Эволюция C++ это путь от простоты к простоте

А. Каленюк, к. т. н.

akalenuk@gmail.com

http://github.com/akalenuk

http://twitter.com/akalenuk2

C++ – это сложно

Количество страниц в разных стандартах на языки:

C++11 ISO/IEC 14882:2011	1338
Ada ISO/IEC 8652:2012	832
C11 ISO/IEC 9899:2011	683
Fortran ISO/IEC 1539-1:2010	603
C# ISO/IEC 23270:2006	584
IsLisp ISO/IEC 13816:2007	127
Algol 60 ISO 1538:1984	18

$$1338 \,/_{18} = 74,(3)$$

Как вы оцениваете свое знание C++
по шкале от 1 до 10

Бьярне Страуструп, оригинальный автор C++

Кто знает, что делает эта функция?

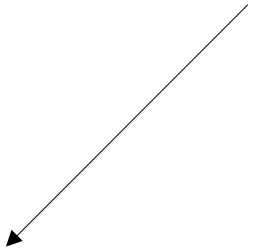
```
int f(int a, int b){
    return a * b;
}
```

Давайте проверим

```
int main(){
    int a, b, c = 2;

    for(int i = 0; i < 6; i++){
        a = b = c;
        c = f(a, b);
        printf("%d * %d = %d\n", a, b, c);
    }
}
```

Какой размерности int?



```
2 * 2 = 4
4 * 4 = 16
16 * 16 = 256
256 * 256 = 65536
65536 * 65536 = 0
0 * 0 = 0
```

Правильный ответ – никто не знает

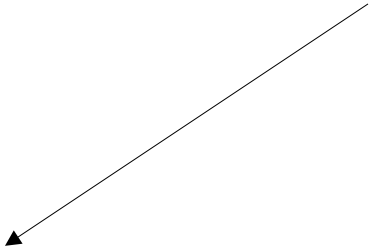
Plain ints have the natural size suggested by the architecture of the execution environment* [ISO-IEC 14882]

* – that is, large enough to contain any value in the range of INT_MIN and INT_MAX, as defined in the header <climits>.

Ок, но тип char гарантировано 8 бит

```
int f(char a, char b){
    return a * b;
}
```

Почему нет переполнения?



```
2 * 2 = 4
4 * 4 = 16
16 * 16 = 256
256 * 256 = 0
```

Кто знает sizeof(a * b)?

Правильный ответ – никто не знает

A prvalue of an integer type other than *bool*, *char16_t*, *char32_t*, or *wchar_t* whose integer conversion rank (4.13) is less than the rank of *int* can be converted to a prvalue of type *int* if int can represent all the values of the source type; otherwise, the source prvalue can be converted to a prvalue of type unsigned *int* [ISO-IEC 14882].

Добавим больше битов

There are five standard signed integer types:
“signed char”,
“short int”,
“int”,
“long int”, and
“long long int”.
In this list, each type provides at least as much storage
as those preceding it in the list [ISO-IEC 14882].

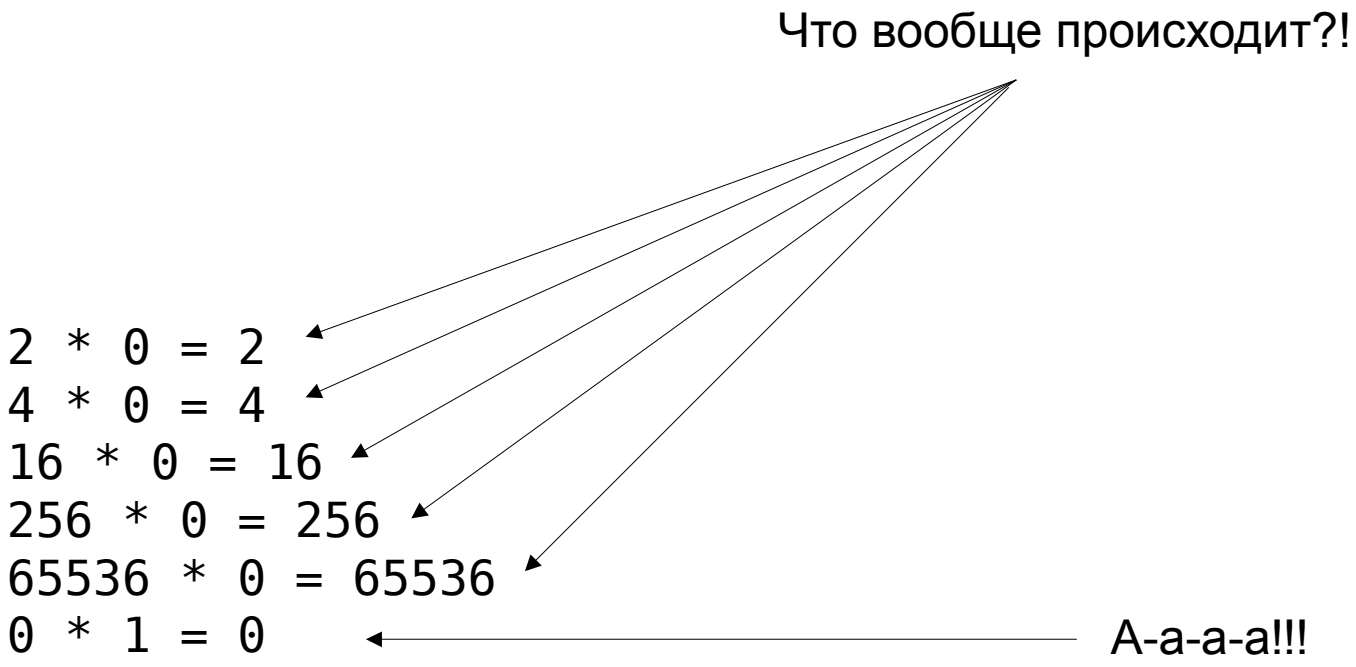
```
long long int f(long long int a, long long b){
    return a * b;
}
```

Кто знает, что выдаст проверка?

```
int main(){
	long long int a, b, c = 2;	// допустим, мы знаем, что это 64 бита

	for(int i = 0; i < 6; i++){
		a = b = c;
		c = f(a, b);
		printf("%d * %d = %d\n", a, b, c);
	}
}
```

Правильный ответ – никто не знает



Вообще-то вышеприведенный код не соответствует стандарту

A length modifier is added to the conversion specification, if needed [ISO-IEC 14882].

```
printf("%lld * %lld = %lld\n", a, b, c);
```

```
2 * 2 = 4
4 * 4 = 16
16 * 16 = 256
256 * 256 = 65536
65536 * 65536 = 4294967296
4294967296 * 4294967296 = 0
```

Простейшая задача посчитать и вывести произведение двух целых чисел оказалась непростой

Откуда берется сложность?

Низкоуровневые абстракции. Числа переполняются по размерности, стек переполняется, память течет и кончается и т. п.

Множество неочевидных нюансов в дизайне языка. Неразграничение контекста, неявные приведения типов, отсутствие защит доступа и т. п.

Исторически библиотеки и соглашения не защищены от неправильного использования. Например, *strcat* не защищена от переполнения буфера, *strncat* не гарантирует завершения нулем, а *strlcat* всем хороша, но вне стандарта.

Почему так?

C++ ведет свою историю от Си.

Язык Си появился в конце шестидесятых как вынужденная альтернатива тяжеловесным компиляторам для работы на слабой машине.

Простой в реализации, а не использовании.

Он не должен был быть изящным, он должен был работать.

Исторически язык для реализации UNIX.

Простота реализации Си – причина его популярности

Простота реализации = дешевизна портирования

Низкие требования = дешевизна железа

Lisp Machine $ 50 000

v.s.

AT&T UNIX PC $ 5 590

Unix and C are the ultimate computer viruses [Richard P. Gabriel].

C++ – высокоуровневые абстракции при низкоуровневом уровне контроля

наследование, инкапсуляция, полифморфизм

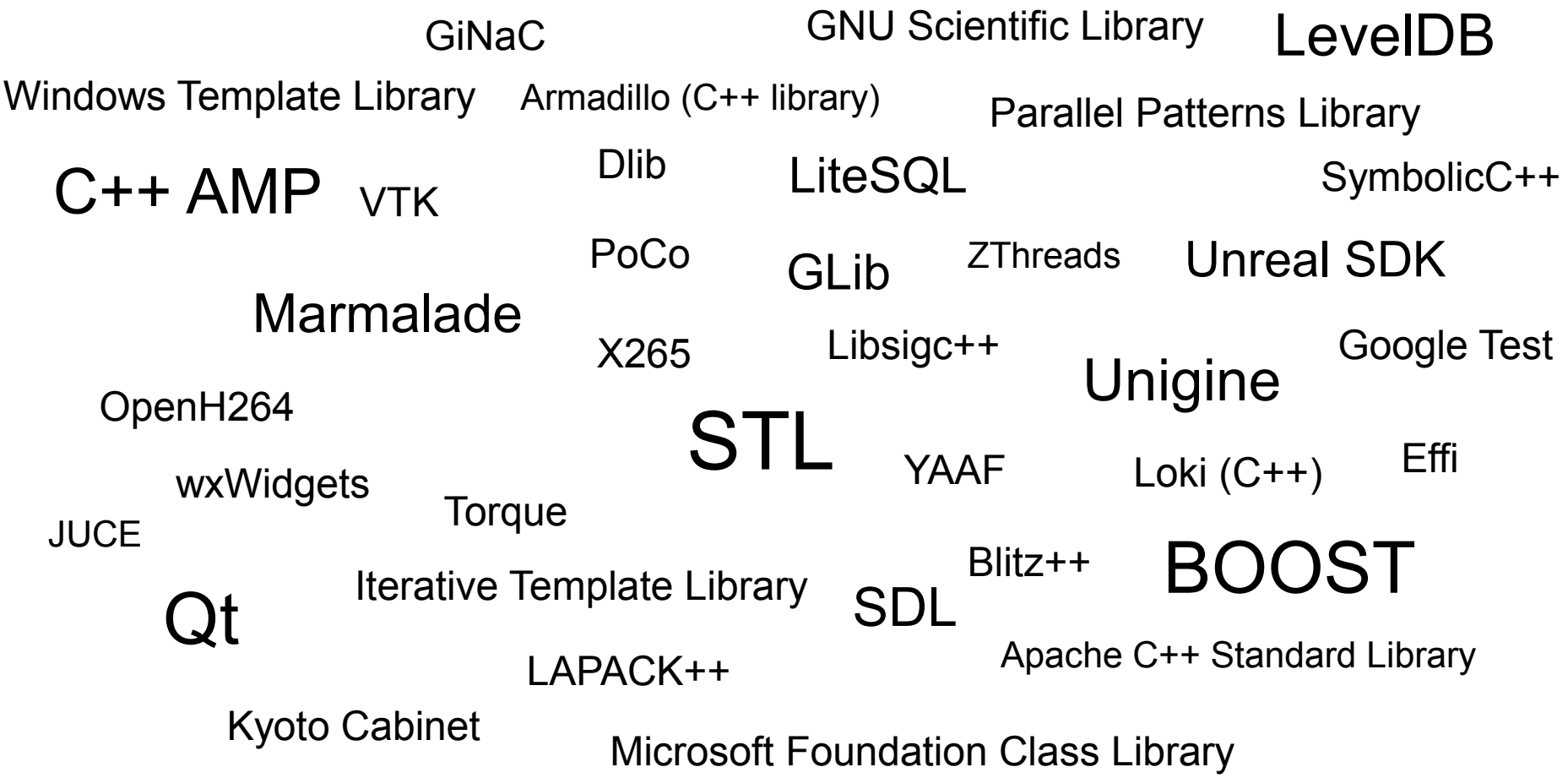
темплейты

исключения

RTTI

⇒ возможность строить годные фреймворки и библиотеки.

Например



Программист на Qt не думает про strcpy.
Вообще.

На том же поле появляются другие языки

(высокоуровневые абстракции, низкоуровневый контроль, но не Си)

Facebook – D

Mozilla Foundation – Rust

Google – Go

и т. п.

D (Dlang)

слайсы

трейты типов

статические условия

текстовые миксины

модули

модульные тесты

контрактное программирование

гарантии чистоты функций

Все из коробки!

Но! Три главные проблемы D

– Библиотеки.
– Инструменты.
– Люди!

Потому что люди, библиотеки и инструменты уже в C++

Проблема курицы и яйца

нет людей ⇒ нет фреймворков,

нет фреймворков ⇒ нет людей

Сообщество C++ знает, что C++ это сложно

Цели комитета по стандартизации:

- Make C++ a better language for systems programming and library building
- Make C++ easier to teach and learn

http://www.stroustrup.com/C++11FAQ.html

Конкретнее

- Поддерживать стабильность и совместимость.
- Предпочитать библиотеки расширениям языка.
- Предпочитать общность частности.
- Подерживать и экспертов, и новичков.
- Увеличить типобезопасность.
- Обеспечивать производительность и способность работать с железом.
- Решать задачи реального мира.

Простота, но не за счет мощности.

Старые фичи по-новому

Типы с гарантированным размером:

```
uint8_t, uint16_t, uint32_t, uint64_t,
int8_t, int16_t, int32_t, int64_t
```

Определение типов

```
int x = 0;
decltype( x ) y = 1;
```

Перечисления под строгой типизацией

```
enum class Color:int{red, green, blue}
```

Константа *nullptr* для строгой же типизации
Имеет тип указателя

Строковые литералы без ескейп-последовательностей

```
string path = R"C:\Windows\Virus"
```

Новые фичи

Автоматическое выведение типов

```
auto superObject = new SuperObject();
```

Умные указатели из коробки

```
Banana* p = new Banana(); // было
unique_ptr<Banana> p(new Banana()); // стало
shared_ptr<Banana> p(new Banana()); // или
weak_ptr<Banana> p(new Banana()); // или даже
```

Анонимные функции

```
Fruit y = Fruit::apple;
auto nApples = count_if(fruits.begin(), fruits.end(), [&y] (Fruit x) {return x == y} )
```

Кортежи

```
tuple<uint8_t, string> t(1, "One")
if( get<0>(t) == 1 ){
    cout << get<1>(t)
}
```

Статические утверждения

```
static_assert( sizeof( x ) == 4, "X size should be 4 bytes");
```

Стандарт толстый, потому что помнит все

Например, триграфы

```
??= → #     ??/ → \     ??' → ^

??( → [     ??) → ]     ??! → |

??< → {     ??> → }     ??- → ~
```

The trigraph sequences enable the input of characters that are not defined in the Invariant Code Set as described in ISO/IEC 646, which is a subset of the seven-bit US ASCII code set [ISO/IEC 9899].

Кроме стандарта языка есть стандарты программирования на нем

ISO/IEC 14882 говорит, что делать можно, а

MISRA-C++:2008,

HICPP,

JSF AV C++,

CERT CSC,

NUREG/CR 6463

и многие другие – что нельзя

Например, HICPP 4.0 (2013)

12.4.4 Write members in an initialization list in the order in which they are declared

Regardless of the order of member initializers in a initialization list, the order of initialization is always:

• Virtual base classes in depth and left to right order of the inheritance graph.

• Direct non-virtual base classes in left to right order of inheritance list.

• Non-static member data in order of declaration in the class definition.

To avoid confusion and possible use of uninitialized data members, it is recommended that the initialization list matches the actual initialization order.

For Example:

```
#include <cstdint>

class B {};
class VB : public virtual B {};
class C {};
class DC : public VB , public C
{
  public :
    DC (): B () , VB () , C () , i (1) , c () // Compliant
    {}
  private :
    int32_t i ;
    C c;
};
```

Инструменты

Статические анализаторы

Динамические анализаторы

Профилировщики

Отладчики

и т. п.

...и заверните все в IDE

Статический анализатор

CppCheck

Разрабатывается с 2009 года, открытый,
около сотни контрибьюторов,
две сотни проверок из коробки,
поддержка библиотек и форматов популярных IDE,
возможность добавлять свои проверки...

Пример работы:

```
memset(ctx, 0, sizeof(ctx));
```

```
[blablabla\md5.c:160]: (warning, inconclusive) Size of pointer
'ctx' used instead of size of its data. This is likely to lead
to a buffer overflow. You probably intend to write
'sizeof(*ctx)'.
```

Динамический анализатор

Dr. Memory

С 2013 года, открытый (LGPL), на базе DynamoRIO

Пример работы:

```
Error #55: POSSIBLE LEAK 80 direct bytes 0x034b00f8-0x034b0148 + 0 indirect bytes
# 0 replace_malloc                 [d:\drmemory_package\common\alloc_replace.c:2292]
# 1 msvcrt.dll!strcpy_s              +0x5e  (0x769ff5d3 <msvcrt.dll+0xf5d3>)
# 2 msvcrt.dll!clearerr_s            +0x337 (0x76a09eed <msvcrt.dll+0x19eed>)
# 3 msvcrt.dll!clearerr_s            +0x27e (0x76a09e34 <msvcrt.dll+0x19e34>)
# 4 cppcheck-gui.exe!?               +0x0   (0x0040104a <cppcheck-gui.exe+0x104a>)
# 5 cppcheck-gui.exe!?               +0x0   (0x00401471 <cppcheck-gui.exe+0x1471>)
# 6 KERNEL32.dll!BaseThreadInitThunk +0x11  (0x767933ca <KERNEL32.dll+0x133ca>)
```

Профилировщик

Gprof — Часть GNU Binutils, разрабатывается с 1988

Пример работы:

% time	cumulative seconds	self seconds	calls	self ms/call	total ms/call	name
83.33	0.03	0.03	500000	0.00	0.00	aa_lagd
16.67	0.03	0.01	1	5.00	5.00	aa_lagd_init
0.00	0.03	0.00	500003	0.00	0.00	aa_and_2
0.00	0.03	0.00	500001	0.00	0.00	aa_trigr
0.00	0.03	0.00	500001	0.00	0.00	dispatch_code_by
0.00	0.03	0.00	500001	0.00	0.00	run
0.00	0.03	0.00	4	0.00	0.00	assert
0.00	0.03	0.00	2	0.00	0.00	aa_or_2
0.00	0.03	0.00	2	0.00	0.00	aa_or_2
0.00	0.03	0.00	2	0.00	0.00	applied_program_run
0.00	0.03	0.00	2	0.00	0.00	log_fill_fb_table
0.00	0.03	0.00	2	0.00	0.00	set_before_equalization_flag
0.00	0.03	0.00	2	0.00	0.00	set_fb_group
0.00	0.03	0.00	1	0.00	0.00	aa_trigr_init

Все это направлено на то, чтобы сделать программирование на C++ проще

Хотите простоты – не сопротивляйтесь переменам.

1. Возьмите современную реализацию языка.
2. Выберите актуальный фреймворк и библиотеки.
3. Используйте все доступные инструменты.
4. Слушайтесь стандартов и руководств.

Что гуглить

ISO/IEC 14882

ISO/IEC 9899

MISRA-C++:2008

HICPP

JSF AV C++

CERT Secure Coding

NUREG/CR 6463

CppCheck Manual

The UNIX-HATERS Handbook

Linden, Deep C Secrets

Stroustroup, The Design and Evolution of C++

Tour of C++11 && “What’s new” in 10 minutes && “What’s new” in depth

Sutter || Meyers || Stroustroup – все что есть, особенно видео с конференций