**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

**Икрамов Абдул Муним**

Проверил: Мусабаев Э.Б.

**Бишкек 2023**

# **Лабораторная работа №2**

**Постановка задачи:**

**Тема: Классы и объекты**

**Задания:**

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.

**Исходный код программы на C++:**

#include <iostream>

#include <string>

#include <algorithm>

#include <ctime>

using namespace std;

class aeroflot {

public:

int number;

string otkuda;

string kuda;

string pribytie;

string otpravlenie;

int numRegstr;

aeroflot() {}

aeroflot(int number, string otkuda, string kuda, string pribytie, string otpravlenie, int numRegstr) {

this->number = number;

this->otkuda = otkuda;

this->kuda = kuda;

this->pribytie = pribytie;

this->otpravlenie = otpravlenie;

this->numRegstr = numRegstr;

}

};

bool sravnit(aeroflot a, aeroflot b) {

return a.kuda < b.kuda;

}

int main() {

int n;

cout << "Введите количество рейсов: ";

cin >> n;

aeroflot\* raises = new aeroflot[n];

// Ввод данных с клавиатуры

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

cout << "Введите данные для рейса #" << i + 1 << ":\n";

cout << "Номер рейса: ";

cin >> raises[i].number;

cout << "Пункт отправления: ";

cin >> raises[i].otkuda;

cout << "Пункт назначения: ";

cin >> raises[i].kuda;

cout << "Время прибытия:";

cin >> raises[i].pribytie;

cout << "Время отправления:";

cin >> raises[i].otpravlenie;

cout << "Номер секции для регистрации: ";

cin >> raises[i].numRegstr;

}

// Сортировка рейсов по пункту назначения

sort(raises, raises + n, sravnit);

// Вывод данных

cout << "\nТаблица рейсов:\n";

cout << "------------------------------------------------------------------------\n";

cout << "| Номер | Отправление | Назначение | Прибытие | Отправление | Секция |\n";

cout << "------------------------------------------------------------------------\n";

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

cout << "| " << raises[i].number << " | " << raises[i].otkuda << " | " << raises[i].kuda << " | " << raises[i].pribytie << " | " << raises[i].otpravlenie << " | " << raises[i].numRegstr << " |\n";

}

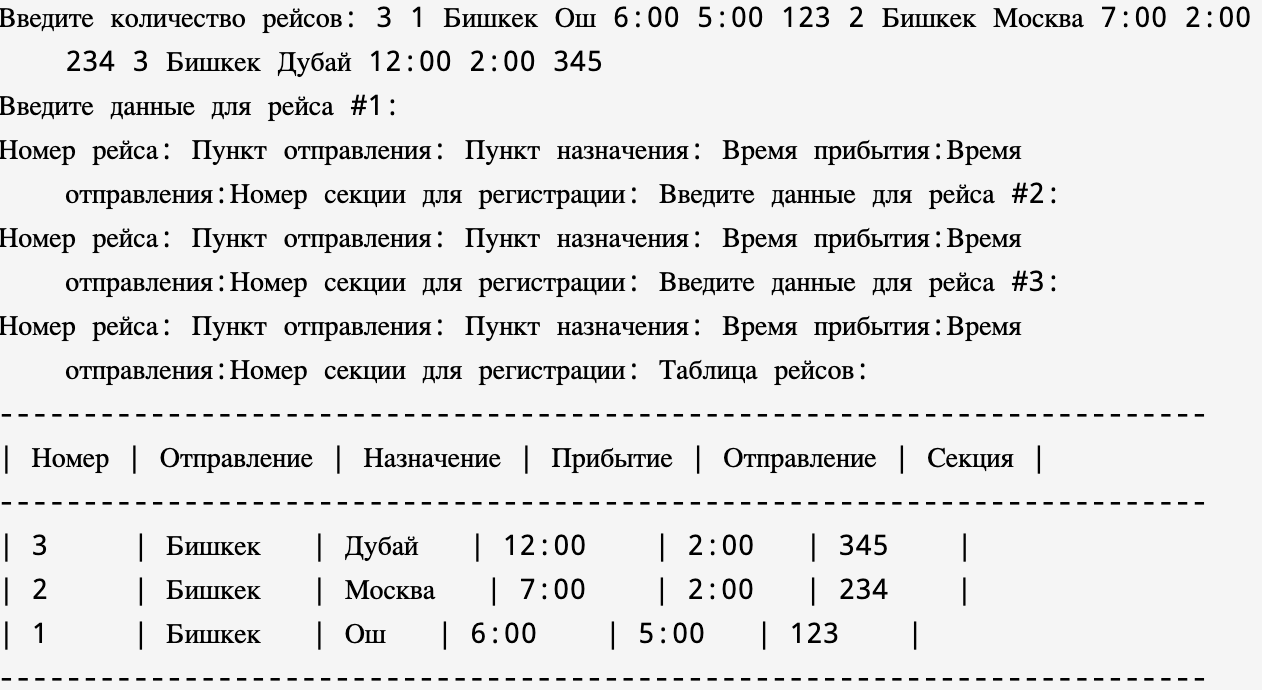
cout << "------------------------------------------------------------------------\n";

delete[] raises;

return 0;

}

Тест:



**2. 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.

**Исходный код программы на C++:**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class Student

{

public:

string name;

string surname;

string patronymic;

int group;

int cours;

Student() {}

Student(string name, string surname, string patronymic, int group, int cours)

{

this->name = name;

this->surname = surname;

this->patronymic = patronymic;

this->group = group;

this->cours = cours;

}

};

void displayStudents(const vector<Student> &students)

{

cout << "\nТаблица студентов:\n";

cout << "------------------------------------------------------------------------\n";

cout << "| Имя | Фамилия | Отчество | Группа | Курс |\n";

cout << "------------------------------------------------------------------------\n";

if (students.empty())

{

cout << "Студентов нет" << endl;

}

else

{

for (const auto &student : students)

{

cout << "|" << student.name << "|" << student.surname << "|" << student.patronymic << "|" << student.group << "| " << student.cours << "|\n";

}

}

cout << "------------------------------------------------------------------------\n";

}

int main()

{

vector<Student> students;

bool exitProgram = false;

cout << "Введите количество студентов: ";

int n;

cin >> n;

// Ввод информации для каждого студента

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

{

cout << "Введите данные для студента " << i + 1 << ":" << endl;

Student newStudent;

cout << "Имя: ";

cin >> newStudent.name;

cout << "Фамилия: ";

cin >> newStudent.surname;

cout << "Отчество: ";

cin >> newStudent.patronymic;

cout << "Группа: ";

cin >> newStudent.group;

cout << "Курс: ";

cin >> newStudent.cours;

students.push\_back(newStudent);

}

while (!exitProgram)

{

cout << "Введите действие над студентами " << endl;

cout << "1 - добавление" << endl;

cout << "2 - удаление" << endl;

cout << "3 - поиск" << endl;

cout << "4 - сортировка" << endl;

cout << "5 - выход" << endl;

int action;

cin >> action;

switch (action)

{

case 1:

{

// Добавление нового студента

Student newStudent;

cout << "Введите данные для нового студента:" << endl;

cout << "Имя: ";

cin >> newStudent.name;

cout << "Фамилия: ";

cin >> newStudent.surname;

cout << "Отчество: ";

cin >> newStudent.patronymic;

cout << "Группа: ";

cin >> newStudent.group;

cout << "Курс: ";

cin >> newStudent.cours;

students.push\_back(newStudent);

}

break;

case 2:

{

// Удаление студента по фамилии

if (!students.empty())

{

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

string surnameToDelete;

cin >> surnameToDelete;

int indexToDelete = -1; // Индекс студента для удаления, -1 означает, что студент не найден

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

{

if (students[i].surname == surnameToDelete)

{

indexToDelete = i;

break;

}

}

if (indexToDelete != -1)

{

// Удаление студента

for (int i = indexToDelete; i < students.size() - 1; ++i)

{

students[i] = students[i + 1];

}

students.pop\_back();

cout << "Студент успешно удален." << endl;

}

else

{

cout << "Студент с указанной фамилией не найден." << endl;

}

}

else

{

cout << "Студентов нет для удаления." << endl;

}

}

break;

case 3:

{

// Поиск студента по фамилии

if (!students.empty())

{

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

string surnameToSearch;

cin >> surnameToSearch;

bool found = false;

for (const auto &student : students)

{

if (student.surname == surnameToSearch)

{

cout << "Студент найден:\n";

displayStudents({student});

found = true;

break;

}

}

if (!found)

{

cout << "Студент с указанной фамилией не найден." << endl;

}

}

else

{

cout << "Студентов нет для поиска." << endl;

}

}

break;

case 4:

// Сортировка студентов по фамилии (пузырьковая сортировка)

if (!students.empty())

{

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

{

for (int j = 0; j < students.size() - i - 1; ++j)

{

if (students[j].surname > students[j + 1].surname)

{

// Обмен элементов

swap(students[j], students[j + 1]);

}

}

}

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

}

else

{

cout << "Студентов нет для сортировки." << endl;

}

break;

case 5:

exitProgram = true;

break;

default:

cout << "Некорректный ввод" << endl;

break;

}

// Вывод информации о студентах

displayStudents(students);

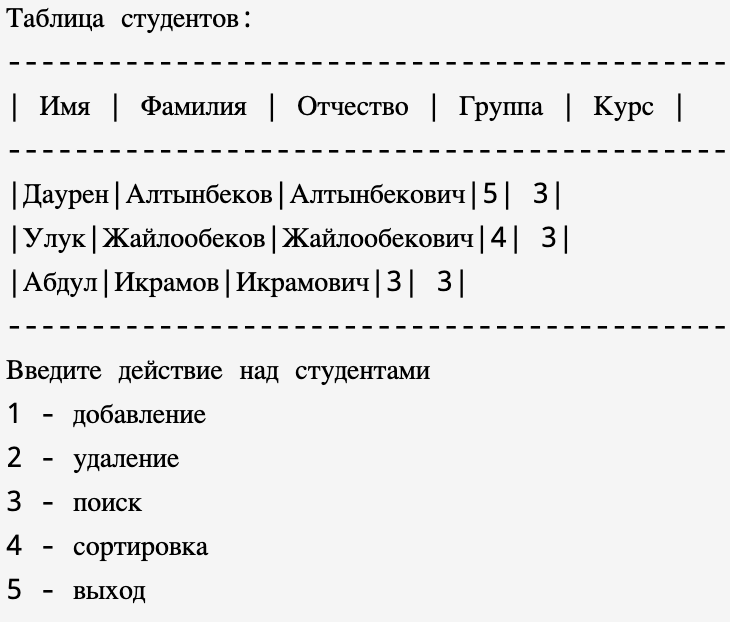
}

return 0;

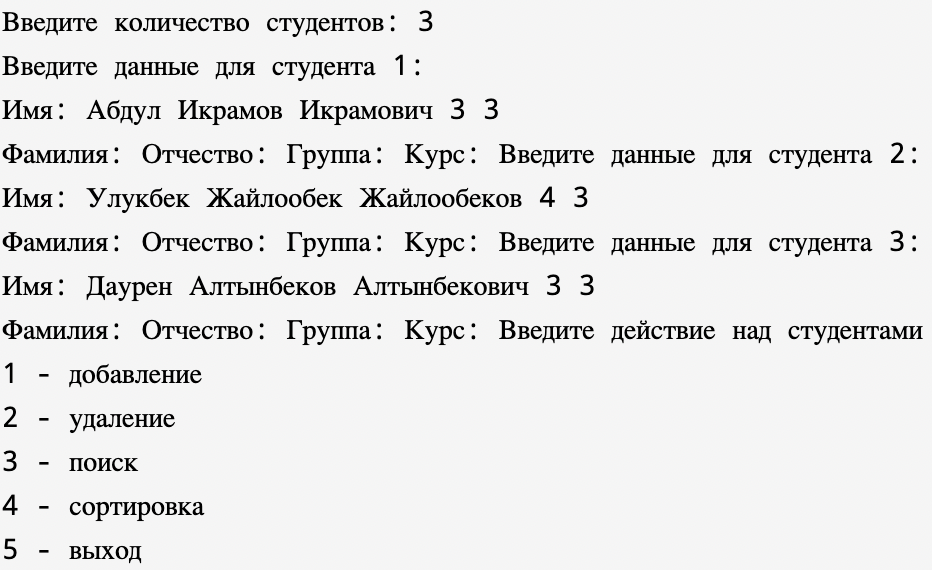
}

Тесты:

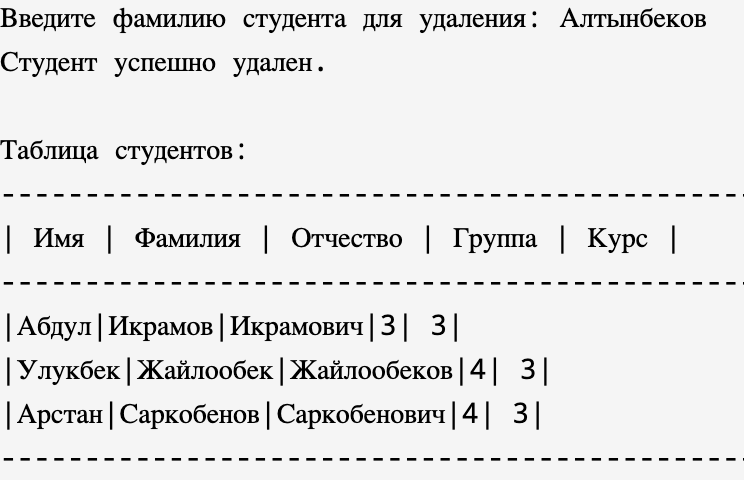
Меню:



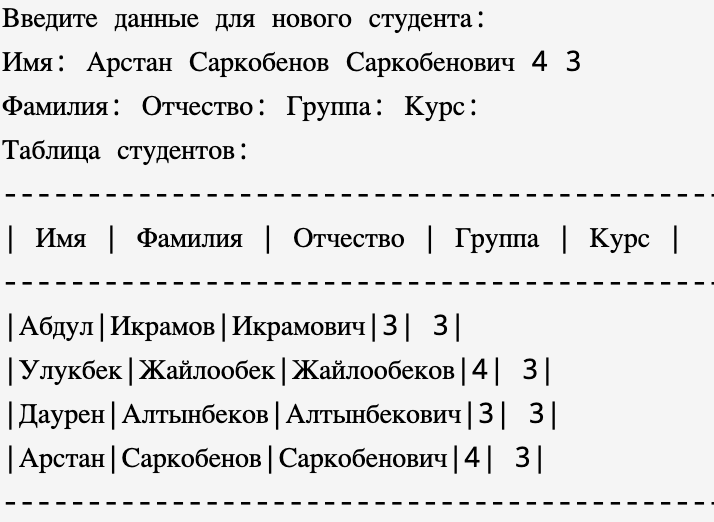
**Добавление студентов(1)**

****

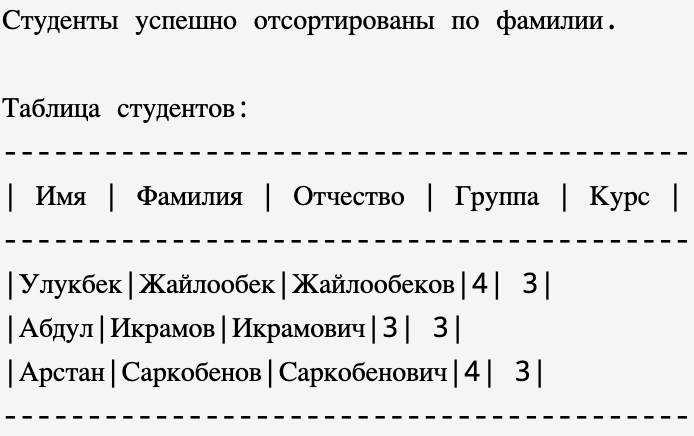
**Удаление студента(2)**



**Поиск студента(3)**



**Сортировка по фамилии(4)**



**3. 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;

**Исходный код программы на C++:**

#include <iostream>

#include <fstream>

#include <stack>

#include <string>

#include <vector>

using namespace std;

class Train {

public:

string type; // тип вагона

int number; // номер вагона

Train(string t, int n) : type(t), number(n) {}

};

class TrainSortingNode {

private:

stack<Train> leftDirection; // Вагоны направляющиеся влево

stack<Train> rightDirection; // Вагоны направляющиеся вправо

public:

void addTrainsToDirection(const Train &train, const string &direction) {

if (direction == "влево") {

leftDirection.push(train);

} else if (direction == "вправо") {

rightDirection.push(train);

} else {

cout << "Некорректное направление.\n";

}

}

void displayTrains() {

cout << "Составы по направлениям:\n";

cout << "Влево:\n";

displayStack(leftDirection);

cout << "Вправо:\n";

displayStack(rightDirection);

}

void displayStack(stack<Train> &s) {

if (s.empty()) {

cout << "Стек пуст.\n";

return;

}

stack<Train> temp = s;

cout << "\nТаблица вагонов:\n";

cout << "------------------------------------\n";

cout << "| тип | номер |\n";

while (!temp.empty()) {

cout << '|' << temp.top().type << " | " << temp.top().number << "|\n";

temp.pop();

}

cout << "------------------------------------\n";

cout << endl;

}

void loadTrainsFromConsole() {

int n;

cout << "Введите количество вагонов: ";

cin >> n;

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

string type, direction;

int number;

cout << "Введите данные для вагона #" << i + 1 << ":\n";

cout << "Тип (грузовой/пассажирский): ";

cin >> type;

cout << "Номер: ";

cin >> number;

cout << "Направление (влево/вправо): ";

cin >> direction;

addTrainsToDirection(Train(type, number), direction);

}

}

};

int main() {

TrainSortingNode sortingNode;

int choice;

do {

cout << "\nМеню:\n";

cout << "1. Ввести составы\n";

cout << "2. Показать составы\n";

cout << "3. Выход\n";

cout << "Выберите действие: ";

cin >> choice;

switch (choice) {

case 1:

sortingNode.loadTrainsFromConsole();

cout << "Составы загружены с клавиатуры.\n";

break;

case 2:

sortingNode.displayTrains();

break;

case 3:

cout << "Выход из программы.\n";

break;

default:

cout << "Некорректный выбор.\n";

}

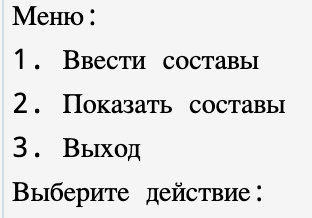
} while (choice != 3);

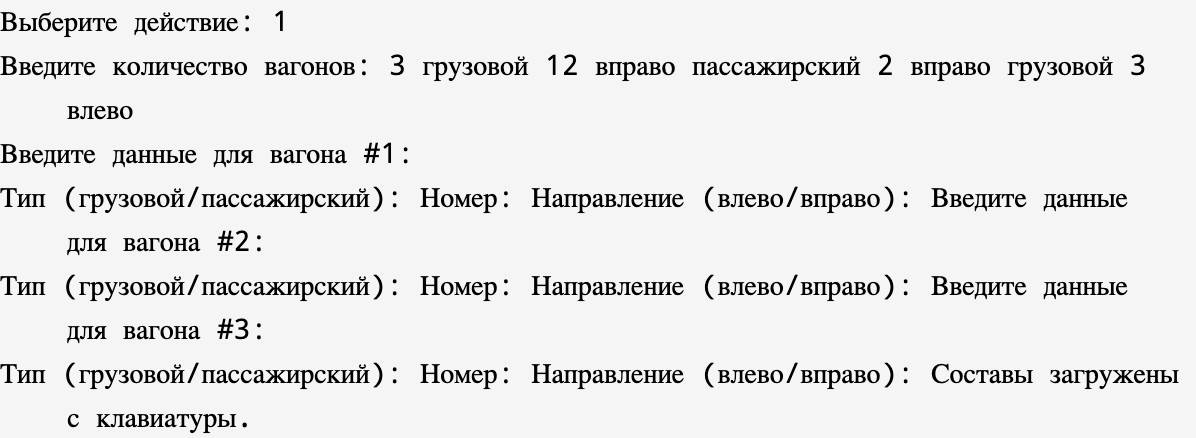
return 0;

}

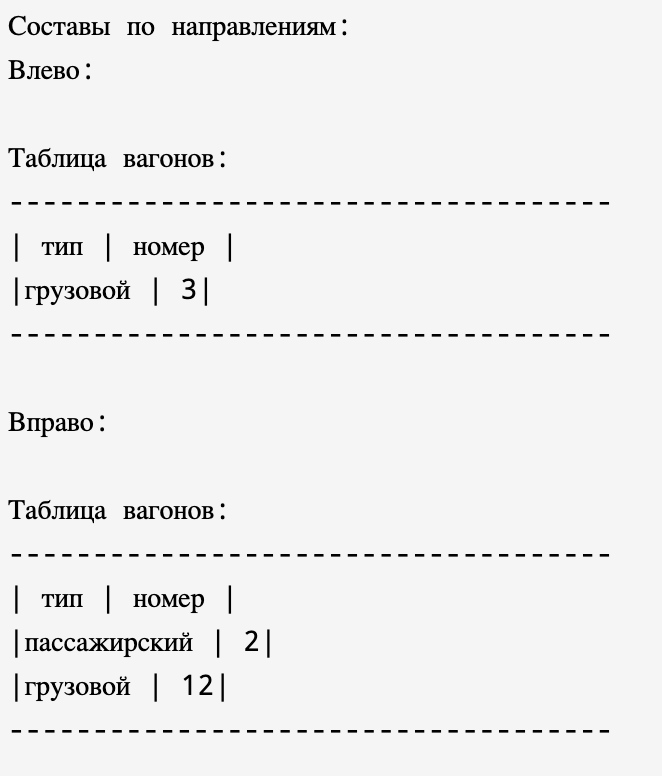
Тесты:

**Ввод составов(1)**





**показ составов (2)**

****