

B.1 — Introduction to C++11

👤 ALEX 🕒 OCTOBER 10, 2023

What is C++11?

On August 12, 2011, the [ISO \(International Organization for Standardization\)](https://www.iso.org/home.html) 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 ([10.8 -- Type deduction for objects using the auto keyword](#))
- char16_t and char32_t and new literals to support them (no tutorial yet)
- constexpr ([5.1 -- Constant variables \(named constants\)](#))
- decltype (no tutorial yet)
- default specifier ([14.11 -- Default constructors and default arguments](#))
- Delegating constructors ([14.12 -- Delegating constructors](#))
- delete specifier ([11.4 -- Deleting functions](#))
- Enum classes ([13.4 -- Scoped enumerations \(enum classes\)](#))
- Extern templates (no tutorial yet)
- Lambda expressions ([20.6 -- Introduction to lambdas \(anonymous functions\)](#)) and captures ([20.7 -- Lambda captures](#))
- long long int ([4.3 -- Object sizes and the sizeof operator](#))

- Move constructor and assignment ([22.3 -- Move constructors and move assignment](#))
- Noexcept specifier (quick mention in [27.4 -- Uncaught exceptions and catch-all handlers](#))
- nullptr ([12.8 -- Null pointers](#))
- override and final specifiers([25.3 -- The override and final specifiers, and covariant return types](#))
- Range-based for statements ([16.8 -- Range-based for loops \(for-each\)](#))
- r-value references ([22.2 -- R-value references](#))
- static_assert ([9.6 -- Assert and static_assert](#))
- std::initializer_list ([23.7 -- std::initializer_list](#))
- Trailing return type syntax ([10.8 -- Type deduction for objects using the auto keyword](#))
- Type aliases ([10.7 -- Typedefs and type aliases](#))
- typedef can now typedef template classes
- Uniform initialization ([4.1 -- Introduction to fundamental data types](#))
- User-defined literals (no tutorial yet)
- Variadic templates (no tutorial yet)
- Two >> symbols without a space between them 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 (basic discussion in [8.14 -- Generating random numbers using Mersenne Twister](#))
- Reference wrappers ([25.9 -- Object slicing](#))
- Regular expressions (no tutorial yet)
- std::auto_ptr has been deprecated ([22.1 -- Introduction to smart pointers and move semantics](#))
- std::tuple (no tutorial yet)
- std::unique_ptr ([22.5 -- std::unique_ptr](#))



Next lesson

B.2 [Introduction to C++14](#)



[Back to table of contents](#)



Previous lesson

A.4 [C++ FAQ](#)



Leave a comment...



Name*



Email*




Notify me about replies:



POST COMMENT

🚩 Find a mistake? Leave a comment above!🔗

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

53 COMMENTS

Newest ▾

We and our partners share information on your use of this website to help improve your experience.

Do not sell my info: ☐

OKAY

×