Введение в функциональное программирование.

Лекция 1.

Специальности : 230105, 010501

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 1961 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.

Рис. 1. Структура функциональной программы.

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.

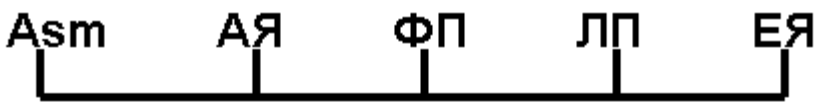
- Краткость программы.

- 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.



Asm – ассемблер
АЯ – алгоритмические (процедурные) языки
ФП – языки функционального программирования
ЛП – языки логического программирования
ЕЯ – Естественные Языки (русский, английский и др.)

Рис.2. Близость к естественному языку.

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).**

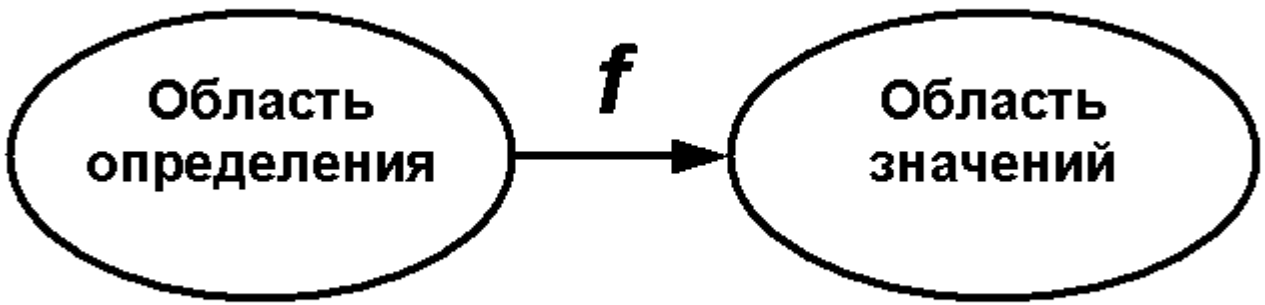


Рис.3.

Требование к описанию функций.

В функциональном программировании *<u>функция должна быть формально определена для всей области определения !</u>*

Пример.

$$sign(x) = \begin{cases} "+", если\ x > 0 \\ "0", если\ x = 0 \\ "-", если\ x < 0 \end{cases}$$

Рис.4.

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.