Валейко Алексей 3530904/90005

Руководитель: Червинский А.П.

**Министерство образования и науки РФ Санкт-Петербургский политехнический университет Петра Великого Институт компьютерных наук и технологий Высшая школа программной инженерии**

Индивидуальное задание ООп

Вариант 5

# Общая постановка задания

В файл вводятся количество строк (n), а далее n строк, в которых содержится Фамилия, Имя водителя, гос. Знак, регион и маршрут автобуса. Программа должна вывести отсортированный по маршруту список, найти наименьший и наибольший номера маршрутов, а также вывести в таблицу все регионы и количество их использований

# Требования

n должно быть целым числом >0

Допускаются только буквы латинского алфавита

Имя и фамилия не могут содержать больше 15 букв и не должны содержать цифр и символов

Гос. Знак должен быть строго в формате БЦЦЦББ (где Б – буква, Ц – цифра)

Регион должен быть отделен пробелом и не может быть отрицательным или больше 999

Номер маршрута не может быть отрицательным, нулевым, больше 999

# Спецификация

Пути к файлам ввода/вывода вводятся пользователем в строках с соответствующими комментариями

При указании неверного пути к файлам выводится сообщение Input/Output file doesn't exist

При некорректно заданном n выводится Wrong number of string

При нарушении требований ввода имени и фамилии, указанных выше, программа выводит: Invalid name

При нарушении формата ввода гос. Знака, выводится: Wrong format of number

При нарушении требований ввода региона, или маршрута, выводятся Invalid name, Invalid route, соответственно

При корректности всех введенных данных программа создает массив, сортирует его и выводит в файл output.txt, указанным первым

Далее, программа выводит во второй файл output1.txt наименьший и наибольший номера маршрутов

И в конце, программа создает список номеров с указанием количества их использований, сортирует по убыванию и вывод в третий файл output2.txt

# Тест план:

Ввод:

1

Zai4ik Nikita V484GQ 124 1

Вывод:

Invalid name

Ввод:

1

Zaichik Nikita 3484GQ 124 1

Вывод:

Wrong format of number

Ввод:

1

Zai4ik Nikita V484GQ 1204 1

Вывод:

Invalid region

Ввод:

1

Zai4ik Nikita V484GQ 124 0

Вывод:

Invalid route

Ввод:

7

Zamyatin Nikita F187TH 178 11

Arshavin Andrey Y648RT 43 2

Alekseenko Nikita X484BC 124 3

Dubrovnyak Grigory U875OI 98 15

Tolstoy Anton A756RO 799 6

Depay Memphis O787AA 81 99

Messi Lionel L901CV 178 184

Вывод:

Output.txt:

Surname Firstname Number Route

Arshavin Andrey Y648RT|43 2

Alekseenko Nikita X484BC|124 3

Tolstoy Anton A756RO|799 6

Zamyatin Nikita F187TH|178 11

Dubrovnyak Grigory U875OI|98 15

Depay Memphis O787AA|81 99

Messi Lionel L901CV|178 184

Output1.txt:

Maximal route: Messi Lionel L901CV|178 184

Minimal route: Arshavin Andrey Y648RT|43 2

Output2.txt:

Route: Counter:

178 2

81 1

98 1

799 1

124 1

43 1

# Программа

## Main.cpp:

#include <iostream>  
#include <string>  
#include <fstream>  
#include <list>  
#include "Bus.h"  
#include "RegistrationPlate.h"  
  
using namespace std;  
  
const string WRONG\_FORM\_NUMBER = "Wrong format of number";  
const string WRONG\_FORM\_REG = "Invalid region";  
const string WRONG\_NAME = "Invalid name";  
const string WRONG\_ROUTE = "Invalid route";  
  
bool compareRegCount(RegistrationPlate &a, RegistrationPlate &b) {  
 return (a.count < b.count);  
}  
  
void check(Bus &other) {  
 if (other.name.length() > 15 or other.surname.length() > 15) {  
 throw WRONG\_NAME;  
 }  
 for (int i = 0; i < other.name.length(); i++) {  
 if (!((other.name[i] >= 'A' and other.name[i] <= 'Z') or (other.name[i] >= 'a' and other.name[i] <= 'z'))) {  
 throw WRONG\_NAME;  
 }  
 }  
 for (int i = 0; i < other.surname.length(); i++) {  
 if (!((other.surname[i] >= 'A' and other.surname[i] <= 'Z') or  
 (other.surname[i] >= 'a' and other.surname[i] <= 'z'))) {  
 throw WRONG\_NAME;  
 }  
 }  
 if (other.number.length() > 6) {  
 throw WRONG\_FORM\_NUMBER;  
 }  
 for (int i = 0; i < 6; i++) {  
 if ((i > 3 or i == 0) and !((other.number[i] >= 'A') and (other.number[i] <= 'Z'))) {  
 throw WRONG\_FORM\_NUMBER;  
 }  
 if ((i > 1) and (i < 4) and !((other.number[i] >= '0') and (other.number[i] <= '9'))) {  
 throw WRONG\_FORM\_NUMBER;  
 }  
 }  
 if ((other.region < 0) or (other.region > 999)) {  
 throw WRONG\_FORM\_REG;  
 }  
 if (other.route < 1 or other.route > 999) {  
 throw WRONG\_ROUTE;  
 }  
}  
  
template<class T>  
T findMax(T \*a, int N) {  
 T max;  
 for (int i = 0; i < N; i++) {  
 if (a[i] > max) {  
 max = a[i];  
 }  
 }  
 return max;  
}  
  
template<class T>  
T findMin(T \*a, int N) {  
 T min = a[0];  
 for (int i = 0; i < N; i++) {  
 if (a[i] < min) {  
 min = a[i];  
 }  
 }  
 return min;  
}  
  
template<class T>  
void sort(T \*a, int n) {  
 T x;  
 int key = 0;  
 for (int i = 0; i < n - 1; i++) {  
 key = i + 1;  
 x = a[key];  
 for (int j = i + 1; j > 0; j--) {  
 if (x < a[j - 1]) {  
 a[j] = a[j - 1];  
 key = j - 1;  
 }  
 }  
 a[key] = x;  
 }  
}  
  
ostream &operator<<(ostream &out, const Bus &other) {  
 out << other.surname << " ";  
 out << setw(10) << other.name << " ";  
 out << setw(10) << other.number << '|' << other.region;  
 out << setw(12) << other.route << endl;  
 return out;  
}  
  
istream &operator>>(istream &in, Bus &other) {  
 in >> other.surname >> other.name >> other.number >> other.region >> other.route;  
 return in;  
}  
  
ostream &operator<<(std::ostream &out, const RegistrationPlate &other) {  
 out << other.region << setw(10) << other.count << endl;  
 return out;  
}  
  
int main() {  
 ifstream in;  
 ofstream out1;  
 ofstream out2;  
 ofstream out3;  
 in.open("/Users/alex/Documents/Progs/Alexei\_Valeiko\_IndOOP/input.txt"); //Put  
 out1.open("/Users/alex/Documents/Progs/Alexei\_Valeiko\_IndOOP/output.txt"); //your  
 out2.open("/Users/alex/Documents/Progs/Alexei\_Valeiko\_IndOOP/output1.txt"); //paths  
 out3.open("/Users/alex/Documents/Progs/Alexei\_Valeiko\_IndOOP/output2.txt"); //there  
 if (!in or !out1 or !out2 or !out3) {  
 cerr << "Input/Output file doesn't exist";  
 exit(2);  
 }  
  
 int n;  
 in >> n;  
 if (n < 1){  
 cerr << "Wrong number of strings";  
 exit(3);  
 }  
 Bus \*buses = new Bus[n];  
 out1 << "Surname" << setw(15) << "Firstname" << setw(15) << "Number" << setw(15) << "Route" << endl;  
  
 for (int i = 0; i < n; i++) {  
 in >> buses[i];  
 try {  
 check(buses[i]);  
 }  
 catch (const string S) {  
 cerr << S;  
 exit(1);  
 }  
 }  
 in.close();  
 sort(buses, n);  
  
  
 list<RegistrationPlate> list;  
 for (size\_t i = 0; i < n; ++i) {  
 RegistrationPlate temp(buses[i].getReg());  
 bool found = false;  
 for (auto &s : list) {  
 if (s == temp) {  
 s++;  
 found = true;  
 break;  
 }  
 }  
 if (!found) {  
 list.push\_back(temp);  
 }  
 }  
 list.sort(compareRegCount);  
 list.reverse();  
  
 out2 << "Maximal route: " << findMax(buses, n) << "Minimal route: " << findMin(buses, n) << endl;  
  
 for (int i = 0; i < n; i++) {  
 out1 << buses[i];  
 }  
  
 out3 << "Route: " << "Counter: " << endl;  
 for (RegistrationPlate s : list) {  
 out3 << s;  
 }  
  
 out1.close();  
 out2.close();  
 out3.close();  
 delete[] buses;  
}

## Bus.h

#ifndef ALEXEY\_PARK\_H  
#define ALEXEY\_PARK\_H  
  
#include <string>  
#include <iostream>  
  
using namespace std;  
  
class Bus {  
 friend void check(Bus &other);  
  
 friend istream &operator>>(istream &in, Bus &other);  
  
 friend ostream &operator<<(ostream &out, const Bus &other);  
  
private:  
 string name, surname;  
 string number;  
 int region;  
 int route;  
public:  
 Bus() {  
 name = "NULL";  
 surname = "NULL";  
 region = 0;  
 route = 0;  
 number = "NULL";  
 }  
  
 Bus(string ln, string fn, string num, int reg, int rt) {  
 name = fn;  
 surname = ln;  
 number = num;  
 region = reg;  
 route = rt;  
 }  
  
 Bus(const Bus &other) {  
 this->name = other.name;  
 this->surname = other.surname;  
 this->number = other.number;  
 this->region = other.region;  
 this->route = other.route;  
 }  
  
 Bus &operator=(const Bus &other) {  
 this->name = other.name;  
 this->surname = other.surname;  
 this->number = other.number;  
 this->region = other.region;  
 this->route = other.route;  
 return \*this;  
 }  
  
 bool operator>(const Bus &other) {  
 return this->route > other.route;  
 }  
  
 bool operator<(const Bus &other) {  
 return this->route < other.route;  
 }  
  
 bool operator>=(const Bus &other) {  
 return this->route >= other.route;  
 }  
  
 bool operator<=(const Bus &other) {  
 return this->route <= other.route;  
 }  
  
 int getReg() {  
 return region;  
 }  
  
 ~Bus() {  
  
 }  
};  
  
  
#endif

## RegistrationPlate.h

#ifndef ALEXEYG\_REGISTRATIONPLATE\_H  
#define ALEXEYG\_REGISTRATIONPLATE\_H  
  
#include <string>  
  
using namespace std;  
  
class RegistrationPlate {  
 friend ostream &operator<<(std::ostream &out, const RegistrationPlate &other);  
  
 friend bool compareRegCount(RegistrationPlate &a, RegistrationPlate &b);  
  
private:  
 unsigned short count;  
 int region;  
public:  
 RegistrationPlate(int reg){  
 region = reg;  
 count = 1;  
 }  
  
 bool operator==(const RegistrationPlate &other) {  
 return this->region == other.region;  
 }  
  
 RegistrationPlate &operator=(int r) {  
 this->region = r;  
 return \*this;  
 }  
  
 const RegistrationPlate operator++(int value) {  
 this->count += 1;  
 }  
  
 void setRegion(int region) {  
 this->region = region;  
 count = 1;  
 }  
  
 bool operator>(const RegistrationPlate &other) {  
 return this->region > other.region;  
 }  
  
 bool operator<(const RegistrationPlate &other) {  
 return this->region < other.region;  
 }  
  
 bool operator!=(const RegistrationPlate &other) {  
 return this->region != other.region;  
 }  
  
 int getReg() {  
 return region;  
 }  
  
};  
  
#endif

# Итоги

Я написал программу, считывающую неотсортированные данные об Автобусном парке (Фамилию, имя водителей, гос. Знак, регион, маршрут автобусов), а далее сортирующую их, выводящую в файл, находящую минимум и максимум среди маршрутов, и выстраивающую таблицу частоты использования регионов в автопарке