Use of C++11 in Geant4

I. Hrivnacova, IPN Orsay

21st Geant4 Collaboration Meeting,
14 September 2016, Ferrara

Outline

- C++11 in Geant4
- Usage of C++11 features
- Geant4 developers experience
- C++14 hightlights

C++11 As A Revolution

- Lambda expressions - lets you define functions locally, at the place of the call
- Automatic type deduction - you can declare objects without specifying their types
- Rvalue references - can bind to “rvalues”, e.g. temporary objects and literals.
- Smart pointers – no delete
- C++ Standard library - new container classes, new algorithm library and several new libraries for regular expressions, tuples, …
- Threading library – thread class

C++11 in Geant4

- Geant4 10.1 - December, 2014
  - Possibility to select compilation using the C++11 Standard at configuration time for capable compilers.
- Geant4 C++11 task force - since February 2015
  - Identify platforms/compilers which can provide valid support for C++11 features and clarify the level of support
  - Define what will be the set of supported systems/compilers for the next Geant4 release, and which should be definitely dropped
  - Identify a minimal set of C++11 features we want to exploit in the current code
- Geant4 10.2 - December, 2015
  - Compilation using the C++11 Standard is enabled by default, therefore requiring compilers supporting C++ (the list of required features is available in Release Notes)
  - Dropped “old” platforms.

C++11 Guidelines Document

- http://geant4.web.cern.ch/geant4/collaboration/c++11_guide.shtml (link)
- Compiled from the following sources:
  - Effective Modern C++ by Scot Meyers (O'Reilly). Copyright 2015 Scot Meyers. 978-1-491-90399-9.
  - ALICE O² C++ Style Guide.
  - cplusplus.com
  - Stack Overflow
  - *It is using style sheets of C++ Google Style guide, Revision 3.274 (link) under the CC-By 3.0 License further modified by ALICE O² project*

- *Presented in Geant4 Collaboration Meeting in Fermilab*

C++11 Mini-Tutorial

- Presented in Geant4 Collaboration Meeting in Fermilab
  - Parallel session 6B “C++11 Migration”
- The guidelines introduced in the C++11 Guidelines Document with more explanations and examples
- Linked from C++11 Guidelines Document

Mini Survey

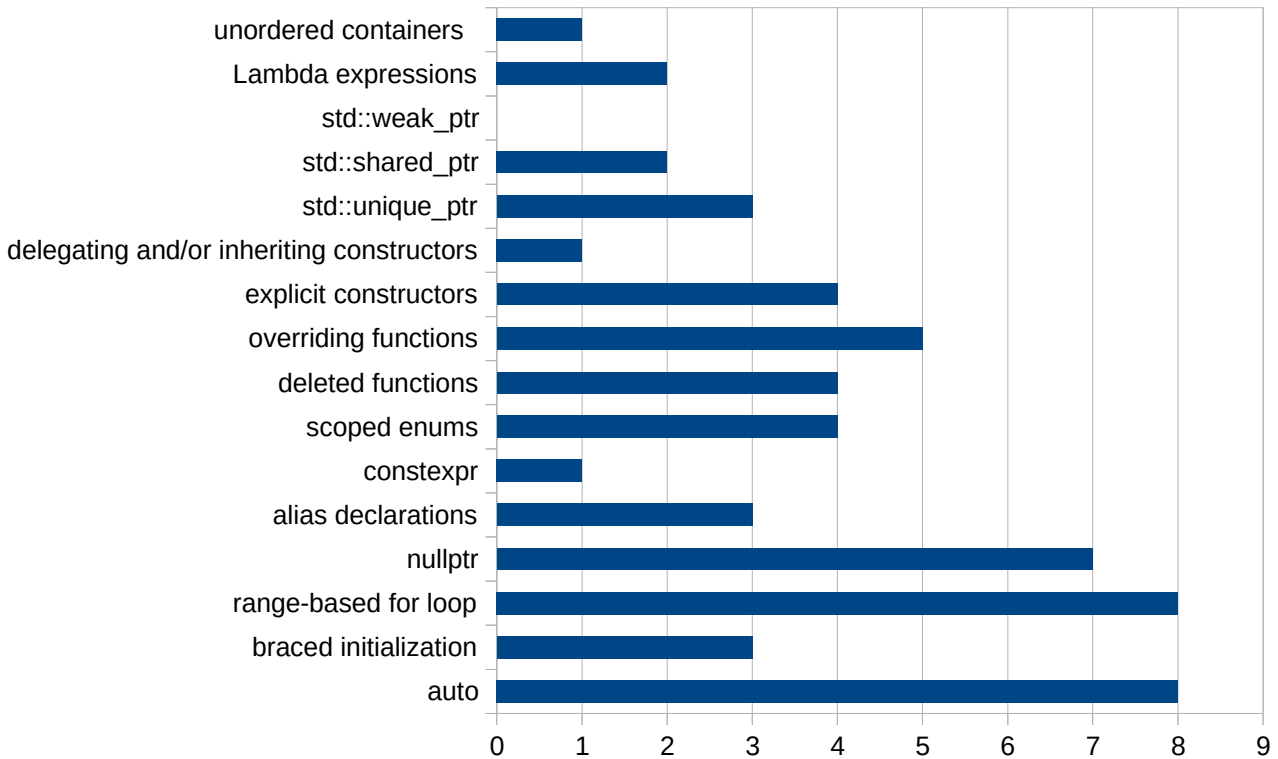
- A questionnaire sent to 11 Geant4 developers (including myself)
  - First to developers who mentionned applying C++11 features in their History file(s)
  - Then to other developers whose names were suggested by the developers asked first
- 8 developers replied:
  - Gabriele Cosmo, Andrea Dotti, Vladimir Ivantchenko, Daren Sawkey, Luciano Pandola, John Alisson, Enrico Bagli, me
  - Thanks to all of them for providing this feedback

Mini Survey (2)

- Did you use the Geant4 C++11 Coding Guidelines document & tutorial?
  - All, except one, replied **YES**
- Did you study C++11 features also from other sources (books, Web sites)
  - All, except one, replied **YES**
  - Books, cppreference website (2x) and stackoverlow
  - Stroustrup's C++11 FAQ was very helpful (2x)
- In which code did you use C++11:
  - Both in newly developed and existing code (4)
  - Only in newly developed code (3),
  - Only in the existing code (1)

Mini Survey (3)

- Which features you use in your code



- I used <chrono> and <thread> to establish the vis thread.
- Also, I used <chrono> for picking in OpenGL. Previously Geant4 just waited for user interaction, consuming 100% CPU. Now it sleeps for 100 ms, not noticeable to the user.
- The CPU usage drops to 1%.

Mini Survey (4)

- Did you get any difficulties with applying some of the guidelines?
- **constexpr**
  - Could be used only since VC13 on Windows was dropped
  - Strongly not recommend in Standard EM – due to known cases in CMS when constexpr kill performance
  - Depends on the context of use, not clear whether “constexpr” was compared to “const” or “static const”
    - Statics get executed only once in your job; const are executed every time the method including them is called…

Personal Comments (1)

- A campain to introduce some (simplest) C++11 features inside EM standard libraries
- **nullptr, delete, override, explicit** - everywere where applicable
- **final** - more delicate we changed our mind from "everywhere" to "in few well defined cases"
- **auto, range base of a loop** - only in few cases, may be in the new code
- **constexpr** - not applicable to physics classes because majority of our constrants are expressions, we still prefer to use "static const G4double" and strongly not recommend constexpr
- **std::unique_ptr** - reqire more code modifications but we would like to consider
- **std::shared_ptr, std::weak_ptr, Lambda expressions, unordered containers -** *seems not applicable or even dangerous*

Personal Comments (2)

- I may say that this helps us to make code more uniform, improve comments, remove few obsolete methods but *do not bring any practical benifits* like improved CPU performance or reduced library size.
- There is a strong requirement that there be no speed degradation in the electromagnetics code so that constrained what C++11 features I could use. These *had to be determined experimentally*.
- Still, I believe we're only scratching the surface of what C++11 provides, given *we're not making use of advanced template programming constructs* in Geant4, *nor we have areas where the design has migrated to make use of such techniques yet*.
- I like C++11 for its new features.

C++14 Highlights

- C++14 is a minor release, featuring mainly bug fixes and small improvements, December 15, 2014
- Some of new features:
  - Function return type deduction
  - Relaxed constexpr restrictions
  - Generic lambdas
  - Variable templates
  - Digit separators in numeric literals
  - The attribute [[deprecated]]
  - std::make_unique can be used like std::make_shared
  - Heterogeneous lookup in standard library associative containers
  - ...

C++14 Highlights (2)

- Compiler support:
  - Clang finished support for C++14 in 3.4 though under the standard name c++1y
  - GCC finished support for C++14 in GCC 5, and made C++14 the default C++ standard in GCC 6
  - Microsoft Visual Studio 2015 has support for some but not all C++14 features