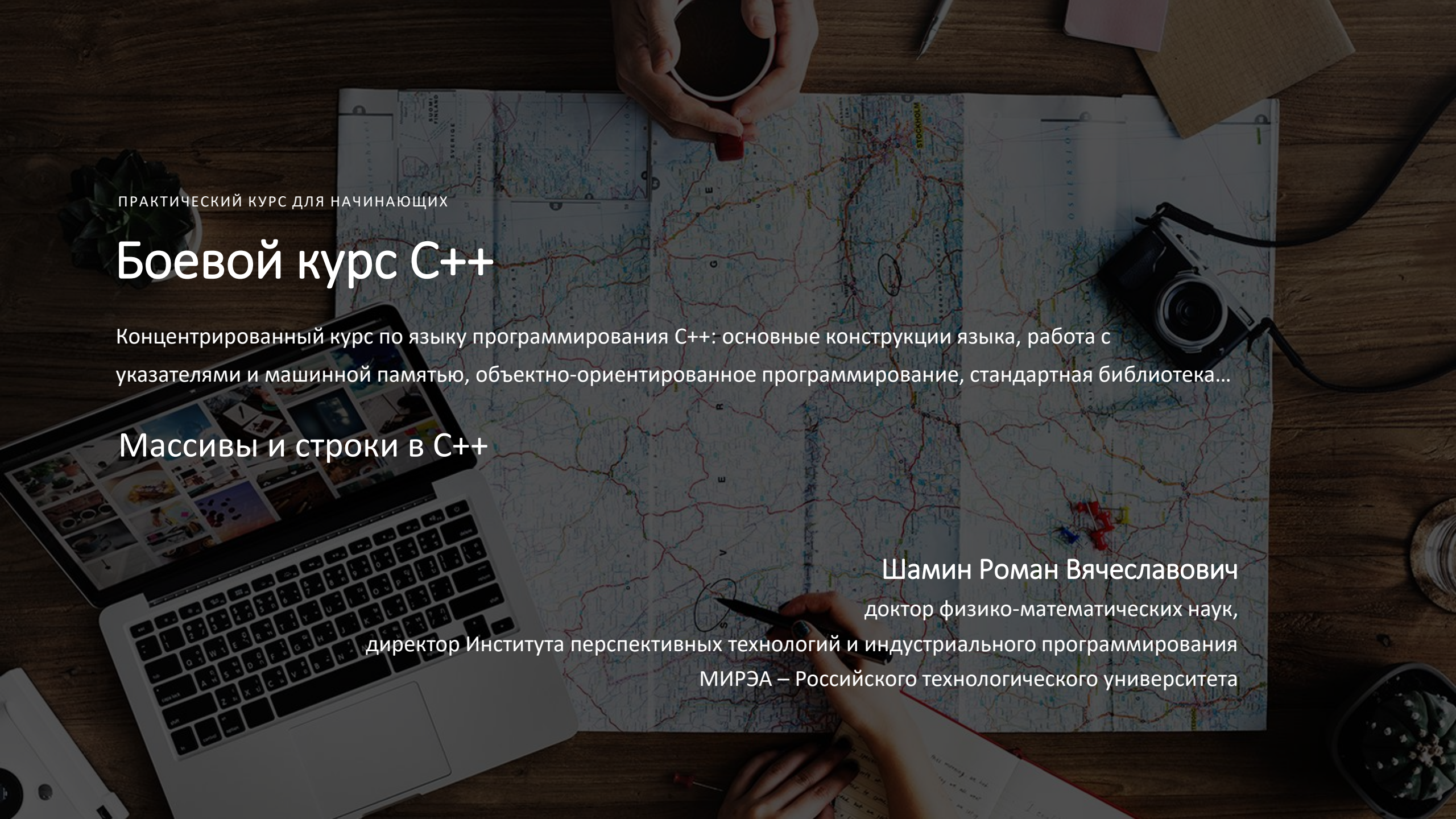


ПРАКТИЧЕСКИЙ КУРС ДЛЯ НАЧИНАЮЩИХ

Боевой курс C++

Концентрированный курс по языку программирования C++: основные конструкции языка, работа с указателями и машинной памятью, объектно-ориентированное программирование, стандартная библиотека…

Массивы и строки в C++

Шамин Роман Вячеславович

доктор физико-математических наук,

директор Института перспективных технологий и индустриального программирования

МИРЭА – Российского технологического университета

Массив – это линейное представление данных одного типа. Массив можно понимать как не одну переменную, а целый набор переменных, доступ к которым осуществляется с помощью целочисленного индекса.

В языке C++ перед использованием массива его нужно объявить. При этом указывается:

- имя массива (как переменную);
- тип данных;
- размер массива (константное выражение).

После объявления массива для доступа к его элементам используем индекс, который начинается с нуля!

```
const int N = 10; // размерность массива
int A[N]; // объявляем и создаем массив
for (int i = 0; i < N; i++)
{
    A[i] = i + 1; // заполняем массив
}
for (int i = 0; i < N; i++)
{
    cout << A[i] << endl; // выводим массив
}
```

Очень и очень важно! Никогда не объявляете массив, указывая конкретное число: int A[10]; ! Создавайте константу для размера массива. Даже если массив хранит дни недели☺

По стандарту языка C++ размер массива должен быть константой, а не переменной. Однако многие компиляторы позволяют создавать массив с динамическим размером. Мы так делать не будем, но в следующей лекции покажем как по стандарту создавать динамические массив.

Посмотреть видео

В языке C++ нет контроля выхода индекса за границы массива. Это серьезная проблема, поскольку программист обязан самостоятельно контролировать границы массива, работая без страховки. Что произойдет, если индекс выйдет за границу массива? В лучшем случае, вы получите случайные данные или нарушите значения других переменных, вероятно произойдет ошибка выполнения программы…

Именно поэтому нужно всегда использовать одну и ту же константу для размера массива при объявлении и при работе с массивом.

Если у Вас есть два массива: `int A[N], B[N];` то Вы не сможете использовать выражения `B = A; // ошибка!`

Копирование массивов нужно делать руками:

```
const int N = 10; // размерность массива
int A[N], B[N]; // объявляем два массива
for (int i = 0; i < N; i++)
{
    A[i] = i + 1; // заполняем первый массив
}
for (int i = 0; i < N; i++)
{
    B[i] = A[i]; // копируем массив поэлементно
}
for (int i = 0; i < N; i++)
{
    cout << B[i] << endl; // выводим второй массив
}
```

Аналогично и сравнение двух массивов нужно делать самому, хотя есть и другие средства в STL (стандартной библиотеке), которую мы рассмотрим позднее.

Язык C++ поддерживает многомерные массивы. Для этого при определении указывается объем массива по всем измерениям.

```
const int N = 5, M = 10; // размерности массива
double A[N][M]; // объявляем двумерный массива
for (int i = 0; i < N; i++)
{
    for (int j = 0; j < M; j++)
    {
        A[i][j] = i * j; // заполняем массив с помощью вложенного цикла
    }
}
for (int i = 0; i < N; i++)
{
    for (int j = 0; j < M; j++)
    {
        cout << A[i][j] << "\t"; // выводим двумерный массив
    }
    cout << endl;
}
```

Будьте осторожны с массивами большой размерности. Например, массив

```
const N = 1000;
double A[N][N][N]; // трехмерный массив, всего миллиард элементов по 8 байт
```

потребует примерно 8 гигабайт памяти.

Инициализация массивов и перебор элементов

В языке C++ можно инициализировать при объявлении, указав значения элементов массива. При этом можно не указывать размер массива – он будет вычислен автоматически:

```
double X[] = {3.14, 2.71, 0, -1.5, 2.41}; // можно не указывать размерность
```

Для многомерных массивов также можно использовать инициализацию:

```
const int N = 3;
int A[][N] = { {10, 5}, {-4, 0}, {7, 15} }; // размерность внутреннего массива необходимо указать
```

Если мы не указывали размерность массива, то как нам потом ее узнать, чтобы организовать перебор? Для перебора массивов можно использовать специальный вариант цикла, который по духу близок циклу в языке Python:

```
    double X[] = {3.14, 2.71, 0, -1.5, 2.41}; // можно не указывать размерность

    int count = 0; // создать счетчик
    for (double x : X) // перебираем массив
    {
        cout << x << "\t"; // печатаем элемент
        count++; // увеличиваем счетчик
    }
    cout << endl << "Количество элементов = " << count;
```

Этот цикл действует следующим образом. Переменная x последовательно принимает все значения из массива.

Как мы уже говорили, в языке C++ нет типа данных для хранения строк. Классическим способом организации строк в языке C было использование массивов из char, где строка заканчивалась нулевым символом. При этом для работы со строками необходимо вызывать отдельные функции. Такие строки называются C-подобными строками.

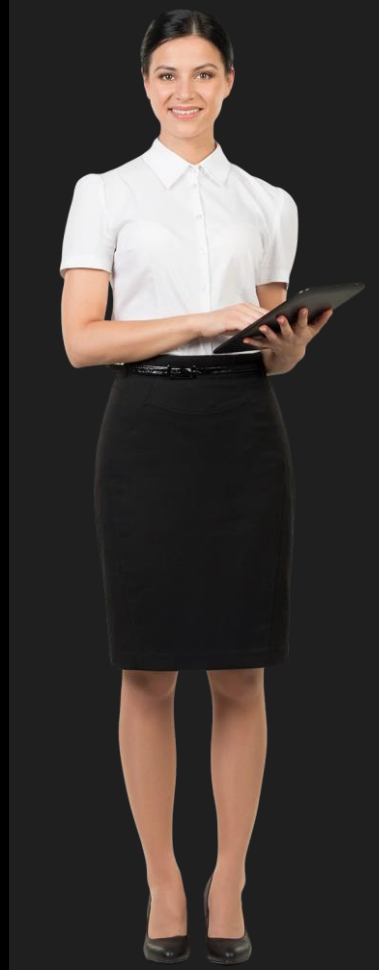
Я стандартной библиотеке C++ есть класс string, который делает возможным работу со строками по-человечески. По сути, это новый тип данных, который имеет у себя различные функции для работы со строками. Обратите внимание, что для ввода строки из cin используется функция getline. Кроме того, могут быть проблемы с русскими буквами при вводе текста через терминал.

Для подключения строк используйте `#include <string>`

```
cout << "Введите две строки > " << endl;
string s1, s2, s3; // определяем три строки
getline(cin, s1); // вводим первую строку
getline(cin, s2); // вводим вторую строку
if (s1 == s2) // сравниваем как обычно
{
    cout << "Строки одинаковые!" << endl;
}
s3 = s1 + " & " + s2; // конкатенация строк
cout << s3 << endl; // выводим строку
```

К отдельным символам строк можно обращаться по индексу как с массивом. Если после переменной типа string набрать точку, то VS Code покажет большое количество функций для поиска в строке, выделения подстрок и т.д. Попробуйте поиграться с ними. Заметим, что количество символов можно получить через функцию length():

```
cout << "Длина строки " << s3.length() << " символов"; // вычисление длины строки
```



Передача массивов в функции

Массивы и строки можно передавать в качестве параметров функций, но при этом эти объекты передаются не по значению, а по ссылке, поэтому изменяя их в функции, мы изменим и оригинал!

```
const int N = 10; // размер массива

void calc(double B[N]) // массив в качестве параметра
{
    cout << B[1] << endl; // выводим один элемент
    B[1] = 2.71; // меняем элемент в массиве
}

int main()
{
    double A[N]; // создаем массив
    A[1] = 3.14; // устанавливаем значение элемента массива
    calc(A); // вызываем функцию
    cout << A[1] << endl; // проверяем значение элемента массива – оно изменилось!
}
```

Мы видим, что массив A был изменен при выполнении функции calc. Почему так произошло мы объясним в следующей лекции.

Скачать текст программы

Посмотреть видео