Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

по лабораторной работе № 10

«Программирование линейных

списков на языке С/С++»

Выполнил

ст. гр. И12д

Серегин А.В.

Проверил:

асс. Забаштанский А.К.

Севастополь

2015

1. Цель работы

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

2. Варианты заданий

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

Таблица 1 - Расписание

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № Поезда | Станция отправки | Станция назначения | Время отправки | Время прибытия | Стоимость билета |

Вариант 20: Функцию, которая удалят из непустого списка L первый элемент.

3. Структурная схема

На рисунках ниже представлены структурные схемы алгоритма программы.



а) б)

Рисунок 1 – Функции ввода-вывода:

a) ввод данных;

б) вывод данных.



Рисунок 2 – Функция AddTrain.



Рисунок 3 – Функция PrintTrainList.



Рисунок 4 – Функция SaveFile.



Рисунок 5 – Функция FreeMem.



Рисунок 6 – Функция LoadFile.



Рисунок 7 – Функция PopTrain.



Рисунок 8 – Функция main.

4. Код программы

#include <iostream>

#include <iomanip>

#include <fstream>

#include <conio.h>

using namespace std;

struct Time

{

int hour;

int minute;

};

//запись о поезде

struct train

{

int number; //номер поезда

char StPoint[8]; //место отправления

char EndPoint[8]; //место прибытия

Time StTime; //время отправления

Time EndTime; //время прибытия

unsigned int Price; //стоимость билета

};

//элемент списка поезда

struct trainlist

{

train key;

trainlist\* next;

};

//считывание данных поезда

train ScanTrain()

{

train Train;

cout << "Введите номер поезда:" << endl;

cin >> Train.number;

cout << "Введите место отправки:" << endl;

cin >> Train.StPoint;

cout << "Введите место прибытия:" << endl;

cin >> Train.EndPoint;

cout << "Введите время отправки:" << endl;

cin >> Train.StTime.hour >> Train.StTime.minute;

cout << "Введите время прибытия:" << endl;

cin >> Train.EndTime.hour >> Train.EndTime.minute;

cout << "Введите стоимость билета:" << endl;

cin >> Train.Price;

return Train;

}

//вывод данных о поезде

void PrintTrain(const train& Train)

{

cout.setf(ios::left);

cout << setw(3) << Train.number;

cout << setw(9) << Train.StPoint;

cout << setw(9) << Train.EndPoint;

cout << setw(2) << Train.StTime.hour << ':' << setw(4) << Train.StTime.minute;

cout << setw(2) << Train.EndTime.hour << ':' << setw(5) << Train.EndTime.minute;

cout << Train.Price << endl;

}

//добавление нового поезда в конец списка

void AddTrain(trainlist\*\* First, trainlist\*\* Last, train NewTrain)

{

if (!\*First)

{

\*First = new trainlist;

(\*First)->key = NewTrain;

(\*First)->next = NULL;

\*Last = \*First;

return;

}

(\*Last)->next = new trainlist;

\*Last = (\*Last)->next;

(\*Last)->key = NewTrain;

(\*Last)->next = NULL;

}

//вывод списка поездов

void PrintTrainList(trainlist\* Train)

{

if (!Train)

{

cout << "Список пуст" << endl;

return;

}

while (Train)

{

PrintTrain(Train->key);

Train = Train->next;

}

}

//удаление первого эл-та списка

void PopTrain(trainlist\*\* FirstTrain)

{

if (!\*FirstTrain)

{

cout << "Поездов нет." << endl;

return;

}

trainlist\* temp;

temp = \*FirstTrain;

\*FirstTrain = (\*FirstTrain)->next;

cout << "Поезд " << temp->key.StPoint << '-'

<< temp->key.EndPoint << " отправлен." << endl;

delete temp;

}

//очистка памяти

void FreeMem(trainlist\*\* FirstTrain)

{

trainlist\* temp;

while (\*FirstTrain)

{

temp = \*FirstTrain;

\*FirstTrain = (\*FirstTrain)->next;

delete temp;

}

}

//загрузка файла и его данных

void LoadFile(fstream& f,trainlist\*\* First, trainlist\*\* Last)

{

f.open("train.dat", ios::in | ios::binary);

if (!f) return;

train temp;

while (!f.eof())

{

if(f.read(reinterpret\_cast<char\*>(&temp), sizeof(train)))

AddTrain(&(\*First), &(\*Last), temp);

}

f.close();

}

//сохранение списка в файл

void SaveFile(fstream& f, trainlist\* Train)

{

f.open("train.dat", ios::out | ios::binary);

while (Train)

{

f.write(reinterpret\_cast<char\*>(&Train->key), sizeof(train));

Train = Train->next;

}

f.close();

}

int main()

{

setlocale(LC\_ALL, "Russian");

fstream f;

trainlist \*FirstTrain = NULL, \*LastTrain = NULL;

LoadFile(f, &FirstTrain, &LastTrain);

char key;

do

{

system("cls");

cout << "1.Add 2.PrintAll 3.Pop1st Esc.Exit" << endl;

key = \_getch();

system("cls");

switch (key)

{

case '1':

AddTrain(&FirstTrain, &LastTrain, ScanTrain());

break;

case '2':

cout << "№ StPoint EndPoint StTime EndTime Price" << endl;

PrintTrainList(FirstTrain);

break;

case '3':

PopTrain(&FirstTrain);

break;

}

system("pause");

} while (key != 27);

SaveFile(f, FirstTrain);

FreeMem(&FirstTrain);

return 0;

}

5. Тестовые примеры

На рисунках ниже представлены тестовые примеры работы программы.



Рисунок 9 – Главное меню.

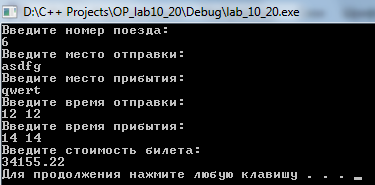


Рисунок 10 – Добавление элемента.

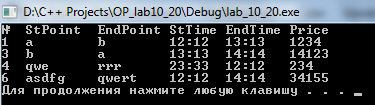


Рисунок 11 – Вывод списка.

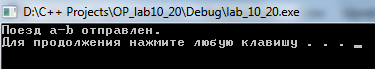


Рисунок 12 – Удаление первого элемента.

Выводы

С помощью линейных списков можно управлять различными данными. Сложностью работы со списками является трудное восприятие хранения данных в памяти компьютера человеком и работа с указателями на области памяти где хранятся эти данные. Минусом работы с односвязными списками является скорость обращение к какому-то конкретному элементу списка.