Informatics basics

Starovoytov Alexandr, Grechko Georgy

<2021-09-04 Sat 13:50>

Contents

- Лекции в pdf формате
- Лекции в виде сайта
- Преподаватель
  - Коновалов Александр Владимирович
  - 89175404352
  - akonovalov@bmstu.ru
  - a.v.konovalov87@mail.ru

1 Основные понятия информатики

1.1 Данные

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

1.2 Алгоритм

это конечная совокупность точно заданных правил решения прозвольного класса задач или набор инструкций, описывающий порядок действий исполнителя для решения некотрой задачи

1.3 Свойства алгоритма:

1. Дискретность
   делится на отдельные элементарные части, отвечающие за определенные действия.
2. Детерминированность
   на одних и тех же входных данных - один и тот же результат.
3. Понятность
   элементы алгоритма должны быть понятны исполнителю.
4. Завершаемость
   если не завершается - то это вычислительный процесс.
5. Массовость
   алгоритм пригоден для решения всех задач данного типа.
6. Результативность
   указывает на наличие таких исходных данных, для которых реализуемый по заданному алгоритму вычислительный процесс должен через конечное число шагов остановиться и выдать искомый результат.

1.4 Компьютерная программа

это алгоритм, записанный на некотором языке программирования

1.5 Язык программирования

формальный язык, предназначенный для записи компьютерных программ

1.6 Компьютер

программно управляемое устройство для обработки информации

1.7 Подпрограмма

некоторый именованный блок кода.
Вызывающая программа приостанавливается, управление передается подпрограмме, по завершению управление передается обратно

1.8 Сопрограмма

в отличие от подпрограммы работает поочередно с вызывающей программой.
При повторном вызове возобновит выполнение с момента, где остановилась

2 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.

2.1 Основные группы парадигм

1. Императивные
   способ записи программ, в котором указывается последовательность действий
   основной признак - оператор присваивания (меняющий значение переменной)
   (a) Структурное программирование
   программа является композицией блоков с одним входом и одним выходом
   (есть операторы ветвления и цикла, но нет goto)
   (b) Процедурное программирование
   совокупность подпрограмм, где одни подпрограммы вызывают другие
   (c) Объектно-ориентированное
   программа рассматривается как набор некоторых взаимодействующих объектов.
   Объект сочетает в себе данные и методы их обработки, методы

вызываются в ответ на сообщение:
посылаем сообщение объекту -> вызывает метод -> возвращает ответ
объекты объединяются в классы

2. Декларативные
   способ записи программ, в котором описывается взаимосвязь между данными, описывается цель, но не последовательность шагов ее вычисления

   (a) Функциональное
   алгоритм записывается как набор взаимосвязанных функций, функции рассматриваются с математической точки зрения, описывает взаимосвязь между данными и результатом

   ```
   x = f(y) + g(z);
   ```

   (b) Логическое (Prolog + отчасти SQL)
   алгоритм описывает взаимосвязь между понятиями.
   Выполнение программы сводится к выполнению запросов

3. Метапрограммирование
   метаирония. Программа рассматривается как данные/объект для другой программы

   (a) Программу пишут программы
   Например: макросы, генераторы кода, шаблоны C++ (фигня нечитаемая зачем он постоянно в руках часы крутит попит симпл димпл круче нет попит симпл димпл попит симпл димпл маленький красивый попит большой и милый)

   (b) Программа взаимодействует с вычислительной средой
   Рефлексия или интроспекция - самоанализ.
   Например, посмотреть поля и значения в объекте класса

2.2 примеры

```
ИП: sort(array);
ФП: sorted_list = sort(list)
    sorted_list - константа, т.к. нет присваивания
ЛП: sort(unsorted, sorted)
```

3 язык Scheme

3.1 информация о языке

Lisp 1950-е годы Джон МакКарти
LISt Processing
Scheme 1970-е годы Абельсон и Сассман
изучаем R5RS
современная редакция R7RS

- другие языки семейства LISP:
  - Common Lisp
  - Clojure
  - Racket

3.2 основные постулаты языка LISP

1. единство кода и данных
2. все есть список
3. выражение является списком
   операция указывается в первом элементе
4. все выражения вычисляют значения (почти все)

3.3 операторы

```
;; <терм> ::= <атом>|<список>
;; <атом> ::= <перем>|<число>|<симол>|<строка>
;; <список> ::= (<термы>)
;; <термы> ::= <пусто> | <терм><термы>
;; <пусто> ::=
;; i-am-variable - имя переменной может включать "-", "+", "*", "/"...
(+ 1 2 3) ;; список
;; "+" - имя функции сложения
;; "1 2 3" - аргументы
;; элементы списка разделяются пробелами
```

```
((a 1)(b 2)(c 3)) ;; список списков

;; по умолчанию выражение вида
;; (f a b c) - вызов функции
;; ((f x) y) - сначала вызывается (f x),
;; то, что она вычислит с аргументом y
;; Особая форма - первый элемент списка - ключевое слово

(define pi 3.1415926)
;; pi - имя переменной
;; 3.14.. - значение

(define pi (* 4 (atan 1)))

(define (f a b c) ;; опеределяет функции
    (+ (* 2 a) (/ b c)))

;; (if <выраж 1>
;;     <выраж 2>
;;     <выраж 3>)

;; Если <выраж 1> - ложь,
;; значением (if ...) станет значение <выраж3>
;; (<выраж 2> не вычисляется), и наоборот

;; #f - ложь
;; #t - истина
;; if считает истиной все что не ложь

(if 1 2 3)

2

(if #f (/ 1 0) (/ 10 5))


2

(if #t (/ 1 0) (/ 10 5))

;; (< x y)
```

```
;; (> x y)
;; (= x y)
;; >=
;; <=
;; (/ x y)
(/ 10 3)
```

(quotient x y) - целое деление (quotient 30 7) -> 4 (remainder x y) -> остаток (remainder 30 7) -> 2

;; “особые формы” (and e1 e2 ... en) - возвращает #f если хотя бы один ei = #f, иначе en (or ...) - вернет первое не ложное значение

(and 1 2 #f (/ 1 0)) -> #f (or (= x 0) (/ 1 0)) - не приведет к ошибке деления на 0, никогда (not x) - логическое отрицание #f -> #t not x -> #f, x != #f вычисления ленивые! ,#+end$_{\text{example}}$

,#+begin$_{\text{src}}$ scheme (+ 1 2 3) #+end$_{\text{src}}$

```
(+ 1 2 3 4)

(define pi (* 4 (atan 1)))

(define (area r) (* r r))

(define e (exp 1))

(define (hypot x y) (sqrt (+ (* x x) (* y y))))
; if всегда в 3 строки
(define (fac x) (if (= x 0)
                    1 (* (fac (- x 1)) x)))

(define (my-positive? n) (> n 0))
(define (my-abs x)
  (if (< x 0)
      (-x)
      x))

(define (factorial n)
  (define loop acc i)
  (if (> i n)
      acc
      (loop (* acc i) (+ i 1)))
```

```
  (loop 1 1))

(define (my-odd n)
  (= 0 (remainder n 2)))

(define (my-even n)
  (= 1 (remainder n 2)))


(my-odd 5)
(my-even 5)
```