



Just because it compiles does
not make it good code

Compiler

human readable code is converted into machine readable code

Makefile

scripts (tools) to help automate the compilation/build, organize dependencies

IDEs

helpful tools to make programming easier

(Visual Studios (Windows Only),

Visual Studio Code, Sublime, Geany, Eclipse, etc.

(Python specific Pycharm, IDLE, etc.)

Flags to compilers

options passed to the program



Code must be

- **Reliable**
- **Maintainable**



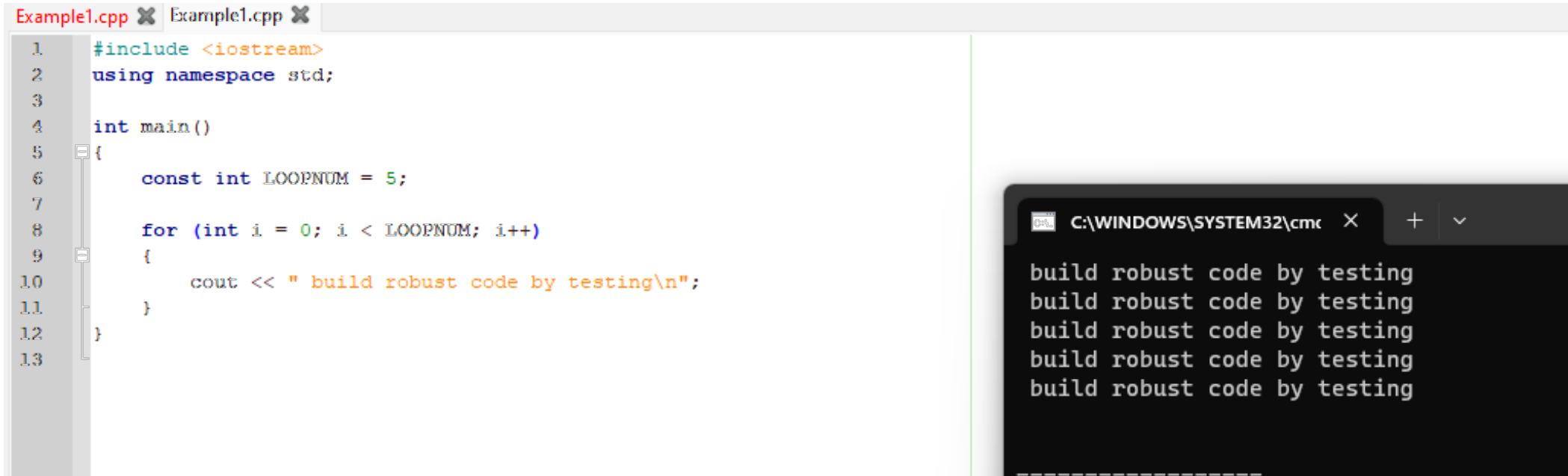
- **Reliable**
 - testing
 - run time errors
 - coding standards / proper syntax
- **Maintainable**
 - coding standards



Following syntax rules

IDEs are usually good about correcting these, however a valid statement compiles even if the intent is wrong

Example:



The image shows a screenshot of a C++ development environment. On the left, a code editor displays the source code for a file named 'Example1.cpp'. The code includes the <iostream> header, uses the std namespace, and defines a main function. Inside main, a constant integer LOOPNUM is set to 5, and a for loop iterates from 0 to 4, printing the string 'build robust code by testing' five times. On the right, a terminal window titled 'C:\WINDOWS\SYSTEM32\cmd' shows the output of the program, which is the same string printed on five separate lines.

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     const int LOOPNUM = 5;
7
8     for (int i = 0; i < LOOPNUM; i++)
9     {
10         cout << " build robust code by testing\n";
11     }
12 }
13
```

```
C:\WINDOWS\SYSTEM32\cmd
build robust code by testing
build robust code by testing
build robust code by testing
build robust code by testing
build robust code by testing
```

Still compiles – perfectly valid, but not giving us correct behavior – **run time behavior is unexpected**

The image shows a screenshot of an IDE (Geany) with a C++ file named 'Example1.cpp'. The code in the editor is as follows:

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     const int LOOPNUM = 5;
7
8     for (int i = 0; i < LOOPNUM; i++);
9     {
10        cout << " build robust code by testing\n";
11    }
12 }
13
```

The IDE's status bar at the bottom shows the compilation command: `g++ -o "Example1" "Example1.cpp" (in directory: C:\Users\Deborah\Desktop\Lecture375BuildingBetterCode)` and the message: `Compilation finished successfully.`

Overlaid on the right side of the IDE is a terminal window titled `C:\WINDOWS\SYSTEM32\cmd`. The terminal output is:

```
build robust code by testing
-----
(program exited with code: 0)
Press any key to continue . . .
```



A compiler is just a program – **the programmer must build good code**

IDEs can help / flags to compilers

For example –Wall to g++

Not necessary, as testing would reveal, however very convenient

The image shows a Geany IDE window with a C++ file named 'Example1.cpp'. The code is as follows:

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     const int LOOPNUM = 5;
7
8     for (int i = 0; i < LOOPNUM; i++);
9     cout << " build robust code by testing\n";
10
11 }
12
13
```

The IDE's status bar shows the compilation command: `g++ -o "Example1" "Example1.cpp" -Wall (in directory: C:\Users\Deborah\Desktop\Lecture375BuildingBetterCode)`. The compiler messages pane displays the following warnings:

```
Example1.cpp: In function 'int main()':
Example1.cpp:8:2: warning: this 'for' clause does not guard... [-Wmisleading-indentation]
   for (int i = 0; i < LOOPNUM; i++);
   ^~~
Example1.cpp:9:2: note: ...this statement, but the latter is misleadingly indented as if it is guarded by the 'for'
   {
   ^
Compilation finished successfully.
```

To the right, a terminal window titled 'C:\WINDOWS\SYSTEM32\cmd.exe' shows the program's output:

```
build robust code by testing
-----
(program exited with code: 0)
Press any key to continue . . .
```

Let's look at the same program in
Visual Studios

From the compiler perspective, this is still valid code and compiles cleanly

Runtime gives unexpected results

The image shows a Visual Studio IDE with a C++ project named 'ConsoleApplication1'. The code in 'Example1.cpp' is as follows:

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     const int LOOPNUM = 5;
7     for (int i = 0; i < LOOPNUM; i++);
8     {
9         cout << " build robust code by testing\n";
10    }
11 }
12
13
```

The code is syntactically valid, but the loop body is empty. The debugger window shows the output: 'build robust code by testing'. The output window shows the following text:

```
Rebuild started...
1>----- Rebuild All started: Project: ConsoleApplication1, Configuration: Debug x64 -----
1>Example1.cpp
1>ConsoleApplication1.vcxproj -> C:\Users\Deborah\Desktop\Lecture375BuildingBetterCode\ConsoleApplication1\x64\Debug\ConsoleApplication1.exe
===== Rebuild All: 1 succeeded, 0 failed, 0 skipped =====
===== Rebuild started at 8:52 AM and took 01.142 seconds =====
```

The output window text is circled in red. The status bar at the bottom indicates 'Rebuild All succeeded'.

You can adjust the properties in several places to reveal suppressed warnings, again do not rely on your IDE. All programs should be tested.

In the options for Properties, you will see various levels using **Warning Level**

For this particular example, Level3 (/W3) does not reveal the problem

If we go to

Level4 (/W4)

Or

EnableAllWarnings (/Wall)

You will find you get many warnings – this can become noise and you miss important warnings (we discussed this with the double vs float earlier this quarter)

Let's take a look For this particular error it still doesn't reveal

```
Example1.cpp x JohnL_GradingCode.cpp
ConsoleApplication1 (Global Scope) main()
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     const int LOOPNUM = 5;
7
8     for (int i = 0; i < LOOPNUM; i++);
9     {
10        cout << " build robust code by testing\n";
11    }
12
13
14
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'ConsoleApplication1' (1 of 1)
- ConsoleApplication1
 - References
 - External Dependencies
 - Header Files
 - Resource Files
 - Source Files
 - Example1.cpp

Microsoft Visual Studio Debug

```
build robust code by testing
C:\Users\Deborah\Desktop\Lecture375BuildingBetterCode\ConsoleApplication1.exe (0) exited with code 0.
To automatically close the console when debugging stops, enable the option 'Automatically close console when debugging stops' in the 'Debugging' tab of the 'Debugging' dialog box.
Press any key to close this window . . .
```

161% No issues found

Output

Show output from: Build

```
Rebuild started...
1>----- Rebuild All started: Project: ConsoleApplication1, Configuration: Debug x64 -----
1>Example1.cpp
1>C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.35.32215\include\limits.h(70,5): warning C4668: '__STDC_WANT_SECURE_LIB__' is not defined as a preprocessor macro, replacing with '0' [C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.35.32215\include\xmemory(154,5)]
1>C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.35.32215\include\xmemory(164,5): warning C4365: 'argument': conversion from 'long' to 'unsigned int', signed/unsigned mismatch [C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.35.32215\include\atomic(93,9)]
1>C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.35.32215\include\atomic(93,9): warning C4365: 'argument': conversion from 'long' to 'unsigned int', signed/unsigned mismatch [C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.35.32215\include\atomic(291,9)]
1>C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.35.32215\include\atomic(291,9): warning C4365: 'argument': conversion from 'long' to 'unsigned int', signed/unsigned mismatch [C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.35.32215\include\atomic(307,9)]
1>C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.35.32215\include\atomic(307,9): warning C4365: 'argument': conversion from 'long' to 'unsigned int', signed/unsigned mismatch
1>ConsoleApplication1.vcxproj -> C:\Users\Deborah\Desktop\Lecture375BuildingBetterCode\ConsoleApplication1\x64\Debug\ConsoleApplication1.exe
1>Done building project "ConsoleApplication1.vcxproj".
1>----- Rebuild All: 1 succeeded, 0 failed, 0 skipped -----
=====
===== Rebuild started at 9:15 AM and took 01.216 seconds =====
```

Always test your code

In addition, if you are building a tool –
wear the hat of your user
(better still – have user testing)

Visual Studios is fairly thorough – however it turns off some warnings by default

For this class, using the Visual Studios compiler defaults is fine

For command line, use `-Wall`

Moral of the story: IDEs are not a replacement for good coding practices and following rules of the language

<https://learn.microsoft.com/en-us/cpp/preprocessor/compiler-warnings-that-are-off-by-default?view=msvc-170&redirectedfrom=MSDN>

Summary:

Practice good code habits

Follow syntax and rules

- Arrays cannot be dynamic (even if the compiler lets you).
Vector arrays can
- Never assume a variable's initial value – always initialize
- Test your code

When in doubt – ask! Class website first