

ООП

Кафедра ИВТ и ПМ

2018



План

Прошлые темы

Qt

Outline

Прошлые темы

Qt



- ▶ Что такое стандарт оформления кода?

- ▶ Что такое стандарт оформления кода?
- ▶ Что такое парадигма программирования?



ООП

- ▶ Что такое стандарт оформления кода?
- ▶ Что такое парадигма программирования?
- ▶ Что такое ООП?

ООП

- ▶ Что такое стандарт оформления кода?
- ▶ Что такое парадигма программирования?
- ▶ Что такое ООП?
- ▶ В чём отличие ООП от структурного программирования?



- ▶ Что такое стандарт оформления кода?
- ▶ Что такое парадигма программирования?
- ▶ Что такое ООП?
- ▶ В чём отличие ООП от структурного программирования?
- ▶ В чём отличие ООП от модульного программирования?



- ▶ Что такое класс?

ООП

- ▶ Что такое класс?
- ▶ Что такое объект?



- ▶ Что такое класс?
- ▶ Что такое объект?

```
class MyClass{  
    int x;  
public:  
    void foo();  
};
```

...

```
MyClass c1, *cp;  
// MyClass - класс (тип)  
// c1 - объект (переменная)  
// cp - указатель на объект (переменная)
```



ООП

- ▶ Что такое поле класса?
- ▶ Что такое метод?



ООП

- ▶ Что такое поле класса?
- ▶ Что такое метод?
- ▶ Какое минимальное количество параметров может быть у метода?



- ▶ Что такое поле класса?
- ▶ Что такое метод?
- ▶ Какое минимальное количество параметров может быть у метода?

Можно объявить метод без параметров, однако в метод неявно передаётся указатель на текущий объект - `this`.

```
class MyClass{  
    int x;  
public:  
    void foo(){  
        this->x = 42;  
        // с точки зрения программиста "идентификатор" this  
        // однако this доступен внутри метода потому,  
        // что при описании метода он неявно объявляется  
        // как формальный параметр  
    }  
};
```



ООП. Основные принципы

Основные принципы ООП?



ООП. Основные принципы

Основные принципы ООП?

- ▶ **Абстрагирование** - выделение значимой информации и исключение из рассмотрения не значимой.
- ▶ **Инкапсуляция** - механизм программирования, объединяющий вместе код и данные, которыми он манипулирует, исключая как вмешательство извне, так и неправильное использование данных.
- ▶ **Наследование** - механизм позволяющий строить новые определения классов на основе определений существующих классов
- ▶ **Полиморфизм** - свойство системы, позволяющее использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта



Интерфейс

Что такое интерфейс класса?

```
class MyClass{  
    int x;  
public:  
    void setX(int xx);  
    int x() const;  
  
    void foo();  
    void bar();  
};
```

Интерфейс класса = способы взаимодействия с этим классом
= методы



Интерфейс

Что такое интерфейс (класс-интерфейс)?

```
class Figure{  
public:  
    virtual float area()=0;  
    virtual float perimeter()=0;  
};
```

Абстрактный класс без полей, с абстрактными (без реализации) методами.



ООП. Основные принципы

Инкапсуляция. Какой из примеров реализует инкапсуляцию?

1. `class Seconds{
 public: float s;};`
2. `class Seconds{
 public:
 float s;
 void set_secs(float s) {...}
 float secs() {...} const;};`
3. `class Seconds{
 float s;
 public:
 void set_secs(float s) {...}
 float secs() {...} const;};`



ООП. Основные принципы

Инкапсуляция. Какой из примеров реализует инкапсуляцию?

1. `class Seconds{
 public: float s;};`
2. `class Seconds{
 public:
 float s;
 void set_secs(float s) {...}
 float secs() {...} const;};`
3. `class Seconds{
 float s;
 public:
 void set_secs(float s) {...}
 float secs() {...} const;};`

Пример 3 и 2 (без сокрытия данных).



ООП. Основные принципы

```
class Fighter{  
    float mass;  
    float max_speed;  
    Armament arm;  
public:  
    ...};  
  
class Airliner{ // passenger aircraft  
    float mass;  
    float max_speed;  
    unsigned capacity;  
public:  
    ...};
```

Проблема?



ООП. Основные принципы

```
class Fighter{
    float mass;
    float max_speed;
    Armament arm;
public:
    ...};

class Airliner{ // passenger aircraft
    float mass;
    float max_speed;
    unsigned capacity;
public:
    ...};
```

Проблема?

У классов одинаковые поля и соответственно методы доступа к полям также должны быть реализованы дважды.



ООП. Наследование

```
class Aircraft{
    float mass;
    float max_speed;
public:
    ...};

class Fighter: public Aircraft{
    Armament arm;
public:
    ...};

class Airliner: public Aircraft{ // passenger aircraft
    unsigned capacity;
public:
    ...
};
```



ООП. Наследование

```
class Aircraft{
    float mass;
    float max_speed;
public:
    ...};

class Fighter: public Aircraft{
    Armament arm;
public:
    ...};

class Airliner: public Aircraft{ // passenger aircraft
    unsigned capacity;
public:
    ...
};
```



ООП. Наследование

Наследование в фреймворках для создание приложений с GUI?

При создании окон:

```
class MainWindow : public QMainWindow{
Q_OBJECT
// макрос для создание метаобъекта
public:
explicit MainWindow(QWidget *parent = 0);
~MainWindow();
private:
// Класс Ui::MainWindow генерируется автоматически из файла интер
// в нём описаны все элементы интерфейса, их расположение и свой
пользователя mainwindow.ui
Ui::MainWindow *ui;
// другие методы и поля ...
}
```



ООП. Прошлые темы

```
class Circle{
    float r;
public:
    float area() const {return M_PI*r*r;}};
class Square{
    float a;
public:
    float area() const {return a*a;}};
...
// Найти общую площадь всех фигур
list<Circle> circles;
list<Square> squares;
...
float S=0;
for (Circle f: circles) S+=f.area();
for (Square f: squares) S+=f.area();
```

Проблема?



ООП. Прошлые темы

```
class Circle{
    float r;
public:
    float area() const {return M_PI*r*r;}};
class Square{
    float a;
public:
    float area() const {return a*a;}};
...
// Найдти общую площадь всех фигур
list<Circle> circles;
list<Square> squares;
...
float S=0;
for (Circle f: circles) S+=f.area();
for (Square f: squares) S+=f.area();
```

Проблема?

Выполнение одинаковых действий с объектами приходится
разделять из-за различий в типах.



ООП. Полиморфизм

```
class Figure{
public: virtual float area() const = 0;};

class Circle: public Figure{
    float r;
public: float area() const {return M_PI*r*r;}
};

class Square: public Figure{
    float a;
public: float area() const {return a*a;}};
};

list<Figure*> figs;
figs.push_back(new Circle());
figs.push_back(new Square());
...
float S = 0;
for (Figure *f: figs) S += f->area();
```



Outline

Прошлые темы

Qt



Сигналы и слоты

Сигнал (signal) - метод вызываемый во время события.

Слот (slot) - метод, принимающий сигнал.

"Соединяя" сигналы и слоты между собой с помощью специальной функции можно добиться автоматического вызова методов одного объекта, на вызов другого в режиме реального времени.



Сигналы и слоты

```
class A : public QObject{
    Q_OBJECT
public:    explicit A(QObject *parent = nullptr);
signals: void my_signal();
};

class B : public QObject{
    Q_OBJECT
public:    explicit B(QObject *parent = nullptr);
public slots: void my_slot(); // должен быть реализован
};

A *a = new A();
B *b = new B();
QObject::connect(a, SIGNAL(my_signal()),
                 b, SLOT(my_slot()));
a->my_signal(); // + автоматический вызов my_slot
```



Сигналы и слоты

Демонстрация работы сигналов и слотов



Ссылки и литература

1. Документация Qt
2. Сигналы и слоты в Qt5



Ссылки и литература

1. Г. Буч. Объектно-ориентированный анализ и проектирование с примерами приложений. 720 с. 2010 г. 700 страниц. Теория. Примеры на C++. Картинки! Вторая половина книги - примеры OOA и OOD с UML диаграммами.
2. MSDN - Microsoft Developer Network
3. Qt 5.X. Профессиональное программирование на C++. Макс Шлее. 2015 и более поздние издания г. 928 с. Книга периодически обновляется с выходом новых версий фреймворка Qt.
4. www.stackoverflow.com - система вопросов и ответов
5. draw.io — создание диаграмм.



Материалы курса

Слайды, вопросы к экзамену, задания, примеры

github.com/VetrovSV/OOP

