1. В здании аэропорта на экранах отображается информация о самолетах, а именно: информация о пункте отправления, пункте назначения, номере рейса, времени прибытия, времени отправления, номере секции для регистрации. Экраны – это средство, которое помогает своевременно зарегистрировать и отправить пассажиров. Важно, чтобы информация на экранах была понятной и верной.

Создайте необходимую информацию в виде таблицы для такого экрана, с помощью класса Aeroflot, содержащего в описании следующие поля: номер рейса; название пункта отправления; название пункта назначения рейса; время прибытия; время отправления; место регистрации.

Напишите код программы, выполняющей следующие действия: ввод с клавиатуры значений полей объектов; сортировку записей в таблице в алфавитном порядке по названию пунктов назначения; вывод на консоль значений полей объектов класса в виде таблицы рейсов; если таких рейсов нет, выдать соответствующее сообщение.

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

class Aeroflot {

public:

string flight\_number;

string departure\_city;

string destination\_city;

string arrival\_time;

string departure\_time;

string registration\_place;

};

class MyClass

{

public:

private:

};

bool compareByDestination(const Aeroflot& a, const Aeroflot& b) {

return a.destination\_city < b.destination\_city;

}

void printFlightTable(const vector<Aeroflot>& flights) {

if (flights.empty()) {

cout << "Нет доступных рейсов." << endl;

return;

}

cout << "Номер рейса\tОтправление\tНазначение\tПрибытие\tОтправление\tМесто регистрации" << endl;

for (const Aeroflot& flight : flights) {

cout << flight.flight\_number << "\t" << flight.departure\_city << "\t" << flight.destination\_city

<< "\t" << flight.arrival\_time << "\t" << flight.departure\_time << "\t" << flight.registration\_place << endl;

}

}

int main() {

setlocale(LC\_ALL, "RU");

vector<Aeroflot> flights;

// Ввод информации о рейсах с клавиатуры

int n;

cout << "Введите количество рейсов: ";

cin >> n;

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

Aeroflot flight;

MyClass flight1;

cout << "Номер рейса: ";

cin >> flight.flight\_number;

cout << "Пункт отправления: ";

cin >> flight.departure\_city;

cout << "Пункт назначения: ";

cin >> flight.destination\_city;

cout << "Время прибытия: ";

cin >> flight.arrival\_time;

cout << "Время отправления: ";

cin >> flight.departure\_time;

cout << "Номер секции для регистрации: ";

cin >> flight.registration\_place;

flights.push\_back(flight);

}

// Сортировка по названию пунктов назначения

sort(flights.begin(), flights.end(), compareByDestination);

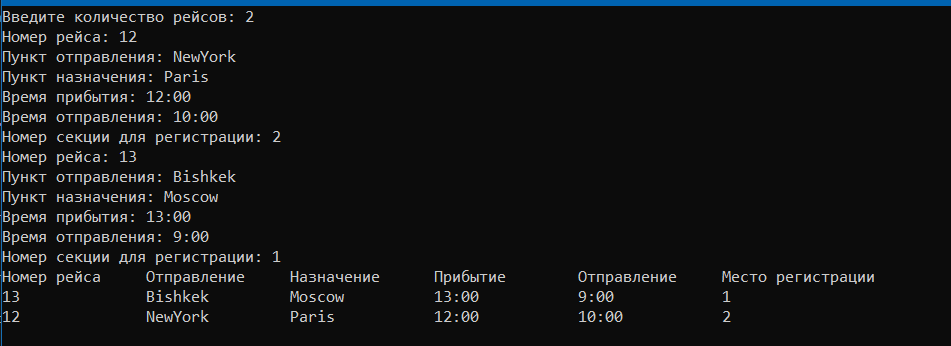
// Вывод таблицы рейсов

printFlightTable(flights);

return 0;

}

Вывод



1. **Опишите класс «студенческая группа».**

Предусмотрите возможность: работы с переменным числом студентов; поиска студента по какому-либо признаку (например, по фамилии, дате рождения или номеру телефона); добавления или удаления записей; сортировки по разным полям.

Разработайте программу, демонстрирующую работу с этим классом.

Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

#include <iostream>

#include <vector>

#include <algorithm>

#include <string>

#include <windows.h>

using namespace std;

class Student {

public:

string firstName;

string lastName;

string dateOfBirth;

string phoneNumber;

};

class StudentGroup {

private:

vector<Student> students;

public:

void addStudent(const string& firstName, const string& lastName, const string& dateOfBirth, const string& phoneNumber) {

students.push\_back({ firstName, lastName, dateOfBirth, phoneNumber });

}

void removeStudentByLastName(const string& lastName) {

students.erase(remove\_if(students.begin(), students.end(),

[lastName](const Student& student) { return student.lastName == lastName; }),

students.end());

}

Student\* findStudentByLastName(const string& lastName) {

auto it = find\_if(students.begin(), students.end(),

[lastName](const Student& student) { return student.lastName == lastName; });

return (it != students.end()) ? &(\*it) : nullptr;

}

void sortStudentsByLastName() {

sort(students.begin(), students.end(), [](const Student& a, const Student& b) {

return a.lastName < b.lastName;

});

}

void displayStudents() {

if (students.empty()) {

cout << "Группа пуста." << endl;

return;

}

cout << "Фамилия\t\tИмя\t\tДата рождения\t\tНомер телефона" << endl;

for (const auto& student : students) {

cout << student.lastName << "\t" << student.firstName << "\t\t"

<< student.dateOfBirth << "\t\t" << student.phoneNumber << endl;

}

}

};

int main() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

setlocale(LC\_ALL, "Ru");

StudentGroup group;

while (true) {

cout << "\nМеню:\n";

cout << "1. Добавить студента\n";

cout << "2. Удалить студента по фамилии\n";

cout << "3. Найти студента по фамилии\n";

cout << "4. Сортировать студентов по фамилии\n";

cout << "5. Вывести список студентов\n";

cout << "0. Выйти из программы\n";

int choice;

cout << "Выберите действие: ";

cin >> choice;

switch (choice) {

case 1: {

string firstName, lastName, dateOfBirth, phoneNumber;

cout << "Введите имя студента: ";

cin >> firstName;

cout << "Введите фамилию студента: ";

cin >> lastName;

cout << "Введите дату рождения студента: ";

cin >> dateOfBirth;

cout << "Введите номер телефона студента: ";

cin >> phoneNumber;

group.addStudent(firstName, lastName, dateOfBirth, phoneNumber);

break;

}

case 2: {

string lastName;

cout << "Введите фамилию студента для удаления: ";

cin >> lastName;

group.removeStudentByLastName(lastName);

break;

}

case 3: {

string lastName;

cout << "Введите фамилию студента для поиска: ";

cin >> lastName;

Student\* foundStudent = group.findStudentByLastName(lastName);

if (foundStudent != nullptr) {

cout << "Студент найден: " << foundStudent->firstName << " " << foundStudent->lastName << endl;

}

else {

cout << "Студент не найден." << endl;

}

break;

}

case 4:

group.sortStudentsByLastName();

cout << "Студенты отсортированы по фамилии." << endl;

break;

case 5:

group.displayStudents();

break;

case 0:

return 0;

default:

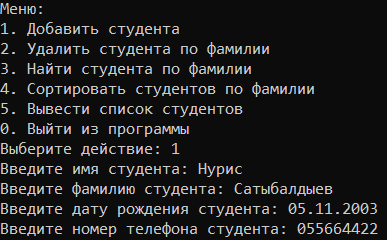
cout << "Неверный выбор. Повторите попытку." << endl;

}

}

return 0;

}

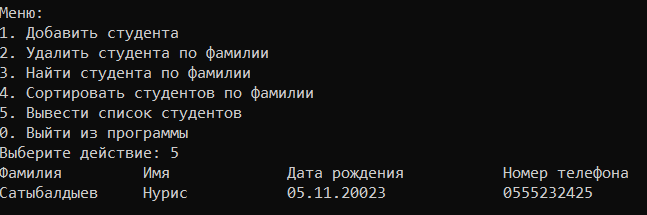


Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание



1. **Опишите класс, реализующий стек (Stack).**

Разработать программу, использующую этот класс для моделирования Т-образного сортировочного узла на железной дороге. Программа должна разделять на два направления состав, состоящий из вагонов двух типов (на каждое направление формируется состав из вагонов одного типа). Предусмотреть возможность формирования состава из файла или с клавиатуры. Возможно использование стандартных функций при работе со стеком в виде контейнера из библиотеки STL:

**push()** - поместить элемент в вершину стека;

**pop()** - удалить элемент из вершины стека;

#include <iostream>

#include <fstream>

#include <stack>

#include <vector>

class Stack {

private:

std::vector<int> data;

public:

void push(int value) {

data.push\_back(value);

}

void pop() {

if (!isEmpty()) {

data.pop\_back();

}

}

bool isEmpty() const {

return data.empty();

}

int top() const {

if (!isEmpty()) {

return data.back();

}

else {

// Возвращаем какой-то специальный флаг или выбрасываем исключение,

// так как в пустом стеке нет вершины

// В данном примере вернем -1 как флаг ошибки

return -1;

}

}

};

void formTrain(Stack& inputStack, Stack& outputStackA, Stack& outputStackB) {

while (!inputStack.isEmpty()) {

int currentWagon = inputStack.top();

inputStack.pop();

// Пример условия для разделения по направлениям

if (currentWagon % 2 == 0) {

outputStackA.push(currentWagon);

}

else {

outputStackB.push(currentWagon);

}

}

}

int main() {

Stack inputStack;

Stack outputStackA;

Stack outputStackB;

int choice;

std::cout << "Choose input method:\n"

<< "1. Manual input\n"

<< "2. Read from file\n";

std::cin >> choice;

if (choice == 1) {

// Ручной ввод

int wagon;

std::cout << "Enter wagon numbers (enter a non-integer to stop):\n";

while (std::cin >> wagon) {

inputStack.push(wagon);

}

}

else if (choice == 2) {

// Ввод из файла

std::ifstream inputFile("train\_composition.txt");

int wagon;

while (inputFile >> wagon) {

inputStack.push(wagon);

}

inputFile.close();

}

else {

std::cout << "Invalid choice. Exiting program.\n";

return 1;

}

formTrain(inputStack, outputStackA, outputStackB);

// Выводим результаты

std::cout << "Train for Direction A:" << std::endl;

while (!outputStackA.isEmpty()) {

std::cout << outputStackA.top() << " ";

outputStackA.pop();

}

std::cout << std::endl;

std::cout << "Train for Direction B:" << std::endl;

while (!outputStackB.isEmpty()) {

std::cout << outputStackB.top() << " ";

outputStackB.pop();

}

std::cout << std::endl;

return 0;

}

Input

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

output

