Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«Российский университет транспорта»

(ФГАОУ ВО РУТ(МИИТ), РУТ (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа № 14

по дисциплине: «Программирование и основы алгоритмизации»

на тему: «Классы и объекты»

Выполнил: ст. гр. ТУУ-111

Ванчаев Д.С.

Вариант №10

06.05.2025

(дата выполнения)

Проверил: к.т.н., доц. Сафронов А.И.

(дата приёмки)

**Не увидел в работе соблюдения принципа «один класс – один файл»**

**Не увидел элементов работы со статическим классом-модулем для хранения общедоступных данных**

Москва – 2025 г.

# **1. Цель работы**

В среде *Microsoft Visual Studio* на языке *Visual C#* в консольном режиме составить программное обеспечение для решения типовых задач программирования по тематике «Классы и объекты» («*Classes and Objects*»), заданных по варианту.

# **2. Формулировка задачи**

# Создайте класс: «Ресторан».

Возможное имя класса: «*clsRestaurant*».

**Поля (*fields*):**

– идентификатор ресторана (тип: *int*);

– наименование ресторана (тип: *string*);

– рейтинг ресторана (тип: *byte*);

– город расположения ресторана (тип: *enum*);

– адрес ресторана (тип: *string*);

– кухня ресторана (тип: *enum*);

– уточнение / комментарий о расположении ресторана (тип: *string*);

– время открытия ресторана (тип: *int*);

– время закрытия ресторана (тип: *int*).

**Конструктор (*instance constructor*):**

1. Входные параметры передаются неразобранной строкой текста, считанной из файла.

**Методы (*methods*):**

1. Фильтр ресторанов по городу (имя: *filtByCityName()*; тип: *clsRestaurant[]* или *List<clsRestaurant>*);

2. Фильтр ресторанов по указанному рейтингу (имя: *filtByRate()*; тип: *clsRestaurant[]* или *List<clsRestaurant>*);

3. Фильтр ресторанов по указанной кухне (имя: *filtByCuisine()*; тип: *clsRestaurant[]* или *List<clsRestaurant>*);

4. Принудительное присвоение конкретного значения рейтинга (имя: *forceRateValue()*; тип: *-*);

5. Расчёт длительности работы в часах (имя: *countWorkHours()*; тип: *double*);

6. Формирование полного адреса ресторана (имя: *printAddress()*; тип: *string*);

7. Сортировка ресторанов по рейтингу (имя: *sortByRate()*; тип: *clsRestaurant[]* или *List<clsRestaurant>*);

8. Формирование времени работы ресторана (имя: *printTimeString()*; тип: *string*).

\* – рейтинг ресторана можно изменять в пределах от 1 до 5 звёзд.

**Хранение элементов (*storage*):**

– в текстовом файле (*PupularRestaurants.txt*) с возможностью дополнения.

**Вывод (*ToString()*):**

«1. «Федя, дичь!» (русская кухня); рейтинг: "\*\*\*\*"; адрес: г. Москва, Мытная ул. 74 (Даниловский рынок); время работы: 12:00 – 23:30».

# **3. Блок-схема алгоритма**

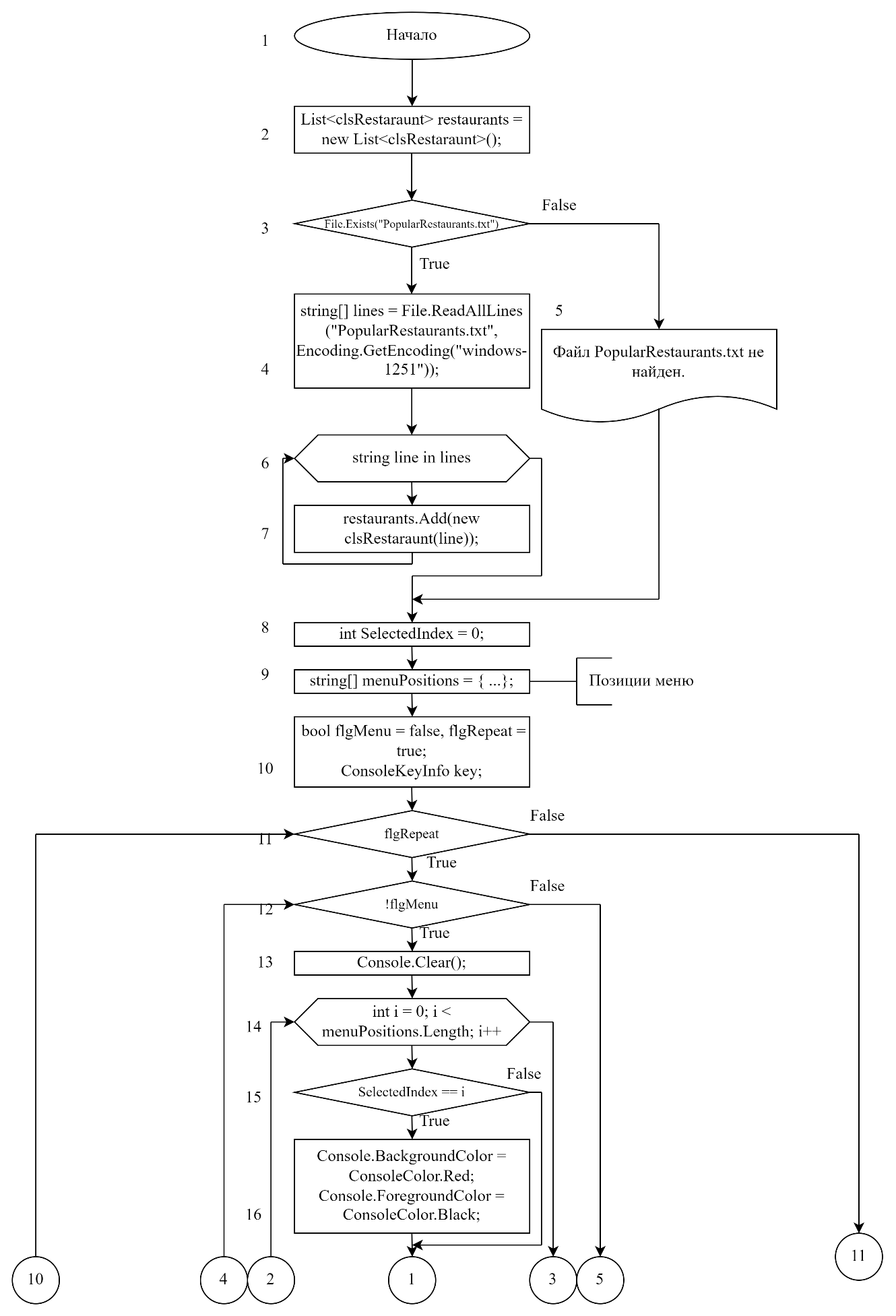
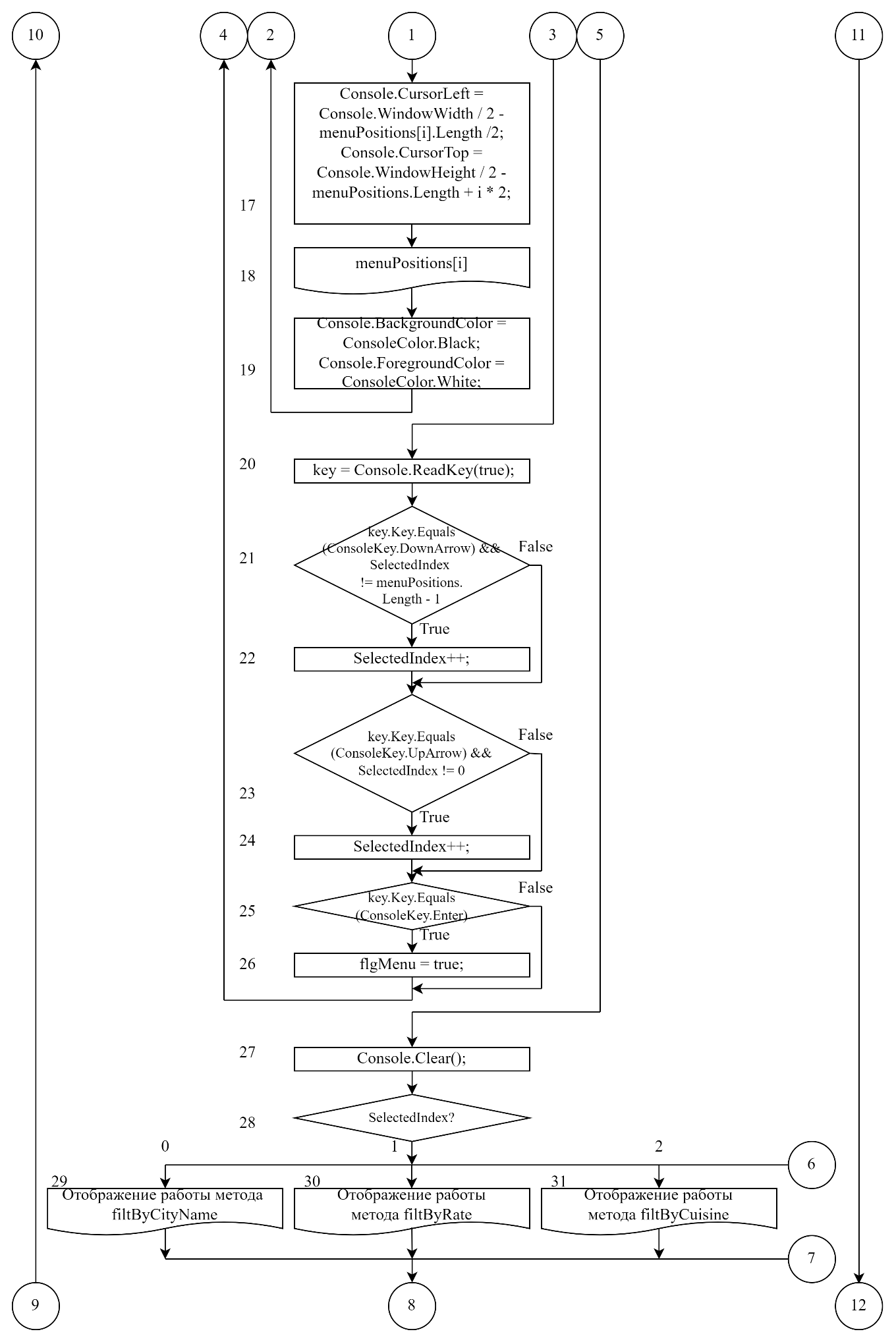
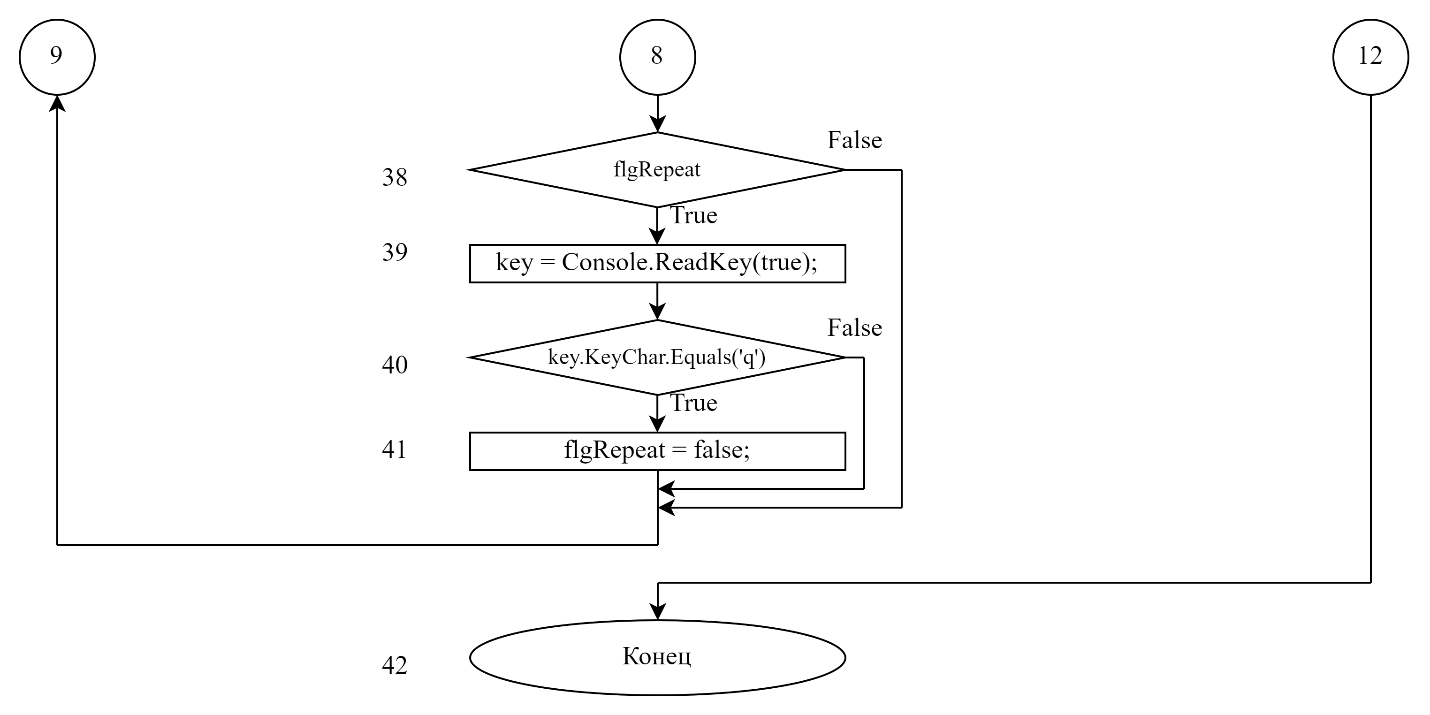


Рисунок 1 – Блок-схема алгоритма

Рисунок 2 – Блок-схема алгоритма



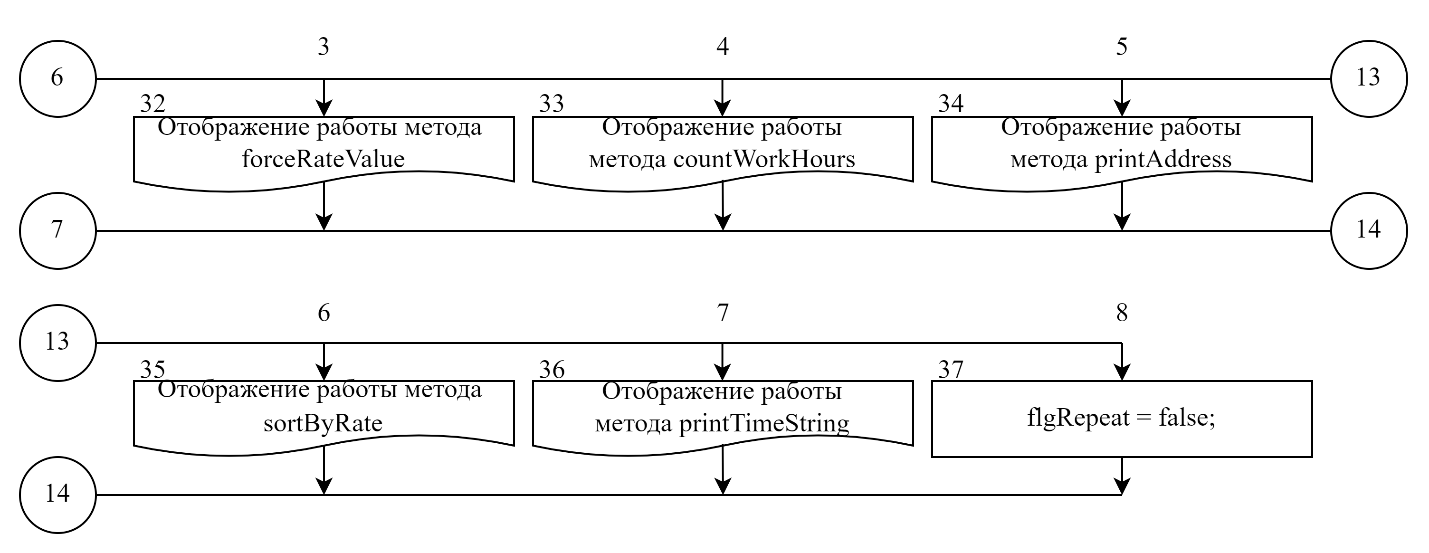
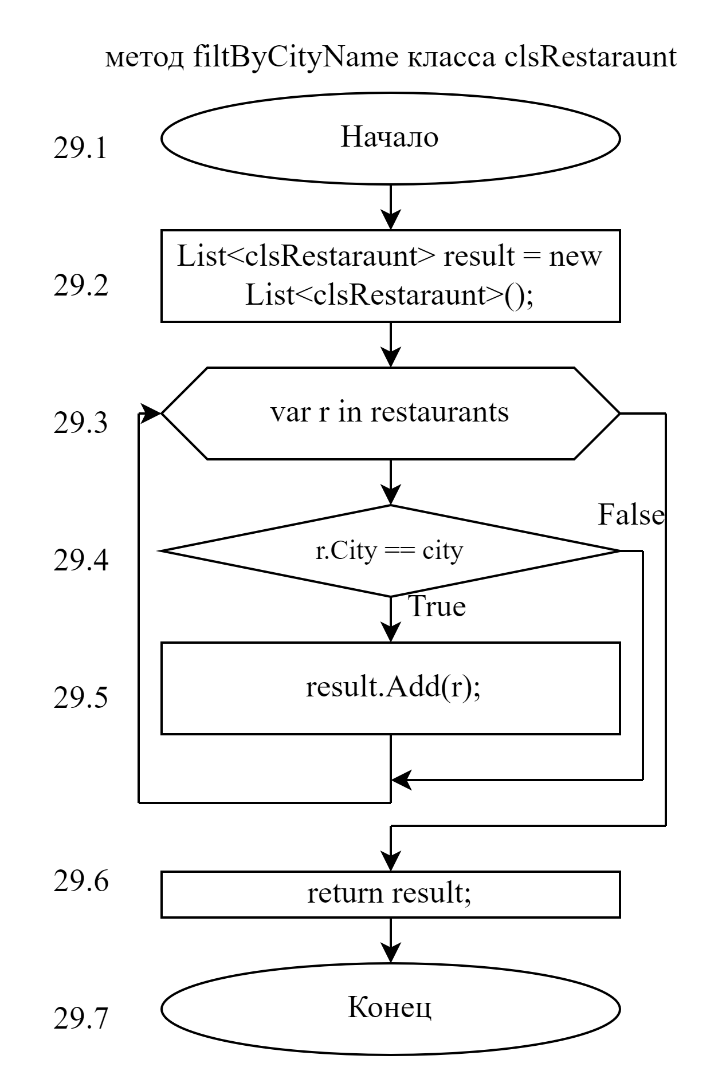
Рисунок 3 – Блок-схема алгоритма

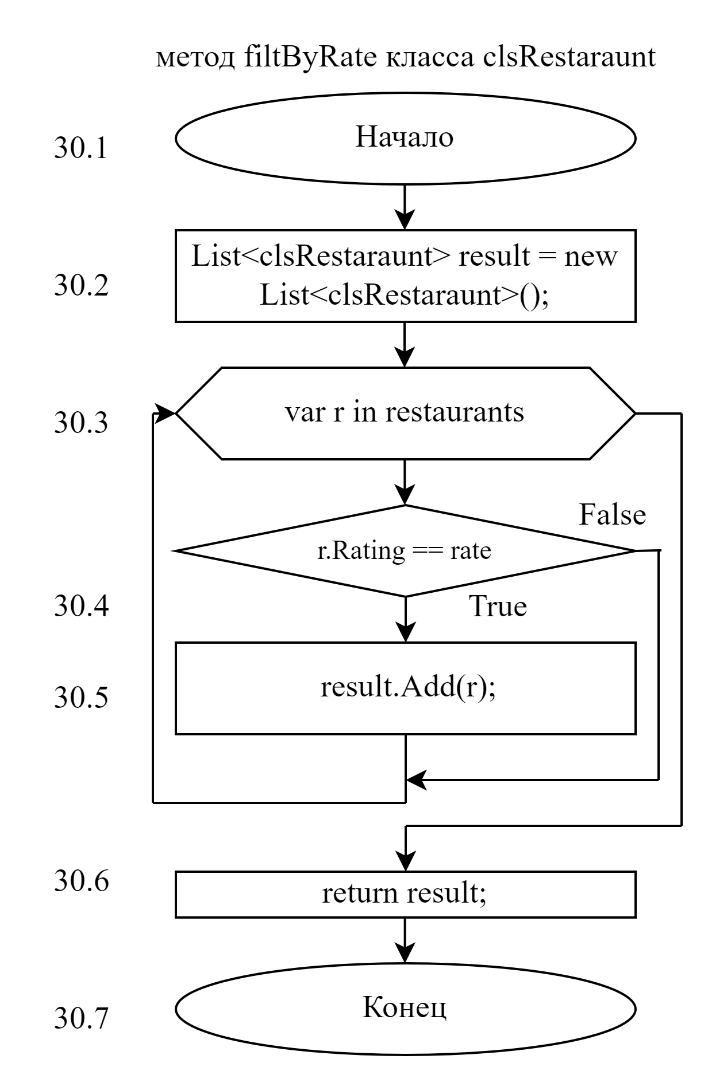
Рисунок 4 – Блок-схема алгоритма



Должна быть жёстка типизация. Элементов Java здесь быть не должно

Возврат оформляется не этим блоком и служебное слово возврата не упоминается

Рисунок 5 – Блок-схема метода *filtByCityName* класса *clsRestaraunt*



Должна быть жёстка типизация. Элементов Java здесь быть не должно

Возврат оформляется не этим блоком и служебное слово возврата не упоминается

Рисунок 6 – Блок-схема метода *filtByRate* класса *clsRestaraunt*



Возврат оформляется не этим блоком и служебное слово возврата не упоминается

Должна быть жёстка типизация. Элементов Java здесь быть не должно

Рисунок 7 – Блок-схема метода *filtByCuisine* класса *clsRestaraunt*

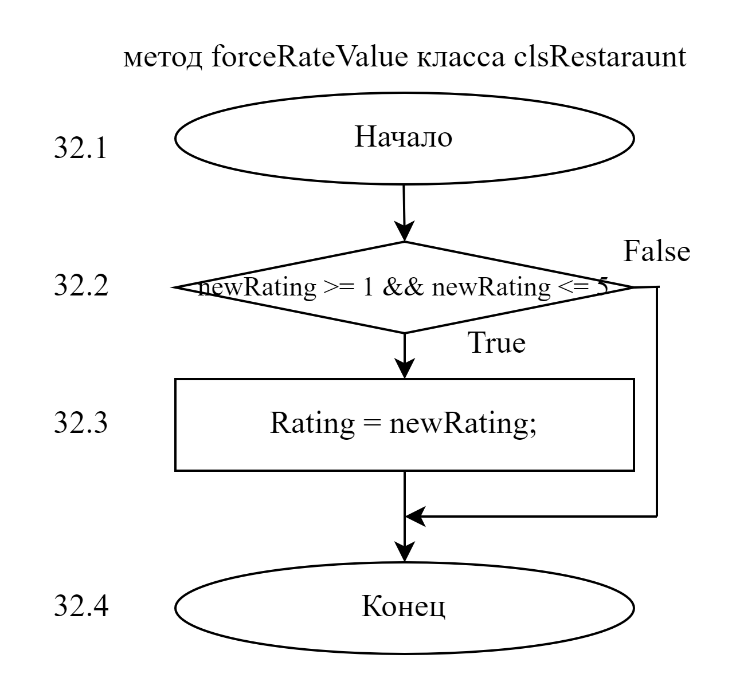


Рисунок 8 – Блок-схема метода *forceRateValue* класса *clsRestaraunt*

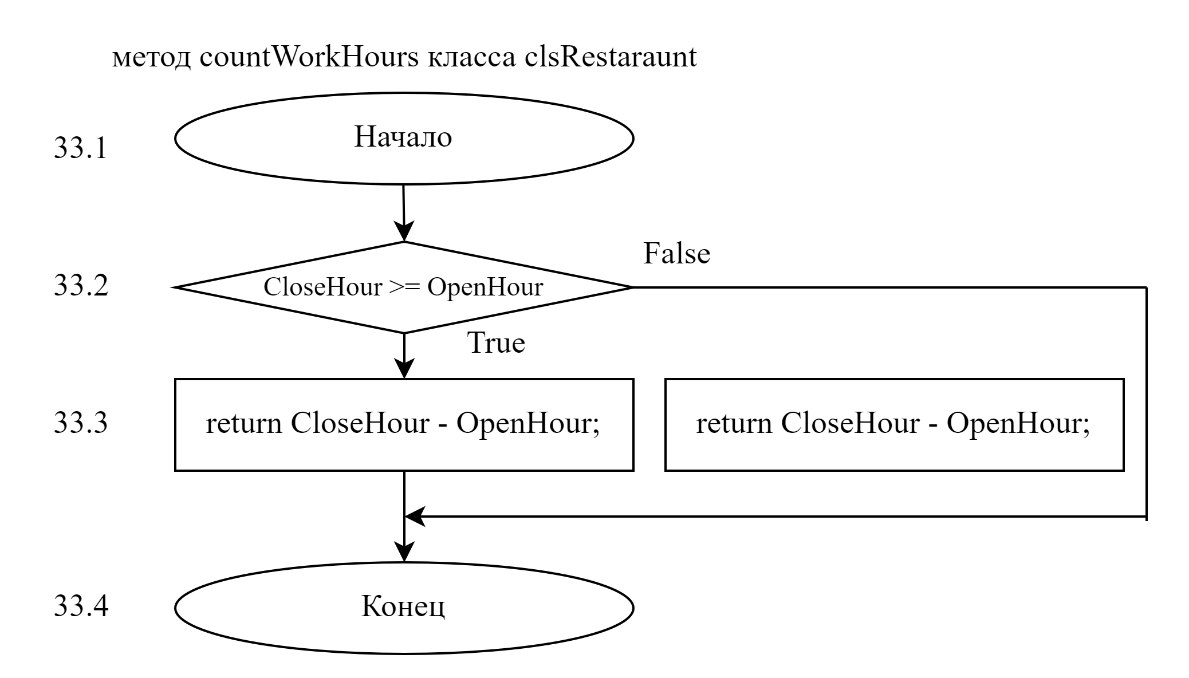
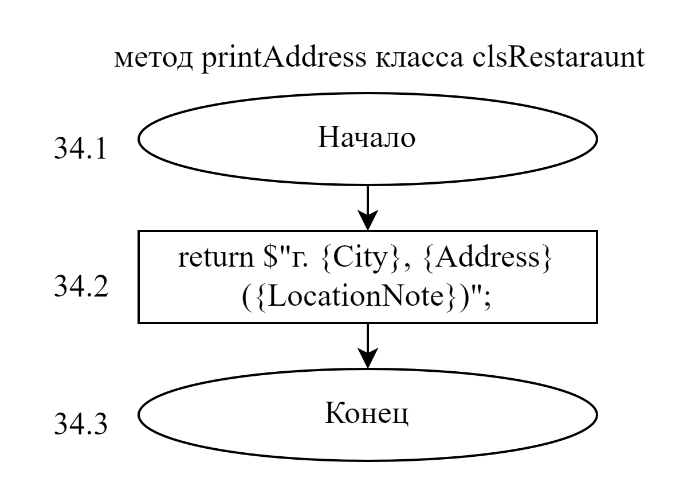
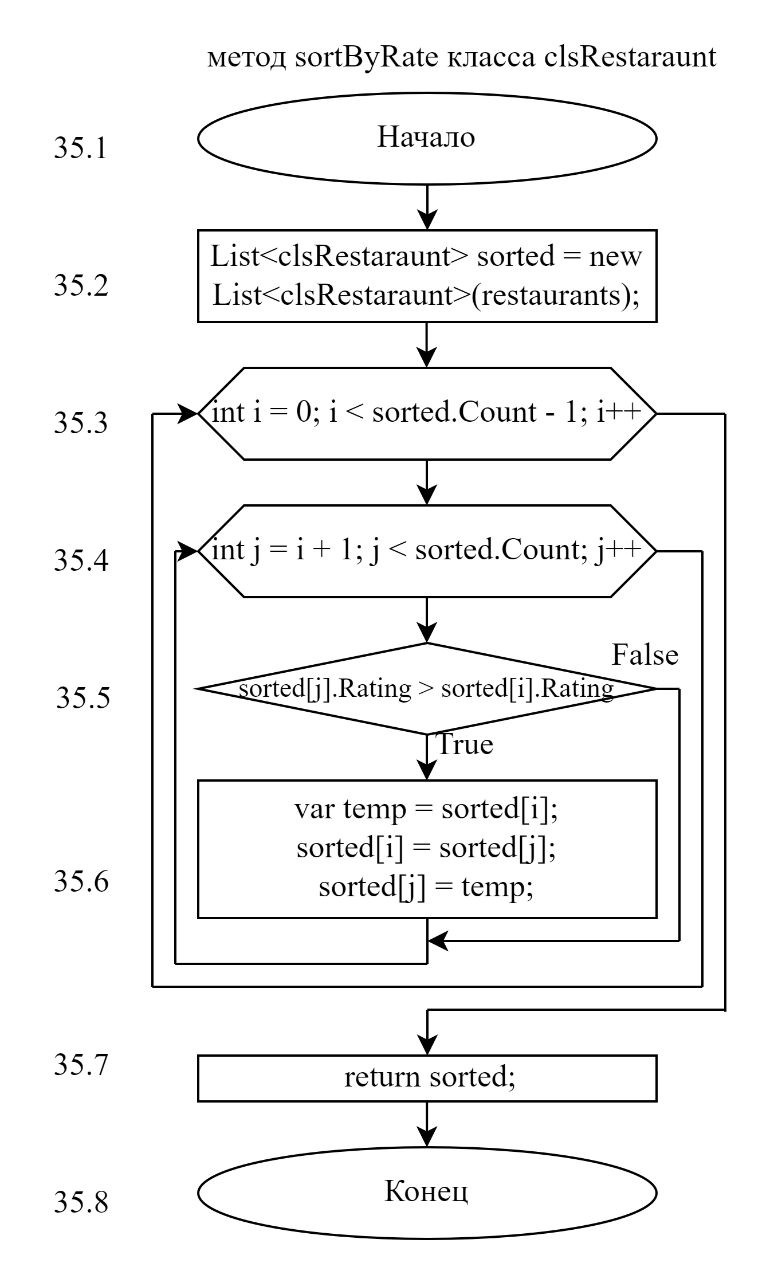


Рисунок 9 – Блок-схема метода *WorkHours* класса *clsRestaraunt*



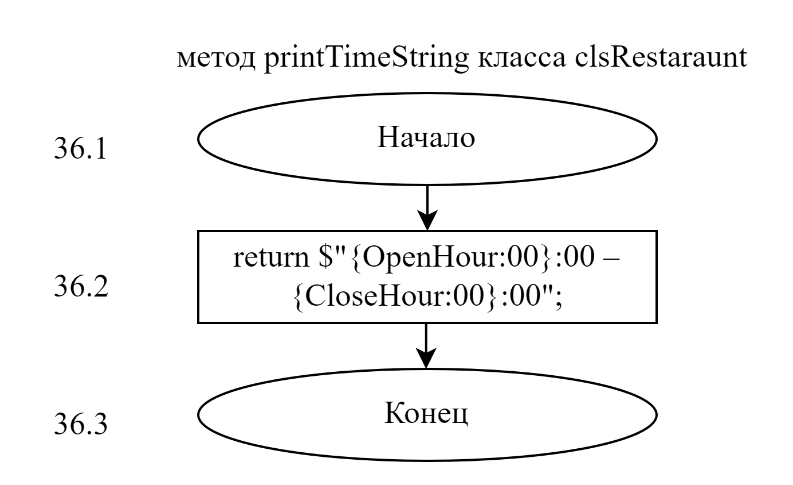
Возврат оформляется не этим блоком и служебное слово возврата не упоминается

Рисунок 10 – Блок-схема метода *PrintAdress* класса *clsRestaraunt*



Возврат оформляется не этим блоком и служебное слово возврата не упоминается

Рисунок 11 – Блок-схема метода *SortByRate* класса *clsRestaraunt*

**

Возврат оформляется не этим блоком и служебное слово возврата не упоминается

Рисунок 12 – Блок-схема метода *printTimeString* класса *clsRestaraunt*

# **4. Подбор тестовых примеров**

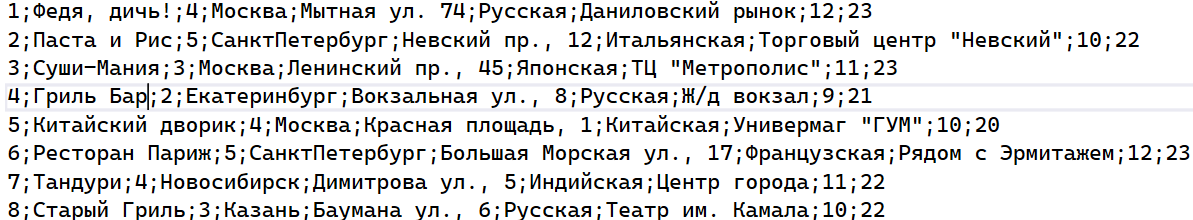
Исходные данные в файле *PupularRestaurants.txt* на Рисунке 13:

Рисунок 12 – Исходные данные

1. Фильтр ресторанов по городу

Город: Москва

Ожидаемый результат: 1. <Федя, дичь!> (русская кухня); рейтинг: "\*\*\*\*"; адрес: г. Москва, Мытная ул. 74 (Даниловский рынок); время работы: 12:00 - 23:00

3. <Суши-Мания> (японская кухня); рейтинг: "\*\*\*"; адрес: г. Москва, Ленинский пр., 45 (ТЦ "Метрополис"); время работы: 11:00 - 23:00

5. <Китайский дворик> (китайская кухня); рейтинг: "\*\*\*\*"; адрес: г. Москва, Красная площадь, 1 (Универмаг "ГУМ"); время работы: 10:00 - 20:00

1. Фильтр ресторанов по рейтингу

Рейтинг: 4

Ожидаемый результат: Введите рейтинг: 4

1. <Федя, дичь!> (русская кухня); рейтинг: "\*\*\*\*"; адрес: г. Москва, Мытная ул. 74 (Даниловский рынок); время работы: 12:00 - 23:00

5. <Китайский дворик> (китайская кухня); рейтинг: "\*\*\*\*"; адрес: г. Москва, Красная площадь, 1 (Универмаг "ГУМ"); время работы: 10:00 - 20:00

7. <Тандури> (индийская кухня); рейтинг: "\*\*\*\*"; адрес: г. Новосибирск, Димитрова ул., 5 (Центр города); время работы: 11:00 - 22:00

1. Фильтр ресторанов по указанной кухне

Кухня: Японская

Ожидаемый результат: 3. <Суши-Мания> (японская кухня); рейтинг: "\*\*\*"; адрес: г. Москва, Ленинский пр., 45 (ТЦ "Метрополис"); время работы: 11:00 - 23:00

1. Принудительное присвоение конкретного значения рейтинга

Ресторан: Паста и Рис

Рейтинг: 2

Ожидаемый результат: 2. <Паста и Рис> (итальянская кухня); рейтинг: "\*\*"; адрес: г. СанктПетербург, Невский пр., 12 (Торговый центр "Невский"); время работы: 10:00 - 22:00

1. Расчёт длительности работы в часах

Ресторан: Тандури

Ожидаемый результат: 11

1. Формирование полного адреса ресторана

Ресторан: Китайский дворик

Ожидаемый результат: Адрес: г. Москва, Красная площадь, 1 (Универмаг "ГУМ")

1. Сортировка ресторанов по рейтингу

Ожидаемый результат: 6. <Ресторан Париж> (французская кухня); рейтинг: "\*\*\*\*\*"; адрес: г. СанктПетербург, Большая Морская ул., 17 (Рядом с Эрмитажем); время работы: 12:00 - 23:00

5. <Китайский дворик> (китайская кухня); рейтинг: "\*\*\*\*"; адрес: г. Москва, Красная площадь, 1 (Универмаг "ГУМ"); время работы: 10:00 - 20:00

1. <Федя, дичь!> (русская кухня); рейтинг: "\*\*\*\*"; адрес: г. Москва, Мытная ул. 74 (Даниловский рынок); время работы: 12:00 - 23:00

7. <Тандури> (индийская кухня); рейтинг: "\*\*\*\*"; адрес: г. Новосибирск, Димитрова ул., 5 (Центр города); время работы: 11:00 - 22:00

3. <Суши-Мания> (японская кухня); рейтинг: "\*\*\*"; адрес: г. Москва, Ленинский пр., 45 (ТЦ "Метрополис"); время работы: 11:00 - 23:00

8. <Старый Гриль> (русская кухня); рейтинг: "\*\*\*"; адрес: г. Казань, Баумана ул., 6 (Театр им. Камала); время работы: 10:00 - 22:00

2. <Паста и Рис> (итальянская кухня); рейтинг: "\*\*"; адрес: г. СанктПетербург, Невский пр., 12 (Торговый центр "Невский"); время работы: 10:00 - 22:00

4. <Гриль Бар> (русская кухня); рейтинг: "\*\*"; адрес: г. Екатеринбург, Вокзальная ул., 8 (Ж/д вокзал); время работы: 09:00 - 21:00

1. Формирование времени работы ресторана

Ресторан: Старый Гриль

Ожидаемый результат: 10:00 - 22:00

# **5. Листинг (код) программы**

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace Classes

{

internal class Program

{

static void Main(string[] args)

{

List<clsRestaraunt> restaurants = new List<clsRestaraunt>();

if (File.Exists("PopularRestaurants.txt"))

{

string[] lines = File.ReadAllLines("PopularRestaurants.txt", Encoding.GetEncoding("windows-1251"));

foreach (string line in lines)

{

restaurants.Add(new clsRestaraunt(line));

}

}

else

{

Console.WriteLine("Файл PopularRestaurants.txt не найден.");

return;

}

int SelectedIndex = 0;

string[] menuPositions = { "Фильтр ресторанов по городу",

"Фильтр ресторанов по указанному рейтингу",

"Фильтр ресторанов по указанной кухне",

"Принудительное присвоение конкретного значения рейтинга",

"Расчёт длительности работы в часах",

"Формирование полного адреса ресторана",

"Сортировка ресторанов по рейтингу",

"Формирование времени работы ресторана",

"Выход" };

bool flgMenu = false, flgRepeat = true;

ConsoleKeyInfo key;

while(flgRepeat)

{

while(!flgMenu)

{

Console.Clear();

for(int i = 0; i < menuPositions.Length; i++)

{

if (SelectedIndex == i)

{

Console.BackgroundColor = ConsoleColor.Red;

Console.ForegroundColor = ConsoleColor.Black;

}

Console.CursorLeft = Console.WindowWidth / 2 - menuPositions[i].Length /2;

Console.CursorTop = Console.WindowHeight / 2 - menuPositions.Length + i \* 2;

Console.WriteLine(menuPositions[i]);

Console.BackgroundColor = ConsoleColor.Black;

Console.ForegroundColor = ConsoleColor.White;

}

key = Console.ReadKey(true);

if (key.Key.Equals(ConsoleKey.DownArrow) && SelectedIndex != menuPositions.Length - 1)

SelectedIndex++;

if (key.Key.Equals(ConsoleKey.UpArrow) && SelectedIndex != 0)

SelectedIndex--;

if (key.Key.Equals(ConsoleKey.Enter))

flgMenu = true;

}

Console.Clear();

switch(SelectedIndex)

{

case 0:

Console.Write("Введите город: ");

string cityInput = Console.ReadLine();

City city = (City)Enum.Parse(typeof(City), cityInput, true);

List<clsRestaraunt> result = clsRestaraunt.filtByCityName(restaurants, city);

foreach (var r in result)

{

Console.WriteLine(r.ToString());

}

break;

case 1:

Console.Write("Введите рейтинг: ");

byte raiting = byte.Parse(Console.ReadLine());

result = clsRestaraunt.filtByRate(restaurants, raiting);

foreach (var r in result)

{

Console.WriteLine(r.ToString());

}

flgMenu = false;

break;

case 2:

Console.Write("Введите кухню: ");

string cuisineInput = Console.ReadLine();

Cuisine cuisine = (Cuisine)Enum.Parse(typeof(Cuisine), cuisineInput, true);

result = clsRestaraunt.filtByCuisine(restaurants, cuisine);

foreach (var r in result)

{

Console.WriteLine(r.ToString());

}

flgMenu = false;

break;

case 3:

Console.Write("Введите ресторан: ");

string restaraunt = Console.ReadLine();

Console.Write("Введите рейтинг: ");

raiting = byte.Parse(Console.ReadLine());

foreach(var r in restaurants)

{

if(r.Name == restaraunt)

{

r.forceRateValue(raiting);

Console.WriteLine(r.ToString());

break;

}

}

flgMenu = false;

break;

case 4:

Console.Write("Введите ресторан: ");

restaraunt = Console.ReadLine();

foreach (var r in restaurants)

{

if (r.Name == restaraunt)

{

Console.WriteLine("Время работы: " + r.countWorkHours());

break;

}

}

flgMenu = false;

break;

case 5:

Console.Write("Введите ресторан: ");

restaraunt = Console.ReadLine();

foreach (var r in restaurants)

{

if (r.Name == restaraunt)

{

Console.WriteLine("Адрес: " + r.printAddress());

break;

}

}

flgMenu = false;

break;

case 6:

result = clsRestaraunt.sortByRate(restaurants);

foreach(var r in result)

{

Console.WriteLine(r.ToString());

}

flgMenu = false;

break;

case 7:

Console.Write("Введите ресторан: ");

restaraunt = Console.ReadLine();

foreach (var r in restaurants)

{

if (r.Name == restaraunt)

{

Console.WriteLine("Время работы: " + r.printTimeString());

break;

}

}

flgMenu = false;

break;

case 8:

flgRepeat = false;

break;

}

if (flgRepeat)

{

key = Console.ReadKey(true);

if (key.KeyChar.Equals('q'))

flgRepeat = false;

}

}

}

}

public enum City

{

Москва,

СанктПетербург,

Казань,

Екатеринбург,

Новосибирск

}

public enum Cuisine

{

Русская,

Итальянская,

Японская,

Китайская,

Французская,

Индийская

}

public class clsRestaraunt

{

public int Id { get; set; }

public string Name { get; set; }

public byte Rating { get; set; }

public City City { get; set; }

public string Address { get; set; }

public Cuisine Cuisine { get; set; }

public string LocationNote { get; set; }

public int OpenHour { get; set; }

public int CloseHour { get; set; }

public clsRestaraunt(string lineFromFile)

{

string[] parts = lineFromFile.Split(';');

Id = int.Parse(parts[0]);

Name = parts[1];

Rating = byte.Parse(parts[2]);

City = (City)Enum.Parse(typeof(City), parts[3]);

Address = parts[4];

Cuisine = (Cuisine)Enum.Parse(typeof(Cuisine), parts[5]);

LocationNote = parts[6];

OpenHour = int.Parse(parts[7]);

CloseHour = int.Parse(parts[8]);

}

// Метод 1 – Фильтр по городу

public static List<clsRestaraunt> filtByCityName(List<clsRestaraunt> restaurants, City city)

{

List<clsRestaraunt> result = new List<clsRestaraunt>();

foreach (var r in restaurants)

{

if (r.City == city)

result.Add(r);

}

return result;

}

// Метод 2 – Фильтр по рейтингу

public static List<clsRestaraunt> filtByRate(List<clsRestaraunt> restaurants, byte rate)

{

List<clsRestaraunt> result = new List<clsRestaraunt>();

foreach (var r in restaurants)

{

if (r.Rating == rate)

result.Add(r);

}

return result;

}

// Метод 3 – Фильтр по кухне

public static List<clsRestaraunt> filtByCuisine(List<clsRestaraunt> restaurants, Cuisine cuisine)

{

List<clsRestaraunt> result = new List<clsRestaraunt>();

foreach (var r in restaurants)

{

if (r.Cuisine == cuisine)

result.Add(r);

}

return result;

}

// Метод 4 – Принудительное изменение рейтинга

public void forceRateValue(byte newRating)

{

if (newRating >= 1 && newRating <= 5)

Rating = newRating;

}

// Метод 5 – Расчёт количества рабочих часов

public double countWorkHours()

{

if (CloseHour >= OpenHour)

return CloseHour - OpenHour;

else

return 24 - OpenHour + CloseHour;

}

// Метод 7 – Сортировка по рейтингу (по убыванию)

public static List<clsRestaraunt> sortByRate(List<clsRestaraunt> restaurants)

{

List<clsRestaraunt> sorted = new List<clsRestaraunt>(restaurants);

for (int i = 0; i < sorted.Count - 1; i++)

{

for (int j = i + 1; j < sorted.Count; j++)

{

if (sorted[j].Rating > sorted[i].Rating)

{

var temp = sorted[i];

sorted[i] = sorted[j];

sorted[j] = temp;

}

}

}

return sorted;

}

public override string ToString()

{

string stars = new string('\*', Rating);

return $"{Id}. «{Name}» ({Cuisine.ToString().ToLower()} кухня); рейтинг: \"{stars}\"; адрес: {printAddress()}; время работы: {printTimeString()}";

}

public string printAddress()

{

return $"г. {City}, {Address} ({LocationNote})";

}

public string printTimeString()

{

return $"{OpenHour:00}:00 – {CloseHour:00}:00";

}

}

}

# **6. Расчет тестовых примеров на ПК**

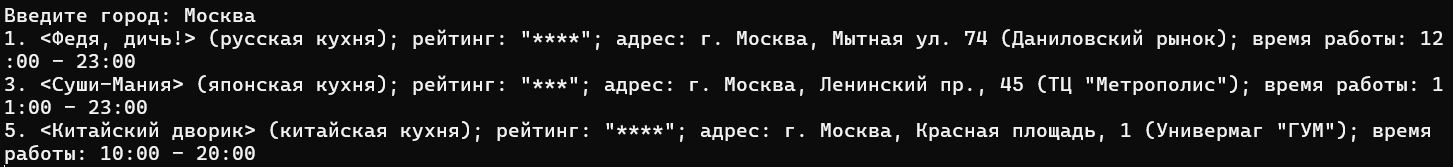


Рисунок 13 – Расчёт первого тестового примера

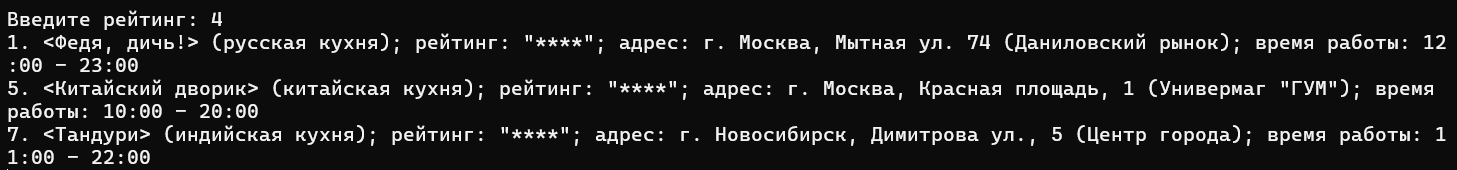


Рисунок 14 – Расчёт второго тестового примера

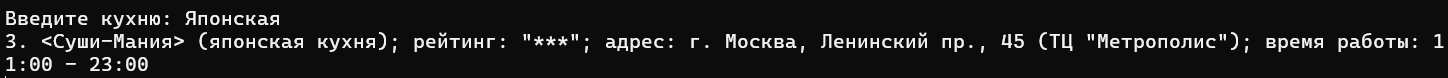


Рисунок 15 – Расчёт третьего тестового примера

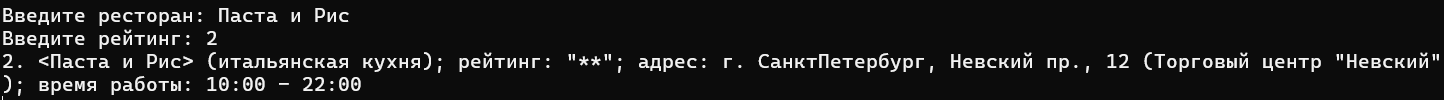


Рисунок 16 – Расчёт четвертого тестового примера



Рисунок 17 – Расчёт пятого тестового примера



Рисунок 18 – Расчёт шестого тестового примера

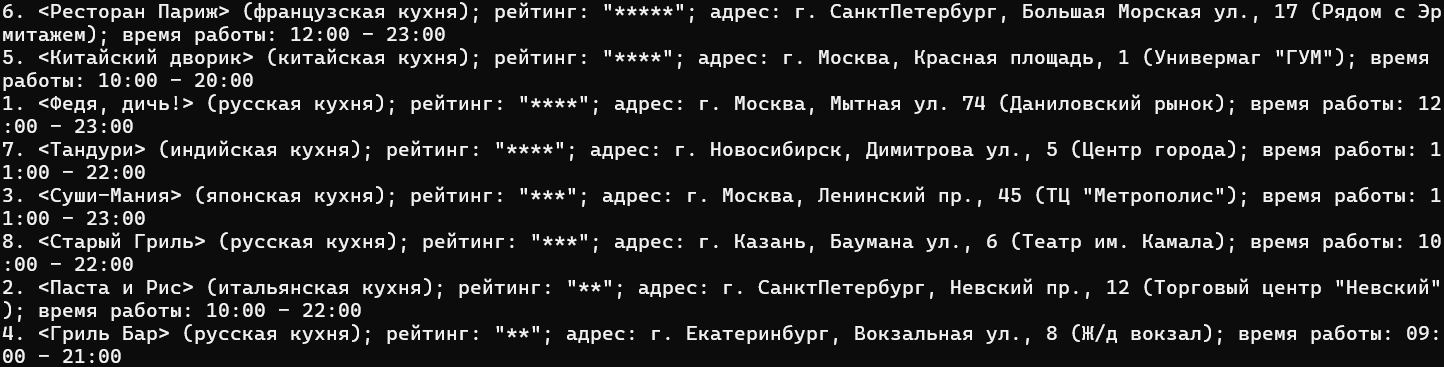


Рисунок 19 – Расчёт седьмого тестового примера



Рисунок 19 – Расчёт восьмого тестового примера

# **7. Вывод по работе**

В среде *Microsoft Visual Studio* на языке *Visual C#* в консольном режиме составлено программное обеспечение для решения типовых задач программирования по тематике «Классы и объекты» («*Classes and Objects*»), заданных по варианту. Был получен навык описания классов, использования объектов класса, переопределение методов, а также построение тестового меню в консоли. Был закреплен навык работы с перечислениями, циклами и списками.