

Программирование на языке C++

...

Лекция 1. История и обзор языка

Эволюция языков программирования



Машинный код

Машинный код — система команд (набор кодов операций) конкретной вычислительной машины, которая интерпретируется непосредственно процессором или микропрограммами этой вычислительной машины.

Программа «Hello, world!» для процессора архитектуры x86:

BB 11 01 B9 0D 00 B4 0E 8A 07 43 CD 10 E2 F9
CD 20 48 65 6C 6C 6F 2C 20 57 6F 72 6C 64 21

Код перевозки штуки	Значение кода	Помощь на улице счета
.	0	0
.	1	
.	2	
.	3	
.	4	
.	5	
.	6	
.	7	
.	8	
.	9	
.	+ водостру.	
.	+ водостру.	
.	+ дебетов.	
.	+ дебетов.	
.	Зачетан	
.	Дебетов. пробит	
.	Перевод адреса	
.	Восприимчивость как	
.	перевод адреса	
.	Зачетан	
.	Восприимчивость	
.	как зачет	
.	пробит	
.	Восприимчивость	
.	как пробит	
.	Принимается (вместе с	
.	мужиком)	

Низкоуровневые языки программирования. Assembler

Язык ассемблера (*assembly language*) —
машинно-ориентированный язык
программирования низкого уровня.

Его команды прямо соответствуют
отдельным командам машины или их
последовательностям

Платформено-зависимый. Языки ассемблера
для различных аппаратных платформ
несовместимы, хотя могут быть в целом
подобны.

```
.global _start
.text
_start:
    # write(1, message, 13)
    mov    $1, %rax           # system call 1 is write
    mov    $1, %rdi           # file handle 1 is stdout
    mov    $message, %rsi     # address of string to output
    mov    $13, %rdx          # number of bytes
    syscall                   # invoke operating system to do the
write

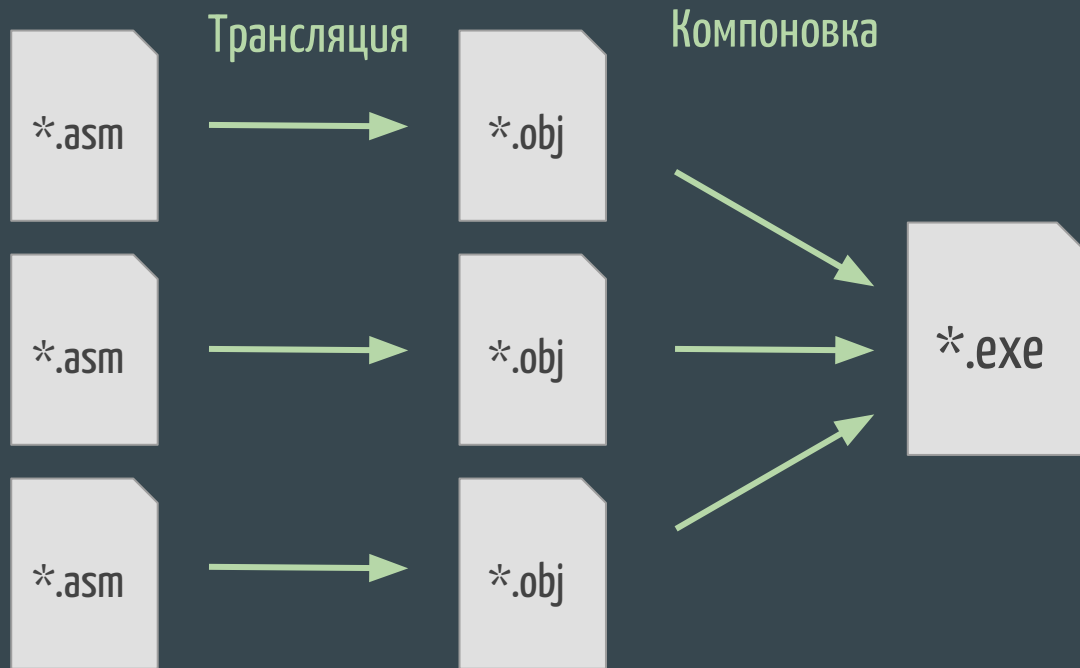
    # exit(0)
    mov    $60, %rax          # system call 60 is exit
    xor    %rdi, %rdi         # we want return code 0
    syscall                   # invoke operating system to exit

message:
    .ascii "Hello, world\n"
```

Assembler. Трансляция и компоновка

Перевод программы на языке ассемблера в исполняемый машинный код производится *ассемблером* — программой-транслятором, которая и дала языку ассемблера его название.

Команды языка ассемблера один к одному соответствуют командам процессора.



Экскурс в историю. Язык Си

1973 Дэннис Ритчи разработал язык Си изначально для реализации ОС UNIX

Ритчи известен как создатель языка программирования Си и ключевой разработчик ОС Unix

Соавтор книги «Язык программирования C», известный как «K/R»



Язык Си



Конструкции близко сопоставляются типичным машинным инструкциям

Нашёл применение в проектах, для которых был свойственен язык ассемблера, в том числе в ОС

Целью языка было облегчение написания больших программ с минимизацией ошибок по сравнению с ассемблером

Оказал существенное влияние на развитие индустрии программного обеспечения, а его синтаксис стал основой для таких языков программирования, как C++, C#, Java и Objective-C.

История языка Си неразрывно связана с историей операционной системы UNIX. Эта система, как и большинство входящих в нее утилит, написана на Си.

Возможности языка Си



Enum

Функции

Структуры

Нуль-терминированные строки

Массивы

Препроцессор

Синонимы типов

Работа с памятью и указатели

```
#include <stdio.h>

int main(void)
{
    #define SQR(x) (x * x)
    printf("%d", SQR(4 + 1)); // ?
    return 0;
}
```


Рождение языка C++

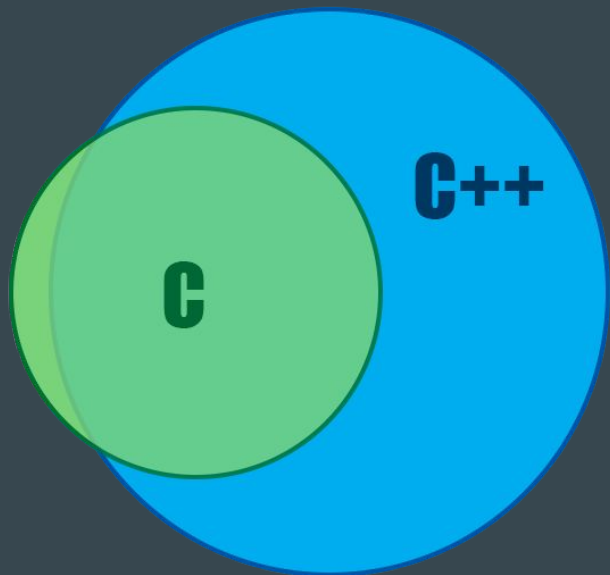
Начало 1980-х. Бьёрном Страуструпом, сотрудником Bell Laboratories для собственных нужд разработана **надстройка над языком Си**.

Изначально получил название **C with classes**

Синтаксис C++ был основан на синтаксисе языка Си с попыткой максимального сохранения совместимости



История языка C++



1983 переименование языка из “C with Classes” в C++

- Виртуальные функции;
- Перегрузка функций и операторов;
- Ссылки;
- Константы

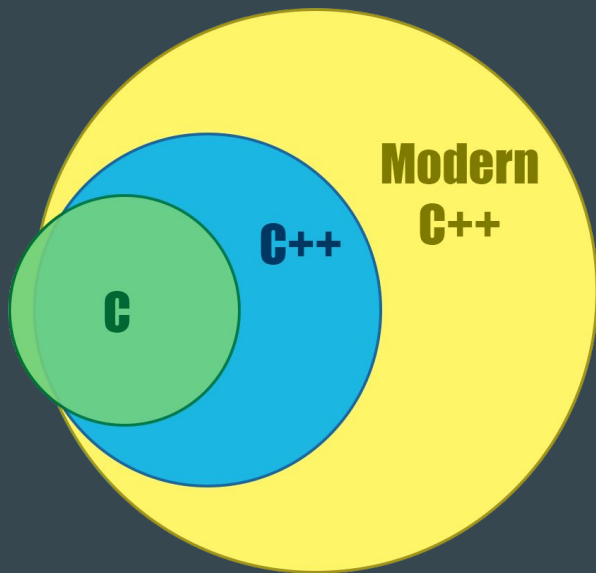
1985 Первый коммерческий выпуск

1989 Выход C++ версии 2.0.

- Наследование;
- Абстрактные классы;
- Статические функции-члены;
- private члены

1998 Опубликован первый стандарт языка C++98

C++ сегодня (C++11/14/17)



Лямбда-функции

Умные указатели

Семантика перемещения

Auto

Range-based циклы

Строго типизированный enum

Nullptr

Override & final

std::optional

Std::filesystem

Многопоточное программирование

Что стоит писать на C++?

Программы с высокими требованиями к ресурсам компьютера: памяти и ЦП

Что пишут на C++:

- Операционные системы
- Драйверы устройств
- Научные программы, в том числе CAD (Computer Aided Design)
- Симуляторы и игры
- Высоконагруженные сервера 24x7

Что не стоит писать на C++?



Библиотеки

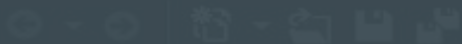
- CRT (C Runtime Library)
- Стандартная библиотека C++
- Стандартная библиотека шаблонов (STL)
- Boost
- Для работы с ОС (WinAPI, POSIX)
- Для работы с UI (QT, WxWidgets)





StephenPrata1 - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help



Build Solution

Ctrl+Shift+B

Rebuild Solution

Clean Solution

Build full program database file for solution

Run Code Analysis on Solution

Alt+F11



Build StephenPrata1

Rebuild StephenPrata1

Run Code Analysis on StephenPrata1

Project Only

Profile Guided Optimization

Batch Build...

Configuration Manager...



Compile

Ctrl+F7

Run Code Analysis on File

Ctrl+Shift+Alt+F7



Сборка программы

Server Explorer Toolbox

limits

Source.cpp

StephenPrata1

1 #inclu

2 #inclu

3 #inclu

4

5 t ma

6

7 st

8

9 in

10 in

11 in

12 st

13 st

14 st

15 st

16 std::cout << "Varriable values is 0" << value1 << " 0" << valu

er

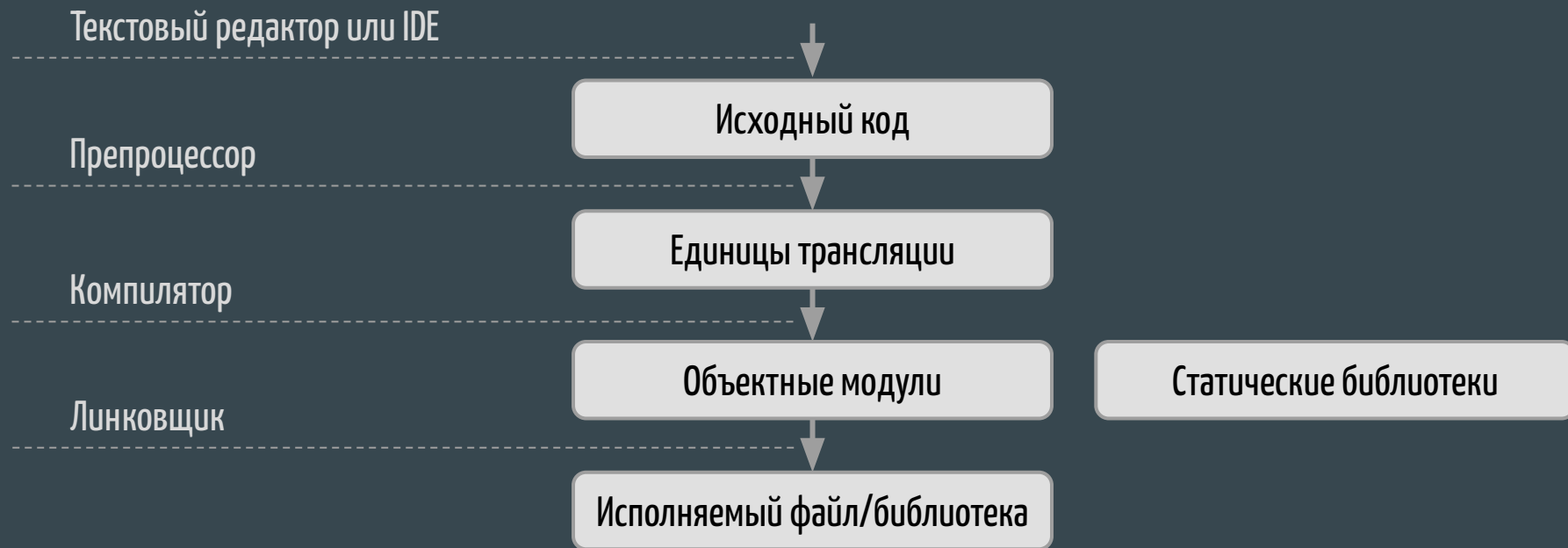
dl;

alue2

<< va

<< valu

Сборка программы



Препроцессор

Преобразовывает исходный файл
программы для дальнейшего
компилирования.

```
g++ -E ./main.cpp -o output.ii
```

```
#include <iostream>
```

```
#define PI 3.1415
```

```
#define MIN(a, b) (((a) < (b)) ? (a) : (b))
```

```
int main()
```

```
{
```

```
#ifdef ANDROID_OS
```

```
    std::cout << "Android OS platform spec output" << std::endl;
```

```
#else
```

```
    std::cout << "Another platform output" << std::endl;
```

```
#endif
```

```
    std::cout << "MIN(3, 7) is " << MIN(3, 7) << std::endl;
```

```
    return 0;
```

```
}
```


Средства разработки

- IDE (MS Visual Studio, Xcode, Eclipse, CLion, QtCreator)
- Компиляторы (MSVC, g++, clang)
- Отладчики (MSVS, gdb)
- Автоматизация сборки (cmake, make)
- Текстовые редакторы (Notepad++, Sublime Text 3, MS Visual Studio Code)

Создание проекта Visual Studio

Конфигурации Release/Debug

Средства отладки кода

```
#include <iostream>
```

```
int factorial(int n)  
{
```

```
    int result{ 1 };
```

```
    while(n > 1)
```

```
    {
```

```
        result *= n--;
```

```
    }
```

```
    return result;
```

```
}
```

```
int main() // Entry point
```

```
{
```

```
    const int n = 7;
```

```
    std::cout << n << "! = " << factorial(n) << "\n";
```

```
    return 0;
```

```
}
```

Параметры командной строки

```
int main(int argc, const char* argv[])
{
    std::cout << "Program name is " << argv[0] << std::endl;
    std::cout << "Input params count " << argc << std::endl;

    if(argc == 1) {
        std::cerr << "Programm param must be factorial arg" << "\n";
        return 1;
    }

    int n = atoi(argv[1]);
    std::cout << n << "! = " << factorial(n) << std::endl;
    return 0;
}
```