

Программирование на языке C++

Лекция 3

Модификаторы доступа

Александр Смаль

Модификаторы доступа

Модификаторы доступа позволяют ограничивать доступ к методам и полям класса.

```
struct IntArray {  
    explicit IntArray(size_t size)  
        : size_(size), data_(new int[size])  
    {}  
    ~IntArray() { delete [] data_; }  
  
    int & get(size_t i) { return data_[i]; }  
    size_t size()      { return size_; }  
  
private:  
    size_t size_;  
    int * data_;  
};
```

Ключевое слово class

Ключевое слово `struct` можно заменить на `class`, тогда поля и методы по умолчанию будут `private`.

```
class IntArray {
public:
    explicit IntArray(size_t size)
        : size_(size), data_(new int[size])
    {}
    ~IntArray() { delete [] data_; }

    int & get(size_t i) { return data_[i]; }
    size_t size()      { return size_; }

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

Инварианты класса

- Выделение *публичного интерфейса* позволяет поддерживать *инварианты класса* (сохранять данные объекта в согласованном состоянии).

```
struct IntArray {  
    ...  
    size_t size_  
    int * data_; // массив размера size_  
};
```

- Для сохранения инвариантов класса:
 - все поля должны быть закрытыми,
 - публичные методы должны сохранять инварианты класса.
- Заккрытие полей класса позволяет абстрагироваться от способа хранения данных объекта.

Публичный интерфейс

```
struct IntArray {  
    ...  
    void resize(size_t nsize) {  
        int * ndata = new int[nsize];  
        size_t n = nsize > size_ ? size_ : nsize;  
        for (size_t i = 0; i != n; ++i)  
            ndata[i] = data_[i];  
        delete [] data_;  
        data_ = ndata;  
        size_ = nsize;  
    }  
private:  
    size_t size_;  
    int * data_;  
};
```

Абстракция

```
struct IntArray {  
public:  
    explicit IntArray(size_t size)  
        : size_(size), data_(new int[size])  
    {}  
    ~IntArray() { delete [] data_; }  
  
    int & get(size_t i) { return data_[i]; }  
    size_t size()      { return size_; }  
  
private:  
    size_t size_;  
    int * data_;  
};
```

Абстракция

```
struct IntArray {  
public:  
    explicit IntArray(size_t size)  
        : data_(new int[size + 1])  
    {  
        data_[0] = size;  
    }  
    ~IntArray() { delete [] data_; }  
  
    int & get(size_t i) { return data_[i + 1]; }  
    size_t size()      { return data_[0]; }  
  
private:  
    int * data_;  
};
```