Стандарт C++ 11

Александр Смаль

Академический университет
24 апреля 2014
Санкт-Петербург

Явное переопределение и финальность

```
struct B {
    virtual void some_func();
    virtual void f(int);
    virtual void g() const;
};

struct D1 : B {
    void sone_func() override;              // error
    void f(int) override;                   // OK
    virtual void f(long) override;          // error
    virtual void f(int) const override;     // error
    virtual int f(int) override;            // error
    virtual void g() const final;           // OK
    virtual void g(long);                   // OK
};

struct D2 final : D1 {
    virtual void g() const;                 // error
};

struct D3 : D2 {};                          // error
```

Мелкие улучшения

1. Исправлена проблема с угловыми скобками: `T<U<int>>`.
2. Добавлена константа для нулевого указателя: `nullptr`.
3. Перечисления со строгой типизацией:

```
enum class Enum1 { Val1, Val2, Val3 = 100, Val4 };
enum class Enum2 : unsigned int { Val1, Val2i };
```

4. Операторы явного преобразования

```
explicit operator bool() {}
```

5. Шаблонный typedef

```
template<class First, class Second, int third>
class SomeType;

template< typename Second>
using TypedefName = SomeType<OtherType, Second, 5>;

typedef void (*OtherType)(double);
using OtherType = void (*)(double);
```

6. union может хранить нетривиальные типы данных (не ссылки).

Шаблоны с переменным числом аргументов

```
template<typename... Values> class list;

typedef list<int, std::vector<int>, std::map<int, int>> L;

void printf(const char *s) {
    while (*s) {
        if (*s == '%' && *(++s) != '%')
            throw std::runtime_error("invalid format");
        std::cout << *s++;
    }
}

template<typename T, typename... Args>
void printf(const char *s, T value, Args... args) {
    while (*s) {
        if (*s == '%' && *(++s) != '%') {
            std::cout << value;
            ++s;
            printf(s, args...);
            return;
        }
        std::cout << *s++;
    }
    throw std::logic_error("extra arguments provided to printf");
}
```

Шаблоны с переменным числом аргументов

```
template <typename... BaseClasses>
struct ClassName : BaseClasses... {
    ClassName (BaseClasses&&... base_classes)
        : BaseClasses(base_classes)...
    {}
};

template<typename Type> struct SharedPtrAllocator {

template<typename ...Args>
std::shared_ptr<Type> construct_with_shared_ptr(Args&&... params)
        return std::shared_ptr<Type>(
            new Type(std::forward<Args>(params)...));
    };
};

template<typename ...Args> struct SomeStruct {
    static const int count = sizeof...(Args);
};
```

Новые строковые литералы

```
u8"I'm a UTF-8 string."          // char[]
u"This is a UTF-16 string."      // char_16_t[]
U"This is a UTF-32 string."      // char_32_t[]
L"This is a wide-char string."   // wchar_t[]

u8"This is a Unicode Character: \u2018."
u"This is a bigger Unicode Character: \u2018."
U"This is a Unicode Character: \U00002018."

R"(The String Data \ Stuff " )"
R"delimiter(The String Data \ Stuff " )delimiter"

LR"(Raw wide string literal \t (without a tab))"
u8R"XXX(I'm a "raw UTF-8" string.)XXX"
uR"*(This is a "raw UTF-16" string.)*"
UR"(This is a "raw UTF-32" string.)"
```

Мелкие улучшения ч.2

C99

1. Добавлен тип `long long int`.
2. Добавлен `static_assert`

```
#include <type_traits>

template<class T>
void run(T * data, size_t n) {
    static_assert(std::is_pod<T>::value, "T isn't POD.");
}
```

3. sizeof без содания экземпляра

```
struct SomeType { OtherType member; };

sizeof(SomeType::member); //only C++11
```

Изменения в стандартной библиотеке

1. Исправлен смысл хинта при вставке в set/map.
2. Метод emplace для контейнеров.

```
template< class... Args >
iterator emplace( const_iterator pos, Args&&... args );
```

3. Методы cbegin и cend (для метапрограммирования, для задания типов через auto).
4. Метод shrink_to_fit для vector-а.
5. В std::list splice — за $O(n)$, size — за $O(1)$.
6. В std::vector добавился прямой доступ к памяти через data()
7. Запрет нескольким string ссылаться на одну и ту же память.
8. Добавлены unordered_set и unordered_map.

```
char const *reg_esp = "[ ,.\\t\\n;:]"; // R"([ ,.\t\n;:])";

std::regex rgx(reg_esp);
std::cmatch match;
char const *target = "Unseen University - Ankh-Morpork";

if( std::regex_search( target, match, rgx ) ) {

    size_t const n = match.size();
    for( size_t a = 0; a != n; a++ ) {
        std::string str(match[a].first, match[a].second);
        std::cout << str << "\n";
    }
}
```