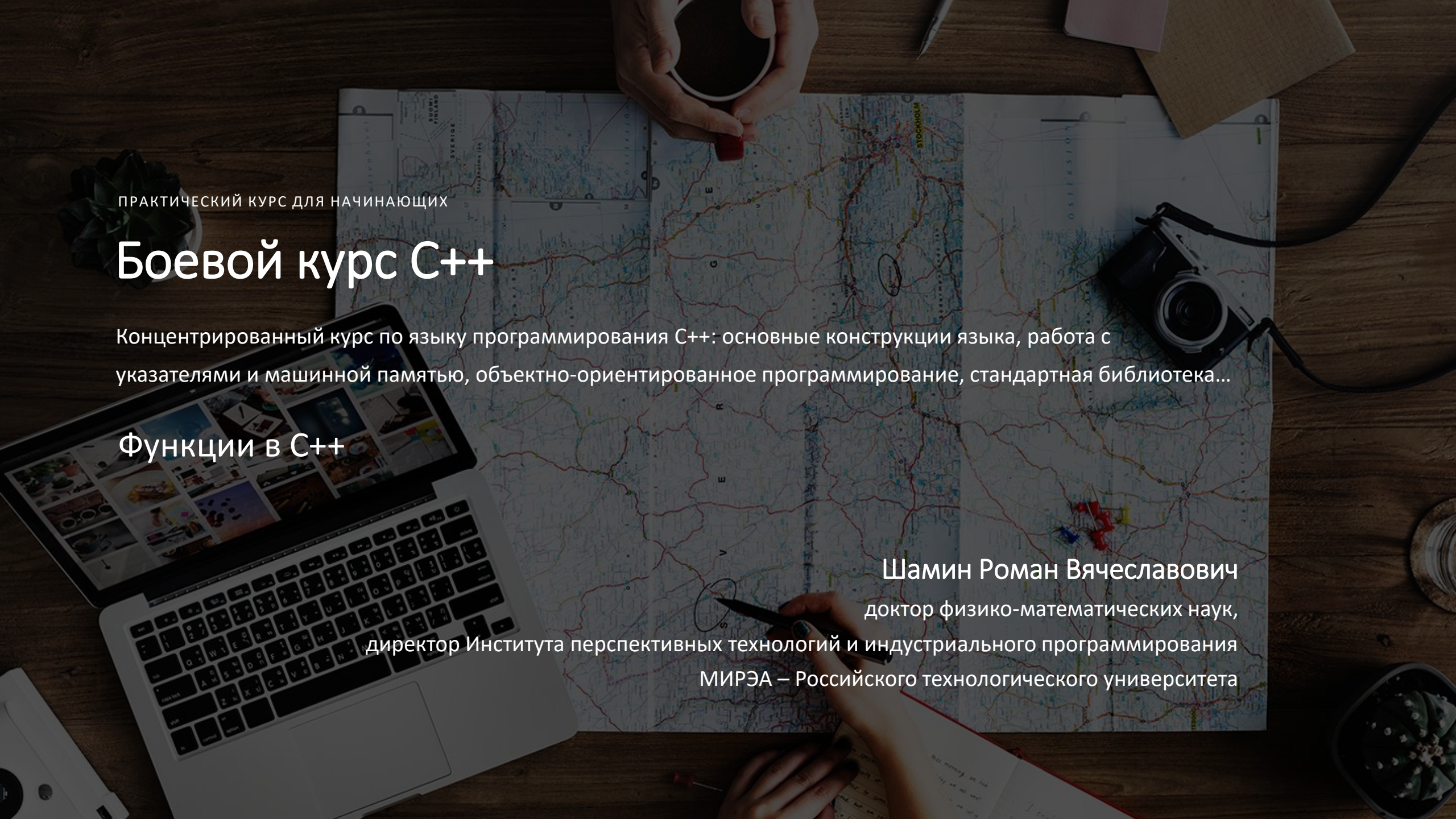

ПРАКТИЧЕСКИЙ КУРС ДЛЯ НАЧИНАЮЩИХ

Боевой курс C++

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

Функции в C++

Шамин Роман Вячеславович
доктор физико-математических наук,
директор Института перспективных технологий и индустриального программирования
МИРЭА – Российского технологического университета

Определение функций

Важнейшим принципом построения программ является разбиение программы на отдельные модули, когда каждый модуль имеет входные параметры, алгоритм работы с этими параметрами и выдает результат выполнения. В языке C++ такими модулями являются функции и классы. Классы мы будем изучать позднее, когда будем рассматривать объектно-ориентированное программирование. Использование функций – это парадигма структурного программирования.

В языке C++ перед использованием функцию следует объявить. При объявлении функции указывается:

- имя функции (как переменной);
- тип возвращаемого значения (только одно значение);
- параметры (параметров может быть сколь угодно много, для каждого указывается тип и название).

Объявление функции может сразу же включать в себя и реализацию, но можно отдельно объявить функцию, а потом реализовать. Пример использования функции:

```
double calc(double x, int n) // функция принимает double и int, возвращает double
{
    double res = 1; // результирующая переменная
    for (int i = 0; i < n; i++) // повторяем n раз
    {
        res *= x; // умножаем res на x
    }
    return res; // возвращаем результат
}
int main()
{
    cout << calc(2.0, 5) << endl; // вызываем функцию
}
```

Посмотреть видео

Параметры и возвращаемое значение

В языке C++ нет процедур как в Pascal, а только функции. Если нет необходимости, чтобы функция возвращала значение, то тип такой функции должен быть void. Соответственно, для завершения выполнения функции необходимо воспользоваться оператором return; без указания возвращаемого значения. Вот пример:

```
void print_star(int n); // только объявляем функцию, но пока не реализуем

int main()
{
    print_star(5); // вызываем функцию
}

void print_star(int n)
{
    int i = 0; // объявляем и инициализируем переменную
    while (true) // «вечный цикл»
    {
        if (i == n)
        {
            return; // завершаем выполнение функции
        }
        cout << '*' << '\t’; // рисуем звездочку и знак табуляции
        i++; // увеличиваем i на 1
    }
}
```

Передача параметров функции

Функция, принимая параметр, может с ним работать как с обычной переменной, но изменение этой переменной не приведут к изменению передаваемого функции параметра.

```
void calc(int n)
{
    n++; // увеличиваем значение на 1
    cout << "Внутри функции: " << n << endl; // будет напечатано 6
}

int main()
{
    setlocale(LC_ALL, "Russian"); // устанавливаем кириллицу
    int n = 5; // объявляем и инициализируем переменную
    calc(n); // вызываем функцию
    cout << "После вызова функции: " << n << endl; // будет напечатано 5
}
```

В дальнейшем мы покажем, как сделать так, чтобы было можно менять значения параметра.
Аргументы функции можно задавать по умолчанию, начиная с конца:

```
int calc(int a, int b = 7) // если не задать b, то его значение будет 7
{
    return a * b; // возвращаем результат
}

int main()
{
    cout << calc(5, 9) << "\t" << calc(5) << endl; // будет напечатано 45    35
}
```

Область видимости переменных

Переменные, которые объявлены внутри функции будут видны только в самой функции и из другой функции они будут не доступны. Более того, локальные переменные (объявленные внутри функции) имеют время жизни только пока выполняется функция. Однако в C++ можно объявлять глобальные переменные, которые будут доступны уже всем функциям и сохранять свои значения. Вот пример:

```
int count = 0; // объявляем и инициализируем глобальную переменную


void print_a()
{
    cout << count << endl; // выводим значение count
    count++; // меняем глобальную переменную - увеличиваем count на 1
}
int main()
{
    print_a(); // печатаем 0
    print_a(); // печатаем 1
    print_a(); // печатаем 2
}
```

Использовать глобальные переменные нужно с особой осторожностью и не злоупотреблять ими, поскольку изменение глобальных переменных нарушает главный принцип работы автономных модулей, который состоит в том, работа модуля не изменяет свое окружение.

Скачать текст программы

Посмотреть видео