Программирование на языке C++
Лекция 4

Особенности наследования в C++

Александр Смаль

Модификаторы при наследовании

При наследовании можно использовать модификаторы доступа:

```
struct A {};
struct B1 : public A {};
struct B2 : private A {};
struct B3 : protected A {};
```

Для классов, объявленных как `struct`, по-умолчанию используется `public`, для объявленных как `class` — `private`.

- **Важно:** *отношение наследования* (в терминах ООП) задаётся только `public`-наследованием.
- Использование `private`- и `protected`-наследований целесообразно, если необходимо не только агрегировать другой класс, но и переопределить его виртуальные методы.

Переопределение private виртуальных методов

```
struct NetworkDevice {
    void send(void * data, size_t size) {
        log("start sending");
        send_impl(data, size);
        log("stop sending");
    }
    ...
private:
    virtual void send_impl(void * data, size_t size)
    {...}
};

struct Router : NetworkDevice {
private:
    void send_impl(void * data, size_t size) {...}
};
```

Реализация чистых виртуальных методов

Чистые виртуальные методы могут иметь определения:

```
struct NetworkDevice {
  virtual void send(void * data, size_t size) = 0;
    ...
};

void NetworkDevice::send(void * data, size_t size) {
    ...
}

struct Router : NetworkDevice {
private:
  void send(void * data, size_t size) {
        // невиртуальный вызов
      NetworkDevice::send(data, size);
    }
};
```

Интерфейсы

- *Интерфейс* — это абстрактный класс, у которого отсутствуют поля, а все методы являются чистыми виртуальными.

```
struct IConvertibleToString {
    virtual ~IConvertibleToString() {}
    virtual string toString() const = 0;
};
```

```
struct IClonable {
    virtual ~IClonable() {}
    virtual IClonable * clone() const = 0;
};
```

```
struct Person : IClonable {
    Person * clone() {return new Person(*this);}
};
```

Множественное наследование

- В C++ разрешено множественное наследование.

```
struct Person {};
struct Student : Person {};
struct Worker : Person {};
struct WorkingStudent : Student, Worker {};
```

- Стоит избегать *наследования реализаций* более чем от одного класса, вместо этого использовать интерфейсы.

```
struct IWorker {};
struct Worker : Person, IWorker {};
struct Student : Person {};
struct WorkingStudent : Student, IWorker {}
```

- Множественное наследование — это отдельная большая тема.