

# Программирование на языке C++

## Лекция 4

Наследование

Александр Смаль

# Наследование

Наследование — это механизм, позволяющий создавать производные классы, расширяя уже существующие.

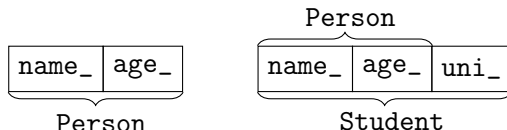
```
struct Person {  
    string name() const { return name_; }  
    int     age()  const { return age_; }  
private:  
    string name_;  
    int age_;  
};  
  
struct Student : Person {  
    string university() const { return uni_; }  
private:  
    string uni_;  
};
```

## Класс-наследник

У объектов класса-наследника можно вызывать публичные методы родительского класса.

```
Student s;  
cout << s.name() << endl  
      << s.age() << endl  
      << s.university() << endl;
```

Внутри объекта класса-наследника хранится экземпляр родительского класса.



## Создание/удаление объекта производного класса

При создании объекта производного класса сначала вызывается конструктор родительского класса.

```
struct Person {  
    Person(string name, int age)  
        : name_(name), age_(age)  
    {}  
    ...  
};  
struct Student : Person {  
    Student(string name, int age, string uni)  
        : Person(name, age), uni_(uni)  
    {}  
    ...  
};
```

После деструктора Student вызывается деструктор Person.

## Приведения

Для производных классов определены следующие приведения:

```
Student s("Alex", 21, "Oxford");  
Person & l = s;    // Student & -> Person &  
Person * r = &s;  // Student * -> Person *
```

Поэтому объекты класса-наследника могут присваиваться объектам родительского класса:

```
Student s("Alex", 21, "Oxford");  
Person p = s; // Person("Alex", 21);
```

При этом копируются только поля класса-родителя (срезка).  
(Т.е. в данном случае вызывается конструктор копирования `Person(Person const& p)`, который не знает про `uni_.`)

## Модификатор доступа protected

- Класс-наследник не имеет доступа к private-членам родительского класса.
- Для определения закрытых членов класса доступных наследникам используется модификатор protected.

```
struct Person {  
    ...  
protected:  
    string name_;  
    int age_;  
};  
  
struct Student : Person {  
    ... // можно менять поля name_ и age_  
};
```