C++0x

Елена Сагалаева, ADD-2010

Сроки

- Стандарт 1998-го года
- Стандарт 2003-го года
- C++0x выйдет не раньше 2011. Потому что процесс разработки очень демократичный.
  - 160 активно работающих человек, на каждом собрании ~60
  - Встречи проходят регулярно по всему миру
  - Final Committee Draft (FCD) был принят в марте этого года

Что хотели от C++0x

- Сделать его лучше для системного программирования и разработки библиотек
- Чтобы его было проще учить
- Обратная совместимость (не вводить новые ключевые слова)

От чего отказались

- Сборка мусора
- Концепции (хотя они уже были реализованы в GCC)
- Урезали потоки (threads)

Очень многое было убрано во время «Компромисса в Кóне», октябрь 2007

Рассматриваемые компиляторы

- GCC 4.5.1 (Опция -std=c++0x). Реализовано больше всего.
- Microsoft Visual C++ 2010
- Intel C++ Compiler 11.0 (Опция /Qstd=c++0x)

Двойные угловые скобки

Было:

```
vector<vector<int> > v;
```

Стало:

```
vector<vector<int>> v;
```

√GCC √MSVC++ √Intel

auto

Угадывание типа по инициализации

```
const int* bar() {…}
auto x = 7;
auto num = bar();
```

√GCC √MSVC++ √Intel

auto

Было:

```
for(vector<int>::iterator p=v.begin(); p!=v.end(); ++p)
        cout<<*p<<'\n';
```

Стало:

```
for(auto p=v.begin(); v!=v.end(); ++p)
        cout<<*p<<'\n';
```

√GCC √MSVC++ √Intel

Списки инициализации (initializer list)

Было:

```
vector <string> vMonths;
vMonths.push_back( "January" );
vMonths.push_back( "February" );
vMonths.push_back( "March" );
```

Стало:

```
vector <string> vMonths = { "January",
                            "February", "March" };
```

√GCC ✗MSVC++ ✗Intel

Списки инициализации

```
map <string, int> mFreq =
{  { "cat", 0 },
   { "catch", 0 },
   { "March", 0 }  };
```

√GCC ✗MSVC++ ✗Intel

Лямбды

Лямбда функции - неименованные функции

```
vector<int> v = { 3, 6, -7, 1};
std::sort(v.begin(), v.end(), [](int a, int b) {
                                  return abs(a)<abs(b);
                              });
```

√GCC √MSVC++ √Intel

lambda-capture

```
// ничего не берем
        [](int x, int y) -> int { return x + y; }
//по значению
        [=](int x, int y) -> int { return x + y + z; }
// по ссылке
        [&](int x, int y) -> int { z++; return x + y + z; }
// по-разному
        [&, z](int x, int y) -> int { return x + y + z; }
```

√GCC √MSVC++ √Intel

Свойства лямбд

- Если return один, то компилятор должен угадать тип возвращаемого значения (Стандарт, 5.1.2/4)
- Каждая лямбда имеет свой тип

```
auto a1=[](int x){return x;}
auto a2=[](int x){return x;}
//a1!=a2
```

Лямбды и замыкания

```
auto a = [](int Init) {
                return ( [=](int Val) {
                                return Init + Val;
                            });
            };
auto a1 = a(5);
cout<<a1(3)<<endl; //8
cout<<a1(2)<<endl; //7
```

√GCC ✗MSVC++ √ Intel

Обработка ошибок в лямбдах

```
std::sort(v.begin(), v.end(), [](int a, int b)) {
                                    return abs(a)<abs(b)
                                };
```

- GCC – огромное сообщение об ошибке – десятки строк. Нужная информация в первых четырех.
- Intel – сообщение повторяется несколько раз. Цикл?

nullptr

Замена для NULL

```
char* p = nullptr;
char* p2 = 0; // 0 работает и p==p2
int i = nullptr; // ошибка, nullptr - не int
```

Нет путаницы с таким случаем:

```
void foo(char *){};
void foo(int){};
foo(nullptr); //вызывает foo(char *)
```

✗GCC √MSVC++ ✗Intel

Атрибуты

```
class CBase {
public:
    virtual void f [[final]] ();
};
class CDerived : public CBase {
public:
    virtual void f (); //ошибка!
}
```

Синтаксис атрибутов еще не утвержден, они нигде не реализованы.

long long int

- Как минимум 64 бита
- Совместимость с C99
- Давно уже реализован в компиляторах

```
long long x = 9223372036854775807LL;
```

Вопросы?

alenacpp.blogspot.com