

# Объектно-ориентированное программирование

## Лекция 2. Парадигмы

П. А. Макаров



СЫКТЫВКАРСКИЙ  
ЛЕСНОЙ  
ИНСТИТУТ

23 сентября 2022 г.

1. Парадигмы как явление
2. Парадигмы программирования
3. Обзор основных парадигм
4. Объектно-ориентированное программирование
5. Совместимость C и C++
6. Контрольные вопросы и задания

Слово “парадигма” заимствовано из греческого языка (*παράδειγμα* — пример/модель).

## Определение 1

Согласно Т. Куну, **парадигма** — это устоявшаяся система научных взглядов, в рамках которой ведутся исследования. Парадигмы сменяют друг друга, уточняя и улучшая наши представления о том или ином явлении.

## Пример 1

Геоцентрическая небесная механика Птолемея и гелиоцентрическая система Коперника.

Некоторые элементы философии UNIX, которую можно представлять как некую общую парадигму разработчика:

- универсальная парадигма “всё есть поток байтов”;
- всё есть текст;
- каждая программа должна решать одну задачу, и делать это хорошо;
- KISS — Keep It Simple, Stupid!

- Модульное и структурное программирование.
- Событийно-управляемое программирование.
- Параллельное программирование.
- Программирование с разделяемыми данными.
- Рекурсия как парадигма.

# Парадигмы программирования

Современный смысл термина



Рис. 1: Роберт Флойд

## Определение 2

*Парадигма программирования — комплекс концепций, используемых программистом при проектировании программы.*

- Императивное программирование.
- Процедурное программирование.
- Функциональное программирование.
- Логическое программирование.
- ООП.

### Базовые термины

- объект, свойства и методы,
- события,
- интерфейс,
- класс.

### Ключевые идеи ООП (four pillars of Object Oriented Programming)

- инкапсуляция (encapsulation),
- абстракция данных (data abstraction),
- полиморфизм (polymorphism),
- наследование (inheritance).



### Определение 3

*Инкапсуляция — это механизм сокрытия реализации данных путем ограничения доступа к публичным методам. Для этого переменные данного экземпляра класса (конкретного объекта) сохраняются приватными (*private*), а методы-аксессоры делаются доступными (*public*).*

```
1 class Employee {
2     private:
3         String name;
4         Date dob;
5     public:
6         String getName() { return name; }
7         void setName(String name) { this.name = name; }
8         Date getDob() { return dob; }
9         void setDob(Date dob) { this.dob = dob; }
10 };
```

Листинг 1: Пример инкапсуляции

### Определение 4

*Абстракция — подход, в котором концепции или идеи, отделяются от конкретного экземпляра класса. Используя абстрактный класс/интерфейс, мы выражаем “намерение” класса, а не его фактическую реализацию. В некотором смысле, один класс не должен знать внутренние детали другого, чтобы использовать его, достаточно знать интерфейсы.*

### Определение 5

*Наследование — способность одного объекта (класса) базироваться на другом объекте (классе). Это главный механизм для повторного использования кода. Наследственное отношение классов четко определяет их иерархию.*

### Определение 6

*Полиморфизм — реализация некоторым множеством способов различных задач с одним и тем же названием (и сходным смыслом).*

- *Статический полиморфизм достигается с помощью перегрузки методов (*method overloading*);*
- *Динамический полиморфизм — с помощью переопределения методов (*method overriding*). Тесно связан с наследованием. С помощью динамического полиморфизма код, который работает на суперклассе (*superclass*), будет работать с любым типом подкласса (*subclass*).*

- Не повторяйся (DRY — Don't repeat yourself).
- Принцип единственной обязанности.
- Принцип открытости/закрытости.
- Принцип подстановки Барбары Лисков.
- Принцип разделения интерфейсов.
- Принцип инверсии зависимостей.

# Совместимость С и С++

```
1 int try;  
2 long new;
```

## Листинг 2: Ключевые слова

```
1 struct mystruct {  
2     int a, b;  
3 };
```

## Листинг 3: Структуры в С++

```
1 typedef struct mystruct {  
2     int a, b;  
3 } mystruct;
```

## Листинг 4: Структуры в С

```
1 struct point {  
2     double x, y, z;  
3 };  
4 point A, B, C; // struct point A, B, C;
```

## Листинг 5: Использование структур в С++

1. Изучите основные идеи философии UNIX, сформулированные Дугом Макилроем, Майком Ганцарзом и Эриком Рэймондом.
2. В чём разница между параллельным программированием и программированием с разделяемыми данными? Действительно ли это различные концепции? Почему?
3. Какие языки позволяют следовать парадигмам, перечисленным в разделе 3?
4. Есть ли существенная разница между императивной и процедурной парадигмами? Какие чисто императивные языки вам известны?
5. Определите значения всех понятий, упомянутых в разделе 4.
6. Какие парадигмы позволяет использовать программисту современный C++? Python?