Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет

им. И.Раззакова

Институт информационных технологий

Кафедра «Программное обеспечение компьютерных систем»

Направление: 710400 «Программная инженерия»

Дисциплина: «Объектно-ориентированное программирование с++»

**ОТЧЕТ**

По лабораторной работе № 2

Тема: «Классы и объекты»

Выполнила: студент группы

ПИ(б)-2-21 Кадырбек кызы Айзат

Проверила: Мусабев Э.Б.

Бишкек – 2024

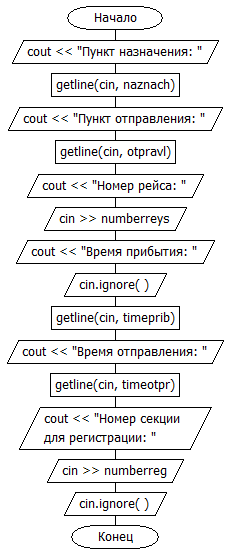
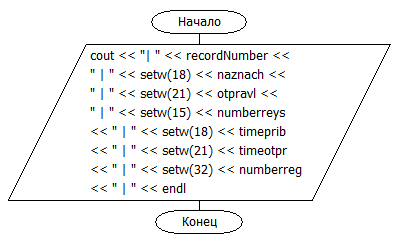
**Задания**

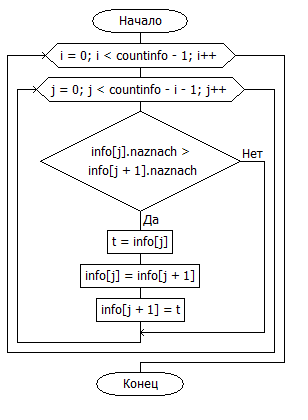
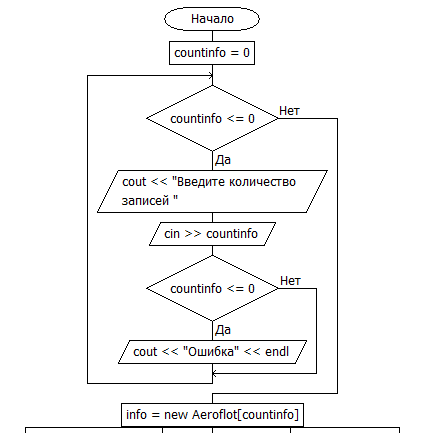
1. В здании аэропорта на экранах отображается информация о самолетах, а именно: информация о пункте отправления, пункте назначения, номере рейса, времени прибытия, времени отправления, номере секции для регистрации. Экраны – это средство, которое помогает своевременно зарегистрировать и отправить пассажиров. Важно, чтобы информация на экранах была понятной и верной.

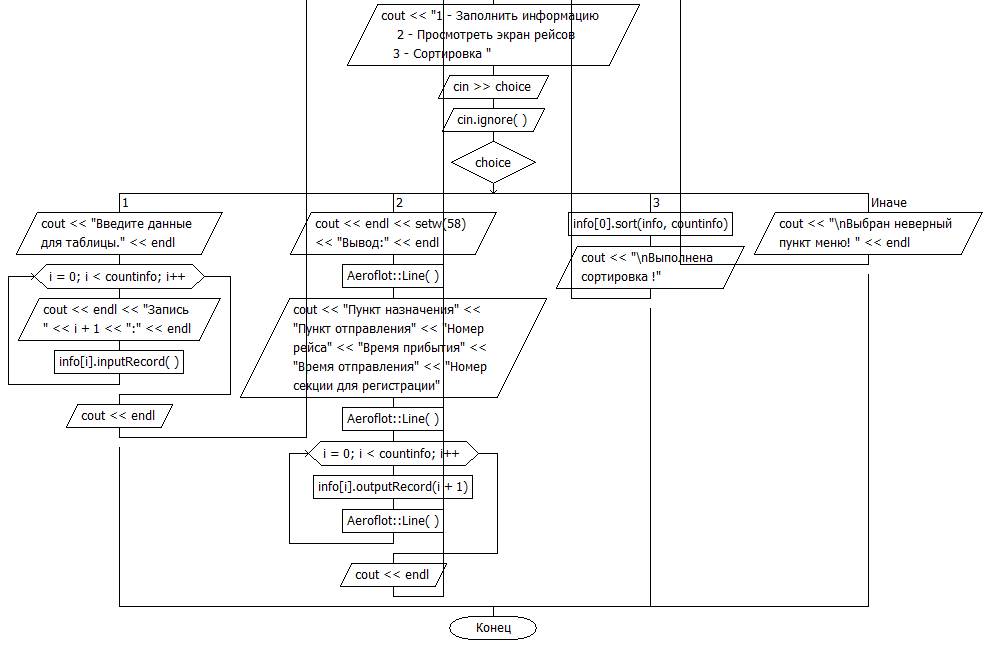
Создайте необходимую информацию в виде таблицы для такого экрана, с помощью класса Aeroflot, содержащего в описании следующие поля: номер рейса; название пункта отправления; название пункта назначения рейса; время прибытия; время отправления; место регистрации.

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.

***Блок-схема:***



***Код:***

#include <iostream>

#include <iomanip>

#include <Windows.h>

#include <string>

using namespace std;

class Aeroflot {

private:

string naznach;

string otpravl;

short numberreys;

string timeprib;

string timeotpr;

short numberreg;

public:

void inputRecord() {

cout << "Пункт назначения: ";

getline(cin, naznach);

cout << "Пункт отправления: ";

getline(cin, otpravl);

cout << "Номер рейса: ";

cin >> numberreys;

cout << "Время прибытия: ";

cin.ignore();

getline(cin, timeprib);

cout << "Время отправления: ";

getline(cin, timeotpr);

cout << "Номер секции для регистрации: ";

cin >> numberreg;

cin.ignore();

}

void outputRecord(int recordNumber) {

cout << "| " << recordNumber << " | " << setw(18) << naznach << " | " << setw(21) << otpravl << " | " << setw(15) << numberreys

<< " | " << setw(18) << timeprib << " | " << setw(21) << timeotpr << " | " << setw(32) << numberreg << " | " << endl;

}

static void Line() {

for (int i = 0; i < 148; i++) {

cout << "-";

}

cout << endl;

}

static void sort(Aeroflot info[], int countinfo) {

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

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

if (info[j].naznach > info[j + 1].naznach) {

Aeroflot t = info[j];

info[j] = info[j + 1];

info[j + 1] = t;

}

}

}

}

};

int main() {

setlocale(LC\_ALL, "");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

short countinfo = 0;

while (countinfo <= 0) {

cout << "Введите количество записей (рейсов) = ";

cin >> countinfo;

if (countinfo <= 0) {

cout << "Ошибка! Повторите ввод " << endl;

}

}

Aeroflot\* info = new Aeroflot[countinfo];

short choice;

menu:

cout << "\n1 - Заполнить информацию о рейсах\n2 - Просмотреть экран рейсов\n3 - Сортировка в алфавитном порядке по месту назначения\n";

cin >> choice;

cin.ignore();

switch (choice) {

case 1:

cout << "Введите данные для таблицы." << endl;

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

cout << endl << "Запись " << i + 1 << ":" << endl;

info[i].inputRecord();

}

cout << endl;

goto menu;

case 2:

cout << endl << setw(58) << "Вывод:" << endl;

Aeroflot::Line();

cout << "| " << "№" << " | " << setw(18) << "Пункт назначения" << " | " << setw(21) << "Пункт отправления" << " | " << setw(15) << "Номер рейса"

<< " | " << setw(18) << "Время прибытия" << " | " << setw(21) << "Время отправления" << " | " << setw(32) << "Номер секции для регистрации" << " | " << endl;

Aeroflot::Line();

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

info[i].outputRecord(i + 1);

Aeroflot::Line();

}

cout << endl;

goto menu;

case 3:

info[0].sort(info, countinfo);

cout << "\nВыполнена сортировка в алфавитном порядке по пункту назначения!\n\n";

goto menu;

default:

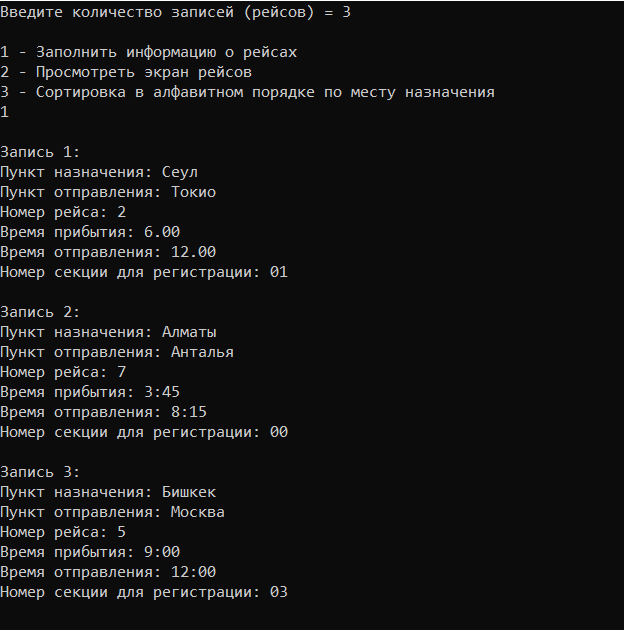
cout << "\nВыбран неверный пункт меню! Пожалуйста, повторите вашу попытку.\n\n" << endl;

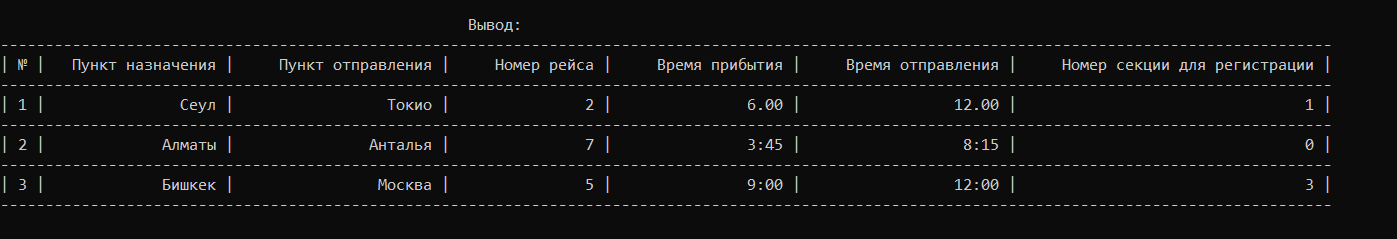
goto menu;

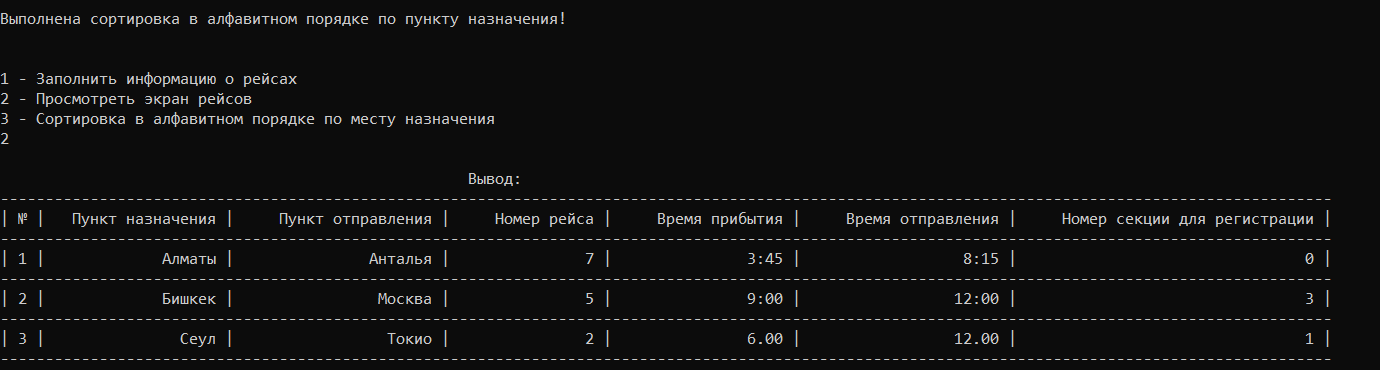
}

}

***Результат:***







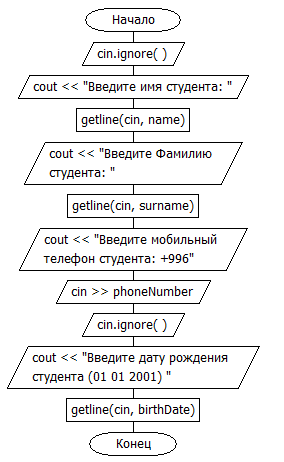
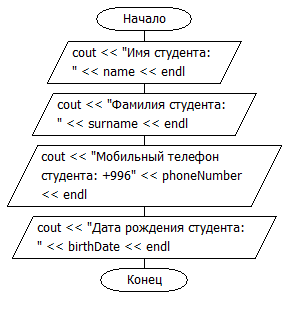
**2. Опишите класс «студенческая группа».**

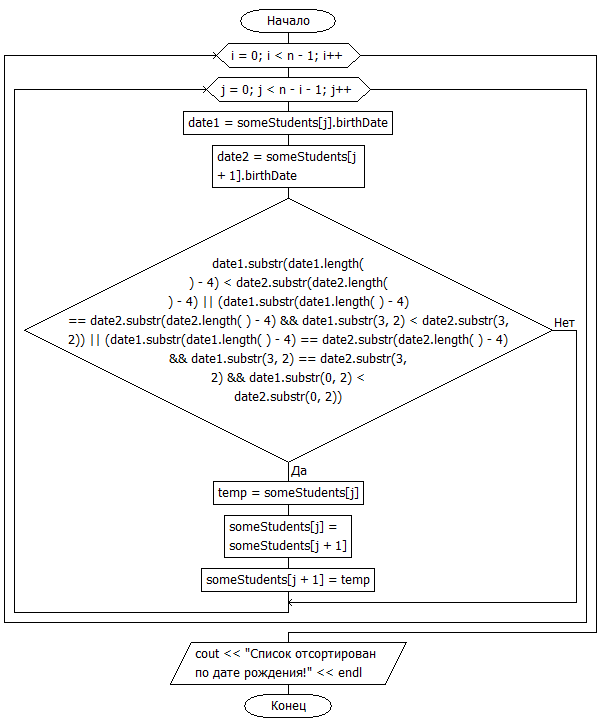
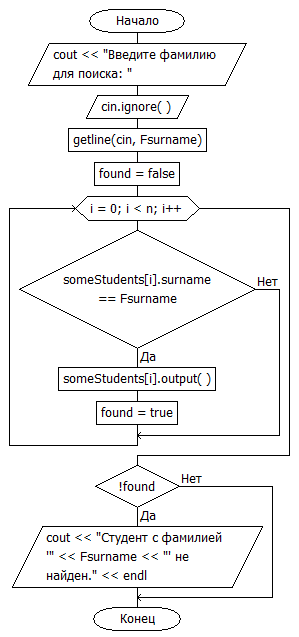
Предусмотрите возможность: работы с переменным числом студентов; поиска студента по какому-либо признаку (например, по фамилии, дате рождения или номеру телефона); добавления или удаления записей; сортировки по разным полям.

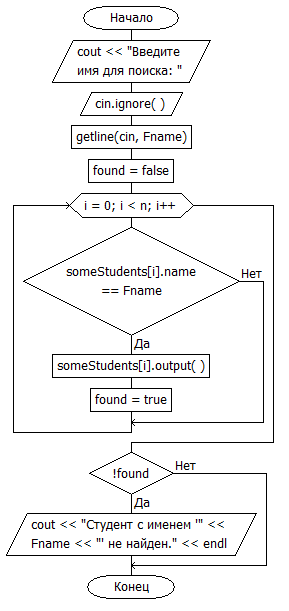
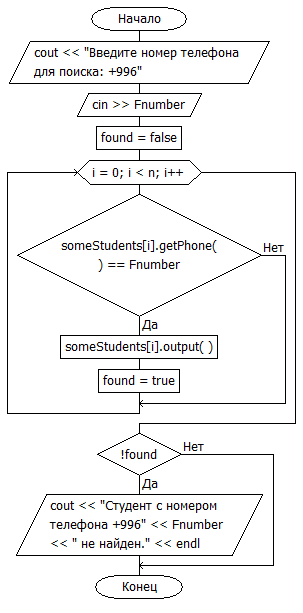
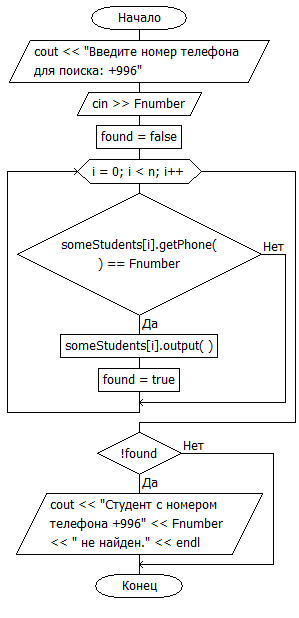
Разработайте программу, демонстрирующую работу с этим классом.

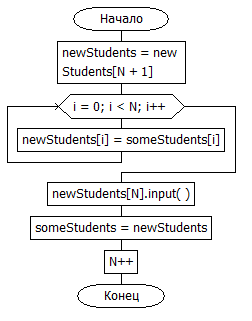
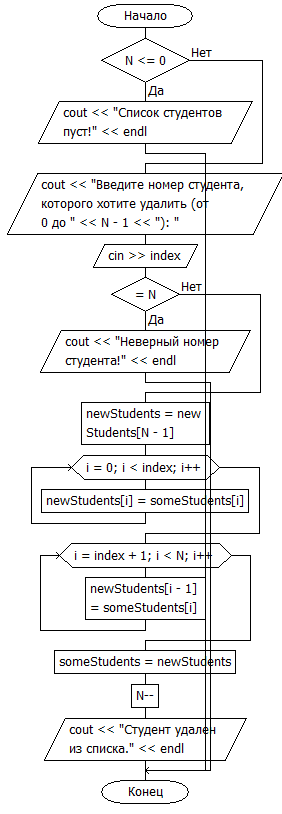
Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

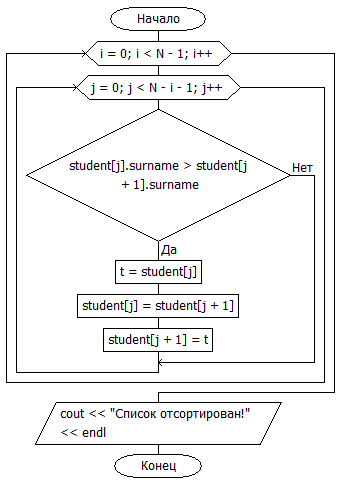
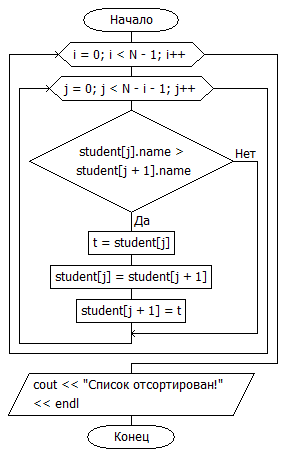
***Блок-схема:***

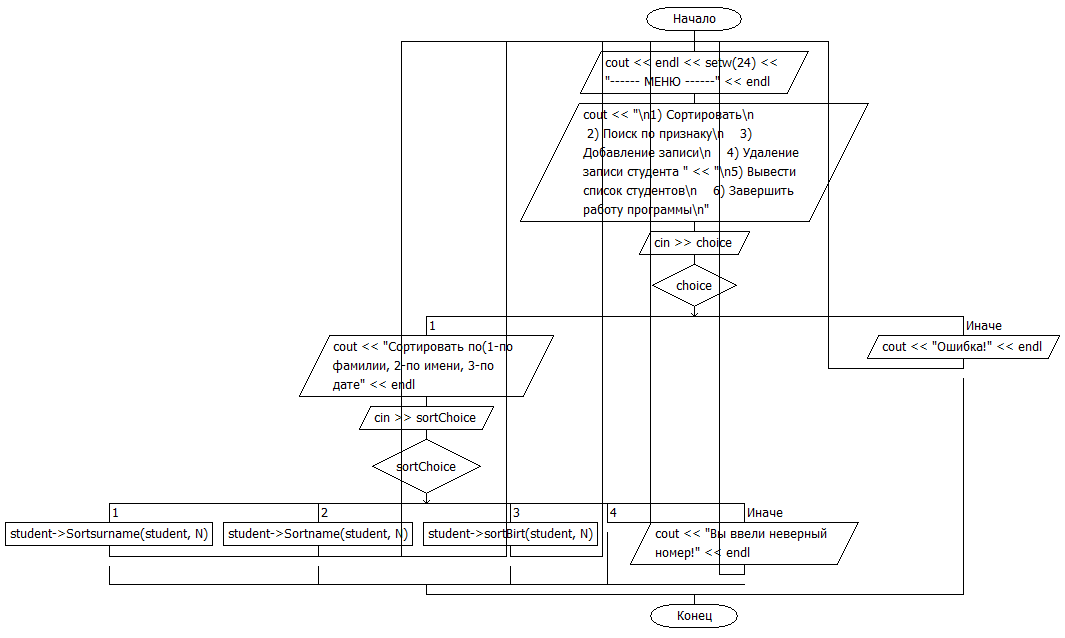
 

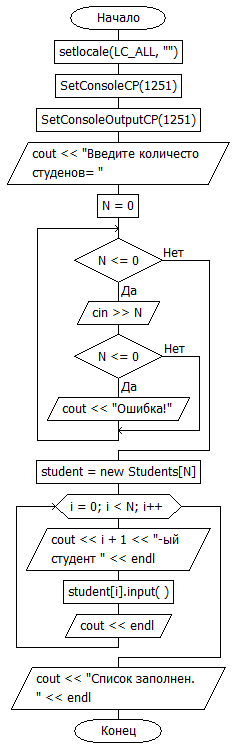
 

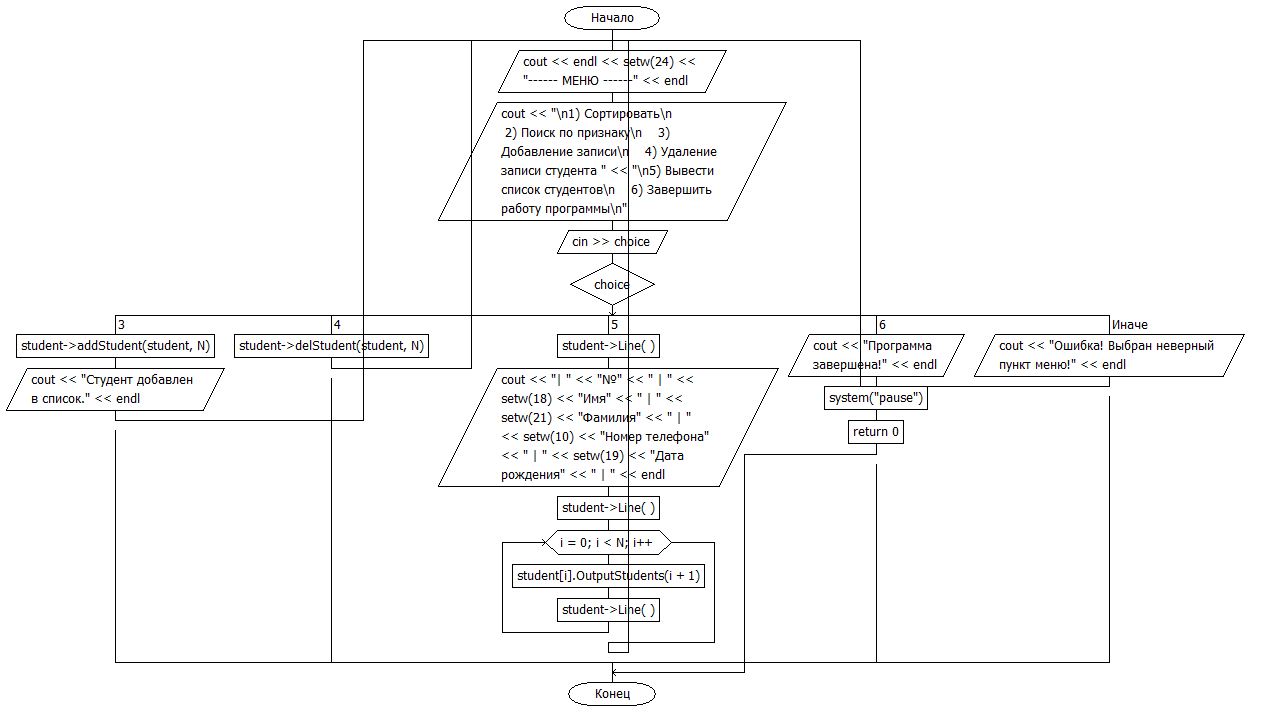
  







***Код:***

#include <iostream>

#include <iomanip>

#include <Windows.h>

#include <string>

using namespace std;

class Students {

private: // Характеристики студента

int phoneNumber;

string name;

string surname;

string birthDate;

public:

// Ввод данных

void input() {

cin.ignore(); // Добавляем cin.ignore() для сброса буфера

cout << "Введите имя студента: ";

getline(cin, name);

cout << "Введите Фамилию студента: ";

getline(cin, surname);

cout << "Введите мобильный телефон студента: +996";

cin >> phoneNumber;

cin.ignore(); // Сбросим символ новой строки после ввода номера телефона

cout << "Введите дату рождения студента (01 01 2001) ";

getline(cin, birthDate);

}

// Вывод данных

void output() {

cout << "Имя студента: " << name << endl;

cout << "Фамилия студента: " << surname << endl;

cout << "Мобильный телефон студента: +996" << phoneNumber << endl;

cout << "Дата рождения студента: " << birthDate << endl;

}

// Сортировка списка по дате рождения

void sortBirt(Students\*& someStudents, short n) {

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

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

// Разделяем дату на составляющие (год, месяц, день)

string date1 = someStudents[j].birthDate;

string date2 = someStudents[j + 1].birthDate;

if (date1.substr(date1.length() - 4) < date2.substr(date2.length() - 4) ||

(date1.substr(date1.length() - 4) == date2.substr(date2.length() - 4) &&

date1.substr(3, 2) < date2.substr(3, 2)) ||

(date1.substr(date1.length() - 4) == date2.substr(date2.length() - 4) &&

date1.substr(3, 2) == date2.substr(3, 2) && date1.substr(0, 2) < date2.substr(0, 2))) {

Students temp = someStudents[j];

someStudents[j] = someStudents[j + 1];

someStudents[j + 1] = temp;

}

}

}

cout << "Список отсортирован по дате рождения!" << endl;

}

// 1 - Поиск по фамилии

void findSurname(Students\* someStudents, short n) {

string Fsurname;

cout << "Введите фамилию для поиска: ";

cin.ignore();

getline(cin, Fsurname);

bool found = false;

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

if (someStudents[i].surname == Fsurname) {

someStudents[i].output();

found = true;

}

}

if (!found) {

cout << "Студент с фамилией '" << Fsurname << "' не найден." << endl;

}

}

// 2 - Поиск по имени

void findName(Students\* someStudents, short n) {

string Fname;

cout << "Введите имя для поиска: ";

cin.ignore();

getline(cin, Fname);

bool found = false;

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

if (someStudents[i].name == Fname) {

someStudents[i].output();

found = true;

}

}

if (!found) {

cout << "Студент с именем '" << Fname << "' не найден." << endl;

}

}

// 3 - Поиск по номеру телефона

void findPhone(Students\* someStudents, short n) {

int Fnumber;

cout << "Введите номер телефона для поиска: +996";

cin >> Fnumber;

bool found = false;

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

if (someStudents[i].getPhone() == Fnumber) {

someStudents[i].output();

found = true;

}

}

if (!found) {

cout << "Студент с номером телефона +996" << Fnumber << " не найден." << endl;

}

}

// 3 - Поиск по дате рождения

void findBirthDate(Students\* someStudents, short n) {

string Fdate;

cout << "Введите дату рождения для поиска (формат: -01 01 2001-): ";

cin.ignore();

getline(cin, Fdate);

bool found = false;

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

if (someStudents[i].getBirthDate() == Fdate) {

someStudents[i].output();

found = true;

}

}

if (!found) {

cout << "Студент с датой рождения '" << Fdate << "' не найден." << endl;

}

}

// Добавление записи студента

void addStudent(Students\*& someStudents, short& N) {

Students\* newStudents = new Students[N + 1]; // Создаю новый список студентов, увеличиваю его размерность на 1

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

newStudents[i] = someStudents[i]; // Добавляю все записи из старого списка

}

newStudents[N].input(); // Записываю данные о последнем студенте

delete[] someStudents; // Удаляю данные в старом списке, очищаю память

someStudents = newStudents; // Старому списку присваиваю новый

N++; // Увеличиваю размерность списка на 1

}

// Удаление записи студента

void delStudent(Students\*& someStudents, short& N) {

if (N <= 0) {

cout << "Список студентов пуст!" << endl;

return;

}

cout << "Введите номер студента, которого хотите удалить (от 0 до " << N - 1 << "): ";

int index;

cin >> index;

if (index < 0 || index >= N) {

cout << "Неверный номер студента!" << endl;

return;

}

// Создаем новый список студентов с уменьшенным размером

Students\* newStudents = new Students[N - 1];

// Копируем записи студентов до удаляемой записи

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

newStudents[i] = someStudents[i];

}

// Копируем записи студентов после удаляемой записи

for (int i = index + 1; i < N; i++) {

newStudents[i - 1] = someStudents[i];

}

delete[] someStudents; // Удаляем старый список

someStudents = newStudents; // Присваиваем новый список

N--; // Уменьшаем размерность списка

cout << "Студент удален из списка." << endl;

}

// Сортировка по алфавиту по фамилии

void Sortsurname(Students student[], short N) {

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

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

if (student[j].surname > student[j + 1].surname) {

Students t = student[j];

student[j] = student[j + 1];

student[j + 1] = t;

}

}

}

cout << "Список отсортирован!" << endl;

}

// Сортировка по алфавиту по имени

void Sortname(Students student[], short N) {

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

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

if (student[j].name > student[j + 1].name) {

Students t = student[j];

student[j] = student[j + 1];

student[j + 1] = t;

}

}

}

cout << "Список отсортирован!" << endl;

}

// Вывод данных о студенте (для поиска)

void OutputStudents(int i) {

cout << "| " << i << " | " << setw(18) << name << " | " << setw(21) << surname << " | " << setw(14) << phoneNumber

<< " | " << setw(19) << birthDate << " | " << endl;

}

int getPhone() { return phoneNumber; }

string getSurname() { return surname; }

string getName() { return name; }

string getBirthDate() { return birthDate; }

// Функция для вывода таблицы (разделяющих линий)

void Line() {

for (int i = 0; i < 89; i++) {

cout << "-";

}

cout << endl;

}

};

int main() {

setlocale(LC\_ALL, "");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "Введите количесто студенов= ";

short N = 0;

while (N <= 0) {

cin >> N;

if (N <= 0) {

cout << "Ошибка! Значение кол-ва записей о студентах должно быть больше 0!";

}

}

Students\* student = new Students[N];

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

cout << i + 1 << "-ый студент " << endl;

student[i].input();

cout << endl;

}

cout << "Список заполнен. Выберите метод, который вы хотите попробовать(выберите соотв.номер)" << endl;

short choice;

menu:

cout << endl << setw(24) << "------ МЕНЮ ------" << endl;

cout << "\n1) Сортировать\n2) Поиск по признаку\n3) Добавление записи нового студента к списку студентов \n4) Удаление записи студента " <<

"\n5) Вывести список студентов\n6) Завершить работу программы\n";

cin >> choice;

switch (choice) {

case 1: // Сортировать

short sortChoice;

cout << "Сортировать по(1-по фамилии, 2-по имени, 3-по дате" << endl;

cin >> sortChoice;

switch (sortChoice) {

case 1: student->Sortsurname(student, N);

goto menu;

case 2: student->Sortname(student, N);

goto menu;

case 3:

student->sortBirt(student, N);

goto menu;

case 4: goto menu;

default: cout << "Вы ввели неверный номер!" << endl;

goto menu;

}

break;

case 2: // Поиск по признаку

short findChoice;

cout << "Поиск по(1-фамилии, 2-по имени, 3-номеру телефона, 4-дате рождения, 5- вернуться в меню)" << endl;

cin >> findChoice;

switch (findChoice) {

case 1: student->findSurname(student, N); goto menu;

case 2: student->findName(student, N); goto menu;

case 3: student->findPhone(student, N); goto menu;

case 4: student->findBirthDate(student, N); goto menu;

case 5: goto menu;

default: cout << "Вы ввели неверный номер!" << endl;

goto menu;

}

case 3: // Добавление записи нового студента к списку студентов

student->addStudent(student, N);

cout << "Студент добавлен в список." << endl;

goto menu;

case 4: // Удаление записи студента

student->delStudent(student, N);

goto menu;

case 5: // Вывод списка студентов

student->Line();

cout << "| " << "№" << " | " << setw(18) << "Имя" << " | " << setw(21) << "Фамилия" << " | " << setw(10) << "Номер телефона"

<< " | " << setw(19) << "Дата рождения" << " | " << endl;

student->Line();

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

student[i].OutputStudents(i + 1);

student->Line();

}

goto menu;

case 6: cout << "Программа завершена!" << endl;

system("pause");

return 0;

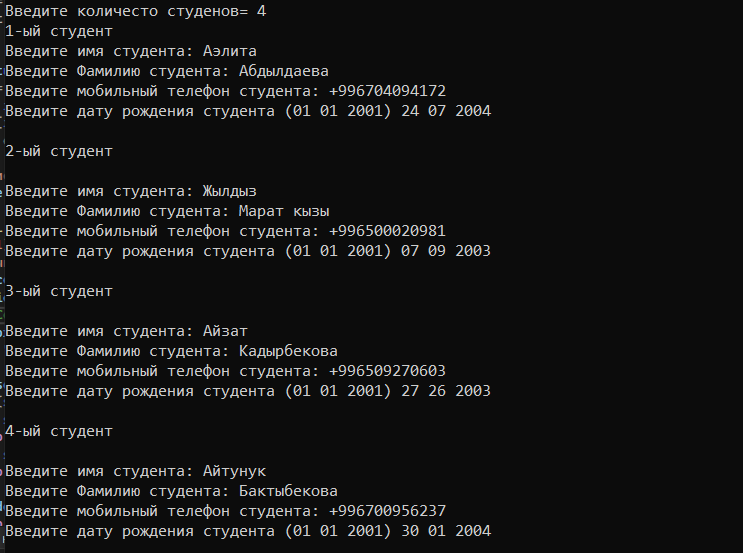
default: cout << "Ошибка! Выбран неверный пункт меню!" << endl;

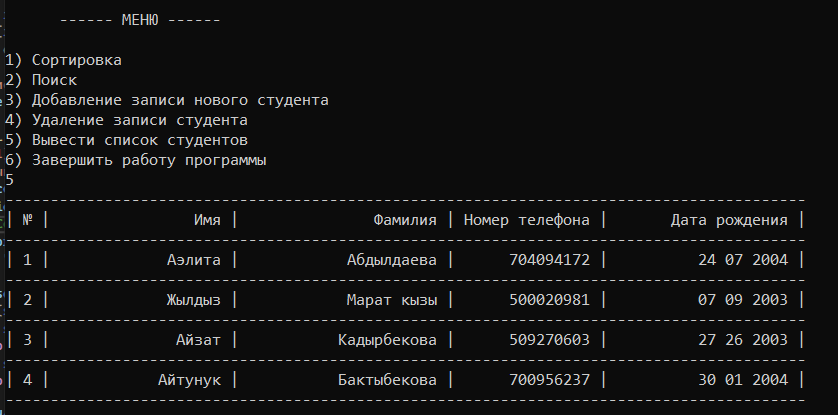
goto menu;

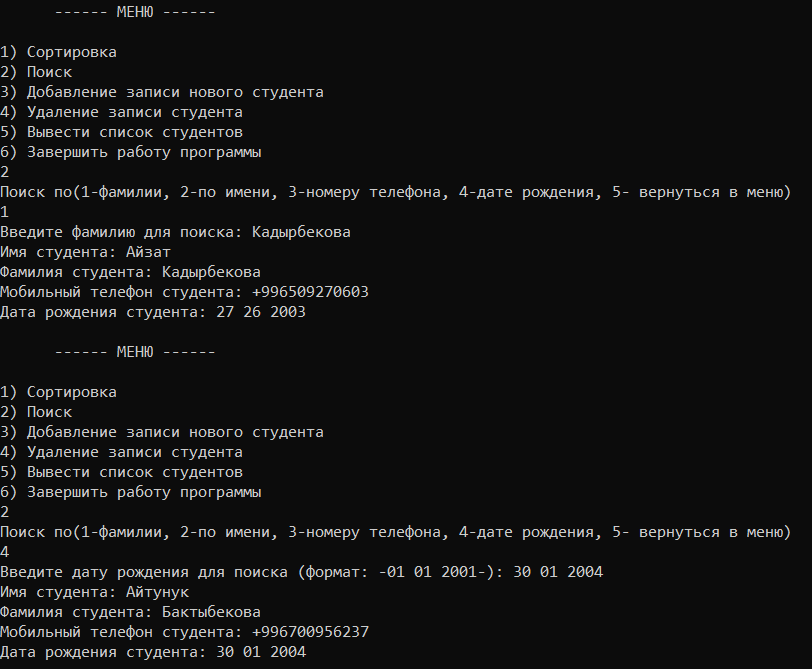
}

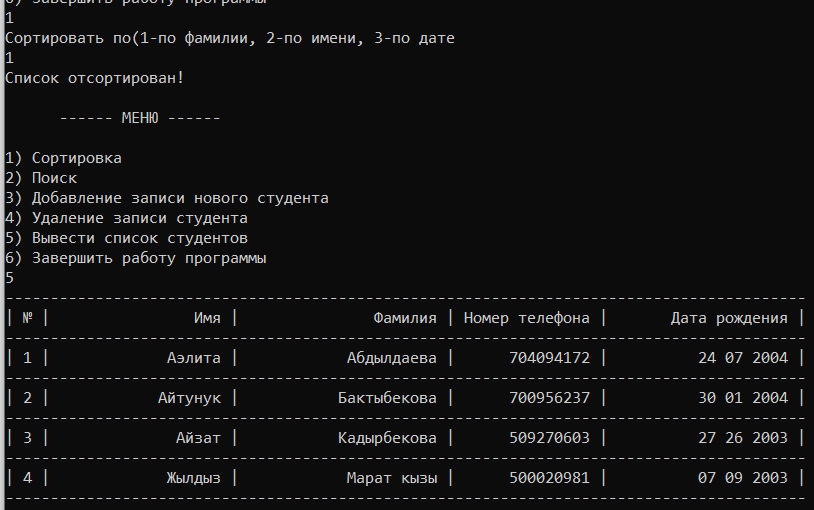
}

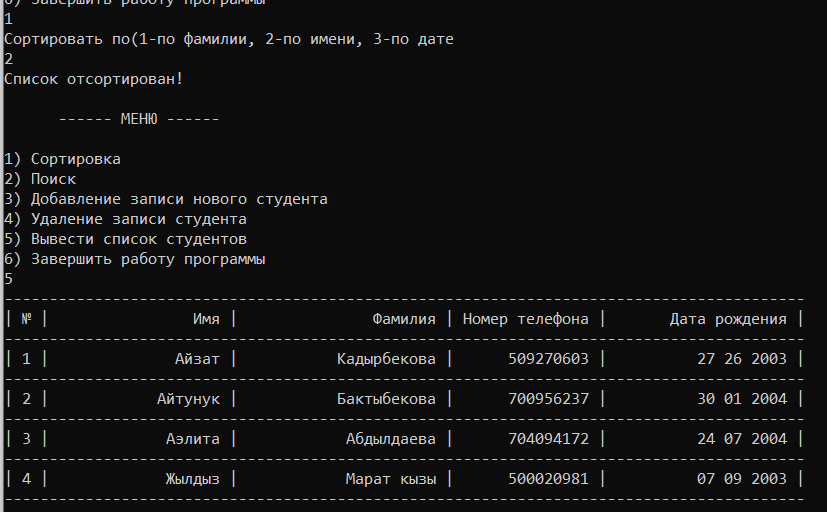
***Результат:***

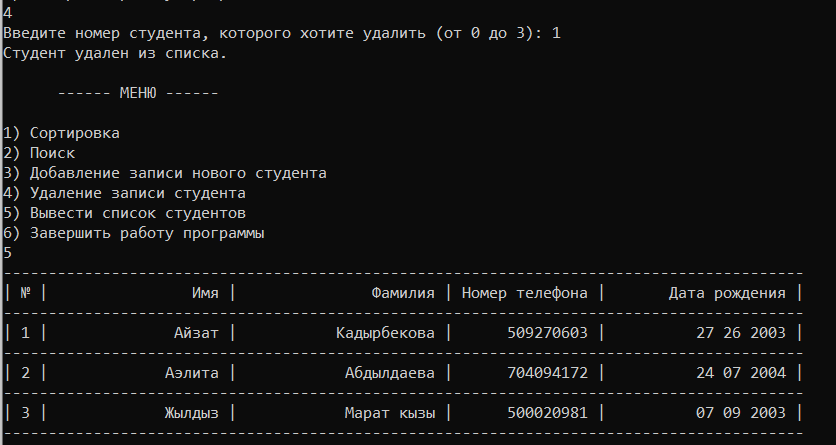


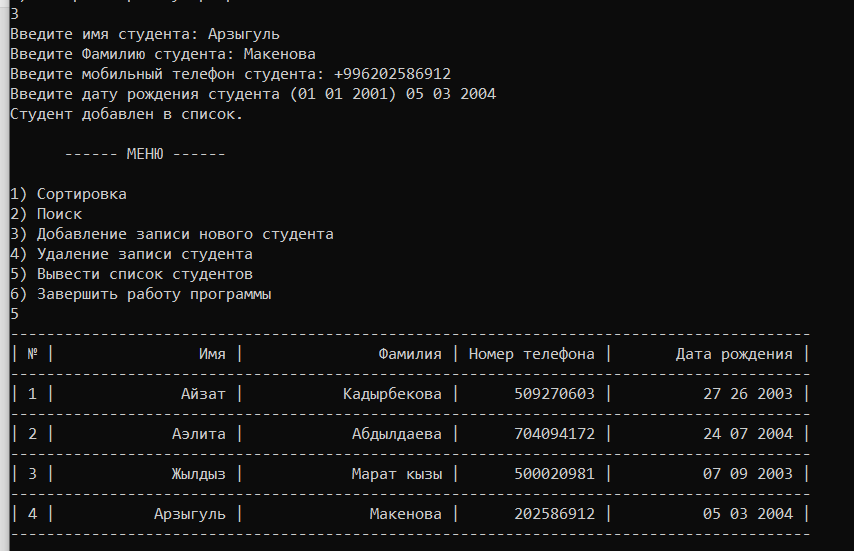












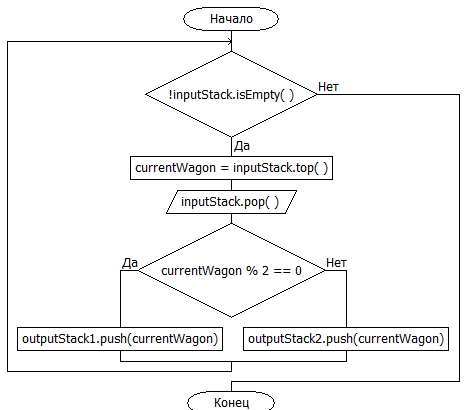
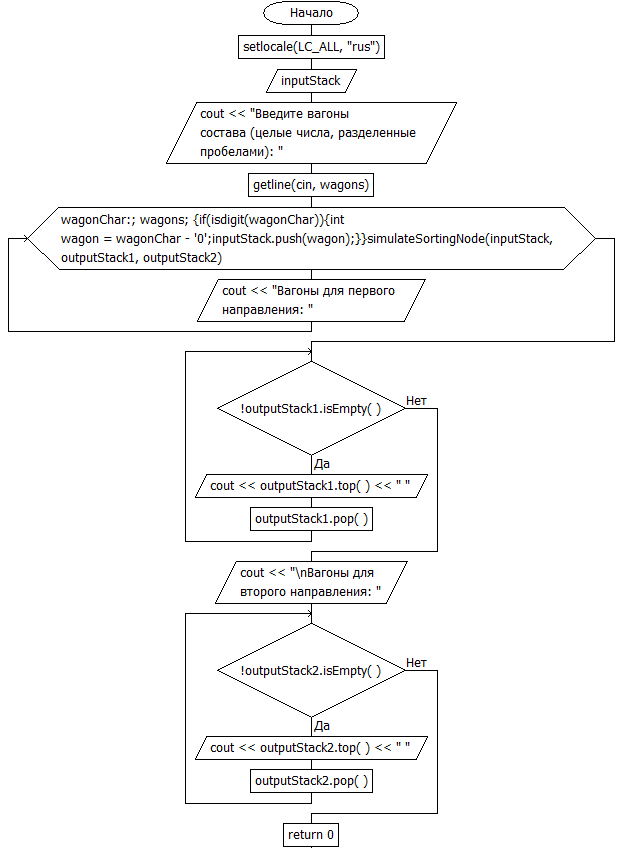
**3. Опишите класс, реализующий стек (Stack).**

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;

***Блок-схема:***



***Код:***

#include <iostream>

#include <stack>

#include <string>

using namespace std;

class Stack {

private:

string elements;

public:

void push(int element) {

elements.push\_back(element + '0'); // Преобразование целого числа в символ и добавление к строке

}

void pop() {

if (!elements.empty()) {

elements.pop\_back();

}

}

bool isEmpty() const {

return elements.empty();

}

int top() const {

if (!elements.empty()) {

return elements.back() - '0'; // Преобразование символа в целое число

}

return -1;

}

};

void simulateSortingNode(Stack& inputStack, Stack& outputStack1, Stack& outputStack2) {

while (!inputStack.isEmpty()) {

int currentWagon = inputStack.top();

inputStack.pop();

// Распределение вагонов по двум направлениям

if (currentWagon % 2 == 0) {

outputStack1.push(currentWagon);

}

else {

outputStack2.push(currentWagon);

}

}

}

int main() {

setlocale(LC\_ALL, "rus");

Stack inputStack;

Stack outputStack1;

Stack outputStack2;

cout << "Введите вагоны состава (целые числа, разделенные пробелами): ";

string wagons;

getline(cin, wagons);

for (char wagonChar : wagons) {

if (isdigit(wagonChar)) {

int wagon = wagonChar - '0';

inputStack.push(wagon);

}

}

simulateSortingNode(inputStack, outputStack1, outputStack2);

// Вывод вагонов для каждого направления

cout << "Вагоны для первого направления: ";

while (!outputStack1.isEmpty()) {

cout << outputStack1.top() << " ";

outputStack1.pop();

}

cout << "\nВагоны для второго направления: ";

while (!outputStack2.isEmpty()) {

cout << outputStack2.top() << " ";

outputStack2.pop();

}

return 0;

}

***Результат:***

