Работа с классами в языке C++

Ограничители доступа:

```
class A {
```

ограничители доступа:

- public: - доступны снаружи класса
- protected: - доступны снаружи класса
- private: - доступны только внутри класса

```
};
```

При наследовании классов передаются только public и protected.

! При создании объекта неявно вызывается конструктор родителя.

Если есть наследование, но создаётся конструктор без аргументов.

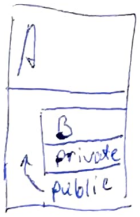
Лишний функционал при наследовании можно удалять, используя конструкцию типа: `method() = delete;`

Также при наследовании могут меняться ограничители доступа:

```
class A: public B {
    public => public
    protected => protected
    private => private
```

```
class A: private B {
    public => private
    protected => private
    private => недоступен
```

```
class A: protected B {
    public => protected
    protected => protected
    private => недоступен
```

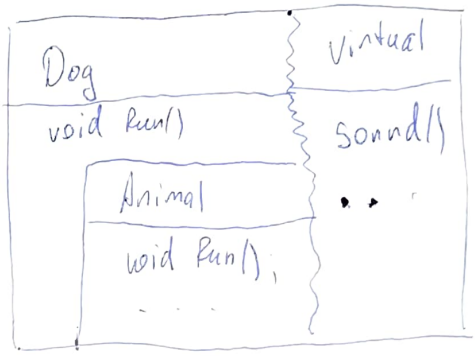


Также в наследнике можно переписать уровень доступа:

```
class Dog: public Animal {
public:
    string type;
protected:
    using Animal::sound;   // ур-ень доступа в Dog к методу sound переписан на protected
};
```

Виртуальная функция - особый тип функций, которая при вызове выполняет "наиболее" дочерний метод, который существует между родительским и дочерними классами. Это свойство ещё известно, как полиморфизм.

```
class Animal {
  public:
    string name;
  public:
    Animal (string new_name = " "): name(new_name){};
    virtual sound();
  } void Run();
```




```
class Dog: public Animal {
  public:
    string type;
  public:
    Dog(string new_name = " ");
    virtual sound() override;
  } void Run();
```

! Если функция определена как виртуальная, то она должна быть виртуальной у всех потомков.

Виртуальные ф-ии - это позднее связывание

! Деструкторы должны быть виртуальными если есть наследование

final - запрещает в дальнейшем наследовать класс и переопределять ф-ию.

```
virtual sound() override final;
```

↑ больше переделывать нельзя

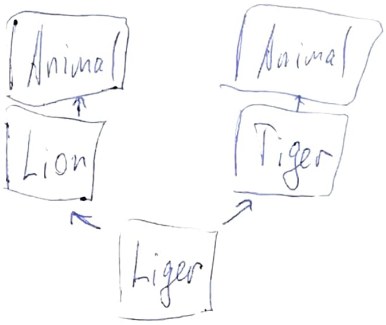
Абстрактные классы – класс, содержащий методы, кот не имеют реализации.

Создаётся с целью создания общего интерфейса между родными реализациями классов, кот. будут производными от абстрактного класса.

У него есть хотя бы одна абстрактная ф-ия. Объекты абстрактного класса не создаются (ошибка компиляции)

```
class IAnimal {
public:
    string get_name (){};
    virtual void sound () = 0; ← абстрактная ф-ия
```

Множественное наследование – позволяет одному дочернему классу иметь несколько родителей.




```
class Animal {
    string name;
};

class Tiger : public Animal {
    int tail_length;
};

class Lion : public Animal {
    int tail_length;
};

class Liger : public Lion, public Tiger {};
```

Но есть проблемы: 1) два экземпляра Animal внутри класса; 2) два одинаковых поля tail_length

Не должно быть одинаковых полей и методов у непосред. родителей, иначе будет неопределенность.

Пользоваться множественным наследованием нужно осторожно!

Дружественные функции и классы — не являются членами класса, однако имеют доступ к его закрытым членам - переменным и ф-циям, кот. имеют спецификатор private.

```
class Graph {
  private:
    Node* root;
  public:
    Node* search(Node *node);
};
class Node {
  private:
    void* data;
    std::list<Node*> neighbors;
    friend class Graph;
};
```

friend - дружба односторонняя.

Node —friend→ Graph

Graph —not friend→ Node

Анонимные объекты - значения без имени.

```
Dog("bobik"); ← анонимный объект
Dog bobik = Dog("bobik");
Dog("bobik").sound();
```