B.1 — Introduction to C++11

BY ALEX ON NOVEMBER 25TH, 2011 | LAST MODIFIED BY ALEX ON JULY 17TH, 2018

What is C++11?

On August 12, 2011, the **ISO (International Organization for Standardization)** approved a new version of C++, called C++11. C++11 adds a whole new set of features to the C++ language! Use of these new features is entirely optional -- but you will undoubtedly find some of them helpful. The prior tutorials have all been updated to be C++11 compliant.

The goals and designs of C++11

Bjarne Stroustrup characterized the goals of C++11 as such:

- Build on C++'s strengths -- rather than trying to extend C++ to new areas where it may be weaker (eg. Windows applications with heavy GUI), focus on making it do what it does well even better.
- Make C++ easier to learn, use, and teach -- provide functionality that makes the language more consistent and easier to use.

To that end, the committee that put the language together tried to obey the following general principles:

- Maintain stability and compatibility with older versions of C++ and C wherever possible. Programs that worked under C++03 should generally still work under C++11.
- Keep the number of core language extensions to a minimum, and put the bulk of the changes in the standard library (an objective that wasn't met very well with this release)
- Focus on improving abstraction mechanisms (classes, templates) rather than adding mechanisms to handle specific, narrow situations.
- Add new functionality for both novices and experts. A little of something for everybody!
- Increase type safety, to prevent inadvertent bugs.
- Improve performance and allow C++ to work directly with hardware.
- Consider usability and ecosystem issues. C++ needs to work well with other tools, be easy to use and teach, etc…

C++11 isn't a large departure from C++03 thematically, but it did add a huge amount of new functionality.

Major new features in C++11

For your interest, here's a list of the major features that C++11 adds. Note that this list is not comprehensive, but rather intended to highlight some of the key features of interest.

- auto (**4.8 -- The auto keyword**)
- char16_t and char_32t and new literals to support them (no tutorial yet)
- constexpr (**2.9 -- Const, constexpr, and symbolic constants**)
- decltype (no tutorial yet)
- default specifier (no tutorial yet)
- Delegating constructors (**8.6 -- Overlapping and delegating constructors**)
- delete specifier(**9.13 -- Converting constructors, explicit, and delete**)
- Enum classes (**4.5a -- Enum classes**)
- Extern templates (no tutorial yet)
- Lambda expressions (no tutorial yet)
- long long int (**2.3 -- Variable sizes and the sizeof operator**)
- Move constructor and assignment (**15.3 -- Move constructors and move assignment**)
- Noexcept specifier (no tutorial yet)
- nullptr (**6.7a -- Null pointers**)
- override and final specifiers(**12.2a -- The override and final specifiers, and covariant return types**)
- Range-based for statements (**6.12a -- For-each loops**)
- r-value references (**15.2 -- R-value references**)
- static_assert (**7.12a -- Assert and static_assert**)
- std::initializer_list (**10.7 -- std::initializer_list**)
- Trailing return type syntax (**4.8 -- The auto keyword**)
- Type aliases (**4.6 -- Typedefs and type aliases**)
- typedef can now typedef template classes

- Uniform initialization (**2.1 -- Fundamental variable definition, initialization, and assignment**)
- User-defined literals (no tutorial yet)
- Variadic templates (no tutorial yet)
- >> will now properly be interpreted as closing a template object

There are also many new classes in the C++ standard library available for use.

- Better support for multi-threading and thread-local storage (no tutorial yet)
- Hash tables (no tutorial yet)
- Random number generation improvements (no tutorial yet)
- Reference wrappers (**12.8 -- Object slicing**)
- Regular expressions (no tutorial yet)
- std::auto_ptr has been deprecated (**15.1 -- Intro to smart pointers and move semantics**)
- std::tuple (quick mention at **7.4a -- Returning values by value, reference, and address**)
- std::unique_ptr (**15.5 -- std::unique_ptr**)


B.2 -- Introduction to C++14


Index


A.3 -- Using libraries with Code::Blocks

Share this:

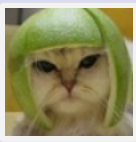
Facebook Twitter Google Pinterest

C++ TUTORIAL | PRINT THIS POST

31 comments to B.1 — Introduction to C++11

DecSco
July 13, 2018 at 7:19 am · Reply

Variatic templates -> Variadic templates



Alex
July 17, 2018 at 3:54 pm · Reply

dhanks for pointing out this error!

😉

Aaron
July 21, 2018 at 6:30 pm · Reply

It's thanks Xd

Siddharth Sharma
March 10, 2018 at 3:33 pm · Reply