C++ 11/14?

Артёмкин Павел
EkbCpp
28.06.2014

Зачем мне это?

Я 16 лет программировал на C++ 98, зачем мне что-то менять?

Зачем мне изучать столько всего нового?

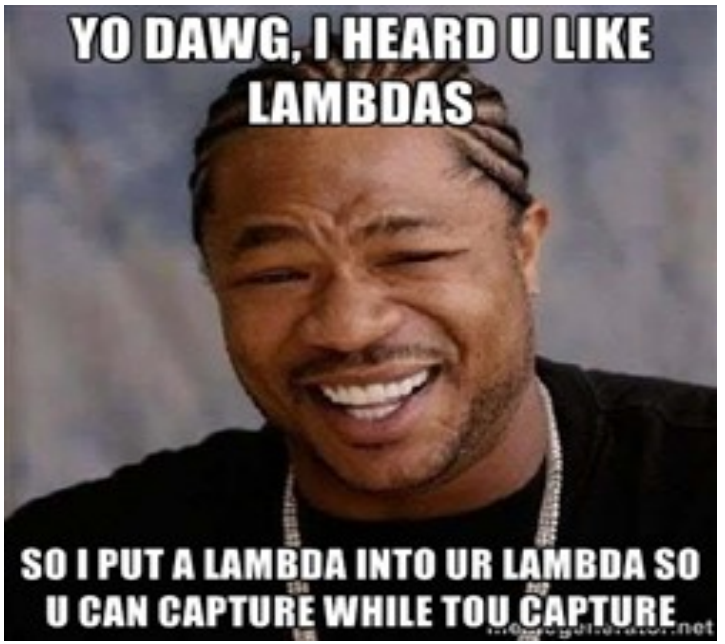
- C++98: 776 страниц.
- C++11: 1353 страницы.
- C++14: 1370 страниц.

Что здесь?

```
1 template <typename T>
2 auto a(T&& t) {
3    auto b = [] (T&& i) {
4          return [i] (const T& j) {
5              return j < i;
6          };
7    };
8    return b(std::forward<T>(t));
9 }
```

C++

```
1  template <typename T>
2  auto make_less(T&& t) {
3     auto less_than = [] (T&& i) {
4           return [i] (const T& j) {
5               return j < i;
6           };
7     };
8     return less_than(std::forward<T>(t));
9  }
10
11 void less_than(int n) {
12    auto less_n = make_less(n);
13    ...
14    std::copy_if(v.begin(), v.end(), std::back_inserter(r), less_n);
15 }
```



Чтобы понимать!

Что надо знать?

- Lambdas
- Auto & decltype
- Variadic templates
- Move-semantic & r-value references

Почему?

- Упрощает часто встречающиеся задачи
- Меньше кода
- Работает по-другому

Задача: вывести все элементы контейнера

std::list<int>

Пример № 1. Проще

Задача: вывести все элементы контейнера.

Решение: Использовать stl функцию для обхода всех элементов контейнера – std::for_each.

```
1 struct print {
2   char sep;
3   print(char s) : sep(s) { }
4   inline void operator () (int x) const {
5     std::cout << x << sep;
6   }
7 };
8
9 std::for_each(c.begin(), c.end(), print('\t'));
```

Пример № 1. Проще

Задача: вывести все элементы контейнера.

Решение: Использовать stl функцию для обхода всех элементов контейнера – std::for_each.

```
1 struct print {
2   char sep;
3   print(char s) : sep(s) { }
4   inline void operator () (int x) const {
5     std::cout << x << sep;
6   }
7 };
8
9 std::for_each(c.begin(), c.end(), print('\t'));
```

Проблемы:

- Надо определять дополнительную структур или функцию.
- Её надо определить в правильном месте (нельзя размещать в текущей функции).
- Коллизии имён.

Пример № 1. Проще

Задача: вывести все элементы контейнера.

Решение: Использовать цикл for.

```
1 for (std::list<int>::iterator i = c.begin(); i != c.end(); ++j) {
2   std::cout << *i << '\t';
3 }
```

Уже лучше, но всё ещё «многабукв».

Пример № 1. Проще

Задача: вывести все элементы контейнера.

Решение: Использовать C++ 11.

```
1 std::for_each(c.begin(), c.end(), [] (int x)
2   { std::cout << x << '\t'; }
3 );
4
5 for (const auto x : c) {
6   std::cout << x << '\t';
7 }
```

Задача: написать меньше кода

Пример № 2. Меньше

Задача: реализовать класс-функтор.

```
 1 template <class F>
 2 class function;
 3
 4 int do(function<int(int,int)> f, int x, int y) {
 5   return f(x,y);
 6 }
 7
 8 int sum(int x, int y) { return x + y; }
 9 int mul(int x, int y) { return x * y; }
10
11 std::cout << do(sum, 3, 5);  // 8
12 std::cout << do(mul, 3, 5);  // 15
```

Пример № 2. Меньше

Задача: реализовать класс-функтор.

Решение: написать реализацю для всех возможных количеств параметров.

```
 1 template <class F>
 2 class function;
 3
 4 template <class R>
 5 class function<R()> { };
 6
 7 template <class R, class P1>
 8 class function<R(P1)> { };
 9
10 template <class R, class P1, class P2>
11 class function<R(P1, P2)> { };
12
13 template <class R, class P1, class P2, class P3>
14 class function<R(P1, P2, P3)> { };
```

Пример № 2. Меньше

Задача: реализовать класс-функтор.

Решение: использовать variadic templates.

```
1 template <class F>
2 class function;
3
4 template <class R, class ... Args>
5 class function<R(Args...)> { };
```

Задача: лучший способ инициализации

Пример № 3. По-другому

Задача: Выбрать наиболее оптимальный вариант для выражения local = f()

```
1 typedef std::list< int > list_t;
2 extern list_t f();
3 list_t local;
4
5 local = f();
```

1. swap(local, f());
2. local = f();
3. f().swap(local);
4. local.swap(f());

Спасибо!

https://www.facebook.com/world.stan

http://www.linkedin.com/pub/pavel-artyomkin/52/b85/b86