Введение в Лисп

Основоположник

- Джон Маккарти предложил проект языка Лисп (LISP - LISt Processing) в качестве средства исследования границ применимости компьютеров, в частности, методом решения задач искусственного интеллекта.

- Существует и активно применяется более трехсот диалектов Лиспа и родственных ему языков: Interlisp, muLisp, Clisp, Scheme, ML, Cmucl, Logo, Hope, Sisal, Haskell, Miranda и др.

Математические основы

Сформулированная Джоном Маккарти (1958) концепция символьной обработки информации компьютером восходит к идеям Черча и других математиков, известным как лямбда-исчисление с конца 20-х годов прошлого века. Выбирая лямбда-исчисление как теоретическую модель, Маккарти предложил рассматривать функции как общее базовое понятие, к которому достаточно естественно могут быть сведены все другие понятия, возникающие при программировании

Лямбда-исчисление

Лямбда-исчисление (λ-исчисление) - формальная система, разработанная американским математиком Алонзо Чёрчем, для формализации и анализа понятия вычислимости.

λ-исчисление может рассматриваться как семейство прототипных языков программирования. Их основная особенность состоит в том, что они являются языками высших порядков. Тем самым обеспечивается систематический подход к исследованию операторов, аргументами которых могут быть другие операторы, а значением также может быть оператор. Языки в этом семействе являются функциональными, поскольку они основаны на представлении о функции или операторе, включая функциональную аппликацию и функциональную абстракцию.

Чистое лямбда-исчисление

Это простейший из семейства прототипных языков программирования, чистое λ-исчисление, термы которого, называемые также объектами («обами»), или λ-термами, построены исключительно из переменных применением аппликации и абстракции. Изначально наличие каких-либо констант не предполагается.

В основу λ-исчисления положены две фундаментальные операции: аппликация (означает применение или вызов функции по отношению к заданному значению) и абстракция (конструирует новые функции по заданным выражениям).

Универсальность понятия «функция»

Универсальность понятия "функция" и разнообразие видов его применения позволяет унифицировать используемые при описании процессов понятия "действие", "значение", "формула", "переменная", "выбор варианта" и пр. Все это - разные категории функций с различными формами унифицированного представления (записи, изображения) в тексте программы и правилами интерпретации (выполнения, вычисления), обеспечивающими получение результата функции при исполнении программы. Аргументами функции могут быть готовые данные или результаты других функций. Возможны ограничения на типы данных, допускаемых в качестве аргументов - тогда речь идет о частичных функциях.

Основные особенности ЛИСПа

- Унификация понятий "функция" и "значение" – представление функций можно строить из их частей и вычислять по мере поступления и обработки информации.
- Интерпретирующий и компилирующий режим работы – в одной программе могут иметься как откомпилированные, так и интерпретируемые функции.
- Интегральность ограничений – если не хватает памяти, то не отдельные блоки данных, а на всю задачу. При этом «мусорщик» автоматически будет искать свободное место.
- Уточняемость решений – реализация языка обычно содержит списки свойств объектов, приспособленные к внешнему доопределению отдельных элементов.
- Множественность определений имен – обеспечение полиморфизма и более общих схем обработки вариантов функциональных построений.

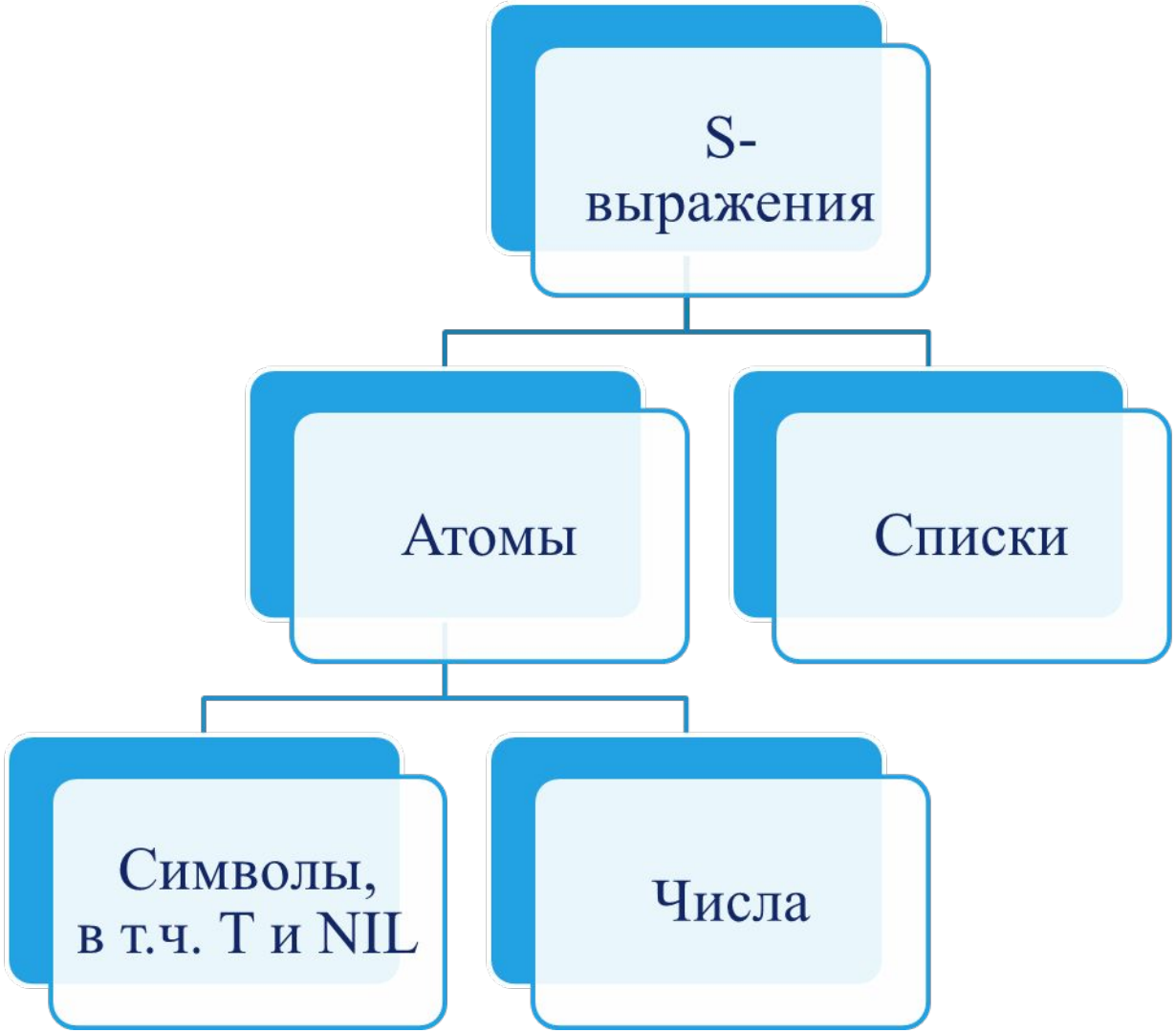
Основные особенности ЛИСПа

- Одинаковая форма данных и программ – списочные структуры. ЛИСП-программы могут обрабатывать не только данные, но и другие ЛИСП-программы и самих себя. При трансляции можно сформировать выражение и проинтерпретировать его в качестве программы и выполнить ее.
- Хранение данных, не зависящее от места размещения в ОЗУ – порядок следования в памяти списочных ячеек с программами и данными не существенен. Добавление/удаление элементов производится без переноса всего списка в другие ячейки памяти.
- Функциональная направленность – в результате каждого действия возникает некоторое значение, которое становится элементом следующих действий.

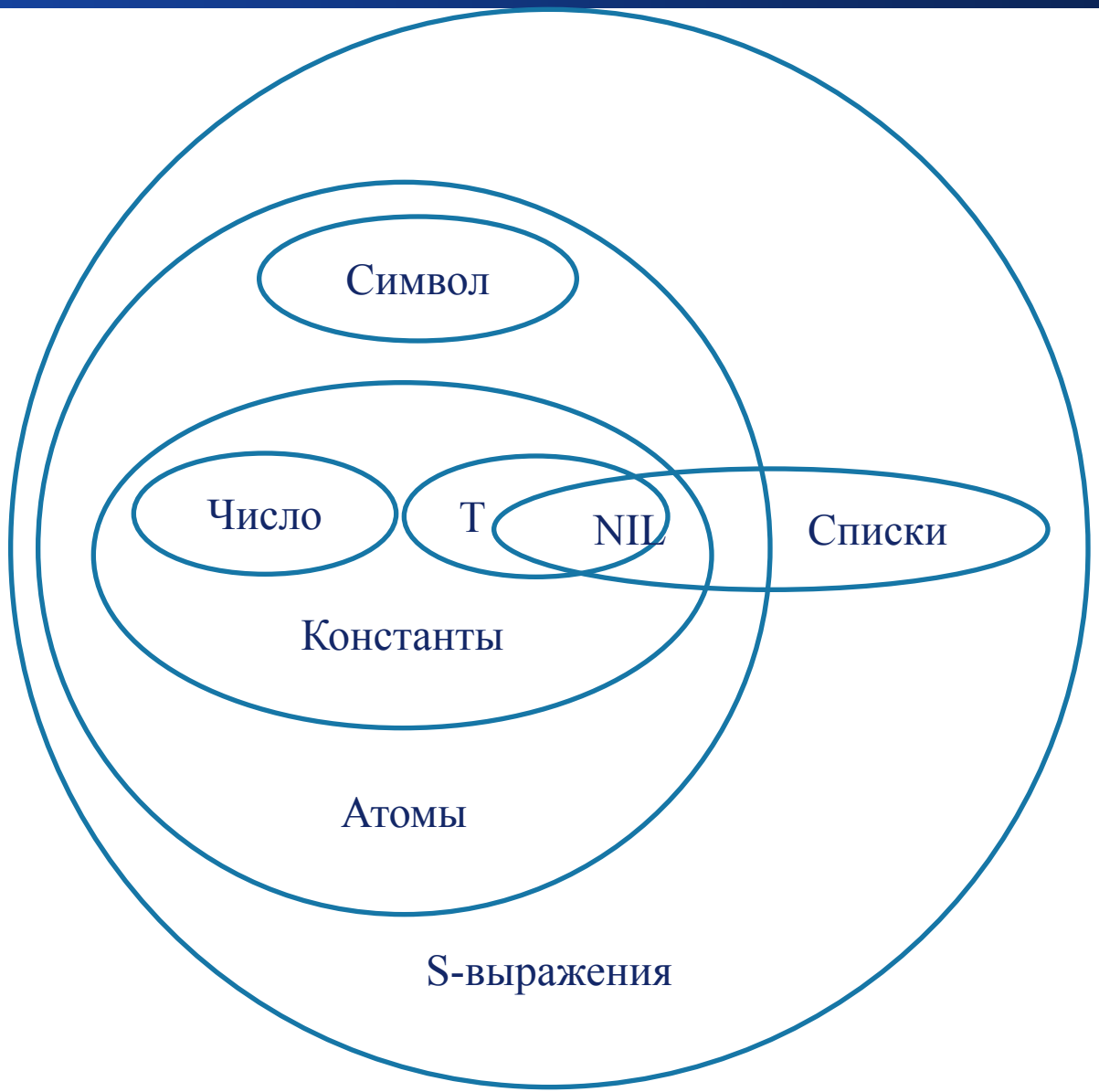
Основные особенности ЛИСПа

- Нестрогая типизация данных – типы сопровождают лишь сами объекты, а не имена символов, списков, функций (позднее связывание). Одни и те же переменные могут представлять объекты разных типов.
- Пошаговое программирование – программирование и тестирование осуществляются функция за функцией, которые последовательно определяются и тестируются.
- ЛИСП одновременно является языком и прикладного, и системного программирования.

Структура данных в ЛИСПе



Чем ошибочна данная схема?



Примеры

- Символы: x, u-1997, символ, function
- Числа: 24, 35.6, 6.3e5
- Списки: (a в (c o) p), (+ 3 6), ((A B) (D C))
- Равнозначность списков:
  - (A B C) = (A (B (C Nil)))
  - ((A B) C) = ((A (B Nil)) (C Nil))
  - (A) = (A Nil)
  - ((A)) = ((A Nil) Nil)
  - (A (B C)) = (A ((B C) Nil))
  - () = Nil
  - (()) = (Nil Nil)

Базис ЛИСПа

Примитивные функции:

- cons – строит списки из бинарных узлов. Первый аргумент произвольного вида – в левой части узла, второй аргумент (список) – в правой части узла.
- car – доступ к первому элементу списка
- cdr – доступ к части списка после удаления его первого элемента
- atom – позволяет различать составные (результат – «ложь») и атомарные (результат – «истина») объекты
- eq – проверка атомарных объектов на равенство

Примеры вычисления по примитивным функциям

- (CAR (CONS x y)) = x
- (CDR (CONS x y)) = y
- (ATOM (CONS x y)) = Nil
- (CONS (CAR x) (CRD x)) = x – для неатомарных x
- (EQ x x) = T – если x – атом
- (EQ x y) = Nil

Примеры вычисления по примитивным функциям

- (CAAR ((A) B C)) = A
- (CADR (A B C)) = B
  вычисляется CDR, затем CAR
- (CADDR (A B C)) = C
  вычисляется дважды CDR, затем CAR
- (CADADR (A (B C) D)) = C
  вычисляется два раза (CDR, затем CAR)

Специальные функции

- quote – объявление константы, предохраняющее аргумент от вычисления
- label / defun / de / csetq – конструктор функций, первый аргумент – имя функции (результат), второй – определение функции
- lambda – определение функции
  (lambda (x1 x2 … xn) fn)
  fn – тело функции
  xi – параметры определения, которые имеют аргументы в теле функции
- cond – ветвление
  (cond (условие действие по первой ветке)
  (T действие по второй ветке))

Примеры вычисления по специальным функциям

- (QUOTE A) – константа A объявлена
- (QUOTE (A B C)) – константа (A B C) объявлена
- (ATOM (QUOTE A)) = T – аргументом предиката является атом A
- (ATOM (QUOTE (A B C))) = Nil – аргументом предиката является список (A B C)
- (ATOM A) – значение не определено, т.к. оно зависит от вхождения переменной A, а ее значение зависит от контекста и должно быть определено или задано до попытки выяснить, атом ли это
- (COND ((EQ (CAR x) (QUOTE A))
  (CONS (QUOTE B) (CDR x)))
  (T x))

Примеры вычисления по специальным функциям

- (LABEL пример - имя новой функции
  (LAMBDA (x) - параметры функции
  (CAR (CDR x))) - тело функции
)

- (LABEL рекурсия (LAMBDA (x) (COND ((ATOM x) x)
  (T (рекурсия (CAR x)))))