B.1 — Introduction to C++11

ALEX JANUARY 26, 2020

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 (8.7 -- Type deduction for objects using the auto keyword)
- char16_t and char32_t and new literals to support them (no tutorial yet)
- constexpr (4.14 -- Const, constexpr, and symbolic constants)
- decltype (no tutorial yet)
- default specifier (no tutorial yet)
- Delegating constructors (12.8 -- Overlapping and delegating constructors)
- delete specifier (13.14 -- Converting constructors, explicit, and delete)
- Enum classes (9.3 -- Enum classes)

- Extern templates (no tutorial yet)
- Lambda expressions (11.13 -- Introduction to lambdas (anonymous functions)) and captures (11.14 -- Lambda captures)
- long long int (4.3 -- Object sizes and the sizeof operator)
- Move constructor and assignment (M.3 -- Move constructors and move assignment)
- Noexcept specifier (quick mention in20.4 -- Uncaught exceptions and catch-all handlers)
- nullptr (10.9 -- Null pointers)
- override and final specifiers(18.3 -- The override and final specifiers, and covariant return types)
- Range-based for statements (10.19 -- For-each loops)
- r-value references (M.2 -- R-value references)
- static_assert (7.17 -- Assert and static_assert)
- std::initializer_list (16.7 -- std::initializer_list)
- Trailing return type syntax (8.7 -- Type deduction for objects using the auto keyword)
- Type aliases (8.6 -- 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)
- >> 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 in9.5 -- Random number generation)
- Reference wrappers (18.9 -- Object slicing)
- Regular expressions (no tutorial yet)
- std::auto_ptr has been deprecated (M.1 -- Intro to smart pointers and move semantics)
- std::tuple (quick mention at11.5 -- Returning values by value, reference, and address)
- std::unique_ptr (M.6 -- std::unique_ptr)

Next lesson
B.2 Introduction to C++14

Back to table of contents

Previous lesson
A.3 Using libraries with
Code::Blocks

Leave a comment... Put C++ code between triple-backticks (markdown style):```Your C++ coc

Name*

Email*

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

Notify me about replies: POST COMMENT

DP N N FOUT

Newest

X

X