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

Михалин. А. В. Вариант №2

1.06.2025

(дата выполнения)

Проверил: к.т.н., доц. Сафронов А.И.

(дата приёмки)

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: «Интернет-сайт».

Возможное имя класса: «*clsSite*».

**Поля (*fields*):**

– уникальный номер сайта в списке (тип: *short*);

– протокол доступа к сайту (тип: *string*);

– доменная зона сайта (тип: *string*);

– хостинг / владелец сайта (тип: *string*);

– верхние уровни домена сайта (тип: *string[]*);

– *IP*-адрес сайта (тип: *long*);

– название заглавной страницы сайта (тип: *string*);

– название текущей просматриваемой страницы сайта (тип: *string*);

– дата посещения сайта (тип: *long*);

– признак защищённости протокола (тип: *bool*).

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

1. Пять входных параметров\*:

– адресная строка сайта,

– строка с заголовком сайта,

– строка с названием текущей страницы сайта,

– строка *IP*-адреса (в формате *IPv4*),

– строка с датой посещения сайта (в формате дд.мм.гггг).

\* – идентификатор сайта не хранится в файле, он назначается по порядковому номеру строки, содержащей, собственно, сайт и его параметры.

**Методы (*methods*):**

1. Формирование адресной строки сайта для вставки в браузер по параметрам сайта: (имя: *createSiteString*; тип: *string*).

2. Срез верхних уровней домена из адресной строки сайта, сформированной для браузера (имя: *getUpperDomainLevels*; тип: *string[]*).

3. Формирование библиографической ссылки\*\* на сайт (имя: *formatBibliography*; тип: *-*).

4. Изменение даты посещения сайта по входному строковому представлению (имя: *changeVisitDateByString*; тип: *long*).

5. Определение защищённости\*\*\* протокола (имя: *detectSecurity*; тип: *bool*).

\*\*\* – последняя буква «*s*» в названии протокола является признаком защищённости протокола, в частности *http*.

**Хранение элементов (*storage*):**

– адреса сайтов и их параметры записаны в текстовый файл; файл дополняется и обновляется только вручную через «Блокнот»; изменения, выполняемые в процессе работы программы, не сказываются на содержимом файла;

– массивы и списки для хранения не предусмотрены.

**Вывод (*ToString()*):**

Выводить информацию о сайте в формате:

«Яндекс [[*https://ya.ru*](https://ya.ru/)] – *IP*: 5.255.255.242»

«Ресторан *MANSARD* [[*http://mansard-moscow.ru*](http://mansard-moscow.ru/)] – *IP*: 185.165.123.36»

\*\* – библиографическая ссылка на сайт: «{название сайта} [Электронный ресурс] : {название текущей страницы сайта}. *URL*: {адресная строка сайта} (дата обращения: {дата в формате дд.мм.гггг})»

Параметрические скобки {} – не печатаемая информация.

Пример: «*Lenta.ru* – Новости России и мира сегодня [Электронный ресурс] : Эффективные кадры. Московский метрополитен увеличивает набор водителей трамвая в два раза. *URL*: *https:/​/​lenta.ru/​articles/​2022/​03/​18/​moskkadry/* (дата обращения: 23.03.2025)».

**3. Блок-схема алгоритма**



Рисунок 3.1 – Блок-схема алгоритма

****

Рисунок 3.2 – Блок-схема алгоритма

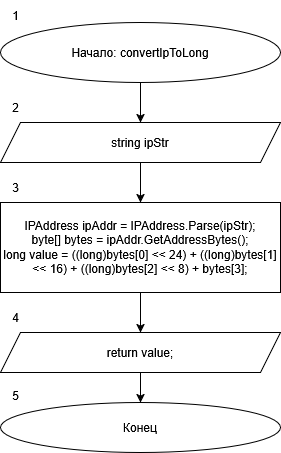
****

Рисунок 3.3 – Блок-схема алгоритма

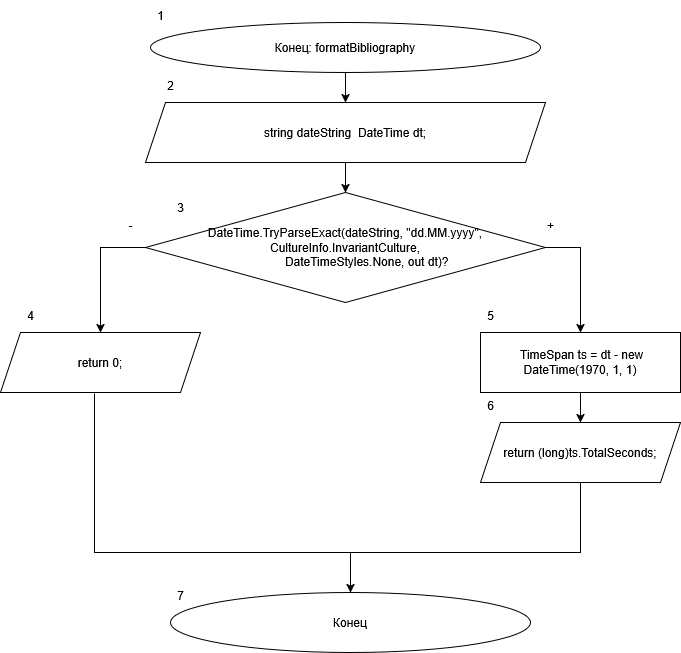
****

Рисунок 3.4 – Блок-схема алгоритма

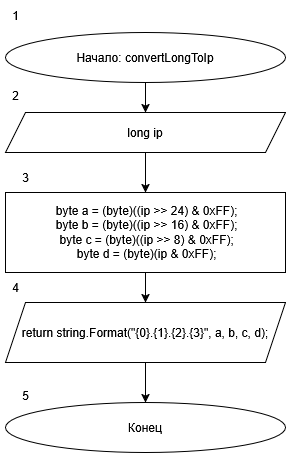
****

Рисунок 3.5 – Блок-схема алгоритма

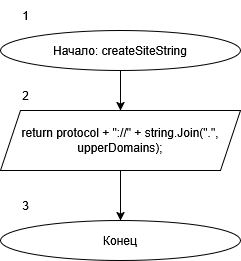
****

Рисунок 3.6 – Блок-схема алгоритма

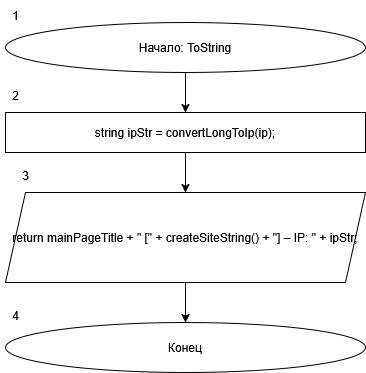
****

Рисунок 3.7 – Блок-схема алгоритма

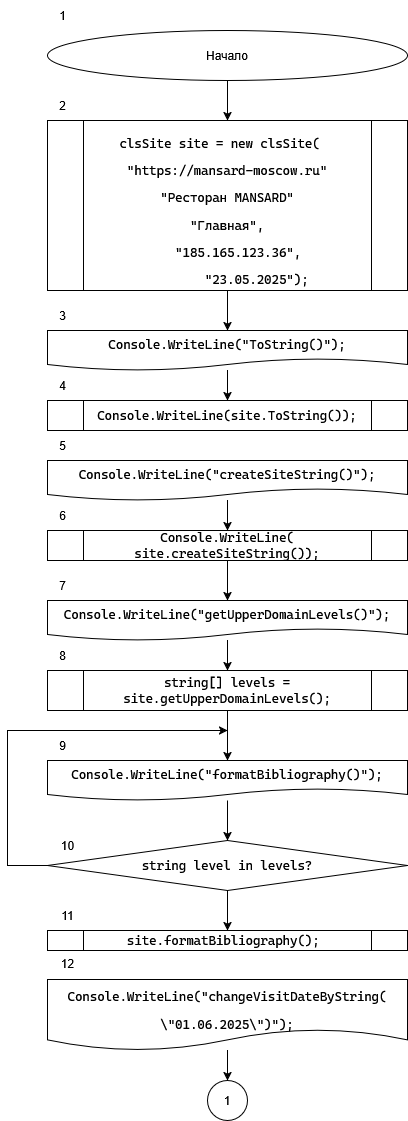
****

Рисунок 3.8 – Блок-схема алгоритма

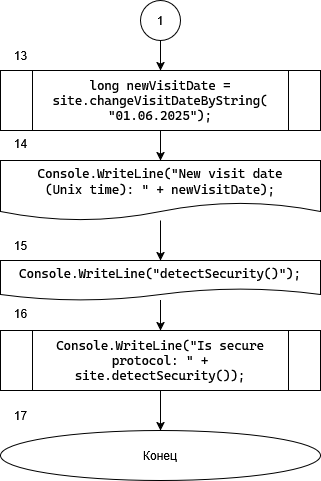


Рисунок 3.9 – Блок-схема алгоритма

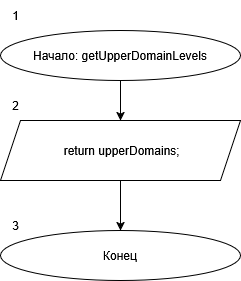
****

Рисунок 3.10 – Блок-схема алгоритма

****

Рисунок 3.11 – Блок-схема алгоритма

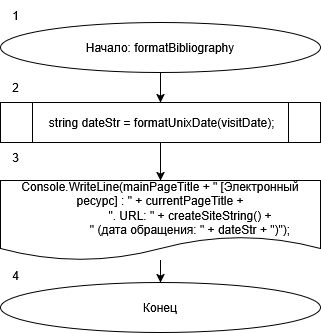
****

Рисунок 3.12 – Блок-схема алгоритма

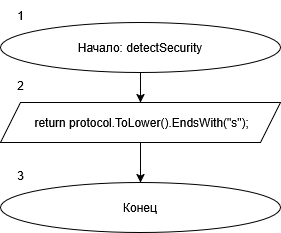
****

Рисунок 3.13 – Блок-схема алгоритма

**4. Подбор тестовых примеров**

Ожидаемый вывод программы :

ToString()

Ресторан MANSARD [https://mansard-moscow.ru] - IP: 185.165.123.36

createSiteString()

https://mansard-moscow.ru

getUpperDomainLevels()

mansard-moscow ru

formatBibliography()

Ресторан MANSARD [Электронный ресурс] : Главная. URL: https://mansard-moscow.ru (дата обращения: 23.05.2025)

changeVisitDateByString("01.06.2025")

New visit date (Unix time): 1748736000

detectSecurity()

Is secure protocol: True

Нажмите любую клавишу чтобы продолжить

**5. Листинг (код) программы**

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Net;

public class clsSite

{

private static short counter = 0; // Статический счётчик для уникального ID

private short id; // Уникальный идентификатор сайта

private string protocol; // Протокол (http, https и т.д.)

private string domainZone; // Доменная зона (например, "com", "ru")

private string hosting; // Хостинг (предпоследний уровень домена)

private string[] upperDomains; // Все уровни домена (массив)

private long ip; // IP-адрес в виде числа

private string mainPageTitle; // Заголовок главной страницы

private string currentPageTitle; // Заголовок текущей страницы

private long visitDate; // Дата посещения в Unix-времени (секунды)

private bool isSecure; // Флаг безопасности (https = true)

public clsSite(string address, string mainTitle, string currentTitle, string ipString, string dateString)

{

this.id = ++counter; // Присваиваем уникальный ID объекту

Uri uri = new Uri(address); // Парсим URL

protocol = uri.Scheme; // Сохраняем протокол

isSecure = detectSecurity(); // Определяем безопасность (https ли)

string host = uri.Host;

upperDomains = host.Split('.');// Разбиваем домен на части

domainZone = upperDomains[upperDomains.Length - 1]; // Получаем доменную зону

hosting = upperDomains.Length > 1 ? upperDomains[upperDomains.Length - 2] : upperDomains[0]; // Получаем "хостинг" — предпоследний уровень домена

this.mainPageTitle = mainTitle;

this.currentPageTitle = currentTitle;

ip = convertIpToLong(ipString); // Конвертируем IP из строки в число

this.visitDate = changeVisitDateByString(dateString); // Конвертируем дату посещения из строки в Unix-время

}

public string createSiteString()

{

// Формируем URL в виде "протокол://домен"

return protocol + "://" + string.Join(".", upperDomains);

}

public string[] getUpperDomainLevels()

{

// Возвращаем массив уровней домена

return upperDomains;

}

public void formatBibliography()

{

// Вывод формата библиографической ссылки с датой посещения

string dateStr = formatUnixDate(visitDate);

Console.WriteLine(mainPageTitle + " [Электронный ресурс] : " + currentPageTitle +

". URL: " + createSiteString() +

" (дата обращения: " + dateStr + ")");

}

public long changeVisitDateByString(string dateString)

{

// Конвертация даты в формате "дд.мм.гггг" в Unix-время (секунды с 1970-01-01)

DateTime dt;

if (DateTime.TryParseExact(dateString, "dd.MM.yyyy", CultureInfo.InvariantCulture,

DateTimeStyles.None, out dt))

{

TimeSpan ts = dt - new DateTime(1970, 1, 1);

return (long)ts.TotalSeconds;

}

return 0; // Возвращаем 0 при ошибке парсинга

}

public bool detectSecurity()

{

// Если протокол заканчивается на 's' (https), возвращаем true

return protocol.ToLower().EndsWith("s");

}

private long convertIpToLong(string ipStr)

{

// Конвертация IP из стандартного формата в число

IPAddress ipAddr = IPAddress.Parse(ipStr);

byte[] bytes = ipAddr.GetAddressBytes();

long value = ((long)bytes[0] << 24) + ((long)bytes[1] << 16) + ((long)bytes[2] << 8) + bytes[3];

return value;

}

private string formatUnixDate(long seconds)

{

// Форматирует Unix-время в строку "дд.мм.гггг"

DateTime dt = new DateTime(1970, 1, 1).AddSeconds(seconds);

return dt.ToString("dd.MM.yyyy");

}

public override string ToString()

{

// Возвращает строковое представление объекта: заголовок и IP-адрес

string ipStr = convertLongToIp(ip);

return mainPageTitle + " [" + createSiteString() + "] – IP: " + ipStr;

}

private string convertLongToIp(long ip)

{

// Конвертация числа обратно в IP-адрес в строковом формате

byte a = (byte)((ip >> 24) & 0xFF);

byte b = (byte)((ip >> 16) & 0xFF);

byte c = (byte)((ip >> 8) & 0xFF);

byte d = (byte)(ip & 0xFF);

return string.Format("{0}.{1}.{2}.{3}", a, b, c, d);

}

}

class Program

{

static void Main()

{// Установка цвета фона консоли белым

Console.BackgroundColor = ConsoleColor.White;

// Установка цвета текста консоли черным

Console.ForegroundColor = ConsoleColor.Black;

// Применяем изменения (очищая окно, чтобы фон изменился у всего окна)

Console.Clear();

// Создаём объект clsSite с информацией о сайте

clsSite site = new clsSite(

"https://mansard-moscow.ru", // URL сайта

"Ресторан MANSARD", // Заголовок главной страницы

"Главная", // Заголовок текущей страницы

"185.165.123.36", // IP-адрес сайта

"23.05.2025" // Дата посещения сайта в формате дд.мм.гггг

);

// Выводим информацию, вызвав метод ToString() класса clsSite

Console.WriteLine("ToString()");

Console.WriteLine(site.ToString());

// Выводим полную строку формирования URL

Console.WriteLine("createSiteString()");

Console.WriteLine(site.createSiteString());

// Получаем и выводим уровни домена (части доменного имени)

Console.WriteLine("getUpperDomainLevels()");

string[] levels = site.getUpperDomainLevels();

foreach (string level in levels)

Console.WriteLine(level);

// Форматируем и выводим библиографическую ссылку с информацией о сайте

Console.WriteLine("formatBibliography()");

site.formatBibliography();

// Конвертируем строку с датой в Unix-время и выводим результат

Console.WriteLine("changeVisitDateByString(\"01.06.2025\")");

long newVisitDate = site.changeVisitDateByString("01.06.2025");

Console.WriteLine("New visit date (Unix time): " + newVisitDate);

// Проверяем, использует ли сайт защищённый протокол (HTTPS)

Console.WriteLine("detectSecurity()");

Console.WriteLine("Is secure protocol: " + site.detectSecurity());

// Ожидаем нажатия клавиши, чтобы программа не закрылась сразу

Console.WriteLine("-------------------------------------------------------------");

Console.WriteLine("Нажмите любую клавишу чтобы продолжить");

Console.ReadKey();

}

}

**6. Расчёт тестовых примеров на ПК**

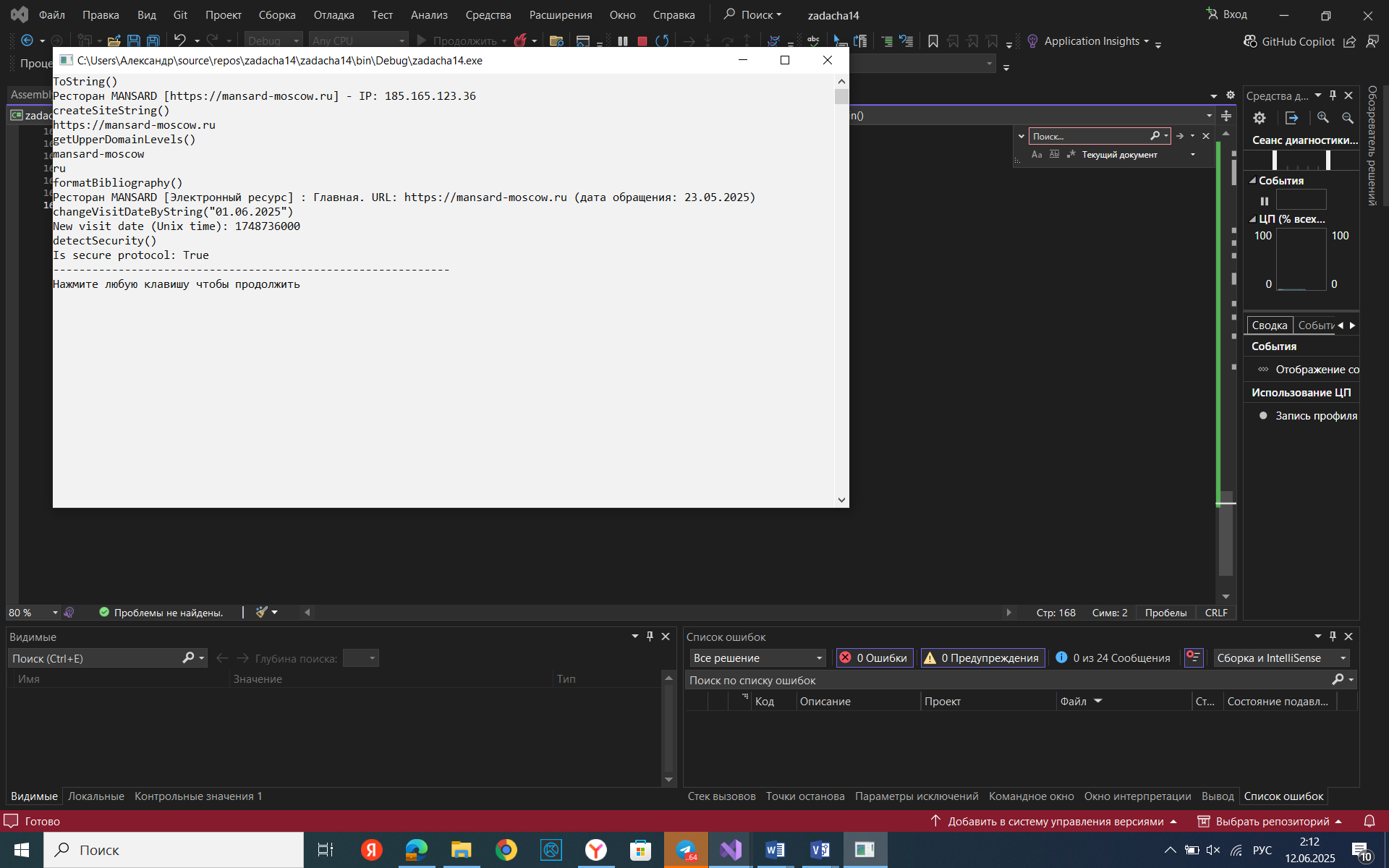


Рисунок 6.1 – Результат работы всей программы

**7. Вывод по работе**

В результате выполнения данной работы было составлено программное обеспечения для решения задания по индивидуальному варианту. Были получены навыки работы с классами и объектами на языке *Visual C#*.