**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

Копжашаров Азамат

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:**

**Исходный код программы на C++:**

// 2.1  
#include <iostream>  
#include <vector>  
#include <algorithm>  
using namespace std;  
class Aeroflot{  
public:  
 int flightNumber;  
 int registrationSection;  
 string departure;  
 string arrivalTime;  
 string destination;  
 string departureTime;  
 Aeroflot(int number, const string& departure, const string& arrival,  
 const string& destination, const string& departureTime, int section)  
 : flightNumber(number), departure(departure), arrivalTime(arrival), destination(destination),  
 departureTime(departureTime), registrationSection(section) {}  
 void outputFlightDetails() const {  
 cout << flightNumber << " " << departure << " " << arrivalTime << " " << destination << " " << departureTime << " " << registrationSection << endl;  
 }  
};  
bool compareFlights(const Aeroflot& f1, const Aeroflot& f2) {  
 return f1.destination < f2.destination;  
}  
int main() {  
 int numOfFlights;  
 cout << "Enter number of flights: " << endl;  
 cin >> numOfFlights;  
 vector<Aeroflot> Flights;  
 cout << "Enter flights in following order:" << endl;  
 cout << "Flight number, departure, arrival time, destination, departure time, registration section" << endl;  
 for(int i = 0; i < numOfFlights; i++){  
 int number, section;  
 string dep, arrTime, dest, depTime;  
 cin >> number >> dep >> arrTime >> dest >> depTime >> section;  
 Flights.emplace\_back(number, dep, arrTime, dest, depTime, section);  
 }  
 sort(Flights.begin(), Flights.end(), compareFlights);  
 for (const auto& flight : Flights) {  
 flight.outputFlightDetails();  
 }  
 return 0;  
}  
/\*  
5  
1 Bishkek 10:10 Osh 10:50 123  
2 Bishkek 12:00 Istanbul 15:30 332  
3 Bishkek 13:20 Amsterdam 16:00 222  
4 Bishkek 11:00 Tokyo 13:00 412  
5 Bishkek 10:30 Moscow 14:50 322  
 \*/

**Тесты:**

# **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.

**Исходные данные:**

**Исходный код программы на C++:**

// 2.2  
#include <iostream>  
#include <vector>  
#include <algorithm>  
using namespace std;  
  
class Student{  
public:  
 string lastName;  
 string firstName;  
 string phoneNumber;  
 string birthDate;  
  
 Student(const string& ln, const string& fn, const string& pn, const string& bd)  
 : lastName(ln), firstName(fn), phoneNumber(pn), birthDate(bd) {}  
  
};  
  
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\* findStudentByLastName(const string& lastName) {  
 auto iter = find\_if(students.begin(), students.end(),  
 [lastName](const Student& student) { return student.lastName == lastName; });  
  
 return (iter != students.end()) ? &(\*iter) : nullptr;  
 }  
  
 void sortStudentsByLastName() {  
 sort(students.begin(), students.end(), [](const Student& a, const Student& b) {  
 return a.lastName < b.lastName;  
 });  
 }  
  
 void displayAllStudents() const {  
 cout << "Last name First name Birthdate Phone number " << endl;  
 for (const auto& student : students) {  
 cout << student.lastName << " " << student.firstName << " "  
 << student.birthDate << " " << student.phoneNumber << endl;  
 }  
 }  
};  
  
int main() {  
 StudentGroup group;  
 int n;  
 do {  
 cout << "Choose action:" << endl  
 << "1. Add student" << endl  
 << "2. Delete student" << endl  
 << "3. Search student by lastname" << endl  
 << "4. Sort students by lastname" << endl  
 << "5. Print all students info" << endl  
 << "0. Exit" << endl;  
 cin >> n;  
 switch (n) {  
 case 1:{  
 string ln, fn, pn, bd;  
 cout << "Input student's data: lastname, firstname, phone number, birthdate" << endl;  
 cin >> ln >> fn >> pn >> bd;  
 group.addStudent(Student(ln, fn, pn, bd));  
 break;  
 }  
 case 2: {  
 string deadman;  
 cout << "Enter student's lastname you want to delete" << endl;  
 cin >> deadman;  
 group.removeStudent(deadman);  
 break;  
 }  
 case 3: {  
 string target;  
 cout << "Enter student's lastname you want to search" << endl;  
 cin >> target;  
 Student \*found = group.findStudentByLastName(target);  
 if (found) {  
 cout << "Student found:" << endl  
 << "Lastname: " << found->lastName << endl  
 << "Firstname: " << found->firstName << endl  
 << "Phone number: " << found->phoneNumber << endl  
 << "Birthdate: " << found->birthDate << endl;  
 } else {  
 cout << "Student wasn't found" << endl;  
 }  
 break;  
 }  
 case 4: {  
 group.sortStudentsByLastName();  
 cout << "Students are sorted" << endl;  
 break;  
 }  
 case 5: {  
 group.displayAllStudents();  
 break;  
 }  
 case 0: {  
 cout << "Shutting down" << endl;  
 break;  
 }  
 default: {  
 cout << "Incorrect input" << endl;  
 }  
 }  
  
 } while(n);  
  
 return 0;  
}  
/\*  
 1  
 Kopzhasharov Azamat 666 28.04.03  
 1  
 Zakirov Mustafa 322 30.09.03  
 1  
 Urmanbetov Sultan 228 06.08.03  
 1  
 test testt 123 12.12.12  
 \*/

/\*  
 2 4 100000 3.14 3.141592 A  
 2 4 100000 3.14 3.141592 z  
 3 4 100000 3.14 3.141592 5  
 3 4 100000 3.14 3.141592 z  
 3 4 12345 2.71 3.141592 !  
 \*/

**Тесты:**

# **Задание №3**

**Постановка задачи:**

**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;

**Исходные данные:**

**Исходный код программы на C++:**

// 2.3  
#include <iostream>  
#include <fstream>  
#include <stack>  
#include <vector>  
using namespace std;  
  
class TrainCar {  
public:  
 int carNumber;  
 TrainCar(int number) : carNumber(number) {}  
};  
  
class TrainSorter {  
private:  
 stack<TrainCar> mainTrack;  
 stack<TrainCar> sideTrackA;  
 stack<TrainCar> sideTrackB;  
  
public:  
 void addToMainTrack(const TrainCar& car) {  
 mainTrack.push(car);  
 }  
  
 void moveToSideTrackA() {  
 if (!mainTrack.empty()) {  
 sideTrackA.push(mainTrack.top());  
 mainTrack.pop();  
 }  
 }  
  
 void moveToSideTrackB() {  
 if (!mainTrack.empty()) {  
 sideTrackB.push(mainTrack.top());  
 mainTrack.pop();  
 }  
 }  
  
 void popFromSideTrackA() {  
 if(!sideTrackA.empty()) sideTrackA.pop();  
 }  
  
 void popFromSideTrackB() {  
 if(!sideTrackB.empty()) sideTrackB.pop();  
 }  
  
 void displayTrainComposition() const {  
 cout << "Main track: ";  
 displayStack(mainTrack);  
 cout << "Side A: ";  
 displayStack(sideTrackA);  
 cout << "Side B: ";  
 displayStack(sideTrackB);  
 }  
  
private:  
 void displayStack(const std::stack<TrainCar>& stack) const {  
 std::stack<TrainCar> tempStack = stack;  
 while (!tempStack.empty()) {  
 std::cout << tempStack.top().carNumber << " ";  
 tempStack.pop();  
 }  
 std::cout << std::endl;  
 }  
};  
  
void readTrainFromFile(TrainSorter& sorter, const string& filename) {  
 ifstream inputFile(filename);  
 if (!inputFile.is\_open()) {  
 cout << "Couldn't open file: " << filename << std::endl;  
 return;  
 }  
 int carNumber;  
 while (inputFile >> carNumber) {  
 sorter.addToMainTrack(TrainCar(carNumber));  
 }  
  
 inputFile.close();  
}  
  
int main() {  
 TrainSorter trainSorter;  
 readTrainFromFile(trainSorter, "trains.txt");  
 int num;  
 do {  
 cout << "What to do?" << endl  
 << "1. Add wagon to main track" << endl  
 << "2. Move wagon to side track A" << endl  
 << "3. Move wagon to side track B" << endl  
 << "4. Delete wagon from side track A" << endl  
 << "5. Delete wagon from side track B" << endl  
 << "6. Print all train sets info" << endl  
 << "0. Exit" << endl;  
 cin >> num;  
 switch (num) {  
 case 1: {  
 int carNumber;  
 std::cout << "Enter wagon: ";  
 std::cin >> carNumber;  
 trainSorter.addToMainTrack(TrainCar(carNumber));  
 break;  
 }  
 case 2:  
 trainSorter.moveToSideTrackA();  
 break;  
 case 3:  
 trainSorter.moveToSideTrackB();  
 break;  
 case 4:  
 trainSorter.popFromSideTrackA();  
 break;  
 case 5:  
 trainSorter.popFromSideTrackB();  
 break;  
 case 6:  
 trainSorter.displayTrainComposition();  
 break;  
  
 case 0:  
 cout << "Shutting down" << endl;  
 break;  
 default:  
 cout << "Incorrect input" << endl;  
 }  
 } while (num);  
 return 0;  
}

**Тесты:**

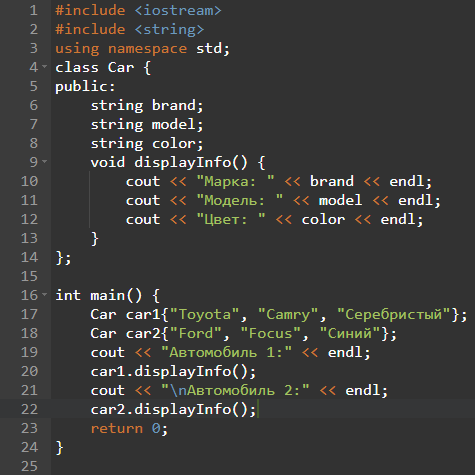
# **Ответы на вопросы:**

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

Класс - это шаблон или описание для создания объектов. Он определяет, какие данные (переменные) и функции (методы) могут быть связаны вместе в одном объекте. Класс описывает структуру и поведение объектов.

Объект - это экземпляр класса, созданный на основе его описания (класса). Он представляет собой конкретный элемент, который содержит данные и может выполнять функции, определенные в классе.

Пример 1: Класс "Автомобиль" и объекты



Пример 2: Класс "Книга" и объекты:



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.*

Существуют **три** 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.

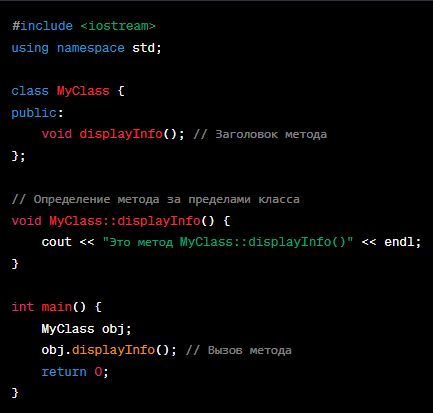
1. *Как называются методы, встроенные в структуру класса?*

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.

Пример:



Здесь displayInfo() имеет заголовок внутри класса MyClass, но его реализация определена за пределами класса в функции MyClass::displayInfo().

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.

Пример доступа к открытому полю и закрытому полю класса в C++:

#include <iostream>

using namespace std;

class MyClass {

public:

int publicField; // Открытое поле

private:

int privateField; // Закрытое поле

public:

// Геттер для закрытого поля

int getPrivateField() {

return privateField;

}

// Сеттер для закрытого поля

void setPrivateField(int value) {

privateField = value;

}

};

int main() {

MyClass obj;

obj.publicField = 42; // Доступ к открытому полю напрямую

obj.setPrivateField(10); // Доступ к закрытому полю через сеттер

cout << "Открытое поле: " << obj.publicField << endl;

cout << "Закрытое поле (через геттер): " << obj.getPrivateField() << endl;

return 0;

}

В этом примере, publicField является открытым полем и к нему можно обращаться напрямую, в то время как privateField - закрытым полем, и для доступа к нему используются геттер и сеттер методы.