0.9 — Configuring your compiler: Build configurations

👤 **ALEX** 🕐 **DECEMBER 8, 2022**

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 (https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/)](https://www.learncpp.com/cpp-tutorial/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.

> **For GCC/G++ users**

Add `-ggdb` to the command line when debugging and `-02` `-DNDEBUG` for release builds. Use the former for now.

For VS Code users

When you first ran your program, a new file called tasks.json was created under the .vscode folder in the explorer pane. Open the tasks.json file, find "args", and then locate the line "${file}" within that section.

Above the "${file}" line, add the following args (one per line) when debugging:
`"-ggdb",`

Above the "${file}" line, add the following args (one per line) for release builds:
`"-02",`
`"-DNDEBUG",`

| B | U | URL | INLINE CODE | C++ CODE BLOCK | HELP! |

```
Leave a comment...
```

Name*

Email*

Find a mistake? Leave a comment!

Avatars from https://gravatar.com/ are connected to your provided email address.

Notify me about replies: 🔔   POST COMMENT

133 COMMENTS   Newest ▼