Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Комсомольский-на-Амуре государственный университет»

Факультет компьютерных технологий

Кафедра «МОП ЭВМ»

ЛАБОРАТОРНАЯ РАБОТА 3

по дисциплине «Логическое программирование»

Списки и бинарные деревья

Студент группы 6ИСб-1 А.М. Закусило

Преподаватель Е.Б. Абарникова

2019

1. **Лист задания**

**Тема:** Списки и бинарные деревья.

**Цель:** Изучить основные операции со списками и бинарными деревьями.

**Задания:** Написать программу для реализации следующих действий:

1. Вставка элементов в список после всех найденных вхождений указанного
2. Реверс списка
3. Объединение трех списков в один в указанном порядке
4. Сортировка списка по убыванию методом вставки
5. **Теоретическое описание**

В Прологе список является рекурсивным типом данных, содержащим внутри произвольное число объектов (целые числа, действительные числа, символы, строки, символические имена и структуры), или не содержащим их вовсе, т.е. пустым списком.

Список состоит из узлов, каждый узел содержит данные (голову) и ссылку на следующий узел (хвост). Совокупность элементов списка заключается в квадратные скобки ([]), элементы друг от друга отделяются запятыми. В программе голова отделяется от хвоста символом «|». Хвост списка является таким же списком, как и исходный, поэтому обрабатывается аналогичным образом (рекурсивно).

Деревья, также, как и списки, являются рекурсивным типом данных. Дерево - это структура данных, которая может быть разделена на корень дерева, левое и правое поддеревья. Так как левое и правое поддеревья в свою очередь являются деревьями, структура рекурсивна. Кроме того, дерево является еще и составным объектом данных. Бинарные деревья задаются с помощью тернарного функтора.

В Прологе существует возможность реализовать сортировку методом вставки или методом простых включений.

Суть алгоритма заключается в том, что предикат принимает два списка: исходный (содержащий некоторые значения) и список для формирования результата (пустой), далее, до наступления терминальной ситуации, от исходного списка рекурсивно отбрасываются головы. После достижения пустоты исходного списка, по точкам отката, список возвращает своё исходное состояние - возвращает себе головы, но кроме этого каждая возвращенная голова вставляется ещё и в результирующий список, стоить отметить что результирующий список всегда отсортирован, то есть элемент будет добавляться в определенную позицию.

1. **Описание предикатов**

sym\_list\_to\_string (symArray, string) – Перевод символьного списка в строку

1 - symArray – Символьный список.

2 - string – Результирующая строка.

string\_to\_list (string, symArray) – Перевод строки в символьный список

1 - string – Строка.

2 - symArray – Результирующий список.

count\_symbol\_before\_separator (string, symbol, integer, integer) – Подсчет количества символов в строке до разделителя или конца строки

1 - string – Входящая строка.

2 - symbol – Разделитель.

3 - integer – Количество символов.

4 - integer – Признак конца строки (1 или 0).

find\_el (symbol, symArray, integer) – Поиск символа в списке

1 - symbol – Искомый символ.

2 - symArray – Символьный список для поиска.

3 - integer – Признак нахождения символа (1 или 0).

insert (symbol, symbol, symArray, symArray) – Вставка символа после каждого вхождения

1 - symbol – Вставляемый символ.

2 - symbol – Символ после которого происходит вставка.

3 - symArray – Исходный символьный список.

4 - symArray – Измененный символьный список.

revers (symArray, symArray) – Реверс символьного списка.

1 - symArray – Исходный символьный список.

2 - symArray – Инвертированный символьный список.

append (integer, integer, symArray, symArray, symArray, symArray) – Объединение трех символьных списков в один в указанном порядке.

1 - integer – Номер списка, который будет добавлен первым.

2 - integer – Номер списка, который будет добавлен вторым.

3 - symArray – Первый символьный список.

4 - symArray – Второй символьный список.

5 - symArray – Третий символьный список.

6 - symArray – Результирующий символьный список.

append (symArray,symArray,symArray) – Слияние символьных двух списков.

1 - symArray – Первый символьный список.

2 - symArray – Второй символьный список.

3 - symArray – Результирующий символьный список.

insert\_sort (symArray, symArray) – Рекурсивный проход по исходному списку пока он не станет пустым, далее при возвращении элементы вставляются в подходящее место среди ранее упорядоченных элементов результирующего списка

1 - symArray– Исходный список.

2 - symArray– Упорядоченный список.

insertEl (symbol, symArray, symArray) – Вставка нового элемента в подходящее место среди ранее упорядоченных элементов

1 – symbol – Элемент для вставки.

2 - symArray – Упорядоченный список.

3 - symArray – Упорядоченный список с новым элементов.

1. **Текст программы**

Листинг 4.1 – Описание доменов программы

|  |
| --- |
| domains  intArray = integer\*  symArray = symbol\* |

Листинг 4.2 – Описание предикатов программы

|  |
| --- |
| predicates  sym\_list\_to\_string(symArray, string)  string\_to\_list(string, symArray)  count\_symbol\_before\_separator(string, symbol, integer, integer)  insert(symbol, symbol, symArray, symArray)  find\_el(symbol, symArray, integer)  revers(symArray, symArray)  revers(symArray, symArray, symArray)  append(integer, integer, symArray, symArray, symArray, symArray)  append(symArray,symArray,symArray)  insert\_sort(symArray, symArray)  insertEl(symbol, symArray, symArray) |

Листинг 4.3 – Символьный список в строку

|  |
| --- |
| sym\_list\_to\_string([], "").  sym\_list\_to\_string([H | T], S) :- T = [], sym\_list\_to\_string(T, S1), !, concat(H, S1, S).  sym\_list\_to\_string([H | T], S) :- sym\_list\_to\_string(T, S1), !, concat(H, ", ", H\_with\_space), concat(H\_with\_space, S1, S). |

Листинг 4.4 – Строку в символьный список

|  |
| --- |
| string\_to\_list("",[]):- !.  string\_to\_list(String,[Symbol|Tail]):- count\_symbol\_before\_separator(String, ",", CountSymbol, TextEnd), TextEnd = 0, !, frontstr(CountSymbol, String, Symbol, StringTailTemp), frontstr(1, StringTailTemp, \_, StringTail), string\_to\_list(StringTail, Tail).  string\_to\_list(String,[Symbol|Tail]):- count\_symbol\_before\_separator(String, ",", CountSymbol, TextEnd), TextEnd = 1, frontstr(CountSymbol, String, Symbol, \_), Tail = []. |

Листинг 4.5 – Подсчет количества элементов до сепаратора или конца строки

|  |
| --- |
| count\_symbol\_before\_separator("", \_, Count, TextEnd):- Count = 0, TextEnd = 1, !.  count\_symbol\_before\_separator(String, Separator, Count, TextEnd):- frontstr(1, String,Symbol,StringTail), Symbol <> Separator, count\_symbol\_before\_separator(StringTail, Separator, CountPlusOne, TextEnd), Count = CountPlusOne + 1, !.  count\_symbol\_before\_separator(\_, \_, Count, TextEnd):- Count = 0, TextEnd = 0. |

Листинг 4.6 – Вхождение элемента в список

|  |
| --- |
| find\_el(\_, [], Result):- Result = 0, !.  find\_el(Symbol, [Head|Tail], Result):- Head <> Symbol, find\_el(Symbol, Tail, Result), !.  find\_el(\_, \_, Result):- Result = 1. |

Листинг 4.7 – Вставка элементов в список после всех найденных вхождений указанного

|  |
| --- |
| insert(\_,\_,[],[]).  insert(X,Head,[Head|Z],[Head,X|T]):- insert(X,Head,Z,T),!.  insert(X,Y,[Head|Z],[Head|N]):- insert(X,Y,Z,N). |

Листинг 4.8 – Реверс списка

|  |
| --- |
| revers(X,Y):- revers([],X,Y).  revers(Y,[],Y).  revers(X1,[Z|X2],Y):- revers([Z|X1],X2,Y). |

Листинг 4.9 – Вхождение элемента

|  |
| --- |
| find\_el(\_, [], Result):- Result = 0, !.  find\_el(Symbol, [Head|Tail], Result):- Head <> Symbol, find\_el(Symbol, Tail, Result), !.  find\_el(\_, \_, Result):- Result = 1. |

Листинг 4.10 – Объединение трех списков в один в указанном порядке

|  |
| --- |
| append([], L, L).  append([N|L1], L2, [N|L3]) :- append(L1,L2,L3).  append(Fisrt, Second, L1, L2, L3, L4) :- Fisrt=1, Second=2, !, append(L1, L2, Res), append(Res, L3, L4).  append(Fisrt, Second, L1, L2, L3, L4) :- Fisrt=1, Second=3, !, append(L1, L3, Res), append(Res, L2, L4).  append(Fisrt, Second, L1, L2, L3, L4) :- Fisrt=2, Second=1, !, append(L2, L1, Res), append(Res, L3, L4).  append(Fisrt, Second, L1, L2, L3, L4) :- Fisrt=2, Second=3, !, append(L2, L3, Res), append(Res, L1, L4).  append(Fisrt, Second, L1, L2, L3, L4) :- Fisrt=3, Second=1, !, append(L3, L1, Res), append(Res, L2, L4).  append(Fisrt, Second, L1, L2, L3, L4) :- Fisrt=3, Second=2, !, append(L3, L2, Res), append(Res, L1, L4). |

Листинг 4.11 – Сортировка по убыванию метод вставки

|  |
| --- |
| insert\_sort([],[]).  insert\_sort([X|Tail],Sorted\_list) :- insert\_sort(Tail,Sorted\_Tail), insertEl(X,Sorted\_Tail,Sorted\_list).    insertEl(X,[Y|Sorted\_list],[Y|Sorted\_list1]) :- str\_int(X,X\_int), str\_int(Y,Y\_int), X\_int < Y\_int, !, insertEl(X,Sorted\_list,Sorted\_list1).  insertEl(X, [Y|Sorted\_list],[X,Y|Sorted\_list]) :- str\_int(X,X\_int), str\_int(Y,Y\_int), X\_int > Y\_int, !.    insertEl(X,[Y|Sorted\_list],[Y|Sorted\_list1]) :- X<Y, !, insertEl(X,Sorted\_list,Sorted\_list1).  insertEl(X,Sorted\_list,[X|Sorted\_list]). |

Листинг 4.12 – Перевод списка в строку

|  |
| --- |
| sym\_list\_to\_string([], "").  sym\_list\_to\_string([H | T], S) :- T = [], sym\_list\_to\_string(T, S1), !, concat(H, S1, S).  sym\_list\_to\_string([H | T], S) :- sym\_list\_to\_string(T, S1), !, concat(H, ", ", H\_with\_space), concat(H\_with\_space, S1, S). |

Листинг 4.13 – Описание окна «Демонстрация работы со списками» (главное меню)

|  |
| --- |
| %BEGIN main\_window, idc\_sort \_CtlInfo  dlg\_main\_window\_eh(\_Win,e\_Control(idc\_sort,\_CtrlType,\_CtrlWin,\_CtlInfo),0):-!,  dlg\_сортировка\_по\_убыванию\_методом\_вставок\_Create(\_Win),  !.  %END main\_window, idc\_sort \_CtlInfo  %BEGIN main\_window, idc\_append \_CtlInfo  dlg\_main\_window\_eh(\_Win,e\_Control(idc\_append,\_CtrlType,\_CtrlWin,\_CtlInfo),0):-!,  dlg\_объединение\_трёх\_списков\_в\_один\_в\_указанном\_порядке\_Create(\_Win),  !.  %END main\_window, idc\_append \_CtlInfo  %BEGIN main\_window, idc\_reverse \_CtlInfo  dlg\_main\_window\_eh(\_Win,e\_Control(idc\_reverse,\_CtrlType,\_CtrlWin,\_CtlInfo),0):-!,  dlg\_revers\_Create(\_Win),  !.  %END main\_window, idc\_reverse \_CtlInfo  %BEGIN main\_window, idc\_insert\_el \_CtlInfo  dlg\_main\_window\_eh(\_Win,e\_Control(idc\_insert\_el,\_CtrlType,\_CtrlWin,\_CtlInfo),0):-!,  dlg\_insert\_el\_Create(\_Win),  !.  %END main\_window, idc\_insert\_el \_CtlInfo  %BEGIN main\_window, idc\_help \_CtlInfo  dlg\_main\_window\_eh(\_Win,e\_Control(idc\_help,\_CtrlType,\_CtrlWin,\_CtlInfo),0):-!,  dlg\_Note("Помощь","Данная программа предназначена для демонстрации возможностей: вставки элемента в список после всех вхождений указанного, реверса списка, объединения трех списков в один, сортировка методом вставок."),  !.  %END main\_window, idc\_help \_CtlInfo  %BEGIN main\_window, idc\_cancel \_CtlInfo  dlg\_main\_window\_eh(\_Win,e\_Control(idc\_cancel,\_CtrlType,\_CtrlWin,\_CtlInfo),0):-!, win\_Destroy(\_Win),  !. |

Листинг 4.14 – Описание окна «ВСТАВКА ЭЛЕМЕНТА В СПИСОК ПОСЛЕ ВСЕХ ВХОЖДЕНИЙ УКАЗАННОГО»

|  |
| --- |
| dlg\_insert\_el\_eh(\_Win,e\_Control(idc\_ok\_sym,\_CtrlType,\_CtrlWin,\_CtlInfo),0):-  LBox\_Handle = win\_GetCtlHandle(\_Win,idc\_sym\_list\_res),    Array\_Handle = win\_GetCtlHandle(\_Win,idc\_edit),  Array\_Text = win\_GetText(Array\_Handle),  Array\_Text <> "",  string\_to\_list(Array\_Text, SymArray),    EditInsert\_Handle = win\_GetCtlHandle(\_Win,idc\_edit\_insert\_sym),  EditInsert\_Text =win\_GetText(EditInsert\_Handle),  EditInsert\_Text <> "",    EditAfter\_Handle = win\_GetCtlHandle(\_Win,idc\_edit\_after\_sym),  EditAfter\_Text =win\_GetText(EditAfter\_Handle),  EditAfter\_Text <> "",  find\_el(EditAfter\_Text, SymArray, FindResult),  FindResult <> 0,    insert(EditInsert\_Text, EditAfter\_Text, SymArray, SymArrayRes),    sym\_list\_to\_string(SymArrayRes,StringArray),  lbox\_Clear (LBox\_Handle),  lbox\_Add(LBox\_Handle, StringArray),  !.    dlg\_insert\_el\_eh(\_Win,e\_Control(idc\_ok\_sym,\_CtrlType,\_CtrlWin,\_CtlInfo),0):-  dlg\_Error("Ошибка", "Ошибка валидации данных. Проверьте, что бы символ, после которого нужно производить вставку, существовал"),  !. |

Листинг 4.15 – Описание окна «Реверс»

|  |
| --- |
| dlg\_revers\_eh(\_Win,e\_Control(idc\_ok\_sym,\_CtrlType,\_CtrlWin,\_CtlInfo),0):-  LBox\_Handle = win\_GetCtlHandle(\_Win,id\_edit),    Array1\_Handle = win\_GetCtlHandle(\_Win,idc\_edit),  Array1\_Text = win\_GetText(Array1\_Handle),  Array1\_Text <> "",    string\_to\_list(Array1\_Text, SymArray),    revers(SymArray,SymArrayRes),  sym\_list\_to\_string(SymArrayRes,StringRes),    lbox\_Clear(LBox\_Handle),  lbox\_Add(LBox\_Handle, StringRes),  !.  %END revers, idc\_ok\_sym \_CtlInfo  dlg\_revers\_eh(\_Win,e\_Control(idc\_ok\_sym,\_CtrlType,\_CtrlWin,\_CtlInfo),0):-!,  dlg\_Error("Ошибка", "Ничего не введено!"),  !. |

Листинг 4.16 – Описание окна «ОБЪЕДИНЕНИЕ ТРЁХ СПИСКОВ В ОДИН В УКАЗАННОМ ПОРЯДКЕ»

|  |
| --- |
| dlg\_объединение\_трёх\_списков\_в\_один\_в\_указанном\_порядке\_eh(\_Win,e\_Control(idc\_ok\_sym,\_CtrlType,\_CtrlWin,\_CtlInfo),0):-  LBox\_Handle = win\_GetCtlHandle(\_Win,idc\_lbox\_res),    Array1\_Handle = win\_GetCtlHandle(\_Win,idc\_edit1),  Array1\_Text = win\_GetText(Array1\_Handle),  string\_to\_list(Array1\_Text, Array1),    Array2\_Handle = win\_GetCtlHandle(\_Win,idc\_edit2),  Array2\_Text = win\_GetText(Array2\_Handle),  string\_to\_list(Array2\_Text, Array2),    Array3\_Handle = win\_GetCtlHandle(\_Win,idc\_edit3),  Array3\_Text = win\_GetText(Array3\_Handle),  string\_to\_list(Array3\_Text, Array3),    Edit1\_Handle = win\_GetCtlHandle(\_Win,idc\_edit\_sym\_first),  Edit1\_Text = win\_GetText(Edit1\_Handle),  str\_int(Edit1\_Text, First),  First>=1,  First<=3,    Edit2\_Handle = win\_GetCtlHandle(\_Win,idc\_edit\_sym\_second),  Edit2\_Text =win\_GetText(Edit2\_Handle),  str\_int(Edit2\_Text, Second),  Second>=1,  Second<=3,    append(First, Second, Array1, Array2, Array3, SymArray3Res),  sym\_list\_to\_string(SymArray3Res, SymArray3ResString),  lbox\_Clear(LBox\_Handle),  lbox\_Add(LBox\_Handle, SymArray3ResString),  !.    dlg\_объединение\_трёх\_списков\_в\_один\_в\_указанном\_порядке\_eh(\_Win,e\_Control(idc\_ok\_sym,\_CtrlType,\_CtrlWin,\_CtlInfo),0):-!,  dlg\_Error("Ошибка", "Ошибка валидации порядка списков"),  !. |

Листинг 4.17 – Описание окна «СОРТИРОВКА ПО УБЫВАНИЮ МЕТОДОМ ВСТАВОК»

|  |
| --- |
| dlg\_сортировка\_по\_убыванию\_методом\_вставок\_eh(\_Win,e\_Control(idc\_ok,\_CtrlType,\_CtrlWin,\_CtlInfo),0):-!,    Array\_Handle = win\_GetCtlHandle(\_Win,idc\_array),  Array\_Text = win\_GetText(Array\_Handle),  string\_to\_list(Array\_Text, Array),    insert\_sort(Array,SymArrayRes),  sym\_list\_to\_string(SymArrayRes,StringRes),    LBox\_Handle = win\_GetCtlHandle(\_Win,id\_edit),  lbox\_Clear(LBox\_Handle),  lbox\_Add(LBox\_Handle, StringRes),  !. |

1. **Программа и методика испытаний**

**5.1** **Наименование испытуемой программы**

Наименование – «Ввод-вывод данных»

**5.2** **Цель испытаний**

Цель проведения испытаний – проверка соответствия характеристик разработанной программы (программного изделия) функциональным и иным, отдельным видам требований, изложенным в программном документе «Техническое задание».

**5.3.** **Требования к программе**

При проведении испытаний функциональные характеристики (возможности) программы подлежат проверке на соответствие требованиям, изложенным в п. «Требования к функциональным характеристикам» Технического задания.

Состав программной документации должен включать в себя:

1. задание;
2. текст программы;
3. описание программы;
4. программу и методики испытаний;

**5.4** **Технические средства, используемые во время испытаний**

Состав используемых во время испытаний технических средств:

• PC совместимый с процессором Intel Core i3-2310M и выше;

• ОЗУ 4096 Мбайт и выше;

• 2048 Мбайт видеопамяти и выше;

• наличие свободного места на жестком диске более 1024 МБайт;

• ОС Windows XP/Vista/Seven;

**5.5** **Программные средства, используемые во время испытаний**

Для проведений испытаний необходимо наличие программы Visual Prolog версии 5.2.

**5.6** **Качественные характеристики, подлежащие оценке**

В ходе проведения испытаний оценке подлежат качественные (функциональные) характеристики программы. Для проверки программы необходимо провести следующие тесты:

a) Проверка на правильность работы программы.

При каждом запуске программа должна выдать точно такой же текст, как и введенный, не должно быть ошибок с кодировкой текста, вне зависимости от количества раз запуска программы.

б) Проверка на надежность системы.

Программа должна быть протестирована на разных компьютерах в течение нескольких дней. В ходе тестирования не должно быть сбоев программы или операционной системы.

**Список использованных источников**

1. Ефимова Е.А. Основы программирования на языке VisualProlog [Электронный ресурс] / Е.А. Ефимова. - 3-е изд. - Электрон. текстовые данные. - М.: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Эр Медиа, 2019. - 265 c. - 978-5-4486-0517-8. - Режим доступа: <http://www.iprbookshop.ru/79710.html>
2. Рублев В.С. Языки логического программирования [Электронный ресурс] / В.С. Рублев. - 2-е изд. - Электрон. текстовые данные. - М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 125 c. - 2227-8397. - Режим доступа: http://www.iprbookshop.ru/73741.html
3. Марков В.Н. Современное логическое программирование на языке Visual Prolog 7.5 [Электронный ресурс] / В.Н. Марков. – М.: BHV, 2016. – 544 с.: ил. - ISBN: 978-5-9775-3487-1. – Режим доступа: <https://www.labirint.ru/books/507843/>
4. Теория и практика логического программирования на языке VisualProlog 7: Учебное пособие для вузов / Цуканова Н.И., Дмитриева Т.А. - М.: Гор. линия-Телеком, 2013. - 232 с.: 60x90 1/16. - (Учебное пособие для высших учебных заведений) ISBN 978-5-9912-0194-0 - Режим доступа: <http://znanium.com/catalog/product/414543>

**ПРИЛОЖЕНИЕ А**

(обязательное)

Результаты работы программы приведены на рисунках 6.1-6.8.

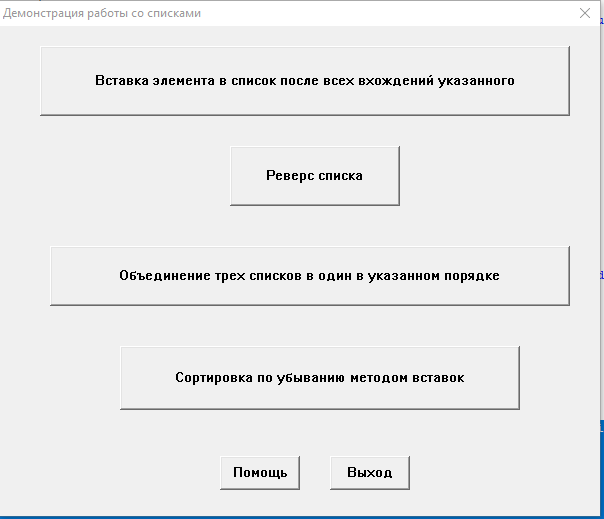


Рисунок 6.1 – Окно «Демонстрация работы со списками» (главное меню)

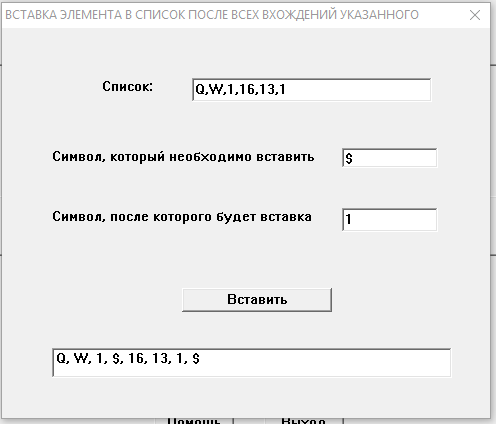


Рисунок 6.2 – Окно «ВСТАВКА ЭЛЕМЕНТА В СПИСОК ПОСЛЕ ВСЕХ ВХОЖДЕНИЙ УКАЗАННОГО»

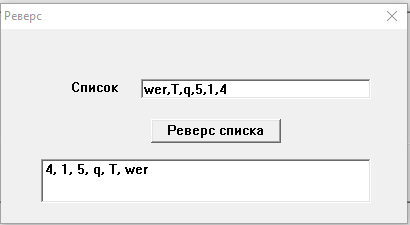


Рисунок 6.3 – Окно «Реверс»

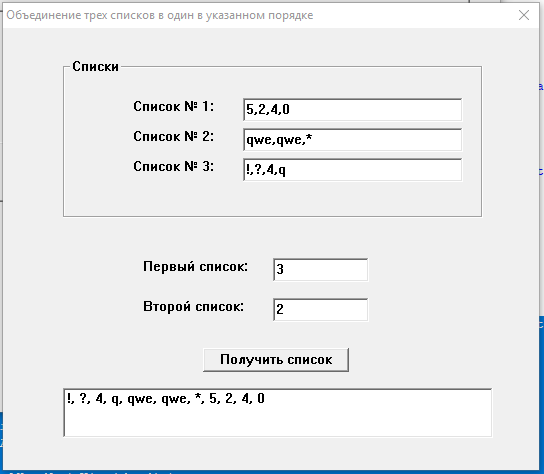


Рисунок 6.4 – Окно «ОБЪЕДИНЕНИЕ ТРЁХ СПИСКОВ В ОДИН В УКАЗАННОМ ПОРЯДКЕ»

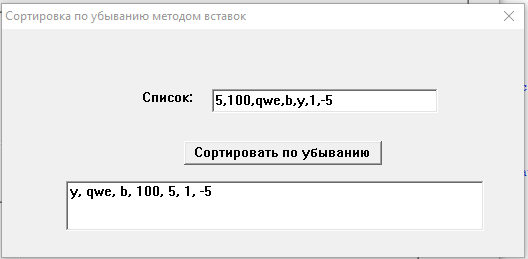


Рисунок 6.5 – Окно «СОРТИРОВКА ПО УБЫВАНИЮ МЕТОДОМ ВСТАВОК»

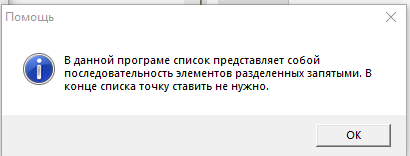


Рисунок 6.6 – Окно «Help»

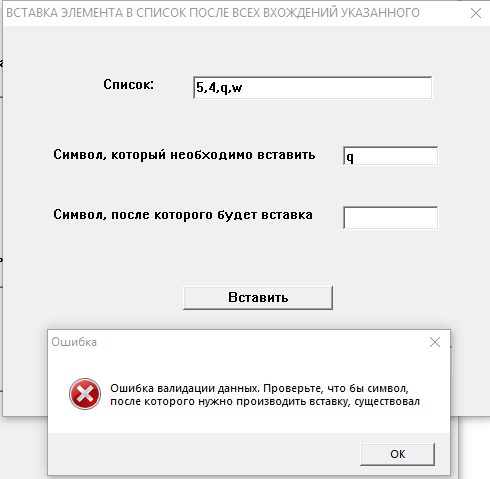


Рисунок 6.7 – Обработка ошибок. Незаполненные поля

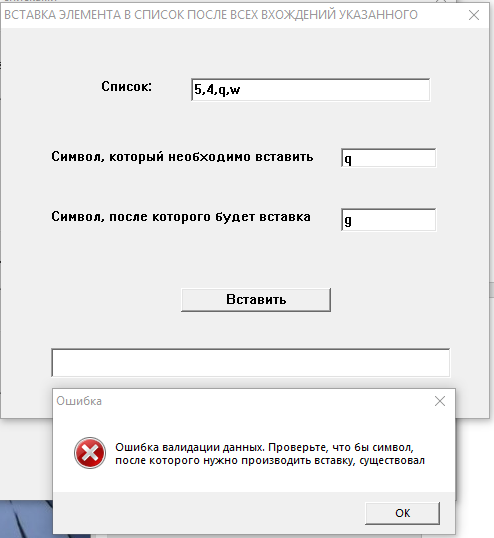


Рисунок 6.8 – Обработка ошибок. Отсутствие символа, после которого будет вставлен элемент

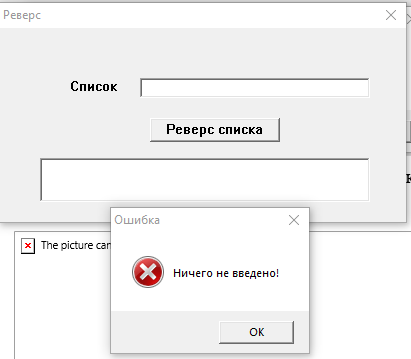


Рисунок 6.9 – Обработка ошибок. Список не введён

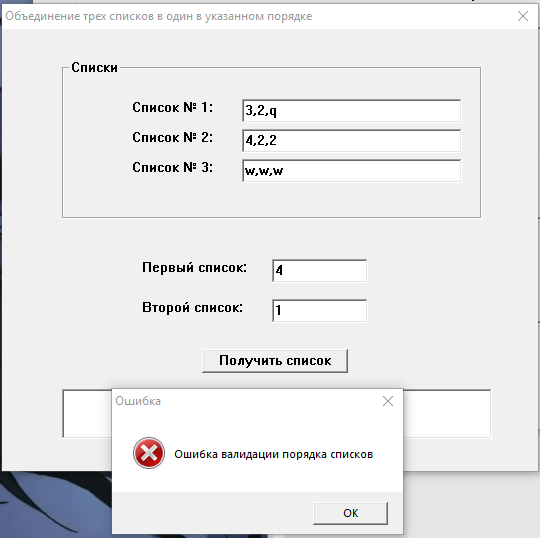


Рисунок 6.10 – Обработка ошибок. Номера списка не существует

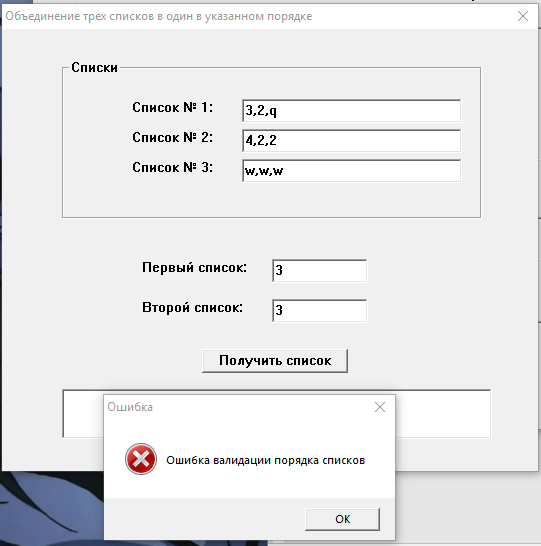


Рисунок 6.11 – Обработка ошибок. Одинаковые номера при выборе списков