**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Программирование на основе классов и шаблонов»

Отчет по домашней работе

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-23Б |  | преподаватель каф. ИУ5 |
| Поляков Данила |  | Правдина А.Д. |
| Подпись и дата: 31.05.2022 |  | Подпись и дата: 31.05.2022 |

Москва, 2022 г.

**Постановка задачи**

Составить программу моделирования работы автобусного парка.

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

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

\* начальное формирование данных о всех автобусах в парке;

\* 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, упорядоченных по номерам автобусов;

\* вывод сведений об автобусах, находящихся в парке, и об автобусах, находящихся на маршруте, упорядоченных по номерам маршрутов.

Хранение всех необходимых списков организовать с применением класса map, в качестве ключа использовать «номер автобуса».

**Текст программы**

**Листинг кода программы:**

**main.cpp**

*#include* <iostream>  
*#include* <vector>  
*#include* "DataBase.hpp"  
  
*void* PrintMenu() {  
 std::vector<std::string> menuHeaders = {"Перезаполнить список маршрутов",  
 "Добавить автобус в парк",  
 "Удалить автобус из парка",  
 "Выезд автобуса на маршрут",  
 "Въезд автобуса в парк",  
 "Список автобусов по номеру",  
 "Список автобусов по маршруту",  
 "Выход"};  
 std::cout << '\n';  
 *for* (size\_t i = 0; i < menuHeaders.size(); ++i) {  
 std::cout << i + 1 << '.' << menuHeaders[i] << '\n';  
 }  
 std::cout << "Введите необходимый вам пункт меню:\n";  
}  
  
*enum* MenuPouints {  
 FEEL\_BUSES = 1,  
 ADD\_BUS,  
 DELETE\_BUS,  
 OUT\_BUS,  
 ENTER\_BUS,  
 BUSES\_BY\_NUMBER,  
 BUSES\_BY\_ROUTE,  
 EXIT  
};  
*enum* SortedBy {  
 NUMBERS,  
 ROUTES  
};  
  
*int* main() {  
 *bool* contqqq = *true*;  
 std::map<std::string, Bus> onRoute, inPark;  
 fillData(inPark, onRoute);  
 *while* (contqqq) {  
 PrintMenu();  
 size\_t menuPoint;  
 std::cin >> menuPoint;  
 *switch* (menuPoint) {  
 *case* FEEL\_BUSES:  
 SeedData(inPark);  
 onRoute.clear();  
 *break*;  
 *case* ADD\_BUS:  
 AddBus(inPark);  
 *break*;  
 *case* DELETE\_BUS:  
 DeleteBus(inPark);  
 *break*;  
 *case* OUT\_BUS:  
 OutBus(inPark, onRoute);  
 *break*;  
 *case* ENTER\_BUS:  
 EnterBus(inPark, onRoute);  
 *break*;  
 *case* BUSES\_BY\_NUMBER:  
 PrintBuses(inPark, onRoute, NUMBERS);  
 *break*;  
 *case* BUSES\_BY\_ROUTE:  
 PrintBuses(inPark, onRoute, ROUTES);  
 *break*;  
 *case* EXIT:  
 updateDB(inPark, onRoute);  
 contqqq = *false*;  
 *break*;  
 *default*:  
 *break*;  
 }  
 }  
 updateDB(inPark, onRoute);  
 *return* 0;  
}

**tasks.hpp**

*// Copyright 2022 qqq <polyakovdd@student.bmstu.ru>  
  
  
#ifndef* CODE\_TASKS\_HPP  
*#define* CODE\_TASKS\_HPP  
  
  
*#include* <iostream>  
*#include* <algorithm>  
*#include* <fstream>  
*#include* <vector>  
*#include* <list>  
*#include* <map>  
*#include* <cmath>  
*#include* "Planet.hpp"  
*#include* "Card.hpp"  
  
*void* printText(*int*);  
  
*void* task1();  
  
*void* task2();  
  
*void* task3();  
  
*void* task4();  
  
*void* task5();  
  
*void* task6();  
  
*void* task7();  
  
*void* task8();  
  
*void* task9();  
  
*#endif //CODE\_TASKS\_HPP*

**Bus.hpp**

*// Copyright 2022 qqq <polyakovdd@student.bmstu.ru>  
  
  
#ifndef* CODE\_BUS\_HPP  
*#define* CODE\_BUS\_HPP  
  
*#include* <iostream>  
*#include* <string>  
*#include* <vector>  
*#include* <fstream>  
  
*class* Bus {  
 std::string \_number = "11111";  
 std::string \_driverInfo = "wqeeqwe";  
 std::string \_routeNumber = "q123";  
  
*public*:  
 Bus() = *default*;  
  
 Bus(std::string number, std::string driver, std::string route) : \_number(number), \_driverInfo(driver),  
 \_routeNumber(route){};  
  
  
 *inline* std::string GetNumber() {  
 *return* \_number;  
 }  
  
 *inline* std::string GetDriver() {  
 *return* \_driverInfo;  
 }  
  
 *inline* std::string GetRoute() {  
 *return* \_routeNumber;  
 }  
  
 *friend* std::ofstream &*operator*<<(std::ofstream &, Bus &);  
  
 *friend* std::ifstream &*operator*>>(std::ifstream &, Bus &);  
  
 *friend* std::istream &*operator*>>(std::istream &, Bus &);  
  
};  
  
  
*#endif //CODE\_BUS\_HPP*

**Bus.cpp**

*// Copyright 2022 qqq <polyakovdd@student.bmstu.ru>  
  
  
#include* "Bus.hpp"  
  
std::ifstream &*operator*>>(std::ifstream &inf, Bus &bus) {  
 inf >> bus.\_number >> bus.\_driverInfo >> bus.\_routeNumber;  
 *return* inf;  
}  
  
std::istream &*operator*>>(std::istream & in, Bus & bus) {  
 in >> bus.\_number >> bus.\_driverInfo >> bus.\_routeNumber;  
 *return* in;  
}

**DataBase.hpp**

*// Copyright 2022 qqq <polyakovdd@student.bmstu.ru>  
  
  
#ifndef* CODE\_DATABASE\_HPP  
*#define* CODE\_DATABASE\_HPP  
  
*#include* <fstream>  
*#include* <map>  
*#include* <algorithm>  
*#include* "Bus.hpp"  
  
*void* SeedData(std::map<std::string, Bus> &);  
  
*void* OutBus(std::map<std::string, Bus> &, std::map<std::string, Bus> &);  
  
*void* AddBus(std::map<std::string, Bus> &);  
  
*void* DeleteBus(std::map<std::string, Bus> &);  
  
*void* EnterBus(std::map<std::string, Bus> &, std::map<std::string, Bus> &);  
  
*void* PrintBuses(std::map<std::string, Bus> &, std::map<std::string, Bus> &, *bool*);  
  
*void* updateDB(std::map<std::string, Bus> &, std::map<std::string, Bus> &);  
  
*void* fillData(std::map<std::string, Bus> &, std::map<std::string, Bus> &);  
  
  
*#endif //CODE\_DATABASE\_HPP*

**DataBase.cpp**

*// Copyright 2022 qqq <polyakovdd@student.bmstu.ru>  
  
  
#include* "DataBase.hpp"  
  
  
*void* print(std::map<std::string, Bus> &buses) {  
 *for* (*auto* &bus: buses) {  
 std::cout << bus.second.GetNumber() << ' ' << bus.second.GetDriver() << ' ' << bus.second.GetRoute() << '\n';  
 }  
}  
  
*void* print(std::vector<Bus> &buses) {  
 *for* (*auto* &bus: buses) {  
 std::cout << bus.GetNumber() << ' ' << bus.GetDriver() << ' ' << bus.GetRoute() << '\n';  
 }  
}  
  
*void* SeedData(std::map<std::string, Bus> &park) {  
 std::map<std::string, Bus> inPark = {std::make\_pair("к181ко32", Bus("к181ко32", "Поляков\_Д.Д.", "32А")),  
 std::make\_pair("л228ох222", Bus("л228ох222", "Журмилов\_В.Д.", "11")),  
 std::make\_pair("н540ск32", Bus("н540ск32", "Поляков\_Д.В.", "19Б")),  
 std::make\_pair("и264иу69", Bus("и264иу69", "Валова\_С.В.", "3.14YES")),  
 std::make\_pair("о222чк01", Bus("о222чк01", "Скиба\_Г.С.", "123")),  
 std::make\_pair("с013мм197", Bus("с013мм197", "Камушкин\_П.Н.", "5")),  
 std::make\_pair("с105рф122", Bus("с105рф122", "Попов\_С.Д.", "324Б")),  
 std::make\_pair("п233мп32", Bus("п233мп32", "Рябец\_А.Д.", "32Б"))  
 };  
 park = inPark;  
  
 std::ofstream outFilePark("InParkDatabase.db");  
 std::ofstream outFileRoute("OnRouteDatabase.db");  
 outFilePark.clear();  
 outFileRoute.clear();  
 outFileRoute << "0\n";  
 outFileRoute.close();  
  
 outFilePark << inPark.size() << '\n';  
 *for* (*auto* busPair: inPark) {  
 outFilePark << busPair.second.GetNumber() << ' ' << busPair.second.GetDriver() << ' '  
 << busPair.second.GetRoute() << '\n';  
 }  
 outFilePark.close();  
  
 std::cout << "Данные обновлены\n";  
}  
  
*void* AddBus(std::map<std::string, Bus> &park) {  
 std::cout << "Введите данные об автобусе через пробел (Номер, Инициалы водителя, Номер маршрута):\n";  
 Bus tempBus;  
 std::cin >> tempBus;  
 park.insert(std::make\_pair(tempBus.GetNumber(), tempBus));  
 std::cout << "Автобус добавлен в парк\n";  
}  
  
*void* DeleteBus(std::map<std::string, Bus> &park) {  
 print(park);  
 std::cout << "Введите номер автобуса, который желаете удалить из парка:\n";  
 std::string tempNumber;  
  
 *bool* entering = *true*;  
 *while* (entering) {  
 std::cin >> tempNumber;  
 *if* (park.find(tempNumber) != park.end()) {  
  
 park.erase(tempNumber);  
 std::cout << "Автобус удалён\n";  
 entering = *false*;  
 } *else* {  
 std::cout << "Автобуса с таким номером не найдено, повторите ввод:\n";  
 }  
 }  
}  
  
*void* OutBus(std::map<std::string, Bus> &park, std::map<std::string, Bus> &route) {  
 *if* (park.empty()) {  
 std::cout << "Сейчас в парке нет автобусов\n";  
 *return*;  
 }  
 std::cout << "Список автобусов в парке:\n";  
 print(park);  
 std::cout << "Введите номер автобуса, который желаете вывести на линию:\n";  
 std::string number;  
 *bool* entering = *true*;  
 *while* (entering) {  
 std::cin >> number;  
 *if* (park.find(number) != park.end()) {  
 entering = *false*;  
 route.insert(std::make\_pair(number, park[number]));  
 park.erase(number);  
 std::cout << "Автобус успешно выведен на линию\n";  
 } *else* {  
 std::cout << "Автобуса с таким номером нет, введите снова:\n";  
 }  
 }  
}  
  
*void* EnterBus(std::map<std::string, Bus> &park, std::map<std::string, Bus> &route) {  
 *if* (route.empty()) {  
 std::cout << "Сейчас на маршруте нет автобусов\n";  
 *return*;  
 }  
 std::cout << "Список автобусов на маршруте:\n";  
 print(route);  
 std::cout << "Введите номер автобуса, который желаете снять с линии:\n";  
 std::string number;  
 *bool* entering = *true*;  
 *while* (entering) {  
 std::cin >> number;  
 *if* (route.find(number) != route.end()) {  
 entering = *false*;  
 park.insert(std::make\_pair(number, route[number]));  
 route.erase(number);  
 std::cout << "Автобус успешно снят с линии\n";  
 } *else* {  
 std::cout << "Автобуса с таким номером нет, введите снова:\n";  
 }  
 }  
}  
  
*void* PrintBuses(std::map<std::string, Bus> &park, std::map<std::string, Bus> &route, *bool* key) {  
 std::vector<Bus> prikolPark;  
 *for* (*auto* bus: park) {  
 prikolPark.push\_back(bus.second);  
 }  
 std::vector<Bus> prikolRoute;  
 *for* (*auto* bus: route) {  
 prikolRoute.push\_back(bus.second);  
 }  
 *if* (!key) {  
 std::sort(prikolPark.begin(), prikolPark.end(), [](Bus &bus1, Bus &bus2) {  
 *return* bus1.GetNumber() > bus2.GetNumber();  
 });  
 std::sort(prikolRoute.begin(), prikolRoute.end(), [](Bus &bus1, Bus &bus2) {  
 *return* bus1.GetNumber() > bus2.GetNumber();  
 });  
 } *else* {  
 std::sort(prikolPark.begin(), prikolPark.end(), [](Bus &bus1, Bus &bus2) {  
 *return* bus1.GetRoute() > bus2.GetRoute();  
 });  
 std::sort(prikolRoute.begin(), prikolRoute.end(), [](Bus &bus1, Bus &bus2) {  
 *return* bus1.GetRoute() > bus2.GetRoute();  
 });  
 }  
 std::cout << "Список автобусов в парке:\n";  
 print(prikolPark);  
 std::cout << "Список автобусов на линии:\n";  
 print(prikolRoute);  
  
}  
  
*void* updateDB(std::map<std::string, Bus> &park, std::map<std::string, Bus> &route) {  
 std::ofstream outFilePark("InParkDatabase.db");  
 std::ofstream outFileRoute("OnRouteDatabase.db");  
 outFilePark.clear();  
 outFileRoute.clear();  
  
  
 outFilePark << park.size() << '\n';  
 *for* (*auto* busPair: park) {  
 outFilePark << busPair.first << ' ' << busPair.second.GetDriver() << ' ' << busPair.second.GetRoute() << '\n';  
 }  
 *if* (park.size() == 0) {  
 outFilePark << 0;  
 }  
 outFilePark.close();  
  
 outFileRoute << route.size() << '\n';  
 *for* (*auto* busPair: route) {  
 outFileRoute << busPair.first << ' ' << busPair.second.GetDriver() << ' ' << busPair.second.GetRoute()  
 << '\n';  
 }  
 *if* (route.size() == 0) {  
 outFileRoute << 0;  
 }  
 outFileRoute.close();  
}  
  
*void* fillData(std::map<std::string, Bus> &park, std::map<std::string, Bus> &route) {  
 std::ifstream inFilePark("InParkDatabase.db");  
 std::ifstream inFileRoute("OnRouteDatabase.db");  
 size\_t count;  
 inFilePark >> count;  
 Bus tempBus;  
 *for* (size\_t i = 0; i < count; ++i) {  
 inFilePark >> tempBus;  
 park.insert(std::make\_pair(tempBus.GetNumber(), tempBus));  
 }  
 inFilePark.close();  
  
 inFileRoute >> count;  
 *for* (size\_t i = 0; i < count; ++i) {  
 inFileRoute >> tempBus;  
 route.insert(std::make\_pair(tempBus.GetNumber(), tempBus));  
 }  
 inFileRoute.close();  
}

**Анализ результатов**

       