

**University American College Skopje**  
**School of Computer Science and Information Technology**  
**Course: Object Programming**

# Debugging in C++ Development Environment in CodeBlocks

**Lecture**

Prepared by: Stojan Kitanov PhD

# Contents

- Debugging
- Debugging in C++ Development Environment
- Debugging a C++ Program in CodeBlocks

# **Debugging**

- **Debugging** is a process of finding and reducing the number of bugs, or defects, in a computer program or a piece of electronic hardware, thus making it behave as expected. A general format to define a structure in C and C++ is the following:

**Bug = Software error**

# Debugging in C++ Development Environment

- Debuggers are software tools which enable the programmer to monitor the execution of a program, stop it, restart it, set breakpoints, and change values in memory. The term debugger can also refer to the person who is doing the debugging.
- The tool we are using in the class has the ability to debug the code and to monitor its execution, step by step.
- Moreover, it is possible to make changes in the code during debugging, which are preserved in the final version of the code
- This way it is accurately reflected where the problems in the code occur, thus allowing us to monitor the whole execution of the program
- Throughout the process, there is an open window for execution, which can be seen at any time.

# Debugging a C++ Program in Codeblocks

- 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.

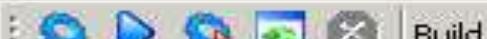
# main.cpp [debug] - Code::Blocks v1.0



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



main() : int



Build target: Debug



Management X

Projects

Workspace  
debug  
  Sources  
    main.cpp

Open files list X

Opened Files

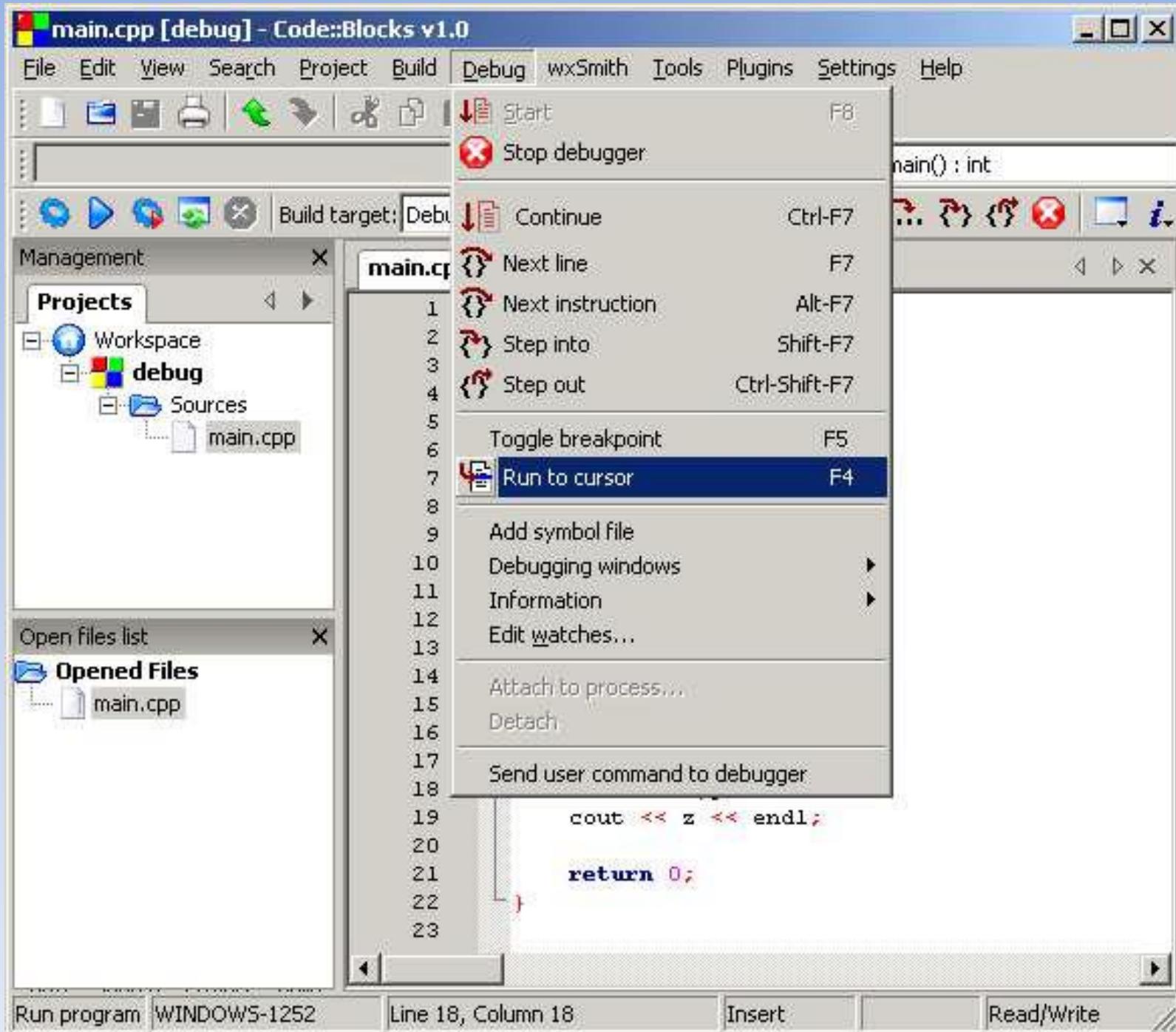
main.cpp

main.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
22 }
```

# Debugging a C++ Program in Codeblocks

- 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.**



# Debugging a C++ Program in Codeblocks

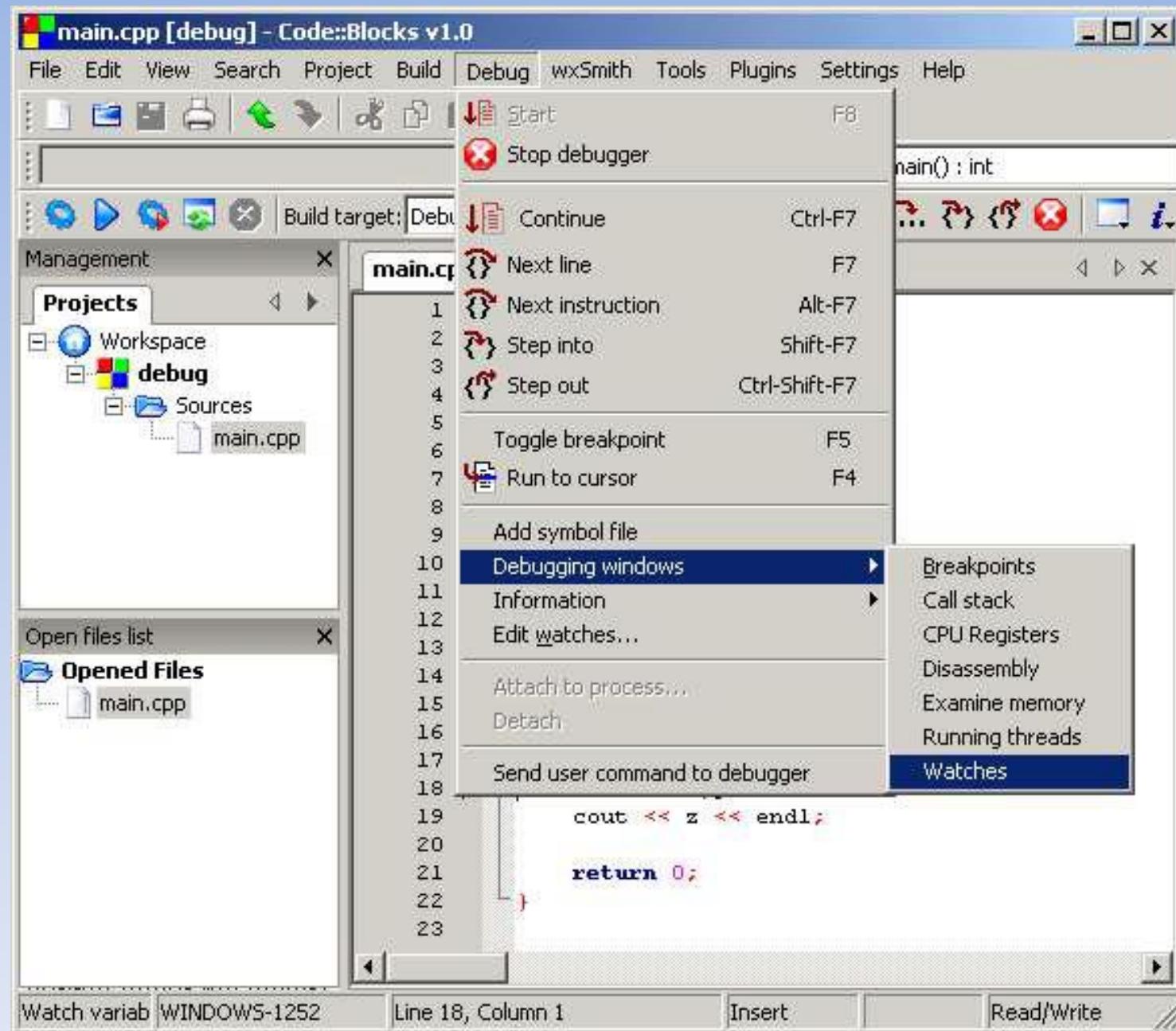
- The next step in debugging a program is to tell the program when to stop running so you can inspect the results. To do this, place the cursor over the line where you want your program to stop.
- For example, the cursor was placed at line 18 (which is hidden behind the menu). This is called a **Breakpoint**.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

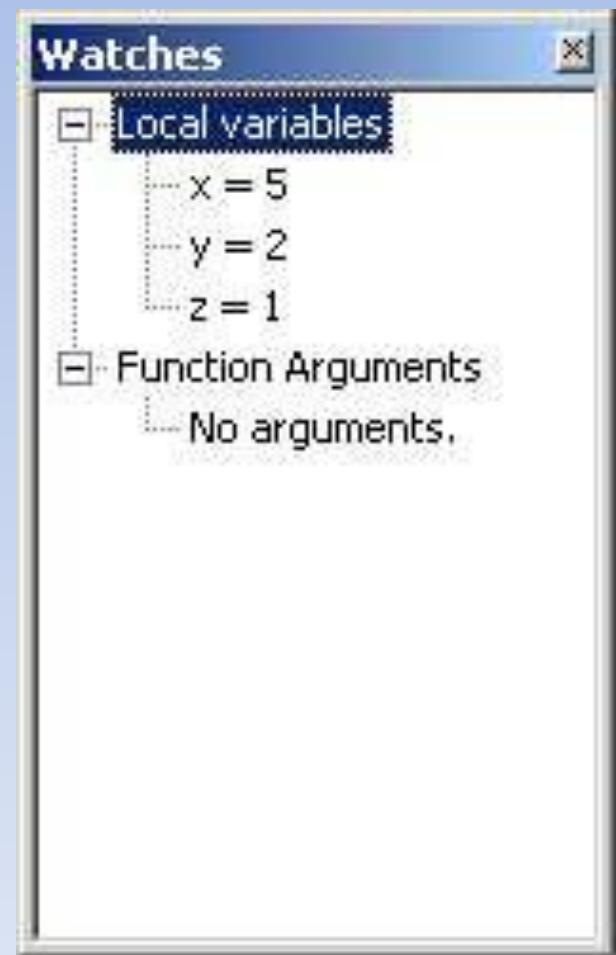
# Debugging a C++ Program in Codeblocks

- 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**.



# Debugging a C++ Program in Codeblocks

- 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.



# Pointers in the Structure

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

- Code::Blocks v1.0

File Edit Project Build Debug wxSmith Tools Plugins Settings Help

Local variables

- x = 5
- y = 2
- z = 1

Function Arguments

- No arguments.

Build target: Debug

main() : int

main.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
22 }
23
```

Open files list

Opened Files

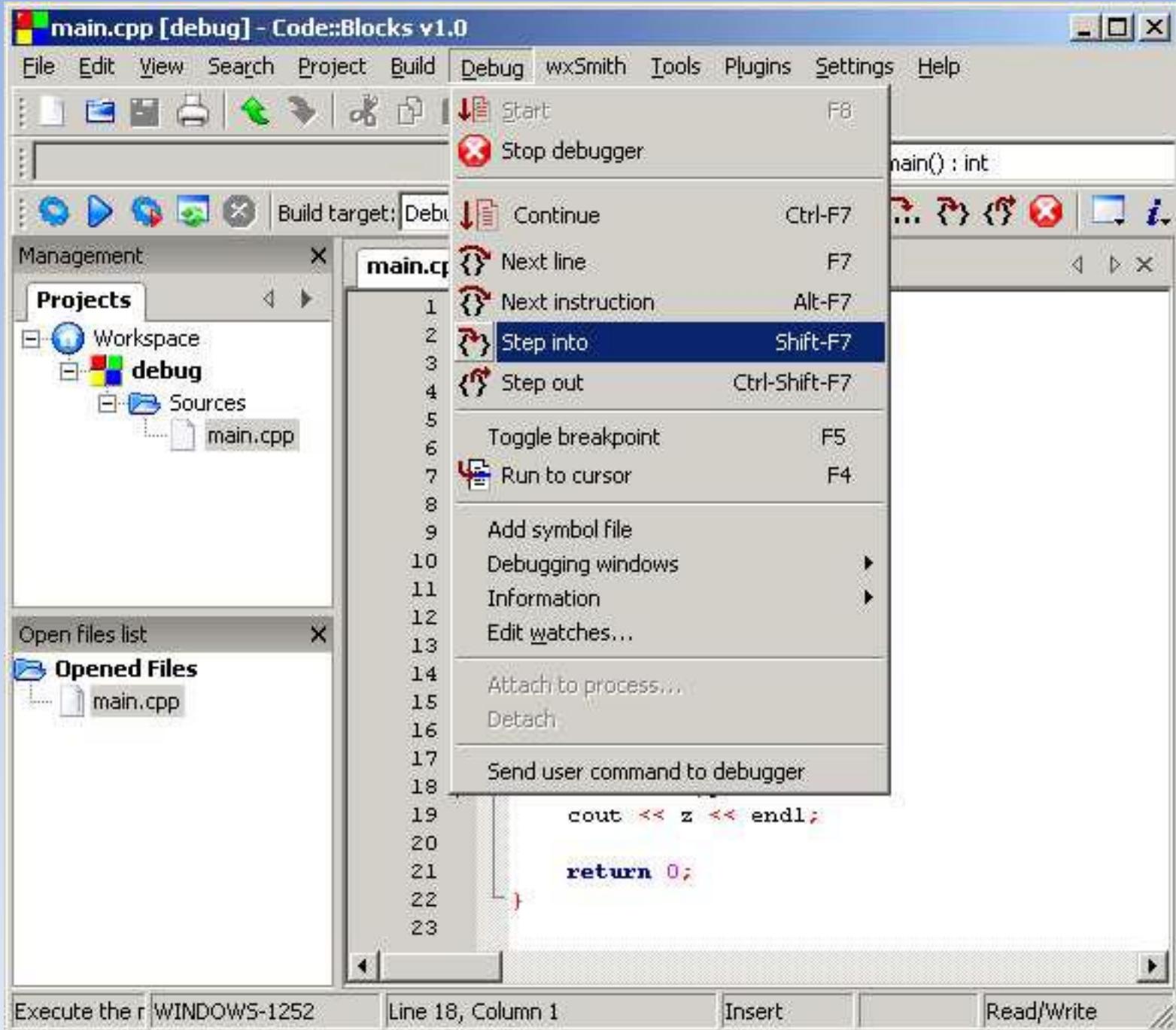
- main.cpp

# Debugging a C++ Program in Codeblocks

- 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.



# Debugging a C++ Program in Codeblocks

- The next step is line 9.

The screenshot shows the Code::Blocks IDE interface. The main window displays the code for 'main.cpp' with the following content:

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

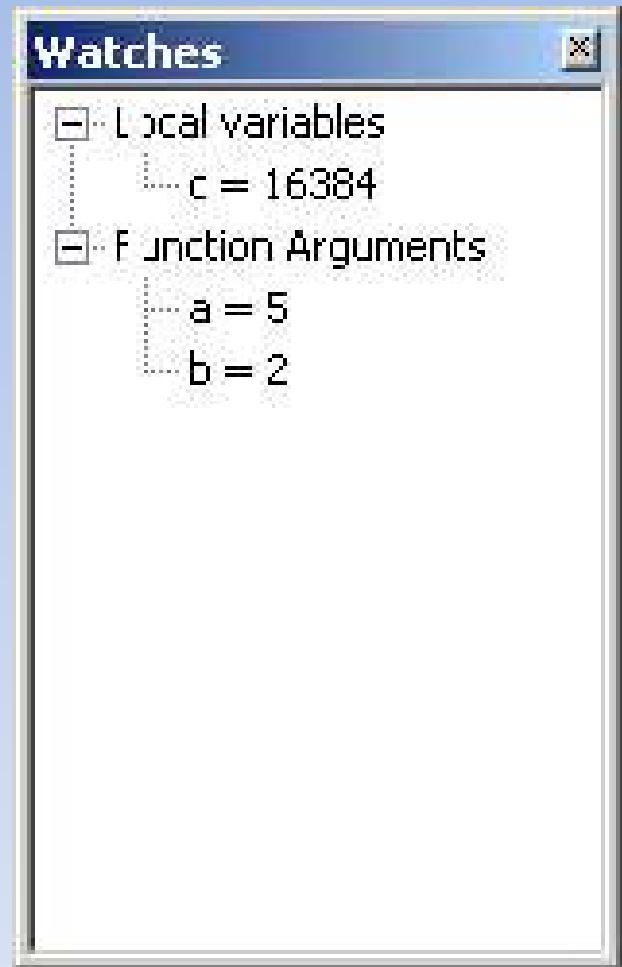
int addem(int, int);

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;
}
```

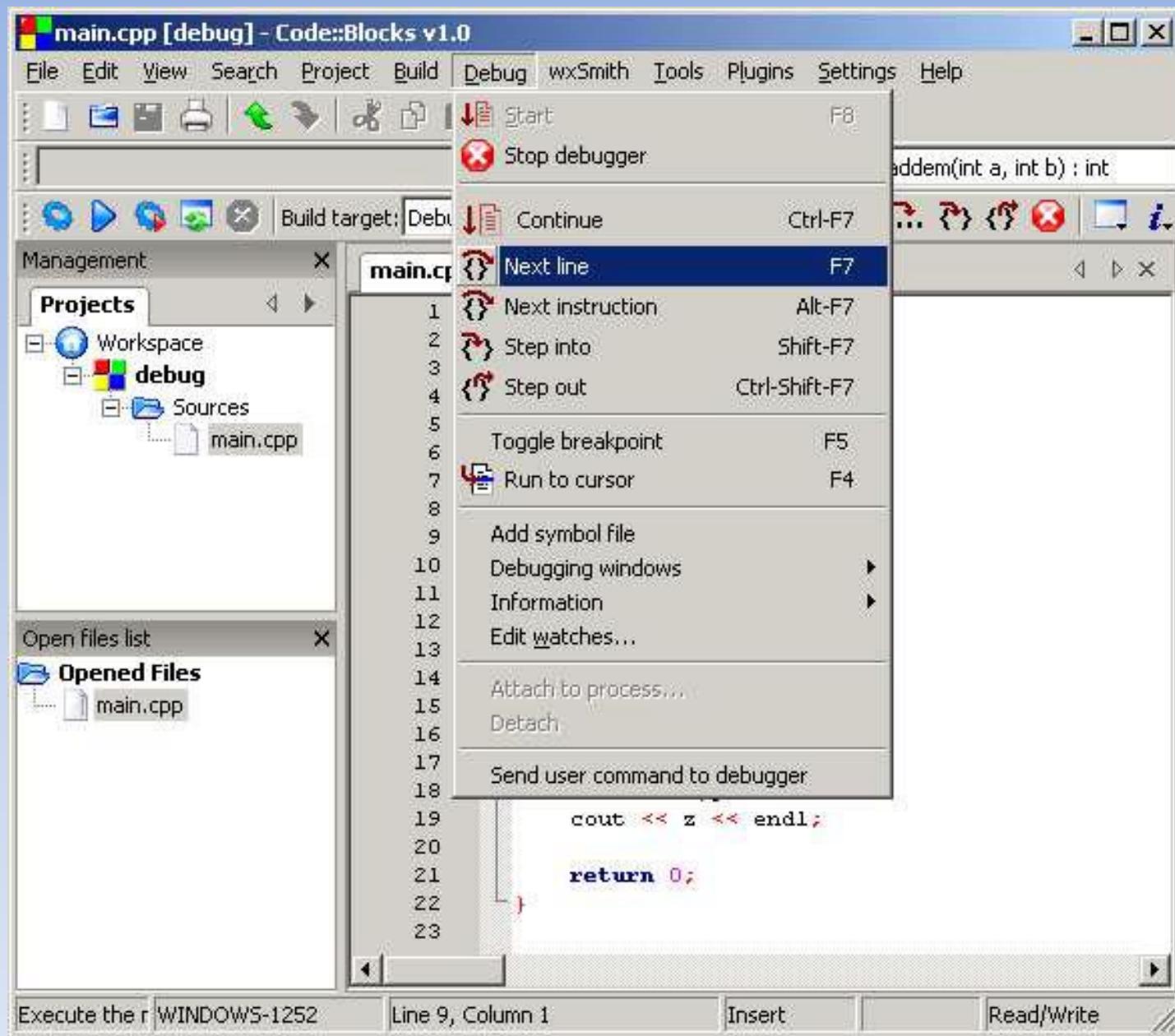
A yellow arrow icon is positioned next to the number 9, indicating a break point has been set on that line. The 'Build target' dropdown shows 'Debug'. The left sidebar includes a 'Projects' view showing a workspace named 'debug' containing a 'Sources' folder with 'main.cpp', and an 'Open files list' showing 'Opened Files' with 'main.cpp'.

- The arguments  $a$  and  $b$  are shown in the Watches window.  
Notice that the local variable  $c$ , which hasnot been set yet, has a random value.



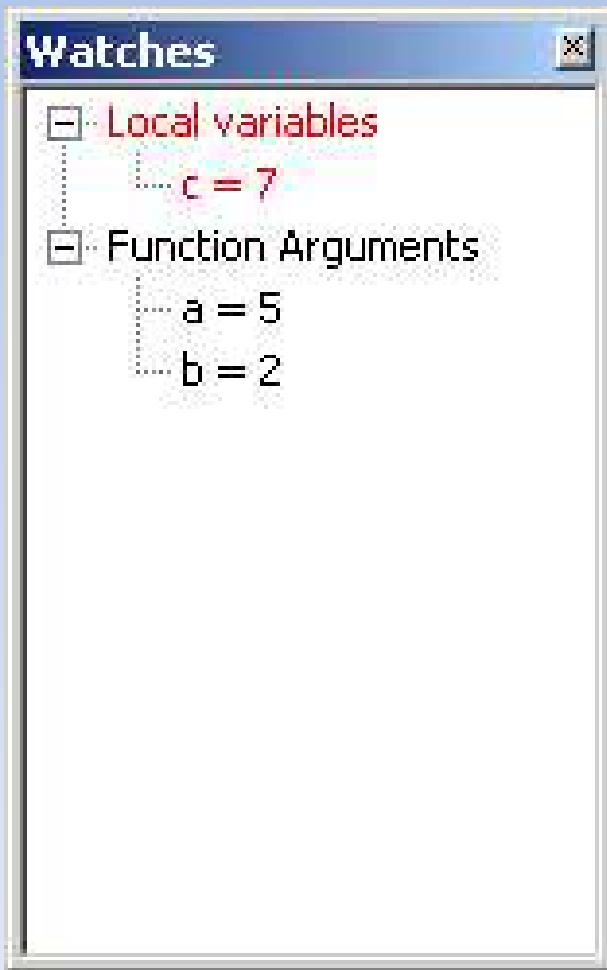
# Debugging a C++ Program in Codeblocks

- 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.



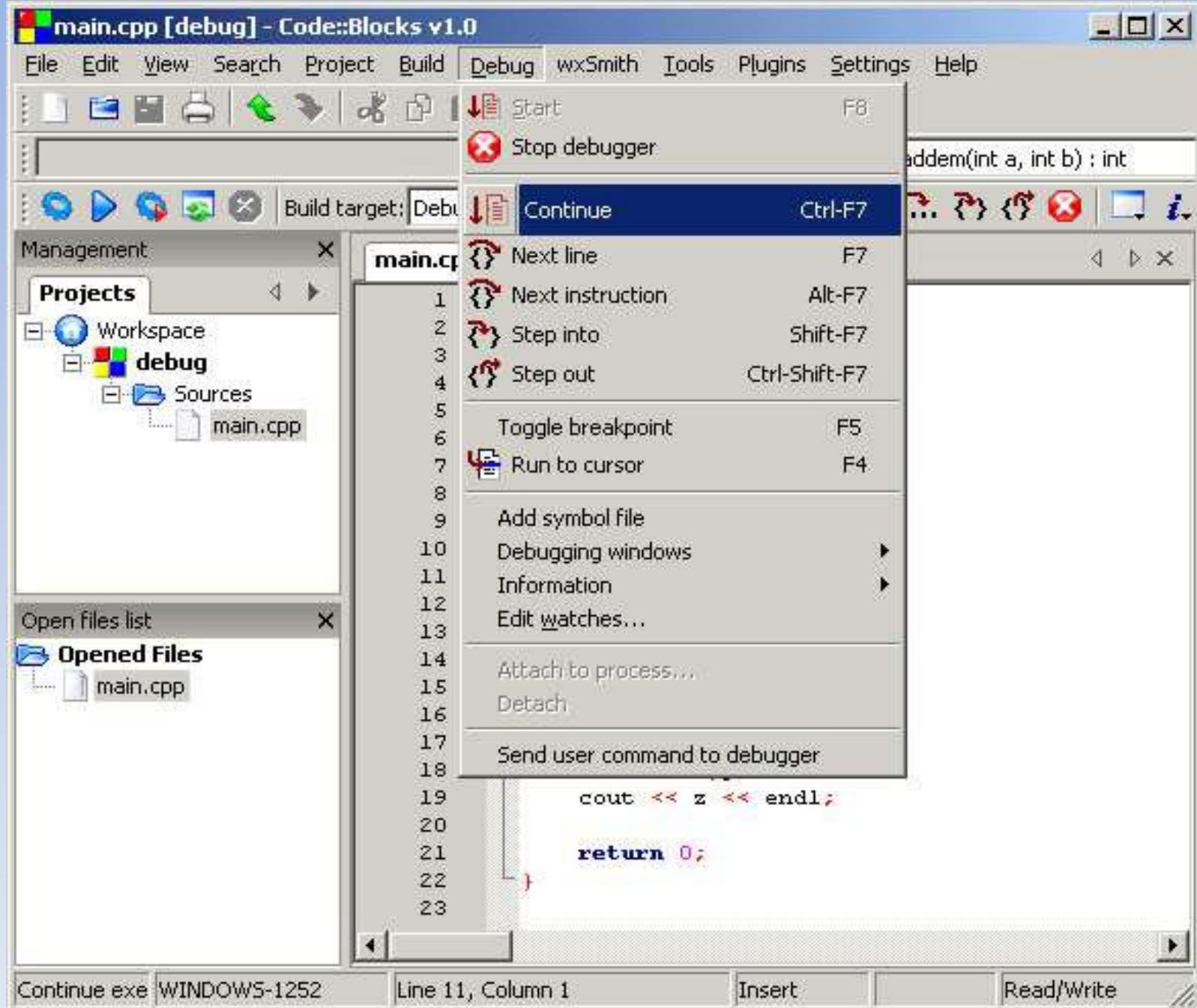
# Debugging a C++ Program in Codeblocks

- The debug window reflects the change of c.



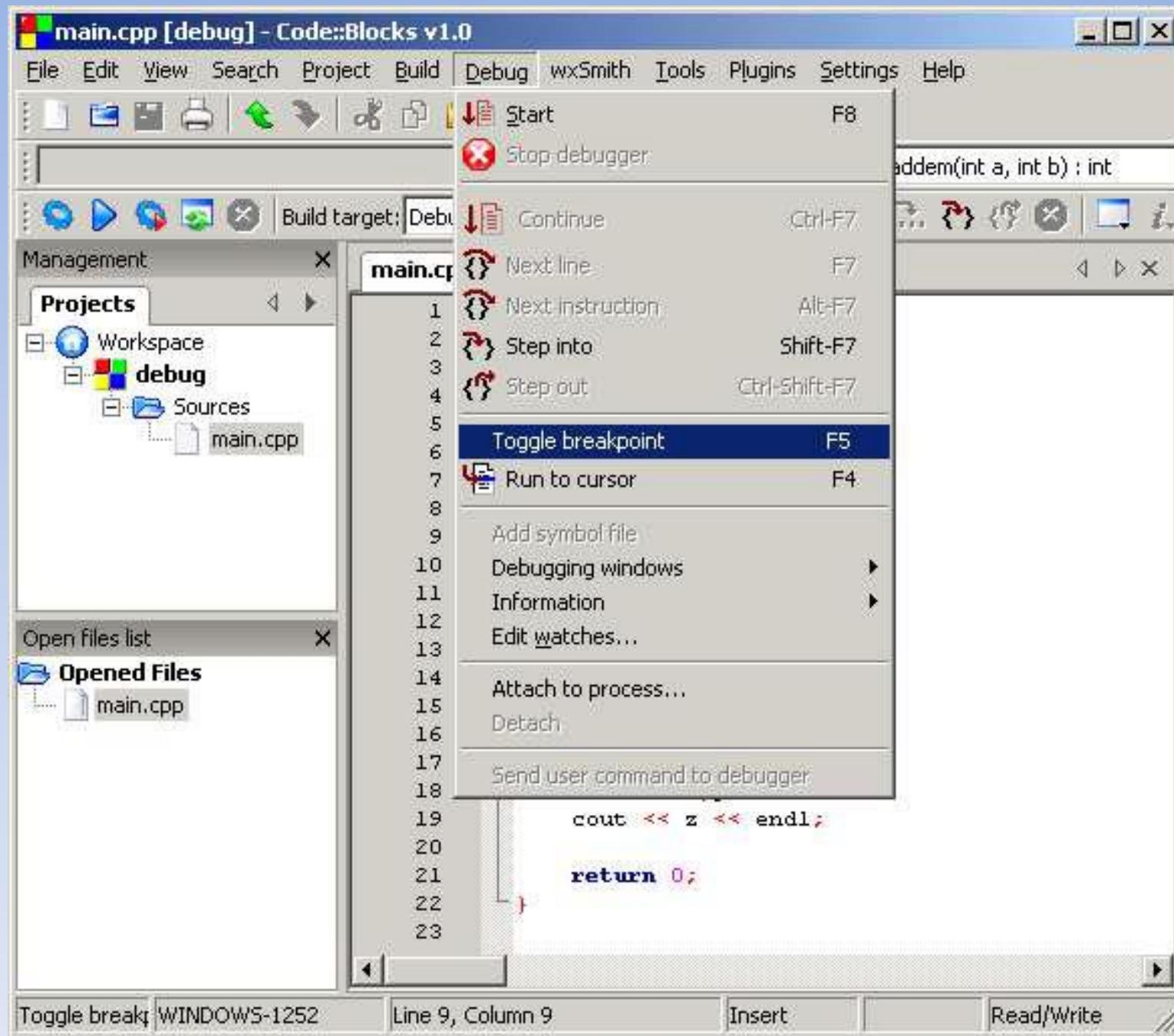
# Debugging a C++ Program in Codeblocks

- 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**.



# Debugging a C++ Program in Codeblocks

- 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.**



# Debugging a C++ Program in Codeblocks

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

The screenshot shows the Code::Blocks IDE interface. The main window displays the code for 'main.cpp'. A red circular marker is placed on the line number 9, indicating a breakpoint. The code defines a function 'addem' and its implementation, along with the main function which calls it and prints the result. The 'Projects' panel on the left shows a workspace named 'debug' containing a source file 'main.cpp'. The status bar at the bottom indicates 'Line 9, Column 11'.

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

int addem(int, int);

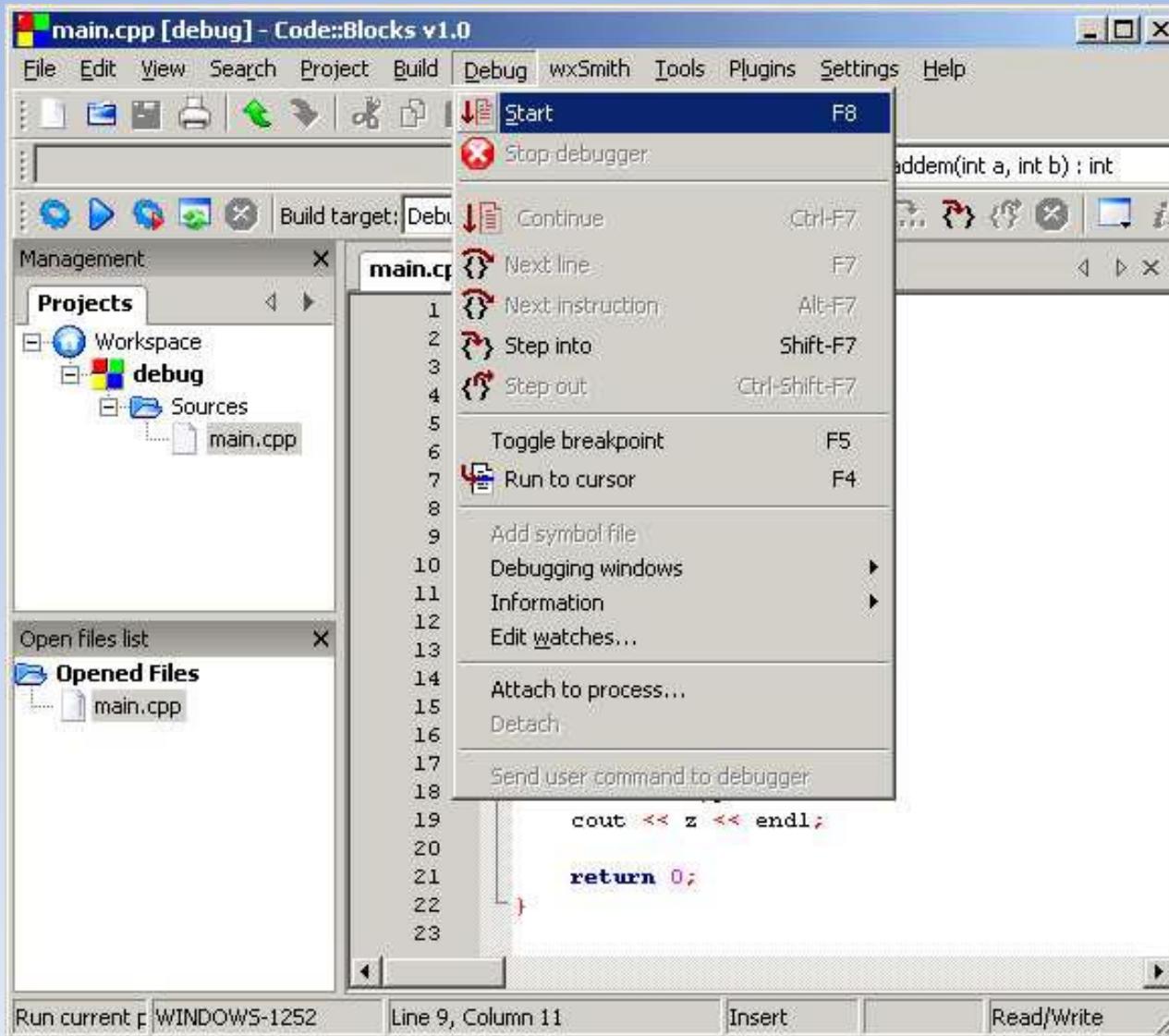
int addem(int a, int b) {
    int c;
    c=a+b;
    return c;
}

int main()
{
    int x=5, y=2, z;
    z=addem(x,y);
    cout << z << endl;
    return 0;
}
```

# Debugging a C++ Program in Codeblocks

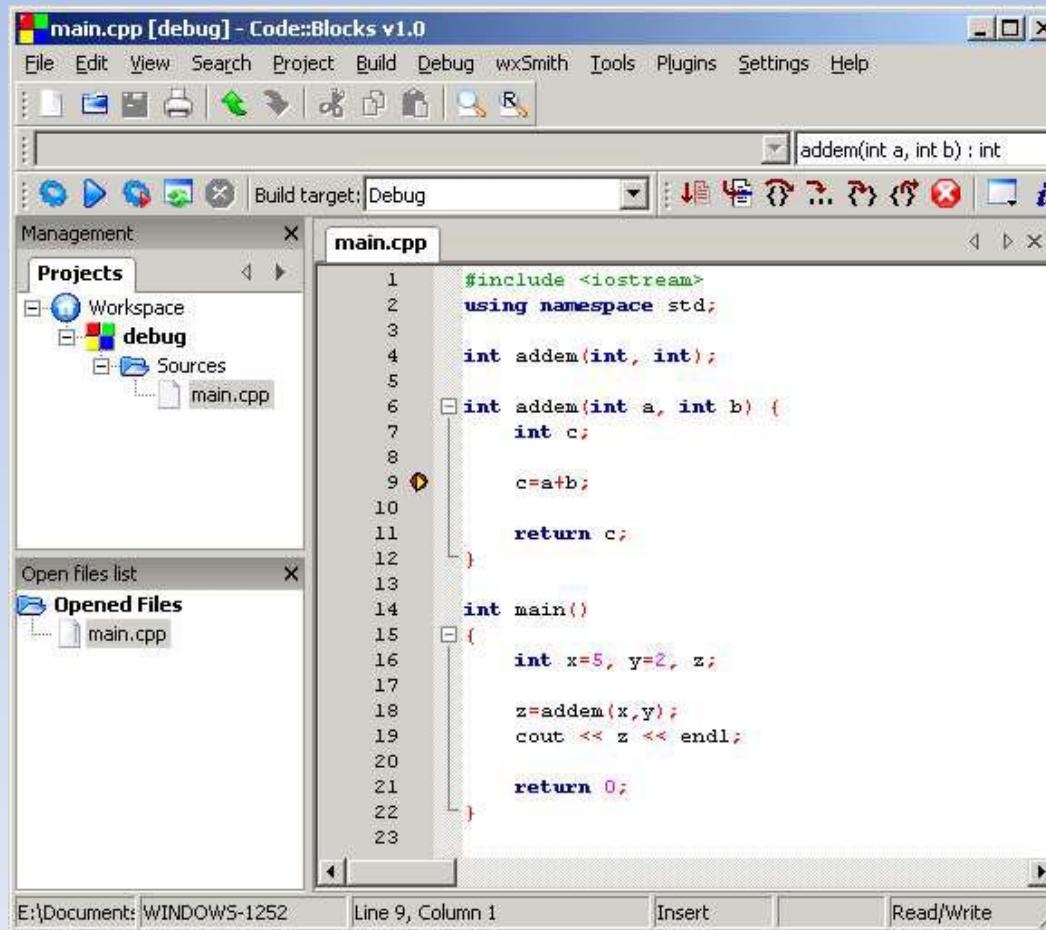
- The program is started by selecting from the **Debug** pull-down menu, **Start**. This will run the program in the debugger until a breakpoint is encountered, at which point the program will pause.

# Debugging a C++ Program in Codeblocks



# Debugging a C++ Program in Codeblocks

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



The screenshot shows the Code::Blocks IDE interface. The main window displays the code editor with the file "main.cpp" open. A red circle with a yellow triangle is visible on the left margin of line 9, indicating a breakpoint. The code in main.cpp is as follows:

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

int addem(int, int);

int addem(int a, int b) {
    int c;
    c=a+b;
    return c;
}

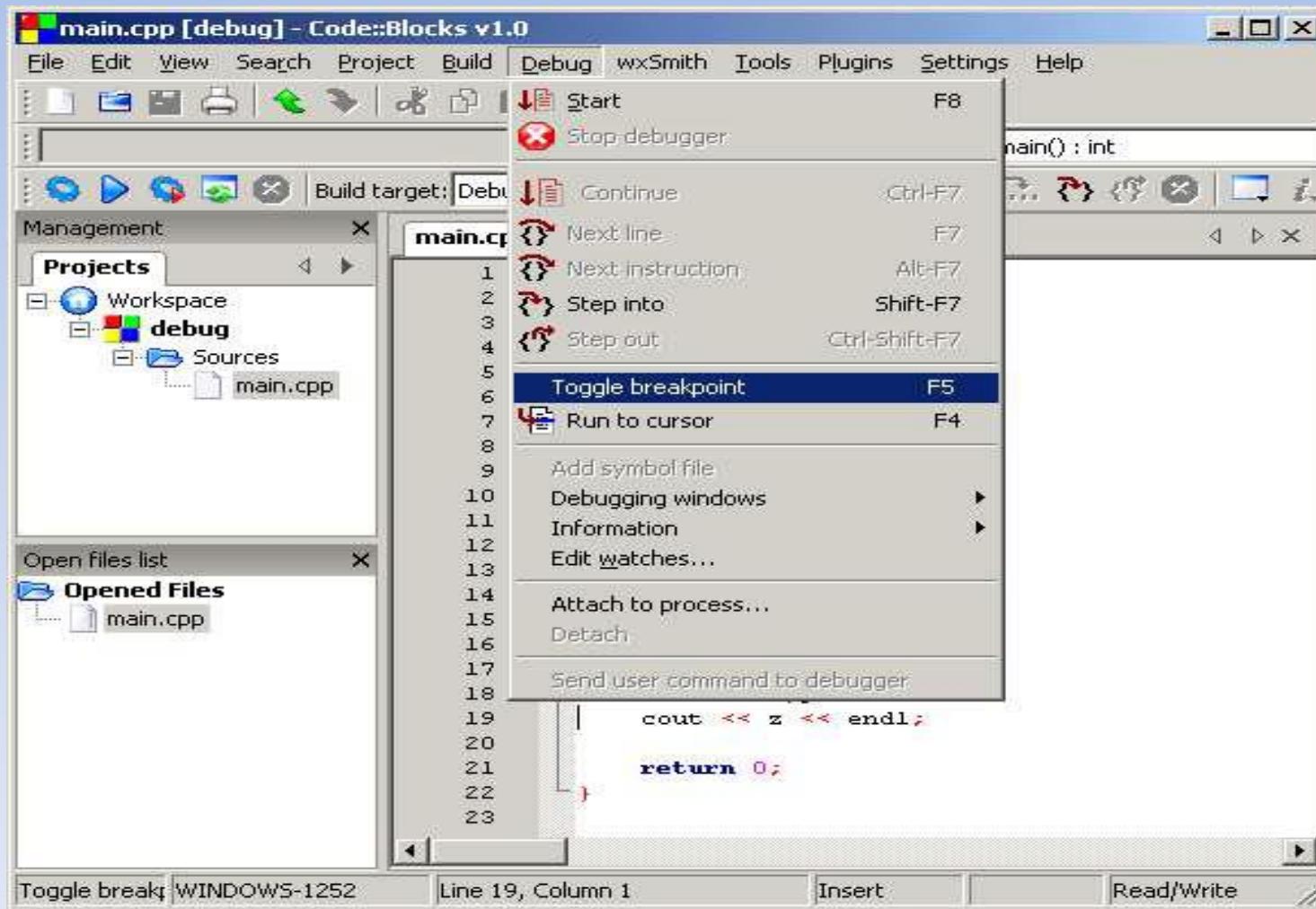
int main()
{
    int x=5, y=2, z;
    z=addem(x,y);
    cout << z << endl;

    return 0;
}
```

The status bar at the bottom shows "E:\Document: WINDOWS-1252", "Line 9, Column 1", "Insert", and "Read/Write".

# Debugging a C++ Program in Codeblocks

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



# Debugging a C++ Program in Codeblocks

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

The screenshot displays the Code::Blocks IDE interface. The main window shows the code editor with the file `main.cpp`. The code contains two breakpoint markers: one yellow circle at line 9 and one red circle at line 19. The code itself is as follows:

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

int addem(int, int);

int addem(int a, int b) {
    int c;
    c=a+b;
    return c;
}

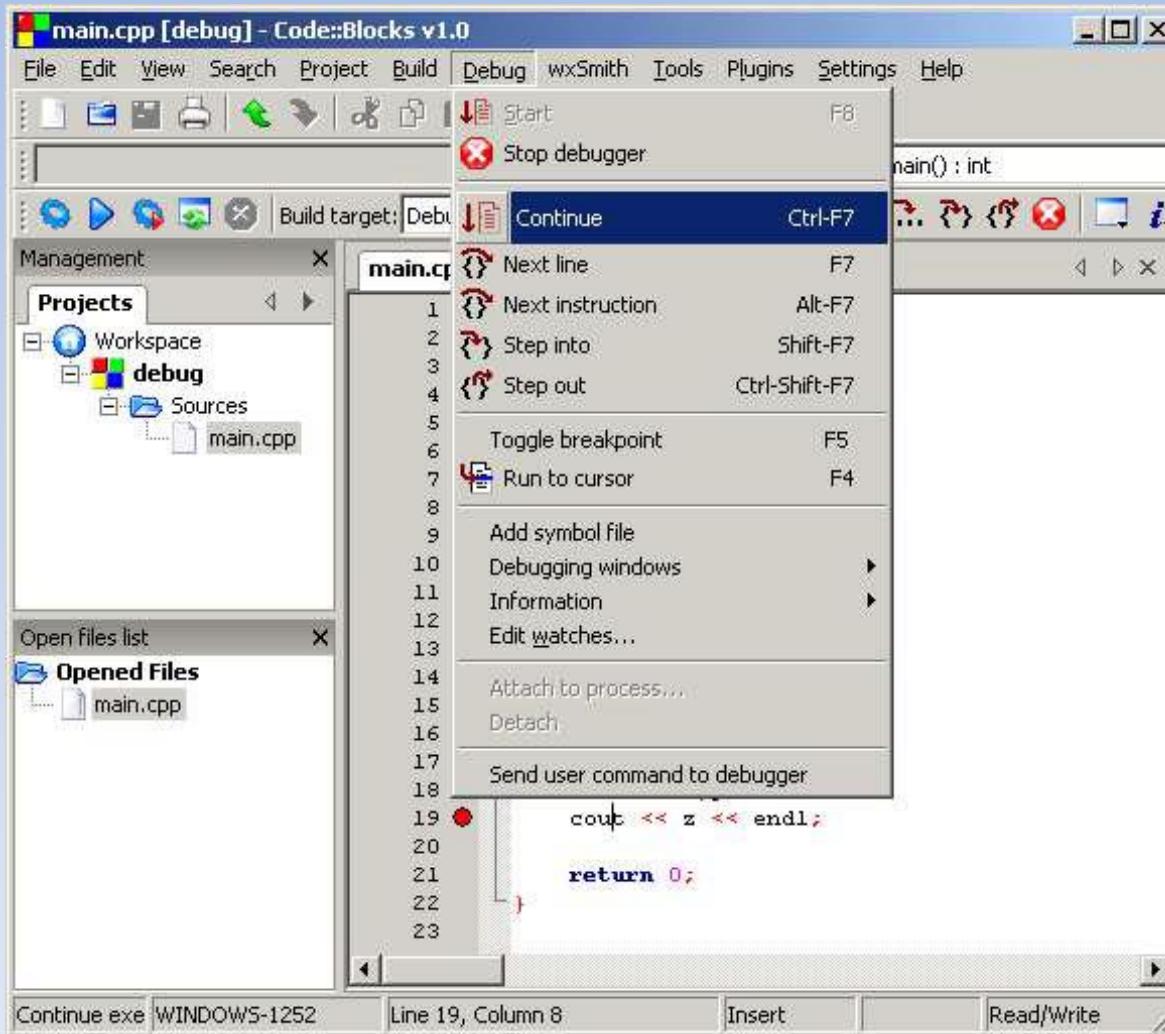
int main()
{
    int x=5, y=2, z;
    z=addem(x,y);
    cout << z << endl;

    return 0;
}
```

The status bar at the bottom indicates "Line 19, Column 8".

# Debugging a C++ Program in Codeblocks

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



# Debugging a C++ Program in Codeblocks

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.

The screenshot shows the Code::Blocks IDE interface. The main window displays the code for 'main.cpp' with two red circular breakpoints set on line 9 and line 19. The status bar at the bottom indicates 'Line 19, Column 1'. The 'Build target' dropdown shows 'Debug'. The 'Management' panel on the left shows a project named 'debug' with a single source file 'main.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;
}
```

# Debugging a C++ Program in Codeblocks

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.