Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

НАПРАВЛЕНИЕ ПОДГОТОВКИ «09.03.04 Программная инженерия»

ОТЧЕТ

по лабораторной работе №11

Название: Рекурсия на Prolog

Дисциплина: Функциональное и логическое программирование

|  |  |  |  |
| --- | --- | --- | --- |
| Студент | ИУ7-65Б  Группа | Подпись, дата | Талышева О.Н.  Фамилия И.О. |
| Преподаватель |  | Подпись, дата | Толпинская Н.Б.  Фамилия И.О. |

Москва, 2025 г.

**Задание**

Используя хвостовую рекурсию, разработать (комментируя назначение аргументов) эффективную программу, позволяющую:

1. Найти длину списка (по верхнему уровню);
2. Найти сумму элементов числового списка;
3. Найти сумму элементов числового списка, стоящих на нечетных позициях исходного списка (нумерация от 0);
4. Сформировать список из элементов числового списка, больших заданного значения;
5. Удалить заданный элемент из списка (один или все вхождения).
6. Объединить два списка.

Для каждого задания:

* Проверить корректность результатов.
* Построить таблицу, отражающую порядок работы системы.

**Результаты работы:**

1. Найти длину списка (по верхнему уровню)

Ниже идут листинги программы с несколькими вариантами вопросов с ответами:

**domains**

list = **integer**\*.

**predicates**

% аргументы: разбираемый список, аккумулятор, результат

lenHelp(list, **integer**, **integer**).

% аргументы: список, результат

len(list, **integer**).

**clauses**

lenHelp([], R, R) :- !.

lenHelp([\_|T], AccLen, R) :-

NewAccLen = AccLen + 1,

lenHelp(T, NewAccLen, R).

len(Arr, R) :- lenHelp(Arr, 0, R).

**goal**

|  |  |
| --- | --- |
| len([1, 2, 3], Res). | Res=3  1 Solution |
| len([], Res). | Res=0  1 Solution |

Таблица для цели len([1, 2, 3], Res).

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Состояние резольвенты | Унификация: T1 = T2 (результат и подстановка) | Действия (прямой ход / откат) |
| 1 | len([1, 2, 3], Res) | T1 = len([1, 2, 3], Res)  T2 = len(Arr, R)  ⟹ Arr=[1,2,3], R=Res | Переход к lenHelp([1,2,3], 0, Res) |
| 2 | lenHelp([1,2,3], 0, Res) | T1 = lenHelp([1|2, 3], 0, Res)  T2 = lenHelp([\_|T], AccLen, R)  ⟹ T=[2,3], AccLen = 0, R=Res | Переход к lenHelp([2,3], 1, Res) |
| 3 | lenHelp([2,3], 1, Res) | T1 = lenHelp([2, 3], 1, Res)  T2 = lenHelp([\_|T], AccLen, R)  ⟹ T=[3], AccLen = 1, R=Res | Переход к lenHelp([3], 2, Res) |
| 4 | lenHelp([3], 2, Res) | T1 = lenHelp([3], 2, Res)  T2 = lenHelp([\_|T], AccLen, R)  ⟹ T=[], AccLen = 2, R=Res | Переход к lenHelp([], 3, Res) |
| 5 | lenHelp([], 3, Res) | T1 = lenHelp([], 3, Res)  T2 = lenHelp([], R, R)  ⟹ R=3, Res=3 | Успех, отсечение ! |

Вывод: Res = 3.

1. Найти сумму элементов числового списка

Ниже идут листинги программы с несколькими вариантами вопросов с ответами:

**domains**

list = **integer**\*.

**predicates**

% аргументы: разбираемый список, аккумулятор, результат

sumElemsHelp(list, **integer**, **integer**).

% аргументы: список, результат

sumElems(list, **integer**).

**clauses**

sumElemsHelp([], R, R) :- !.

sumElemsHelp([H|T], AccSumElems, R) :-

NewAccSumElems = AccSumElems + H,

sumElemsHelp(T, NewAccSumElems, R).

sumElems(Arr, R) :- sumElemsHelp(Arr, 0, R).

**goal**

|  |  |
| --- | --- |
| sumElems([1, 2, 3], Res). | Res=6  1 Solution |
| sumElems([], Res). | Res=0  1 Solution |

Таблица для цели sumElems([1, 2, 3], Res).

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Состояние резольвенты | Унификация: T1 = T2 (результат и подстановка) | Действия (прямой ход / откат) |
| 1 | sumElems ([1, 2, 3], Res) | T1 = sumElems ([1, 2, 3], Res)  T2 = sumElems (Arr, R)  ⟹ Arr=[1,2,3], R=Res | Переход к sumElemsHelp ([1,2,3], 0, Res) |
| 2 | sumElemsHelp ([1,2,3], 0, Res) | T1 = sumElemsHelp ([1|2, 3], 0, Res)  T2 = sumElemsHelp ([H|T], AccSumElems, R)  ⟹ H=1, T=[2,3], AccSumElems = 0, R=Res | Переход к sumElemsHelp ([2,3], 1, Res) |
| 3 | sumElemsHelp ([2,3], 1, Res) | T1 = sumElemsHelp ([2, 3], 1, Res)  T2 = sumElemsHelp ([H|T], AccSumElems, R)  ⟹ H=2, T=[3], AccSumElems = 1, R=Res | Переход к sumElemsHelp ([3], 3, Res) |
| 4 | sumElemsHelp ([3], 3, Res) | T1 = sumElemsHelp ([3], 3, Res)  T2 = sumElemsHelp ([H|T], AccSumElems, R)  ⟹ H=3, T=[], AccSumElems = 3, R=Res | Переход к sumElemsHelp ([], 6, Res) |
| 5 | sumElemsHelp ([], 6, Res) | T1 = sumElemsHelp ([], 6, Res)  T2 = sumElemsHelp ([], R, R)  ⟹ R=6, Res=6 | Успех, отсечение ! |

Вывод: Res = 6.

1. Найти сумму элементов числового списка, стоящих на нечетных позициях исходного списка (нумерация от 0)

Ниже идут листинги программы с несколькими вариантами вопросов с ответами:

**domains**

list = **integer**\*.

**predicates**

% аргументы: разбираемый список, аккумулятор, четность индекса, результат

sumOddElemsHelp(list, **integer**, **integer**, **integer**).

% аргументы: список, результат

sumOddElems(list, **integer**).

**clauses**

sumOddElemsHelp([], R, \_, R) :- !.

sumOddElemsHelp([\_|T], AccSumElems, 0, R) :- sumOddElemsHelp(T, AccSumElems, 1, R).

sumOddElemsHelp([H|T], AccSumElems, 1, R) :-

NewAccSumElems = AccSumElems + H,

sumOddElemsHelp(T, NewAccSumElems, 0, R).

sumOddElems(Arr, R) :- sumOddElemsHelp(Arr, 0, 0, R).

**goal**

|  |  |
| --- | --- |
| sumOddElems([1, 2, 3, 4, 5], Res). | Res=6  1 Solution |
| sumOddElems([], Res). | Res=0  1 Solution |

Таблица для цели sumOddElems([1, 2, 3], Res).

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Состояние резольвенты | Унификация: T1 = T2 (результат и подстановка) | Действия (прямой ход / откат) |
| 1 | sumOddElems([1,2,3], Res) | T1 = sumOddElems([1,2,3], Res)  T2 = sumOddElems(Arr, R)  ⟹ Arr=[1,2,3], R=Res | Переход к sumOddElemsHelp([1,2,3], 0, 0, Res) |
| 2 | sumOddElemsHelp([1,2,3], 0, 0, Res) | T1=sumOddElemsHelp([1,2,3], 0, 0, Res)  T2=sumOddElemsHelp([\_|T], AccSumElems, 0, R)  ⟹ T=[2,3], AccSumElems=0, R=Res | Переход к sumOddElemsHelp([2,3], 0, 1, Res) |
| 3 | sumOddElemsHelp([2,3], 0, 1, Res) | T1=sumOddElemsHelp([2,3], 0, 1, Res)  T2=sumOddElemsHelp([H|T], AccSumElems, 1, R)  ⟹ H=2, T=[3], AccSumElems=0, R=Res | Переход к sumOddElemsHelp([3], 2, 0, Res) |
| 4 | sumOddElemsHelp([3], 2, 0, Res) | T1=sumOddElemsHelp([3], 2, 0, Res)  T2=sumOddElemsHelp([\_|T], AccSumElems, 0, R)  ⟹ T=[], AccSumElems=2, R=Res | Переход к sumOddElemsHelp([], 2, 1, Res) |
| 5 | sumOddElemsHelp([], 2, 1, Res) | T1=sumOddElemsHelp([], 2, 1, Res)  T2= sumOddElemsHelp([], R, \_, R)  ⟹ R=2, Res=2 | Успех, отсечение ! |

Вывод: Res = 2.

1. Сформировать список из элементов числового списка, больших заданного значения

Ниже идут листинги программы с несколькими вариантами вопросов с ответами:

**domains**

list = **integer**\*.

**predicates**

% аргументы: разбираемый список, аккумулятор, граничный элемент, результат

sumMoreElemsHelp(list, **integer**, **integer**, **integer**).

% аргументы: список, граничный элемент, результат

sumMoreElems(list, **integer**, **integer**).

**clauses**

sumMoreElemsHelp([], R, \_, R) :- !.

sumMoreElemsHelp([H|T], AccSumElems, BordElem, R) :-

H > BordElem,

NewAccSumElems = AccSumElems + H,

sumMoreElemsHelp(T, NewAccSumElems, BordElem, R),

!.

sumMoreElemsHelp([\_|T], AccSumElems, BordElem, R) :- sumMoreElemsHelp(T, AccSumElems, BordElem, R).

sumMoreElems(Arr, BordElem, R) :- sumMoreElemsHelp(Arr, 0, BordElem, R).

**goal**

|  |  |
| --- | --- |
| sumMoreElems([1, 2, 3, 4, 5], 3, Res). | Res=9  1 Solution |
| sumMoreElems([], 3, Res). | Res=0  1 Solution |

Таблица для цели sumMoreElems([1, 2, 3], 2, Res).

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Состояние резольвенты | Унификация: T1 = T2 (результат и подстановка) | Действия (прямой ход / откат) |
| 1 | sumMoreElems([1, 2, 3], 2, Res) | T1=sumMoreElems([1,2,3], 2, Res)  T2=sumMoreElems(Arr, BordElem, R)  ⟹ Arr=[1,2,3], BordElem=2, R=Res | Переход к sumMoreElemsHelp([1,2,3], 0, 2, Res) |
| 2 | sumMoreElemsHelp([1,2,3], 0, 2, Res) | T1=sumMoreElemsHelp([1,2,3], 0, 2, Res)  T2=sumMoreElemsHelp([H|T], AccSumElems, BordElem, R)  ⟹ H=1, T=[2,3], AccSumElems=0, BordElem=2, R=Res | H > BordElem не выполняется  откат |
| 3 | sumMoreElemsHelp([1,2,3], 0, 2, Res) | T1=sumMoreElemsHelp([1,2,3], 0, 2, Res)  T2=sumMoreElemsHelp([\_|T], AccSumElems, BordElem, R)  ⟹ T=[2,3], AccSumElems=0, BordElem=2, R=Res | Переход к sumMoreElemsHelp([2,3], 0, 2, Res) |
| 4 | sumMoreElemsHelp([2,3], 0, 2, Res) | T1=sumMoreElemsHelp([2,3], 0, 2, Res)  T2=sumMoreElemsHelp([H|T], AccSumElems, BordElem, R)  ⟹ H=2, T=[3], AccSumElems=0, BordElem=2, R=Res | H > BordElem не выполняется  откат |
| 5 | sumMoreElemsHelp([2,3], 0, 2, Res) | T1=sumMoreElemsHelp([2,3], 0, 2, Res)  T2=sumMoreElemsHelp([\_|T], AccSumElems, BordElem, R)  ⟹ T=[3], AccSumElems=0, BordElem=2, R=Res | Переход к sumMoreElemsHelp([3], 0, 2, Res) |
| 6 | sumMoreElemsHelp([3], 0, 2, Res) | T1=sumMoreElemsHelp([3], 0, 2, Res)  T2=sumMoreElemsHelp([H|T], AccSumElems, BordElem, R)  ⟹ H=3, T=[], AccSumElems=0, BordElem=2, R=Res | Переход к sumMoreElemsHelp([], 3, 2, Res) |
| 7 | sumMoreElemsHelp([], 3, 2, Res) | T1=sumMoreElemsHelp([], 3, 2, Res)  T2= sumMoreElemsHelp([], R, \_, R)  ⟹ R=3, Res=3 | Успех, отсечение ! |

Вывод: Res = 3.

1. Удалить заданный элемент из списка (один или все вхождения)

Ниже идут листинги программы с несколькими вариантами вопросов с ответами:

**domains**

list = **integer**\*.

**predicates**

% аргументы: стартовый список, удаляемый элемент, результат

delElemArr(list, **integer**, list).

**clauses**

delElemArr([], \_, []) :- !.

delElemArr([DelElem|T], DelElem, R) :- delElemArr(T, DelElem, R), !.

delElemArr([H|T], DelElem, [H|R]) :- delElemArr(T, DelElem, R).

**goal**

|  |  |
| --- | --- |
| delElemArr([1, 2, 3, 4, 5], 3, Res). | Res=[1, 2, 4, 5],  1 Solution |
| delElemArr([], 3, Res). | Res=[]  1 Solution |

Таблица для цели delElemArr([1, 2], 2, Res).

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Состояние резольвенты | Унификация: T1 = T2 (результат и подстановка) | Действия (прямой ход / откат) |
| 1 | delElemArr([1, 2], 2, Res) | T1= delElemArr([1, 2], 2, Res)  T2= delElemArr([H|T], DelElem, [H|R])  ⟹ H=1, T=[2], DelElem=2, [1|R] = Res | Переход к delElemArr([2], 2, R) |
| 2 | delElemArr([2], 2, R) | T1= delElemArr([2], 2, R)  T2= delElemArr([DelElem |T], DelElem, [R])  ⟹ H=2, T=[], DelElem=2, R=R | Переход к delElemArr([], 2, R) |
| 3 | delElemArr([], 2, R) | T1= delElemArr([], 2, R)  T2= delElemArr([], \_, [])  ⟹ R=[] | Успех, отсечение ! |

Вывод: Res = [1].

1. Объединить два списка

Ниже идут листинги программы с несколькими вариантами вопросов с ответами:

**domains**

list = **integer**\*.

**predicates**

% аргументы: стартовый список 1, стартовый список 2, объединённый список

unionArr(list, list, list).

**clauses**

unionArr([], Arr2, Arr2) :- !.

unionArr(Arr1, [], Arr1) :- !.

unionArr([H|T], Arr2, [H|R]) :- unionArr(T, Arr2, R).

**goal**

|  |  |
| --- | --- |
| unionArr([1, 2, 3, 4, 5], [6, 7, 8], Res). | Res = [1,2,3,4,5,6,7,8]  1 Solution |
| unionArr([1, 2, 3], [], Res2). | Res2 = [1,2,3]  1 Solution |
| unionArr([], [1, 2, 3], Res3). | Res3 = [1,2,3]  1 Solution |
| unionArr([], [], Res4). | Res4 = []  1 Solution |

Таблица для цели unionArr([1, 2], [3], Res).

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Состояние резольвенты | Унификация: T1 = T2 (результат и подстановка) | Действия (прямой ход / откат) |
| 1 | unionArr([1, 2], [3], Res) | T1= unionArr([1, 2], [3], Res)  T2= unionArr([H|T], Arr2, [H|R])  ⟹ H=1, T=[2], Arr2=[3], [1|R] = Res | Переход к unionArr ([2], [3], R) |
| 2 | unionArr([2], [3], R) | T1= unionArr([2], [3], R)  T2= unionArr([H|T], Arr2, [H|R])  ⟹ H=2, T=[], Arr2=[3], [2|R] = R | Переход к unionArr ([], [3], R) |
| 3 | unionArr([], [3], R) | T1= unionArr([], [3], R)  T2= unionArr([], Arr2, Arr2)  ⟹ Arr2=[3], R=[3] | Успех, отсечение ! |

Вывод: Res = [1, 2, 3].