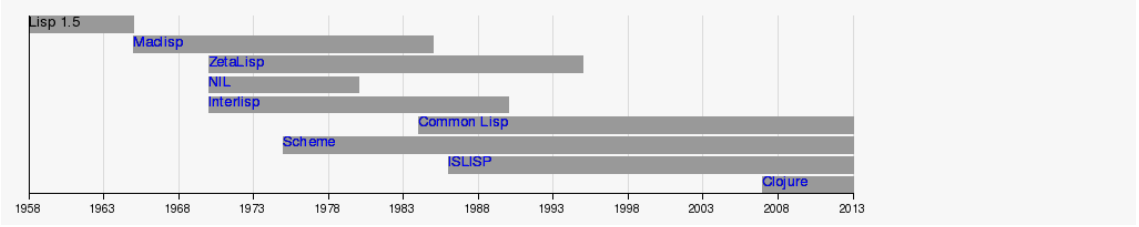
ДЕКЛАРАТИВНОЕ ПРОГРАММИРОВАНИЕ

Лекция 1. Введение в функциональное программирование.

Концепция функционального программирования.

Функциональным называется программирование при помощи функций в математическом их понимании. Функциональное программирование основано на следующей идее : в результате каждого действия возникает значение, которое может быть аргументом следующего действия. Программы строятся из логически расчлененных определений функций. Каждое определение функции состоит из организующих вычисления управляющих структур и из вложенных, в том числе вызывающих самих себя (рекурсивных) вызовов функций.

Язык LISP (LISt Processing) – язык программирования высокого уровня, разработан в 1958 году Дж. Маккарти. В основе Лиспа лежит функциональная модель вычислений, ориентированная прежде всего на решение задач нечислового характера.



Символьная обработка и искусственный интеллект.

Искусственный интеллект (ИИ) – область исследований по моделированию интеллектуальной деятельности человека для решения различных задач. ИИ находится на стыке ряда наук : информатики, языкознания (математической лингвистики), психологии (когнитивной науки) и философии. Задачи ИИ требуют работы с данными и знаниями в виде символьных структур. В обработке символьной информации важна не только форма рассматриваемых знаний, но и их содержание и значение. Причиной возникновения в конце 1950-х годов потребности в специализированных инструментальных программных средствах символьной обработки послужила неспособность процедурных языков отражать естественным образом в виде чисел и массивов объектов и ситуаций реального мира. Указанный недостаток процедурных средств, в частности, затруднял реализацию применяемых в решении задач ИИ эвристических методах.

Особенности функционального программирования.

1.Вызов функций является единственной разновидностью действий, выполняемых в функциональной программе,

2.В алгоритмических языках программа является последовательностью операторов, вызовов процедур в соответствии с алгоритмом. В функциональном программировании программа состоит из вызовов функций (рис. 1) и описывает то, что нужно делать и что собой представляет результат решения, а не как нужно действовать для получения результата.

Особенности функционального программирования (продолжение 1).

3. Основными методами программирования являются суперпозиция функций и рекурсия.

4.Функциональное программирование есть программирование, управляемое данными. В строго функциональном языке однажды созданные (введенные)

данные не могут быть изменены !

5.В алгоритмических языках с именем переменной связана некоторая область памяти, соответствие строго сохраняется в течение всего времени выполнения программы. В функциональном программировании переменная обозначает только имя некоторой структуры, имена символов, переменных, списков, функций и других объектов не закреплены предварительно за какими-либо типами данных. В ФП одна и та же переменная в различные моменты времени может представлять различные объекты.

Особенности функционального программирования (продолжение 2).

6.В языках функционального программирования программа и обрабатываемые ею данные имеют единую списочную форму представления.

7.Функциональное программирование предполагает наличие функционалов – функций, аргументы и результаты которых могут быть функциями.

Всякий язык функционального программирования предполагает наличие ядра, называемого строго функциональным языком.

Требования к строго функциональному языку.

1.Всякая функция должна однозначно определять результат по любому набору аргументов.

2.Отсутствует оператор присваивания.

3.Переменная обозначает только имя структуры.

4.В языке присутствуют функционалы.

Основные преимущества языков ФП.

-Краткость программы.

-Функциональные программы поддаются формальному анализу легче своих аналогов на алгоритмических языках за счет использования математической функции в качестве основной конструкции.

-Возможность реализации на ЭВМ с параллельной архитектурой.

Близость к естественному языку.

Лисп – интерпретируемый бестиповой язык символьной обработки. Сходство с машинным языком : единая форма представления данных и программ. Близость к ЕЯ обеспечивается за счет декларативности ЛИСП – программ. Наряду с типичными для “декларативных” языков средств, Лисп допускает применение некоторых структур данных АЯ, не допустимых в ЛП, в частности, массивов, а также имеет встроенные “ассемблерные” функции.

Применение языков функционального программирования.

•Системы автоматизированного проектирования.

•Программирование игр.

•Математическая лингвистика.

•Реализация ленивых вычислений.

Ленивые вычисления.

Ленивые вычисления (англ. lazy evaluation) - концепция в некоторых языках программирования, согласно которой вычисления следует откладывать до тех пор, пока не понадобится их результат (вызов по необходимости в противоположность традиционному вызову по значению). При этом функции можно передавать как само значение аргумента (как в традиционных т.н. энергичных вычислениях), так и указатель на него, если достаточно указателя, то значение можно не вычислять. Примеры : добавление узла в дерево, конкатенация списков. Ключевым требованием при этом является нестрогость функции по отношению к данному аргументу.

Ленивые вычисления позволяют гарантировать, что вычисляться будут только те данные, которые требуются для получения конечного результата, и тем самым позволяют программисту описывать только зависимости функций друг от друга и не следить за тем, чтобы не осуществлялось “лишних” вычислений.

Ленивые вычисления естественным образом легли на функциональную парадигму программирования.

Применение ленивых вычислений.

–Оптимизация кода. Смысл : аргументы, значения которых не требуются, не будут вычисляться.

–Бесконечные и циклические структуры данных - при описании функций, которые порождают произвольное количество элементов структуры данных.

–Для удобства формы записи некоторых алгоритмов (числа Фибоначчи, приближенный метод Ньютона и т.п.).

Реализация ленивых вычислений в некоторых известных языках.

Ряд современных функциональных языков имеют встроенные средства реализации ленивых вычислений.

–Haskell. Не имеет оператора присвоения, а только операцию определения функциональной зависимости. Имеет ленивые списки, позволяющие программистам оперировать бесконечными последовательностями. Позволяет создавать неленивые типы данных.

–Mathematica (язык программирования) - допускает как ленивые (оператор определения “:=“) так и неленивые (оператор присвоения “=“) вычисления.

–Python - имеет возможность создавать ленивые определения с помощью ключевого слова “lambda”.

Функциональные языки программирования, реализующие ленивые вычисления, зарекомендовали себя как инструменты, удобные для прототипирования и быстрой разработки программного обеспечения, а также для проектирования электронно-вычислительных устройств.

Функции в функциональном программировании.

Определение. В математическом понимании функция является правилом сопоставления каждому элементу области определения функции в точности одного элемента из области значений (рис. 3).

Требование к описанию функций.

В функциональном программировании функция должна быть формально определена для всей области определения !

Пример.

Описание функций в функциональном программировании.

В функциональных языках функции описываются следующим образом :

Sign(x)=если x>0 то плюс

иначе если x=0 то нуль

иначе минус.

Пример.

Найти максимальное из 6-ти значений : a,b,c,d,e,f.

Функция “максимум из двух” : Функция “наибольшее из трех” : max(x,y)=если x>y то x иначе y наиб(x,y,z)=max(x,max(y,z))

Три варианта описания решения задачи “максимум из шести” :

1). max(наиб(a,b,c), наиб(d,e,f))

2). max(max(a, max(b,c)), max(d, max(e,f))

3). наиб(max(a,b), max(c,d), max(e,f))

Основная литература по курсу.

1.Хювенен Э., Сеппянен Й. Мир Лиспа. В 2-х т. Пер. с финск. – М.: Мир, 1990.

2.Хендерсон П. Функциональное программирование. Применение и реализация : Пер. с англ. – М.: Мир, 1983.

3.Филд А., Харрисон П. Функциональное программирование : Пер. с англ. – М.: Мир, 1993.

4.Морозов А.Н. Функциональное программирование : курс лекций. // http://www.marstu.mari.ru:8101/mmlab/home/lisp/title.htm

5.Информатика и программирование шаг за шагом : Язык программирования LISP. // http://it.kgsu.ru/Lisp/oglav.html

6.АВТОЛИСП – язык графического программирования в системе

AutoCAD. // http://kappasoft.narod.ru/info/acad/lisp/a_lisp.htm

7.XLISP Home Page // http://www.mv.com/ipusers/xlisper/

Дополнительная литература.

8.Мауэр У. Введение в программирование на языке ЛИСП : Пер. с

англ. – М.: Мир, 1976.

9.Лавров С.С., Силагадзе Г.С. Автоматическая обработка данных. Язык Лисп и его реализация. – М.: Наука, 1978.

10.Вячеслав А. Функциональное программирование для всех :

пер. Линкер Н. // RSDN Magazine. – 2006. - №2. -
http://www.rsdn.ru/article/funcprog/fp.xml#ECNAC

Литература по тематике самостоятельной работы студентов.

11.Хомский Н. Аспекты теории синтаксиса. – МГУ, 1972.

12.Гладкий А.В. Формальные грамматики и языки. – М.: Наука, 1973.

13.Виноград Т. Программа, понимающая естественный язык. –

М.: Мир, 1976.

14.Льюис Ф., Розенкранц Д., Стирнз Р. Теоретические основы проектирования компиляторов. – М.: Мир, 1979.

15.Бек Леланд Л. Введение в системное программирование. –

М.: Мир, 1988.

16.Белоногов Г.Г., Новоселов А.П. Автоматизация процессов накопления, поиска и обобщения информации. – М.: Наука, 1979.

17.Белоногов Г.Г. и Богатырев В.И. Автоматизированные информационные системы. Под ред. К.В. Тараканова. – М.: Сов.

радио, 1973.

Лекция 1 (часть2). Базовые функции языка Лисп.

Основные определения.

Определение 1. Символом называется отличное от числа имя, состоящее из букв, цифр и специальных знаков, которое обозначает некоторый объект реального мира.

Определение 2. Атомарными объектами или просто атомами в функциональных языках называются простейшие объекты, из которых строятся остальные структуры.

Определение 3. К константам в функциональном программировании относят логические значения T и NIL. Иногда к множеству констант относят также числа. В этом случае говорят о способности символа обозначать объекты. Если символ может обозначать более одного объекта, то его следует рассматривать как переменную.

Определение 4. S-выражениями называют символьные выражения, которые образуют области определений и области значений функций Лиспа.

Виды S-выражений.

Списки как средство представления знаний.

Атомы и списки являются основными типами данных в Лиспе.

Определение. Списком в Лиспе называется упорядоченная последовательность S-выражений, заключенная в круглые скобки. Элементы списка отделяются друг от друга пробелами.

Пример :

(a b c) – список из трех элементов, ((a b c)) – список из одного элемента, ((a b) c) – список из двух элементов.

Одной из отличительных особенностей Лиспа является единая- форма представления данных и программного кода (функций). Для обозначения списка, используемого как данные, и блокировки вычислений в Лиспе определена функция QUOTE :

(QUOTE (a b c)) ⇔ ‘(a b c)

(a b c) – вызов функции с именем a и фактическими параметрами b,c.

Встроенные функции Лиспа.

- Арифметические (+, -, *, /)
- Логические (AND, OR, NOT)

Деление производится с точностью до седьмого знака.

Пример :

$ (/ 1 3) Результат :

0.3333333

Поскольку Лисп является декларативным языком, то в нем используется предикативная форма записи. Для сравнения :

Паскаль.	Лисп.
a + b	(+ a b)
(a + b)/(b - c)	(/ (+ a b)(- b c))
a and b and c	(and a b c)

Базовые функции обработки списков.

В Лиспе для построения, разбора и анализа списков существуют простые базовые функции, к которым сводятся символьные вычисления и которые в нотации данного языка следует рассматриваются как система аксиом (алгебра обработки списков). К базисным функциям обработки символьных выражений в muLISP'е относят :

- Функции-селекторы CAR и CDR;
- Функцию-конструктор CONS;
- Предикаты ATOM, EQ, EQUAL.

Селекторы

Определение. Селектором называется функция, осуществляющая выборку элемента объекта данных.

Функция CAR возвращает в качестве значения голову списка :

Функция CDR возвращает в качестве значения хвост списка.

Примеры :

Список	CAR	CDR
(a b c)	a	(b c)
((a)(b c))	(a)	((b c))
(a)	a	nil
()	nil	nil

Конструктор.

Конструктором называется функция, осуществляющая построение объекта данных. Функция CONS строит новый список из переданных ей в качестве аргументов произвольного s-выражения и списка. При этом первый аргумент включается в список-результат в качестве головы, второй – в качестве хвоста.

Примеры

S-выражение	список-аргумент	результат
(a)	(b c)	((a) b c)
(a)	NIL	((a))
NIL	(b c)	(NIL b c)
a	NIL	(a)
a	b	(a.b)

Связь между конструкторами и селекторами.

Функции CAR и CDR являются обратными для конструктора CONS.

Пример.

(CONS (CAR ‘(a b c))(CDR ‘(a b c))) дает исходный список ‘(a b c)

Композиция селекторов.

Путем комбинации селекторов CAR и CDR можно выделять произвольный элемент списка. В Лиспе допускается сокращенная запись композиции нескольких селекторов (в muLISPe – не более четырех) в виде одного вызова функции.

Пример.

Дан список списков :

‘((a b)(1 2)(3 4 5 6)(7 8 9) 10)

Выделить второй элемент третьего подсписка. Решение.

Выделяем третий подсписок :

(CAR (CDR (CDR ‘((a b)(1 2)(3 4 5 6)(7 8 9) 10))))

Выделяем второй элемент третьего подсписка :

(CAR (CDR (CAR (CDR (CDR ‘((a b)(1 2)(3 4 5 6)(7 8 9) 10)))))) Сокращенная запись : (CADAR (CDDR ‘((a b)(1 2)(3 4 5 6)(7 8 9) 10)))

Конструирование списков в Лиспе.

Помимо примитивных функций CAR, CDR и CONS списочного ассемблера (уровня ячеек памяти виртуальной Лисп-машины), для построения списков в Лиспе существуют функции более высокого уровня.

Функция list создает список из произвольных элементов. Вызов list эквивалентен суперпозиции вызовов CONS, причем вторым аргументом последнего вызова является nil, который служит основой для наращивания списка.

Пример :

(list ‘a ‘b ‘c) эквивалентно

(cons ‘a (cons ‘b (cons ‘c nil))) в muLISP’e, либо (cons 'a (cons 'b (cons 'c '()))) в newLISP-tk

Выделение элемента списка.

Для реализуемой с помощью селекторов и их суперпозиции выборки элементов списков в Коммон Лиспе и muLISP’е существуют функции выделения заданного элемента : FIRST, SECOND, THIRD, FOURTH, … и более общая функция NTH,

выделяющая n-й элемент списка : (nth n список). Следует помнить, что функция nth отсчет элементов списка производит с нуля. В newLISP-tk реализованы first и nth.

Пример :

(nth 5 ‘(1 2 3 4 5)) дает nil, (nth 4 ‘(1 2 3 4 5)) дает 5.

Последний элемент списка можно выделить с помощью функции LAST. При этом вызов (last список) в muLISP’е дает последний непустой хвост списка список.

Пример :

(last ‘(1 2 3)) дает (3)

(car (last ‘(1 2 3))) возвращает последний элемент списка, то есть 3.

Предикатные функции в LISP'е.

Определение. Предикатами в функциональном программировании называются функции, которые проверяют выполнение некоторого условия и возвращают логическое значение T или NIL.

ATOM, EQ и EQUAL являются базовыми предикатами Лиспа. С их помощью и используя другие базовые функции, можно задавать более сложные предикаты, которые будут проверять наличие более сложных свойств.

Предикат ATOM проверяет, является ли аргумент атомом.

(atom s-выражение) возвращает T, если s-выражение является атомом, NIL – в противном случае.

Предикат EQ проверяет тождественность двух атомарных s-выражений. (eq 'a 'b) дает NIL, (eq 'a 'a) дает T, (eq a a) дает T, (eq 1 1) дает T,

но (eq '(1 2) '(1 2)) дает NIL.

Предикат EQUAL проверяет тождественность произвольных s- выражений.

Примеры : (equal '(1 2) '(1 2)) дает T, (equal '(1 2) '(2 1)) дает NIL.

Следует отметить, что (EQUAL список NIL)⇔(NULL список)