C++

Разбор домашней работы

Практика

15 минут

Теория

15 минут

Разберем задачи

Вектор

```
class MyVector {
    int data[1000], len;
public:
    MyVector(int len = 0, int value = 0);

    void read();
    void print();
};
```

Вектор

```
class MyVector {
    int data[1000], len;
public:
    MyVector(int len = 0, int value = 0) : len(len) {
        for (int i = 0; i < len; i++) data[i] = value;
    }

    void read() {
        for (int i = 0; i < len; i++) cin >> data[i];
    }
    void print() {
        for (int i = 0; i < len; i++) cout << data[i] <<" ";
    }
};
```

Обращение к элементам

```
class MyVector {
    int data[1000], len;
public:
    MyVector(int len = 0, int value = 0);

    void read();
    void print();

    int get(int i);
    void set(int i, int value);
};
```

Обращение к элементам

```
class MyVector {
    int data[1000], len;
public:
    MyVector(int len = 0, int value = 0);

    void read();
    void print();

    int get(int i) { return data[i]; }
    void set(int i, int value) { data[i] = value;  }
};
```

pop_back

```
class MyVector {
    int data[1000], len;
public:
    MyVector(int len = 0, int value = 0);

    void read();
    void print();

    int get(int i);
    void set(int i, int value);

    void pop_back();
};
```

pop_back

```
class MyVector {
    int data[1000], len;
public:
    MyVector(int len = 0, int value = 0);

    void read();
    void print();

    int get(int i);
    void set(int i, int value);

    void pop_back() { len--; }
};
```

push_back

```
class MyVector {
    int data[1000], len;
public:
    MyVector(int len = 0, int value = 0);

    void read();
    void print();

    int get(int i);
    void set(int i, int value);

    void pop_back();
    void push_back(int value);
};
```

push_back

```
class MyVector {
    int data[1000], len;
public:
    MyVector(int len = 0, int value = 0);

    void read();
    void print();

    int get(int i);
    void set(int i, int value);

    void pop_back();
    void push_back(int value) { data[len++] = value; }
};
```

Правильно ли в данном случае делать обычный массив?

Что изменится, если сделать массив динамическим?

MyVector

```
class MyVector {
    int* data, len;
public:
    MyVector(int len = 0, int value = 0) : len(len) {
        data = new int[len];
        for (int i = 0; i < len; i++) data[i] = value;
    }

    void read();
    void print();

    int get(int i);
    void set(int i, int value);

    void pop_back();
    void push_back(int value);
};
```

Как освобождать такую память?

Деструктор

Деструктор - специальный тип метода класса, который выполняется при удалении объекта класса.

Когда объект автоматически выходит из области видимости или же динамический объект удаляется с помощью delete, запускается деструктор.

Как и конструктор, имя деструктора совпадает с именем класса, но в начале добавляется ~:

```
~имя класса() {}
```

MyVector

```
class MyVector {
    int* data, len;
public:
    MyVector(int len = 0, int value = 0) : len(len) {
        data = new int[len];
        for (int i = 0; i < len; i++) data[i] = value;
    }

    ~MyVector() { delete[] data; }

    void read();
    void print();
    int get(int i);
    void set(int i, int value);
    void pop_back();
    void push_back(int value);
};
```

Сколько деструкторов может быть у класса?

Практика

15 минут

Теория

10 минут

Что произойдет, если передать объект нашего класса в функцию?

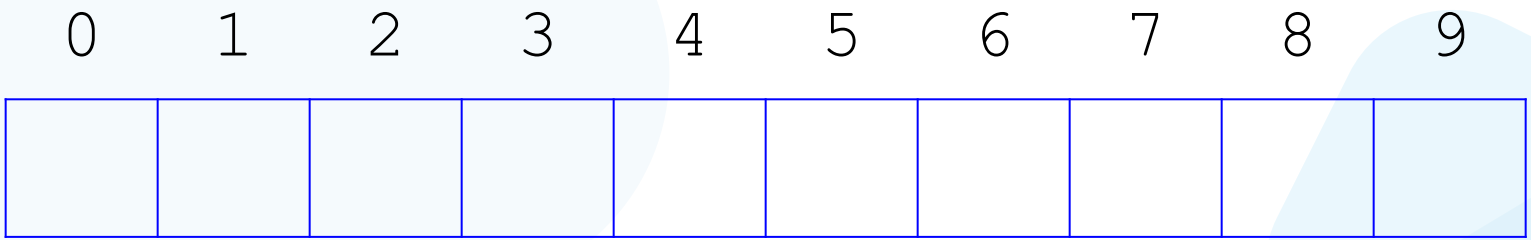
```
int main()
```

```
int count(MyVector a, int value)
```

Все данные объекта копируются, в том числе адрес массива

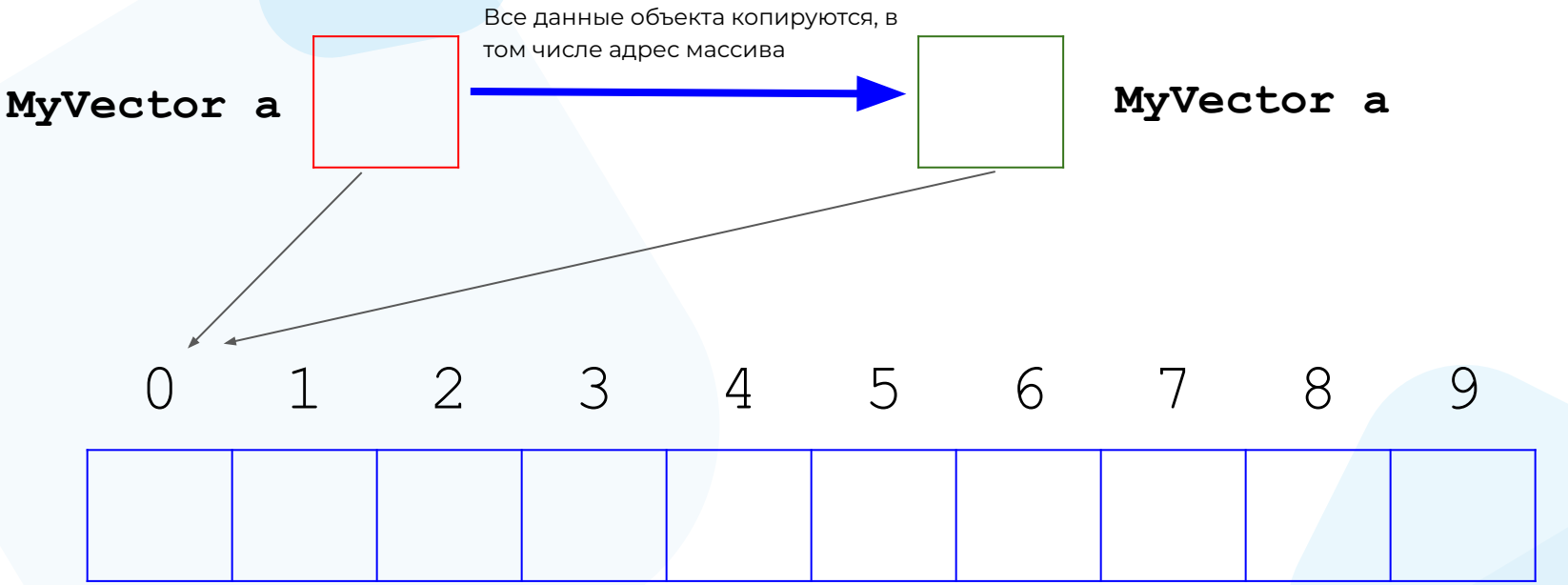
MyVector a

MyVector a



```
int main()
```

```
int count(MyVector a, int value)
```



Один и тот же динамический массив в двух разных объектах

Чем плохо то, что оба объекта работают с одним массивом?

Как исправить проблему?

Переопределим копирование

Конструктор копирования

Конструктор копирования - особый тип конструктора, который используется для создания нового объекта через копирование существующего.

Как и с конструктором по умолчанию, для класса автоматически создается конструктор копирования, который побитово копирует один объект в другой.

Т.е. данные, находящиеся вне объекта (например, динамические данные), скопированы автоматически не будут.

Конструктор копирования

Конструктор копирования как параметр принимает константную ссылку на объект этого же класса. Например:

```
MyClass(const MyClass& a) { … }
```

Почему константа?

Почему ссылка?

Конструктор копирования

Конструктор копирования как параметр принимает константную ссылку на объект этого же класса. Например:

```
MyClass(const MyClass& a) { … }
```

Почему константа? Копируемый объект ни при каких обстоятельствах не должен быть изменен. Иначе это не копирование.
Почему ссылка? Если передавать копию объекта, то для запуска конструктора копирования нужно будет запустить это же копирование.

MyVector

```
class MyVector {
    int* data, len;
public:
                MyVector(int len = 0, int value = 0);
    ~MyVector();
    MyVector(const MyVector& v) {
        len = v.len;
        data = new int[len];
        for (int i = 0; i < len; i++) data[i] = v.data[i];
    }
    void read();
    void print();
    int get(int i);
    void set(int i, int value);
    void pop_back();
    void push_back(int value);
};
```

Практика

20 минут

Итоги урока

1) Повторили работу с динамической памятью
2) Узнали, что такое деструктор и как его создавать
3) Узнали, новый тип конструктора - конструктор копирования
4) Создали свой класс вектора как динамического массива