2 семестр

Объектно-ориентированное программирование

Практические занятия

БИВТ-24-17

Надежда Анисимова

ms teams m2102039@edu.misis.ru

Структура курса

01

Итоговая оценка формируется с учетом всех пунктов справа

- **Домашние задания**

  Ориентировочно 7 лабораторных работ

- **Контрольная работа**

  Одна контрольная в середине семестра

- **Индивидуальный проект на c++**

  Защита проекта с презентацией в конце семестра

- **Экзамен**

  Формат уточняется

Типы данных

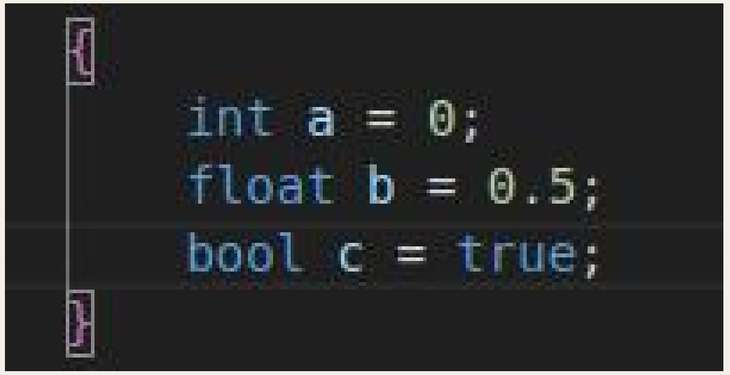
02

C++ — КОМПИЛИРУЕМЫЙ СТАТИЧЕСКИ ТИПИЗИРОВАННЫЙ ЯЗЫК ПРОГРАММИРОВАНИЯ

- **КОМПИЛИРУЕМЫЙ**

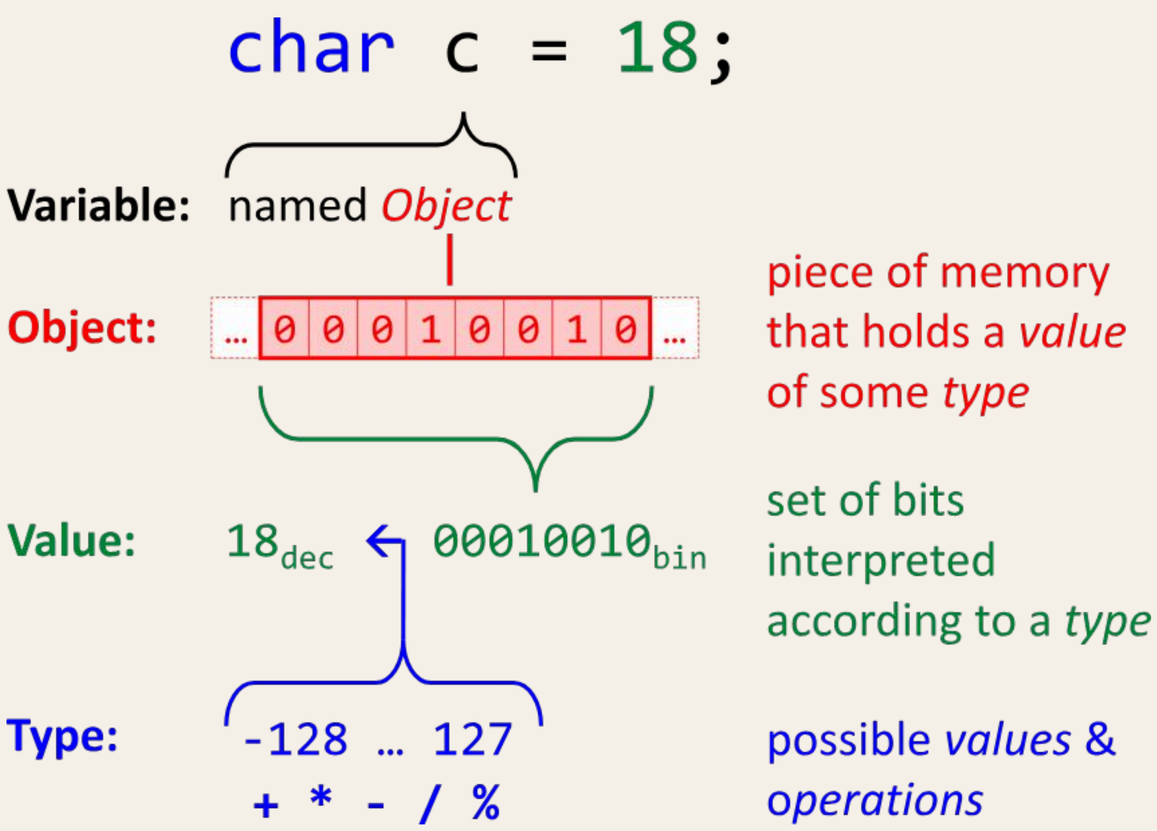
Компилятор – служебная программа, которая преобразовывает код на c++ в машинный исполняемый код

- **СТАТИЧЕСКАЯ ТИПИЗАЦИЯ**



ошибки ловятся на этапе компиляции(compile-time), а не исполнения (run-time)

документация по c++ cppreference.com



- Кусочек памяти
- Интерпретация битов согласно типу
- Допустимые значения

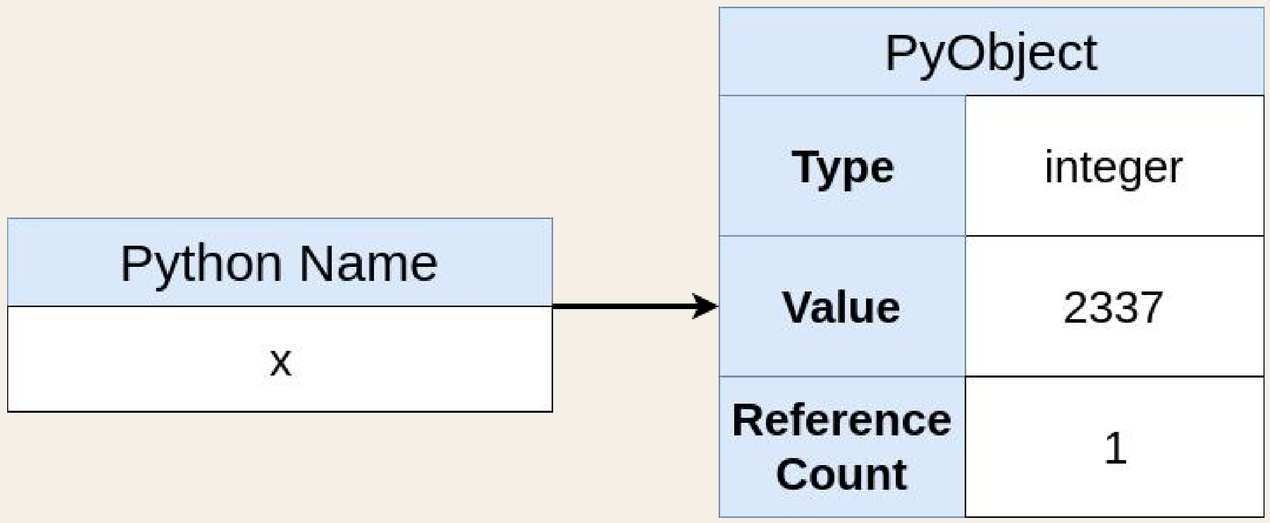
В Python **НЕТ ПЕРЕМЕННЫХ** в стандартном понимании этого термина

В Python переменные – это ссылки на объекты в памяти

c++

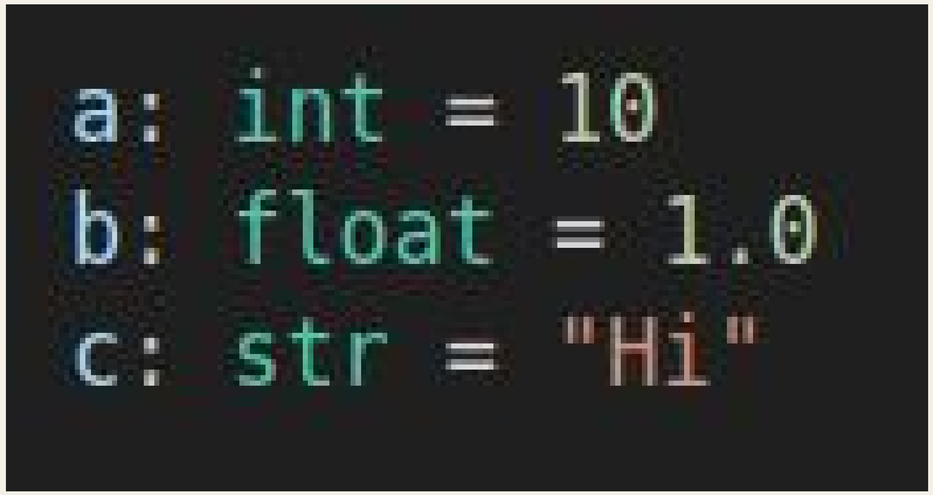
X	
Location	0x7f1
Value	2338

python



Типизация – динамическая

~~Типизация~~
Аннотация типов



Подсказки программисту, которые очень упрощают работу

Напоминалка

- Переменная - именованное хранилище, именованный участок памяти под определенный тип

- Ссылка - альтернативное имя переменной (объекта), псевдоним

Оператор формата **&d**, где d - объявляемое имя

```
int value = 50;
int& refValue = value;
```

Ссылка на константу (reference to const)

Ссылка - альтернативное имя, поэтому сама по себе константной быть не может, но может ссылаться на объект, свойства которого станут константными

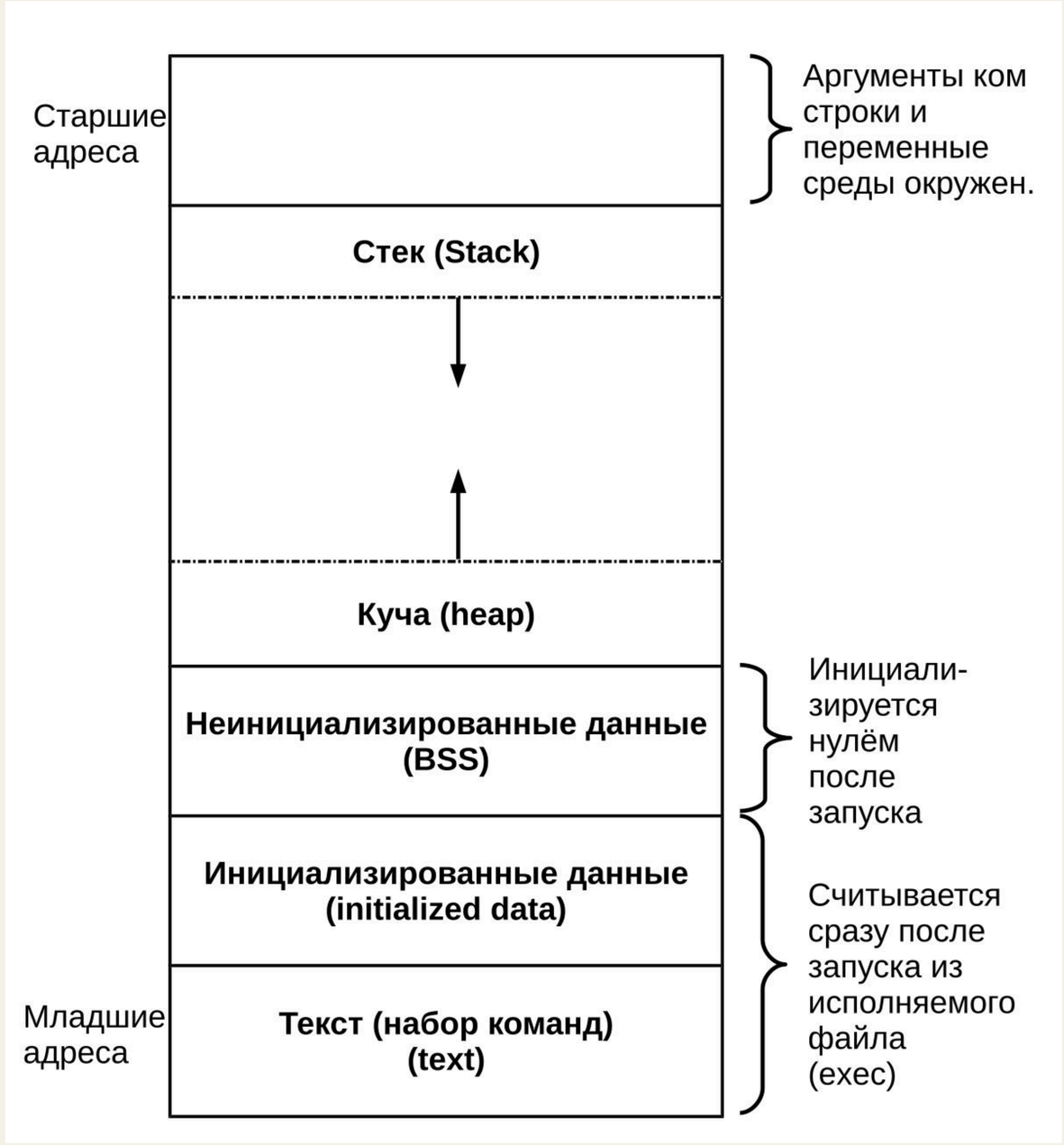
```
// ok
const int val = 50;
const int& ref = val;

int val = 50;
const int& ref = val;
```

```
// ошибка
const int val = 50;
int& ref = val;
```

Память, выделяемая под программу

Memory layout – организации памяти для различных компонентов программы.

Когда мы запускаем нашу программу, операционная система **выделяет блок памяти для нашего процесса**. Этот блок памяти содержит различные сегменты



Массивы стандартной библиотеки

Массив в стиле С

std::array

std::vector

C-style массив

std::array<int, N> a;

std::vector<int> v;

T a[N];

тип

кол-во элементов

тип

кол-во элементов

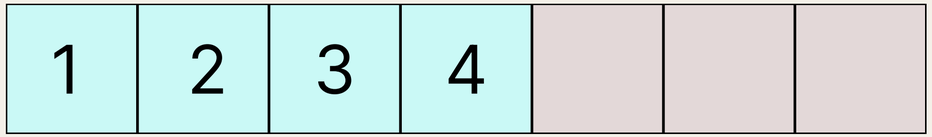
тип

Фиксированный размер

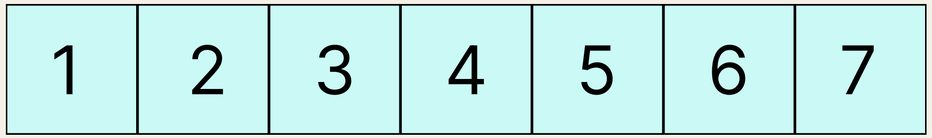
Size vs Capacity

Size - количество элементов в массиве

Capacity - сколько элементов выделено по факту под капотом

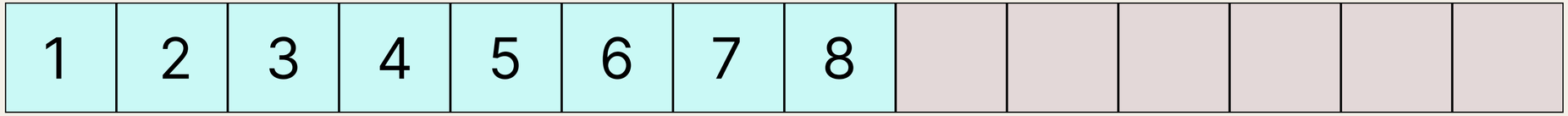


size = 4; capacity = 7;



size = 7; capacity = 7;

Элементы копируются в новую непрерывную область памяти



size = 8; capacity = 14; - увеличилось в 2 раза

Напоминалка

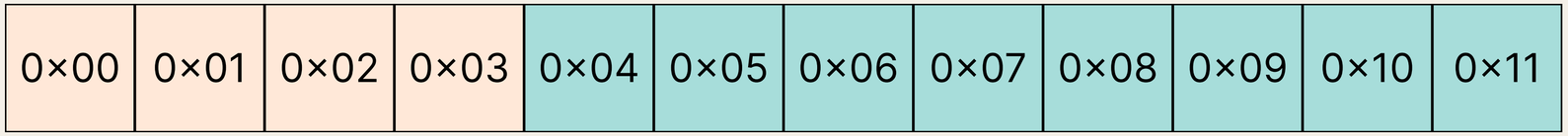
- Переменная - именованное хранилище, именованный участок памяти под определенный тип

Указатель – переменная, значением которой является адрес ячейки памяти

Указатель типа **T*** хранит адрес объекта типа **T**

```
int value;
```

```
int* ptr;
```



Обе переменные лежат в памяти стека

Указатель на константу (pointer to const)

Аналогично ссылке, с помощью указателя на объект можно наложить дополнительное ограничение в виде константности, то есть запретить через указатель менять значение переменной

*ptr = 30; - запретить так менять значение

// ok

```
const int val = 50;
const int* ptr = &val;
```

```
int val = 50;
const int* ptr = &val;
```

// ошибка

```
const int val = 50;
int* ptr = &val;
```

Константный указатель (const pointer)

Но в отличие от ссылки, указатель - объект, поэтому можно так же запретить менять значение самого указателя, то есть запретить присваивать ему новый адрес

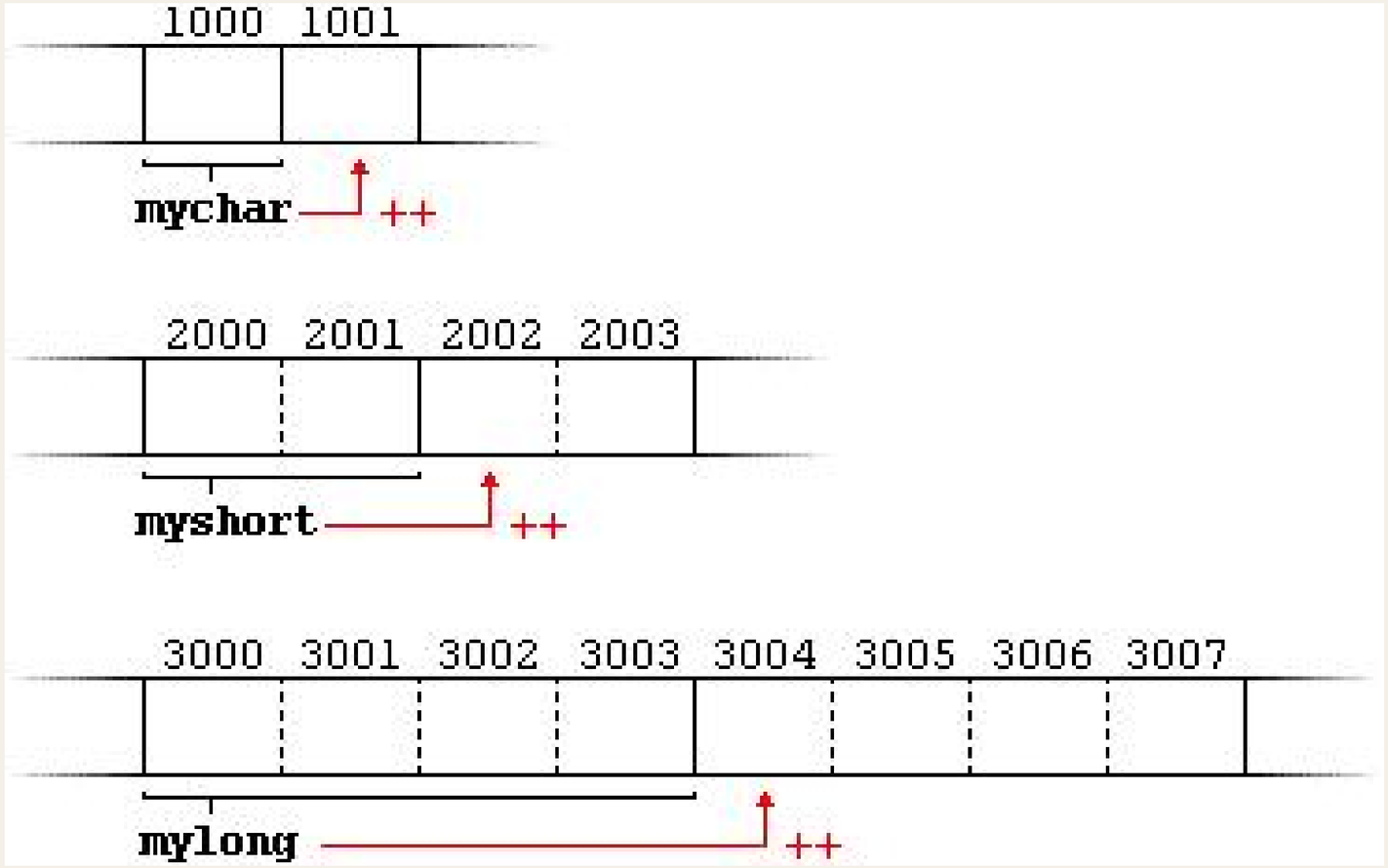
```
cptr = &value; - запретить так менять значение
```

```
int *const cptr = &value;
```

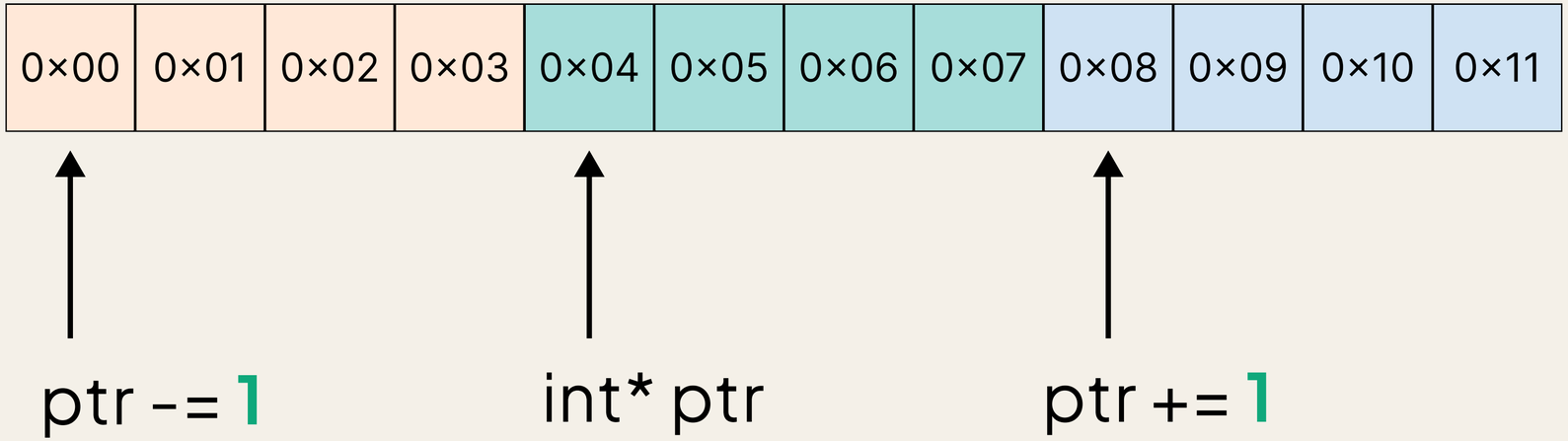
после * добавить const

Для указателей определены только операции **сложения и вычитания**

НО, при увеличении указателя на 1, он будет указывать на следующий элемент того же типа



Указатель + **N** <=> Адрес + **N** * sizeof(тип)



На практике арифметика указателей имеет смысл, если указатель на массив, то есть если вы точно знаете, что хранится в последующем/предыдущем участке памяти

new, delete – операторы для работы с динамической памятью

new – для выделения памяти под указанный тип данных с автоматическим вызовом конструктора;

delete – освобождение памяти с автоматическим вызовом деструктора.

```
int* p = new int; //выделение в куче памяти под int
delete p; //освобождение выделенной памяти
```

new, delete – операторы для работы с динамической памятью

new [] – для выделения памяти под указанное число объектов;

delete [] – для освобождения памяти массива объектов.

```
double* arr = new double[10]; //выделение под 10 double и указатель на первый элемент

delete[] arr; // освобождение памяти для всх 10 элементов
```

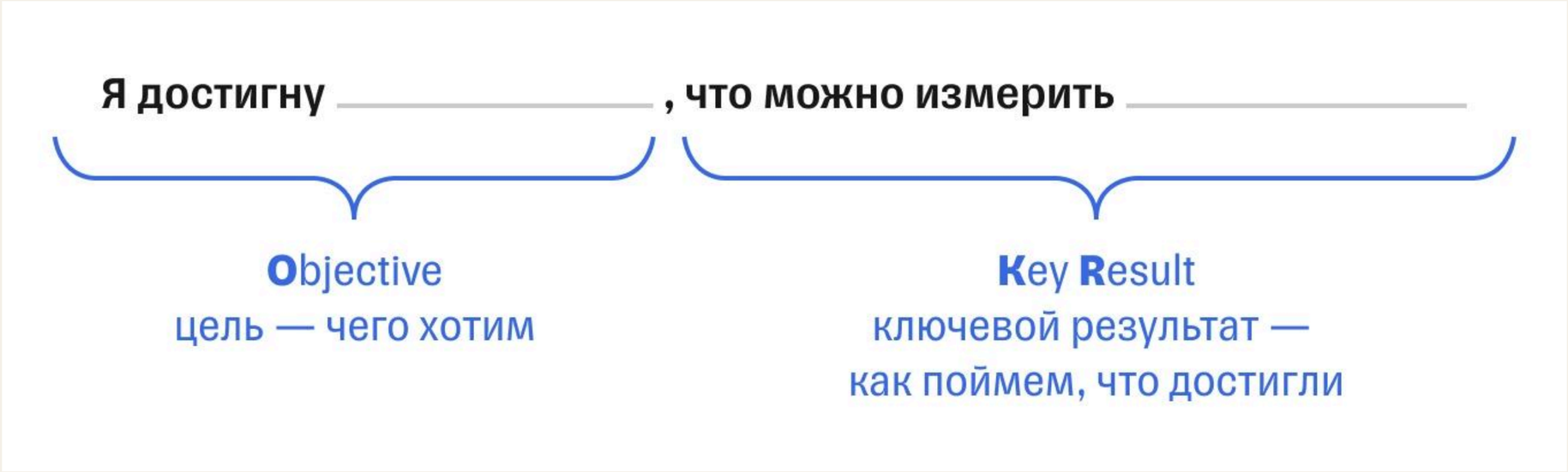
А помните свои цели, которые ставили в начале семестра?


Целеполагание
Мотивация
Рефлексия

Целеполагание через фреймворки

Целеполагание через фреймворки

OKR, или Objectives and Key Results



Законом оптимума мотивации – Закон Йеркса — Додсона

Рефлексия — это осмысление того опыта, который произошёл с человеком

Если совсем глобально мыслить, то рефлексия — это не способ оценивать свои достижения, а способ смотреть на жизнь в целом как на процесс обучения и развития

- What? / Why? / So what? (Что? / Зачем? / И что теперь?)
- Рефлексия по модели Гиббса

1. Описание (что случилось?)
2. Чувства (что вы думали и чувствовали?)
3. Оценка (что было хорошо, а что плохо?)
4. Анализ (какой смысл это несет?)
5. Вывод (что вы бы могли еще сделать?)
6. План действий (если это снова случится, что будете делать?)

Домашнее задание

Класс своего вектора – DynamicArray