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 Code Blocks	4
Step 1: Download the Software	4
Step 2: Install the Software	
Step 3: Customization of the Code::Blocks User Interface (Optional)	
First Project	
Adding Files To Your Project	
Debugging a Program.	
If you are using the Mac OS, you will need to do the following:	
Pre-Install steps:	
Appendix A:	
Installing Code::Blocks under Mac OS X and Linux	62
Installation for Fedora 8 Linux:	
Installation for Ubuntu 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 graphicallybased 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 Setu	p		
-	License Ag Please revie	reement w the license terms before ins	talling CodeBlocks.
Press Page Down t	o see the rest of the a	greement.	
	ENERAL PUBLIC LICEN ion 3, 29 June 2007	ISE	<u> </u>
Everyone is permi	7 Free Software Found tted to copy and distrit ument, but changing it		
F	reamble		
The GNU General software and othe	Public License is a free r kinds of works.	, copyleft license for	-
		, click I Agree to continue. You	u must accept the
agreement to insta	II CODEDIOCKS.		
ullsoft Install System	v2,35		
		< Back I Agr	ree Cancel

Select I Agree.

Choose Components Choose which features of CodeBlocks you want to install.
u want to install and uncheck the components you don't want to nue.
Standard: Core plugins, core tools, and core lexers
Default install Ontrib Plugins C::B Share Config MinGW Compiler Suite
Description
Position your mouse over a component to see its description.
< Back Next > Cancel

Take the default settings by pressing Next.

Choose I Choose ti Blocks in the follo der. Click Install t	the folder	r in which to ii der. To install	in a differe	77.570.709	, click Browse
Blocks in the follo	wing fold	der. To install	in a differe	77.570.709	, click Browse
				nt folder,	, click Browse
CodeBlocks				Brow	/se
.,35					
		< <u>B</u> ack	Inst	tal	Cancel
	CodeBlocks 5MB B 2.35	5MB B	5MB B 2.35	5MB B 2.35	5MB B 2.35

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:	VINDOW5-1255
General settings	TAB options	End-of-line options
	Use TAB character TAB indents TAB size in spaces: 4	 Show end-of-line chars ✓ Strip trailing blanks ✓ End files with blank line ✓ Ensure consistent EOLs ✓ End-of-line mode:
Folding	Indent options	Other options
Margins and caret	 Auto indent Smart indent Backspace unindents Show indentation guides Show spaces: No 	 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
The second secon	Editor title is the file's O name only (no path information) O relative filename (to the project file)	
	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	
	Margins and caret
	Left margin Width for line numbers (in chars): ✓ Add/remove breakpoints by left-clicking
General settings	Right margin Right margin hint: Visible line Colour: Hint column: 72
Folding	Caret Width: 1
Margins and caret	Period (in milliseconds): 0 500 500
	OK Cancel

- Choose the Source formatter Caret tab on the right of the window.
 - 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.)

Your screen should now look like the figure:

Configure editor		×
	Sourc	e formatter
Mouse Drag Scrolling Mouse Drag Scrolling Keyboard shortcuts Keyboard shortcuts wxSmith settings source formatter	Style Indentation Styles ANSI K&R Linux GNU Java Custom	<pre>Sample Sample namespace foospace { int Foo() { if (isBar) { bar(); return 1; } else return 0; } } Preview</pre>
	OK	Cancel
	UK	

e.

d.

i.

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:

	Help files	
% .	C++ Help	Add Rename Delete
HeaderFixup configuration		Up Down
2		
Help files		
)	C:\Program Files\CodeBlocks\cpp.chm	
Thread search	TIP: \$(keyword) will be replaced by the word under the cursor This is the default help file (shortcut: F1) This line represents a full command to be executed Open this file with the embedded help viewer (only for HTML files)	
Anter van cont Anter Universitätige Vice- met dom	Preserve keyword case	
To-do list	Default keyword value:	
	OK Cancel	

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.

i.

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.

Console application	and the second	×
Console	Please select the language you want to use. Please make a selection C C C	
	< <u>B</u> ack <u>Einish</u> <u>C</u> a	ncel

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



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



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.

Console application	×
🚮 Console	Please select the compiler to use and which configurations you want enabled in your project. Compiler: GNU GCC 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

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.



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::Blocks v	
Eile Edit View Search Project Bu	uild Debug wxSmith Iools Plugins Settings Help
	P 🛍 🔍 🕵
🕴 💊 🌔 🧐 🗔 🙆 🛛 Build target:	Debug
14872700	<i>i.</i>
Management ×	
Projects Symbols 4	
🖻 📕 First Program	
Sources	
maintpp	
Open files list 🗙 🗙	Messages X
🔁 Opened Files	Code::Blocks 🔰 Code::Blocks Debug 4
	Loading: AStylePlugin Autosave BYOGames CBProfiler CB_Koders
	ClassWizard CodeCompletion CodeSnippets CodeStatistics Compiler copystrings Debugger FilesExtensionHandler DevPakUpdater DragScroll
	EnvVars Source Exporter HelpPlugin cbKeyBinder ScriptedWizard
	ToDoList wxSmith wxSmithMime wxSmithWizard WindowsXPLookNFeel
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...**.



Add files to proje	ct				<u>? ×</u>
Look jn:	🔄 First Progra	m	•	+ 🗈 💣 🗉	•
Kistory	First Program	n.cbp			
Desktop My Documents					
My Computer	File <u>n</u> ame:			•	<u>O</u> pen
My Network P	Files of <u>type</u> :	All files (*.*)			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.

File, New, Empty file.

First Program] - Code::Blocks v1.0	
File Edit View Search Project Build De	bug wxSmith <u>T</u> ools Plugins <u>S</u> ettings <u>H</u> elp
New	Empty file Ctrl-Shift-N
Open Ctrl-O Open default workspace <u>R</u> ecent projects <u>R</u> ecent files	Project Build target File Custom From user template
Import project	The set complete
Save Ctrl-S	
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	le::Blocks 🔌 Code::Blocks Debug 🕴 🔹 🕨
Save wor <u>k</u> space as Close workspace	StylePlugin Autosave BYOGames CBProfiler CB_Koders rd CodeCompletion CodeSnippets CodeStatistics Compiler is Debugger FilesExtensionHandler DevPakUpdater DragScroll
S Close file Ctri-W	ource Exporter HelpPlugin cbKeyBinder ScriptedWizard vxSmith wxSmithMime wxSmithWizard WindowsXPLookNFeel
Close all files Ctrl-Shift-W	cartup script
Close project	
Close all projects	

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 in:	🔄 First Program	n	•	+ 🗈 💣 🎟-	
History Desktop My Documents	C main.cpp				
My Computer My Computer 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.

ielect the targets this file should belong to Debug	Wildcard select
✓ Release	Toggle selection
	Select All
	Deselect All
	Selected: 2

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] - [Code::Blocks v1.0		
Eile Edit View Search Project	<u>Build Debug</u> wxSmith	<u>T</u> ools Plugins <u>S</u> etting	gs <u>H</u> elp
1 🖸 🖬 🖨 🔦 🔖 🔏	D 🛍 🔒 🕾		
		*	
🕴 💊 🜔 🥵 🗔 🙆 🛛 Build target	: Debug		
14 🖷 🖓 🚡 🏷 🖑 😣 🗌	I i.		
Management	× sample.cpp		4 Þ 🗙
Projects Symbols 4			
Open files list	× Messages		×
Dened Files	Code::Blocks	🕦 Code: :Blocks Debug	3
	ClassWizard CodeCom copystrings Debugger EnvVars Source Export	Autosave BYOGames CBP oletion CodeSnippets Cod FilesExtensionHandler De er HelpPlugin cbKeyBinde nithMime wxSmithWizard	deStatistics Compiler WPakUpdater DragScroll er ScriptedWizard
E:\Documeni default	ine 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 Progr	am] - Co	de::Blocks v1.0			
File Edit View Search P	roject B Add files Add files Remove	 recursively	Tools Plugins	Settings Hel	P
। । • • • • • • • • • • • • • • • • • •	Project t				
Management Projects Symbols	Build opt Propertie				4 Þ 🗙
First Program		<u> </u>			Þ
Open files list	×	Messages			×
Opened Files		🚺 Code::Blocks	V) Code:(Block	s Debug	۹ ک
		Loading: AStylePlugin ClassWizard CodeCom copystrings Debugger EnvVars Source Expor ToDoList wxSmith wxS Running startup script	pletion CodeSnipp FilesExtensionHar ter HelpPlugin cbKr mithMime wxSmith	ets CodeStatis Idler DevPakUp eyBinder Script	tics Compiler odater DragScroll tedWizard
Remove files default	Lin	e 1, Column 1	Insert	1	Read/Write

From the Project menu select, Remove files.

main.cpp	Wildcard select
sample.cpp	Toggle selection
	Select All
	Deselect All
	Selected: 1

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*.



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			
Eile Edit View Search Proj	ject <u>B</u> uild <u>D</u> ebug	wxSmith Tools	Pļugins <u>S</u> e	ettings <u>H</u> elp	
	🔥 🗗 🛍 🔜	R			
				*	
i 😒 👂 🥵 🐼 🛛 Build t	arget: Debug]		
i 💵 🖷 🖓 🐍 🏷 🖉 🕼) 🗔 i.				
Management ×	main.cpp				4 Þ 🗙
Workspace First Program Sources main.cpp Open files list	2	nclude <iost t main() std::cout return 0;</iost 		world!" << std	::endl;
🔁 Opened Files	Messages				×
	📝 Code::Block	s 🔰 Code::Bl	ocks Debug	Search results	⊲ ►
	CodeCompletion Co FilesExtensionHand	deSnippets Code: ler DevPakUpdate der ScriptedWiza dowsXPLookNFee	Statistics Com er DragScroll E rd ToDoList w	iller CB_Koders ClassW piler copystrings Debu invVars Source Export xSmith wxSmithMime	gger 👘
E:\Document WINDOWS-1252	Line 1, Column 1		Insert	Read/W	/rite

🗧 main.cpp [First Program]] - Code::Blocks v1.0		
	oject Build Debug wxSmith Add files Add files recursively Remove files Project tree • Set programs' arguments	Tools Plugins Settings	Help
	Build options		4 Þ 🗙
Projects 4 Workspace First Program Sources main.cpp	2 3 int main() 4 : (5 std::: 6 return 7 }	cout << "Hello world	d!" << std;:endl;
Dened Files main.cpp Untitled2	Messages Code::Blocks Co Loading: AStylePlugin Autosav CodeCompletion CodeSnippets FilesExtensionHandler DevPakl HelpPlugin cbKeyBinder Scripte wxSmithWizard WindowsXPLoo Dupping charture script	CodeStatistics Compiler cop Jpdater DragScroll EnvVars dWizard ToDoList wxSmith v	Koders ClassWizard oystrings Debugger Source Exporter
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.**

ľ
_
-
С-е
<u>ب</u>

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.

<mark>- main.cpp [First Program]</mark> File <u>E</u> dit <u>V</u> iew Sea <u>r</u> ch Proj	<mark>- Code::Blocks v1.0</mark> ect <u>Build D</u> ebug wxSmith <u>T</u> ools Pl़	lugins <u>S</u> ettings <u>H</u> elp	
L III III A N N N N N N N N N N N N N N N	Compile current file Ctrl-Sh Run C arge 🎧 Build and run	Ctrl-F9 hift-F9 :trl-F10 F9 :trl-F11	
Management X			d þ×
Projects 4	Build <u>w</u> orkspace Reb <u>ui</u> ld workspace Clean workspace Second Errors	world!" << std:	:endl;
	Select target	🕨 🗸 Debug	
Open files list X	Export Makefile	Release	
Opened Files main.cpp Untitled2	Me Suld log & Build messages	S Debugger	∢ ⊳
	Setting breakpoints Debugger name and version: G At E:/DOCUME~1/goetz/MYDOCU~ Debugger finished with statu	1/FIRSTP~1/main.cpp:5	
uild target 'WINDOWS-1252	Line 6, Column 11 Ins	sert Read/Wr	ite

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.

ile Edit View Search Pro	ect Build Debug wxSmith	Tools Plugins Settin	gs Help
	🦧 😜 <u>B</u> uild	Ctrl-F9	
	Compile current <u>fi</u> le	Ctrl-Shift-F9	main() : int
' 💊 🕨 🥵 🐼 Build t		Ctrl-F10	
		F9	
비 🖶 🖓 🚡 🏷 🖉 🕼	🕽 🛛 💽 Rebuil <u>d</u>	Ctrl-F11	
lanagement X	Clean		4 Þ ×
Workspace	Reb <u>uild workspace</u> Clean workspace <u>Abort</u> <u>Errors</u> Select target		rld!" << std::endl;
pen files list 🛛 🗙 🗙	Export Makefile		I
Opened Files	Me:		
main.cpp	Suild log 🛛 🗳 Build m	essages 🛛 🚫 Debu	gger 🔹 🕴
	Setting breakpoints Debugger name and ver At E:/DOCUME~1/goetz/ Debugger finished wit	MYDOCU~1/FIRSTP~.	

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

-main.cpp [First Program]	- Code::Blocks v1.0	
<u>File E</u> dit <u>V</u> iew Sea <u>r</u> ch Pro	iect <u>Build D</u> ebug wxSmith <u>I</u> ools Plugins <u> </u>	9 9
💵 🏪 🖓 🔐 🏷 💔 🄇 Management 🛛 🗙 🛠		1 4 b x
Projects 4 Workspace First Program Sources Main.cpp Open files list ×	Build workspace Rebuild workspace Clean workspace Abort Errors Select target	world!" << std::endl;
Opened Files main.cpp Untitled2	Export Makefile	×
. Ontriedz	Process terminated with status O seconds) O errors, O warnings) (O minutes, 18
Build and rur WINDOWS-1252	Line 1, Column 1 Insert	Read/Write

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.

main.cpp [First Program] - Code::Blocks	s v1.0
<u>File E</u> dit <u>V</u> iew Sea <u>r</u> ch <u>P</u> roject <u>B</u> uild <u>D</u> eb	bug wxSmith <u>T</u> ools Plugins <u>S</u> ettings <u>H</u> elp
New	▶ <mark>등 </mark>
🔄 Open Ctrl-O	main() : int
Open default workspace	
Recent projects	•
Recent files	•
Import project	•
Gtrl-5	finclude <iostream></iostream>
Ddve ds	nt main()
J Save as Save all files Ctrl-Shift-S	
	<pre>std::cout << "Hello world!" << std::end return 0;</pre>
Save all files Ctrl-Shift-S	std::cout << "Hello world!" << std::end
Save all files Ctrl-Shift-S Save project	std::cout << "Hello world!" << std::end
Save all files Ctrl-Shift-S Save project Save project as	std::cout << "Hello world!" << std::end
Save all files Ctrl-Shift-S Save project Save project as Save project as user-template Save all projects	<pre>std::cout << "Hello world!" << std::end return 0;</pre>
Save all files Ctrl-Shift-S Save project Save project as Save project as user-template Save project as user-template	std::cout << "Hello world!" << std::end
Save all files Ctrl-Shift-S Save project Save project as Save project as user-template Save all projects Save workspace Save workspace	<pre>std::cout << "Hello world!" << std::end return 0; Build messages Debugger 4 kpoints</pre>
Save all files Ctrl-Shift-S Save project Save project as Save project as user-template Save all projects Save all projects Save workspace Save workspace Save workspace	std::cout << "Hello world!" << std::end return 0; Build messages Debugger 4 kpoints e and version: GNU gdb 6.3
Save all files Ctrl-Shift-S Save project Save project as Save project as user-template Save all projects Save all projects Save workspace Save workspace Save workspace Save workspace Close workspace Siglose file Ctrl-W	<pre>std::cout << "Hello world!" << std::end return 0; Build messages Debugger 4 kpoints</pre>
Save all files Ctrl-Shift-S Save project Save project as Save project as user-template Save all projects Save workspace Save workspace as Close workspace Save workspace	<pre>std::cout << "Hello world!" << std::end return 0; Build messages O Debugger kpoints e and version: GNU gdb 6.3 ~1/goetz/MYDOCU~1/FIRSTP~1/main.cpp:5</pre>

When you are done, save all your files by pulling down the File menu and clicking on Save all files.
Now you can select to save the project:

main.cpp [First Program] - Code::Blocks	v1.0		
File Edit View Search Project Build Debi New	ug wxSmith <u>T</u> ools	Pļugins <u>S</u> ettings I	Help
🔁 Open Ctrl-O		mair	n() : int
Open default workspace <u>R</u> ecent projects <u>R</u> ecent files	·]	
Import project			4 Þ 🗙
Save as Save all files Ctrl-Shift-S Save project	.nt main() std::cout return 0;	<< "Hello world	!" << std::endl;
Sa <u>v</u> e project as <u>S</u> ave project as user-template Save all projects			<u> </u>
Save <u>w</u> orkspace Save wor <u>k</u> space as	🔗 Build message	s 🚫 Debugge	د م ا
Close workspace	kpoints e and version:	GNU gdb 6.3	<u> </u>
S ⊆lose file Ctrl-W Close all files Ctrl-Shift-W Close project	이 아님님 (NYA CHERY 2017) CHERY ()	CU~1/FIRSTP~1/ms	ain.cpp:5
Close all projects	11	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		<u>?</u> ×
Look in:	🔄 First Program 💽 🖝 🛅 🕇	
History Desktop	⊇ obj ➡ First Program.cbp	
My Documents		
My Network P		<u>O</u> pen 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.

main.cpp [debug] - Code::I	Blocks v1.0				- 🗆 ×
<u>File Edit View Search Proje</u>	ect <u>B</u> uild <u>D</u> e	bug wxSmith <u>T</u> ools Pļu	gins <u>S</u> et	tings <u>H</u> elp	
1 🖬 🖬 🖨 ا 🔦 🔖 🛛	* 0 6	🔜 🖳			
				main() : int	
i 😒 👂 🧊 🐷 Build ta	arget: Debug	•	順 悟	ලි බ. ලි ගි	😧 🗔 i.
Management ×	main.cpp				d ⊳×
Projects 4 🕨	1	#include <iostream< td=""><td><u>n</u>></td><td></td><td></td></iostream<>	<u>n</u> >		
🖃 🕥 Workspace	2	using namespace st	d;		
🗄 🗧 debug	3				
🖻 🥅 Sources	4	int addem(int, int	t);		
in ain.cpp	5			14-1	
	6	int addem(int a, i int c;	int b)	f	
	7	int c;			
	9	c=a+b;			
	10	C diby			
	11	return c;			
On an Olan link	12	L)			
Open files list X	13				
Opened Files	14	int main()			
main.cpp	15		10		
	16 17	int x=5, y=2,	Z;		
	18	z=addem(x,y);			
	19	cout << z << 6	endl;		
	20				
	21	return 0;			
	22	L-}			
	23				
	4				F
E:\Document: WINDOWS-1252	Line 16, (olumn 21 Ins	ert	Re	ad/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.**

		Debug wxSmith Tools Plugins		s <u>H</u> elp I
		📔 Start	F8	
		🐼 Stop debugger		nain() : int
😒 👂 💫 🗔 🙆 Build ta	rget: Debi	Continue Cl	trl-F7	🚠 ኛን 🔇 🗔 🕯
lanagement X	main.cr	😚 Next line	F7	d þ x
Projects 4 🕨	1	Next instruction	lt-F7	
- Workspace	2	🔥 Step into Shi	ft-F7	
e Page Sources	4	😚 Step out 🛛 Ctrl-Shi	ft-F7	
main.cpp	5	Toggle breakpoint	F5	
	7	Run to cursor	F4	
	8 ⁻ 9	Add symbol file	2	
	10	Debugging windows	•	
	11 12	Information	•	
pen files list X	13	Edit <u>w</u> atches		
Opened Files main.cpp	14 15	Attach to process		
mannepp	16	Detach		
	17 18	Send user command to debugger		
	19	cout <mark><<</mark> z <mark><<</mark> endl		20)
	20 21	return 0;		
	22	L.)		
	23			

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.

Stop debugger Management Management <	ld Debug wxSmith Tools Plugins Settings Help
Image: Image	
Management main.cr Projects 1 Workspace 2 debug 3 Sources 3 main.cpp 6 7 Sources 6 7 8 9 Add symbol file 11 12 13 0pen files list 14 15 16 17 18 10 11 11 11 12 13 14 15 16 17 18 18 10 11 12 13 14 15 16 17 18 10 11 12 13 14 15 16 16 17 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 <th>Stop debugger nain() : int</th>	Stop debugger nain() : int
Projects 4 Workspace 1 Sources 3 main.cpp 4 Open files list X Opened Files 14 Main.cpp 12 Main.cpp 12 Main.cpp 1 Main.cpp <	ebi 🚛 Continue 🛛 🖸 Ctrl-F7 📅 🖓 🔇 🗔 🧃
Projects 4 1 Image: Sources source	🔐 🕜 Next line F7 🛛 🖉 🗛 🖓
Workspace debug Sources main.cpp Toggle breakpoint F5 Toggle breakpoint F5 Run to cursor F4 Add symbol file Debugging windows Information Information Information CPU Register Disassembly Edit watches Detach Information Detach Running thre Send user command to debugger	
Sources main.cpp Main.cpi Main.cpi Main.cpi Main.cpi Main.c	
main.cpp S main.cpp 6 7 8 9 Add symbol file 10 0 <t< td=""><td>4 Step out Ctrl-Shift-F7</td></t<>	4 Step out Ctrl-Shift-F7
7 Run to cursor F4 8 Add symbol file 10 Debugging windows Breakpoints 11 Information Call stack 13 Edit watches CPU Register 14 Attach to process Disassembly 15 Detach Running thre 16 Detach Running thre 17 Send user command to debugger Watches	
9 Add symbol file 10 Debugging windows Breakpoints 11 Information Call stack 12 Edit watches CPU Register 13 Attach to process Disassembly 14 Attach to process Examine men 16 Detach Running thre 18 Send user command to debugger Watches	7 🏰 Run to cursor 🛛 🗛 🖌
Dpen files list X Depend files 11 Information Call stack CPU Register Disassembly Disassembly Disassembly Eath Watches Examine men Information Examine men	
Impendision Call stack Copened Files Edit watches CPU Register Disassembly Impendition Edit watches CPU Register Disassembly Impendition Edit watches Impendition Call stack Impendition Coll stack <td>Ecologiand mildouth</td>	Ecologiand mildouth
Opened Files 13 Disassembly Imain.cpp 15 Attach to process Disassembly Interport 15 Detach Examine men Interport 16 Detach Running three Interport 18 Send user command to debugger Watches	
Imain.cpp 15 Attach to process Examine men 16 Detach Running thre 17 18 Send user command to debugger Watches	
17 18 Send user command to debugger Watches	5 Attach to process Examine memory
	7 Sept user command to debugger Watches
19 cout << z << endl; 20 21 22 -} 23	return 0;

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 $\times \& y$ 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.

<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	h <u>T</u> ools Plugins <u>S</u> ettings l	<u>H</u> elp
	🔏 🗗 🛿 🐙 Start	F8	
	😡 Stop debugg	ger nair	n() : int
😒 👂 🧐 🐷 🙆 Build ta	arget: Debi 🚛 Continue	Ctrl-F7	শৈ 🖸 🖸 🗔 i
1anagement X	main.cr 🕜 Next line	F7	4 Þ ×
Projects 4 >	1 💦 Next instruct	tion Alt-F7	
🛛 😡 Workspace	2 3 Thep into	Shift-F7	
e Page Sources	4 💔 Step out	Ctrl-Shift-F7	
main.cpp	5 6 Toggle breakp	point F5	
	7 🧏 Run to curso	or F4	
	8 9 Add symbol fil	le	
	10 Debugging wir	ndows 🔸	
200 - 2011 - 201	11 Information	•	
open files list X	13 Edit <u>w</u> atches.		
main.cpp	14 Attach to prov 15 Detach 16 Detach	cess	
	17 18Send user con	mmand to debugger	
	19 cout 20	: << z << endl; mn 0;	
	<[]		

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.

- Code [debug] - Code	:::Blocks v1.0	_ 🗆 ×
<u>File Edit View Search Pr</u>	oject <u>B</u> uild <u>D</u> ebug wxSmith <u>T</u> ools Plugins <u>S</u> ettings <u>H</u> u	elp
	💰 🗗 😩 🔍 🕵	
	<u>*</u>	
i 💊 👂 🥵 🐷 Build	target: Debug 💽 🚺 👫 🍄 🔭 🚠	ዮን (ኛ 😡 🗔 i
Management >	main.cpp	d ⊳×
Projects 4 🕨	1 #include <iostream></iostream>	
🖃 🕜 Workspace	2 using namespace std;	
🖻 💾 debug	3	
🖻 🥦 Sources	<pre>4 int addem(int, int);</pre>	
main.cpp	5	
	6 Gint addem(int a, int b) { 7 int c;	
	8	
	9) c=a+b;	
	10	
	11 return c;	
Open files list >	12 }	
Opened Files	13 14 int main()	
main.cpp		
	16 int x=5, y=2, z;	
	17	
	18 z=addem(x,y);	
	19 cout << z << endl;	
	20 21 return 0;	
	21 return 0;	
	23	
		•
E:\Document: WINDOWS-1252	Line 2, Column 12 Insert	Read/Write

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.

<u>jile E</u> dit <u>V</u> iew Sea <u>r</u> ch <u>P</u> roj	ect <u>B</u> uild	Debug wxSmith Tools Plugins Sett	ings <u>H</u> elp
🔄 🗃 🗃 📇 🔦 🔖	801	↓ Start F8	
		😡 Stop debugger	ddem(int a, int b) : int
💊 խ 😘 🗔 🙆 Build t	arget: Debu	Continue Ctrl-F7	ት. 🔿 🕫 😡 🗔 i
lanagement X	main.cp	😯 Next line F7	X 4 D
Projects 4 >	1 2 3 4	Next instruction Alt-F7 Next instruction Step into Step into Shift-F7 Step out Ctrl-Shift-F7	
🖻 🦰 Sources	5 6 7 8	Toggle breakpoint F5	
ppen files list X	9 10 11 12 13	Add symbol file Debugging windows Information Edit <u>w</u> atches	•
• Opened Files main.cpp	14 15 16	Attach to process Detach	
	17	Send user command to debugger	
	19 20 21 22 23	<pre>cout << z << endl; return 0; }</pre>	

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	×
E Local variables	
c = 7	
E Function Arguments	
a = 5	
i b = 2	
94.82	
	_

The debug window reflects the change of $\ensuremath{\mathtt{c}}.$

	▶ & ₽	↓ Estart F8 Stop debugger	addem(int a, int b) : int
	uild target: Debu	Continue Ctrl-F7	🗂 🔿 🥂 😡 🗔 i
Anagement Projects 4 Workspace debug	X main.cr	Next line F7 Next instruction Alt-F7 Step into Shift-F7 Step out Ctrl-Shift-F7	4 b ×
inain.cpp	5 6 7 8	Toggle breakpoint F5	
pen files list	9 10 11 12 13	Add symbol file Debugging windows Information Edit <u>w</u> atches	•
Opened Files main.cpp	14 15 16	Attach to process Detach	
	17 18 19	Send user command to debugger	
	20 21 22 23	return 0;	

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**.

<u>File Edit View Search Proje</u>	t <u>B</u> uild <u>D</u> ebug wxSmith	h <u>T</u> ools Plugins <u>S</u> ettings	<u>H</u> elp
🗋 🖻 🖬 🖨 ا 🔦 📎 🛛	🔏 🗗 🚺 💵 Start	F8	
	😡 Stop debug	ger	iddem(int a, int b) : int
😒 👂 ୠ 🗔 🙆 Build tar	rget: Debi	Ctrl-F7	ी. 🏷 (१ 🙆 🗔 🕯
Management X	main.cr 🕜 Next line	F7	4 Þ ×
Projects 4 >	1 🕜 Next instruc	tion Alt-F7	
🗄 😡 Workspace	2 🏠 Step into	Shift-F7	
e Page Sources	3 4 💔 Step out	Ctrl-Shift-F7	
main.cpp	5 6 Toggle break	point FS	
	7 🧏 Run to curso	or F4	
	8 Add symbol fi 9 Add symbol fi 10 Debugging wi	X	
open files list 🛛 🗙	11 Information 12 Edit <u>w</u> atches,	•	
• Opened Files	14 Attach to pro 15 Detach 16 Detach	cess	
	17 18 Send user cor	nmand to debugger	
	19 cout 20	; << z << endl; mn 0;	
oggle break; 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] -		
	Project Build Debug wxSmith Tools Plugins ;	Settings Help
1 🖻 🖬 🖨 🔦	🕨 💰 🗗 🋍 🖳 🖳	
		addem(int a, int b) : int
S S S S S	uild target: Debug 💽 🚺 👫 🍕	i 🖓 T. 🏷 (f 🥝 🗖 i
Management	X main.cpp	4 Þ 🗙
Projects	1 #include <iostream></iostream>	
🖃 🕜 Workspace	2 using namespace std;	
🚊 🏪 debug	3	
🖻 🔁 Sources	<pre>4 int addem(int, int);</pre>	
i 📄 main	5 6 ⊡int addem(int a, int b	A 1
	7 int c;	7 (
	8	
	9 🔶 c=a+b;	
	10	
	11 return c;	
Open files list	X 12 L	
🔁 Opened Files	14 int main()	
i main.cpp	15 🖂 (
1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 -	16 int x=5, y=2, z;	
	17	
	18 z=addem(x,y); 19 cout << z << endl;	
	20	
	21 return 0;	
	22 L _}	
	23	
		•
:\Document: WINDOWS-	52 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.

<u> Edit View Search Proje</u>	ct <u>B</u> uild [Debug wxSmith Tools Pl	ugins <u>S</u> etting:	s <u>H</u> elp
	201	📔 <u>S</u> tart	F8	
		🕃 Stop debugger		addem(int a, int b) : int
😒 👂 👒 🗔 🙆 Build ta	rget: Debi	Continue	Ctrl-F7	ी. 🔁 🖉 🙆 🗔 🕯
1anagement ×	main.cr	🕜 Next line	F7	A Þ×
Projects 4 >	1	😚 Next instruction	Alt-F7	
🗄 😡 Workspace	2	Step into	Shift-F7	
e Page Sources	3 4 {	😗 Step out 🛛 🖓	trl-Shift-F7	
main.cpp	5	Toggle breakpoint	F5	
	7 5	🖶 Run to cursor	F4	
	8 - 9	Add symbol file		
	10	Debugging windows	•	
	11 12	Information	•	
Open files list X	13	Edit <u>w</u> atches		
🔁 Opened Files	14 15 16	Attach to process Detach		
	17 18	Send user command to del	pugger	
	19 20 21 22 23	cout << z << return 0; -}	endl;	
	•			

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.



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

ile <u>E</u> dit <u>Y</u> iew Sea <u>r</u> ch <u>P</u> roje	ct <u>B</u> uild <u>D</u> ebug wxSmith (<u>T</u> ools Plugins <u>S</u> ettings l	Help
🗋 🗃 🛱 📥 💊 📘	🔏 🗗 🖊 Start	F8	
	😧 Stop debugger	nair	n() : int
🛇 👂 😘 🗔 🙆 Build ta	rget: Debi 🚛 Continue	Ctrl-F7	ኛን 🕼 🖾 i
anagement ×	main.cr 🕜 Next line	F7	4 Þ ×
Projects 4 🕨	1 😚 Next instruction	Alt-F7	
Workspace	2 🏠 Step into	Shift-F7	
e Pag	4 😗 Step out	Ctrl-Shift-F7	
main.cpp	5 6 Toggle breakpoin	t F5	
	7 \ 🕌 Run to cursor	F4	
	8 Add symbol file	25	
	10 Debugging windo	ws 🕨	
	11 Information	*	
pen files list X • Opened Files	13 Edit <u>w</u> atches		
main.cpp	14 Attach to process 15 Detach 16 Detach	5	
	17 18 Send user comma	nd to debugger	
	19 cout < 20	< z << endl;	
	21 return 22 -} 23	0;	
			10

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.

		Debug wxSmith Iools Plugins Sett	- 1
		😡 Stop debugger	nain() : int
😒 👂 🥵 🐷 🛛 Build ta	arget: Debi	Continue Ctrl-F7	🕺 🕄 😗 🚱 🗔 🕯
Management X Projects 4 O Workspace O Workspace O Sources Management A A A A A A A A A A A A A A A A A A A	main.cp	Next line F7 Next instruction Alt-F7 Step into Shift-F7 Step out Ctrl-Shift-F7	
	5 6 7 8	Toggle breakpoint F5 F3 Run to cursor F4	
pen files list X	9 10 11 12 13	Add symbol file Debugging windows Information Edit <u>w</u> atches	*
Dpened Files	14 15 16	Attach to process Detach	
	17 18	Send user command to debugger	
	19 20 21 22 23	<pre>cout << z << endl; return 0; }</pre>	

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



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

For Mac OS X 10.5, you need to install Xcode Tools version 3.1 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:

Installing Code::Blocks 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 -O- | sudo apt-key add wget -q http://apt.wxwidgets.org/key.asc -O- | 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.