МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Логическое программирование»
Тема: Структуры данных
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 структур данных, освоение таких структур как список и дерево.

Вариант 1

Задание 1. Списки

Определить, является ли заданный список упорядоченным по возрастанию или убыванию.

```
?- is_ordered([1,2,3], X).
X = asc
```

Задание 2. Деревья

Создайте предикат, проверяющий, что дерево является двоичным справочником.

```
?- is_ordered(tr(2,tr(7, nil, nil),tr(3,tr(4, nil, nil), tr(1, nil, nil)))).
No
```

Порядок выполнения работы

Задание 1

Реализуется предикат `is_ordered`, принимающий в качестве параметров список и переменную, в которую будет записан результат выполнения программы. Если предикат вызывается с пустым списком или списком, состоящим из одного элемента, такой список считается упорядоченным как по возрастанию, так и по убыванию – для каждого из вариантов реализованы соответствующие два правила:

```
is_ordered([], asc).
is_ordered([], desc).
is_ordered([_], asc).
is_ordered([_], desc).
```

Для списков, состоящих из двух и более элементов, создается правило, определяющее, что список упорядочен по возрастанию. Список упорядочен по возрастанию, если первый его элемент меньше или равен второму, а список,

полученный из текущего удалением первого элемента, также упорядочен по возрастанию:

```
is_ordered([X,Y|Tail], asc) :- X =< Y, is_ordered([Y|Tail], asc).
```

Аналогично создается правило, определяющее, что список упорядочен по убыванию. Список упорядочен по убыванию, если первый его элемент больше или равен второму, а список, полученный из текущего удалением первого элемента, также упорядочен по убыванию:

```
is_ordered([X,Y|Tail], desc) :- X >= Y, is_ordered([Y|Tail], desc).
```

Также реализовано правило, определяющее, что список не упорядочен. Список не упорядочен, если он не упорядочен по возрастанию и не упорядочен по убыванию:

```
is_ordered(Arr, not_ordered) :- \+is_ordered(Arr, asc),
\+is_ordered(Arr, desc).
```

Задание 2

Двоичный справочник – это такое двоичное дерево, у которого элементы левого поддерева меньше корня родительского дерева, а элементы правого поддерева больше или равен корню родительского дерева. Далее дерево, являющееся двоичным справочником, будет также называться упорядоченным.

Пустое поддерево обозначается атомом `nil`. Поддерево с пустым корнем всегда считается упорядоченным.

```
is_ordered_inner(nil, _, _).
```

Реализованная программа поддерживает неполные деревья, записанные в следующих форматах:

- <Число> <=> tr(<Число>, nil, nil)
- tr(<Число>) <=> tr(<Число>, nil, nil)
- tr(<Число1>, <Число2>) <=> tr(<Число1>, <Число2>, nil)

Для того, чтобы осуществить приведенные выше преобразования, заданы следующие правила:

```
is_ordered_inner(Value, Min, Max) :- integer(Value),
is_ordered_inner(tr(Value), Min, Max).
```

```
is_ordered_inner(tr(Value), Min, Max) :- is_ordered_inner(tr(Value,
nil), Min, Max).
is_ordered_inner(tr(Value, Left), Min, Max) :-
is_ordered_inner(tr(Value, Left, nil), Min, Max).
```

После выполнения этих правил дерево будет иметь вид tr(<Число>, <Дерево>, <Дерево>). Предикат `is_ordered_inner`, помимо самого дерева, принимает в качестве параметров минимальное и максимальное значение, между которыми должны находиться элементы дерева. Для проверки того, что значение зажато между двумя граничными значениями, реализовано следующее правило:

```
is_clamped(Value, Min, Max) :- integer(Value), (Min = nil,!; Value
>= Min), (Max = nil,!; Value < Max).
```

Дерево является упорядоченным, если его корень зажат между граничными значениями, а его левое и правое поддеревья также упорядочены. Причем в качестве верхней границы для левого поддерева и нижней границы для правого поддерева задается корень родительского дерева.

```
is_ordered_inner(tr(Value, Left, Right), Min, Max) :-
is_clamped(Value, Min, Max), is_ordered_inner(Left, Min, Value),
is_ordered_inner(Right, Value, Max).
```

Предполагается, что для того, чтобы проверить, что дерево является двоичным справочником, пользователь будет вызывать предикат `is_ordered`, который выполняется, если выполнен предикат `is_ordered_inner`. Для этого реализовано дополнительное правило:

```
is_ordered(Tree) :- is_ordered_inner(Tree, nil, nil),!.
```

На рис. 1 на примере продемонстрирован порядок выполнения программы.

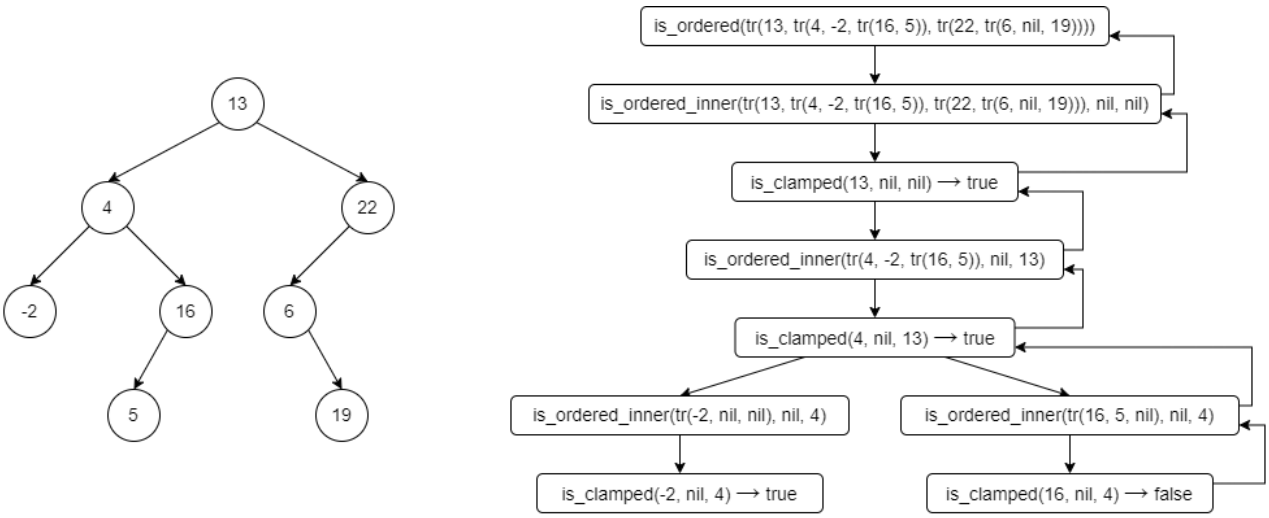


Рисунок 1 – Порядок выполнения программы, определяющей, упорядочено ли дерево

Полный текст программы, содержащей решения заданий 1 и 2, приведен в приложении А.

Примеры вызова правил и результаты их выполнения

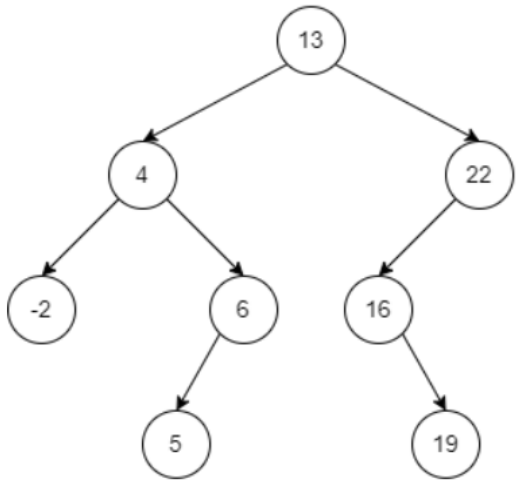
Примеры выполнения предиката, определяющего, в каком порядке упорядочены элементы списка, приведены на рис. 2.

```
| ?- is_ordered([-1, 0, 1, 2, 3], X).

X = asc ?

yes
| ?- is_ordered([1, 0, -1, -2, -3], X).

X = desc ?

yes
| ?- is_ordered([-1, 0, -1, -2, -3], X).

X = not_ordered

yes
| ?- is_ordered([-1], X).

X = asc ? ;

X = desc ? ;

no
| ?- is_ordered([], X).

X = asc ? ;

X = desc ? ;

no
| ?- is_ordered([1, 2, 3], asc).

true ?

yes
| ?- is_ordered([4, 1, 2, 3], asc).

no
```

Рисунок 2 – Пример выполнения программы, определяющей, в каком порядке упорядочен список

Примеры выполнения предиката, определяющего, является ли дерево двоичным справочником, приведены на рис. 3-5. Рис. 5 также демонстрируют возможность различного формата ввода дерева.

- Дерево:

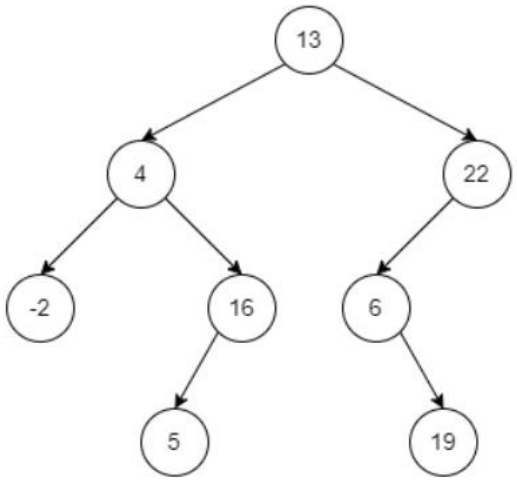


Результат работы программы:

```
| ?- is_ordered(tr(13, tr(4, -2, tr(6, 5)), tr(22, tr(16, nil, 19)))).

yes
```

Рисунок 3

- Дерево:

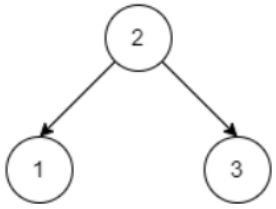


Результат работы программы:

```
| ?- is_ordered(tr(13, tr(4, -2, tr(16, 5)), tr(22, tr(6, nil, 19)))).

no
```

Рисунок 4

- Дерево:



Результат работы программы:

```
| ?- is_ordered(tr(2, 1, 3)).

yes
| ?- is_ordered(tr(2, tr(1), tr(3))).

yes
| ?- is_ordered(tr(2, tr(1, nil, nil), tr(3, nil, nil))).

yes
```

Рисунок 5

Выводы

Была реализована программа на языке Пролог, определяющая, является список упорядоченным по возрастанию или убыванию, а также является ли дерево двоичным справочником.

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 задания на списки трудностей не возникло. В ходе выполнения задания на деревья возникли следующие трудности:

- Для того, чтобы обеспечить возможность вводить дерево в разных форматах, пришлось создать три дополнительных правила, которые несколько загромождают программу;

- Изначально осуществлялась проверка того, что корень левого поддерева меньше корня родительского дерева, а корень правого поддерева больше или равен корню родительского поддерева, однако при тестировании программы обнаружилось, что такой подход обеспечивает упорядоченность только на одном уровне. Поэтому был разработан способ, позволяющий обеспечить то, что все элементы левого поддерева меньше корня, а все элементы правого – больше или равны корню. Согласно этому способу, при рекурсивном вызове предиката задаются минимальное и максимальное значение, зависящие от уже просмотренной «верхушки» дерева, между которыми может находиться корень рассматриваемого поддерева.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
/*
    Задание 1.
    1. Определить, является ли заданный список упорядоченным по
возрастанию или убыванию.

    ?- is_ordered([1,2,3], X).
    X = asc

    ----

    Вывод программы:
        asc - по возрастанию
        desc - по убыванию
        not_ordered - список не отсортирован
*/

is_ordered([], asc).
is_ordered([], desc).
is_ordered([_], asc).
is_ordered([_], desc).
is_ordered([X,Y|Tail], asc) :- X =< Y, is_ordered([Y|Tail], asc).
is_ordered([X,Y|Tail], desc) :- X >= Y, is_ordered([Y|Tail], desc).
is_ordered(Arr, not_ordered) :- \+is_ordered(Arr, asc),
\+is_ordered(Arr, desc).

/*
    Задание 2.
    1. Создайте предикат, проверяющий, что дерево является двоичным
справочником.

    ?- is_ordered(tr(2,tr(7, nil, nil),tr(3,tr(4, nil, nil), tr(1,
nil, nil)))).
    No

    ----

    Программа поддерживает неполные деревья:
        <Число>                 <=> tr(<Число>, nil, nil)
        tr(<Число>)             <=> tr(<Число>, nil, nil)
        tr(<Число1>, <Число2>)  <=> tr(<Число1>, <Число2>, nil)

    Например:

    ?- is_ordered(tr(1,-1,1)).
    Yes

    ?- is_ordered(tr(1,1,1)).
    No

    ?- is_ordered(tr(1,tr(1),1)).
    No
```

```
    ?- is_ordered(tr(1,tr(0),1)).
    Yes
*/

is_ordered_inner(nil, _, _).
is_ordered_inner(Value, Min, Max) :- integer(Value),
is_ordered_inner(tr(Value), Min, Max).
is_ordered_inner(tr(Value), Min, Max) :- is_ordered_inner(tr(Value,
nil), Min, Max).
is_ordered_inner(tr(Value, Left), Min, Max) :-
is_ordered_inner(tr(Value, Left, nil), Min, Max).
is_ordered_inner(tr(Value, Left, Right), Min, Max) :-
is_clamped(Value, Min, Max), is_ordered_inner(Left, Min, Value),
is_ordered_inner(Right, Value, Max).

is_clamped(Value, Min, Max) :- integer(Value), (Min = nil,!; Value
>= Min), (Max = nil,!; Value < Max).

is_ordered(Tree) :- is_ordered_inner(Tree, nil, nil),!.
```