**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ им. И. РАЗЗАКОВА**

**Кафедра «Программное обеспечение компьютерных систем»**

**Отчет**

**ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ С++**

**Лабораторная работа №2**

**Выполнила: Табылдиева Шоола ПИ-3-21**

**Проверил: Мусабаев Э.Б.**

**Бишкек 2024**

**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.**

**Код:**

#include <iostream>

#include <vector>

#include <algorithm>

#include <iomanip>

class aeroflot {

public:

int flightnumber;

std::string departurepoint;

std::string destinationpoint;

std::string arrivaltime;

std::string departuretime;

int registrationsection;

aeroflot(int number, const std::string& departure, const std::string& destination,

const std::string& arrival, const std::string& departuretime, int section)

: flightnumber(number),

departurepoint(departure),

destinationpoint(destination),

arrivaltime(arrival),

departuretime(departuretime),

registrationsection(section) {}

void displayflightinfo() const {

std::cout << "| " << std::setw(13) << flightnumber << " | " << std::setw(18) << departurepoint

<< " | " << std::setw(22) << destinationpoint << " | " << std::setw(14) << arrivaltime

<< " | " << std::setw(17) << departuretime << " | " << std::setw(21) << registrationsection << " |" << std::endl;

}

static void displaytableheader() {

std::cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << std::endl;

std::cout << "| flight number | departure point | destination point | arrival time | departure time | registration section |" << std::endl;

std::cout << "----------------------------------------------------------------------------------------------------------------------------" << std::endl;

}

static void displaytablefooter() {

std::cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << std::endl;

}

};

bool comparebydestination(const aeroflot& a, const aeroflot& b) {

return a.destinationpoint < b.destinationpoint;

}

int main() {

std::vector<aeroflot> flights;

int numflights;

std::cout << "enter the number of flights: ";

std::cin >> numflights;

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

int number, section;

std::string departure, destination, arrival, departuretime;

std::cout << std::endl;

std::cout << "enter the flight number: ";

std::cin >> number;

std::cout << "enter the departure point: ";

std::cin >> departure;

std::cout << "enter the destination point: ";

std::cin >> destination;

std::cout << "enter the arrival time: ";

std::cin >> arrival;

std::cout << "enter the departure time: ";

std::cin >> departuretime;

std::cout << "enter the registration section number: ";

std::cin >> section;

flights.emplace\_back(number, departure, destination, arrival, departuretime, section);

}

std::sort(flights.begin(), flights.end(), comparebydestination);

aeroflot::displaytableheader();

for (const auto& flight : flights) {

flight.displayflightinfo();

}

if (flights.empty()) {

std::cout << "no available flights." << std::endl;

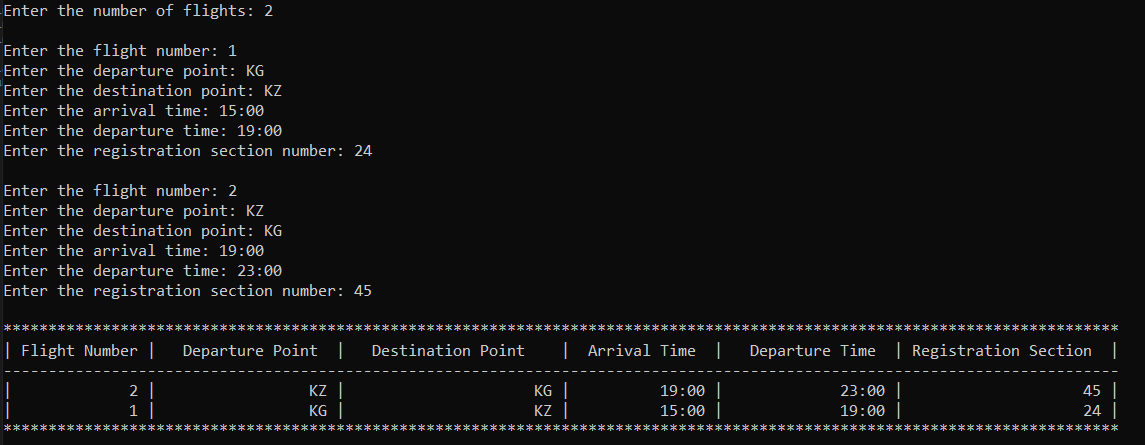
}

aeroflot::displaytablefooter();

return 0;

}

**Результат:**



**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.**

**Код:**

#include <iostream>

#include <vector>

#include <algorithm>

class Student {

public:

std::string firstName;

std::string lastName;

std::string birthDate;

std::string phoneNumber;

Student(const std::string& first, const std::string& last, const std::string& birth, const std::string& phone)

: firstName(first), lastName(last), birthDate(birth), phoneNumber(phone) {}

void displayStudentInfo() const {

std::cout << "Name: " << firstName << " " << lastName << "\tBirthdate: " << birthDate << "\tPhone: " << phoneNumber << std::endl;

}

};

class StudentGroup {

private:

std::vector<Student> students;

public:

void addStudent(const Student& student) {

students.push\_back(student);

}

void removeStudentByLastName(const std::string& lastName) {

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

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

}

void findStudentByLastName(const std::string& lastName) const {

for (const auto& student : students) {

if (student.lastName == lastName) {

std::cout << "Student found: ";

student.displayStudentInfo();

return;

}

}

std::cout << "Student with last name '" << lastName << "' not found." << std::endl;

}

void sortStudentsByLastName() {

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

return a.lastName < b.lastName;

});

}

void displayAllStudents() const {

std::cout << "\nAll students in the group:\n";

for (const auto& student : students) {

student.displayStudentInfo();

}

std::cout << std::endl;

}

};

int main() {

StudentGroup group;

group.addStudent(Student("Shoola", "Tabyldieva", "01-02-2003", "234566543345"));

group.addStudent(Student("Ksenia", "Park", "12-02-2004", "23456765433456"));

group.addStudent(Student("Alex", "Johnson", "14-06-2003", "3456788765"));

int choice;

std::string lastName;

do {

std::cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

std::cout << "\nStudent Group Menu:\n"

<< "1. Display all students\n"

<< "2. Add student\n"

<< "3. Remove student by last name\n"

<< "4. Find student by last name\n"

<< "5. Sort students by last name\n"

<< "6. Exit\n";

std::cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << std::endl;

std::cout << "Enter your choice: ";

std::cin >> choice;

std::cout << std::endl;

switch (choice) {

case 1:

group.displayAllStudents();

break;

case 2:

{

std::string first, last, birth, phone;

std::cout << "Enter student information:\n";

std::cout << "First name: ";

std::cin >> first;

std::cout << "Last name: ";

std::cin >> last;

std::cout << "Birthdate: ";

std::cin >> birth;

std::cout << "Phone number: ";

std::cin >> phone;

group.addStudent(Student(first, last, birth, phone));

std::cout << "Student added successfully.\n";

}

break;

case 3:

std::cout << "Enter the last name of the student to remove: ";

std::cin >> lastName;

group.removeStudentByLastName(lastName);

std::cout << "Student removed successfully.\n";

break;

case 4:

std::cout << "Enter the last name of the student to find: ";

std::cin >> lastName;

group.findStudentByLastName(lastName);

break;

case 5:

group.sortStudentsByLastName();

std::cout << "Students sorted by last name.\n";

break;

case 6:

std::cout << "Exiting the program.\n";

break;

default:

std::cout << "Invalid choice. Please enter a valid option.\n";

break;

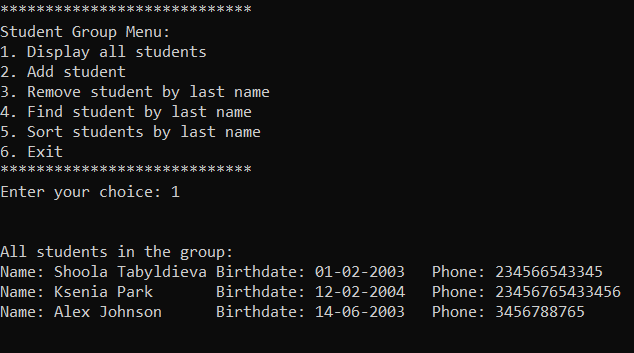
}

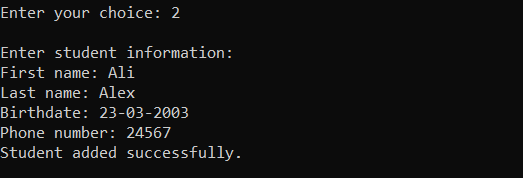
} while (choice != 0);

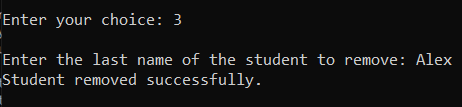
return 0;

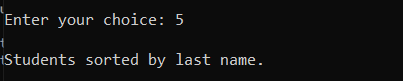
}

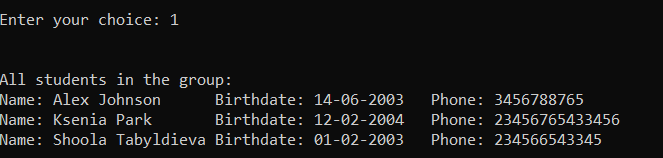
**Результат:**











**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;**

**Код:**

#include <iostream>

#include <fstream>

#include <stack>

class TrainStation {

private:

std::stack<int> leftDirection;

std::stack<int> rightDirection;

public:

void addWagon(int type) {

if (type == 1) {

leftDirection.push(type);

std::cout << "Wagon of type 1 added to the left direction.\n";

}

else if (type == 2) {

rightDirection.push(type);

std::cout << "Wagon of type 2 added to the right direction.\n";

}

else {

std::cout << "Invalid wagon type.\n";

}

}

void sendTrain() {

std::cout << "\nSending trains...\n";

while (!leftDirection.empty()) {

std::cout << "Train going left with wagon type " << leftDirection.top() << "\n";

leftDirection.pop();

}

while (!rightDirection.empty()) {

std::cout << "Train going right with wagon type " << rightDirection.top() << "\n";

rightDirection.pop();

}

}

};

int main() {

TrainStation station;

int wagonType;

int choice;

do {

std::cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

std::cout << "\nTrain Station Menu:\n"

<< "1. Add wagon\n"

<< "2. Send trains\n"

<< "0. Exit\n";

std::cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << std::endl;

std::cout << "Enter your choice: ";

std::cin >> choice;

switch (choice) {

case 1:

std::cout << "Enter wagon type (1 or 2): ";

std::cin >> wagonType;

station.addWagon(wagonType);

break;

case 2:

station.sendTrain();

break;

case 0:

std::cout << "Exiting the program.\n";

break;

default:

std::cout << "Invalid choice. Please enter a valid option.\n";

break;

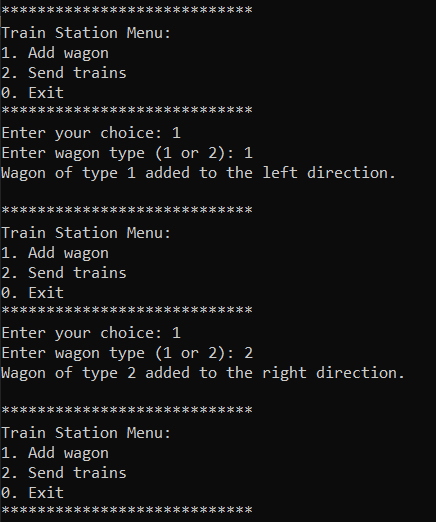
}

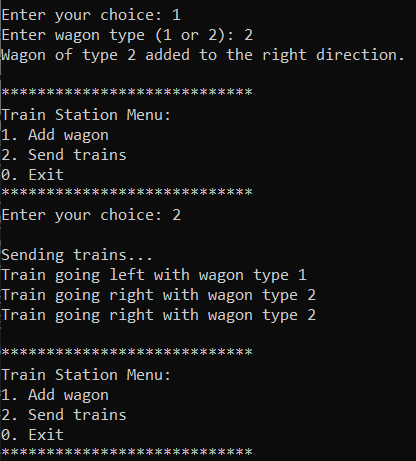
} while (choice != 0);

return 0;

}

**Результат:**





**ВОПРОСЫ К ЗАЩИТЕ ЛАБОРАТОРНОЙ РАБОТЫ №2**

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

**Класс (class)** - это абстрактный тип данных, заданный программистом. Класс описывает структуру своих объектов (поля) и их поведение (методы). Класс представляет модель реального объекта в виде данных и функций для работы с ними. Данные класса называются полями, а функции класса методами.

// Пример класса "Автомобиль"

class Car {

public:

// Поля класса

string brand;

int year;

double price;

// Метод класса для вывода информации

void displayInfo() {

cout << "Brand: " << brand << "\nYear: " << year << "\nPrice: $" << price << endl;

}

};

**Объект (object)** - это экземпляр класса, т.е. его физическая реализация. Практически класс может пониматься как некий шаблон, по которому создаются объекты — экземпляры данного класса. Все экземпляры одного класса созданы по одному шаблону, поэтому имеют один и тот же набор полей и методов. Значения этих полей разные для разных объектов.

Пример:

int main() {

// Создание объекта класса "Автомобиль"

Car myCar;

// Установка значений полей объекта

myCar.brand = "Toyota";

myCar.year = 2022;

myCar.price = 25000.0;

// Вызов метода для вывода информации

myCar.displayInfo();

return 0;

}

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

**Смысл определения класса**: фактически определение класса задает внутреннюю структуру объектов класса, т.е. из каких полей они состоят и какими методами пользуется объект класса при работе с этими полями. При определении класса не происходит выделения физической памяти. Память отводится при объявлении (создании) объекта.Общий вид синтаксиса класса**:**

*// имя должно быть уникальным среди имен других классов, определенных в вашей программе.*

**сlass <**имя\_ класса**>**

**{**

**private:**

<сокрытые поля и методы класса>

**public:**

<общедоступные поля и методыкласса**>**

**} [**список объектов]**;**

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.

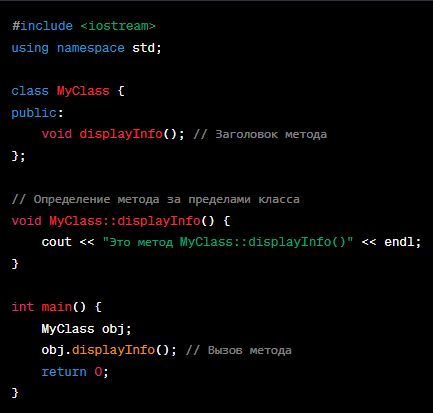
1. **Как называются методы, встроенные в структуру класса?**

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.