**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

(СПбГУТ)

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ **(ИТПИ)**

Кафедра программной инженерии и вычислительной техники **(ПИиВТ)**

Дисциплина: «Разработка приложений искусственного интеллекта в киберфизических системах»

Лабораторная работа №3.

**Тема: «Разработка интерполяционного поиска»**

Выполнил:

Студент группы ИКПИ-23

Харлова А.А

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Принял:

Ерофеев С.А.

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_

2024 г.

**Постановка задачи**

Разработать программу на языке PROLOG, которая осуществляет интерполяционный поиск в списке, отсортированном с помощью алгоритма сортировки расчёской.

**Алгоритм решения**

Задача реализации интерполяционного поиска значения в спике подрузамевает первоначальную сортировку данного списка. В ходе выполнения работы необходимо реализовать два независимых алгоритма поиска и сортировки, после чего объединить их в одну программу. Алгоритм решения кратко представлен на рисунке 1.

**Описание программы**

В таблице 1 приведено описание термов, используемых в программе.

|  |  |  |
| --- | --- | --- |
| Терм | Тип | Описание |
| option | integer | Число типа "integer". Способ ввода/вывода данных программы. |
| index | integer | Число типа "integer". Индекс списка чисел. |
| list | real\* | Список чисел типа "real". |
| stream | file | Стандартный предикат для обработки файлов, который позволяет использовать файлы произвольного доступа. |

*Таблица 1. Используемые термы*

Программа разделена на 4 модуля: "SEARCH.PRO", "SORT.PRO", "STRUCTS.PRO", "IO.PRO". В таблицах 2-5 приведено описание предикатов, используемых в каждом модуле программы.

Модуль "SEARCH.PRO" является основным и содержит термы, указанные в таблице 1, точку входа в программу, реализованный алгоритм для решения задачи, предикаты для работы интерполяционного поиска. Описание предикатов данного модуля – таблица 2.

|  |  |
| --- | --- |
| Предикат | Описание |
| start() | Запуск программы |
| interpolationSearch(list, real, index) | Интерполяционный поиск |
| interpolationSearchCycle(list, index, index, real, index) | Цикл интерполяционного поиска |
| searchConditions(real, index, index, real, index, index) | Условия следующего шага поиска |
| searchFormula(list, index, index, real, integer) | Формула для вычисления следующего шага поиска |

*Таблица 2. Предикаты модуля "SEARCH.PRO"*

В модуль "SORT.PRO" включены предикаты для алгоритма сортировки расчёской. Описание предикатов данного модуля – таблица 3.

|  |  |
| --- | --- |
| Предикат | Описание |
| combSort(list, list) | Сортировка расчёской |
| gapShift(integer, integer) | Следующий шаг сортировки расчёской |
| combSortCycle(list, integer, list, integer) | Основной цикл сортировки |
| sortIteratorInit(list, integer, list) | Итератор списка с определенным шагом |
| sortIterator(list, index, integer, integer, list) | Цикл прохода по списку для сортировки с определенным шагом |
| sortConditions(list, index, index, list) | Условия перестановки элементов списка |

*Таблица 3. Предикаты модуля "SORT.PRO"*

Модуль "STRUCTS.PRO" определяет предикаты для работы со структурой данных, такой как список, и взаимодействие с ней, а также общие операции преобразования типов данных. Описание предикатов данного модуля – таблица 4.

|  |  |
| --- | --- |
| Предикат | Описание |
| realToInt(real, integer) | Приведение числа "real" к числу типа "integer" |
| get(index, list, real) | Получение элемента списка по индексу |
| lenth(list, integer) | Длина списка |
| replace(index, list, real, list) | Замена элемента списка по индексу |
| swap(list, index, index, list) | Поменять местами два элемента списка |

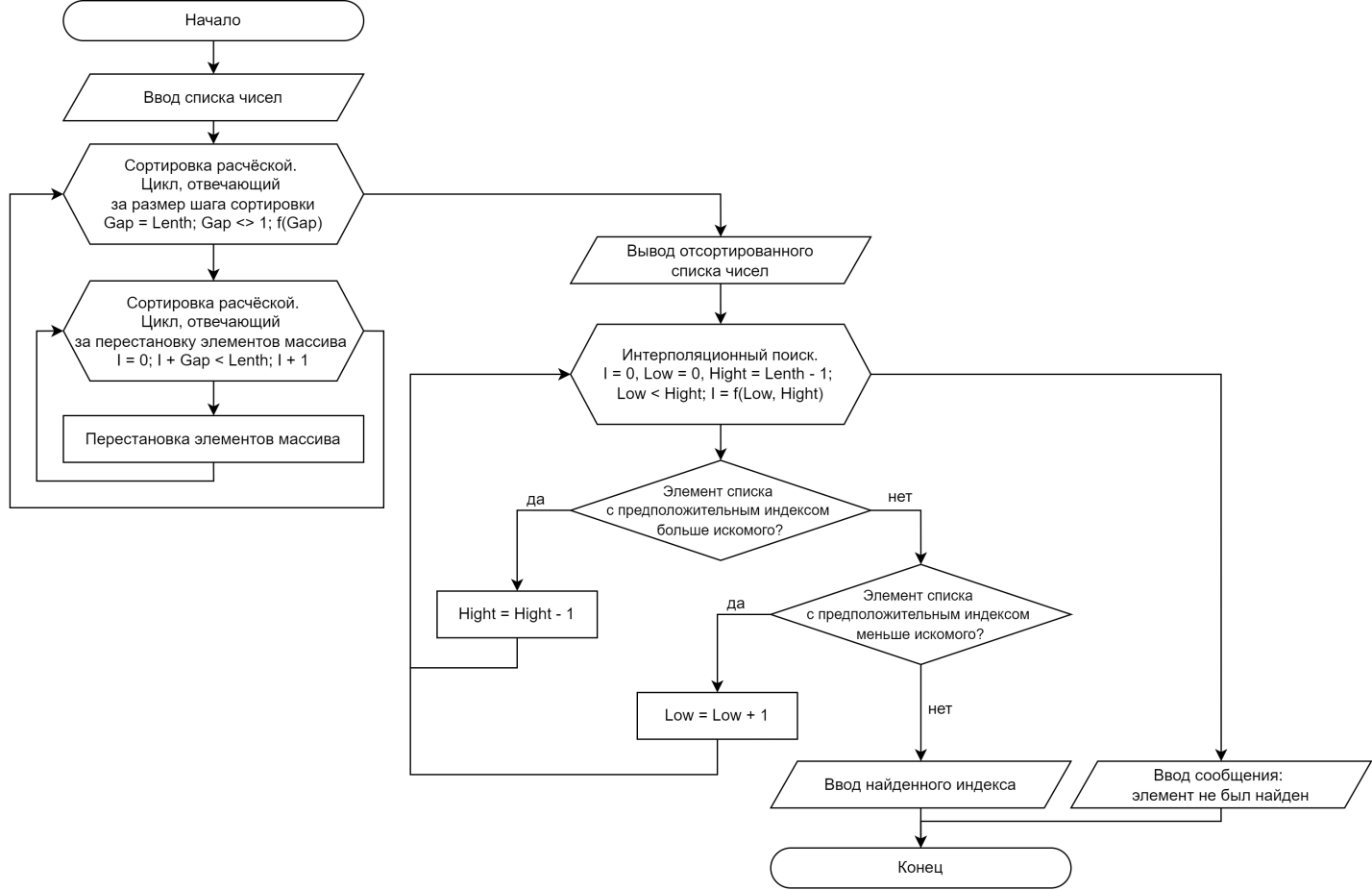
*Таблица 4. Предикаты модуля "STRUCTS.PRO"*

В четвертом модуле "IO.PRO" реализованы предикаты для взаимодействия программы с пользователем, например, ввод и вывод данных, отображение меню программы. Описание предикатов данного модуля – таблица 5.

|  |  |
| --- | --- |
| Предикат | Описание |
| menu(option) | Пользовательское меню |
| correctMenuOption(option) | Коррестность выбора опции пользовательского меню |
| inputFilePath(string, string) | Путь к файлу |
| getFilePath(char, string, string) | Выбор файла |
| inputList(list, option) | Ввод списка |
| inputValue(real, option) | Ввод числа типа "real" |
| outputIndex(index, option) | Форматированный вывод индекса списка |
| outputList(list, string, option) | Форматированный вывод списка |
| outputData(list, list, real, index, option) | Форматированный вывод входных и выходных данных в файл |

*Таблица 5. Предикаты модуля "IO.PRO"*

**Алгоритм решения**

****

**Используемые формулы**

В ходе разработки программы были использованы формулы для расчёта параметров сортировки расчёской и интерполяционного поиска. Все применённые формулы представлены в таблице 6.

|  |  |
| --- | --- |
| Формула | Описание |
| - *искомый размер шага;*  - *предыдущий размер шага;*  - *коэффициент частоты шага.* | Формула нахождения следующего размера шага сортировки расчёской на основе предыдущего |
| - *предположительный индекс искомого элемента в списке*;  - *нижняя граница диапазона поиска;*  - *верхняя граница диапазона поиска;*  *alue* - *искомый элемент списка;*  - *элемент списка с индексом, равным нижней границе диапазона поиска;*  - *элемент списка с индексом, равным верхней границе диапазона поиска.* | Формула интерполяционного поиска |

*Таблица 6. Используемые формулы*

**Вывод**

Была разработана программа на языке PROLOG, осуществляющая интерполяционный поиск в списке, отсортированном с помощью алгоритма сортировки расчёской. В ходе разработки были введены необходимые предикаты. Выполнено тестирование. Поставленная задача выполнена.

**Код программы**

**Модуль "SEARCH.PRO":**

DOMAINS

option = integer

index = integer

list = real\*

file = stream

PREDICATES

start()

interpolationSearch(list, real, index)

interpolationSearchCycle(list, index, index, real, index)

searchConditions(real, index, index, real, index, index)

searchFormula(list, index, index, real, integer)

INCLUDE "STRUCTS.PRO"

INCLUDE "SORT.PRO"

INCLUDE "IO.PRO"

GOAL

start.

CLAUSES

% ----------------------- Начало программы -----------------------

start() :-

menu(Option),

inputList(InputList, Option),

combSort(InputList, SortedList),

outputList(SortedList, "Sorted list", 1),

inputValue(Value, 1),

interpolationSearch(SortedList, Value, Index),

outputIndex(Index, 1),

outputData(InputList, SortedList, Value, Index, Option), !;

write("Try again..."), nl.

% -------------------- Интерполяционный поиск --------------------

interpolationSearch(List, Value, Index):-

lenth(List, Lenth),

HightIndex = Lenth - 1,

interpolationSearchCycle(List, 0, HightIndex, Value, Index).

% Цикл поиска

interpolationSearchCycle(List, LowIndex, HightIndex, Value, FoundIndex):-

searchFormula(List, LowIndex, HightIndex, Value, Index),

get(Index, List, FoundValue),

FoundValue = Value, FoundIndex = Index, !;

searchFormula(List, LowIndex, HightIndex, Value, Index),

get(Index, List, FoundValue),

searchConditions(Value, LowIndex, HightIndex, FoundValue,NewLowIndex , NewHightIndex),

NewLowIndex < NewHightIndex,

interpolationSearchCycle(List, NewLowIndex, NewHightIndex, Value, FoundIndex), !; FoundIndex = -1.

% Условия следующего шага

searchConditions(Value, LowIndex, HightIndex, FoundValue, NewLowIndex, NewHightIndex):-

FoundValue > Value, NewLowIndex = LowIndex, NewHightIndex = HightIndex - 1, !;

FoundValue < Value, NewLowIndex = LowIndex + 1, NewHightIndex = HightIndex.

% Формула интерполяционного поиска

% index = low + [(val-lys[low])\*(high-low) / (lys[high]-lys[low])]

searchFormula(List, LowIndex, HightIndex, Value, Index):-

get(LowIndex, List, LowElement),

get( HightIndex, List, HightElement),

Index = LowIndex + (Value-LowElement)\*(HightIndex-LowIndex) / (HightElement-LowElement).

**Модуль "SORT.PRO":**

% -------------------- Сортировка расчёской --------------------

CONSTANTS

coefficient = 1.3

PREDICATES

gapShift(integer, integer)

combSort(list, list)

combSortCycle(list, integer, list, integer)

sortIteratorInit(list, integer, list)

sortIterator(list, index, integer, integer, list)

sortConditions(list, index, index, list)

CLAUSES

combSort(Data, SortedData):-

lenth(Data, Lenth),

Gap = Lenth - 1,

combSortCycle(Data, Gap, SortedData, 0).

% Внешний цикл сортировки

combSortCycle(List, Gap, SortedList, N):-

sortIteratorInit(List, Gap, NewList), NewN = N + 1,

gapShift(Gap, NewGap), NewGap >= 1,

combSortCycle(NewList, NewGap, SortedList, NewN), !;

sortIteratorInit(List, Gap, NewList),

SortedList = NewList.

% Итератор цикла сортировки с заданным шагом

sortIteratorInit([], \_ , []).

sortIteratorInit(List, Gap, NewList):-

lenth(List, Lenth),

sortIterator(List, 0, Gap, Lenth, NewList).

sortIterator(List, I, Gap, Lenth, NewList):-

J = I + Gap,

J < Lenth,

sortConditions(List, I, J, ListTmp),

NewI = I + 1,

sortIterator(ListTmp, NewI, Gap, Lenth, NewList), !;

NewList = List.

% Условия сортировки

sortConditions(List, I, J, ListTmp):-

get(I, List, Element1),

get(J, List, Element2),

Element1 > Element2,

swap(List, I, J, ListTmp), !;

ListTmp = List.

% Новый шаг

gapShift(Gap, NewGap):-

Div = Gap / coefficient,

NewValue = Div,

realToInt(NewValue, NewGap).

**Модуль "STRUCTS.PRO":**

% -------------------- Структуры данных --------------------

PREDICATES

realToInt(real, integer)

get(index, list, real)

lenth(list, integer)

replace(index, list, real, list)

swap(list, index, index, list)

CLAUSES

% ------------------------- Общее ---------------------------

% Округление значения до целого числа

realToInt(Real, Integer):-

Integer = Real, Integer <= Real, Real >= 0, !;

Integer = Real - 1, Real >= 0, !;

Integer = Real, Integer >= Real, Real < 0, !;

Integer = Real + 1.

% ------------------------- Списки --------------------------

% Получение элемента списка по индексу

get(0, [Element|\_], Element).

get(Index, [\_|Tail], Element) :-

IndexNew = Index - 1,

get(IndexNew, Tail, Element).

% Замена элемента списка по индексу

replace(0, [\_|Tail], NewElement, [NewElement|Tail]) .

replace(Index, [H|Tail], NewElement, [H|NewList]) :-

IndexNew = Index - 1,

replace(IndexNew, Tail, NewElement, NewList).

% Поменять местами два элемента списка

swap(List, Index1, Index2, NewList):-

get(Index1, List, Element1),

get(Index2, List, Element2),

replace(Index1, List, Element2, ListTmp),

replace(Index2, ListTmp, Element1, NewList).

% Длина списка

lenth([], 0).

lenth([\_|T], Lenth) :-

lenth(T, LenthNew),

Lenth = LenthNew + 1.

**Модуль "IO.PRO":**

% ------------------ Ввод/вывод данных ------------------

% (включена проверка корректности ввода)

PREDICATES

menu(option)

correctMenuOption(option)

inputList(list, option)

inputFilePath(string, string)

getFilePath(char, string, string)

inputValue(real, option)

outputIndex(index, option)

outputList(list, string, option)

outputData(list, list, real, index, option)

CLAUSES

% -------------------- Ввод данных ------------------------

% Стартовое меню

menu(Option):-

nl, write("===== PROGRAM STARTED ====="), nl,

write("1. I/O console"), nl,

write("2. I/O file"), nl,

readint(Option),

correctMenuOption(Option), !;

write("[ ERROR ] Incorrect input"), nl, fail.

% Ввод имени файла в системе

inputFilePath(FilePath, DefaultFilePath):-

write("Dafault path: ", DefaultFilePath), nl,

write("Change? [y/n]"), nl,

readchar(Action),

getFilePath(Action, FilePath, DefaultFilePath); fail.

% Выбор файла

getFilePath(Action, FilePath, DefaultFilePath):-

Action = 'n', FilePath = DefaultFilePath, !;

Action = 'y', nl, write("New absolute filepath: "), readln(FilePath), !;

nl, write("[ ERROR ] Incorrect input"), nl, fail.

% Ввод списка из файла

inputList(List, 2) :-

inputFilePath(FilePath, "input.txt"),

existfile(FilePath),

openread(stream, FilePath),

readdevice(stream),

readterm(list, List),

closefile(stream),

readdevice(keyboard).

% Ввод списка с консоли

inputList(List, 1):-

write("Enter list:"), nl,

write(" "),

readterm(list, List).

% Ввод значения, которое следует найти

inputValue(Value, 1):-

write("Enter value to find: "),

readReal(Value).

% -------------------- Проверка ввода -----------------------

% Корректная опция меню

correctMenuOption(Option):-

Option = 1; Option = 2.

% --------------------- Вывод данных ------------------------

% Вывод списка в консоль

outputList(SortedList, Tag, 1):-

write(Tag, ":"), nl,

write(" "),

write(SortedList), nl.

% Вывод индекса в консоль

outputIndex(Index, 1):-

Index >= 0, write("Index: ", Index), nl, !;

write("No value in the list"), nl.

% Вывод результата в файл

outputData(List, SortedList, Value, Index, Option):-

Option = 1, !;

Option = 2,

nl, write("Writing data to the file..."), nl,

inputFilePath(FilePath, "output.txt"),

openwrite(stream, FilePath),

writedevice(stream),

outputList(List, "List", 1),

outputList(SortedList, "Sorted list", 1),

write("Value: ", Value), nl,

outputIndex(Index, 1),

closefile(stream),

writedevice(screen),

write("Input data was written successfully"), nl, !;

write("[ ERROR ] Can't write data to the file"), nl, fail.