

In-Class Exercise 1 – Getting Set Up & Started

The purpose of this In-Class assignment is to ensure that you

1. have the tools properly set up for future assignments,
2. know what documentation you must include in every program, and
3. know how to submit your programs.
4. can follow instructions

Read the whole assignment because you are responsible for the whole assignment. That's the first instruction to follow.

Grading

You will be graded on updating your Canvas profile as well as completing the programming assignment.

Fill out your Canvas profile

Fill out (or update) your Canvas profile so the instructor, TAs, and your fellow students can get to know who you are. **Include a picture of yourself**, one **where we can see your face** to get to know you!

At the top-left of the page, click on Account, then select Profile. Click the "**Edit Profile**" button, and fill in information in the "**Bio**" section. Fill out a short personal blurb – your major, hobbies or interests, why you're taking the course, and anything else fun that you'd like the rest of the folks in the class to know.

You must upload a headshot of yourself by clicking on the circle towards the left of the screen. Please use a picture where your face is clearly visible, not your favorite pic where your face is only 10th of the image. You may not use a logo, avatar, or any image that does not have you in it. Please, help me try to get to know you a little better!

Also update your Notifications (These settings are only suggestions.):

Notifications

Profile
Files
Settings
ePortfolios
Content
Migrations
SPOT

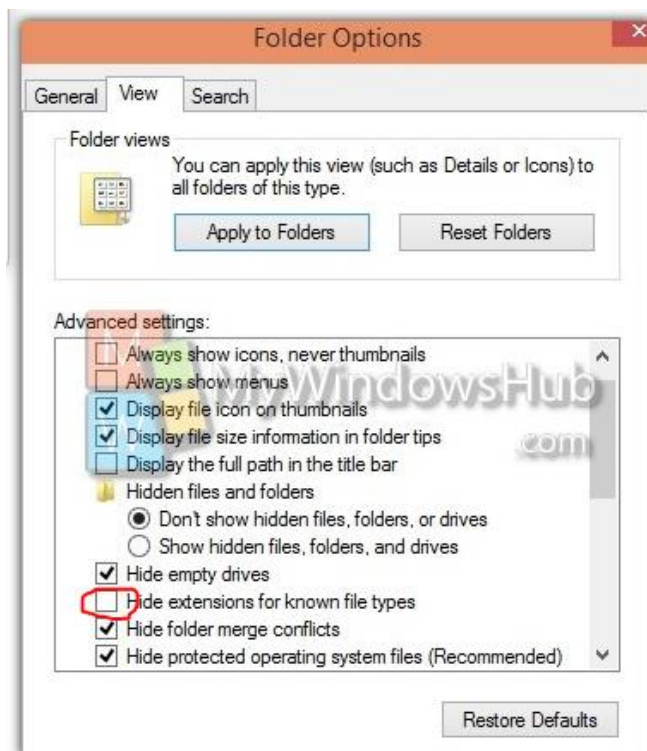
Notification Preferences

☒ Notify me right away ☐ Send daily summary ☐ Send weekly summary ☐ Do not send me anything

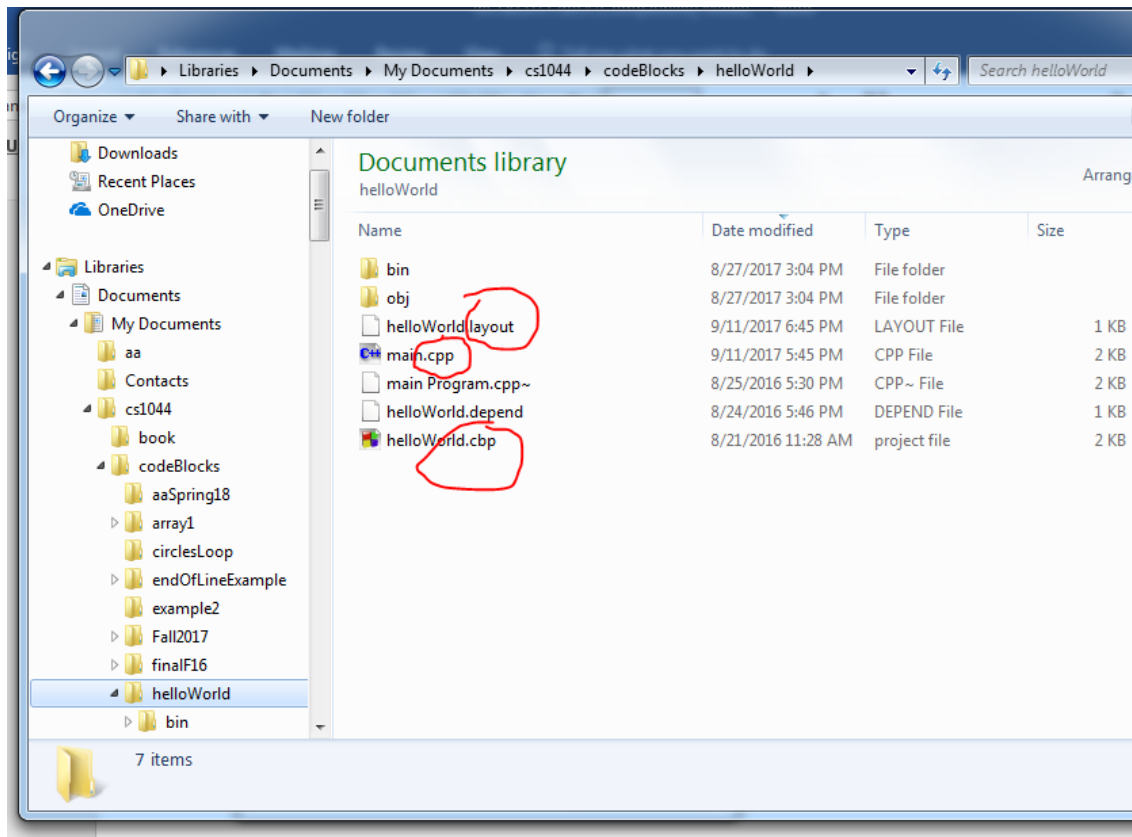
Course Activities	Email Address tedds1@vt.edu
Due Date	<input checked="" type="checkbox"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/>
Grading Policies	<input checked="" type="checkbox"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/>
Course Content	<input checked="" type="checkbox"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/>
Files	<input checked="" type="checkbox"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/>
Announcement	<input checked="" type="checkbox"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/>
Announcement Created By You	<input checked="" type="checkbox"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/>
Grading	<input checked="" type="checkbox"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/>
Invitation	<input checked="" type="checkbox"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/>
All Submissions	<input checked="" type="checkbox"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/>
Late Grading	<input checked="" type="checkbox"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/>
Submission Comment	<input checked="" type="checkbox"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/>

Set File Explorer (or Windows Explorer) to show file type

FOR WINDOWS (not Apple Macs): You need to show the file types (file suffixes or extensions) to display when you are prompted to select a file. To do this in Windows 7, use Windows Search (i.e. click the Start or Windows button) and type Folder options. For Windows 10 start File Explorer, select View and on the far right select Options. For both Windows 7 & 10, you should see something like this when you click on the View tab at the top of the window:



By default, the little box circled in red is checked. UN-CHECK it, please, then click OK. Now, your folder display (and open-a-file-to-submit-it pop up box) will show file extensions like circled here:



This helps you all to submit the correct files when you are submitting your programs for grading. This alleviates a lot of frustration for both students and us. Please do this.

Installing the development tools

We are using CodeBlocks this semester. If you need help installing it, you should see us during office hours before the second class. We will start the first program together in class.

Go to: <http://www.codeblocks.org/downloads>

Click on the **Download the binary release**

There are different ways to download and install Code::Blocks on your computer:

- **Download the binary release**
This is the easy way for installing Code::Blocks. Download the setup file, run it on your computer. Can't get any easier than that!
- **Download a nightly build:** There are also more recent so-called *nightly builds* available in the **forums** and **Jens' Fedora repository**. Other distributions usually follow provide consider nightly builds to be *stable*, usually, unless stated otherwise.
- **Download the source code**
If you feel comfortable building applications from source, then this is the recommended way. It puts you in great control and also makes it easier for you to update to newer versions contributing them back to the community so everyone benefits.
- **Retrieve source code from SVN**
This option is the most flexible of all but requires a little bit more work to setup. It gives you the latest code. We do at the time we do it. No need to wait for the next stable release to benefit from bug-fixing.

Click on the Download from **Sourceforge.net** for version for your computer/OS:
(I suggest codeblocks-17.12mingw-setup.exe for Windows because this includes the compiler.)

Please select a setup package depending on your platform:


- Windows XP / Vista / 7 / 8.x / 10
- Linux 32 and 64-bit
- Mac OS X

NOTE: For older OS'es use older releases. There are releases for many OS version and platforms on the **Sourceforge.net** page.

NOTE: There are also more recent *nightly builds* available in the **forums** or (for Ubuntu users) in the **Ubuntu PPA repository**. Please note that we consider nightly builds to be *stable*, usually.

NOTE: We have a **Changelog for 20.03**, that gives you an overview over the enhancements and fixes we have put in the new release.

NOTE: The default builds are 64 bit (starting with release 20.03). We also provide 32bit builds for convenience.

 **Windows XP / Vista / 7 / 8.x / 10:**

File	Date	Download from
codeblocks-20.03-setup.exe	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03-setup-nonadmin.exe	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03-nosetup.zip	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03mingw-setup.exe	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03mingw-nosetup.zip	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup.exe	02 Apr 2020	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup-nonadmin.exe	02 Apr 2020	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-nosetup.zip	02 Apr 2020	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-setup.exe	02 Apr 2020	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-nosetup.zip	02 Apr 2020	FossHUB or Sourceforge.net

Save the file and execute it to install the software. We will look at this in class.

Creating a new project in CodeBlocks

(Creating a project in XCode is in a separate file)

*****Never use any *spaces* in file names or folder names.** It causes errors submitting your programs.

Each programming assignment that you work on in this class, whether it is a small homework assignment or one of the larger projects, should exist in a **separate project** in CodeBlocks. This keeps your work well-organized and ensures that work that you do on one assignment does not interfere with work on another assignment. Please notice **WHERE** the project is being created, i.e. in which folder.

We will go thru creating your first project in class.

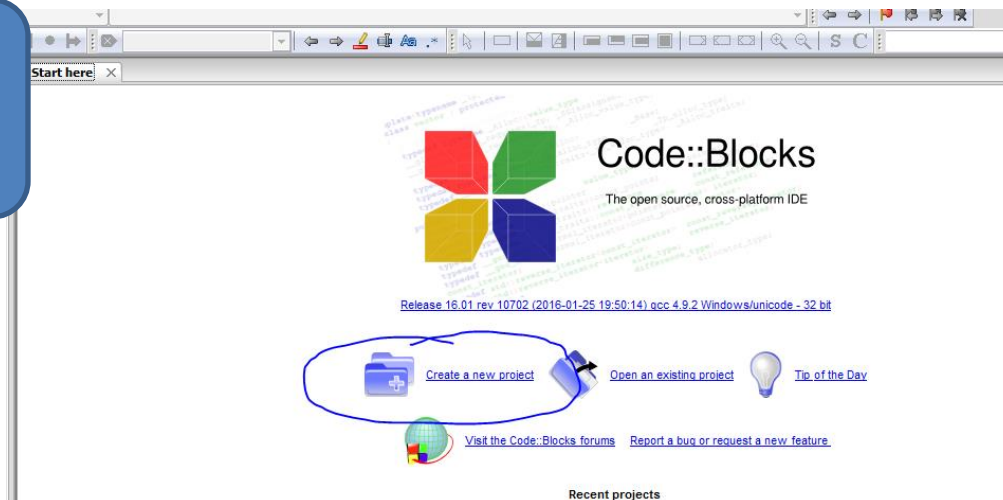
Writing your first program (in CodeBlocks)

Now you're going to write a very simple C++ program so that you can get yourself accustomed to the tools you'll be using in class. We'll be discussing all of the concepts below in lecture during the first couple of days of class, but it's certainly not a bad thing if you want to go ahead and get started!

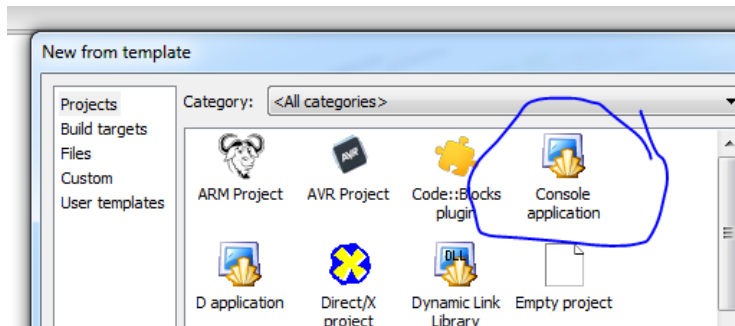
Create a new Project for every assignment (keep old ones around):

1. Click "Create a New Project"

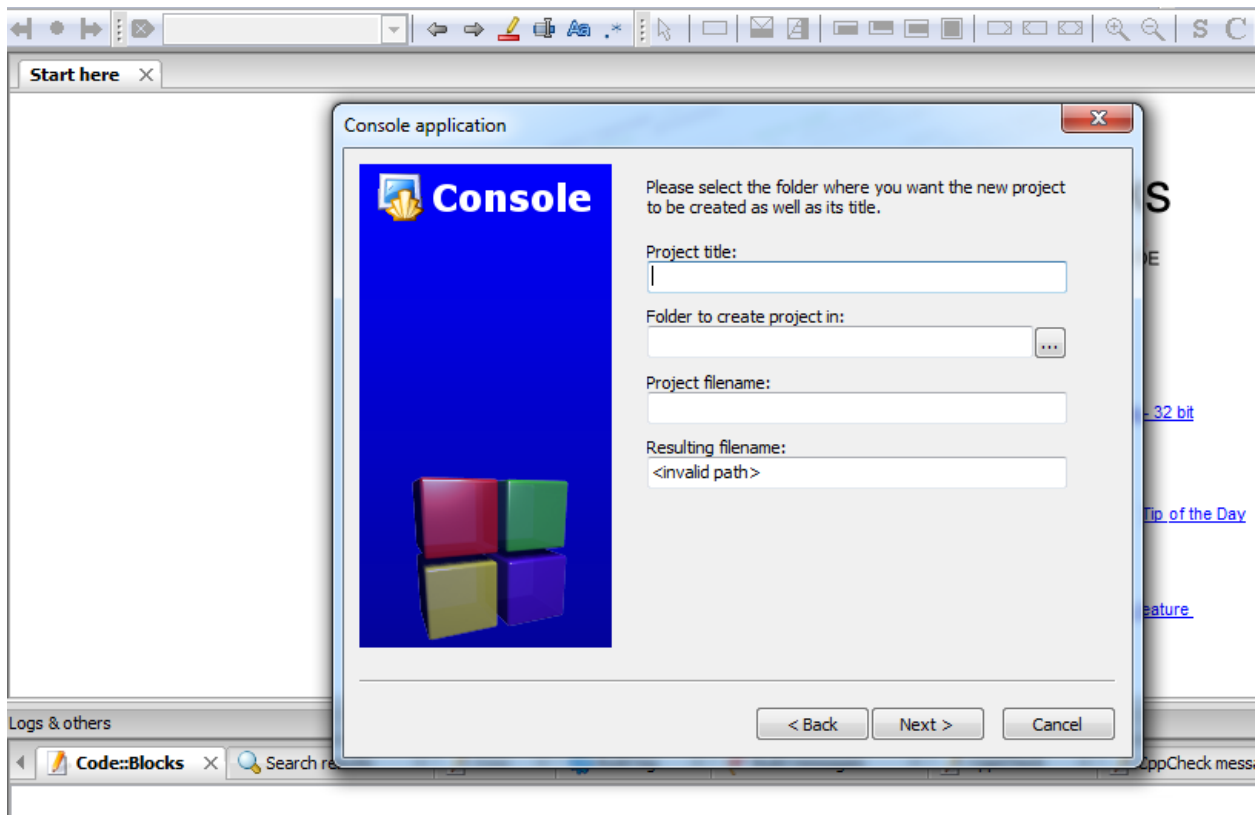
You MUST
do this for
every
assignment!



2. Double-Click Console Program

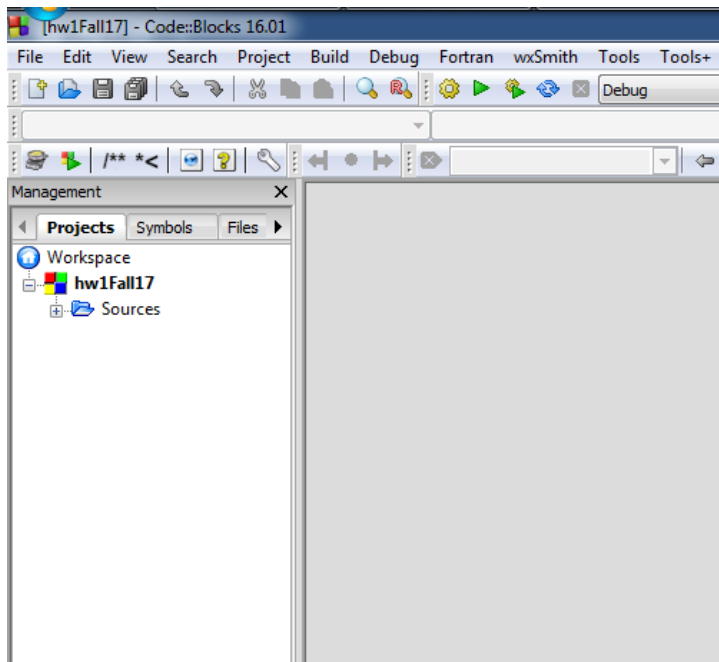


3. Click Next 1 or 2 time until you see

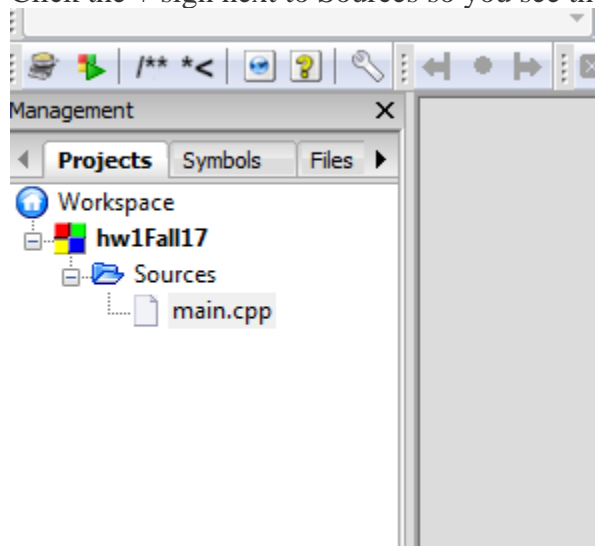


4. **Enter** a Project Title, like Homework1F17
5. **Select a Folder** where you want to save your projects, like MyDocuments\cs1044
6. Click Next
7. Click Finish

Your screen should look like this:



8. Click the + sign next to Sources so you see the source program files, e.g. main.cpp



9. If you want to, you can right-click on the file main.cpp, and select Rename... to change the file name to something like homework1.cpp (You **must** have the .cpp suffix!!!)
10. You have just created a Project that holds your program, and you are ready to start editing your program...

Edit the program file *filename1.cpp*:

1. Double-click the .cpp file you just created (e.g. main.cpp) to open it in the source code editor. CodeBlocks starts you off with some code, although other editors just start with an empty file. The following describes creating the code from an empty file, but you should read through it to understand

2. We need to use the `#include` keyword to include "header files" that contain features that we want to use in our programs. In this case, we need to include the `iostream` header, which lets us read and write text output. So, type the following as the first line of the file:

```
#include <iostream>
```

3. After that line, enter the following:

```
using namespace std;
```

As a rule, you'll want to include this line at the top of all of your programs, just **under** your `#include` statements. This feature adds into your program a "namespace" named `std`. You can think of namespaces as "folders" that contain functions and data types; for example, `std::cout` refers to something named `cout` inside the `std` namespace. By having this line of code (using `namespace std;`) you can write shorter code by leaving off the `std::` prefix.

4. Every C++ program must have a **function** named `main`. The `main()` function is the starting point of your program – the code within the braces `{ }` will be executed one statement at a time (in order) whenever you run it. Add this code to your source file:

```
int main()  
{  
    return 0;  
}
```

This function is the smallest valid C++ program that you can write. What does it do? Nothing! The `int` before the function's name is its **return type**; the `main` function must return an integer that tells the operating system whether the program ran successfully or not. In our case, we return 0, which means everything went ok. (Our programs in this class will never return anything other than 0.)

The parentheses after the name of the function represent **parameters** that the function takes. Parameters allow us to pass values into functions. In this case, since the parentheses are empty, we're saying that the `main` function does not take any parameters. (It optionally can, but we'll never use them.)

Lastly, the curly braces after the function are where we place the code that belongs to that function. As we'll see throughout the course, we can separate the logic of our programs into multiple functions that keep the responsibilities of different parts of our programs organized.

5. Now let's have this program do something. We'll use the `cout` stream to print this text to the screen: (from <http://www.imdb.com/title/tt0062622/quotes>, 2001: A Space Odyssey)

```
Dave Bowman: Hello, HAL. Do you read me, HAL?
```

```
HAL: Affirmative, Dave. I read you.
```

```
Dave Bowman: Open the pod bay doors, HAL.
```

```
HAL: I'm sorry, Dave. I'm afraid I can't do that.
```

```
Dave Bowman: What's the problem?
```

```
HAL: I think you know what the problem is just as well as I do.
```


Dave Bowman: What are you talking about, HAL?

HAL: This mission is too important for me to allow you to jeopardize it.

6. To accomplish this, type code like the following around each quoted line you want to print:

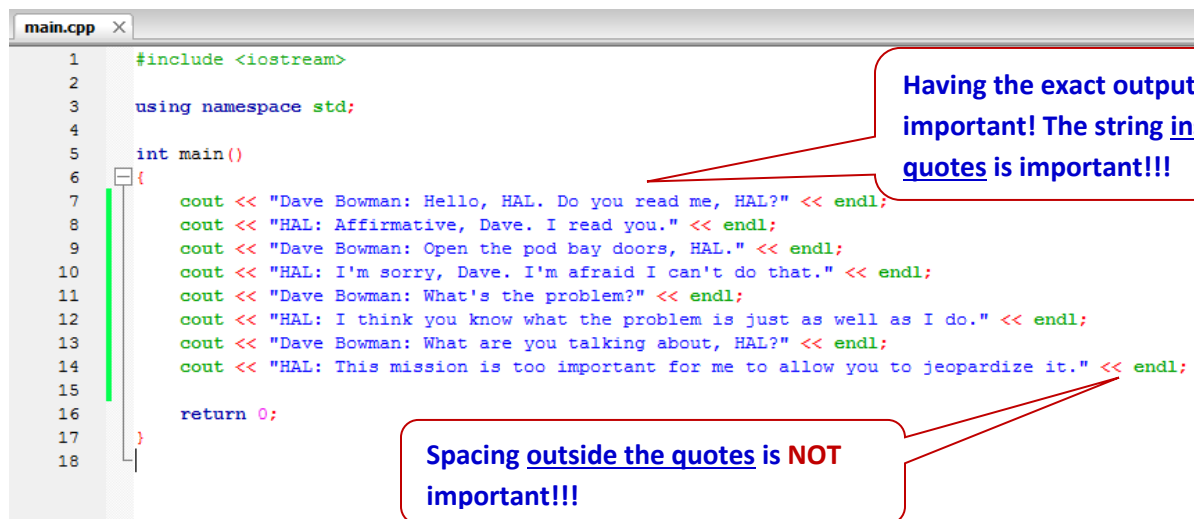
```
cout << "Dave Bowman: Hello, HAL. Do you read me, HAL?" << endl;
```

The << symbol is an **operator** that we use to write data to an output stream; in this case, cout, which represents the screen. (Other output streams can represent files on your disk.)

Lastly, the **endl** symbol says that the cursor should be moved down to the next line.

Copy the Hal/Dave text above **and Paste** the text into CodeBlocks, and type the “cout” and “endl” around the quoted lines!!!

7. Write cout statements for each line you want to print. Your code should look like this:



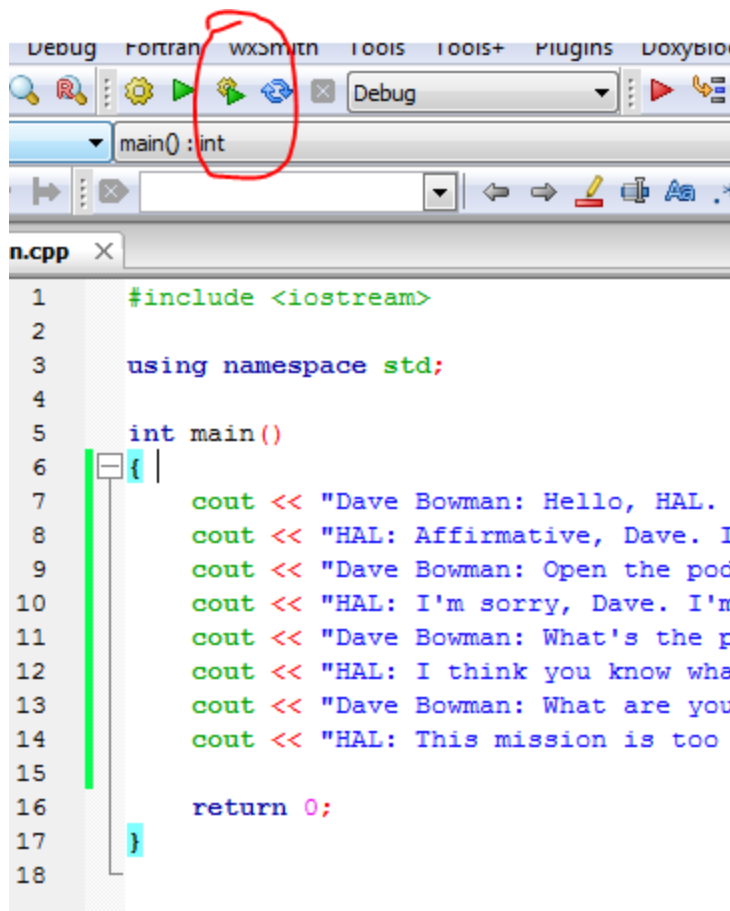
```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Dave Bowman: Hello, HAL. Do you read me, HAL?" << endl;
8      cout << "HAL: Affirmative, Dave. I read you." << endl;
9      cout << "Dave Bowman: Open the pod bay doors, HAL." << endl;
10     cout << "HAL: I'm sorry, Dave. I'm afraid I can't do that." << endl;
11     cout << "Dave Bowman: What's the problem?" << endl;
12     cout << "HAL: I think you know what the problem is just as well as I do." << endl;
13     cout << "Dave Bowman: What are you talking about, HAL?" << endl;
14     cout << "HAL: This mission is too important for me to allow you to jeopardize it." << endl;
15
16     return 0;
17 }
18
```

Having the exact output text is important! The string inside the quotes is important!!!

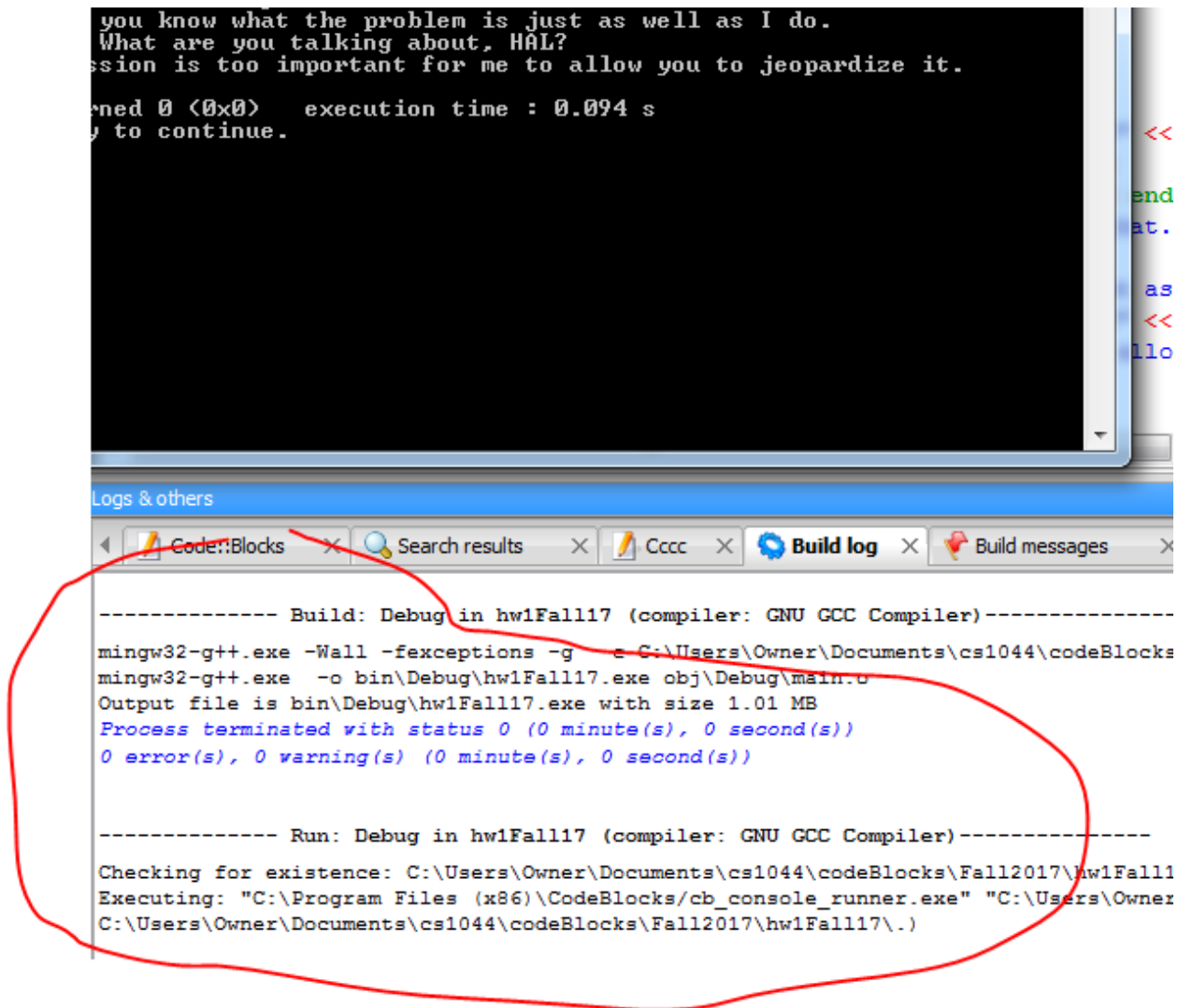
Spacing outside the quotes is **NOT** important!!!

8. Now test your program by compiling and running it. Click the “Build and run” button in the toolbar circled in the image below.

In XCode on the Mac, you have a window at the bottom where output is printed!



When you run your program in CodeBlocks, you get a black window that your program runs in. Beneath that popped up window in CodeBlocks, it shows you messages saying it has built and run your code, which I have circled below.



```
you know what the problem is just as well as I do.
What are you talking about, HAL?
ssion is too important for me to allow you to jeopardize it.

ned 0 (0x0)   execution time : 0.094 s
y to continue.
```

Logs & others

CodeBlocks Search results Cccc Build log Build messages

```
----- Build: Debug in hw1Fall17 (compiler: GNU GCC Compiler)-----
mingw32-g++.exe -Wall -fexceptions -g -o C:\Users\Owner\Documents\cs1044\codeBlocks
mingw32-g++.exe -o bin\Debug\hw1Fall17.exe obj\Debug\main.o
Output file is bin\Debug\hw1Fall17.exe with size 1.01 MB
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))

----- Run: Debug in hw1Fall17 (compiler: GNU GCC Compiler)-----
Checking for existence: C:\Users\Owner\Documents\cs1044\codeBlocks\Fall12017\hw1Fall1
Executing: "C:\Program Files (x86)\CodeBlocks\cb_console_runner.exe" "C:\Users\Owner
C:\Users\Owner\Documents\cs1044\codeBlocks\Fall12017\hw1Fall17\."
```

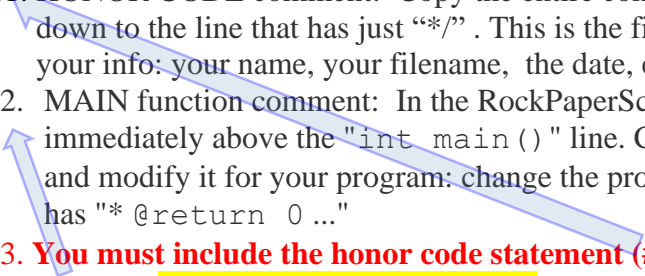
If you do not see the black window pop up, CodeBlocks (or XCode) prints errors that are in your code. The error messages mainly just tell you on which line CodeBlocks has a problem understanding what you wanted; frequently, your error is on the line above the one CodeBlocks indicates. If you have errors, you need to fix errors, and try building again:

9. Congratulations! You've just finished your first C++ program.

But you are not done yet!

10. Before you submit your program though, you must document your program and include an honor code statement. All of your programs must include an honor code statement, or you will lose major points! Go to the Canvas homepage and find the example code to copy a template of the documentation and honor code:

There is an example of how to format and comment your code on Canvas [here](#) [EXAMPLE Code: RockPaperScissors.cpp](#). This is in the Administrative "Module" on the Canvas homepage. (<https://canvas.vt.edu/courses/115374/modules/items/1000845>) .

- 
11. HONOR CODE comment: Copy the entire comment at the top of the file from “/**” down to the line that has just “*/” . This is the file header comment. Change it to have your info: your name, your filename, the date, etc.
 12. MAIN function comment: In the RockPaperScissors.cpp, go down to the /** comment immediately above the "int main()" line. Copy that entire comment from /** to */ and modify it for your program: change the program description but leave the line that has " * @return 0 ..."
 13. **You must include the honor code statement (#11) and the main program comment (#12) for every one of your programs, or you will lose points. Do this when you start writing your programs!**
 14. Step-by-step instructions will not be provided for each assignment. That is what you need to learn to do on your own. Perform steps similar to the previous steps for future assignments.

Submitting Your Work

We may or may not get to this point today! Once complete, submit your work to Web-CAT to be graded. There are instructions on the Canvas Homepage in the Administrative Module! We will also go thru this in class next session.