Министерство образования и науки Российской Федерации

Севастопольский Государственный Университет

Кафедра ИС

ОТЧЕТ

по лабораторной работе №7

ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ

СПИСКОВ НА ЯЗЫКЕ C/C++

Выполнил: ст. гр. ИТб-11

Куркчи А.Э.

Проверил:

Забаштанский А.К.

Севастополь

2015

# 1. Цель

Изучение списковых структур данных и приобретение навыков разработки и отладки программ, использующих динамическую память. Исследование особенностей организации списков средствами языка С/С++.

2. Постановка задач

Представить одну из приведенных ниже таблиц в виде линейного списка L, элементами которого являются строки таблицы. Написать функции организации, добавления элемента в список, исключения элемента из списка, просмотра списка, а также одну из функций 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

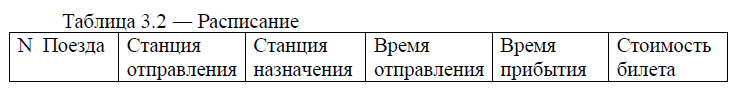


Таблица 3.2. Функцию, которая вставляет в начало очереди новый элемент.

3. Структурная схема

На рисунке 1 изображена схема алгоритма работы программы.

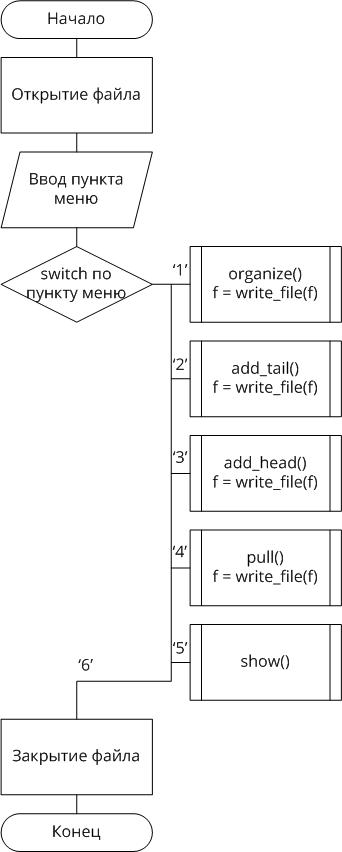
****

Рисунок 1 – Структурная схема

На рисунке 2 изображена схема алгоритма работы подпрограммы add\_tail.

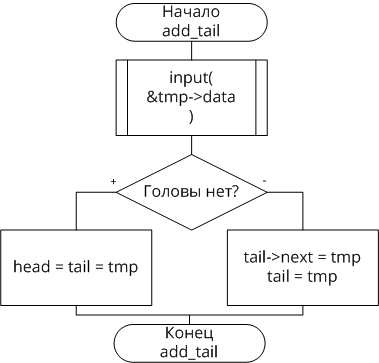
****

Рисунок 2 – Структурная схема add\_tail

На рисунке 3 изображена схема алгоритма работы подпрограммы add\_head.

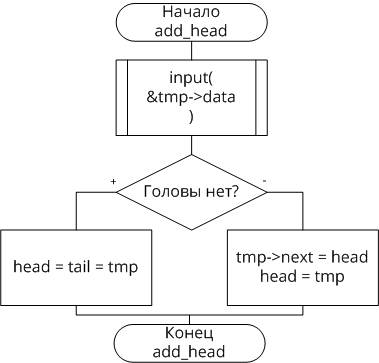
****

Рисунок 3 – Структурная схема add\_head

На рисунке 4 изображена схема алгоритма работы подпрограммы input.

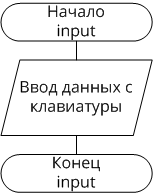
****

Рисунок 4 – Структурная схема input

На рисунке 5 изображена схема алгоритма работы подпрограммы organize.

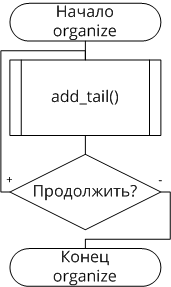
****

Рисунок 5 – Структурная схема organize

На рисунке 6 изображена схема алгоритма работы подпрограммы show.

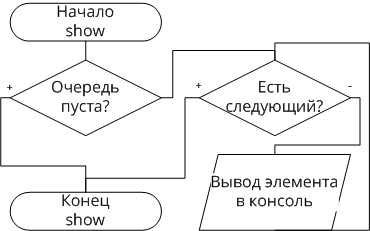
****

Рисунок 6 – Структурная схема show

На рисунке 7 изображена схема алгоритма работы подпрограммы read\_file.

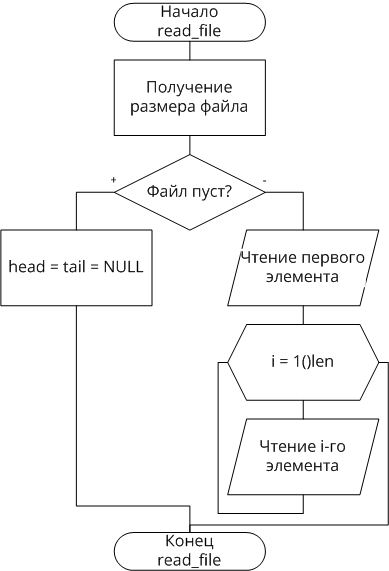
****

Рисунок 7 – Структурная схема read\_file

На рисунке 8 изображена схема алгоритма работы подпрограммы write\_file.

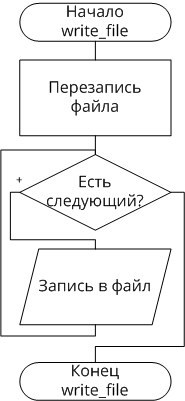
****

Рисунок 8 – Структурная схема write\_file

На рисунке 9 изображена схема алгоритма работы подпрограммы pull.

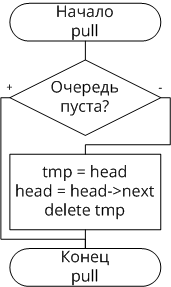
****

Рисунок 9 – Структурная схема pull

4. Текст программы

#include <stdio.h>

#include <iostream>

#include <iomanip>

#define STR\_SIZE 12

#define TIME\_SIZE 11

**using** **namespace** std**;**

struct data\_t **{**

int id**;**

char from**[**STR\_SIZE**];**

char to**[**STR\_SIZE**];**

char dep**[**TIME\_SIZE**];**

char ar**[**TIME\_SIZE**];**

float price**;**

**};**

struct list\_t **{**

data\_t data**;**

list\_t **\***next**;**

**};**

const int size\_d **=** **sizeof(**data\_t**);**

list\_t **\***head**,\***tail**;**

// Чтение файла

void read\_file**(**FILE **\***f**)** **{**

fseek**(**f**,**0**,**SEEK\_END**);**

long len **=** ftell**(**f**)/**size\_d**;**

rewind**(**f**);**

**if(**len **==** 0**)** **{**

head **=** tail **=** **NULL;**

**}** **else** **{**

list\_t **\***tmp**;**

tmp **=** **new** list\_t**;**

fread**(&**tmp**->**data**,**size\_d**,**1**,**f**);**

tmp**->**next **=** **NULL;**

head **=** tail **=** tmp**;**

**for(**int i**=**1**;**i**<**len**;**i**++)** **{**

tmp **=** **new** list\_t**;**

fread**(&**tmp**->**data**,**size\_d**,**1**,**f**);**

tmp**->**next **=** **NULL;**

tail**->**next **=** tmp**;**

tail **=** tmp**;**

**}**

**}**

**}**

// Запись файла

FILE**\*** write\_file**(**FILE **\***f**)** **{**

fclose**(**f**);**

f **=** fopen**(**"data.bin"**,**"w+b"**);**

list\_t **\***tmp **=** head**;**

**while(**tmp **!=** **NULL)** **{**

fwrite**(&**tmp**->**data**,**size\_d**,**1**,**f**);**

tmp **=** tmp**->**next**;**

**}**

**return** f**;**

**}**

// Ввод данных с клавиатуры

void input**(**data\_t **\***data**)** **{**

cout **<<** "Train number:" **<<** endl**;**

cin **>>** data**->**id**;**

cin**.**get**();**

cout **<<** "Departure point:" **<<** endl**;**

cin**.**getline**(**data**->**from**,**STR\_SIZE**);**

cout **<<** "Arrival point:" **<<** endl**;**

cin**.**getline**(**data**->**to**,**STR\_SIZE**);**

cout **<<** "Departure time:" **<<** endl**;**

cin**.**getline**(**data**->**dep**,**TIME\_SIZE**);**

cout **<<** "Arrival time:" **<<** endl**;**

cin**.**getline**(**data**->**ar**,**TIME\_SIZE**);**

cout **<<** "Ticket price:" **<<** endl**;**

cin **>>** data**->**price**;**

cin**.**get**();**

**}**

// Добавить элемент в хвост

void add\_tail**()** **{**

list\_t **\***tmp **=** **new** list\_t**;**

input**(&**tmp**->**data**);**

tmp**->**next **=** **NULL;**

**if(**tail **==** **NULL)** **{**

head **=** tail **=** tmp**;**

**}** **else** **{**

tail**->**next **=** tmp**;**

tail **=** tmp**;**

**}**

**}**

// Добавить элемент в голову

void add\_head**()** **{**

list\_t **\***tmp **=** **new** list\_t**;**

input**(&**tmp**->**data**);**

tmp**->**next **=** head**;**

**if(**head **==** **NULL)** **{**

head **=** tail **=** tmp**;**

**}** **else** **{**

head **=** tmp**;**

**}**

**}**

// Организация очереди

void organize**()** **{**

char c**;**

**do** **{**

add\_tail**();**

cout **<<** "Continue oranization? (y/n): "**;**

cin **>>** c**;**

cin**.**get**();**

**}** **while(**c **==** 'y'**);**

**}**

// Удаление с головы

void pull**()** **{**

**if(**head **==** **NULL)** **{**

cout **<<** "Queue is empty" **<<** endl**;**

**}** **else** **{**

list\_t **\***tmp **=** head**;**

head **=** head**->**next**;**

**if(**head **==** **NULL)** **{**

tail **=** **NULL;**

**}**

**delete** tmp**;**

cout **<<** "Train removed" **<<** endl**;**

**}**

cout **<<** "Press enter to continue..." **<<** endl**;**

cin**.**get**();**

**}**

// Печать очереди

void show**()** **{**

**if(**head **==** **NULL)** **{**

cout **<<** "Queue is empty" **<<** endl**;**

**}** **else** **{**

cout **<<** "+---+------------+------------+-----------+-----------+---------+" **<<** endl**;**

cout **<<** "| # | From | To | Departure | Arrival | Price |" **<<** endl**;**

cout**.**precision**(**2**);**

list\_t **\***tmp **=** head**;**

**while(**tmp **!=** **NULL)** **{**

cout **<<** "+---+------------+------------+-----------+-----------+---------+" **<<** endl**;**

cout **<<** "|" **<<** setw**(**3**)** **<<** tmp**->**data**.**id **<<** "|" **<<** setw**(**12**)** **<<** tmp**->**data**.**from **<<** "|" **<<** setw**(**12**)** **<<** tmp**->**data**.**to **<<** "|" **<<** setw**(**11**)** **<<** tmp**->**data**.**dep **<<** "|" **<<** setw**(**11**)** **<<** tmp**->**data**.**ar **<<** "|" **<<** setw**(**9**)** **<<** tmp**->**data**.**price **<<** "|" **<<** endl**;**

tmp **=** tmp**->**next**;**

**}**

cout **<<** "+---+------------+------------+-----------+-----------+---------+" **<<** endl**;**

**}**

cout **<<** "Press enter to continue..." **<<** endl**;**

cin**.**get**();**

**}**

int main**(**int argc**,** char**\*\*** argv**)** **{**

FILE **\***f**;**

f **=** fopen**(**"data.bin"**,**"r+b"**);**

**if(!**f**)** **{**

f **=** fopen**(**"data.bin"**,**"w+b"**);**

**if(!**f**)** **{**

cout **<<** "Can`t open (create) file" **<<** endl**;**

**return** 1**;**

**}**

**}**

read\_file**(**f**);**

char c**;**

**while(**1**)** **{**

cout **<<** "1 - Organize" **<<** endl**;**

cout **<<** "2 - Add in tail" **<<** endl**;**

cout **<<** "3 - Add in head" **<<** endl**;**

cout **<<** "4 - Pull" **<<** endl**;**

cout **<<** "5 - Show" **<<** endl**;**

cout **<<** "6 - Exit" **<<** endl**;**

cout **<<** "Choose your destiny: "**;**

cin **>>** c**;**

cin**.**get**();**

**switch(**c**)** **{**

**case** '1'**:**

organize**();**

f **=** write\_file**(**f**);**

**break;**

**case** '2'**:**

add\_tail**();**

f **=** write\_file**(**f**);**

**break;**

**case** '3'**:**

add\_head**();**

f **=** write\_file**(**f**);**

**break;**

**case** '4'**:**

pull**();**

f **=** write\_file**(**f**);**

**break;**

**case** '5'**:**

show**();**

**break;**

**case** '6'**:**

fclose**(**f**);**

**return** 0**;**

**}**

**}**

fclose**(**f**);**

**return** 0**;**

**}**

5. Тестовые примеры

На рисунке 10 представлено меню программы

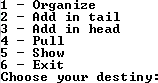


Рисунок 10 – Меню программы

На рисунке 11 представлен вывод студентов в таблице

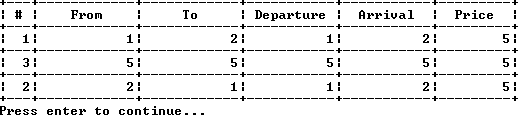


Рисунок 11 – Вывод студентов

Вывод

В ходе лабораторной работы были изучены списковые структуры данных и приобретены навыки разработки и отладки программ, использующих динамическую память. Исследованы особенности организации списков средствами языка C/C++.