C++ Club

11 August 2017

Post-Toronto mailing

ISO C++ blog post

Uses and abuses of metaclasses

Proposal PDF | Reddit post

- `$unfriendly`: prohibit friends
- `$memorise`: remember last N values of each member variable
- `$packed`: reorder member variables for optimal packing
- `$proxy`: proxy each member variable to network
- `$serializable`: add a version number and `serialize()` function
- QT support without MOC
- No need for SWIG anymore! Generate interop code automatically.
- `$mock`: create a mock implementation of every virtual function
- `$json`, `$xml` for automatic deserialization of JSON/XML
- `$table`, `$model`: describe database tables/ORM

P0696R1: Remove abbreviated functions and template-introduction syntax from the Concepts TS

Link

> *<…> the features currently defined in the Concepts TS raise concerns regarding mutation of code undergoing maintenance and difficulties programmers may face in navigating the differences in requirements and behaviors exhibited by functions and function templates that are made more subtle by the current design.*

P0726R0: Does the Concepts TS Improve on C++17?

Link | Reddit thread

- One of the primary goals of the C++ Concepts TS is to provide better template error messages. <...> this paper evaluates the error messages for a few simple STL use cases. The results are surprising, and not encouraging.

P0726R0: Does the Concepts TS Improve on C++17?

Link | Reddit thread

- One of the primary goals of the C++ Concepts TS is to provide better template error messages. <...> this paper evaluates the error messages for a few simple STL use cases. The results are surprising, and not encouraging.
- The Concepts TS is also intended to simplify Generic Programming, but in practice, concept requirements are far from simple. The Ranges TS <...> demonstrates the challenges of making concepts work in the real world.

P0726R0: Does the Concepts TS Improve on C++17?

Link | Reddit thread

- One of the primary goals of the C++ Concepts TS is to provide better template error messages. <…> this paper evaluates the error messages for a few simple STL use cases. The results are surprising, and not encouraging.
- The Concepts TS is also intended to simplify Generic Programming, but in practice, concept requirements are far from simple. The Ranges TS <…> demonstrates the challenges of making concepts work in the real world.
- Finally, C++17 already has the tools to express requirements on templates, check valid syntax, and compose those checks into reusable entities, providing most of the features of concepts without new language extensions. The Concepts TS adds significant complexity and little value to C++17. Until demonstrable benefits outweigh the costs, it should not be merged.

Switch the Ranges TS to Use Variable Concepts

Link

> *The reasons for dropping function-style concepts from the Ranges TS then are: * To eliminate syntactic noise. * To eliminate user confusion that comes at a concept's point-of-use, where sometimes parens are needed (in requires clauses) and sometimes not (when the concept is used as a placeholder). * To avoid depending on a feature that may get dropped. * To avoid becoming a reason to keep a little-loved feature.*

The long-arrow operator in C++

Post

```cpp
template <typename T>
class wrap {
public:
    T* operator->()     { return &t; }
    T& operator--(int) {  return t; }

private:
    T t;
};

wrap<wrap<std::string>> wp1;
wp1--->length();

wrap<wrap<wrap<std::string>>> wp2;
wp2----->length();
```

Cfront rel. 1

Source