**`МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

по дисциплине «Объектно-ориентированное проектирование и программирование»

«Обработка строковых данных. Регулярные выражения»

Выполнил студент

группы ИТП– 21

*Кислов Я.В.*

Проверил преподаватель

*Карась О.В.*

Гомель 2025

Лабораторная работа №4

«Обработка строковых данных. Регулярные выражения»

**Цель работы:** Освоить методы обработки строковых данных в C#, включая поиск, разбиение и анализ текста. Изучить применение регулярных выражений для сложных паттернов поиска и валидации данных.

**Задание:**

1. Разработать приложения, решающие поставленную задачу (таблица 4.1 и таблица 4.

2) 2. Разработать не менее 15 модульных тестов для тестирования



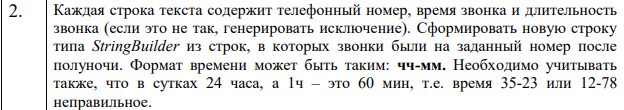


Рисунок 1 – Вариант заданий

**Ход работы:**

Для решения первой задачи был разработан статический метод который возвращает кортеж из количества уникальных символов и список их соответственно. Метод принимает строку, проверка на null и далее логика на проверку уникальных элементов. Код программы в приложении А. Результат выполнения на рисунке 2.

Для решения второй задачи был разработан статический метод true в том случае, если данные которые вводятся по условию задачи введены по формату. Были использованны регулярные выражения. Код программы в приложении А. Выполнения программы на рисунке 3.

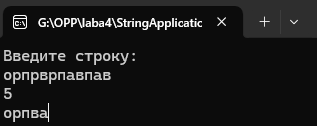


Рисунок 2 – Результат выполнения 1 задания

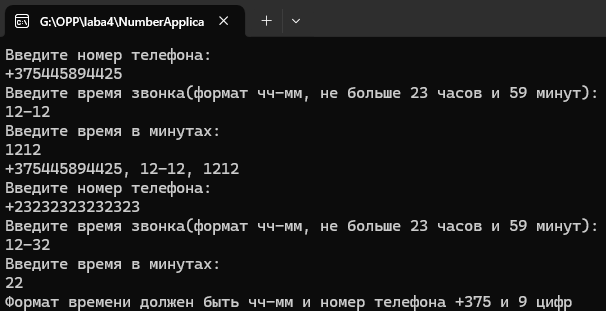


Рисунок 3 – Результат выполнения задания 2.

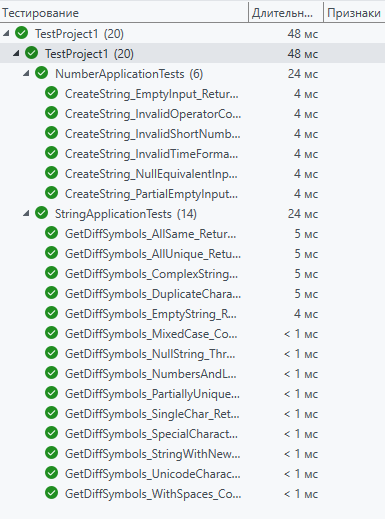


Рисунок 4 – Результат выполнения тестов

Вывод: В ходе выполнения работы были освоены базовые методы обработки строк в C#, включая разбиение строки на подстроки, проверку формата данных и подсчет символов. Также были применены принципы объектно-ориентированного программирования для структурирования кода.

**Приложение А**

**Текст программы**

Текст программы консоли

using System.Runtime.Intrinsics.Arm;

using System.Text;

using System.Text.RegularExpressions;

namespace NumberApplication

{

public class Program

{

static void Main(string[] args)

{

List<StringBuilder> list = new List<StringBuilder>();

var stringBuilder = new StringBuilder();

try

{

while (true)

{

if (CreateString(stringBuilder))

{

list.Add(stringBuilder);

foreach (var item in list)

{

Console.WriteLine(item);

}

}

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

public static bool CreateString(StringBuilder str)

{

string patternTime = @"^([0-9]|1[0-9]|2[0-3])-([0-5][0-9])$";

string patternNumber = @"^\+375(29|33|44|25)\d{7}$";

Console.WriteLine("Введите номер телефона: ");

string? number = Console.ReadLine();

Console.WriteLine("Введите время звонка(формат чч-мм, не больше 23 часов и 59 минут): ");

string? timeStamp = Console.ReadLine();

Console.WriteLine("Введите время в минутах: ");

string? timePeriod = Console.ReadLine();

bool isSuccses = false;

if (string.IsNullOrEmpty(number) || string.IsNullOrEmpty(timeStamp) || string.IsNullOrEmpty(timePeriod))

{

Console.WriteLine("Ошибка: все поля должны быть заполнены");

return false;

}

if (string.IsNullOrEmpty(number) || string.IsNullOrEmpty(timeStamp) || string.IsNullOrEmpty(timePeriod))

throw new ArgumentNullException();

if (Regex.IsMatch(timeStamp, patternTime) && Regex.IsMatch(number,patternNumber))

{

str.AppendFormat("{0}, {1}, {2}", number, timeStamp, timePeriod);

isSuccses = true;

}

else

Console.WriteLine("Формат времени должен быть чч-мм и номер телефона +375 и 9 цифр");

return isSuccses;

}

}

}

using System;

namespace StringApplication

{

public class Program

{

static void Main(string[] args)

{

try

{

Console.WriteLine("Введите строку: ");

string? str = Console.ReadLine();

var (count, symbols) = GetDiffSymbols(str);

Console.WriteLine(count);

foreach (char c in symbols)

Console.Write(c);

Console.ReadKey();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

public static (int count, List<char> symbols) GetDiffSymbols(string str)

{

if (str is null)

{

throw new Exception();

}

List<char> list = new List<char>();

if (str.Length == 0)

return (0, list);

list.Add(str[0]);

for (int i = 1; i < str.Length; i++)

{

for (int j = 0; j < list.Count; j++)

{

if (list[j] != str[i])

{

if (list.Exists(c => c == str[i]))

break;

list.Add(str[i]);

}

}

}

return (list.Count, list);

}

}

}