**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Логическое программирование»**

Тема: Рекурсия и операторы

Вариант 1

|  |  |  |
| --- | --- | --- |
| Студентка гр. 8382 |  | Кузина А.М. |
| Студентка гр. 8382 |  | Кулачкова М.К. |
| Студент гр. 8382 |  | Мирончик П.Д. |
| Преподаватель |  | Родионов С.В. |

Санкт-Петербург

2022

**Цель работы**

Изучение рекурсии на языке Пролог, освоение принципов решения типовых логических программ.

**Задание**

**Вариант 1**

Реализовать поиск минимального элемента в заданном списке. Записать вызов предиката в естественной операторной форме.

**Порядок выполнения работы**

Реализован предикат min\_elem для нахождения минимального элемента в списке. Программа состоит из трех правил. Первое правило задает поведение для в случае списка из одного элемента – минимальным будет единственный элемент списка. Второе правило делит список на голову и хвост и рекурсивно вызывает предикат min\_elem для хвоста, находя его минимальный элемент. Если минимальный элемент хвоста меньше головы списка, минимальный элемент хвоста будет минимальным элементом всего списка. Третье правило также рекурсивно вызывает предикат min\_elem для хвоста списка, находя его минимальный элемент, и в случае, если голова меньше минимального элемента хвоста, голова будет минимальным элементом всего списка.

min\_elem([Elem], Elem).

min\_elem([Elem | Tail], TailMin) :- min\_elem(Tail, TailMin), Elem >= TailMin.

min\_elem([Elem | Tail], Elem) :- min\_elem(Tail, TailMin), Elem < TailMin.

На рис. 1 схематично изображена последовательность исполнения программы для конкретного примера.

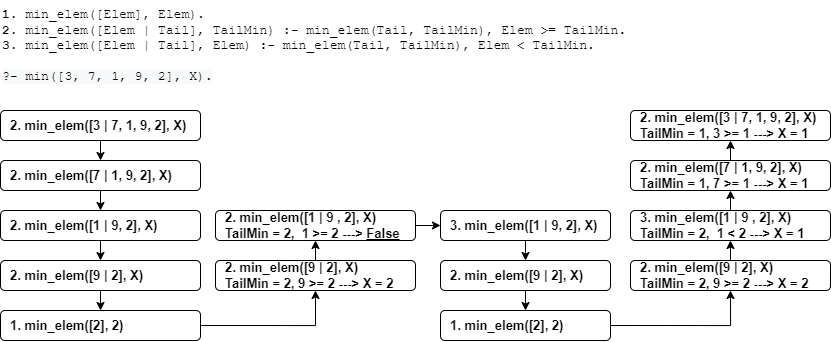


Рисунок – Последовательность исполнения программы

Затем вызов предиката реализуется в виде оператора. Для создания оператора используется встроенный предикат op(Приоритет, Тип, Имя). Приоритет оператора задает порядок выполнения операций в выражениях, содержащих более одного оператора. Тип оператора задает позицию, будет оператор префиксным, постфиксным или инфиксным, и ассоциативность оператора. Название созданного оператора связывается с реализованным предикатом поиска наименьшего элемента в списке.

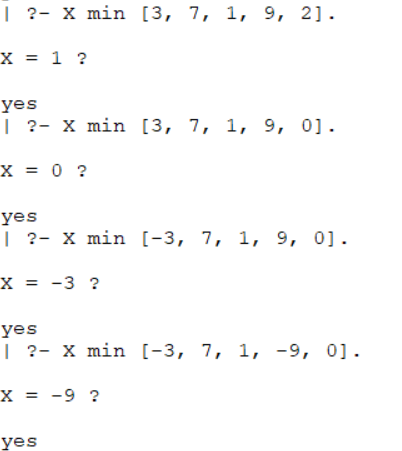
:-op(100, xfx, min).

Elem min X:- min\_elem(X, Elem).

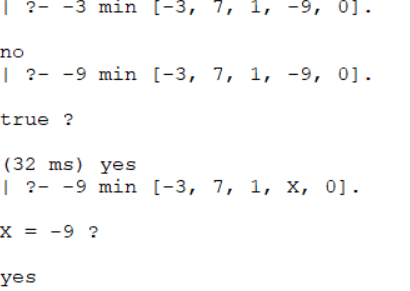
Полный текст программы приведен в приложении А.

**Примеры вызова правил и результаты их выполнения**

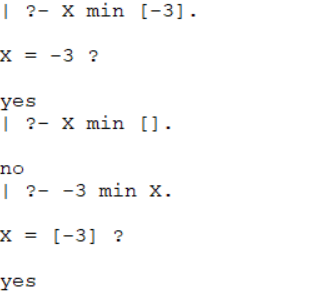
На рис. 2-4 приведены примеры вызова предиката (в операторной форме) с результатами выполнения программы.



Рисунок



Рисунок



Рисунок

**Выводы**

Была реализована программа на языке Пролог, находящая минимальный элемент списка с использованием рекурсии.

Зоны ответственности членов бригады:

* Кузина А.М. – тестирование программы;
* Кулачкова М.К. – составление отчета;
* Мирончик П.Д. – написание программы.

Каждый участник бригады проконтролировал действия других участников и разобрался в проделанной ими работе

В ходе выполнения лабораторной работы возникли следующие трудности:

* Изначально была предпринята попытка написать третье правило без повторного поиска минимального элемента хвоста, однако полученная программа возвращала неправильный ответ на вопросы вида

?- y min [y, x2, …, xN].,

где y, x2, …, xN не являются переменными (явно заданы), а y не является минимальным элементом. Например, на вопрос

?- 3 min [3, 2, 1].

программа отвечала Yes. Повторное нахождение минимального элемента хвоста, реализованное в конечной версии программы, увеличивает время выполнения программы.

Приложение А

исходный код программы

/\*

Задание:

Найти минимальный элемент в заданном списке

?- min\_elem([3,7,1,9,2], X).

X = 1

Бригада 1 группы 8382 - Кузина, Кулачкова, Мирончик

\*/

min\_elem([Elem], Elem).

min\_elem([Elem | Tail], TailMin) :- min\_elem(Tail, TailMin), Elem >= TailMin.

min\_elem([Elem | Tail], Elem) :- min\_elem(Tail, TailMin), Elem < TailMin.

:-op(100, xfx, min).

Elem min X:- min\_elem(X, Elem).