# Лабораторная работа 7. Файлы и строки.

## Задание.

1. **Проект1**

Реализуйте консольное меню, которое выполняет следующие действия:

1 – устанавливает текущий диск/каталог

2 – выводит список всех каталогов в текущем (пронумерованный)

3 – выводит список всех файлов в текущем каталоге (пронумерованнный)

4 – выводит на экран содержимое указанного файла (по номеру)

5 – создает каталог в текущем

6 – удаляет каталог по номеру, если он пустой

7 – удаляет файлы с указанными номерами

8 – выводит список всех файлов с указанной датой создания (ищет в текущем каталоге и подкаталогах)

9 – выводит список всех текстовых файлов, в которых содержится указанный текст (ищет в текущем каталоге и подкаталогах)

0 – выход

Приложение должно быть универсальным – т.е. имена файлов и каталогов, текст, дата вводятся с клавиатуры.

Для демонстрации проекта подготовьте папку с вложенными папками и файлами, по пунктам 8 и 9 поиск должен возвращать несколько файлов.

class Program

{

static void CurrentDirDispl(DirectoryInfo curr)

{

Console.WriteLine($"- - -\nТекущая директория: \"{curr.FullName}\"\n- - -");

}

static void f1(ref DirectoryInfo d)

{// установить текущий диск/каталог

CurrentDirDispl(d);

Console.WriteLine("---\nУстановите текущий диск/каталог:");

string path = Console.ReadLine();

DirectoryInfo newDirect = new DirectoryInfo(path);

if (newDirect.Exists)

{

d = newDirect;

CurrentDirDispl(d);

}

else

Console.WriteLine("Вы ввели не верный путь!");

Console.ReadKey();

}

static void f2(DirectoryInfo d)

{// вывод списка всех каталогов в текущем (пронумерованный)

CurrentDirDispl(d);

Console.WriteLine("---\nвывод списка всех каталогов в текущем (пронумерованный):");

int i = 0;

foreach (DirectoryInfo directory in d.GetDirectories())

{

Console.WriteLine(i + ".) " + directory.Name);

i++;

}

Console.ReadKey();

}

static void f3(DirectoryInfo d)

{// вывод списка всех файлов в текущем каталоге (пронумерованнный)

CurrentDirDispl(d);

Console.WriteLine("---\nвывод списка всех файлов в текущем каталоге (пронумерованный):");

int i = 0;

foreach (FileInfo file in d.GetFiles())

{

Console.WriteLine(i + ".) " + file.Name);

i++;

}

Console.ReadKey();

}

static void f4(DirectoryInfo d)

{// вывод на экран содержимого указанного текстового файла (по номеру)

CurrentDirDispl(d);

f3(d);

Console.WriteLine("---\nВведите номер файла:");

int index;

if (!Int32.TryParse(Console.ReadLine(), out index))

{

Console.WriteLine("Индекс неверен!");

return;

}

FileInfo[] files = d.GetFiles();

if (files[index].Extension != ".txt")

{

Console.WriteLine("Это не текстовый файл!");

return;

}

StreamReader sr = new StreamReader(files[index].FullName);

string str = sr.ReadToEnd();

Console.WriteLine("содержимое файла:" + str);

sr.Close();

Console.ReadLine();

}

static void f5(DirectoryInfo d)

{// создание каталога в текущем

CurrentDirDispl(d);

Console.WriteLine("---\nсоздание каталога в текущем");

Console.WriteLine("Введите имя нового каталога:");

string newFolder = Console.ReadLine();

try

{

d.CreateSubdirectory(newFolder);

Console.WriteLine($"Каталог с именем \"{newFolder}\" успешно создан!");

}

catch(Exception e)

{

Console.WriteLine(e.Message);

}

Console.ReadKey();

}

static void f6(DirectoryInfo d)

{// удаление каталога по номеру, если он пустой

CurrentDirDispl(d);

f2(d);

Console.WriteLine("---\nудаление каталога по номеру, если он пустой");

Console.WriteLine("Введите имя номер каталога:");

int index;

if(!Int32.TryParse(Console.ReadLine(), out index))

{

Console.WriteLine("Индекс неверен!");

return;

}

DirectoryInfo dirToDel = d.GetDirectories()[index];

try

{

dirToDel.Delete();

Console.WriteLine($"Пустой каталог \"{dirToDel.Name}\" удален!");

}

catch(Exception e)

{

Console.WriteLine(e.Message);

}

Console.ReadKey();

}

static void f7(DirectoryInfo d)

{// удаление файлов с указанными номерами

CurrentDirDispl(d);

f3(d);

Console.WriteLine("---\nудаление файлов с указанными номерами");

Console.WriteLine("Введите номера первого и последнего файлов из диапазона для удаления:");

int index1, index2;

if(!Int32.TryParse(Console.ReadLine(),out index1) || !Int32.TryParse(Console.ReadLine(), out index2))

{

Console.WriteLine("неверные индексы!");

return;

}

FileInfo[] files = d.GetFiles();

if (index1 > index2 || index2 >= files.Length)

{

Console.WriteLine("неверные номера файлов");

return;

}

for (int i = index1; i <= index2; i++)

files[i].Delete();

Console.WriteLine(index2 - index1 + 1 + " файлов удалено");

Console.ReadKey();

f3(d);

}

static void f8(DirectoryInfo d)

{// вывод списка всех файлов с указанной датой создания (ищет в текущем каталоге и подкаталогах)

CurrentDirDispl(d);

Console.WriteLine("---\nвывод списка всех файлов с указанной датой создания (ищет в текущем каталоге и подкаталогах)");

Console.WriteLine("Введите дату для поиска в формате dd.mm.yyyy :");

string date = Console.ReadLine();

DateTime dt;

while (!DateTime.TryParseExact(date, "dd.MM.yyyy", null, System.Globalization.DateTimeStyles.None, out dt))

{

Console.WriteLine("Неверный формат даты, попробуйте еще раз!");

date = Console.ReadLine();

}

int i = 0;

foreach(FileInfo file in d.GetFiles("\*", SearchOption.AllDirectories))

{

if (file.CreationTime.Date == dt.Date)

{

Console.WriteLine(file.FullName + " " + file.CreationTime);

i++;

}

}

if(i==0)

Console.WriteLine("файлы с датой создания {0} не найдены",dt.ToString("dd.MM.yyyy"));

Console.ReadKey();

}

static void f9(DirectoryInfo d)

{// вывод списка всех текстовых файлов, в которых содержится указанный текст (ищет в текущем каталоге и подкаталогах)

CurrentDirDispl(d);

Console.WriteLine("---\nвывод списка всех текстовых файлов, в которых содержится указанный текст (ищет в текущем каталоге и подкаталогах))");

Console.WriteLine("Введите текст для поиска в файлах");

string str = Console.ReadLine();

FileInfo[] files = d.GetFiles("\*", SearchOption.AllDirectories);

int i = 0;

StreamReader sr;

foreach (FileInfo x in files)

{

if (x.Extension != ".txt") continue;

sr = new StreamReader(x.FullName);

string buf = sr.ReadToEnd();

sr.Close();

if (buf.Contains(str))

Console.WriteLine(++i + ")" + x.DirectoryName + "\\" + x.Name);

}

if (i == 0)

Console.WriteLine("файлы с текстом \"{0}\" не найдены", str);

Console.ReadKey();

}

static void Main()

{

DirectoryInfo dir = new DirectoryInfo(@"D:\"); // текущий каталог

try

{

while (true)

{

CurrentDirDispl(dir);

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("\* Главное меню \*");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("1 – установить текущий диск/каталог");

Console.WriteLine("2 – вывод списка всех каталогов в текущем");

Console.WriteLine("3 – вывод списка всех файлов в текущем каталоге");

Console.WriteLine("4 – вывод на экран содержимого указанного файла ");

Console.WriteLine("5 – создание каталога в текущем");

Console.WriteLine("6 – удаление каталога по номеру, если он пустой");

Console.WriteLine("7 – удаление файлов с указанными номерами");

Console.WriteLine("8 – вывод списка всех файлов с указанной датой создания");

Console.WriteLine("9 – вывод списка всех текстовых файлов, в которых текст");

Console.WriteLine("0 – выход");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

int choice = Convert.ToInt32(Console.ReadLine());

switch (choice)

{

case 1:

f1(ref dir);

break;

case 2:

f2(dir);

break;

case 3:

f3(dir);

break;

case 4:

f4(dir);

break;

case 5:

f5(dir);

break;

case 6:

f6(dir);

break;

case 7:

f7(dir);

break;

case 8:

f8(dir);

break;

case 9:

f9(dir);

break;

case 0: return;

}

}

}

catch (FileNotFoundException e)

{

Console.WriteLine("Неверное имя файла");

}

catch (DirectoryNotFoundException e)

{

Console.WriteLine("Неверное имя каталога");

}

catch (Exception e)

{

Console.WriteLine("Ошибка " + e.Message);

}

}

}

1. **Проект2**

Используйте любой класс с наследованием от другого класса или интерфейса из работы №2.

Добавьте статический метод, который запишет в текстовый файл всю информацию о типе вашего класса (рефлексия). Имя файла – параметр метода.

Добавьте экземплярный метод, который будет сохранять в бинарный файл всю информацию о текущем объекте. Имя файла – параметр метода.

Метод, который будет читать информацию из бинарного файла и возвращать готовый объект. Имя файла – параметр метода.

Добавьте методы, которые сериализуют и десериализуют объекты вашего класса. Имя файла – параметр метода.

// класс троллейбус наследуется от класса наземного транспорта из лабы №4

[Serializable]

public class Tbus : Land

{

public string Model { get; set; }

public Tbus(string model, int maxspeed, int passenger, string typeEngine) : base(maxspeed, passenger, typeEngine)

{

Model = model;

}

public override void Move() // реализация абстрактного метода базового класса Transport

{

Console.WriteLine("Я еду по автодороге");

}

public new void DisplayType() // скрываем метод родительского класса Land

{

Console.WriteLine($"Я являюсь троллейбусом {Model}");

}

public override string Info() // переопределение метода абстрактного родительского класса Bus

{

return $"Название модели:{Model};" + base.Info();

}

//Добавьте статический метод, который запишет в текстовый файл всю информацию о типе вашего класса (рефлексия). Имя файла – параметр метода.

public static void SaveClass(string filename)

{

Type t = typeof(Tbus);

StreamWriter f = new StreamWriter(filename);

f.WriteLine("Полное имя класса:" + t.FullName);

if (t.IsAbstract) f.WriteLine("Абстрактный класс");

if (t.IsClass) f.WriteLine("Обычный класс");

if (t.IsInterface) f.WriteLine("Интерфейс");

if (t.IsEnum) f.WriteLine("Перечисление");

if (t.IsSealed) f.WriteLine("Закрыт для наследования");

f.WriteLine("Базовый класс - " + t.BaseType);

FieldInfo[] fields = t.GetFields();

if (fields.Count() > 0)

f.WriteLine("\*\*\* Поля класса: \*\*\*");

foreach (var field in fields)

{

f.WriteLine(field);

}

PropertyInfo[] properties = t.GetProperties();

if (properties.Count() > 0)

f.WriteLine("\*\*\* Свойства класса: \*\*\*");

foreach (var property in properties)

{

f.WriteLine(property);

}

MethodInfo[] methods = t.GetMethods();

if (methods.Count() > 0)

f.WriteLine("\*\*\* Методы класса: \*\*\*");

foreach (var method in methods)

{

f.WriteLine(method);

}

f.Close();

}

// Добавьте экземплярный метод, который будет сохранять в бинарный файл всю информацию о текущем объекте. Имя файла – параметр метода.

public void SaveObject(string filename)

{

FileStream fs = new FileStream(filename, FileMode.Create, FileAccess.Write);

BinaryWriter bw = new BinaryWriter(fs);

bw.Write(this.Model);

bw.Write(this.MaxSpeed);

bw.Write(this.Seats);

bw.Write(this.EngineType);

fs.Close();

}

// Метод, который будет читать информацию из бинарного файла и возвращать готовый объект. Имя файла – параметр метода.

public static Tbus LoadObject(string filename)

{

FileStream fs = new FileStream(filename, FileMode.Open, FileAccess.Read);

BinaryReader br = new BinaryReader(fs);

string model = br.ReadString();

int speed = br.ReadInt32();

int seats = br.ReadInt32();

string engineType = br.ReadString();

fs.Close();

return new Tbus(model, speed, seats, engineType);

}

// Добавьте методы, которые сериализуют и десериализуют объекты вашего класса. Имя файла – параметр метода.

public void Serialize(string filename)

{

Stream fs = new FileStream(filename, FileMode.Create);

BinaryFormatter fmt = new BinaryFormatter();

fmt.Serialize(fs, this);

fs.Close();

}

public static Tbus Deserialize(string filename)

{

Stream fs = new FileStream(filename, FileMode.Open);

BinaryFormatter fmt = new BinaryFormatter();

Tbus tbus = (Tbus)fmt.Deserialize(fs);

fs.Close();

return tbus;

}

class Program

{

static void Main(string[] args)

{

Tbus tbus = new Tbus("Maz 54", 90, 56, "электро");

Console.WriteLine("Вывод инфо о троллейбусе:");

tbus.DisplayType();

tbus.Move();

Console.WriteLine(tbus.Info());

Tbus.SaveClass("Tbus.txt"); // сохранили в файл информацию о классе

Console.WriteLine("Сохранили информацию о классе");

tbus.SaveObject("tbus.bin"); // сохранили в файл информацию об объекте

Console.WriteLine("Сохранили информацию об объекте");

tbus.Model = "Liaz";

tbus.Seats = 100; // изменили объект

Console.WriteLine("Изменили параметры троллейбуса:");

Console.WriteLine(tbus.Info()); // вывели информацию о троллейбусе

tbus = Tbus.LoadObject("tbus.bin"); // прочитали объект

Console.WriteLine("Прочитали сохраненные значения экземпляра троллейбуса");

Console.WriteLine(tbus.Info()); // вывели информацию о троллейбусе

tbus.Serialize("tbus.dat");

Console.WriteLine("Сериализовали экземпляр троллейбуса");

tbus.Model = "Neoplan 3";

tbus.Seats = 30; // изменили объект

Console.WriteLine("Изменили параметры троллейбуса:");

Console.WriteLine(tbus.Info()); // вывели информацию о троллейбусе

tbus = Tbus.Deserialize("tbus.dat");

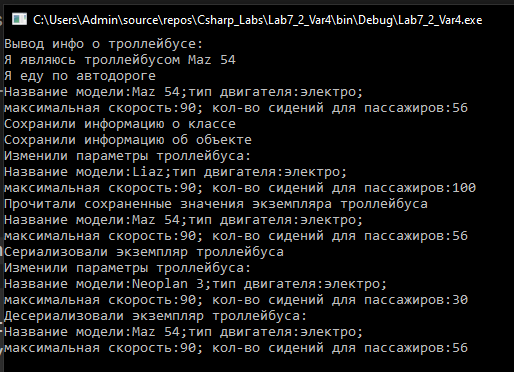
Console.WriteLine("Десериализовали экземпляр троллейбуса:");

Console.WriteLine(tbus.Info()); // вывели информацию о троллейбусе

Console.ReadKey();

}

}



1. **Проект3**

Для строки StringBuilder реализуйте следующие действия:

1 – ввод строки с клавиатуры (указывать размер)

2 – вывод строки

3 – после указанного символа каждый раз вставить \*

4 – заменить один символ на другой

5 – удалить все вхождения указанной подстроки

0 – выход

class Program

{

//Проект3

public static void f1(ref StringBuilder builder)

{

Console.WriteLine("Введите строку:");

builder = new StringBuilder(Console.ReadLine());

Console.WriteLine("Задайте емкость буфера:");

int capacity;

if (!Int32.TryParse(Console.ReadLine(), out capacity))

return;

builder.EnsureCapacity(capacity);

}

public static void f2(ref StringBuilder builder)

{

Console.WriteLine(builder + "\nемкость буфера:" + builder.Capacity);

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

}

public static void f3(ref StringBuilder b)

{

Console.WriteLine("Вставить символ \"\*\" после символа:");

char ch;

if(!char.TryParse(Console.ReadLine(),out ch))

{

Console.WriteLine("Неверный символ!");

return;

}

b.Replace(ch.ToString(), ch + "\*");

}

public static void f4(ref StringBuilder b)

{

Console.WriteLine("Заменить символ:");

char ch1;

if (!char.TryParse(Console.ReadLine(), out ch1))

{

Console.WriteLine("Неверный символ!");

return;

}

char ch2;

if (!char.TryParse(Console.ReadLine(), out ch2))

{

Console.WriteLine("Неверный символ!");

return;

}

b.Replace(ch1, ch2);

}

public static void f5(ref StringBuilder b)

{

Console.WriteLine("Удалить все вхождения подстроки:");

b.Replace(Console.ReadLine(),"");

}

static void Main(string[] args)

{

StringBuilder strBuilder = new StringBuilder(0);

while (true)

{

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

Console.WriteLine("| Меню |");

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

Console.WriteLine("1 – ввод строки с клавиатуры(указывать размер)");

Console.WriteLine("2 – вывод строки");

Console.WriteLine("3 – после указанного символа каждый раз вставить \*");

Console.WriteLine("4 – заменить один символ на другой");

Console.WriteLine("5 – удалить все вхождения указанной подстроки");

Console.WriteLine("0 - выход");

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

Console.WriteLine("текущее значение:");

f2(ref strBuilder);

int choise;

Console.WriteLine("Ваш выбор:");

if (!Int32.TryParse(Console.ReadLine(), out choise)) {

Console.WriteLine("Неверный выбор!");

continue;

}

switch (choise)

{

case 1: f1(ref strBuilder);break;

case 2: f2(ref strBuilder); break;

case 3: f3(ref strBuilder); break;

case 4: f4(ref strBuilder); break;

case 5: f5(ref strBuilder); break;

case 6: return;

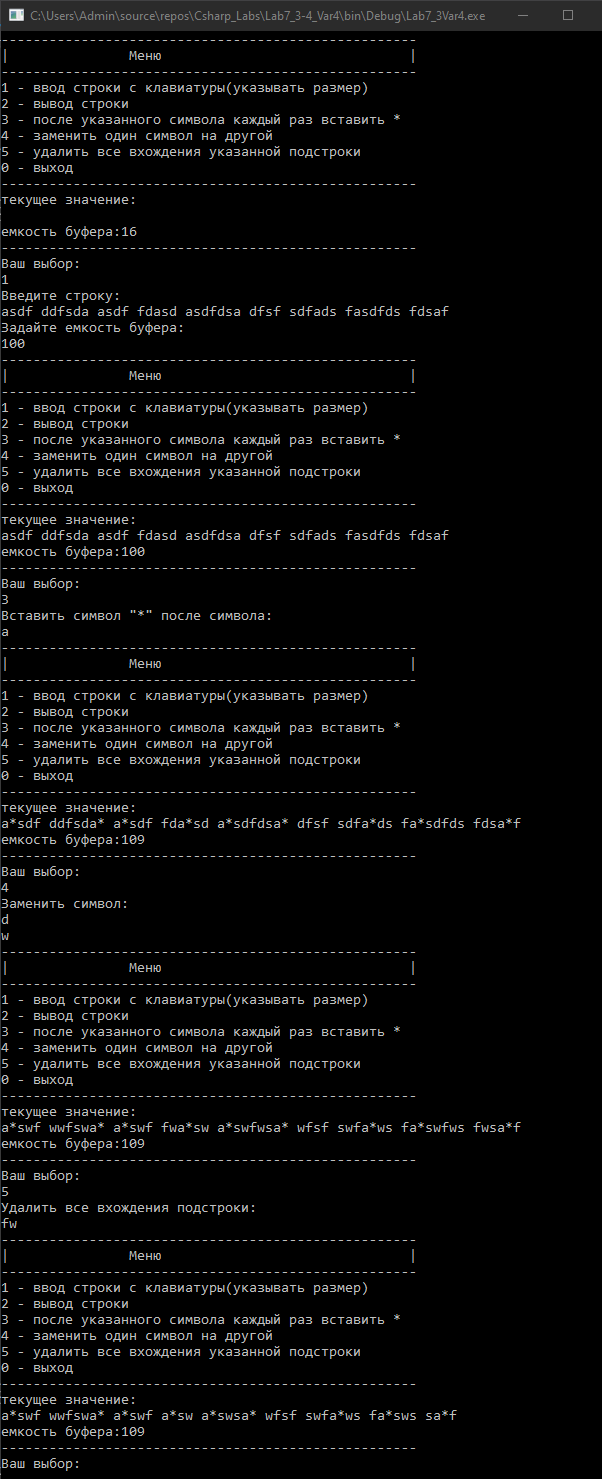
default:return;

}

}

}

}



1. **Проект4 (по вариантам)**

Напишите регулярное выражение и найдите все совпадения в блоке текста.

Вывести все найденные совпадение

1. Поиск телефонного номера +375(29)111-11-11
2. Поиск e-mail
3. Поиск ip 123.15.12.56 - корректно 467.8976.9876.8764 - нет
4. Поиск корректного времени 13:45:35 - корректно 99:66:77 – некорректно

class Program

{

static void Main(string[] args)

{

//4.Проект4(по вариантам)

//Напишите регулярное выражение и найдите все совпадения в блоке текста.

//Вывести все найденные совпадение. 4-вариант. Поиск корректного времени 13:45:35 - корректно 99:66:77 - некорректно

string data = "45:78:99sd dffds gfds12:12:12 ggggsd789:12:80 fsa23:59:59df25:59:59fds fdfsdf 22:60:45ds.kjhj 23:00:0024:00:00";

while (true)

{

Console.WriteLine("--- Поиск корректного времени в формате HH24:MM:SS ---");

Console.WriteLine("Блок текста:\n" + data);

Regex regex = new Regex(@"([01]\d|2[0-3]):[0-5]\d:[0-5]\d");

MatchCollection matchCollection = regex.Matches(data);

Console.WriteLine("\*\*\*\nВсе найденные совпадения корректного времени:");

foreach (Match m in matchCollection)

{

Console.WriteLine(m.Value);

}

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

Console.WriteLine("Введите строку для добавления в конец имеющегося блока текста (для выхода - 0):");

string newstring = Console.ReadLine();

if (newstring.Length == 1)

{

int quit;

if (Int32.TryParse(newstring, out quit))

if (quit == 0)

return;

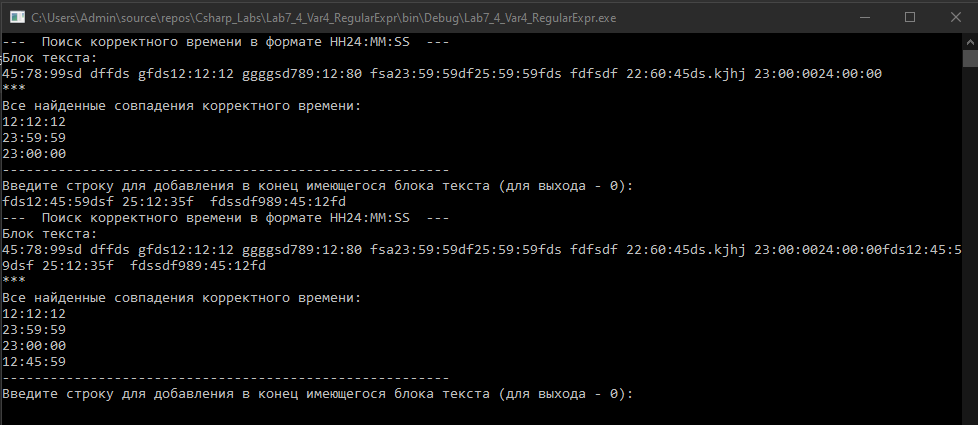
}

data += newstring;

}

}

}



1. Поиск комментариев HTML <!-- -->
2. Поиск корректной даты 12.11.2020 – корректно 39.16.2020 - некорректно
3. Проверка номера кредитной карты: 9999 9999 9999 9999