Справочник по формулам Maxima, используемых при работе со списками.

Списки – базовые строительные блоки для Maxima. Чтобы создать список необходимо в квадратных скобках записать все его элементы через запятую. Список может состоять из одного элемента: в квадратных скобках указывается один элемент. Список может быть пустым, об этом указывают пустые квадратные скобки.

Пример:

```
-->     a1:[1,6,-3,s,2*t];
(%o1)   [1,6,-3,s,2*t]
 -->    a2:[4];
(%o2)   [4]
 -->    b:[];
(%o3)   []
```

Элементом списка может быть и другой список.

Примечание: нумерация команд может быть другой.

```
-->     d:[1,2,[6,-7],[s,t,q]];
(%o4)   [1,2,[6,-7],[s,t,q]]
 -->    r:[a1,a2,3,8];
(%o5)   [[1,6,-3,s,2*t],[4],3,8]
```

Ссылка на элемент списка. Чтобы вывести на экран один из элементов списка нужно записать имя списка, а затем в квадратных скобках указать номер интересующего элемента.

```
-->     d[1];
(%o6)   1
 -->    r[2];
(%o7)   [4]
 -->    r[1];
(%o8)   [1,6,-3,s,2*t]
```

Пример работы со списками:

```
-->     list1:[1,2,3,x,x+y];
(%o10)  [1,2,3,x,y+x]
 -->    list2:[];
(%o11)  []
```

--> list3:[3];

(%o12) [3]

--> list4:[1,2,[3,4],[5,6,7]];

(%o16) [1,2,[3,4],[5,6,7]]

--> list4[1];

(%o15) list4[1]

--> list[3];

(%o17) 3

--> list4[3][2];

(%o18) 4

Функции для элементарных операций со списками

Функция ***length*** возвращает число элементов списка (при этом элементы списка сами могут быть достаточно сложными конструкциями):

--> **length**(list4);

(%o20) 4

--> **length**(list3);

(%o21) 1

Функция ***copylist(expr)*** возвращает копию списка expr:

--> list1:[1,2,3,x,x+y];

(%o22) [1,2,3,x,y+x]

--> list2:**copylist**(list1);

(%o23) [1,2,3,x,y+x]

Функция ***makelist*** создаёт список, каждый элемент которого генерируется из некоторого выражения.

Возможны два варианта вызова этой функции:

1)makelist(expr,i,i0,i1) – возвращает список, j-й элемент которого равен **ev(expr,i = j)**, при этом индекс j меняется от i0до i1. То есть общий вид функции имеет вид:
makelist(выражение, переменная, начальное значение, конечное значение)
Выражение – это некоторое выражение (функция), в которое подставляются значения переменной.

Переменная – имя аргумента в выражении.

Начальное значение – первое подставляемое значение.

Конечное значение – последнее подставляемое значение.

Пример работы:

--> **makelist**(2*x,x,1,5);

(%o24) [2,4,6,8,10]

2**) makelist(expr,x,list)** – возвращает список, j-й элемент которого равен **ev(expr,x = list[j]),** при этом индекс j меняется от 1 до **length(list).** В данном случае значения переменной берутся из списка list – от первого значения до последнего значения.

--> makelist(concat(x,i),i,1,6);

(%o25) [x1,x2,x3,x4,x5,x6]

--> list:[1,2,3,4,5,6,7];

(%o26) [1,2,3,4,5,6,7]

--> makelist(exp(i),i,list);

(%o27) [e,e^2,e^3,e^4,e^5,e^6,e^7]

Во многом аналогичные действия выполняет функция **create_list(form,x1,list1,...,xn,listn).** Эта функция строит список путём вычисления выражения **form**, зависящего от x1, к каждому элементу списка list1(аналогично **form**, зависящая и от x2, применяется к list2и т.д.).

--> create_list(x^i,i,[1,3,7]);

(%o28) [x,x^3,x^7]

--> create_list([i,j],i,[a,b],j,[e,f,h]);

(%o29) [[a,e],[a,f],[a,h],[[],e],[[],f],[[],h]]

*Функция **append*** позволяет склеивать списки. При вызове **append (list_1, \dots, list_n)** возвращается один список, в котором за элементами list1 следуют элементы list2 и т.д. вплоть до listn.

--> append([1],[2,3],[4,5,6,7]);

(%o30) [1,2,3,4,5,6,7]

Создать новый список, компонуя элементы двух списков поочерёдно в порядке следования, позволяет функция **join(k,m).** Новый список содержит k1, затем m1, затем k2, m2 и т.д.

--> join([1,2,3],[10,20,30]);

(%o31) [1,10,2,20,3,30]

--> join([1,2,3],[10,20,30,40]);

(%o32) [1,10,2,20,3,30]

*Функция **cons(expr,list)*** создаёт новый список, первым элементом которого будет expr, а остальные элементы списка list.

Функция ***endcons(expr,list)*** также создаёт новый список, первые элементы которого – элементы списка list, а последний – новый элемент expr.

```
-->     cons(x,[1,2,3]);

(%o33) [x,1,2,3]

-->     endcons(x,[1,2,3]);

(%o34) [1,2,3,x]
```

Функция ***reverse*** меняет порядок элементов в списке на обратный.

```
-->     list1:[1,2,3,x];

(%o35) [1,2,3,x]

 -->    list2:reverse(list1);

(%o36) [x,3,2,1]
```

Функция ***member(expr1,expr2)*** возвращает true, если expr1 является элементом списка expr2, и false в противном случае.

```
-->     member (8, [8, 8.0, 8b0]);

(%o37) true

 -->    member (8, [8.0, 8b0]);

(%o38) false

 -->    member (b, [[a, b], [b, c]]);

(%o39) false

 -->    member ([b, c], [[a, b], [b, c]]);

(%o40) true
```

Функция ***rest(expr)*** выделяет остаток после удаления первого элемента списка expr. Можно удалить первые n элементов, используя вызов rest(expr,n).

Функция ***last(expr)*** выделяет последний элемент списка expr (аналогично first – первый элемент списка).

```
-->     list1:[1,2,3,4,a,b];

(%o41) [1,2,3,4,a,[]]

 -->    rest(list1);

(%o42) [2,3,4,a,[]]

 -->    rest(%);

(%o43) [3,4,a,[]]

 -->    last(list1);
```

(%o44) []

--> rest(list1,3);

(%o45) [4,a,[]]

*Функция **sum(expr,i,in, ik)*** суммирует значения выражения expr при изменении индекса i от in до ik.

*Функция **product(expr,i,in,ik)*** перемножает значения выражения expr при изменении индекса i от in до ik.

--> product (x + i*(i+1)/2, i, 1, 4);

(%o46) (x+1)*(x+3)*(x+6)*(x+10)

--> sum (x + i*(i+1)/2, i, 1, 4);

(%o47) 4*x+20

--> product (i^2, i, 1, 4);

(%o48) 576

--> sum (i^2, i, 1, 4);

(%o49) 30

Функции, оперирующие с элементами списков

*Функция **map(f,expr1,...,exprn)*** позволяет применить функцию (оператор, символ операции) f к частям выражений expr1, expr2,...,exprn. При использовании со списками применяет f к каждому элементу списка. **Следует обратить** внимание, что f – именно имя функции (без указания переменных, от которых она зависит).

--> map(ratsimp, x/(x^2+x)+(y^2+y)/y);

(%o50) y+1/(x+1)+1

--> map("=",[a,b],[-0.5,3]);

(%o51) [a=-0.5,[]=3]

--> map(exp,[0,1,2,3,4,5]);

(%o52) [1,%e,%e^2, %e^3,%e^4,%e^5]

--> f(x):=x^2;

(%o53) f(x):=x^2

--> map(f,[1,2,3,4,5]);

(%o54) [1,4,9,16,25]

*Функция **apply*** применяет заданную функцию ко всему списку (список становится списком аргументов функции; при вызове (F,[x1,...,xn] вычисляется выражение F(arg1,...,argn)). Следует учитывать, что apply не распознаёт ординарные функции и функции от массива.

```
-->     L : [1, 5, -10.2, 4, 3];
(%o55) [1,5,-10.2,4,3]
 -->    apply(max,L);
(%o56) 5
 -->    apply(min,L);
(%o57) -10.2
```