

C++

Урок 15

A hand-drawn blue oval frame with a double-line border, centered on the page. The word "Повторение" is written inside this frame in a bold, black, sans-serif font.

Повторение

**Назовите основные парадигмы
ООП**



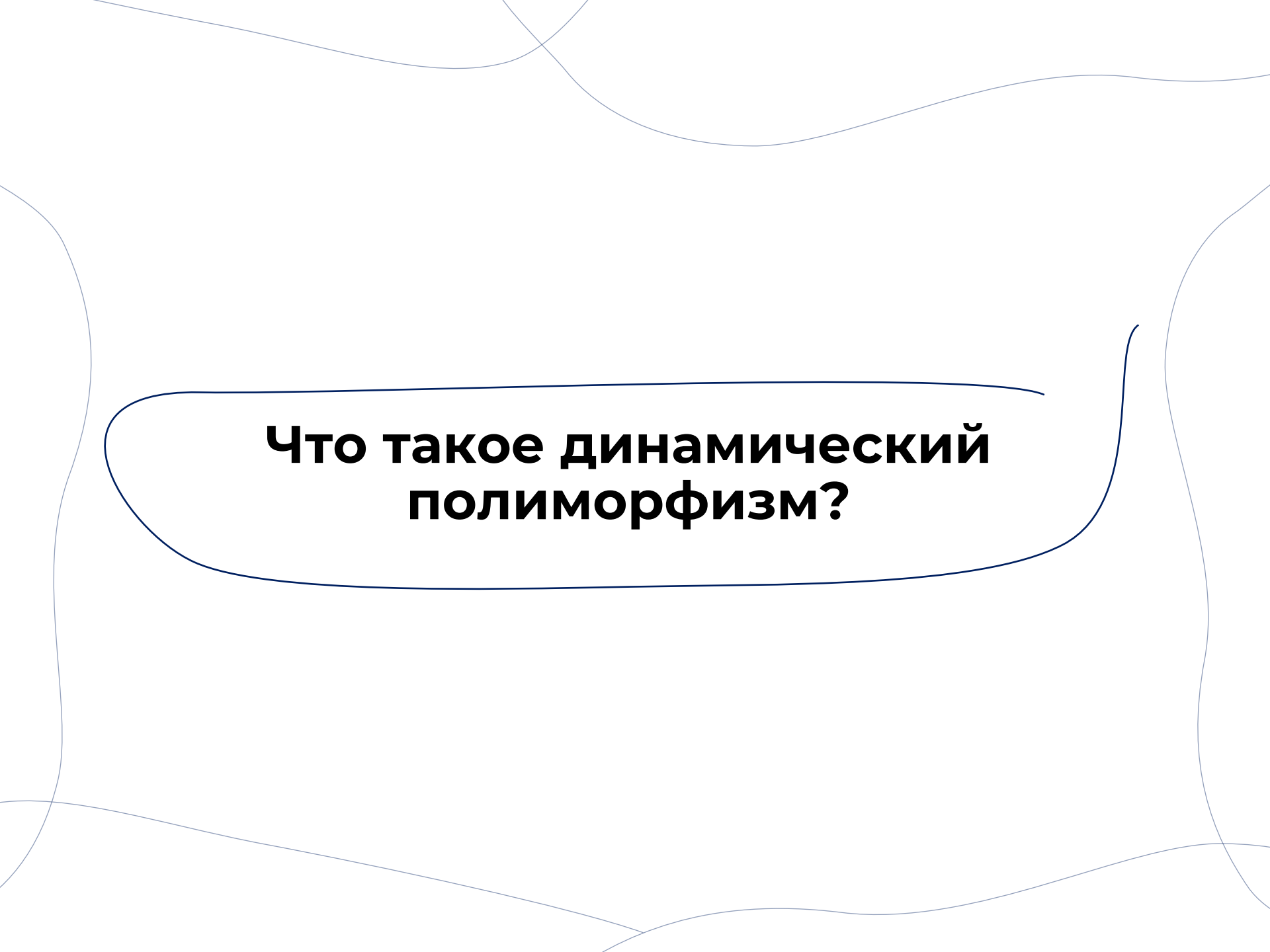
Что такое наследование?



Что такое абстракция?



**С помощью чего реализуется
абстракция?**



**Что такое динамический
полиморфизм?**

**Что делает чисто виртуальная
ф-ия?**

A hand-drawn blue oval frame with a double-line border, centered on the page. The word "Вектор" is written inside this frame.

Вектор

**Что такое динамический
массив?**

Может ли его размер меняться?

Вектор

Вектор — это контейнер, который очень похож на одномерный динамический массив.

Вектор

Вектор — это контейнер, который очень похож на одномерный динамический массив.

Для работы с вектором нужно подключить библиотеку `vector`.

```
#include <vector>
```

Вектор

```
// vector<тип данных> название (размер вектора);  
int len;  
cin >> len;  
vector<int> animals(len);
```

Создание вектора целых чисел размера len

Вектор

```
for (int i = 0; i < animals.size(); i++)
{
    cin >> animals[i];
}
//animals.size() - ф-ия размера контейнера (кол-ва элем)

for (int i = 0; i < animals.size(); i++)
{
    cout << animals[i] <<" ";
}
```

Ввод и вывод элементов контейнера

Вектор

```
int len;  
cin >> len;  
  
vector<int> animals;           //создали пустой вектор  
cout << animals.size() << endl; // 0  
  
// animals.size() => ф-ия подсчет кол-ва элем  
// animals.resize() => ф-ия изменения размера вектора  
  
int temp = 10;  
animals.resize(temp);  
cout << animals.size() << endl; // 10  
  
// push_back(val) => ф-ия вставки в конец эл-та val и размер контейнера +1  
// pop_back() => ф-ия удаления последнего эл-та и размер контейнера -1  
  
animals.pop_back();  
cout << animals.size() << endl; // 9
```

Встроенные ф-ии у вектора

Задача

Задача: напишите программу, которая будет считывать значения в вектор пока пользователь не введет 0.

Вектор

```
vector<int> animals;
int temp;
cin >> temp;
cout << animals.size() << endl;
while (temp)
{
    animals.push_back(temp);
    cin >> temp;
}
cout << animals.size() << endl;
for (int i = 0; i < animals.size(); i++)
{
    cout << animals[i] << " ";
}
```

Задача

Задача: удалите последние два эл-та в векторе.

Итераторы

Итераторы

Для работы с вектором удобнее использовать **итераторы**.

Итератор - объект для перебора элементов контейнера.

Для понимания будем думать, что итератор = указатель на элемент.

Итераторы

Виды итераторов:

`begin()` – указатель на первый эл-т

`end()` – указатель на элемент за последним

Итераторы

Заполним контейнер `vec` (5) след элементами:

2.9	1.3	6.5	0.5	8.6
------------	------------	------------	------------	------------

Итераторы

Заполним контейнер `vec` (5) след элементами:

2.9	1.3	6.5	0.5	8.6
------------	------------	------------	------------	------------

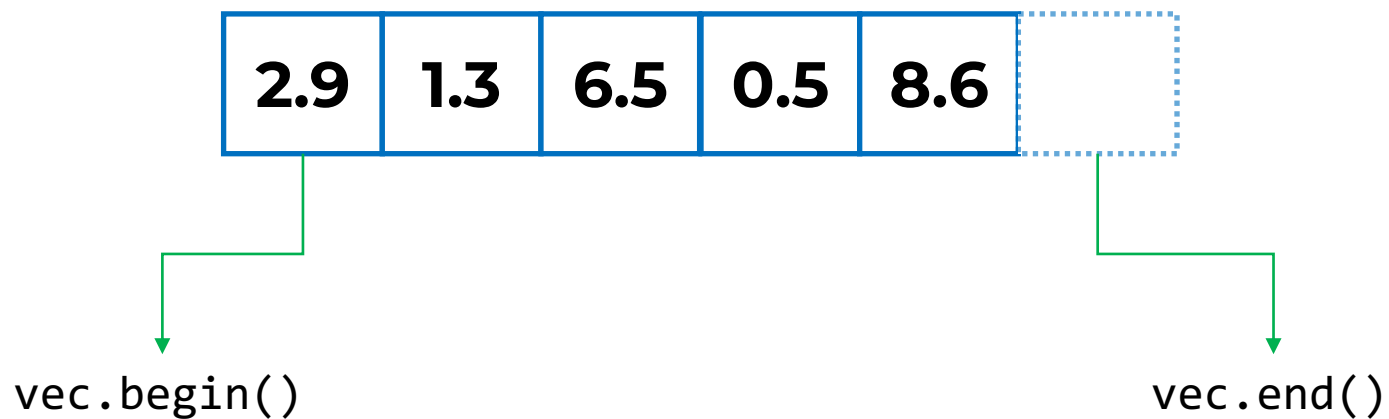
`vec.begin()`

A green line starts from the bottom of the first cell of the vector (2.9), goes down, then left, then down again to point at the text 'vec.begin()'.

<code>vec.begin()</code>

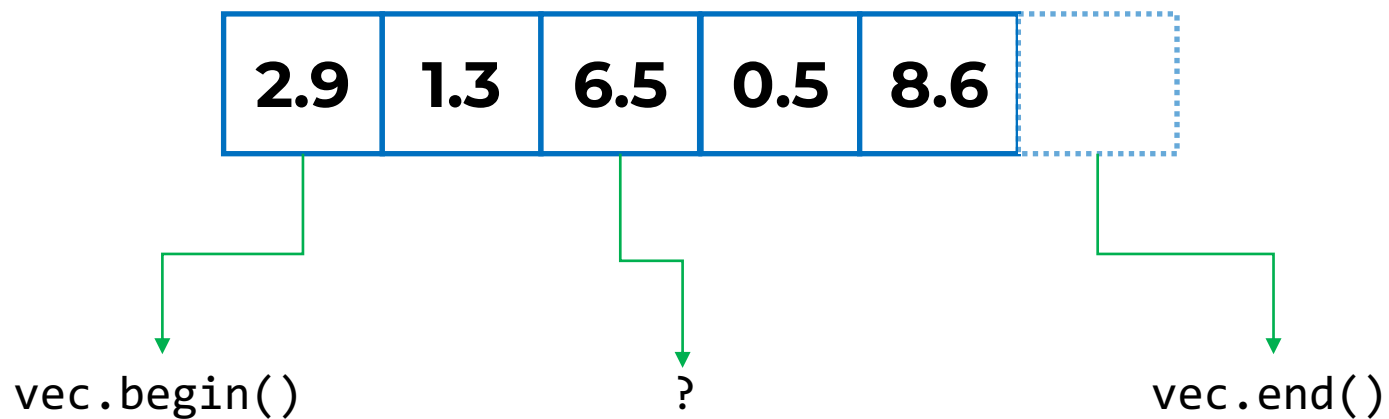
Итераторы

Заполним контейнер `vec` (5) след элементами:



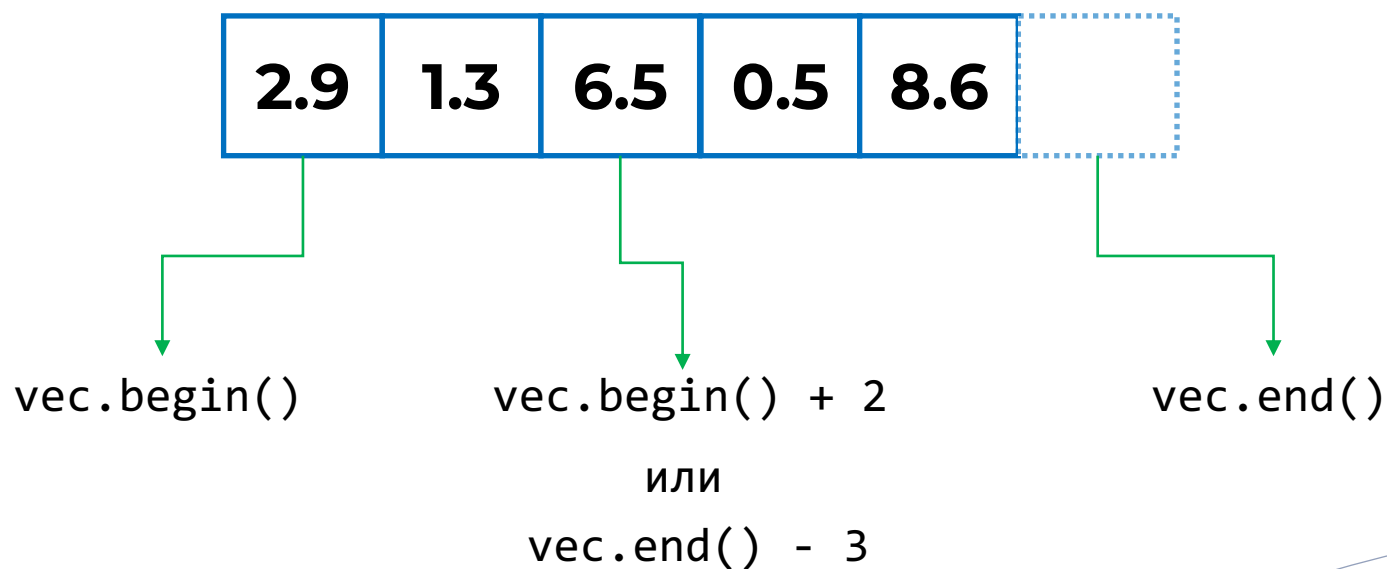
Итераторы

Заполним контейнер `vec` (5) след элементами:



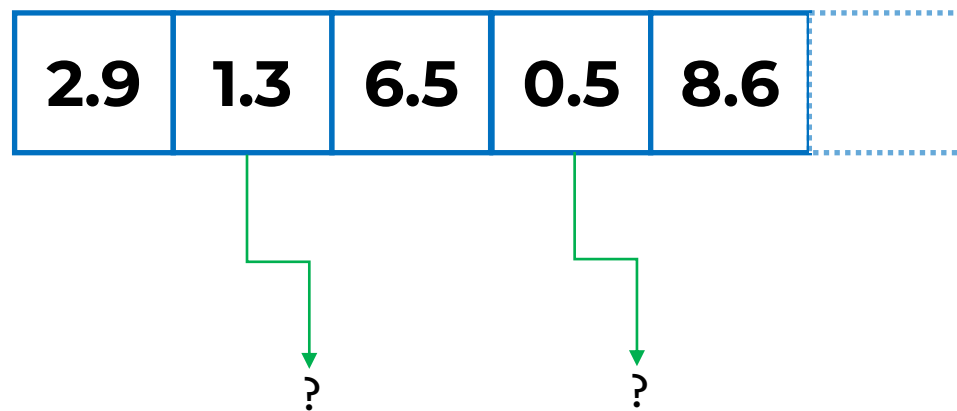
Итераторы

Заполним контейнер `vec` (5) след элементами:



Итераторы

Заполним контейнер `vec` (5) след элементами:



Итераторы

Заполним контейнер `vec` (5) след элементами:



`vec.begin() + 1` `vec.end() - 2`

**Как получить значение по
указателю?**

Итераторы

Заполним контейнер `vec` (5) след элементами:



`*vec.begin() = 2.9`

`*(vec.end() - 2) = 0.5`

Итераторы

Заполним контейнер `vec` (5) след элементами:


2.9	1.3	6.5	0.5	8.6
------------	------------	------------	------------	------------



Итераторы

Заполним контейнер `vec` (5) след элементами:

2.9	1.3	6.5	0.5	8.6
------------	------------	------------	------------	------------



`*(vec.begin() + 2) = 6.5`

Вектор

```
for (auto iter = vec.begin(); iter != vec.end(); iter++)  
{  
    cin >> *iter;  
}  
  
for (auto iter = vec.begin(); iter != vec.end(); iter++)  
{  
    cout << *iter << " ";  
}
```

Ввод и вывод элементов контейнера

Вектор

```
for (auto iter = vec.begin(); iter != vec.end(); iter++)  
{  
    cin >> *iter;  
}  
  
for (auto iter = vec.begin(); iter != vec.end(); iter++)  
{  
    cout << *iter << " ";  
}
```

Что за auto?

Вектор

```
vector<int>::iterator iter = vec.begin(); // чтобы не писать это  
auto iter = vec.begin(); //дадим компилятору понять, что за тип будет у переменной iter
```

Итераторы

Вставка и удаление:

`insert(vec.begin() + 2, 46)` - вставка элемента 46 на позицию с индексом 2. Размер вектора ++.

`vec.erase(vec.begin() + 2)` - удаление эл-та, находящегося на позиции с индексом 2. Размер вектора --.

`vec.erase(vec.begin(), vec.begin() + 2)` - удаление промежутка (первых двух элементов), удаление до второго аргумента не включ.

Задача

На прием к ветеринару приходят сначала собаки, а потом кошки, приходят сделать взвешивание (вещ числа). В результате измерений получается два вектора и это не очень удобно.

Задача: вставьте веса кошек внутрь вектора весов собак на позицию pos.

Задача

На прием к ветеринару приходят животные и измеряют свой вес.

Задача: выведите медианное значение веса. (не путать со средним арифметическим)

Задача

На прием к ветеринару приходят животные и измеряют свой вес. Однако случаются ошибки аппаратуры и записывается отрицательные значения.

Задача: удалите все отрицательные значения.

Вектор

Вектор относится к STL контейнерам.

Методы в vector:

- `find(начало, конец, ключ)` – возвращает итератор на найденный элемент. Если не нашел эл-т, то итератор на конец.
- `count (начало, конец, ключ)` - возвращает кол-во эл-тов по заданному ключу. В `map` вернет либо 0, либо 1.
- `erase (ключ)` – удаляет эл-т по заданному ключу.
- `size()` – кол-во эл-тов в словаре.
- `clear()` – удаление всех эл-тов из словаря.
- `empty()` – определяет пустой ли словарь.
- `swap(контейнер)` – меняет элементы между двумя контейнерами

Вектор

Для комфортной работой с векторами также используют библиотеку `algorithm`.

`algorithm` — заголовочный файл в стандартной библиотеке языка программирования C++, включающий набор функций для выполнения алгоритмических операций над контейнерами STL, каким и является вектор.

[Документация и набор ф-ий тут](#)

Вектор

```
sort(dogs.begin(), dogs.end());    // – сортировка элементов по возрастанию (quick sort)  
reverse(dogs.begin(), dogs.end()); //переворот вектора
```

Сортировка и переворот контейнера

Задача

На прием к ветеринару приходят животные и измеряют свой вес. Однако случаются ошибки аппаратуры и записывается странные значения.

Задача: удалите все значения значения меньше нуля и больше 100.

Подсказка: использовать `remove_if` в связке с `erase`.

Вектор

```
bool rem(int a)
{
    return ((a < 0) || (a > 100));
}

void show(int a)
{
    cout << a << " ";
}

int main()
{
    vector<int> vec(5);
    int i = -3;
    for (auto &it : vec)
    {
        it = i;
        i += 50;
    }

    auto addr = remove_if(vec.begin(), vec.end(), rem);
    // remove_if вернет адрес на эл-т следующий за последним правильным, но не изменит размер контейнера
    vec.erase(addr, vec.end()); //для изменения длины
    for_each(vec.begin(), vec.end(), show);
}
```

Использование ф-ий из algorithm

Задача

На прием к ветеринару приходят животные и измеряют свой вес. Однако случаются ошибки аппаратуры и записывается странные значения.

Задача: посчитайте кол-во положительных значений.

Подсказка: использовать `count_if`.

Вектор

```
bool check(int a)
{
    return (a > 0);
}

void show(int a)
{
    cout << a << " ";
}

int main()
{
    vector<int> vec(5);
    int i = -3;
    for (auto &it : vec)
    {
        it = i;
        i += 50;
    }
    for_each(vec.begin(), vec.end(), show);
    cout << endl;
    int cnt = count_if(vec.begin(), vec.end(), check); // возвращает int, а аргумент – булевая ф-ия
    cout << cnt;
}
```

Использование ф-ий из algorithm