В. Э. Вольфенгаген,
О. В. Воскресенская

ОСНОВНЫЕ КОНСТРУКЦИИ ЯЗЫКА ЛИСП

Лабораторная работа

Москва 1983

МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ СССР

МОСКОВСКИЙ ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ ИНЖЕНЕРНО-ФИЗИЧЕСКИЙ ИНСТИТУТ

В. Э. Вольфенгаген, О. В. Воскресенская

ОСНОВНЫЕ КОНСТРУКЦИИ ЯЗЫКА ЛИСП

Лабораторная работа

Утверждена
редсоветом института
в качестве учебного пособия

Москва 1983

УДК 681.3

Вольфенгаген В.Э., Воскресенская О.В. Основные конструкции языка ЛИСП. Лабораторная работа. – М.: Изд. МИФИ, 1983, 16с.

В учебном пособии описан синтаксис языка ЛИСП, основные операторы языка, приведены примеры JCL-карт для запуска программ.

Учебное пособие предназначено для студентов, аспирантов и инженеров, специализирующихся в области искусственного интеллекта, баз данных и языках программирования, а также может быть использовано слушателями факультета повышения квалификации.

Рецензенты:
В.Н.Жданов, В.Т.Олейников

Цель: изучение языка программирования ЛИСП, расширенного средствами работы с внешней памятью, и пакетов JCL-карт для работы с интерпретатором с языка ЛИСП.

ВВЕДЕНИЕ

ЛИСП является языком высокого уровня, предназначенным для решения разнообразных задач искусственного интеллекта. ЛИСП – не только самый раепространенный язык обработки списков, но и служит основой новых мощных языков высокого уровня типа *QLISP*, *MLISP*, *INTERLISP* и т.д.

ОБЩЕЕ ОПИСАНИЕ

Описывается интерпретатор с языка ЛИСП в системе ОС ЕС ЭВМ. Основной операцией интерпретатора является последовательность действий:

1) чтение очередного выражения;
2) вычисление значения выражения;
3) повторение с шага 1.

ЛИСП-система состоит из множества встроенных функций, реализованных на языке АССЕМБЛЕРА, или определенных в языке ЛИСП. Значения выражений вычисляются в режиме интерпретации. Отлаженные функции пользователя помещаются в личную библиотеку, из которой осуществляется автоматический вызов функций и программ в процессе вычислений.

Общая организация памяти в ЛИСП-системе дана ниже:

1) системные программы (интерпретатор и встроенные функции);
2) *FWS* – область свободной памяти (для размещения информационных ячеек атомов и списков);
3) *FS* – область полных слов (для размещения чисел и внешних представлений для атомов);

4) *STACK* (*LIFO*-стек) – адреса возвратов для рекурсии и адреса информационных ячеек, соответствующие именам функций);

5) *ALIST* стек (таблица имен переменных и их значений и промежуточные результаты вычислений);

6) внешняя память для размещения пользовательских ассемблерных программ.

Информационные ячейки строятся для списков и атомов и состоят из следующей информации:

1 байт	тип атома, идентификатор списка, бит для мусорщика
2–4 байты	ссылка на значение
5 байт	длина атома и бит для трассировки
6–8 байты	ссылка в область полных слов

Информационные ячейки размещают в области свободной памяти *FWS*, адресуются из области *STACK* и указывают на область полных слов *FS*, в которой находятся числа и внешние представления атомов.

ОПИСАНИЕ ЯЗЫКА

Для обозначения различных объектов в языке ЛИСП применяются атомы. Атомом называется произвольная последовательность букв, цифр и литер, заключенная между двумя ограничителями языка ЛИСП (ограничителями являются левая и правая скобки, точка и пробел). Алфавит языка включает в себя буквы русского и латинского алфавита, цифры, ограничители и литеры:

\+ – x / , : ; = ! % .

Списком называется конечная последовательность элементов, заключенная в круглые скобки. Элементом списка может быть атом или список. Элементы списка разделяются пробелом. Скобки в ЛИСПе – важные синтаксические единицы. Отсутствие парной скобки или неправильная постановка скобок является грубой ошибкой и делает программу бессмысленной.

Выражением является список, атом или точечная пара. Выражения могут иметь значения, которые в свою очередь являются выражениями. Выражения, имеющие значения – это константы, переменные, наименование функций и обращения к функциям. С каждой функцией связывается набор ее переменных, значения которых определяются в момент обращения. Значения констант остаются неизменными до переопределения. Значением имени функции является определяющее выражение этой функции.

Обращение к функции имеет вид списка, в котором первый элемент есть имя функции, следующие элементы списка – аргументы функции, являющиеся выражениями ЛИСПа.

Программа на языке ЛИСП состоит из последовательности обращений к функциям, которые выполняются в заданном порядке. Функция может быть вычислена, если вычислены значения ее аргументов и если она была определена с помощью функций *SEXPR* или *SFEXPR*.

ОПИСАНИЕ ВСТРОЕННЫХ ФУНКЦИЙ

Для формального описания функций языка введем следующую нотацию:

A – атом, *F* – атом, имя функции, *L* – список, *N* – число, *NIL* – пустой список, *ANY* – любое *S* – выражение, *Bool* – булевское выражение, *u*, *v*, ... – переменные, *SEQ* – последовательность *S* – выражений, имя: тип – имя имеет тип (например, атом), имя ( ): тип – результат имеет тип; *val*: значение, вырабатываемое функцией.

Далее дается определение основных функций ЛИСПа.

QUOTE(u:ANY):ANY, val:u – функция, подавляющее вычисление аргумента, одноместная, значением функции является сам аргумент

CAR(u:L):ANY, *val*: голова списка

CDR(u:L):L или *NIL*, *val*: хвост списка

CONS(u:ANY, v:L):L, *val*: в начало *L* вставляется *u*

APPEND(u:L, v:L):L; *val*: сливаются два списка в один

JOIN(u:L, v:L):L, *val*: два списка сливаются в один, но элементы первого списка помещаются в обратном порядке

NULL(u:L):Bool, *val*: *T* (истина), если *L* пуст, *NIL* – в противном случае

ATOM(u:ANY):Bool, *val*: T, если *u:A*, *NIL*, если *u:L*

EQ(u:ANY, v:A): Bool, причем любая одна из переменных должна быть атомом, val: T, если $u = v$, иначе NIL

EQUAL(u:ANY, v:ANY):Bool, val:T, если $u = v$, иначе NIL

EVAL(u:ANY): ANY, val: вычисляется значение от значения u

SETQ (u:A, v:ANY):ANY, val: атому u присваивается значение v

SET(u:A, v:ANY): ANY, val: значению атома u присваивается значение v

CSETQ(u:A, v:ANY):ANY, val: константе u присваивается значение v

CSET(u:A, v:ANY):ANY, val: значению константы u присваивается значение v

MSETI(u:A, v:N, w:ANY), val: в массив с именем u на v -позицию заносится данное w

EV(u:A, v:N):ANY, val: из массива u выбирается данное по индексу v и выдается в качестве результата

PUSH(u:ANY, v:L):L, val: в начало списка v вставляется значение выражения u значение v становится равным (CONS и v)

POPUP(u:A, v:L): L, val: переменной u присваивается голова списка v, список v становится равным (CDR v)

POP(u:L):L, val: список u становится равным (CDR u)

JOIN(u:L, v: NIL):L, val: список u в обратном порядке

COND(u_1:L, ... u_n: L), u_i - $(w_1$:ANY, w_2 : ANY): ANY, val: первое выполнившееся условие w_1 из некоторого u_j инициирует выполнение соответствующего w_2, значением функции будет результат выполнения последнего оператора из w_2 и NIL, если ни

одно из условий w_i в u_j, $j = \overline{1,n}$ не выполнено, w_i может быть атомом

SEXPR(u:A, v:ANY):u, val: заводится константа с именем u, значением которой является определяющее выражение v. При обращении к u аргументы будут вычисляться

SFEXPR(u:A, v:ANY):u, val: аналогично SEXPR, но при обращении аргументы вычисляться не будут

LAMBDA(u:L, w:SEQ), val: описание тела функции, u список связанных переменных, w - тело функции

PROG(u:L, w: SEQ), val: описание вычислений, u - список программных переменных, значение которых устанавливается NIL при входе в PROG

RETURN(u:ANY), val: возврат из функции со значением u в точку, следующую за обращением к функции, если голова u : F - обращение к другой функции

GO(u:A), val: переход по метке заданной внутри PROG

READ, val: следующее S -выражение

PRINT(u: ANY): ANY, val: будет отпечатано значение u

BEFORE(u: A, v:A): Bool, val: Т, если u лексикографически меньше, чем v иначе NIL

OR(u_1:ANY, ... u_n: ANY)Bool, val: Т, если хотя бы одно u_i отлично от NIL

AND(u_1:ANY, ... u_n : ANY): Bool, val: T, если все u_i отличны от NIL

NOT(u:ANY): Bool, val: Т, если u равно NIL, иначе NIL

MEMBER(u:ANY, v:ANY):Bool, val: T, если u включено в v, иначе *NIL*

LIST(u_1:ANY,...u_N:ANY):L, val: строится список из значений элементов $u_1, \ldots u_N$

АРИФМЕТИЧЕСКИЕ ФУНКЦИИ

Числа представляются в формате целых и с плавающей точкой десятичных чисел и шестнадцатиричных чисел. В памяти ЭВМ все числа представлены как десятичные с плавающей точкой и занимают по 8 байт. Формат представления следующий:

[±]<целое без – знака> [. <целое без–знака>] [E [±] <целое без – знака>]

Формат для 16–ичных чисел: X'<набор 16–ичных цифр>

Максимально можно представить числа до 10^{75} с точностью 56 двоичных разрядов:

ADD1(u:N):N, val: число, увеличенное на 1

SUB1(u:N):N, val: число на 1 меньшее

TIMES(u_1:N,...u_n:N):N, val: произведение чисел

PLUS(u_1:N,...u_n:N):N, val: сумма чисел

DIFFERENCE(u_1:N, u_2:N):N, val: разность чисел

QUOTIENT(u_1:N, u_2:N):N, val: целая часть от деления u_1 на u_2

MINUS(u_1:N):N, val: число с обратным знаком

MAX(u_1:N,... u_n:N):N, val: максимальное число

MIN(u_1:N,... u_n:N):N, val: минимальное число

GREATERP(u_1:N, u_2:N):Bool, val: Т, если u_2 меньше u_1, иначе *NIL*

LESSP(u_1:N, u_2:N):Bool, val: Т, если u_2 больше u_1

ZEROP(u:N):Bool, val: Т, если $u = 0$

Более подробное описание языка ЛИСП приведено в работах [1, 2].

ОРГАНИЗАЦИЯ ОБМЕНА ИНФОРМАЦИИ С ВНЕШНЕЙ ПАМЯТЬЮ

В языке ЛИСП данные представляются в виде информационных ячеек, которые существуют для всех атомов, чисел и списков, и собственных представлений для чисел и атомов. Информационные ячейки занимают 8 байт. На внешней памяти программы и данные представлены в символьном виде – на каждый символ тратится 1 байт. Математической моделью внешней памяти является так называемая виртуальная лента – ось положительных целых чисел от 1 до N, где N – размер выделенной области для реализации виртуальной ленты в байтах. Чтобы поместить информацию на диск, необходимо написать команду (*LAY* x K), где x – это имя, значение которого перемещается на дисковую память, K – адрес первого байта, с которого будет помещаться эта информация. Результатом функции будет адрес байта, следующего за областью, в которой поместился x. Считать информацию с диска можно с помощью функции (*TAKE* K), где K – числовой атом, адрес первого байта, с которого будет осуществлено считывание. Если по байту K помещалась левая скобка, то будет считано выражение (любой длины) по принципу спаривания скобок. Если байт K пустой, то будет считана информация первого непустого байта виртуальной ленты по принципу парных скобок, либо будет считан атом. Если байт K приходится на середину атома, то будет считана оставшаяся часть атома. Значением функции является считанное выражение:

LAY(x:A, v:N):N, val: значение x записывается с v байта на внешнюю память

TAKE(v:N):ANY, val: значение по адресу v считывается в оперативную память

Более сложные функции для работы с внешней памятью описаны в [2].

Примеры. Рассмотрим приемы программирования. Считаем выражение и присвоим его в качестве значения константе A:

```
(CSETQ A (READ))
(A1 B1 C1)
```

Возьмем начало и "хвост" списка A , и присвоим эти значения переменным x и Y соответственно.

```
(SETQ x (CAR A))(SETQ Y(CDR A))
```

Вставим в начало списка Y атом G1 и затем соединим полученный список со списком (1 2 3).

```
(SETQ Y (CONS (QUOTE G1) Y))
```

либо

```
(PUSH (QUOTE G1) Y)
(SETQ Y (APPEND Y (QUOTE (1 2 3))))
```

либо

```
(SETQ Y(JOIN(JOIN Y NIL)(QUOTE(1 2 3))))
```

Узнаем значение Y :

```
(EVAL Y)
```

даст в результате

```
(G1 B1 C1 1 2 3)
```

Рассмотрим Y как массив и поместим по индексу 5 значение 12.

```
(MSETI Y 5 12)
```

Проверим, если по индексу 2 в массиве Y находится атом, заменим его списком, состоящим из этого же элемента:

```
(SETQ z(EV Y 2))
(COND((ATOM z)(MSETI Y 2(LIST z))))
```

Выполним (EVAL Y) , получим

```
(G1 (B1) C1 1 12 3)
```

Проверим, содержится ли элемент D в Y :

```
(MEMBER(QUOTE D)Y)
```

получим NIL;

```
(MEMBER(QUOTE(B1))Y)
```

даст T.

Сделаем список из элементов B , C , D и присвоим его переменной u в качестве значения:


```
(SETQ u (LIST(QUOTE B)(QUOTE C)(QUOTE D)))
```

Запомним обращение к функции CAR, применяемой к списку u.

```
(SETQ v (LIST (QUOTE CAR)(QUOTE u)))
```

(EVAL v) дает в качестве результата B.

Напечатаем значение v -(PRINT v)

напечатает (CAR u).

Проверим, лексикографически больше первый или третий элемент u :

```
(BEFORE(CAR u)(CADDR u))
```

даст в качестве результата T .

Напишем программу попарного умножения чисел из двух списков A и B :

```
(SETQ A (QUOTE (2 4 6)))
(SETQ B (QUOTE (3 5 7)))

(SEXPR УМНОЖ (LAMBDA(x y)(PROG(CO x1 Y1 z)
        (SETQ CO y)
M2      (POPUP x1 x)
M1      (POPUP y1 y)
        (PUSH (TIMES x1 y1) z)
        (COND (y (GO M1))
              (x (SETQ y CO)(GO M2)))
        (RETURN (PRINT (JOIN z NIL)))
        )))
```

Обратимся к этой функции

(УМНОЖ A B)

получим в результате

(6 10 14 12 20 28 18 30 42)

В функции УМНОЖ запрограммирован следующий алгоритм: просмотр элементов списков выполняется в двух циклах, для сохранения второго списка используется переменная *GO*. Результат формируется в виде списка произведений чисел в обратном порядке.

Возврат из *PROG* выполняется с помощью функции *RETURN* со значением *Z* .

Переменные *X1* и *Y1* содержат очередные элементы списков *X* и *Y* .

Обратить внимание на запись проверки, не пуст ли список *(COND (Y (GO M1)))*. Эта проверка лаконичнее, чем *(COND ((NULL Y) ....))*

ПАКЕТ ЗАПУСКА ЛИСП-СИСТЕМЫ (ИНТЕРПРЕТАТОР ПАНТЕЛЕЕВА)

Интерпретаор с языка ЛИСП находится на Вашем томе с меткой *LISPOK* . Пусть Ваши отлаженные функции хранятся в библиотеке БИБЛ в модулях, имена которых совпадают с именами функций (по одной функции в каждом модуле). Пусть отлаживаемая часть программы находится в модуле МОД библиотеки БИБЛ 2. С внешней памятью Вы работать не собираетесь. Тогда Вам нужен следующий пакет *JCL* -карт:

```
1 //A      JOB  REGION=200K, TIME=2
2 //        EXEC PGM=LISP
3 //STEPLIB DD UNIT=SYSDA,VOL=SER=LISPOK,DSN=LLISP,DISP=SHR
4 //DDLISPO DD SYSOUT=A
5 //LISPLIB DD UNIT=SYSDA,VOL=SER=LISPOK,DSN=BUBA,DISP=SHR
6 //DDLISPI DD UNIT=SYSDA,VOL=SER=LISPOK,DSN=BUBA2(MOD),DISP=SHR
```

Если у Вас нет библиотеки отлаженных функций и не нужен механизм автоматического вызова, опустите 5-ю карту.

При необходимости работы с базой данных на диске необходимо:

выделить физически область памяти на диске;

загрузить в эту область некоторые системные программы.

Для выполнения первого этапа необходимо следующее задание:

```
//OLYAKA JOB MSGLEVEL=(1,1)
// EXEC PGM=IEFBR14
//A DD UNIT=SYSDA,VOL=SER=LISPOK,DSN=VOLYA,DISP=(MOD,DELETE)
// EXEC PGM=BIPBAM
//STEPLIB DD DSN=ASNMASL,
//  UNIT=SYSDA,
//  VOL=SER=LISPOK,
//  DISP=SHR
//SYSPRINT DD SYSOUT=A
//GO.BIM DD DUMMY
//BAM DD  UNIT=SYSDA,DSN=VOLYA,
//    VOL=SER=LISPOK,
//    DCB=(DSORG=DA,BLKSIZE=3504,RECFM=F),
//    SPACE=(CYL,(8,1)),DISP=(NEW,KEEP)
//GO.SYSIN DD *
NP=2;
NNNN=0;
MMMM=300;
/*
//
```

VOLYA – имя Вашей виртуальной ленты. 2-е и 3-е предложения уничтожают набор данных с этим именем, если он существует. Далее идут команды создания набора.

8-е предложение описывает физическую организацию виртуальной ленты. В частности, создается виртуальная лента объемом 8 цилиндров, разбитая на блоки длиной 3504, количество которых равно *3ФФ* (параметр MMMM, 14-я карта). Число цилиндров должно содержать заданное параметром MMMM количество блоков.

```
//OLYARB JOB MSGLEVEL=(1,1),REGION=100K
// EXEC PGM=LISP,PARM=',,,,,,33700'
//STEPLIB DD UNIT=SYSDA,VOL=SER=LISPOK,DISP=SHR,DSN=ASNMASL
//        DD UNIT=SYSDA,VOL=SER=LISPOK,DISP=SHR,DSN=LLISP
//DADP DD UNIT=SYSDA,VOL=SER=LISPOK,DISP=OLD,DSN=VOLYA
//DDLISPO DD SYSOUT=A
//DDLISPI DD UNIT=SYSDA,VOL=SER=LISPOK,DISP=SHR,DSN=MASS(GENERAL)
//        DD UNIT=SYSDA,VOL=SER=LISPOK,DISP=SHR,DSN=MASS(KILPH)
//        DD UNIT=SYSDA,VOL=SER=LISPOK,DISP=SHR,DSN=MASS(INQUIRPH)
//        DD UNIT=SYSDA,VOL=SER=LISPOK,DISP=SHR,DSN=MASS(INSEBH)
//        DD UNIT=SYSDA,VOL=SER=LISPOK,DISP=SHR,DSN=MASS(CHANPH)
//        DD UNIT=SYSDA,VOL=SER=LISPOK,DISP=SHR,DSN=MASS(GPPH)
//        DD UNIT=SYSDA,VOL=SER=LISPOK,DISP=SHR,DSN=MASS(BUILPH)
//
```

Для работы с описанными выше функциями *LAY* и *TAKE* достаточно 7-й карты, остальные 8 – 13-е карты обеспечивают загрузку программных фаз, реализующих более сложные функции работы с базой данных.

Далее приводится пример запуска ЛИСП – системы для работы с виртуальной лентой:

```
//DEJOIN JOB MSGLEVEL=(1,1),REGION=200K,TIME=15
//       EXEC PGM=LISP,PARM='D,,,,,33000'
//STEPLIB DD UNIT=2314,VOL=SER=LISPOK,DSN=ASNMASL,DISP=SHR
//        DD UNIT=2314,VOL=SER=LISPOK,DSN=LLISP,DISP=SHR
//DATA    DD UNIT=2314,VOL=SER=LISPOK,DSN=VOLYA,DISP=OLD
//DDLISPO DD SYSOUT=A,DCB=BLKSIZE=80
//MYOD    DD SYSOUT=*,DCB=(RECFM=F,BLKSIZE=120)
//LISPLIB DD UNIT=2314,VOL=SER=LISPOK,DISP=SHR,DSN=NOLYA
//DDLISPI DD UNIT=2314,VOL=SER=LISPOK,DSN=MASS(START),DISP=SHR
//        DD UNIT=2314,VOL=SER=LISPOK,DSN=MASS(POPPY),DISP=SHR
//        DD UNIT=2314,VOL=SER=LISPOK,DSN=VOLYA(NEW2),DISP=SHR
```

Здесь *LISPOK* означает физическую метку тома, *VOLYA* – имя пользовательской виртуальной ленты.

8-е предложение описывает библиотеку для автоматического вызова функций из библиотеки *NOLYA* (имя модуля в библиотеке должно совпадать с именем определяемой функции, раздел библиотеки содержит ровно одну функцию, имя раздела должно удовлетворять требованиям ОС).

9 – 10-е предложения определяют входные наборы для работы с базой данных. Интерпретатор с языка ЛИСП находится в библиотеке *LLISP* (4-е предложение), процедуры работы с базой данных в библиотеке *ASNMASL* (3-е предложение), вызов интерпретатора осуществляется во 2-м предложении. Виртуальная лента описывается в 5-м предложении. В 11-м предложении описывается исходный модуль с программой пользователя, находящейся в библиотеке. В случае, если исходный модуль находится на перфокартах, последнее предложение заменить на

```
//   DD *
```

Если не требуется работы с базой данных, 3-е, 5-е, 9-е, 10-е предложения опускаются. Может быть опущено 8-е предложение.

СПИСОК ЛИТЕРАТУРЫ

1. Лавров С.С., Силагадзе Г.С. Автоматическая обработка данных. Язык ЛИСП и его реализация. – М.: Наука, 197

2. Булкин М.Э., Габович Ю.Р., Пантелеев А.Г. Методические рекомендации по программированию и эксплуатации интерпретатора для алгоритмического языка ЛИСП в ОС ЕС. – Киев: НИАСС Госстроя УССР, 1981, 89 с.

Вячеслав Эристович Вольфенгаген,
Ольга Владимировна Воскресенская

ОСНОВНЫЕ КОНСТРУКЦИИ ЯЗЫКА ЛИСП

Лабораторная работа

Редактор О.А.Сафронова
Техн. редактор Н.М.Воронцова
Корректор Е.А.Жадан

Л.- 99297 Подписано в печать 22/II - 1983 г.
Формат 60x84 1/16 Объем 1 п.л. Уч-изд.л. 0,7
Тираж 650 экз. Цена 6 коп. Изд. № 013-1
Заказ 2960

Типография МИФИ, Каширское шоссе, 31