C++ — Lurkmore

> «There are just two kinds of languages: the ones everybody complains about and the ones nobody uses. »
>
> — *Bjarne Stroustrup*

> «На говне сметану не сделаешь »
>
> — *Народ о **Qt** для C++*

Ветеран движения

C++ (*си-плюс-плюс, плюсы́, срр, кресты, приплюснутый, чсвшн. The Programming Language*) — язык программирования, о котором слышали многие, но которым, судя по всему, пока никто полностью не овладел. Создан мужиком типично нердической внешности (очки, свитер, лысина) с забавной фамилией Страуструп с целью поднять зарплаты специалистов в индустрии, что, судя по всему, ему успешно удалось.

Первоначально представлял собой язык Си (в него же и компилировался, чтобы затем использовать многочисленные готовые компиляторы Си под конкретную платформу), к которому прикрутили «классы» (сишные структуры с функциями внутри) и оператор ++ («увеличить на единицу»/«сделать шаг вперёд»), от которого и произошло название. Ныне же, со всеми доработками и библиотеками, ~~представляет свалку метафор и паттернов~~ является универсальным ЯП, поддерживаемым на чуть менее, чем всех платформах. Очень сильно потеснил своего предка, благодаря чему большинство соответствующих инструментов теперь маркируется как «C/C++». Благодаря такой генетике на нём до сих пор можно писать шустрый и не требующий больших ресурсов код, способный без тормозов ~~и мыла~~ влезть в ~~жопу~~ каждую кофеварку.

Язык быстро стал заложником собственной популярности: из-за дрочки на совместимость с тоннами практически сразу же написанного кода его развитие пошло по «системе Ниппель», когда невозможно вычистить что-то из нового стандарта, не доставив анальной боли той или иной группе хомяков. Одновременно с этим, он очень сильно отличается от старичка Си (и других популярных языков) тем, что акцент в развитии сделан не на функционале (которым стандартная библиотека крайне скудна), а на конь-септуальности[1]. Благодаря сочетанию этих двух факторов, в одном C++'ном проекте можно встретить 10 и более разных типов строк[2].

Резюме Моторина Вадима Ивановича (C++)

Типичный программист на C++

Никому не принадлежит. Если Java pwned by Oracle и сильно прихватизирована силами зла, а C# анально имеем Microsoft'ом, плюсики разрабатываются комитетом. Отсюда, в основном, и проистекают его многочисленные достоинства и недостатки:

- **Весьма невысокая выразительность**. Ввести новое ключевое слово, даже если оно сто лет назад было зарезервировано и об этом написано на каждом заборе, гарантированно означает сделать чей-то продукт некомпилируемым, а значит представители владельцев этого продукта в комитете будут упираться раком и голосовать против. Поэтому то, что в других языках будет принято с песней и энтузиазмом, в C++ будет не принято вовсе или реализовано самым неочевидным способом[3].
- **Отсталость**. Причины те же.
- Всё ж таки, комитет не просто штаны протирает, а реально отстаивает усреднённые интересы IT'шного бизнеса, а значит и типичного сферического разработчика в вакууме.
- Можно не бояться, что завтра компания-владелец свернёт лавочку, язык предаст анафеме, начнёт стращать судом опенсорсников и срать разработчикам в души и карманы.

Чудо-оружие эпохи технофашизма, о чём громко говорит имя описывающего его стандарта: ANSI ISO/IEC 14882:1998. Самый новый стандарт называется C++20 (ISO/IEC 14882:2020).

Как же так вышло?

Всё началось ~~в океане~~ с того, что парень по имени Алан Кей изобрёл[4] НЁХ под названием ООП (воплотив свои идеи в языке *SmallTalk*). История этого изобретения не только доставляет, но и способна навести на кое-какие умные мысли. Как-то раз Алан, ползая раком по коридору с распечатками своих программ, подметил случай вопиющей дискриминации: оказалось, что обычные переменные можно объединить в структуры (что примерно соответствует созданию шаблона карточки), а вот функции — хрен. Это делало простые переменные гражданами первого класса, а функции на их фоне — гражданами каких-то других классов. «Шо за несправедливость!», — подумал Алан, вспоминая, как за несколько дней до этого профессор в Университете Юты Боб Бартон отлил в граните: «**Базовый принцип рекурсивного проектирования состоит в том, что сущности на любом уровне вложенности должны обладать равными возможностями**». Оказалось, что уравнять функции и переменные в праве находиться внутри структур, это как дать возможность к карточке добавлять кнопки, при нажатии на которые карточка сама может слазить в картотеку, распечататься на принтере и т.п.

Есть два способа выучить C++ за 21 день: фантастический…

Есть два способа выучить C++ за 21 день: фантастический…

Принцип этот дичайше доставляет тем, что будучи применён ещё раз, ИЧСХ — снова к функциям, но только в другом контексте (функции сделали гражданами первого класса, разрешив наряду с данными передавать их как параметры в другие функции, а также хранить их внутри переменных, описывая в виде простых выражений) — он порождает не менее великую НЁХ — ФП. Так что, анон, подумай на досуге, кого ещё можно с кем уравнять в правах — глядишь, изобретёшь что-то не менее великое, чем ООП и ФП и твой фотопортрет спрячут в хрестоматию.

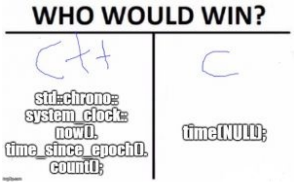
Как бы то ни было, в начале 80-х, примерно через 15 лет после описываемых событий, компания Bell Labs поручила своему сотруднику, Бьярне Страуструпу, написать какую-то информационную систему под собственные нужды (что-то связанное с телефонными вызовами, как и положено компании Белла). Бьярне подошёл к задаче как типичный, сцуко, программист: он решил, что для напейсания ему требуется ~~свой фреймворк~~ ~~своя библиотека~~ свой язык! Во всех имеющихся языках имелся фатальный недостаток. А вот конец этой истории оказался вовсе даже не типичным: вместо того, чтобы проебать все сроки, нихуя не написать и быть уволенным за долбоебизм и мегаломанию, Бьерне ВНЕЗАПНО закончил работу над языком, написал к нему транслятор и даже требуемую изначально информационную систему. Богатырь — не вы!

…и реалистичный.

…и реалистичный.

Изначально Страусик хотел писать всё на Симуле — языке, содержавшем зайчатки ООП ещё в те времена, когда Алан Кей не разродился Смолтоком. Однако оказалось, что результат такого программирования на доступном ему железе тормозит как Крузис на Спектруме. А говнокодить на низкоуровневом языке *BCPL* в силу ущербности последнего ему показалось дольше, чем написать свой язык с блекджеком. В результате он взял за основу Си, приделал к нему ООП в духе Симулы/Смолтока, и написал *cfront* — транслятор из нового улучшенного Си («Си с классами») в олдфажный Си. Язык неожиданно для Бьярне стал дичайше популярен среди его коллег, например, а потом — и среди прочих мимокрокодилов.

Первое время свистоперделки к языку он добавлял в одного, но позже передал эти обязанности комитету по стандартизации. Где-то примерно в это же время русский по фамилии **СТ**епанов и китаец по фамилии **Л**и набыдлячили библиотеку *STL*, что, якобы, расшифровывается как *Standard Template Library* (но мы-то знаем!). С одной стороны, это была хорошая, годная библиотека, ибо позволяла при помощи шаблонов работать с такими сущностями, как строки и массивы. С другой — она была полна дичайших изъёбств в названиях (вместо Array — vector, вместо Add() — push_back() и т.п.) и спорных архитектурных решений (считать алгоритмы не частью объектов, как, казалось бы, следует из духа ООП, а независимыми сущностями). К восторгу одних и ужасу других, эта библиотека стала частью стандартной библиотеки языка.



Позже выяснилось, что строковый тип, который нихуя не умеет работать с текстом (заменять фрагменты, например) — это, конечно, круто, стильно, молодёжно, но немного недостаточно для плодотворного программирования. То же самое касалось отсутствующих регекспов, работы со временем и другого функционала, доступного в прочих языках из коробки, но требующего от сиплюсплюсника прибегать к старым добрым, но устаревшим сишным функциям. Тогда другие умники, чьи имена не столь известны, запилили *boost* — хтонический многогигабайтный ужоснах, построенный на тех же принципах, что и STL, но дающий искомый функционал. О его простоте и дружелюбности красноречиво говорит название модуля для работы со временем: *Boost.Chrono*. К ещё большей пополбни обладающих вкусом месье части Буста тоже начали стандартизировать. Параллельно с этим язык обзаводился совершенно инопланетным синтаксисом (аж целых несколько операторов приведения типов, один из которых выглядит как *std::uintptr_t v1 = reinterpret_cast<std::uintptr_t>(&i);* и прочий BDSM).

Наконец, под влиянием более молодых конкурентов — всяких шарпов — к языку прикрутили элементы ФП (лямбды и прочее), отчего язык окончательно начал выглядеть инфернально.

Тем не менее, он и в 2021 году остаётся чуть ли не единственным языком, гарантирующим максимальную переносимость, наличие библиотек на все случаи жизни, отсутствие нежданчика с производительностью и независимость от каких-либо крупных вендоров.

Священная книга

Как и положено каждой уважающей себя секте, программисты на C++ обзавелись собственным священным писанием — Стандартом. Именно так, с большой буквы и никак иначе (см. эту статью, например).

Стандарт состоит из множества нумерованных разделов, подразделов и подраздельчиков (типа, «7.6.2.9»), исчерпывающе описывающих, почему твой говнокод обязательно приведёт к *Undefined Behavior* — ситуации, когда в зависимости от фазы Луны, разработчика твоего компилятора, версии операционной системы и других факторов произойдёт что угодно, что совершенно невозможно предсказать заранее. Например, твой компьютер превратится в кашалота или горшок с петунией.

Великие гуру способны, глядя в произвольный исходник, по памяти называть номера разделов, которые нарушает каждая строчка такого кода.

И к чему пришло?

Язык ~~цветёт и~~ пахнет. В смысле, вполне себе жив. Если верить TIOBE (это такое кармадрочерство для языков, может, не самое справедливое или объективное, но на другие никто не ссылается), уже долгие, долгие годы в Царя горы играют Java с Pure C, по очереди сталкивая друг друга с первого места на второе. А наш герой стабильно входит в пятёрку самых популярных, выше Шарпа, но ниже Питона, лол.

Язык прочно занял свои ниши — движки игр, например — и вряд ли их сдаст в ближайшее время.

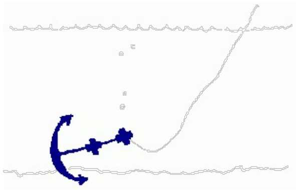
В вебе, как бы ни казалось обратное, он тоже незримо присутствует. Взять хоть браузеры, которые его отображают. Или Мускул/Машу, где данные анонимуса хранит большая часть сайтов. Все они написаны на связке языков, в которой плюсы занимают почётное место.

Ах да, как сообщают с мест, битва за карточку метрополитена проиграна Жабе. Это печально.

Хелловорлды

Самый элементарный

```
#include <cstdio>
int main()
{
        puts( "Hello, World" );
        return 0;
}
```



DNIWE EBANOE

Вариант 0

```
#include <iostream>
int main()
{
  std::cout << "Hello, World\n";
}
```

Сферический в вакууме.

Отмечается, что "\n" не эквивалентно "std::endl", так что вариант #1 более верен.

Вариант 1

```
#include <iostream>
int main()
{
  std::cout << "Hello, World" << std::endl;
}
```

Вариант 2

```
#include <stdio.h>
int main(void)
{
  const char *message[] = {"Hello ", "World"};
  int i;

  for(i = 0; i < 2; ++i)
    printf("%s", message[i]);
  printf("\n");
}
```

На самом деле это сишный код, но сипипи компилером он тоже жуётся.

Вариант 3

```
#include <iostream.h>
#include <string.h>

class string
{
private:
  int size;
  char *ptr;

public:
  explicit string(const char* chrs = 0) : size(chrs ? strlen(chrs) : 0)
  {
    ptr = new char[size + 1];
    if (chrs)
      strcpy(ptr, chrs);
    else
      ptr[size] = 0;
  }

  string(const string &s) : size(s.size)
  {
    ptr = new char[size + 1];
    strcpy(ptr, s.ptr);
  }

  ~string()
  {
    delete [] ptr;
  }

  friend ostream &operator <<(ostream &, const string &);
  string &operator=(const char *);
  string &operator=(const string&);
};

ostream &operator<<(ostream &stream, const string &s)
{
  return(stream << s.ptr);
}

string &string::operator=(const string &s)
{
  this->~string ();
  new (this) string (s);
  return(*this);
}

string &string::operator=(const char *chrs)
{
  *this = string(chrs);
  return(*this);
}

int main()
{
  string str;

  str = "Hello World";
  cout << str << endl;

  return(0);
```

```
}
```

Как то так.

Вариант 4

```
#include <cstdio>
#define HelloWorld
#define BEGIN {
#define END }
#define PROGRAM int main()
#define WRITELN(a) puts(a);

PROGRAM HelloWorld

BEGIN
        WRITELN("Hello, World")
END
```

Для олдфагов. На настоящем паскале выглядит, впрочем, несколько иначе.

Вариант 5

```
#include <iostream>
#include <boost/mpl/vector_c.hpp>
#include <boost/mpl/for_each.hpp>
#include <boost/lambda/lambda.hpp>

typedef boost::mpl::vector_c<char,'H','e','l','l','o',',', ' ','W','o','r','l','d'> string;

int main()
{
        boost::mpl::for_each<string>(std::cout<<boost::lambda::_1);
        std::cout<<std::endl;
};
```

Шаблонно-бустовый

Вариант 6

```
#include <string.h>
#include <unistd.h>
int main()
{
  char *str="Hello, World!";
  write(1, str, strlen(str));
  return 0;
}
```

По принципу ассемблера. Тоже сишный код, кстати.

Вариант 7

```
#include <iostream>
#include <list>
#include <algorithm>

using namespace std;
int main()
{
    list<string> hw {"hello", " ", "world"};

    for (auto &s: hw)
        cout << s;
    cout << endl;

    for_each(hw.begin(), hw.end(), [](string &s){
        cout << s;
    });
    cout << endl;
}
```

C++11 ^__^. Здесь присутствуют такие ништяки как: initializer lists, type inference (т.е. auto), range-based for-loop, lambda и for_each. К сожалению, регэкспы пока в gcc-4.6* не пашут.

Вариант 8

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
        char str[strlen("Hello, World!") + 1];

        strcpy(str, "Hello, World!");
        cout << str;
        return 0;
}
```

Исправленный пример от Анонимуса.

Вариант 9

```
#include <stdio.h>

int main()
 {
        char * h="Hello, World!\n\0";
        for(;*h;putchar(*h++));
        return 0;
 }
```

Вариант на Си: Побуквенный вывод фразы - "Hello, World!".

Вариант 10

```
using namespace std;
#include<ofstream>

class hello{
public:
hello()
{
 ofstream hello;
 hello.open ("hello.com");
 hello << "»..№..ґ.Љ.CH.вщH Hello, World!";
 hello.close();
 system("hello.com");
}}put;

int main(){
return 0;
}
```

Братишка! Я тебе покушать принёс! Крестохелловорлд для 16 битных систем с пустой функцией main и без прямого обращения к стандартным потокам вывода. На Windows 7, Linux не работает к сожалению.

Последний раз запускал на XP SP2 пару лет назад.

Вариант 11

```
#include<iostream>
#include<conio.h>

using namespace std;
```

```
int main()
{
cout<<"Hello world!!!\n";
_getch();
return 0;
}
```

Пример от Анонимуса[2]

Вариант 12

```
#include <iostream>

int main()
{
 char *s;

 s = "Hello, World!\n";

 std::cout << s;

 return 0;
}
```

Пример от Анонимуса [3]

Вариант 13

```
#define _str "Hello, World!\n"
#include <cstdio>
#include <cstdlib>
#include <cstring>

int main()
{
 char *s;

 s = (char *)malloc(15);

 strcpy(s, _str);
 printf("%s", s);

 return 0;
}
```

Пример от Анонимуса [4]

Вариант 14

```
#define _str "Hello, World!\n"
#include <cstdio>
#include <cstring>

int main()
{
 char *s;

 s = strdup(_str);

 printf("%s", s);

 return 0;
}
```

Пример от Анонимуса [5]. От предыдущего отличается тем, что strdup() сама выделяет память.

Вариант 15

```
#include <cstdio>

int main(void)
{
 for(const char* c = "Hello, World!"; *c; putchar(*c++));
 return 0;
}
```

Пример от Анонимуса [6]. Как вариант 9, только правильно.

Вариант 16

```
#include <iostream>
#include <conio.h>
using namespace std;

void hello(){
        cout << "Hello, world!";
}

void main(){
        hello();
        _getch();
}
```

Вариант 17

```
#include <iostream>
#include <algorithm>

void foo(char ch)
{
        std::cout << ch;
}

int main()
{
        char* str = "Hello, world!";
        std::for_each(str, str + strlen(str), foo);

        return 0;
}
```

Вариант 18

```
#include <iostream>
#include <string>
#include <iterator>

int main()
{
        std::string str = "Hello, world!";
        std::copy(begin(str), end(str), std::ostream_iterator<char>(std::cout));
        return 0;
}
```

Вариант 19

```
#include <cstdio>

void main()
{
        char line[30000]={0};
        int cell=0;
```

```
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        while(line[cell])
        {
                cell++;
                line[cell]++;
                line[cell]++;
                line[cell]++;
                line[cell]++;
                line[cell]++;
                line[cell]++;
                line[cell]++;
                cell++;
                line[cell]++;
                line[cell]++;
                line[cell]++;
                line[cell]++;
                line[cell]++;
                line[cell]++;
                line[cell]++;
                line[cell]++;
                line[cell]++;
                line[cell]++;
                cell++;
                line[cell]++;
                line[cell]++;
                line[cell]++;
                cell++;
                line[cell]++;
                cell--;
                cell--;
                cell--;
                cell--;
                line[cell]--;
        }
        cell++;
        line[cell]++;
        line[cell]++;
        putchar(line[cell]);
        cell++;
        line[cell]++;
        putchar(line[cell]);
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        putchar(line[cell]);
        putchar(line[cell]);
        line[cell]++;
        line[cell]++;
        line[cell]++;
        putchar(line[cell]);
        cell++;
        line[cell]++;
        line[cell]++;
        putchar(line[cell]);
        cell--;
        cell--;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        line[cell]++;
        putchar(line[cell]);
        cell++;
        putchar(line[cell]);
        line[cell]++;
        line[cell]++;
        line[cell]++;
        putchar(line[cell]);
        line[cell]--;
        line[cell]--;
        line[cell]--;
        line[cell]--;
        line[cell]--;
        line[cell]--;
        putchar(line[cell]);
        line[cell]--;
        line[cell]--;
        line[cell]--;
        line[cell]--;
        line[cell]--;
        line[cell]--;
        line[cell]--;
        line[cell]--;
        putchar(line[cell]);
        cell++;
        line[cell]++;
        putchar(line[cell]);
        cell++;
        putchar(line[cell]);
}
```

А программист на BrainFuck'е видит так.

Вариант 20

```
#include <iostream>
#include <cstring>

char print(const char* str )
{
        if (*str == 'H')
                return *str;

        std::cout << print(str-1);
        return *str;
```

```
}

int main()
{
        char str[] = "Hello world!";
        std::cout << print(str+strlen(str)) << std::endl;

        return 0;
}
```

Вариант 21

```
#include <iostream>

int hi() {
        std::cout << "Hello World!" << std::endl;
        return 0;
}

int i = hi();

int main(void) {
        return 0;
}
```

Вариант с пустым main() для эстетов.

Вариант 22

```
#include "stdafx.h"
#include "iostream"
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
        cout << "Hello, world!\n";
        return 0;
}
```

вариант M$ visual studio 2013

Вариант 23

```
#include <Windows.h>

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{
        MessageBox(NULL, L"This is Win32 API Hello World!", L"Hello World!", MB_OK);
        return 0;
}
```

Гуй-версия — вместо консоли выводит окошко с сообщением. Работает только в Microsoft Windows, использует Windows API. Сишным компилятором тоже должно жеваться.

Вариант 24

```
#include <iostream>
#include <iterator>

int main()
{
        std::ostream_iterator<const char*>(std::cout) = "Hello, World!\n";
        return EXIT_SUCCESS;
}
```

Вариант 25

```
#include <iostream>

template <typename... Arguments>
class HW;


template <class T>
class HW<T>
{
public:
    HW(T t): ch(t) {}
    ~HW() { std::cout << ch;}
private:
    T ch;
};

template <typename T, typename... ARGS>
class HW<T, ARGS...> : public HW<ARGS...>
{
public:
    HW(T t, ARGS... args):HW<ARGS...>(args...), ch(t){}
    ~HW() { std::cout << ch;}
private:
    T ch;
};

template <typename... ARGS>
void Print(ARGS... args)
{
    HW<ARGS...> hw(args...);
    (void)hw;
}

int main()
{
    Print('H','e','l','l','o',' ','W','o','r','l','d','\n');
}
```

Вариант 26

```
#include <iostream>
#include <conio.h>

using namespace std;

int main()
{
        char ch = 0;

        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        cout << ch;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        cout << ch;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
```

```
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        cout << ch;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        cout << ch;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        cout << ch;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        cout << ch;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++;
        cout << ch;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        cout << ch;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        cout << ch;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        cout << ch;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        cout << ch;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        cout << ch;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--; ch--;
        ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++; ch++;
        ch++; ch++; ch++;
        cout << ch;

        _getch();
        return 0;
}
```

Инкрементно-декрементный вариант.

Вариант 27

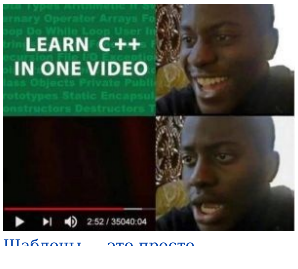
```
#include <stdio.h>

asm(
    ".PRINT:\n"
    ".ascii \"Hello, World\\0\"\n"
);

int main(){
        asm(
                "leaq   .PRINT(%rip), %rcx\n"
                "call   printf\n"
        );
        return 0;
}
```

С использованием ассемблерной вставки.

Шаблоны



Шаблоны — это просто

Десятое правило Гринспена гласит: «Любая достаточно сложная программа на C или Фортране содержит заново написанную, неспецифицированную, глючную и медленную реализацию половины языка Common Lisp». Шаблоны являются таким недолиспом, встроенным в язык C++.

Шаблоны.они (

- Полны по Тьюрингу. Что это значит? Это значит, что если написать достаточно сложный код с шаблонами, бедный компилятор, пытаясь их скомпилировать, начнёт выполнять вычисления, которые можно сделать *произвольными* (и тем самым решить любую задачу, например, эмулировать любой, самый сложный комп, хоть Apple с M1). Только вот входные данные для таких вычислений надо тоже явно задавать в коде. Практическая польза от *шаблонного метапрограммирования* находится где-то на уровне: «Если купить новый Форд Фокус, отковырять пластиковую заглушку в крыше, вынуть оттуда маленькую лампочку, которая горит, когда открыты двери, примотать её к голове, на спину повесить автомобильный аккумулятор, соединить проводами и нырнуть головой в унитаз, то можно будет пугать светом всех встречных кротов».

Шаблоны — это просто.

- Вычисляются на этапе компиляции, что с учётом первого преимущества позволяет добиться сколь угодно долгой сборки программы.
- Статические, а не динамические, как дженерики, что позволяет устраивать километровые забеги, выясняя, что круче.
- Семантические, а не текстуальные, как макросы, что позволяет не только устраивать километровые забеги, выясняя, что круче, но и вводить в ступор читателей кода, используя и то и другое одновременно в одном фрагменте.
- Лежат в основе *STL*, который состоит из них чуть менее чем полностью, и *boost*, который состоит из них чуть более чем на семь четвёртых.

);

Примеры

Вычисление факториала на стадии компиляции

```
#include <iostream>
template<int N>
struct fact
{
   enum
   {
      value = N * fact<N - 1>::value
   };
};
template<>
struct fact<0>
{
   enum
   {
      value = 1
   };
};

int main()
{
   const int n = 6;
   std::cout << n << "! = " << fact<n>::value << std::endl;
   return EXIT_SUCCESS;
}
```

А это — факториал от авторов Phoenix-а, без всяких б-гомерзких enum-ов и static-ов

```
#include <vector>
#include <algorithm>
#include <iostream>
#include <boost/phoenix/core.hpp>
#include <boost/phoenix/function.hpp>

struct factorial_impl
{
    template <typename Sig>
    struct result;

    template <typename This, typename Arg>
    struct result<This(Arg)>
        : result<This(Arg const &)>
    {};

    template <typename This, typename Arg>
    struct result<This(Arg &)>
    {
        typedef Arg type;
    };

    template <typename Arg>
    Arg operator()(Arg n) const
    {
        return (n <= 0) ? 1 : n * this->operator()(n-1);
    }
};


int
main()
{
    using boost::phoenix::arg_names::arg1;
    boost::phoenix::function<factorial_impl> factorial;
    int i = 4;
    std::cout << factorial(i)() << std::endl;
    std::cout << factorial(arg1)(i) << std::endl;
    return 0;
}
```

Шаблоны — это просто

```
template<typename C> class Basic_operations
{
public:

  bool operator==(const C& a)const
  {
    if(derived().size() != a.size()) return false;
    for(int i=0; i < derived().size(); ++i)
      if(derived()[i] != a[i]) return false;

    return true;
  }

  bool operator!=(const C&)const; // определение где-то но не здесь
  // ... еще ряд функций;

  // доступ к операциям класс C
  const C& derived()const {return static_cast<const C&>(*this);}
};

template<typename T> class Math_container: public Basic_operations<Math_container<T> >
{
public:
  std::size_t size()const;
  T& operator[](size_t);
  const T& operator[](size_t)const;
};
```

Классический пример с автоматической перегрузкой оператора сравнения для наследников класса Basic_operations.

Библиотека Analog Literals

Пользуясь этой библиотекой, можно писать такой код:

```
assert( top( o-------o
             |L       \
             | L       \
             |  o-------o
             |  !       !
             !  !       !
             o  |       !
              L |       !
               L|       !
                o-------o ) == ( o-------o
```

```
                                 |        !
                                 !        !
                                 o--------o ) );
```

Моаг здесь

Шаблоны здесь нужны для того, чтобы значения этих литералов вычислялись на этапе компиляции, являясь так называемыми expression templates.

Согласно альтернативному мнению шаблоны превращают C++ в тормозное функциональное говно!

STL

Как было сказано выше, Алан Кей пропагандировал идею, что «Алгоритмы + структуры данных = объекты».

А некто Степанов в ответ придумал, что алгоритмы и структуры данных должны быть порознь. Идея посетила его светлую голову, когда он находился в состоянии бреда, вызванного отравлением то ли рыбой, то ли грибами. С появлением C++ и шаблонов в нём, идея оказалась воплотима, и была разработана библиотека STL, позже вошедшая в Стандарт.

В качестве структур данных выступают контейнеры, такие, как массив, список, словарь, а в качестве алгоритмов — сотни какой-то непонятной хрени вроде lexicographical_compare и set_symmetric_difference.

STL.он (

- На 3,14159% реализуется разработчиком компилятора (из-за расовых разногласий по поводу ОС), поэтому существует десяток реализаций, каждая со своими клопами.
- Содержит разрыв шаблона. Шаблон *vector<T>* — контейнер STL типа вектор, содержащий T. Но *vector<bool>* — ни разу не контейнер STL и никаких bool не содержит. Нормально, да? Но всем абсолютно похуй, взамен юзают *vector<char>*.
- Реализует множества и словари не на хеш-таблицах, а с применением деревьев поиска, что обычно печально с точки зрения производительности. (Хинт: есть версии, реализованные на хеш-таблицах, нужно просто добавить префикс *hash_* или *unordered_*).
- Имеет громоздкий в использовании API из-за стремления к максимальной общности. Например, для получения значения по ключу из ассоциативного массива (без потенциального изменения состояния последнего) необходимо написать аж две сотни букв. Для кого-то лечится макросами вроде *BOOST_FOREACH* и *RANGE*.
- Вызывает привыкание:

  STL … It's also something that, when you first encounter it, makes you wonder what the hell the designer was smoking. And once you've gotten used to it, you start wondering why other library designers don't start smoking the same thing.

);

Итоги

По мере своего старения и уменьшения головного мозга, Александр Александрович полностью разочаровался не только в C++, но и в объектно-ориентированном программировании вообще. Вот что он пишет про ООП:

> «Я уверен, что ООП методологически неверна. Она начинает с построения классов. Это как если бы математики начинали с аксиом. Но реально никто не начинает с аксиом, все начинают с доказательств. Только когда найден набор подходящих доказательств, лишь тогда на этой основе выводится аксиома. То есть, в математике вы заканчиваете аксиомой. То же самое и с программированием: сначала вы должны начинать развивать алгоритмы, и только в конце этой работы приходите к тому, что вы в состоянии сформулировать четкие и непротиворечивые интерфейсы. Именно из-за этой неразберихи в ООП так популярен рефакторинг — из-за ущербности парадигмы вы просто обречены на переписывание программы, уже в тот самый момент, когда только задумали её спроектировать в ООП-стиле».

Пруфлинк: «Почему объектно-ориентированное программирование провалилось».

Это как если бы Папа Римский вышел на балкон в Ватикане и сказал: «Молиться Иисусу — методологически неверно, Аллах акбар!»

К сожалению, несмотря на такой каминг-аут, откатить все STL'ные правки из стандарта КЕМ не представляется возможным в силу причин, описанных выше.

В середине десятых Александр Александрович выступал перед сотрудниками Яндекса, где долго и с видимым удовольствием рассказывал, какими эпитетами награждают его в письмах благодарные программисты и в турне по каким местам предлагают отправиться. К ещё большему сожалению, он объявил, что решительно отказывается их посещать.

CUDA и OpenCL

> *для тех, кого забанили в педивикии: OpenCL != OpenGL ; OpenCL != OpenML ; OpenCL != OpenAL*

Матановые библиотеки, позволяющие получить на C++ ещё больше скорости, так как вместо медленного центрального процессора задействуют графический процессор или другой вычислительный ускоритель.

Своим появлением обязаны развитию графических шейдеров, изначально представлявших собой коротенькие инструкции для операций с текстурами (типа сложить текстуру A с текстурой B). Со временем сложность шейдеров нарастала, позволяя реализовывать более сложные операции, а помимо текстурных шейдеров появились вершинные шейдеры — для изменения координат вершины треугольника[5]. В общем, компьютерная графика развивалась, и по вычислительной мощности[6] графические процессоры превзошли центральные процессоры общего назначения.

Затем появился PhysX — специализированный процессор для работы с физикой, на которую ранее в компьютерных играх тупо забивали болт. Этот процессор превосходил по своей мощности центральные процессоры общего назначения, так что при воспроизведении какого-нибудь физического эффекта центральный процессор тормозил, а PhysX показывал невозможное. Но был один минус: физический ускоритель PhysX стоил ощутимых денег, и потому многих начала так же ощутимо душить жаба, при том что далеко не все игры поддерживали новые физические спецэффекты, и получалось, что недешёвую карту физического ускорителя нужно покупать, чтобы узреть красоту в 3,5 играх при отсутствии какого-либо другого влияния на игровой процесс (второй Half-Life с его гравитационной пушкой ещё не появился).

И пока хомячки пищали, но готовили свои кошельки… вдруг эсперты выяснили, что несложные физические вычисления, типа сложить пару-тройку векторов, можно выполнить и через вершинный шейдер, а если извернуться, то можно посчитать физику и на текстурных шейдерах. Правда, всё это с бутылочным горлышком перегонки массивов данных из обычной памяти в графическую и обратно с участием центрального процессора. В общем, радужные перспективы PhysX, стали для специалистов и экспертов тускнеть и становиться туманными, в то время как для обычных хомячков, «эффективных» менеджеров с их гуманитарным образованием, и биржевых трейдеров, торгующих акциями, перспективы продолжали казаться такими же радужными… => его создатели постарались его побыстрее продать, пока не упали акции, а nVidia постаралась его побыстрей купить, пока это не сделали конкуренты. В итоге nVidia:

1. включила все наработки PhysX в свои видеокарты,
2. переделала текстурные и вершинные шейдеры в шейдеры общего назначения,
3. запилила CUDA, что позволяет считать произвольный матан на шейдерах общего назначения.

WIN!

Boost

В некотором царстве, в некотором государстве, два C++ программиста, приняв пару стаканов вина, обсуждали разработку открытых библиотек, которые должны были бы содержать всё необходимое, не включенное в недавно вышедший Стандарт. Один из них упомянул, что Герб Саттер готовил пропозицию языка программирования Booze, который должен был быть лучше, чем Java. Смысл этой остроумной шутки в том, что java — сорт кофе, а booze — «бухло». Продолжением игры слов стало название «Boost» для набора открытых библиотек, куда на сегодняшний день вошли около сотни библиотек, а некоторые из них даже были запилены в нынешний или будущий Стандарт (пруфлинк — FAQ буста).

Использование этого набора библиотек в промышленном коде является предметом споров, причём не только в форумах интернетов, но и IRL с заказчиком, начальником или коллегой. Консерваторам не очевиден профит применения проверенных решений вместо написания своих велосипедов, также их пугает размер и сложность буста. На первый взгляд, опасения кажутся небезосновательными — эта штука собирается полностью около часа на не самом слабом ПК и занимает несколько гигабайт в собранном виде со всеми либами во всех конфигурациях; многие библиотеки кажутся полными матана и возникают сомнения в способности быдлокодеров использовать их. На самом деле, вышеприведенные опасения — хуита, а поддержка своих велосипедов или написание «минималистичного» кода обходятся дороже.

Qt

Ещё в 90-х двое троллей решили подшутить над владельцами тёплых ламповых машинок, и показать миру быстрый и лёгкий GUI. В итоге была организована компания Trolltech, и их детище — Qt получило неиллюзорный коммерческий и просто народный успех. Позже это всё богатство было продано Nokia, затем перепродано Digia, в свете анальной связи Nokia с Microsoft. Алсо «Qt developers are happy developers. 95% Satisfaction.» (c) www.qt.io

Как забыть C++

Достоинства

- GUI можно строить покрасивее, чем в дэльфи и прочих — есть даже поддержка CSS, HTML и даже пародия на JavaScript, в виде QML.
- Весьма и весьма шустрая и небольшая библиотека.
- Сигналы и слоты.
- Имеются порты на десятки языков.
- Одна из основных сред слюникса.
- Полноценный фреймворк, даже есть пародия на GC (все объекты наследующие QObject удаляются автоматически).
- Парадигма MVC, ограждающая вменяемых программистов от быдлокодеров[7].
- Плазма не падает.

Недостатки

- Таки довольно хуёвая кроссплатформенность.
- Несмотря на простоту, школьникам всё равно не даётся.
- Нужно полное понимание ООП, из-за чего школьникам и быдло/С-кодерам не даётся.
- Зависимости: если решил внести в софтину новую функциональность — нужно пересобирать всю библиотеку (если же распространять «как есть» — вес библиотек составляет почти 30мб).
- На нищебродских машинках тормозит, ибо слишком широк и создан без понятия цели создания.
- В версиях до 4.* все кнопки, полоски просмотра и прочие элементы окон рисовались везде одинаково (и везде одинаково убого), начиная с 4 под каждую систему всё прорисовывается как нативное.
- Moc компилятор.
- Был создан, как легковесная графическая библиотека, а превратился в широкий глобус с чуть менее видными краями.
- Сборка под виндами — весьма тоскливое занятие.
- Qt Creator и плагины для прочих IDE могут в один прекрасный день перестать видеть qmake (для Qt < 5.x)

Где он живёт

> «Большинство C++-программистов не знают, что делает большинство C++-программистов. »
>
> — *Андрей Александреску*

В наши дни C++ программисты по-прежнему очень нужны. Когда такого программиста удаётся поймать, его сажают за компьютер и заставляют:

- Поддерживать стародивный C++ (а то и Pure C) код.
- Писать большие коробочные игры (куча готовых движков/библиотек плюс быстрота)
- Задрачивать системное/низкоуровневое программирование (драйвера, работа с портами, ядра ОС и т. п.)
- Писать ядро высоконагруженных систем, для которых Java слишком требовательна, а Erlang тормознут. Apache, например. Или Dropbox.
- Писать приложения конкурентоспособность которых определяется быстродействием (от агрегаторов фондовых бирж до движка практически каждого браузера и т. д.)
- Писать ядро приложения, которое обрамляет код на более простом и понятном языке. Всякий научный и узкоспецифический софт вроде Mathematica.

Разумеется, адекватные программисты пишут на C++ только основные части, самые требовательные к нагрузкам и памяти. Для интерфейсов, внешней логики и прочей функциональщины используют внешние языки (Lua, Ruby, Boost:Python), которые милосердней к пользователю. А то и вовсе компилируют код в библиотеку и подключают к своему уютненькому проектику на PHP.

А вот всякий интерпрайз и корпоративные приложения выбирают Java и C#. Для них C++ решительно не подходит:

- Сложен и настроить в нём можно всё. За счёт этого не помещается в голове даже у самого Страуструпа.
- Чтобы работать стало легко и приятно, надо осилить STL/Boost. А про них в учебниках пишут редко и скомкано: большинство программистов до сих пор учится по книгам, которые описывает не столько C++, сколько именно C с классами. Желающие писать на C++ крутотень и зарабатывать OVER9000 баксов должны узнавать что-то новое каждый день и неустанно постигать библиотеки (начинаешь с книжки 21st Century C: C Tips from the New School и далее по нарастающей)
- Крайне сложная формализация. Поэтому в то время, как под Python/Java/C# есть ReSharper, который находит очепятки в коде прямо на лету, на C++ приходится каждый раз компилировать и распутывать запутанные ошибки компилятора, который в очередной раз подавился шаблоном. Спасибо, что подсветка синтаксиса есть.
- Язык универсален — а веб-морду всё равно не сделаешь. Конечно, есть WT, но без сборки мусора пыхтеть придётся долго.

Писать большой проект с нуля и на C++ заслуженно считается малоудачной идеей. Любой программист прокачавшийся выше хеллоувордщика допускает на 10 000 строк кода примерно одинаковое количество ошибок (которое варьируется от 1 у Кнутов и Кормаков до 10 000+ у типичного быдлокодера). А код на C++ обычно длиннее, чем на более поздних языках, и багов, разумеется, будет намного больше. В разы проще написать пусть медленный, но макет на чём-то хоть и медленном, но со сборкой мусора (обычно это C#, Java или Python, так как умеют читать C++±библиотеки), а потом переписать особенно тормозящие участки (которые, как водится, окажутся совсем не там, где ты думал). Наконец, бывает так, что единственная скорость, которой хочет заказчик — это скорость разработки и он *покупает макет и начинает юзать его как нормальную программу*, оставляя программистов наедине с баблом и удивлением.

В результате мы имеем всякие экзотические конфигурации вроде заводского сервера для станков на MINIX (видимо, переделывали из собственной дипломной работы, вдохновлённой общеизвестной книгой Таненбаума).

А заказчик, помучившись год, начинает требовать версию 2.0, с кучей новых фич, а «скорости мне хватает». В результате макет переделывается, и снова переделывается, и снова… с перспективой превратиться в операционную систему на managed-коде, которая, в отличии от Фантом ОС, ещё и запускается.

А разгадка одна: в некоторых самоучителях (aka «tutorial»-ах) можно прочесть, что сабж++ и не рыба, и не мясо, но является языком ВНЕЗАПНО «среднего уровня».

- Означает сие следующее: сабж неплохо портируется на какие угодно платформы и архитектуры (ибо сравнительно малые трудозатраты на сей кропотливый процесс), и при том он держит как минимум марку «малый джентльменский набор», случись потребность перекомпилировать некий высокоуровневый (быдло)код. И потому удобно софт автоматом декомпилировать для последующего refactoring или даже reverse engineering не в ассемблер, а именно в Си или же СиПиПи, оно же ЦоПеПе.

- С одной стороны, C++ aka *CPP* уступает в читабельности языкам высокого уровня (и притом отнюдь не только Пайтону), для хоть какой-то эргономичности язык требует доработки напильником (вроде готового фреймворка, заточенного под некую задачу) или нуждается в костыле, делающего язык визуально понятным (даже отечественный производитель с 1996-го для подобных нужд вывел в общественное достояние костыль-"язык «Д.Р.А.К.О.Н.», чтобы «гибрydный» DRAKON-CPP можно было хоть как-то читать непрофессионалу с учёной степенью по международному социологическому менеджменту.

Почему «быдлокодерский»

> «А вы друзья, как ни садитесь —
> всё в программисты не годитесь… »
>
> — *Эпос о Гильгамеше, XXII век до н.э.*

Тысячи возможностей выстрелить себе в ногу в языке являются результатом совмещения высокоуровневых концепций ООП с более низкоуровневым языком C и говорят о продуманности дизайна C++ и уважении принципов обратной совместимости. Отсутствие сборки мусора говорит о попытках экономии памяти — а ещё о том, что без глобальных переменных, или как их называют теперь, «Паттерн Одиночка», программировать не получится. Зато конечный результат почему-то есть. И откуда же он взялся?…

Типичный представитель

Аргументы в пользу принадлежности C++ к «илитарным» языкам не выдерживают критики. Порог вхождения низок; как показывает опыт, каждую обезьянку, пишущую на решётках или на жаббе, можно заставить писать на C++ (хотя далеко не каждую — читать). При этом, засилье шаблонов STL и Boost побуждает горевателей делать даже простейшие вещи вроде сортировки данных в массиве предельно громоздкими, ресурсоёмкими и неочевидными методами.

Избыток сложности не может компенсировать низкий порог вхождения, что влечёт за собой гигабайты говнокода, кривых программ, и, как следствие наличия большого числа «типа шарящих» — низкую оплату труда письма на нём. Зачем нанимать профессионала за пять косых, если можно нанять десяток голодных студентов за пятьсот, которые хоть и наебошат говно, но зато наебошат же.

Студентами и написано на C++ относительно большое количество программ, а дописывается еще бóльшим количеством индусов. Если системный код Windows написан на C, то свистелки и перделки, вроде IE, шелла, и прочего — чаще всего на C++. Даже мышкой херачить можно: для этого есть Видимая Студия и Борланд Ц-Быдлер. Достаточно знать 15 функций и хорошо манипулировать операторами if и while, и «программный продукт» готов.

А то, что на выходе получается АдовЪ ГовнокодЪ — никого не волнует.

Положение в современном мире


++Ↄ профессионал смотрит на тебя как на…

Серьёзный бизнес мало интересуют субъективные рассуждения школяров-задротов, наподобие изложенных выше, и поэтому на реальную роль C++ действуют совершенно иные факторы. А именно:

1. Переусложнённость, сочетание прямой работы с памятью с грязной реализацией ООП, что для новых проектов означает увеличенный бюджет и более высокий риск фейла.
2. Универсальность, никому, как правило, не нужная.
3. Все прошивки для всех микроконтроллеров (их сложно найти только в лампах накаливания) пишутся на C. Кто-то где-то пытался их писать на паскале, но уже поздно.
4. Большой объём уже написанного на C++ кода, который приходится поддерживать. Или же кода, написанного на C и которому на кой-то хер нужно было сменить парадигму.
5. ~~Свежее мясо~~ Новых программистов обычно уже не учат на этом языке (а учат этой вашей Java, C# и прочим более устойчивым к быдлокодерству языкам), и это уже вовсе не тот язык, который они знают лучше всего (а stl и boost студентота не знает чуть менее, чем вовсе).

Кроме того, если посмотреть на рейтинг популярности языков на сайте www.tiobe.com, то можно заметить две вещи. Во-первых, по популярности C++ никогда не обходил своего папу, а лишь однажды к нему приблизился, но было это в конце 90-х. Во-вторых, с появлением яблофона и яблолопаты, идейный брат C++, Objective-C, который используется почти исключительно Яблокорпорацией для их продукции и дотоле почти неизвестный, стал также набирать в популярности. Того и глядишь, через пару лет мы узнаем, почему Objective-C на самом деле непереносимо ужасный язык.

Олсо C++ практически (чуть более, чем целиком) не используется для написания этих ваших Linux Kernels & Daemons, что кагбе намекает всем нам.

И не забудьте ещё раз поправить именно эту часть статьи и донести до нас Правду!

И наконец, —

Луговский ставит последние точки над «ё»:

> C++ — довольно таки примитивное, но монстровое поделие, полное исторически сложившихся нелепых нагромождений. Человек, который хорошо в нем ориентируется — это хорошее зубрилко, а не хороший программист. Умение героически преодолевать трудности, которые создает твой собственный инструмент, вместо того, чтобы решать непосредственно прикладную задачу, в современном мире ценится разве что только среди прыщавых сосок. Работодатель же это сомнительное умение не ценит, и совершенно справедливо.
>
> В общем, так: хороший программист обязан знать Си. Хороший программист может знать C++, но это не обязательно уже. Главное, чтоб C и C++ не были единственными доступными программисту инструментами — иначе это адски паршивый программист.

— *SQL.ru*

См. также

- Быдлокодер
- Няшная Сишка
- Умение разбираться в чужом коде
- Только закончил собирать
- C++ considered harmful, гарантируют даже Линус и RMS
- Почему C++ не торт
- Лжестрауструп об ужасных плюсах

Примечания

1. ↑ Нет, библиотек, реализующих любой функционал, до жопы и более. Но вот необходимость тащить стороннюю библиотеку (одну из сотен) для вещей, которые *встроены* в остальные языки дарит море радости и счастья.
2. ↑ И это только более-менее *стандартных* типов строк! Общее же число реализаций строковых классов на порядки превосходит сакральное число.
3. ↑ Пример. Вместо записи:

   var i = 5;

   (что значит: «Компилятор, объяви переменную i и сам догадайся, какого она типа, раз в неё надо записать 5»), принятой в чуть менее, чем всех языках, C++ предлагает писать:

   auto i = 5;.
4. ↑ В том же смысле, в каком Virgin/Westwood изобрели жанр RTS: разрозненные элементы были и до этого, но кому-то надо было их собрать.
5. ↑ В современной компьютерной графике модель представляет собой многогранник из множества треугольников
6. ↑ Если измерять в флопсах, то есть скорости выполнения операция над числами с плавающей запятой/точкой.
7. ↑ см. w:Магическая кнопка

Языки программирования
++i + ++i 1C AJAX BrainFuck C Sharp C++ Dummy mode Erlang Forth FUBAR God is real, unless explicitly declared as integer GOTO Haskell Ifconfig Java JavaScript LISP My other car Oracle Pascal Perl PHP Prolog Pure C Python RegExp Reverse Engineering Ruby SAP SICP Tcl TeX Xyzzy Анти-паттерн Ассемблер Быдлокодер Выстрелить себе в ногу Грязный хак Дискета ЕГГОГ Индусский код Инжалид дежице Капча КОИ-8 Костыль Лог Метод научного тыка Очередь Помолясь Проблема 2000 Программист Процент эс Рекурсия Свистелки и перделки Спортивное программирование СУБД Тестировщик Умение разбираться в чужом коде Фаза Луны Фортран Хакер Языки программирования

Software
12309 1C 3DS MAX 8-bit Ache666 Alt+F4 Android BonziBuddy BrainFuck BSOD C++ Chaos Constructions Cookies Copyright Ctrl+Alt+Del Denuvo DOS DRM Embrace, extend and extinguish FL Studio Flash FreeBSD GIMP GNU Emacs Google Google Earth I2P Internet Explorer Java Lolifox LovinGOD Low Orbit Ion Cannon Me MediaGet MenuetOS Microsoft Miranda Movie Maker MS Paint Open source Opera PowerPoint PunkBuster QIP Quit ReactOS Rm -rf SAP SecuROM Sheep.exe Skype StarForce Steam T9 Tor Vi Windows Windows 7 Windows Phone 7 Windows Phone 8 Windows Vista Wine Winlogon.exe Wishmaster Word ^H ^W Автоответчик Антивирус Ассемблер Баг Билл Гейтс и Стив Джобс Блокнот Бот Ботнет Браузер Варез Винлок Вирусная сцена Генерал Фейлор Глюк Гуй Даунгрейд Демосцена Джоэл Спольски Донат Защита от дурака Звонилка Интернеты Кевин Митник Китайские пингвины Костыль Красноглазики Леннарт Поттеринг Линуксоид Линус Торвальдс Лог Ман Машинный перевод Мегапиксель

w:C++ en.w:C++ ae:C++