**8 ЛАБОРАТОРНАЯ РАБОТА №8**

**«ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ СПИСКОВ»**

**8.1 Цель работы**

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

**8.2 Вариант задания – 21**

Требуется представить таблицу 8.1 в виде линейного списка L, элементами которого являются строки таблицы. Написать процедуры организации, добавления элемента в список, исключения элемента из списка, удаления списка, просмотра списка, а также процедуру, которая удаляет из непустого списка L последний элемент.

Значения и количество записей в таблице студент выбирает самостоятельно. Исходные данные после организации списка должны сохраняться в файле и при повторном запуске программы считываться из файла. Количество строк таблицы не задается.

**Таблица 8.1 – Анкета**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N | ФИО | Год рождения | Пол | Семейное состояние | Количество детей | Оклад |

**8.3 Порядок выполнения работы**

8.3.1 Разработать алгоритм решения задачи, разбив его на процедуры.

8.3.2 Разработать структурную схему алгоритма решения задачи.

8.3.3 Разработать программу на языке Pascal.

8.3.4 Разработать тестовые примеры.

8.3.5 Выполнить отладку программы.

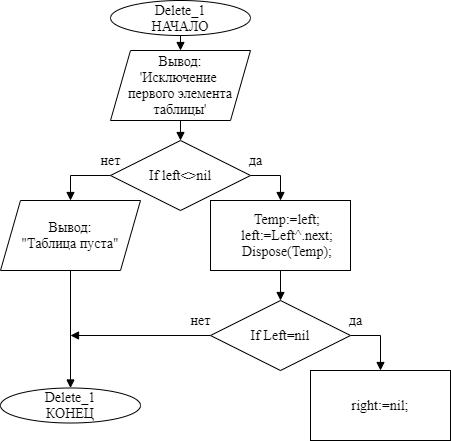
8.3.6 Сделать выводы по проделанной работе.

**8.4 Ход работы**

8.4.1 Для задания был разработан алгоритм решения задачи:

Был подключен модуль «сrt», чтобы организовать удобную работу пользователя с программой. Так же для удобства пользователя в основном блоке программы выводится надпись «что делает программа», а также работа с программой осуществлена с помощью оператора выбора «case», который поможет выбирать нужные действия которые должна выполнять программа. Программа может выполнить 8 задач, 7 из которых – наши процедуры и одна встроенная процедура выхода из программы. Первая задача ­­– это процедура создания таблицы, в которой автоматически удаляется предыдущая если такова имелась. Если вводится символ ‘\*’ вместо фамилии, то считывание строк таблицы заканчивается. Во второй процедуре происходит добавление элемента в таблицу. Третья процедура удаляет первый элемент таблицы. Четвёртая удаляет последний элемент. Пятая процедура выводит таблицу на экран. Шестая процедура сохраняет таблицу в файл, для того чтобы потом можно было вернуться к сохранённому виду таблицы. Седьмая процедура выгружает таблицу из файла для работы с ней.

8.4.2 Разработана структурная схема алгоритма решения задачи и представлена

на рисунках 8.1, 8.2, 8.3, 8.4.

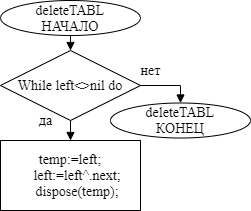


Рисунок 8.1 – Структурные схемы процедур удаления первого элемента таблицы и удаления всей таблицы

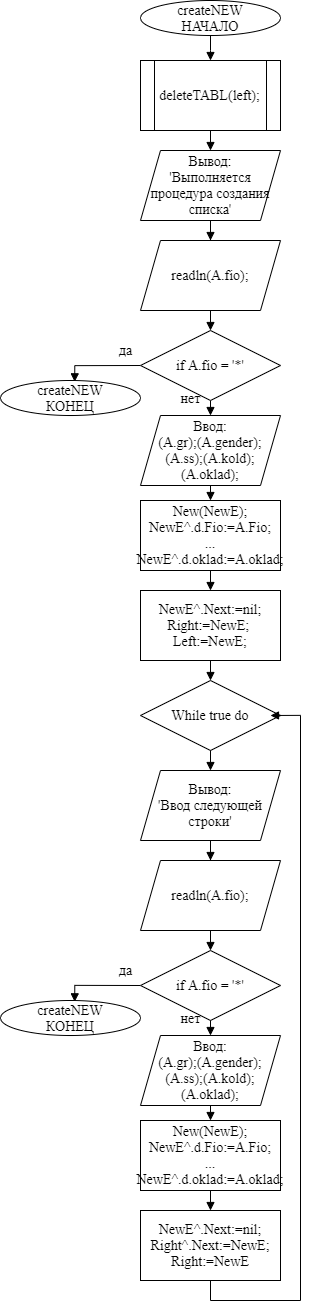
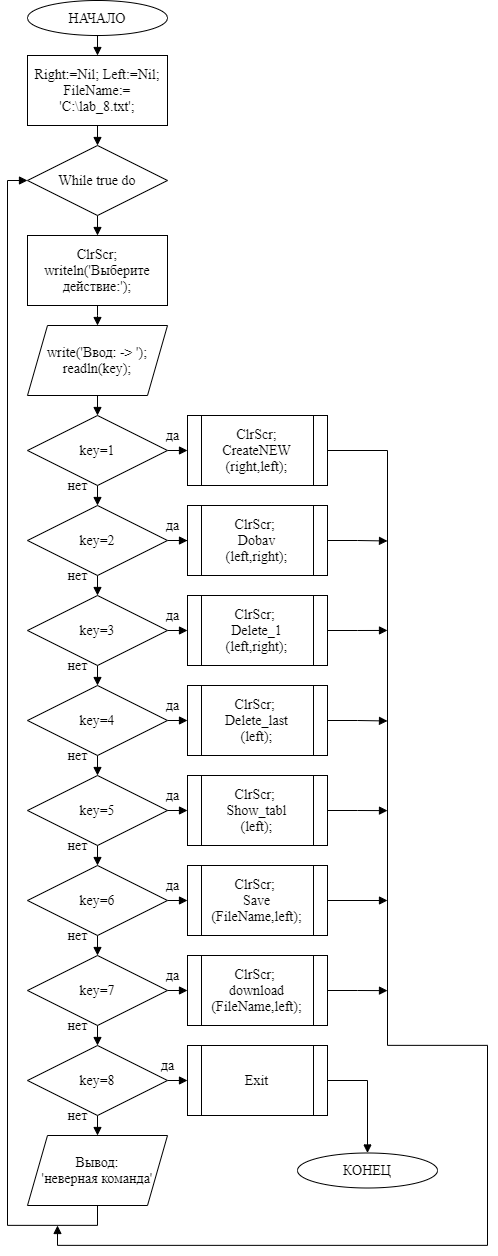
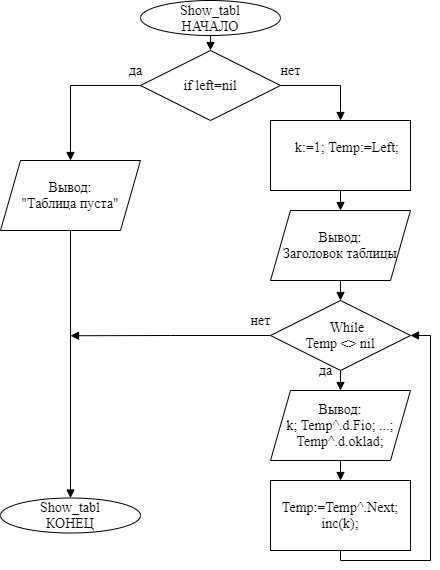
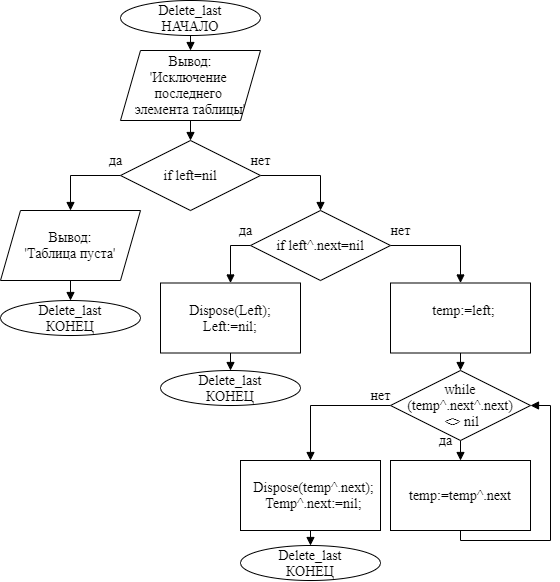


Рисунок 8.2 – Структурные схемы основной программы и процедуры создания таблицы



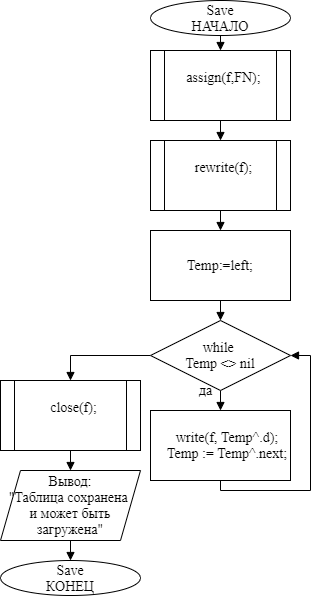
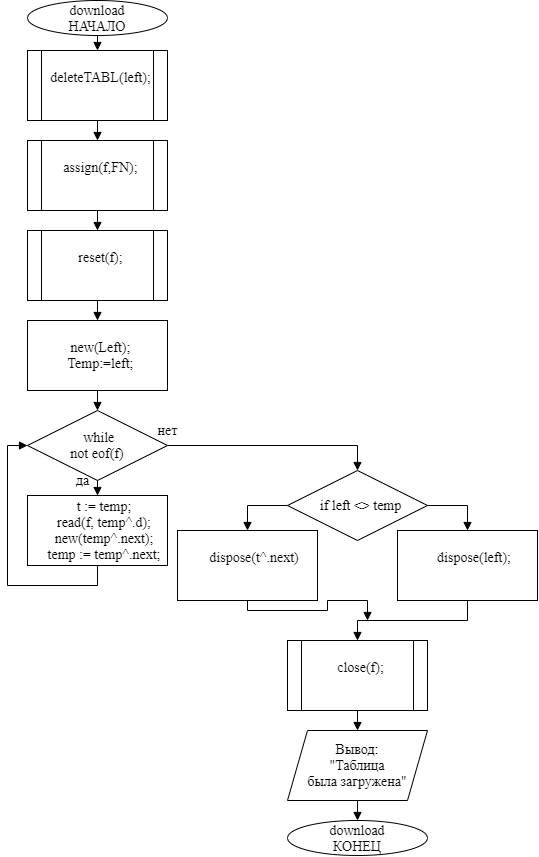


Рисунок 8.3 – Структурные схемы процедур удаления последнего элемента таблицы, демонстрации таблицы и сохранения таблицы в файл



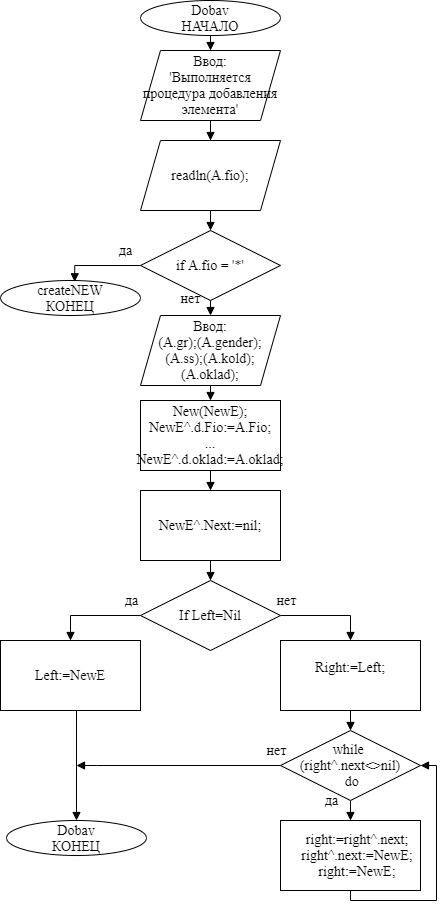


Рисунок 8.4 – Структурные схемы процедур добавления элемента в таблицу и выгрузки таблицы из файла в программу

8.4.3 Написана программа согласно вышеописанного алгоритма.

**Program** lab\_8;

**Uses** Crt;

**Type**

Anketa = **record**

n,gr,kold,oklad:integer;

fio,ss:string[20];

gender:char;

**end**;

Ukaz=^pole;

pole = **record**

d:anketa;

next:Ukaz;

**end**;

File\_name=string;

//Удаление таблицы

//===================================

**procedure** deleteTABL(**var** left:Ukaz);

**Var** temp:ukaz;

**begin**

**while** left <> nil **do**

**begin**

temp:=left;

left:=left^.next;

dispose(temp);

**end**

**end**;

//Создание таблицы

//===========================================================

**Procedure** createNEW (**var** right,left:Ukaz);

**Var** A:anketa;

NewE:Ukaz;

**Begin**

deleteTABL(left); //процедура удаления старой таблицы

Writeln('=========================================');

Writeln('Выполняется процедура создания списка');

Writeln('(+ очистка пред. таблицы если она имеется)');

Writeln('=========================================');

Write('Введите фамилию и инициалы (\* - выход): '); readln(A.fio);

**if** A.fio = '\*' **then exit**;

Write('Введите год рождения: '); readln(A.gr);

Write('Введите пол (м/ж): '); readln(A.gender);

Write('Введите семейное состояние: '); readln(A.ss);

Write('Введите количество детей: '); readln(A.kold);

Write('Введите оклад: '); readln(A.oklad);

**New**(NewE);

NewE^.d.Fio:=A.Fio;

NewE^.d.gr:=A.gr;

NewE^.d.gender:=A.gender;

NewE^.d.ss:=A.ss;

NewE^.d.kold:=A.kold;

NewE^.d.oklad:=A.oklad;

NewE^.Next:=nil;

Right:=NewE; //Right - указатель начала таблицы

Left:=NewE; //Left - указатель конца таблицы

**While** true **do**

**begin**

writeln();

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

writeln();

Write('Введите фамилию и инициалы (\* - выход): '); readln(A.fio);

**if** A.fio = '\*' **then exit**;

Write('Введите год рождения: '); readln(A.gr);

Write('Введите пол (м/ж): '); readln(A.gender);

Write('Введите семейное состояние: '); readln(A.ss);

Write('Введите количество детей: '); readln(A.kold);

Write('Введите оклад: '); readln(A.oklad);

**New**(NewE);

NewE^.d.Fio:=A.Fio;

NewE^.d.gr:=A.gr;

NewE^.d.gender:=A.gender;

NewE^.d.ss:=A.ss;

NewE^.d.kold:=A.kold;

NewE^.d.oklad:=A.oklad;

NewE^.Next:=nil;

Right^.Next:=NewE;

Right:=NewE

**end**;

**end**;

//Добавить в конец таблицы один элемент

//===========================================================

**Procedure** Dobav (**Var** left,right:ukaz);

**Var** A:anketa;

newE:Ukaz;

**Begin**

Writeln('=========================================');

Writeln('Выполняется процедура добавления элемента');

Writeln('=========================================');

Write('Введите фамилию и инициалы (\* - выход): '); readln(A.fio);

**if** A.fio = '\*' **then exit**;

Write('Введите год рождения: '); readln(A.gr);

Write('Введите пол (м/ж): '); readln(A.gender);

Write('Введите семейное состояние: '); readln(A.ss);

Write('Введите количество детей: '); readln(A.kold);

Write('Введите оклад: '); readln(A.oklad);

**New**(NewE);

NewE^.d.Fio:=A.Fio;

NewE^.d.gr:=A.gr;

NewE^.d.gender:=A.gender;

NewE^.d.ss:=A.ss;

NewE^.d.kold:=A.kold;

NewE^.d.oklad:=A.oklad;

NewE^.Next:=nil;

**If** Left=Nil **Then** {Если добавляется первый элемент, то}

Left:=NewE {инициализировать указатель головы}

**Else**

**begin**

Right:=Left;

**while** (right^.next<>nil) **do**

right:=right^.next;

right^.next:=NewE;

right:=NewE;

**end**;

Writeln('Нажмите "Enter" для продолжения'); readln;

**end**;

//Удалить первый элемент таблицы

//===========================================================

**Procedure** Delete\_1(**Var** left,right:ukaz);

**Var** temp:ukaz;

**Begin**

Writeln('Исключение первого элемента таблицы');

**If** left<>nil **then**

**begin**

Temp:=left;

left:=Left^.next;

Dispose(Temp);

//если удалили последний элемент

**If** Left=nil **then** right:=nil;

**end**

**else**

Writeln('--------ERROR: таблица пуста--------');

Writeln('Нажмите "Enter" для продолжения'); readln;

**end**;

//МОЯ ПРОЦЕДУРА

//Удаление последнего элемента таблицы

//===========================================================

**Procedure** Delete\_last (**Var** Left:ukaz);

**Var** Temp:ukaz;

**Begin**

Writeln('Исключение последнего элемента таблицы');

**if** left=nil **then**

**begin**

Writeln('--------ERROR: таблица пуста--------');

Writeln('Нажмите "Enter" для продолжения'); readln; **Exit**;

**end**;

**if** left^.next=nil **then**

**begin**

Dispose(Left);

Left:=nil;

Writeln('Нажмите "Enter" для продолжения'); readln; **Exit**;

**end**;

temp:=left;

**while** (temp^.next^.next) <> nil **do** temp:=temp^.next;

Dispose(temp^.next);

Temp^.next:=nil;

Writeln('Нажмите "Enter" для продолжения'); readln;

**end**;

//Показать таблицу

//===========================================================

**Procedure** Show\_tabl(**const** left:ukaz);

**Var** k:integer;

temp:ukaz;

**Begin**

**if** left=nil **then**

**begin**

writeln();

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

writeln(' ERROR: Таблица пуста ');

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

writeln();

**end**

**else**

**begin**

k:=1; Temp:=Left;

writeln();

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

writeln(' ТАБЛИЦА ');

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

writeln();

writeln(

'№':4, ' | ',

'ФИО':20, ' | ',

'Г.р.':5, ' | ',

'Пол':3, ' | ',

'Семейное состояние':20, ' | ',

'Дети':4, ' | ',

'Оклад':7

);

writeln('------------------------------------------------------');

**While** Temp <> nil **do**

**begin**

writeln(

k:4, ' | ',

Temp^.d.Fio:20, ' | ',

Temp^.d.gr:5, ' | ',

Temp^.d.gender:3, ' | ',

Temp^.d.ss:20, ' | ',

Temp^.d.kold:4, ' | ',

Temp^.d.oklad:7

);

Temp:=Temp^.Next; inc(k);

**end**;

**end**;

Writeln('Нажмите "Enter" для продолжения'); readln;

**end**;

//Сохранить таблицу

//===========================================================

**Procedure** Save(**const** FN:file\_name; **var** Left:ukaz);

**var** temp:ukaz;

f: **file of** anketa;

**Begin**

assign(f,FN);

rewrite(f);

Temp:=left;

**while** Temp <> nil **do**

**begin**

write(f, Temp^.d);

Temp := Temp^.next;

**end**;

close(f);

writeln();

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

writeln(' Таблица была сохранена и может быть загружена ');

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

writeln();

Writeln('Нажмите "Enter" для продолжения'); readln;

**end**;

//Загрузить таблицу

//===========================================================

**Procedure** download(**const** FN:file\_name; **var** Left:ukaz);

**Var** t,temp:Ukaz;

f: **file of** anketa;

**Begin**

deleteTABL(left);

assign(f,FN);

reset(f);

**new**(Left);

Temp:=left;

**while not** eof(f) **do**

**begin**

t := temp;

read(f, temp^.d);

**new**(temp^.next);

temp := temp^.next;

**end**;

**if** left <> temp **then**

dispose(t^.next)

**else**

dispose(left);

close(f);

writeln();

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

writeln(' Таблица была загружена и может быть использована ');

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

writeln();

Writeln('Нажмите "Enter" для продолжения'); readln;

**end**;

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

**Var** key:byte;

NewE,Left,Right,Temp:Ukaz;

A:anketa;

Filename:File\_name;

{============== Основная программа ==============}

**BEGIN**

Right:=Nil; Left:=Nil;

//......................................\\

FileName:='C:\PABCWork.NET\lab\_8.txt';

//......................................\\

**while** true **do**

**begin**

ClrScr;

writeln();

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

writeln('Выберите действие:');

writeln('1 - Создание таблицы (+ удаление старой)');

writeln('2 - Добавление элемента в таблицу');

writeln('3 - Удалить первого');

writeln('4 - Удалить последнего'); //МОЁ

writeln('5 - Показать таблицу');

writeln('6 - Сохранить таблицу в файл');

writeln('7 - Выгрузить таблицу из файла');

writeln('8 - Выход из программы');

write('Ввод: -> '); readln(key);

**case** key **of**

1: **begin** ClrScr; CreateNEW(right,left); **end**;

2: **begin** ClrScr; Dobav(left,right); **end**;

3: **begin** ClrScr; Delete\_1(left,right); **end**;

4: **begin** ClrScr; Delete\_last(left); **end**;

5: **begin** ClrScr; Show\_tabl(left); **end**;

6: **begin** ClrScr; Save(FileName,left); **end**;

7: **begin** ClrScr; download(FileName,left); **end**;

8: **Exit**;

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

**end**;

**end**;

**END**.

8.4.4 Выполнена отладка программы. Результаты тестирования отображены на рисунках 8.5, 8.6, 8,7, 8,8. На рисунках 8.5 и 8.6 видно, как запускается программа, которая предлагает выбрать одну из восьми процедур и мы как сознательные пользователи выбираем первую процедуру и начинаем заполнять анкеты студентов.

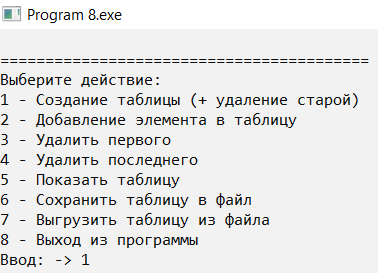


Рисунок 8.5 – Главное меню программы

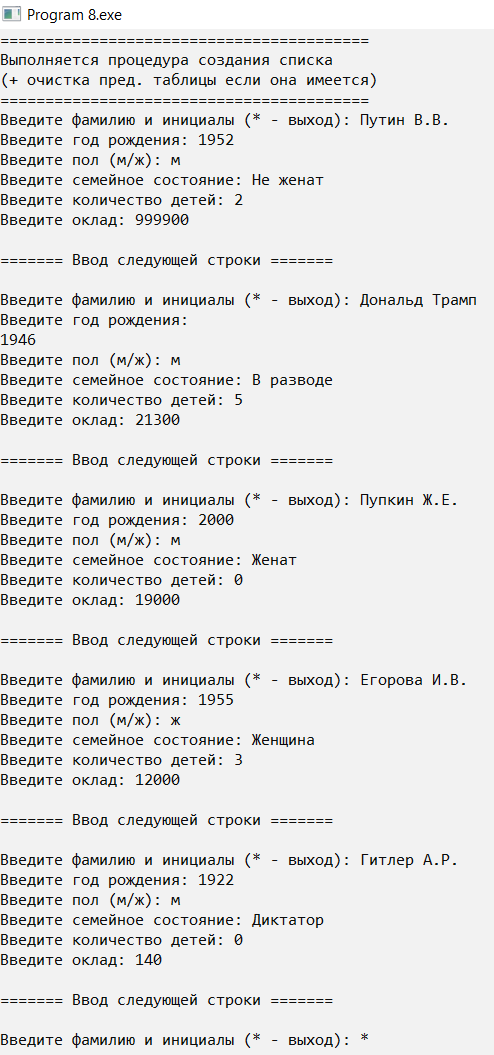
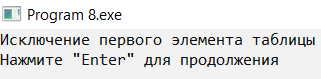
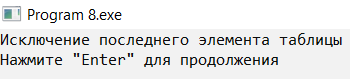


Рисунок 8.6 – Создание таблицы

На рисунках 8.7 и 8.8 мы запускаем процедуры удаления первого и последнего элемента, а на рисунке 8.9 демонстрируем что у нас получилось с помощью пятой процедуры демонстрации таблицы.

  
Рисунок 8.7 – Процедура исключения Рисунок 8.8 – Процедура исключения первого элемента таблицы последнего элемента таблицы

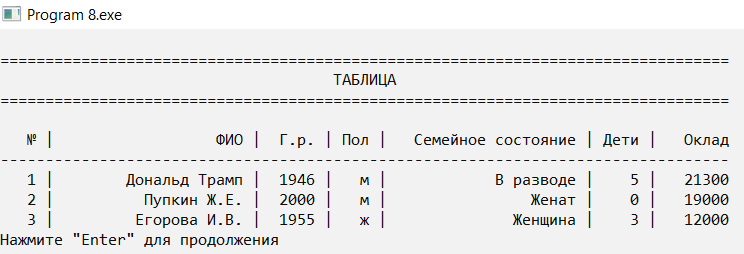


Рисунок 8.9 – Процедура демонстрации таблицы

На рисунке 8.10 продемонстрировано действие процедуры добавления элемента в таблицу. На рисунке 8.11 мы сохраняем таблицу в файл, затем закрываем программу и снова запускаем, на рисунке 8.12 запускаем процедуру выгрузки таблицы из файла и на рисунке 8.13 показываем что всё прошло успешно, за счёт демонстрации таблицы пятой процедурой.

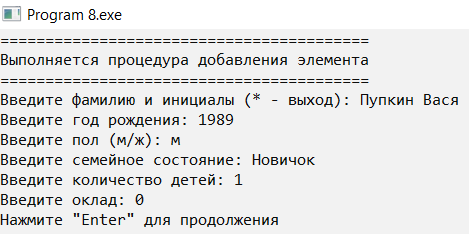


Рисунок 8.10 – Процедура добавления элемента в таблицу

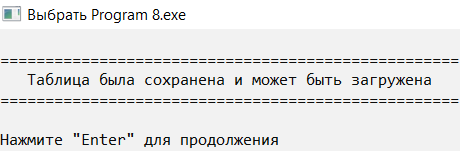
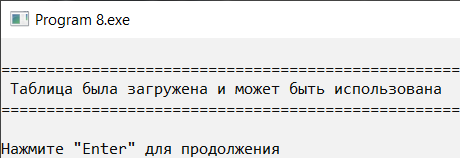
 

Рисунок 8.11 – Процедура сохранения Рисунок 8.12 – Процедура выгрузки таблицы в файл таблицы из файла

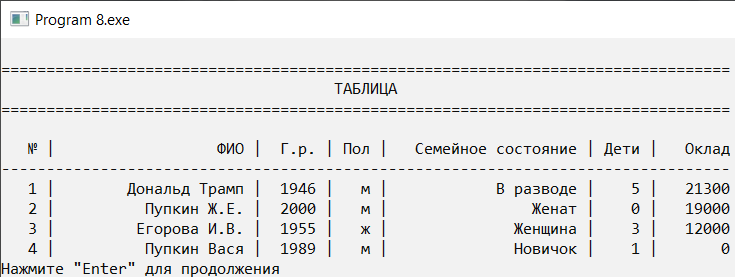


Рисунок 8.12 – Демонстрация таблицы выгруженной из файла

Результаты тестирования полностью соответствуют ожиданиям.

**Выводы**

В ходе выполнения данной лабораторной работы были освоены основные приёмы программирования операций над линейными списками. Был изучен новый тип данных – ссылочный и способы обращения к элементам этого типа. Были подробно исследованы динамические переменные. Изучены специальные переменные с указателями, специальные процедуры для работы с динамической памятью: Dispose, GetMem, FreeMem, New. Полученные во время разработки навыки помогут разрабатывать более сложные программы в будущем.