: Save as <u>t</u> ype:	sample.cpp C/C++ files		•	<u>S</u> ave Cancel

Give a name to the file. Pick a name that is related to the content of the file. Here it is called sample.cpp. C++ files need to be of the type cpp. Press Save to save the file.

Multiple selection Select the targets this file should belong to:	<u>_0×</u>
<ul> <li>✓ Debug</li> <li>✓ Release</li> </ul>	Wildcard select Toggle selection Select All Deselect All Selected: 2
OK Cancel	]

Press Select All to have this file saved as both Debug & Release targets. Press OK when done.

A target is a type of compiled version. You can work with a debug target, which will allow you to test the program using a debugger. A debug target will be large in size, because it has extra information in it to allow you to test for errors. A release target is smaller in size, because it does not have the debugging information. When you are ready to give other people (such as your Instructor) your finished program, you should give them the release target.

🗧 sample.cpp [First Program] ·	- Cod	le::Blocks v1.0			
<u>File E</u> dit <u>V</u> iew Sea <u>r</u> ch <u>P</u> roject	Bui	ld <u>D</u> ebug wxSmith <u>T</u> ools	Pļugins <u>S</u> et	tings <u>H</u> elp:	
i 🗋 🐸 🖬 🖧 🔦 🔖 🚽	ſß	🖺 🔍 🕵			
				<b>v</b>	
🕴 💊 🌔 🦚 🗔 🛛 Build targ	et: D	ebug 💌	]		
i 💵 🖷 😚 🐍 🏷 🔗 😣		i.			
Management	×	sample.cpp			4 Þ ×
Projects     Symbols     4       Image: Workspace     Image: Workspace     Image: Workspace       Image: Image: Workspace     Image: Workspace       Image: Image	▶ 	1 2			
Open files list	×	Messages			×
Dened Files		🚺 Code::Blocks 🔌	iode::Blocks De	ebug	۹ ک
		Loading: AStylePlugin Autosa ClassWizard CodeCompletion copystrings Debugger FilesEx EnvVars Source Exporter Help ToDoList wxSmith wxSmithMir Running startup script	CodeSnippets tensionHandler pPlugin cbKeyB ne wxSmithWiz	CodeStatistic r DevPakUpda inder Scripted	s Compiler ater DragScroll dWizard XPLookNFeel
E:\Document default	Line	1, Column 1	Insert		Read/Write

The Sources now has sample.cpp as a source file in addition to the main.cpp file.

Since the sample.cpp is not needed for your project, please remove it.

sample.cpp [First Program	m] - Code::Blocks v1.0				_ 0	×
	ject Build Debug wxSmi	h Tools:	Plugins Sel	ttings Help		_
	Add files					
	Add files recursively					_
	Remove files					
🗄 🛇 🕞 🚱 🔽 🖉 🖡	Project tree	,	Ŀ			
i 💵 🧏 😚 🚠 🏷 🦿	et programs' arguments					
Management	Build options					x
Projects Symbols F	Properties					
🖃 🕜 Workspace	2					
🖻 📇 First Program						
🖻 🔁 Sources						
main.cpp						
i sample.cpp						
Open files list	× Messages					×
Dened Files	📝 Code::Block	s 🚺 🤇	iode::Blocks De	ebug	٩	►
	Loading: AStylePlug					
	ClassWizard CodeC copystrings Debug					
	EnvVars Source Ex	orter Help	oPlugin cbKeyB	inder Scripter	dWizard	
	ToDoList wxSmith v		ne wxSmithWiz	ard Windows	XPLookNFeel	
	Running startup sc	ipc				•
Remove files default	Line 1, Column 1		Insert		Read/Write	

From the Project menu select, Remove files.

Multiple selection	
Select files to remove from First Program:	
main.cpp	Wildcard select
sample.cpp	Toggle selection
	Select All
	Deselect All
	Selected: 1
OK Cancel	

Place a check mark next to any file(s) that you wish to remove. Press OK when you are done.



You will need to confirm that you wish to remove the file(s). Press *Yes*, if you are sure you want to remove them. Otherwise press *No*.

First Program] - Code::Blocks v	/1.0			
<u>File E</u> dit <u>V</u> iew Sea <u>r</u> ch <u>Project</u> <u>B</u> u		Plugins Setting		
1 🖻 🖬 🖨 🔧 🗟 t				
:		~		
i 🤝 ┝ 🦚 🐼 🛛 Build target:	Debug 💌	1		
i 💵 🖷 😚 🐍 🏷 🖑 😆 🛄	i.	_		
Management ×				
Projects Symbols				
🖃 😡 Workspace				
🗄 💾 First Program				
🖻 🔁 Sources				
i main.cpp				
Open files list 🛛 🗙	Messages			×
Dened Files	🚺 Code::Blocks 🔌 🕬	ode::Blocks Debug	,	< ▶
	Loading: AStylePlugin Autosav			
	ClassWizard CodeCompletion copystrings Debugger FilesEx			roll
	EnvVars Source Exporter Help			- On
	ToDoList wxSmith wxSmithMin			el 🚺
	Running startup script			-
Welcome to	<i>.</i>			

You will now see an updated listing of the Sources in your file. You should now see only Main.cpp. In the Open Files list, there may be a file called **!Untitled. Please ignore this.** 

To edit a file from your project, double click on it's name from **Sources** and it will appear in the window with line numbers. You can now edit the file and prepare your program.



🗧 main.cpp [First Program] -	- Code::Blocks v1.0	
<u>File E</u> dit <u>V</u> iew Sea <u>r</u> ch <u>P</u> roje	ect <u>B</u> uild <u>D</u> ebug wxSmith <u>T</u> ools	: Plugins <u>S</u> ettings <u>H</u> elp
	dd files	
	dd files recursively emove files	
i 🛇 🕨 🧊 🐼 E 🔐	oject tree	
	et programs' arguments	
Management Bu	uild options	4 Þ ×
Projects 4 Pr	operties <iost< td=""><td>tream&gt;</td></iost<>	tream>
Workspace	2 3 int main() 4 ⊟ ( 5   std::cout 6   return 0; 7 -} 8	<< "Hello world!" << std::endl;
Open files list 🛛 🗙	•	
Dened Files	Loading: AStylePlugin Autosave BYO CodeCompletion CodeSnippets Code	
Set the proje WINDOWS-1252	Line 1, Column 1	Insert Read/Write

In order to check that Debug is running, you can use the **Project** pull-down menu and click on **Build Options.** 

Project build options		
Project build options	Selected compiler         GNU GCC Compiler         Compiler       Linker         Directories       Commands       Custom variables       "Make" commands         Policy:       Append target options to project options       Image: Compiler Flags       Other options       Image: Compiler Flags         Compiler Flags       Other options       #defines       Image: Compiler Flags       Other options       Image: Compiler Flags         Categories:               Categories:                Categories:	
	<u>O</u> K <u>C</u> ancel	

When this is done, the **Project Build options** window will come up. Make sure that the **Produce debugging symbols** [-g] is checked.

Press OK when done.



After clicking on done, the system will return to Main.cpp. When testing your code, make sure that **Debug is selected as the target to use.** This way when you **Compile** your program, you will have a Debug version available. To compile a file means to take the instructions that you have written and translate it into machine code for the computer to understand.



Compile your file from the Build pull-down menu by clicking on Compile current file (Ctrl-Shift-F9).



Test the project from the Build Pull-down menu, by clicking on Build and Run. This step will **build** an executable file for you. A project build will take the compiled versions of your source files and combine them into one program.

You are able to press F9, which is a keyboard shortcut that will build your project and run it at the same time. As you gain more experience with the system, it will be easier to just press F9 to Build & Run your program. The Message window will indicate if there are any errors during a compile or build phase.



This is the output from my first program. Notice that besides displaying "Hello world!" it also says to "Press any key to continue" with the program paused. Pressing any key will exit the program.

If you execute the program by going to a console window you will not see the "Press any key to continue" message:



Notice that there are double quotes around the file name. This is because there is a space in the name. If

you execute this program by double clicking on it's icon, the program would close right away. That is because the pause statement is only done when you run your program in Code Blocks.

When you are done, save all your files by pulling down the File menu and clicking on Save all files.

🗕 main.cpp [First Program] - Code::Blocks v1.0	
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help	
New K	
🖆 Open Ctrl-O 🔽 main() : int	
Open default workspace	
Recent projects	
Recent files	
Import project	⊳×
Eave Ctrl-5	
Save as	
Save all files Ctrl-Shift-S std::cout << "Hello world!" << std::	ondl ·
Save project Court of the section of	
Sa <u>v</u> e project as	
Save project as user-template	
Save all projects	
Save workspace	×
Save workspace as	
Close workspace kpoints	
e and version: GNU gdb 6.3	
Close file Ctrl-W ~1/goetz/MYDOCU~1/FIRSTP~1/main.cpp:5	
Close all files Ctri-shirt-w	•
Close project Close all projects 11 Insert Read/Writ	e //

Now you can select to save the project:

<mark>–</mark> main.cpp [First Program]	- Code::Blo	cks v1	.0					_ 🗆 ×
<u>File Edit View Search Proje</u>	ect <u>B</u> uild	<u>D</u> ebug	wxSmith	<u>T</u> ools	Plugins	<u>S</u> ettings	Help	
New		•	B					
🔁 Open	Ctrl	• [				🔽 m	ain() : ir	it
Open default workspace		F		-	[			
<u>R</u> ecent projects		- • F						
<u>R</u> ecent files		-> _						
Import project		→						$\triangleleft \triangleright \mathbf{x}$
🔚 Save	Ctrl		include <	(iost)	ream≻			
🛃 Save <u>a</u> s			t main()					
Save all files	Ctrl-Shift-9	5	_					
- Course our side t			std::c return		<< "Hel	llo worl	d!" ≺	< std::endl;
Save project		_						
Save project as Save project as user-template								
Save all projects								F
Save aj projects		-						×
Save <u>w</u> orkspace		- E	🐣 Build me	econec	; (	) Debugg	er	
Save wor <u>k</u> space as		H	A Dalla Ille	sssayes	' 🚬 🍢	Debugg		
Close workspace			points					<b>_</b>
😧 Close file	Ctrl-		e and version: GNU gdb 6.3 ~1/goetz/MYDOCU~1/FIRSTP~1/main.cpp:5					
Close all files	Ctrl-Shift-V		shed wit					
Close project	Car Driftery							-
Close all projects		1	1		Insert			Read/Write

When you exit the program, you may be asked to save the **Workspace** and the **Layout**. The Layout refers to the placement of various windows that you may have positioned. Generally you would select to Save the Layout (unless you know you really do not want it saved). The Workspace refers to the projects you are working on. It is possible for you to be working on multiple projects within your workspace. Saving your workspace will allow you to return to the same set of projects when you next open Code Blocks.

#### To open a project



From the File menu select Open.
Open file		? ×
Look jn:	🔄 First Program 💽 🔶 📸 📰 🗸	
History Desktop My Documents My Computer	⊇ obj ≸ First Program.cbp	
My Network P		<u>Open</u> Cancel

From the **Files of type:** in the window, select "Code::Blocks project files" and then select the .cbp file pertaining to your program.

Press Open when done.



The project has reopened. You can get more space to see your program, if you close the **Messages** window. Pressing **F2** toggles the display of the messages. The **Messages** window has been turned off for the remainder of this tutorial, to allow more space to be visible on the screen.

*Note*: You may also open a project directly from Windows Explorer by double-clicking on the file with the .cbp extension.

# Debugging a Program

As your programs become more complicated, there will be a need to trace the program execution step by step or place break points where you wish the program to pause. This is where a debugger is utilized. A debugger can pause your program and you can watch the values of the variables that you have defined.

The following is a sample program that can be traced "line by line" while watching what happens as each line of code is executed.

Hain.cpp [debug] - Code::B	locks v1.0				
	t <u>B</u> uild <u>D</u> e	bug wxSmith <u>T</u> ools	Plugins <u>S</u> etti	ngs <u>H</u> elp	
i 🗋 🛤 🖨 🔦 🔖 🛛	288	O. R.			
				-	
<u>:</u> ]				main() : int	
🕴 😒 🌔 🧊 🐼 🖉 Build tar	get: Debug	•	i 📲 📲 🤇	ን ብ ብ የ	š 😧 🗔 i.
Management ×	main.cpp	]			4 Þ 🗙
Projects 4 🕨	1	#include <iostr< th=""><th>eam≻</th><th></th><th></th></iostr<>	eam≻		
🖃 🕜 Workspace	2	using namespace	std;		
🖃 📕 debug	3				
🖻 🔁 Sources	4	int addem(int,	int);		
main.cpp	5				
	6 7	int addem(int a int c;	(, int b) (		
	8	Int c,			
	9	c=a+b;			
	10				
	11	return c;			
Open files list X	12	L)			
	13				
Opened Files	14	int main()			
i 📄 main.cpp	15 16	□ { int x=5, y=			
	10	Inc x-5, y-	2, 21		
	18	z=addem(x,y	);		
	19	cout << z <			
	20				
	21	return 0;			
	22	L)			
	23				
	•				▶
E:\Document: WINDOWS-1252	Line 16, C	Column 21	Insert	R	ead/Write

First, it is necessary to set a place in the code to have the program pause. This is done by using the **Debug** pull-down menu and clicking on **Run to Cursor. The cursor should be over the first line of code where you wish to start the tracing process. This starts the debugging process.** 



The next step in debugging a program is to tell the program when to stop running so you can inspect the results. To do this, place the cursor over the line where you want your program to stop. For example, the cursor was placed at line 18 (which is hidden behind the menu). This is called a **Breakpoint.** Now you can instruct the debugger to run the program up to the cursor's position (line number).



The program will generate a blank window. It is blank, since that program has yet to execute any line that displays something.

<mark>–</mark> main.cpp [debug] - Code::	Blocks v1.	.0			
File Edit View Search Proje	ect Build	Debug wxSmith	Tools Plugins	Settings	Help
🗋 🗃 📓 📥   🔦 🔖	a 6 I	_		F8	
:		🐼 Stop debugger			nain() : int
🕴 💊 🜔 🦚 🐼 🛛 Build ta	arget: Debu	Continue	C	trl-F7	🚠 ኛን 😚 🙆 🗔 🥻
Management X	main.ct	😚 Next line		F7	4 Þ 🗙
Projects 4 🕨	1	😚 Next instruction	n A	lt-F7	
🖃 😡 Workspace	2	矜 Step into	Shi	ft-F7	
en e	3 4	🔥 Step out	Ctrl-Shi	ft-F7	
main.cpp	5	Toggle breakpoir	nt	F5	
	7	🕌 Run to cursor		F4	
	8	Add symbol file			
	10	Debugging windo	ows	•	Breakpoints
	11 12	Information		•	Call stack
Open files list 🛛 🗙	13	Edit <u>w</u> atches			CPU Registers
Dened Files	14 15	Attach to proces	is		Disassembly Examine memory
	16	Detach			Running threads
	17 18	Send user comm	and to debugger		Watches
	19 20	cout «	< z << endl	2	
	20	return	<b>1</b> 0;		
	22	L)			
	23				
Watch variab WINDOWS-1252	Line 18	3, Column 1	Insert		Read/Write

To watch certain variables during the execution of the program, you should open the Watches window. This will show you the variables in your code. This is accomplished by going to the **Debug** pull-down menu and clicking on **Debugging Windows** and then **Watches**.



These are the watches that the debugger is displaying. Notice that x & y have the correct values. Variable z has not been assigned it's value on line 18 yet. The current value is a random value.



Line 18 has a yellow marker on the left side. This indicates that the program has paused on that line, which is the breakpoint.

<mark>-</mark> main.cpp [debug] - Code::E	Blocks v1.	0		
<u>File E</u> dit <u>V</u> iew Sea <u>r</u> ch <u>P</u> roje	ct <u>B</u> uild	Debug wxSmith Tools	Plugins <u>S</u> etting:	s <u>H</u> elp
1 🖻 🖻 🖨 🔦 🔖 🛛	a 🗗 I	💵 Start	F8	
1		🐼 Stop debugger		nain() : int
i 😒 👂 🥵 🐼 🛛 Build ta	rget: Debi	Continue	Ctrl-F7	🗈 改 🧭 🙆 🗔 🚛
Management X	main.cr	😚 Next line	F7	4 Þ ×
Projects 4 >	1	😚 Next instruction	Alt-F7	
- Workspace	2 3	矜 Step into	Shift-F7	
e <b>debug</b>	4	😚 Step out	Ctrl-Shift-F7	
main.cpp	5	Toggle breakpoint	F5	
	7	🕌 Run to cursor	F4	
	8	Add symbol file		
	10	Debugging windows	+	
	11 12	Information	+	
Open files list X	13	Edit <u>w</u> atches		
Dened Files	14 15	Attach to process		
	15	Detach		
	17 18	Send user command to	debugger	
	19	cout << z -	<< endl;	
	20 21	return 0;		
	22	L)		
	23			
	•			
Execute the r WINDOWS-1252	Line 18	3, Column 1	Insert	Read/Write

To determine how your program will function when calling functions such as:

```
z = addem(x,y);
```

**Step info** (Shift-F7) can be selected from the Debug pull-down menu.

<mark>–</mark> main.cpp [debug] - Code::E	Blocks v1.0	
<u>File Edit V</u> iew Sea <u>r</u> ch Proje	ect <u>B</u> uild <u>D</u> ebug wxSmith <u>T</u> ools Plugins <u>S</u> ettings <u>H</u> elp	
E 🗋 🛤 🛗 📥 🔦 🔖 🗌	🔏 🗗 😭 💁 🕵	
🕴 💊 🌔 🦚 🐼 🖉 Build ta	arget: Debug 💽 🔢 👫 🔭 👬 🤭 🕻	) 🗔 i.
Management X	main.cpp	$\triangleleft \triangleright \times$
Projects 4 🕨	l #include <iostream></iostream>	
🖃 🕜 Workspace	2 using namespace std;	
🚊 📕 debug	3	
Sources	4 int addem(int, int); 5	
i main.cpp	5 6 ⊟int addem(int a, int b) {	
	7 int c;	
	8	
	9 👂 c=a+b;	
	10	
	11 return c;	
Open files list 🛛 🗙	13	
🔁 Opened Files	14 int main()	
i 📄 main.cpp	15 🗆 (	
	16 int x=5, y=2, z;	
	17	
	18 z=addem(x,y); 19 cout << z << endl:	
	19 cout << z << endl; 20	
	21 return 0;	
	22 )	
	23	
		Þ
E:\Document: WINDOWS-1252	Line 2, Column 12 Insert Read/V	Vrite //

The next step is line 9.



The arguments *a* and *b* are shown in the Watches window. Notice that the local variable *c*, which has not been set yet, has a random value.

<mark>–</mark> main.cpp [debug] - Code::I	Blocks v1.	.0		
<u>Eile E</u> dit <u>V</u> iew Sea <u>r</u> ch <u>P</u> roje	ect <u>B</u> uild	Debug wxSmith Tools	Plugins <u>S</u> etting	is <u>H</u> elp
1 🖻 🖻 🖨 🔦 🔖	a 61	💵 Start	F8	
:		🚱 Stop debugger		addem(int a, int b) : int
i 😒 🕨 🧊 🐼 Build ta	arget: Debu	Continue	Ctrl-F7	🚠 ኛን 🐼 🗔 🖡
Management ×	main.ct	😚 Next line	F7	4 Þ 🗙
Projects 4 🕨	1	Next instruction	Alt-F7	
🖃 😡 Workspace	2	矜 Step into	Shift-F7	
e <b>debug</b>	4	😚 Step out	Ctrl-Shift-F7	
main.cpp	5	Toggle breakpoint	F5	
	7	🕌 Run to cursor	F4	
	8	Add symbol file		
	10	Debugging windows	•	
	11	Information	+	
Open files list 🛛 🗙	12 13	Edit <u>w</u> atches		
🔁 Opened Files	14	Attach to process		
i 📄 main.cpp	15 16	Detach		
	17	Send user command to	debugger	
	19	cout << z	<< endl;	
	20 21	return 0;		
	22	L Peturn 0,		
	23			
	•			Þ
Execute the r WINDOWS-1252	Line 9,	, Column 1	Insert	Read/Write

To proceed to the next line of code, select **Next line** from the Debug menu.

Pressing **F7** is a useful keyboard shortcut and will become second nature as you become familiar with the system.

Watches	×
🖃 Local variables	
c=7	
E Function Arguments	
- a = 5	
b = 2	
<u> </u>	

The debug window reflects the change of c.

📙 main.cpp [debug] - Code::	Blocks v1.	.0		
	ect <u>B</u> uild	Debug wxSmith Tools	Plugins <u>S</u> etting F8	s <u>H</u> elp
		🐼 Stop debugger		addem(int a, int b) : int
	arget: Debu	Continue	Ctrl-F7	ት. የን (ኛ 😡 🗔 🚛
Management X		😚 Next line	F7	4 Þ ×
Workspace	1 2	<ul> <li>Next instruction</li> <li>Step into</li> </ul>	Alt-F7 Shift-F7	
e- <b>P</b> debug	3 4	( Step out	Ctrl-Shift-F7	
i main.cpp	5 6 7	Toggle breakpoint 🕌 Run to cursor	F5 F4	
Open files list X	8 9 10 11 12 13	Add symbol file Debugging windows Information Edit <u>w</u> atches	) }	
Dened Files	14 15 16	Attach to process Detach		
	17 18 19 20 21	Send user command to cout << z < return 0;		
	22 23			Þ
Continue exe WINDOWS-1252	Line 1:	I, Column 1	Insert	Read/Write

When you are done debugging, you can click on **Continue and your program will run to completion.** This is better than selecting to **Stop debugger**. The reason it is better to **Continue**, is because the program comes to a natural end, rather than aborting. However if your program is stuck in a loop, or you are sure you can exit safely, you can select from the Debug menu **Stop Debugger**.

<mark>–</mark> main.cpp [debug] - Code::	Blocks v1.	.0		
<u>File E</u> dit <u>V</u> iew Sea <u>r</u> ch <u>P</u> roje	ect <u>B</u> uild	Debug wxSmith Tools	Plugins <u>S</u> ettings	<u>H</u> elp
🗋 🖆 🚔 📥   🔦 🔖	a 🗗 🕻	<b>↓</b> ≧ <u>S</u> tart	F8	
:		🐼 Stop debugger		ddem(int a, int b) : int
i 😒 🕨 🥵 🐼 🛛 Build ta	arget: Debi	Continue	Ctrl-F7	ት 🤭 🖑 😆 🗔 🚛
Management X	main.ct	😚 Next line	F7	4 Þ 🗙
Projects 4 >	1	😚 Next instruction	Alt-F7	
🖃 😡 Workspace	2	🔥 Step into	Shift-F7	
e Sources	4	🔥 Step out	Ctrl-Shift-F7	
main.cpp	5	Toggle breakpoint	F5	
	7	🕌 Run to cursor	F4	
Open files list ×	8 9 10 11 12 13	Add symbol file Debugging windows Information Edit <u>w</u> atches	*	
Dened Files	14 15 16	Attach to process Detach		
	17 18	Send user command to		
	19 20 21 22 23	cout << z · return 0; }	<< endl;	
Toggle breakt WINDOWS-1252	Line 9,	Column 9	Insert	Read/Write

You can further define places in your program to pause and allow you to inspect the code. This is done by setting breakpoints in your code. You can have zero or more breakpoints in your code. When the debugger encounters a breakpoint, the program pauses and the debugger will allow you to inspect your code. **The breakpoint remains until you remove it. It can be Toggled with F5.** 

📙 main.cpp [debug] - Code::t	llocks v1.0
<u>File E</u> dit <u>V</u> iew Search Proje	ct <u>B</u> uild <u>D</u> ebug wxSmith <u>T</u> ools P <u>l</u> ugins <u>S</u> ettings <u>H</u> elp
🗋 🗃 🛗 📥 🔦 🔖	💰 🗗 🋍 🔍 🕓
	addem(int a, int b) : int
🕴 💊 🌔 🦚 🐼 🖉 Build ta	rget: Debug 💽 🚺 📲 😵 抗 🏞 🖑 😆 🗔 🧃
Management ×	main.cpp d b ×
Projects 4 🕨	1 #include <iostream></iostream>
🖃 😡 Workspace	2 using namespace std;
🚊 📕 debug	3
Sources	<pre>4 int addem(int, int);</pre>
i main.cpp	5 6 <b>int</b> addem(int a, int b) {
	7 int c;
	8
	9 🔶 c=a+b;
	10
	11 return c;
Open files list X	12 <sup>L</sup> ) 13
🔁 Opened Files	14 int main()
main.cpp	15 🖂 (
	16 int x=5, y=2, z;
	17
	18 z=addem(x,y);
	19 cout << z << endl; 20
	20 21 return 0;
	23
	- I I I
E:\Document: WINDOWS-1252	Line 9, Column 11 Insert Read/Write

A breakpoint has been set at line 9. The red circle indicates that there is a breakpoint in the code.

Hain.cpp [debug] - Code::Blocks v1	.0	
<u>File E</u> dit <u>V</u> iew Sea <u>r</u> ch <u>P</u> roject <u>B</u> uild	Debug wxSmith Tools Plugins Settin	ngs <u>H</u> elp
🗄 🗋 🖼 🚔 📥 🔦 🔖 💰 🕑	<mark>↓</mark> È <u>S</u> tart F8	
:	😣 Stop debugger	addem(int a, int b) : int
j 😒 ┝ 🦚 🐼 🛿 Build target: Deb	Continue Ctrl-F7	🗟 🔭 🥙 🖾 🖾
Management × main.c	😯 Next line 🛛 🖓	4 b x
	Next instruction Alt-F7	
□ Workspace 2	Step into Shift-F7	
English Sources	💔 Step out Ctrl-Shift-F7	
main.cpp 5	Toggle breakpoint F5	
7	Run to cursor F4	
89	Add symbol file	
10	Debugging windows	
Open files list X 12	Edit <u>w</u> atches	
Dened Files	Attach to process	-
15 main.cpp 15	Detach	
17	Send user command to debugger	
19	cout << z << endl;	-
20	return 0;	
22	L)	
23		
Run current p WINDOWS-1252 Line 9	, Column 11 Insert	Read/Write

The program is started by selecting from the **Debug** pull-down menu, **Start**. This will run the program in the debugger until a breakpoint is encountered, at which point the program will pause.

📕 main.cpp [debug] - Code::I	Blocks v1.0		
		vxSmith <u>T</u> ools Plugins <u>S</u> ettings ł	
	a de 🛍 🔍 🤅		
<u>1</u>			em(int a, int b) : int
🕴 😒 🍃 🧊 🐼 🛛 Build ta	arget: Debug	💽 🕴 📲 😯 🔭	🔿 🖑 😡 🗔 ∔
Management ×	main.cpp		$\triangleleft \triangleright \times$
Projects	l #in	clude <iostream></iostream>	
⊡ · 🕜 Workspace		ng namespace std;	
🖻 📕 debug	3 4 int	addem(int, int);	
🖻 🔁 Sources	5	addem(Inc, Inc),	
i main.cpp	6 🗔 int	addem(int a, int b) (	
	7	int c;	
	8 9 <b>(</b> )	c=a+b;	
	10	c-arb,	
	11	return c;	
Open files list X	12 -}		
Opened Files	13 14 int	main()	
main.cpp	15 🗆 (		
	16	<pre>int x=5, y=2, z;</pre>	
	17		
	18	z=addem(x,y); cout << z << endl;	
	20		
	21	return 0;	
	22 <sup>L</sup> ) 23		
	23		
E:\Document: WINDOWS-1252	Line 9, Column 1	Insert	Read/Write

When the program pauses at the break point, a red circle with a yellow triangle mark will appear at the breakpoint.

Hain.cpp [debug] - Code::Blocks v1	I.0 X
<u>File Edit View Search Project Build</u>	Debug wxSmith Tools Plugins Settings Help
🗋 🛤 🚔   🔦 🍾   🖧 🕑	I J Etart F8
:	Stop debuggernain() : int
i 😒 👂 🥵 🐼 🛛 Build target: Deb	🚛 Continue Ctrl-F7 🔝 🏷 🤔 🛄 🍋
	😯 Next line F7 d b 🗙
	Rext instruction Alt-F7
Workspace 2	Step into Shift-F7
Eng Sources	😚 Step out Ctrl-Shift-F7
main.cpp 5	Toggle breakpoint F5
7	Run to cursor F4
8	Add symbol file
10	Debugging windows
12	Information
Open files list × 13 Opened Files 14	Edit <u>w</u> atches
] main.cpp 15	Attach to process Detach
16	
18	Send user command to debugger
19 20	cout << z << endl;
21	return 0;
22 23	-,
•	
Toggle breakt WINDOWS-1252 Line 1	9, Column 1 Insert Read/Write

You can set multiple breakpoints. The keyboard shortcut **F5** allows you to toggle the breakpoint at any line.



This screen shows breakpoints on lines 9 and 19, but line 9 indicates that the code has executed to that point.

File Edit View Search Project Build Debug wxSmith Iools Plugins Settings Help   Imain(): int   Imain	nain.cpp [debug] - Code::Blocks v1.0							
Management   Projects   Workspace   Workspace   Sources   Management   Sources   Management   Management   Workspace   Management   Management   Workspace   Management   M		_	J∰ Start	F8				
Projects Imain.cp   Imain.cpp     Imain.cpi     Imain.cpi     Imain.cpi     Imain.cpi     Imain.cpi     Imain.cpi     Imain.cpi	🔆 💊 🖒 🦚 🗔 🗭 Build tarc	get: Debi				😢 🗔 i.		
Attach to process 15 16 17 Send user command to debugger 18 19 Coupt << z << endl; 20 21 return 0; 23	Projects     Image: Constraint of the sector o	1 2 3 4 5 6 7 8 9 10 11 12 13	<ul> <li>Next instruction</li> <li>Step into</li> <li>Step out</li> <li>Toggle breakpoint</li> <li>Run to cursor</li> <li>Add symbol file</li> <li>Debugging windows</li> <li>Information</li> </ul>	Alt-F7 Shift-F7 Ctrl-Shift-F7 F5				
Continue exe WINDOW5-1252 Line 19, Column 8 Insert Read/Write		15 16 17 18 19 20 21 22 23	Detach Send user command to coupt << z return 0; }	<< endl;				

Selecting Continue from the Debugger menu will run the program till the next breakpoint.

🗕 main.cpp [debug] - Code::Blocks v1.0							
<u>File Edit View Search Proje</u>							
i 🗋 🖴 🚔 🔦 🔖 🦂 🗗 🏦 🔍 🕵							
:	main() : int						
🗄 💊 🜔 🦚 🐼 😵 Build ta	rget: Debug 🗾	💷 😣 የን 🙃 🖓 📲					
Management X	main.cpp	4 ▷ 🗙					
Projects 4 🕨	1 #include <iostre< td=""><td>am&gt;</td></iostre<>	am>					
🖃 🞧 Workspace	2 using namespace	std;					
🖃 📕 debug	3						
Sources	4 int addem(int, i	.nt);					
main.cpp	5						
mannepp	6 ⊟ int addem(int a,	, int b) (					
	7 int c;						
	8 9 🔴 c=a+b;						
	10 C-a+b,						
	11 return c;						
	12						
Open files list 🛛 🗙	13						
🔁 Opened Files	14 int main()						
main.cpp	15 🖂 (						
	16 int x=5, y=2	2, z;					
	17						
	18 z=addem(x,y)						
	19 <b>0</b> cout << z <<	cendl;					
	20						
	21 return 0; 22 -)						
	22 - }						
	20						
		<u> </u>					
E:\Document: WINDOWS-1252	Line 19, Column 1	Insert Read/Write					

Now the program stops at line 19, because the program reached the second breakpoint. Press Ctrl-F7 to continue. Now the program runs till the end of the program, because there are no further breakpoints to encounter. When you exit Code:Blocks you may be presented with the following window.



Say "Yes" to save the Workspace. This saves settings of the workspace you are working on.

You now know the basics of how to use the compiler, work with a project, and use the debugger. Good luck with your programming!

# Appendix A

# Installation of Code::Blocks for Mac OS X and Linux

# If you are using the Mac OS, you will need to do the following:

## **Pre-Install steps:**

Here is a step by step instruction file for setting up X11 and XcodeTools for the Mac.

- 1. Make sure that **X11** is installed in your system (check in your Application menu). If not, you'll need to install it from your Mac OS disc.
  - Insert the disc and find the optional packages installer, open it, check the X11 box, and install from that.
  - If you are using 10.1 through 10.3, you can download X11 from <u>Apple</u>.
- 2. **Install Developer Tools:** You may be able to skip this step, if you are using Mac OS X 10.4 *Tiger and have installed the Developer tools.* 
  - As a test, open up **xterm**. You cannot use **terminal** with Code Blocks. xterm is part of X11 and that is why you'll need to have X11 installed.
  - Run the command **gcc**. If you get an error that there are no input files, that means gcc exists and you are good to go. If you get an error saying the command is not found, you need to install the Developer Tools.
  - If you need to install the Developer tools, you can download them from <u>Apple</u>, or you can install them from your Mac OS disc. Locate the Xcode installer on the installation discs and double click to install. Check your discs for Xcode or Developer Tools.
     Run the installer and install everything other than the documentation:
    - For Mac OS X 10.3, you need to install Xcode Tools version 1.2 or later
    - For Mac OS X 10.4, you need to install Xcode Tools version 2.2 or later

**Download Code::Blocks**. Uncompress the zip file and place CodeBlocks.app where you like it. The suggested location is /Developer/Applications or ~/Applications.

## **Installation for Fedora 8 Linux:**

#### <u>Installing Code::Blocks</u> In a console window, become **root** and then execute the following command: **yum install codeblocks**

<u>Running Code::Blocks</u> In the Applications Pull Down Menu, go to Programming and click on: Code::Blocks IDE *or* Open a console window and then execute the following command: **codeblocks** 

# Installation for Ubuntu Linux:

Installation Instructions From: http://forums.codeblocks.org/index.php/topic,8208.msg61085.html#msg61085

#### 1) Add the repositories to /etc/apt/sources.list:

Open the file with a graphical editor as root. Paste the following line in a terminal: gksu gedit /etc/apt/sources.list

Paste this at the end: # codeblocks deb http://lgp203.free.fr/ubuntu/ gutsy universe # wx widgets deb http://apt.wxwidgets.org/ gutsy-wx main

Note: Depending on what version of Ubuntu you are using, in step 1 you may need to replace gutsy with feisty.

#### 2) To make sure your package system trusts these sources. Add their keys.

```
Enter these two lines at the terminal:
wget -q http://lgp203.free.fr/public.key -0- | sudo apt-key add -
wget -q http://apt.wxwidgets.org/key.asc -0- | sudo apt-key add -
```

And update the packages by entering the following lines on the terminal: sudo apt-get update sudo apt-get upgrade

#### 3) Install Code::Blocks

Enter the following line in the terminal: sudo apt-get install libcodeblocks0 codeblocks libwxsmithlib0 codeblocks-contrib

You are able to step 3 whenever you want you to get the latest nightly build.

You should see Code::Blocks in the Programming Languages listing of your programs.