Enabling `-std=c++14` flag in Code::Blocks

I have installed Code::Blocks for Windows and want to compile C++14 code like generic lambdas but the binary version of Code::Blocks that I've installed from codeblocks.org doesn't support the flag `-std=c++14`.

How do I update the compiler and enable `-std=c++14` flag for Code::Blocks?

c++ codeblocks c++14 generic-lambda

edited Jul 2 '15 at 8:41

asked Jul 1 '15 at 21:26

Andreas DM

4,890 4 15 39

I would recommend installing MinGW-w64. It is an active fork of the original MinGW which has now fallen into disrepair. The link given in the current top answer is one guy's particular build of MinGW-w64; however if you get it from the official site you can use the online installer and just pick the options you want. – M.M Jul 2 '15 at 10:41

2 Answers

May a humble newbie make one small suggestion? A small modification to test C++14 code, to allow resulting .exe file to be run independently of the IDE it was created in, slightly modified test program follows:

```
#include <iostream>
#include <string>

using namespace std;

auto main() -> int
{
    auto add_two([](auto x, auto y){ return x + y; });

    cout << add_two("I"s, "t"s) << " works!" << endl;

    cout << "press enter to continue..." << endl;
    cin.ignore(10, '\n');
    cin.get();
}
```

Thank you all, peace to all fellow coders, especially Igor Tandetnik.

answered Dec 23 '15 at 18:51

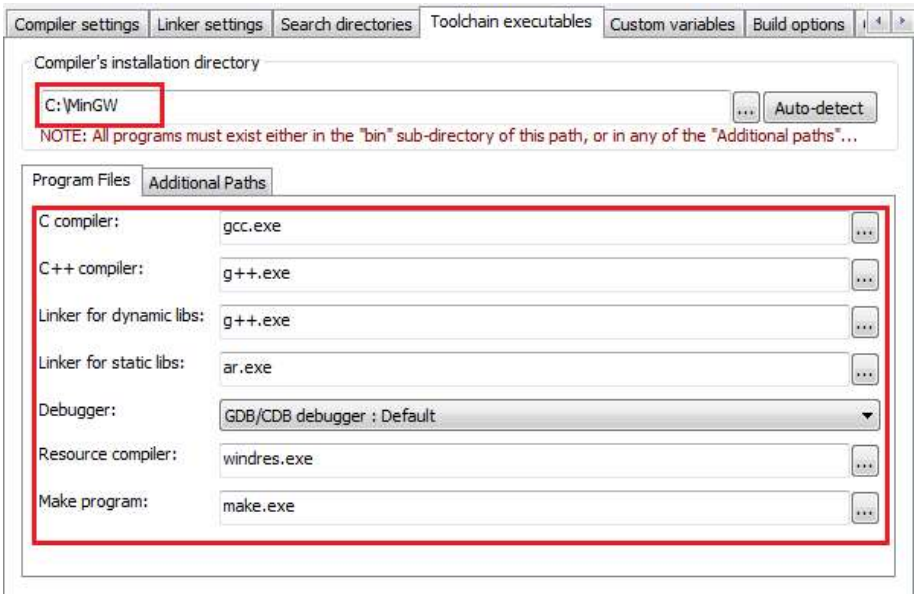
newmanadam

13 4

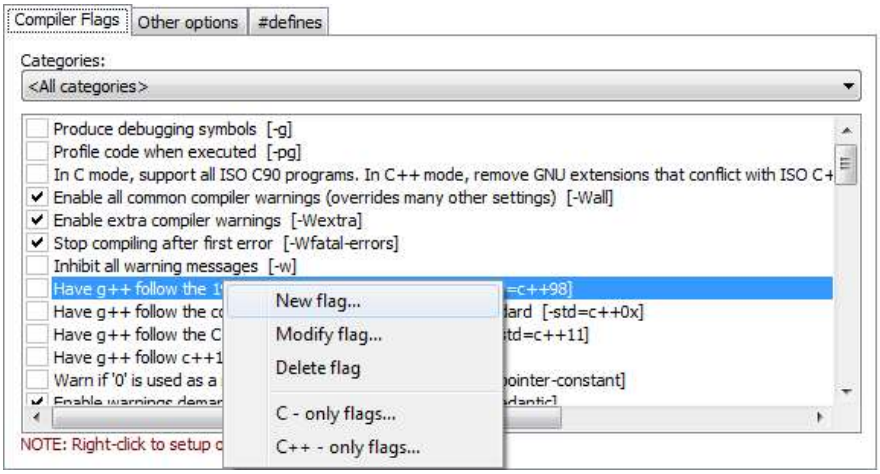
To compile your source code using C++14 in Code::Blocks, you first of all need to download and install a compiler that supports C++14 features.

Here's how you can do it on Windows:

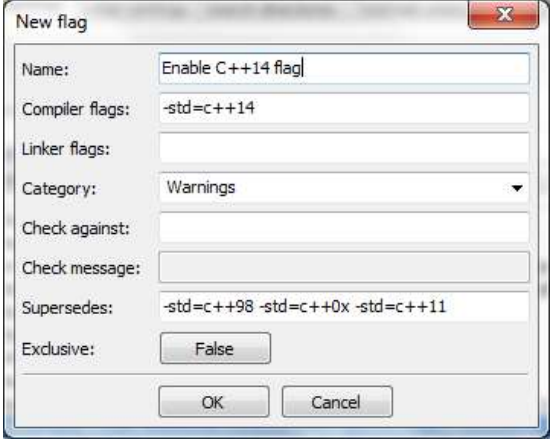
1. Download MinGW from here (particular build) or from official site to choose options
2. Extract it to for example: C:\ (result will be C:\MinGW)
3. Open Code::Blocks
4. Go to Settings => Compiler.
5. Go to "Toolchain Executables".
6. In the top field "Compiler's installation directory", change the directory to the one where you extracted the compiler. E.g C:\MinGW.
7. Change all the necessary files under "Program Files" to match the files under C:\MinGW\bin:



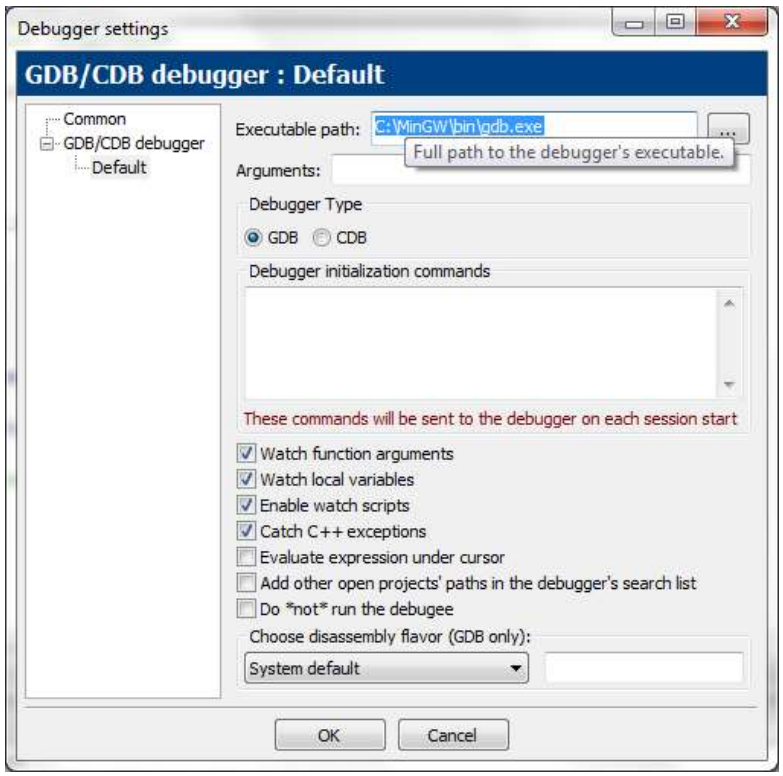
8. Before you hit "OK", go to the leftmost tab "Compiler settings".
9. Select "Compiler Flags".
10. For simplicity, right click in the list somewhere and select "New Flag":



11. Type in the following and click "OK", and tic the box of the flag you just created:



12. Lastly, you need to specify the debugger path. Go to "Settings" => "Debugger", click "Defualt" on left hand side and enter the new full path of the executable:



Now, try to compile a program with C++14 features:

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

auto main() -> int
{
    auto add_two([](auto x, auto y){ return x + y; });

    cout << add_two("I"s, "t"s) << " works!" << endl;
}
```

edited Jul 2 '15 at 11:26

answered Jul 1 '15 at 21:26

Andreas DM

4,890 4 15 39

1 I have absolutely no intention of doing this on any machine I manage, and yet I'm compelled to uptick this. This answer is *Stellar* with a capital-S. – WhozCraig Jul 1 '15 at 22:27

Another thing to do is to right-click in the "Settings > Compiler" page, and add `-std=c++14` to the list of flags excluded from C builds. Otherwise, building a `.c` file will get that flag, causing a warning. Then, you set your project to have both `-std=c++14 -std=c11` . – M.M Jul 27 '15 at 2:50

I followed the procedure, the testing code compiles with no problem. But the following code still causes error `char *m_data = new char[14]{ "Hello, World!" };` this line of code compiles on VS 2015 though. – Chen Nov 13 '16 at 18:48