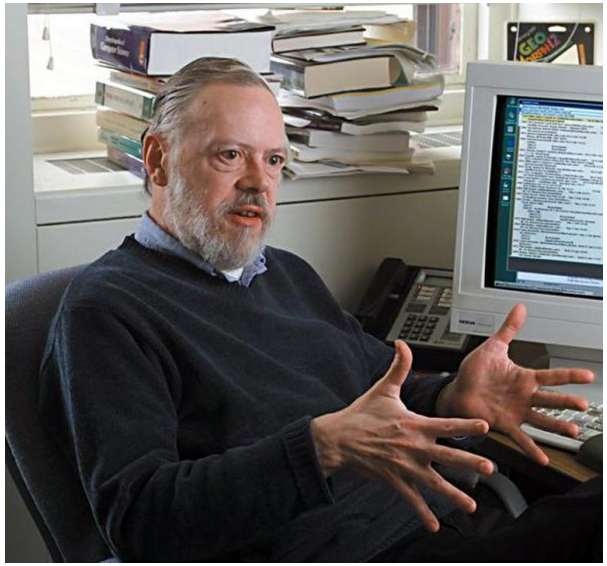
C++ course: 1. Introduction

2017 by Oleksiy Grechnyev

C and C++



Dennis Ritchie, the man behind C (1972) and Unix

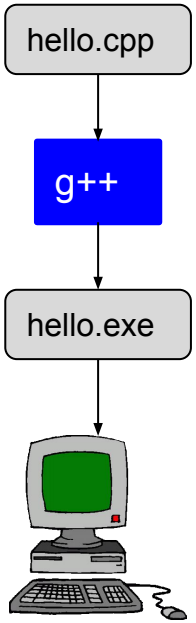
C is an imperative (procedural) language

Bjarne Stroustrup, the man who created C++ (1983)

C++ is an object-oriented language

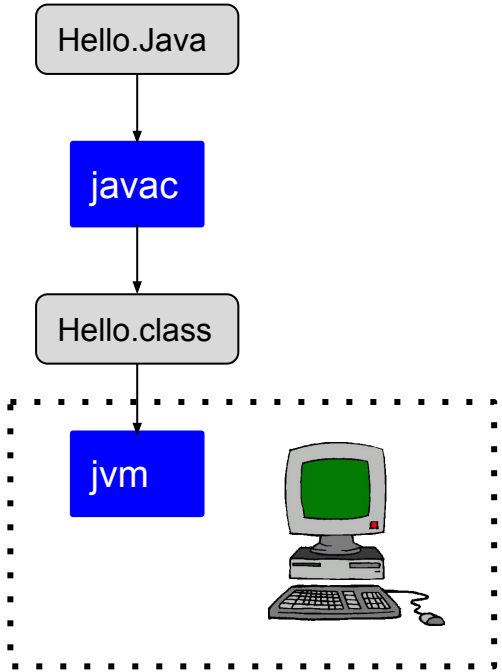
Compiler vs Interpreter

Compile to machine code



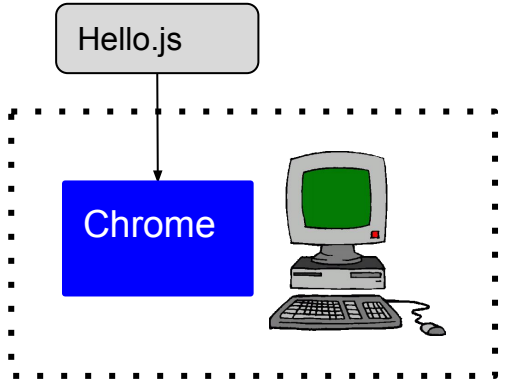
C, C++, Go, Fortran, Assembler, Pascal, Ada ...

Compile to bytecode



Java, Scala, Python, C#, ...

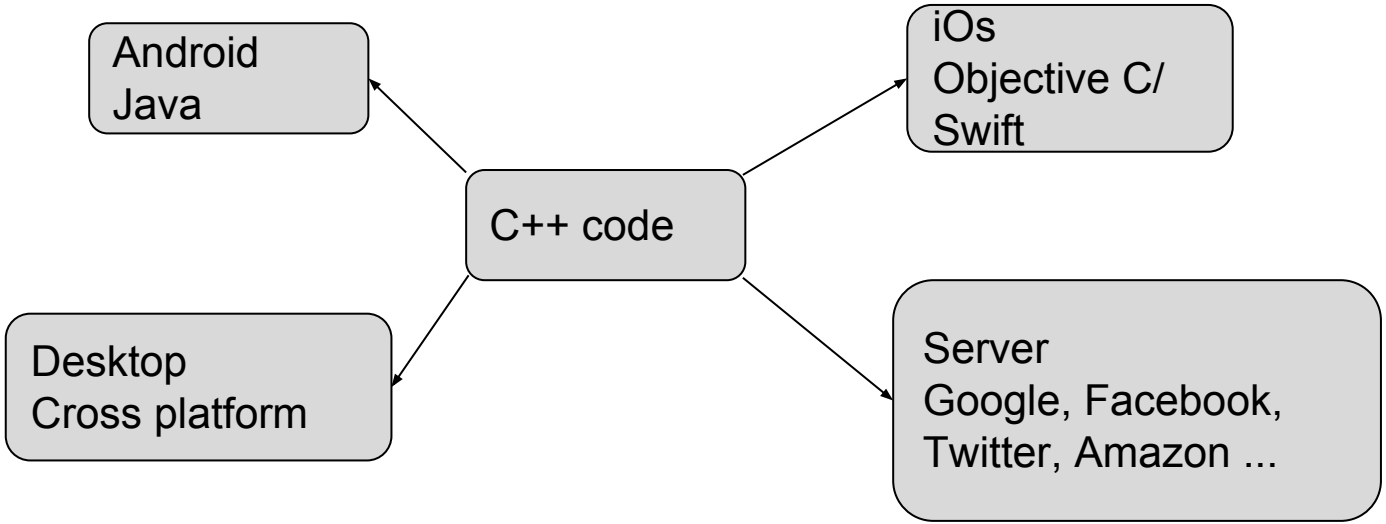
Interpret



JavaScript, Ruby, Matlab, PHP, Basic, Logo ...

Why C++ ?

- Creates .EXE files and libraries .A, .SO/.DLL - создает файлы .EXE и библиотеки .A, .SO/.DLL
- Can use any C libraries and OS API - может использовать любые C библиотеки и OS API
- Efficient, yet high level - Эффективный, но высокоуровневый
- Portable - Переносимый
- Used for desktop applications, mobile and server back-ends



C++ compilers

- **gcc** Linux, Windows (MinGW, MSYS, Cygwin)
- **clang** MacOS, iOS, Android, Linux, Windows
- **Microsoft CL** Windows
- **Intel** C++ Linux, Windows (BAD for AMD)
- **AMD AOCC** Linux

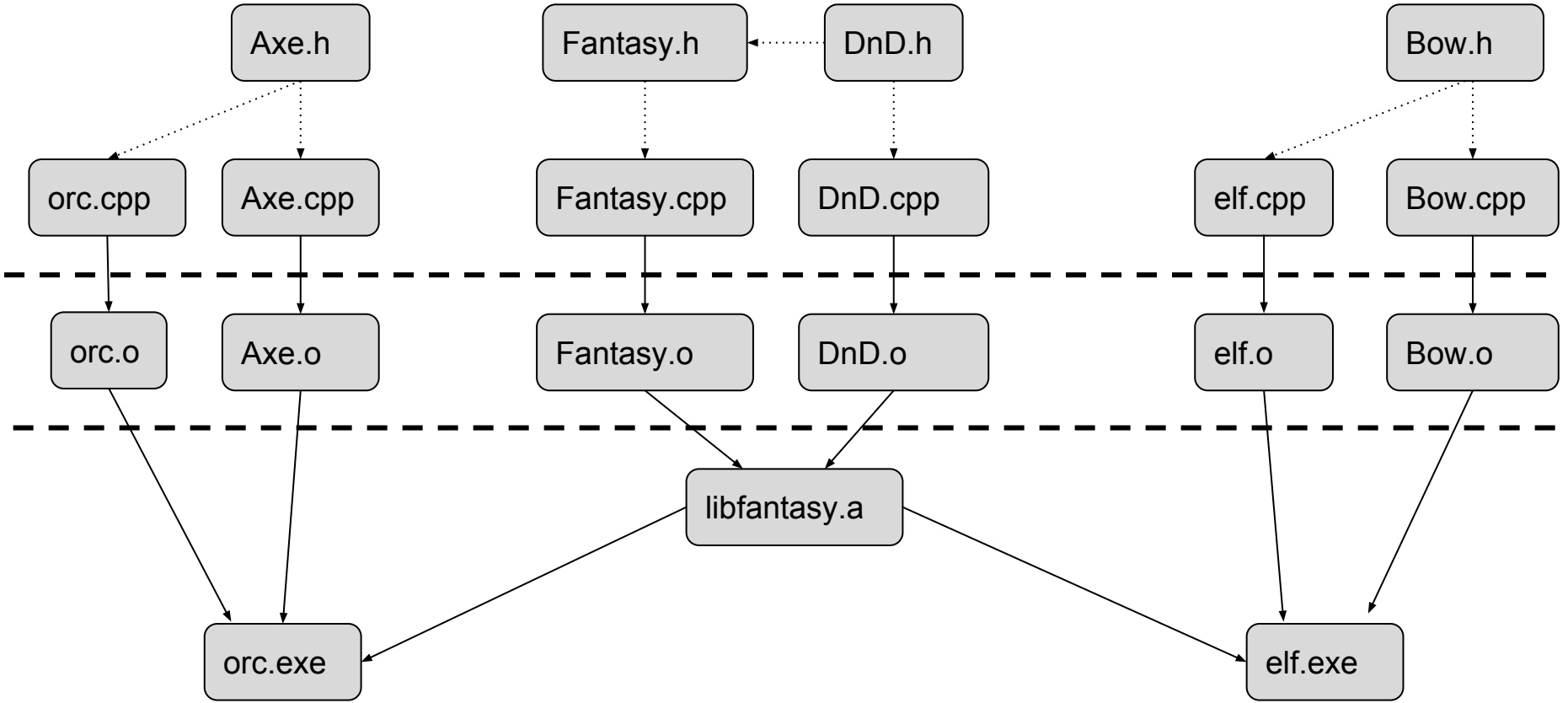
Что поставить под Windows ?

Для умных: **mingw-w64** под **msys2**, удобная система пакетов
http://www.msys2.org/
https://stackoverflow.com/questions/30069830/how-to-install-mingw-w64-and-msys2

Для ленивых:
Visual studio, Qt Creator, Code.Blocks , MinGW, что угодно ...

Обязательно !
CMake или пакетом под msys2, или Windows CMake

Build process



Build automation systems and IDEs

Build automation systems (Build tools) - Сборщики

- **CMake**
- make
- qmake
- nmake

IDEs - Среды разработки

- CLion (Commercial, slow)
- Qt Creator
- Code.Blocks
- KDevelop
- Visual Studio
- Xcode

For beginners - NO IDEs! Новичкам - никаких IDE!

Example 1_1: Hello world

The course repo https://github.com/agrechnev/cpp-course

```
#include <iostream>

int main(){
        std::cout << "Carthago delenda est." << std::endl;
        return 0;
}
```

To compile it - Чтобы скомпилировать

```
g++ -o hello hello.cpp
```

To build examples with CMake - Чтобы собрать с помощью CMake

```
mkdir build
cd build
cmake ..
cmake --build .
```

Example 1_2: C++ at a glance

Include several headers - Включаем несколько хедеров

```
#include <iostream>
#include <vector>
```

Command line arguments - Аргументы командной строки

```
int main(int argc, char **argv){
    using namespace std;
    cout << "argc = " << argc << endl << endl;
    for (int i = 0; i < argc; ++i)
        cout << "argv[" << i << "] = " << argv[i] << endl;
```

char ** - указатель на указатели (массив указателей) на тип **char** (1 байт)
char * - C-строка, заканчивается символом '\0' (обычно не используется в C++)
argv[i] - i-я строка в массиве **argv** (обращаемся к указателю как к массиву)
int - целый тип со знаком (обычно 4 байта = 32 бит)
argv[0] - имя программы, индексы массива начинаются с нуля !

Defining a class

Имя класса, *access modifiers* (модификаторы доступа) **public**, **private**, **protected**

```
class Warrior {
public:  // Public stuff goes here
```

Constructor - конструктор

```
Warrior(const std::string &name, const std::string &weapon, int age) :
            name(name),  // Init class field 'name' with argument 'name'
            weapon(weapon),
            age(age) {
        std::cout << "Constructor : " << str() << std::endl;
    }
```

const std::string & - Ссылка на константу (нельзя менять) типа **std::string** (C++ строка)
Такая передача параметров избегает *копирования*
name(name) - инициализация поля класса **name** параметром конструктора **name**
str() - метод класса (см. ниже)

Class Fields - Поля Класса

```
private: // Private stuff goes here
    std::string name;
    std::string weapon;
    int age;
}; // Note the semicolon ';' here, it's important !
```

Warrior::str() is *declared* (объявлен) inside class and *defined* (определен) outside it

```
std::string str() const;
... }; ...
std::string Warrior::str() const{
    return "name = " + name + ", weapon = " + weapon + ", age = " +
std::to_string(age);
}
```

Getter

```
const std::string &getName() const {
        return name;
    }
```

Create **std::vector** *container* of **Warrior** objects - создать контейнер **std::vector**

```
vector<Warrior> warriors{
     {"Brianna", "Lightsaber", 17},
     {"Sita", "Spear", 15},
     {"Jean Grey", "Telekinesis", 25},
     Warrior("Ashe", "Zodiac Spear", 19)
};
```

Add a few more **Warrior** objects - Добавить объекты

```
warriors.emplace_back("Jaheira", "Club+5", 110); // Construct in-place. Best !
Warrior wz("Zoe Maya Castillo", "Fists", 20);
warriors.push_back(wz); // OOPS! A copy operation!
warriors.push_back(Warrior("Casca", "Sword", 24)); // move operation
```

emplace_back - создать новый объект прямо внутри контейнера
push_back - добавить существующий объект в контейнер (copy or move)
Warrior wz("Zoe Maya Castillo", "Fists", 20); - Создать локальную переменную типа **Warrior** с параметрами конструктора

Print all elements of std:::vector

Range **for** loop
Использует ссылку **const Warrior &** чтобы избежать копирования

```
for (const Warrior & w: warriors)
    cout << w.str() << endl;
```

Traditional **for** loop
Индекс начинается с нуля !

```
for (int i = 0; i < warriors.size(); ++i)
        cout << warriors[i].str() << endl;
```

References. Литература.

Books:

1. S.B. Lippman, J. Lajoie, B.E. Moo, *C++ primer* (2012).
2. Scottt Meyers, *Effective Modern C++* (2014).
3. N.M. Josutttis, *The C++ Standard Library* (2014).
4. Bjarne Stroustrup, *Programming: Principles and Practice Using C++* (2014).

Read only books/tutorials on C++ 11 or later ! C++ 98 = EVIL !!!
Читайте только литературу по C++ 11 или новее!

Other resources:

1. http://en.cppreference.com/w/
2. http://www.cplusplus.com/
3. https://stackoverflow.com/
4. https://www.google.com
5. https://stackoverflow.com/questions/388242/the-definitive-c-book-guide-and-list