缩进文本或代码

键入文本或代码时，会根据所选文本编辑器或代码样式首选项自动缩进文本或代码。选择一个块以在按 **Tab** 键时缩进它。按 **Shift+Tab** 键可减少缩进。您可以禁用自动缩进。

您可以为以下各项指定缩进：

- C++文件
- 断续器文件
- 尼姆文件
- 其他文本文件

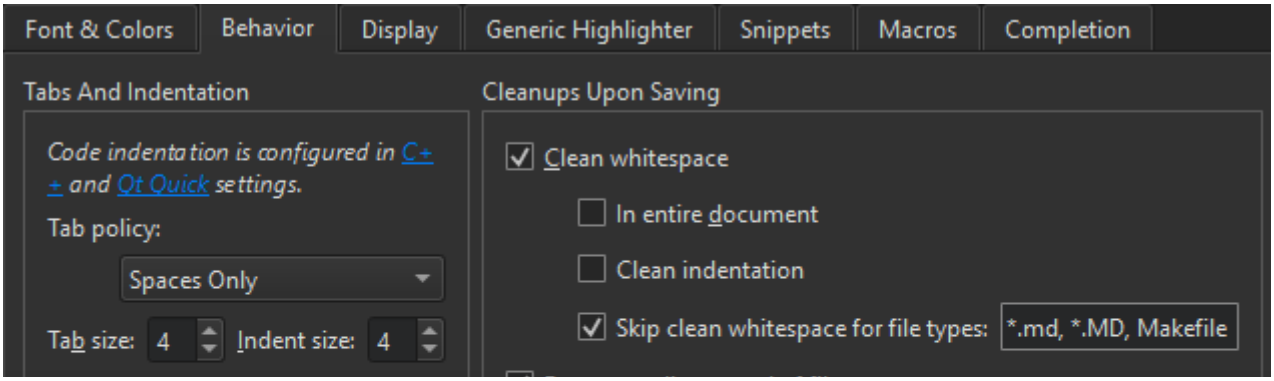
要修复编辑器中当前打开的文件中的缩进，请在"**编辑>高级**"菜单中选择选项，或使用键盘快捷键：

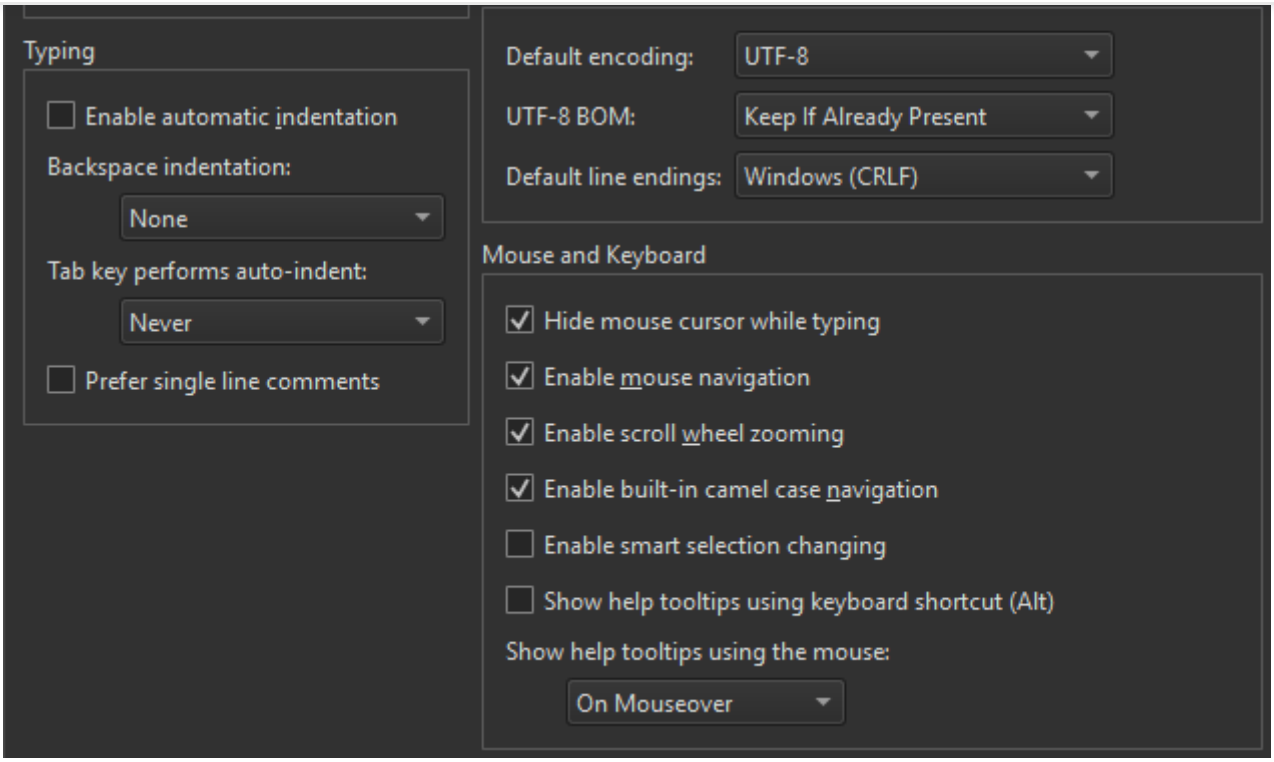
- 若要自动缩进突出显示的文本，请选择"**自动缩进所选内容**"或按 Ctrl+I。
- 若要自动设置突出显示的文本的格式，请选择"**自动设置所选内容的格式**"或按 Ctrl+;。
- 若要调整所选段落的换行，请选择"**重新包装段落**"或按 Ctrl+E，然后按 R。
- 若要切换文本换行，请选择"**启用文本环绕**"或按 Ctrl+E，然后按 Ctrl+W。
- 若要在编辑器中可视化空白，请选择"**可视化空白**"或按 Ctrl+E，然后按 Ctrl+V。
- 若要清除当前打开的文件中的所有空格字符，请选择"**清理空格**"。

指定缩进设置

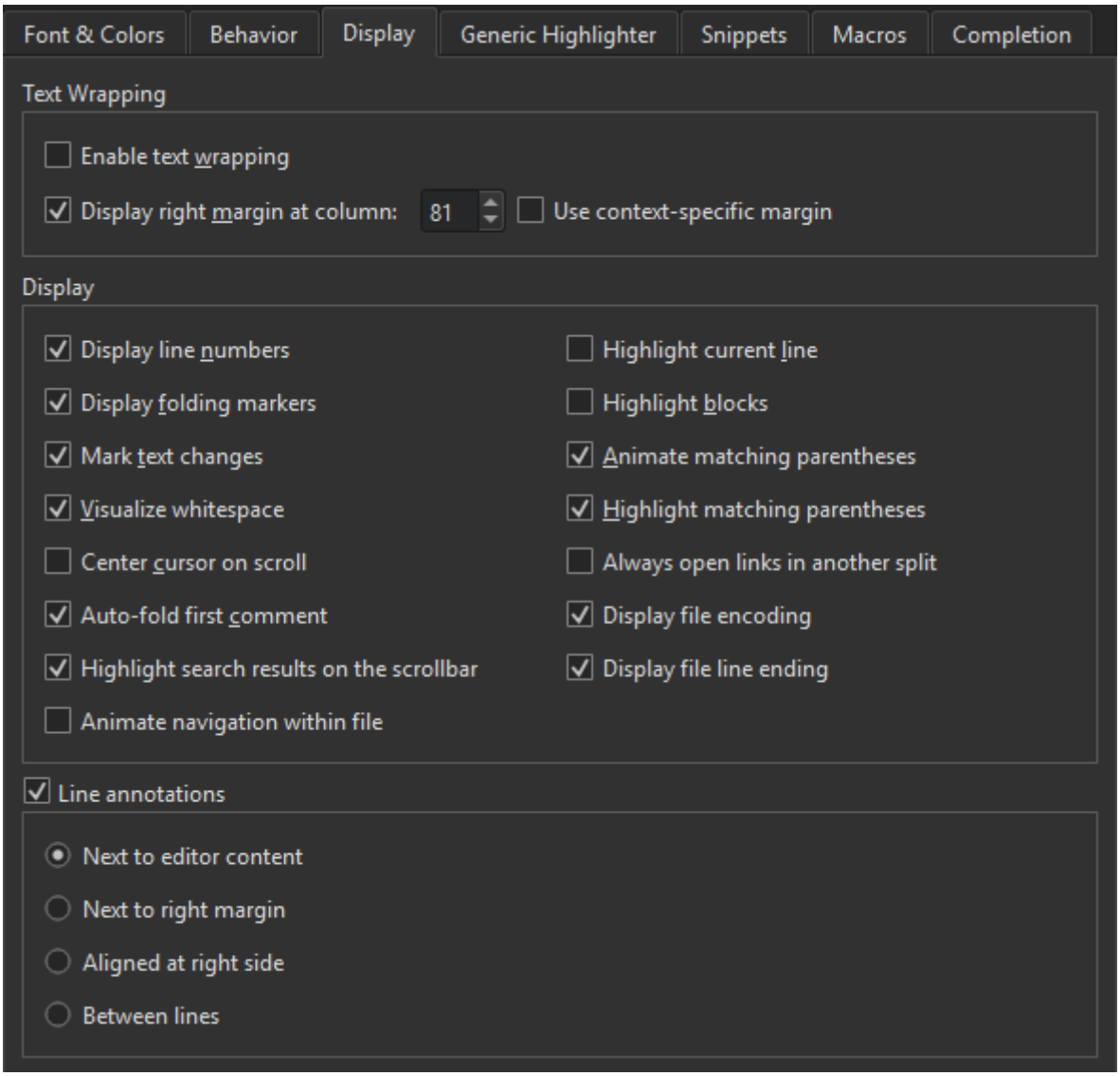
还可以为每个项目单独指定缩进。您可以指定多组代码样式设置，并轻松地在它们之间切换。此外，还可以导入和导出代码样式设置。

若要在保存文件时根据缩进设置自动修复缩进，请选择"**编辑>首选项**">"**文本编辑器>行为>**"**清除空格>清除缩进**"。选中"**跳过文件类型的干净空格**"复选框以排除指定的文件类型。





要在编辑器中可视化空白，请选择**"编辑>首选项">文本编辑器>"显示>可视化空白"**。

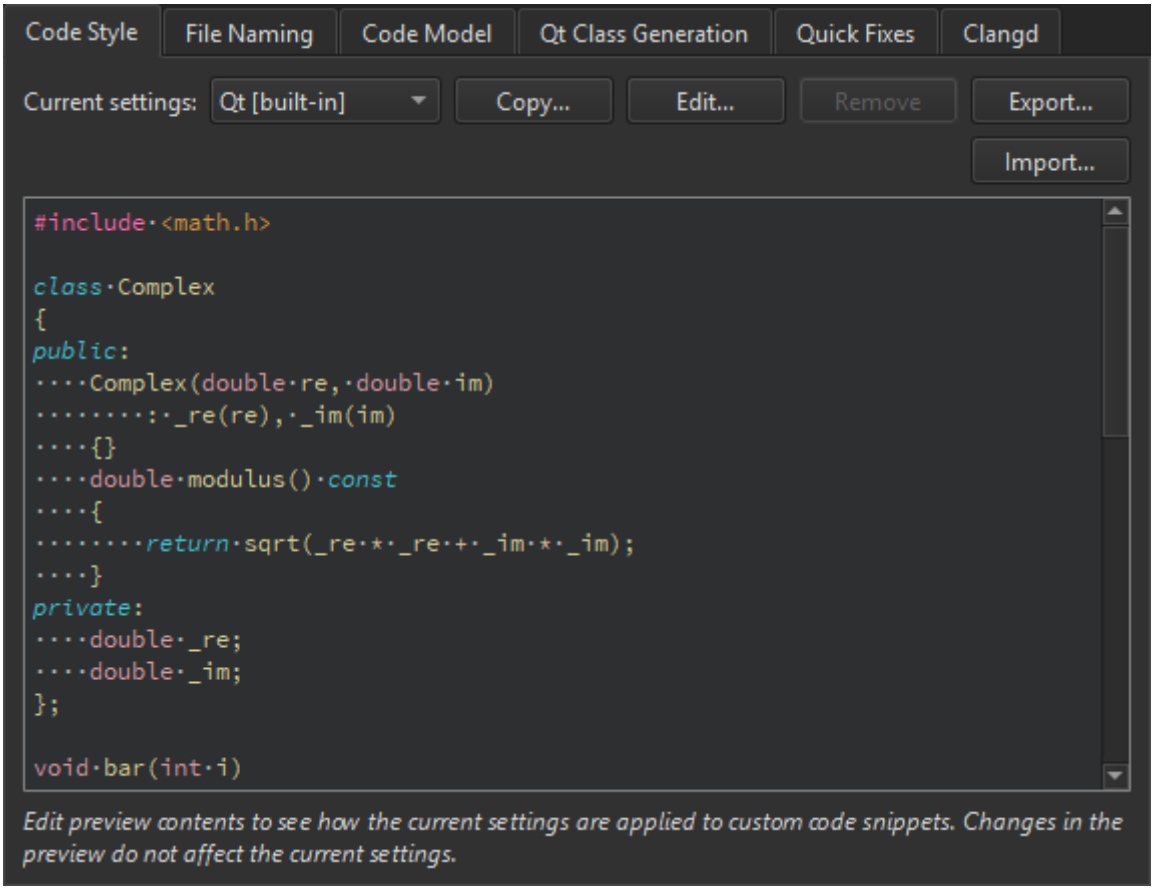


check box. For example, the margin could be set by the option of the Clang Format plugin.`ColumnLimit`

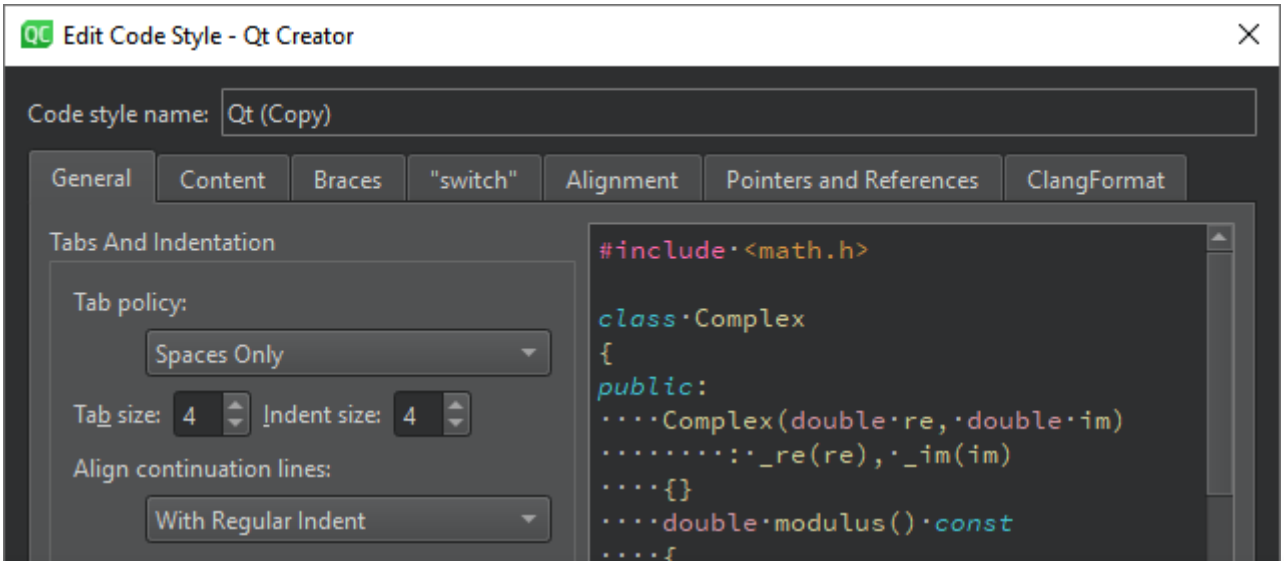
Indenting C++ Files

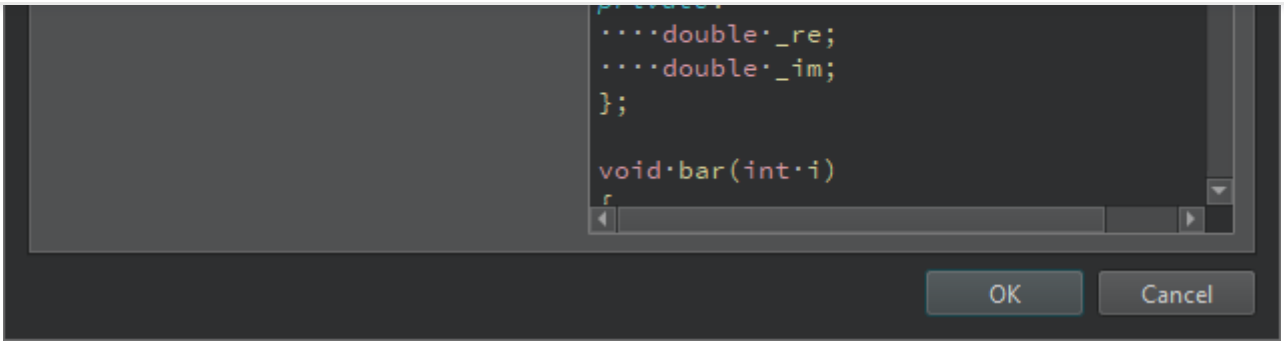
To specify indentation settings for the C++ editor:

1. Select **Edit** > **Preferences** > **C++**.
2. In the **Current settings** field, select the settings to modify and click **Copy**.



3. Give a name to the settings and click **OK**.
4. Click **Edit** to specify code style settings for the project.





You can specify how to:

- Interpret the **Tab** and **Backspace** key presses.
- Indent the contents of classes, functions, blocks, and namespaces.
- Indent braces in classes, namespaces, enums, functions, and blocks.
- Control switch statements and their contents.
- Align continuation lines.
- Bind pointers (*) and references (&) in types and declarations to identifiers, type names, or left or right or keywords.`constvolatile`
- Name getter functions.

You can use the live preview to see how the preferences change the indentation.

To specify different settings for a particular project, select **Projects** > **Code Style**.

Automatic Formatting and Indentation

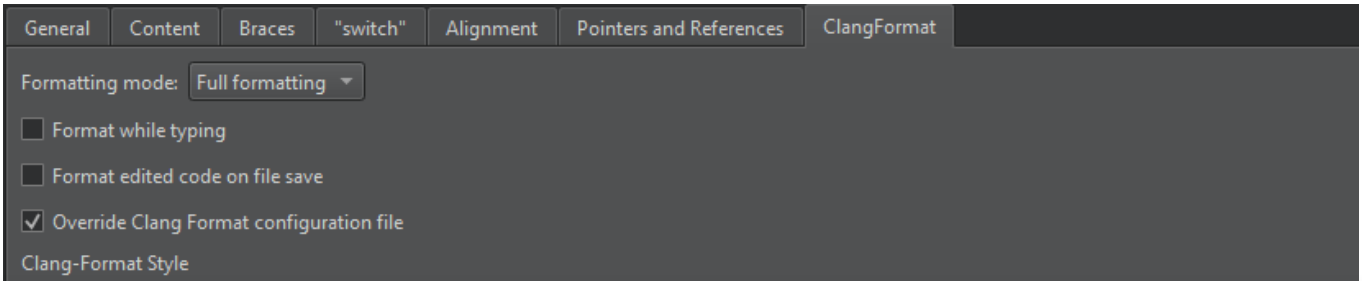
The Clang Format plugin uses the LibFormat library for automatic formatting and indentation.

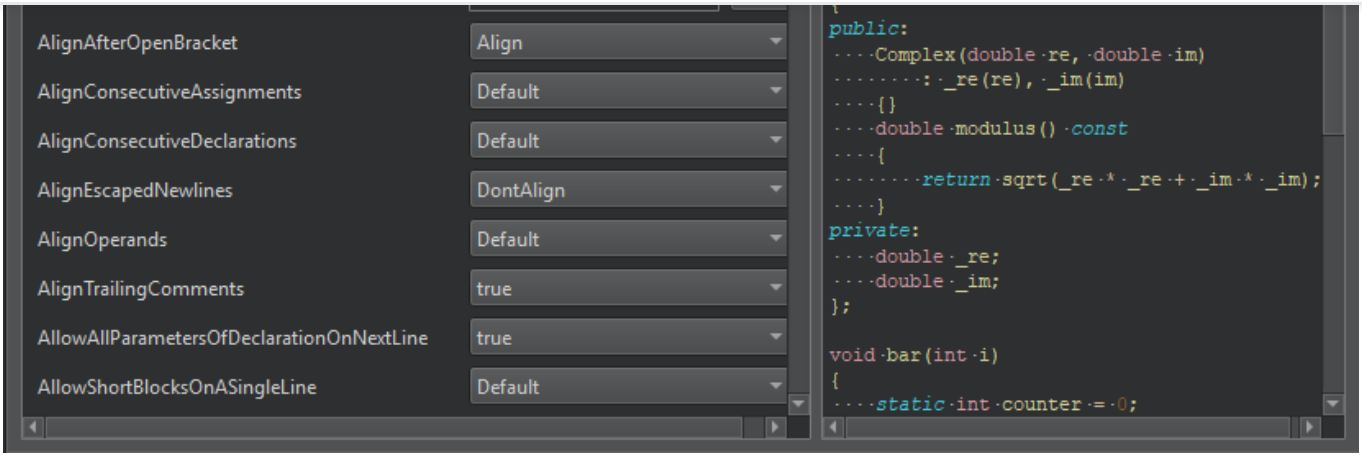
To enable the plugin, select **Help** > **About Plugins** > **C++** > **ClangFormat**. Then select **Restart Now** to restart Qt Creator and load the plugin.

> **Note:** If you enable Clang Format, do not use the Beautifier because combining them can provide unexpected results.

You can use Clang Format to enforce a coding style for a project or the whole organization. Create a file that contains the Clang Format Style Options to use and save it in the root folder of the project or one of its parent folders. The plugin searches for the Clang format file recursively from the directory that contains the source file up to the file system root.`.clang-format`

To override the file globally for all projects, select **Edit** > **Preferences** > **C++** > **Copy** > **Edit** > **ClangFormat** > **Override Clang Format configuration file**.`.clang-format`





In **Formatting mode**, select **Indenting Only** to only indent code. Select **Full Formatting** to use the **Ctrl+I** keyboard shortcut to format code instead of indenting. To apply the formatting while you type, select **Format while typing**. To apply the formatting to the edited code when you save the file, select **Format edited code on file save**.

This creates a local configuration file that overrides the one stored in the file system.

To override the file for a particular project, create a copy of the built-in style and edit its settings by selecting **Projects** > **Project Settings** > **Code Style** > **Copy** > **Edit** > **ClangFormat** > **Override Clang Format configuration file**.`.clang-format`

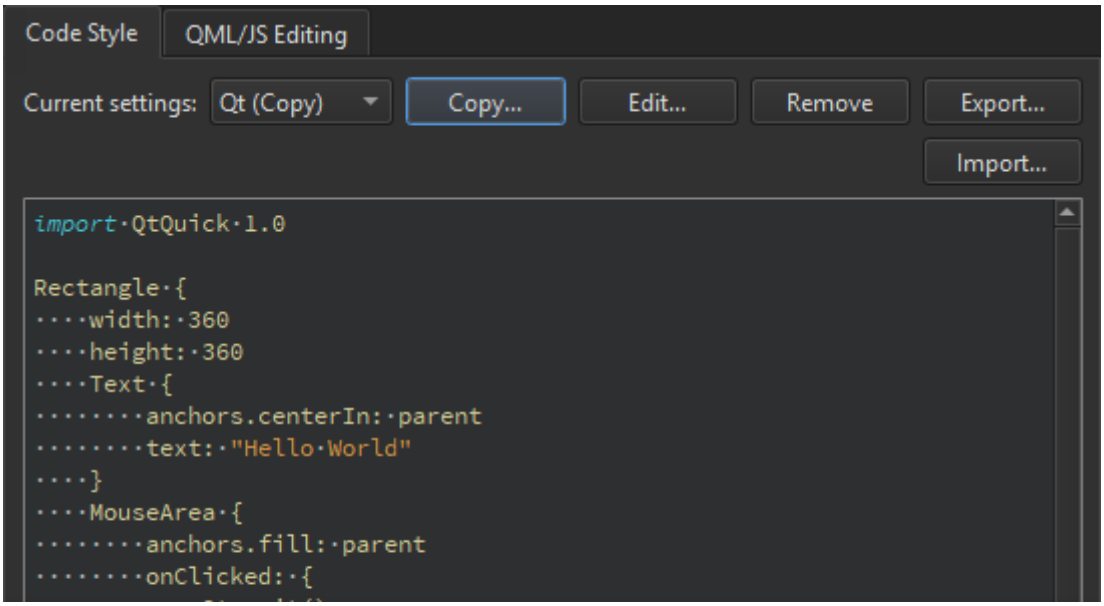
You can create files that contain the configuration options of a certain predefined style from the command line. For example, to create a format file for the LLVM style, enter the following command:`.clang-format`

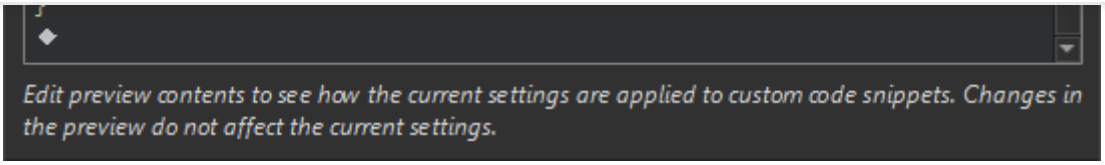
```
clang-format -style=llvm -dump-config > .clang-format
```

Indenting QML Files

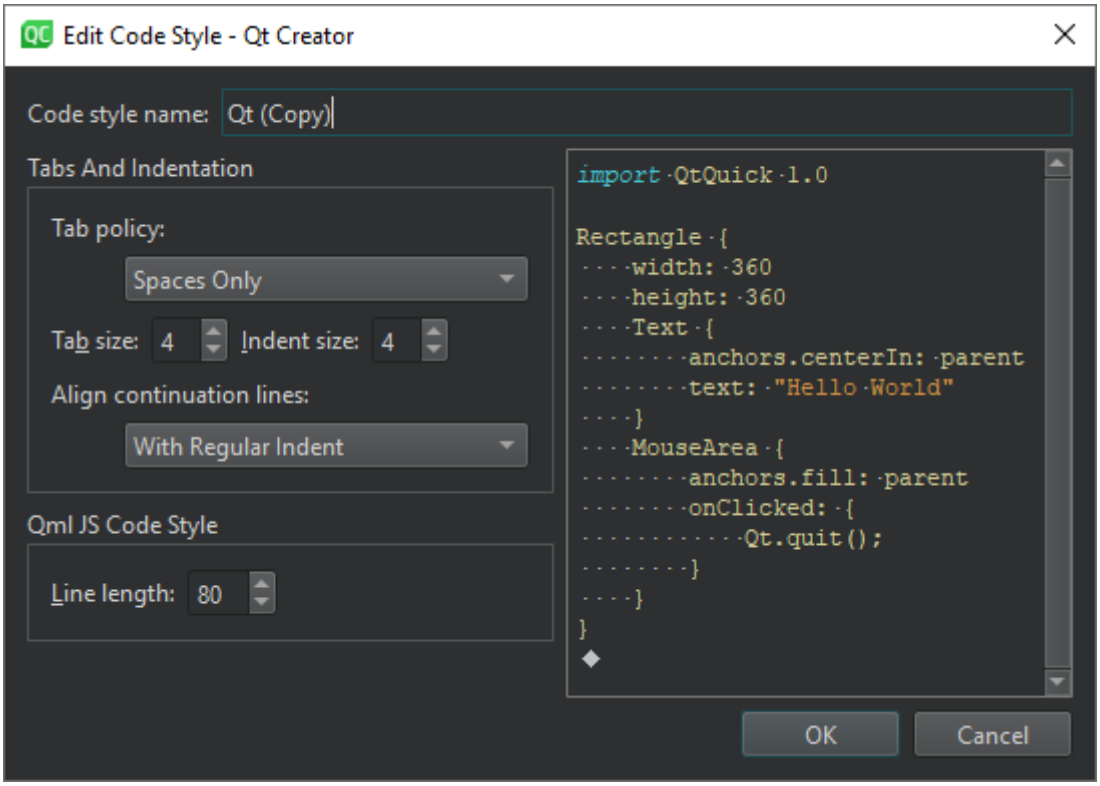
To specify settings for the Qt Quick editor:

1. Select **Edit** > **Preferences** > **Qt Quick**.
2. In the **Current settings** field, select the settings to modify and click **Copy**.





3. Give a name to the settings and click **OK**.
4. Click **Edit** to specify code style settings for the project.



You can specify how to interpret the **Tab** key presses and how to align continuation lines.

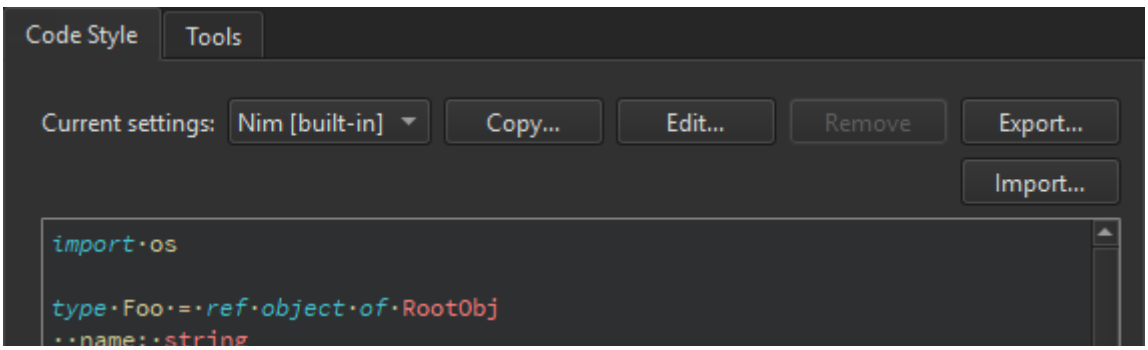
In **Line length**, you can adjust the maximum line length for code lines.

To specify different settings for a particular project, select **Projects** > **Code Style**.

Indenting Nim Files

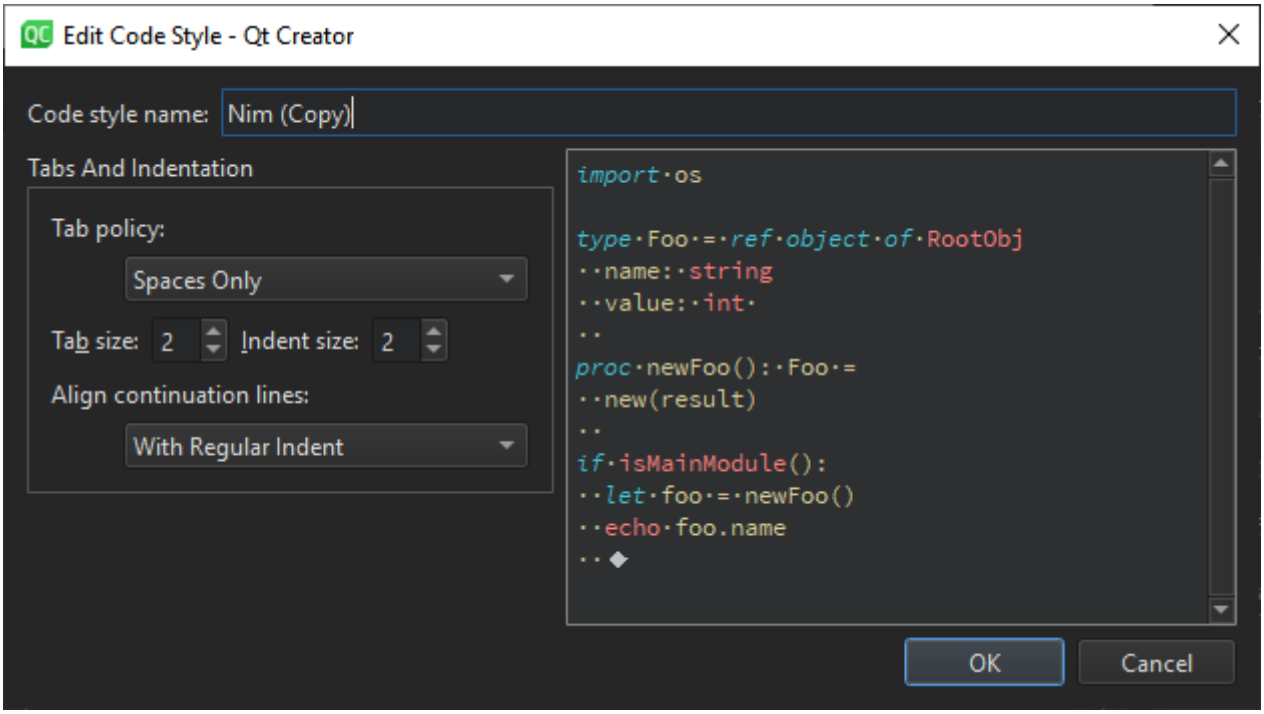
To specify settings for the Nim editor (experimental):

1. Select **Edit** > **Preferences** > **Nim**.
2. In the **Current settings** field, select the settings to modify and click **Copy**.



```
··new(result)
··
if·isMainModule():
··let·foo·=·newFoo()
··echo·foo.name
··◆
```

3. Give a name to the settings and click **OK**.
4. Click **Edit** to specify code style settings for the project.

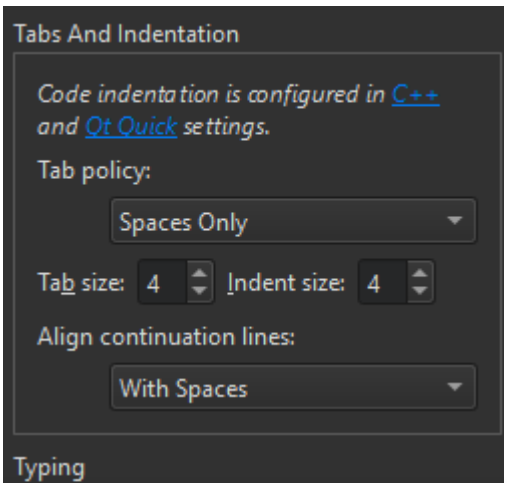


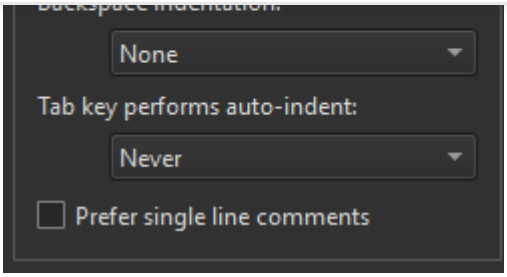
You can specify how to interpret the **Tab** key presses and how to align continuation lines.

To specify different settings for a particular project, select **Projects** > **Code Style**.

Indenting Other Text Files

To specify indentation settings for text files that do not contain C++ or QML code (such as Python code files), select **Edit** > **Preferences** > **Text Editor** > **Behavior**.





To specify different settings for a particular project, select **Projects** > **Editor**.

You can specify how to interpret the **Tab** and **Backspace** key presses and how to align continuation lines.

Specifying Tab Settings

You can specify tab settings at the following levels:

- For all C++ files
- For all QML files
- For all other text files
- For C++ files in a project
- For QML files in a project
- For other text files in a project

Specifying Tabs and Indentation

You can specify tab policy and tab size in the **Tabs and Indentation** group. In the **Tab policy** field, select whether to use only spaces or only tabs for indentation, or to use a mixture of them.

By default, the tab length in code editor is 8 spaces and the indent size is 4 spaces. You can specify the tab length and indent size separately for each project and for different types of files.

You can have continuation lines aligned with the previous line. In the **Align continuation lines** field, select **Not at all** to disable automatic alignment and indent continuation lines to the logical depth. To always use spaces for alignment, select **With Spaces**. To follow the **Tab policy**, select **With Regular Indent**.

Setting Typing Preferences

When you type text or code, it is indented automatically according to the selected text editor or code style preferences. To set typing preferences, select **Edit** > **Preferences** > **Text Editor** > **Behavior** > **Typing**.

To disable automatic indentation, deselect the **Enable automatic indentation** check box.

You can specify how the indentation is decreased when you press **Backspace** in the **Backspace indentation** field. To go back one space at a time, select **None**. To decrease indentation in leading white space by one level, select **Follows Previous Indents**. To move back one tab length if the character to the left of the cursor is a space, select **Unindents**.

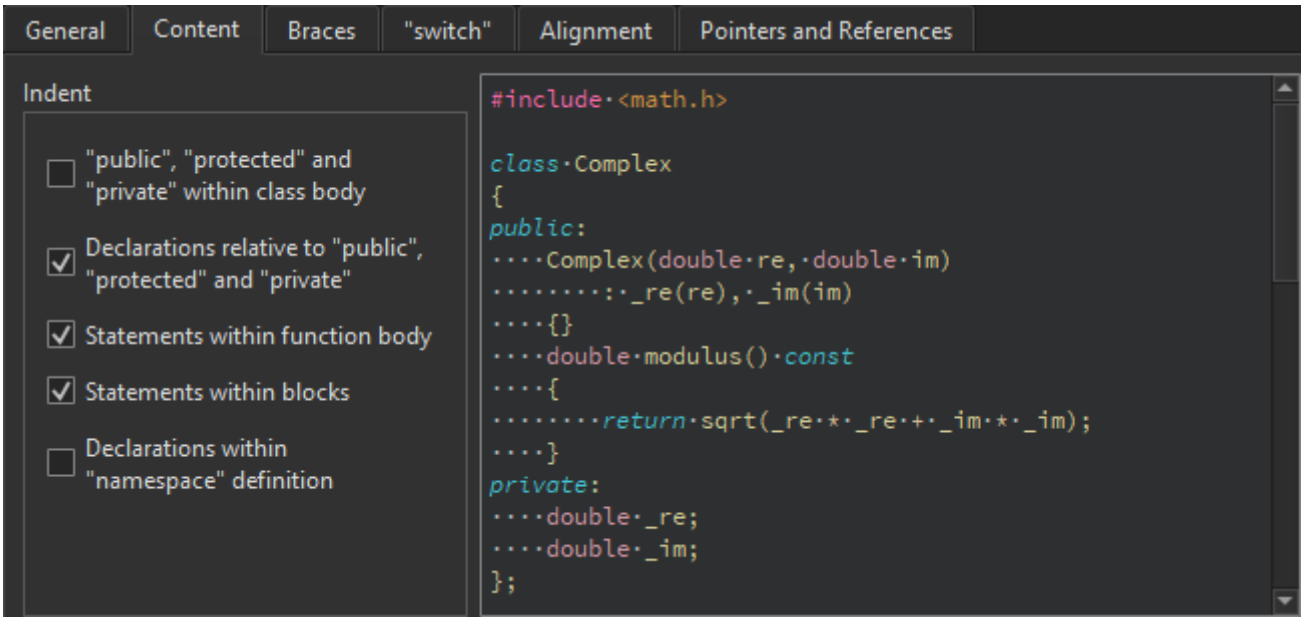
You can specify whether the **Tab** key automatically indents text when you press it. To automatically indent text, select **Always** in the **Tab key performs auto-indent** field. To only indent text when the cursor is located within leading white space, select **In Leading White Space**.

Your highlight definition file can contain definitions for both multi and single line comments. To apply the single line

Specifying Settings for Content

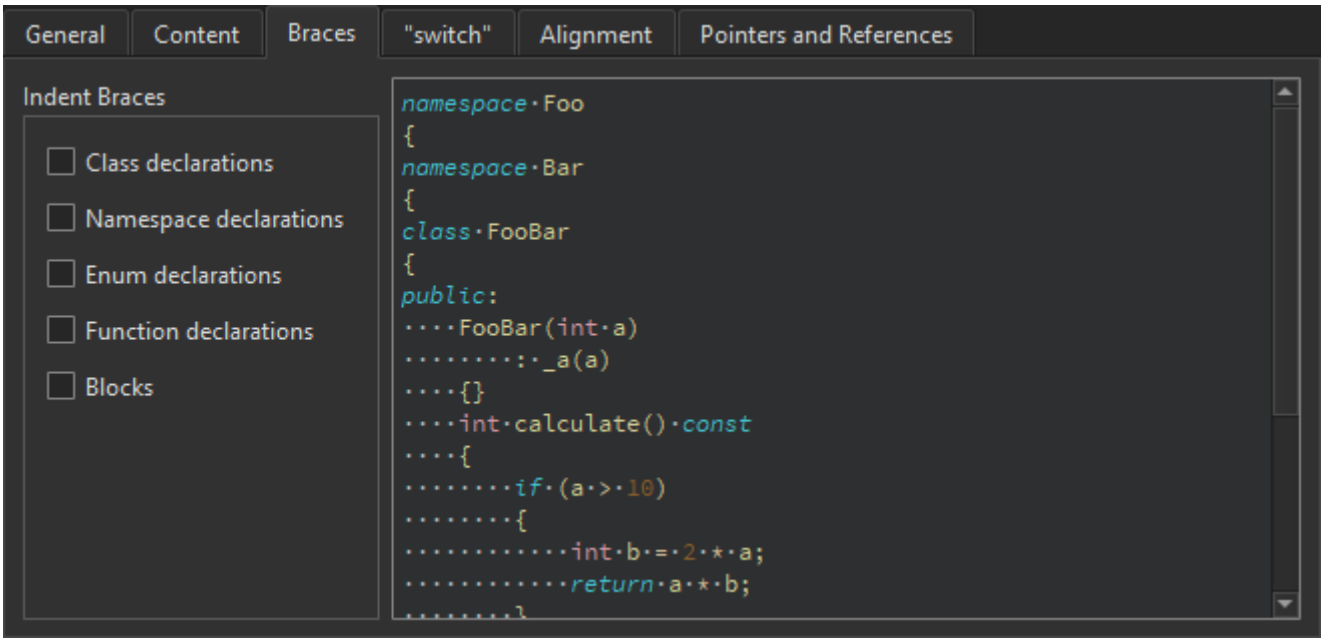
You can indent public, protected, and private statements and declarations related to them within classes.

You can also indent statements within functions and blocks and declarations within namespaces.



Specifying Settings for Braces

You can indent class, namespace, enum and function declarations and code blocks.



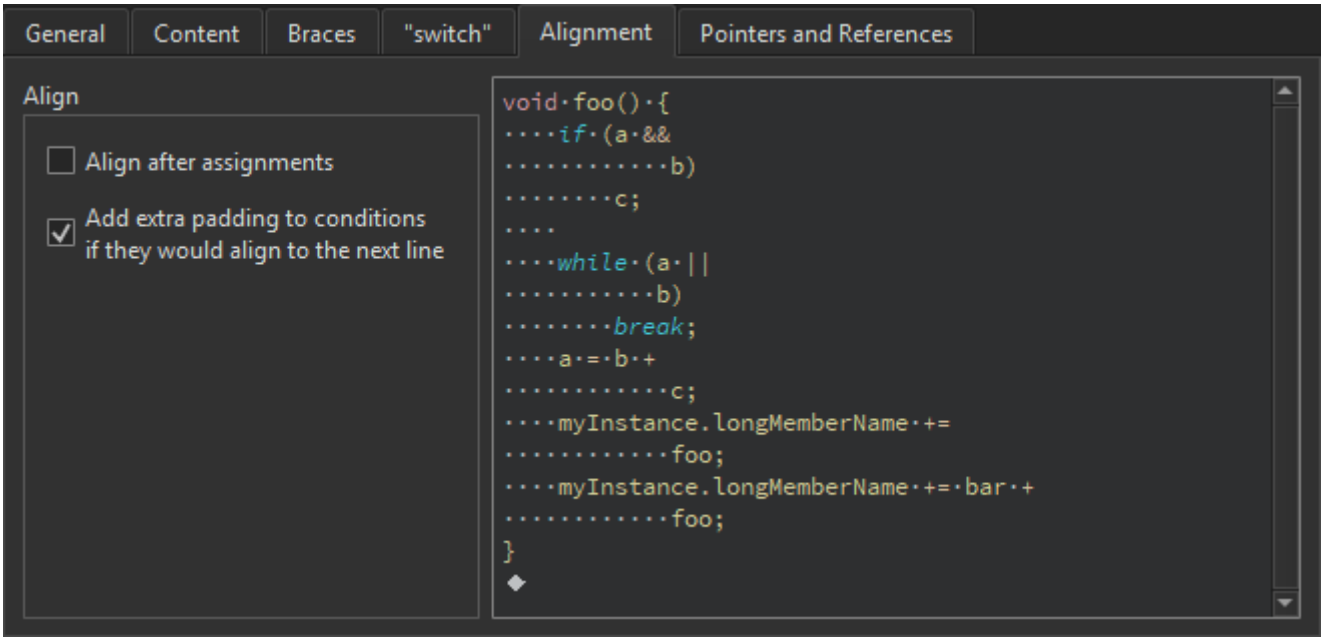
Specifying Settings for Switch Statements

You can indent case or default statements, or statements or blocks related to them within switch statements.

Specifying Alignment

To align continuation lines to tokens after assignments, such as or , select the **Align after assignments** check box. You can specify additional settings for aligning continuation lines in the **General** tab.=+=

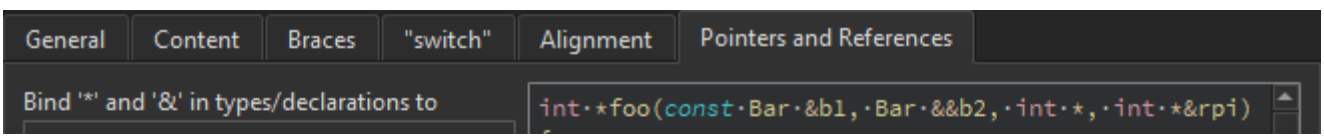
You can also add spaces to conditional statements, so that they are not aligned with the following line. Usually, this only affects statements.`if`

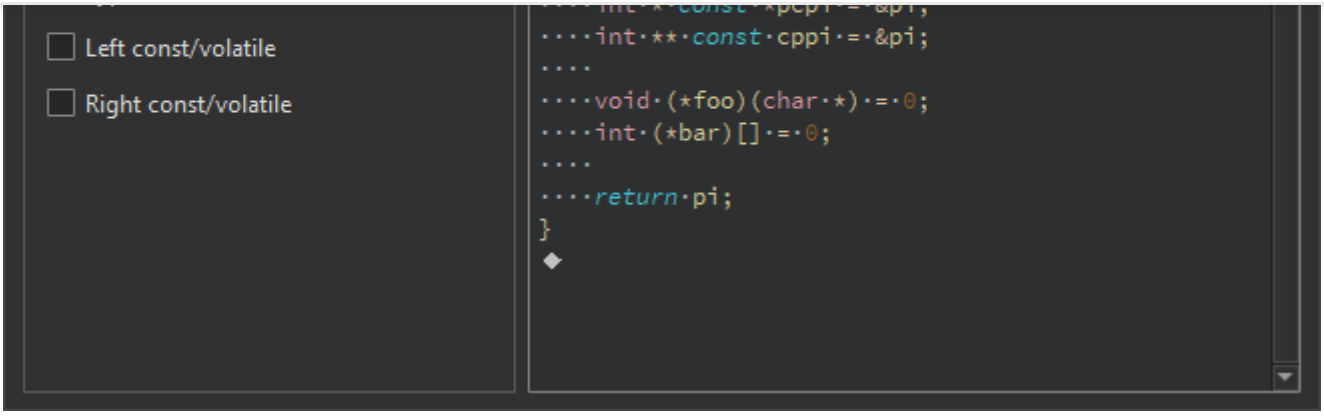


Binding Pointers and References

To bind pointers () and references () in types and declarations to identifiers, type names, or left or right or keywords, select the check boxes in the **Pointers and References** tab.*&`constvolatile`

The and characters are automatically bound to identifiers of pointers to functions and pointers to arrays.*&





‹ Completing Code

Using Qt Quick Toolbars ›