C++

Урок 11

Разбор ДЗ

Разбор дз

4 задача:

```
void proverka(Animal &test)
{
    (test.col > 10) ? test.col = 10 : (test.col < 0) ? test.col = 0
                                                     : test.col;
}
void inc(Animal &test)
{
    cout << "Скорость увеличена " << endl;
    test.skorost++;
    cout << "Его информация:";
    test.show();
}
```

Реализация функций (1 вар)

Разбор дз

4 задача:

```
class Animal
{
public:
    string name;
    int opit;
    double ves;
    double skorost;
    int col;

    void read()
    {
        cout << "Введите имя: " << endl;
        cin >> name;
        cout << "Введите сколько опыта: " << endl;
        cin >> opit;
        cout << "Введите вес инвентаря:" << endl;
        cin >> ves;
        cout << "Введите скорость:" << endl;
        cin >> skorost;
        cout << "Введите кол-во жизни:" << endl;
        cin >> col;
    }
    void show()
    {
        cout << "Персонаж создан!" << endl;
        cout << "Его информация:" << name << ", " << opit << ", "
             << "xp, " << ves << " kg, " << skorost << " sp, " << col << " hp, " << endl;
    }

    void inc(){
        col++; //функция является методом класса
    }
```

Main()

Не много ли места занимает функция read() и show()?

Можем ли мы определить метод класса в виде прототипа?

Разбор дз

Объявим прототипы внутри класса:

```
class Animal
{
public:
    string name;
    int opit;
    double ves;
    double skorost;
    int col;

    void read(); //прототип метода
    void show(); //прототип метода
    void inc();  //прототип метода
};
```

Прототипы методов

Разбор дз

тип данных название класса : : имя функции (аргументы)

```
void Animal::read() //реализация функции read
{
    cout << "Введите имя: " << endl;
    cin >> name;
    cout << "Введите сколько опыта: " << endl;
    cin >> opit;
    cout << "Введите вес инвентаря:" << endl;
    cin >> ves;
    cout << "Введите скорость:" << endl;
    cin >> skorost;
    cout << "Введите кол-во жизни:" << endl;
    cin >> col;
}

void Animal::show() //реализация функции show
{
    cout << "Персонаж создан!" << endl;
    cout << "Его информация:" << name << ", " << opit << ", "
         << "xp, " << ves << " kg, " << skorost << " sp, " << col << " hp, " << endl;
}

void Animal::inc() //реализация функции inc
{
    col++; //функция является методом класса
}
```

Реализация методов

Разбор дз

5 задача:

Задача: Добавьте ф - ию, которая при вызове уменьшает вес инвентаря на 50 кг. Если вес инвентаря будет равен нулю или отрицательному числу, то игра завершаете и выводится сообщение.

Разбор дз

5 задача:

```
int main()
{
    Animal kot, pes, bars;
    kot.read();
    kot.show();
    valid(kot);
    bool flag = 1;
    while (flag)
    {
        flag = play(minInvent(kot));
        kot.show();
    }
}
```

Объявляем прототипы

Разбор дз

5 задача:

```
int minInvent(Animal &test)
{
    cout << "Инвентарь уменьшен " << endl;
    test.ves -= 50;
    if (test.ves < 0)
    {
        test.ves = 0;
    }
    return test.ves;
}

bool play(int num)
{
    if (num == 0)
        return 0;
    else
    {
        return 1;
    }
}
```

Реализация функций

Повтороение

Что такое класс?

Что такое поля класса?

Что такое методы класса?

Расскажите синтаксис написания класса

Можно ли передать класс по ссылке?

В чем разница когда передаем класс по ссылке?

Инкапсуляция

Инкапсуляция

Инкапсуляция - это набор инструментов для управления доступом к данным или методам которые управляют этими данными.

Инкапсуляция

Существует три типа **модификаторов доступа**:

- публичные (**public**) данные—доступны всем (+);
- защищенные (**protected**)—доступны только классу и дочерним классам (#);
- приватные (**private**) —доступны только классу которому они принадлежат (-).

Инкапсуляция

Инкапсуляция реализуется с помощью методов (ф-ии), которые получают доступ к приватным полям объекта:

- Геттеры (возвратные ф-ии для получения значения поля);
- Сеттеры (ф-ии для изменения значения поля).

Инкапсуляция

Этапы:

```
class Animal
{
    // private:
    string name;
    int opit;
    double ves;
    double skorost;
    int col;
```

1. Делаем поля приватными

Инкапсуляция

Этапы:

```
public:
    void setAll(string name, int opit, double ves, double skorost, int col)
    {
        if (skorost < 0)
            skorost = 0;
        if (ves < 0)
            ves = 0;
        name = name;
        opit = opit;
        ves = ves;
        skorost = skorost;
        col = col;
    }

    string getN()
    {
        return name;
    }

    int getXP()
    {
        return opit;
    }

    double GetW()
    {
        return ves;
    }

    double getV()
    {
        return skorost;
    }

    int getcol()
    {
        return col;
    }
};
```

2. Добавляем Геттеры и Сеттеры

Инкапсуляция

Этапы:

```
void read(Animal &test)
{
    string names;
    int op;
    double we;
    double vel;
    int count;
    cin >> names >> op >> we >> vel >> count;
    test.setAll(names, op, we, vel, count);
}

void show(Animal &test)
{
    cout << "Персонаж создан!" << endl;
    cout << "Его информация:" << test.getN() << ", " << test.getXP() << ", "
         << "xp, " << test.GetW() << " kg, " << test.getV() << " sp, " << test.getcol() << " hp ";
}

int main()
{
    Animal kot;
    read(kot);
    show(kot);
}
```

3. Используем, где необходимо

Задача

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

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

Ф-ия main()

Конструкторы

конструктор

Конструктор – функция, которая задает значения объектам (само вызывающийся сеттер).

Конструкторы имеют **имена**, **совпадающие с именами классов**, и **не** имеют возвращаемых значений.

Конструктор можно **перегружать**.

конструктор

```
class Animal
{

    string name;
    int vozr;

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

public:
    void setAll(string test, int age)
    {
        name = test;
        vozr = age;
        valid();
    }

    string getN()
    {
        return name;
    }

    int getA()
    {
        return vozr;
    }
};
```

конструктор

```
class Animal
{

    string name;
    int vozr;

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

public:
    void setAll(string test, int age)
    {
        name = test;
        vozr = age;
        valid();
    }

    string getN()
    {
        return name;
    }

    int getA()
    {
        return vozr;
    }
};
```

Нашли конструктор?

конструктор

```
class Animal
{

    string name;
    int vozr;

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

public:
    void setAll(string test, int age)
    {
        name = test;
        vozr = age;
        valid();
    }

    string getN()
    {
        return name;
    }

    int getA()
    {
        return vozr;
    }
};
```

Нашли конструктор? Его тут **нет**

конструктор

```
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();
    }

    string getN()
    {
        return name;
    }

    int getA()
    {
        return vozr;
    }
};
```

Конструктор класса Animal

конструктор

```
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();
    }

    string getN()
    {
        return name;
    }

    int getA()
    {
        return vozr;
    }
};
```

```
int main()
{
    Animal cot("Том", 12); //создали объект и в момент его создания инициализировали его поля
}
```

Вызов конструктора из main()

конструктор

```
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();
    }

    string getN()
    {
        return name;
    }

    int getA()
    {
        return vozr;
    }
};
```

```
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
}
```

Объявим еще один объект

конструктор

```
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();
    }

    string getN()
    {
        return name;
    }

    int getA()
    {
        return vozr;
    }
};
```

```
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
}
```

Скомпилируется ли программа?

конструктор

```
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();
    }

    string getN()
    {
        return name;
    }

    int getA()
    {
        return vozr;
    }
};
```

```
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
}
```

Скомпилируется ли программа? **НЕТ**

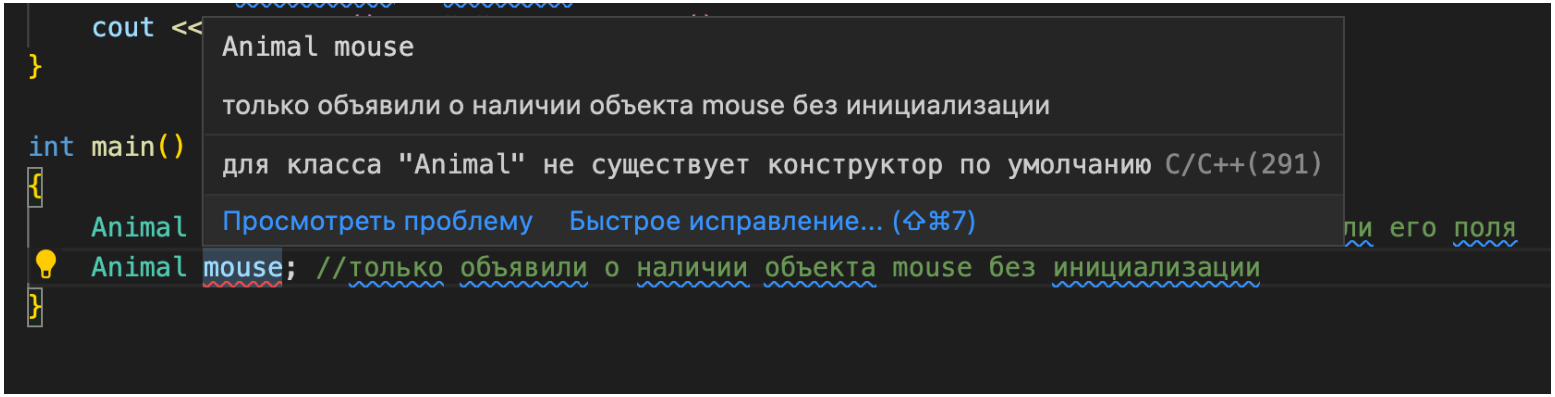
конструктор

```
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();
    }

    string getN()
    {
        return name;
    }

    int getA()
    {
        return vozr;
    }
};
```

```
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
}
```

Скомпилируется ли программа? Почему?

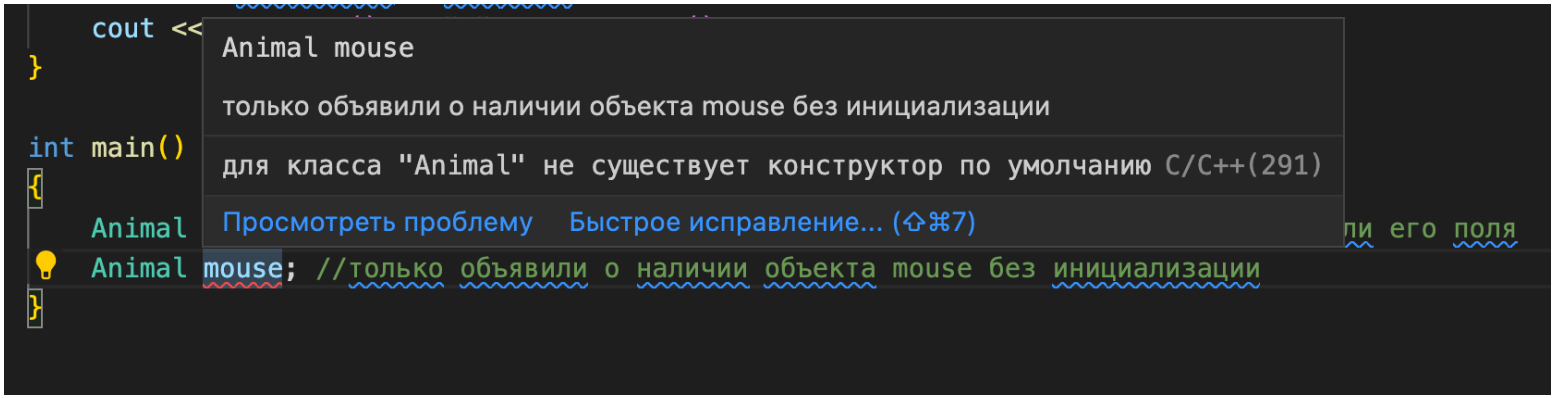
конструктор



Посмотрим на то, что говорит компилятор

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.



Посмотрим на то, что говорит компилятор

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);
}
```

Список инициализации

конструктор

Список инициализации – вариант записи конструктора, у которого значения в поля объекта заносятся напрямую, ускоряя работу.

конструктор

```
Animal(string test, int age) : name(test), vozr(age) //список иниуиализации класса Animal
{
    valid(); //тело конструктора
}
```

Конструктор со списком инициализации

конструктор

```
Animal(string name, int vozr) : name(name), vozr(vozr) //список иниуиализации класса Animal
{
    valid(); //тело конструктора
}
```

Конструктор со списком инициализации с одинаковыми полями

Задача

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

Задача:. С помощью списка инициализации 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)

конструктор

```
class Animal
{

    string name;
    int vozr;

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

public:
    Animal(string name = "спросить кличку", int vozr = 0) : name(name), vozr(vozr)
 //список иниуиализации класса Animal
    {
        valid();
    }

    string getN()
    {
        return name;
    }

    int getA()
    {
        return vozr;
    }
};
```

Конструктор со списком инициализации + конструктор с одинаковыми полями

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 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

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();

    Corsina &join(const Corsina &test)
    {
        for (auto &iter : test.magasine)
        {
            this->magasine.push_back(iter);
        }
        return *this;
    }
};

void show(const double &); //прототип функции вывода значений

int main()
{
    Corsina perv, vtor;
    perv.read();
    vtor.read();
    perv.join(vtor).show();
}
```

Реализация задачи 5

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();
}
```