Министерство образования и науки Российской Федерации

Севастопольский Государственный Университет

Кафедра ИС

ОТЧЕТ

по лабораторной работе №1

ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ

СПИСКОВ

Выполнил: ст. гр. ИТб-11

Куркчи А.Э.

Проверил:

Забаштанский А.К.

Севастополь

2014

# 1. Цель

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

2. Постановка задач

Представить одну из приведенных ниже таблиц в виде линейного списка L, элементами которого являются строки таблицы. Написать процедуры организации, добавления элемента в список, исключения элемента из списка, просмотра списка, а также одну из процедур в соответствии с вариантом, приведенным ниже.

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.

Вариант №2

Таблица 3.2. Процедуру, которая вставляет в начало списка L

новый элемент Е.

3. Структурная схема

На рисунке 1 изображена схема алгоритма работы программы.

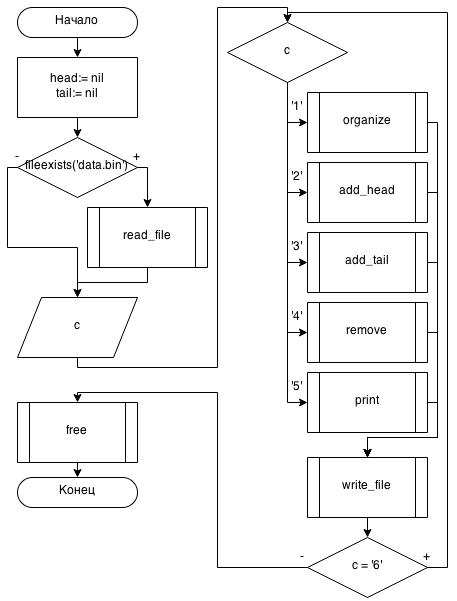
****

Рисунок 1 – структурная схема

На рисунке 2 изображена схема алгоритма работы процедуры free

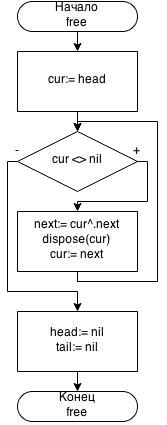


Рисунок 2 – структурная схема процедуры free

На рисунке 3 изображена схема алгоритма работы функции input

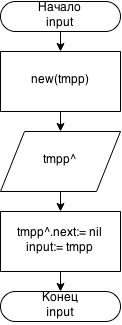


Рисунок 3 – структурная схема функции input

На рисунке 4 изображена схема алгоритма работы процедуры add\_head

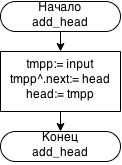


Рисунок 4 – структурная схема процедуры add\_head

На рисунке 5 изображена схема алгоритма работы процедуры add\_tail

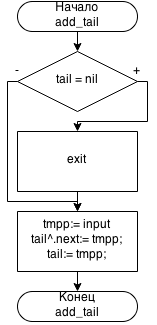


Рисунок 5 – структурная схема процедуры add\_tail

На рисунке 6 изображена схема алгоритма работы процедуры organize

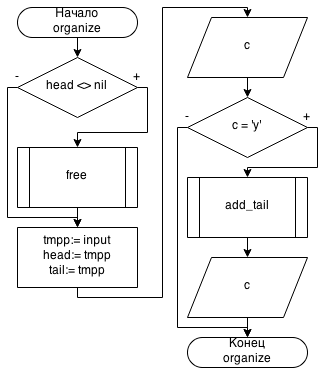


Рисунок 6 – структурная схема процедуры organize

На рисунке 7 изображена схема алгоритма работы процедуры print

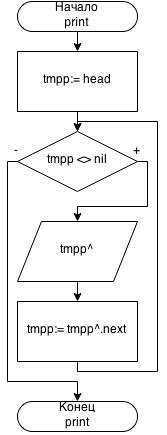


Рисунок 7 – структурная схема процедуры print

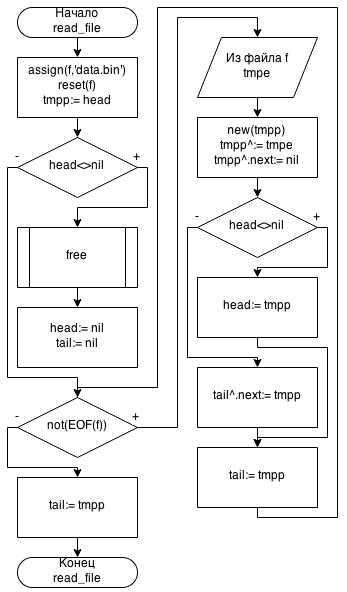
На рисунке 8 изображена схема алгоритма работы процедуры read\_file

Рисунок 8 – структурная схема процедуры read\_file

На рисунке 9 изображена схема алгоритма работы процедуры write\_file

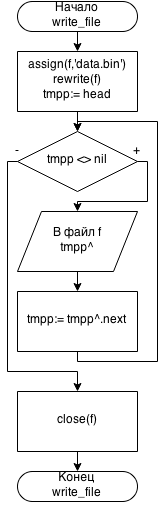


Рисунок 9 – структурная схема процедуры write\_file

На рисунке 10 изображена схема алгоритма работы процедуры remove

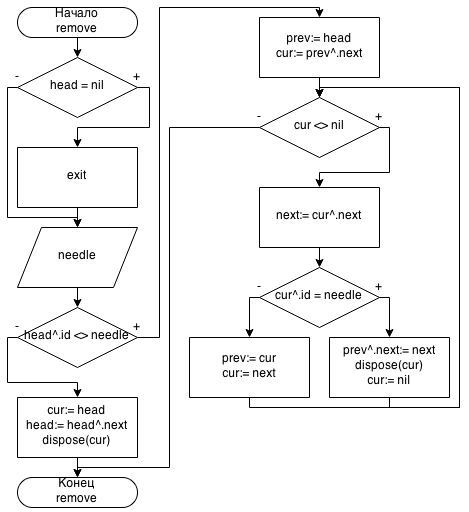


Рисунок 10 – структурная схема процедуры remove

4. Текст программы

**program** lab01;

**uses** crt;

**type** TRowPointer=^TRow;

TRow=**record**

id:integer;

origin,destination:string[16];

time\_departure,time\_arrival:string[5];

price:integer;

next:TRowPointer;

**end**;

**var** head,tail:TRowPointer;

**procedure** free;

**var** cur,next:TRowPointer;

**begin**

cur:= head;

**while**(cur <> nil) **do begin**

next:= cur^.next;

dispose(cur);

cur:= next;

**end**;

head:= nil;

tail:= nil;

**end**;

**procedure** read\_file;

**var** f:**file of** TRow;

tmpp:TRowPointer;

tmpe:TRow;

**begin**

assign(f,'data.bin');

reset(f);

**if**(head <> nil) **then**

free;

head:= nil;

tail:= nil;

**while**(**not**(EOF(f))) **do begin**

read(f,tmpe);

**new**(tmpp);

tmpp^:=tmpe;

tmpp^.next:= nil;

**if**(head = nil) **then begin**

head:= tmpp;

**end else begin**

tail^.next:= tmpp;

**end**;

tail:= tmpp;

**end**;

close(f);

**end**;

**procedure** write\_file;

**var** f:**file of** TRow;

tmpp:TRowPointer;

**begin**

assign(f,'data.bin');

rewrite(f);

tmpp:= head;

**while**(tmpp <> nil) **do begin**

write(f,tmpp^);

tmpp:= tmpp^.next;

**end**;

close(f);

**end**;

**function** input:TRowPointer;

**var** tmpp:TRowPointer;

**begin**

**new**(tmpp);

write('Введите номер поезда: ');

readln(tmpp^.id);

write('Введите пункт отправления: ');

readln(tmpp^.origin);

write('Введите пункт назначения: ');

readln(tmpp^.destination);

write('Введите время отправления: ');

readln(tmpp^.time\_departure);

write('Введите время прибытия: ');

readln(tmpp^.time\_arrival);

write('Введите стоимость билета: ');

readln(tmpp^.price);

tmpp^.next:= nil;

input:= tmpp;

**end**;

**procedure** add\_head;

**var** tmpp:TRowPointer;

**begin**

writeln('Добавление элемента в начало');

tmpp:= input;

tmpp^.next:= head;

head:= tmpp;

writeln('Элемент добавлен');

writeln('Для продолжения нажмите клавишу Enter');

readln;

**end**;

**procedure** add\_tail;

**var** tmpp:TRowPointer;

**begin**

**if**(tail = nil) **then begin**

writeln('Рассписание должно быть организованно');

**exit**;

**end**;

writeln('Добавление элемента в конец');

tmpp:= input;

tail^.next:= tmpp;

tail:= tmpp;

writeln('Элемент добавлен');

writeln('Для продолжения нажмите клавишу Enter');

readln;

**end**;

**procedure** organize;

**var** tmpp:TRowPointer;

c:char;

**begin**

**if**(head <> nil) **then**

free;

writeln('Организация рассписания');

tmpp:= input;

head:= tmpp;

tail:=tmpp;

writeln('Первый элемент добавлен');

writeln('Продолжить? (y/n)');

readln(c);

**while**(c = 'y') **do begin**

clrscr;

add\_tail;

writeln('Продолжить? (y/n)');

readln(c);

**end**;

**end**;

**procedure** remove;

**var** prev,cur,next:TRowPointer;

needle:integer;

**begin**

**if**(head = nil) **then begin**

writeln('Рассписание должно быть организованно');

**exit**;

**end**;

write('Введите номер поезда для удаления: ');

readln(needle);

**if**(head^.id = needle) **then begin**

cur:= head;

head:= head^.next;

dispose(cur);

**end else begin**

prev:= head;

cur:= prev^.next;

**while**(cur <> nil) **do begin**

next:= cur^.next;

**if**(cur^.id = needle) **then begin**

prev^.next:= next;

dispose(cur);

cur:= nil;

**end else begin**

prev:= cur;

cur:= next;

**end**;

**end**;

**end**;

writeln('Элемент удалён из рассписания');

writeln('Для продолжения нажмите клавишу Enter');

readln;

**end**;

**procedure** print;

**var** tmpp:TRowPointer;

**begin**

tmpp:= head;

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

writeln('| Номер | Пункт отправления | Пункт назначения | Отправление | Прибытие | Цена |');

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

**while**(tmpp <> nil) **do begin**

writeln('|',tmpp^.id:7,'|',tmpp^.origin:19,'|',tmpp^.destination:19,'|',tmpp^.time\_departure:13,'|',tmpp^.time\_arrival:13,'|',tmpp^.price:10,'|');

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

tmpp:= tmpp^.next;

**end**;

writeln('Для продолжения нажмите клавишу Enter');

readln;

**end**;

**var** c:char;

**begin**

head:= nil;

tail:= nil;

**if**(fileexists('data.bin')) **then**

read\_file;

**repeat**

clrscr;

writeln('Выберите действие:');

writeln('1 - Организация');

writeln('2 - Добавление в начало');

writeln('3 - Добавление в конец');

writeln('4 - Удаление элемента');

writeln('5 - Печать рассписания');

writeln('6 - Выход');

readln(c);

clrscr;

**case** c **of**

'1':organize;

'2':add\_head;

'3':add\_tail;

'4':remove;

'5':print;

**end**;

write\_file;

**until** c = '6';

free;

**end**.

5. Тестовые примеры

На рисунке 11 представлено меню программы

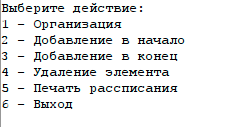


Рисунок 11 – меню программы

На рисунке 12 представлен результат работы пункта меню 5 «Печать расписания»

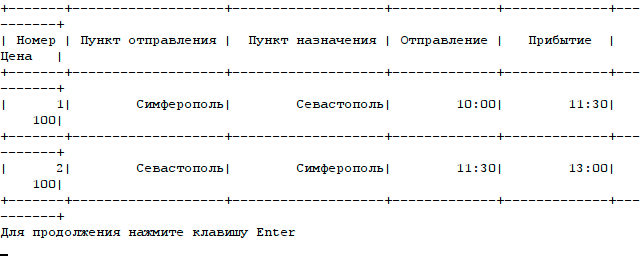


Рисунок 12 – расписание

Вывод

В ходе лабораторной работы были исследованы списковые структуры данных и приобретены навыки разработки и отладки программ, использующих динамическую память. Были исследованы особенности использования переменных ссылочного типа.