how to use codeblocks

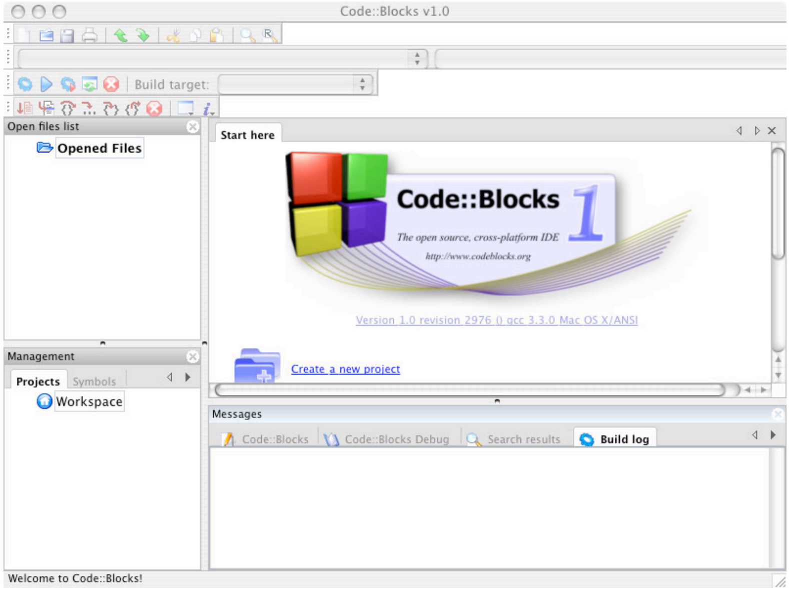
Note that the screen images below were taken on a Mac. Except for the last step (running the program), the steps should be the same when running CodeBlocks on a Windows or Linux machine.

---

step 1: starting up CodeBlocks

The picture on the right shows the screen when you first start up CodeBlocks. It has a number of different parts to it. Depending on what you are doing with CodeBlocks, different parts of the window will be filled in with information. These will be explained below, as they come up.

Note that CodeBlocks, like most IDEs (Integrated Development Environments), has many, many different things you can do with it, and hence many different menu options. Don't worry about all of them --- only worry about the ones you need to use. Those are outlined in the steps below.

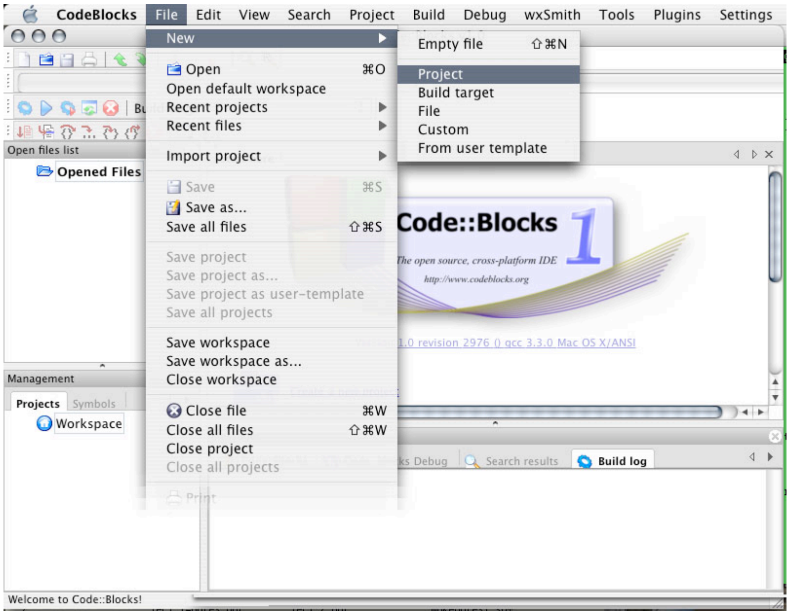


---

step 2: opening a Project

Programs in CodeBlocks (and most IDEs) are organized into entities called ***projects***. The first thing you need to do when you start up CodeBlocks is open a project. This could be an existing project, or a new project. In the example here, we will open a new project.

As shown in the screen on the right, from the **File** menu, select **New** and then **Project**.

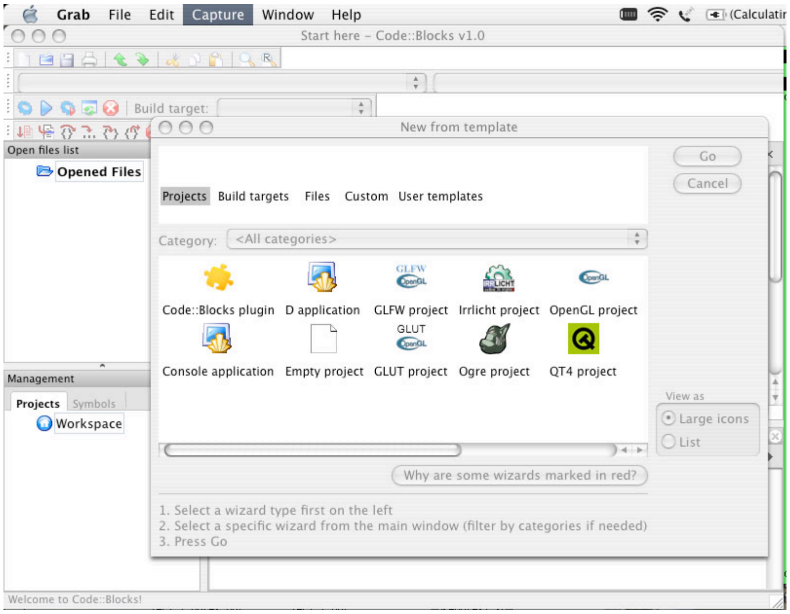


---

step 3: selecting new Project target

When you create a new project, you need to specify it's ***target***. The target is how you want to run (or "execute") the program you are creating. In the example here, we are going to run the program from the Mac OSX Terminal *command line* (see the last step). If you are running CodeBlocks in Linux or Windows, you can probably run the program without going to a terminal window (but we'll update this document with that information when we have it...).

When you select "File - New - Project", the small window shown on the right will pop up in front of your main CodeBlocks window. Click on **Console application** and then click on the **Go** button.

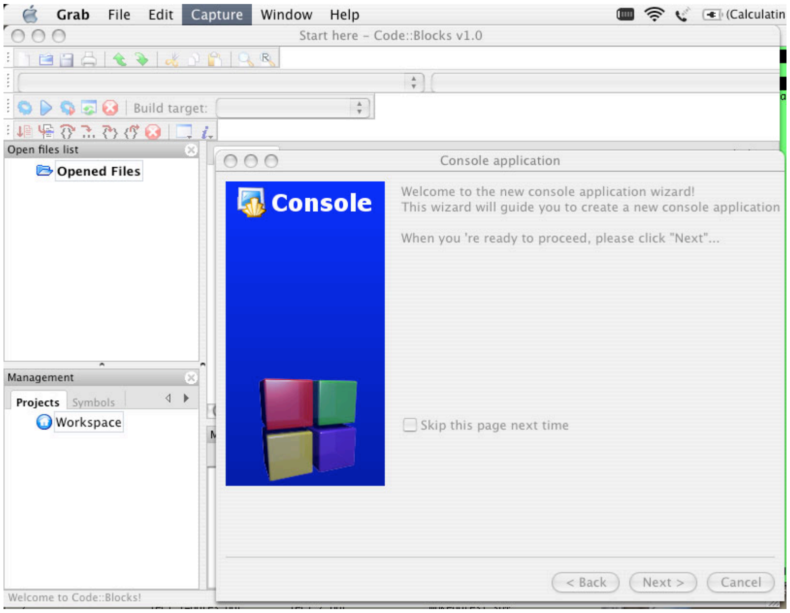


1. Select a wizard type first on the left
2. Select a specific wizard from the main window (filter by categories if needed)
3. Press Go

step 4: setting up a console application

After you have selected the console application as the target for your project, you will see the console wizard, as shown on the right. You can just click on the **Next >** button.

Note that you can click on **Skip this page next time** once you are comfortable with setting up console applications.



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

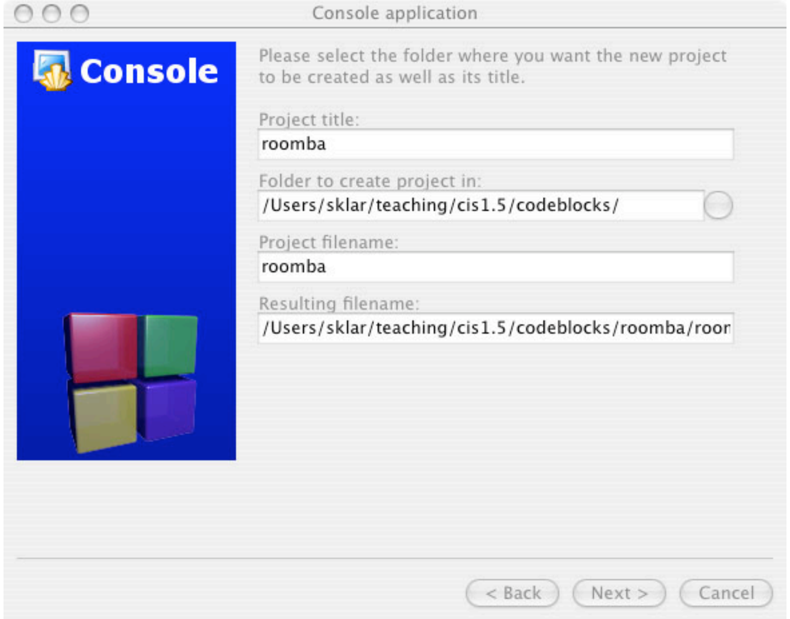
step 5: specifying the project title and location

Now you need to enter a title for the project and specify the location (i.e., folder) where the project will be stored on your computer.

As shown in the screen on the right, in the box labeled **Project title:**, type the name of your project. In this example, ours is called *roomba*.

Then specify the name of the folder where you want the project's files to be stored. In this example, our folder is called */Users/sklar/teaching/cis1.5/codeblocks*. This means that a folder will be created called "roomba" that is a subfolder (i.e., underneath) the /Users/sklar/teaching/cis1.5/codeblocks folder.

Then click on the **Next >** button.



Please select the folder where you want the new project to be created as well as its title.

Project title: roomba

Folder to create project in: /Users/sklar/teaching/cis1.5/codeblocks/

Project filename: roomba

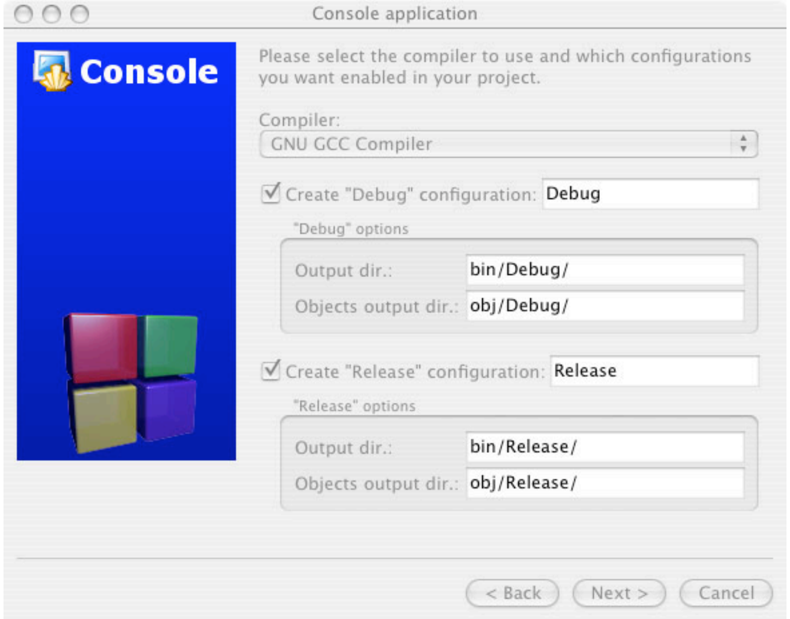
Resulting filename: /Users/sklar/teaching/cis1.5/codeblocks/roomba/roor

step 6: specifying the compiler and debugger

You can specify the *compiler* and *debugger* to use with CodeBlocks. The compiler is the translator that converts your program from C++ to machine language. The debugger is a program that can help you figure out where you might have bugs (i.e., mistakes) in your program.

Use the default values for these. Don't worry about changing them. And for now, you don't really need to worry about what they are.

So, when you see the screen shown on the right, you just need to click on the **Next >** button.



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

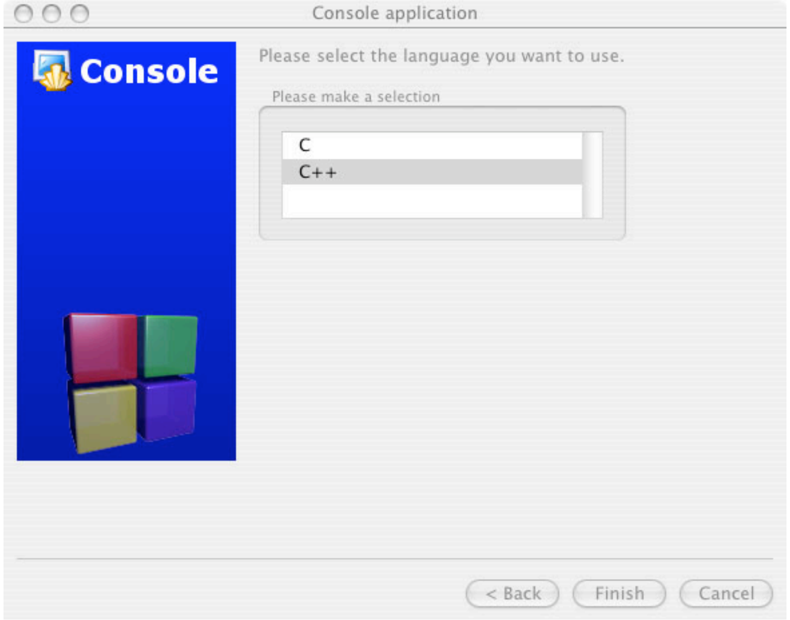
Output dir.: bin/Release/

Objects output dir.: obj/Release/

step 7: specifying the language

You can also specify the language that you are writing your program in. CodeBlocks works with both the **C** and **C++** languages.

For cis1.5, we are using C++. So select **C++** on the screen shown to the right, and then click on the **Finish** button.



Please select the language you want to use.

Please make a selection

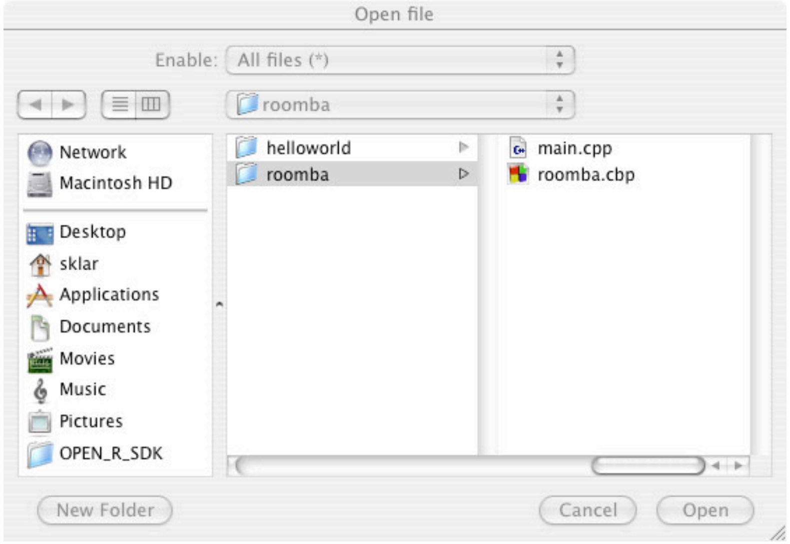
C

C++

step 8: selecting the project

Now you have created the project. CodeBlocks creates a ***main*** source code for you when it creates the new project. So now you have to open that source code file. First, you have to select the project folder where the source code file is stored.

Select **File** and then **Open**, and you will see a small window pop up in front of the main CodeBlocks window, as shown on the right. Highlight the project folder; the one we created here is called ***roomba***.



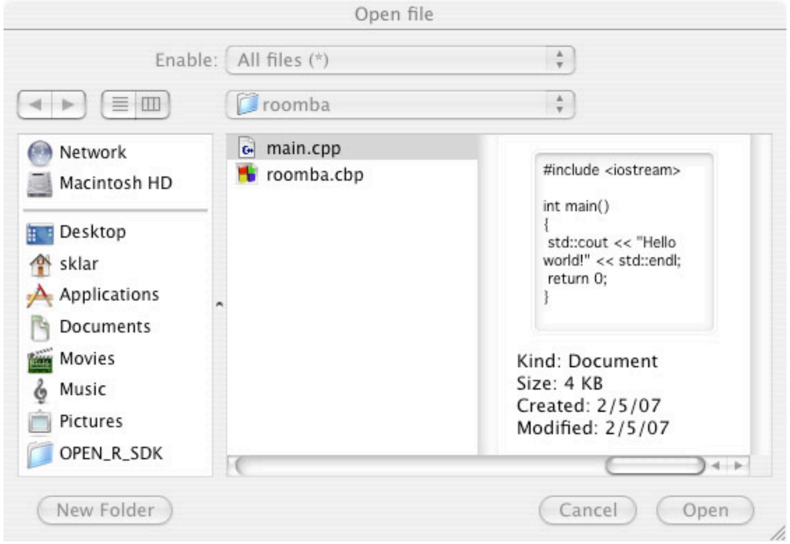
Enable: All files (*)

roomba

step 9: opening the source code file

Inside the project folder, there are two files, as shown in the screen on the right. Both are created by CodeBlocks when you create a new project. The first file in the list is called ***main.cpp***. This is the main source code file. Note that the file extension is ***.cpp*** --- this indicates that it is a **C++** source code file. This is the file we will edit (as instructed below). The second file in the list is called ***roomba.cbp***. The file extension, ***.cbp***, indicates that the file is a CodeBlocks Project.

Select **main.cpp** and click on the **Open** button (in the lower right corner off the window, as shown on the right).



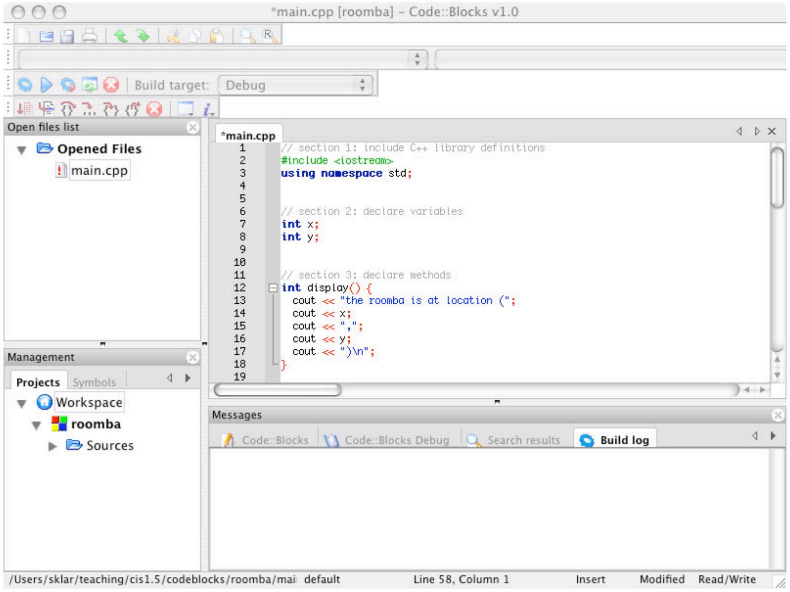
```
#include <iostream>

int main()
{
 std::cout << "Hello
world!" << std::endl;
 return 0;
}
```

Kind: Document
Size: 4 KB
Created: 2/5/07
Modified: 2/5/07

step 10: editing the source code file

Now you should see the source code, like we typed in during class, appear in the ***edit*** section of the CodeBlocks main window. As shown in the screen on the right, this is in the middle of the righthand side of the CodeBlocks main window.

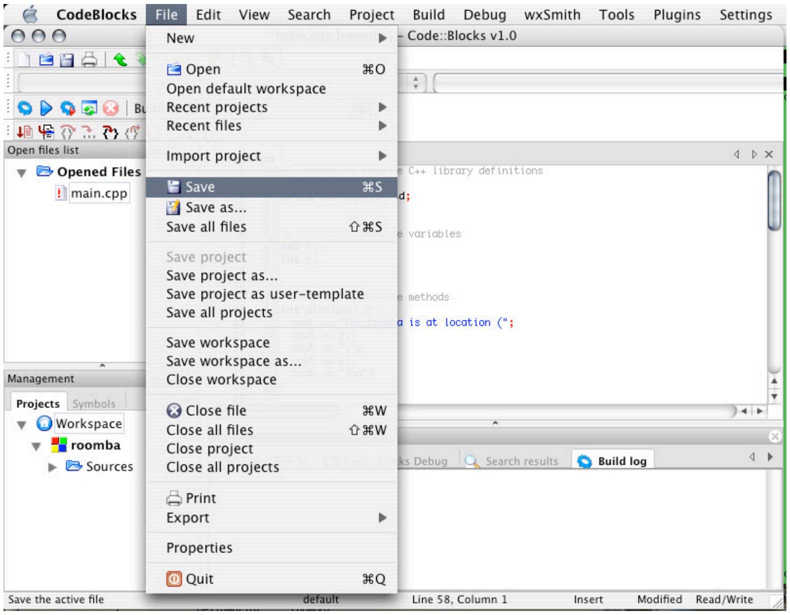
Note that the code is written in several colors. This feature is called ***syntax highlighting***. It is helpful. Punctuation marks are highlighted in red. C++ keywords are highlighted in blue. Pre-processor commands (we'll discuss these later) are highlighted in green. Comments are highlighted in grey. Identifiers (like names of variables and methods) are highlighted in black.



```
// section 1: include C++ library definitions
#include <iostream>
using namespace std;


// section 2: declare variables
int x;
int y;


// section 3: declare methods
int display() {
  cout << "the roomba is at location (";
  cout << x;
  cout << ",";
  cout << y;
  cout << ")\n";
}

```

step 11: saving the source code file

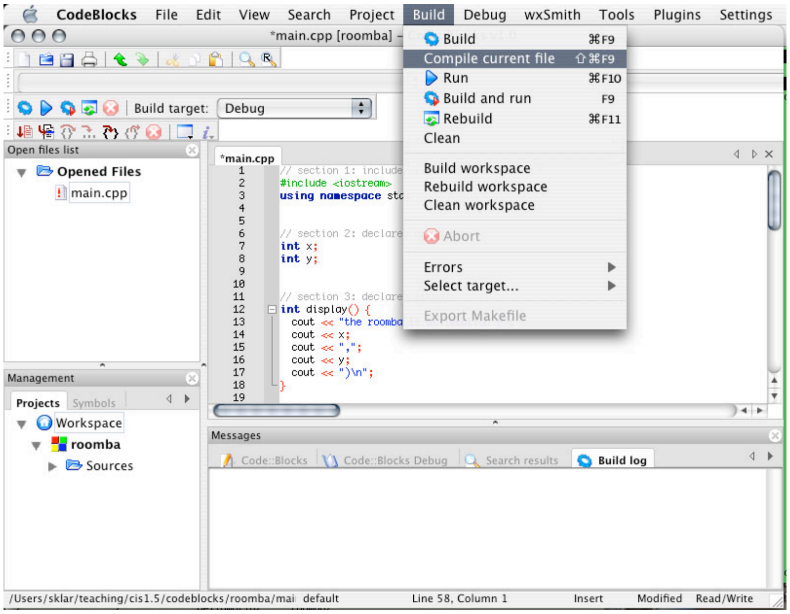
When you are done editing the file (and also periodically while you are editing it), you should ***save*** your work. Select **File** and then **Save**, as shown on the right.



step 12: compiling the source code file

In order to create a file that you can run from your source code file, you need to perform two steps. The first step is to ***compile*** the source code, and the second step is to ***build*** an "executable" file.

To perform the first step, select **Build** and then **Compile current file** from the menus, as shown.



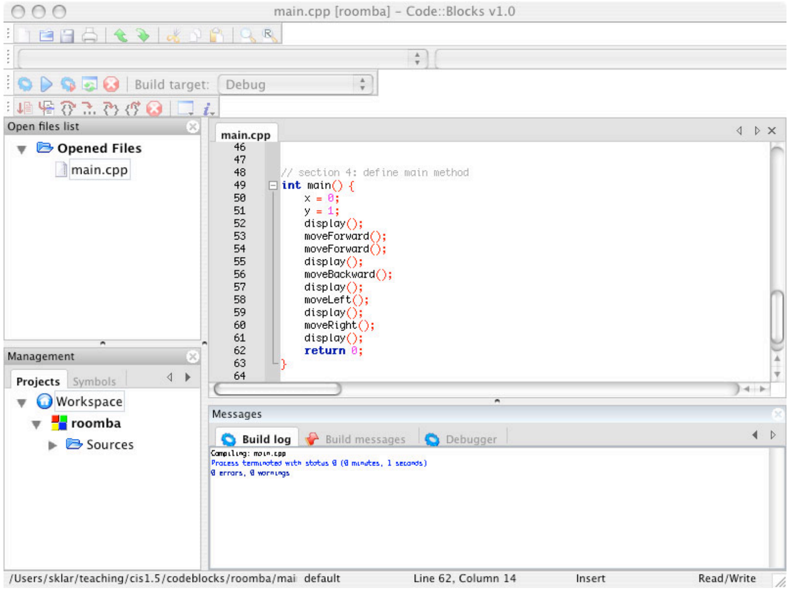
```
// section 1: include
#include <iostream>
using namespace st


// section 2: declare
int x;
int y;


// section 3: declare
int display() {
  cout << "the roomba
  cout << x;
  cout << ",";
  cout << y;
  cout << ")\n";
}
```

step 13: checking the compiler messages

When the compiler runs, messages will show up in the section of the CodeBlocks main window that is labeled ***Build log***. As shown on the right, that is in the lower right corner of the CodeBlocks main window.

In our example, we have no error messages from the compiler. Yippee! (You know this because the end of the message says: **0 errors, 0 warnings**).

But normally, at least the first time you try to compile, you will probably have messages appearing there. You will need to read through each message and correct your source code file by editing it, saving it again and compiling it again--- until there are no more errors in the Build Log message window.


main.cpp [roomba] – Code::Blocks v1.0
Build target: Debug
Open files list
Opened Files
main.cpp
Management
Projects Symbols
Workspace
roomba
Sources
main.cpp
Messages
Build log Build messages Debugger
/Users/sklar/teaching/cis1.5/codeblocks/roomba/mai default Line 62, Column 14 Insert Read/Write

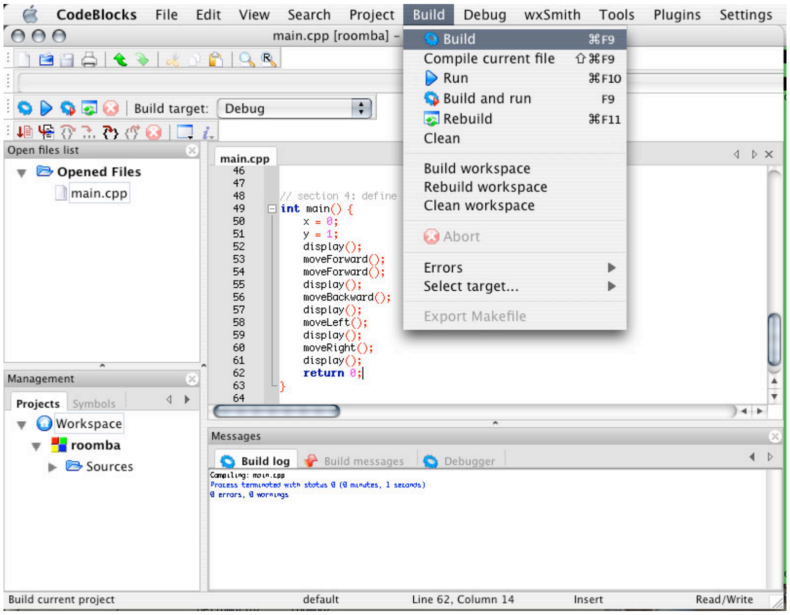
```


// section 4: define main method
int main() {
    x = 0;
    y = 1;
    display();
    moveForward();
    moveForward();
    display();
    moveBackward();
    display();
    moveLeft();
    display();
    moveRight();
    display();
    return 0;
}

```

```
Compiling: main.cpp
Process terminated with status 0 (0 minutes, 1 seconds)
0 errors, 0 warnings
```

step 14: building the application

After your program compiles correctly (with no errors and hopefully no warnings), you need to ***build*** an ***application*** from the compiler's output. This just means that you are converting the compiler's translated version of your source code into a file that your computer will know how to execute (i.e., run).

Do this by selecting **Build** and then **Build** (again) from the menus, as shown on the right.



```


// section 4: define
int main() {
    x = 0;
    y = 1;
    display();
    moveForward();
    moveForward();
    display();
    moveBackward();
    display();
    moveLeft();
    display();
    moveRight();
    display();
    return 0;
}

```

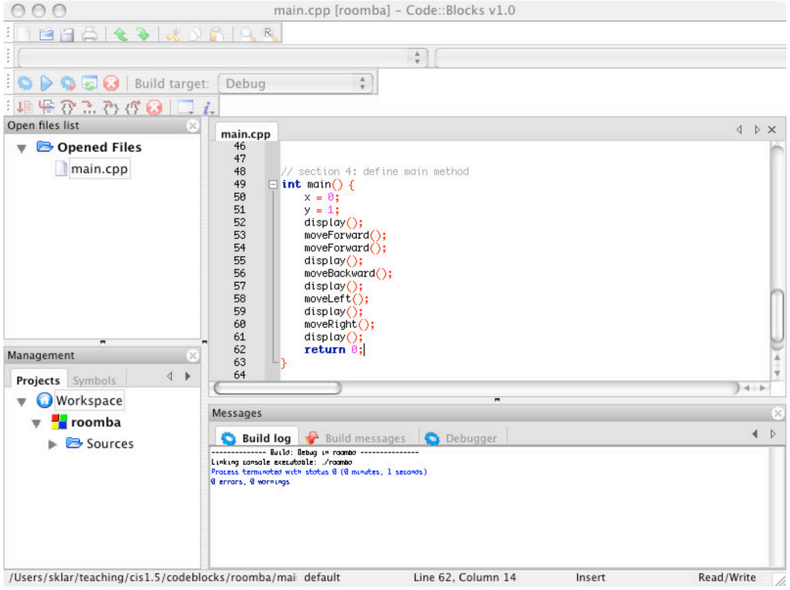
```
Compiling: main.cpp
Process terminated with status 0 (0 minutes, 1 seconds)
0 errors, 0 warnings
```

step 15: checking the build messages

Just as you checked to make sure that the compiler ran without errors, you also need to make sure that the builder ran without errors. The ***Build log*** section of the CodeBlocks window will indicate if there are any errors, as shown on the right. This is the same section of the window where the compiler writes its messages.

We don't have any builder errors in our example. If you did have errors, you would need to go back to the edit step (above), fix your errors, then save and compile again; then build again.

If the builder runs without errors, then you are ready to run the program and test if it does what you want it to do.



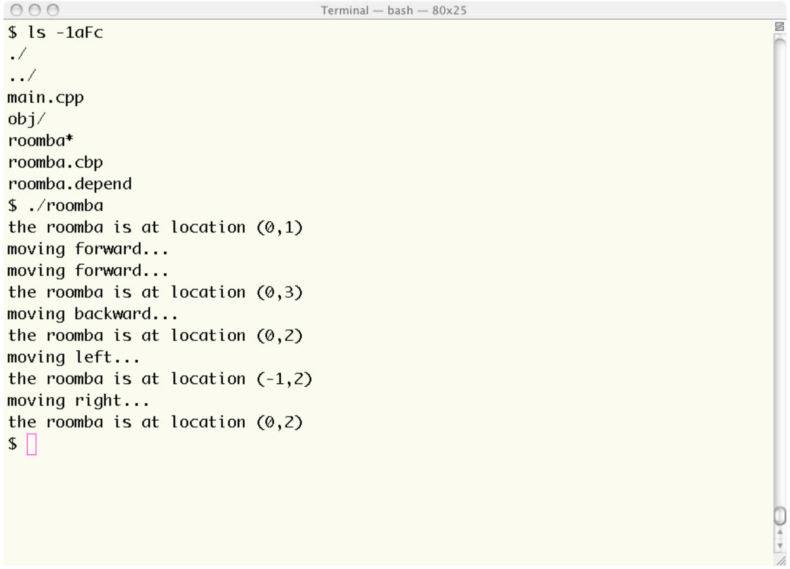
```


// section 4: define main method
int main() {
    x = 0;
    y = 1;
    display();
    moveForward();
    moveForward();
    display();
    moveBackward();
    display();
    moveLeft();
    display();
    moveRight();
    display();
    return 0;
}

```

```
-------------- Build: Debug in roomba ---------------
Linking console executable: ./roomba
Process terminated with status 0 (0 minutes, 1 seconds)
0 errors, 0 warnings
```

---

step 16: running the program

In Mac OS X, you will need to run the program outside of CodeBlocks, using the Mac Terminal application. This application is inside the top-level **Applications** folder, which you can locate using the **Finder**. **Terminal** is inside the **Utilities** folder that is inside the **Applications** folder. Double-click on the **Terminal** icon, and a window will open that looks like the one on the right.



```
$ ls -1aFc
./
../
main.cpp
obj/
roomba*
roomba.cbp
roomba.depend
$ ./roomba
the roomba is at location (0,1)
moving forward...
moving forward...
the roomba is at location (0,3)
moving backward...
the roomba is at location (0,2)
moving left...
the roomba is at location (-1,2)
moving right...
the roomba is at location (0,2)
$ 
```

You will need to use some ***command-line*** commands inside Terminal. The first thing you need to do is go to ("open") the folder where your program is saved. Do this by entering the letters **cd**, followed by the project's folder name (see step 5 above). In our example, this would be:

```
$ cd
/Users/sklar/teaching/cis1.5/codeblocks/roomba
```

followed by the **enter** (or **return**) key.

Note that the **$** is called the "prompt". It is an indication that the Terminal is waiting for you to enter a command.

Note that the "cd" and the folder name need to be entered on the same line, even though they are shown on two lines above (that's just because the screen isn't wide enough to show them on the same line). You only hit "enter" after you've entered both "cd" and the full path name.

Once you have entered the "cd" command, you should be in the folder where your project files are stored. You can check this by entering the **ls** command (followed by hitting the "enter" key). You should see a list of files, including **roomba** (with no extension). This is our *executable* program file.

To run the program, at the **$** prompt, enter **./** followed by the name of the executable program. In this example, you enter:

```
$ ./roomba
```

The output of running the program is shown.