



C++

## Разбор домашней работы



# Наследование

# Теория



15 минут



# Наследование

**Наследование** - возможность описания нового класса на основе уже существующего с заимствованием данных и функционала.

# Наследование

**Базовый класс (предок)** - класс, от которого ведётся наследование.

**Производный класс (наследник)** - класс, наследуемый от базового.

**Синтаксис:**

```
class Имя: public имя_базового_класса {};
```

**Сделаем класс “транспорт”.  
Что может характеризовать  
транспорт?**

**А что транспорт может  
делать?**



**Есть ли какие-то конкретные  
виды транспорта,  
отличающиеся друг от друга?**

# Пример

```
class Point {  
    double x, y;  
public:  
    Point(double _x = 0, double _y = 0) {  
        x = _x; y = _y;  
    }  
};  
class Point3D: public Point {  
    double z; //x и y есть в предке  
public:  
    Point3D(double _x = 0, double _y = 0, double _z = 0) {  
        x = _x; y = _y;  
        z = _z;  
    }  
};
```

# Пример

```
class Point {  
    double x, y;  
public:  
    Point(double _x = 0, double _y = 0) {  
        x = _x; y = _y;  
    }  
};
```

```
class Point3D: public Point {  
    double z; //x и y есть в предке  
public:  
    Point3D(double _x = 0, double _y = 0, double _z = 0) {  
        x = _x; y = _y;  
        z = _z;  
    }  
};
```

Ошибка компиляции



# **Почему возникает ошибка компиляции?**

# protected

При наследовании приватные поля базового класса недоступны в наследнике.

**protected** - модификатор доступа, недоступный вне класса, как и `private`, но при наследовании он переходит как `protected` и доступен в наследнике.

# Пример

```
class Point {  
protected:  
    double x, y;  
public:  
    Point(double _x = 0, double _y = 0) {  
        x = _x; y = _y;  
    }  
};  
  
class Point3D: public Point {  
    double z; //x и y есть в предке  
public:  
    Point3D(double _x = 0, double _y = 0, double _z = 0) {  
        x = _x; y = _y;  
        z = _z;  
    }  
};
```

**Что произойдет, если в наследнике описать метод, который уже есть в предке?**

# Конструктор базового класса

При наследовании конструктор базового класса можно вызвать, используя список инициализации.

Т.е. не нужно в классе-наследнике прописывать инициализацию полей базового класса. Более того, в списке инициализации можно указать **только** поля текущего класса и/или конструктор базового класса.



# Пример

```
class Point {  
protected:  
    int x, y;  
public:  
    Point(int x = 0, int y = 0): x(x), y(y) {}  
};  
class Point3D: public Point {  
    int z;  
public:  
    Point3D(int x = 0, int y = 0, int z = 0): Point(x, y), z(z) {}  
};
```

# Пример

```
class Point {  
protected:  
    int x, y;  
public:  
    Point(int x = 0, int y = 0): x(x), y(y) {}  
};  
class Point3D: public Point {  
    int z;  
public:  
    Point3D(int x = 0, int y = 0, int z = 0): x(x), y(y), z(z) {}  
};
```

# Практика



20 минут



# Разбор



15 минут



**Разберем задачу,  
вызывающую больше всего  
проблем**

# Практика



30 минут



# Итоги урока

- 1) Что такое наследование
- 2) Зачем нужно наследование
- 3) Новый модификатор доступа `protected`
- 4) Конструктор базового класса и список инициализации
- 5) Перегрузка методов