ООП

Семестр 2

Лекция 1

Черновик

Кафедра ИВТ и ПМ

2019

План

- Прошлые темы
- Стандартная библиотека (продолжение)
  - Контейнеры
    - stack
    - set
    - queue
    - map
    - regex
    - numeric
  - algorithm
- Классы в C#

Outline

ООП

- Что такое стандарт оформления кода?

ООП

- 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?

ООП

- 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?

ООП

- Что такое класс?

ООП

- Что такое класс?
- Что такое объект?

ООП

- 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?

ООП

- 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" th
      // 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)?

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 capasity;
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 capasity;
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 capasity;
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 capasity;
public:
    ...
};
```

ООП. Наследование

Наследование в фреимворках для создание приложений с GUI?

ООП. Наследование

Наследование в фреимворках 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 инте
// 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;}};
...
// 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?

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;}};
};

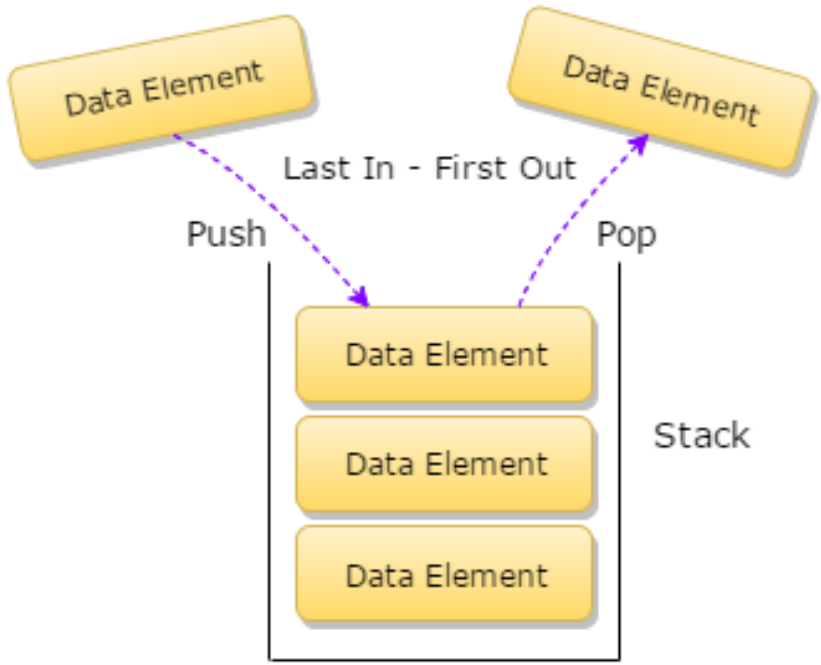
list<Figure*> figs;
figs.push_back(new Circle());
figs.push_back(new Square());
...
float S = 0;
for (Figure *f: figs) S += f->area();
```

Outline

Outline

Стэк

Стек (stack — стопка) — абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO (last in — first out, «последним пришёл — первым вышел»).



Стэк

```
#include <stack>

stack<int> s1;
s1.push(20);
s1.push(14);
s1.push(42);

int a;
a = s1.size(); // 3
a = s1.top(); // 42
s1.pop();
a = s1.top(); // 14
s1.pop();
a = s1.top(); // 20
s1.pop();

a = s1.empty(); // true (1)
```

http://www.cplusplus.com/reference/stack/stack/stack/

Множество

set

queue

map

regex

numeric

Outline

algorithm

Outline

Типы данных

Data Type	Range
byte	0 .. 255
sbyte	-128 .. 127
short	-32,768 .. 32,767
ushort	0 .. 65,535
int	-2,147,483,648 .. 2,147,483,647
uint	0 .. 4,294,967,295
long	-9,223,372,036,854,775,808.. 9,223,372,036,854,775,807
ulong	0 .. 18,446,744,073,709,551,615
float	-3.402823e38 .. 3.402823e38
double	-1.7976931348623e308.. 1.7976931348623e308
decimal	-79228162514264337593543950335 to 79228162514264337593543950335
char	A Unicode character.
string	A string of Unicode characters.
bool	True or False.
object	An object.

Структура программы

```
// подключение модулей
using System;

namespace YourNamespace
{
    class YourClass
    {
    }

    class YourMainClass
    {
        static void Main(string[] args)
        {
            //Your program starts here...
        }
    }
}
```

Классы в C#

- Класс - ссылочный тип
- Сборщик мусора

Классы в C#

Объявление классов и создание объектов

```
// перед классом модификатор доступа
public class MyClass
{
// поля и методы...
} // здесь нет точки с запятой

 class Program   {
    static void Main(string[] args) {
        // создание экземпляра класса
        // и сохранение его адреса в ссылке
        MyClass object1 = new MyClass();
        // объявление ссылки на класс
        MyClass object2;
        MyClass object3 = new MyClass();
        MyClass object4 = object3;
        // object3 и object4 идентичны,
        // т.е. указывают на один и тот же объект
        }
    }
```

Классы в C#

Поля

```
class SampleClass
{
    // модификатор доступа указывается перед каждым полем
    public string sampleField;

    // закрытое поле
    private string sampleField2;

    // открытое поле. константа
    public const int months = 12;
}
```

Классы в C#

Методы

```
class Example{
    public void method1(){
        Console.WriteLine("method1");
    }
    public void method3(int x){
        Console.WriteLine("method2" + x.ToString());
    }
}
//...
class YourMainClass
{
    static void Main(string[] args)
    {
        Example ex = new Example();
        ex.method1();
        ex.method3();
    }
}
```

- параметры значения
- ссылочные параметры
  модификатор ref при объявлении и вызове
- выходные параметры
  модификатор out при объявлении и вызове. Метод *обязан* присвоить им значение. Позволяет вернуть из метода несколько переменных
- массивы параметров
  позволяет передавать в метод переменное число аргументов.

Свойство — это член, предоставляющий гибкий механизм для чтения, записи или вычисления значения частного поля. Свойства можно использовать, как если бы они были членами общих данных, но фактически они представляют собой специальные методы, называемые методами доступа.

Классы в C#

Свойства

```
class TimePeriod
{  // поле
   private double _seconds;

   // свойство
   public double Hours
   {
        // получение значения
       get { return _seconds / 3600; }

       // задание значения
       set {
       // value - входной параметр
          if (value < 0 || value > 24) // проверка предусловий
             throw new ArgumentOutOfRangeException(
                    \$"{nameof(value)} must be between 0 and 24.");
          _seconds = value * 3600;
       }
   }
}
```

Классы в C#

Свойства. Использование

```
static void Main(string[] args)
        {
            TimePeriod p = new TimePeriod();
            p.Hours = 20;   // вызывается метод set
            double h = p.Hours; // вызывается метод get
        }
```

Ссылки и литература

1. dotnetfiddle.net - онлайн интерпретатор C#

Материалы курса

Слайды, вопросы к экзамену, задания, примеры

github.com/VetrovSV/OOP