What happened to…: Deprecated and Removed Features of C++

Billy Baker

billy.baker@flightsafety.com

CppCon 2016

FlightSafety international

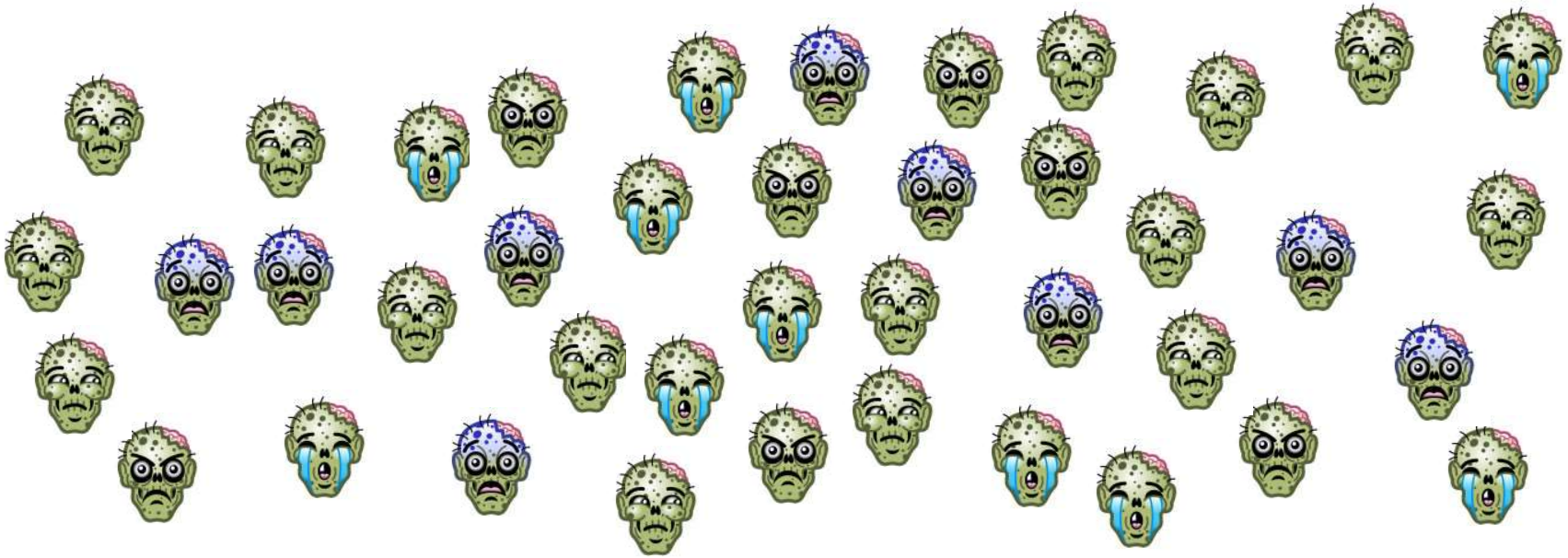
A little background

- Developer of deterministic real-time software for flight simulators
- ISO C++ Committee member
  - LWG, SG14
  - Author of N4168 (removing `auto_ptr`)
- Part-time graduate student (PhD Computer Science)

So far at CppCon 2016

- Bjarne's CppCon 2016 keynote talked about language evolution and compatibility from a 30,000 foot view
- Alisdair Meredith's talk included deprecated and removed features in C++14 and 17 but not in depth

This is not a horror story



http://ralphcosentino.com/project/free-zombie-emoji/

Maintenance can be difficult

- A variety of aviation authorities certify commercial flight simulators
  - FAA (US)
  - EASA (Europe)
  - CAAC (China)
  - DGAC (France)
  - …
- Flight simulators may be in service for 20-30+ years

why doesn't the standard C++ committee push updates more aggressively?

"I am asking myself for a while now: why do we rarely see deprecated features for C++."

youshouldnameit (June 2016)

https://www.reddit.com/r/cpp/comments/4qly1e/why_doesnt_the_standard_c_committee_push_updates

Don’t believe what you read

- From a 2015 non-programming conference paper

  “Recently, C++ exceptions have been deprecated and as such, many C++ projects explicitly recommend not using them.”

- Even though games and some other areas may avoid exceptions, **there are no plans to deprecate/remove exceptions from C++.**

Annex D

Mailing	Content
Pre-Chicago, 2013 N3691 Working Draft	5174 words, 11 sections
Pre-Urbana, 2014 N4140 Working Draft	
Pre-Kona, 2015 N4527 Working Draft	
Post-Oulu, 2016 N4606 Working Draft	

Remove `char* gets(char*)`

- N3733 GB 9
  - C11 no longer defines the dangerous `gets` function
    - Buffer overflows
  - Strike from `<cstdio>`

Remove `char* gets(char*)`

- Resolution
  - Remove `gets` from `std` namespace
  - Added note
    [c.files] [Note: C++ does not define the function `gets`. – end note]

Deprecate `rand` and `random_shuffle`

- N3733 US 21
  - Deprecate items from N3547
- Walter E. Brown's papers
  - N3547 and N3742 proposed deprecating `rand`, `srand`, `RAND_MAX`, and `random_shuffle` as well as adding a few algorithms
  - N3755 created for only the deprecation portion

Deprecate `rand` and `random_shuffle`

- C++11 discouragement

> The `rand` function has the semantics specified in the C standard, except that the implementation may specify that particular library functions may call `rand`. It is implementation-defined whether the `rand` function may introduce data races (17.6.5.9). [Note: The random number generation (26.5) facilities in this standard are often preferable to `rand`. — end note]

Deprecate `rand` and `random_shuffle`

- Resolution
  - Deprecate only `random_shuffle`
  - N3841 and N3924 added additional discouragement for `rand` usage

- Note: Library Fundamentals v2 (N4600) has `std::experimental::randint` as a `std::rand` replacement

Deprecate `rand` and `random_shuffle`

- C++14 discouragement

> The `rand` function has the semantics specified in the C standard, except that the implementation may specify that particular library functions may call `rand`. It is implementation-defined whether the `rand` function may introduce data races (17.6.5.9). [Note: The random number generation (26.6) facilities in this standard are often preferable to `rand`, because `rand`'s underlying algorithm is unspecified. Use of `rand` therefore continues to be nonportable, with unpredictable and oft-questionable quality and performance. — end note]

Remove `char*` streams (`strstream`)

- N3733 CH 9
  - Delete D.7 from the standard
  - Dangerous to use
  - Interface does not meet current requirements

Remove `char*` streams (`strstream`)

- Background
  - Deprecated in C++98
  - No replacement in the standard
- Resolution
  - No changes to Annex D

Deprecate `async`

- N3733 US 26

  “Deprecate `std::async` due to the inability to reconcile the blocking semantics of the destructor of the returned values with the growing expected semantics of `std::future`'s destruction.”

Deprecate `async`

- Background
  - N3777 (H. Sutter)
    - Wording for deprecation
  - N3780 (N. Josuttis)
    - Deprecation is the worst of all options
- Resolution
  - No deprecation
  - Apply N3776 (H. Sutter) wording for `~future`

Annex D

Mailing	Content
Pre-Chicago, 2013 N3691 Working Draft	5174 words, 11 sections
Pre-Urbana, 2014 N4140 Working Draft	5397 words, 12 sections
Pre-Kona, 2015 N4527 Working Draft	
Post-Oulu, 2016 N4606 Working Draft	

CppCon 2014

- Grill the Committee
  - Howard Hinnant wished `auto_ptr` would be removed
  - For the November, 2014 meeting in Urbana
    - N4168 (B. Baker)
    - N4190 (STL)

Remove trigraphs??!

- N3981 and N4086 (R. Smith)

  "Note that the mapping from physical source file characters to the basic source character set is implementation-defined. If trigraphs are removed from the language entirely, an implementation that wishes to support them can continue to do so: its implementation-defined mapping from physical source file characters to the basic source character set can include trigraph translation (and can even avoid doing so within raw string literals). **We do not need trigraphs in the standard for backwards compatibility.**"

Remove trigraphs??!

- N4120 IBM response (M. Wong, et al)

  "In an ASCII world, they are an annoyance because they get quietly replaced in quoted strings, such that strange combinations of leading ?? with any of =,(,),<,>,/,',!,and – can become surprisingly replaced with some single character."

  - See N2910 for technical arguments for trigraphs
  - Clang 3.5, GCC 4.7 (at least), MSVC (VS2010+) default to trigraphs off

Remove trigraphs??!

- IBM response (continued)

  "After significant consultations within IBM, it is IBM's position that for the harmony of the greater C++ community, we will not oppose C++17 because of the removal of trigraphs."

- Resolution
  - Remove trigraphs

Add `uncaught_exceptions`

- N3614, N4152, and N4259 (H. Sutter)
  - See GotW #47 (November 1998)
  - Add `uncaught_exceptions` (plural)
  - LWG asked that `uncaught_exception` (singular) be deprecated
- Resolution
  - Deprecate `uncaught_exception`

Remove deprecated function objects

- N4190 (STL)
  - Remove `unary_function`/`binary_function`
    - Unnecessary given perfect forwarding and `decltype`
    - Class can define `argument_type`, etc typedefs rather than inherit from these function objects
    - Numeric conversion from Boost still using `unary_function` (Paul Bristow, July, 2016)
  - Remove `ptr_fun`
    - Function pointers can be used with `bind` and `function`
  - Remove `mem_fun`/`mem_fun_ref`
    - Superseded by `mem_fn`

Remove deprecated binders

- N4190 (STL)
  - Remove `bind1st`/`bind2nd`
    - Superseded by `bind`

Remove `auto_ptr`

- N4190 (STL) and N4168 (B. Baker)
  - Deprecated in C++11 (N1856 from 2005)
  - Use `unique_ptr`
  - clang-tidy has had a checker for modernization for several years

Remove `random_shuffle`

- N4190 (STL)
  - Deprecated in 2013 for C++14
  - Use `shuffle` in place of `random_shuffle(first, last, rng)`
  - We discourage `rand` usage yet `random_shuffle(first, last)` was permitted to use it

Remove all that N4190 stuff

- Resolution
  - Remove features listed in N4190

“The removal of features was applauded.”

Annex D

Mailing	Content
Pre-Chicago, 2013 N3691 Working Draft	5174 words, 11 sections
Pre-Urbana, 2014 N4140 Working Draft	5397 words, 12 sections
Pre-Kona, 2015 N4527 Working Draft	3605 words, 9 sections
Post-Oulu, 2016 N4606 Working Draft	

Remove `register`

- N4340 and P0001 (A. Meredith)

  "The `register` keyword has no normative function, yet occupies a place in the grammar that must be specified, so seems a good candidate for similar cleanup."

  - Deprecated in C++11
- Resolution
  - Remove `register` and add a note that `register` is reserved for future use

Remove `operator++` for `bool`

- P0002 (A. Meredith)
  - Deprecated in C++98
  - For C++14, Core chair suggested removal
  - C `<stdbool.h>` compatibility concern
    - C++ never supported `operator--` on `bool`

Remove `operator++` for `bool`

- Resolution
  - Remove `operator++` for `bool`

Remove deprecated iostreams aliases

- P0004 (A. Meredith)
  - Deprecated in C++98
    - Old iostreams members [depr.ios.members]
    - `ios_base::io_state/open_mode/seek_dir/streamoff/streampos, basic_streambuf::sbumpc, …`
  - Removal expected to have a lower impact than `auto_ptr`
  - No changes to `strstream`

Remove deprecated iostreams aliases

- Resolution
  - Remove deprecated iostreams aliases

Add `not_fn`

- P0005 (A. Meredith) and P0090 (STL)
  - N4076 added `not_fn`
  - Allows deprecation of `unary_negate`/`binary_negate` and `not1`/`not2`
    - Last components that depend on embedded typedefs
  - Extensive usage experience via Boost

Add `not_fn`

- Resolution
  - Add `not_fun` to C++17
  - Deprecate of `unary_negate/binary_negate` and `not1/not2`

Deprecate adapatable function typedefs

- P0005 (A. Meredith) and P0090 (STL)
  - Remove all uses of `result_type`, `argument_type`, `first_argument_type`, and `second_argument_type`
  - No wait, remove from all but hash and function
  - No wait, remove from all but function
  - Just deprecate

Deprecate adaptable function typedefs

- Resolution
  - Deprecate *weak result type* wording
  - Deprecate `result_type`, `argument_type`, `first_argument_type`, and `second_argument_type` keeping `result_type` in `function`

The zombie clause



- P0005 added [zombie.names]

  "reserving certain names for use by previous standards. This continues to reserve those names for use by the library, meaning that users are not permitted to start using these names as macros. It also allows vendors to remove these identifiers at a time of their choosing, rather than immediately on the release of the new standard. This will mitigate the previous removal of deprecated features like `auto_ptr`."

Deprecate std::iterator

- P0174 (A. Meredith)
  - Standard no longer mandates the use that iterator adaptors derive from `std::iterator`
  - Typedefs provide better clarity than `void` arguments
  - Standard does not use `std::iterator`

Deprecate redundant `std::allocator` members

- P0174 (A. Meredith)
  - Members of `allocator` can be satisfied with members of `allocator_traits<allocator<T>>`
  - `addressof` supersedes `allocator<T>::address`
  - `allocator<void>` does not have `allocate` and `deallocate`

Deprecate is_literal trait

- P0174 (A. Meredith)
  - Not harmful
  - Really want to know if “a specific construction would produce constant initialization”
  - Removal needs a paper to remove *literal type* from the core language

Deprecate temporary buffer APIs

- P0174 (A. Meredith)
  - Lacks exception safety
  - Implementation are no better than new
  - If not deprecated, then design needs to be completed

Deprecate raw storage iterators

- P0174 (A. Meredith)
  - Another incomplete design
  - In 20 years, no papers to complete functionality
  - Not possible to use allocator's `construct`
  - No replacement at this time

Deprecate the P0174 stuff

- Resolution
  - Deprecate `iterator`
  - Deprecate redundant `allocator` members
  - Deprecate `is_literal`
  - Deprecate temporary buffer APIs
  - Deprecate raw storage iterators

Update to C11

- P0063 (C. Nelson, H. Boehm)
  - Remove note on `gets`
- Resolution
  - C++17 refers to C11

Annex D

Mailing	Content
Pre-Chicago, 2013 N3691 Working Draft	5174 words, 11 sections
Pre-Urbana, 2014 N4140 Working Draft	5397 words, 12 sections
Pre-Kona, 2015 N4527 Working Draft	3605 words, 9 sections
Post-Oulu, 2016 N4606 Working Draft	5687 words, 13 sections

Musings on future deprecations and removals

- N4190 (STL) comments on deprecating `bind` in favor of generic lambdas
- How about `<codecvt>`?
- P0408 (P. Sommerlad) might affect `strstream`'s status
- P0003 (A. Meredith)
  - Remove deprecated exception specifications (`throws()`)

Musings on future deprecations and removals

- P0174 (A. Meredith)
  - Deprecate `vector<bool>` specialization
    - Previous attempt to deprecate (N2204)
  - Deprecate algorithms with half an input range
    - Revisit in the future possibly after Ranges TS
  - Replace `value_compare` with unspecified type

Other talks/resources

- Alisdair Meredith – C++17 in Breadth
  - Monday, September 19, 2016, 2:00PM and 3:15PM
- Patrice Roy – The Exception Situation
  - Tuesday, September 20, 2016, 9:00AM
- Walter E. Brown – What C++ Programmers Need to Know about Header <random>
  - Thursday, September 22, 2016, 2:00PM
- Jason Turner – Language Features Removed from C++17
  - C++ Weekly Episode #26

Thank you

- Do you agree with youshouldnameit?

  "I have noticed the progression in deprecating old features, which is a good thing. I still feel like the language could be a lot easier at its core, while still keeping the possibility to do complex things."

- Questions?