МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное учреждение высшего образования  
«САНКТ-ПЕТЕРБУГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

КУРСОВОЙ ПРОЕКТ  
ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доц., к.т.н |  |  |  | А.В.Туманова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ |
| **РАЗРАБОТКА ПРОГРАММЫ**  **«Маршруты»** |
| по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134К |  |  |  | Опарин С.Н. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

1. **Постановка задачи**
   1. Задачей курсового проекта является разработка программы маршрутов с использованием заданных структур данных, которая позволяет вводить информацию, хранить её в файле, осуществлять поиск, модификацию, сортировку и удаление данных.
   2. Тип хранимой информации и задание на поиск определяются в соответствии с номером варианта:

**Вариант №9**

Предметная область – «Маршруты».

Данные о маршруте хранятся в структуре с именем MARSH, содержащей следующие поля:

• название начального пункта маршрута;

• название конечного пункта маршрута;

• номер маршрута.

Задание на поиск: найти маршрут, номер которого введѐн с клавиатуры

1. **Описание структур данных**

Данные о маршруте хранятся в структуре MARSH

struct MARSH {

string startPoint; //Начальная точка маршрута

string endPoint; //Конечная точка маршрута

string routeNumber; //Номер маршрута

База данных хранится на диске в виде текстового файла(data.txt). Для работы с БД в процессе выполнения программы используется линейный двухсвязный список.

Номера маршрутов в программе являются уникальными.

1. **Описание программы и созданных функций**

Программа реализована на языке С++ в виде консольного приложения. В главной функции main() реализовано меню пользователя, в котором каждому действию соответствует определенная цифра. Реализованы следующие функции для работы с данными: добавление, удаление, редактирование, вывод всех маршрутов, поиск маршрута по его номеру, выгрузка данных в файл.

***Добавление записи***

void push\_back(string \_startPoint, string \_endPoint, string \_routeNumber)

Входные данные: начальный пункт маршрута, конечный пункт маршрута, номер маршрута(все в строковом типе данных)

Выходные данные: отсутствуют

Функция получает информацию по маршруту для создания новой ячейки в списке. Создается объект нашей структуры MARSH, которой присваивается наш первый элемент списка и начинается проверка на то, что номер маршрута, который мы хотим добавить, является уникальным, то есть в списке нет другого маршрута с данным номером.

После того, как проверка на уникальность номера маршрута пройдена, идет следующая проверка на пустоту нашего списка.

Создается новый объект структуры MARSH с параметрами из нашей функции и, если список является пустым, то наш элемент становится первым в списке, иначе он встает после последнего элемента и становится последним. В конце выводится сообщение о том, что маршрут был успешно добавлен.

***Удаление записи***

void remove(string \_val)

Входные данные: Номер маршрута, который мы хотим удалить(в строковом формате)

Выходные данные: Отсутствуют

Функция получает номер маршрута, который нужно удалить.

Если список пустой, то просто выходим

Иначе проверяем является ли номер маршрута первым. Если номер является первым, то вызывается функция удаления первого элемента, в которой создается объект MARSH и ему присваивается первый элемент списка, потом первым элементом делаем следующий, а исходный первый удаляем.

Если номер маршрута является последним, то вызывается функция удаления последнего элемента, внутри которой происходит проверка на то, что в списке единственный элемент является, как первым, так и последним соответственно. Если это является истиной, то происходит вызов функции удаления первого элемента, которую я описал ранее, иначе создается объект MARSH и ему присваивается первый элемент списка, потом пробегаются по всему списку до последнего и удаляется.

Третий случай, когда удаляется элемент между первым и последним, содержит следующие инструкции:

Создается два объекта MARSH и им присваиваются первый элемент списка и последующий за ним. Идет проход по всему списку и если такого маршрута вообще нет в списке, то выход.

Иначе меняем местами маршруты и удаляем тот, который подходит по номеру и выводим сообщение об успешном удалении.

***Редактирование записи***

Конкретной функции редактирования у меня нет, но есть алгоритм в меню.

Идет проверка на пустоту списка, чтобы не искать элемент в пустом списке. Если список пуст, то выводится сообщение о том, что искать нечего, потому что список является пустым, и выход из кейса редактирования.

Иначе ищем в списке маршрут, который мы хотим отредактировать; если он не нашелся, то выводим сообщение о том, что такого маршрута нет в списке и выходим, иначе вводим новые точки маршрута и заносим их в ячейку маршрута. В конце выводим сообщение о том, что маршрут был успешно отредактирован.

***Вывод записей***

void print()

Входные данные: отсутствуют

Выходные данные: отсутствуют

Функция начинается с проверки на пустоту списка, после создается объект MARSH и ему присваивается первый элемент списка. Запускается цикл, проходим по каждому элементу списка и выводим его поля.

***Поиск записи***

MARSH\* find(string \_val)

Входные данные: номер маршрута, который нужно найти(строковый тип данных)

Выходные данные: маршрут, который мы искали

Создается объект MARSH и ему присваивается первый элемент списка. Пробегаемся в цикле по всему списку, пока не находим нужный нам маршрут. Если он нашелся, то возвращаем его, иначе nullptr.

***Выгрузка данных в файл***

void exportToFile(NodeList& list)

Входные данные: Ссылка на список, который записываем в файл

Выходные данные: Отсутствуют

Начинается функция с создания переменной для хранения пути до файла, куда будет записываться элементы списка.

После дописываем в путь к файлу название самого файла, открываем поток записи в файл.

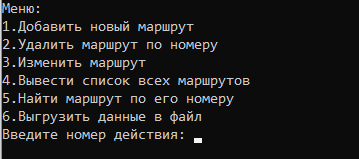
Проверяем открылась ли запись в файл; если проверка не пройдена, то выводится сообщение о том, что запись в файл заблокирована и выходим, иначе создается объект MARSH и ему присваивается первый элемент списка. Затем в цикле пробегаемся по каждому элементу и записываем их файл.

В конце закрываем поток записи и выводим сообщение о том, что запись в файл завершена.

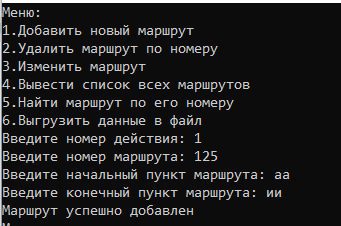
1. **Описание пользовательского интерфейса**

Меню пользователя

1. Добавить маршрут
2. Удалить маршрут
3. Редактировать маршрут
4. Вывести список всех маршрутов
5. Найти маршрут
6. Выгрузить данные в файл
7. **Результат тестирования программы**
   1. При запуске консольного приложения запускается меню main.

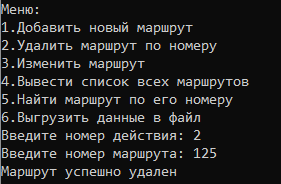


5.2 При выборе действия 1 вызывается функция push\_back



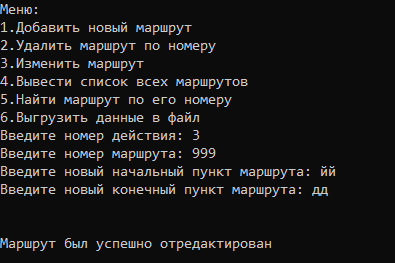
После ввода всех требуемых данных выводится сообщение об успешном добавлении маршрута

5.3 При выборе действия 2 вызывается функция remove



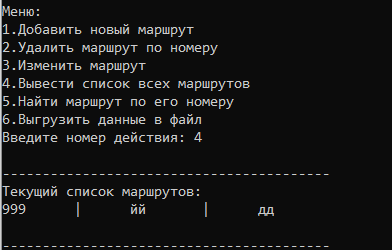
После ввода номера маршрута, который нужно удалить, выводится сообщение об успешном удалении маршрута

5.4 При выборе действия 3 выполняются операции из функции main



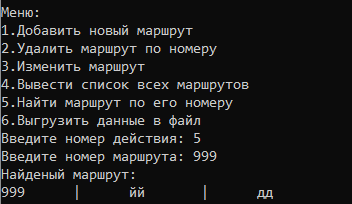
После ввода номера маршрута и новых пунктов маршрута выводится сообщение о том, что маршрут был успешно редактирован

5.5 При выборе действия 4 выполняется функция print



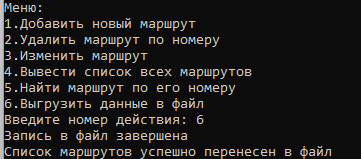
В консоль выводится список всех маршрутов

5.6 При выборе действия 5 выполняется функция find



После ввода номера маршрута, который мы хотим найти, выводится вся информация о маршруте

5.7 При выборе действия 6 выполняется функция exportToFile



Все записи записываются в базу данных

1. **Заключение**

Преимущества программы:

Есть пользовательский интерфейс, который способствует понятному использованию функционала. Можно вводить названия как на русском, так и на английском. Есть проверки на пустоту списка, которые помогают не делать лишние действия с пустым списком. Есть проверка на уникальность номера маршрута.

Недостатки программы:

Небольшое количество проверок на ввод, отсутствие сортировки списка, нет разделение кода по файлам.

1. **Листинг программы**

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <Windows.h>

#include <shlobj.h>

#include <vector>

using namespace std;

bool initAdd = true;

// ноды (элементы) списка

struct MARSH {

string startPoint; //Начальная точка маршрута

string endPoint; //Конечная точка маршрута

string routeNumber; //Номер маршрута

MARSH\* next;

MARSH(string \_startPoint, string \_endPoint, string \_routeNumber) {

startPoint = \_startPoint;

endPoint = \_endPoint;

routeNumber = \_routeNumber;

next = nullptr;

}

};

struct NodeList {

MARSH\* first;

MARSH\* last;

NodeList() : first(nullptr), last(nullptr) {}

bool is\_empty() {

return first == nullptr;

}

void push\_back(string \_startPoint, string \_endPoint, string \_routeNumber) {

MARSH\* tmp = first;

while (tmp) {

if (tmp -> routeNumber == \_routeNumber) {

cout << endl << "Маршрут с таким номером уже существует" << endl << endl;

return;

};

tmp = tmp -> next;

}

MARSH\* p = new MARSH(\_startPoint, \_endPoint, \_routeNumber);

if (is\_empty()) {

first = p;

last = p;

return;

}

last -> next = p;

last = p;

if (!initAdd) cout << endl << "Маршрут был успешно добавлен" << endl << endl;

}

void print() {

if (is\_empty()) return;

MARSH\* p = first;

cout << endl << "-----------------------------------------" << endl;

cout << "Текущий список маршрутов:" << endl;

while (p) {

cout << p -> routeNumber << "\t | \t" << p -> startPoint << "\t | \t" << p -> endPoint << endl;

p = p -> next;

}

cout << endl << "-----------------------------------------" << endl;

}

MARSH\* find(string \_val) {

MARSH\* p = first;

while (p && p -> routeNumber != \_val) p = p->next;

return (p && p -> routeNumber == \_val) ? p : nullptr;

}

void remove\_first() {

if (is\_empty()) return;

MARSH\* p = first;

first = p -> next;

delete p;

}

void remove\_last() {

if (is\_empty()) return;

if (first == last) {

remove\_first();

return;

}

MARSH\* p = first;

while (p -> next != last) p = p -> next;

p -> next = nullptr;

delete last;

last = p;

}

void remove(string \_val) {

if (is\_empty()) return;

if (first -> routeNumber == \_val) {

remove\_first();

return;

}

else if (last -> routeNumber == \_val) {

remove\_last();

return;

}

MARSH\* slow = first;

MARSH\* fast = first -> next;

while (fast && fast -> routeNumber != \_val) {

fast = fast -> next;

slow = slow -> next;

}

if (!fast) {

cout << endl << "Маршрут с таким номером не найден" << endl << endl;

return;

}

slow -> next = fast -> next;

delete fast;

cout << endl << "Маршрут с номером " << (\_val) << " был успешно удалён" << endl << endl;

}

};

string getNumber() {

string res;

while (true) {

cout << "Введите номер маршрута: ";

cin >> res;

if (res.size() == 3) break;

else cout << "Неверный формат" << endl;

}

return res;

}

vector<string> split(string s, string delimiter) {

size\_t pos\_start = 0, pos\_end, delim\_len = delimiter.length();

string token;

vector<string> res;

while ((pos\_end = s.find(delimiter, pos\_start)) != string::npos) {

token = s.substr(pos\_start, pos\_end - pos\_start);

pos\_start = pos\_end + delim\_len;

res.push\_back(token);

}

res.push\_back(s.substr(pos\_start));

return res;

}

void importFromFile(NodeList& list) {

static char desktop[MAX\_PATH + 1];

SHGetSpecialFolderPathA(HWND\_DESKTOP, desktop, CSIDL\_DESKTOP, FALSE);

std::ostringstream file\_path;

file\_path << desktop << "\\data.txt";

ifstream input\_data(file\_path.str().c\_str(), ios\_base::in);

if (!input\_data.is\_open()) {

cout << "Загрузочный файл не был найден" << endl << endl;

return;

}

string line;

string delimiter = ":";

while (getline(input\_data, line, '\n')) {

vector<string> v = split(line, delimiter);

list.push\_back(v.at(1), v.at(2), v.at(0));

}

input\_data.close();

}

void exportToFile(NodeList& list) {

static char desktop[MAX\_PATH + 1];

SHGetSpecialFolderPathA(HWND\_DESKTOP, desktop, CSIDL\_DESKTOP, FALSE);

std::ostringstream file\_path;

file\_path << desktop << "\\data.txt";

ofstream out\_file(file\_path.str().c\_str(), ios\_base::out);

if (!out\_file.is\_open()) {

cout << "Запись в файл заблокирована" << endl;

return;

}

MARSH\* p = list.first;

while (p) {

out\_file << p -> routeNumber << ":" << p -> startPoint << ":" << p -> endPoint << endl;

p = p -> next;

}

out\_file.close();

cout << "Запись в файл завершена" << endl;

}

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

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

NodeList list;

MARSH\* finded;

importFromFile(list);

initAdd = false;

bool menu = true;

int task;

string number, start, end;

string newStart, newEnd;

while (menu) {

cout << "Меню:" << endl;

cout << "1.Добавить новый маршрут" << endl;

cout << "2.Удалить маршрут по номеру" << endl;

cout << "3.Изменить маршрут" << endl;

cout << "4.Вывести список всех маршрутов" << endl;

cout << "5.Найти маршрут по его номеру" << endl;

cout << "6.Выгрузить данные в файл" << endl;

cout << "Введите номер действия: ";

cin >> task;

switch (task)

{

case 1:

number = getNumber();

cout << "Введите начальный пункт маршрута: ";

cin >> start;

cout << "Введите конечный пункт маршрута: ";

cin >> end;

list.push\_back(start, end, number);

cout << "Маршрут успешно добавлен" << endl;

break;

case 2:

if (list.is\_empty()) {

cout << "Список пуст, поэтому удалять нечего" << endl << endl;

break;

}

number = getNumber();

list.remove(number);

cout << "Маршрут успешно удален" << endl;

break;

case 3:

if (list.is\_empty()) {

cout << "Список пуст, поэтому изменять нечего" << endl << endl;

break;

}

number = getNumber();

finded = list.find(number);

if (finded == nullptr) {

cout << endl << "Маршрута с заданным номером не существует" << endl << endl;

break;

}

cout << "Введите новый начальный пункт маршрута: ";

cin >> newStart;

cout << "Введите новый конечный пункт маршрута: ";

cin >> newEnd;

finded -> startPoint = newStart;

finded -> endPoint = newEnd;

cout << endl << endl << "Маршрут был успешно отредактирован" << endl << endl;

break;

case 4:

if (list.is\_empty()) {

cout << "Список пуст, поэтому выводить нечего" << endl << endl;

break;

}

list.print();

break;

case 5:

if (list.is\_empty()) {

cout << "Список пуст, поэтому искать нечего" << endl << endl;

break;

}

number = getNumber();

finded = list.find(number);

if (finded == nullptr) {

cout << "Маршрут с таким номером не найден" << endl;

break;

}

cout << "Найденый маршрут: " << endl;

cout << finded->routeNumber << "\t | \t" << finded -> startPoint << "\t | \t" << finded -> endPoint << endl;

break;

case 6:

if (list.is\_empty()) {

cout << "Список пустой, поэтому переносить в файл нечего" << endl << endl;

break;

}

exportToFile(list);

cout << "Список маршрутов успешно перенесен в файл " << endl;

break;

}

}

}

1. Содержание
2. Постановка задачи……………………………………………2
3. Описание структур данных………………………………......2
4. Описание программы и созданных функций………..………3
5. Описание пользовательского интерфейса………..….………7
6. Результат тестирования……………………………….………7
7. Заключение.…………………………………………..………..9
8. Листинг программы…………………………………..………10