



Современный C++

Обзор C++17. Часть вторая.



Меня хорошо видно && слышно?



Ставим "+", если все хорошо
"-", если есть проблемы

Тема вебинара

If constexpr. Structured bindings. Statements with initializer. `std::filesystem`



Владимир Щерба

C++/Rust Backend, Quadrivium

Telegram: @harm



План вебинара

Constexpr If

Inline variables

Structured bindings

If with init

std::filesystem

Sequence points

constexpr If

- Проблема:
 - Мы хотим ветвления на этапе компиляции
- Решения до C++ 17:
 - SFINAE – тяжело читать и писать, вообще не похоже на императивное программирование
- Чего мы хотим:
 - Ветвления как в обычном императивном коде, но на этапе компиляции
- Решение:
 - `constexpr if`

Inline variables

- Проблема:
 - Мы хотим писать header-only библиотеки
- Решения до C++ 17:
 - Использовать другие механизмы кроме static переменных, мириться с дублированием данных в разных единицах трансляции
- Чего мы хотим:
 - Просто определять static-поля в заголовках
- Решение:
 - inline

Structured bindings

- Проблема:
 - Часто нужно раскладывать составные типы на отдельные переменные
- Решения до C++ 17:
 - `std::tie`
 - Просто делать это вручную
- Чего мы хотим:
 - Поддержки со стороны компилятора для этого действия
- Решение:
 - Structured bindings
 - <https://cppinsights.io/s/596636c0>

If with init

- Проблема:
 - Часто переменные объявляются только для использования в условии + теле if'a
- Решения до C++ 17:
 - Объявлять переменные в более широкой области видимости
 - Вручную делать отдельный блок
- Чего мы хотим:
 - Удобный и читаемый способ объявлять переменные, которые видно только в условии и теле if'a
- Решение:
 - If with init
- Также относится к switch (но не к while)

std::filesystem

- Проблема:
 - В STL нет библиотеки для работы с файловой системой, приходится отдельно добавлять зависимости для этого
- Решения до C++ 17:
 - Использовать внешние зависимости
- Чего мы хотим:
 - Инструментария “из коробки” для простых операций с файловой системой
- Решение:
 - std::filesystem - <https://en.cppreference.com/w/cpp/filesystem>

Sequence point

- Проблема:
 - В целях оптимизации компилятор может менять порядок выполнения некоторых операций с побочными эффектами, что приводит к UB в неинтуитивных местах
- Решения до C++ 17:
 - Быть внимательным и использовать статические анализаторы
- Чего мы хотим:
 - Более интуитивных правил в стандарте
- Решение:
 - Более строгие правила точек следования в C++ 11 и C++ 17

Sequence point

- value computation – вычисление значения выражения
- side-effect – изменение объекта, вызов системной функции (например, для записи в файл)

Pre-C++-11

- sequence point – момент в последовательности выполнения, когда все побочные эффекты предыдущих вычислений завершены, а новые еще не начались
 - В конце каждого полного выражения (обычно разделены ;)
 - Между вызовами функций
 - Между вычислением всех аргументов функции и ее вызовом
 - Между операндами операторов && || ?: ,
- Правила неопределенного поведения:
 - Изменение значения между двумя точками следования
 - Использование значения измененного между точками значения объекта

Pre-C++

```
i = ++i + i++;      // undefined behavior
i = i++ + 1;        // undefined behavior
i = ++i + 1;        // undefined behavior
++ ++i;             // undefined behavior
f(++i, ++i);        // undefined behavior
f(i = -1, i = -1);  // undefined behavior
```

```
cout << i << i++;  // undefined behavior
a[i] = i++;        // undefined behavior
```

C++ 11

- Sequence points заменены на отношение “sequenced before/after”
- Если A “sequenced before” B, вычисление A завершится до начала вычисления B
- Если A не следует перед B и B не следует перед A, то либо они будут вычислены в произвольном порядке но не будут пересекаться (indeterminately sequenced), либо либо могут пересекаться (unsequenced)
- Правила неопределенного поведения:
 - Если два побочных эффекта на одном участке памяти unsequenced
 - Если побочный эффект на участке памяти unsequenced относительно вычисления значения использующего этот участок памяти

Правила определения последовательности

1. Полное выражение *следует перед* следующим полным выражением
2. Вычисление значения (но не побочные эффекты) операндов оператора *следует перед* вычислением значения результата оператора
3. При вызове функции вычисление значения и побочные эффекты аргументов следуют перед телом функции
4. Вычисление значения встроенного пост-инкремента и -декремента следует перед его побочным эффектом
5. Побочный эффект пре-инкремента и -декремента следует перед его вычислением значения
6. ...и еще множество правил, полный список есть тут:
https://en.cppreference.com/w/cpp/language/eval_order

Что добавили в C++17 (неполный список)

1. Вычисление значения и побочные эффекты каждого параметра функции неопределенно следуют относительно любого другого параметра (т. е. если логика кода зависит от порядка аргументов, это теперь unspecified а не undefined behaviour)
2. То же самое относится к инициализатору со скобками
3. Перегруженные операторы следуют правилам для встроенных операторов
4. В любом присваивании $E1 @ = E2$ все вычисления и побочные эффекты $E2$ следуют перед вычислениями и побочными эффектами $E1$
5. В операторах сдвига $<<$ и $>>$ левый операнд следует перед правым

Что нам это дает

```
i = ++i + 2;           // well-defined
i = i++ + 2;           // undefined behavior until C++17
f(i = -2, i = -2);     // undefined behavior until C++17
f(++i, ++i);           // undefined behavior until C++17, unspecified after C++17
i = ++i + i++;         // undefined behavior
```

```
cout << i << i++;     // undefined behavior until C++17
a[i] = i++;            // undefined behavior until C++17
n = ++i + i;           // undefined behavior
```

Итоги вебинара

`constexpr If`

`Inline variables`

`Structured bindings`

`If with init`

`std::filesystem`

`Sequence points`



Материалы

- https://en.cppreference.com/w/cpp/language/if#Constexpr_If
- <https://hackernoon.com/a-tour-of-c-17-if-constexpr-3ea62f62ff65>
- <https://habr.com/ru/post/351970/>
- https://en.cppreference.com/w/cpp/language/structured_binding
- https://en.cppreference.com/w/cpp/language/if#If_Statements_with_Initializer
- <https://pabloariasal.github.io/2019/02/28/cpp-inlining/>
- https://en.cppreference.com/w/cpp/language/storage_duration
- <https://en.cppreference.com/w/cpp/language/definition>
- <http://alenacpp.blogspot.com/2005/11/sequence-points.html>
- <https://www.bfilipek.com/2017/08/cpp17-details-filesystem.html>



Делимся впечатлениями

Q&A

**Заполните, пожалуйста,
опрос о занятии
по ссылке в чате**

Спасибо!