

C++

Урок 12

A hand-drawn blue oval frame with a slightly irregular, sketchy border, centered on the page. It contains the text 'Разбор ДЗ'.

Разбор ДЗ

Разбор дз

1 задача:

Задача: Создайте класс фигура, у которой есть два поля (две стороны) a и b . Требуется посчитать периметр и площадь.

Разбор дз

1 задача:

```
int main()
{
    figure kv1;
    read(kv1);
    show(per(kv1));
    show(sq(kv1));
}
```

main()

Разбор дз

1 задача:

```
class figure
{
public:
    int a;
    int b;
};
```

Класс figure

Разбор дз

1 задача:

```
void read(figure &test)
{
    cin >> test.a >> test.b;
}

void show(double number)
{
    cout << number << endl;
}

double per(figure &test)
{
    return 2 * (test.a + test.b);
}

double sq(figure &test)
{
    return (test.a * test.b);
}
```

Функции

Разбор дз

2/3 задача:

Задача: Создайте класс фигура, у которой есть два поля (две стороны) a и b . Требуется посчитать периметр и площадь+ валидация значений.

Разбор дз

2/3 задача:

```
int main()
{
    figure kv1;
    read(kv1);
    show(per(kv1));
    show(sq(kv1));
}
```

main()

Разбор дз

2/3 задача:

```
class figure
{
    double a;
    double b;

public:
    void set(double a_, double b_)
    {
        a = a_;
        b = b_;
        if (a < 0)
            a = 0;
        if (b < 0)
            b = 0;
    }

    int get_a()
    {
        return a;
    }
    int get_b()
    {
        return b;
    }
};
```

Класс figure

Разбор дз

2/3 задача:

```
void read(figure &test)
{
    double a, b;
    cin >> a >> b;
    test.set(a, b);
}

void show(double number)
{
    cout << number << endl;
}

double per(figure &test)
{
    return 2 * (test.get_a() + test.get_b());
}

double sq(figure &test)
{
    return (test.get_a() * test.get_b());
}
```

Функции

Разбор дз

4 задача:

Задача: Добавьте функцию сравнения двух фигур. Функция `comparisson` принимает на вход два аргумента (`фигура1`, `фигура2`). Требуется посчитать во сколько одна фигура больше другой в соотношении площадь/периметр.

Разбор дз

4 задача:

```
int main()
{
    figure kv1, kv2;
    read(kv1);
    read(kv2);
    comparisson(kv1, kv2);
}
```

main()

Разбор дз

4 задача:

```
class figure
{
    double a;
    double b;

public:
    void set(double a_, double b_)
    {
        a = a_;
        b = b_;
        if (a < 0)
            a = 0;
        if (b < 0)
            b = 0;
    }

    int get_a()
    {
        return a;
    }
    int get_b()
    {
        return b;
    }
};
```

Класс figure

Разбор дз

4 задача:

```
void read(figure &test)
{
    double a, b;
    cin >> a >> b;
    test.set(a, b);
}

void show(double number)
{
    cout << number << endl;
}

double per(figure &test)
{
    return 2 * (test.get_a() + test.get_b());
}

double sq(figure &test)
{
    return (test.get_a() * test.get_b());
}

void comparisson(figure &a, figure &b)
{
    double compare = 0;
    double val1 = sq(a) / per(a);
    double val2 = sq(b) / per(a);
    cout << "Отношение двух фигур равно: " << val1 / val2 << endl;
}
```

Функции

Разбор дз

5 задача:

Задача: Добавьте функцию `set`, которая заносит в поля объекта `result` значения медианы стороны `a` (объектов `kv1`, `kv2`, `kv3`) и медиану сторон `b` от объектов `kv1`, `kv2`, `kv3`.

Разбор дз

5 задача:

```
int main()
{
    figure kv1, kv2, kv3;
    read(kv1);
    read(kv2);
    read(kv3);
    figure result;
    set(result, med_a(kv1, kv2, kv3), med_b(kv1, kv2, kv3));
    show(result);
}
```

main() функция

Разбор дз

5 задача:

```
class figure
{
    double a;
    double b;

    void check()
    {
        if (a < 0)
            a = 0;
        if (b < 0)
            b = 0;
    }

public:
    void set(double a_, double b_)
    {
        a = a_;
        b = b_;
        check();
    }

    void set(double *a_, double *b_)
    {
        a = *a_;
        b = *b_;
        check();
    }
    int get_a()
    {
        return a;
    }
    int get_b()
    {
        return b;
    }
};
```

класс figure

Разбор дз

5 задача:

```
void read(figure &test)
{
    double a, b;
    cin >> a >> b;
    test.set(a, b);
}

void show(double number)
{
    cout << number << endl;
}

void show(figure &test)
{
    cout << test.get_a() << " " << test.get_b() << endl;
}

double per(figure &test)
{
    return 2 * (test.get_a() + test.get_b());
}

double sq(figure &test)
{
    return (test.get_a() * test.get_b());
}
```

Функции 1 часть

A hand-drawn blue oval frame with a double-line border, centered on the page. The word "Повторение" is written inside this frame in a bold, black, sans-serif font.

Повторение



**Неявный указатель
this**

this

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

- Тип указателя = имя класса.
- `this` - зарезервированное слово 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();
}
```

A hand-drawn blue oval frame with a slightly irregular, sketchy border, centered on the page. It contains the word 'Наследование' in bold black text.

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 есть те же поля, что и у животного + высота прыжка.

У дикого животного есть те же поля, что и у животного + булево 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

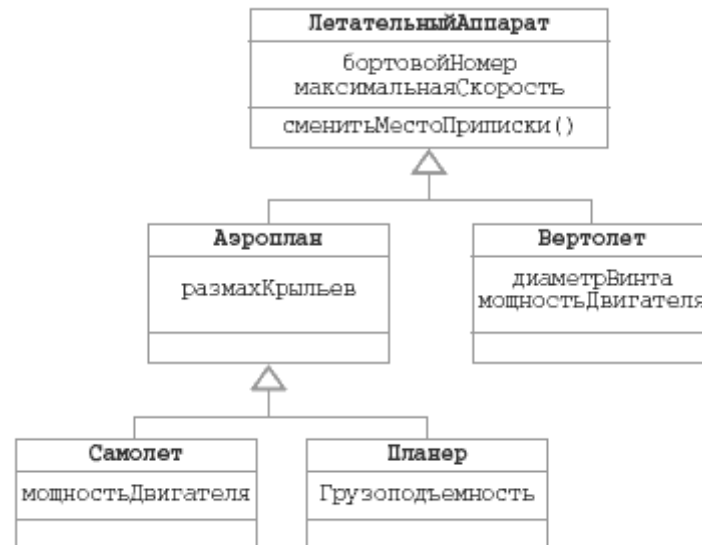
Базовый класс – тот, от которого реализуется механизм наследования

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. Protected 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

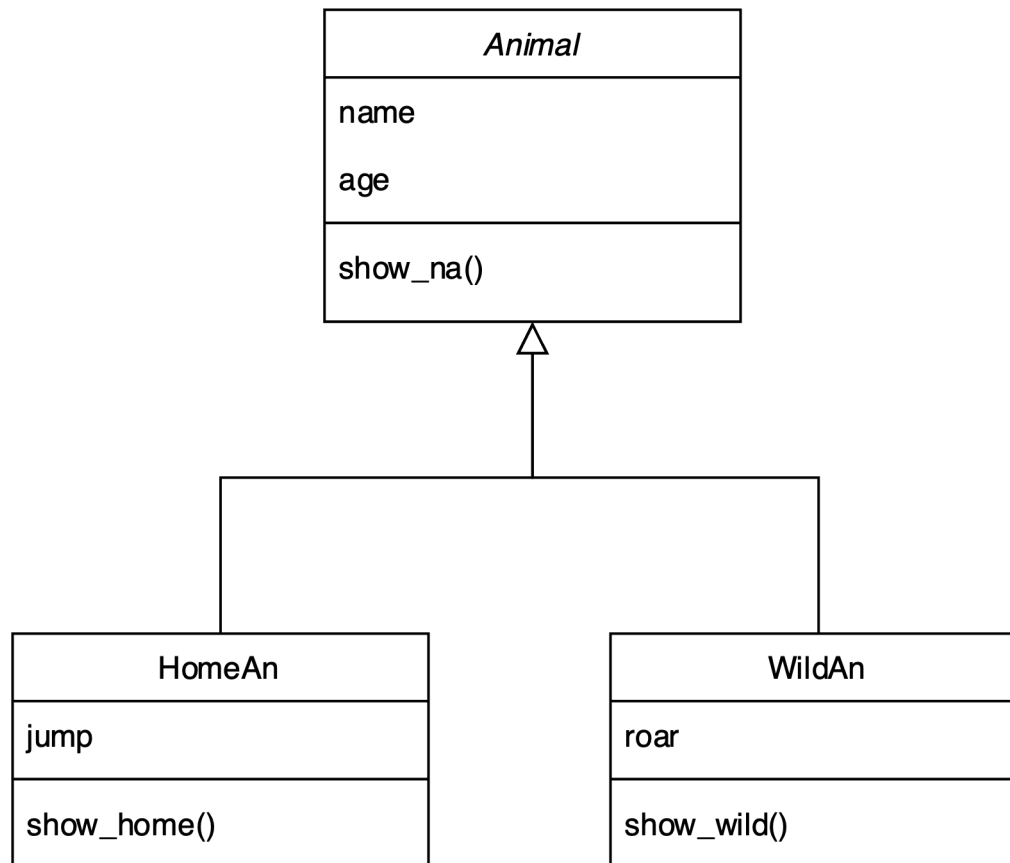


Диаграмма классов

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;
    }
};
```

```
std::string Animal::name
член "Animal::name" (объявлено в строке 10) недоступно C/C++(265)
Просмотреть проблему Быстрое исправление... (⌘.)
age = 0, double jump = 0) : name(name), age(age), jump(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;
    }
};
```

```
std::string Animal::name
член "Animal::name" (объявлено в строке 10) недоступно C/C++(265)
Просмотреть проблему Быстрое исправление... (⌘.)
ge = 0, double jump = 0) : name(name), age(age), jump(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

```
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)

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...](#) (C#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);
}
```