C++

Урок 18

Разбор ДЗ

Неявный указатель this

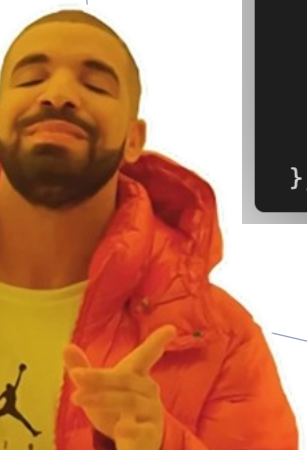
this

```
class Corsina
{
    vector<double> magasine;
    int size;

    void readS();
    void scan();

public:
    Corsina(int size = 0);
    void read(); // прототп метода считывания
    void show();
    void razm();

    void join(const Corsina &test)
    {
        for (auto &iter : test.magasine)
        {
            this->magasine.push_back(iter); //this указывает на тот класс, к которому применяют функцию
        }
    }
};
```



Лучше указать на поле объекта

this

this — это неявный указатель на адрес объекта класса, который является скрытым первым параметром любого метода.

- Тип указателя = имя класса.
- this - зарезервированное словом C++;
- Явно объявить, инициализировать либо изменить указатель this нельзя;
- Обращение к полю/методу объекта происходит через ->

Для чего нужен this?

this

```
class Animal
{
    int age;
    string name;

public:
    Animal(string name = " ", int age = 0)
    {
        this->name = name;
        this->age = age;
    }
};

int main()
{
    Animal Cat("Barsik", 12);
}
```

При написании конструктора с одинаковыми аргументами

this

```
class Animal
{
    int age;
    string name;

public:
    Animal(string name = " ", int age = 0)
    {
        this->name = name;
        this->age = age;
    }
    void sum(const Animal &an)
    {
        this->age += an.age;
    }
    void show()
    {
        cout << name << " " << age;
    }
};

int main()
{
    Animal Cat("Barsik", 12), Lion("Alex", 8);
    Cat.sum(Lion); // this = Cat
    Cat.show();
}
```

Для реализации функции, которая в роли аргумента получает объект класса

this

```
class Animal
{
    int age;
    string name;

public:
    Animal(string name = " ", int age = 0)
    {
        this->name = name;
        this->age = age;
    }
    Animal &sum(const Animal &an){
        this->age += an.age;
        return *this;
    }
    void show()
    {
        cout << name << " " << age;
    }
};

int main()
{
    Animal Cat("Barsik", 12), Lion("Alex", 8);
    Cat.sum(Lion).show(); // this = Cat
}
```

Для реализации цепочек вызова методов

Задача

На прием к ветеринару пришли Барсик, Алекс и Глория.

Задача:. Посчитайте средний вес и запишите его Глории.

```
int main()
{
    Animal Cat("Barsik", 12.4), Lion("Alex", 8.1), Hippo("Глория", 34.45);
    Hippo.middle(Cat, Lion).show();
}
```

Задача

На прием к ветеринару пришли Барсик, Алекс и Глория. Но не все указали свое имя + у системы есть некоторые критерии, которым нужно следовать.

Задача:. Посчитайте средний вес, сделайте проверку, что он не меньше 20 (иначе прибавьте 5) + если не указано имя, то сообщите об этом (переприсвойте поле имени).

```
int main()
{
    Animal Cat("Barsik", 10.2), Lion("Alex", 8), Hippo(" ", 0.45);
    Hippo.middle(Cat, Lion).check().empty().show();
}
```

Наследование

Что такое наследование?

Где мы его уже встречали?

Задача

Есть класс Animal, с полями name и age и метод show(), который выводит информацию полей.

Задача: создайте еще 2 класса: HomeAn и WildAn.

У домашнего животного есь те же поля, что и у животного + высота прыжка.

У дикого животного есь те же поля, что и у животного + булевое поле рева..

Задача

Есть класс Animal, с полями name и age и метод show(), который выводит информацию полей.

main:

```
int main()
{
    Animal Cat("Barsik", 12);
    HomeAn Puff("Leopold", 10, 5.5);
    WildAn Wolf("Alex", 15, 1);
    Cat.show();
    Puff.show();
    Wolf.show();
}
```

Дублировали ли мы код?

Удобно ли это было?

наследование

Наследование — парадигма ООП, при которой дочерний объект получает те же поля и методы, что и в базовом классе.

Наследование позволяет определить базовый класс для определенных функций (доступа к данным или действий), а затем создавать производные классы, которые наследуют или переопределяют функции базового класса.

наследование

Базовый класс – тот, от которого реализуется механизм наследования

Дочерний класс – потомок базового класса.

наследование

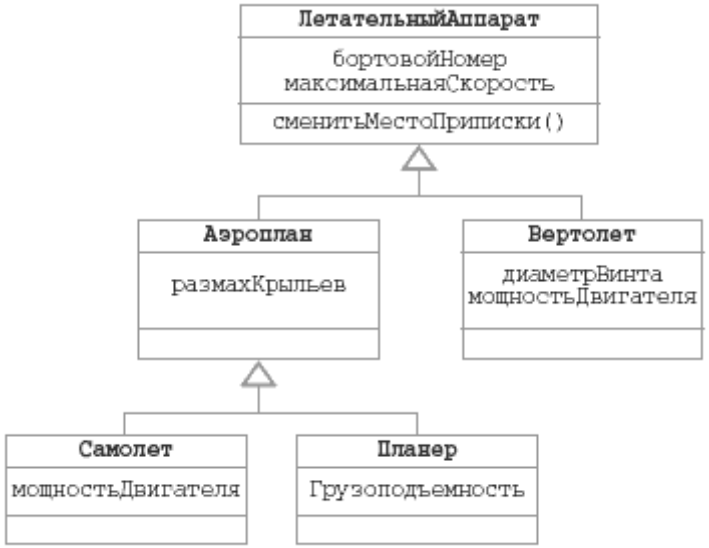
Базовый класс – тот, от которого реализуется механизм наследования

Дочерний класс – потомок базового класса. (получает те же поля и методы, что и в базовом классе)

наследование

Базовый класс – тот, от которого реализуется механизм наследования

Дочерний класс – потомок базового класса. (получает те же поля и методы, что и в базовом классе)



наследование

```
class A{

};

class B: [доступ] A{ //B - дочерний класс от класса A

};
```

Синтаксис наследования

наследование

Модификаторы наследования:

public – публичные члены базового класса доступны. Приватные члены базового класса недоступны. Proteted члены доступны внутри дочернего класса.

private – задается по умолчанию, может отсутствовать. И публичные и приватные члены базового класса недоступны.

protected – в базовом классе элементы, объявленные как protected, снаружи класса трактуются как private. Но в классах-наследниках эти поля доступны.

наследование

Модификатор доступа	Модификатор наследования		
	Private	Protected	Public
Private	Нет доступа	Нет доступа	Нет доступа
Protected	Private	Protected	Protected
Public	Private	Protected	Public

наследование

Protected – модификатор доступа, который позволяет получить доступ к полям и методам базового класса из дочернего.

наследование

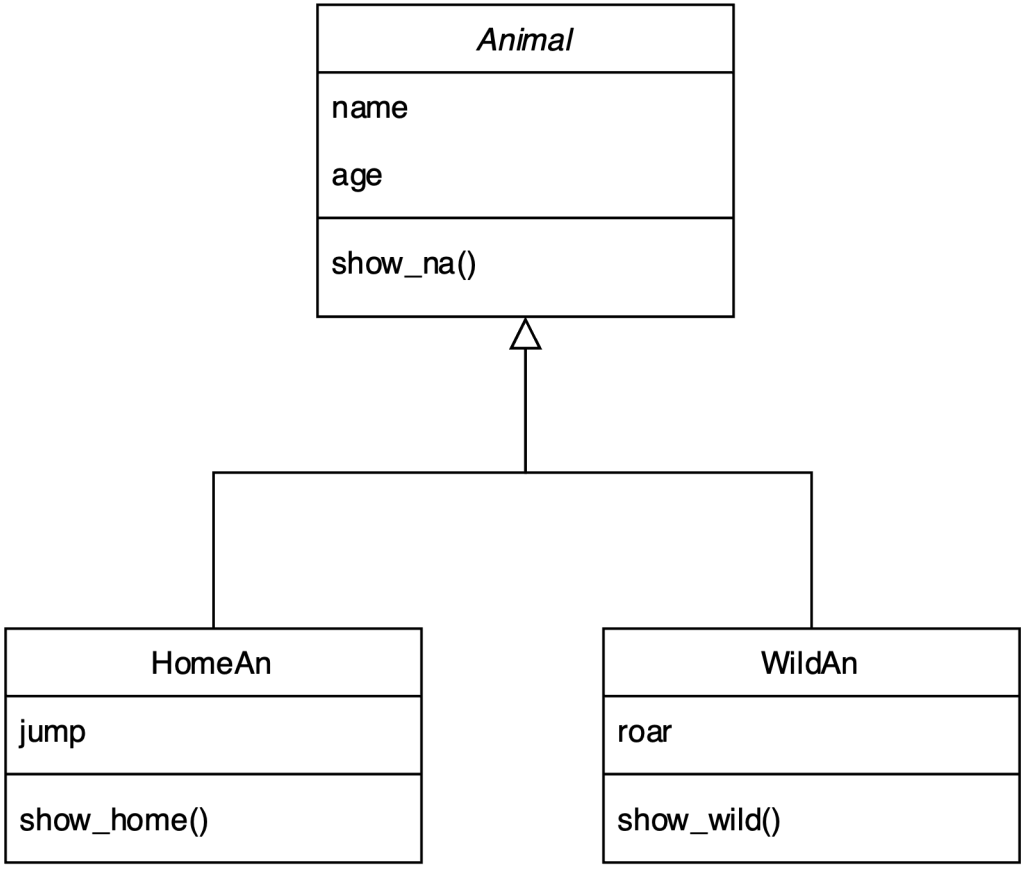


Диаграмма классов

наследование

```
class Animal
{
    int age;
    string name;

public:
    Animal(string name = " ", int age = 0)
    {
        this->name = name;
        this->age = age;
    }

    void show_na()
    {
        cout << name << " " << age << endl;
    }
};
```

Класс Animal

наследование

```
class HomeAn : public Animal
{
    double jump;

public:
    HomeAn(string name = " ", int age = 0, double jump = 0) : name(name), age(age), jump(jump) {}

    void show_home()
    {
        // cout << name << " " << age << " " << jump << endl;
        show_na(); //используем ф-ию базового класса
        cout << " " << jump;
    }
};
```

Класс HomeAn

наследование

```
class HomeAn : public Animal
{
    double jump;

public:
    HomeAn(string name = " ", int age = 0, double jump = 0) : name(name), age(age), jump(jump) {}

    void show_home()
    {
        // cout << name << " " << age << " " << jump << endl;
        show_na(); //используем ф-ию базового класса
        cout << " " << jump;
    }
};
```

```
ge = 0, double jump = 0) : name(name), age(age), jump(jump) {}
```

std::string Animal::name

член "Animal::name" (объявлено в строке 10) недоступно C/C++(265)

Просмотреть проблему Быстрое исправление... (⌘.)

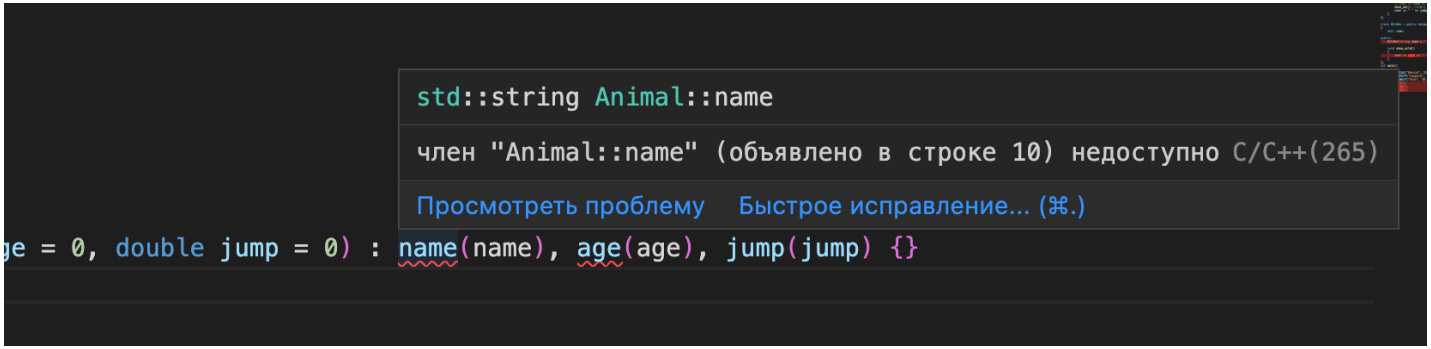
Класс HomeAn

наследование

```
class HomeAn : public Animal
{
    double jump;

public:
    HomeAn(string name = " ", int age = 0, double jump = 0) : name(name), age(age), jump(jump) {}

    void show_home()
    {
        // cout << name << " " << age << " " << jump << endl;
        show_na(); //используем ф-ию базового класса
        cout << " " << jump;
    }
};
```



Как исправить ошибку?

наследование

```
class Animal
{
protected: // модификатор доступа, который дает доступ к полям age, name в дочернем классе
    int age;
    string name;

public:
    Animal(string name = " ", int age = 0)
    {
        this->name = name;
        this->age = age;
    }

    void show_na()
    {
        cout << name << " " << age << endl;
    }
};
```

Класс Animal

наследование

```cpp
class Animal
{
protected: // модификатор доступа, который дает доступ к полям age, name в дочернем классе
    int age;
    string name;

public:
    Animal(string name = " ", int age = 0)
    {
        this->name = name;
        this->age = age;
    }

    void show_na()
    {
        cout << name << " " << age << endl;
    }
};
```

```cpp
class HomeAn : public Animal
{
    double jump;

public:
    HomeAn(string name = " ", int age = 0, double jump = 0) : name(name), age(age), jump(jump) {}

    void show_home()
    {
        show_na(); //используем ф-ию базового класса
        cout << " " << jump;
    }
};
```

std::string Animal::name

"name" не является нестатическим элементом данных или базовым классом для класса "HomeAn" C/C++ (292)

Просмотреть проблему Быстрое исправление... (⇧⌘7)

Класс Animal + HomeAn

наследование

```
class Animal
{
protected: // модификатор доступа, который дает доступ к полям age, name в дочернем классе
    int age;
    string name;

public:
    Animal(string name = " ", int age = 0)
    {
        this->name = name;
        this->age = age;
    }

    void show_na()
    {
        cout << name << " " << age << endl;
    }
};
```

```
class HomeAn : public Animal
{
    double jump;

public:
    HomeAn(string name = " ", int age = 0, double jump = 0) : name(name), age(age), jump(jump) {}

    void show_home()
    {
        show_na(); //используем ф-ию базового класса
        cout << " " << jump;
    }
};
```

```
std::string Animal::name

"name" не является нестатическим элементом данных или базовым классом для класса "HomeAn" C/C++ (292)

Просмотреть проблему  Быстрое исправление... (⇧⌘7)
```

Как решить проблему?

наследование

```
class Animal
{
protected: // модификатор доступа, который дает доступ к полям age, name в дочернем классе
    int age;
    string name;

public:
    Animal(string name = " ", int age = 0)
    {
        this->name = name;
        this->age = age;
    }

    void show_na()
    {
        cout << name << " " << age << endl;
    }
};

class HomeAn : public Animal
{
    double jump;

public:
    HomeAn(string name = " ", int age = 0, double jump = 0) : Animal(name, age), jump(jump) {}

    void show_home()
    {
        show_na(); //используем ф-ию базового класса
        cout << " " << jump;
    }
};
```

Использовать конструктор базового класса

наследование

```
class Animal
{
protected: // модификатор доступа, который дает доступ к полям age, name в дочернем классе
    int age;
    string name;

public:
    Animal(string name = " ", int age = 0)
    {
        this->name = name;
        this->age = age;
    }

    void show_na()
    {
        cout << name << " " << age;
    }
};

class HomeAn : public Animal
{
    double jump;

public:
    HomeAn(string name = " ", int age = 0, double jump = 0) : Animal(name, age), jump(jump) {}

    void show_home()
    {
        show_na(); //используем ф-ию базового класса
        cout << " " << jump << endl;
    }
};

class WildAn : public Animal
{
    bool roar;

public:
    WildAn(string name = " ", int age = 0, bool roar = 0) : Animal(name, age), roar(roar) {}

    void show_wild()
    {
        show_na();
        cout << " " << roar << endl;
    }
};
```

Класс Animal и его дочерние

наследование

```
int main()
{
    Animal Cat("Barsik", 12);
    HomeAn Puff("Leopold", 10, 5.5);
    WildAn Wolf("Alex", 15, 1);
    Cat.show_na();
    cout <<endl;
    Puff.show_home();
    Wolf.show_wild();
}
```

Main()

наследование

```
int main()
{
    Animal Cat("Barsik", 12);
    HomeAn Puff("Leopold", 10, 5.5);
    WildAn Wolf("Alex", 15, 1);
    Cat.show_na();
    cout <<endl;
    Puff.show_home();
    Wolf.show_wild();
}
```

```
Barsik 12
Leopold 10 5.5
Alex 15 1
```

Main()

Задача

Есть класс Animal, с полями name и age и метод show(), который выводит информацию полей.

main:

```
int main()
{
    Animal Cat("Barsik", 12);
    HomeAn Puff("Leopold", 10, 5.5);
    WildAn Wolf("Alex", 15, 1);
    Cat.show();
    Puff.show();
    Wolf.show();
}
```

конструктор

Причина: по умолчанию у каждого объекта есть неявный конструктор (конструктор по умолчанию), когда мы переопределили функцию (написали другой конструктор), то неявный конструктор больше не будет работать.

Решение: переопределить вручную конструктор без аргументов.

```
class Animal
{

    string name;
    int vozr;

    void valid()
    {
        if (vozr < 0)
            vozr = 0;
    }

public:
    Animal(string test, int age) //конструктор класса Animal
    {
        name = test;
        vozr = age;
        valid();
    }
    Animal(){} //конструктор по умолчанию, который ничего не делает
```

Конструктор + конструктор по умолчанию

конструктор

```
int main()
{
    Animal cot("Том", 12); //создали объект и в момент его создания инициализировали его поля
    Animal mouse; //только объявили о наличии объекта mouse без инициализации
}
```

Посмотрим на то, что говорит компилятор

Задача

На осмотр в ветклинику хозяева приводят своих животных. У каждого животного есть кличка, вес, размер и пол.

Задача:. С помощью конструктора задайте значение объектам. Не все так просто: если пользователь не написал кличку, то должно быть написано ”спросить кличку”, вес и размер не могут быть отрицательными. (проверку реализуем в конструкторе)

Задача

```
int main()
{
    Animal cot("Том", 12);
    Animal mouse, dog("Лайка"), zebra(23);

    show(cot);
    show(dog);
    show(zebra);
    show(mouse);
}
```