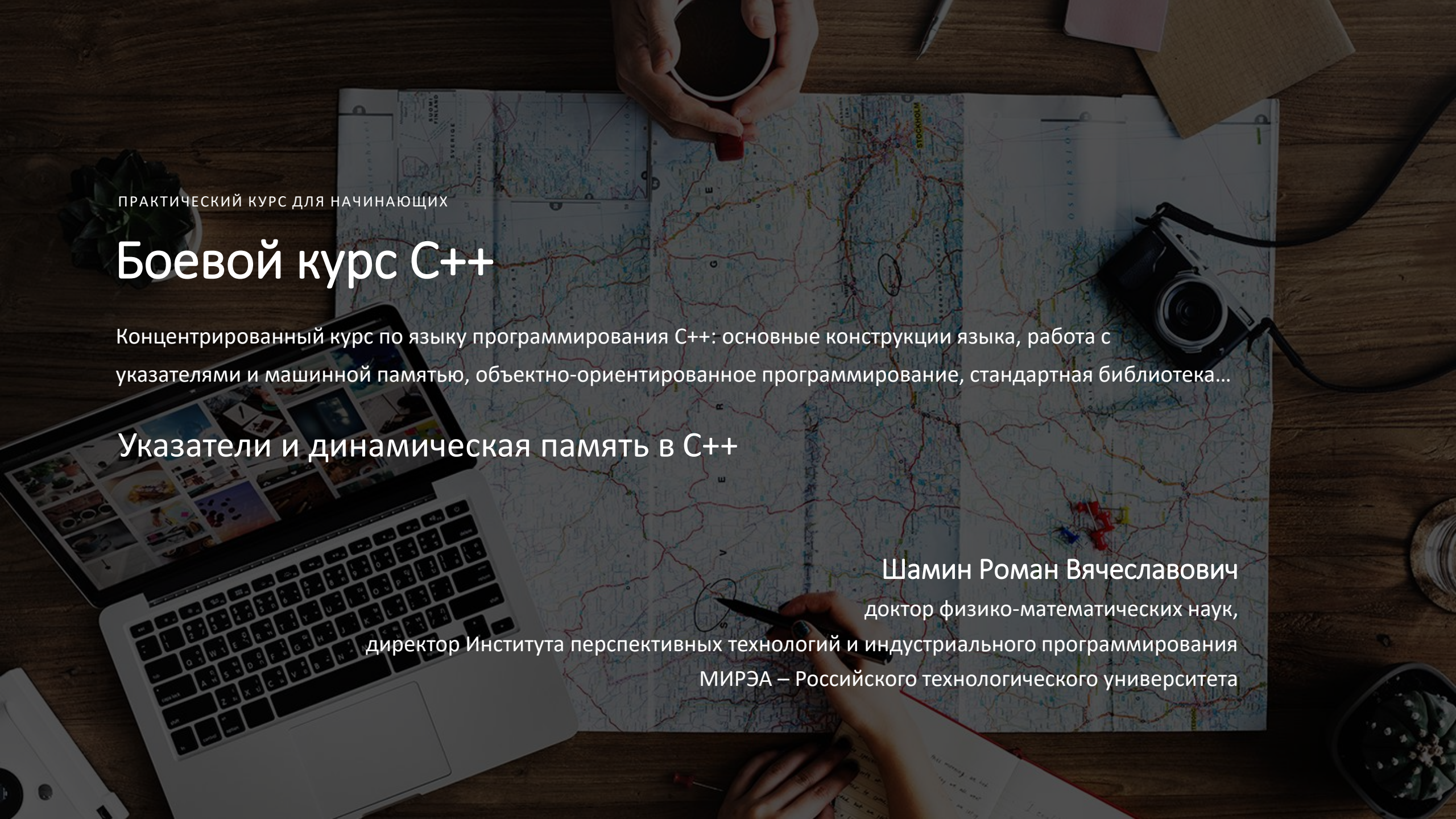


ПРАКТИЧЕСКИЙ КУРС ДЛЯ НАЧИНАЮЩИХ

Боевой курс C++

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

Указатели и динамическая память в C++

Шамин Роман Вячеславович

доктор физико-математических наук,

директор Института перспективных технологий и индустриального программирования

МИРЭА – Российского технологического университета

Указатель – это переменная, которая хранит не данные, а адрес другой переменной.

В языке C++ указатели типизированные. Это означает, что при объявлении указателя мы должны указать на переменную какого типа будет указывать наш указатель.

Объявление указателя выглядит следующим образом:

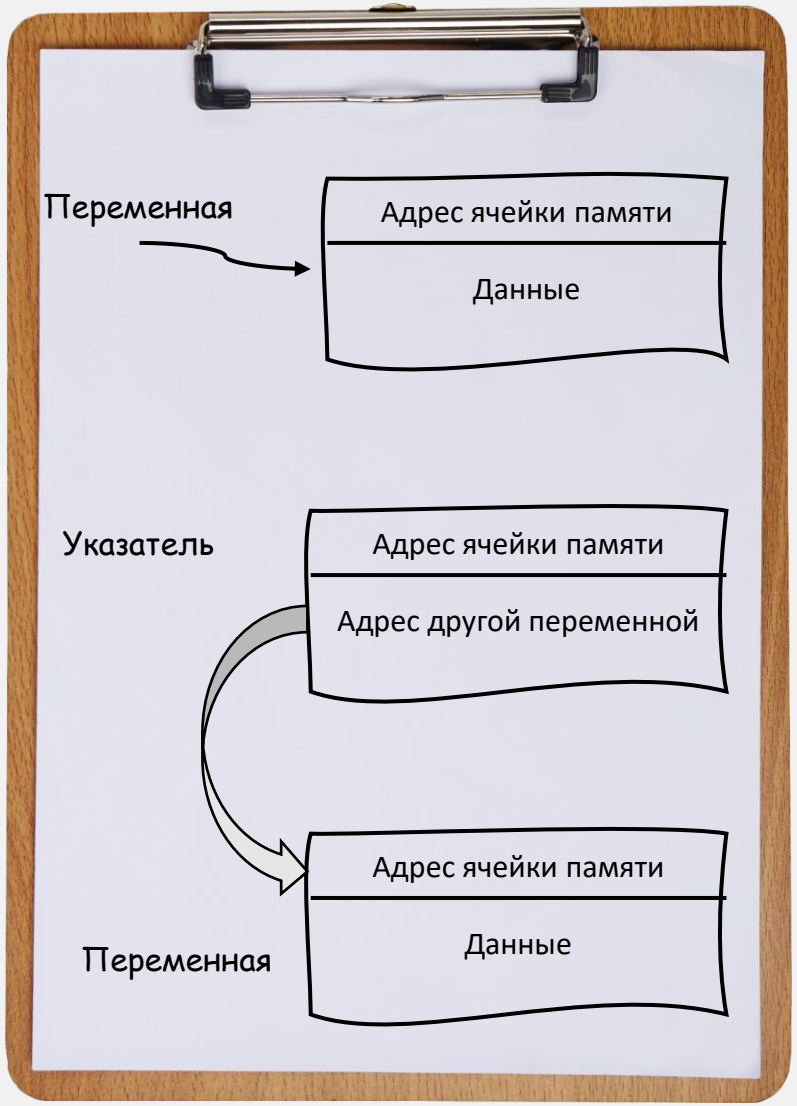
```
int *p; // объявить указатель на int
int a = 12; // создать переменную
p = &a; // присвоить указателю адрес переменной a
*p = 21; // обратиться к адресу на который указатель
cout << "Значение переменной a = " << a << endl;
cout << "Адрес переменной a = " << p << endl;
```

В результате выполнения программы будет выведено примерно следующее:

```
Windows PowerShell
PS C:\cpp> ./p.exe
Значение переменной a = 21
Адрес переменной a = 0x61fe14
```

Конкретное значение адреса переменной у Вас будет другое, этот адрес может меняться при разных запусках программы. Вы не сможете присвоить указателю конкретное значение, кроме как адреса другой переменной или нулевого значения:

```
p = nullptr; // нулевой указатель (NULL, 0)
```



Посмотреть видео

Арифметика указателей и связь с массивами

Указатели имеют тесную связь с массивами. Указателю можно присвоить имя массива (если, конечно, совпадают типы данных) и использовать указатель как массив. Но при этом можно к указателю добавить целое число, и тогда он будет указывать на соответствующий элемент массива. Вот пример:

```
const int N = 10;
int A[N] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; // создаем и инициализируем массив
int *p; // объявляем указатель

p = A; // присваиваем указателю массив, это тоже самое, что p = &A[0];

for (int i = 0; i < N; i++)
{
    cout << p[i] << "\t"; // можно считать, что p – это массив:)
}
cout << endl;

for (int i = 0; i < N; i++)
{
    cout << *p << "\t"; // но можно обращаться к тому элементу, на кого указывает p
    p = p + 1; // сдвигаем указатель на следующую позицию
}
```

То, что к указателю можно добавлять и вычитать целые числа называется арифметикой указателей. Наша программа два раза печатает массив A, обращаясь к нему через указатель разными способами.

Динамическое выделение памяти

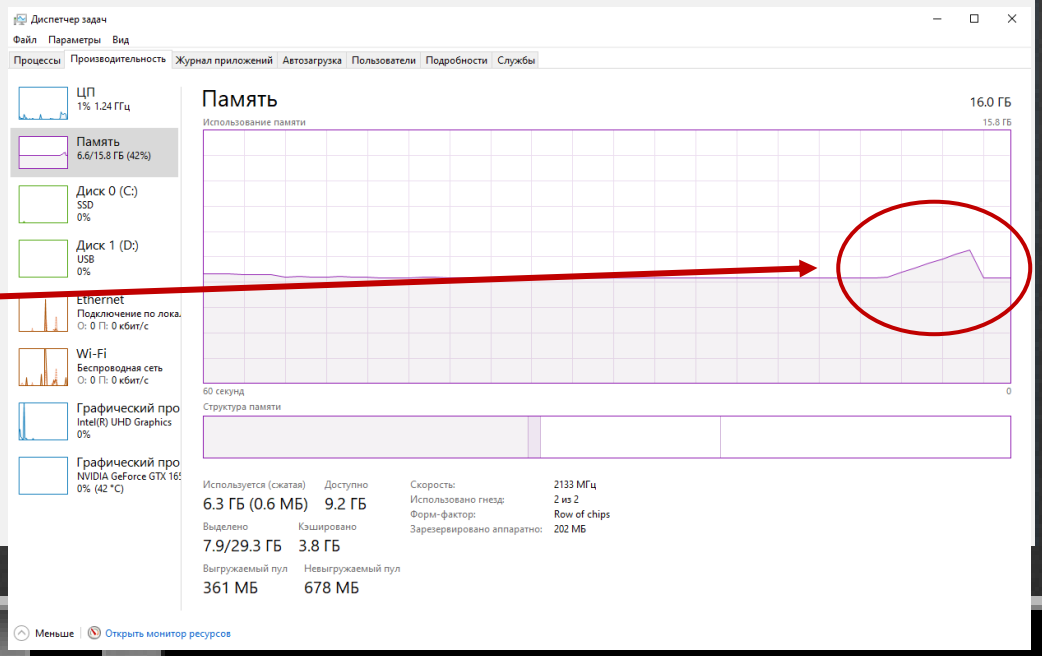
В языке C++ для указателя можно создать новую переменную (без имени) доступ к которой будет только через указатель. При этом нужно соблюдать правило – если Вы создали новую переменную, то потому Вы должны ее удалить из памяти. Вот как это делается:

```
double *p; // объявляем указатель
p = new double(3.14); // выделяем память под double, инициализируем ее и адресуем p
cout << *p; // используем указатель
delete p; // освобождаем память
```

Наиболее важное использование указателей – это создание динамических массивов. Делается это с помощью указателей:

```
int N; // переменная, а не константа
cout << "Введите количество байт > ";
cin >> N; // запрашиваем у пользователя размер массива
char *p = new char[N]; // выделяем память динамически!
for (int i = 0; i < N; i++)
{
    p[i] = 'A'; // заполняем память
}
delete[] p; // освобождаем память
```

Момент выделения и освобождения памяти



С помощью указателей можно изменять аргументы у функции. Для этого нужно передавать указатели на переменные:

```
void swap(int *a, int *b) // аргументы - указатели
{
    int c = *a; // сохраняем значение
    *a = *b; // меняем значения
    *b = c;
}
int main()
{
    int x = 3, y = 5; // создаем две переменные
    swap(&x, &y); // передаем адреса переменных
    cout << "x = " << x << "\ty = " << y;
}
```

Если функция хочет вернуть несколько значений – это можно сделать, вернув массив значений:

```
const int N = 2;

double *two(double x) // возвращаем указатель
{
    double *res = new double[N]; // выделяем память
    res[0] = x * x; // используем как массив
    res[1] = x * x * x;
    return res; // вернуть указатель
}
```

```
int main()
{
    double *A = two(3);
    cout << A[0] << "\t" << A[1];
    delete[] A; // не забываем
                // освободить память
}
```

Скачать текст программы

Посмотреть видео