**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. И. РАЗЗАКОВА**

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

**ОТЧЕТ**

**по лабораторной работе №2**

**Выполнила:** студент группы ПИ-3-21

Джанышбекова Акмарал

**Проверила:** Мусабаев Э. Б.

**Бишкек 2024**

**Задание №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; 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 программы:**

#include <string>

#include <iostream>

#include <iomanip>

#include <Windows.h> время

using namespace std;

class AEROFLOT {

string departureLocation;

string destinationLocation;

string departureTime;

string arrivalTime;

string registrationLocation;

int flightNumber;

public:

void SetFlightInfo()

{

cout << "Введите название пункта отправления:" << endl;

cin >> departureLocation;

cout << "Введите название пункта назначения:" << endl;

cin >> destinationLocation;

cout << "Введите время отправки:" << endl;

cin >> departureTime;

cout << "Введите время прибытия:" << endl;

cin >> arrivalTime;

cout << "Введите место регистрации:" << endl;

cin >> registrationLocation;

cout << "Введите номер рейса:" << endl;

cin >> flightNumber;

cin.ignore();

}

void DisplayFlightInfo() {

cout << setw(15) << departureLocation << setw(15) << destinationLocation << setw(12) << departureTime << setw(12) << arrivalTime << setw(10) << flightNumber << setw(15) << registrationLocation << endl;

cout << "------------------------------------------------------------------------------------------------------" << endl;

}

string GetDestinationLocation() {

return destinationLocation;

}

int GetFlightNumber() {

return flightNumber;

}

};

void BubbleSort(AEROFLOT flights[], size\_t size) {

AEROFLOT temp;

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

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

if (flights[j].GetDestinationLocation() > flights[j + 1].GetDestinationLocation()) {

temp = flights[j];

flights[j] = flights[j + 1];

flights[j + 1] = temp;

}

}

}

cout << "Сортировка выполнена успешно!" << endl;

}

void Search(AEROFLOT flights[], size\_t size) {

bool found = false;

string destination;

cout << "Введите название пункта назначения:" << endl;

cin >> destination;

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

if (destination == flights[i].GetDestinationLocation()) {

flights[i].DisplayFlightInfo();

found = true;

}

}

if (!found)

cout << "Рейс не найден!" << endl;

}

int main()

{ setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int flightCount;

cout << "Введите количество рейсов:";

cin >> flightCount;

AEROFLOT flightInfo[10];

int choice;

do

{ cout << "Выберите действие:" << endl;

cout << "1-Ввод" << endl;

cout << "2-Вывод" << endl;

cout << "3-Сортировка" << endl;

cout << "4-Поиск рейса" << endl;

cout << "0-Выход" << endl;

cin >> choice;

switch (choice) {

case 0:

return 0;

break;

case 1:

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

flightInfo[i].SetFlightInfo();

}

cout << endl;

break;

case 2:

cout << setw(15) << "| Пункт отпр. " << setw(15) << "| Пункт назнач. " << setw(12) << "| Время отпр. " << setw(12) << "| Время приб. " << setw(10) << "|Номер рейса " << setw(10) << "| Место регистрации | " << endl;

cout << " |=============|===============|=============|=============|============|===================|" << endl;

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

{flightInfo[i].DisplayFlightInfo();}

cout << endl;

break;

}

case 3:

BubbleSort(flightInfo, flightCount);

cout << endl;

break;

case 4:

Search(flightInfo, flightCount);

cout << endl;

break;

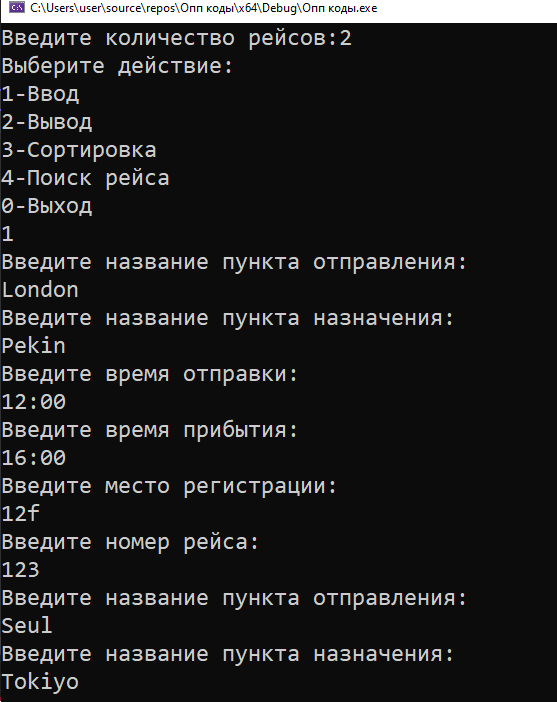
}

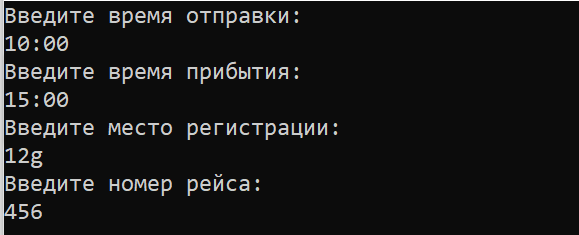
} while (choice != 0);

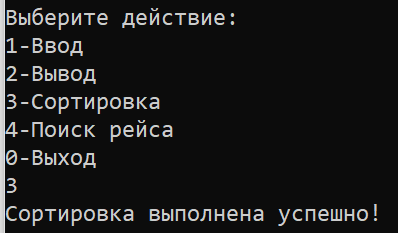
return 0;

}

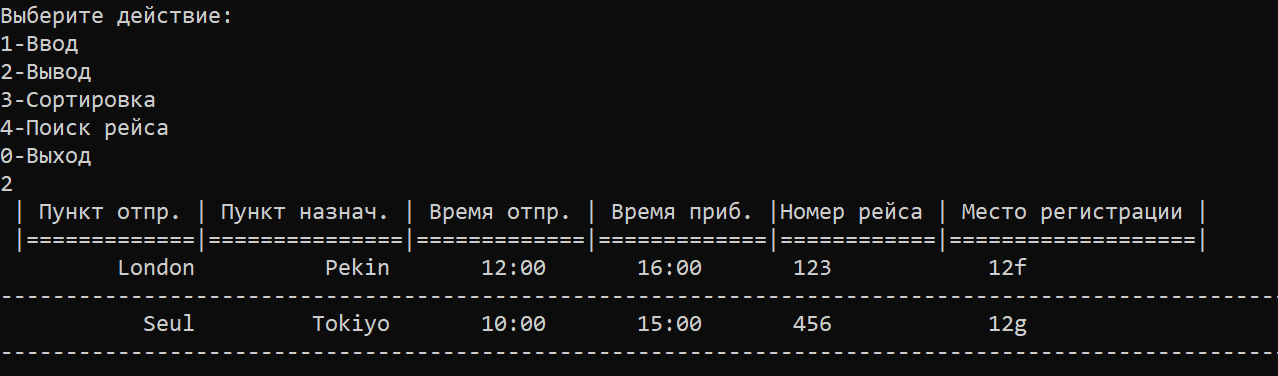
**Результат работы программы:**



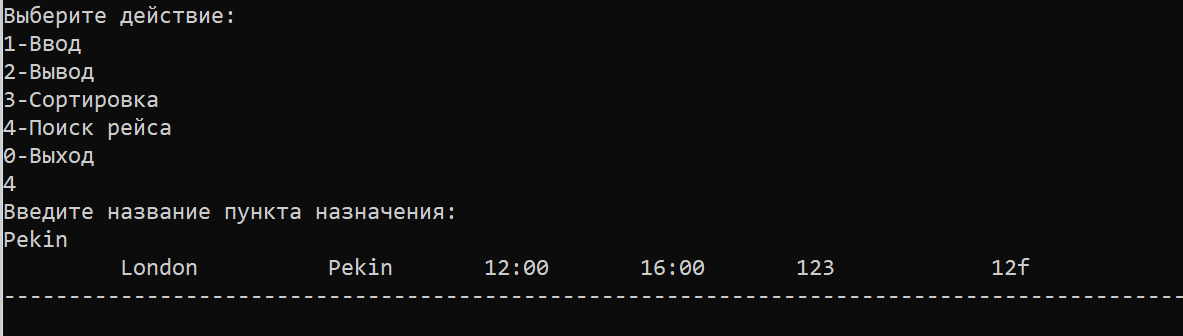




Вывод после сортировки:



Поиск рейса:



**Задание №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 программы:**

#include <iostream>

#include <vector>

#include <algorithm>

#include <string>

#include <iomanip>

#include <Windows.h>

using namespace std;

class Student {

public:

string firstName;

string lastName;

string dateOfBirth;

string phoneNumber;

};

class StudentGroup {

private:

vector<Student> students;

public:

// Метод для добавления студента

void addStudent(const Student& student) {

students.push\_back(student);

}

// Метод для поиска студента по фамилии

Student\* findStudentByLastName(const string& lastName) {

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

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

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

}

// Метод для сортировки студентов по фамилии

void sortByLastName() {

sort(students.begin(), students.end(),

[](const Student& a, const Student& b) { return a.lastName < b.lastName; });

}

// Метод для сортировки студентов по имени

void sortByFirstName() {

sort(students.begin(), students.end(),

[](const Student& a, const Student& b) { return a.firstName < b.firstName; });

}

// Метод для сортировки студентов по дате рождения

void sortByDateOfBirth() {

sort(students.begin(), students.end(),

[](const Student& a, const Student& b) { return a.dateOfBirth < b.dateOfBirth; });

}

// Метод для вывода информации о студентах

void displayStudents() {

for (const auto& student : students) {

cout << "Студент: " << student.firstName << " " << student.lastName

<< ", Дата рождения: " << student.dateOfBirth

<< ", Номер телефона: " << student.phoneNumber << "\n";

}

}

// Метод для удаления студента по имени и фамилии

void removeStudentByName(const string& firstName, const string& lastName) {

students.erase(

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

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

students.end());

}

};

int main() {

setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

StudentGroup group;

int choice = -1;

int numStudents = 0; // Переменная для хранения количества студентов

while (choice != 0) {

cout << "Меню:\n";

cout << "1-Ввод количества студентов\n"; // Добавлен пункт ввода количества студентов

cout << "2-Ввод данных студентов\n"; // Изменен пункт на ввод данных студентов

cout << "3-Вывод всех студентов\n"; // Изменен пункт на вывод всех студентов

cout << "4-Сортировка по фамилии\n"; // Добавлен пункт сортировки по фамилии

cout << "5-Сортировка по имени\n"; // Добавлен пункт сортировки по имени

cout << "6-Сортировка по дате рождения\n"; // Добавлен пункт сортировки по дате рождения

cout << "7-Поиск студента\n";

cout << "8-Удаление студента\n"; // Изменен пункт на удаление студента

cout << "0-Выход\n";

cout << "Выберите действие: ";

cin >> choice;

switch (choice) {

case 1: {

// Ввод количества студентов

cout << "Введите количество студентов: ";

cin >> numStudents;

break;

}

case 2: {

// Ввод данных студентов

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

Student student;

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

cout << "Имя: ";

cin >> student.firstName;

cout << "Фамилия: ";

cin >> student.lastName;

cout << "Дата рождения (ГГГГ-ММ-ДД): ";

cin >> student.dateOfBirth;

cout << "Номер телефона: ";

cin >> student.phoneNumber;

group.addStudent(student);

}

break;

}

case 3:

// Вывод всех студентов

cout << "\nВсе студенты:\n";

group.displayStudents();

break;

case 4:

// Сортировка по фамилии

group.sortByLastName();

cout << "\nПосле сортировки по фамилии:\n";

group.displayStudents();

break;

case 5:

// Сортировка по имени

group.sortByFirstName();

cout << "\nПосле сортировки по имени:\n";

group.displayStudents();

break;

case 6:

// Сортировка по дате рождения

group.sortByDateOfBirth();

cout << "\nПосле сортировки по дате рождения:\n";

group.displayStudents();

break;

case 7: {

// Поиск студента по фамилии

string searchLastName;

cout << "\nВведите фамилию для поиска: ";

cin.ignore();

getline(cin, searchLastName);

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

if (foundStudent != nullptr) {

cout << "\nНайденный студент:\n";

cout << "Студент: " << foundStudent->firstName << " " << foundStudent->lastName

<< ", Дата рождения: " << foundStudent->dateOfBirth

<< ", Номер телефона: " << foundStudent->phoneNumber << "\n";

}

else {

cout << "\nСтудент с фамилией '" << searchLastName << "' не найден.\n";

}

break;

}

case 8: {

// Удаление студента по имени и фамилии

string removeFirstName, removeLastName;

cout << "\nВведите имя и фамилию для удаления (через пробел): ";

cin >> removeFirstName >> removeLastName;

group.removeStudentByName(removeFirstName, removeLastName);

cout << "\nПосле удаления студента '" << removeFirstName << " " << removeLastName << "':\n";

group.displayStudents();

break;

}

case 0:

// Выход из программы

cout << "Выход из программы.\n";

break;

default:

// Некорректный выбор

cout << "Некорректный выбор. Попробуйте снова.\n";

break;

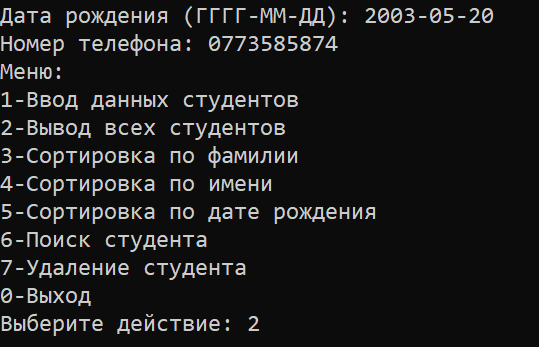
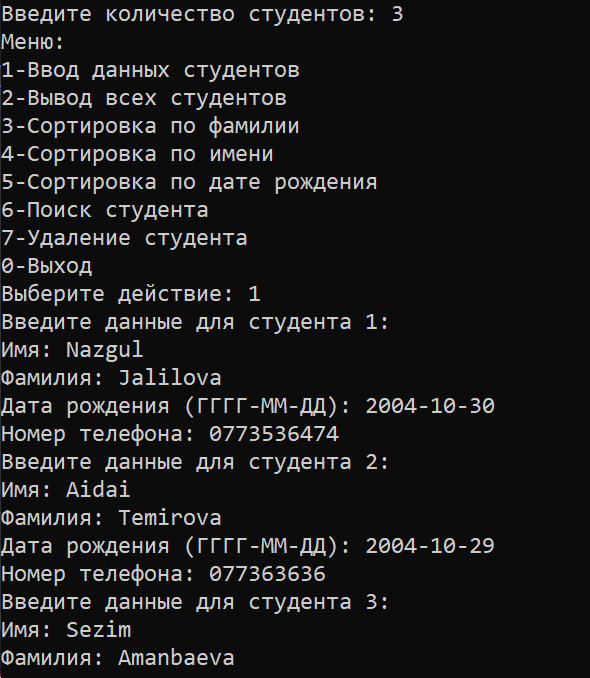
}

}

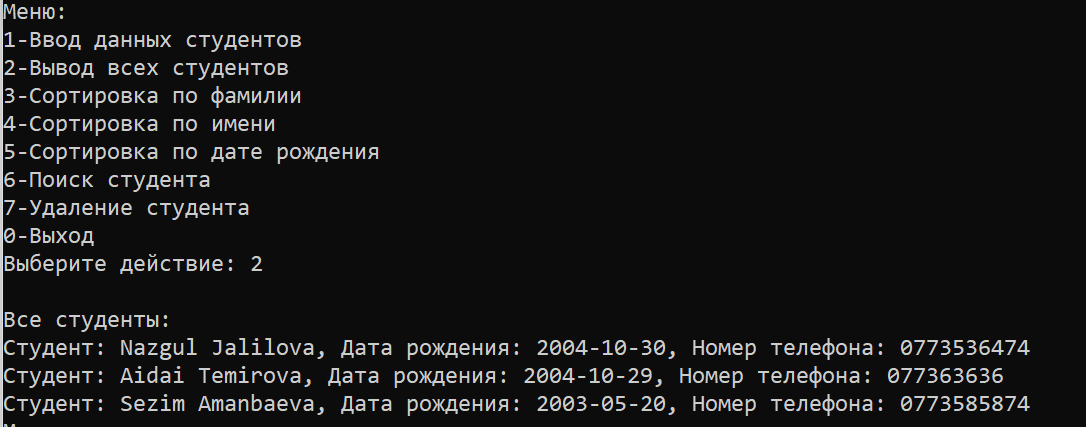
return 0;

}

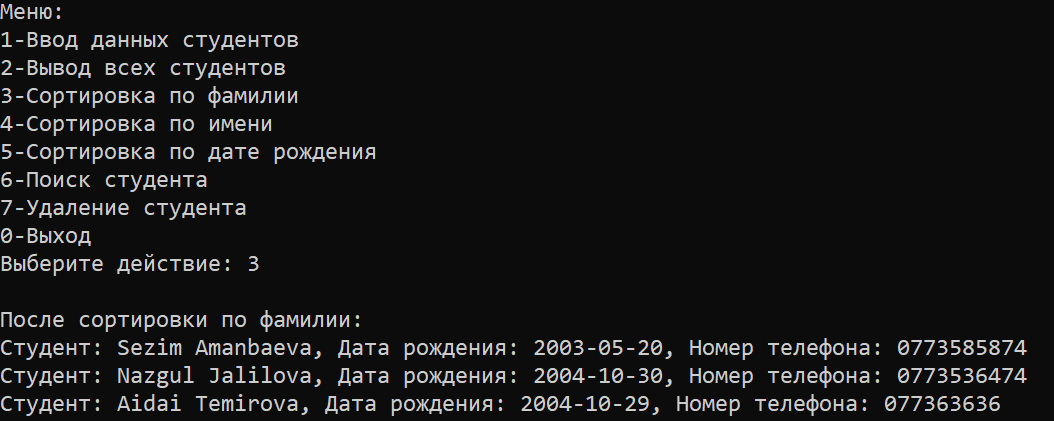
**Результат работы программы:**



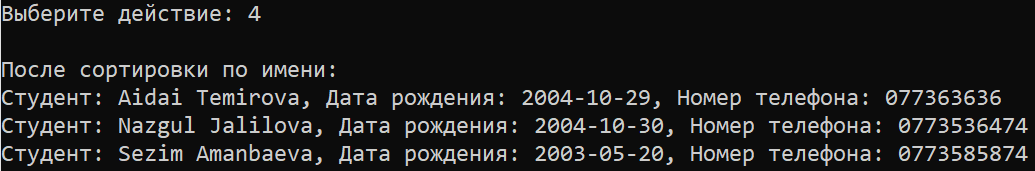
Вывод:



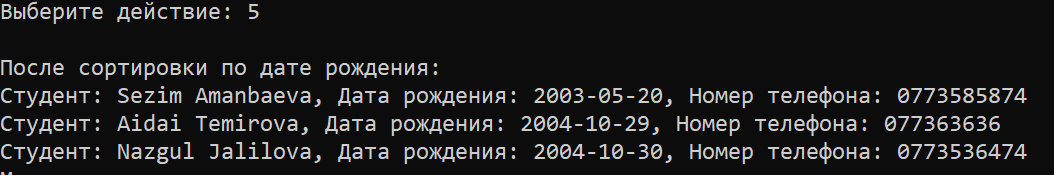
Сортировка по фамилии:



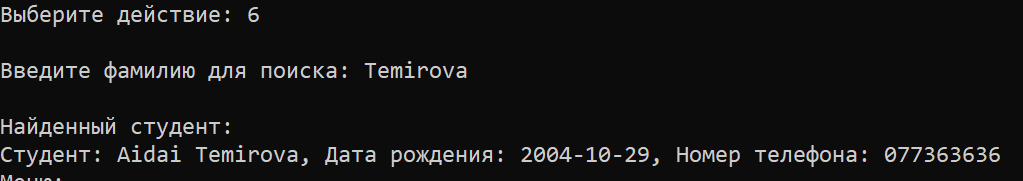
Сортировка по имени:



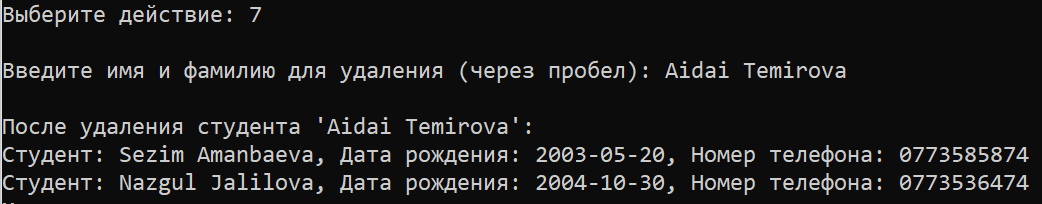
Сортировка по дате рождения:



Поиск по фамилии:



Удаление студента:



**Задание №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 программы:**

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <fstream>

#include <stack>

using namespace std;

class STACK {

private:

stack<int> myStack1; // Основной стек, в который будут вводиться данные

stack<int> myStack3; // Стек для четных чисел

stack<int> myStack4; // Стек для нечетных чисел

public:

void input(int a) {

myStack1.push(a); // Добавить введенное значение в основной стек

}

void raz() {

if (myStack1.empty()) {

cout << "нет данных" << endl;

}

else {

while (!myStack1.empty()) {

if (myStack1.top() % 2 == 0) {

myStack3.push(myStack1.top()); // Поместить четное число в стек myStack3

}

else {

myStack4.push(myStack1.top()); // Поместить нечетное число в стек myStack4

}

myStack1.pop(); // Удалить верхний элемент из основного стека

}

cout << "деление завершено" << endl;

}

}

void file\_Show() {

ifstream fin;

fin.open("akmaral.txt"); // Открыть файл для чтения

if (!fin.is\_open()) {

cout << "Файл не открылся" << endl;

}

else {

int a;

while (fin >> a) {

myStack1.push(a); // Считать данные из файла и добавить в основной стек

}

}

fin.close(); // Закрыть файл

cout << "данные считаны из файла" << endl;

}

void show() {

stack<int> myStack2 = myStack1; // Создать копию основного стека для вывода

if (!myStack2.empty()) {

while (!myStack2.empty()) {

cout << myStack2.top() << endl; // Вывести верхний элемент

myStack2.pop(); // Удалить верхний элемент из копии стека

}

}

else {

cout << "стек пуст!" << endl;

}

}

void chetn() {

stack<int> myStack2 = myStack3; // Создать копию стека четных чисел

if (!myStack2.empty()) {

while (!myStack2.empty()) {

cout << myStack2.top() << endl; // Вывести верхний элемент

myStack2.pop(); // Удалить верхний элемент из копии стека

}

}

else {

cout << "нет таких вагонов!" << endl;

}

}

void ne\_chetn() {

stack<int> myStack2 = myStack4; // Создать копию стека нечетных чисел

if (!myStack2.empty()) {

while (!myStack2.empty()) {

cout << myStack2.top() << endl; // Вывести верхний элемент

myStack2.pop(); // Удалить верхний элемент из копии стека

}

}

else {

cout << "нет таких вагонов!" << endl;

}

}

};

int main() {

setlocale(LC\_ALL, "ru");

srand(static\_cast<unsigned>(time(NULL))); // Инициализация генератора случайных чисел

int col = 0, val = 0;

STACK obj; // Создание объекта класса STACK

cout << "Введите количество вагонов: " << endl;

cin >> col; // Ввод количества вагонов

ofstream fout("akmaral.txt"); // Открыть файл для записи

if (!fout.is\_open()) {

cout << "Файл не открылся!" << endl;

}

else {

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

fout << rand() % 21 << endl; // Случайно генерировать числа и записывать их в файл

}

fout.close(); // Закрыть файл

}

while (val != 6) {

system("CLS"); // Очистить консоль

cout << "\n Меню: " << endl;

cout << "0 - Ввод данных " << "\n 1 - Считывание из файла " << "\n 2 - Вывод всех вагонов " << "\n 3- Разделить вагоны " << "\n 4 - Вывод четных вагонов" << "\n 5 - Вывод нечетных вагонов" << "\n 6 - Выход";

cout << endl << "Выберите из меню: ";

cin >> val; // Выбор действия из меню

if (val == 0) {

system("CLS"); // Очистить консоль

cout << "Введите номера вагонов : " << endl;

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

int zn;

cin >> zn; // Ввод номеров вагонов

obj.input(zn); // Добавление номера в основной стек

}

cout << endl << "данные записаны!" << endl << endl;

system("pause"); // Пауза для продолжения

}

else if (val == 1) {

system("CLS"); // Очистить консоль

obj.file\_Show(); // Считать данные из файла

system("pause"); // Пауза для продолжения

}

else if (val == 2) {

system("CLS"); // Очистить консоль

cout << "Все вагоны: " << endl << endl;

obj.show(); // Вывод всех вагонов

system("pause"); // Пауза для продолжения

}

else if (val == 3) {

system("CLS"); // Очистить консоль

obj.raz(); // Разделить вагоны

system("pause"); // Пауза для продолжения

}

else if (val == 4) {

system("CLS"); // Очистить консоль

cout << "Вагоны с четными номерами:" << endl << endl;

obj.chetn(); // Вывод четных вагонов

system("pause"); // Пауза для продолжения

}

else if (val == 5) {

system("CLS"); // Очистить консоль

cout << "Вагоны с нечетными номерами:" << endl << endl;

obj.ne\_chetn(); // Вывод нечетных вагонов

system("pause"); // Пауза для продолжения

}

else if (val == 6) {

cout << "До свидания!" << endl;

break; // Выход из цикла при выборе опции "Выход"

}

else {

cout << "Неверный выбор. Попробуйте снова." << endl;

system("pause");

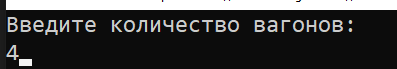
return 0;

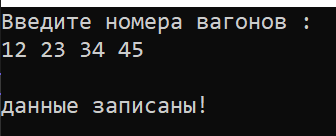
}

}

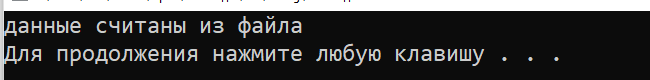
}

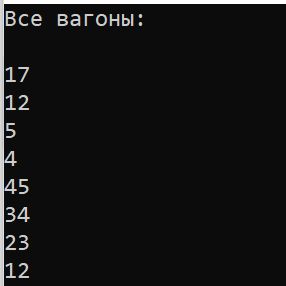
**Результат работы программы:**



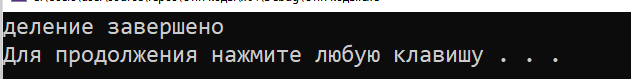


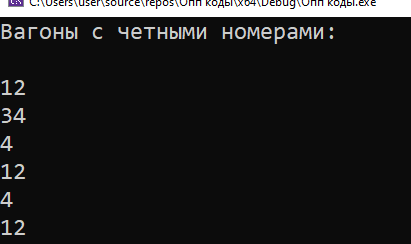
Считывание из файла:

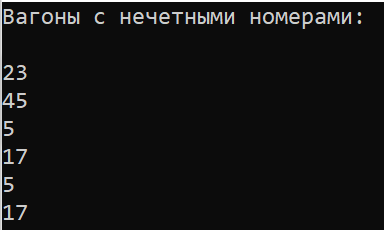




Разделение вагонов:







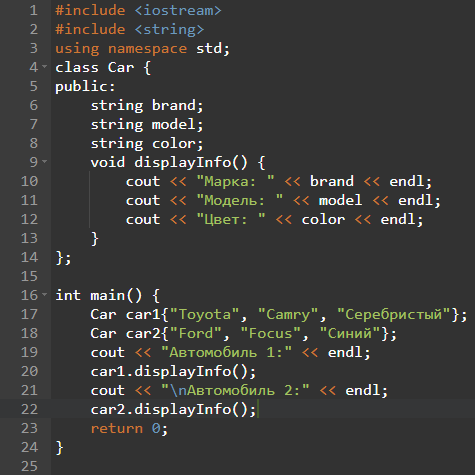
**ВОПРОСЫ К ЗАЩИТЕ ЛАБОРАТОРНОЙ РАБОТЫ №2**

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.

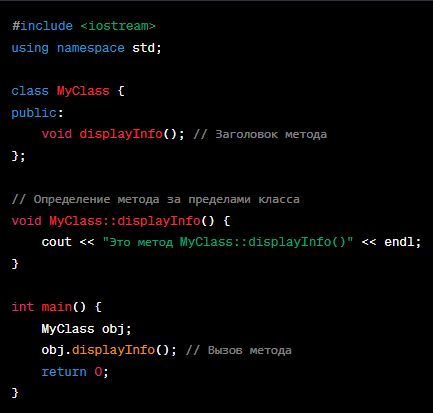
1. *Как называются методы, встроенные в структуру класса?*

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.