

# **Технология программирования**

Осенний семестр 2016 г.

Кафедра управления и информатики НИУ «МЭИ»

# Состав курса

## ■ Язык C++

- Встроенные и стандартные типы данных, языковые конструкции
- Объектно-ориентированное программирование
- Работа с указателями, динамической памятью, блоками данных
- Работа с окружением (файлами и пр.)

## ■ Алгоритмы и структуры данных

- Анализ сложности операций
- Обзор основных алгоритмов и структур данных

## ■ Средства разработки

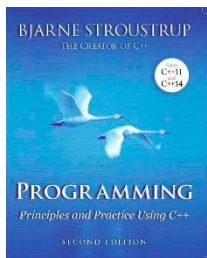
- Системы контроля версий (Git)
- Системы сборки (make, CMake)
- Модульное тестирование и отладка



# Учебный процесс

- Лекции: 8 шт. первые 8 недель.
- Лабораторные работы: 8 шт. раз в 2 недели.
- Зачет с оценкой по результатам ЛР.
- <http://uui.mpei.ru/study/courses/sdt> — всё,
  - но потом :-)
- Преподаватели:
  - Козлюк Дмитрий Александрович (лекции, ЛР)
  - Никитин Вадим Владимирович (лекции, ЛР)
  - Мохов Андрей Сергеевич (ЛР)

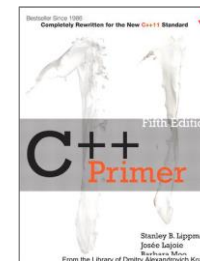
# Литература



- ❑ Bjarne Stroustrup. *Programming: Principles and Practices using C++* (2014 или 2011):

Страуструп Б. Программирование: принципы и практика использования C++. — СПб., «Вильямс». — 2012 г.

- ❑ Stanley B. Lippman. *C++ Primer*, 5<sup>th</sup> Ed. Липман С. C++: базовый курс, 5 изд.



[стандарт]

- ❑ ISO/IEC 14882—2014 (Draft)  
<http://open-std.org/JTC1/SC22/WG21>

- Сайт «C++ Reference»

[K&R]

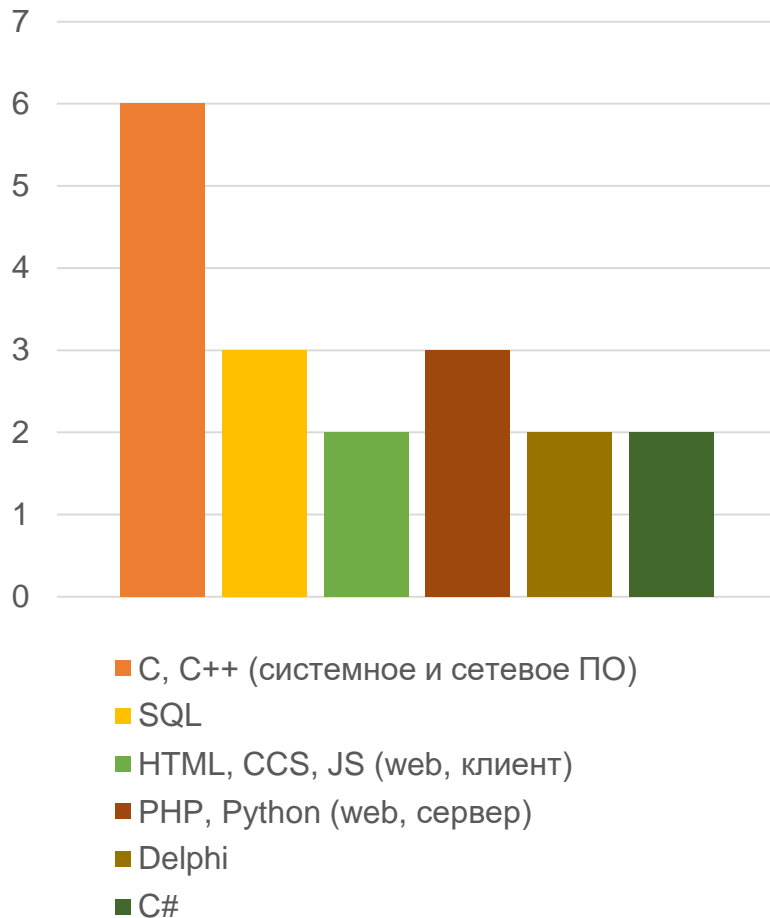
- ❑ Brian Kernighan & Dennis Ritchie. *The C Programming Language*, 2<sup>nd</sup> Edition.

Керниган Б., Ритчи Д. Язык программирования C.

# Основы языка C++

# Почему C++?

Выпускники используют  
(2014)



# Почему не Python/SQL/?..

Системы промышленной автоматизации:

- низкого уровня:
  - встраиваемые (спец. устройства);
  - системное ПО (драйверы, ...);
- реального времени (производительные);
- сложные, но надежные.

# Привлекательность C++

1. Zero-overhead abstractions

2. Maps directly to hardware

- 1 + 2 = программы одновременно понятны и быстры.
- Совместимость:
  - высокая совместимость с языком C;
  - обратная совместимость (backwards compatibility).
- Гибкое управление ресурсами:
  - доступ к выделению памяти и т. п. вручную;
  - возможность автоматического управления ресурсами.
- Солидная поддержка, стандартизация.

# От идеи к воплощению: выразительность средств языка

«Взять» = «объявить переменную»;  
есть ли объявления переменных?

Есть ли у переменных типы?  
Когда они указываются?  
Когда и как контролируются?

«Взять целое число, изначально нуль...»

Можно ли указать значение при объявлении?

**var** x = 0;

x = 0

**var** X: Integer;

**const** X: Integer := 0;

**int** x = 0;




# Структура программы на C++

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.      // Вывод строки на экран.
6.      cout << "Hello, world!\n";
7.  }
```

# Типы данных и переменные

Тип данных определяет формат данных и операции над значением.

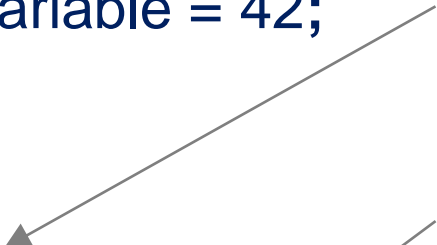
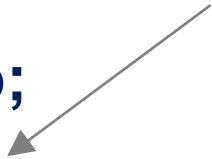
Можно и желательно указывать начальное значение.



```
int negative = -42;           // целое со знаком
unsigned int count = 66;      // целое без знака
double pi = 3.14159265358979; // действительное
char letter = 'A';           // символ
string name = "Dmitry";      // строка
bool condition = true;       // логическое значение
```

См. подробности в раздаточном материале.

# Неизменяемые переменные

- Переменные, которые нельзя изменить.
  - Значение должно быть задано сразу.
- **const int variable = 42;**
  - **int const variable = 42;**  Может зависеть от переменных.
- **int a = 10;**  
**int b = 20;**  
**int const sum = a + b;** 
  - Не может зависеть от переменных и **const**.
  - Может зависеть от **constexpr**.
- **double constexpr PI = 3.14;**
  - Только для констант этапа компиляции ( $\pi$ ,  $e$ , ...).

# ВВОД И ВЫВОД

1. `#include "sdt.h" // Файл из [PPP], подключает нужное.`

2.

3. `int main()`

4. `{`

5. `cout << "Enter your name and age: ";`

6. `string name;`

7. `unsigned int age;`

8. `cin >> name >> age;`

9. `const unsigned int next = age + 1;`

10. `cout << "Hello, " << name << ", next year "`

11. `<< "you will be " << next << " years old.\n";`

12. `}`

dmitry@lab: ~/mpei/c++2014

Enter your name and age: **Shepard 34**

Hello, Shepard, next year you will be 35 years old.

# Операторы и выражения

- Арифметика и отношения как в Pascal.



$9 / 5 == 1$   
 $9.0 / 5 == 1.8$

- Остаток от деления:  $a \% b$
- Проверка равенства:  $x == y$ 
  - неравенства:  $x != y$



- Присваивание:  $n = 5;$        $k = p = 0;$ 
  - До присваивания значение не определено!

- Сокращенное присваивание:

$q += 5;$      $// q = q + 5$

$v *= 2;$      $// v = v * 2$

- Инкремент и декремент:

$int i = 5;$

$++i;$        $// 6, i == 6$

$i--;$        $// 6, i == 5$

- Логические операции:

- «И»:       $\&\&$

- «ИЛИ»:     $||$

- «НЕ»       $!$

- $n < 20 \&\& (n == 2 || n > 9)$

# Проверки условий

❑ **if** (*условие*)  
    *действие-1*  
**else**  
    *действие-2*

❑ **if** (*условие-1*)  
    *действие-1*  
**else if** (*условие-2*)  
    *действие-2*  
**else**  
    *действие-3*

❑ **if** (t == 100) {  
    c++;  
} **else**  
    t \*= 2;



❑ **if** (x % 2)  
    cout << "odd";  
**else if** (x % 3)  
    q++;  
**else** {  
    cout << x << '\n';  
}

# Циклы **while** и **do...while**

## **C++**

- ☐ **while** (*условие*)  
*тело цикла*
- ☐ **do**  
*тело цикла*  
**while** (*условие*);
- ☐ **break**;
- ☐ **continue**;

## **Pascal, Delphi**

- ☐ **while** *условие* **do**  
*тело цикла*
- ☐ **repeat**  
*тело цикла*  
**until not** *условие*;
- ☐ **Break**;
- ☐ **Continue**;

# Цикл **for**

**for** (*инициализация; условие; действие*)  
*тело цикла*

```
for (int i = 0; i < 10; ++i)  
    cout << i << '\n';
```

Подобно коду на C++...

```
int i = 0;  
while (i < 10) {  
    cout << i << '\n';  
    ++i;  
}
```

Подобно коду:

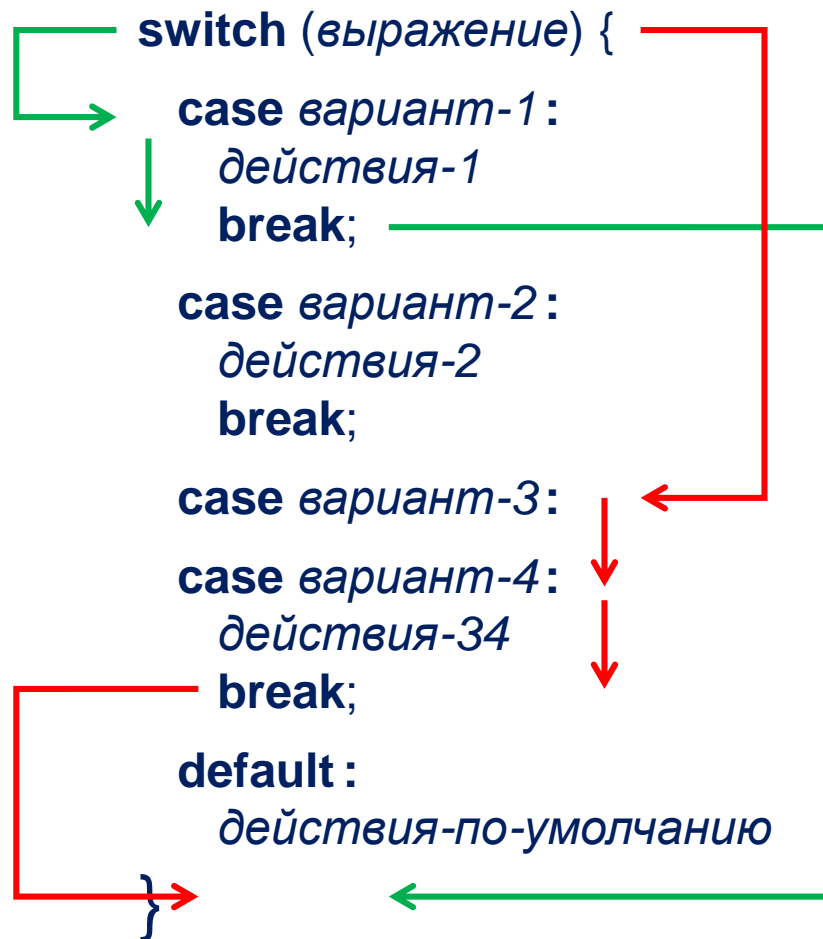
```
инициализация;  
while (условие) {  
    тело цикла;  
    действие;  
}
```

...и на Pascal:

```
for l := 0 to 9 do  
    WriteLn(l);
```



# Переключатель (switch)

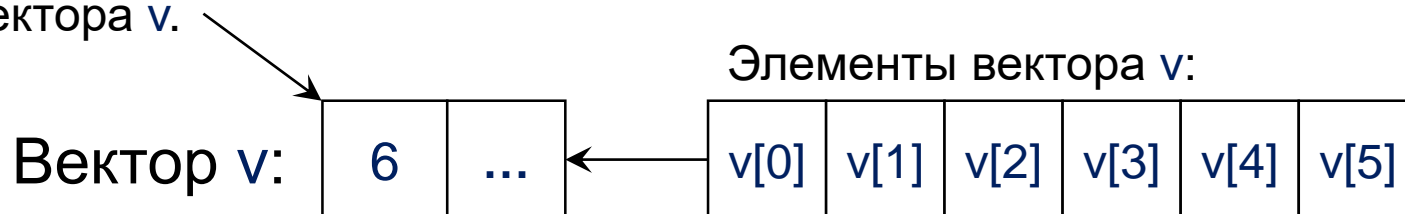


```
char answer;  
cout << "Yes or No? ";  
cin >> answer;  
  
switch (answer) {  
  case 'y':  
  case 'Y':  
    // Ответ «Да».  
    break;  
  case 'n':  
  case 'N':  
    // Ответ «Нет».  
    break;  
  default:  
    cout << "Type Y or N!";  
}
```

# Тип `vector<T>`, «вектор»

«Динамический массив» элементов типа `T`.

Размер вектора `v`.



В круглых скобках передаются значения.

```
vector<double> v(10);
```

В угловых скобках «передаются» типы.

Смысл:

«выполнить *операцию* с вектором `v`, используя *аргументы*».

`v.операция( аргументы );`

# Использование **vector<T>**

- Вектор из чисел:

```
vector<double> numbers;
```

- Вектор из 5 чисел:

```
vector<double> numbers(5);
```

```
vector<double> numbers {1, 2, 3, 4, 5 };
```

- Вектор из 5 нулей:

```
vector<double> zeros(5, 0);
```

- Третье число (нумерация элементов с 0):

```
double third = numbers[2];
```

- Узнать размер:

```
unsigned int size = numbers.size();
```

# Изменение `vector<T>`

- Изменить элемент:  
`numbers[3] = 9;`
- Добавить элемент в конец:  
`numbers.push_back(42.99);`
- Удалить последний элемент:  
`numbers.pop_back();`
- Изменить размер до 10 элементов:  
`numbers.resize(10);`
- Очистить вектор:  
`numbers.clear();`

# Статистические расчеты

```
vector<double> xs;  
  
int n;  
cin >> n;  
xs.resize(n);  
for (int i = 0; i < n; ++i)  
    cin >> xs[i];  
  
double mean = 0;  
for (double x : xs)  
    mean += x;  
mean /= xs.size();
```

```
double variance = 0;  
if (n > 1) {  
    for (double x : xs) {  
        double d = x - mean;  
        variance += d * d;  
    }  
    variance /= n - 1;  
}
```

# Литература к лекции

- *Programming Principles and Practices Using C++:*
  - глава 1: о программировании в целом;
  - главы 2—4: типы данных, выражения, конструкции;
  - упражнения (drills).
- *C++ Primer:*
  - глава 1: пошаговое написание программы;
  - главы 2—5: типы данных, выражения, конструкции;
  - упражнения.
- *C++ Reference* (<http://cppreference.com>)