**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

**Лабораторная работа №4**

**по курсу «Логическое программирование»**

Студент: Архипов М. Д.

Группа: М8О-407Б-18

Преподаватель: Левинская М. А.

Сошников Д.В.

Оценка:

Москва 2021

**1)** **Постановка задачи:** Познакомиться на практике с методами анализа естественно-языковых текстов системах логического программирования, реализовать несложный фрагмент естественного-языкового интерфейса к модельной задаче и протестировать его на ряде примеров.

Реализовать синтаксический анализ арифметических выражений для перевода его в префиксную форму. В выражении допустимы знаки +,-,\*,/.

В качестве системы программирования был выбран SWI Prolog 7.3.22

**2) Алгоритм работы программы:** Пролог-программа осуществляет разбор заданного выражения и преобразует его в префиксную форму при помощи предиката calculate(List,Res).  Предикаты expression и subexpression преобразуют заданное арифметические выражение в следующий вид:

«1\*2+3/4  ->  plus multiply(is\_int(1),is\_int(2)) divide(is\_int(3),is\_int(4))». После преобразования полученный набор букв конвертируется в читабельную префиксную форму вида «+ \* 1 2 / 3 4».

**2) Листинг программы:**

symbol(1,"+").

symbol(1,"-").

symbol(2,"\*").

symbol(2,"/").

term(1,"+",Exp1,Exp2,plus1(Exp1,Exp2)).

term(1,"-",Exp1,Exp2,minus(Exp1,Exp2)).

term(2,"\*",Exp1,Exp2,multiply(Exp1,Exp2)).

term(2,"/",Exp1,Exp2,divide(Exp1,Exp2)).

%Конвертация из стандартной записи в промежуточную

expression(N,List,Other,Exp) :- N<3,

N1 is N+1,

                                expression(N1,List,List1,Exp1),!,

subexpression(N,List1, Other,Exp1,Exp).

expression(3,[Int| Other], Other,is\_int(I)) :- I=Int.

subexpression(N,[S|List], Other,Exp1,Exp) :- symbol(N,S),

N1 is N+1,

                                             expression(N1,List,List1,Exp2),!,

                                             term(N,S,Exp1,Exp2,Term),

                                             subexpression(N,List1,Other,Term,Exp).

subexpression(\_,List,List,Exp,Exp).

%Конвертация из промежуточной записи в префиксную

convert(plus1(Exp1,Exp2),["+"|List]) :- convert(Exp1,L1),

                                        convert(Exp2,L2),

                                        append(L1,L2,List).

convert(minus(Exp1,Exp2),["-"|List]) :- convert(Exp1,L1),

  convert(Exp2,L2),

  append(L1,L2,List).

convert(multiply(Exp1,Exp2),["\*"|List]) :- convert(Exp1,L1),

                                           convert(Exp2,L2),

                                           append(L1,L2,List).

convert(divide(Exp1,Exp2),["/"|List]) :- convert(Exp1,L1),

                                         convert(Exp2,L2),

                                         append(L1,L2,List).

convert(is\_int(I),[Int]) :- I=Int.

%Основной предикат

calculate(List,Res) :- expression(1,List,\_,Term),

  convert(Term,Res).

**4) Протокол тестирования:**

?- calculate([5,"+",2,"\*",3],Res).

Res = ["+", 5, "\*", 2, 3] ;

false.

?- calculate([1,"\*",а,"-",2,"/",b],Res).

Res = ["-", "\*", 1, а, "/", 2, b] ;

false.

**5) Вывод:** в результате выполнения данной лабораторной работы мной было выяснено, что при помощи языка Prolog очень удобно осуществлять разбор арифметических выражений. Программа на прологе получается значительно короче, чем на императивных языках программирования за счет многократного использования рекурсии и достаточно удобной структуры – списка.