Программирование на языке C++
Лекция 3

Конструктор копирования и оператор присваивания

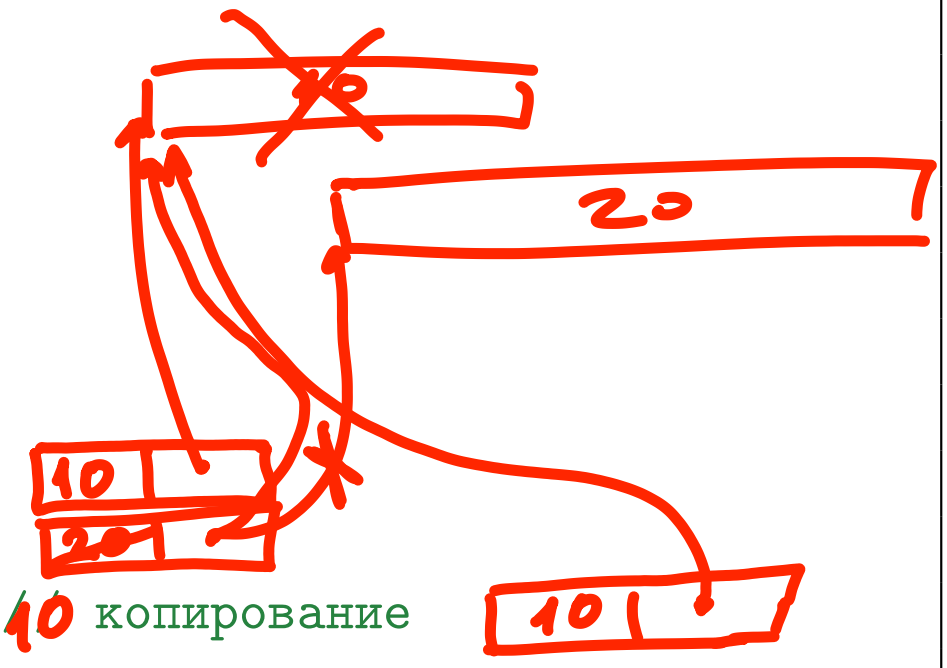
Александр Смаль

Копирование объектов

```
struct IntArray {
    ...
private:
    size_t size_;
    int *  data_;
};

int main() {
  IntArray a1(10);
  IntArray a2(20);
  IntArray a3 = a1; // копирование
  a2 = a1; // присваивание

    return 0;
}
```



Конструктор копирования

- Если не определить конструктор копирования, то он сгенерируется компилятором.

```
struct IntArray {
    IntArray(IntArray const& a)
        : size_(a.size_), data_(new int[size_])
    {
        for (size_t i = 0; i != size_; ++i)
            data_[i] = a.data_[i];
    }
    ...
private:
    size_t size_;
    int *  data_;
};
```

Оператор присваивания

- Если не определить оператор присваивания, то он тоже сгенерируется компилятором.

```
struct IntArray {
    IntArray & operator=(IntArray const& a)
    {
        if (this != &a) {
            delete [] data_;
            size_ = a.size_;
            data_ = new int[size_];
            for (size_t i = 0; i != size_; ++i)
                data_[i] = a.data_[i];
        }
        return *this;
    }
    ...
};
```

Метод swap

```
struct IntArray {
    void swap(IntArray & a) {
        size_t const t1 = size_;
        size_ = a.size_;
        a.size_ = t1;

        int * const t2 = data_;
        data_ = a.data_;
        a.data_ = t2;
    }
    ...
private:
    size_t size_;
    int *  data_;
};
```

Метод swap

Можно использовать функцию std::swap и файла algorithm.

```
#include <algorithm>

struct IntArray {
    void swap(IntArray & a) {
        std::swap(size_, a.size_);
        std::swap(data_, a.data_);
    }
    ...
private:
    size_t size_;
    int *  data_;
};
```

Реализация оператора = при помощи swap

```
struct IntArray {
    IntArray(IntArray const& a)
        : size_(a.size_), data_(new int[size_]) {
        for (size_t i = 0; i != size_; ++i)
            data_[i] = a.data_[i];
    }
    IntArray & operator=(IntArray const& a) {
        if (this != &a)
            IntArray(a).swap(*this);
        return *this;
    }
    ...
private:
    size_t size_;
    int *  data_;
};
```

- IntArray t(a);
- t.swap(*this);

Запрет копирования объектов

Для того, чтобы запретить копирование, нужно объявить конструктор копирования и оператор присваивания как `private` и не определять их.

```
struct IntArray {
    ...
private:
    IntArray(IntArray const& a);
    IntArray & operator=(IntArray const& a);

    size_t size_;
    int *  data_;
};
```

Методы, генерируемые компилятором

Компилятор генерирует четыре метода:

1. конструктор по умолчанию,
2. конструктор копирования,
3. оператор присваивания,
4. деструктор.

Если потребовалось переопределить конструктор копирования, оператор присваивания или деструктор, то нужно переопределить и остальные методы из этого списка.