

C++

# Урок 14

A hand-drawn blue oval frame with a double-line border, centered on the page. The text is positioned in the middle of this frame.

**Разбор ДЗ**

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

**Повторение**



**Что такое конструктор?**

**Можно ли перегружать  
конструктор?**



**Что такое наследование?**

**Какие есть модификаторы  
доступа?**



**Какие используем в  
наследовании?**



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

# **Динамический полиморфизм**

# Динамический полиморфизм

Виды полиморфизма:

- **Статический.** Достигается путём использования перегруженных функций (раннее связывание).
- **Динамический.** Используется наследование в сочетании с виртуальными функциями (позднее связывание).

# Динамический полиморфизм

**Виртуальная функция** – это функция, объявляемая в базовом классе и переопределяемая в производном классе.

Производный класс по своему усмотрению реализует виртуальную функцию. Чтобы объявить виртуальную функцию, используется ключевое слово **virtual**.



**Абстракция**

# Абстракция



**Абстракция** — процесс отвлечения (разделения) от тех или иных характеристик объекта для их избирательного анализа;



# Абстракция

**Абстракция** – это метод программирования (проектирования), основанный на **разделении** интерфейса и реализации.

# Абстракция

Плюсы абстракции:

**Дочерние классы** защищены от непреднамеренных ошибок на уровне пользователя, которые могут повредить состояние объекта.

**Реализация** дочерних может расширяться или дорабатываться.

**Возможность** через абстрактный класс создавать объекты и использовать те методы, которые определены типом дочернего потомка.

# Задача

**Задача:** Напишите реализацию класса Rectangle (прямоугольник) с приватными полями длина и ширина. Также реализуйте методы подсчета периметра (per) и площади (sq).



# Задача

**Задача:** Напишите реализацию класса Rectangle (прямоугольник) с приватными полями длина и ширина. Также реализуйте методы подсчета периметра (per) и площади (sq).

```
int main()
{
    Rect kv(5, 5);
    cout << kv.per() << " " << kv.pl() << endl;
}
```

# Абстракция

```
class Rect
{
private:
    double weight, height;

public:
    Rect(double weight = 0, double height = 0) : weight(weight), height(height) {}
    void read()
    {
        cin >> weight >> height;
    }
    void show()
    {
        cout << weight << " " << height;
    }

    double per()
    {
        return 2 * (weight + height);
    }

    double pl()
    {
        return (weight * height);
    }
};
```

Класс Rect

# Задача

**Задача:** Напишите реализацию класса Circle (круг) с приватными полями радиус. Также реализуйте методы подсчета длины окружности (per) и площади (sq).

# Задача

**Задача:** Напишите реализацию класса Circle (круг) с приватными полями радиус. Также реализуйте методы подсчета длины окружности (per) и площади (sq).

```
int main()
{
    Circle kr(4.12);
    cout << kr.per() << " " << kr.pl() << endl;
}
```

# Абстракция

```
class Circle
{
private:
    double rad;

public:
    Circle(double rad = 0) : rad(rad) {}
    void read()
    {
        cin >> rad;
    }
    void show()
    {
        cout << rad;
    }

    double per()
    {
        return 2 * (PI * rad);
    }

    double pl()
    {
        return (PI * rad * rad);
    }
};
```

Класс Circle



**Что общего между классами?**



**В чем различия?**

**Можно ли обобщить?**





**Можем ли мы реализовать  
механизм наследования?**

**От Квадрата круг?  
От Круга квадрат?**

**А что если создать общий класс  
Фигура?**

# Абстракция

Мы не можем наследовать круг от прямоугольника тк прямоугольник **не частный случай** круга и наоборот.

В данной задаче мы пойдем по пути **абстрагирования** (создание общего класса Figure) от того, кто будет наследником: круг, квадрат, шестиугольник, n – угольник.

В абстрактном классе Figure будут 3 метода: подсчет периметра, площади, деструктор.

**ВАЖНО:** объект абстрактного класса **не может быть создан**, тк он не является завершенным.

**А какая реализация будет у этих методов?**

# Абстракция

При реализации абстрактного класса часть его методов не будут иметь реализацию

# Абстракция

При реализации абстрактного класса часть его методов не будут иметь реализацию

```
class Figure
{
public:
    virtual double pl() = 0;
    virtual double per() = 0;
};
```

Класс Figure с чисто виртуальными функциями

# Абстракция

**Чисто виртуальная функция** (pure virtual function) является функцией, которая объявляется в базовом классе, но не имеет в нем определения.

Поскольку она не имеет определения, то есть тела в этом базовом классе, то всякий производный класс обязан переопределять данную ф-ию.

Для объявления чисто виртуальной функции используется следующая общая форма:

```
virtual тип имя_функции(список параметров) = 0;
```



# Абстракция

```
const double PI = 3.14;

class Figure
{
public:
    virtual double pl() = 0;
    virtual double per() = 0;
};

//Чист вирт ф-ии - ф-ии, кот не имеют тела.

class Rect : public Figure
{
private:
    double weight, height;

public:
    Rect(double weight = 0, double height = 0) : weight(weight), height(height) {}

    double per()
    {
        return 2 * (weight + height);
    }

    double pl()
    {
        return (weight * height);
    }
};

class Circle : public Figure
{
private:
    double rad;

public:
    Circle(double rad = 0) : rad(rad) {}

    double per()
    {
        return 2 * (PI * rad);
    }

    double pl()
    {
        return (PI * rad * rad);
    }
};
```

Реализуем механизм наследования

# Задача

**Задача:** Допишите ф-ии ввода и вывода полей.

# Абстракция

Для того, чтобы увидеть прелесть абстракции, посмотрим на `main()`.

Через указатель базового класса на объект дочернего класса, будем вызывать тот метод, который нужен объекту.

```
int main()
{
    string what;
    Figure *ptr;
    cin >> what;
    if (what == "Rect")
    {
        Rect re_ptr;
        ptr = &re_ptr;
    }
    if (what == "Cirk")
    {
        Circle ce_ptr;
        ptr = &ce_ptr;
    }

    ptr->read();
    ptr->show();
    cout << "Периметр: " << ptr->per() << " Площадь:" << ptr->pl() << endl;
}
```

Реализация `main()`

# Абстракция

В конечном итоге мы работаем с **абстрактным объектом**, который **сам** понимает на что он ссылается.

```
Rect  
2 5  
Квадрат: 2 5. Периметр: 14 Площадь:10
```

```
Cirk  
4  
Круг: 4. Периметр: 25.12 Площадь:50.24
```

# Абстракция

```
int main()
{
    string what;
    Figure *ptr;
    cin >> what;
    if (what == "Rect")
    {
        ptr = new Rect;
    }
    if (what == "Cirk")
    {
        ptr = new Circle;
    }

    ptr->read();
    ptr->show();
    cout << "Периметр: " << ptr->per() << " Площадь:" << ptr->pl() << endl;
}
```

Альтернативная реализация main()

# Задача

**Задача:** Добавьте метод `compare`, который ищет отношение периметра к площади.