

С++ для новичков.



Путь камикадзе

Путь камикадзе



Мы тоже немного пилоты
@supapro

Все хотят изучать C++

- Ну, может быть и не все...
- Но почти каждый день появляются новые и новые люди, желающие изучить C++.
- Иногда они хотят писать быстрые программы
- Иногда они хотят создавать ИИ
- Иногда они хотят на самом деле изучать *чистый C*, но не знают об этом, потому что отделить одно от другого сложно для новичка.

Но что же их привлекает?

Привлекательность C/C++

C и C++ очень распространены.

- Почти все базовые компоненты и службы всех вычислительных систем написаны на C или C++
 - все операционные системы
 - все СУБД
 - все базовые компоненты WEB
 - все сетевые службы, сетевые устройства

Привлекательность C/C++

- Также на C/C++ написаны
 - множество игр и игровых "движков"
 - библиотеки AI и машинного обучения
 - библиотеки распознавания текста, речи
 - библиотеки обработки изображения
 - ПО банков и платёжных систем
 - Криптографическое ПО
 - Криптовалюты

С и С++ применяются для создания ПО с высокой производительностью

Что же лучше? С или С++?

- С -- довольно слабый, неудобный язык, не обладающий большой мощностью и выразительностью.
- С++ обладает всеми сильными сторонами С, но даёт большее.
- Линус Торвальдс полагает, что С лучше
- Мы уважаем его выбор, но конечно с ним не согласны.



Всё это делает C++ привлекательным языком

(в глазах начинающих)



Изучу его - буду богом!

Кто и зачем изучает C/C++



- Студенты ВУЗ, СТО
 - заинтересованные (33% от студентов)
 - в силу наличия в программе (66% от студентов)
- Специалисты
 - расширение знаний и навыков (второй язык)
 - переквалификация (падение спроса на основное направление либо скука)
- Непрофессионалы
 - "а вот я хочу"
 - "чудики"

Но какой результат можно ожидать от изучения C++?

Сложный ли C++ язык?



- Да! Очень сложный!

Почему C++ такой сложный?



C++ является одним из самых сложных из существующих языков программирования.

- C++ базируется на C. Сохраняет обратную совместимость.
Общая история двух родственных языков составляет почти 50 лет!
- C++ - гибридный язык с поддержкой различных парадигм программирования.
- C++ стандартизирован, есть много реализаций-компиляторов, каждый со своими расширениями.
- C++ переносим, но на каждой платформе есть своя специфика.
- Программа на C/C++ не обязана быть даже целиком валидной с точки зрения языка, но может при этом работать.

Почему C++ такой сложный?



Внутри C++ содержится как минимум несколько языков по различию в подходах

- Чистый C
- C с классами (C++ до 98го)
- C++ с шаблонами и обобщённым программированием
- C++ с метапрограммами из 2000-ных
- Функциональный C++11
- Современный C++17/20

Почему С++ такой сложный?



Документация на С++ огромна.

Год	Объём спецификации языка
1990	453 стр
1998	776 стр
С++11	1353 стр
С++14	1370 стр
С++17	стр

С чего начинается знакомство с любым языком программирования?

- Принципы функционирования программы (90% Машина Тьюринга)
- Типы данных языка
- Операторы и управляющие конструкции языка
- Системы сборки, поставки и развёртывания

Работа с каждым языком начинаеся с Hello World!

Пишем HELLOWORLD на C++

- <http://cpp.sh/>

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

int main()
{
    std::string name;
    std::cout << "What is your name? ";
    getline (std::cin, name);
    std::cout << "Welcome to C++, " << name << "!\n";
}
```

Просто и понятно.
Но хочется большего.

Пишем HELLOWORLD на C++

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

int main()
{
    std::string name;
    std::cout << "What is your name? ";
    getline (std::cin, name);
    std::cout << "Welcome to C++, " << name << "!\n"; <== Хотим имя БОЛЬШИМИ буквами!
}
```

- Ищем функцию ...

std::toupper

Defined in header `<cctype>`

```
int toupper( int ch );
```

Преобразует символ в верхний регистр в соответствии с правилами преобразования символов, определённых текущей локалью.

В стандартной локали "C", следующие символы нижнего регистра `abcdefghijklmnopqrstuvwxyz` заменяются соответствующими символами верхнего регистра `ABCDEFGHIJKLMNOPQRSTUVWXYZ`.

*Класс! сейчас сделаем!

Пишем HELLOWORLD на C++

```
#include <iostream>
#include <string>
#include <cctype>

int main()
{
    std::string name;
    std::cout << "What is your name? ";
    getline (std::cin, name);
    int i = 0;
    std::cout << "Hello, " << char(toupper( name[i++] ))
               << char(toupper( name[i++] ))
               << char(toupper( name[i++] ))
               << char(toupper( name[i++] ))
               << " !"
               << std::endl;
}
```


Пишем HELLOWORLD на C++

```
#include <iostream>
#include <string>
#include <cctype>

int main()
{
    std::string name;
    std::cout << "What is your name? ";
    getline (std::cin, name);
    int i = 0;
    std::cout << "Hello, " << char(toupper( name[i++] ))
               << char(toupper( name[i++] ))
               << char(toupper( name[i++] ))
               << char(toupper( name[i++] ))
               << char(toupper( name[i++] ))
               << char(toupper( name[i++] ))
               << char(toupper( name[i++] ))
               << char(toupper( name[i++] ))
               << char(toupper( name[i++] ))
               << char(toupper( name[i++] ))
               << char(toupper( name[i++] ))
               << char(toupper( name[i++] ))
               << " !"
               << std::endl;
}
```

Первый прогон!

options

compilation

execution

What is your name? Gennady

Первый прогон!

options

compilation

execution

```
What is your name? Gennady  
Hello, YDANNEG !
```

Первый прогон!



Первый прогон!

```
22     << char(toupper( name[i++] ))  
23     << char(toupper( name[i++] ))  
24     << char(toupper( name[i++] ))  
25     << " !"  
26     << std::endl;
```

```
27  
2 26:38: warning: operation on 'i' may be undefined [-Wsequence-point]
```

Главный принцип работы программы C++



Неопределённое поведение.

Когда это случилось?

Не было ничего сложного!

- ни сложных, многофайловых проектов
- ни безумных конструкций препроцессора
- ни сложных математических операций
- ни адского темплейтного метапрограммирования
 - ЭТО ВООООЩЕ ДЕТСКИЙ КОД!
- такое можно было бы писать в школе!

Неопределённое поведение.

Что же это такое?

- Что говорит стандарт ANSI/ISO?

```
1.3.24 undefined behavior  
behavior for which this International Standard imposes no requirements
```

вроде не страшно...

- Что говорит cppreference?

```
Renders the entire program meaningless if certain rules of the language are violated.
```

- Так что, вся моя программа неправильная?

Неопределённое поведение.

Что же это такое?

undefined behavior - there are **no restrictions on the behavior of the program**. Compilers are **not required to diagnose undefined behavior** (although many simple situations are diagnosed), and the compiled program is **not required to do anything meaningful**.

Самый главный слайд.

UV значит

- Неизвестность
- Помощи не будет !
- Ты один, а вокруг опасность!



Типы данных

Предположим, вы не испугались UB...

Какие типы данных в распоряжении программиста C/C++?
Давайте разбираться...

Классификация типов

- fundamental types:
 - тип `void`
 - тип `std::nullptr_t`
 - арифметические типы
 - типы с плавающей точкой (`float`, `double`, `long double`)
 - целочисленные типы
 - тип `bool`;
 - символьные типы:
 - короткие символьные типы (`char`, `signed char`, `unsigned char`);
 - широкие символьные типы (`char16_t`, `char32_t`, `wchar_t`);
 - знаковые целые типы (`short int`, `int`, `long int`, `long long int`);
 - беззнаковые целые типы (`unsigned short int`, `unsigned int`, `unsigned long int`, `unsigned long long int`);
 - Составные типы
 - ...

Пишем toupper на C++

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

int main(int argc, char *argv[])
{
    std::string s;
    std::cout << "Input some strings\n";
    getline (std::cin, s);
    for(unsigned c = 0; c < s.size(); ++c)
    {
        if( s[c] == ' ')
            std::cout << s[c];
        else
            std::cout << char(s[c] - ('a' - 'A'));
    }
    std::cout << "\n";
}
```

- Запускаем!

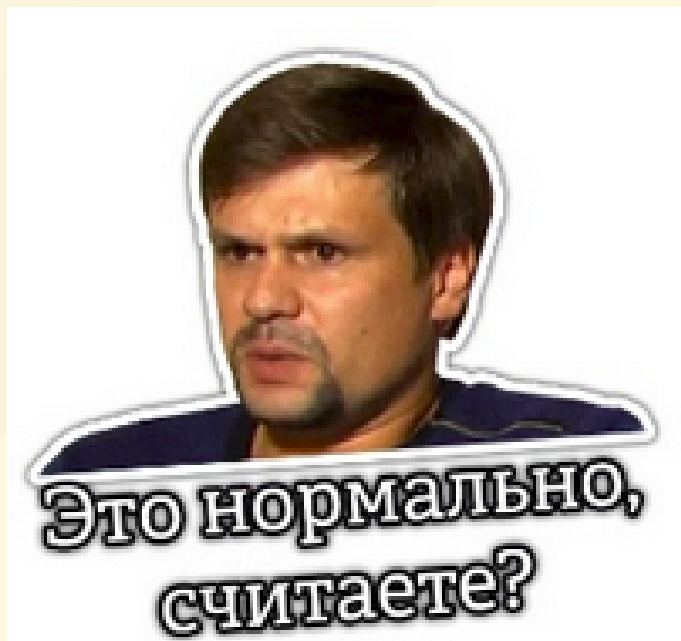
Запускаем toupper на C++

```
Input some strings  
the quick brown fox jumps over a lazy dog  
THE QUICK BROWN FOX JUMPS OVER A LAZY DOG
```

- Всё хорошо?

Классификация типов

- fundamental types:
 - арифметические типы
 - целочисленные типы
 - **символьные типы:**
 - char



Классификация типов

На самом деле

- в С и С++ нет типов данных для поддержки символов.

есть элементарные функции для классификации и тривиальных преобразований символов в рамках кодировки ASCII (т.н. Latin-1) и библиотеки операционных систем и специализированные для работы с текстом, не входящие в стандарт.
`char`, `char16_t`, `char32_t`, `wchar_t` -- это просто числа

- `wchar_t` - вроде бы по описанию, символ Unicode?

Нет, в С++ и С вообще нет встроенной поддержки Unicode, и она не появится ранее 20 или даже 23 го года.

- Вы ещё хотите писать обработку текстов на С++?

Типы данных в C++

Ничего не забыли?

- А где же строки символов, текст?

В C и C++ нет поддержки строк символов на уровне языка.

- Вместо поддержки строкового типа в C и C++ есть представления данных и (в C++) специальный класс, входящие в стандартные библиотеки, и реализующие работу с текстом.

Вы ещё хотите писать обработку текстов на C++?

Типы данных в C++

Ничего не забыли?

- ок, а даты, время ? Такие важные структуры данных, где они?

В C++ и C нет встроенной поддержки типов данных "дата", "время" и нет операций с ними

- Поддержки типов "дата" и "время" реализованы
 - в C в стандартной библиотеке языка как структура особого вида и несколько функций для работы с ней.
 - C++ в стандартной библиотеке поддержка интервальных данных появилась только в 2011 году.
 - Поддержки дат, календарей в C++ нет до сих пор

Сборка программ

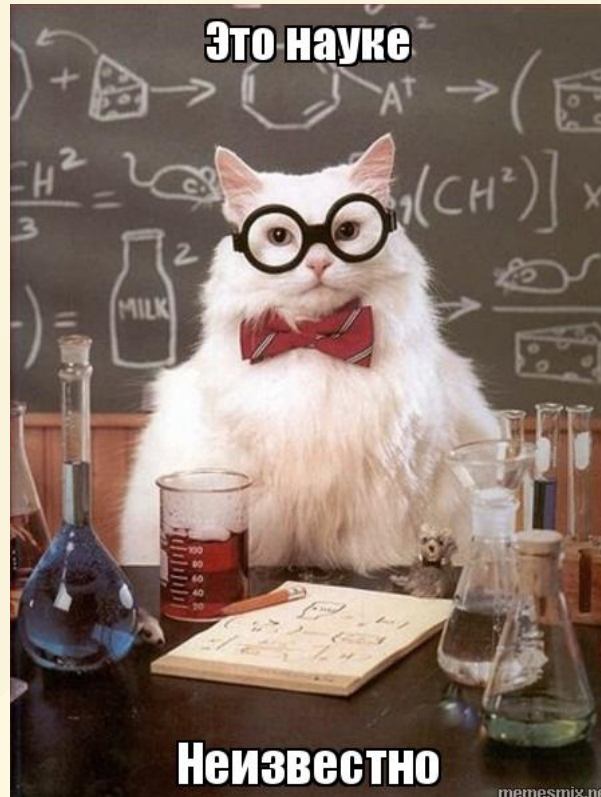
Ну, ладно, живут же как-то люди с этим.

А как же нам собирать нашу программу?



- Как из исходного текстового файла получить работающую программу?

Как из исходного текстового файла
получить работающую программу?



- (Это не шутка)

Как из исходного текстового файла получить работающую программу?

- Стандарт языка совсем не описывает процесс получения исполняемой программы из исходного кода
- Описывается только 9-тифазный процесс преобразования
- При этом применяется так называемая отдельная компиляция.

Раздельная компиляция

Среди компилируемых языков есть два противоположных подхода

- Паскаль:
 - один большой файл с исходным кодом.
 - компиляция в один шаг с получением исполняемого приложения
- Фортран/С/С++
 - много файлов с исходным кодом
 - компиляция каждого файла в промежуточный объектный код (полумашинный)
 - сборка из нескольких файлов объектного кода исполняемого приложения
 - при определённых условиях допустима сборка объектных модулей, полученных из исходных кодов, написанных на разных языках программирования.

Раздельная компиляция - проблемы

Главный бич начинающих при сборке

- При сборке начинающих поджидают две главные проблемы
 - Undefined Reference
 - Это когда что-то не определено
 - Нарушение ODR (One Definition Rule)
 - Это когда что-то определено несколько раз или по-разному

Раздельная компиляция - проблемы

- Не смотря на очевидность UR и ODRV, очень сложно объяснить начинающему что же у него не так в программе.
- Отсутствие определённого процесса сборки не позволяет дать совет о том, как собирать программу на другой платформе.
- В будущем в стандарте 2020 ожидается введение модулей исходных кодов, которые должны были бы исправить проблему, но сделают всё ещё хуже:
 - код будет написан 2мя способами, по-старому и по-новому.
 - простоты и изящности это не добавит

Надежда есть!



Да-да!

Мы спасём вас!



С++ -- не первый язык!

* Если вы не гений, и не собираетесь стать профессиональным разработчиком ПО, не изучайте С++.

Изучение С++ как первого языка программирования слишком сложно и бесполезно, если вы не собираетесь этим профессионально заниматься.



С++ -- не первый язык!

- Непрофессионалам нечего делать среди С++ разработчиков.
- Студентам лучше избегать изучения С++, если они не планируют заниматься разработкой ПО

Если всё же надо изучать C++

Принцип постепенного усложнения.

- Не торопиться. Не гнать вперёд. Не пытаться сделать всё и сразу.
- Начинать с классических консольных приложений
- Изучать язык, а не операционную систему, в которой он работает
- Изучать язык, а не библиотеки для него.
- Да, будет не так эффектно



Принцип постепенного усложнения.



Принцип постепенного усложнения.

- Приложения можно разделить на
 - консольные/терминальные
 - сервисы/демоны
 - встраиваемые/прошивки
 - приложения GUI
 - сайты / WEB
 - игры 2D/3D
- Желательно учиться разрабатывать приложения именно в таком порядке.
это примерный порядок возрастания сложности и увеличения роли не-C++ в этом

Мы спасём вас!

Не изучать устаревшее.



- С++ развивается и одни возможности приходят на смену другим.
- Нужно изучать новые которые пришли на смену старым
- хотя это очень сложно сделать.

Мы спасём вас!

Не делать того, что не понимаешь.



- На C++ очень важно вникать в то, что пишешь в программе, понимать каждую строку, каждый символ
- Если ты делаешь это, ты учишься
- Если ты не делаешь это, ты ходишь по 10 раз по одним и тем же граблям.

Мы спасём вас!

Проверять каждую строчку!



- Очень много ошибок у начинающих по невнимательности.
- Надо проверять код.

Мы спасём вас!

Проверять каждую строчку!



- Очень много ошибок при неправильном использовании библиотечных функций.
- Надо тщательно читать документацию на функции.
 - что функция делает
 - в каких условиях работает, а когда не может работать
 - что принимает и что возвращает
 - как сигнализирует об ошибках
- Надо проверять код.

Подводим итоги ...

- С++ не подходит для изучения программирования. Нужен другой язык
 - python
 - D или Rust? Может Fortran?
- С++ только для профессионального использования.
- Если ты не собираешься его использовать, изучать его бессмысленно
- При необходимости изучать С++ нельзя торопиться
 - Наскоком не получится
 - Постепенно увеличивать сложность приложений.
 - Сначала -- только обработка данных
 - Не изучать устаревшие части языка.
 - Внимательно писать код

Литература и ссылки

- <https://ru.cppreference.com>
- <https://github.com/CppCon/CppCon2017>
- <https://isocpp.org/blog>
- <https://github.com/masterziv/meetup.git>
- <https://t-do.ru//ProCxx> или @proscxx
- <https://t-do.ru//supapro> или @supapro

Конец



План

- Причины привлекательности C/C++
- Категории изучающих
- Причины сложности C++
 - UB
 - Базовые типы данных и их особенности
 - раздельная компиляция
 - системы сборки
 - (ИСКЛЮЧЕНО) принцип программного управления и почему он нарушается
- Внимание! Мины!
 - C++ не первый язык!
 - Принцип постепенного усложнения.
 - (ограничение по видам приложений, используемым библиотекам и т.п.)
 - Не изучать устаревшее.
 - Не делать того, что не понимаешь.
 - Проверять каждую строчку.
- Выводы