0.9 — Configuring your compiler: Build configurations

BY ALEX ON FEBRUARY 17TH, 2015 | LAST MODIFIED BY ALEX ON NOVEMBER 17TH, 2019

A **build configuration** (also called a **build target**) is a collection of project settings that determines how your IDE will build your project. The build configuration typically includes things like what the executable will be named, what directories the IDE will look in for other code and library files, whether to keep or strip out debugging information, how much to have the compiler optimize your program, etc... Generally, you will want to leave these settings at their default values unless you have a specific reason to change something.

When you create a new project in your IDE, most IDEs will set up two different build configurations for you: a release configuration, and a debug configuration.

The **debug configuration** is designed to help you debug your program, and is generally the one you will use when writing your programs. This configuration turns off all optimizations, and includes debugging information, which makes your programs larger and slower, but much easier to debug. The debug configuration is usually selected as the active configuration by default. We'll talk more about debugging techniques in a later lesson.

The **release configuration** is designed to be used when releasing your program to the public. This version is typically optimized for size and performance, and doesn't contain the extra debugging information. Because the release configuration includes all optimizations, this mode is also useful for testing the performance of your code (which we'll show you how to do later in the tutorial series).

When the *Hello World* program (from lesson **0.7 -- Compiling your first program**) was built using Visual Studio, the executable produced in the debug configuration was 65kb, whereas the executable built in the release version was 12kb. The difference is largely due to the extra debugging information kept in the debug build.

Although you can create your own custom build configurations, you'll rarely have a reason to unless you want to compare two builds made using different compiler settings.

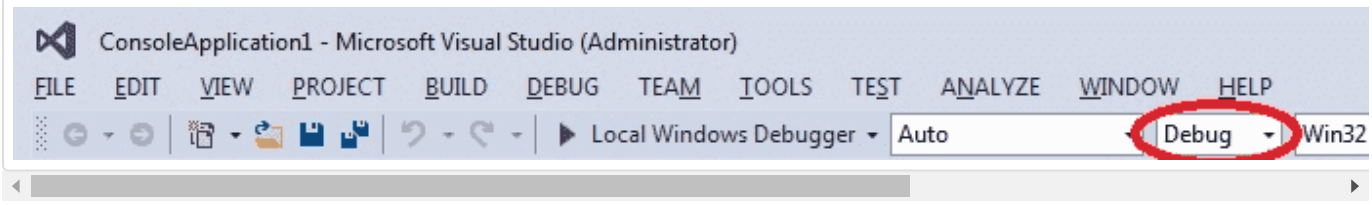
Best practice

Use the *debug* build configuration when developing your programs. When you're ready to release your executable to others, or want to test performance, use the *release* build configuration.

Switching between build configurations

For Visual Studio users

There are multiple ways to switch between *debug* and *release* in Visual Studio. The easiest way is to set your selection directly from the *Solution Configurations* dropdown in the *Standard Toolbar Options*:

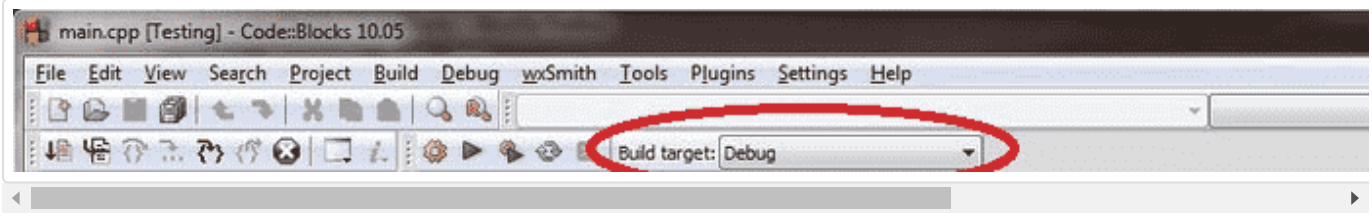


Set it to *Debug* for now.

You can also access the configuration manager dialog by selecting *Build menu* > *Configuration Manager*, and change the *active solution configuration*.

For Code::Blocks users

In Code::Blocks, you should see an item called *Build Target* in the *Compiler toolbar*:



Set it to *Debug* for now.

0.10 -- Configuring your compiler: Compiler extensions

Index

0.8 -- A few common C++ problems

C++ TUTORIAL | PRINT THIS POST

98 comments to 0.9 — Configuring your compiler: Build configurations

« Older Comments 1 2

Shiba
December 26, 2019 at 11:57 pm · Reply

I think it would be great and much appreciated if you guys can update the whole IDE configuration part with Visual Basic Code examples as well. Thank you for all the continuous effort so far.

Inu
February 4, 2020 at 4:56 pm · Reply

I think you meant to say 'Visual *Studio* Code' here. I concur with your suggestion, however, my guess as to the reason they don't have a bunch of different compiler/IDE stuff is because 1) they *do* have the options for a few really popular compilers/IDEs. And 2) I believe (but I'm not sure since I haven't looked ahead) they only talk about the specifics of compiling and IDE stuff in these early lessons. So once you get past them it won't make a difference which compiler/IDE you're using.

nascardriver
February 6, 2020 at 9:08 am · Reply

Visual Studio Code isn't an IDE, it's a text editor. You invoke the compiler manually, in which case you can follow the instructions about gcc. Or you use a plugin to invoke the compiler, then you need to search for instructions of that plugin.

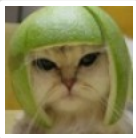
Vitaliy Sh.
December 24, 2019 at 7:07 pm · Reply

The trouble with the images here

When
When a 1680:1050 monitor is split between a web browser (attached left), and a text editor (attached right), i'm experience a trouble.

What
I'm can't see the whole images from s "For Visual Studio users" and "For Code::Blocks users", and also can't scroll them side-wise, to see more.

Q: Can your, please, fix it?
Proposal: <pre> tag (tested on my mobile, no breakage), or cropping.



Alex
January 1, 2020 at 9:36 pm · Reply

I tried to fix it in a slightly different way. Let me know whether it works for you.

Vitaliy Sh.
January 1, 2020 at 9:50 pm · Reply

The new look of the site is great on my halved desktop browser. Wide and convenient. Thanks, sir.
For a quite a few days before, and today: Everything is works.
But the mobile version (~4.7") need to be zoomed a lot, for me to see the diagram's letters. Your rather will be unable to fix it ;)

Vitaliy Sh.
November 18, 2019 at 3:14 am · Reply

There is no reference about using g++'s -g flag here, to match the lesson's 0.7 "If you're using g++ on the command line".

Maybe remove it from 0.7, because:
1) The "Yes, but we don't recommend it for beginners." from 0.6
2) The man for g++ -g looks rather arcane. My IDE default to -g for debug build config, though.
3) Sir nascardriver reply on one of my comments like "there is simplier alternatives to that script, like Cmake"; about *makefile* (which is "build configurations on command line"?) in 0.5: "However, because the makefiles are not part of the C++ core language, nor do you need to use them to proceed, we'll not discuss them as a part of this tutorial series.".

P.S.: In 0.10, 0.11, and 0.12 there is "For GCC/G++ users".

nascardriver
November 18, 2019 at 6:17 am · Reply

There is no -g flag in 0.7's "If you're using g++ on the command line". Are you referring to something else?

Vitaliy Sh.
November 18, 2019 at 8:02 am · Reply

Else, sir.
0.7, 0.10, 0.11, 0.12 all have examples with a g++ on the c-line. That - 0.9 - is have none.

Proposal:
Deletion (all g++ examples) *
||
Addition (g++ -g there).

* Because there is no simple way in spirit of that lessons to mimic the "Build Configurations" without the (C)Makefiles (0.5 - "we'll not discuss them").

nascardriver
November 18, 2019 at 8:08 am · Reply

There are, and should be, examples about how to use g++ on the command line.
There should be no examples about Makefiles.
There is no example about g++ in this lesson, because there are no build-configurations in g++ (Only in build systems). -g is not a build configuration.

Vitaliy Sh.
November 18, 2019 at 8:34 am · Reply

The g++ examples there is to show that it is exist to us? Chapter 3 is "g++"-less.

nascardriver
November 18, 2019 at 8:42 am · Reply

The g++ examples are there so you can use g++.
Chapter 3 is about debugging in an IDE. It's irrelevant which compiler the IDE is using.

Vitaliy Sh.
November 18, 2019 at 8:52 am · Reply

"Yes, but we don't recommend it for beginners.". And me anyway need to read about proper using it elsewhere, because search for "debug" on the main page shows only 3rd chapter. And any proper pupil expected to know how to debug? Thank you for all the reply, sir, but i've not get it in :) (yet?).

Also, "debug" on the Site Index: 0.4, 0.9, 3.6, 7.12a: no "g++". That looks like a unneeded weight for people of my level, at least there.

nascardriver
November 18, 2019 at 9:01 am · Reply

Using a command-line compiler isn't recommended for beginners, but a lot of people want to use a command-line compiler. The g++ instructions aren't perfect, but it's better than not having them at all.
If you're not using a graphical debugger (In an IDE or editor), you should look into gdb (A command line debugger).

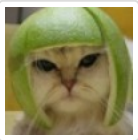
Vitaliy Sh.
November 18, 2019 at 9:15 am · Reply

I am using IDE, sir. The step, next, explore x is to complicated to me for now.

Vitaliy Sh.
November 12, 2019 at 10:35 pm · Reply

… you'll rarely have a reason to unless …

Possible typo: no comma before "unless".



Alex
November 14, 2019 at 7:03 pm · Reply

Thanks for all the great suggestions!

Vitaliy Sh.
November 17, 2019 at 7:56 am · Reply

Hi sir!

The comma is still missed, somehow:
…
reason to unless
…

oxygène
April 8, 2019 at 11:49 pm · Reply

Am I the only one using kdevelop for my first program? It's awesome though, don't forget to install cmake before creating your first project.

Teddy
October 8, 2019 at 2:48 pm · Reply

I started using kdevelop myself to compile some github projects. It has come a long way for sure. The ability to import CMake "project" to create a workspace is very slick. KDE has started porting some apps to Microsoft store so I assume KDevelop will make its way there eventually.

Dan Satt
March 17, 2019 at 2:24 pm · Reply

I'm using the new MSVS 2019 RC, and the default build config is debug x86. I have the option to switch to x64, but I can't particularly think of why I would. Any insight?

nascardriver
March 18, 2019 at 5:59 am · Reply

Hi Dan!

Compiling in x86-64 (x64) allows the compiler to use instructions that are unavailable in IA-32 (x86), potentially speeding up your program.
64bit allows you to access more memory than you could possibly have, while 32 bit is limited to around 4GB.
IA-32 support is decreasing, you might find yourself wanting to use a library which is only available in x86-64.

Vinyamin
January 9, 2019 at 4:57 pm · Reply

Why do I not have the Build menu?

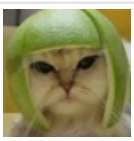
Mark
October 24, 2018 at 12:35 pm · Reply

Since you inserted a new lesson prior to this one (0.8—A few common C++ problems), the following sentence in the fifth paragraph needs to reflect that.

“When the Hello World program from >>the previous lesson<< was built using Visual Studio, …”

Should be corrected to read:

“When the Hello World program from >>lesson 0.7<< was built using Visual Studio, …”

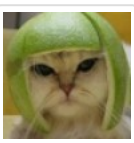


Alex
October 25, 2018 at 12:36 pm · Reply

Indeed. Thanks for pointing this out. Fixed!

Ryan
September 23, 2018 at 12:08 am · Reply

Hello! I believe I found a typo in the bottom section. Where you said 'In Code::Blocks, you should see aitem called Build Target', I think you meant to say 'an item' instead of 'aitem'.



Alex
September 24, 2018 at 7:52 am · Reply

Definitely. Thanks for pointing this out.

Mark
May 17, 2018 at 2:36 pm · Reply

Hi,
I can't find the Build target toolbar, because I am using codeblocks 17.12. Another question that is when and why do you need to use the highlight option? And last question is what are the "match case" and "use regex" options used for?
Thanks.

Willie
March 10, 2018 at 1:30 pm · Reply

Hello Alex.
Is there a easy to read section where the variables are listed in a table form so the values can be seen and whatever else ....possibly how the variables are written?
Trying to make it easy.

Alex
March 11, 2018 at 12:26 pm · Reply

I don't think I understand what you mean. Variables aren't mentioned in this lesson, so I'm not sure what problem you are trying to solve for.

Willie
March 11, 2018 at 1:21 pm · Reply

Hello ,
I wasn't trying to solve or solving anything I'm writing an easy method for reference so I can see tools and how I could use them or more appropriate - how they are used.

Alex
March 11, 2018 at 3:53 pm · Reply

You're looking for an easy way to see all the variables in scope and their values?
If so, then some IDEs offer this as a debugging tool. In debugging mode, Visual Studio's watch windows auto tab does this.

Willie
March 12, 2018 at 1:52 am · Reply

Thanks Alex.

Prynce
March 10, 2018 at 9:46 am · Reply

Hello!!!
Thanks so much I was able to write my first c++ program using Dev c++
"Hello world!" Was displayed on my screen

Prabhat Kumar Roy
December 4, 2017 at 4:59 am · Reply

Dear Sir!

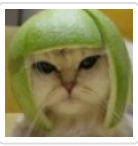
I am revising your lessons after completing the chapter 1 (including its sub-chapters).
I have downloaded and installed Code Block-version 16.01 (Release 16.01 rev 10702 (2016-01-25 19:50:14) gcc 4.9.2 Windows/unicode - 32 bit).

I failed to locate any 'Build target' there even after pasting my 'HellowWorld' program which otherwise runs perfectly.

Can you please guide me in this respect?

Thank you.

Best regards.



Alex
December 6, 2017 at 7:29 pm · Reply

Did you set up both a release and debug release configuration when your project was created? Check out the screenshot here: https://i.ytimg.com/vi/ccYmlFRLCWE/maxresdefault.jpg -- it shows where you should be able to see your active build target on the interface (in this screenshot, the Debug configuration is selected)

Kushal Singh
November 20, 2017 at 10:10 am · Reply

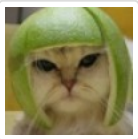
Thanks :)

Kushal Singh
November 8, 2017 at 12:31 pm · Reply

Hello Alex,

When I navigate to Debug dir of my project folder , I find files with extensions: idb,pdb and pch. what are these files for and when are they generated ?

Regards



Alex
November 8, 2017 at 11:54 pm · Reply

.idb and .pdb are debug files produced by Visual Studio. .pch is a precompiled header file. These are all generated when you compile your program in debug configuration with precompiled headers enabled (which they are by default).

Kushal Singh
November 8, 2017 at 9:27 am · Reply

How can I check the size of executable produced in debug/release configuration ?

Alex

November 8, 2017 at 11:53 pm · Reply

Use your OS file explorer/browser and find where your IDE put the executable for each of your configurations. This is most likely in a subfolder inside your project directory.

phong nguyen
June 16, 2017 at 8:44 pm · Reply

Hi, I just want to say thank you for the continuous efforts over the past couple of years. You rock!!!

Ahtazaz Khan
May 3, 2017 at 8:51 pm · Reply

Hi, i'm new here, i love your effort. i'm doing bachelor in Computer science. I have a little bit programming skills.But now i have a aim to learn Programming from the 0.I found your work best.Amazing tutorials.I have a question about building configurations.I have don't clear the build-configuration.Mean,What is the purpose of Build Configuration.How and when debug and release configurations use...?

Alex
May 4, 2017 at 11:10 am · Reply

You'll generally use a debug configuration when developing your code or running it for your own purposes. This enables all debugging functionality, making it easier to find errors. If you decide to release your executable to others, you'll build it using a release configuration. This strips out all debugging information and uses more compiler optimizations, making your code smaller and faster.

You can define custom release configurations as well, but there's no need unless you have special circumstances.

Ahtazaz Khan
May 4, 2017 at 8:50 pm · Reply

Awesome response...i got it... Thanks

« Older Comments 1 2