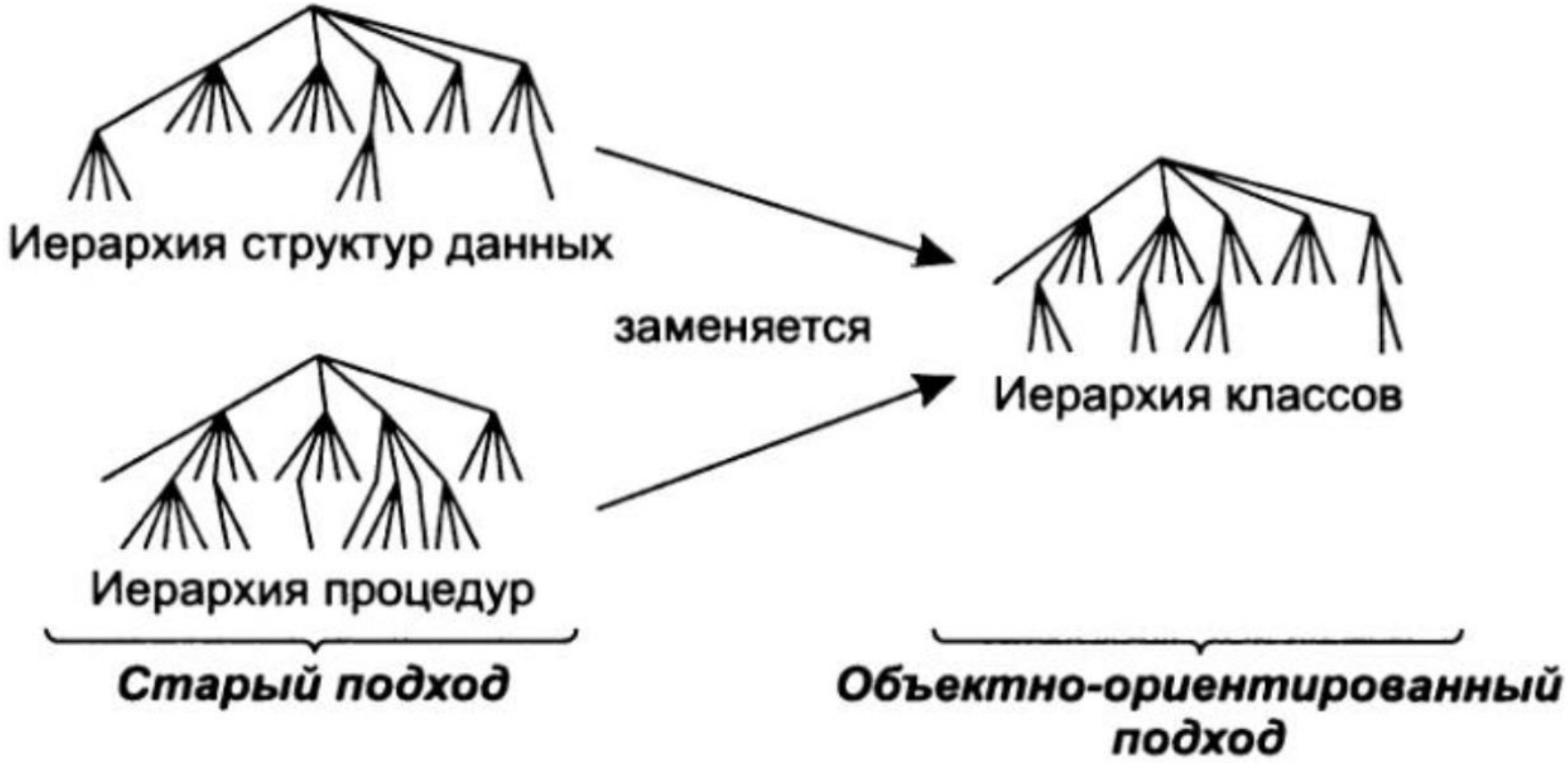
Переход от процедурного подхода к объектно-ориентированному

1. Процедурный и объектно-ориентированный подходы
2. Пример перехода от процедурного стиля к ООП
3. Примеры непонимания ООП
4. Пример создания классов в разных ОО-языках

Преподаватель:
Ботов Дмитрий Сергеевич

Переход от процедурного подхода к ООП

Единая иерархия операций и структур данных:



Пока в процедурном программировании…

Шаг 0. Всё в одной функции

```
#include <math.h>
#include <stdio.h>

// Подсчитать периметр треугольника

void main()
{
    float x1 = 6, y1 = 5;
    float x2 = 3, y2 = 3;
    float x3 = 1, y3 = 2;

    float perimeter;

    perimeter = sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
    perimeter += sqrt((x2-x3)*(x2-x3)+(y2-y3)*(y2-y3));
    perimeter += sqrt((x3-x1)*(x3-x1)+(y3-y1)*(y3-y1));

    printf("perimeter of the triangle = %f", perimeter);
}
```

Все еще в процедурном программировании…

Шаг 1. Вынесение дублирующегося кода в функцию

```
float getDistance(float x1, float y1, float x2, float y2)
{
    return sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
}
```

Все еще в процедурном программировании…

Шаг 1. Использование функции

```
void main()
{
    float x1 = 6, y1 = 5;
    float x2 = 3, y2 = 3;
    float x3 = 1, y3 = 2;

    float perimeter;

    perimeter = getDistance(x1, y1, x2, y2);
    perimeter += getDistance(x2, y2, x3, y3);
    perimeter += getDistance(x3, y3, x1, y1);

    printf("perimeter of the triangle = %f", perimeter);
}
```

Структуры в процедурной (структурной) парадигме

Шаг 2. Выделение структуры и использование в качестве параметров функции

```
struct Point
{
    float x;
    float y;
};

float getDistance(Point p1, Point p2)
{
    return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}
```

Структуры в процедурной (структурной) парадигме

Шаг 3. Выделение структуры и использование в качестве параметров функции

```
struct Point
{
    float x;
    float y;
};

float getDistance(Point p1, Point p2)
{
    return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}
```

Структуры в процедурной (структурной) парадигме

Шаг 3. Использование структуры

```
void main()
{
    Point a, b, c;
    a.x = 6, a.y = 5;
    b.x = 3, b.y = 3;
    c.x = 1, c.y = 2;

    float perimeter;

    perimeter = getDistance(a, b);
    perimeter += getDistance(b, c);
    perimeter += getDistance(c, a);

    printf("perimeter of the triangle = %f", perimeter);
}
```

Начинаем переходить к ООП

Шаг 4. Переход к классам: объединение данных и операций над ними

```
class Point
{
private:
    float x;
    float y;

public:

    Point(float x, float y);

    float getDistance(Point p);
};
```

Начинаем переходить к ООП

Шаг 4. Реализация класса Point

```
Point::Point(float x, float y)
{
    this->x = x;
    this->y = y;
}

float Point::getDistance(Point p)
{
    return sqrt((x-p.x)*(x-p.x)+(y-p.y)*(y-p.y));
}
```

Пример внедрения ООП в код

Шаг 4. Использование возможностей класса

```
void main()
{
    Point a(6,5), b(3,3), c(1,2);

    float perimeter;

    perimeter = a.getDistance(b);
    perimeter += b.getDistance(c);
    perimeter += c.getDistance(a);

    printf("perimeter of the triangle = %f", perimeter);
}
```

Пример внедрения ООП в код

Шаг 5. Использование классов в качестве основы других классов

```
class Triangle
{
private:
    Point a;
    Point b;
    Point c;

public:
    Triangle(Point a, Point b, Point c);
    float getPerimeter();
};
```

Пример внедрения ООП в код

Шаг 5. Реализация класса Triangle

```
Triangle::Triangle(Point a, Point b, Point c)
{
    this->a = a;
    this->b = b;
    this->c = c;
}

float Triangle::getPerimeter()
{
    float perimeter = a.getDistance(b);
    perimeter += b.getDistance(c);
    perimeter += c.getDistance(a);
    return perimeter;
}
```

Пример внедрения ООП в код

Шаг 5. Использование класса Triangle

```
void main()
{
    Point a(6,5), b(3,3), c(1,2);
    Triangle myTriangle(a,b,c);

    printf("perimeter of the triangle = %f",
           myTriangle.getPerimeter());
}
```

Пример непонимания ООП

```
class Point
{
private:
    float x;
    float y;

public:

    Point(float x, float y);

    float getDistance(int x2, int y2);
};

float Point::getDistance(int x2, int y2)
{
    return sqrt((x-x2)*(x-x2)+(y-y2)*(y-y2));
}
```

Не делайте так:
Использование примитивных типов вместо созданных классов

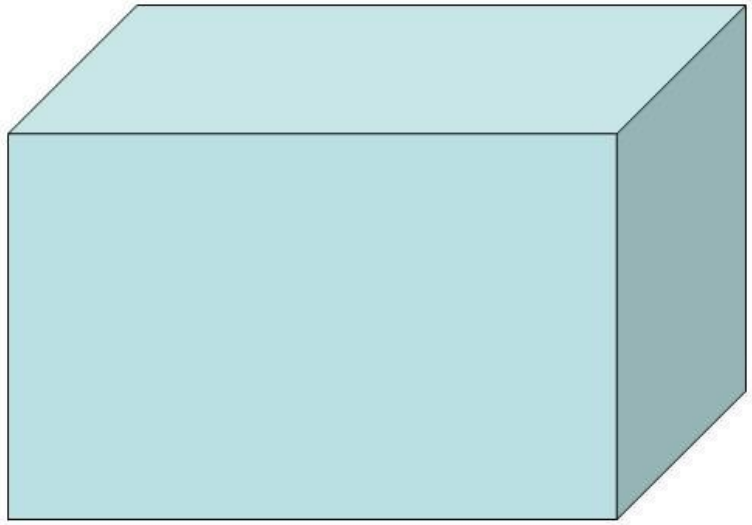
Пример непонимания ООП

```
class Triangle
{
private:
     Point a;
     Point b;
     Point c;

public:
     Triangle(Point a, Point b, Point c);
     float getPerimeter(Point a, Point b, Point c);
};

float Triangle::getPerimeter(Point a, Point b, Point c)
{
     float perimeter = a.getDistance(b);
     perimeter += b.getDistance(c);
     perimeter += c.getDistance(a);
     return perimeter;
}
```

Не делайте так:

Передача состояния объекта через параметры методов: непонимание связи состояния и поведения объекта

Пример простейшего объекта



Прямоугольный параллелепипед

Класс на C++

```
// объявление класса
class Box
{
   private:
      int _width;
      int _height;
      int _length;

   public:
      Box(int width, int height, int length);
      int getVolume();
      void printBox();
}
```

Класс на C++

```
// определение конструктора и методов класса
    Box::Box(int width, int height, int length)
    {
        _width  = width;
        _height = height;
        _length = length;
    }

    int Box::getVolume()
    {
        return _width * _height * _length;
    }

    void Box::printBox()
    {
        printf("Box: %d %d %d ", _width, _height, _length);
        printf("Volume = %d\n", getVolume());
    }
```

Тот же класс на Java

```
public class Box {
    private int _width;
    private int _height;
    private int _length;

    public Box(int width, int height, int length) {
        _width  = width;
        _height = height;
        _length = length;
    }

    public int getVolume() {
        return _width * _height * _length;
    }

    public void printBox() {
        System.out.print("Box: "+ _width + _height + _length);
        System.out.println("Volume = " + getVolume());
    }
}
```

И опять этот же класс, но на C#

```
public class Box {
    private int _width;
    private int _height;
    private int _length;

    public Box(int width, int height, int length) {
        _width  = width;
        _height = height;
        _length = length;
    }

    public int getVolume() {
        return _width * _height * _length;
    }

    public void printBox() {
        Console.Write("Box: "+ _width + _height + _length);
        Console.WriteLine("Volume = " + getVolume());
    }
}
```