**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. И.Раззакова**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра: **Программное обеспечение компьютерных систем**

Курс «Объектно-ориентированное программирование»

**ОТЧЕТ**

**Лабораторная работа №2**

Выполнил: студент группы ПИ-3-21

Урманбетов Султан

Проверил: Мусабаев Э.Б.

**Бишкек 2024**

# **Задание №1**

**Постановка задачи:**

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.

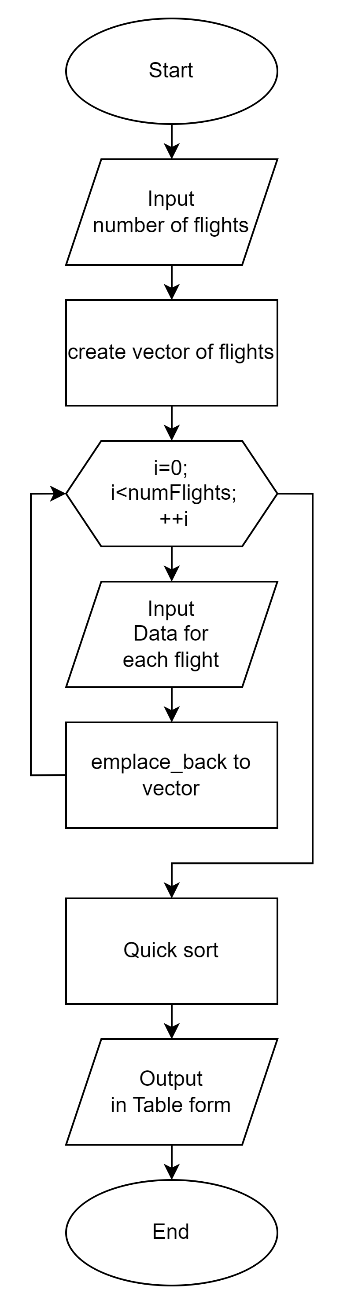
Напишите код программы, выполняющей следующие действия: ввод с клавиатуры значений полей объектов; сортировку записей в таблице в алфавитном порядке по названию пунктов назначения; вывод на консоль значений полей объектов класса в виде таблицы рейсов; если таких рейсов нет, выдать соответствующее сообщение.

**Исходные данные:**

информация о самолетах:

информация о пункте отправления, пункте назначения, номере рейса, времени прибытия, времени отправления, номере секции для регистрации

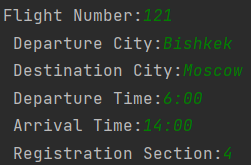
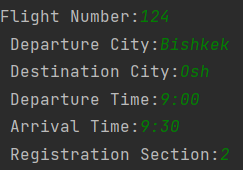
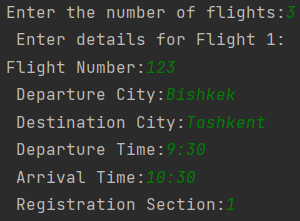
**Блок-схема:**

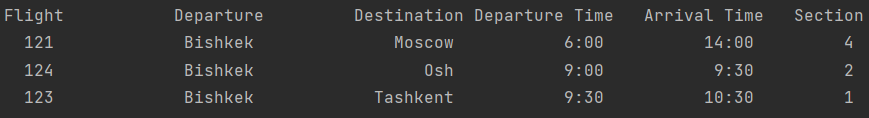


**Исходный код программы на C++:**

#include <bits/stdc++.h>  
using namespace std;  
  
class Aeroflot {  
public:  
 int flightNumber;  
 string departureCity;  
 string destinationCity;  
 string departureTime;  
 string arrivalTime;  
  
 int registrationSection;  
  
 Aeroflot(int flightNum, const string& depCity, const string& destCity,  
 const string& depTime, const string& arrTime, int regSection):  
 flightNumber(flightNum),  
 departureCity(depCity),  
 destinationCity(destCity),  
 departureTime(depTime),  
 arrivalTime(arrTime),  
 registrationSection(regSection) {}  
  
 void displayFlightInfo() const {  
 cout<<setw(5)<<flightNumber  
 <<setw(20)<<departureCity  
 <<setw(20)<<destinationCity  
 <<setw(15)<<departureTime  
 <<setw(15)<<arrivalTime  
 <<setw(10)<<registrationSection<<endl;  
 }  
};  
  
bool compareByDestination(const Aeroflot& a, const Aeroflot& b) {  
 return a.destinationCity < b.destinationCity;  
}  
  
int main() {  
 int numFlights;  
 cout<<"Enter the number of flights: ";  
 cin>>numFlights;  
 vector<Aeroflot> flights;  
  
 for (int i=0; i<numFlights; ++i) {  
 int flightNum, regSection;  
 string depCity, destCity, arrTime, depTime;  
  
 cout<<"Enter details for Flight "<<i + 1<<":"<<endl;  
 cout<<"Flight Number: ";  
 cin>>flightNum;  
 cout<<"Departure City: ";  
 cin>>depCity;  
 cout<<"Destination City: ";  
 cin>>destCity;  
 cout<<"Departure Time: ";  
 cin>>depTime;  
 cout<<"Arrival Time: ";  
 cin>>arrTime;  
 cout<<"Registration Section: ";  
 cin>>regSection;  
  
 flights.emplace\_back(flightNum, depCity, destCity,depTime, arrTime, regSection);  
 }  
  
 sort(flights.begin(), flights.end(), compareByDestination);  
  
 cout<<endl<<setw(5)<<"Flight"<<setw(20)<<"Departure"<<setw(20)  
 <<"Destination"<<setw(15)<<"Departure Time"<<setw(15)  
 <<"Arrival Time"<<setw(10)<<"Section"<<endl;  
  
 for (const auto& flight : flights)  
 flight.displayFlightInfo();  
  
 if (flights.empty())  
 cout<<"No flights available."<<endl;  
  
 return 0;  
}

**Тест:**





# **Задание №2**

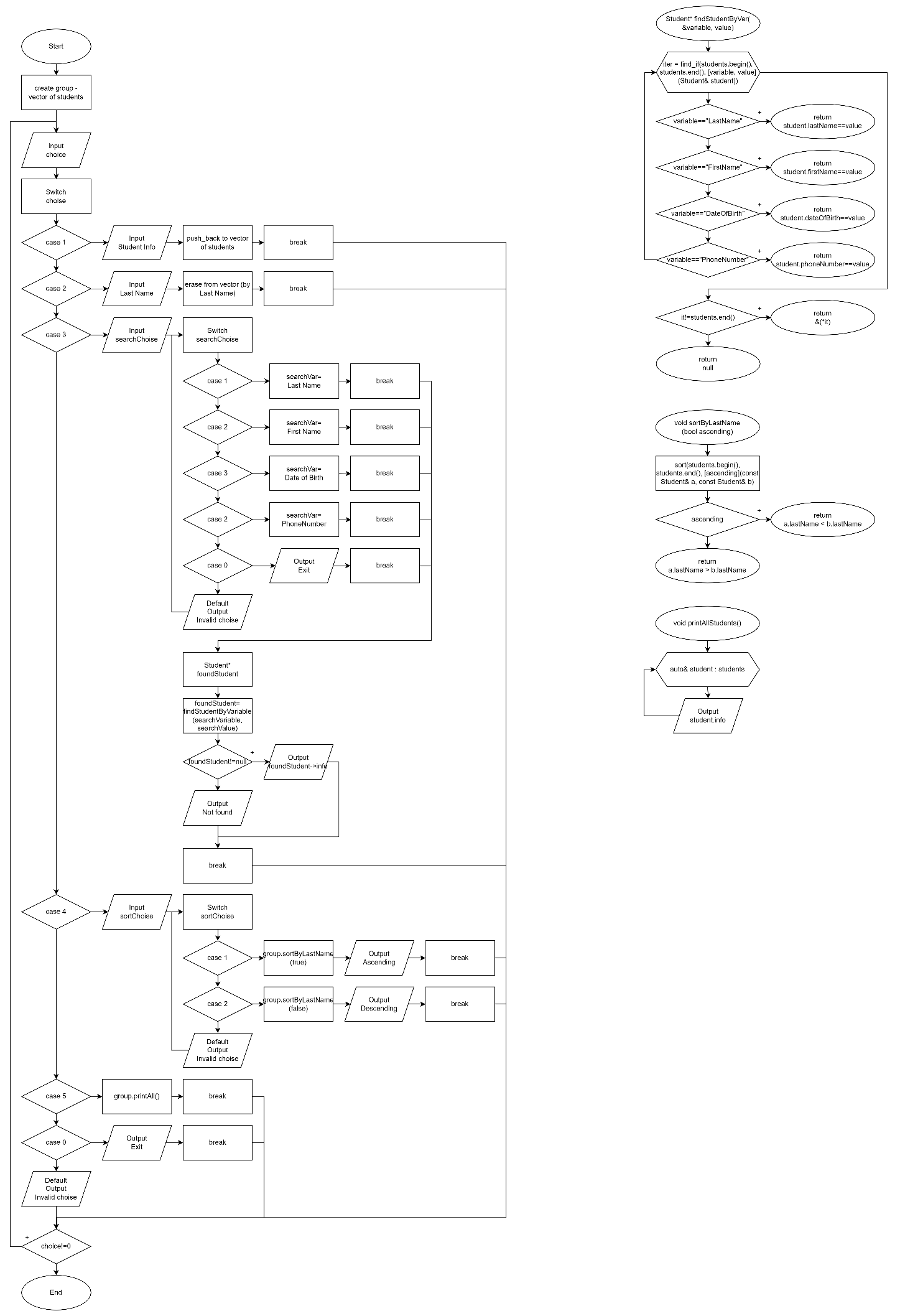
**Постановка задачи: Опишите класс «студенческая группа».**

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.

**Блок-схемы:**

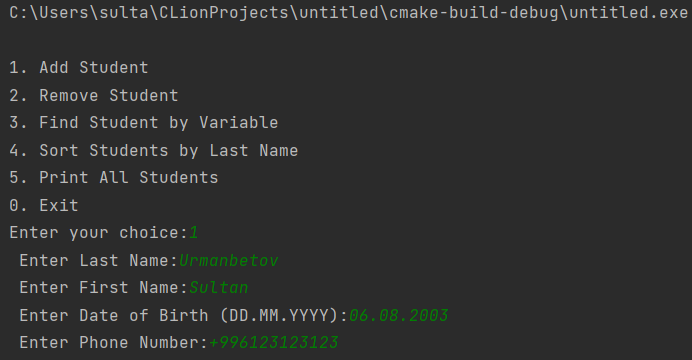


**Исходный код программы на C++:**

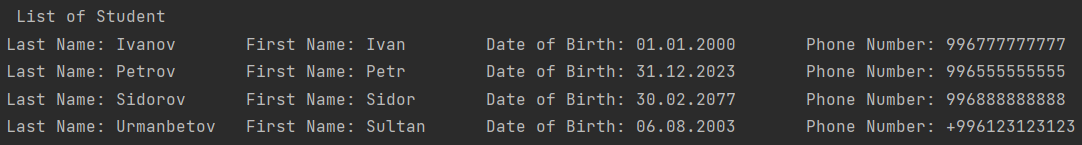
#include <bits/stdc++.h>  
using namespace std;  
  
class Student {  
public:  
 string lastName;  
 string firstName;  
 string dateOfBirth;  
 string phoneNumber;  
  
 Student(const string& last, const string& first, const string& dob, const string& phone)  
 : lastName(last), firstName(first), dateOfBirth(dob), phoneNumber(phone) {}  
};  
  
class StudentGroup {  
private:  
 vector<Student> students;  
  
public:  
 void addStudent(const Student& student) {  
 students.push\_back(student);  
 }  
  
 void removeStudent(const string& lastName) {  
 students.erase(remove\_if(students.begin(), students.end(),  
 [lastName](const Student& student) { return student.lastName == lastName; }), students.end());  
 }  
  
 Student\* findStudentByVariable(const string& variable, const string& value) {  
 auto it=find\_if(students.begin(), students.end(),  
 [variable, value](const Student& student) {  
 if (variable == "LastName") return student.lastName == value;  
 else if (variable == "FirstName") return student.firstName == value;  
 else if (variable == "DateOfBirth") return student.dateOfBirth == value;  
 else if (variable == "PhoneNumber") return student.phoneNumber == value;  
 return false;  
 });  
  
 return (it != students.end()) ? &(\*it) : nullptr;  
 }  
  
 void sortByLastName(bool ascending) {  
 sort(students.begin(), students.end(), [ascending](const Student& a, const Student& b) {  
 if (ascending) {  
 return a.lastName < b.lastName;  
 } else {  
 return a.lastName > b.lastName;  
 }  
 });  
 }  
  
 void printAllStudents() {  
 cout<<"List of Student"<<endl;  
 for (const auto& student : students) {  
 cout<<"Last Name: "<<student.lastName<<"\tFirst Name: "<<student.firstName  
 <<"\tDate of Birth: "<<student.dateOfBirth<<"\tPhone Number: "<<student.phoneNumber<<endl;  
 }  
 }  
};  
  
int main() {  
 StudentGroup group;  
  
 group.addStudent(Student("Ivanov", "Ivan", "01.01.2000", "996777777777"));  
 group.addStudent(Student("Petrov", "Petr", "31.12.2023", "996555555555"));  
 group.addStudent(Student("Sidorov", "Sidor", "30.02.2077", "996888888888"));  
  
 int choice, searchChoice, sortChoice;  
 string searchVariable, searchValue;  
  
 do {  
 cout<<"\n1. Add Student\n2. Remove Student\n3. Find Student by Variable\n"  
 "4. Sort Students by Last Name\n5. Print All Students\n0. Exit\n";  
 cout<<"Enter your choice: ";  
 cin>>choice;  
  
 switch (choice) {  
 case 1: {  
 string last, first, dob, phone;  
 cout<<"Enter Last Name: ";  
 cin>>last;  
 cout<<"Enter First Name: ";  
 cin>>first;  
 cout<<"Enter Date of Birth (DD.MM.YYYY): ";  
 cin>>dob;  
 cout<<"Enter Phone Number: ";  
 cin>>phone;  
  
 group.addStudent(Student(last, first, dob, phone));  
 break;  
 }  
 case 2: {  
 cout<<"Enter Last Name to remove: ";  
 cin>>searchValue;  
 group.removeStudent(searchValue);  
 break;  
 }  
 case 3: {  
 cout<<"Enter your choice: ";  
 cout<<"\n1. Last Name\n2. First Name\n3. Date Of Birth\n4. PhoneNumber\n0. Exit\n";  
 cin>>searchChoice;  
 switch (searchChoice) {  
 case 1:  
 searchVariable="LastName";  
 break;  
 case 2:  
 searchVariable="FirstName";  
 break;  
 case 3:  
 searchVariable="DateOfBirth";  
 break;  
 case 4:  
 searchVariable="PhoneNumber";  
 break;  
 case 0:  
 cout<<"Exiting program.\n";  
 break;  
 default:  
 cout<<"Invalid choice. Please enter a valid option.\n";  
 continue;  
 }  
 cout<<"Enter Value to search: ";  
 cin>>searchValue;  
  
 Student\* foundStudent=group.findStudentByVariable(searchVariable, searchValue);  
 if (foundStudent != nullptr) {  
 cout<<"Student found:\n";  
 cout<<"Last Name: "<<foundStudent->lastName<<"\tFirst Name: "<<foundStudent->firstName  
 <<"\tDate of Birth: "<<foundStudent->dateOfBirth<<"\tPhone Number: "<<foundStudent->phoneNumber<<endl;  
 } else {  
 cout<<"Student not found.\n";  
 }  
 break;  
 }  
 case 4:{  
 cout << "Enter your choice for sorting:\n";  
 cout << "1. Sort by Last Name Ascending\n";  
 cout << "2. Sort by Last Name Descending\n";  
 cin >> sortChoice;  
 switch (sortChoice) {  
 case 1:  
 group.sortByLastName(true);  
 cout << "Students sorted by Last Name (Ascending).\n";  
 break;  
 case 2:  
 group.sortByLastName(false);  
 cout << "Students sorted by Last Name (Descending).\n";  
 break;  
 default:  
 cout << "Invalid choice. Please enter a valid option.\n";  
 }  
 break;  
 }  
 case 5:  
 group.printAllStudents();  
 break;  
 case 0:  
 cout<<"Exiting program.\n";  
 break;  
 default:  
 cout<<"Invalid choice. Please enter a valid option.\n";  
 }  
 } while (choice!=0);  
  
 return 0;  
}

**Тесты:**

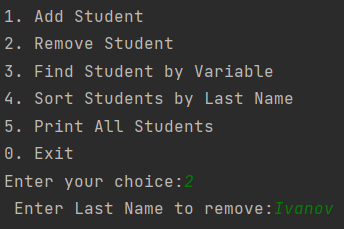
**Тест№1: Add Student**

****

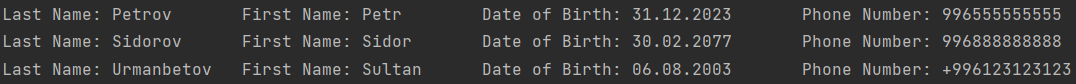
**Check**

****

**Тест№2: Remove Student**

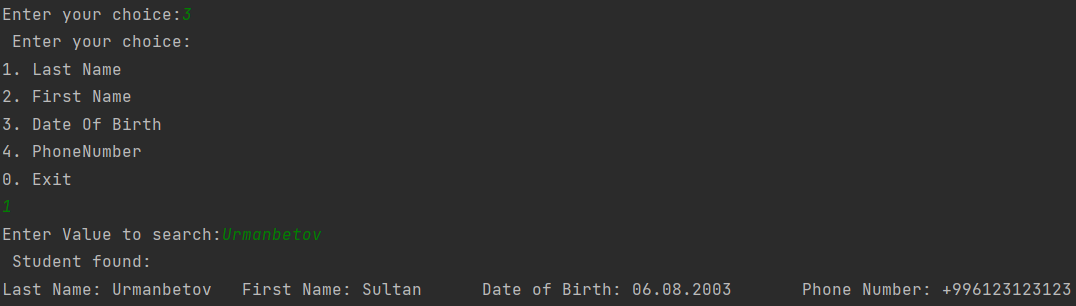
****

**Check**

****

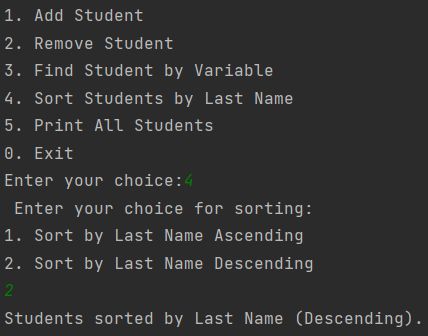
**Тест№3: Search Student**

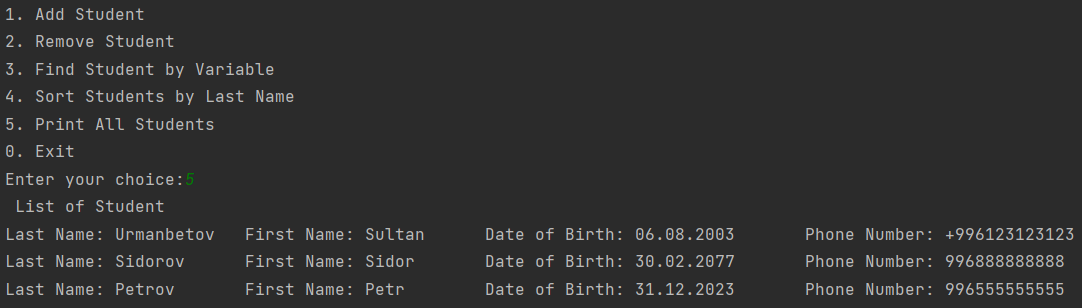
**By LastName**

****

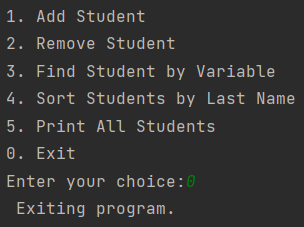
**By Date of Birth**

**Тест№4: Sort Student By Last Name (Ascending/Descending)**

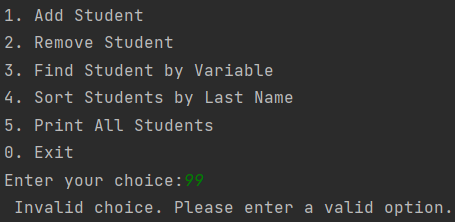
****

**Тест№5: Print All Students **

**Тест№0: Exit**

****

**Тест№?: Choice**

****

# **Задание №3**

**Постановка задачи:**

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;

**Исходный код программы на C++:**

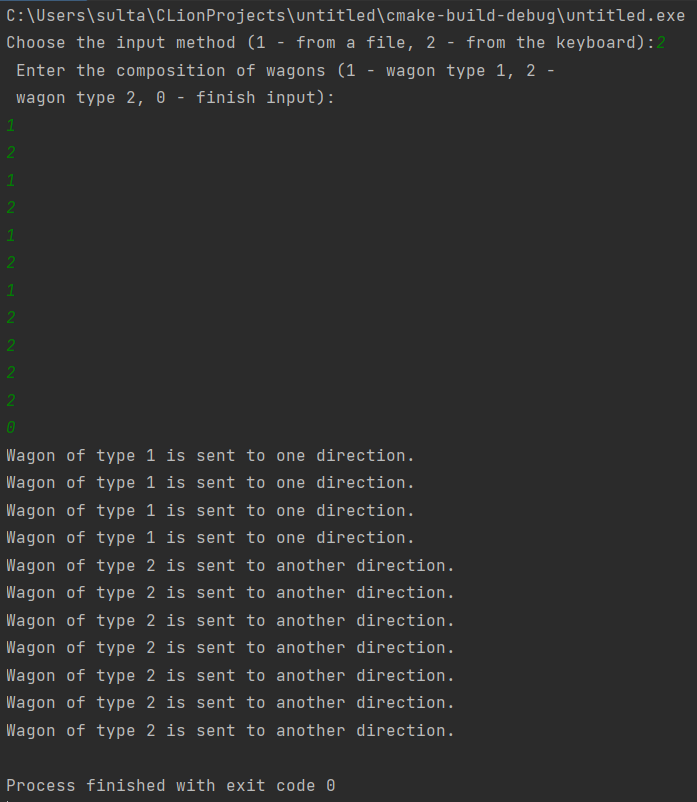
#include <bits/stdc++.h>  
using namespace std;

class TSortNode {  
public:  
 void addWagon(int wagonType) {  
 if (wagonType == 1) {  
 stack1.push(wagonType);  
 } else if (wagonType == 2) {  
 stack2.push(wagonType);  
 }  
 }  
 void splitComposition() {  
 while (!stack1.empty()) {  
 cout << "Wagon of type 1 is sent to one direction." << endl;  
 stack1.pop();  
 }  
  
 while (!stack2.empty()) {  
 cout << "Wagon of type 2 is sent to another direction." << endl;  
 stack2.pop();  
 }  
 }  
  
private:  
 stack<int> stack1;  
 stack<int> stack2;  
};  
  
int main() {  
 TSortNode sortingNode;  
 cout << "Choose the input method (1 - from a file, 2 - from the keyboard): ";  
 int inputChoice;  
 cin >> inputChoice;  
  
 if (inputChoice == 1) {  
 ifstream inputFile("input.txt");  
 if (!inputFile.is\_open()) {  
 cout << "Error opening the file." << endl;  
 return 1;  
 }  
  
 int wagonType;  
 while (inputFile >> wagonType) {  
 sortingNode.addWagon(wagonType);  
 }  
  
 inputFile.close();  
 } else if (inputChoice == 2) {  
 cout << "Enter the composition of wagons (1 - wagon type 1, 2 - wagon type 2, 0 - finish input):" << endl;  
 int wagonType;  
  
 while (true) {  
 cin >> wagonType;  
 if (wagonType == 0) {  
 break;  
 }  
  
 sortingNode.addWagon(wagonType);  
 }  
 } else {  
 cout << "Invalid choice." << endl;  
 return 1;  
 }  
 sortingNode.splitComposition();  
  
 return 0;  
}

**Тесты:**

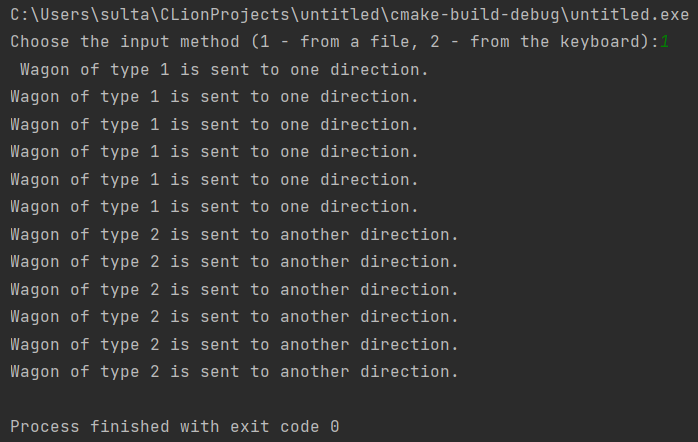
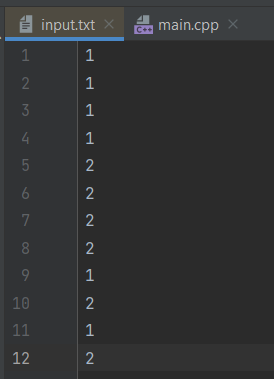
**Тест№1:**

**keyboard**

****

**Тест№2:**

**File**



**Ответы на вопросы:**

1. **Как вы понимаете, что такое класс и объект? Приведите примеры.**

Шаблоном или описанием объекта является класс, а объект представляет экземпляр этого класса. Можно еще провести следующую аналогию. У нас у всех есть некоторое представление о человеке - наличие двух рук, двух ног, головы, туловища и т.д. Есть некоторый шаблон - этот шаблон можно назвать классом. Реально же существующий человек (фактически экземпляр данного класса) является объектом этого класса.

1. **Каков синтаксис и смысл определения класса? Как называются составные части класса?**

**Синтаксис определения класса:** Класс определяется с использованием ключевого слова **class**, за которым следует имя класса. Тело класса заключается в фигурные скобки **{}** и содержит переменные и функции-члены класса.

**Смысл определения класса** - это как план для создания объектов. Класс говорит, какие данные (переменные) и действия (функции) могут быть у объектов этого класса, но сам по себе класс не занимает память. Память выделяется только когда мы создаем объект на основе этого класса. Класс - это как чертеж, а объект - это уже дом, построенный по этому чертежу.

**Составные части класса называются полями и методами.**

Поля - это переменные, которые хранят данные объекта класса. Например, в классе "Автомобиль" поля могут быть "марка", "модель", "цвет".

Методы - это функции, которые определяют действия, которые объект класса может выполнять. Например, в классе "Автомобиль" методом может быть "завести двигатель".

1. **Что такое идентификаторы доступа к элементам класса? Дайте определения.**

3 основных идентификатора доступа:

* **public:** Элементы класса, объявленные с public, являются общедоступными и могут быть доступны извне класса. Это означает, что данные элементы (поля и методы) могут быть использованы в других частях программы.
* **private:** Элементы класса, объявленные с private, являются закрытыми и недоступными извне класса. Они могут быть использованы только внутри самого класса.
* **protected:** Элементы класса, объявленные с protected, имеют ограниченный доступ и могут быть использованы как внутри класса, так и в производных классах (наследниках). Они не доступны извне класса и его производных классов.

Идентификаторы доступа позволяют регулировать уровень инкапсуляции и защиты данных и методов в классе, обеспечивая контролируемый доступ к ним из других частей программы.

1. **Что такое инкапсуляция?**

**Инкапсуляция** – это механизм, который объединяет данные и код, манипулирующий с этими данными, а также защищает и то и другое от внешнего вмешательства или неправильного использования. Позволяет скрыть конкретную реализацию класса, облегчая отладку и модификацию программ.

1. **Где в программе с классами можно создавать объекты?**

Список объектов можно определить позднее в функции **main().**

1. **Каков смысл объявления объектов?**

Смысл объявления объектов заключается в создании конкретных экземпляров классов, которые могут хранить данные и использовать методы, определенные в классе, для выполнения конкретных задач в программе.

1. **В какой момент метод готов для вызова объектом?**

Метод готов для вызова объектом в момент, когда объект был создан и инициализирован, и метод был объявлен и определен в классе.

1. **Как называются методы, встроенные в структуру класса?**

Методы, встроенные в структуру класса, называются методами-членами класса или методами объекта класса.

1. **Как определяется метод, если внутри класса записан только его заголовок, сам метод определен вне класса?**

Если внутри класса записан только заголовок метода, а его определение находится вне класса, то это называется "разделение метода" (separate method definition). Это позволяет вынести реализацию метода за пределы класса, что может быть полезно для улучшения читаемости кода и разделения интерфейса класса от его реализации.

1. **Если в классе два поля данных и два объекта, сколько полей принадлежит каждому объекту? Совпадет ли имена и значения этих полей для объектов?**

Если в классе есть два поля данных и создано два объекта этого класса, то каждому объекту принадлежит полный набор этих двух полей данных. Имена полей для обоих объектов будут одинаковыми, но значения полей могут различаться, так как каждый объект имеет свои собственные значения для своих полей данных.

Другими словами, имена полей будут общими для всех объектов данного класса, но значения полей будут индивидуальными для каждого объекта.

1. **Тиражируются ли методы класса?**

Да, методы класса тиражируются, что означает, что они существуют только в одном экземпляре в памяти, независимо от количества объектов этого класса, и могут вызываться для каждого объекта этого класса.

1. **Как в программе написать доступ к открытым и закрытым полям класса?**

Для доступа к открытым полям класса (публичным членам) можно использовать их имена напрямую из объекта класса. Для доступа к закрытым полям (приватным членам) класса, обычно используют публичные методы (геттеры и сеттеры) класса.