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

Version 8.02

Table of Contents

Introduction

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

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

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

You may find additional information regarding Code::Blocks at: http://www.codeblocks.org/

A complete manual for Code::Blocks is available here: http://www.codeblocks.org/user-manual

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

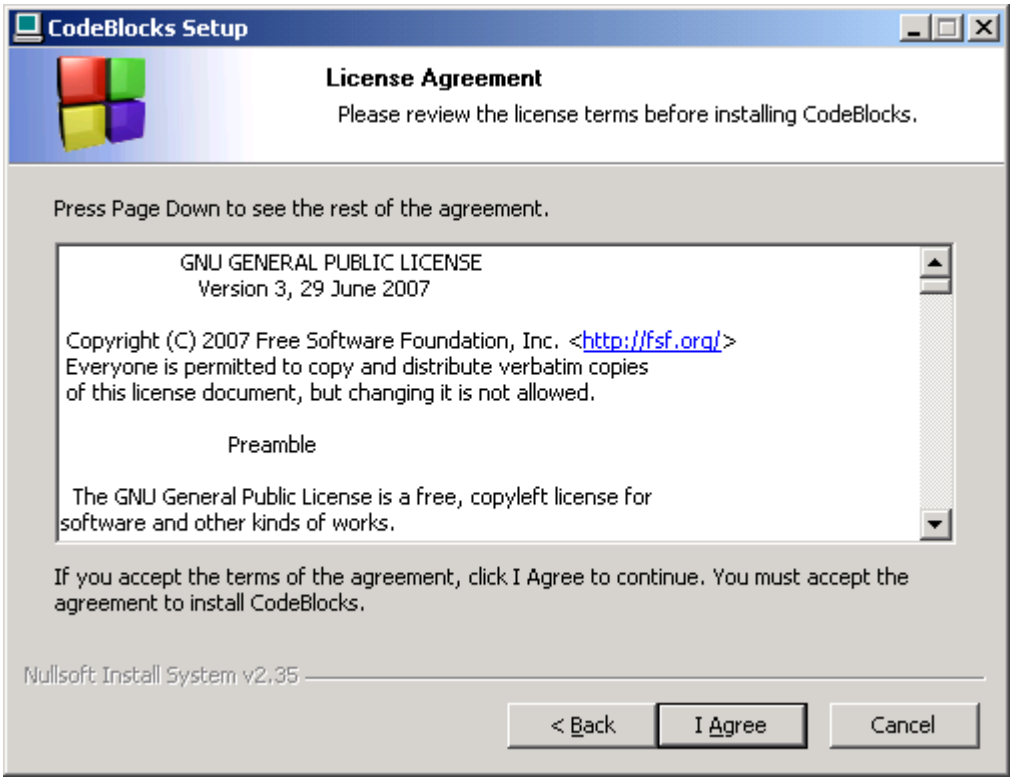
Save the file to your hard disk and remember its location. Proceed to the next page in order to continue the installation.

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

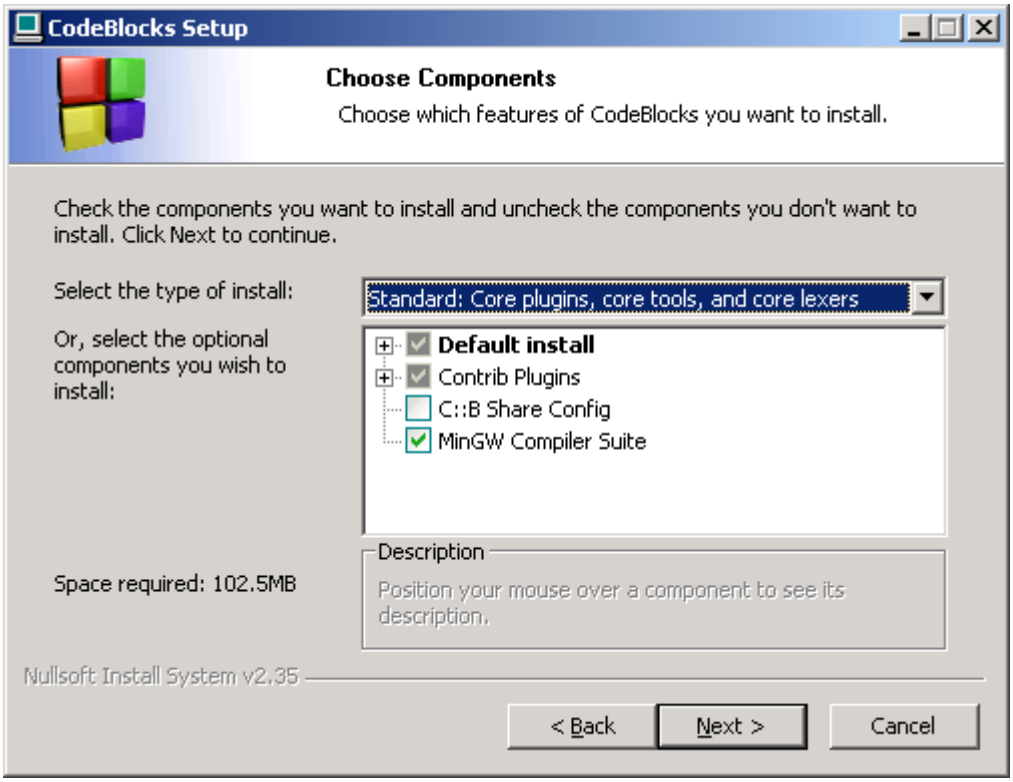
Step 2: Install the Software

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

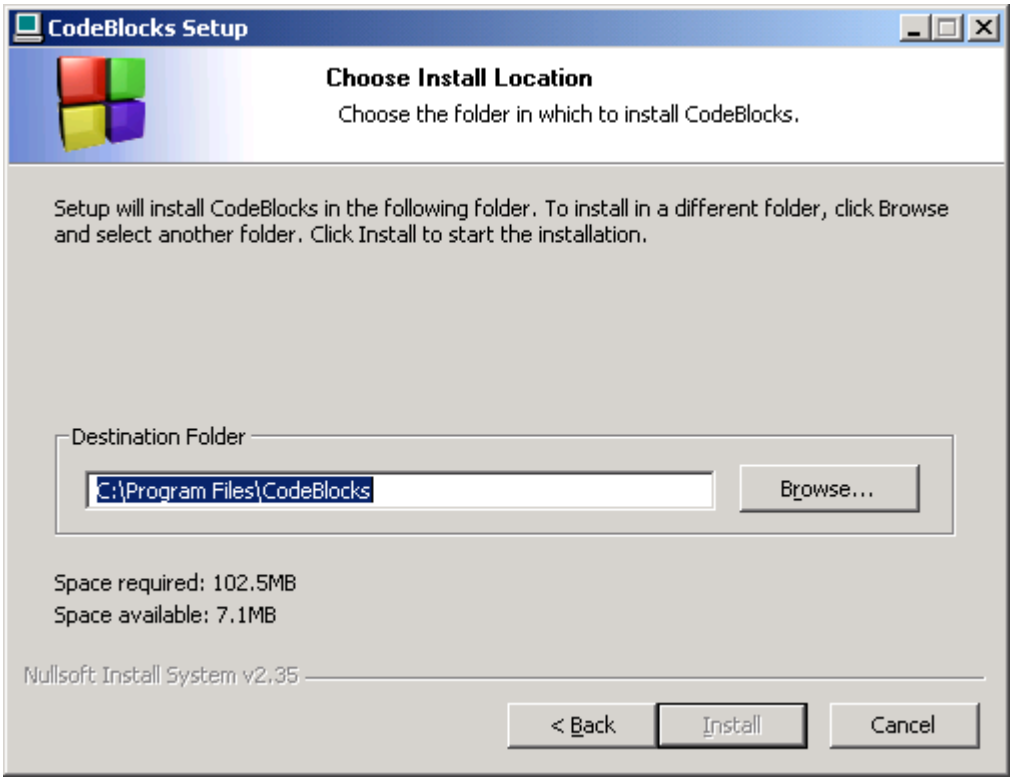
Click **Next.**



Select **I Agree**.



Take the default settings by pressing **Next**.



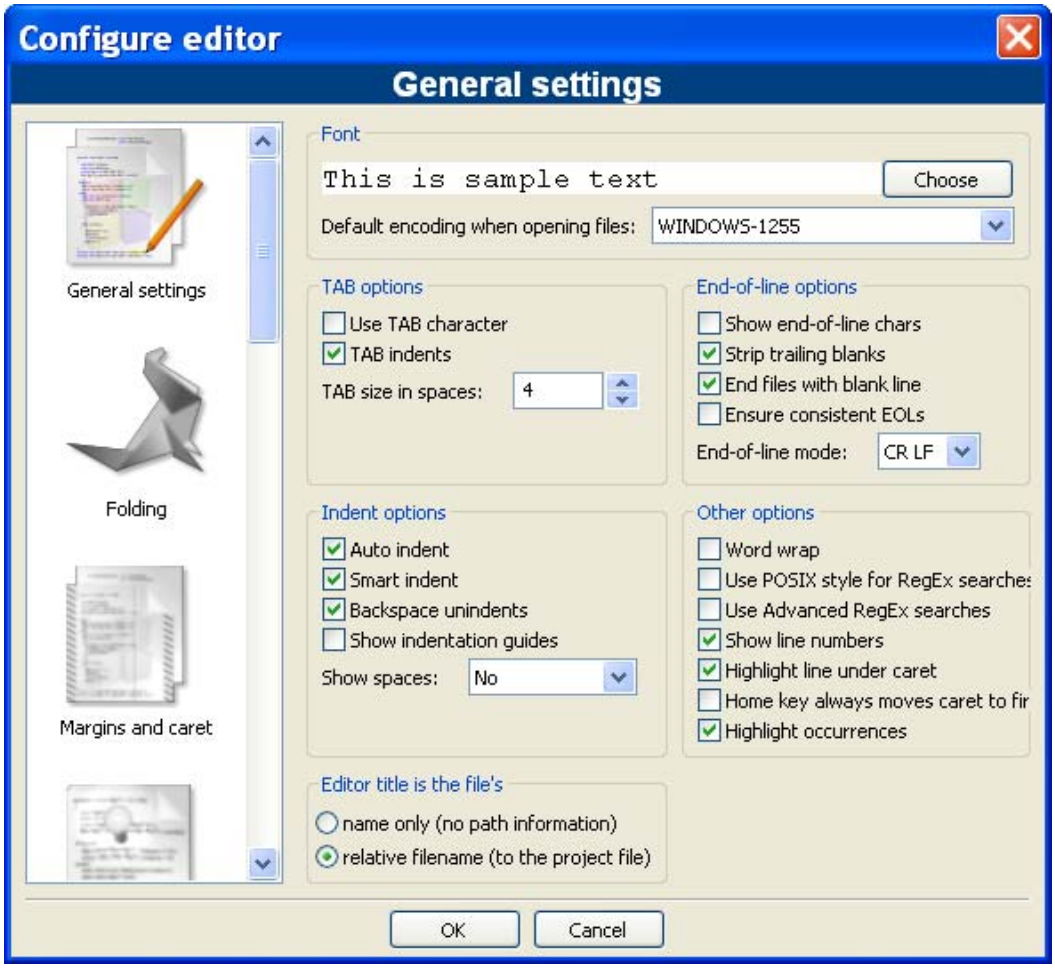
Take the default folder to install CodeBlocks to and then select Install.

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:

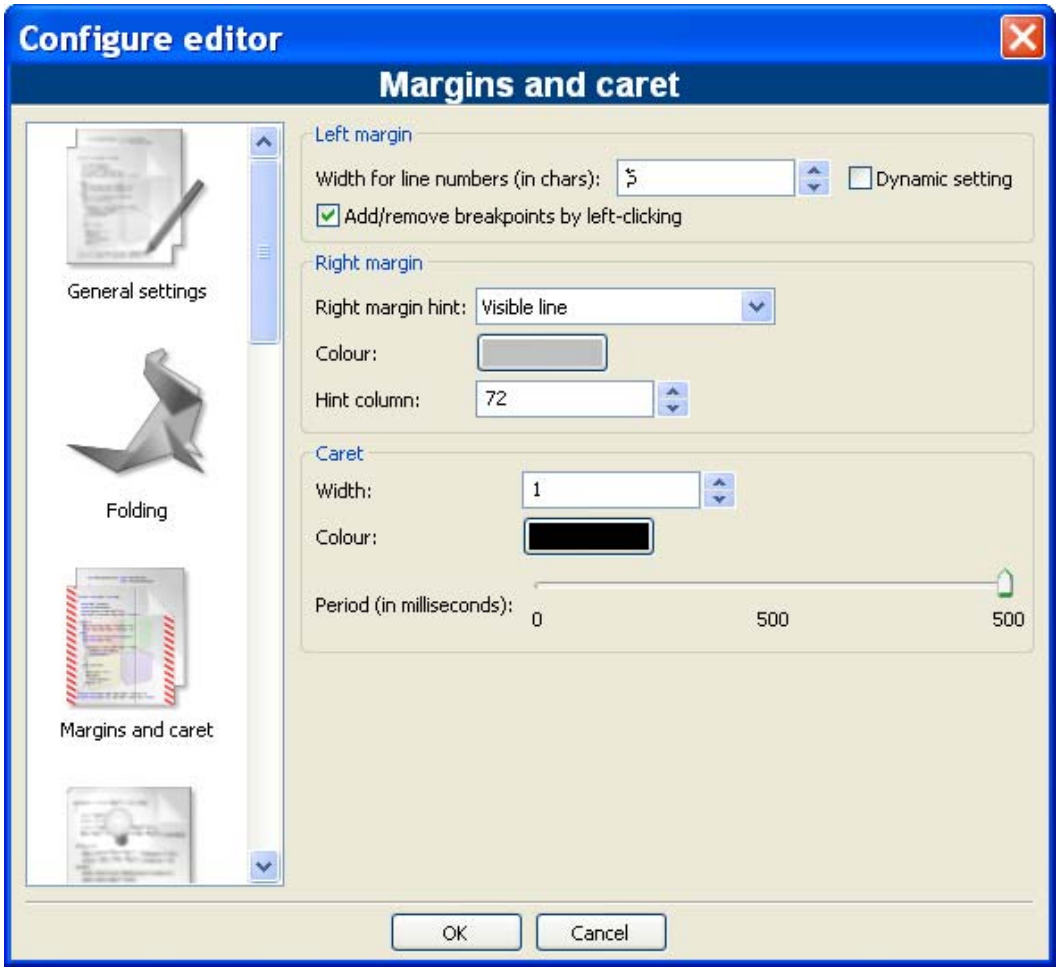
1. Configure the editor:
    a. Choose *Editor* from the *Settings* Menu
    b. Under the *General Setting* tab
        i. Change the font size to 10 or 12 point (Use the *Choose* button.)
        ii. Under *Other Options* place a check mark the following options:
            - "Show line numbers"
 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

Your screen should now look like the figure:



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

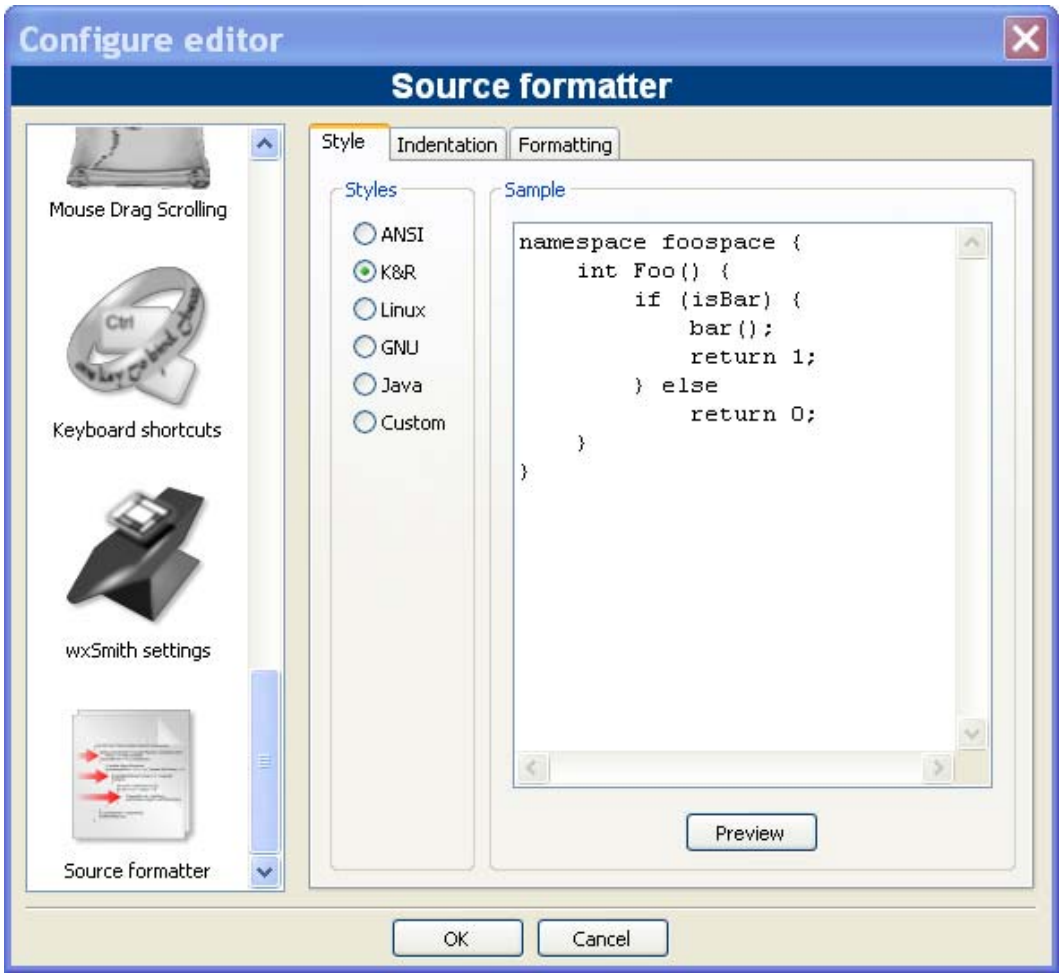
Your screen should now look like the figure:



d. Choose the Source formatter Caret tab on the right of the window.

   i. Select *K&R* from the style menu (Note: Your instructor may use a different style – if so, examine each style and choose the style that matches your instructor's preferred indentation.)

Your screen should now look like the figure:



e. Choose *OK* in order to save your customizations.

2. Configure the *Help* files

    a. Download the C++ help file from

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

    b. Save the cpp.htm file to your c:/Program Files/Codeblocks directory

    c. After saving the file, right-click on the file, choose *Properties*, and unblock the file so that it will be accessible to the Code::Blocks IDE.

    d. Choose *Environment* from the *Settings* Menu

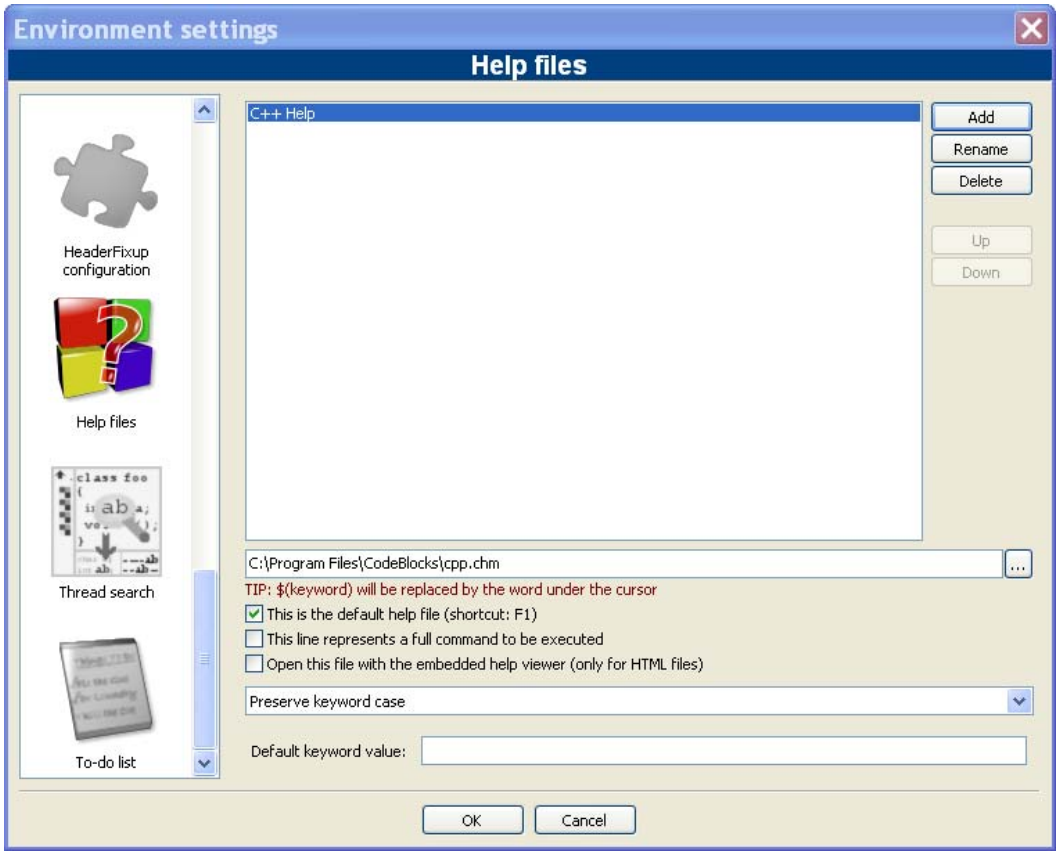
    e. On the tab to the left of the window scroll down to the *Help Files* section

    f. Choose Add and enter "C++ Help"

    g. Use the file browser button  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:



    i. Choose *OK* in order to save your customizations.

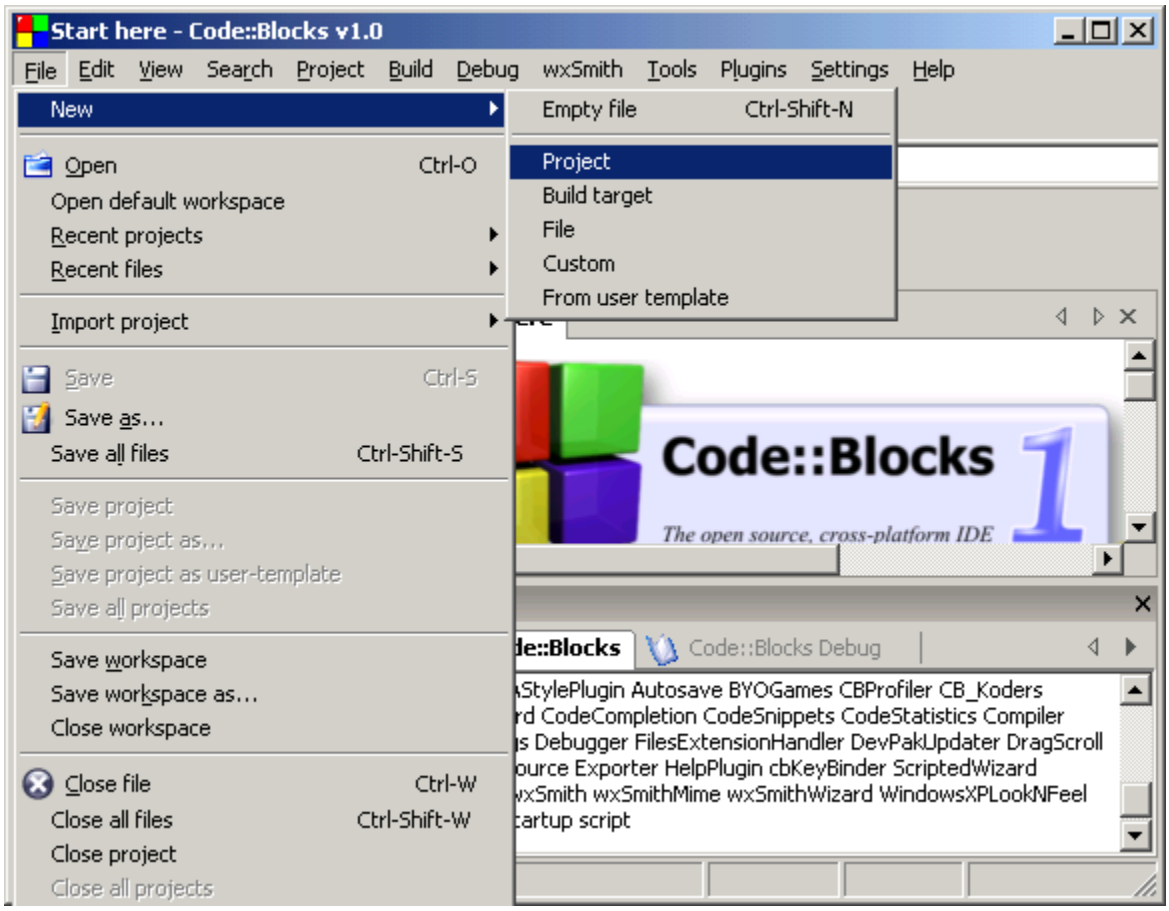
By pressing F1 you will now be able to obtain help on the word under your cursor.

First Project

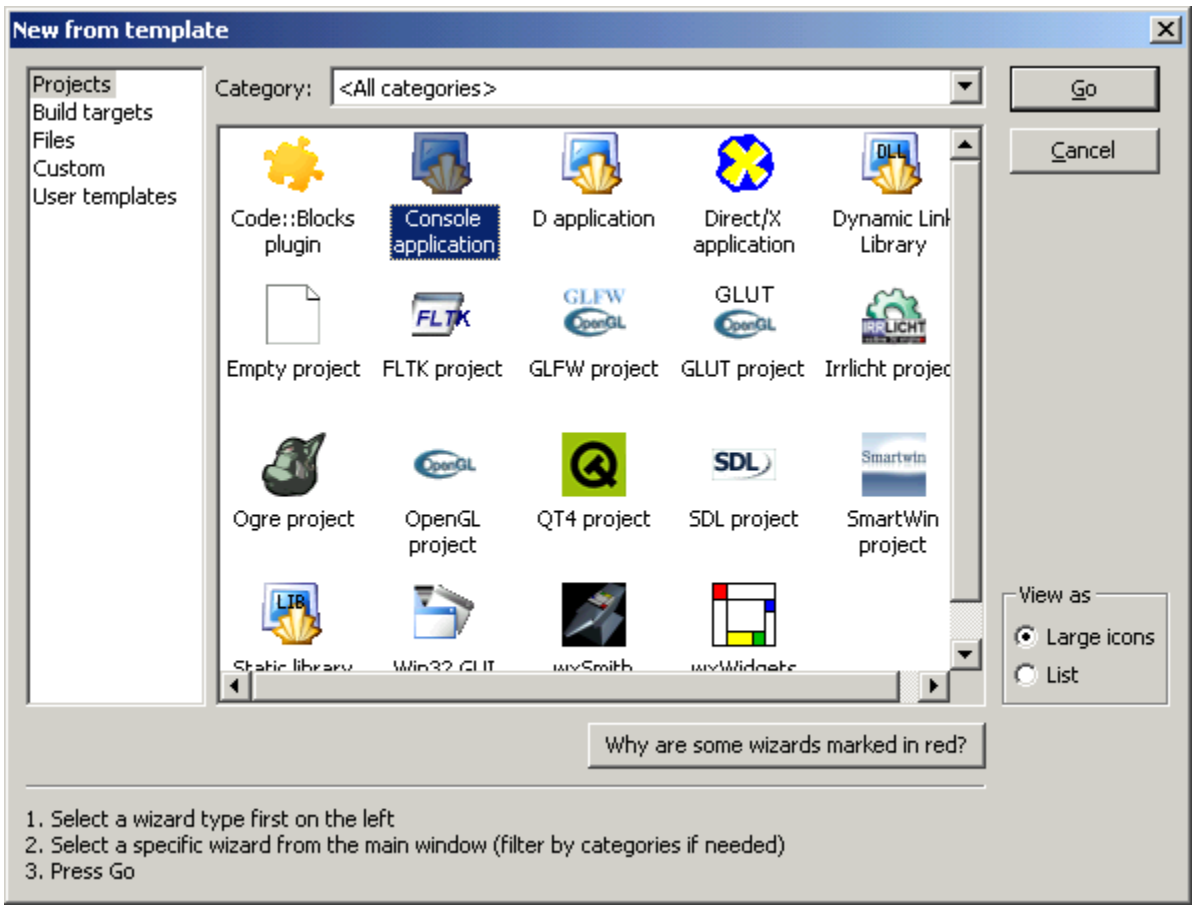
After you have finished downloading and setting up the Code::Blocks system, you can be in to write code. Code::Blocks creates what is called a **Workspace** to keep track of the **project** you are working on. It is possible for you to be working on multiple projects within your workspace. A **project**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

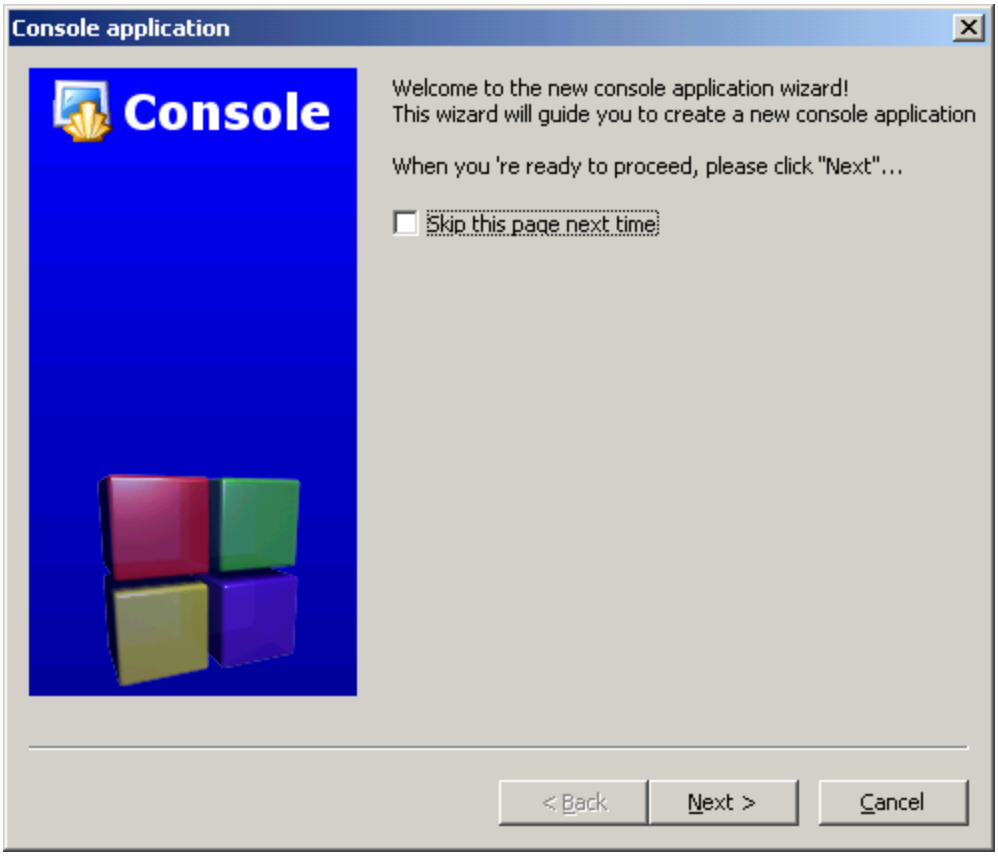
Setting up a **project** allows you to keep track of all the files in an organized way. When first starting out in computer programming, generally your projects will consist of a single source file. However as you gain experience and work on more complex projects, you will have projects containing many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

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

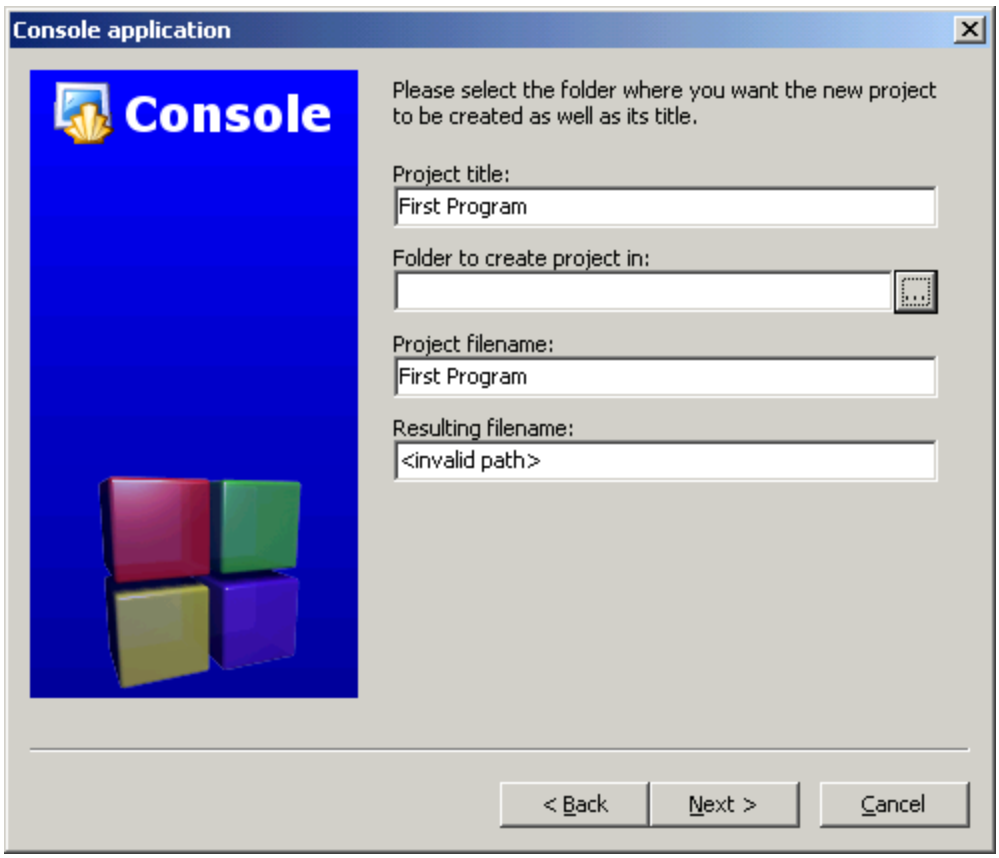


This will bring up the **New from template** window. Opening (clicking on) **Console Application** will then allow you to write a program on the console. The other application are for developing more advanced types of applications. After selecting **Console application,** click on the Go button to begin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



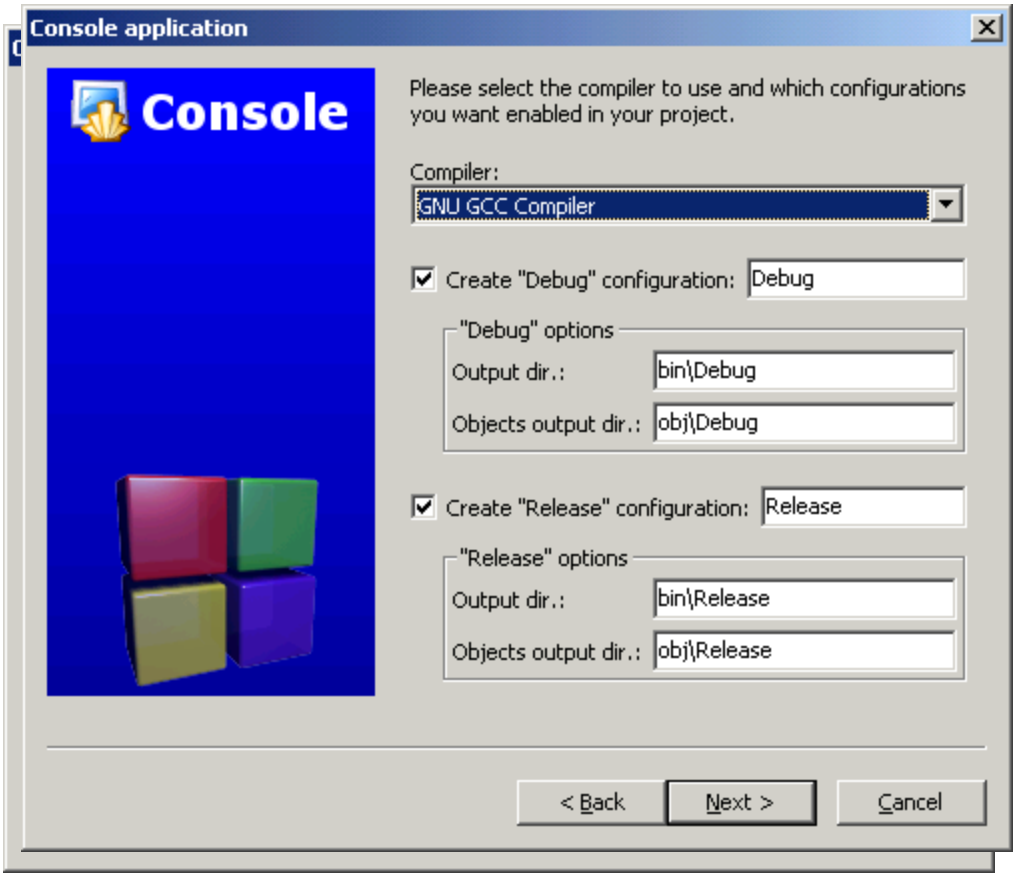


Press Next to go to the next step.



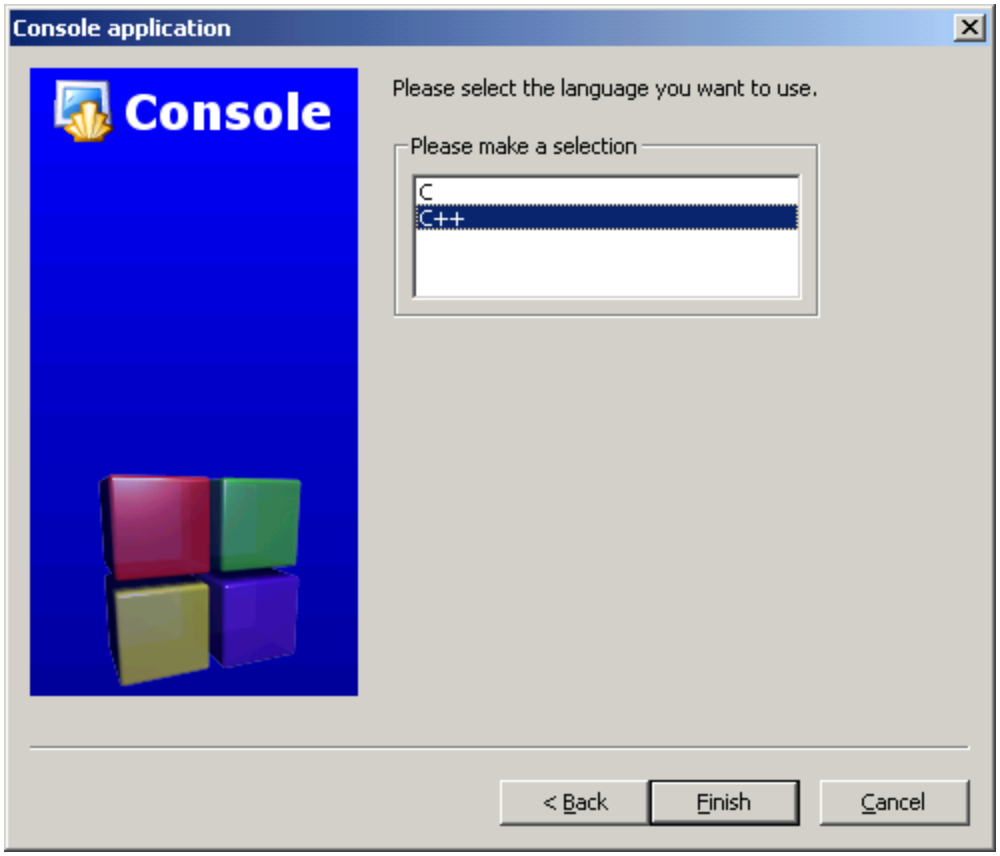
Start by filling in the Project Title. You will notice that the Project Filename automatically becomes the same name. If you wish, you can change the filename, but for simplicity leave it as is. To specify the location of the folder to contain the project, click on the "..." button (selected in the picture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

Press *Ok* after selecting My Documents

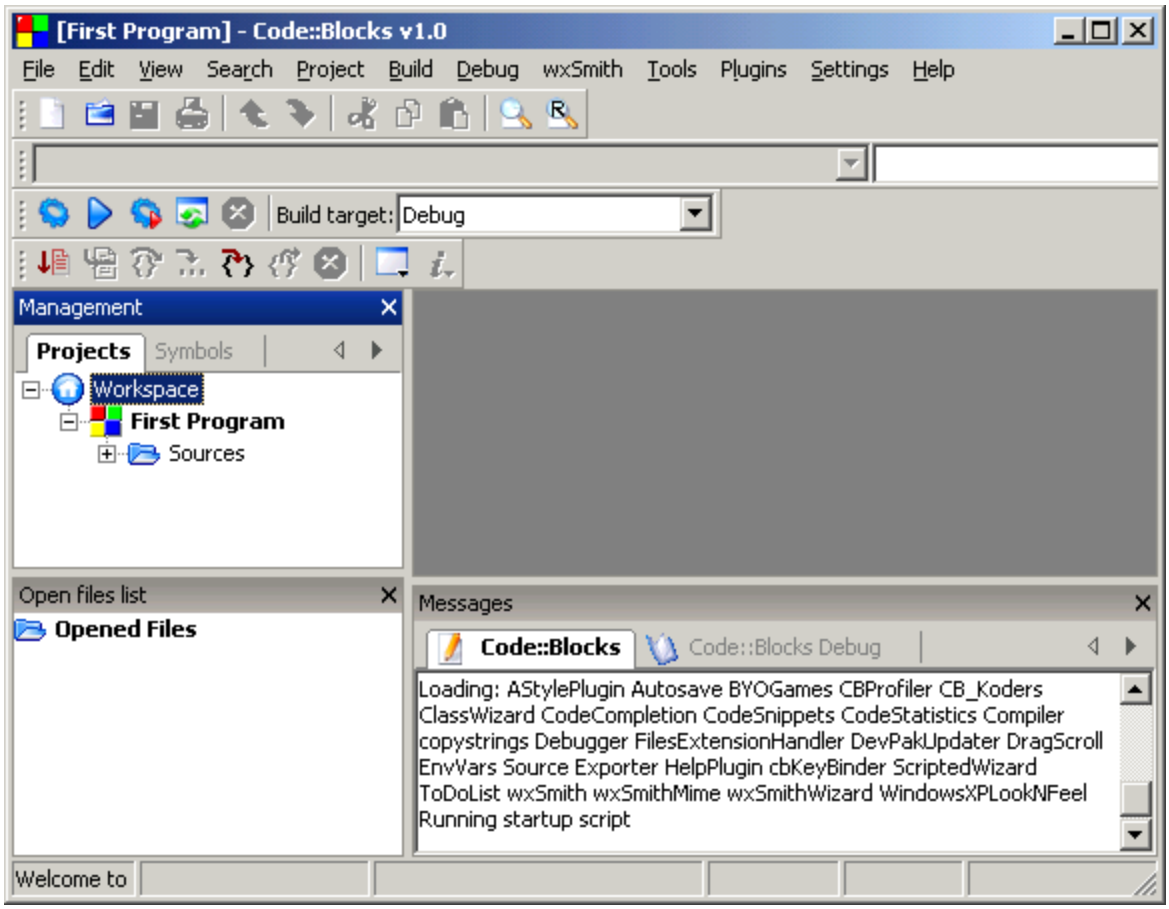


When the directory for your project has been selected, the system will return to the **Console application**. As shown, Code Blocks will create a directory called First Program (Project Title) and returns your selected directory in **Folder to create project in**. Inside that directory will be the P**roject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

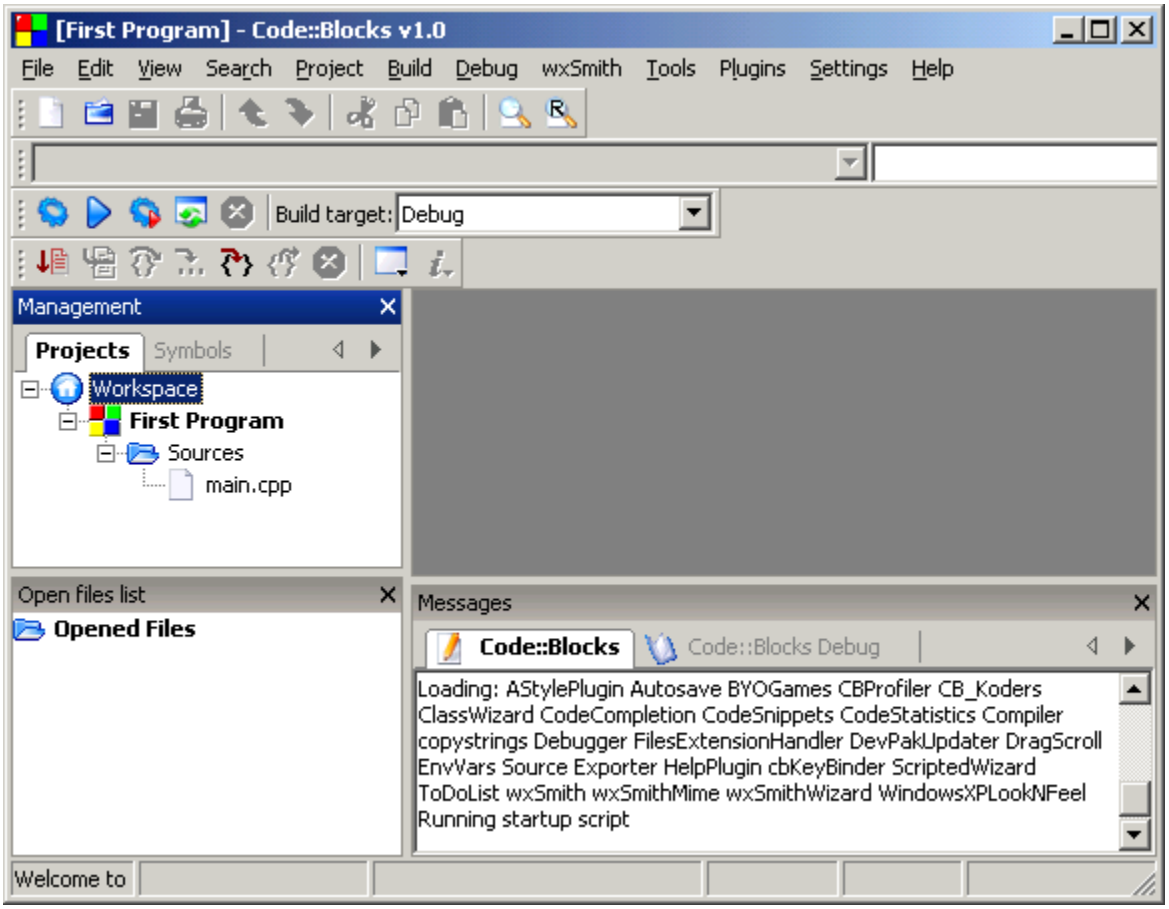
The next window to pop up will be the Compiler screen. This specifies where the **Debug** and **Release** compiled versions of your program will be placed. Leave this setting alone and press Next.



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



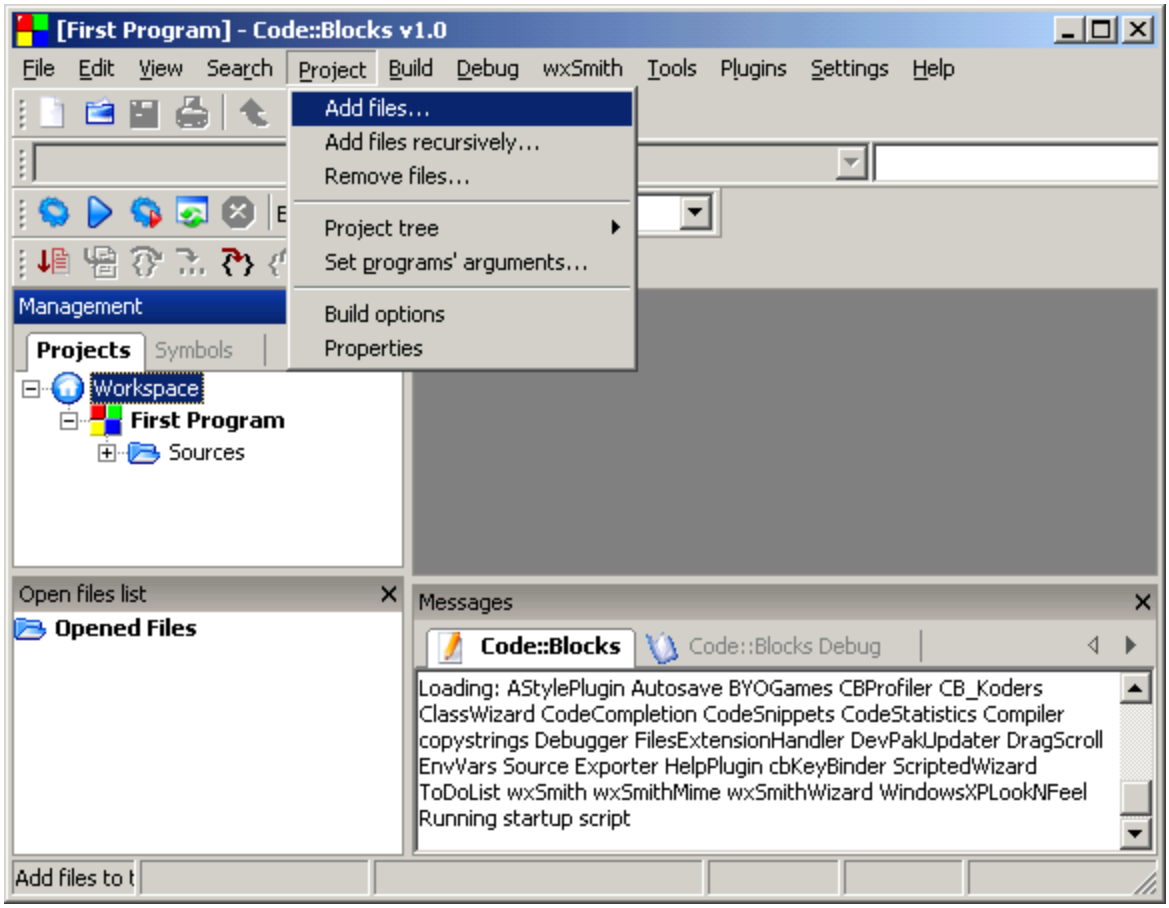
The system will then return to the **[First Program]** window and you are ready to write your program. It should be noted that the Build target is **Debug**, which will allow you to use the debugger to find errors. In the **Management** area of the screen (Shift-F2 toggles the Management display), you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

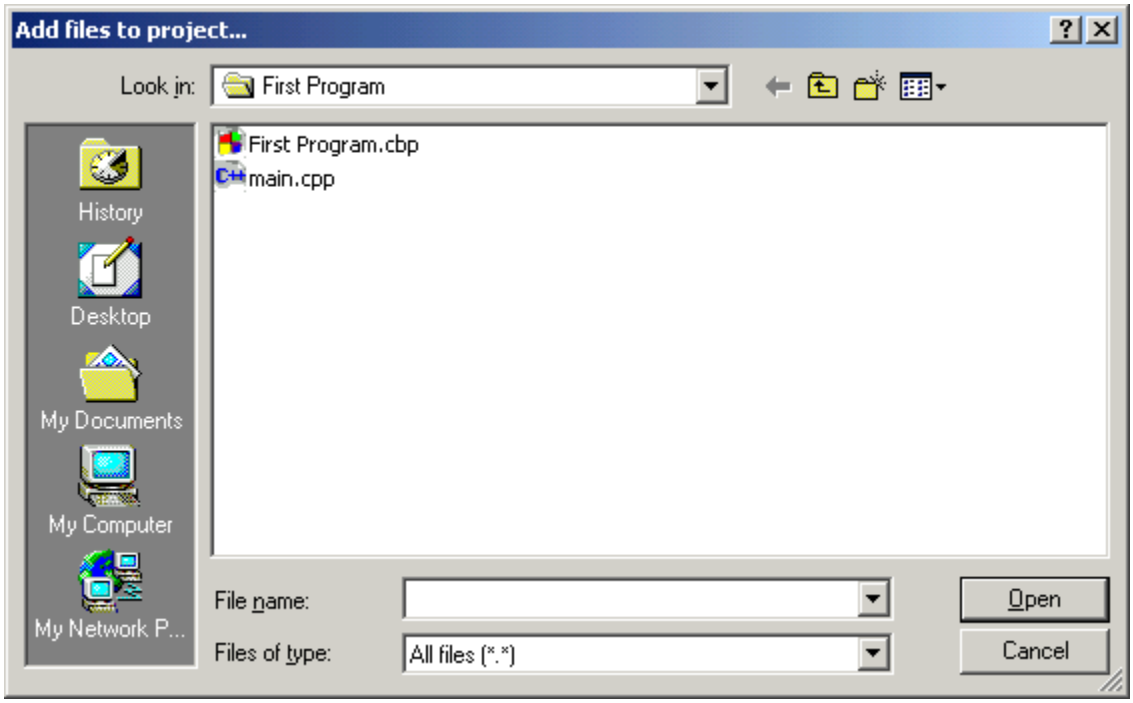


Under Sources, there is a file called main.cpp, which is automatically created for you when you build a console application.

Adding Files To Your Project

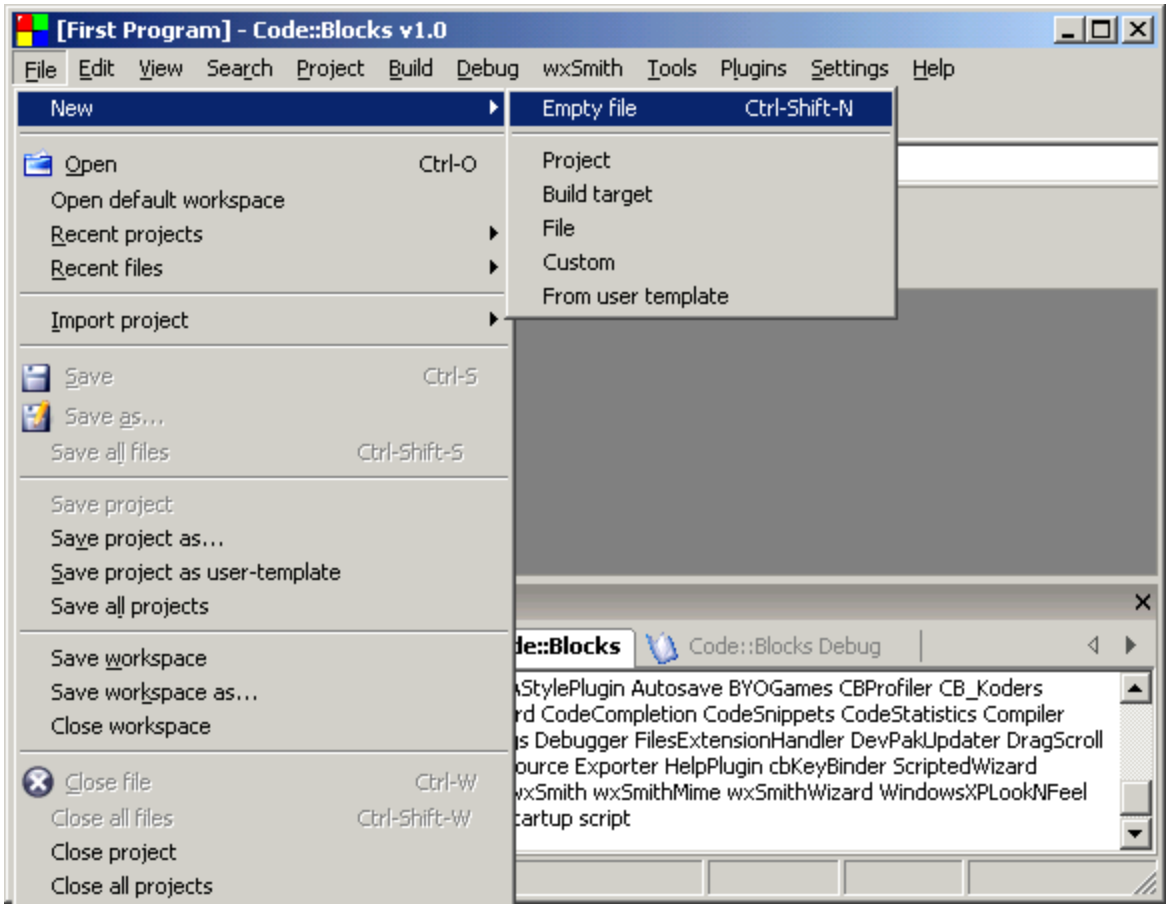
If you have a project with additional existing files, go to the Project menu and select “Add files.” This will bring in the files associated with your program. You also have the option to **Remove files**, performing **Build options** and to **Set programs’ arguments…**.





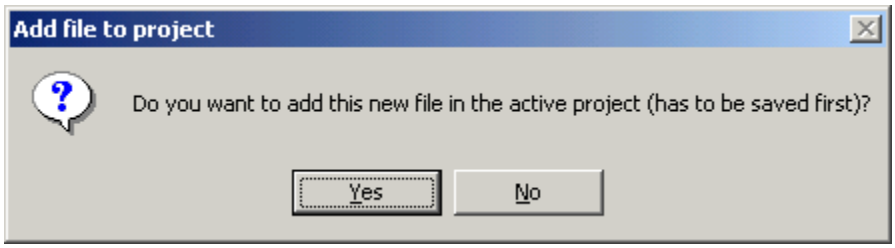
Clicking on **Add files to project,** will bring up a window so you can browse to where your files that you wish to add are**.** Select any additional file you want to add and press Open. The file will then be added to your project.

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



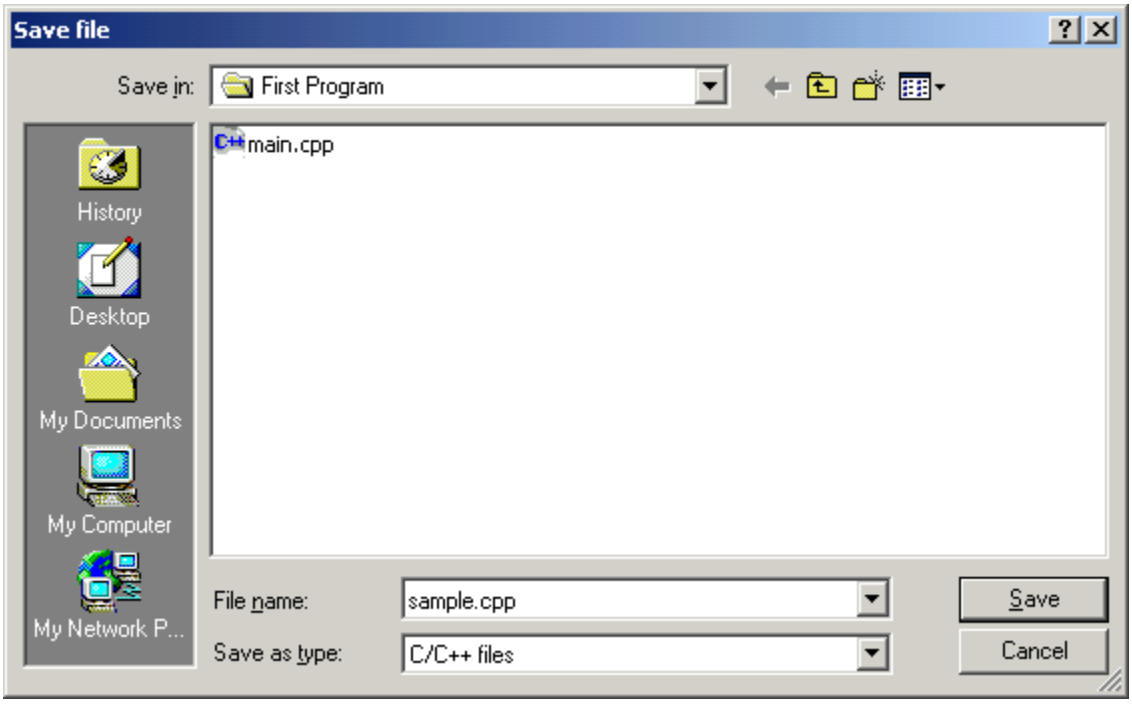
File, New, Empty file.

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

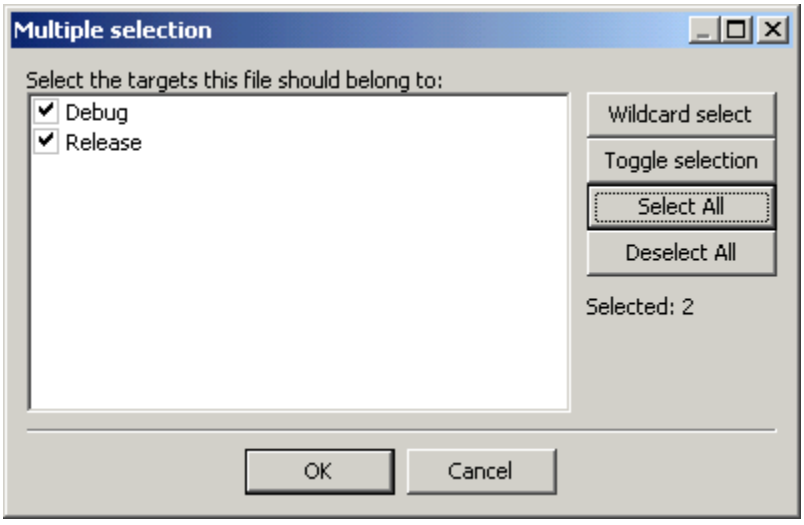


Choose *Yes*.

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

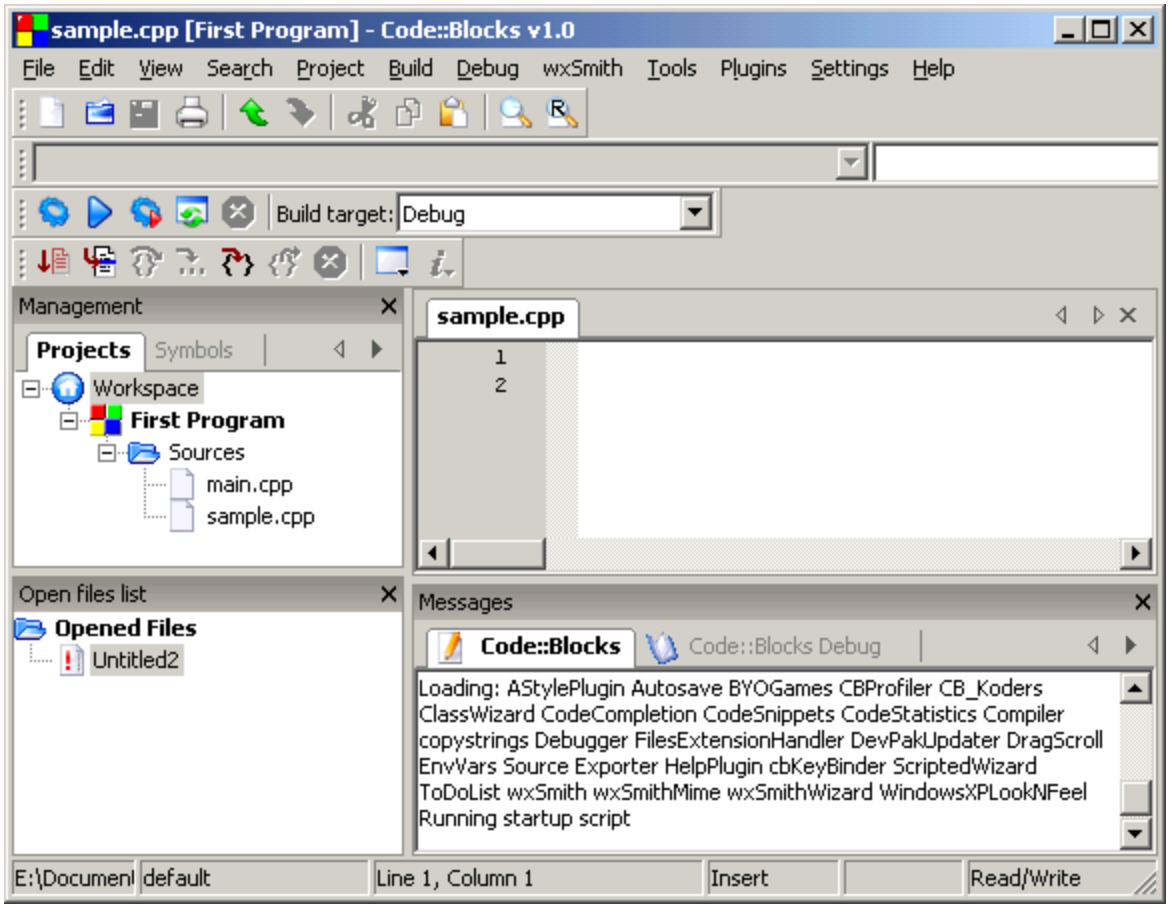


Give a name to the file. Pick a name that is related to the content of the file. Here it is called sample.cpp. C++ files need to be of the type cpp. Press Save to save the file.



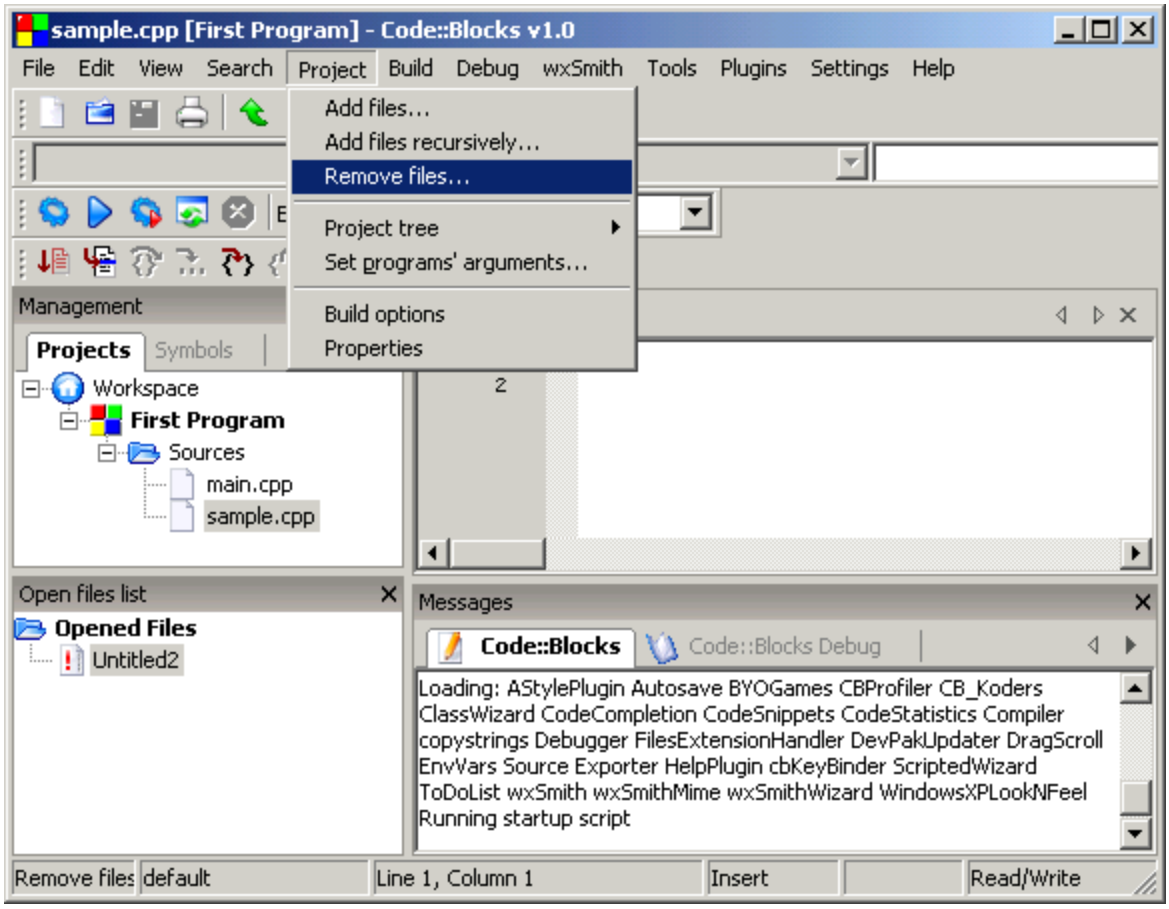
Press Select All to have this file saved as both Debug & Release **targets**. Press *OK* when done.

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.

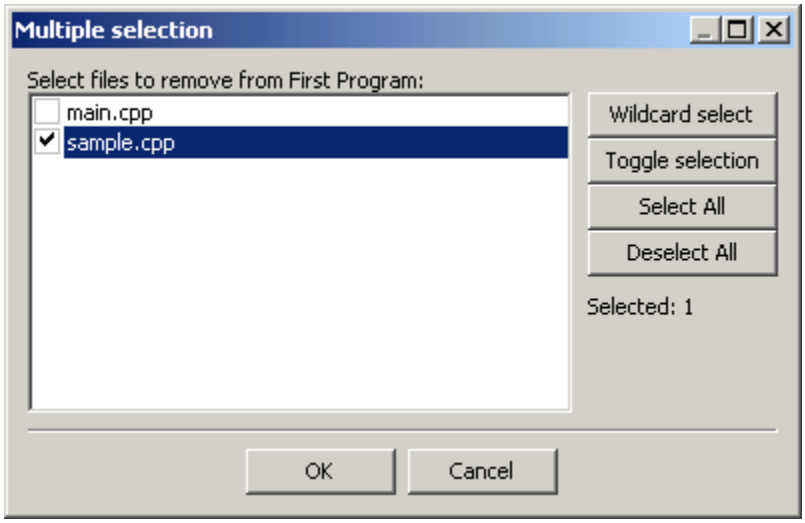


The Sources now has sample.cpp as a source file in addition to the main.cpp file.

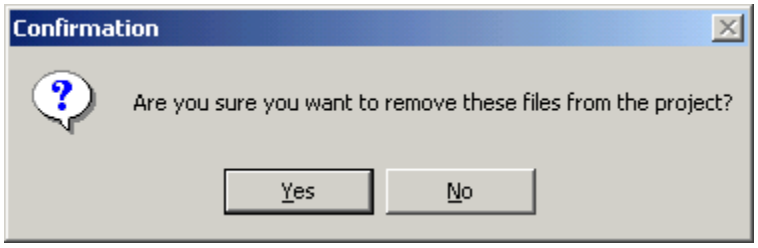
Since the sample.cpp is not needed for your project, please remove it.



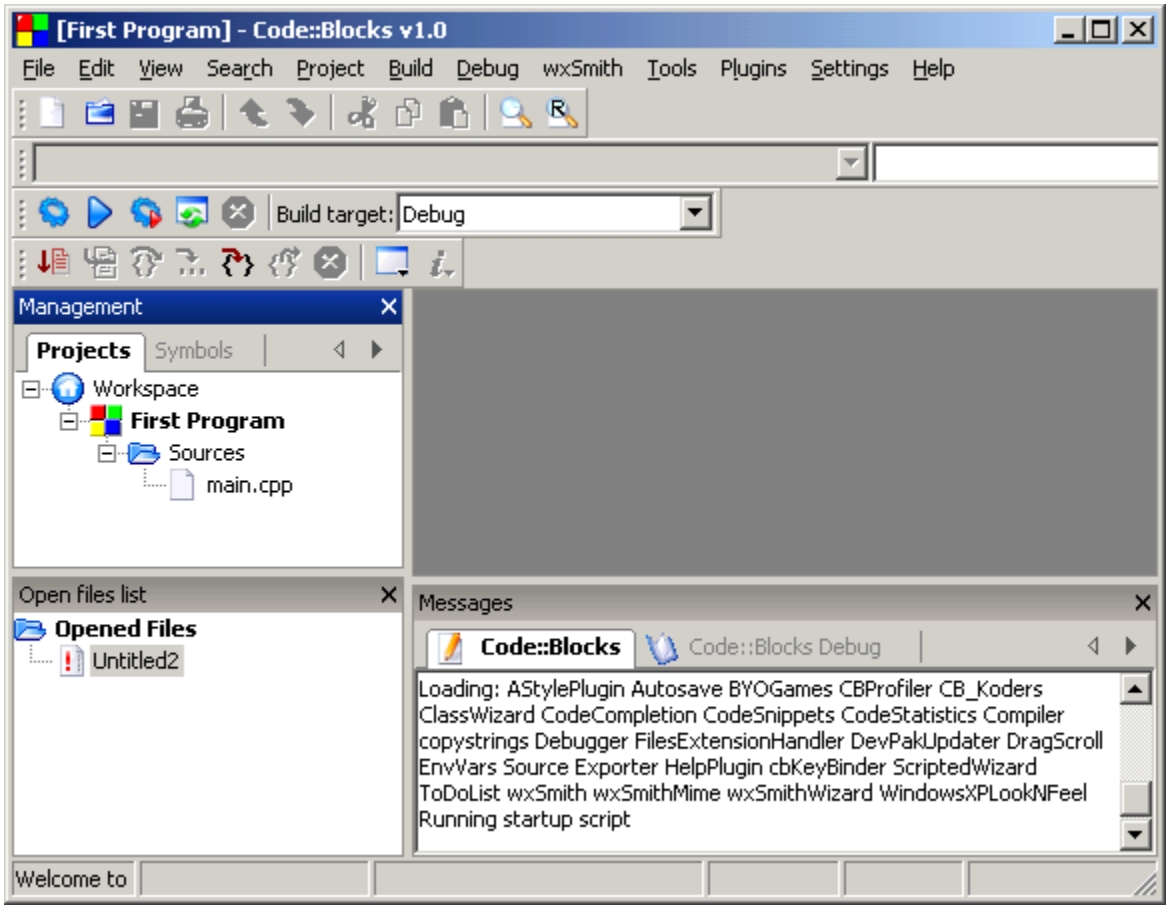
From the Project menu select, Remove files.



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

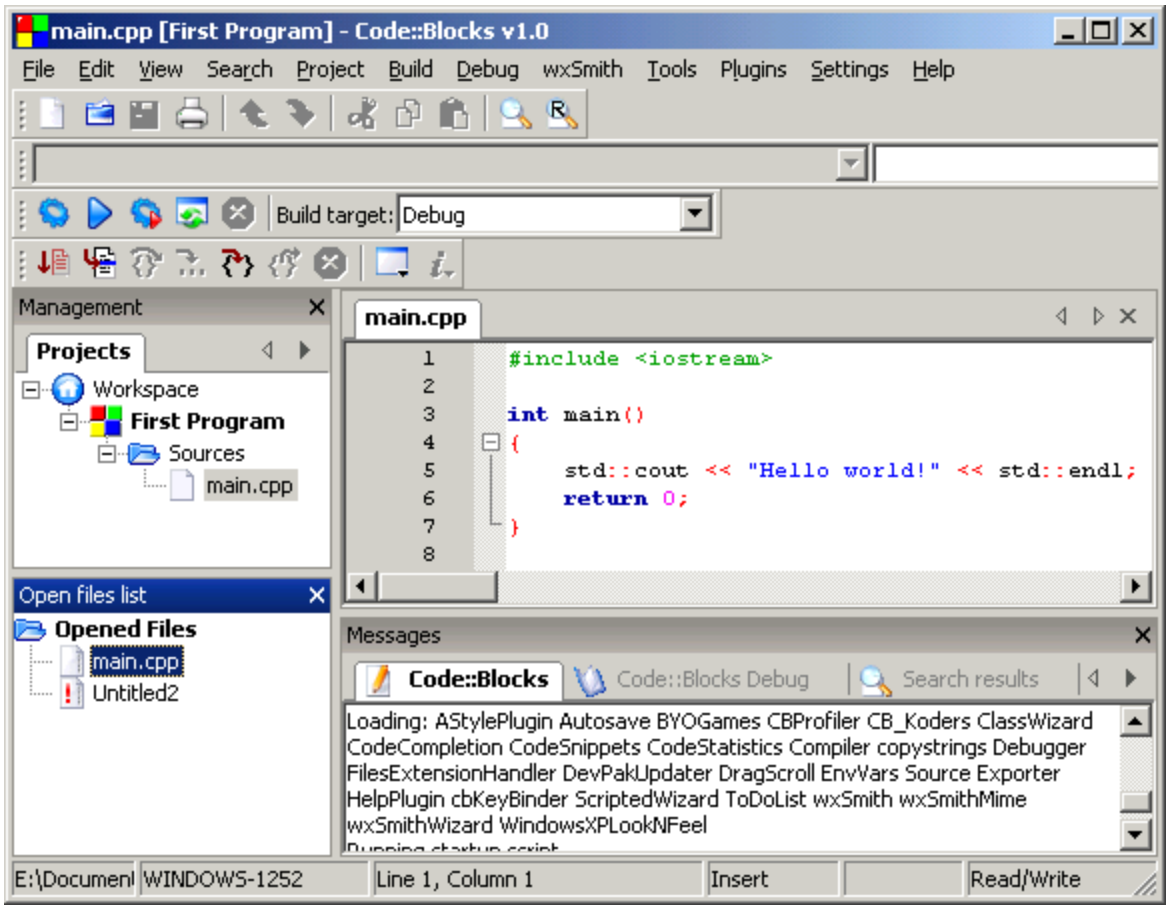


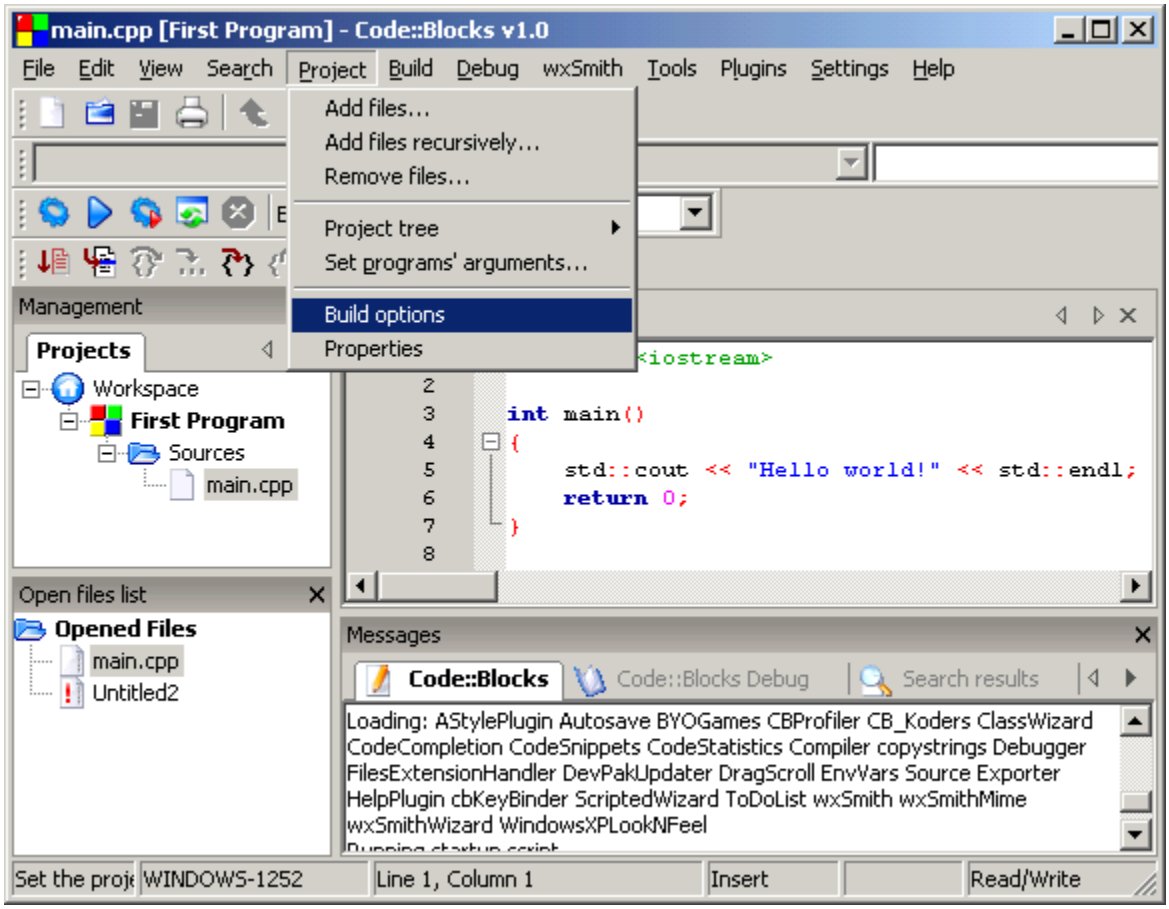
You will need to confirm that you wish to remove the file(s). Press *Yes*, if you are sure you want to remove them. Otherwise press *No*.



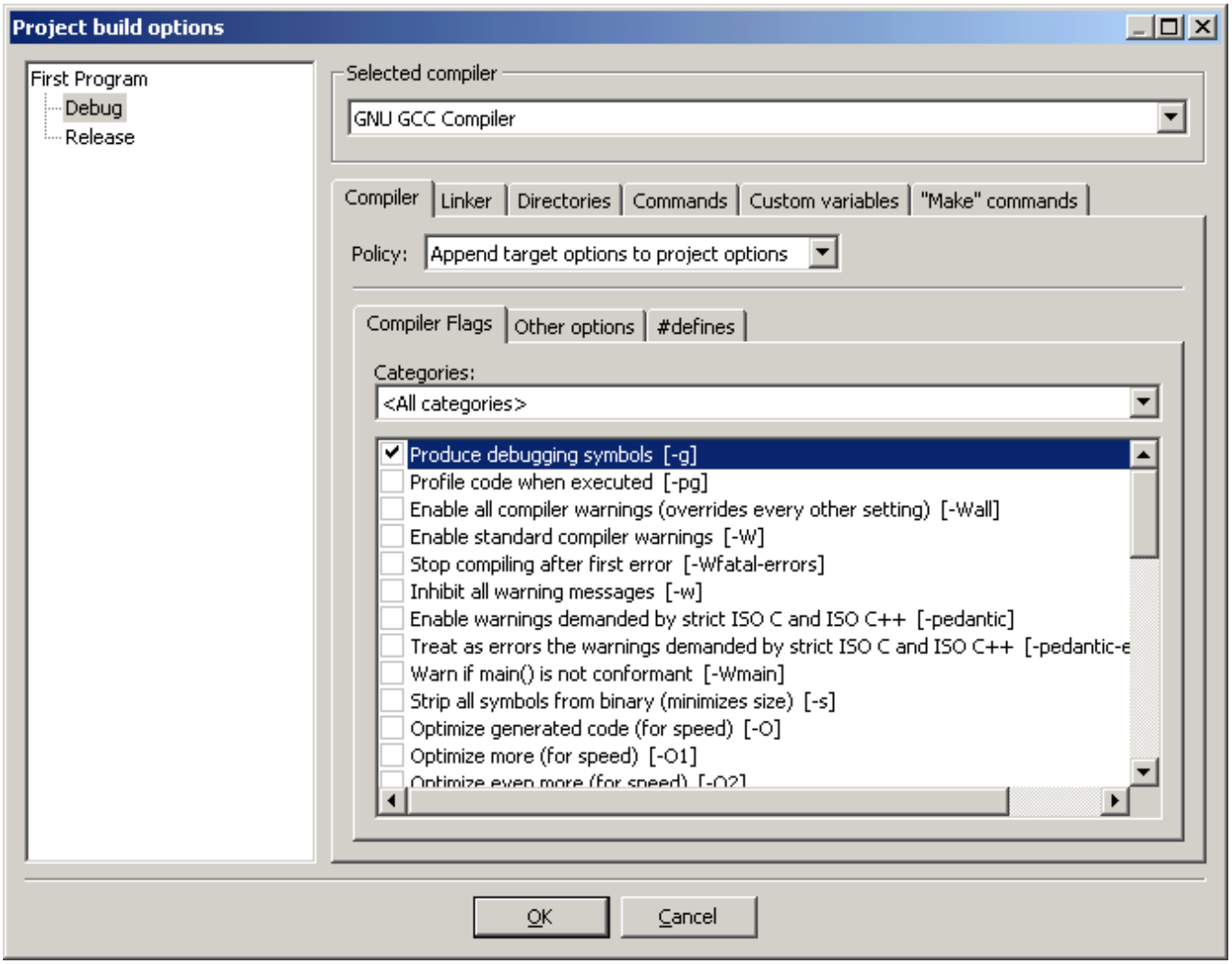
You will now see an updated listing of the Sources in your file. You should now see only Main.cpp. In the Open Files list, there may be a file called !**Untitled. Please ignore this.**

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.



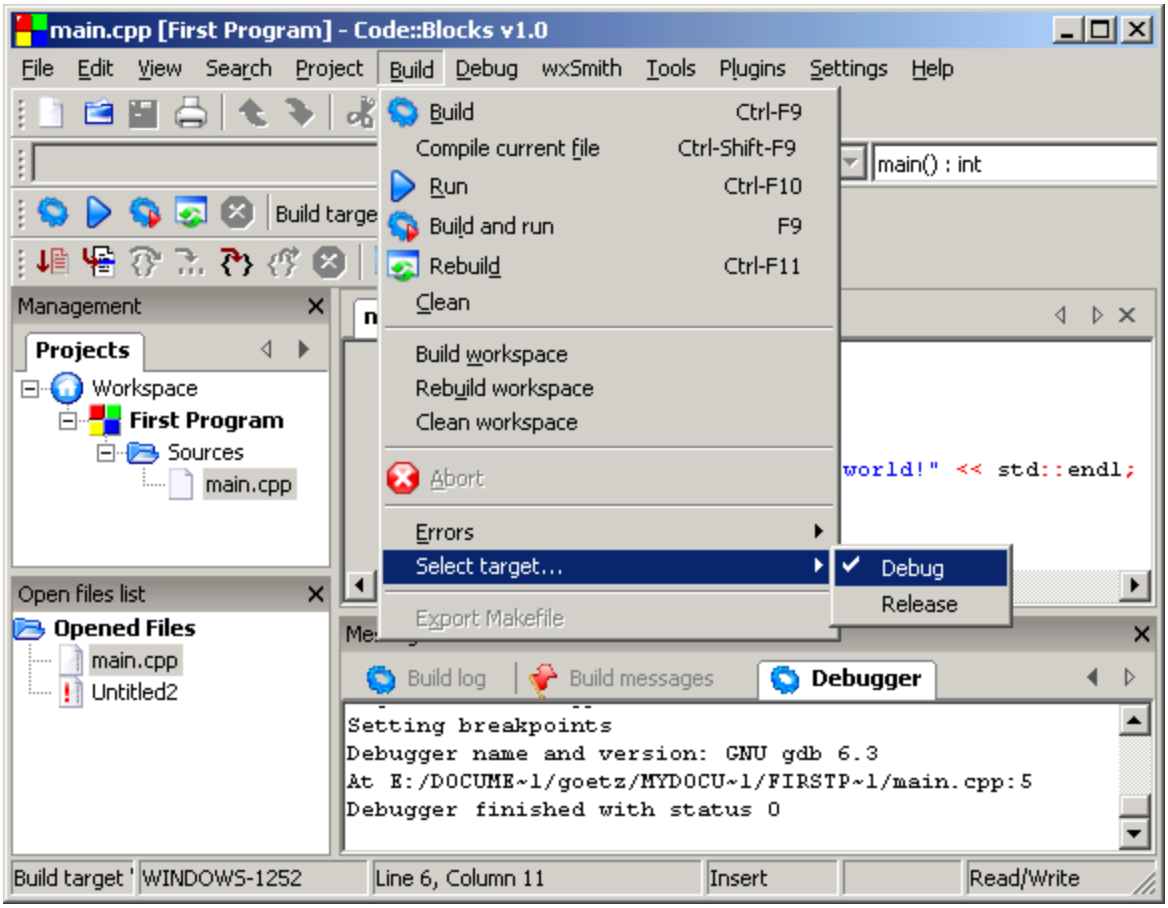


In order to check that Debug is running, you can use the **Project** pull-down menu and click on **Build Options.**

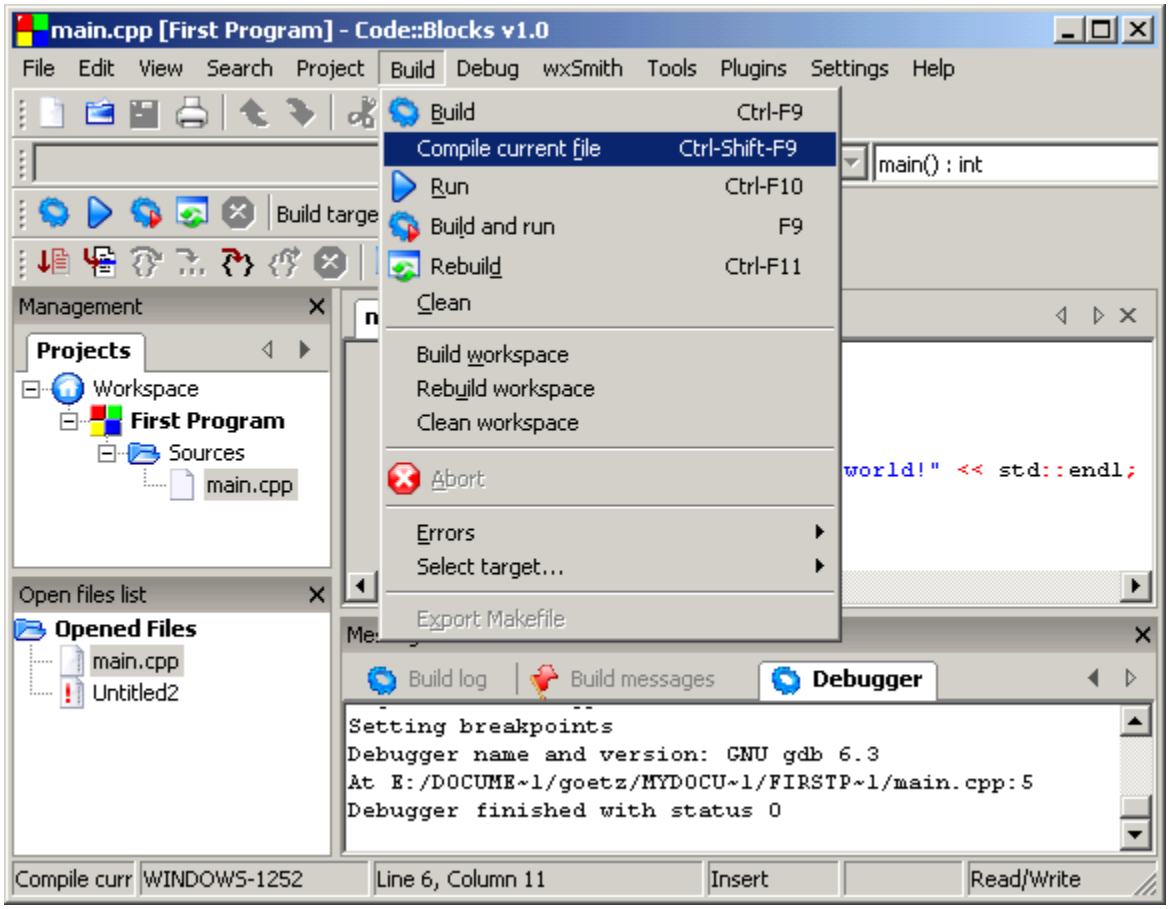


When this is done, the **Project Build options** window will come up. Make sure that the **Produce debugging symbols [-g]** is checked.

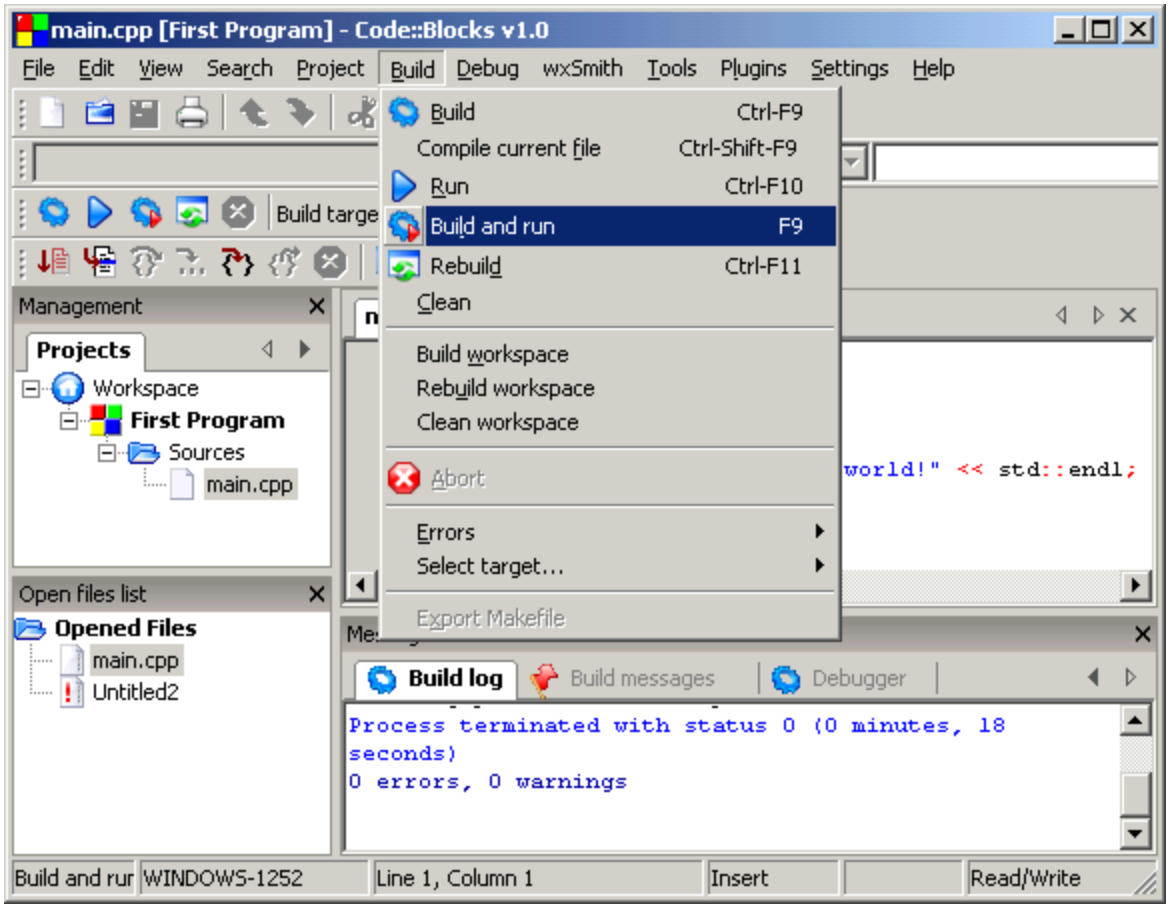
Press *OK* when done.



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

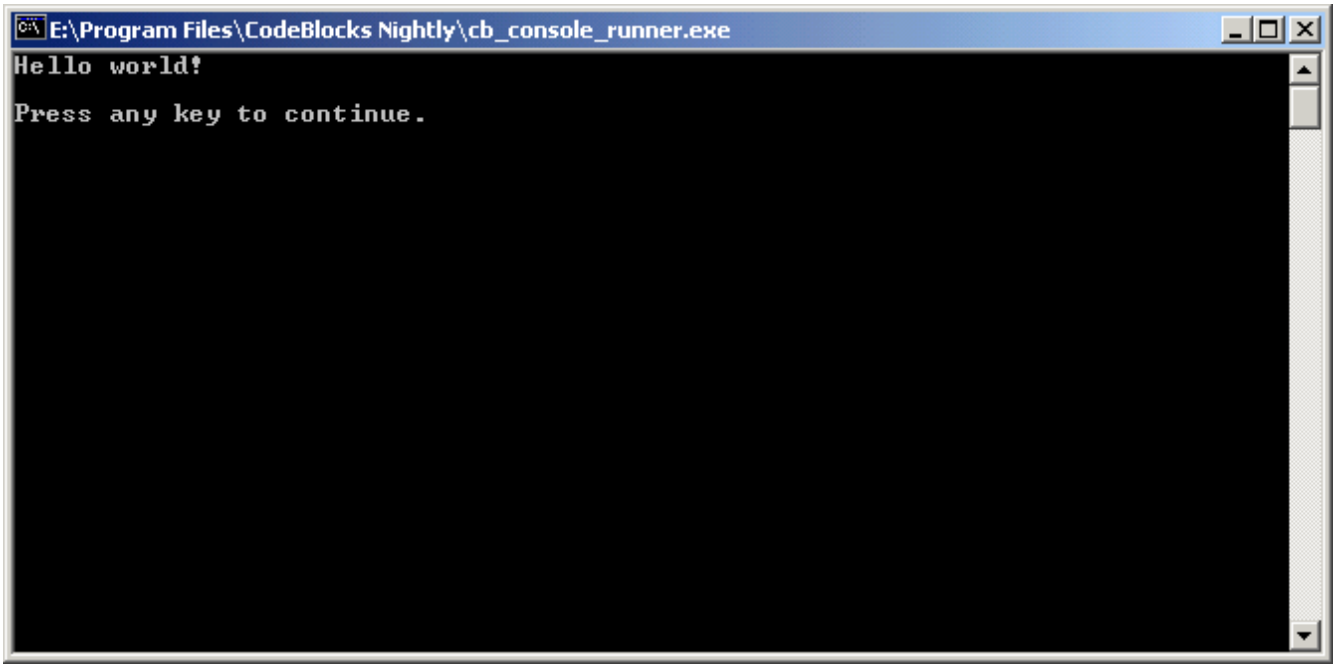


Compile your file from the Build pull-down menu by clicking on **Compile current file** (Ctrl-Shift-F9).



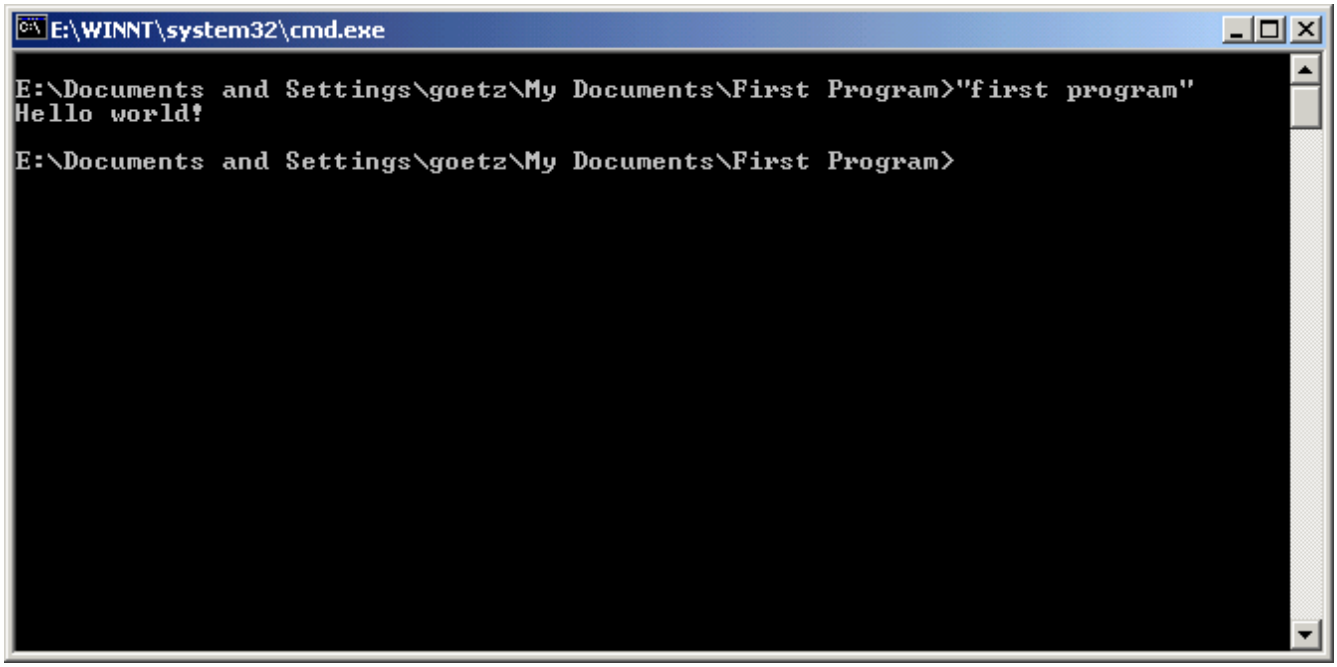
Test the project from the Build Pull-down menu, by clicking on Build and Run. This step will **build** an executable file for you. A project build will take the compiled versions of your source files and combine them into one program.

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



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

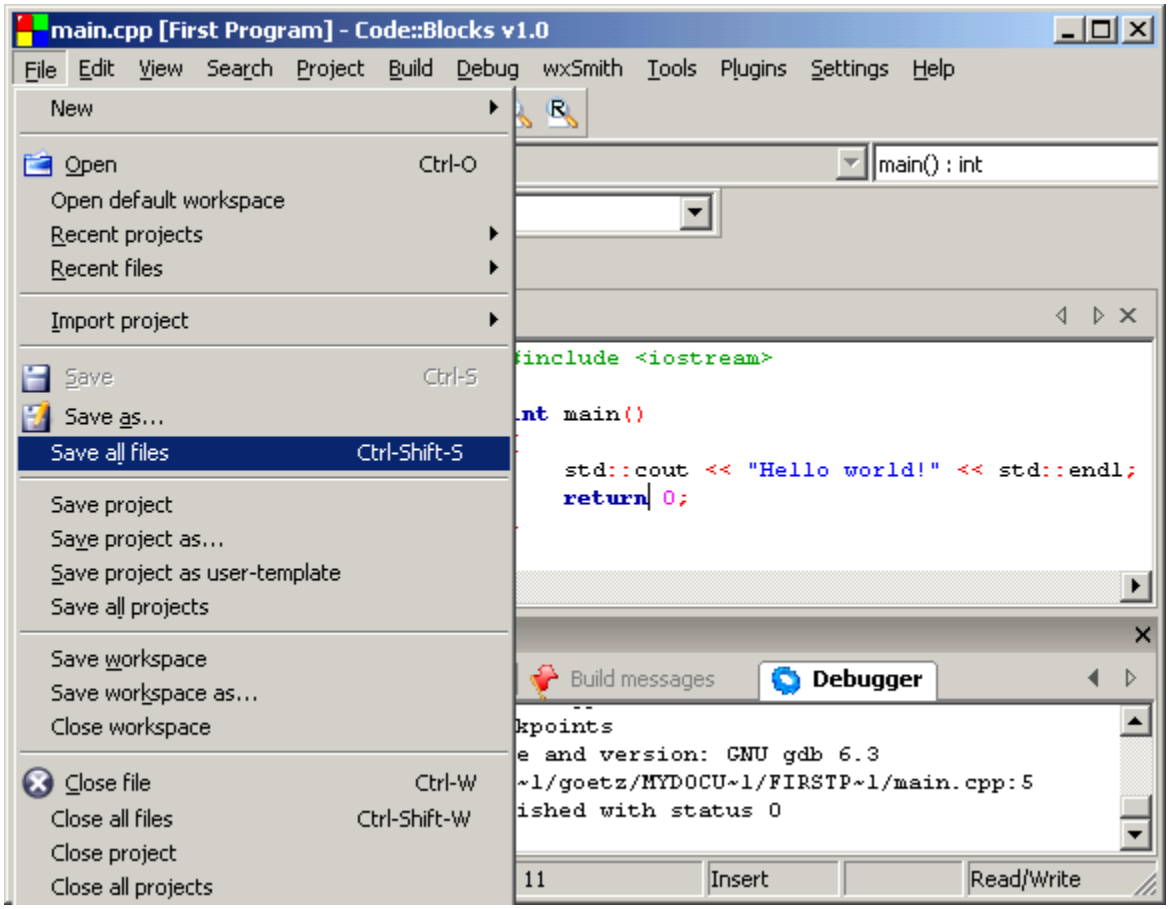
If you execute the program by going to a console window you will not see the “Press any key to continue” message:



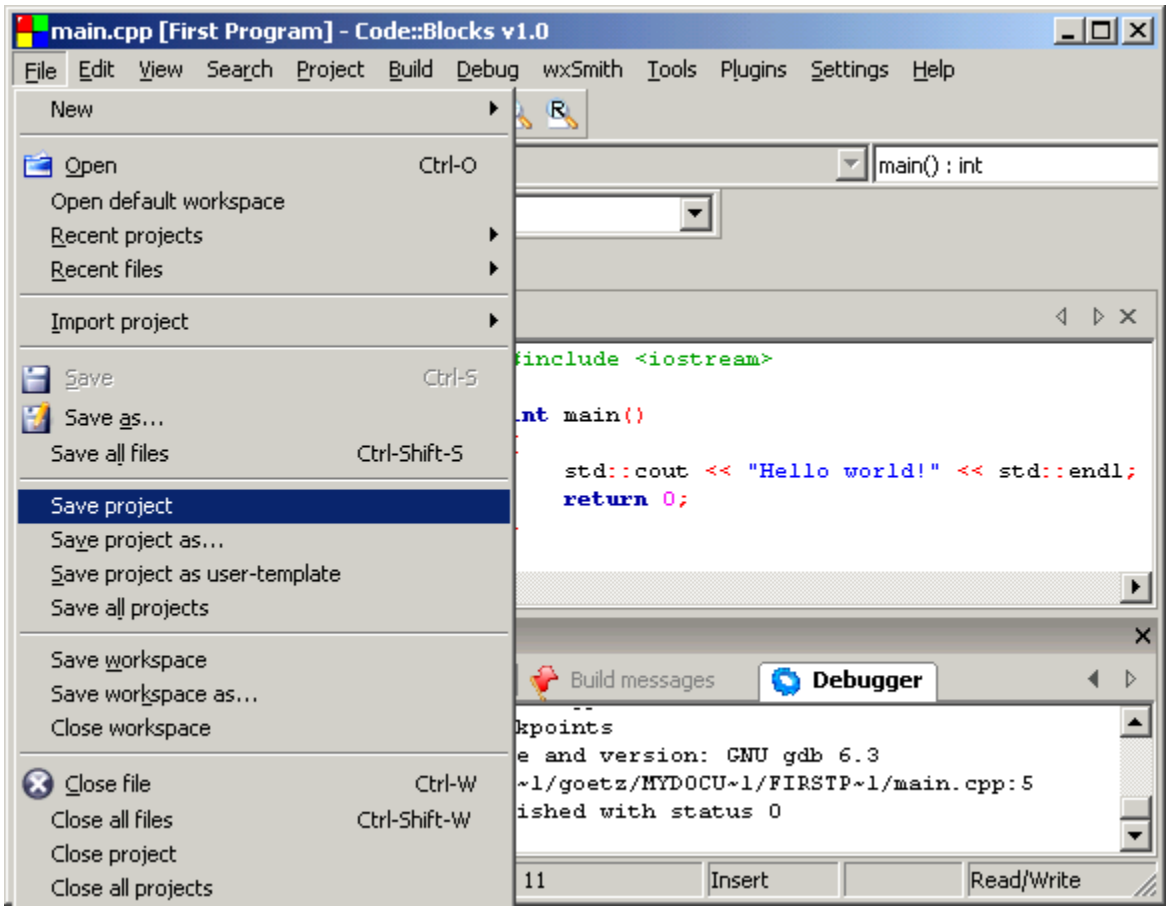
Notice that there are double quotes around the file name. This is because there is a space in the name. If

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

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

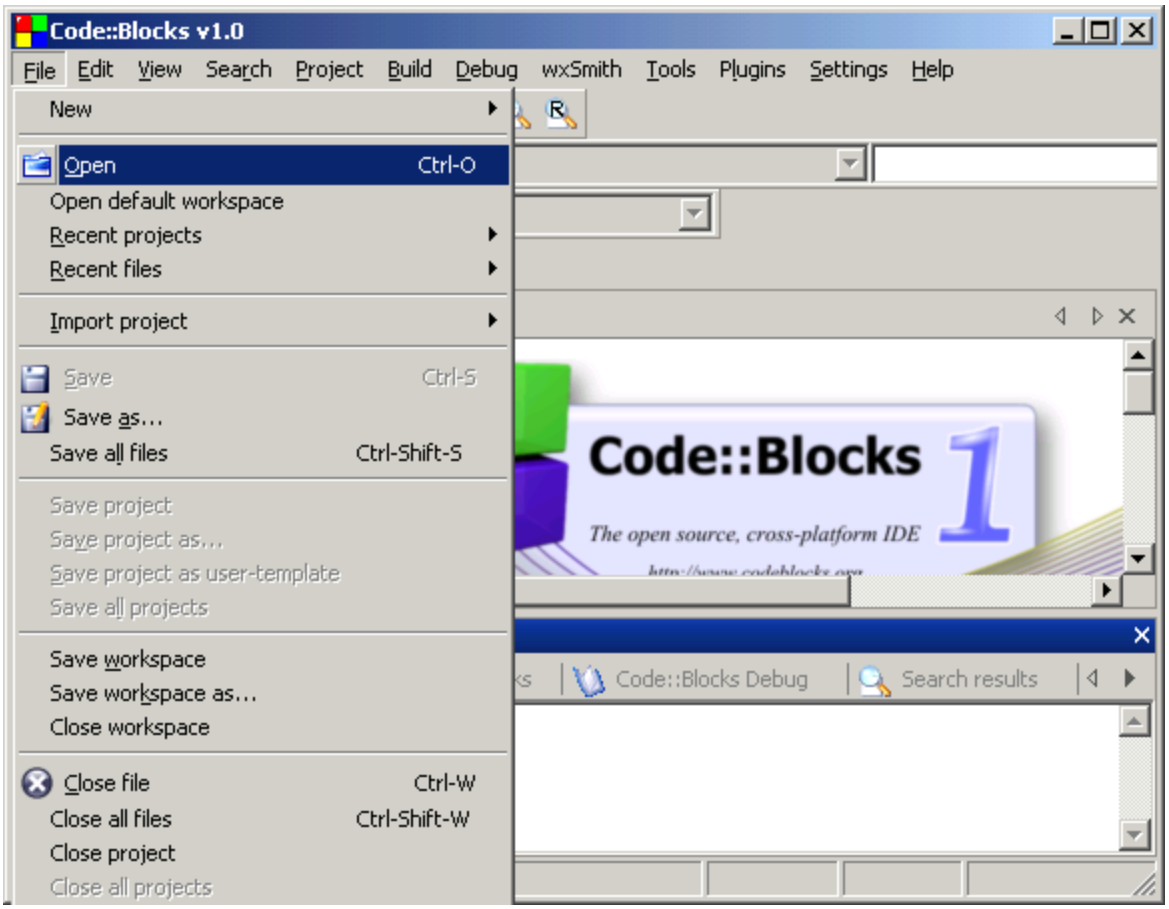


Now you can select to save the project:

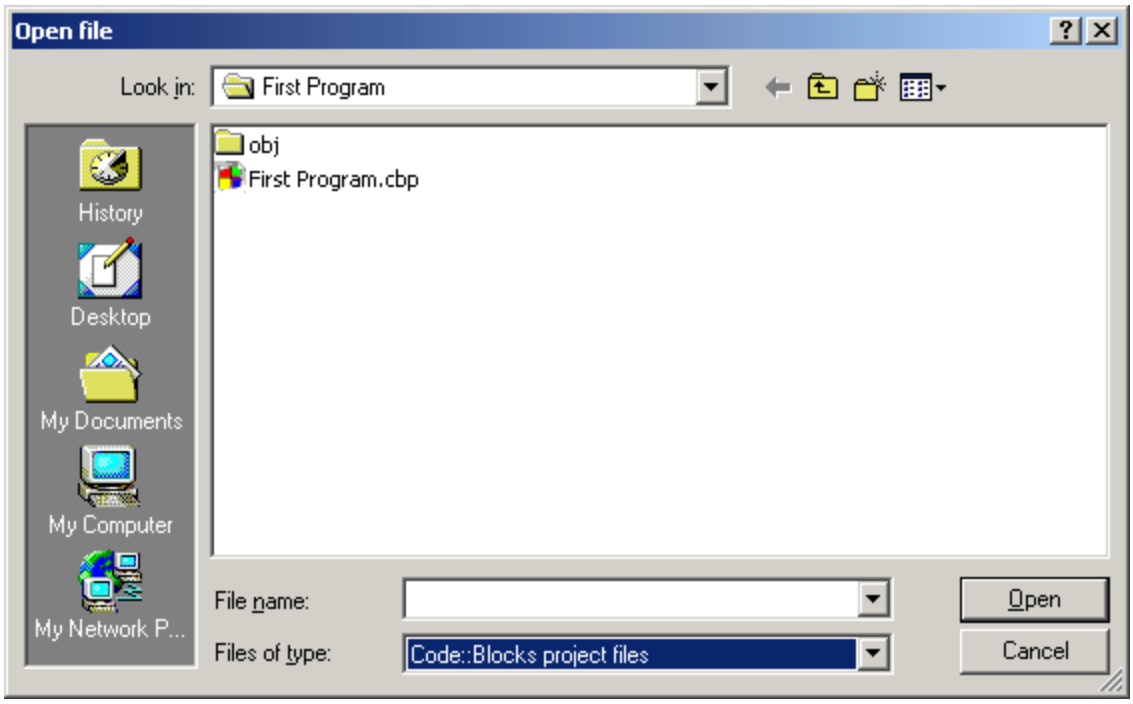


When you exit the program, you may be asked to save the **Workspace** and the **Layout**. The Layout refers to the placement of various windows that you may have positioned. Generally you would select to Save the Layout (unless you know you really do not want it saved). The Workspace refers to the projects 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

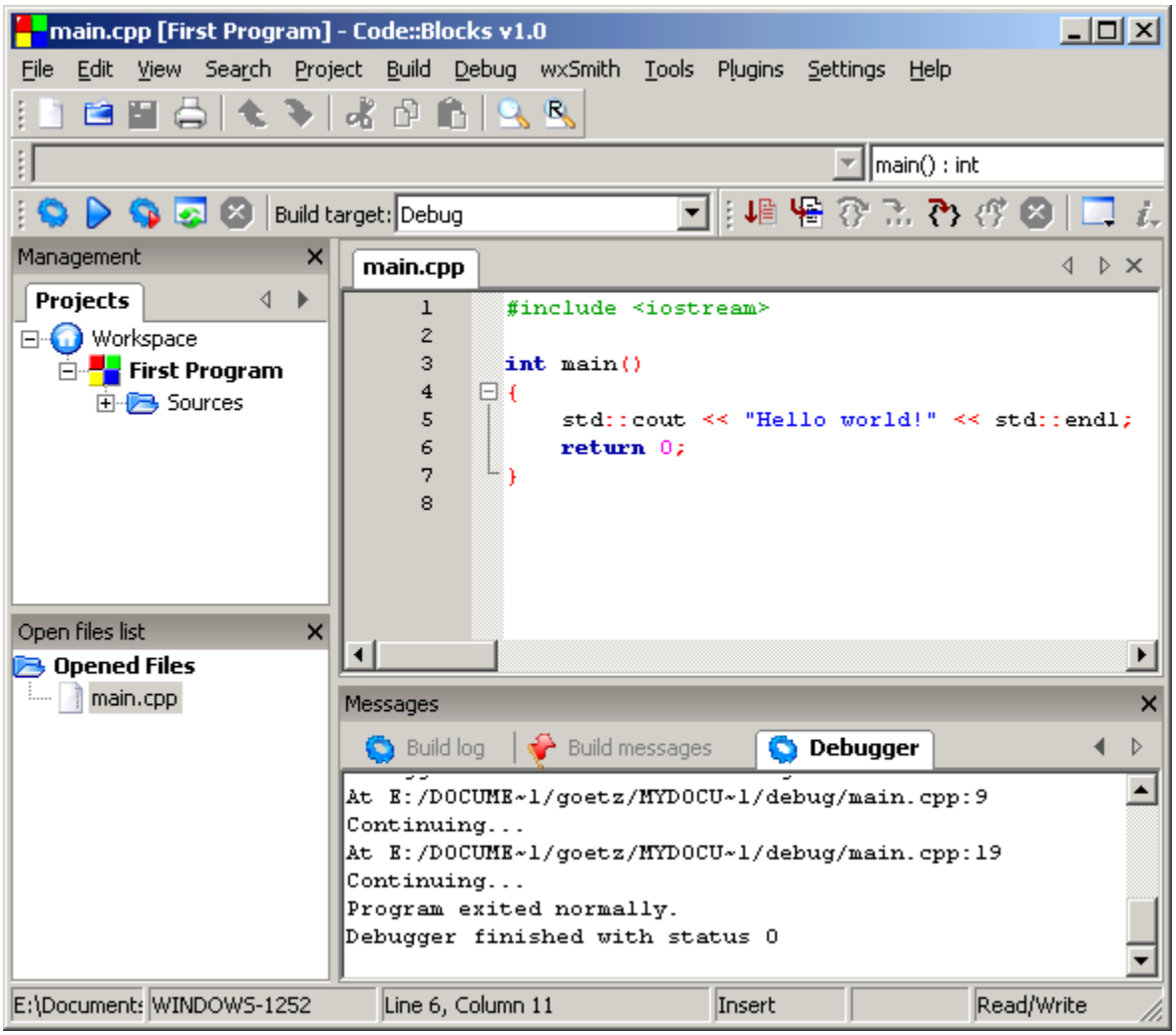


From the File menu select Open.



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.



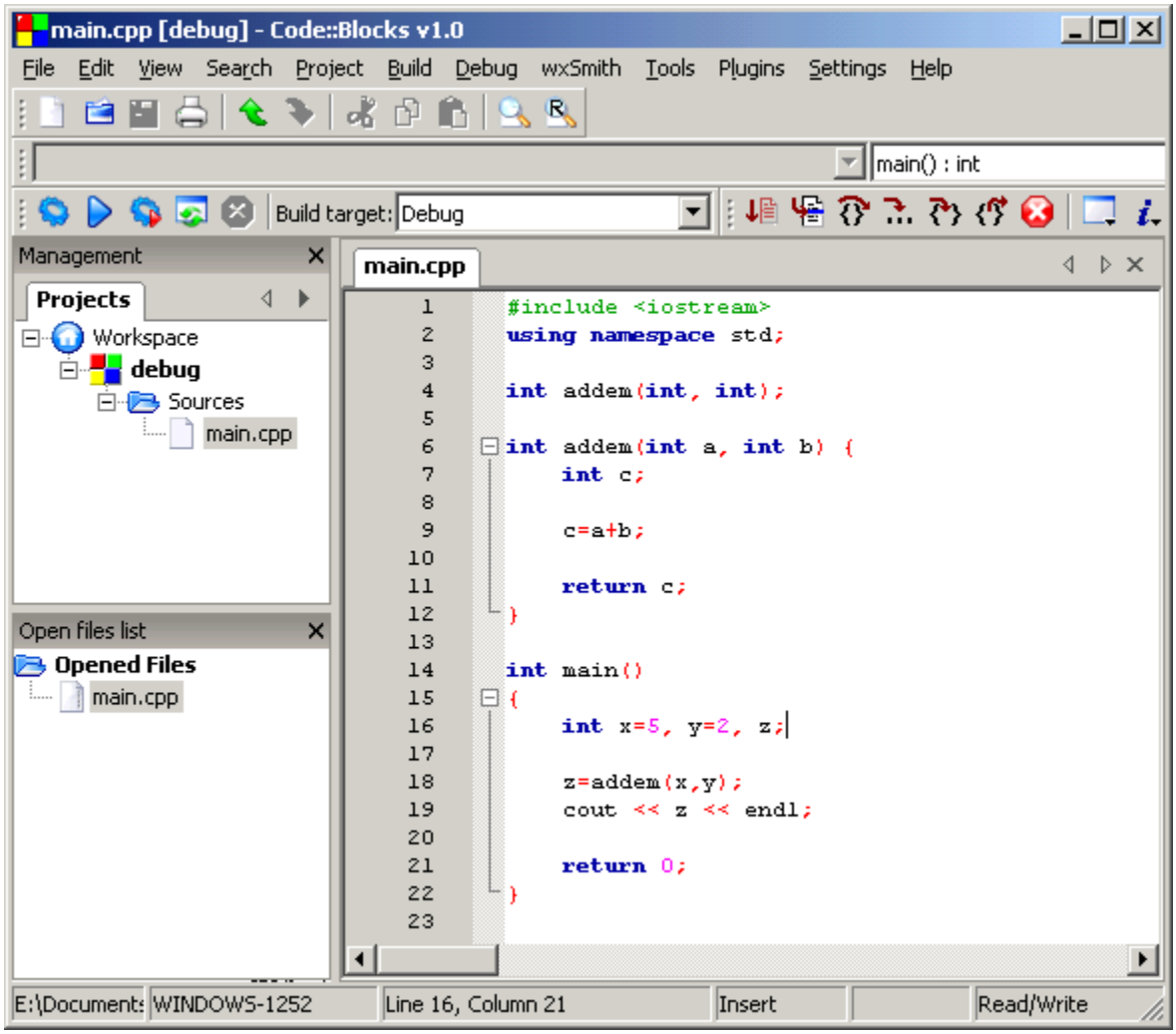
The project has reopened. You can get more space to see your program, if you close the **Messages** window. Pressing **F2** toggles the display of the messages. The **Messages** window has been turned off for the remainder of this tutorial, to allow more space to be visible on the screen.

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

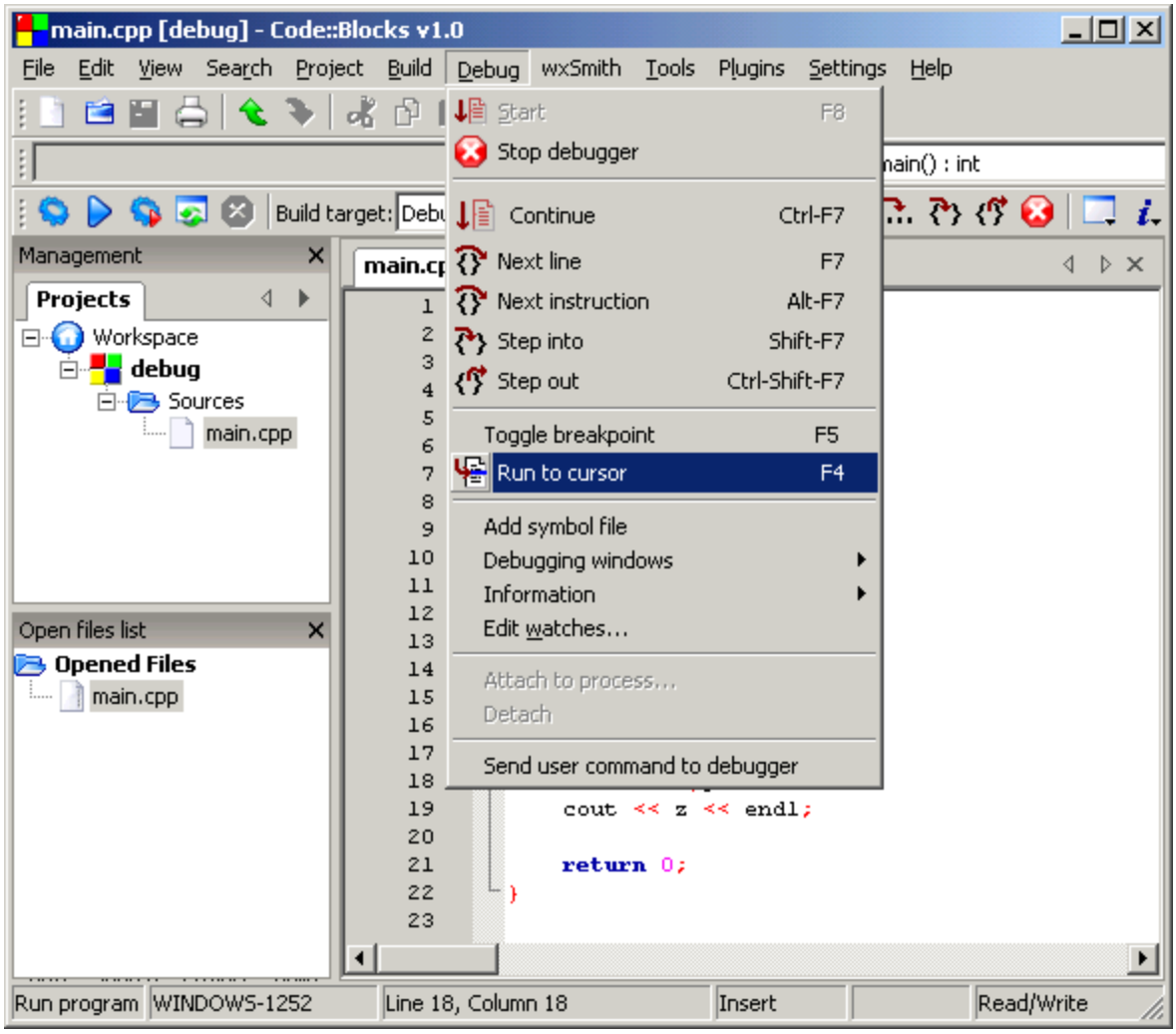
Debugging a Program

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.

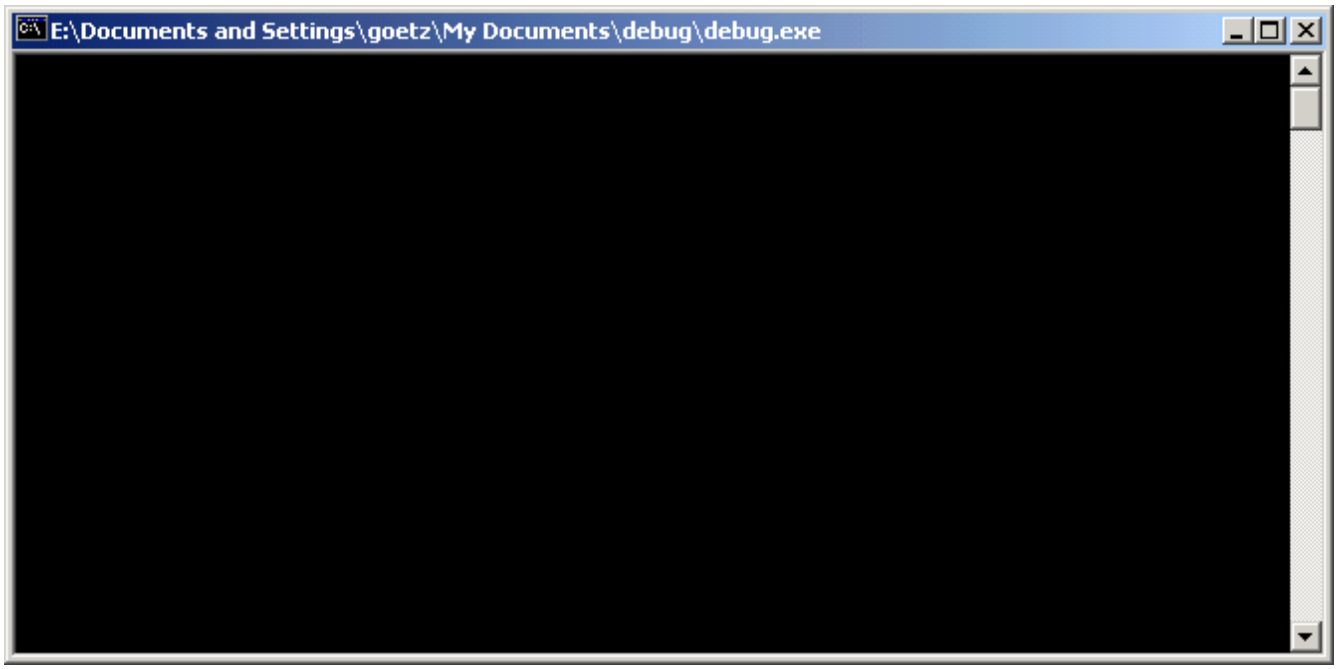
The following is a sample program that can be traced “line by line” while watching what happens as each line of code is executed.



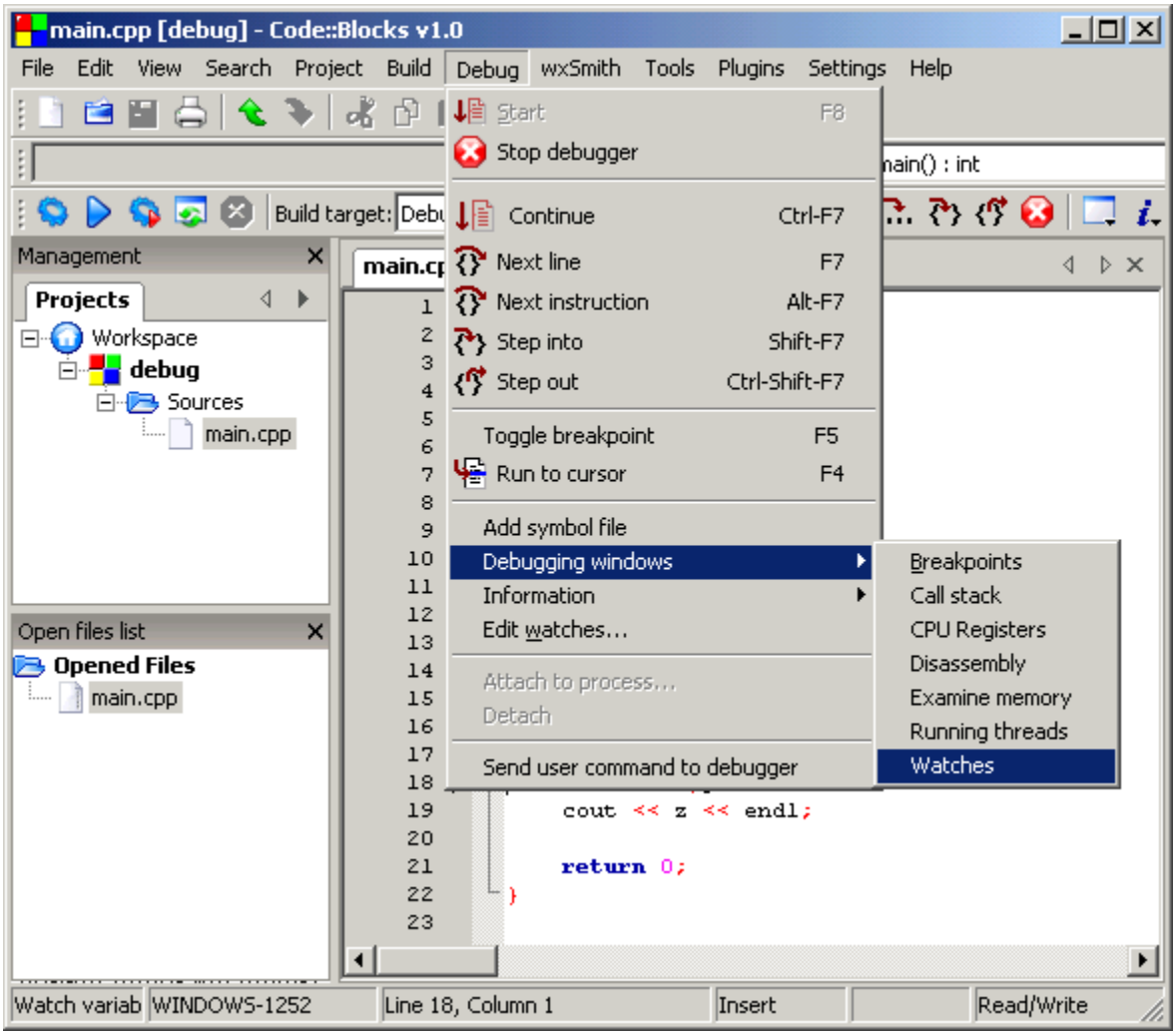
First, it is necessary to set a place in the code to have the program pause. This is done by using the **Debug** pull-down menu and clicking on **Run to Cursor. The cursor should be over the first line of code where you wish to start the tracing process. This starts the debugging process.**



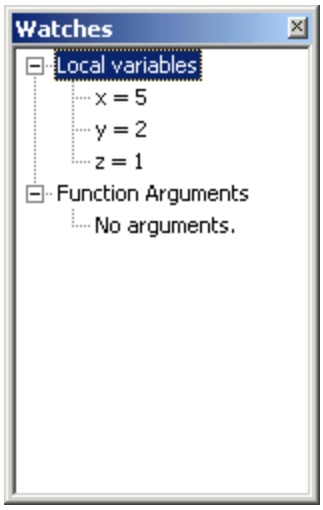
The next step in debugging a program is to tell the program when to stop running so you can inspect the results. To do this, place the cursor over the line where you want your program to stop. For example, the cursor was placed at line 18 (which is hidden behind the menu). This is called a **Breakpoint.**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



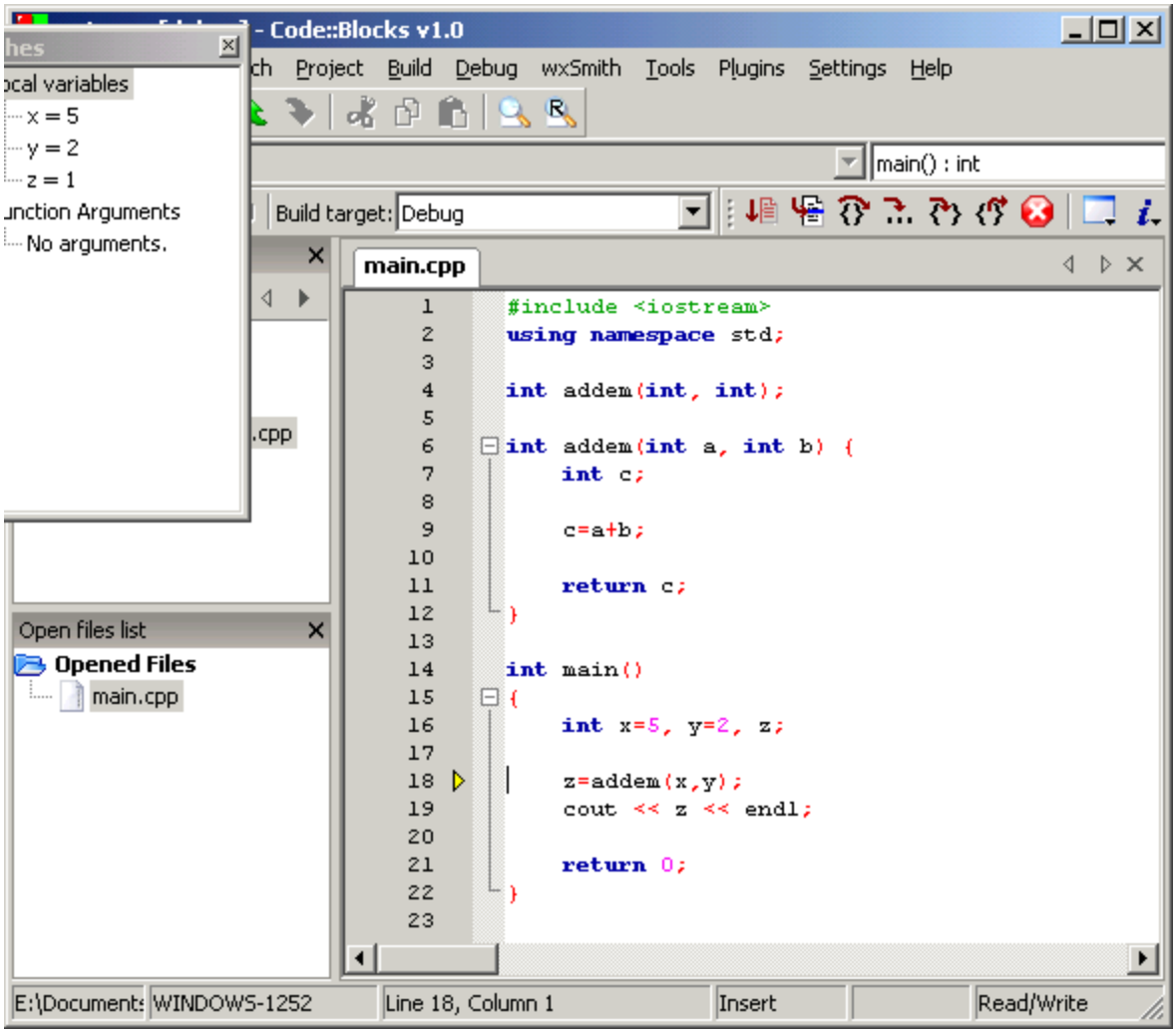
The program will generate a blank window. It is blank, since that program has yet to execute any line that displays something.



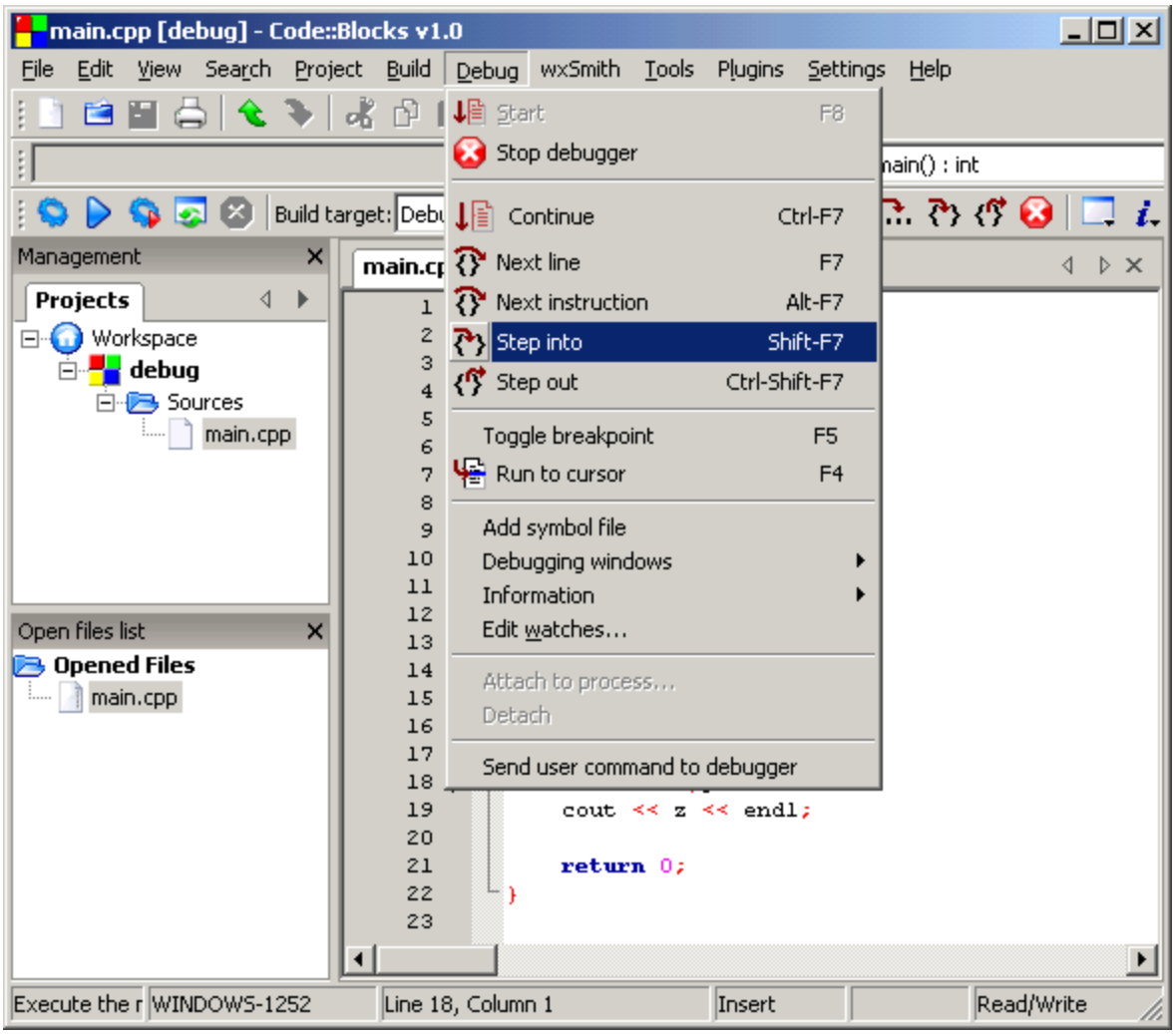
To watch certain variables during the execution of the program, you should open the Watches window. This will show you the variables in your code. This is accomplished by going to the **Debug** pull-down menu and clicking on **Debugging Windows** and then **Watches**.



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



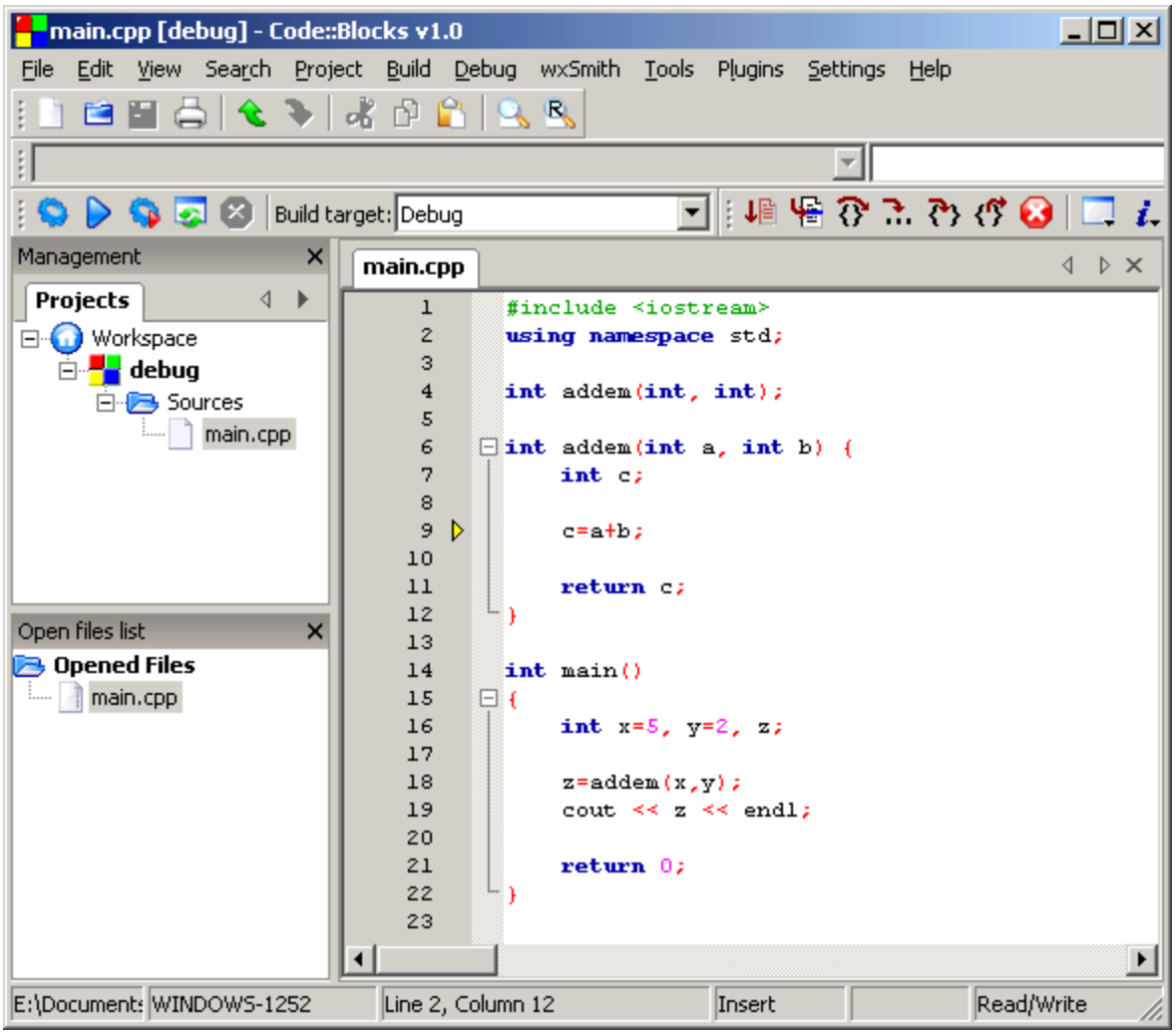
Line 18 has a yellow marker on the left side. This indicates that the program has paused on that line, which is the breakpoint.



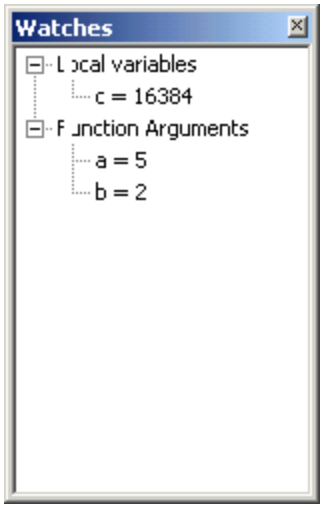
To determine how your program will function when calling functions such as:

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

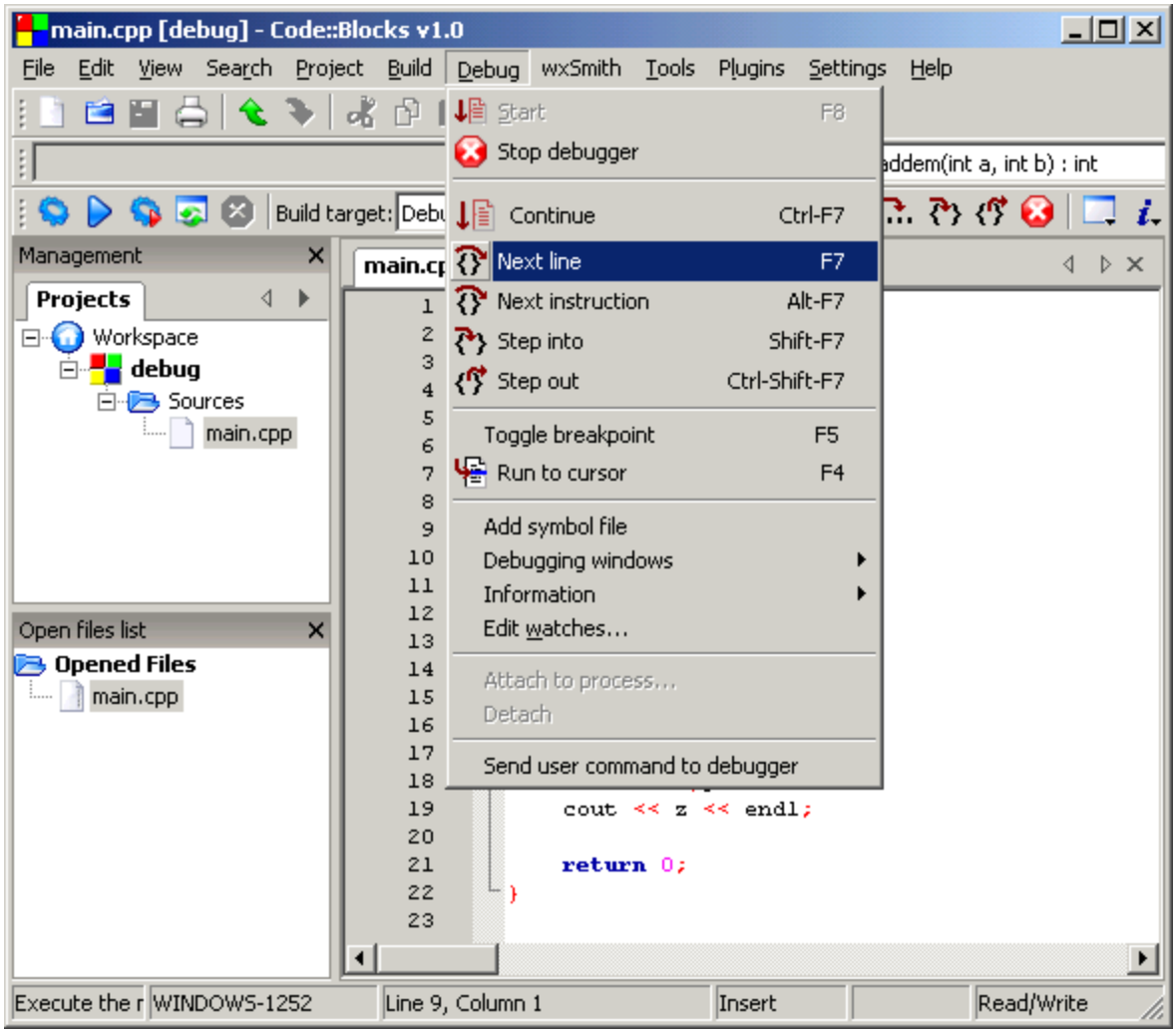
Step info (Shift-F7) can be selected from the Debug pull-down menu.



The next step is line 9.

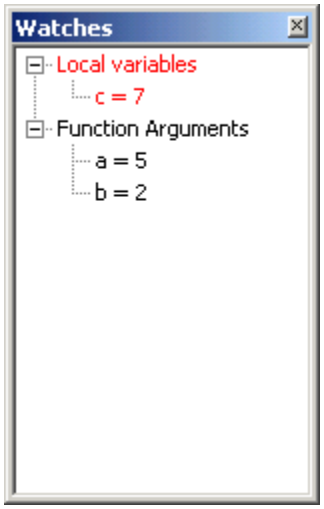


The arguments *a* and *b* are shown in the Watches window. Notice that the local variable c, which has not been set yet, has a random value.

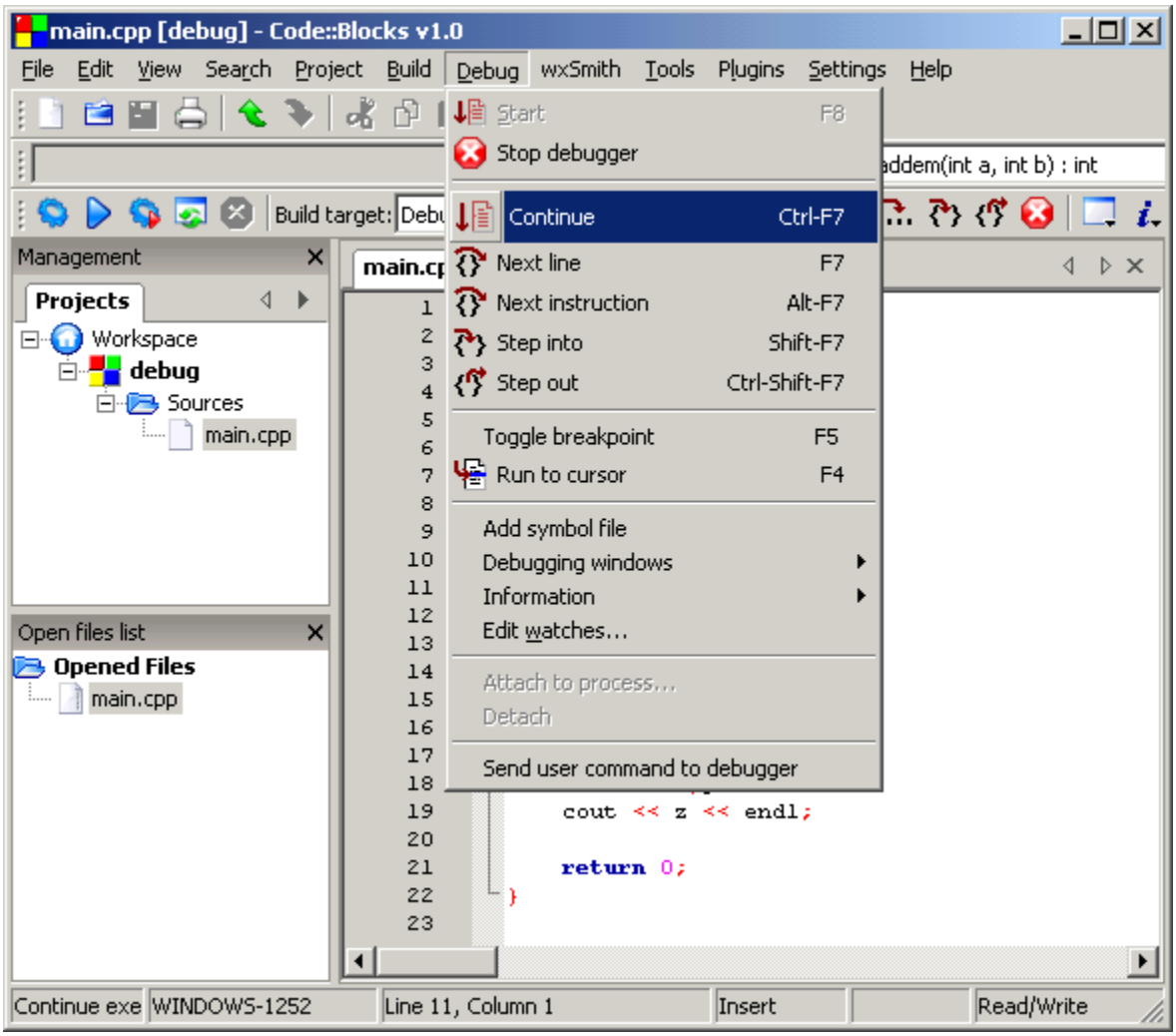


To proceed to the next line of code, select **Next line** from the Debug menu.

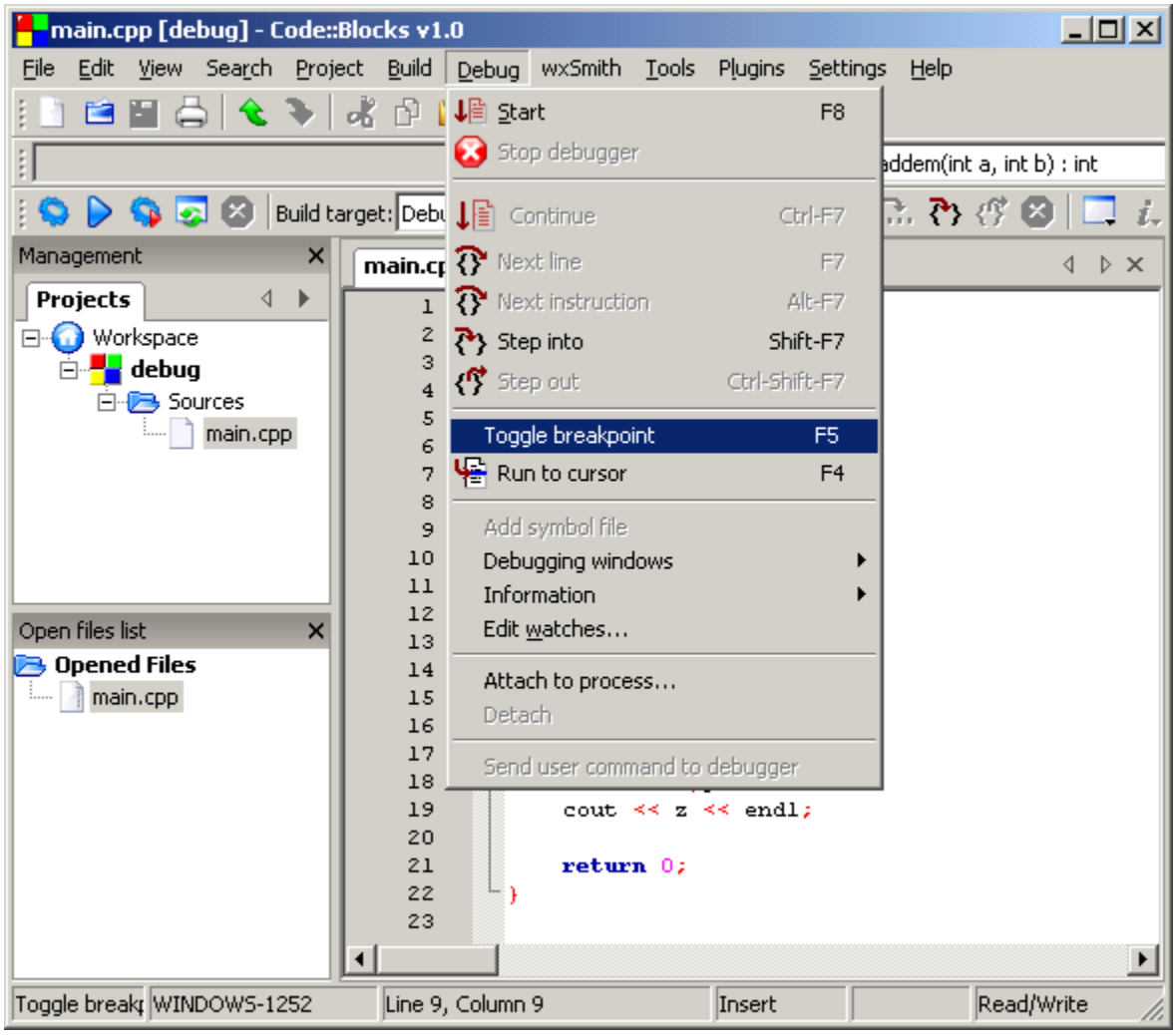
Pressing **F7** is a useful keyboard shortcut and will become second nature as you become familiar with the system.



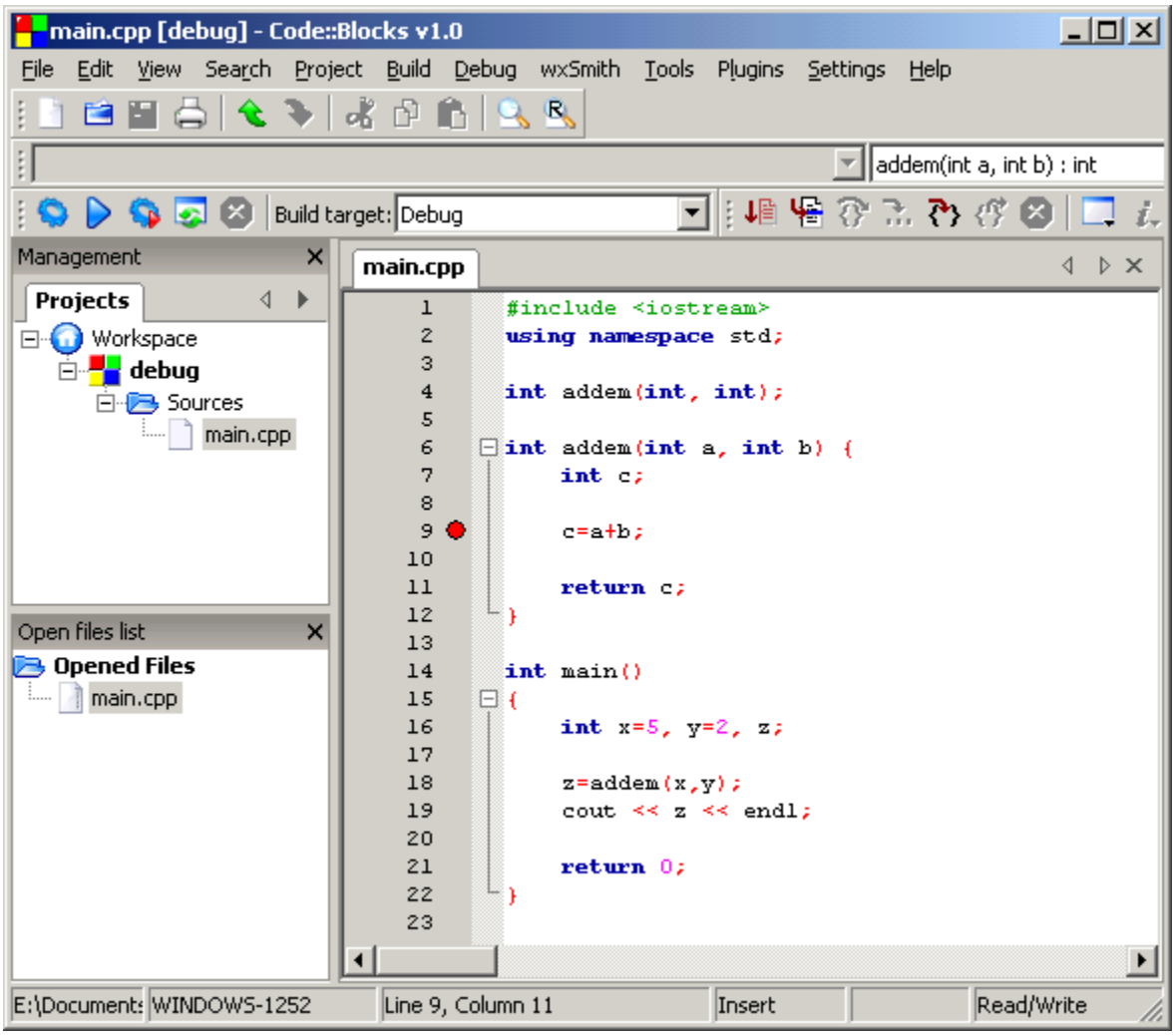
The debug window reflects the change of `c`.



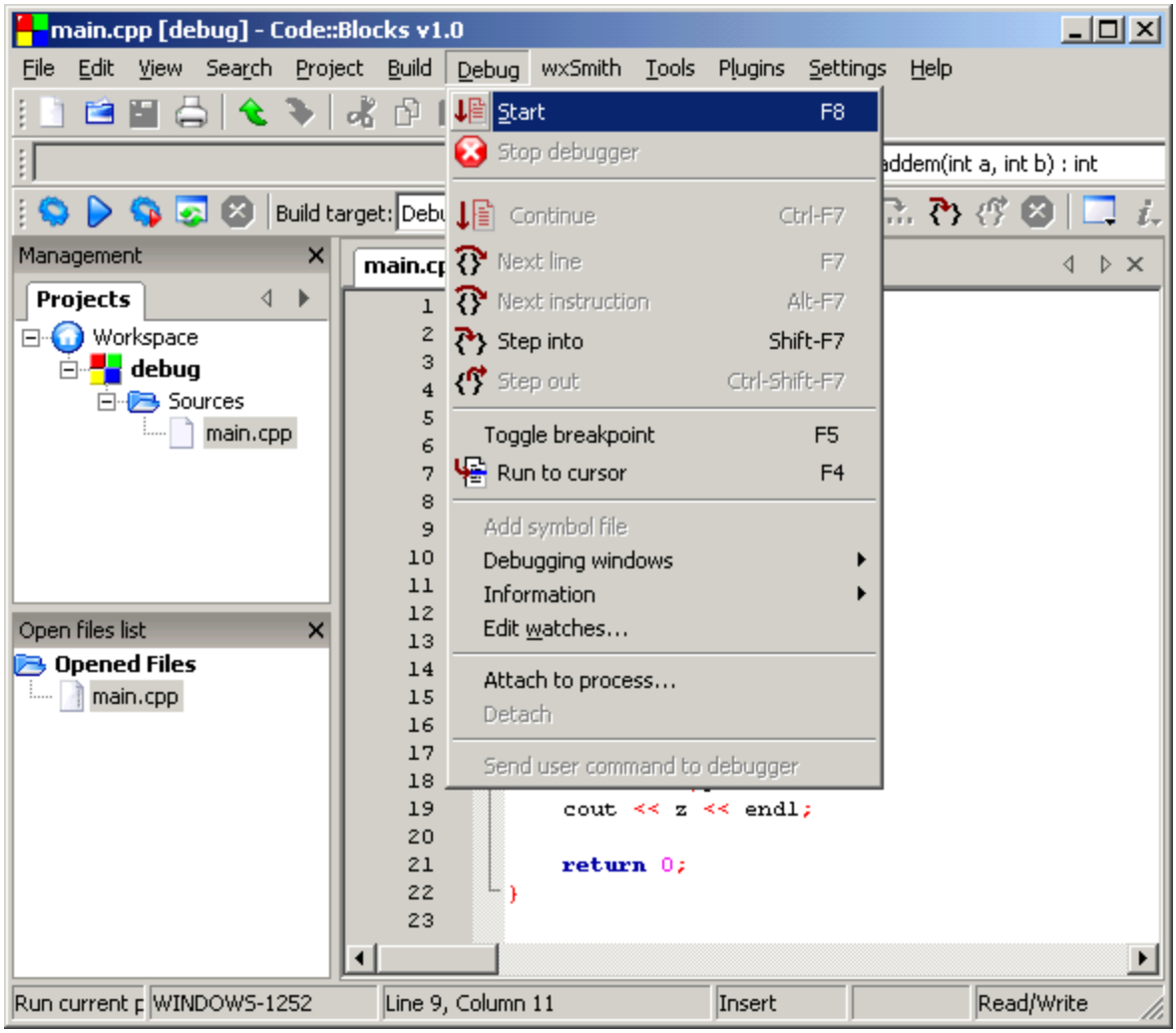
When you are done debugging, you can click on **Continue and your program will run to completion.** This is better than selecting to **Stop debugger**. The reason it is better to **Continue**, is because the program comes to a natural end, rather than aborting. However if your program is stuck in a loop,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



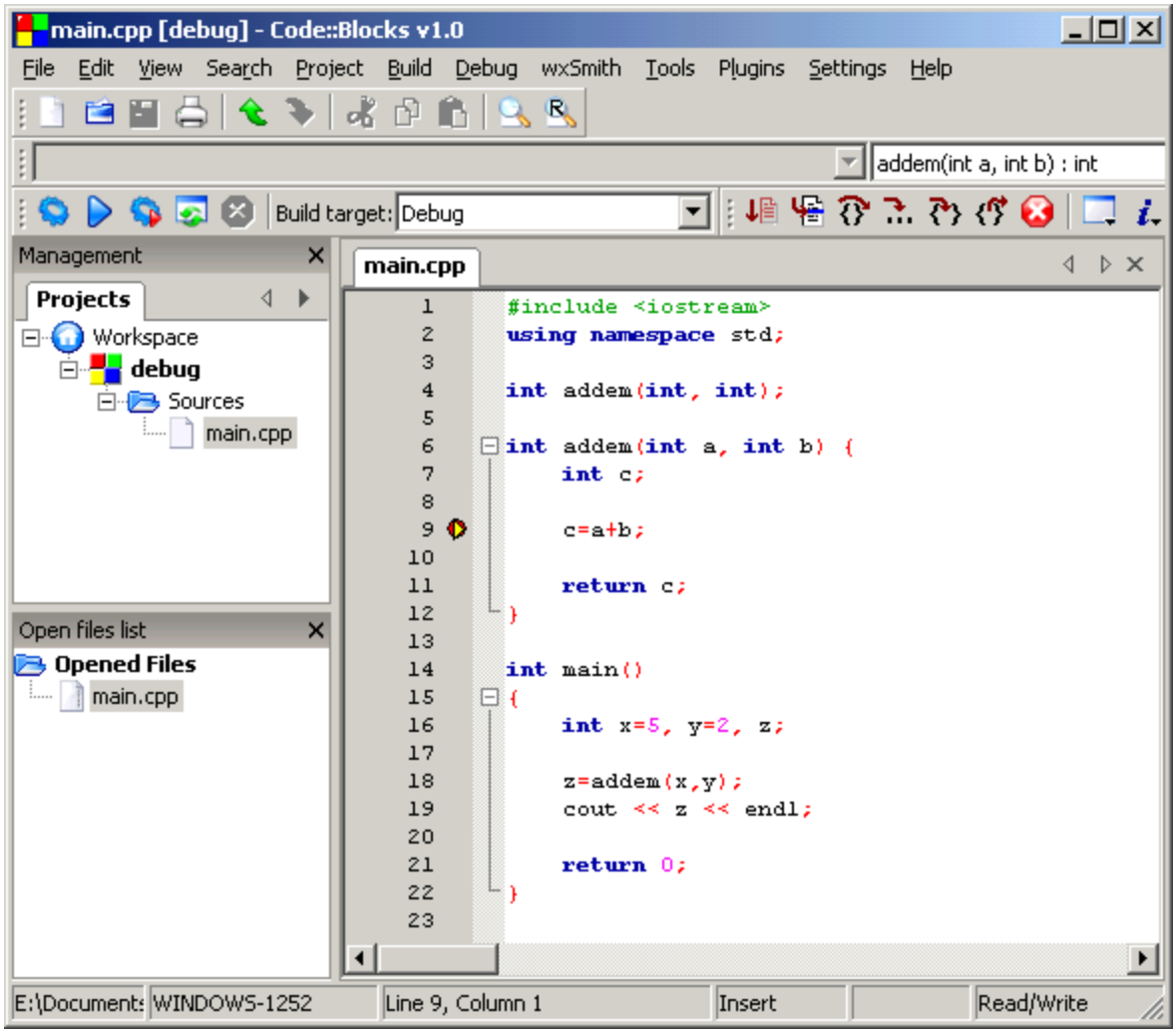
You can further define places in your program to pause and allow you to inspect the code. This is done by setting breakpoints in your code. You can have zero or more breakpoints in your code. When the debugger encounters a breakpoint, the program pauses and the debugger will allow you to inspect your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



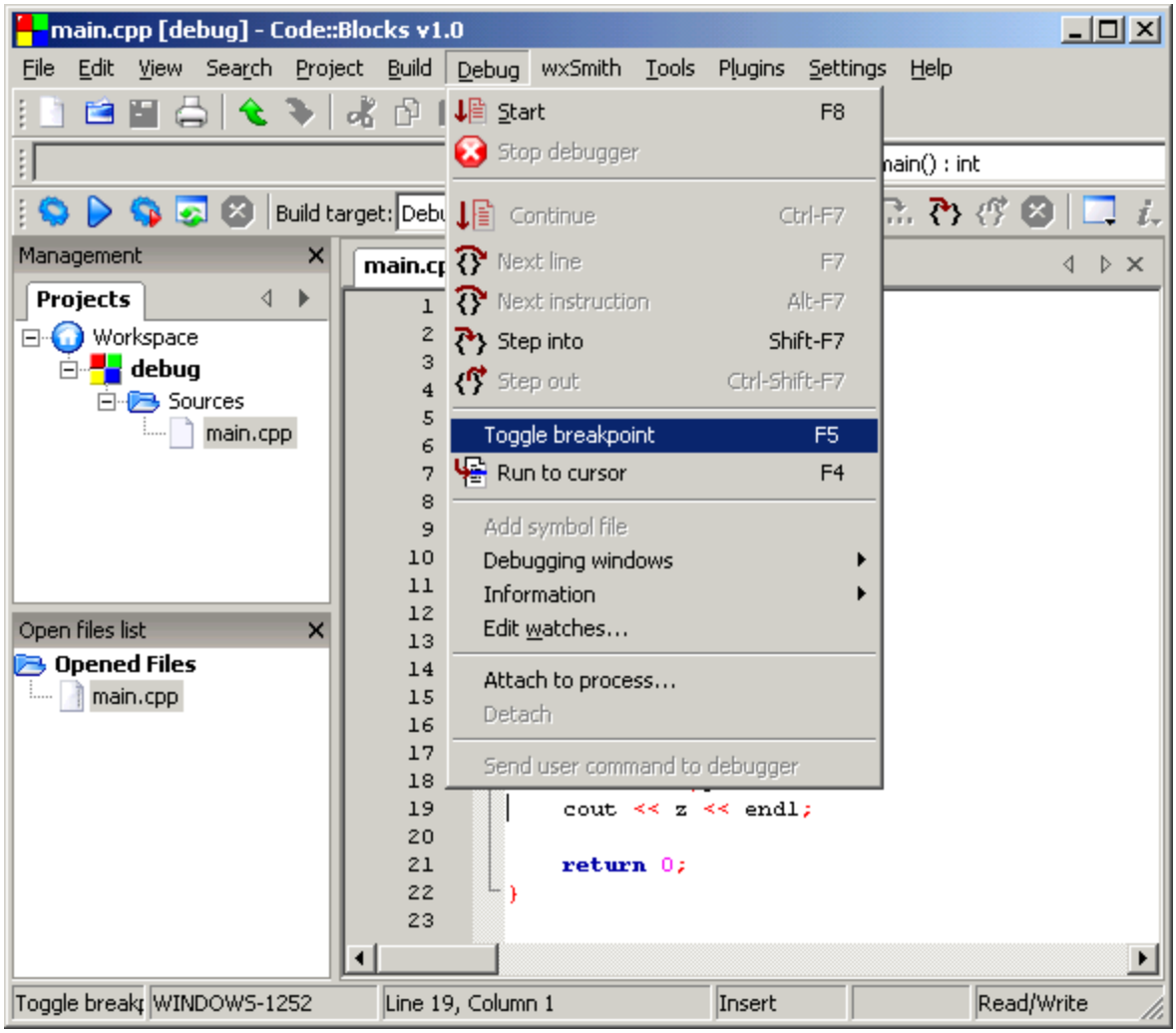
A breakpoint has been set at line 9. The red circle indicates that there is a breakpoint in the code.



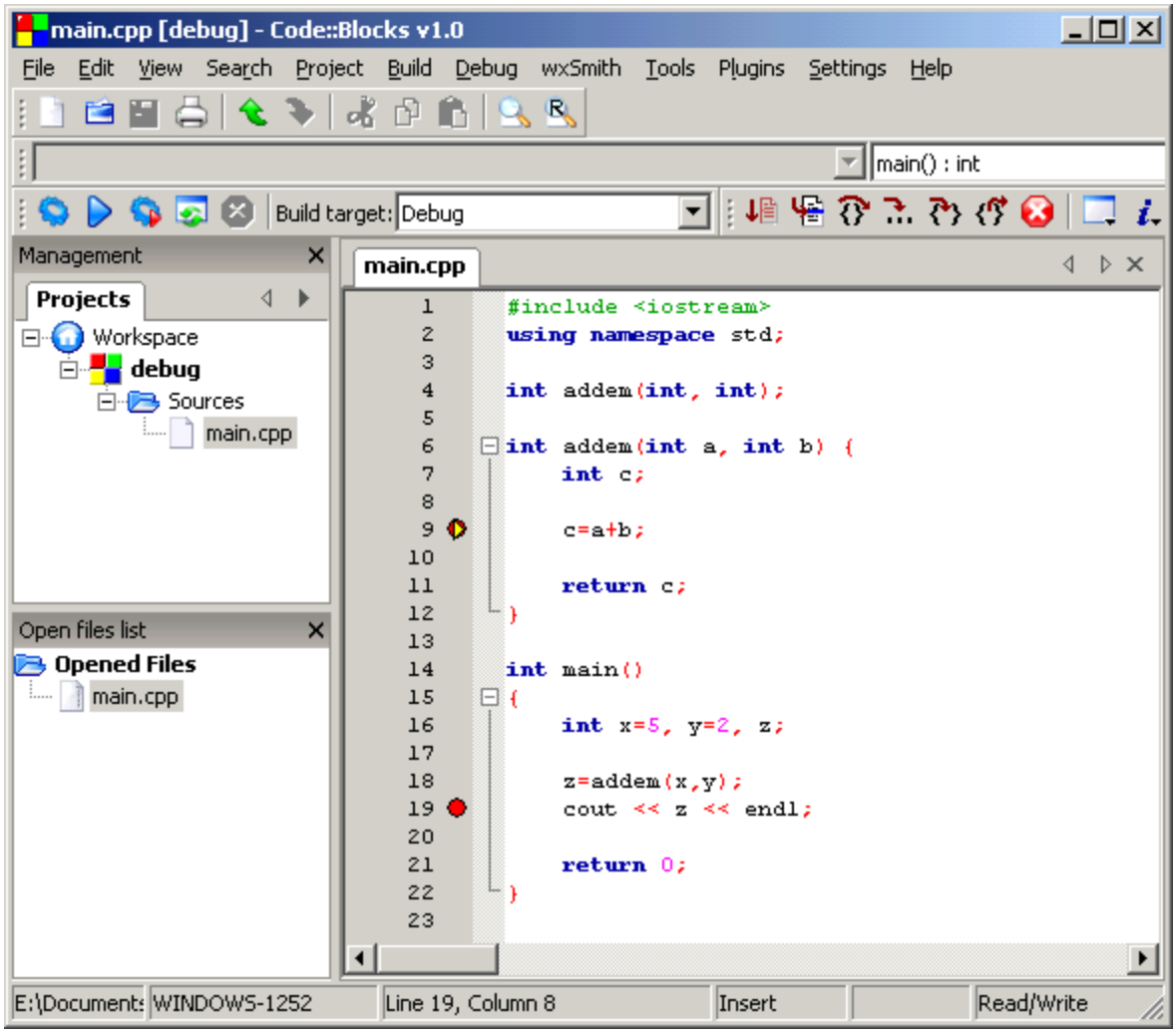
The program is started by selecting from the **Debug** pull-down menu, **Start**. This will run the program in the debugger until a breakpoint is encountered, at which point the program will pause.



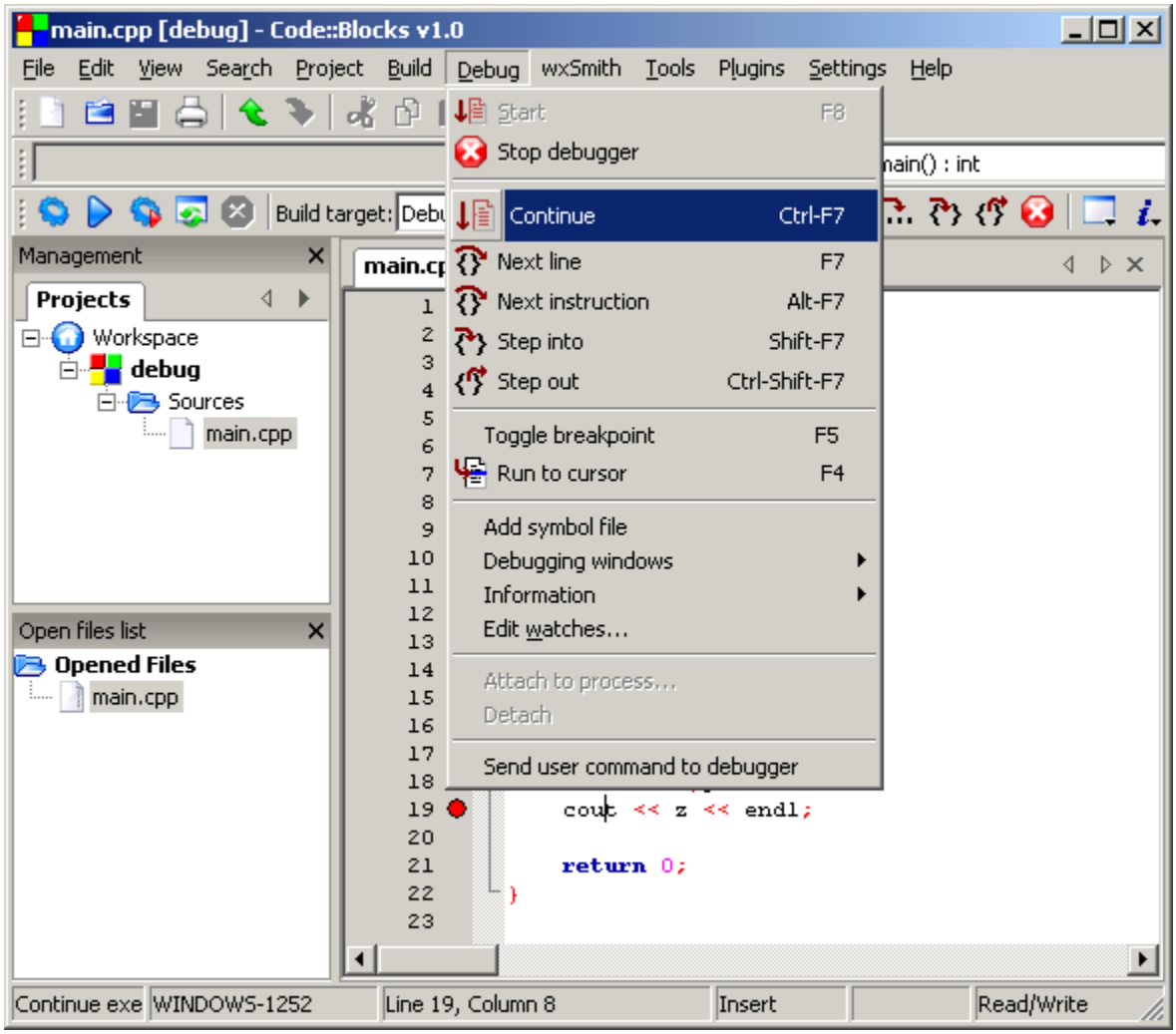
When the program pauses at the break point, a red circle with a yellow triangle mark will appear at the breakpoint.



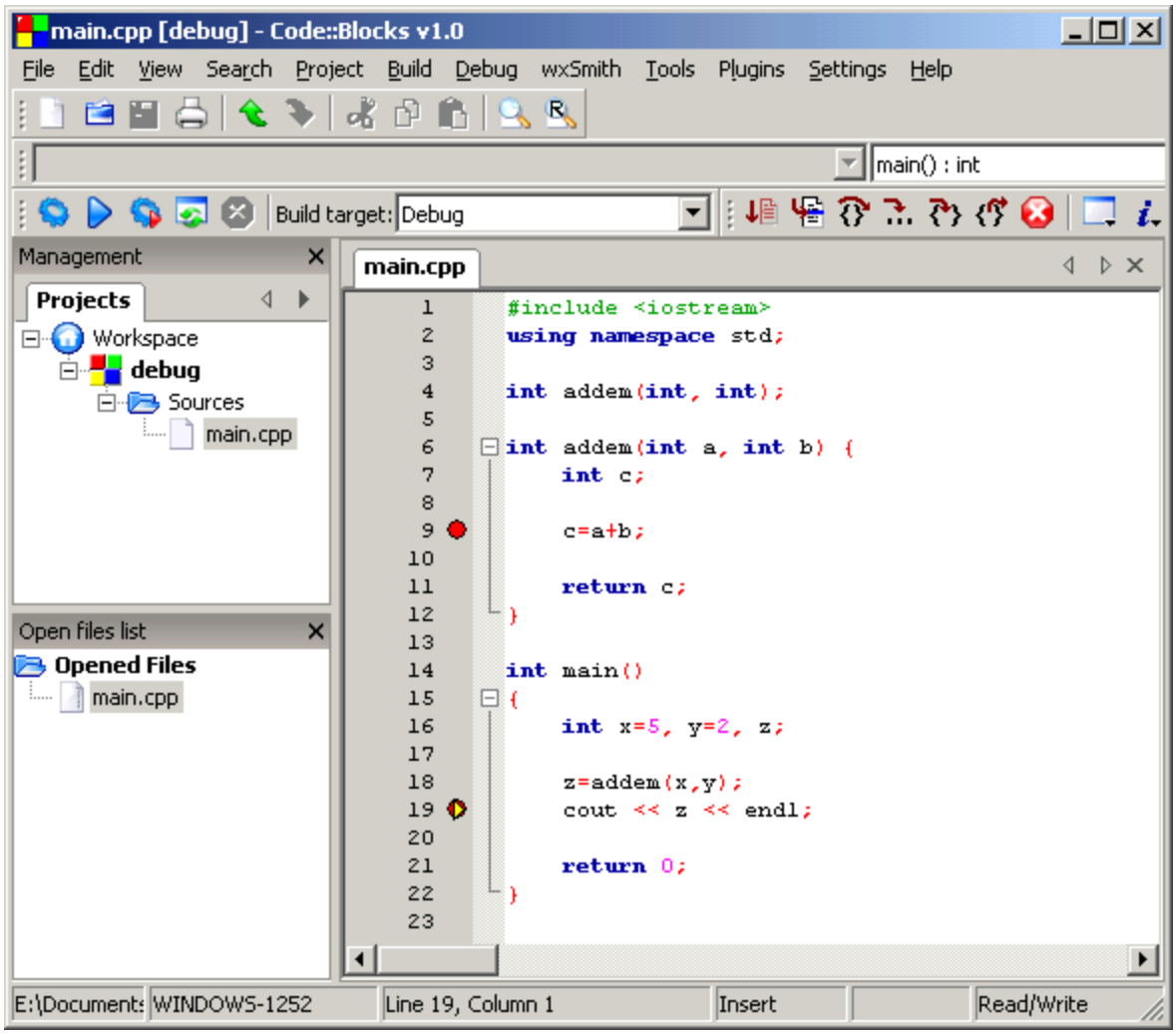
You can set multiple breakpoints. The keyboard shortcut **F5** allows you to toggle the breakpoint at any line.



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

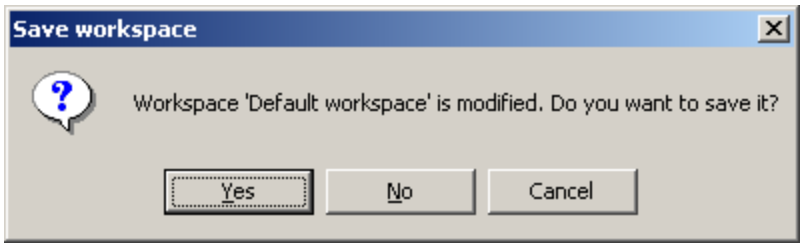


Selecting Continue from the Debugger menu will run the program till the next breakpoint.



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 Apple.
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 Apple, or you can install them from your Mac OS disc. Locate the Xcode installer on the installation discs and double click to install. Check your discs for Xcode or Developer Tools. **Run the installer and install everything other than the documentation:**
        - For Mac OS X 10.3, you need to install Xcode Tools version 1.2 or later
        - For Mac OS X 10.4, you need to install Xcode Tools version 2.2 or later

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

Installation for Fedora 8 Linux:

Installing Code::Blocks

In a console window, become **root** and then execute the following command:

yum install codeblocks

Running Code::Blocks
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.