1. Лабораторная работа № 8

«Программирование линейных списков»

* 1. Цель работы

Исследование списковых структур данных и приобретение навыков разработки и отладки программ, использующих динамическую память. Исследование особенностей использования переменных ссылочного типа.

## Постановка задачи

Научится использовать стеки, очереди, ссылки.

* 1. Ход выполнения работы

Вариант выполнения задания – 19.

* + 1. Описание алгоритма решения задачи
       1. Определение входных и выходных данных

В меню на вход поступает число от 1 до 8, любое другое число будет проигнорировано. В соответствие с выбранным действием запускается выполнение процедуры.

При создании очереди или добавлении элемента в таблицы требуется ввести Фамилию, Имя, Отчество (строковые типы), оценки по Истории, Физике и Математике (число от 1 до 100 включительно).

На каждое действие пользователя программа реагирует сообщением.

* + - 1. Структурная схема алгоритма

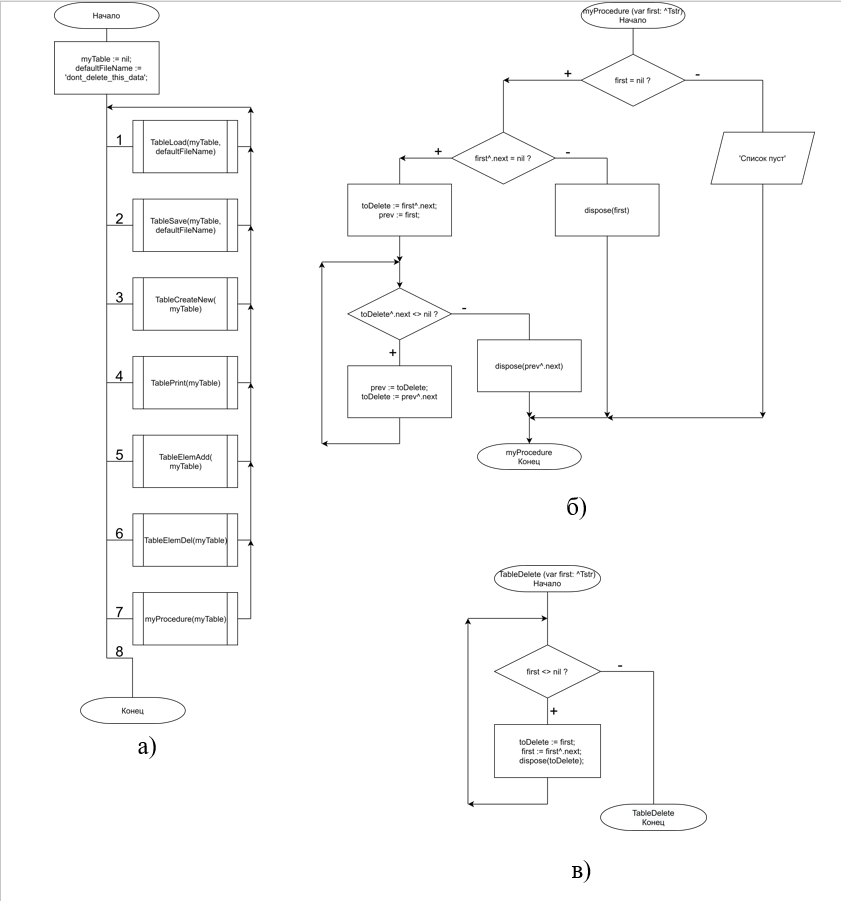


Рисунок 8.1 – Структурные схемы: а) основная программа;

б) процедура myProcedure; в) процедуры TableDelete

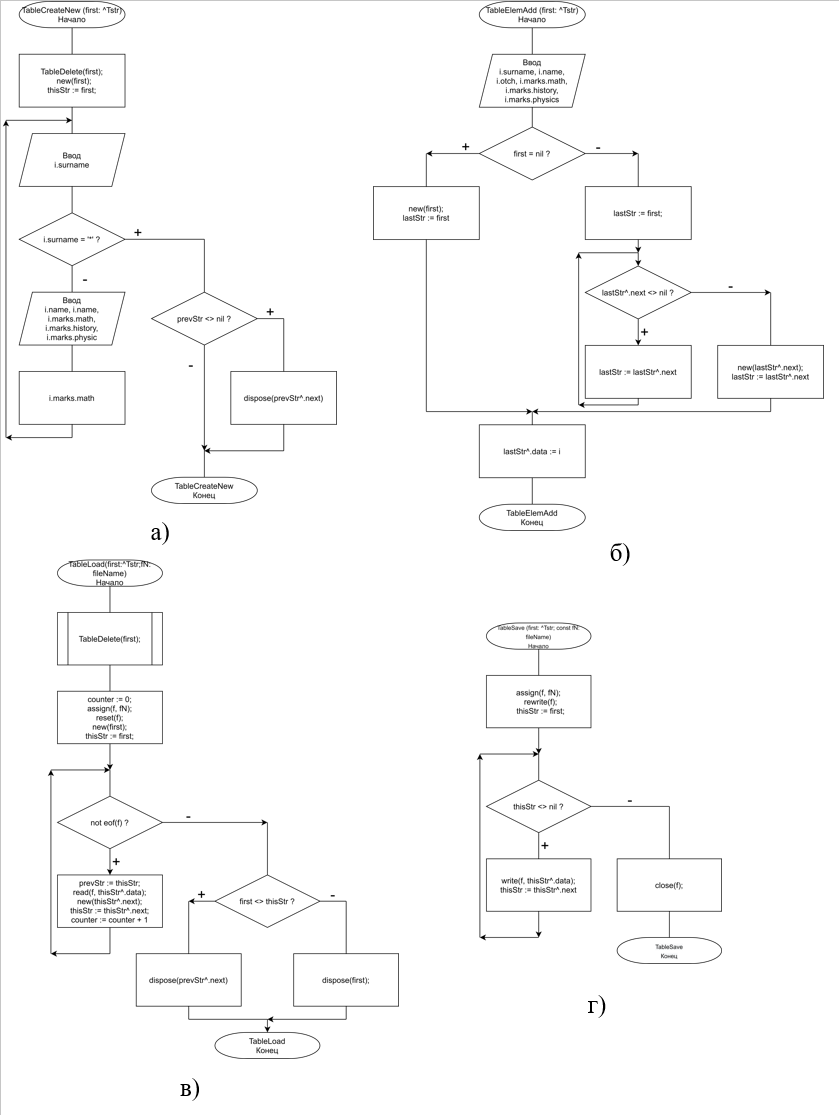


Рисунок 8.2 – Структурные схемы: а) процедура TableCreateNew;

б) процедура TableElementAdd; в) процедуры TableLoad;

г) процедуры TableSave

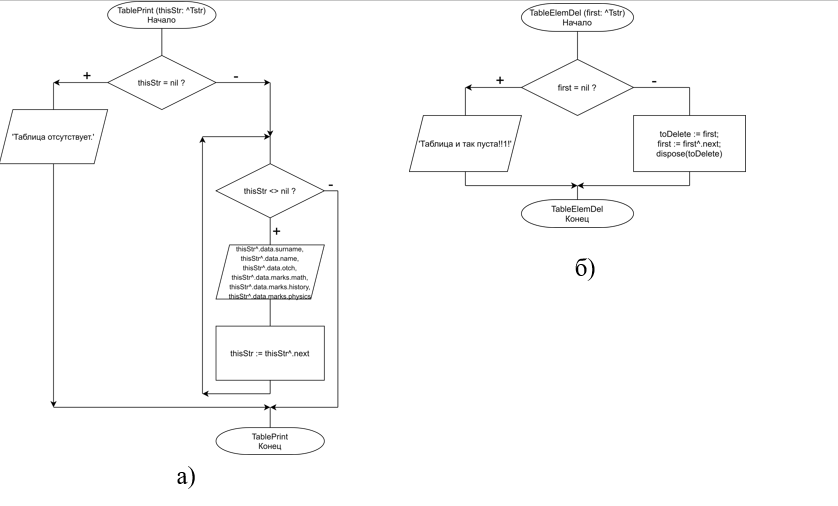


Рисунок 8.3 – Структурные схемы: а) процедура TablePrint;

б) процедура TableElemDel

* + 1. Текст программы

Type

form = record

surname, name, otch: string[64];

marks: record

math, history, physics: 0..100;

end;

end;

Tstr = record

data: form;

next: ^Tstr;

end;

fileName = string[128];

// Удаление списка

procedure TableDelete (var first: ^Tstr);

var toDelete: ^Tstr;

begin

while first <> nil do

begin

toDelete := first; first := first^.next; dispose(toDelete);

end

end;

// Создание таблицы

procedure TableCreateNew (var first: ^Tstr);

var thisStr, prevStr: ^Tstr; i:form;

begin

// Удаление таблицы и создание новой

writeln();

writeln('=========================================');

writeln('Инициализация процедуры создания таблицы.');

writeln('(!) Внимание: таблица в оперативной памяти');

writeln(' была удалена. ');

writeln('=========================================');

writeln();

TableDelete(first);

new(first);

thisStr := first;

i.marks.history := 0;

// Ввод новых значений

while true do

begin

write('Введите Фамилию (\* для выхода): '); readln(i.surname);

if i.surname = '\*' then

begin

if prevStr <> nil then dispose(prevStr^.next);

if first^.data.marks.history = 0 then

begin

dispose(first);

first := nil

end;

break

end;

write('Введите Имя: '); readln(i.name);

write('Введите Отчество: '); readln(i.otch);

write('Введите Оценку: Математика: '); readln(i.marks.math);

write('Введите Оценку: История: '); readln(i.marks.history);

write('Введите Оценку: Физика: '); readln(i.marks.physics);

new(thisStr^.next);

thisStr^.data := i;

prevStr := thisStr;

thisStr := thisStr^.next;

writeln();

writeln('======= Ввод следующей строки =======');

writeln();

end;

end;

// Добавление элемента в список

procedure TableElemAdd (var first: ^Tstr);

var lastStr: ^Tstr; i: form;

begin

writeln();

writeln('=========================================');

writeln('Инициализация процедуры добавления ');

writeln(' строки в конец! ');

writeln('=========================================');

writeln();

// Получение данных от пользователя

write('Введите Фамилию: '); readln(i.surname);

write('Введите Имя: '); readln(i.name);

write('Введите Отчество: '); readln(i.otch);

write('Введите Оценку: Математика: '); readln(i.marks.math);

write('Введите Оценку: История: '); readln(i.marks.history);

write('Введите Оценку: Физика: '); readln(i.marks.physics);

if first = nil then

begin

// Если таблица отсутствует, то создадим новую

new(first);

lastStr := first

end

else

begin

// Если таблица существует, то найдём последний элемент

lastStr := first;

while lastStr^.next <> nil do

lastStr := lastStr^.next;

new(lastStr^.next);

lastStr := lastStr^.next

end;

lastStr^.data := i;

writeln();

writeln('=========================================');

writeln('Операция завершена: строка была успешно ');

writeln(' добавлена! ');

writeln('=========================================');

writeln();

end;

// Исключение первого элемента из списка

procedure TableElemDel (var first: ^Tstr);

var toDelete: ^Tstr;

begin

if first = nil then

begin

writeln();

writeln('=========================================');

writeln('ERROR: Таблица и так пуста!!! ');

writeln('=========================================');

writeln();

end

else

begin

toDelete := first;

first := first^.next;

dispose(toDelete);

writeln();

writeln('=========================================');

writeln('Операция завершена: строка была успешно ');

writeln(' удалена! ');

writeln('=========================================');

writeln();

end;

end;

// Печать списка

procedure TablePrint (thisStr: ^Tstr);

var counter: integer;

begin

if thisStr = nil then

begin

writeln();

writeln('=========================================');

writeln('ERROR: Таблица и так пуста!!! ');

writeln('=========================================');

writeln();

end

else

begin

counter := 0;

while thisStr <> nil do

begin

counter := counter + 1;

writeln(

counter, ' | ',

thisStr^.data.surname:20, ' | ',

thisStr^.data.name:20, ' | ',

thisStr^.data.otch:20, ' | ',

thisStr^.data.marks.math:3, ' | ',

thisStr^.data.marks.history:3, ' | ',

thisStr^.data.marks.physics:3

);

thisStr := thisStr^.next

end;

end;

end;

// Сохранение в файл

procedure TableSave (var first: ^Tstr; const fN: fileName);

var f: file of form;

thisStr: ^Tstr;

begin

assign(f, fN); rewrite(f); thisStr := first;

while thisStr <> nil do

begin

write(f, thisStr^.data); thisStr := thisStr^.next

end;

close(f);

writeln();

writeln('=========================================');

writeln('Операция завершена: таблица была сохранена');

writeln(' и может быть загружена! ');

writeln('=========================================');

writeln();

end;

// Загрузка из файла

procedure TableLoad (var first: ^Tstr; const fN: fileName);

var f: file of form;

thisStr, prevStr: ^Tstr;

counter: integer;

begin

TableDelete(first);

counter := 0;

assign(f, fN);

if not fileExists(fN) then

begin

writeln();

writeln('=========================================');

writeln('ERROR: Файл не возможно прочитать!!! ');

writeln('=========================================');

writeln();

end

else

begin

reset(f); new(first); thisStr := first;

while not eof(f) do

begin

prevStr := thisStr;

read(f, thisStr^.data);

new(thisStr^.next);

thisStr := thisStr^.next;

counter := counter + 1

end;

if first <> thisStr then

dispose(prevStr^.next)

else

dispose(first);

close(f);

writeln();

writeln('=========================================');

writeln('Операция завершена: таблица была загружена!');

writeln('=========================================');

writeln();

end;

end;

// \*new\* Удаление последнего элемента

procedure myProcedure (var first: ^Tstr);

var prev, toDelete: ^Tstr;

begin

if first = nil then

begin

writeln();

writeln('=========================================');

writeln('ERROR: Таблица и так пуста!!! ');

writeln('=========================================');

writeln();

end

else

if first^.next = nil then

dispose(first)

else

begin

toDelete := first^.next;

prev := first;

while toDelete^.next <> nil do

begin

prev := toDelete;

toDelete := prev^.next

end;

dispose(prev^.next)

end;

writeln();

writeln('=========================================');

writeln('Операция завершена: Элемент удалён! ');

writeln('=========================================');

writeln();

end;

var answer: integer;

myTable: ^Tstr;

defaultFileName: fileName;

begin

// ...

myTable := nil;

defaultFileName := 'dont\_delete\_this\_data';

// Основная программа

while true do

begin

writeln();

writeln('=========================================');

writeln('Выберите действие: ');

writeln('1 - Загрузить из файла (файл: ', defaultFileName, '). ');

writeln('2 - Сохранить в файл (файл: ', defaultFileName, '). ');

writeln('3 - Очистить таблицу в памяти и создать новую. ');

writeln('4 - Показать таблицу из памяти. ');

writeln('5 - Добавить в конец. ');

writeln('6 - Удалить первый. ');

writeln('7 - Удалить последний. \* my \* ');

writeln('8 - Выход из программы. ');

write('Ввод: -> '); readln(answer);

case answer of

1: TableLoad(myTable, defaultFileName);

2: TableSave(myTable, defaultFileName);

3: TableCreateNew(myTable);

4: TablePrint(myTable);

5: TableElemAdd(myTable);

6: TableElemDel(myTable);

7: myProcedure(myTable);

8: break;

else writeln('(!) Неверная команда!');

end;

end;

end.

* + 1. Описание тестовых примеров

Тестовые примеры продемонстрированы на Рисунках 8.4-8.9.

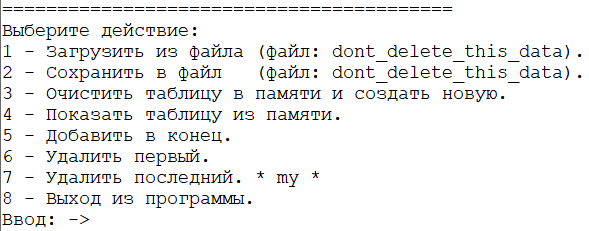


Рисунок 8.4 – Меню программы

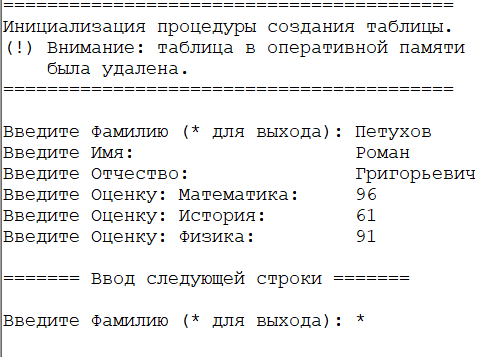


Рисунок 8.5 – Организация очереди

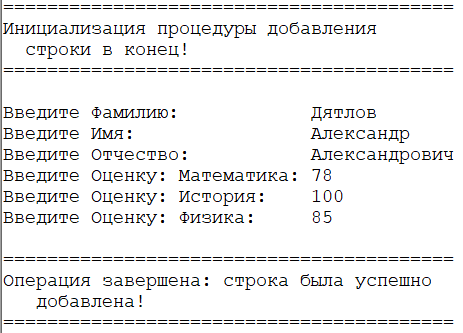


Рисунок 8.6 – Добавление строки в конец

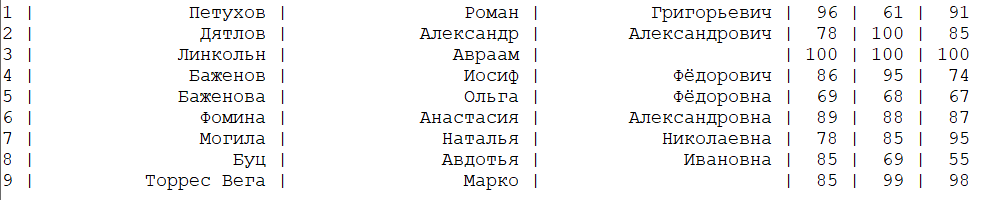


Рисунок 8.7 – Вывод таблицы (таблица заполнена)

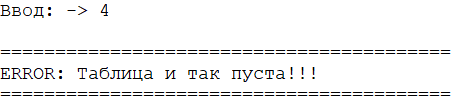


Рисунок 8.8 – Вывод таблицы (таблица пустая)

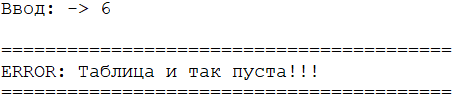


Рисунок 8.9 – Удаление последнего элемента в пустой таблице

На Рисунке 8.4 показано работающее меню программы.

На Рисунке 8.5 показана работоспособность процедуры организации очереди. Были введены данные, а затем ведено ‘\*’, тем самым ввод был завершён. В результате была создана очередь из одного элемента.

На Рисунке 8.6 показана работоспособность процедуры добавления элемента в конец таблицы. В итоге имеется таблица, состоящая из двух элементов.

Было добавлено ещё 7 строк. Содержимое таблицы продемонстрировано на Рисунке 8.7.

Если будет создана пустая таблица, то при попытке её вывода программа выдаст ошибку (Рисунок 8.8).

Если будет создана пустая таблица, то при попытке удаления первой или последней строчки таблицы программа выдаст ошибку (Рисунок 8.9).

## Вывод

При выполнении данной лабораторной работы были получены навыки разработки приложений, умеющих работать со ссылками и списками; написаны процедуры, осуществляющие главные операции над списками, а именно: Организация очереди, Добавление элементов в конец списка, Удаление первого и последнего элемента, Печать списка. Также были повторно закреплены навыки отладки программы, работы с файлами, а именно чтение данных из бинарного типизированного файла и запись в него. Полученные навыки в будущем помогут создавать новые таблицы данных, а также деревья, в том числе и бинарные.