|  |  |  |
| --- | --- | --- |
| Защищено:  Папин А.В..    "\_\_" \_\_\_\_\_\_\_\_\_\_\_2022 г. |  | Демонстрация:  Папин А.В..    "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

**Отчет по домашнему задании по курсу**

**программирование на основе классов и шаблонов (ПКШ)**

#### Тема работы: " Имитация работы автобусного парка "

33

(количество листов)

Вариант № 4

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| студент группы ИУ5Ц-41Б Папин Алексей | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (подпись) |
| Правдина А.Д. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

Москва, МГТУ - 2022

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**СОДЕРЖАНИЕ ОТЧЕТА**

[1. Цель домашней работы 2](#_Toc104901727)

[2. Индивидуальное задание №1. 2](#_Toc104901728)

[2.1. Постановка задачи 2](#_Toc104901729)

[3. Листинг программы: 3](#_Toc104901730)

[Class.h (файл заголовок) 3](#_Toc104901731)

[Class.cpp (вспомогательный файл для выполнений функции) 6](#_Toc104901732)

[first.cpp (основной исходной файл для выполнений заданий) 17](#_Toc104901733)

[4. Результаты работы программы: 29](#_Toc104901734)

# **Цель домашней работы**

Изучение, построение алгоритмов и бесперебойных инфраструктур программы, выполняющейся имитацию работы автобусного парка.

# **Индивидуальное задание №1.**

## **Постановка задачи**

Написать программу моделирования работы автобусного парка.

Сведения о каждом автобусе содержат: номер автобуса, фамилию и инициалы водителя, номер маршрута. База хранится в текстовом файле, его размер может быть произвольным.

Программа должна обеспечивать выбор с помощью меню и выполнение следующих функций:

1. начальное формирование данных о всех автобусах в парке в виде списка;
2. имитация выезда автобуса из парка: вводится номер автобуса; программа удаляет данные об этом автобусе из списка автобусов, находящихся в парке, и записывает эти данные в список автобусов, находящихся на маршруте;
3. имитация въезда автобуса в парк: вводится номер автобуса; программа удаляет данные об этом автобусе из списка автобусов, находящихся на маршруте, и записывает эти данные в список автобусов, находящихся в парке;
4. вывод сведений об автобусах, находящихся в парке, и об автобусах, находящихся на маршруте.

Для представления необходимых списков использовать класс list или vector.

# **Листинг программы:**

## **Class.h (файл заголовок)**

#pragma once

#include <iostream>

#include <string>

#include <fstream>

#include <vector>

#pragma warning(disable : 4996)

using namespace std;

/\*

Написать программу моделирования работы автобусного парка.

Сведения о каждом автобусе содержат: номер автобуса, фамилию и инициалы

водителя, номер маршрута. База хранится в текстовом файле, его размер может быть

произвольным.

Программа должна обеспечивать выбор с помощью меню и выполнение следующих

функций:

- начальное формирование данных о всех автобусах в парке в виде

списка;

- имитация выезда автобуса из парка: вводится номер автобуса;

программа удаляет данные об этом автобусе из списка автобусов,

находящихся в парке, и записывает эти данные в список автобусов,

находящихся на маршруте;

- имитация въезда автобуса в парк: вводится номер автобуса;

программа удаляет данные об этом автобусе из списка автобусов,

находящихся на маршруте, и записывает эти данные в список

автобусов, находящихся в парке;

- вывод сведений об автобусах, находящихся в парке, и об автобусах,

находящихся на маршруте.

Для представления необходимых списков использовать класс list.

\*/

// Проверка на правильность заполнения данных

bool check\_cin\_number\_bus(string str);

bool check\_cin\_FIODRIVER(string str);

// Функция, выводящаяся подсказку по правильности оформлении

void support\_build\_text();

class BUS {

private:

// Номер автобуса, пример АА 559 70.

// INFO (Первые две символы - русские алфавитные буквы, далее,

// три цифры - любые современные арабские цифры, от 0 до 9, далее,

// две произвольные цифры - номер региона)

string number\_bus;

// Фамилия и инициалы водителя, пример, Игнатьев В.В.

string FIODRIVER;

// Номер маршрута, 440

int route\_number;

// Автобус где находится? В парке или на маршруте

// Если в парке, то 0, если на маршруте, то 1

int status;

// Идентификатор класса, чтобы легче работать с определенным объектом

int id;

static int count;

public:

BUS();

BUS(int NUMBER\_BUS, string FIO, int ROUTNUM, int WHERE);

//~BUS() { cout << "DESTRUCTION! ~BUS" << endl; }

// Функция, возвращающаяся определенные значения

string get\_number\_bus();

string get\_FIODRIVER();

int get\_status();

int get\_route\_number();

int get\_id();

// Функция, принимающаяся определенные значения

void set\_number\_bus(string num);

void set\_FIODRIVER(string FIO);

void set\_status(int WHERE);

void set\_route\_number(int route);

void set\_id(int ID);

// Поток вывода в консольном окне

friend ostream& operator<<(ostream& output, BUS& obj);

// Поток ввода пользователем

friend istream& operator>>(istream& input, BUS& obj);

};

// Печатает все данные

void print(vector<BUS> bus);

// Печатает только данные, в которых содержатся автобусы, находящихся в парке

void print\_also\_park(vector<BUS> bus);

// Печатает только данные, в которых содержатся автобусы, находящихся на маршруте

void print\_also\_route(vector<BUS> bus);

// Работа с текстовым файлом

void writeTextFile(fstream& DB, vector<BUS> Class\_BUS);

void readTextFile(fstream& DB);

// Чтение и изменение объекта класса

void read\_and\_add\_into\_class\_TextFile(fstream& DB, vector<BUS>& newBUS, int& size);

// Определение и удаление номер текущего автобуса с автобусного парка и добавление в список маршрутов

void delete\_number\_bus(vector<BUS>& bus, string number);

// Определение и добавление номер текущего автобуса на автобусный парк в списке автобусного парка

void add\_number\_bus(vector<BUS>& bus, string number);

// Проверяет, пустой ли класс?

bool check\_class(vector<BUS> bus);

// Выявляет размер файла

int size\_file(fstream& DB);

// Функция, предлагающаяся распечатать все данные

void offer\_for\_printing(vector<BUS> bus);

// Функция, требующая повторно ввести

void cin\_number\_request(string& number);

// Проверка на существующего номера

bool check\_cin\_number(vector<BUS> bus, string number, int& id);

// Нахождение по id объекта и изменение

void check\_edit\_class\_number\_bus(vector<BUS>& bus, string str, int& id);

// Проверка на присуствии совпадающих номеров в базе данных

bool check\_number\_bus(vector<BUS> DB, BUS bus);

## **Class.cpp (вспомогательный файл для выполнений функции)**

#include "Class.h"

#include <iostream>

#include <string>

#include <fstream>

#pragma warning(disable : 4996)

using namespace std;

// Методы классы, описанные вне класса

// Конструкторы без параметров

BUS::BUS() { id = count++; number\_bus = ""; FIODRIVER = ""; route\_number = NULL; status = NULL; }

// Конструкторы с параметрами

BUS::BUS(int NUMBER\_BUS, string FIO, int ROUTNUM, int WHERE) {

number\_bus = NUMBER\_BUS; FIODRIVER = FIO; route\_number = ROUTNUM; status = WHERE;

}

// Функция, возвращающаяся определенные значения

string BUS::get\_number\_bus() { return number\_bus; }

string BUS::get\_FIODRIVER() { return FIODRIVER; }

int BUS::get\_status() { return status; }

int BUS::get\_route\_number() { return route\_number; }

int BUS::get\_id() { return id; }

// Функция, принимающаяся определенные значения

void BUS::set\_number\_bus(string num) { number\_bus = num; }

void BUS::set\_FIODRIVER(string FIO) { FIODRIVER = FIO; }

void BUS::set\_status(int WHERE) { status = WHERE; }

void BUS::set\_route\_number(int route) { route\_number = route; }

void BUS::set\_id(int ID) { id = ID; }

// Поток вывода в консольном окне

ostream& operator<<(ostream& output, BUS& obj) {

output << "Идентификатор класса: " << obj.id << endl;

output << "Номер автобуса: " << obj.number\_bus << endl

<< "Фамилия и инициалы водителя: " << obj.FIODRIVER << endl

<< "Номер маршрута: " << obj.route\_number << endl;

//output << "Статус автобуса (0 - парк, 1 - маршрут): " << obj.status << endl;

return output;

}

// Поток ввода пользователем

istream& operator>>(istream& input, BUS& obj) {

// Проверка ошибки. Если есть ошибка, тогда true

bool check = true;

// Если пользователь допускает несколько ошибок по оформлению, тогда ему вновь предлагают подсказку по оформлении

int count\_warn = 0;

bool check\_warn = true;

string str\_check = "";

int count\_str\_value = 0;

cin.ignore();

while (check\_warn) {

// Если пользователь вводит несколько ошибок, тогда ему снова предлагают

if (count\_warn == 3) {

cout << endl;

cout << "Вы ввели несколько ошибок подряд!" << endl;

support\_build\_text(); count\_warn = 0;

cin.ignore();

}

while (check) {

if (count\_warn == 3) break;

cout << "Введите номер автобуса: ";

getline(input, obj.number\_bus, '\n');

// Проверка заполнения

if (check\_cin\_number\_bus(obj.number\_bus) == false) {

cout << "Ошибка оформления! Введите заново!" << endl;

count\_warn++;

}

else {

count\_warn = 0;

check = false;

}

}

check = true;

while (check) {

if (count\_warn == 3) break;

cout << "Введите фамилия и инициалы водителя: ";

getline(input, obj.FIODRIVER, '\n');

// Проверка заполнения

if (check\_cin\_FIODRIVER(obj.FIODRIVER) == false) {

cout << "Ошибка оформления! Введите заново!" << endl;

count\_warn++;

}

else {

count\_warn = 0;

check = false;

}

}

check = true;

while (check) {

if (count\_warn == 3) break;

cout << "Введите номер маршрута: ";

getline(input, str\_check, '\n');

for (int i = 0; i < str\_check.size(); i++) {

if (str\_check[i] >= '0' && str\_check[i] <= '9')

count\_str\_value++;

}

if (count\_str\_value == str\_check.size()) {

obj.route\_number = stoi(str\_check);

if (obj.route\_number > 0 && obj.route\_number < 1000) {

count\_warn = 0;

check = false;

}

else

count\_warn++;

}

else {

count\_str\_value = 0;

count\_warn++;

}

}

// По умолчанию автобус находится в парке

if (count\_warn != 3) {

obj.status = 0;

check\_warn = false;

}

else

check = true;

}

//cout << endl;

return input;

}

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

// Остальные функции

// Печатает все данные

void print(vector<BUS> bus) {

for (int i = 0; i < bus.size(); i++) {

cout << bus[i] << endl;

}

}

// Печатает только данные, в которых содержатся автобусы, находящихся в парке

void print\_also\_park(vector<BUS> bus) {

// Проверка на присутствии данных, если их нет, то false

bool check = false;

for (int i = 0; i < bus.size(); i++)

if (bus[i].get\_status() == 0)

check = true;

if (check)

for (int i = 0; i < bus.size(); i++) {

if (bus[i].get\_status() == 0) {

cout << bus[i] << endl;

}

}

else

cout << "Отсутствует" << endl << endl;

}

// Печатает только данные, в которых содержатся автобусы, находящихся на маршруте

void print\_also\_route(vector<BUS> bus) {

// Проверка на присутствии данных, если их нет, то false

bool check = false;

for (int i = 0; i < bus.size(); i++)

if (bus[i].get\_status() == 1)

check = true;

if (check)

for (int i = 0; i < bus.size(); i++) {

if (bus[i].get\_status() == 1) {

cout << bus[i] << endl;

}

}

else

cout << "Отсутствует" << endl << endl;

}

// Проверка на правильность заполнения данных

// Проверка на номер автобуса: АА 559 70

bool check\_cin\_number\_bus(string str) {

int count = 0;

int size = str.size();

bool check\_number = false;

// Первые две символы должны быть заглавными

bool check\_letter = false;

if (size <= 10) {

for (int i = 0; i < size; i++)

if ((str[i] >= '0') && (str[i] <= '9')) {

count++;

// Если есть число

check\_number = true;

}

if (!check\_number) {

cout << "Ошибка! Отсутвуют числа!" << endl;

return false;

}

if (str[0] >= 'А' && str[0] <= 'Я' && str[1] >= 'А' && str[1] <= 'Я')

check\_letter = true;

else {

cout << "Первые две символы должны быть заглавными русскими буквами!" << endl;

return false;

}

// Проверка на инициалы

int pos\_point0 = 0;

int pos\_point1 = 0;

pos\_point0 = str.find(' ', pos\_point0);

//cout << "pos\_point0: " << pos\_point0 << endl;

str = str.replace(pos\_point0, 1, ".");

//cout << "str.erase: " << str << endl;

pos\_point1 = str.find(' ', pos\_point1);

str = str.replace(pos\_point0, 1, " ");

//cout << "pos\_point1: " << pos\_point1 << endl;

if (pos\_point1 - pos\_point0 != 4) {

cout << "После заглавной русской буквы должны стоять три любые цифры" << endl;

return false;

}

if (count == size - 4 && check\_number == true && check\_letter == true)

return true;

else

return false;

}

else {

cout << "Ошибка! Слишком длинный номер!" << endl;

return false;

}

}

// Проверка на ФИО водителя и его инициалы: Игнатьев В.В.

bool check\_cin\_FIODRIVER(string str) {

// Количество пробелов должен быть 1

const int space = 1;

// Количество точек должен быть 2

const int point = 2;

int check\_space = 0;

int check\_point = 0;

// Позиция точек

int pos\_point0 = 0;

int pos\_point1 = 0;

// Не должен присутствовать любое число

bool check\_number = false;

// Проверка на инициалы

bool check\_initials = false;

int size = str.size();

if (size >= 7) {

for (int i = 0; i < size; i++) {

// Количество пробелов должен быть 1

if (str[i] == ' ') check\_space++;

// Количество точек должен быть 2

if (str[i] == '.') check\_point++;

if ((str[i] >= '0') && (str[i] <= '9')) {

// Если есть число

check\_number = true;

}

}

if (!(str[0] >= 'А' && str[0] <= 'Я')) {

cout << "Фамилия должна начинаться с заглавной буквой!" << endl;

return false;

}

if (check\_space != 1) {

cout << "Ошибка! Пробел должен быть один!" << endl;

return false;

}

if (check\_point != 2) {

cout << "Ошибка! Точки должны быть две!" << endl;

return false;

}

if (check\_number) {

cout << "Ошибка! Присутствуют числа!" << endl;

return false;

}

// Проверка на инициалы

pos\_point0 = str.find('.', pos\_point0);

//cout << "pos\_point0: " << pos\_point0 << endl;

str = str.replace(pos\_point0, 1, ",");

//cout << "str.erase: " << str << endl;

pos\_point1 = str.find('.', pos\_point1);

str = str.replace(pos\_point0, 1, ".");

//cout << "pos\_point1: " << pos\_point1 << endl;

if (pos\_point1 - pos\_point0 != 2) {

cout << "Ошибка заполнения инициалы! Инициалы должны быть одними заглавными буквами.\nНапример: Фамилия И.О." << endl;

return false;

}

// Игнатьев В.В.

// Инициалы должны быть заглавными русскими буквами

if (str[size - 2] >= 'А' && str[size - 2] <= 'Я')

if (str[size - 4] >= 'А' && str[size - 4] <= 'Я')

check\_initials = true;

if (!check\_initials) {

cout << "Ошибка! Инициалы должны быть заглавными русскими буквами!" << endl;

return false;

}

// Общая проверка

if (check\_space == space && check\_point == point && check\_number == false && check\_initials == true)

return true;

else

return false;

}

else {

cout << "Короткие фамилия и инициалы!" << endl;

return false;

}

}

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

// Работа с текстовыми файлами

// Запись в текстовыый файл

void writeTextFile(fstream& DB, vector<BUS> BUS) {

DB.open("Moscow.txt", ios\_base::out);

if (!DB)

cout << "Файл открыть невозможно\n";

// Цикл записи файла

for (int i = 0; i < BUS.size(); i++)

// Запись в файл

DB << BUS[i].get\_id() << endl

<< BUS[i].get\_number\_bus() << endl

<< BUS[i].get\_FIODRIVER() << endl

<< BUS[i].get\_route\_number() << endl

<< BUS[i].get\_status() << endl;

DB.clear();

// Закрытие файла

DB.close();

}

// Распечатка файла

void readTextFile(fstream& DB) {

BUS bus\_buff;

DB.open("Moscow.txt", ios\_base::in);

if (!DB.is\_open()) {

cout << "Файл отсутствует! Открыть его невозможно!\n";

}

else {

char buff[100];

while (!DB.eof()) {

DB.getline(buff, 100, '\n');

cout << buff << endl;

}

}

DB.clear();

DB.close(); // Закрытие файла

}

// Чтение и изменение объекта класса

void read\_and\_add\_into\_class\_TextFile(fstream& DB, vector<BUS>& newBUS, int& size) {

DB.open("Moscow.txt", ios\_base::in);

int count = 0, i = 0;

size = size\_file(DB);

string str = "";

DB.open("Moscow.txt", ios\_base::in);

//vector<BUS> newBUS(size);

newBUS.resize(size);

if (!DB.is\_open()) {

//cout << "Файл отсутствует! Открыть его невозможно!\n";

}

else {

char buff[100] = "";

while (!DB.eof()) {

DB.getline(buff, 100, '\n');

if (buff[0] != '\0') {

//cout << buff << endl;

str = buff;

switch (count) {

case 0: { newBUS[i].set\_id(atoi(buff)); count++; break; }

case 1: { newBUS[i].set\_number\_bus(str); count++; break; }

case 2: { newBUS[i].set\_FIODRIVER(str); count++; break; }

case 3: { newBUS[i].set\_route\_number(atoi(buff)); count++; break; }

case 4: { newBUS[i].set\_status(atoi(buff)); count = 0; i++; break; }

}

}

}

}

DB.clear();

DB.close(); // Закрытие файла

}

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

// Дополнительные поддержки

// Определение и удаление номер текущего автобуса с автобусного парка и добавление в список маршрутов

void delete\_number\_bus(vector<BUS>& bus, string number) {

string tmp = "";

// Проверяет на наличие присутствии совпадающего номера

int check = NULL;

for (int i = 0; i < bus.size(); i++) {

tmp = bus[i].get\_number\_bus();

if (tmp == number && bus[i].get\_status() == 1) {

check = 2;

break;

}

if (tmp == number) {

//bus.erase(bus.begin() + i);

// Если в парке, то 0, если на маршруте, то 1

bus[i].set\_status(1);

check = 1;

// Размер изменился

//size = bus.size();

break;

}

else {

check = 0;

}

}

if (check == 0)

cout << "Нет совпадающего номера, которого Вы ввели ранее!" << endl;

else if (check == 1)

cout << "Есть совпадающий номер!" << endl;

else if (check == 2)

cout << "Ошибка! Данный номер автобуса находится на маршруте! Выход на главное меню!" << endl;

}

// Определение и добавление номер текущего автобуса на автобусный парк в списке автобусного парка

void add\_number\_bus(vector<BUS>& bus, string number) {

string tmp = "";

// Проверяет на наличие присутствии совпадающего номера

int check = NULL;

for (int i = 0; i < bus.size(); i++) {

tmp = bus[i].get\_number\_bus();

if (tmp == number && bus[i].get\_status() == 0) {

check = 2;

break;

}

if (tmp == number) {

bus[i].set\_status(0);

check = 1;

break;

}

else {

check = 0;

}

}

if (check == 0)

cout << "Нет совпадающего номера, которого Вы ввели ранее!" << endl;

else if (check == 1)

cout << "Есть совпадающий номер!" << endl;

else if (check == 2)

cout << "Ошибка! Данный номер автобуса находится в парке! Выход на главное меню!" << endl;

}

// Проверяет, пустой ли класс?

bool check\_class(vector<BUS> bus) {

// Если пустой, тогда true

bool check = false;

if (bus.size() == NULL) check = true;

else {

for (int i = 0; i < bus.size(); i++) {

if (bus[i].get\_number\_bus() == "" && bus[i].get\_FIODRIVER() == "" && bus[i].get\_route\_number() == NULL && bus[i].get\_status() == NULL )

check = true;

}

}

return check;

}

// Выявляет размер файла

int size\_file(fstream& DB) {

int count = 0;

if (!DB.is\_open()) {

//cout << "Файл отсутствует! Открыть его невозможно!\n";

}

else {

char buff[100] = "";

while (!DB.eof()) {

DB.getline(buff, 100, '\n');

//cout << buff << endl;

if (strlen(buff) != 0)

count++;

}

}

DB.clear();

DB.close(); // Закрытие файла

return (count / 5);

}

// Функция, выводящаяся подсказку по правильности оформлении

void support\_build\_text() {

bool check = true;

// Если пользователь допускает несколько раз ошибок, тогда ему вновь предлагают включить функцию на подсказку

//int count = 0;

string request = "";

cout << "Нужна подсказка по оформлению?\nВведите 'да' или 'нет'" << endl;

while (check) {

cout << "Ваш ответ: ";

cin >> request;

if (request == "да") {

cout << "Пример заполнения: " << endl

<< "Номер автобуса: АА 559 70 \nСправочник по оформлению:\n\tАА - любые заглавные русские алфавиты, только 2 символы;\n\t559 - любые цифры, только 3 символы;\n\t70 код региона, 2 или 3 символы." << endl << endl

<< "Фамилия и инициалы водителя: Игнатьев В.В." << endl

<< "Номер маршрута: 440" << endl << endl;

check = false;

}

else if (request == "нет") {

cout << "Ответ принят. Не подсказывать." << endl;

check = false;

}

else {

cout << "Ошибка! Введите только 'да' или 'нет' !" << endl;

}

}

}

// Функция, предлагающаяся распечатать все данные

void offer\_for\_printing(vector<BUS> bus) {

string request;

// Проверяет на правильность заполнения

int check = 0;

while (check == 0) {

cout << "Распечатать все данные? Введите да или нет или другое." << endl;

cout << "Ваш ответ: ";

cin >> request;

cout << endl;

if (request == "да") {

cout << "Список автобусов, находящихся в парке:" << endl;

print\_also\_park(bus);

cout << "Список автобусов, находящихся на маршруте:" << endl;

print\_also\_route(bus);

check = 1;

break;

}

else if (request == "нет") {

cout << "Ответ принят. Не распечатать." << endl;

check = 1;

break;

}

else if (request == "другое") {

check = 2;

cout << "Распечатать данные автобуса, находящегося в парке или на маршруте?\nВведите парк или маршрут или назад." << endl;

while (check == 2) {

cin >> request;

if (request == "парк") {

print\_also\_park(bus);

check = 3;

break;

}

else if (request == "маршрут") {

print\_also\_route(bus);

check = 3;

break;

}

else if (request == "назад") {

check = 0;

}

else {

cout << "Ошибка! Введите только 'парк' или 'маршрут' или 'назад'!" << endl;

}

}

}

else

cout << "Ошибка! Введите только 'да' или 'нет' или 'другое'!" << endl;

}

cout << endl;

}

// Функция, требующая повторно ввести для правильности оформлении

void cin\_number\_request(string& number) {

bool check = true;

while (check) {

cout << "Номер автобуса: ";

getline(cin, number, '\n');

if (check\_cin\_number\_bus(number) == false)

cout << "Ошибка оформления! Введите заново!" << endl;

else

check = false;

}

}

// Проверка на существующего номера

bool check\_cin\_number(vector<BUS> bus, string number, int& id) {

for (int i = 0; i < bus.size(); i++) {

if (bus[i].get\_number\_bus() == number) {

id = bus[i].get\_id();

return true;

}

}

return false;

}

// Проверка на присуствии совпадающих номеров в базе данных

bool check\_number\_bus(vector<BUS> DB, BUS bus) {

for (int i = 0; i < DB.size(); i++)

if (DB[i].get\_number\_bus() == bus.get\_number\_bus())

return true;

return false;

}

## **first.cpp (основной исходной файл для выполнений заданий)**

// DZ\_04.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include <iostream>

#include "Class.h"

#pragma warning(disable : 4996)

using namespace std;

int BUS::count = 0;

int main()

{

system("chcp 1251 > nul");

cout << "Домашнее задание:\n Моделирование работы автобусного парка\n Папин А.В., ИУ5Ц-41Б \n Вариант № 4\n\n";

system("pause");

// Размер объекта класса

int size = NULL;

// Сигнал предупреждения о отсутствии содержимого файла

bool warning = true;

bool check = true;

//BUS \*Moscow = new BUS[size];

// Список автобусов, находящихся в парке и на маршруте

vector <BUS> Moscow(size);

//Moscow.resize(size);

// Создаем новый файл

fstream DB;

while (true) {

system("cls");

// Проверка на пустого класса

if (check\_class(Moscow) == true) {

//cout << "Пустой класс" << endl;

// Если пустой класс, то из файла заполяют в класс

read\_and\_add\_into\_class\_TextFile(DB, Moscow, size);

// Если содержимое в файле нет, тогда в классе нет содержимых,

// теперь потребуется подтверждении о отсутствии содержимых в классе

if (!Moscow.empty())

warning = false;

}

else {

warning = false;

//print(Moscow);

//cout << "Класс не пустой" << endl;

//cout << endl;

// Файл не пустой

}

cout << "№1. Начальное формирование данных" << endl;

if (!warning) {

cout << "№2. Имитация выезда автобуса из парка" << endl;

cout << "№3. Имитация въезда автобуса в парк" << endl;

cout << "№4. Вывод сведений об автобусах" << endl;

cout << "№5. Операция над данными" << endl;

}

cout << "№0. Выход с программы" << endl;

int sw;

// Не позволяет пользователю выбрать другой пункт

while (check) {

cout << "Введите номер пункт: ";

cin >> sw;

if (warning) {

if (sw == 1 || sw == 0) check = false;

else

cout << "Ошибка! Введите перечень доступных пунктов!" << endl;

}

else {

check = false;

}

}

check = true;

switch (sw)

{

case 1:

{

system("cls");

cout << "№1. Начальное формирование данных" << endl;

cout << "Введите новый размер объект класса: ";

cin >> size;

Moscow.resize(size);

// Поддержка на правильность оформления

support\_build\_text();

for (int i = 0; i < size; i++) {

cin >> Moscow[i];

cout << endl;

}

// Присвоим начальные формированные данные

writeTextFile(DB, Moscow);

system("pause");

break;

}

case 2:

{

system("cls");

cout << "№2. Имитация выезда автобуса из парка" << endl;

string number, request;

bool check = true;

// Функция, предлагающаяся на распечатку всех данных

offer\_for\_printing(Moscow);

// Функция, подсказывающаяся на правильность оформлении

support\_build\_text();

cout << "Введите номер автобуса, который собирался выехать с автобусного парка" << endl;

cin.ignore();

cin\_number\_request(number);

// Определение и удаление номер текущего автобуса с автобусного парка и добавление в список маршрутов

delete\_number\_bus(Moscow, number);

// Присвоим начальные формированные данные

writeTextFile(DB, Moscow);

system("pause");

break;

}

case 3:

{

system("cls");

cout << "№3. Имитация въезда автобуса в парк" << endl;

string number, request;

bool check = true;

// Функция, предлагающаяся на распечатку всех данных

offer\_for\_printing(Moscow);

// Функция, подсказывающаяся на правильность оформлении

support\_build\_text();

cout << "Введите номер автобуса, который собирался въехать в автобусный парк" << endl;

cin.ignore();

cin\_number\_request(number);

add\_number\_bus(Moscow, number);

// Присвоим начальные формированные данные

writeTextFile(DB, Moscow);

system("pause");

break;

}

case 4:

{

system("cls");

cout << "№4. Вывод сведений об автобусах." << endl;

cout << "Список автобусов, находящихся в автобусном парке: " << endl << endl;

// ЧТЕНИЕ И РАСПЕЧАТКА ФАЙЛА

//readTextFile(DB);

print\_also\_park(Moscow);

cout << "Список автобусов, находящихся на маршруте: " << endl << endl;

print\_also\_route(Moscow);

system("pause");

break;

}

case 5:

{

system("cls");

cout << "№5. Операция над данными" << endl;

cout << "Введите нижеперечисленные пункты, которых вы будете взаимодействовать" << endl;

cout << "№1. Добавление данных" << endl;

cout << "№2. Изменение данных" << endl;

cout << "№3. Удаление данных" << endl;

cout << "№4. Отсортировать по алфавитному порядку по возрастанию" << endl;

cout << "№5. Отсортировать по алфавитному порядку по убыванию" << endl;

cout << "№0. Назад в меню" << endl;

cout << "Введите номер пункт: ";

cin >> sw;

switch (sw)

{

case 1:

{

cout << "№1. Добавление данных" << endl;

// Поддержка на правильность оформлении

support\_build\_text();

BUS tmp;

cin >> tmp;

if (check\_number\_bus(Moscow, tmp)) {

cout << "Данный новый номер совпадает с номером в базе данных! Отмена добавление. Возврат в меню!" << endl;

system("pause");

}

else {

// Добавление элемент в конец вектора

Moscow.emplace\_back(tmp);

}

break;

}

case 2:

{

system("cls");

cout << "№2. Изменение данных" << endl;

string number = "";

bool check\_0 = true;

// Если пользователь вводит несколько ошибок, тогда перебрасывает в меню

int count\_warn = 0;

int id = NULL;

// Функция, предлагающаяся на распечатку всех данных

offer\_for\_printing(Moscow);

cin.ignore();

// Функция, подсказывающаяся на правильность оформлении

support\_build\_text();

cin.ignore();

cout << endl;

cout << "Введите номер автобуса для поиска" << endl;

while (check\_0) {

cin\_number\_request(number);

if (!check\_cin\_number(Moscow, number, id)) {

cout << "Данный номер не существует в базе данных. Повторно введите" << endl;

count\_warn++;

}

else {

cout << "Данный номер существует в базе данных!" << endl;

check\_0 = false;

}

if (count\_warn == 3) break;

}

if (count\_warn == 3) {

cout << "Вы ввели несколько ошибок. Возврат в главное меню" << endl;

break;

}

cout << endl;

cout << "Введите номер пункта для изменения данных:" << endl;

cout << "№1. Номера автобуса" << endl;

cout << "№2. Фамилия и инициалы водителя" << endl;

cout << "№3. Номер маршрута" << endl;

cout << "№0. Назад" << endl;

cout << "Введите номер пункт: ";

cin >> sw;

cout << endl;

switch (sw)

{

case 1:

{

string str;

cout << "Введите новый номер автобуса: ";

cin.ignore();

getline(cin, str, '\n');

if (!check\_cin\_number\_bus(str)) {

cout << "Ошибка оформления! Возврат в главное меню." << endl;

system("pause");

break;

}

for (int i = 0; i < Moscow.size(); i++) {

if (Moscow[i].get\_id() == id) {

Moscow[i].set\_number\_bus(str);

break;

}

}

writeTextFile(DB, Moscow);

break;

}

case 2:

{

string str;

cout << "Введите новые фамилия и инициалы водителя: ";

cin.ignore();

getline(cin, str, '\n');

if (!check\_cin\_FIODRIVER(str)) {

cout << "Ошибка оформления! Возврат в главное меню." << endl;

system("pause");

break;

}

for (int i = 0; i < Moscow.size(); i++) {

if (Moscow[i].get\_id() == id) {

Moscow[i].set\_FIODRIVER(str);

break;

}

}

writeTextFile(DB, Moscow);

break;

}

case 3:

{

int new\_route;

cout << "Введите новый номер маршрута: ";

cin >> new\_route;

if (!(new\_route > 0 && new\_route < 1000)) {

cout << "Ошибка! Номер должен быть не более 1000 и не менее 1!\nВозврат в главное меню." << endl;

system("pause");

break;

}

for (int i = 0; i < Moscow.size(); i++) {

if (Moscow[i].get\_id() == id) {

Moscow[i].set\_route\_number(new\_route);

break;

}

}

writeTextFile(DB, Moscow);

break;

}

case 0:

{

cout << "Возврат в главное меню" << endl;

break;

}

default:

{

cout << "Ошибка! Указанный номер пункта отсутствует! Возврат к главному меню!" << endl;

system("pause");

break;

}

}

break;

}

case 3:

{

cout << "№3. Удаление данных" << endl;

string number = "";

bool check\_0 = true;

// Если пользователь вводит несколько ошибок, тогда перебрасывает в меню

int count\_warn = 0;

int id = NULL;

// Функция, предлагающаяся на распечатку всех данных

offer\_for\_printing(Moscow);

cout << "Введите номер автобуса для поиска" << endl;

cin.ignore();

// Функция, подсказывающаяся на правильность оформлении

support\_build\_text();

cin.ignore();

while (check\_0) {

cin\_number\_request(number);

if (!check\_cin\_number(Moscow, number, id)) {

cout << "Данный номер не существует в базе данных. Повторно введите" << endl;

count\_warn++;

}

else {

cout << "Данный номер существует в базе данных!" << endl;

check\_0 = false;

}

if (count\_warn == 3) break;

}

if (count\_warn == 3) {

cout << "Вы ввели несколько ошибок. Возврат в главное меню" << endl;

break;

}

for (int i = 0; i < Moscow.size(); i++) {

if (Moscow[i].get\_id() == id) {

Moscow.erase(Moscow.begin() + i);

break;

}

}

writeTextFile(DB, Moscow);

break;

}

case 4:

{

system("cls");

cout << "№4. Отсортировать по алфавитному порядку по возрастанию" << endl;

// а, б, в, г, д, е, ё, ж, з, и, й, к, л, м, н, о, п, р, с, т, у, ф, х, ц, ч, ш, щ, ъ, ы, ь, э, ю, я

cout << "Введите номер пункта для изменения данных:" << endl;

cout << "№1. Сортировать по номеру автобуса" << endl;

cout << "№2. Сортировать по фамилию водителя" << endl;

cout << "№3. Сортировать по номеру маршруту" << endl;

cout << "№0. Назад" << endl;

cout << "Введите номер пункт: ";

cin >> sw;

cout << endl;

switch (sw)

{

case 1:

{

int str\_size0 = NULL, str\_size1 = NULL, size = NULL, length = 0;

string str0 = "", str1 = "";

BUS tmp;

for (int k = 0; k < Moscow.size(); k++)

for (int i = 0; i < Moscow.size() - 1; i++) {

str\_size0 = Moscow[i].get\_number\_bus().size();

str\_size1 = Moscow[i + 1].get\_number\_bus().size();

if (str\_size0 > str\_size1)

size = str\_size1;

else

size = str\_size0;

for (int j = 0; j < 2; j++) {

str0 = str0 + Moscow[i].get\_number\_bus()[j];

str1 = str1 + Moscow[i + 1].get\_number\_bus()[j];

if (str0 > str1) {

tmp = Moscow[i];

Moscow[i] = Moscow[i + 1];

Moscow[i + 1] = tmp;

break;

}

}

str0 = ""; str1 = "";

}

writeTextFile(DB, Moscow);

break;

}

case 2:

{

int str\_size0 = NULL, str\_size1 = NULL, size = NULL, length = 0;

string str0 = "", str1 = "";

BUS tmp;

for (int k = 0; k < Moscow.size(); k++)

for (int i = 0; i < Moscow.size() - 1; i++) {

str\_size0 = Moscow[i].get\_FIODRIVER().size();

str\_size1 = Moscow[i + 1].get\_FIODRIVER().size();

if (str\_size0 > str\_size1)

size = str\_size1;

else

size = str\_size0;

for (int j = 0; j < 2; j++) {

str0 = str0 + Moscow[i].get\_FIODRIVER()[j];

str1 = str1 + Moscow[i + 1].get\_FIODRIVER()[j];

if (str0 > str1) {

tmp = Moscow[i];

Moscow[i] = Moscow[i + 1];

Moscow[i + 1] = tmp;

break;

}

}

str0 = ""; str1 = "";

}

writeTextFile(DB, Moscow);

break;

}

case 3:

{

int str0, str1, size = 0, sizecheck = 0;

BUS tmp;

bool value\_check = true;

for (int i = 0; i < Moscow.size() - 1; i++) {

for (int j = 0; j < Moscow.size() - 1; j++) {

str0 = Moscow[j].get\_route\_number();

str1 = Moscow[j + 1].get\_route\_number();

if (str0 > str1)

sizecheck = str1;

else

sizecheck = str0;

while (value\_check) {

if (sizecheck == 0) { value\_check = false; break; }

sizecheck = sizecheck / 10;

size++;

}

if (str0 > str1) {

tmp = Moscow[j];

Moscow[j] = Moscow[j + 1];

Moscow[j + 1] = tmp;

//break;

}

}

}

writeTextFile(DB, Moscow);

break;

}

case 0:

{

cout << "Возврат в главное меню" << endl;

break;

}

default:

{

cout << "Ошибка! Указанный номер пункта отсутствует! Возврат к главному меню!" << endl;

system("pause");

break;

}

}

break;

}

case 5:

{

system("cls");

cout << "№5. Отсортировать по алфавитному порядку по убыванию" << endl;

// а, б, в, г, д, е, ё, ж, з, и, й, к, л, м, н, о, п, р, с, т, у, ф, х, ц, ч, ш, щ, ъ, ы, ь, э, ю, я

cout << "Введите номер пункта для изменения данных:" << endl;

cout << "№1. Сортировать по номеру автобуса" << endl;

cout << "№2. Сортировать по фамилию водителя" << endl;

cout << "№3. Сортировать по номеру маршруту" << endl;

cout << "№0. Назад" << endl;

cout << "Введите номер пункт: ";

cin >> sw;

cout << endl;

switch (sw)

{

case 1:

{

int str\_size0 = NULL, str\_size1 = NULL, size = NULL, length = 0;

string str0 = "", str1 = "";

BUS tmp;

for (int k = 0; k < Moscow.size(); k++)

for (int i = 0; i < Moscow.size() - 1; i++) {

str\_size0 = Moscow[i].get\_number\_bus().size();

str\_size1 = Moscow[i + 1].get\_number\_bus().size();

if (str\_size0 > str\_size1)

size = str\_size1;

else

size = str\_size0;

for (int j = 0; j < 2; j++) {

str0 = str0 + Moscow[i].get\_number\_bus()[j];

str1 = str1 + Moscow[i + 1].get\_number\_bus()[j];

if (str0 < str1) {

tmp = Moscow[i];

Moscow[i] = Moscow[i + 1];

Moscow[i + 1] = tmp;

break;

}

}

str0 = ""; str1 = "";

}

writeTextFile(DB, Moscow);

break;

}

case 2:

{

int str\_size0 = NULL, str\_size1 = NULL, size = NULL, length = 0;

string str0 = "", str1 = "";

BUS tmp;

for (int k = 0; k < Moscow.size(); k++)

for (int i = 0; i < Moscow.size() - 1; i++) {

str\_size0 = Moscow[i].get\_FIODRIVER().size();

str\_size1 = Moscow[i + 1].get\_FIODRIVER().size();

if (str\_size0 > str\_size1)

size = str\_size1;

else

size = str\_size0;

for (int j = 0; j < 2; j++) {

str0 = str0 + Moscow[i].get\_FIODRIVER()[j];

str1 = str1 + Moscow[i + 1].get\_FIODRIVER()[j];

if (str0 < str1) {

tmp = Moscow[i];

Moscow[i] = Moscow[i + 1];

Moscow[i + 1] = tmp;

break;

}

}

str0 = ""; str1 = "";

}

writeTextFile(DB, Moscow);

break;

}

case 3:

{

int str0, str1, size = 0, sizecheck = 0;

BUS tmp;

bool value\_check = true;

for (int i = 0; i < Moscow.size() - 1; i++) {

for (int j = 0; j < Moscow.size() - 1; j++) {

str0 = Moscow[j].get\_route\_number();

str1 = Moscow[j + 1].get\_route\_number();

if (str0 > str1)

sizecheck = str1;

else

sizecheck = str0;

while (value\_check) {

if (sizecheck == 0) { value\_check = false; break; }

sizecheck = sizecheck / 10;

size++;

}

if (str0 < str1) {

tmp = Moscow[j];

Moscow[j] = Moscow[j + 1];

Moscow[j + 1] = tmp;

//break;

}

}

}

writeTextFile(DB, Moscow);

break;

}

case 0:

{

cout << "Возврат в главное меню" << endl;

break;

}

default:

{

cout << "Ошибка! Указанный номер пункта отсутствует! Возврат к главному меню!" << endl;

system("pause");

break;

}

}

break;

}

case 0:

{

cout << "Возврат в главное меню" << endl;

break;

}

default:

{

cout << "Ошибка! Указанный номер пункта отсутствует! Возврат к главному меню!" << endl;

system("pause");

break;

}

}

break;

}

case 0:

{

cout << "Выход с программы" << endl;

system("pause");

exit(1);

break;

}

default:

{

cout << "Ошибка! Указанный номер пункта отсутствует! Возврат к главному меню!" << endl;

system("pause");

break;

}

}

}

}

# **Результаты работы программы:**

Лабораторная работа № 6

Папин А.В., ИУ5Ц-41Б

Вариант № 4

№1. 2.1.3. Макрокоманды и переменные этапа компиляции

№2. 2.1.4. Шаблоны функции Swap

№3. 2.1.5. Шаблон функции с несколькими параметрами (по варианту)

№4. 2.1.6. Специализация шаблона функции

№5. 2.1.7. Шаблон класса с внутренними методами с одним параметром

№6. 2.1.8. Шаблон класса с внутренними методами с двумя параметрами

№7. 2.1.9. Шаблон класса с внешним описанием методов и двумя параметрами

№8. 2.1.10. Описание нового класса на основе шаблона класса

№9. 2.1.11. Описание нового шаблона класса на основе шаблона класса

№10. 2.1.12. Задание нового типа на основе шаблона (typedef)

№0. Выход с программы

Введите номер пункт: 1

№1. 2.1.3. Макрокоманды и переменные этапа компиляции

Сумма всех элементов массивы: 55

До SWAP

x = 10; y = 20

После SWAP

x = 20; y = 10

a = 1; b = 2; c = 3;

Сумма трех элементов (через макрос): 6

№2. 2.1.4. Шаблоны функции Swap

Целый тип (указатель):

До SWAP

x = 10; y = 20

После SWAP

x = 20; y = 10

Вещественный тип (указатель):

До SWAP

z = 3.1416; d = 1.474

После SWAP

z = 1.474; d = 3.1416

№3. 2.1.5. Шаблон функции с несколькими параметрами (по варианту)

Минимальный элемент массива(целочисленный тип): 0

Минимальный элемент массива(вещественный тип): 1.77

Минимальный элемент массива(вещественный тип): 0.13

№4. 2.1.6. Специализация шаблона функции

Строка:

Меркурий Венера Земля Марс Юпитер Сатурн Уран Нептун

Короткая строка:

Вызов функции специализации с типом string

Самая короткая строка: Уран

Вызов функции специализации с незивестным типом

Самая короткая строка: Уран

№5. 2.1.7. Шаблон класса с внутренними методами с одним параметром

b = 1; k = 2; x = 3

Функция y: 7

y = b + k \* x

y = 7 = 1 + 2 \* 3

b = 1.12346; k = 2.87654; x = 3.1416

Функция y: 10.1604

y = b + k \* x

y = 10.1604 = 1.12346 + 2.87654 \* 3.1416

b = 9.556; k = 6.2354; x = 5.6

Функция y: 44.4742

y = b + k \* x

y = 44.4742 = 9.556 + 6.2354 \* 5.6

№6. 2.1.8. Шаблон класса с внутренними методами с двумя параметрами

Функция y: 7

y = b + k \* x

y = 7 = 1 + 2 \* 3

Цвет закраски: красный

Функция y: 10.1604

y = b + k \* x

y = 10.1604 = 1.12346 + 2.87654 \* 3.1416

Цвет закраски: синий

Функция y: 44.4742

y = b + k \* x

y = 44.4742 = 9.556 + 6.2354 \* 5.6

Цвет закраски: зеленый

№7. 2.1.9. Шаблон класса с внешним описанием методов и двумя параметрами

Функция y: 223593591

y = b + k \* x

y = 223593591 = 49235756 + 821231235 \* 568216473

Цвет закраски: красный

№8. 2.1.10. Описание нового класса на основе шаблона класса

y = b + k \* x

y = 176087 = 1.423 + 324.2 \* 543.14

Цвет закраски: красный

№9. 2.1.11. Описание нового шаблона класса на основе шаблона класса

y = b + k \* x

y = 103143 = 3.423 + 424.2 \* 243.14

Цвет закраски: красный

y = b + k \* x

y = 176087 = 1.423 + 324.2 \* 543.14

Цвет закраски: зеленый

y = b + k \* x

y = 22 = 4 + 9 \* 2

Цвет закраски: синий

№10. 2.1.12. Задание нового типа на основе шаблона (typedef)

y = b + k \* x

y = 29 = 1 + 4 \* 7

Цвет закраски: красный

Лабораторная работа № 6

Папин А.В., ИУ5Ц-41Б

Вариант № 4

№1. 2.1.3. Макрокоманды и переменные этапа компиляции

№2. 2.1.4. Шаблоны функции Swap

№3. 2.1.5. Шаблон функции с несколькими параметрами (по варианту)

№4. 2.1.6. Специализация шаблона функции

№5. 2.1.7. Шаблон класса с внутренними методами с одним параметром

№6. 2.1.8. Шаблон класса с внутренними методами с двумя параметрами

№7. 2.1.9. Шаблон класса с внешним описанием методов и двумя параметрами

№8. 2.1.10. Описание нового класса на основе шаблона класса

№9. 2.1.11. Описание нового шаблона класса на основе шаблона класса

№10. 2.1.12. Задание нового типа на основе шаблона (typedef)

№0. Выход с программы

Введите номер пункт: 0

Выход с программы

Для продолжения нажмите любую клавишу . . .