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

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 in many instances 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.

Table of Contents

The first step in using the Code::Blocks IDE is to download it. The IDE can be obtained either for Windows-based or Linux-based systems at:

http://www.codeblocks.org/

http://www.codeblocks.org/downloads.shtml

The Linux based IDE can be downloaded as an **.rpm** file (Red Hat Package Manager and can be installed using the rpm utility within Linux). You can download the open source, cross platform Free C++ IDE at the following locations.

For convenience, the Windows-based system has been configured for you as a self extracting executable installation file (**.exe** file) and placed at:

http://www.sci.brooklyn.cuny.edu/~goetz/codeblocks/

This site has two installation files. One file is called **MinGW**, which is the **C++ compiler** and another file **Code Blocks Nightly,** which can be downloaded to any folder, including the Desktop.

These two files will then be used to install the IDE.

The installation instructions explained on the following pages, refer *only* to the files downloaded from the self installing exe files found on http://www.sci.brooklyn.cuny.edu/~goetz/codeblocks/ .
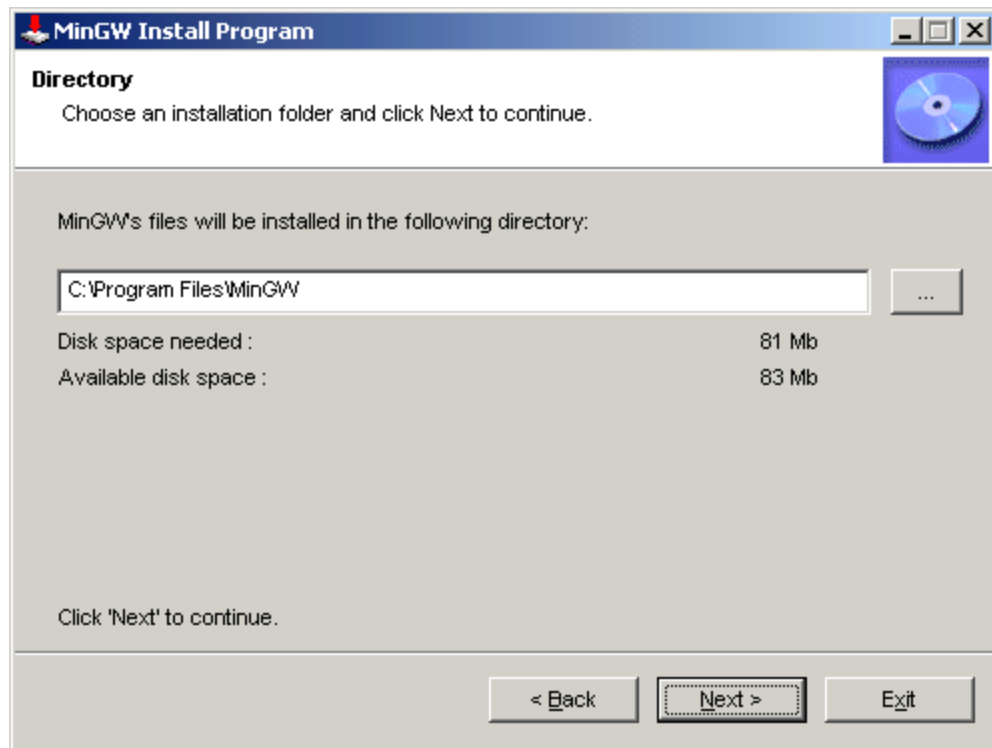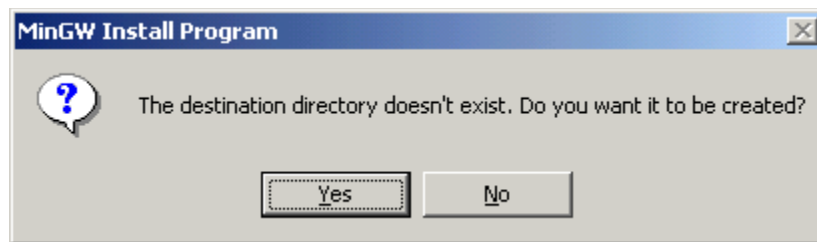
Installation

Installing the compiler

After downloading the two files, MinGW and The Nightly builds, the first step will be to install the MinGW compiler. This is accomplished by opening (clicking on) the install file (MinGW), bringing up the following window:
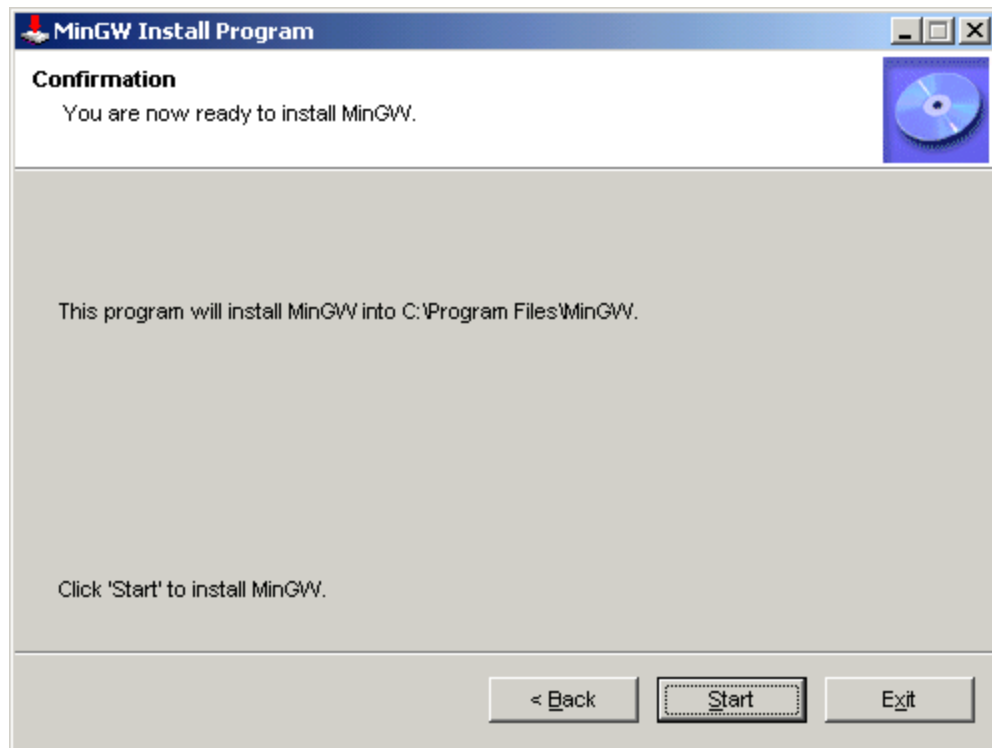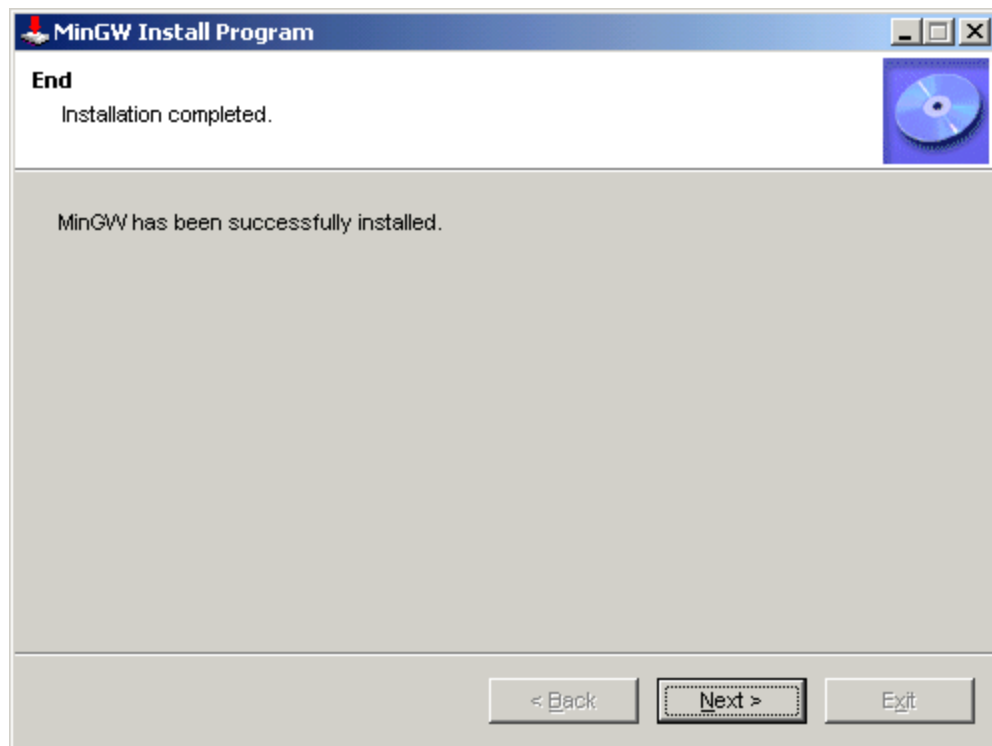

Choose Next to continue

You may select a different installation directory. However it's easiest to take the default.
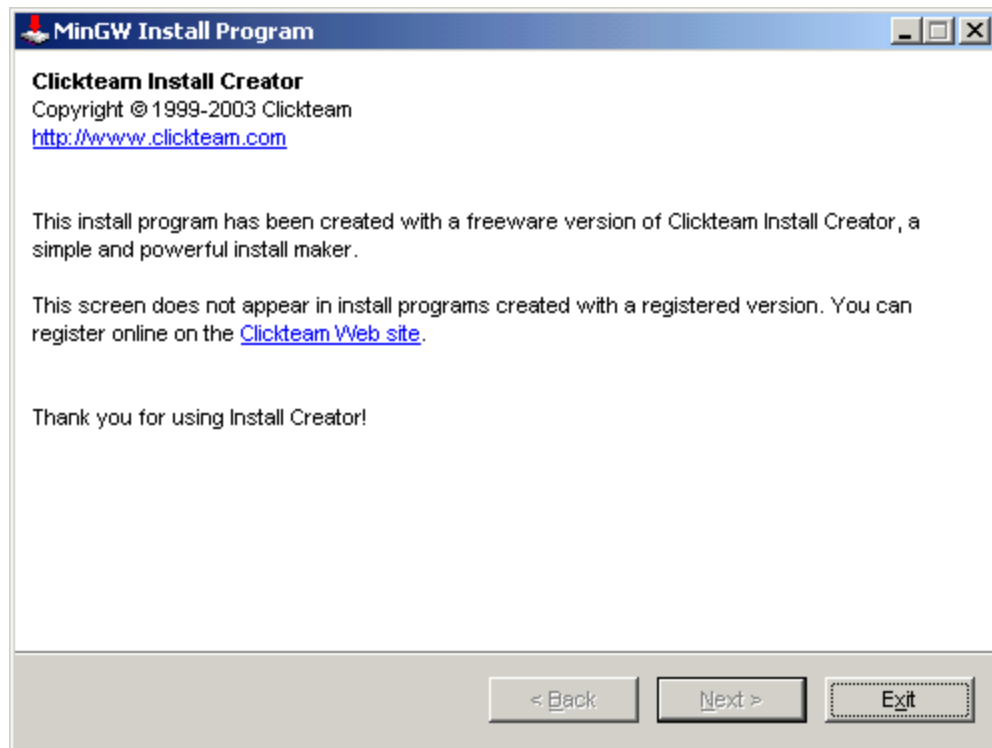Choose Next to continue.

Choose Yes.

When you are ready, choose Start to start the installation. Please wait while the installation is preformed.
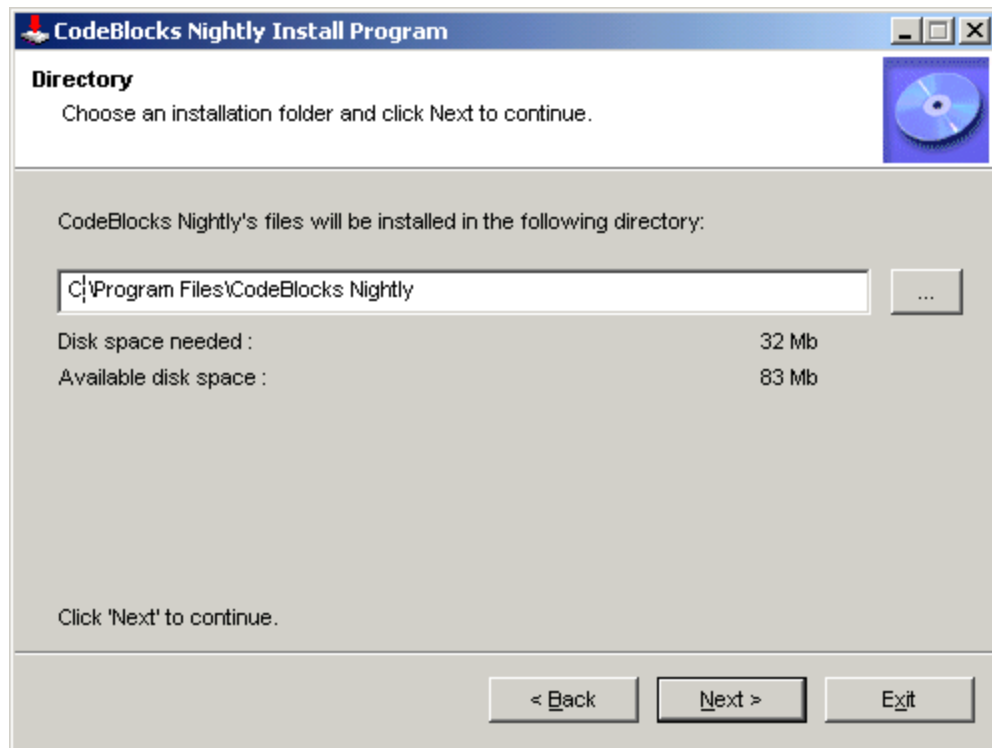

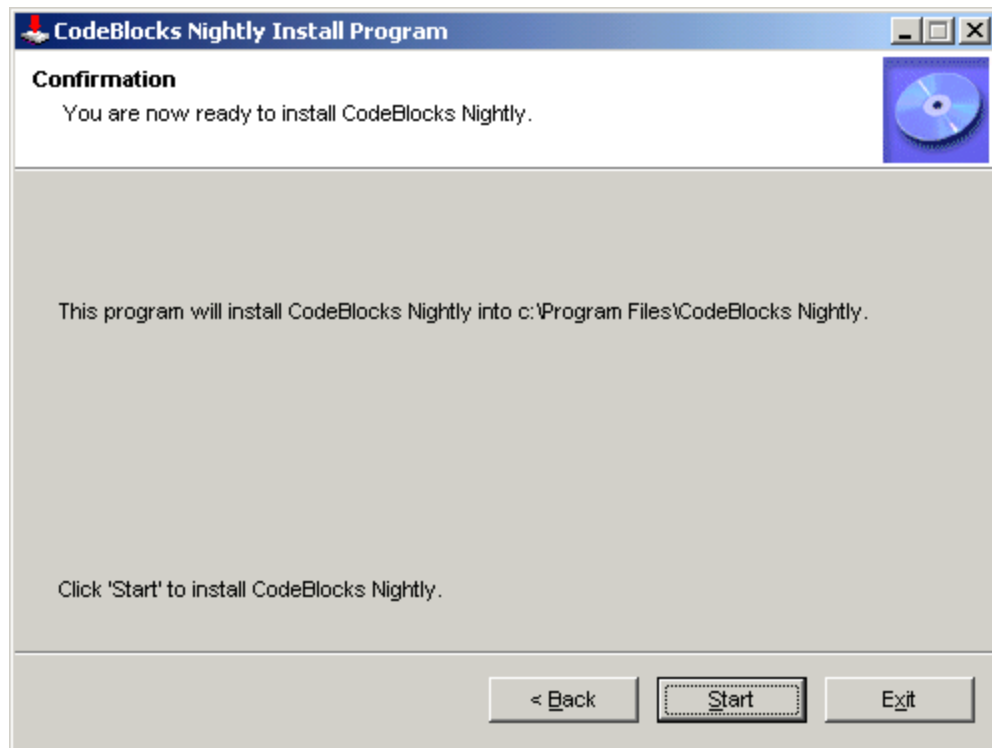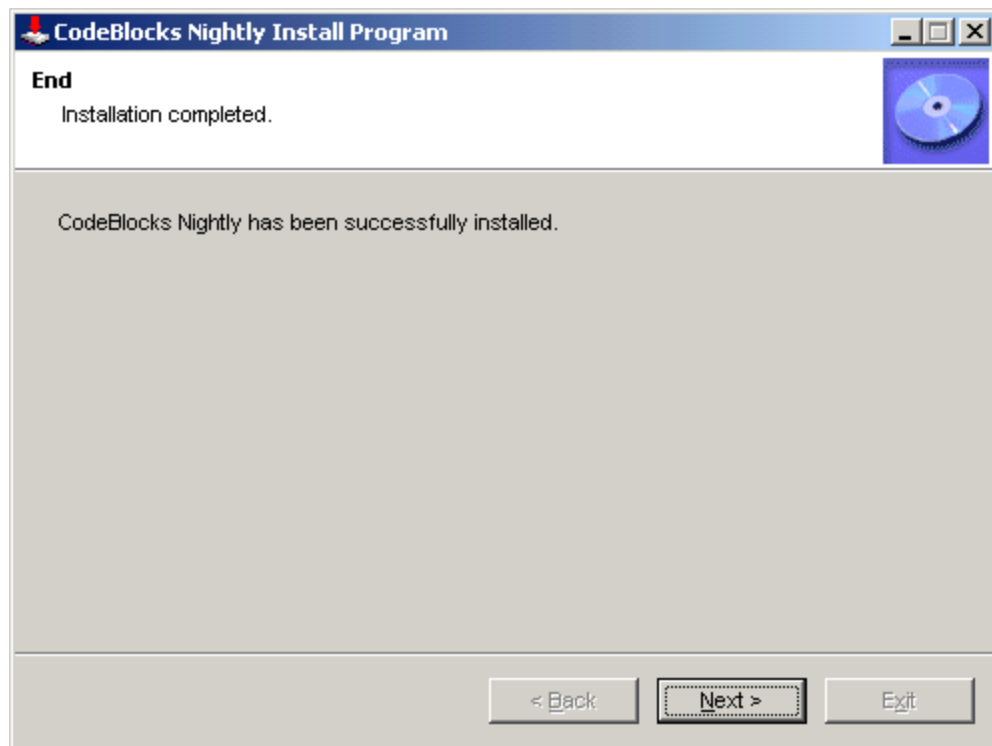
Choose Next to go to the exit screen.

MinGW Install Program

Clickteam Install Creator
Copyright © 1999-2003 Clickteam
http://www.clickteam.com

This install program has been created with a freeware version of Clickteam Install Creator, a simple and powerful install maker.

This screen does not appear in install programs created with a registered version. You can register online on the Clickteam Web site.

Thank you for using Install Creator!

< Back | Next > | Exit

Choose Exit to exit the application.

Installation of Code Blocks

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

Choose Next to continue.

CodeBlocks Nightly Install Program

Directory
Choose an installation folder and click Next to continue.

CodeBlocks Nightly's files will be installed in the following directory:

C:\Program Files\CodeBlocks Nightly

| Disk space needed : | 32 Mb |
| Available disk space : | 83 Mb |

Click 'Next' to continue.

< Back   Next >   Exit

You may select a different installation directory. However it's easiest to take the default.
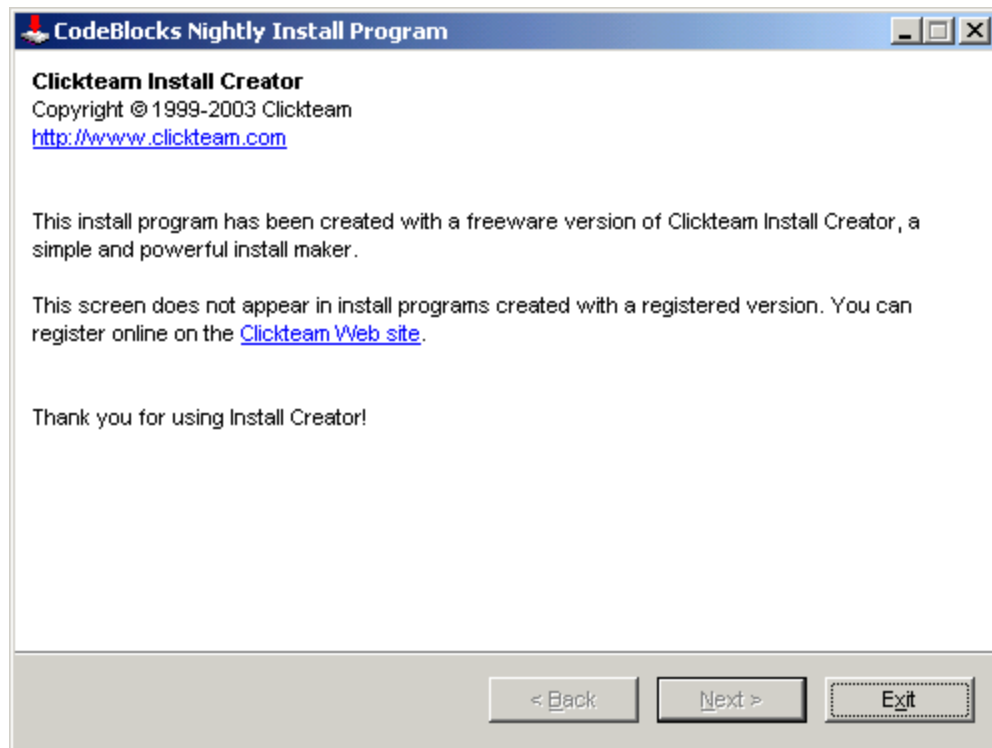Choose Next to continue.

MinGW Install Program

The destination directory doesn't exist. Do you want it to be created?

Yes   No

Choose Yes.

When you are ready, choose Start to start the installation. Please wait while the installation is preformed.

Choose Next to go to the exit screen.

CodeBlocks Nightly Install Program
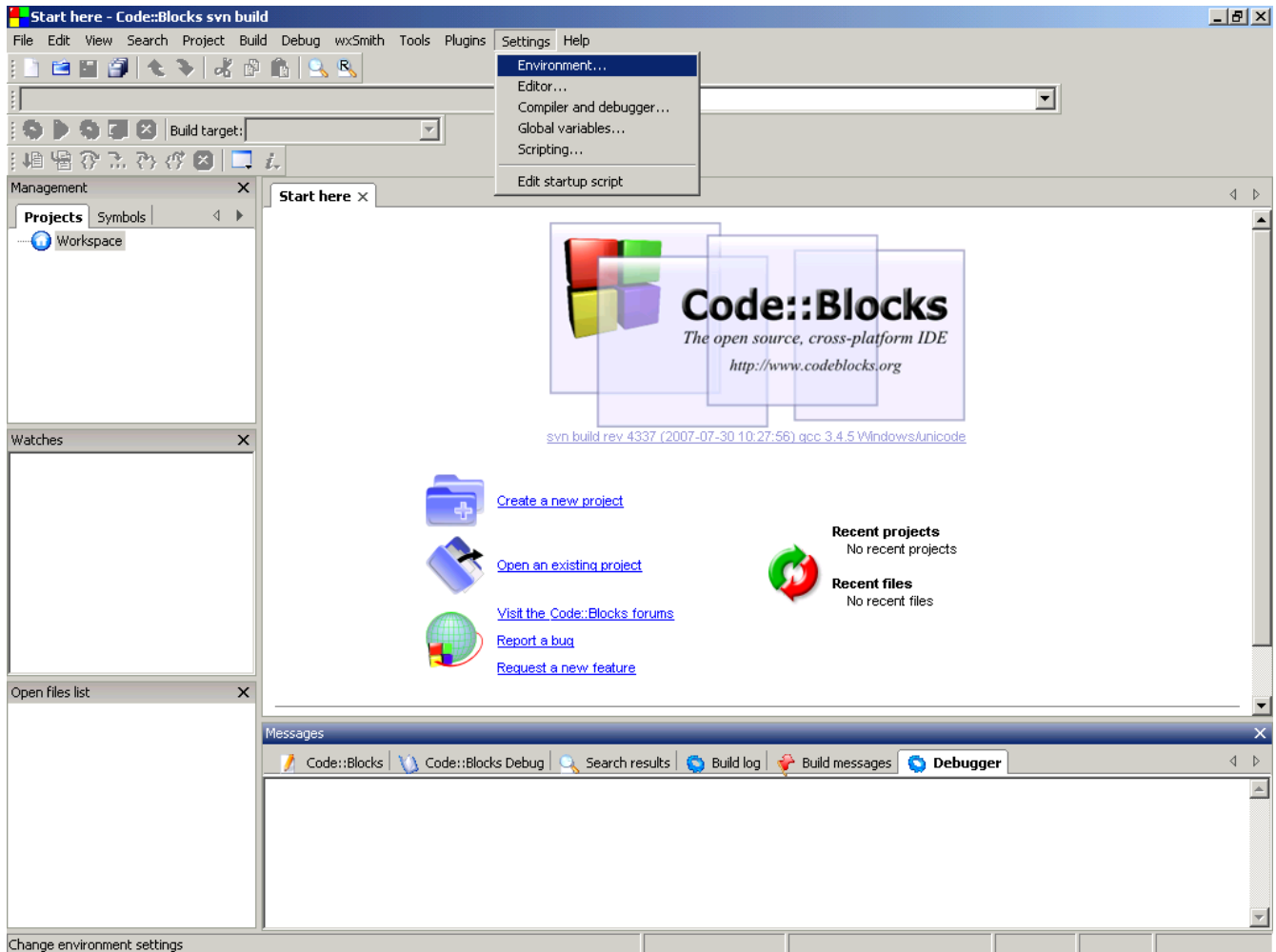
Clickteam Install Creator
Copyright © 1999-2003 Clickteam
http://www.clickteam.com

This install program has been created with a freeware version of Clickteam Install Creator, a simple and powerful install maker.

This screen does not appear in install programs created with a registered version. You can register online on the Clickteam Web site.

Thank you for using Install Creator!

< Back | Next > | Exit

Choose Exit to exit the application.

When you have finished the Compiler and CodeBlocks installation, you should have two folders in the **Programs Files** folder: MinGW and CodeBlocks Nightly.

After installing Code::Blocks, and icon will be generated on your desktop, which is a shortcut to the Code::Blocks program:

Code Blocks

Starting Code::Blocks

After you open up Code Blocks, you should setup the association of C++ to be used within the Code Blocks' IDE. However if you are using another C++ IDE and do not wish to associate C++ files with Code Blocks, you may skip the following steps.

From the **Settings** menu, select **Environment**.

Environment settings

General settings

General settings

Note... app...

- ☑ Show splash screen on start-up
- ☑ Allow only one running instance (needs application restart to take effect)
- ☐ Check & set file associations (Windows only)   [Set now]   [Manage...]
- ☑ Check for externally modified files
- ☑ Ignore Invalid Targets

On application start-up
- ○ Open default workspace
- ● Open blank workspace

On project load
- ○ Open all project files
- ● Open last open files
- ○ Do not open any files

Shell to run commands in: /bin/sh -c

Terminal to launch console programs: xterm -T $TITLE -e

[OK]   [Cancel]

To have Code Blocks associated with C++ files, click on the **Set Now** button. Then click on **OK**.

You can also make changes to the Code::Blocks editor by clicking on the **Settings** pull-down menu and clicking on **Editor**. However, unless your professor tells you otherwise, you should leave these settings alone.

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**.
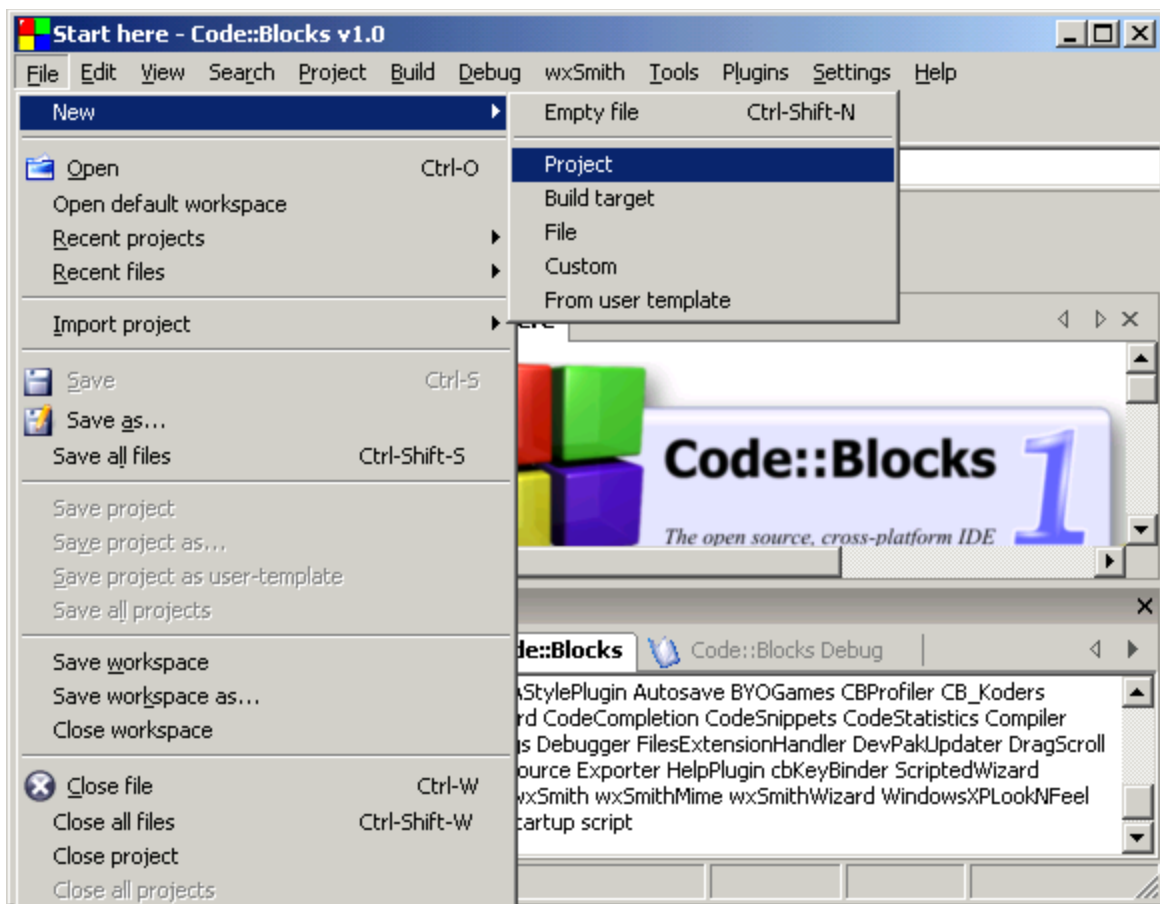
This will bring up the **New from template** window. Opening (clicking on) **Console Application** will then allow you to write a program on the console. The other application are for developing more advanced types of applications. After selecting **Console application,** click on the Go button to begin using the Console Application Wizard.
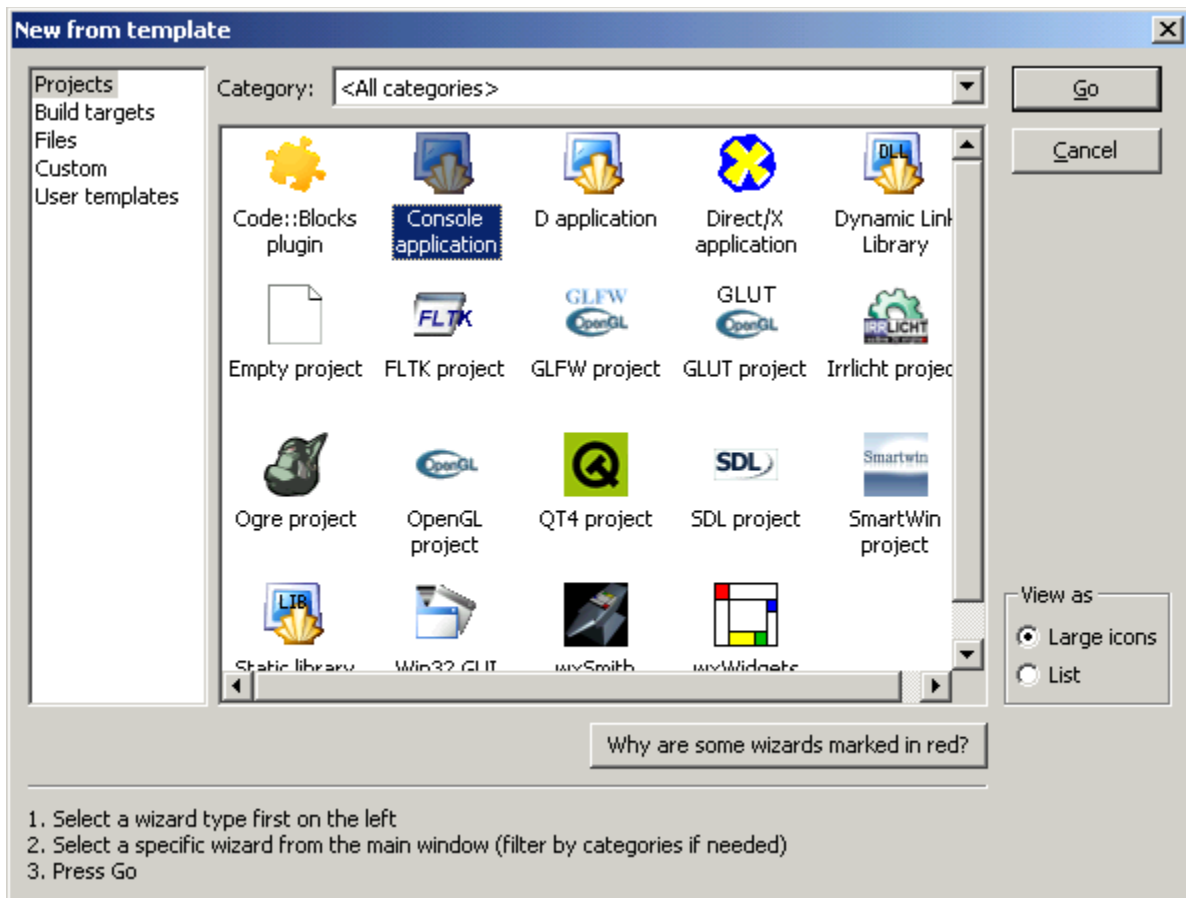
Press Next to go to the next step.
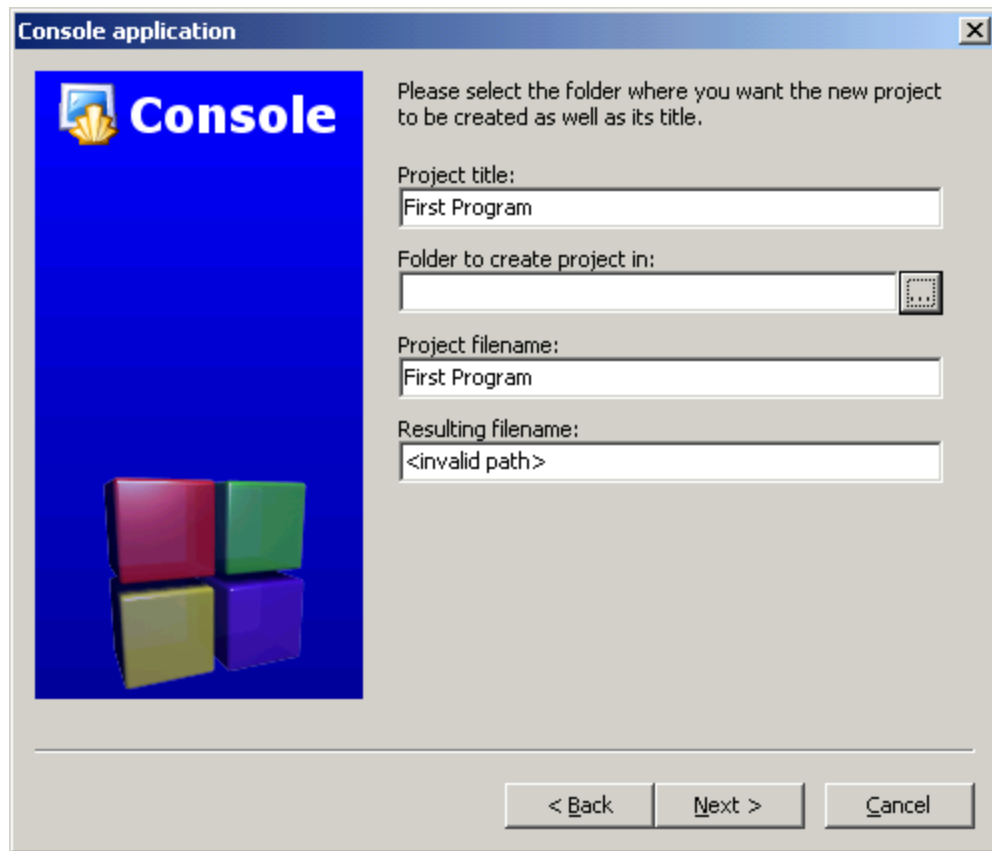
Start by filling in the Project Title. You will notice that the Project Filename automatically becomes the same name. If you wish, you can change the filename, but for simplicity leave it as is. To specify the location of the folder to contain the project, click on the "..." button (selected in the picture above) and browse to a folder on your drive to store the project. Generally, you can save it in My Documents.
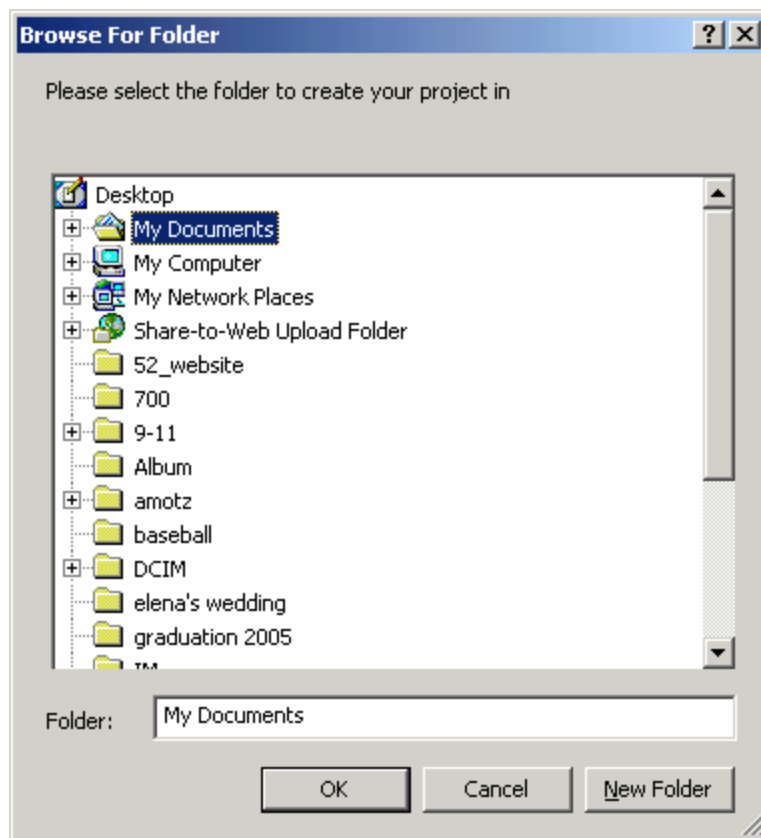
Press Ok after selecting My Documents
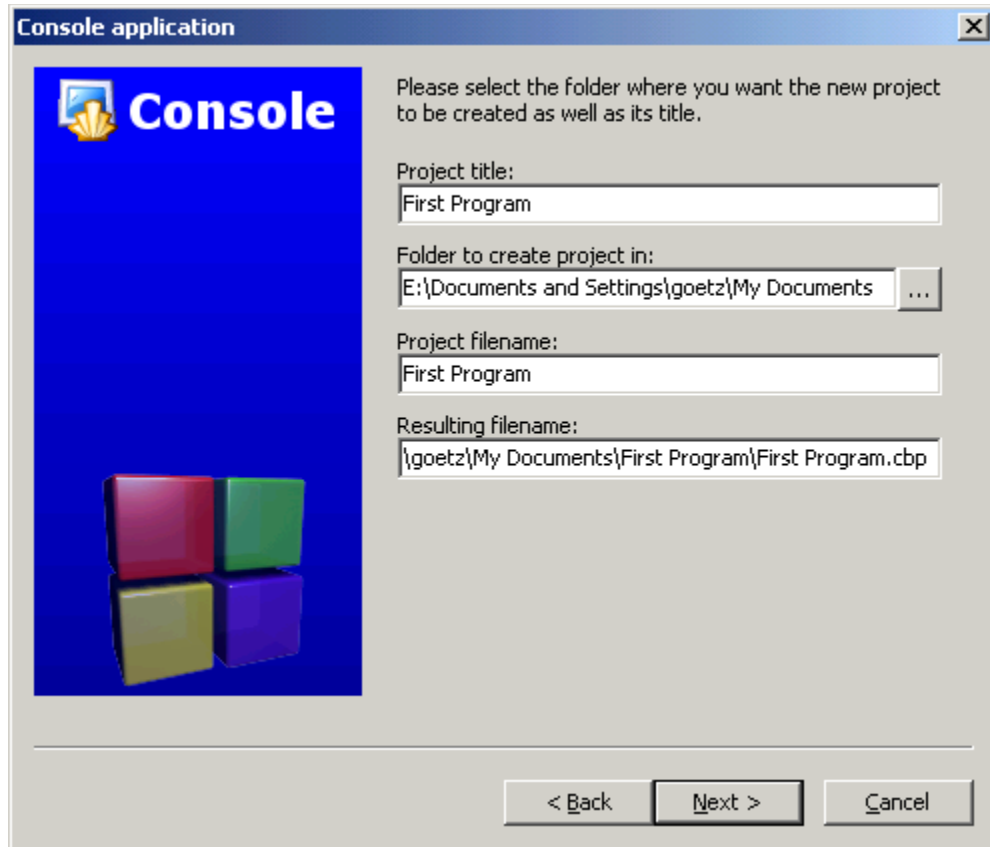
When the directory for your project has been selected, the system will return to the **Console application**. As shown, Code Blocks will create a directory called First Program (Project Title) and returns your selected directory in **Folder to create project in**. Inside that directory will be the **Project 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.
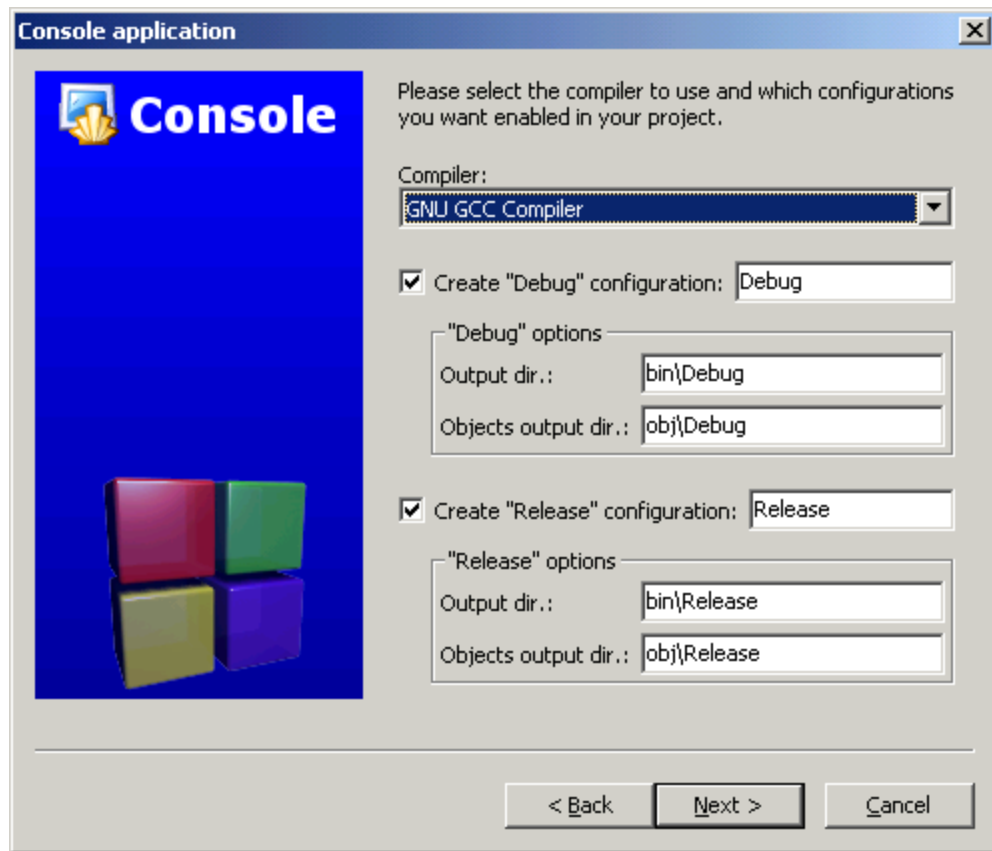
The Next window to pop up will be the Compiler screen. This specifies where the **Debug** and **Release** compiled versions of your program will be placed. Leave this setting alone and press Next.
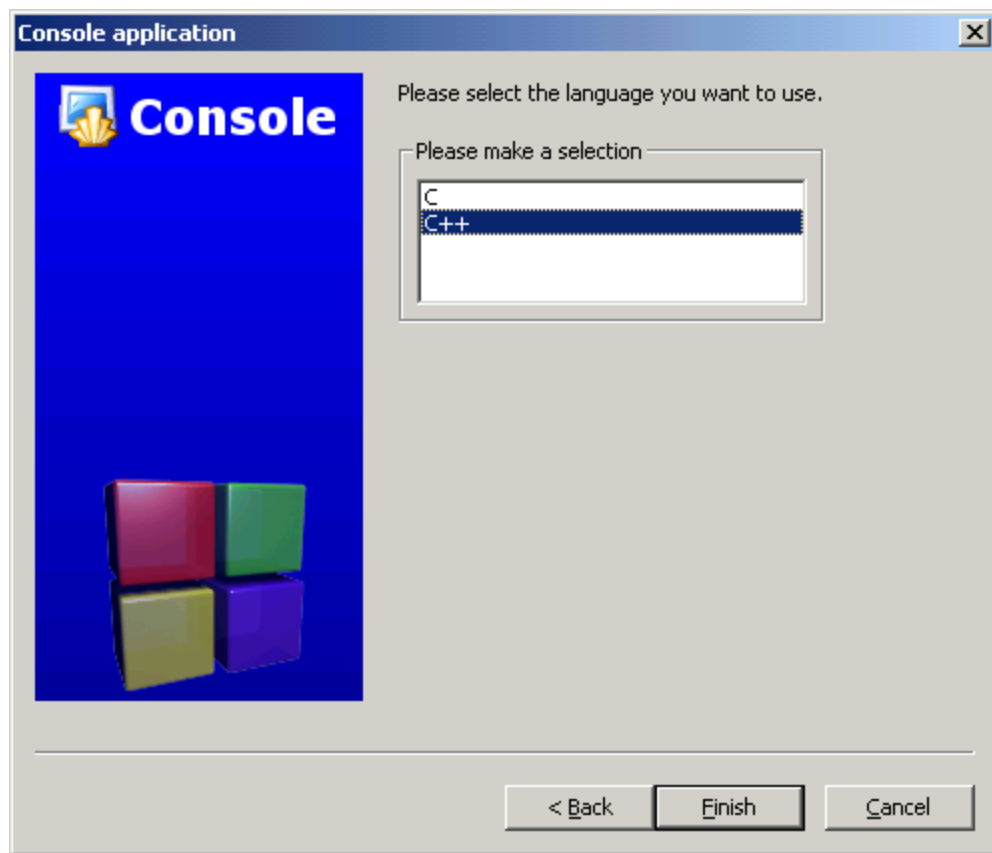
The next window allows you to choose the language that you will use. Select the language as C++, then press Finish.
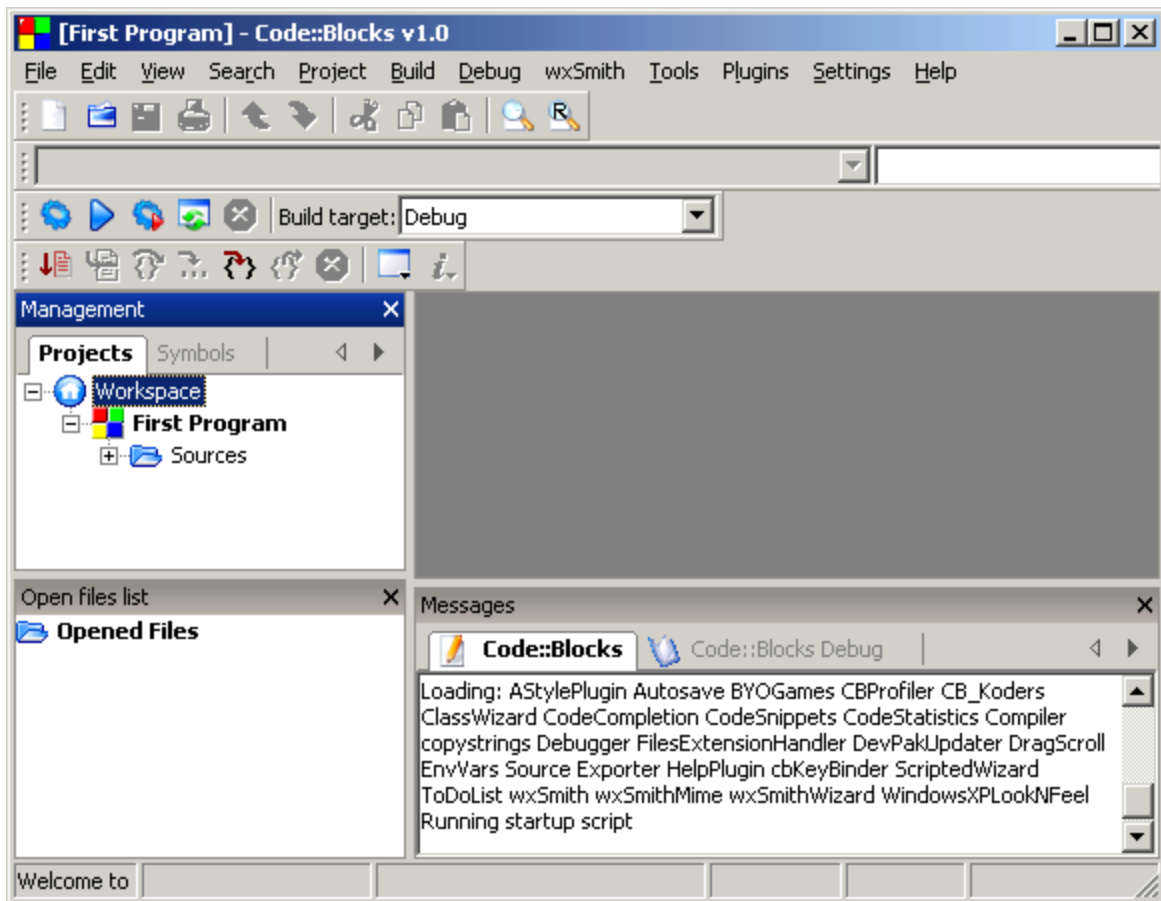
The system will then return to the **[First Program]** window and you are ready to write your program. It should be noted that the Build target is **Debug**, which will allow you to use the debugger to find errors. In the **Management** area of the screen (Shift-F2 toggles the Management display), you 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.
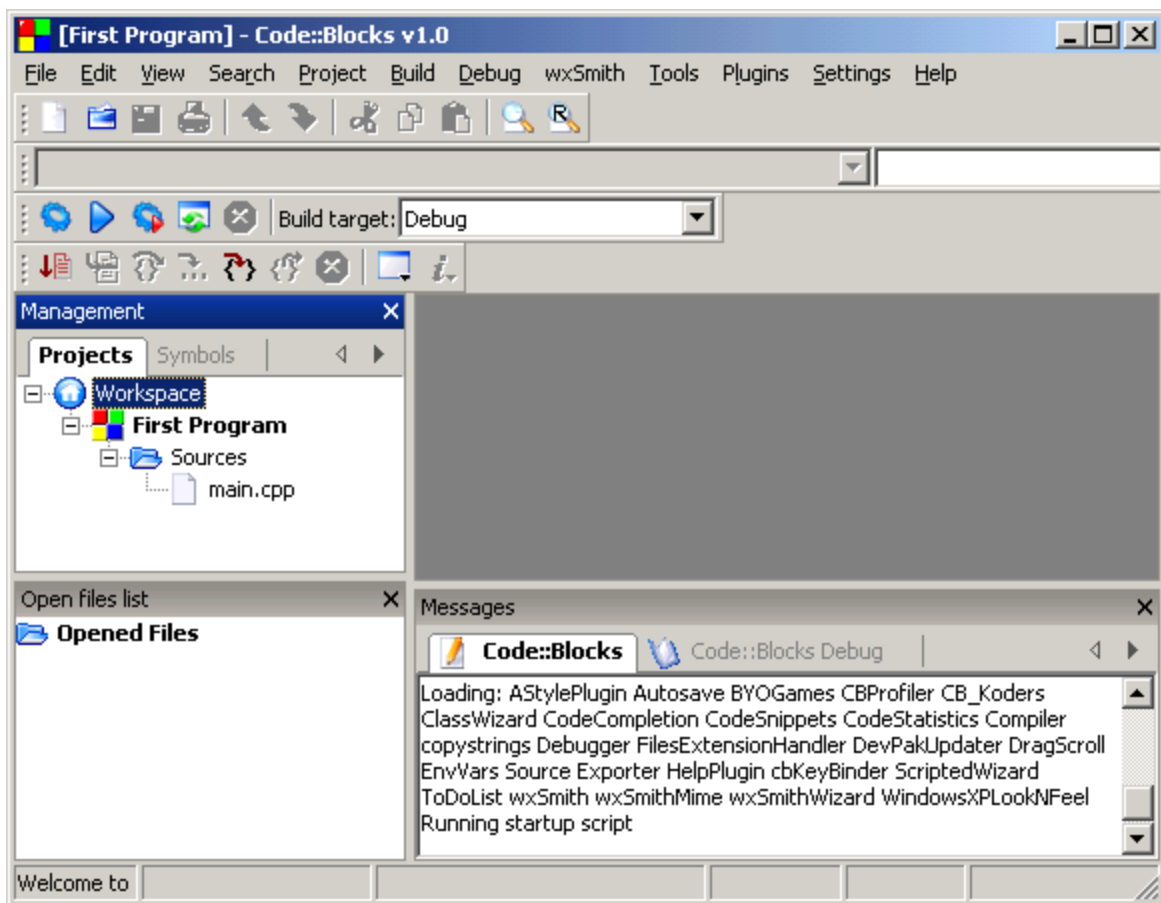
Under Sources, there is a file called main.cpp, which is automatically created for you when you build a console application.

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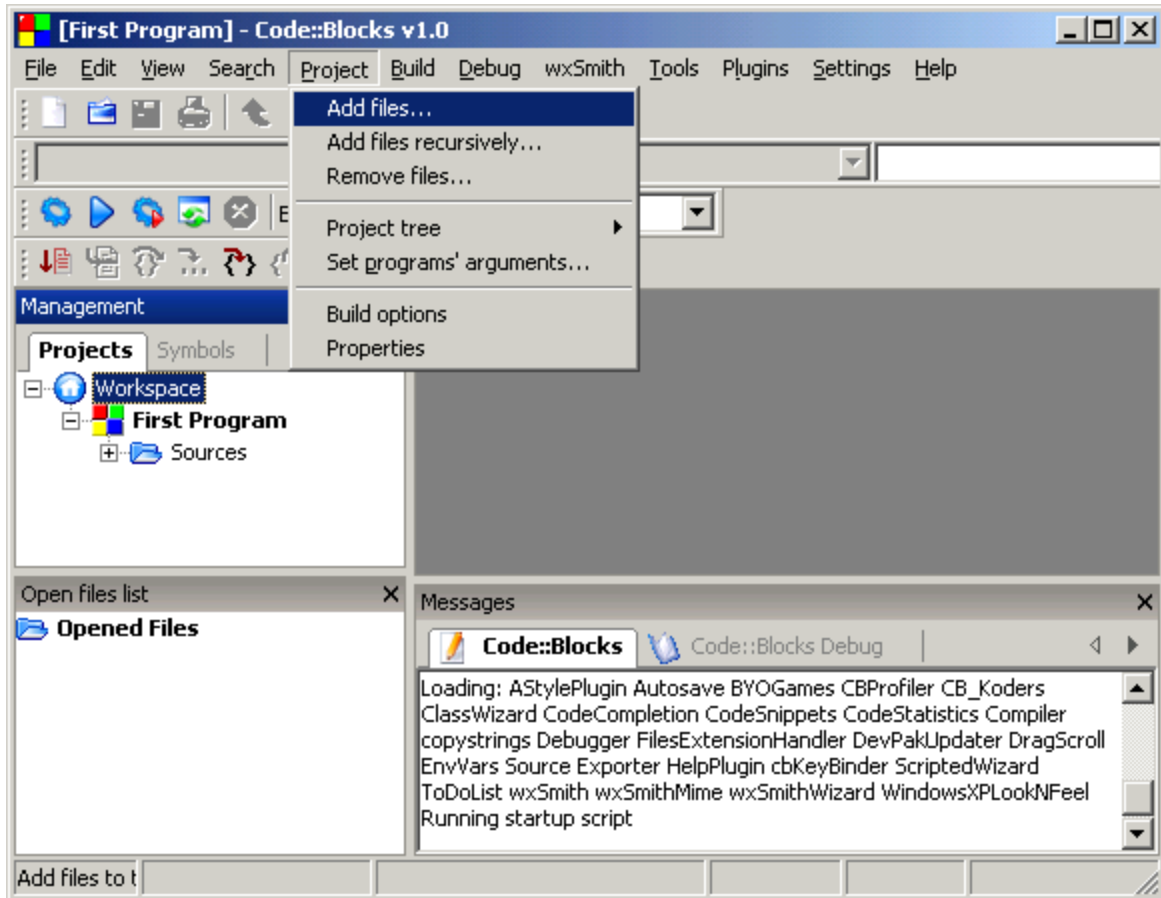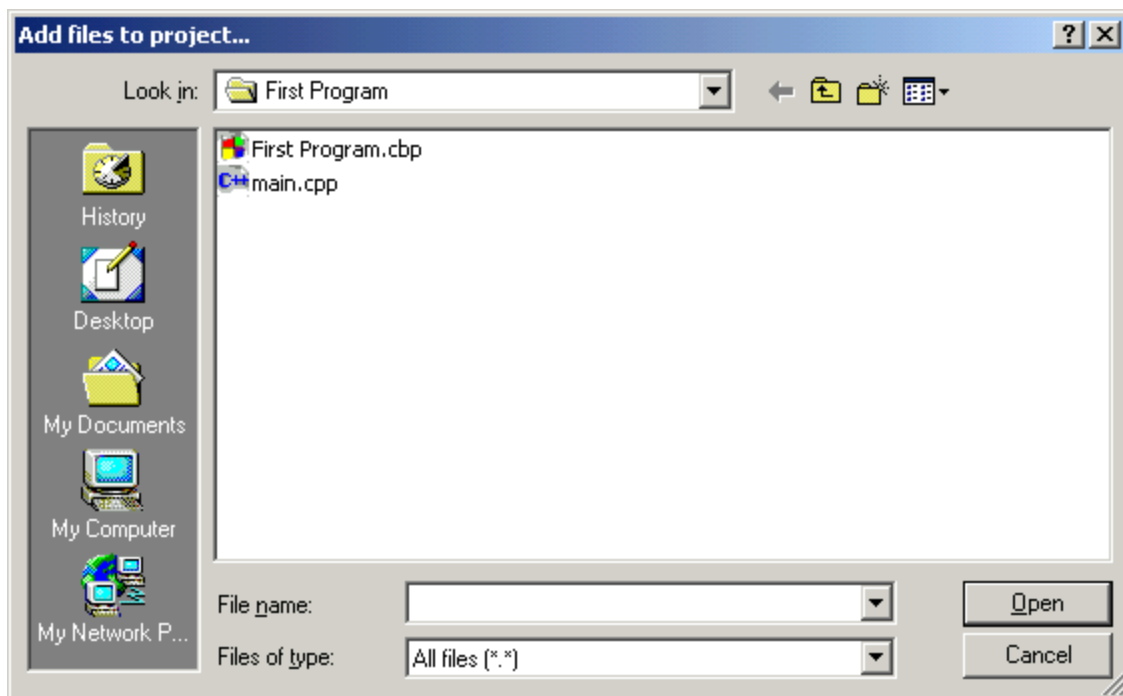
Clicking on **Add files to project,** will bring up a window so you can browse to where your files that you wish to add are**.** Select any additional file you want to add and press Open. The file will then be added to your project.

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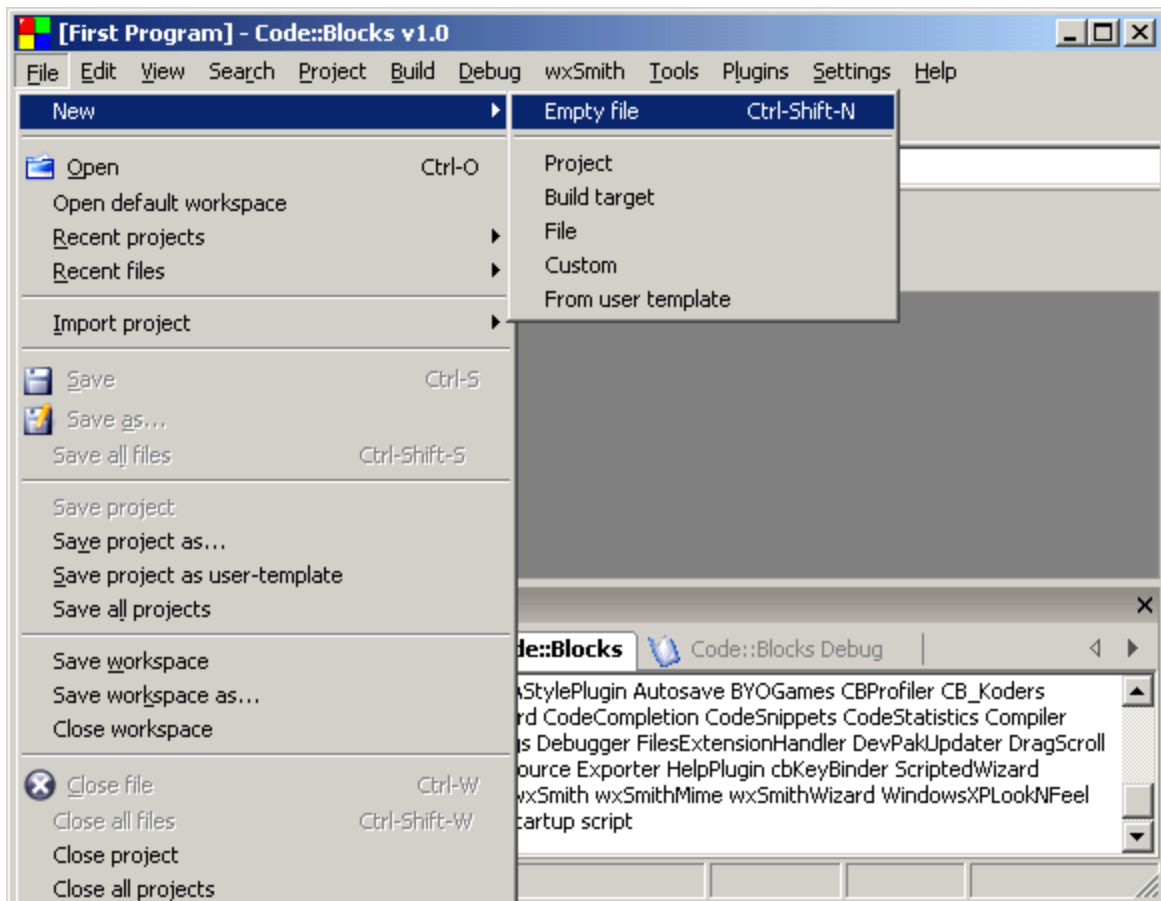
File, New, Empty file.

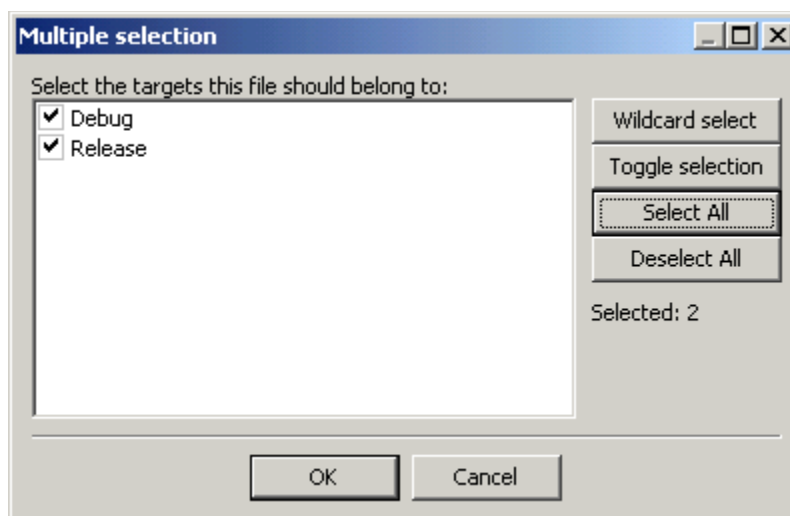You will be asked if you want to add this file to the project.
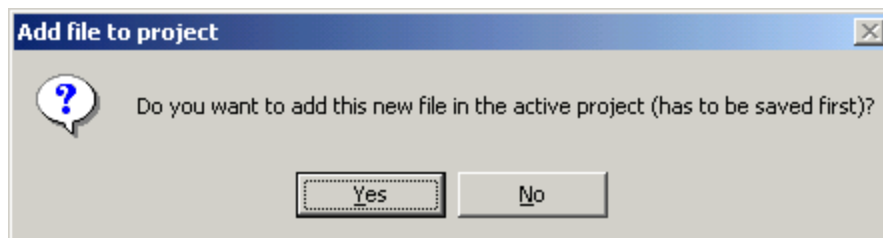

Choose Yes.

Code Blocks will ask for a file name to save the file as:
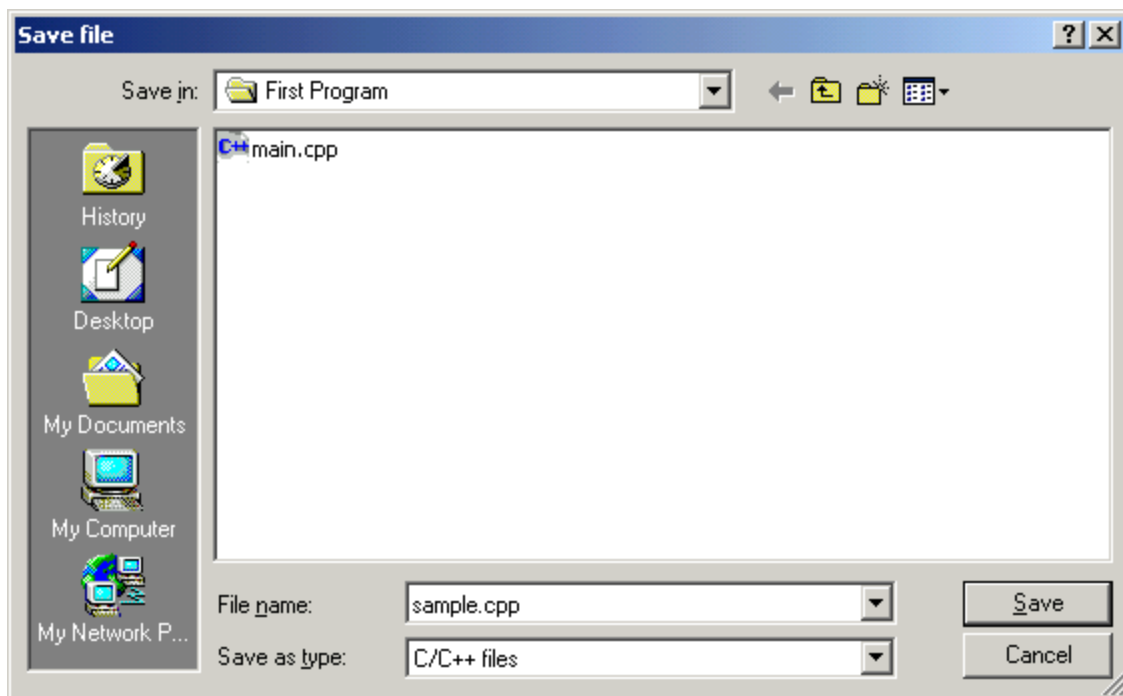
Give a name to the file. Pick a name that is related to the content of the file. Here it is called sample.cpp. C++ files need to be of the type cpp. Press Save to save the file.


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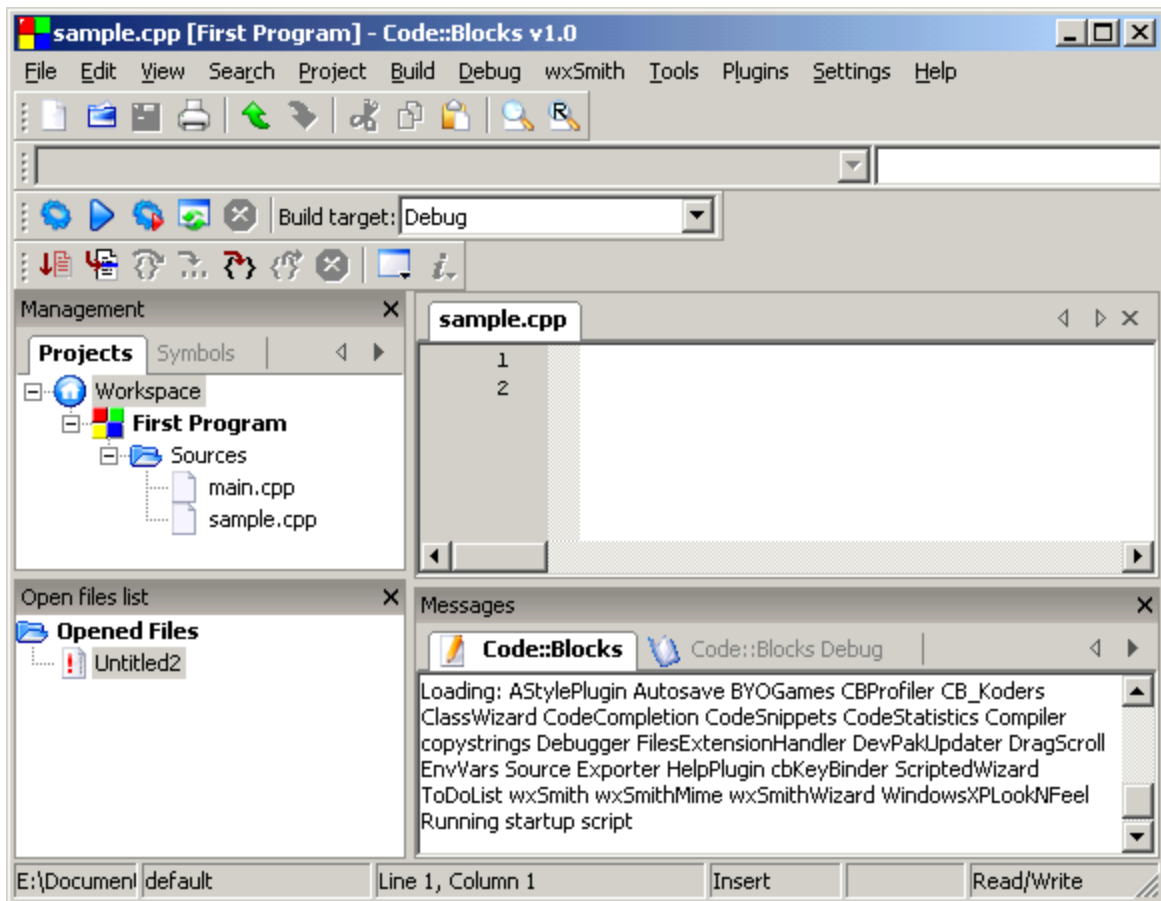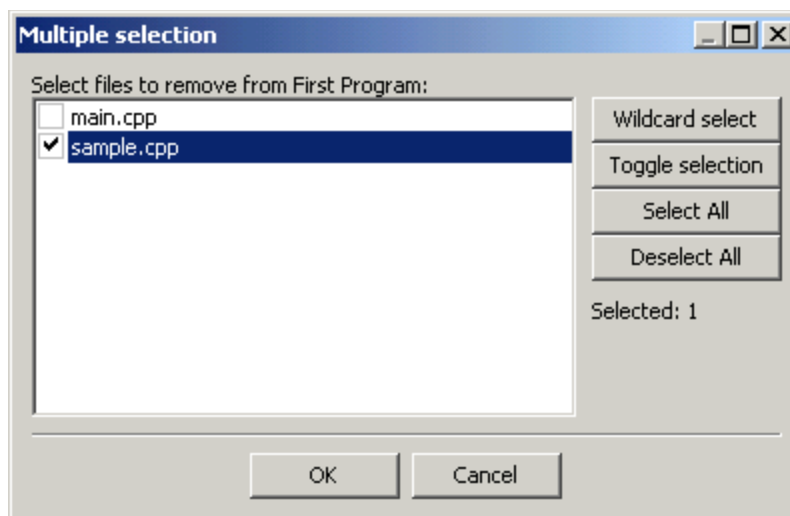
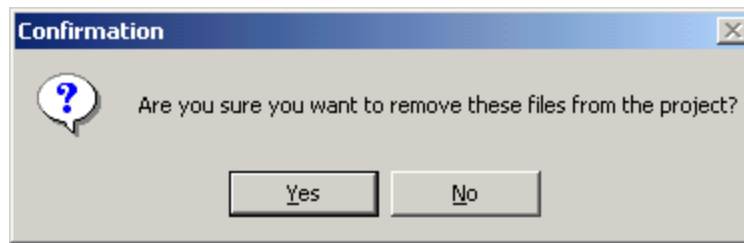The Sources now has sample.cpp as a source file in addition to the main.cpp file.

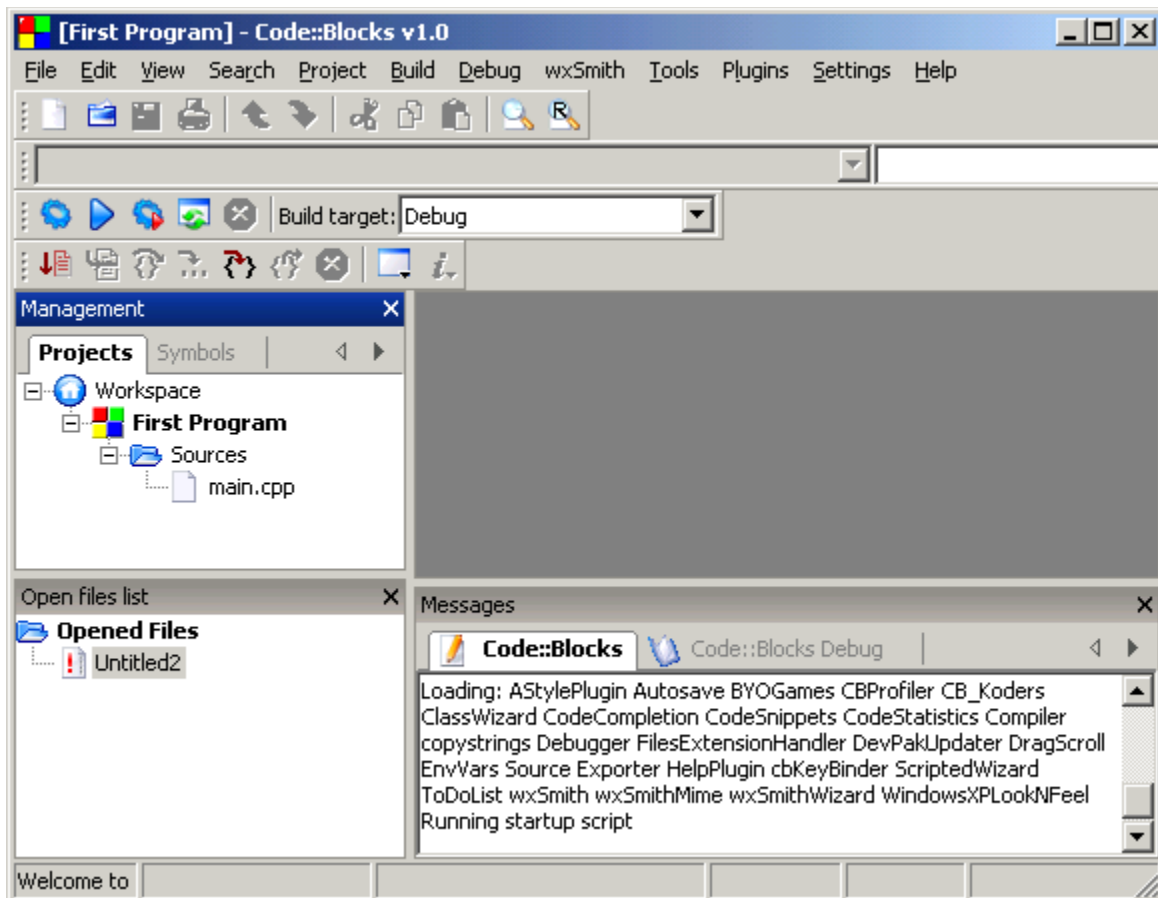Since the sample.cpp is not needed for your project, please remove it.
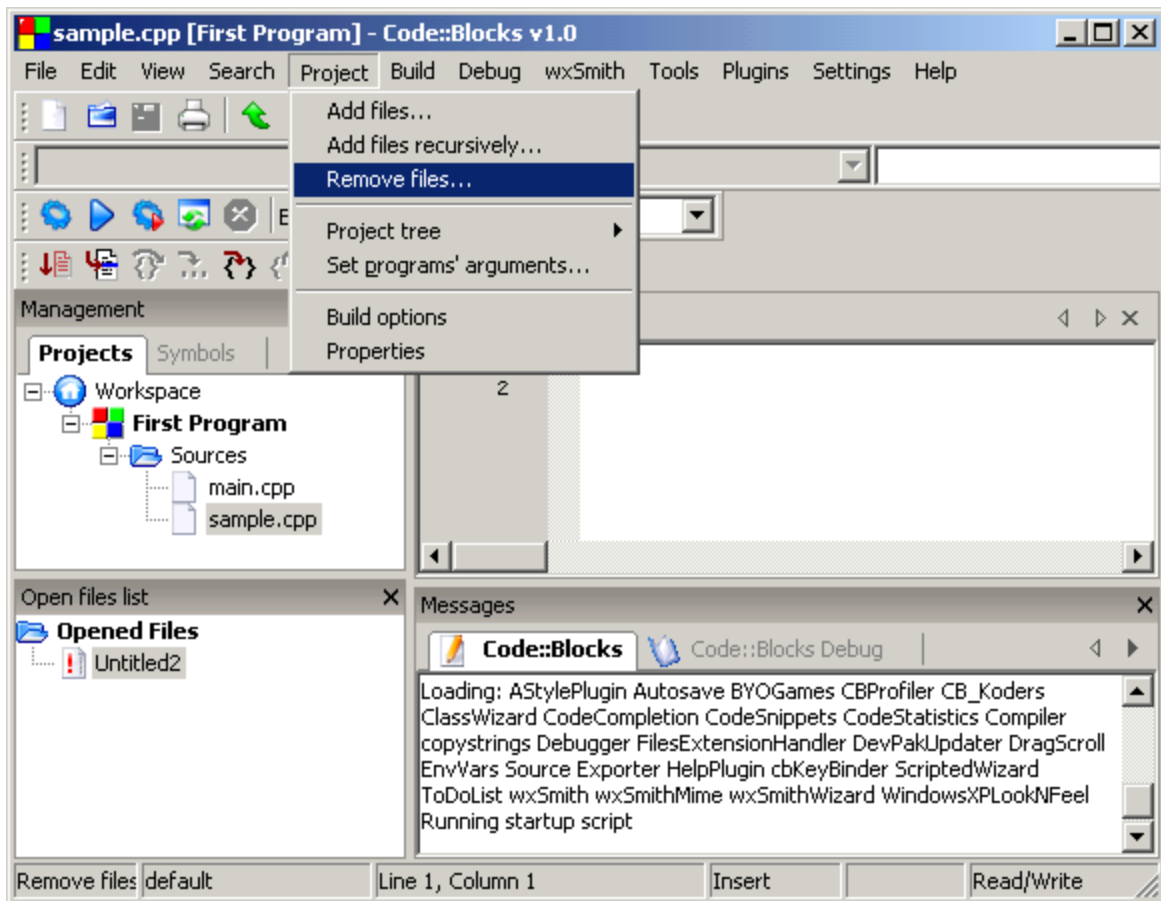
From the Project menu select, Remove files.

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.
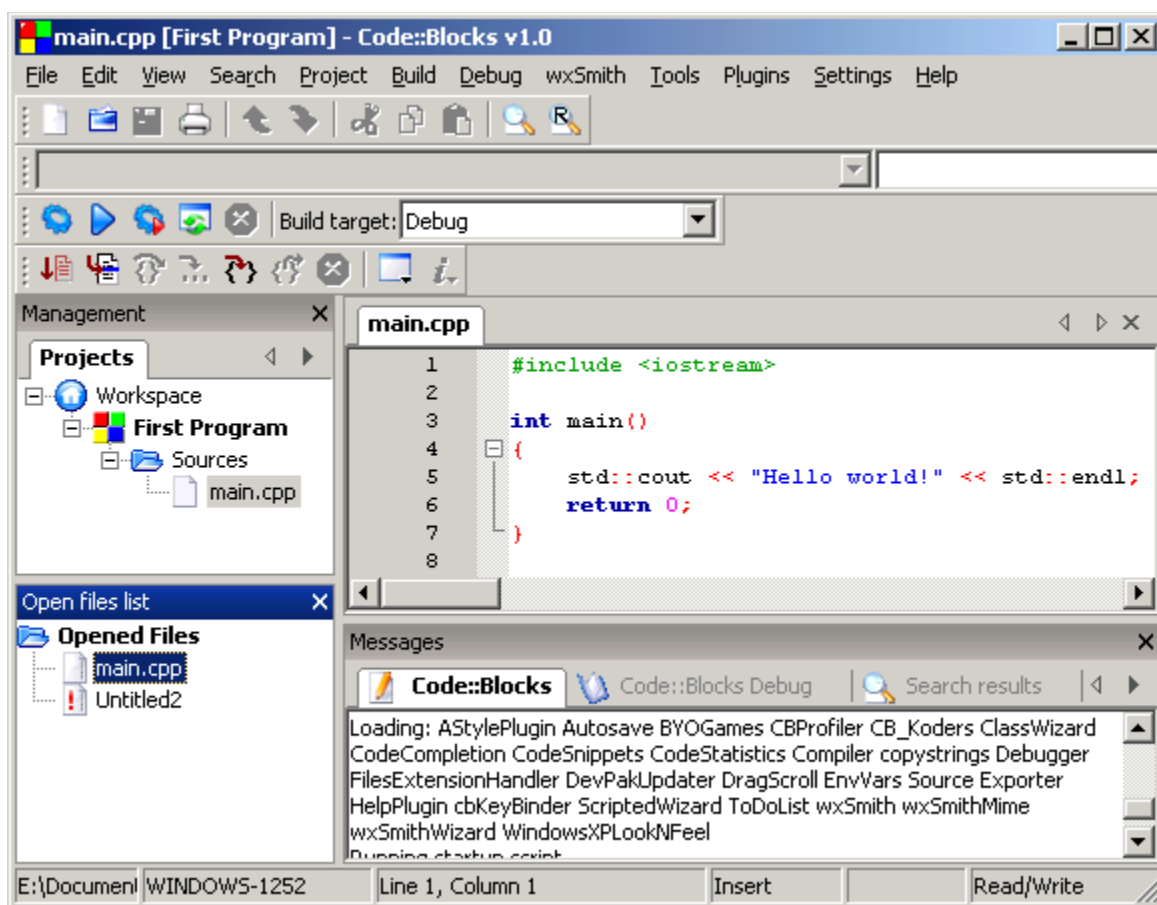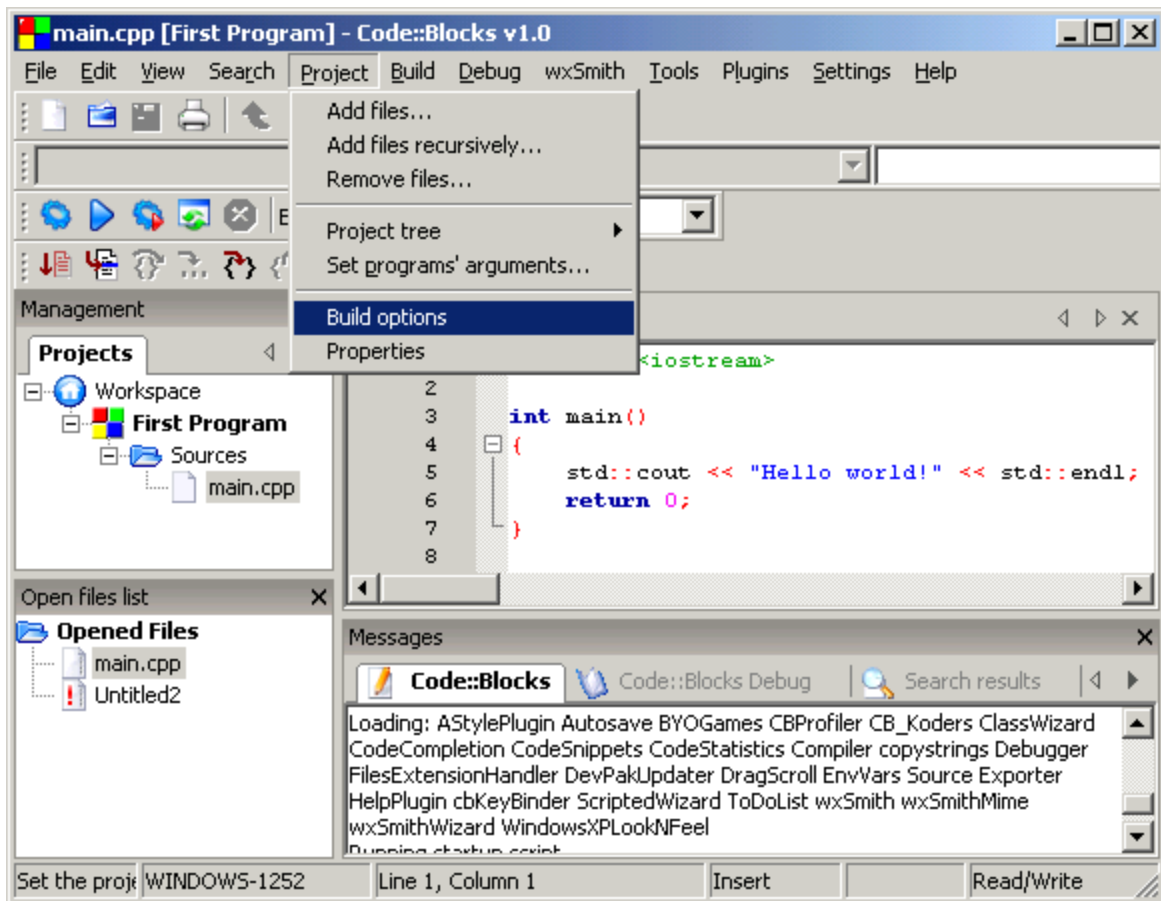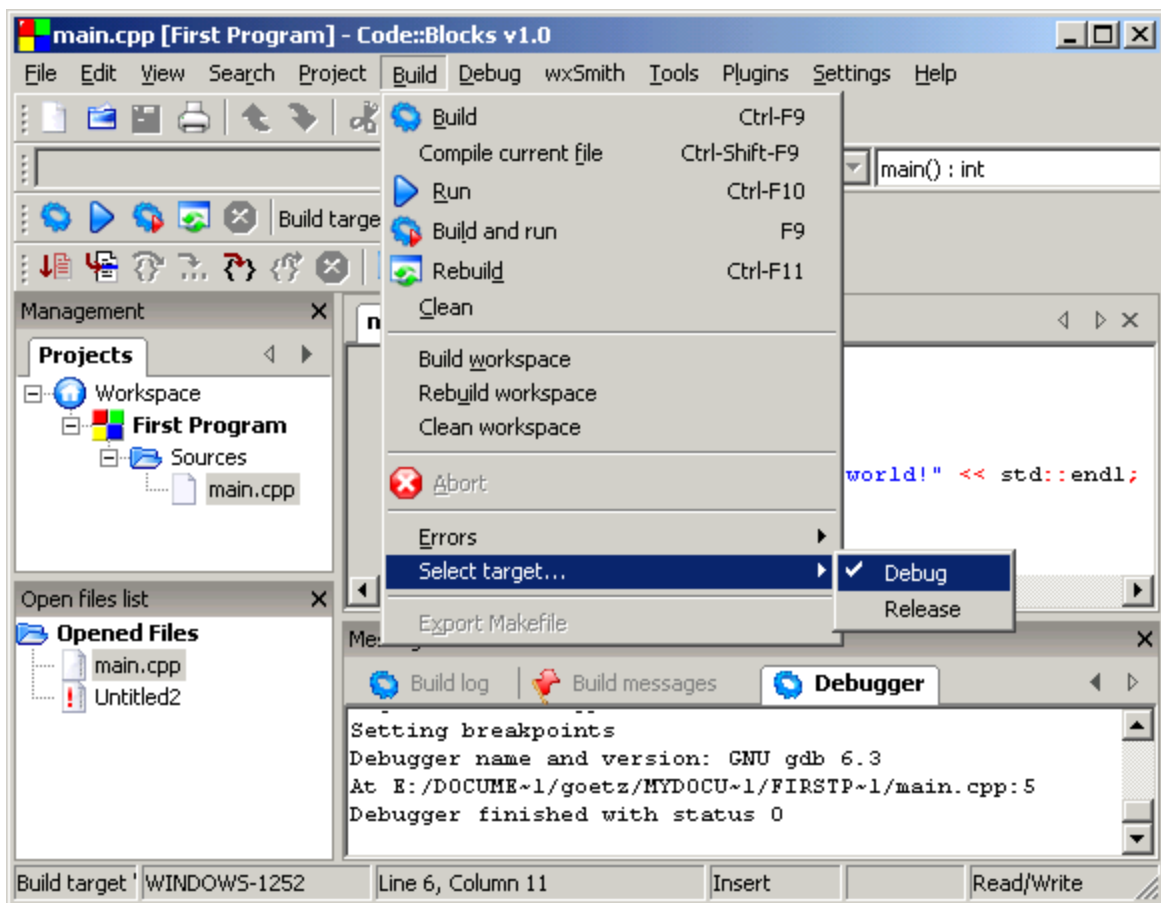
In order to check that Debug is running, you can use the **Project** pull-down menu and click on **Build Options.**
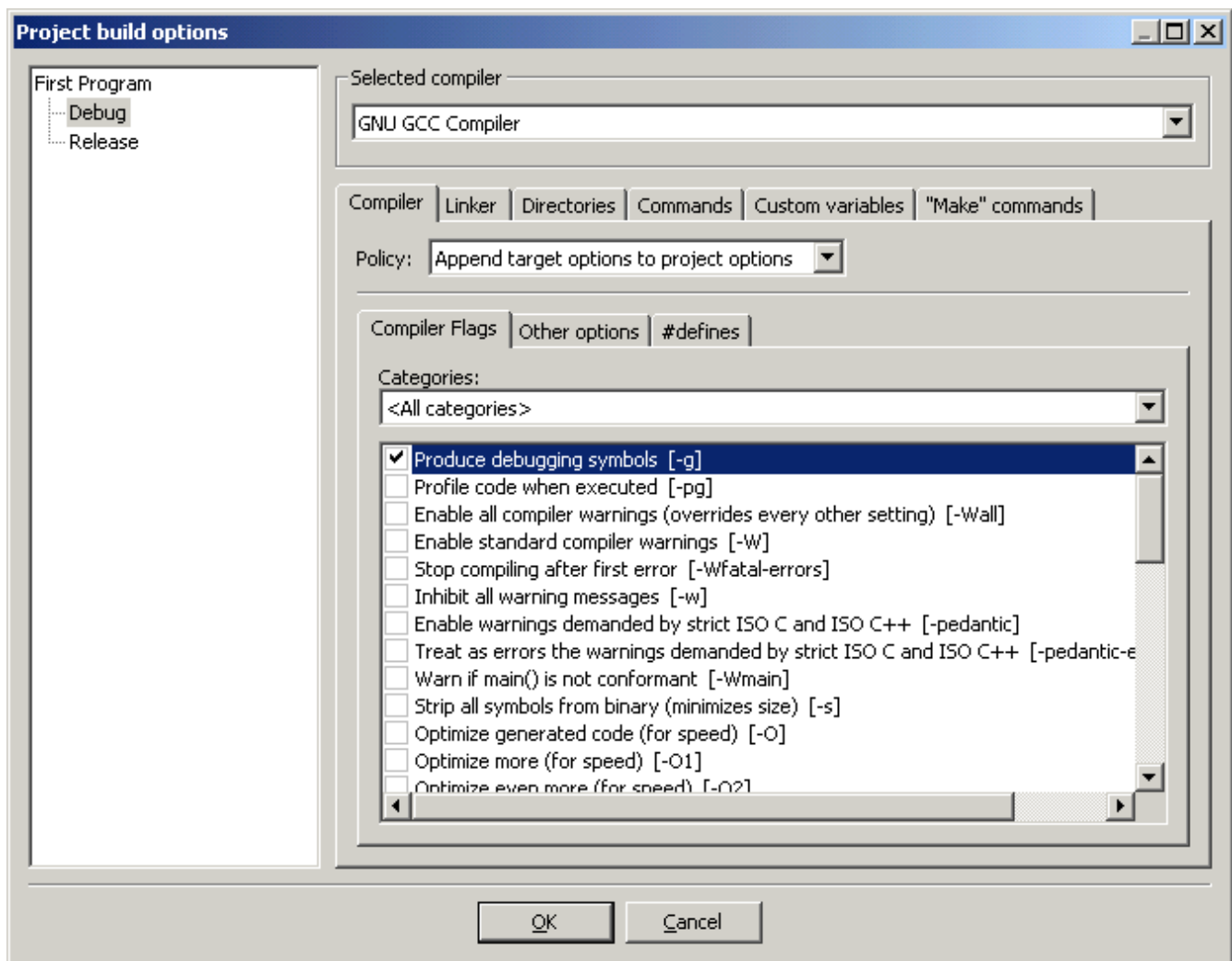
When this is done, the **Project Build options** window will come up. Make sure that the **Produce debugging symbols [-g]** is checked.

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.
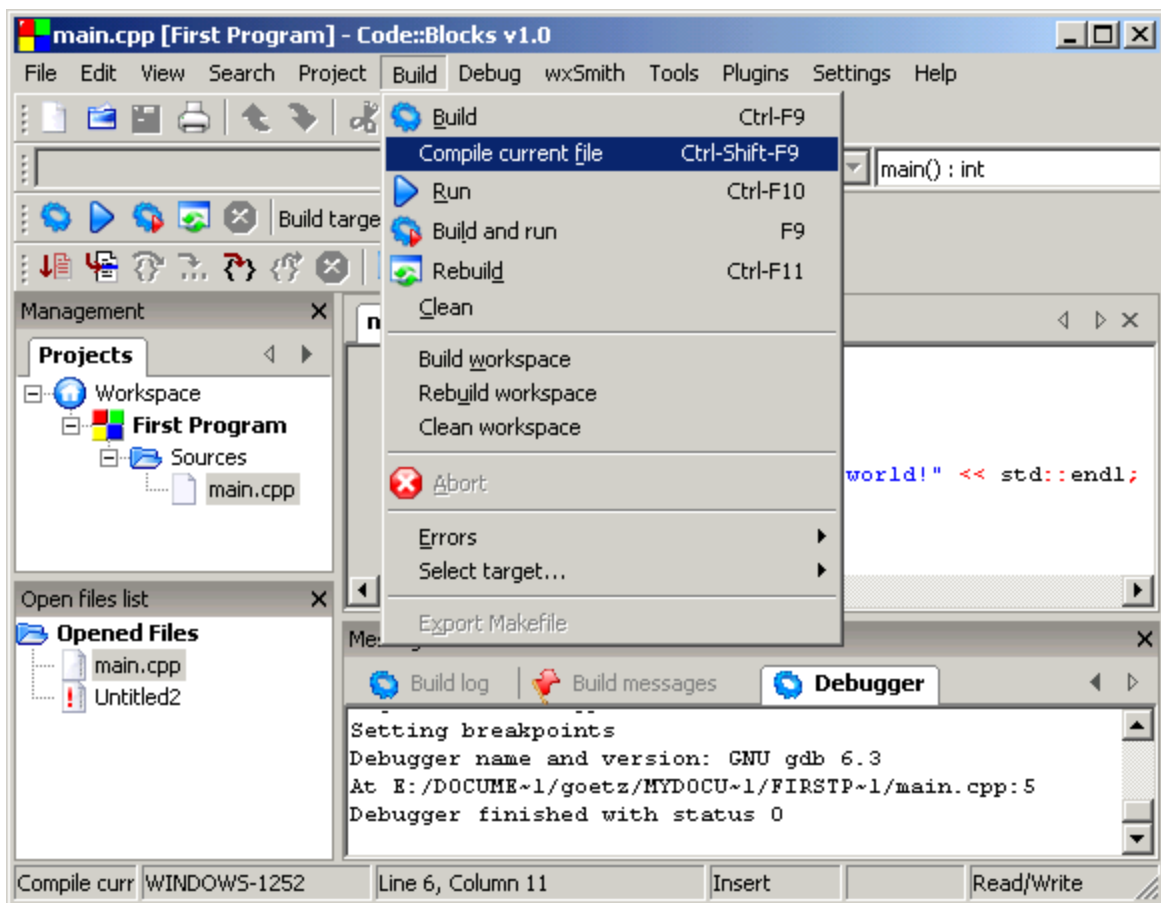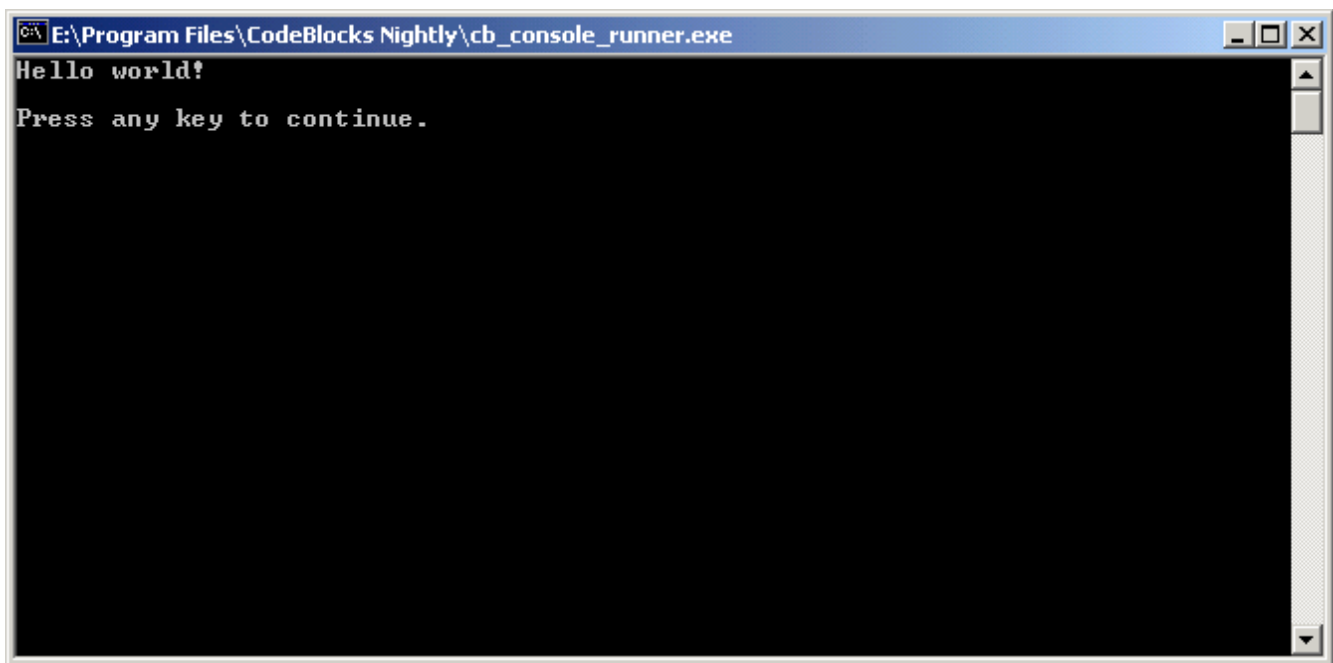
Compile your file from the Build pull-down menu by clicking on **Compile current file** (Ctrl-Shift-F9).
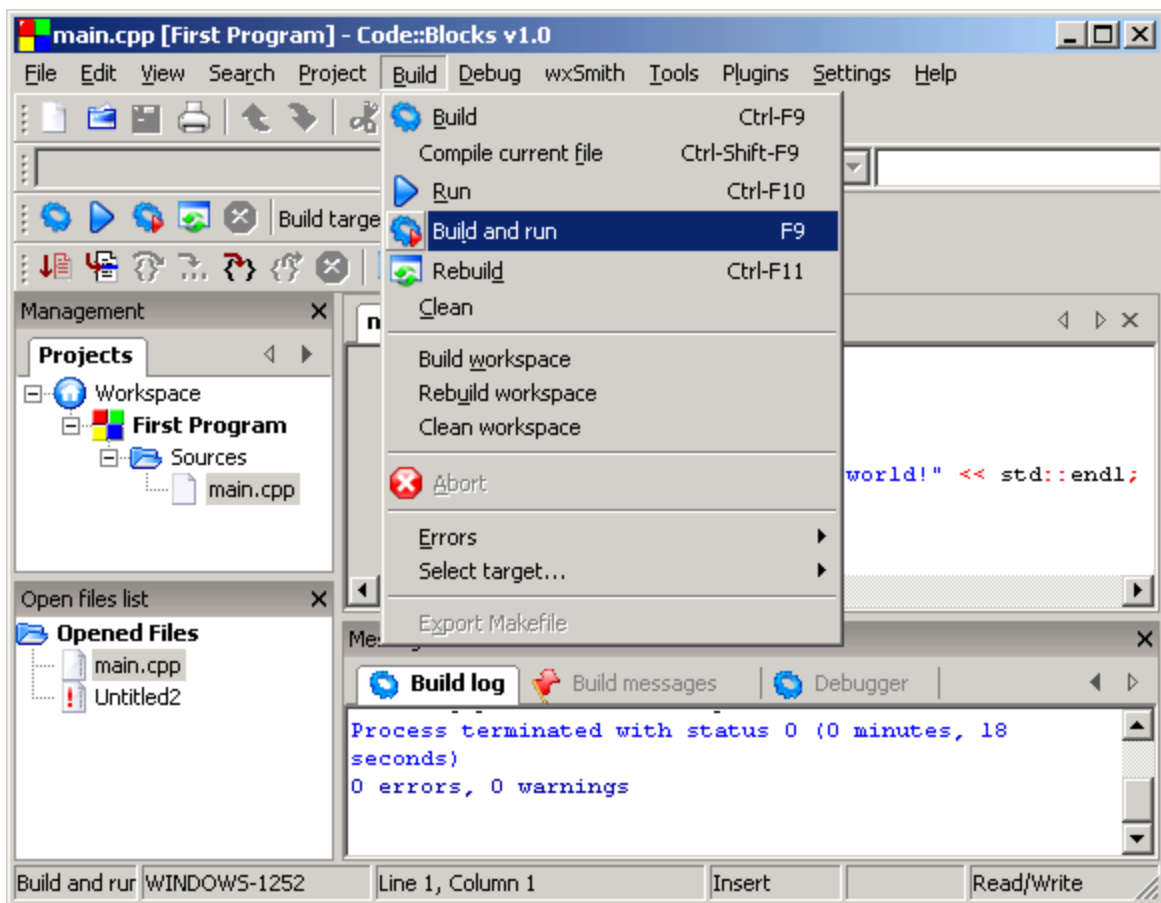
Test the project from the Build Pull-down menu, by clicking on Build and Run. This step will **build** an executable file for you. A project build will take the compiled versions of your source files and combine them into one program.
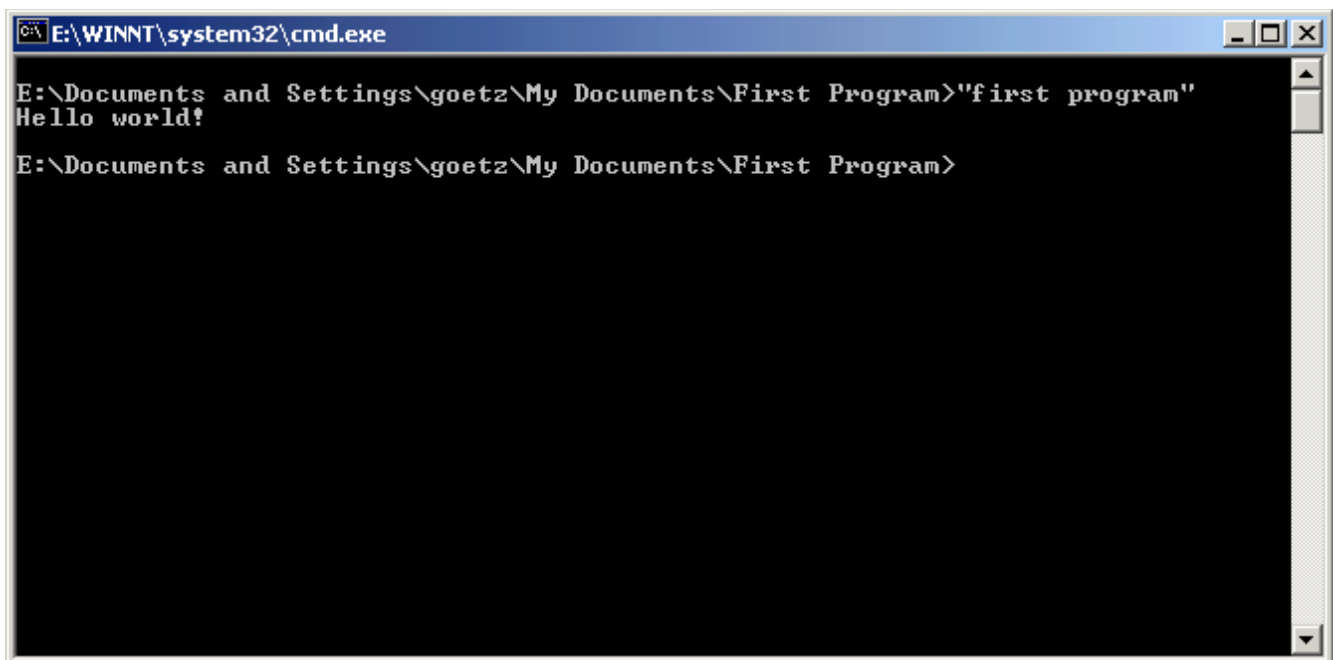
You are able to press F9, which is a keyboard shortcut that will build your project and run it at the same time. As you gain more experience with the system, it will be easier to just press F9 to Build & Run your program. The Message window will indicate if there are any errors during a compile or build 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**.

```
main.cpp [First Program] - Code::Blocks v1.0

File  Edit  View  Search  Project  Build  Debug  wxSmith  Tools  Plugins  Settings  Help

   New                              ▶                                    main() : int

   📁 Open                    Ctrl-O
      Open default workspace
      Recent projects                ▶
      Recent files                   ▶                                         ◁ ▷ ✕

      Import project                 ▶        #include <iostream>

   💾 Save                    Ctrl-S          int main()
   📝 Save as...
      Save all files          Ctrl-Shift-S        std::cout << "Hello world!" << std::endl;
                                                   return 0;
      Save project
      Save project as...
      Save project as user-template
      Save all projects                                                              ▶

      Save workspace                                                            ✕
      Save workspace as...
      Close workspace              Build messages    🔵 Debugger           ◁ ▷
                                   kpoints                                         ▲
   ❌ Close file              Ctrl-W     e and version: GNU gdb 6.3
      Close all files         Ctrl-Shift-W  ~1/goetz/MYDOCU~1/FIRSTP~1/main.cpp:5
      Close project                         ished with status 0
      Close all projects                 11          Insert              Read/Write
```

Now you can select to save the project:
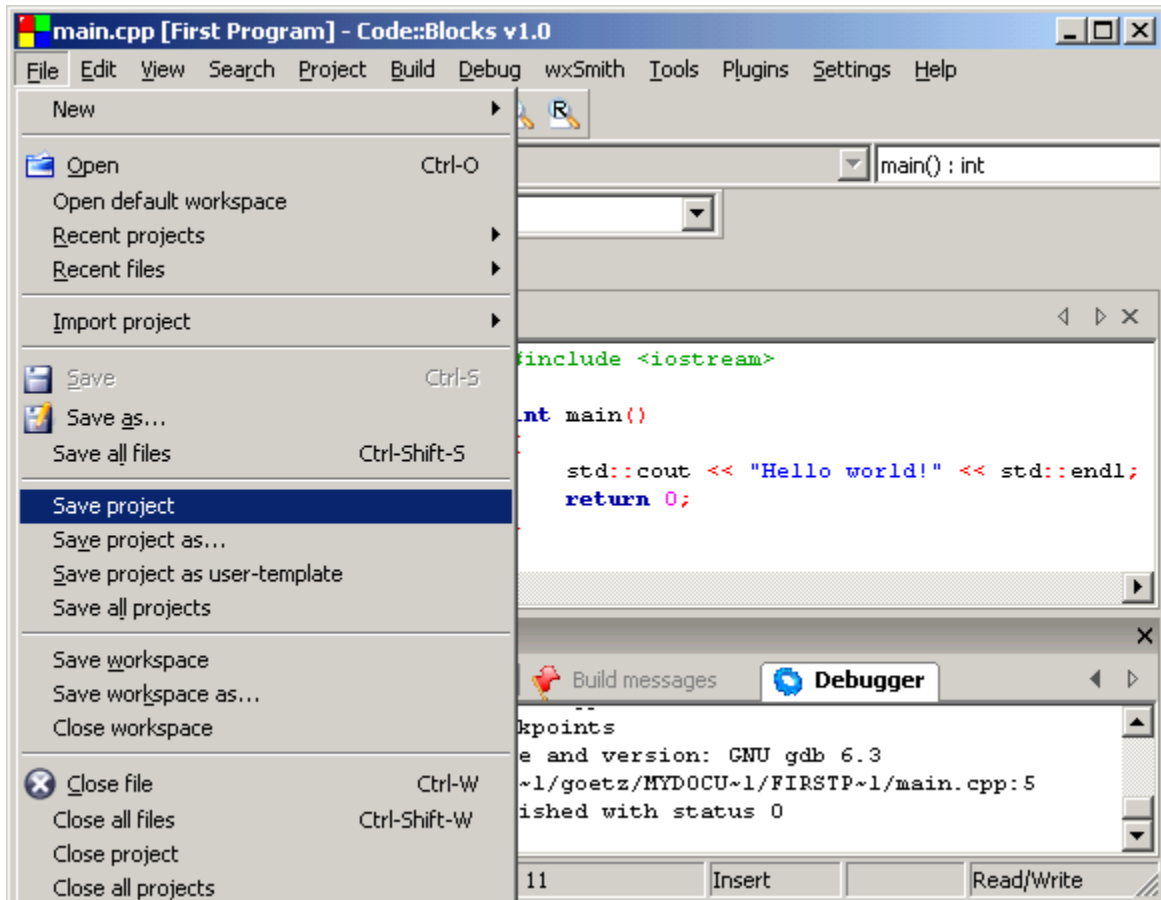
When you exit the program, you may be asked to save the **Workspace** and the **Layout**. The Layout refers to the placement of various windows that you may have positioned. Generally you would select to Save the Layout (unless you know you really do not want it saved). The Workspace refers to the projects 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.
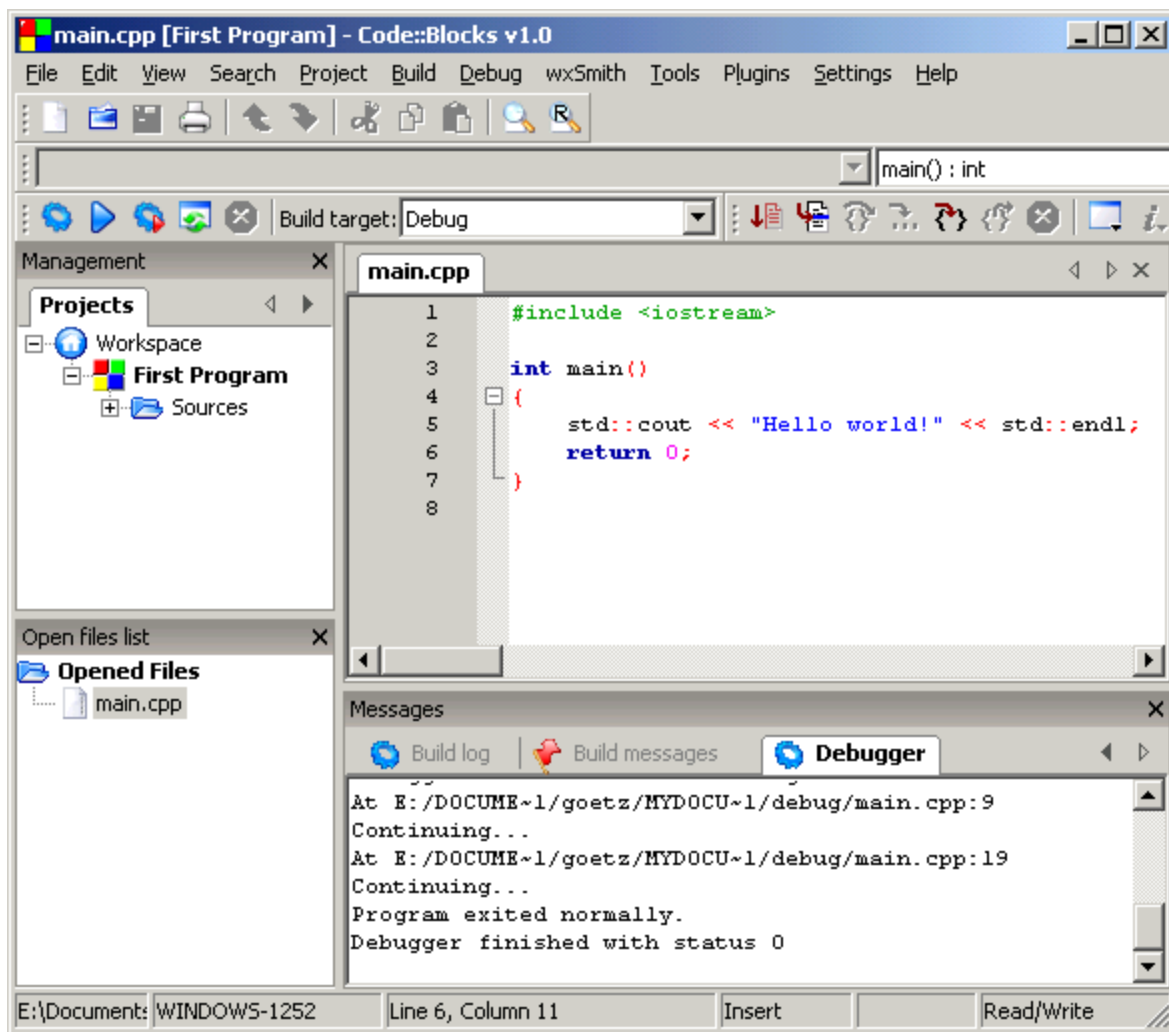
To open a project
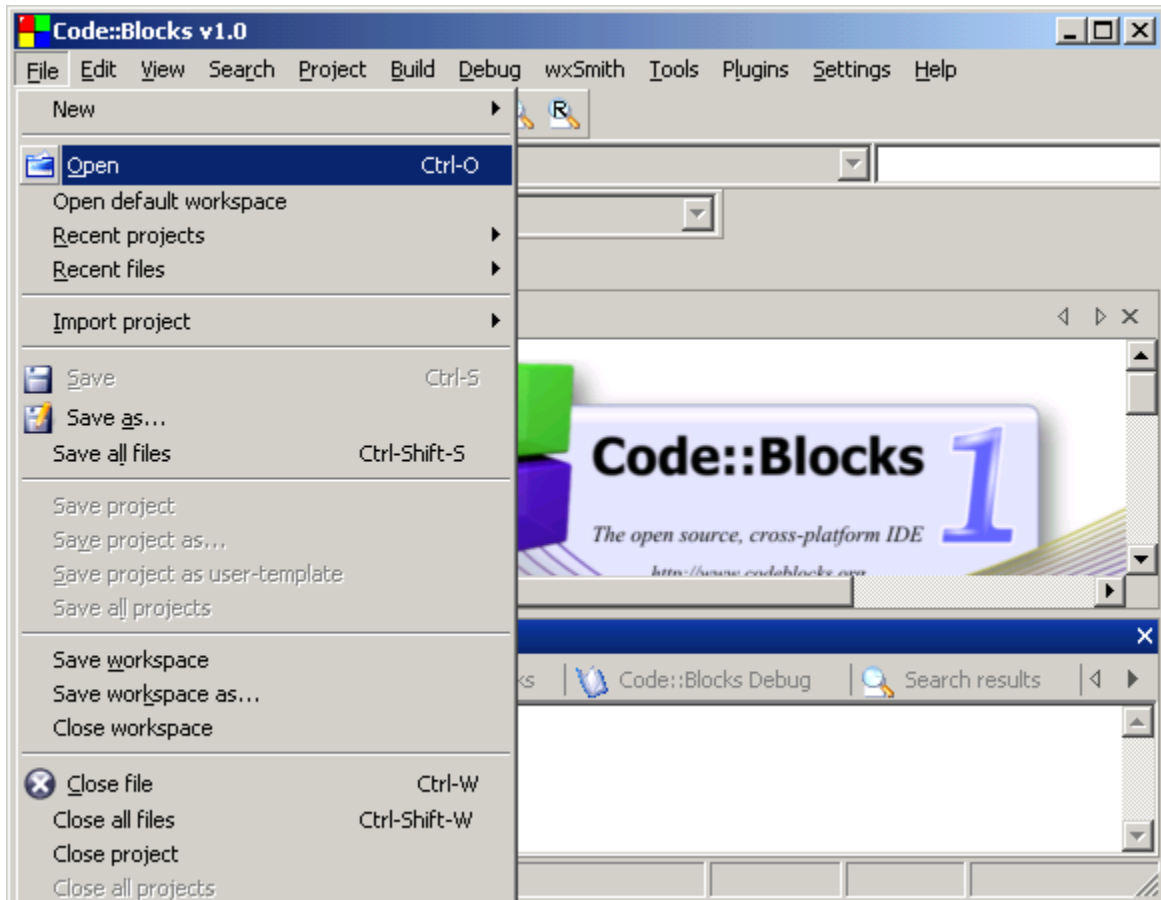
From the File menu select Open.
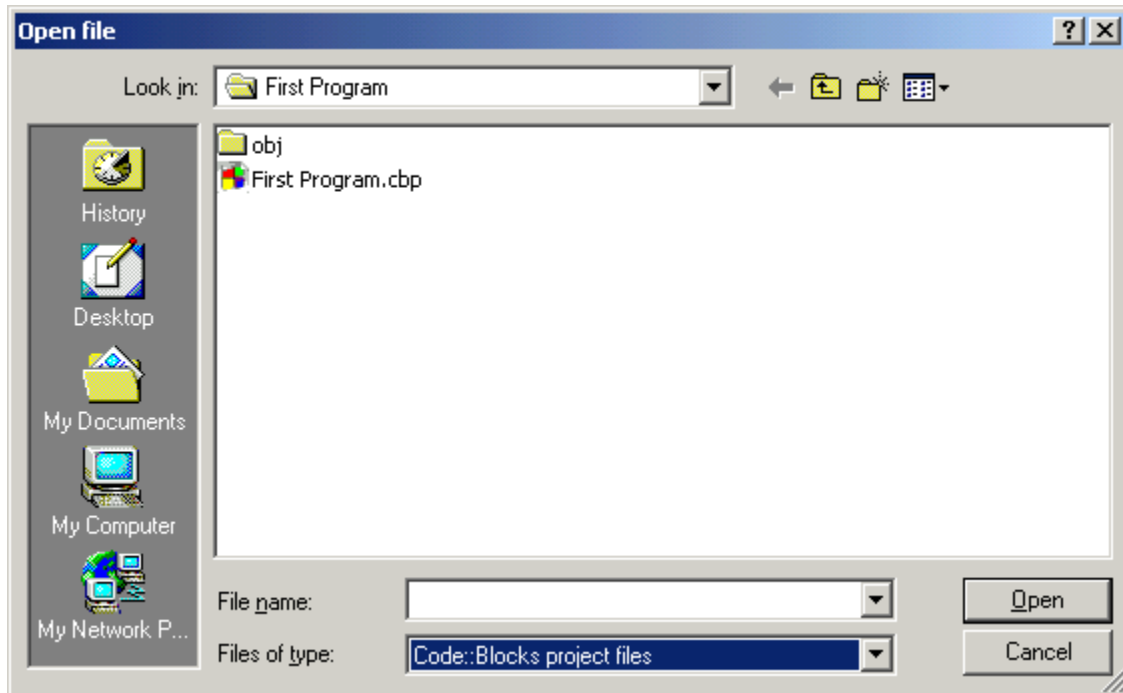
From the **Files of type:** in the window, select "Code::Blocks project files" and then select the .cbp file pertaining to your program.
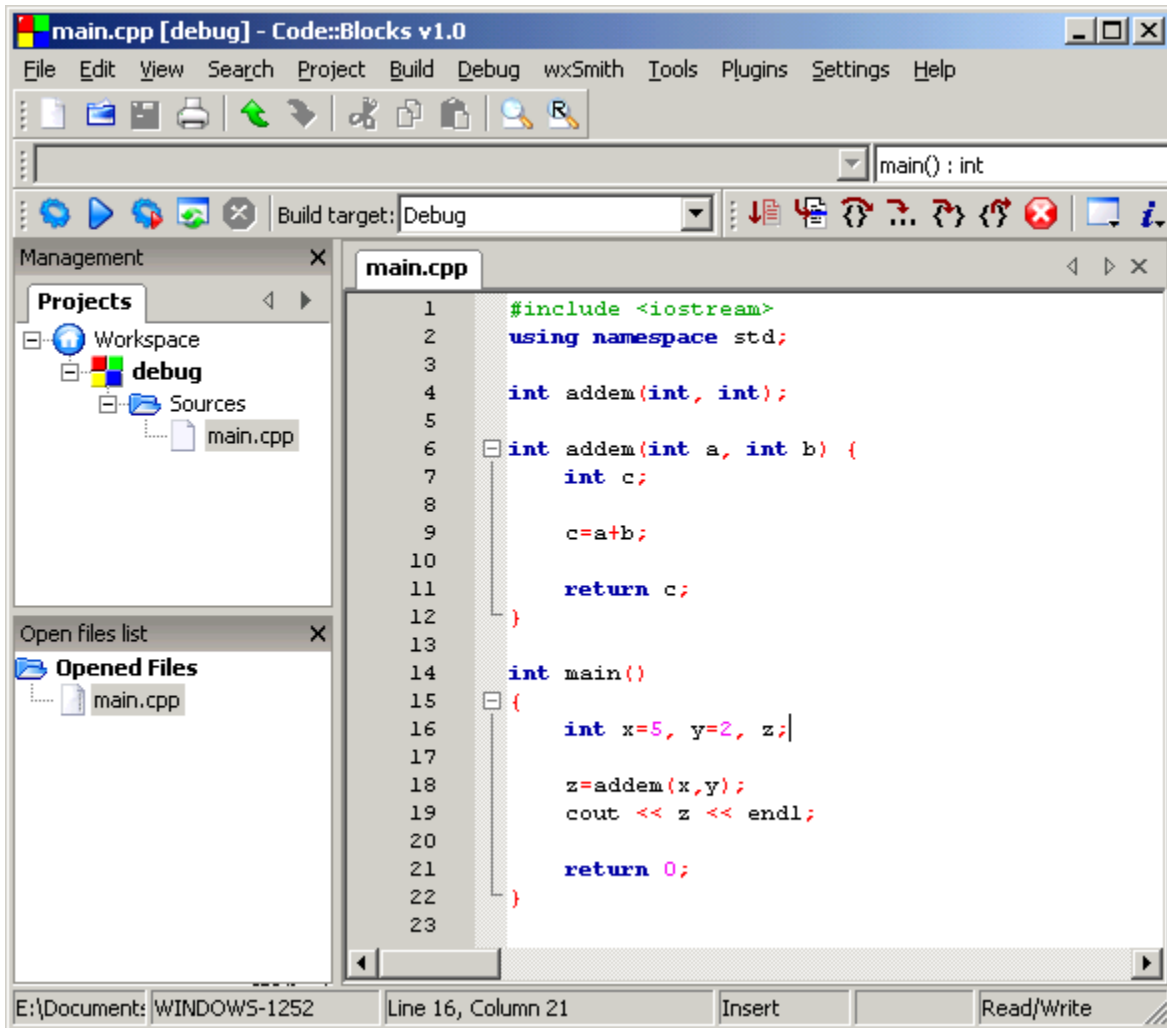
Press Open when done.

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.
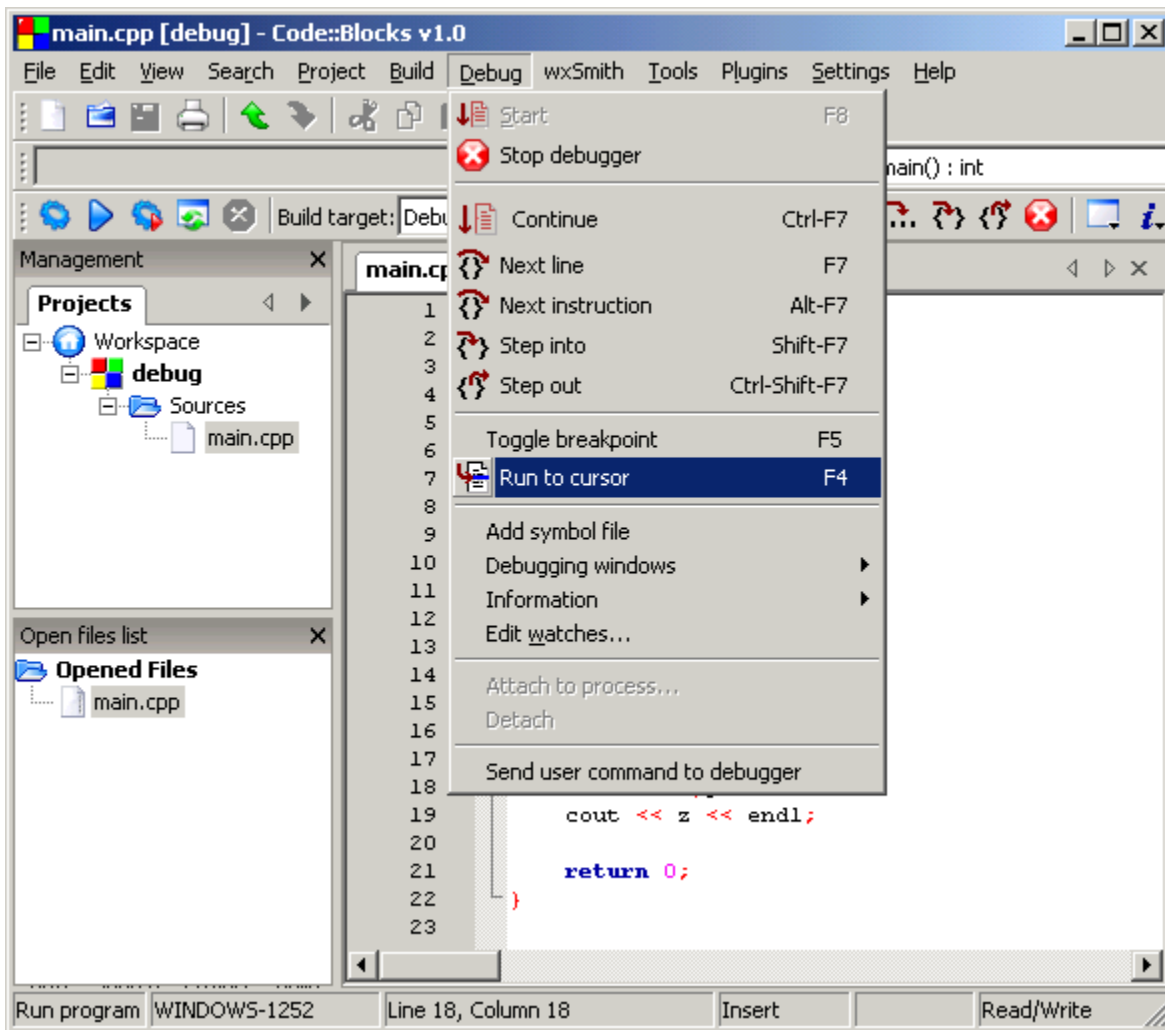
Debugging a Program

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.

```cpp
#include <iostream>
using namespace std;

int addem(int, int);

int addem(int a, int b) {
    int c;

    c=a+b;

    return c;
}

int main()
{
    int x=5, y=2, z;

    z=addem(x,y);
    cout << z << endl;

    return 0;
}
```

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).
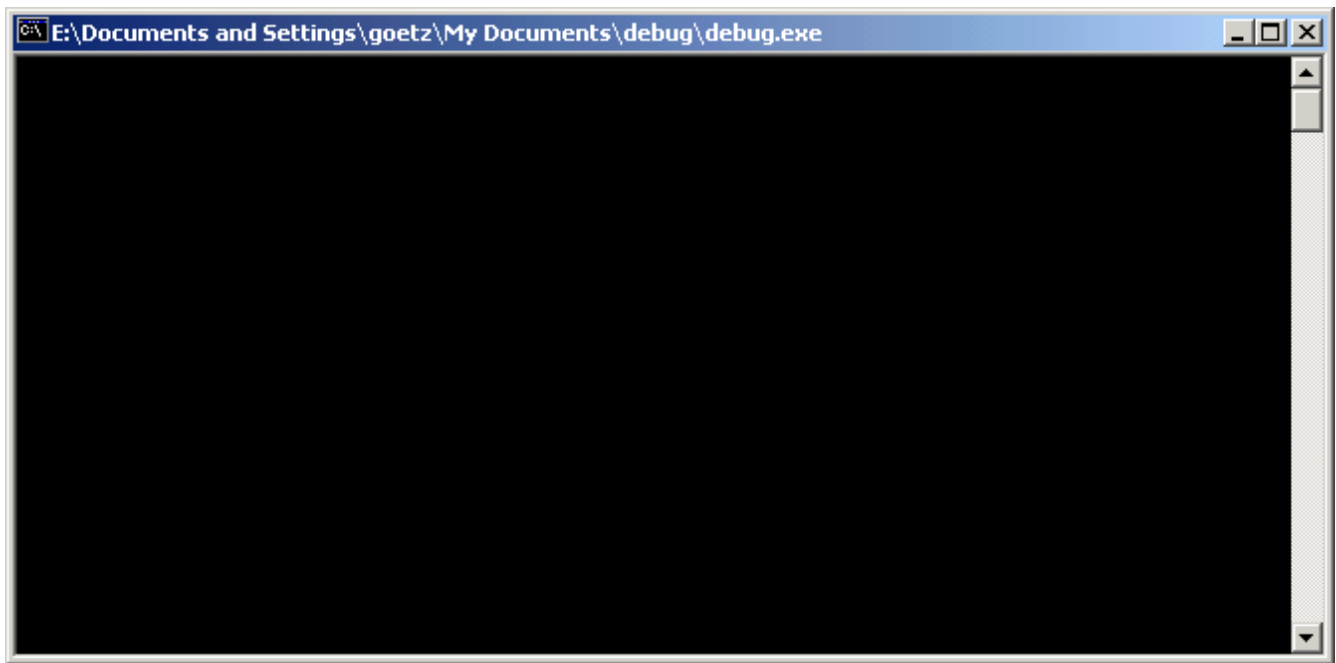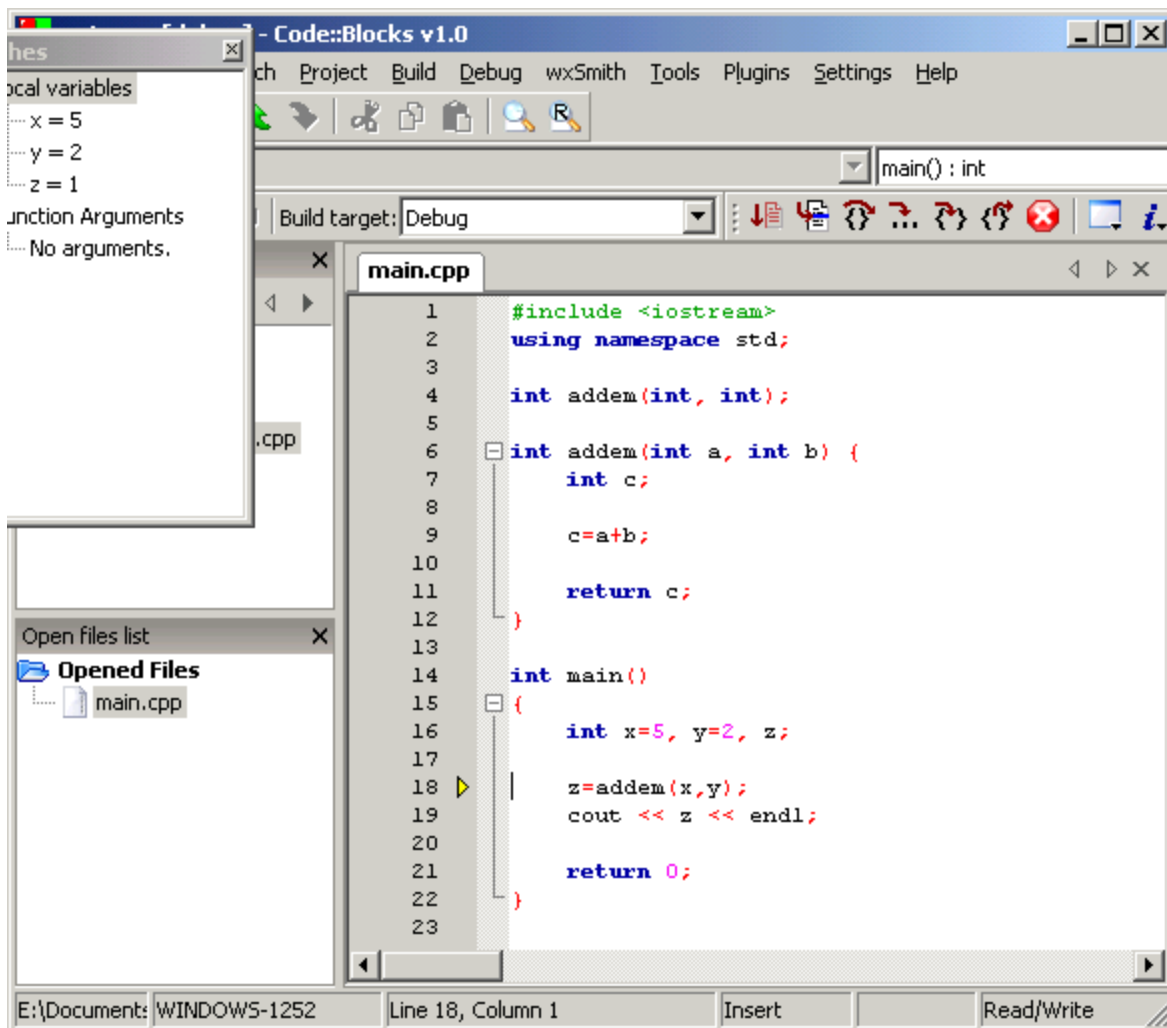
The program will generate a blank window. It is blank, since that program has yet to execute any line that displays something.
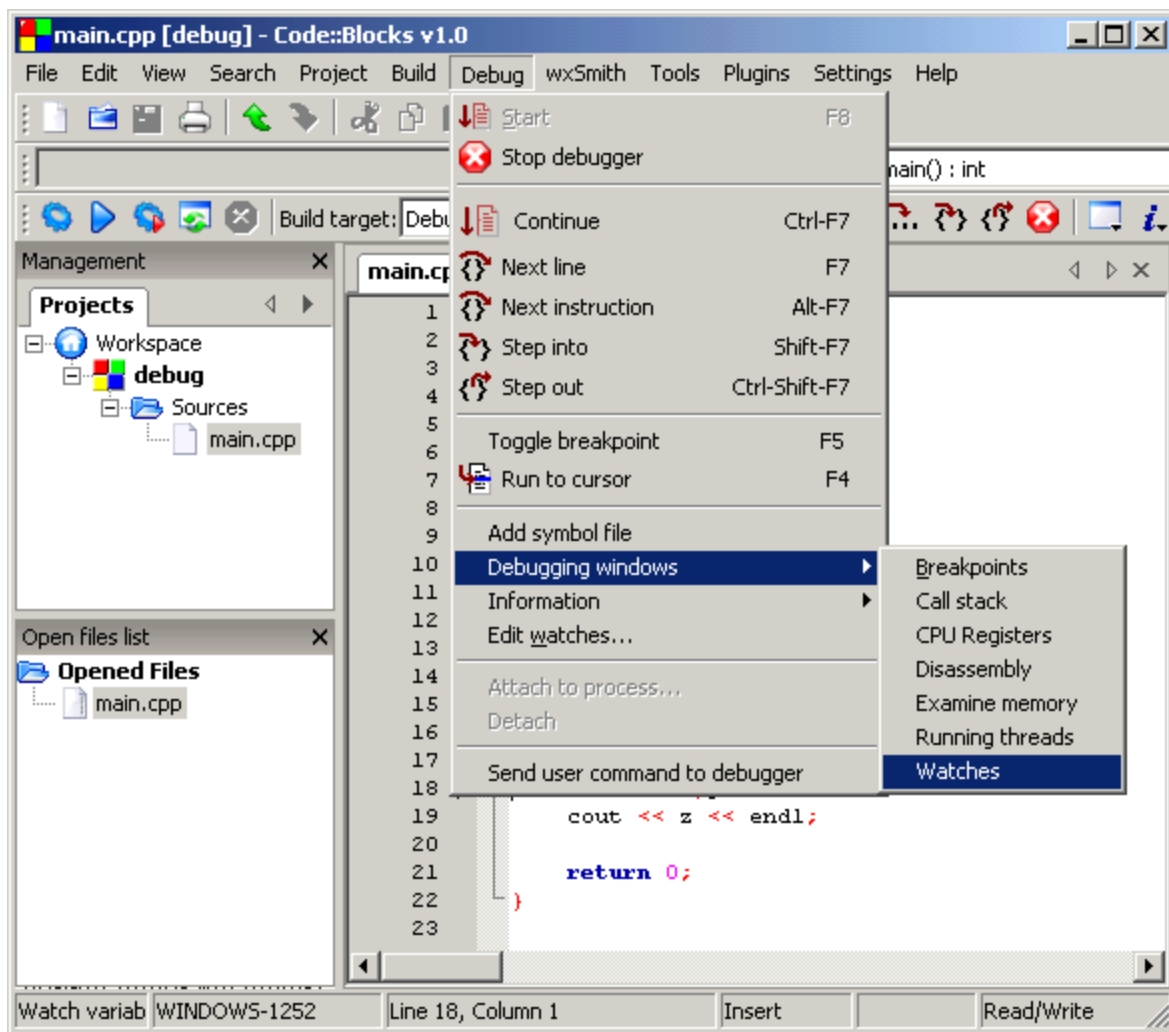
To watch certain variables during the execution of the program, you should open the Watches window. This will show you the variables in your code. This is accomplished by going to the **Debug** pull-down menu and clicking on **Debugging Windows** and then **Watches**.

Watches

☐ Local variables
  x = 5
  y = 2
  z = 1
☐ Function Arguments
  No arguments.
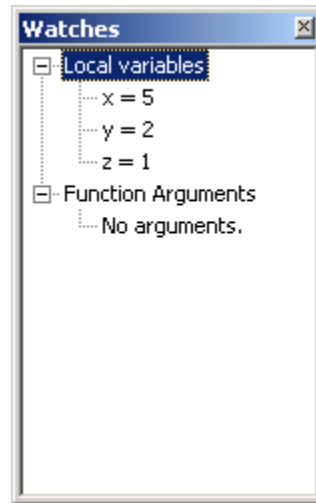
These are the watches that the debugger is displaying. Notice that x & y have the correct values. Variable z has not been assigned it's value on line 18 yet. The current value is a random value.

hes ✕

ocal variables
   x = 5
   y = 2
   z = 1
  unction Arguments
   No arguments.

Code::Blocks v1.0

ch Project Build Debug wxSmith Tools Plugins Settings Help

main() : int

Build target: Debug

.cpp

main.cpp

```cpp
1    #include <iostream>
2    using namespace std;
3
4    int addem(int, int);
5
6    int addem(int a, int b) {
7         int c;
8
9         c=a+b;
10
11        return c;
12    }
13
14    int main()
15    {
16        int x=5, y=2, z;
17
18        z=addem(x,y);
19        cout << z << endl;
20
21        return 0;
22    }
23
```

Open files list ✕

Opened Files
  main.cpp

E:\Document: | WINDOWS-1252 | Line 18, Column 1 | Insert | Read/Write

Line 18 has a yellow marker on the left side. This indicates that the program has paused on that line, which is the breakpoint.
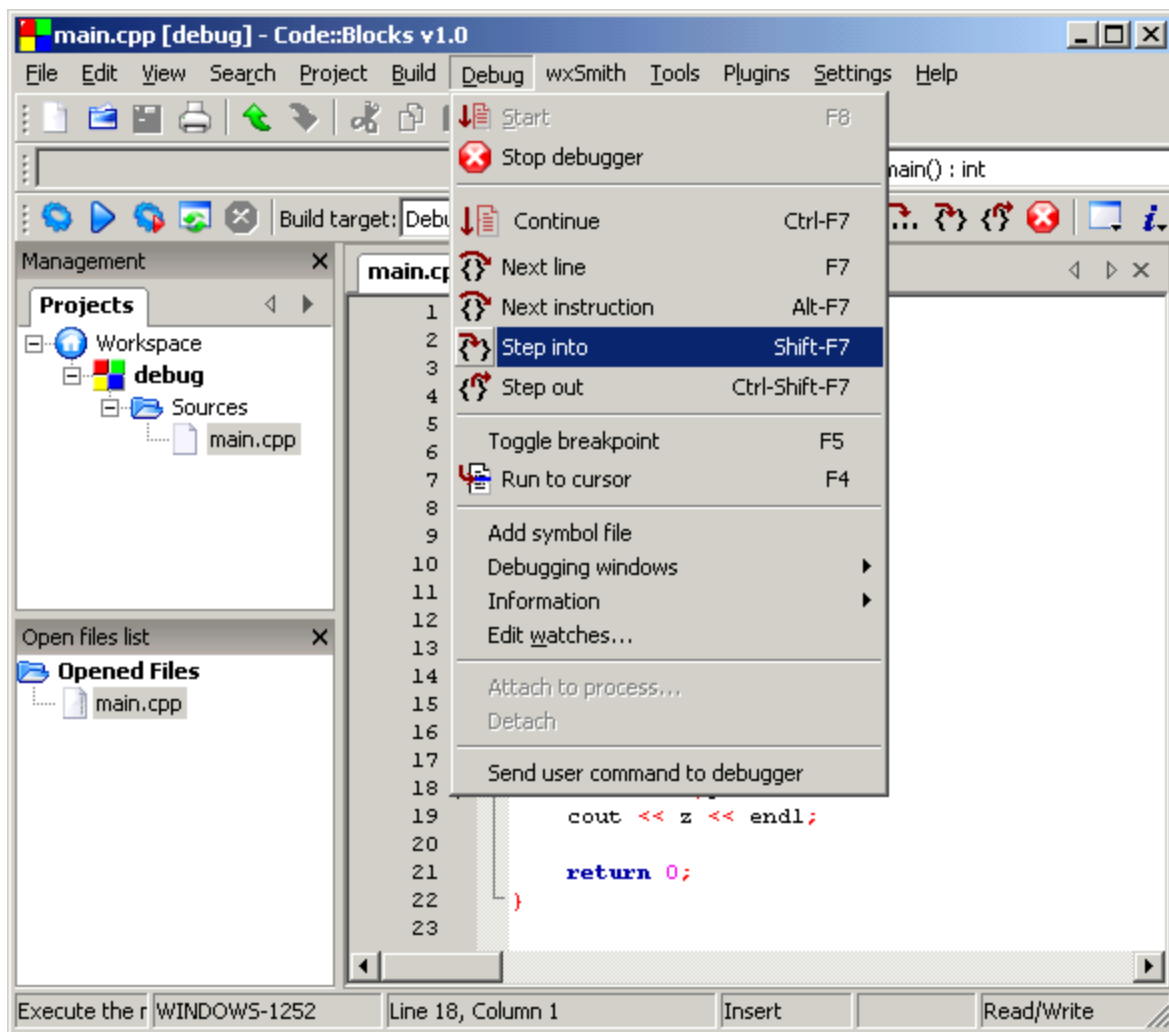
To determine how your program will function when calling functions such as:
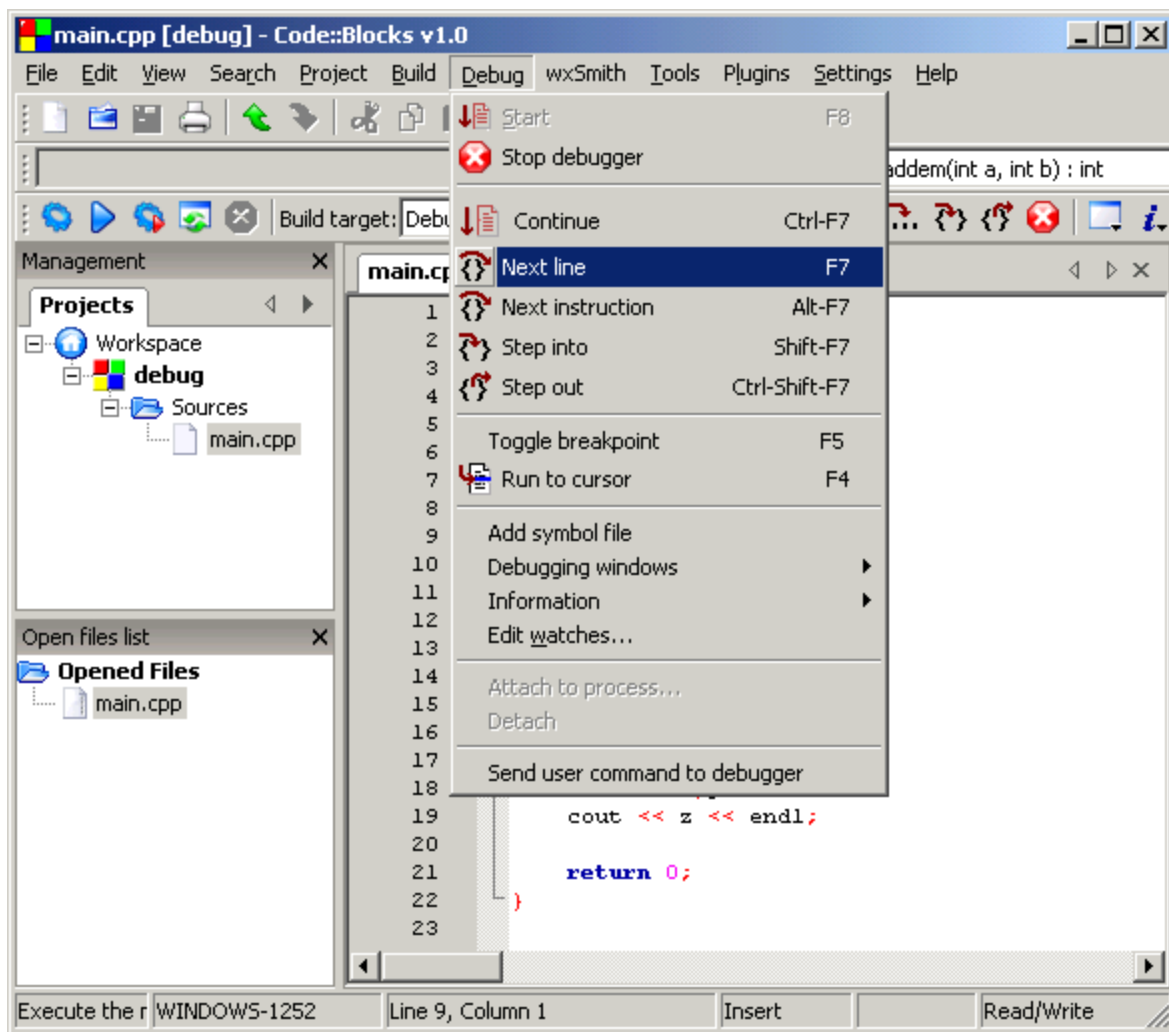
```
z=addem(x,y);
```

Step info (Shift-F7) can be selected from the Debug pull-down menu.
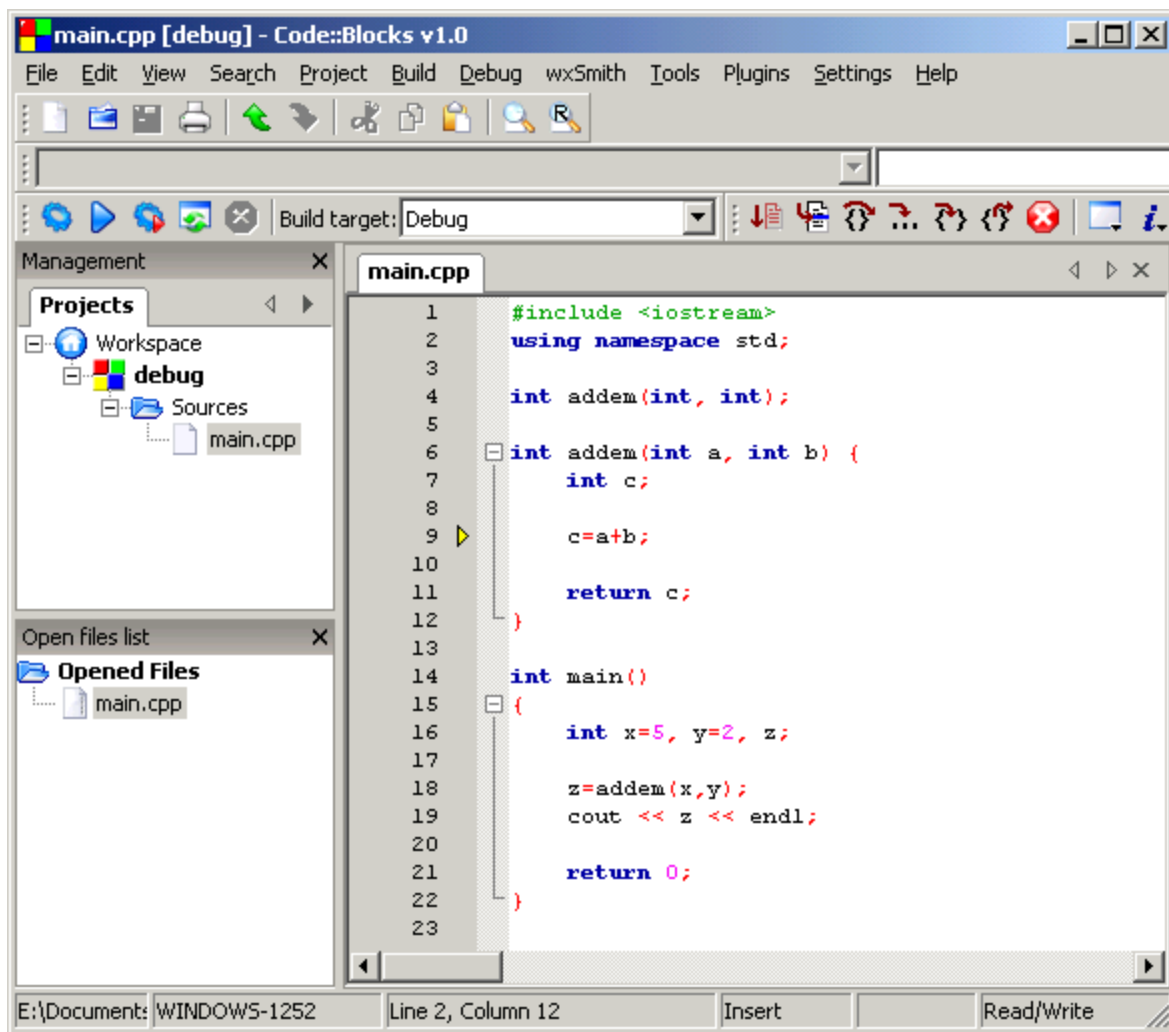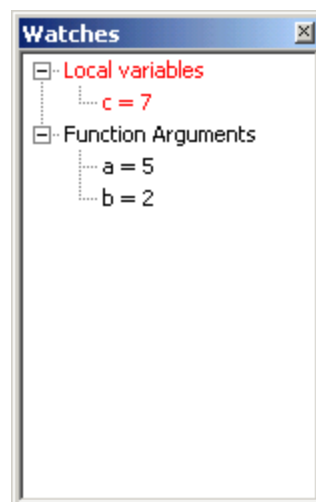
The next step is line 9.
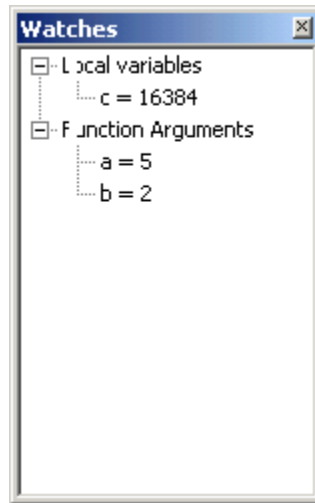
The arguments `a` and `b` are shown in the Watches window. Notice that the local variable `c`, which has not been set yet, has a random value.
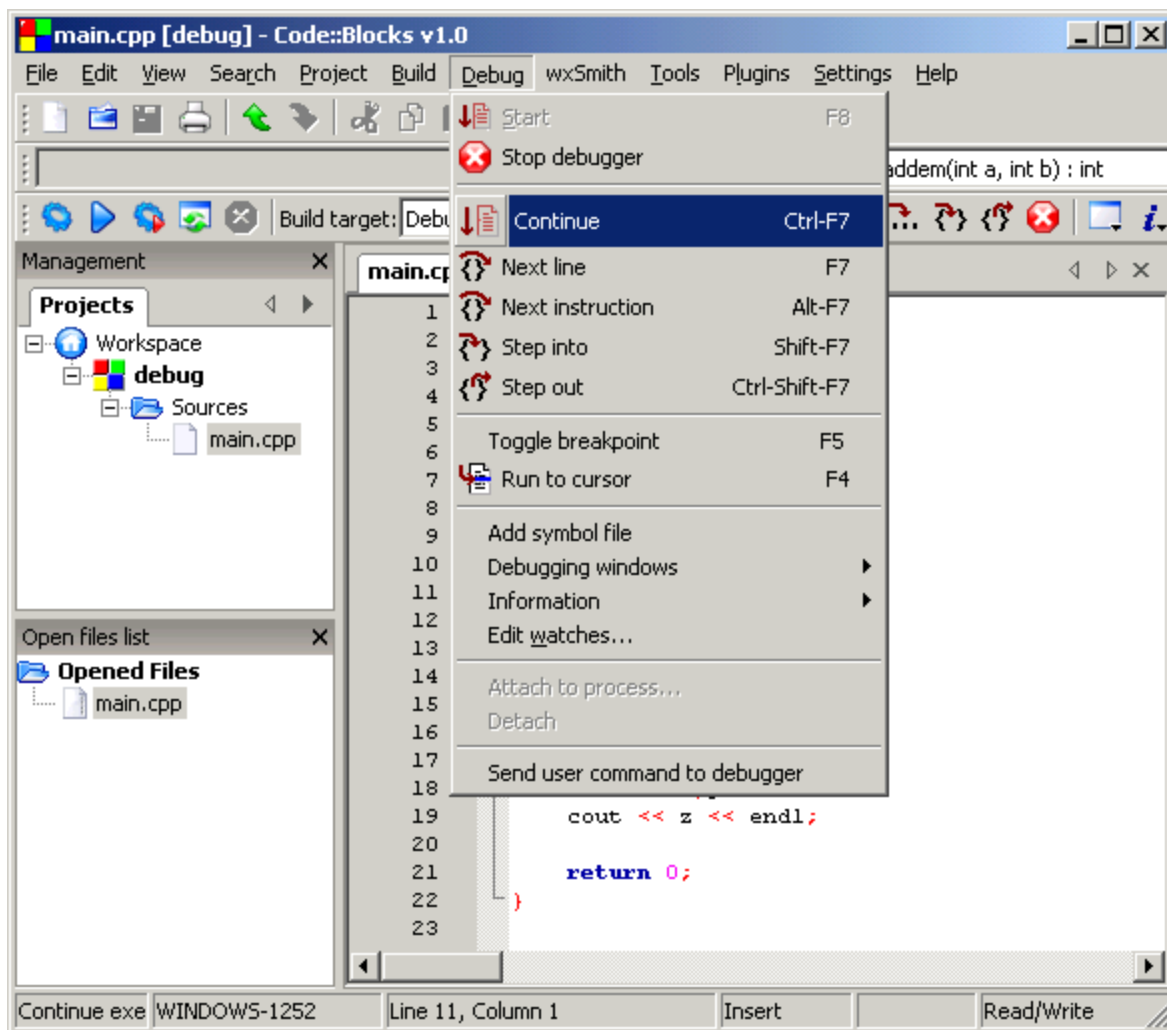
To proceed to the next line of code, select **Next line** from the Debug menu.

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**.
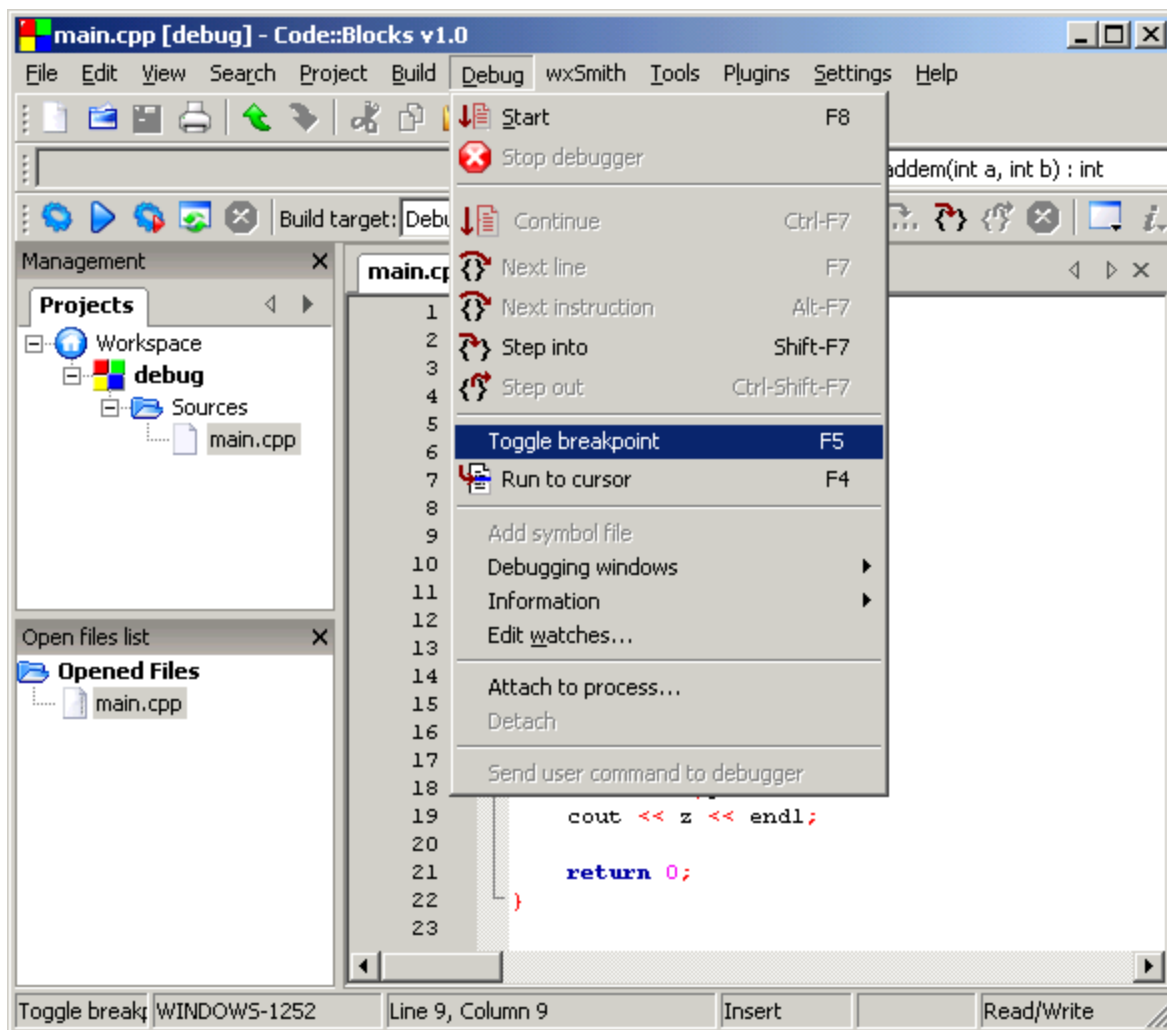
You can further define places in your program to pause and allow you to inspect the code. This is done by setting breakpoints in your code. You can have zero or more breakpoints in your code. When the debugger encounters a breakpoint, the program pauses and the debugger will allow you to inspect your code. **The breakpoint remains until you remove it. It can be Toggled with F5.**
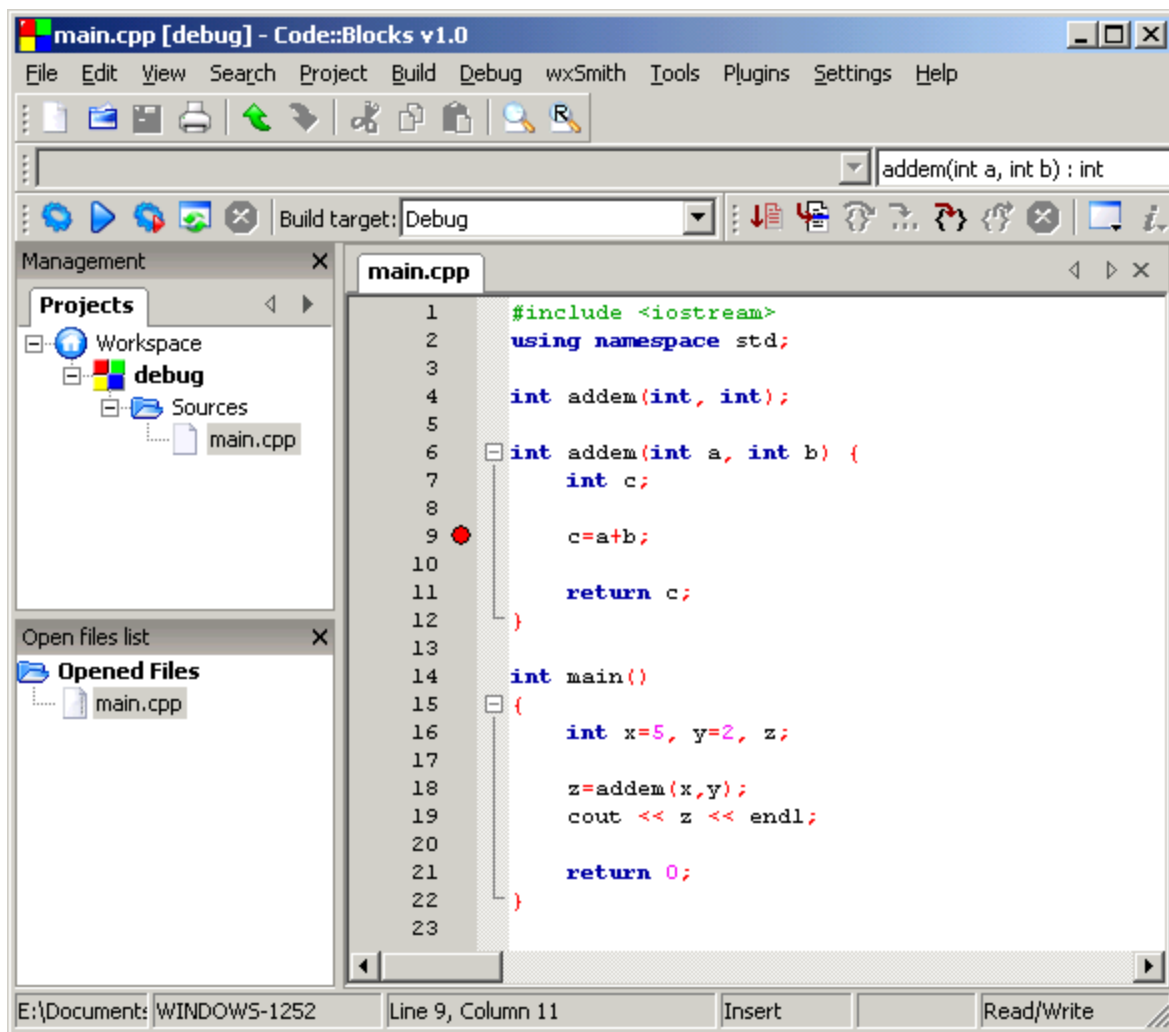
A breakpoint has been set at line 9. The red circle indicates that there is a breakpoint in the code.
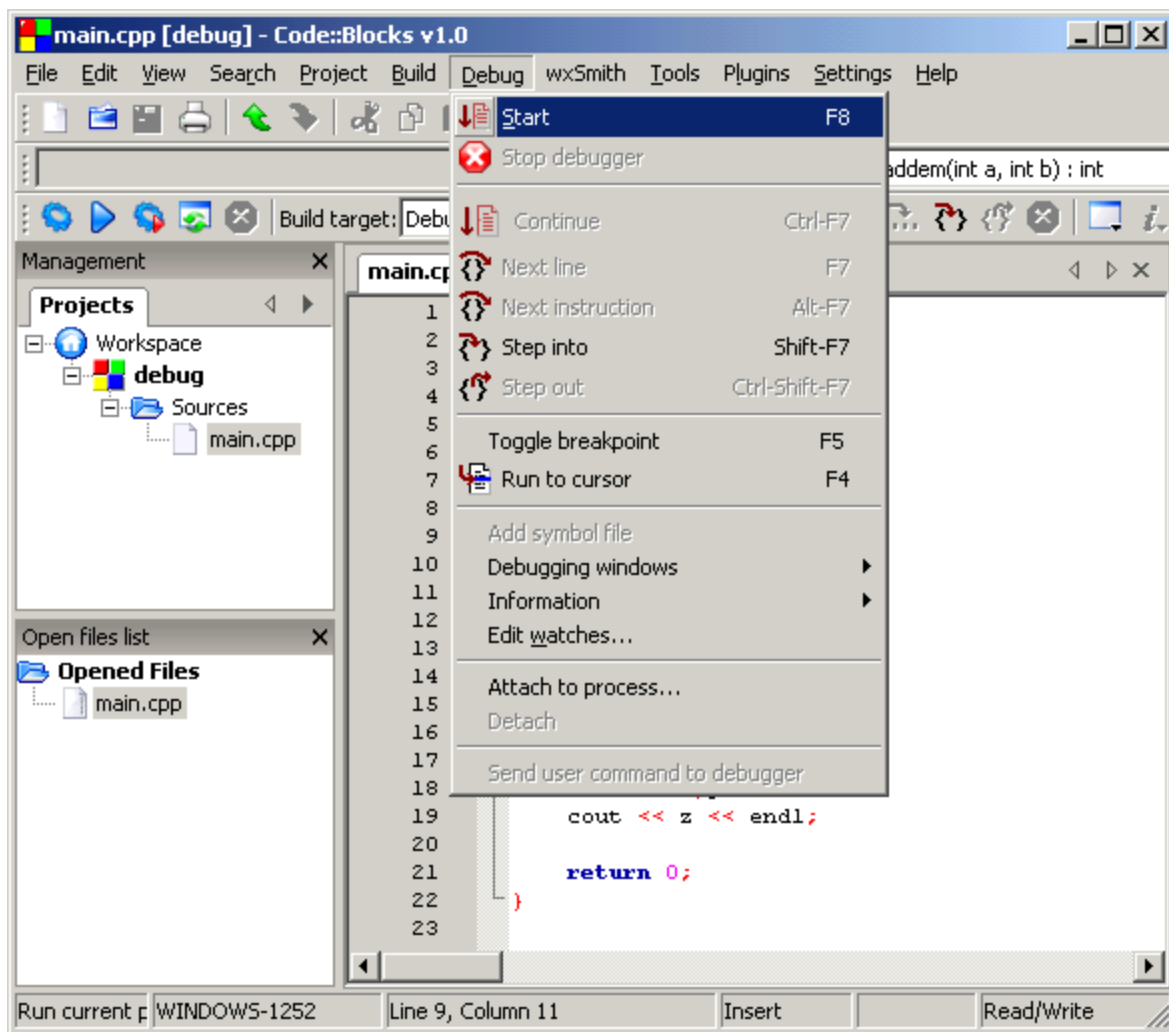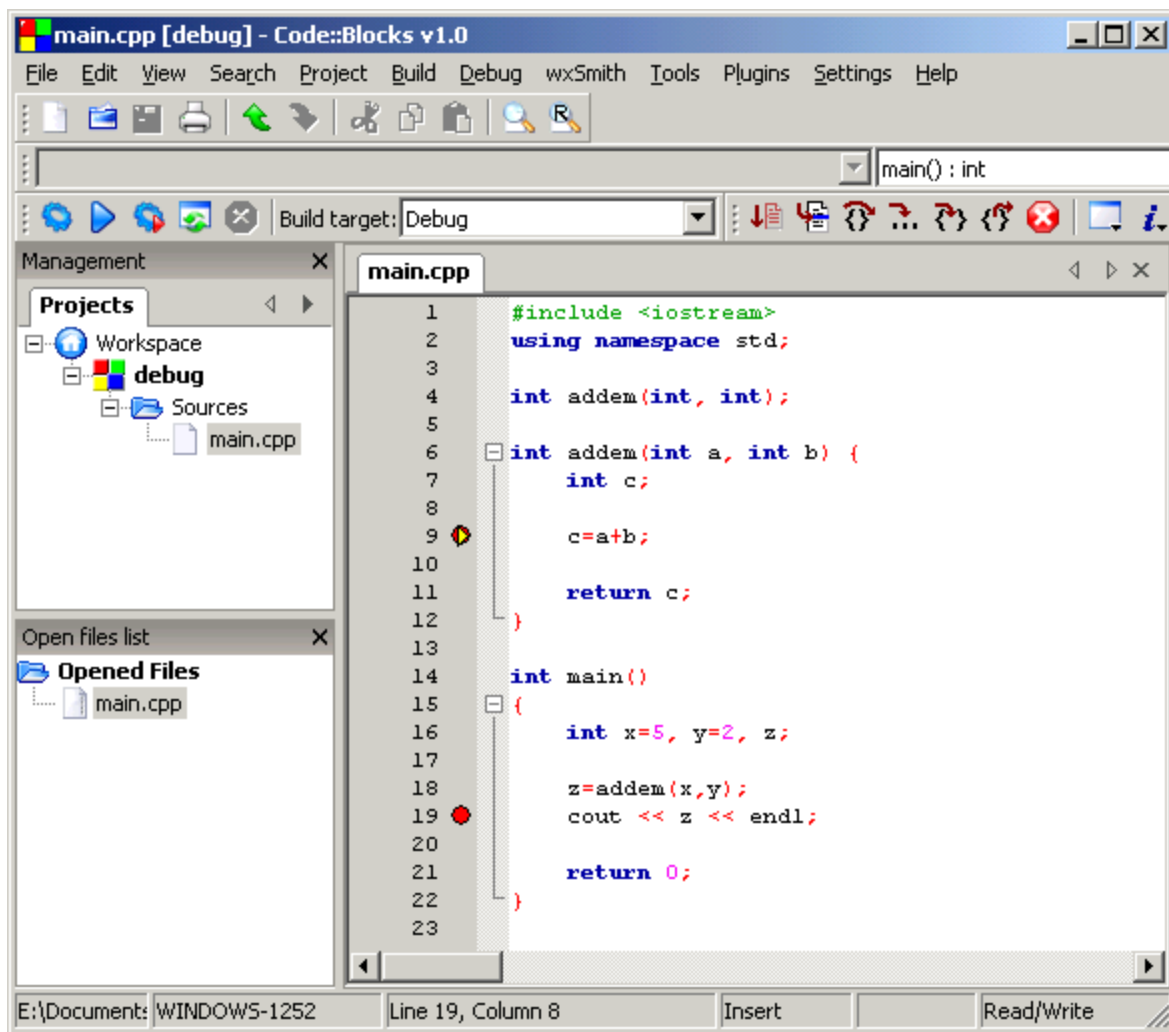
The program is started by selecting from the **Debug** pull-down menu, **Start**. This will run the program in the debugger until a breakpoint is encountered, at which point the program will pause.
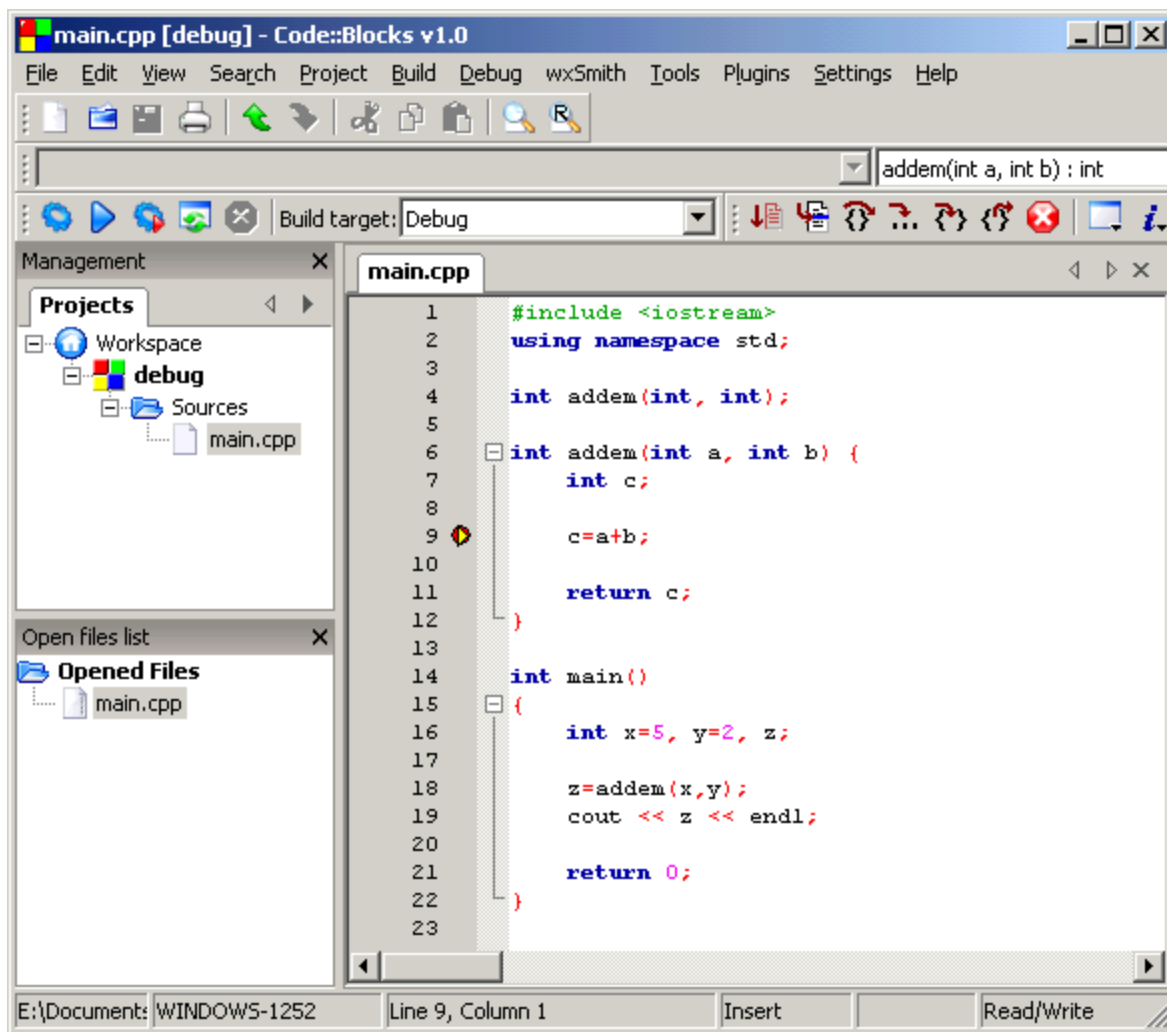
When the program pauses at the break point, a red circle with a yellow triangle mark will appear at the breakpoint.
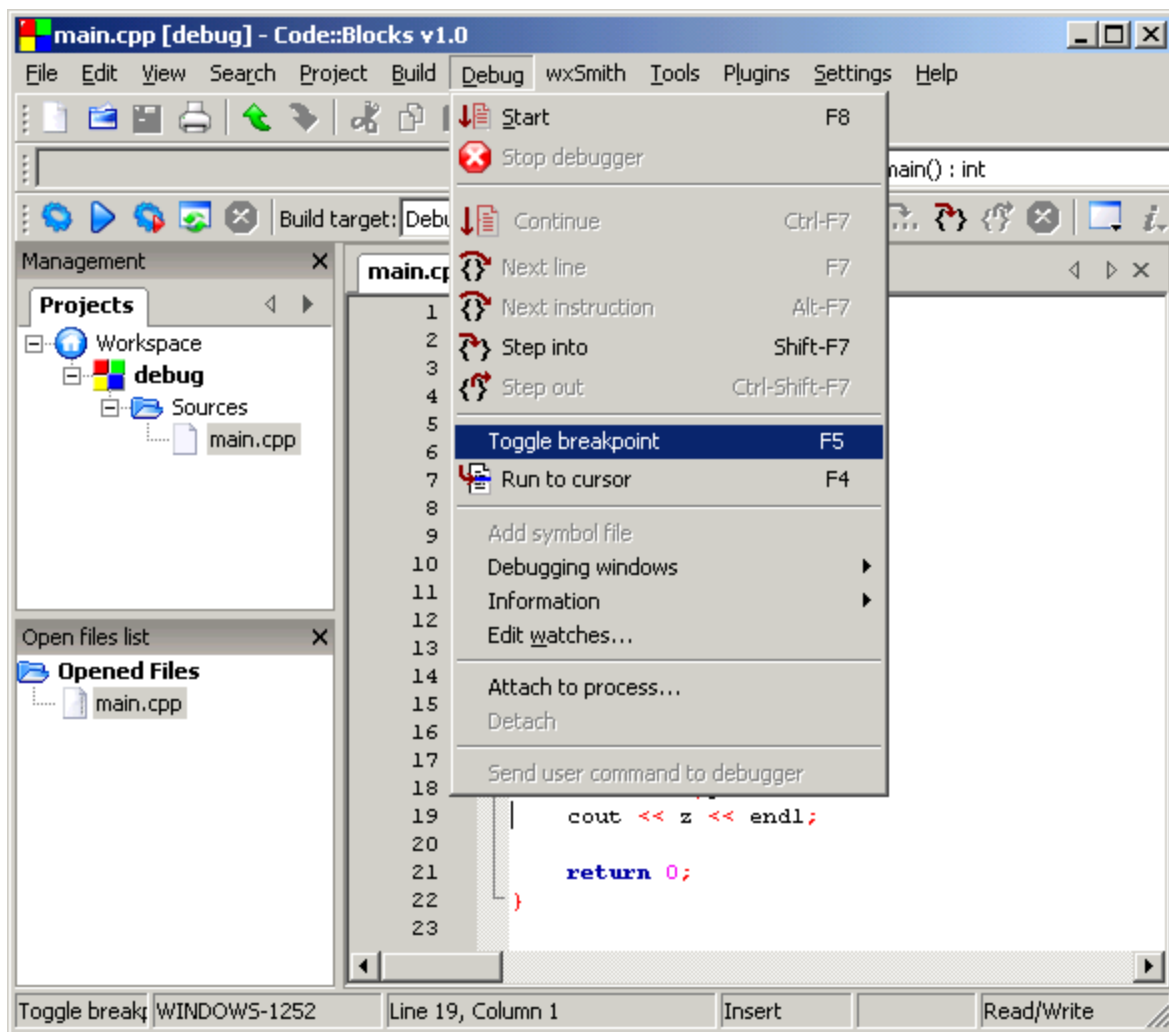
You can set multiple breakpoints. The keyboard shortcut **F5** allows you to toggle the breakpoint at any line.

```cpp
#include <iostream>
using namespace std;

int addem(int, int);

int addem(int a, int b) {
    int c;

    c=a+b;

    return c;
}

int main()
{
    int x=5, y=2, z;

    z=addem(x,y);
    cout << z << endl;

    return 0;
}
```

This screen shows breakpoints on lines 9 and 19, but line 9 indicates that the code has executed to that point.
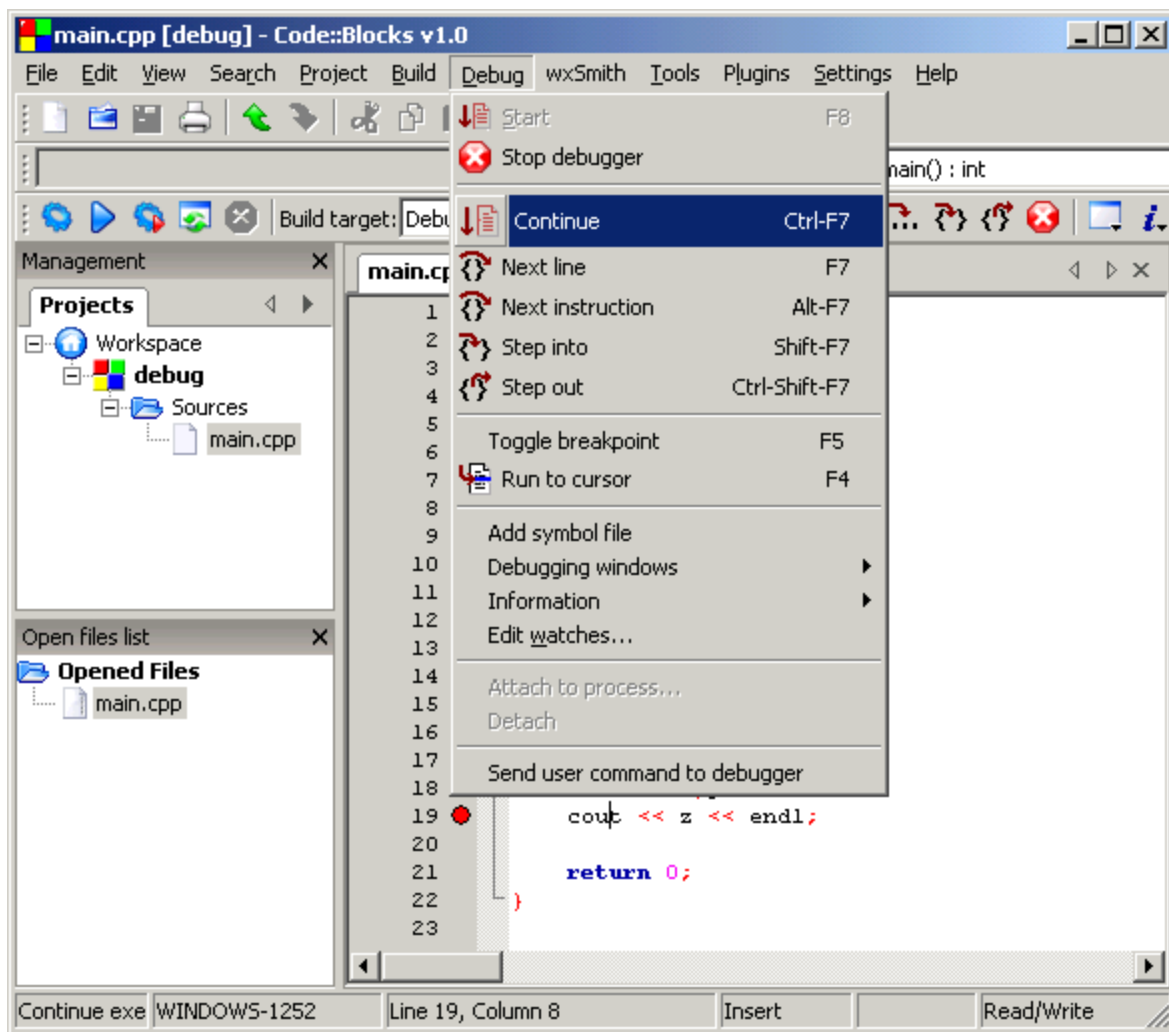
Selecting Continue from the Debugger menu will run the program till the next breakpoint.
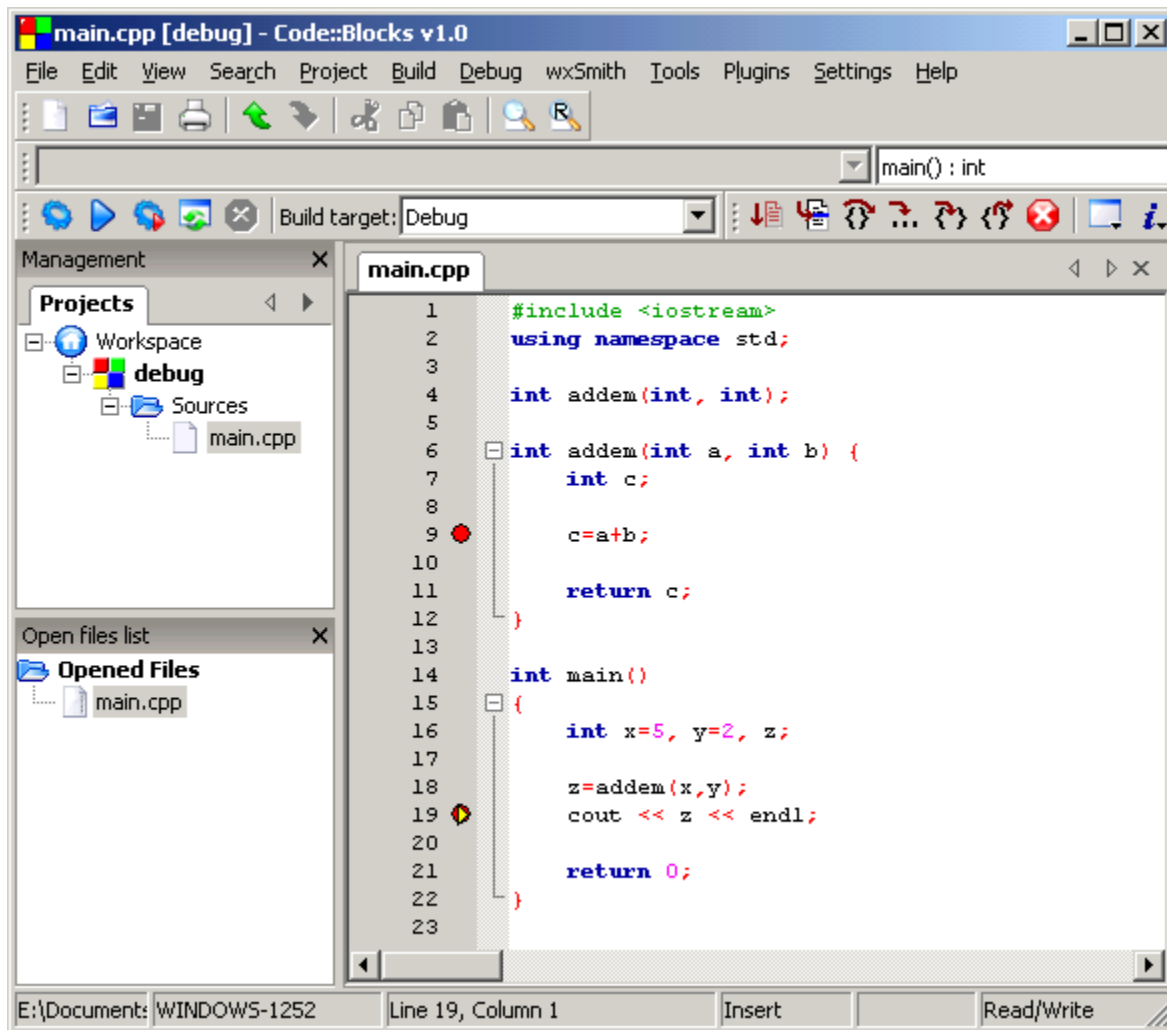
Now the program stops at line 19, because the program reached the second breakpoint.
Press Ctrl-F7 to continue. Now the program runs till the end of the program, because there are no further breakpoints to encounter.

When you exit Code:Blocks you may be presented with the following window.
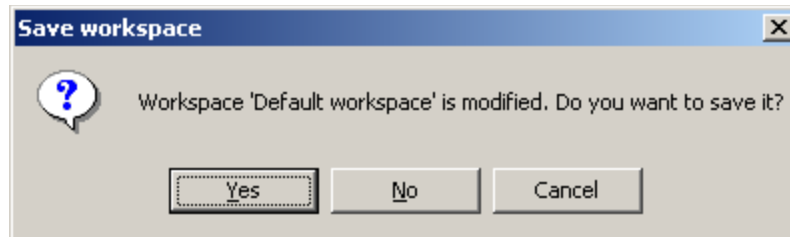
Say "Yes" to save the Workspace. This saves settings of the workspace you are working on.

You now know the basics of how to use the compiler, work with a project, and use the debugger. Good luck with your programming!