

# 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++

C++ (си-плюс-плюс, *плюсы, спп, кресты, приплюснутый, чевши*. *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 собой 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 жёну каждую кофеварку.

Язык быстро стал заложником собственной популярности: из-за дочки 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<sup>[1]</sup>. Благодаря сочетанию этих двух факторов, в одном C++-ном проекте можно встретить *10 и более разных типов строк*<sup>[2]</sup>.

*Никому не принадлежит*. Если *Java* *owned* by *Oracle* и сильно прихвачивается *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 некомпilierуемым, 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<sup>[3]</sup>.
- **Отсталость**. Причины те же.
- Все 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 предстает анафеме, начнёт *страшать 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<sup>[4]</sup> *НЭХ* 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 уравнивая 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.

Принцип этот дайчайше 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 *НЭХ* — ФП. Так что, анон, подумай на досуге, кое ещё можно с кем уравнивать 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*, сдуко, 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, Бьярне *ВНЕЗАПНО* закончил работу над языком, написал к нему транслятор и даже требуемую изначально информационную систему. Богатырь — не вы!

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 *Крузик* 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 *сfront* — транслятор из нового улучшенного Си («Си с классами») в олдфайный Си. Язык неожиданно для Бьярне стал дайчайше популярней среди его коллег, а потом — и среди прочих *мимокрокдиллов*.

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 Степанов* и *китаец по фамилии Ли* набыдляли 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 дайчайших изъёбств в названиях (вместо *Atgu* — *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 силпослупника прибегать к старым добрым, но устаревшим сишным функциям. Тогда другие *умники, чьи имена не столь известны*, записали *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 попобели *обладающих вкусом меся* 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 независимости 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



Ветеран движения

Резюме Моторина  
Вадима Ивановича  
(C++)  
Типичный программист  
на C++



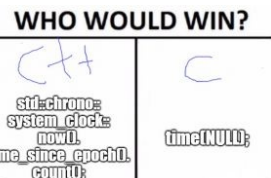
Есть два способа выучить C++ за 21 день: фантастический...

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



...и реалистичный.

...и реалистичный.



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 **ТЮБЕ** (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 <stdio.h>
int main()
{
    puts( "Hello, World" );
    return 0;
}
```

### 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);
}

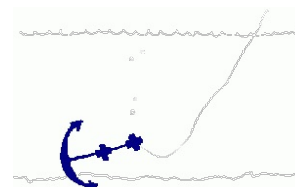
string &string::operator=(const string &s)
{
    this->~string ();
    new (this) string (s);
    return(*this);
}

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

int main()
{
    string str;

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

    return(0);
}
```



DNIWE EBANOE



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

```











## Достоинства

- GUI можно строить по красивее, 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 **слионкса**.
- Полноценный фреймворк, даже есть пародия на GC (все объекты наследующие **QObject** удаляются автоматически).
- Парадигма **MVC**, ограждающая вменяемых программистов от **быдлокодеров**<sup>[7]</sup>.
- **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 софтинку новую функциональность — нужно пересобрать всю библиотеку (если же распространять «как есть» — вес библиотек составляет почти 30мб).
- На **нищевродских 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.
- Мос 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 С++-программистов не знают, что делает большинство С++-программистов.»

— Андрей Александреску

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

- Поддерживать стародивный С++ (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 С++ 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 проекту на **PHP**.

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

- Сложен и настроить в нём можно всё. За счёт этого не помещается в голову 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 учатся по книгам, которые описывают не столько С++, сколько именно С с классами. Желющие писать на С++ 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 опечатки в коде прямо на лету, на С++ 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 С++ заслуженно считается малоудачной идеей. Любой программист прокачавший 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 С++ 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 С±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 фиш, 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»-ax) 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, С++ 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 «гибридный» **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 **С** и говорят о продуманности дизайна С++ 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?...

Аргументы в пользу принадлежности С++ к «литерарным» языкам не выдерживают критики. Порог вхождения низок; как показывает опыт, каждую обезьянку, пишущую на **решётках** или на **жаббе**, можно заставить писать на С++ (хотя далеко не каждую — читать). При этом, засилье шаблонов STL и Boost побуждает гореваителей 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 шарших» — низкую оплату труда письма на нём. Зачем нанмать 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 С++ относительно большое количество программ, а дописывается еще большим количеством индусов. Если системный код Windows написан на С, то свистелки и перделки, вроде **IE**, шелла, и прочего — чаще всего на С++. Даже мышкой херачить можно: для этого есть Видимая Студия и Борланд Ц-Бидлер. Достаточно знать 15 функций и хорошо манипулировать операторами if и while, и «программный продукт» готов.



Как забыть C++



Типичный представитель

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

## Положение в современном мире

Серьёзный бизнес мало интересует 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 С. 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 С и которому на кой-то хер нужно было сменить парадигму.
5. Свежее-мяеё 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](http://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 яблочфона и яблочопаты, идейный брат C++, Objective-C, который используется почти исключительно Яблочнокорпорацией для их продукции и дотеле 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 С и C++ не были единственными доступными программисту инструментами — иначе это адски паршивый программист.

— SQL.ru



C++ профессионал смотрит на тебя как на...



## 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 1 И это только более-менее *стандартных* типов строк! Общее же число [реализаций](#) строковых классов на порядки превосходит [сакральное число](#).
- 3 1 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 1 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 1 В современной компьютерной графике модель представляет собой многогранник из множества треугольников
- 6 1 Если измерять в флопсах, то есть скорости выполнения операция над числами с плавающей запятой/точкой.
- 7 1 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 Xuzzy Анти-паттерн Ассемблер Быдлокодер Выстрелить себе в ногу Грязный хак Дискета ЕГТОГ Индусский код Инжалид дежище Капча КОИ-8 Костыль Лог Метод научного тыка Очередь Помоясь 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 Вarez Винлок Вирусная сцена Генерал Фейлор Глюк Гуй Даунгрейд Демосцена Джеол Спольски Донат Защита от дурака Звонилка Интернеты Кевин Митник Китайские пингвины Костыль Красноглазки Леннарт Поттеринг Линуксоид Линус Торвальдс Лог Ман Машинный перевод Мегапиксель

w:C++ ep:w:C++ ae:C++