[ИУ9] Основы информатики

Лекция 1. Основные понятия информатики и программирования

Данные — представление фактов, понятий, инструкций в форме, приемлемой для обмена, интерпретации или обработки человеком или с помощью автоматических средств.

Алгоритм — конечная совокупность точно заданных правил решения произвольного класса задач или набор инструкций, описывающий порядок действий исполнителя для решения некоторой задачи.

Свойства алгоритма:

1. **Дискретность** — наличие структуры, разбитие на отдельные команды, понятия, действия.
2. **Детерминированность** — для одного и того же набора данных всегда один и тот же результат.
3. **Понятность** — элементы алгоритма должны быть понятны исполнителю.
4. **Завершаемость** — алгоритм завершается за конечное число шагов.
5. **Массовость** — применимость алгоритма для некоторого класса похожих задач.
6. **Результативность** — алгоритм должен выдавать результат.

Компьютерная программа — алгоритм, записанный на некотором языке программирования.

Язык программирования — формальный язык, предназначенный для записи компьютерных программ.

Компьютер — универсальное программно-управляемое устройство для обработки информации (данных).

Парадигмы программирования — совокупность идей и понятий, определяющих стиль написания компьютерных программ (подход к программированию). Это способ концептуализации, определяющий организацию вычислений и структурирование работы, выполняемой компьютером.

Основные парадигмы программирования делятся на три большие группы:

1. Императивное программирование.
2. Декларативное программирование.

3. Метапрограммирование.

Императивное программирование — способ записи программ, в котором указывается последовательность действий.

Основной признак императивной парадигмы (группы парадигм) — оператор деструктивного присваивания. Слово «деструктивное» означает, что присваивание может изменять значение, хранящееся в переменной — старое теряется безвозвратно, заменяясь новым значением.

Декларативное программирование — способ записи программ, в котором описываются взаимосвязь между данными; описывается цель, а не последовательность шагов для её достижения. Деструктивного присваивания в декларативной парадигме нет. Возможно лишь однократное присваивание значения при создании новой переменной.

Метапрограммирование — программа становится объектом управления со стороны программы — той же или другой.

В императивной группе выделяют три основные парадигмы.

1. **Структурное программирование** — каждый блок программы имеет ровно один вход и ровно один выход (кострукции вроде `goto`, `break`, `return` из середины функции, `continue` запрещены). В программе используются три основные управляющие констркции: следование (`{…}` в Си, `begin` в Scheme), ветвление (`if` / `else` в Си, `if` и `cond` в Scheme) и цикл (`while`, `for` в Си, `do` в Scheme).
2. **Процедурное программирование** — в рамках этого подхода программа рассматривается как набор подпрограмм, которые вызывают друг друга.
3. **Объектно-ориентированное программирование (ООП)** — программа описывается как набор взаимодействующих друг с другом объектов. Объект объединяет в себе данные и поведение (код), объекты могут посылать друг другу сообщения.

Декларативная парадигма:

1. **Функциональное программирование** — алгоритм описывается как набор функций; порядок вычисления функций не существенен и на результат влиять не должен. В ленивых языках (например, Haskell) функции вызываются только когда нужен их результат.
2. **Логическое программирование** — алгоритм описывает взаимосвязь между понятиями; выполнение программы сводится к выполнению запросов. Представлено почти исключельно языком Prolog, сильно отчасти — SQL.

Пример. Требуется отсортировать последовательность чисел по возрастанию. Как это будет выглядеть в разных парадигмах.

1. Императивное программирование. Последовательность находится в массиве `numbers`. Для упорядочивания вызывается процедура `sort(array)`, меняющая содержимое

своего аргумента:

```
{ Паскаль }
sort(numbers);
```

После вызова процедуры в массиве `numbers` будут находиться те же числа, что и ранее, но в порядке возрастания.

2. Функциональное программирование. Имеем список `numbers`, функция `sort` формирует новый список `sorted_numbers`, где будут располагаться те же числа, но по возрастанию:

```
-- Хаскель
sorted_numbers = sort numbers
```

Содержимое списка `numbers` остаётся прежним.

3. Логическое программирование. Тут всё интересно. Определяем предикат

```
% Пролог
unsorted_sorted(Unsorted, Sorted) :- …
```

Теперь рассмотрим обращения к предикату:

```
?- unsorted_sorted([8, 2, 5, 1], X).
  X = [1, 2, 5, 8];
false.
```

Получили сортированный список для несортированного

```
?- unsorted_sorted(X, [1, 2, 3]).
  X = [1, 2, 3];
  X = [1, 3, 2];
  X = [2, 1, 3];
  X = [2, 3, 1];
  X = [3, 1, 2];
  X = [3, 2, 1];
false.
```

Нашлись все перестановки сортированного списка.

```
?- unsorted_sorted(X, [1, 3, 2]).
false.
```

Для исходного списка не по возрастанию предикат не выполняется.

Парадигма метапрограммирования:

1. **Программы пишут программы:** макросы, генераторы кода, шаблонное метапрограммирование в C++.
2. **Рефлексия (интроспекция)** — программы взаимодействуют с вычислительной средой.

Подпрограмма — именованный блок кода; вызывающая программа приостанавливается, управление передаётся подпрограмме. При завершении работы подпрограммы вызывающая программа возобновляет свою работы; процедуры, функции, методы — разновидности подпрограмм.

Сопрограмма — в отличие от подпрограмм работает поочередно с вызывающей программой, при следующем вызове она возобновляет свою работу с точки остановки.