Code::Blocks Students' Guide

This document is meant to make it easy for you to install the IDE we will be using this semester, ensure it is working, learn about its most useful features. It is organized as follows;

- **S1 Installing Code::Blocks IDE**
- **S2** Testing the IDE
- **S3 Debugging Programs**

Feel free to ask any question you might have on the offering's forums. Make sure you work on all steps in this guide, including preparing a .zip file of the entire PA203 project folder you when you are done. This will prepare you for dealing with future graded assignments.

S1 – Installing Code::Blocks IDE

S1.1 – Downloading & Installing the IDE

The installation process is pretty straightforward. Download the **codeblocks**-**10.05mingw-setup.exe** file from blackboard and run it. Accept all default settings.

Run Code::Block as soon as the installation is finished.

S1.2 – First run

You will see a dialog window showing all available compilers. This installer also installs the GNU GCC Compiler on your machine which is the only one which will be compatible with the systems we will be using to grade your assignments. Select GNU GCC Compiler from the list and click on Set as default.

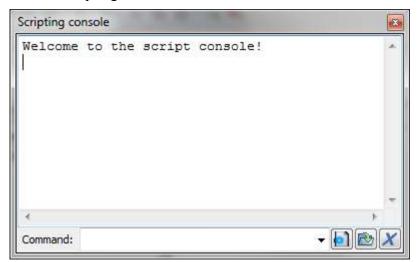
Compiler	Status	<u>^</u>	Set as default
GNU GCC Compiler	Detected		
Microsoft Visual C++ Toolkit 2003			
Microsoft Visual C++ 2005/2008			
Borland C++ Compiler (5.5, 5.82)		E	
Digital Mars Compiler			
OpenWatcom (W32) Compiler			
GNU GCC Compiler for MSP430			
Cygwin GCC			
LCC Compiler			
Intel C/C++ Compiler			
SDCC Compiler			
Tiny C Compiler		-	
efault compiler: GNU GCC Compiler			

You should now see 3 windows opened;

- The Tip of the Day – Feel free to close it for now

🔮 Did	you know
	sable these tips on startup i k the "Show tips at startup"

The scripting console – feel free to close it for now.



The main IDE panel

_

-

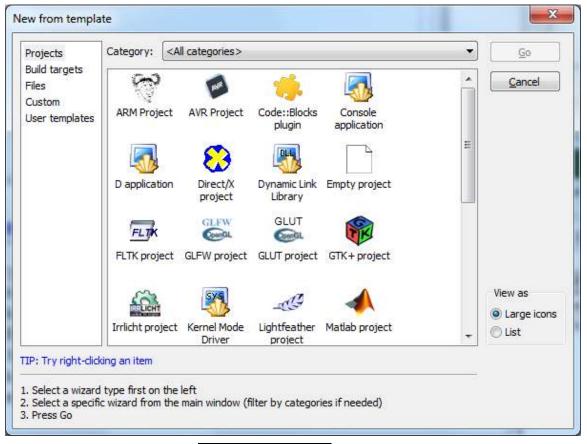
Start here - Code Hiocka 10/5		
	Build Debug webmith Joole Plugim Settings Help	
10 GHOILS NO	6 Q 4	
1	•	
0 > 9 0 0 8.44 target		
48 9 1 2 7 0 0 0 3		
Vonagement X	Starthere ×	
I Projecta Symbols Rass.	Surface a	
- O Workspace		
	Vantita Casa-Backa kurua Baseria kas Baseria kas Recent projecta Na recent projecta	Inter
	Lap àother	×
	CodectRiccles × Consectments Collecting # Build ressures Collectinger	
	Codectionces × Seerch results Solution Thad resuges Sublight	
		ŝ.
Welcome to Code:Blocks!		defeak

You are now ready to work with this IDE.

S2 – Testing the IDE

S2.1 – Preparing a new project

Let us prepare a new project by navigating to the Code::Block menu $File \rightarrow New \rightarrow$ **Project**. This will open the project pane illustrated in the screenshot below.



We will be always selecting Console Application in this offering.

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" Skip this page next time
	< Back Next > Cancel

Go for Next and Skip this page next time if you want to.

Console application	Please select the language you want to use.
	C C++
	< Back Next > Cancel

We are going to write C programs in this offering so the selection is fairly obvious.

In the next panel, you will have to provide a projec tittle, up to you but I recommend you use the name of the assignment e.g. **PA203**. In this example, I'm selecting to have the project folder on my desktop.



By default, Code::Block will be ready to compile two versions of your project. The **Debug** version is the one you'll work with, troubleshoot and then submit. The **Release** one won't be something we'll even use in this offering.

Console application	you want enabled in you	r to use and which configurations r project.
	Compiler: GNU GCC Compiler	•
	Create "Debug" confi "Debug" options Output dir.: Objects output dir.:	bin\Debug\
	Create "Release" con "Release" options	figuration: Release
	Output dir.: Objects output dir.:	bin\Release\ obj\Release\
	< <u>B</u> ac	k <u>Einish</u> <u>C</u> ancel

When you click Finish, you'll be back to the main window with a brand new project opened for you.

H [PA303] - Code::Blocks 10.05	
	Build Debug wxSmith Tools Plugins Settings Help
	🖿 🕋 🔍 🔍 📔 🎯 🕨 🏇 🌚 🛛 Build target: Debug 🛛 🚽 🕴 😁 🕜
	▼
Management X	
I Projects Symbols Resourt	
- Workspace	
PA303	
B. B. Bources	
	L
	Logs & others X
	Code::Blocks 🔍 Search results 🥎 Build log 📌 Build messages 😒 Del 🕨
	Command:
-	
Welcome to Cc	default

If you open the **Sources** tree on the left panel you will find a main.c already there for you.

 main.c [PA303] - Code::Blocks 10.0 <u>File Edit View Search Project</u> <u>Project</u> <u>Project</u> 	<u>Build Debug</u> wxSmith <u>T</u> ools Plugins <u>S</u> ettings <u>H</u> elp
Management X	main.c ×
Projects Symbols Resou ► Workspace PA303 Sources Imain.c	<pre>1 #include <stdio.h> 2 #include <stdib.h> 3 4 int main() 5 = { 6 printf("Hello world!\n"); 7 return 0; 8 } 9</stdib.h></stdio.h></pre>
	Logs & others X
	Command:
C:\CLUE\Assign WINDOWS-1252	Line 1, Column 1 Insert Read/Write default

S2.2 – Compiling and Executing a project

In order to run your program, you must build it. This entails compiling each of the .c source files which are part of your project, linking them together along with any needed library with a tool named the linker. Refer to the first module's videos lectures for more information about this whole process.

You will achieve this by using the Build \rightarrow Build menu or the Ctrl-F9 shortcut. The Build log tab in the bottom panel will show you any potential errors.

Logs & others				
Code::Blocks	🔍 Search results	💊 Build log 🛛 🗙	📌 Build messages	🚫 Debugger
Output size is	e executable: bin\ 25.28 KB ted with status 0	-		
3\main.c				WINDOWS-1252

When your project is built, you may run it with "Build" \rightarrow "Run or Ctrl-F10 shortcut. The interaction with your program will take place in DOS text Console window which will prompt you to press any key before to close when your program is done executing.



S3 – Debugging Programs

This section supplements the previous ones by showing you how to use Code::Block debugger to troubleshoot your programs.

S3.1 – Why using a Debugger at all?

One of the real benefits of using Code::Block is that you are able to execute step by step a program and display the value of key variables at each step. However, keep in mind you would be able to do the same thing by adding a few instructions to display the values of these same variables as your program execute. So, again, make sure you do not spend hours learning to use the debugger when you should be instead learning to program.

S3.2 – Preparing your program to be ran with the debugger

For the sake of demonstrating how the debugger works, we are going to use a simple "Hello World" program.

Eile Edit View Search Project	Build Debug wxSmith Tools Plugins Settings Help Image: Debug Image: Debug
Management × Projects Symbols Resol Workspace PA303 Sources main.c	<pre>main.c × 1</pre>
	Logs & others
	Code::Blocks Search results Suid log X Build log X Build messages Debugger Build: Debug in PA303 Target is up to date. Nothing to be done.

As you may notice in the above screenshot, the project is already built and ready to execute. You may not debug any project which is not already building properly an executing. If you project produces bad results or "crashes" while running, it's ok. This is why we are going to troubleshoot it. But a debugger won't be able to help with syntax errors. So make sure that you have fixed these before to debug.

S3.3 – Setting up "BreakPoints"

When you use a debugger, you need to have an idea of where things are going wrong in your program. The best way to do this, is to put a breakpoint so that, when you run your program with the debugger, execution will pause at that line in the code.

Select a line of code and use $\boxed{\text{Debug}} \rightarrow \boxed{\text{Toggle Breakpoint}}$ to set or unset a breakpoint at this line. You may also use the $\boxed{\text{F5}}$ shortcut instead. Notice the red dot on the break point line below;

H main.c [HellowWorld] - Code::Bloc	ks 10.05
File Edit View Search Project Image: I	Build Debug wxSmith Tools Plugins Settings Help 1 🕋 🔍 🔍 🔋 🍪 🕨 🏇 🌚 Build target: Debug 🛛 🕇 🕼 🏰 🖓 ੋ 🏠 🌾
	→ [main() : int
Management X Projects Symbols Resor Workspace HellowWorld Sources main.c	<pre>main.c × 1 #include <stdio.h> 2 #include <stdlib.h> 3 4 void f(int data){ 5 printf("The value of x is %d\n",data); 7 8 10 int main() 11 if int x = 42; 13 14 printf("Hello world!\n"); 15 f(x); return 0; 18 19 10 Logs & others</stdlib.h></stdio.h></pre>
C:\Users\alessio\Desktop\Hello' WINE	00WS-1252 Line 11, Column 2 Insert Read/Write default

Now, we may run our program with the debugger by selecting $\boxed{\text{Debug}} \rightarrow \boxed{\text{Start}}$. A popup will inform you that the IDE will now display your project in Debug Mode.

_				
The p	perspective 'Code::E	Blocks default' ha	s changed. Do	you want to save i
	e			7
	Yes	No	Cance	

The, the program will start executing until it reaches your breakpoint. The display Console will be blank since we inserted a breakpoint before any of our printf.

C:\Users\alessio\Desktop\HelloWorld\HellowWorld\bin\Debug\HellowWorld.exe	x
	*
	E
	-
	-

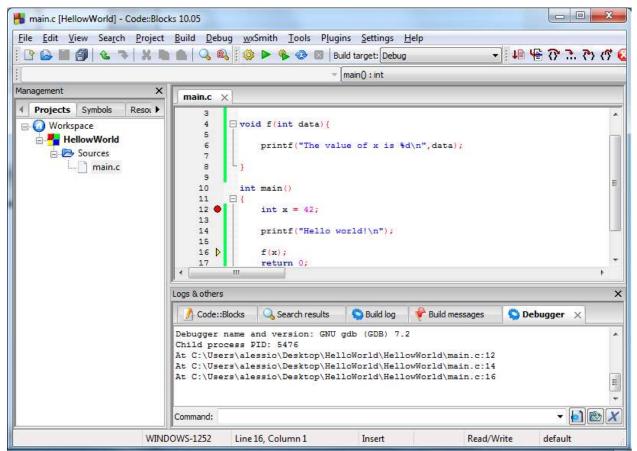
Meanwhile, the IDE window will mark that we are at the break point with a right pointing triangle.

H main.c [HellowWorld] - Code::Block	ks 10.05
<u>File Edit View Search Project</u>	<u>B</u> uild <u>D</u> ebug <u>w</u> xSmith <u>T</u> ools Plugins <u>S</u> ettings <u>H</u> elp
🗄 🕒 🖬 🎒 🔍 🥆 🛛 X 🖿	। 💼 🔍 🔍 🔋 🌼 🕨 🏶 🚳 Build target: Debug 🔹 📢 👫 😯 🔭 🥐 (🕇)
1	✓ [main(): int
Management X	main.c ×
Projects Symbols Resol Workspace HellowWorld Sources main.c	3 Image: state of the st
C:\Users\alessio\Desktop\Hello\ WIND	OWS-1252 Line 13, Column 1 Insert Read/Write default

S3.4 – Stepping through your programs

Right now, we might just select $\boxed{\text{Debug}} \rightarrow \boxed{\text{Continue}}$ and the debugger would just run until the program is done. Generally though, we don't want to do this. We want to "step" line per line throught the program. There are two ways of stepping

Next Line or Next Instruction will do pretty much what it sounds it'd do. Here is the result after using it twice;



Now that we are on a function call, things are different. If we use the same way of stepping, the whole function call will be executed and we'll find ourselves on line #17 without having a possibility to step through the code of the function.

This is when you might want to use Step into instead which will allow you to get into the function's code and step through it.

main.c [HellowWorld] - Code::Blo	
	t <u>B</u> uild <u>D</u> ebug <u>w</u> xSmith <u>T</u> ools P <u>I</u> ugins <u>S</u> ettings <u>H</u> elp
🗄 🕒 🖬 🞒 🕲 🥆 🗙 🛛	h 🧥 🔍 🔍 🕴 🏟 🕨 🏀 🖾 🛛 Build target: Debug 💿 💽 🖓 👫 🏠 🕐 🕻
#	√ [f(int data) : void
Management X Projects Symbols Resor V Workspace HellowWorld Sources main.c	
C:\Users\alessio\Desktop\Hello\ WIN	IDOWS-1252 Line 6, Column 1 Insert Read/Write default

If you want to step through all its code, go for it. If at some point you just want to run until the function returns, then use the Step Out option.

Hain.c [HellowWorld] - Code::Bloc	ks 10.05
<u>File Edit View Search Project</u>	<u>B</u> uild <u>D</u> ebug <u>w</u> xSmith <u>T</u> ools P <u>l</u> ugins <u>S</u> ettings <u>H</u> elp
🚹 🕒 🖩 🞒 🔍 🥆 🛛 X	। 💼 🛛 🔍 🔍 🔋 🎯 🕨 🏶 🚳 🛛 Build target: Debug 💿 🚽 🗄 🏰 🏠 🔭 🥐 🔇
**	→ [main() : int
Management X	main.c ×
 Projects Symbols Resol ► Workspace HellowWorld Sources main.c 	4 > void f(int data) { 5 printf("The value of x is %d\n", data); 7 } 9 int main() 11 - { 12 int x = 42; 13 14 14 printf("Hello world!\n"); 16 f(x); 17 > 18 > 2 Odde::Blocks Q Search results 9 Build log 9 Build messages 9 Debugger ×
	Child process PID: 5476 At C:\Users\alessio\Desktop\HelloWorld\HellowWorld\main.c:12 At C:\Users\alessio\Desktop\HelloWorld\HellowWorld\main.c:14 At C:\Users\alessio\Desktop\HelloWorld\HellowWorld\main.c:16 At C:\Users\alessio\Desktop\HelloWorld\HellowWorld\main.c:6 At C:\Users\alessio\Desktop\HelloWorld\HellowWorld\main.c:17 Command:
WIND	OWS-1252 Line 17, Column 1 Insert Read/Write default

S3.5 – Inspecting Variables Values on the go

The other main feature of a debugger is to allow you to keep an eye on a list of variables as you step through the program. Let's finish the previous run and start a new one with $\boxed{\text{Debug}} \rightarrow \boxed{\text{Start}}$.

<pre>Projects Symbols Resor ></pre>	H main.c [HellowWorld] - Code::Blo	cks 10.05					
<pre>Management Management Management Management Projects Symbols Reso.</pre>	<u>File Edit View Search Project</u>	<u>B</u> uild <u>D</u> ebug <u>w</u>	xSmith <u>T</u> ools P <u>l</u> ug	ins <u>S</u> ettings <u>I</u>			
Management X Management X Projects Symbols Resor Workspace Multiple Sources Main.c Source	1 C 🕞 🖬 🞒 🕲 🥆 🛛 🗶 🛙	h 🖍 🔍 🔍 🤅	🕽 🕨 🏇 🌚 Bi	uild target: Debug	-	· • • •	වී 🚠 එ 🖑 🌔
<pre>Projects Symbols Resor ></pre>	:[-	main() : int			
<pre>Workspace printf("The value of x is \$d\n", data); printf("The value of x is \$d\n", data); f f</pre>	Management X	main.c ×					
CVI Jerris Versia Ve	Workspace	6 7 8 9 10 in 11 = { 12 13 14 4 III Logs & others Logs & others Code::Blocks Adding file: b Starting debug done Registered new Registered new Setting breakp Debugger name Child process At C:\Users\al	<pre>it main() int x = 42; printf("Hello w printf("Hello w Search results oin\Debug\HellowWo gger: type: wxString type: STL String type: STL Vector oints and version: GNU PID: 704</pre>	orld!\n"); Build log rld.exe gdb (GDB) 7.3	Build messages	Debu	gger ×
	C:\Users\alessio\Desktop\Hello\ WIN	DOWS-1252 Line	e 12, Column 1	Insert	Read/M	Vrite (default

We are going to open the window holding all our variables monitoring by selecting **Debug** \rightarrow **Debugging Windows** \rightarrow **Watches**.



This panel shows all the local variables, here x only, and functions arguments if we are in a function accepting parameters, here none. We haven't stepped through the line of our program initializing x so it is right now holding a random value.

Note for version 12.11 or later; please note that the Watches window has been updated starting in version 12.11. The same information is displayed but the layout is a bit different. Do not let that confuse you.

Eile Edit View Sea <u>r</u> ch Project	Build Debug wx		ns <u>S</u> ettings <u>I</u> ild target: Debug		₩₩∂?	A vet
	· · · · · · · · · · · · · · · · · · ·		main() : int			
			main(): int			
lanagement X	main.c ×					
Projects Symbols Resol □ • • • •	5 6 7 8 -}	printf("The val	ue of x is %	d\n",data);		
Sources	9	<pre>main() int x = 42;</pre>				
	14 14	printf("Hello w	orld!\n");			
	() • () • () • ()					F
	Logs & others					
	Code::Blocks	Search results	S Build log	Build messages	Debugger 🗙	
	Starting debugg		Build log	touro messages	bebugger X	
	<pre>done Registered new type: wxString Registered new type: STL String Registered new type: STL Vector Setting breakpoints Debugger name and version: GNU gdb (GDB) 7.2 Child process PID: 704 At C:\Users\alessio\Desktop\HelloWorld\HellowWorld\main.c:12 At C:\Users\alessio\Desktop\HelloWorld\HellowWorld\main.c:14</pre>					
	Command:				-	

Now we've stepped through the initialization, look at your watch window now.



The red indicates what was modified since the last line which was executed. In addition to watching the variables the debugger finds for you, you may also add your own watches by selecting Debug \rightarrow Edit Watches.

Note for version 12.11 or later; You may now just type the name of the variable you want to watch on the last line of the watches window.

The variables we specify by hand are listed at the bottom of our watch window