**`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 № 5

по дисциплине «Объектно-ориентированное проектирование и программирование»

«Парсинг текстовых данных»

Выполнил студент

группы ИТП– 21

*Кислов Я.В.*

Проверил преподаватель

*Карась О.В.*

Гомель 2025

Лабораторная работа №5

«Парсинг текстовых данных»

**Цель работы:** изучить применение класса stringbuilder в языке программирования C# для эффективного создания и редактирования строк, а также освоить методы и свойства, используемые при работе с динамическими строками.

**Задание:**

1. Создать объект класса StringBuilder на основе исходной строки.
2. Реализовать добавление новой подстроки в конец строки с помощью метода Append.
3. Выполнить вставку подстроки в заданную позицию исходной строки через метод Insert.
4. Удалить часть строки методом Remove.
5. Заменить выбранную подстроку внутри строки на другую с помощью метода Replace.
6. Определить длину итоговой строки и вывести её на экран.
7. Преобразовать объект StringBuilder в обычную строку методом ToString и вывести результат.



Рисунок 1 – Вариант заданий

**Ход работы:**

Для решения задачи был разработан класс для сортировки IP формата v4 из строки, которая в свою очередь вводил пользователь вручную. Код программы в приложении А. Выполнение программы показано на рисунке 2.

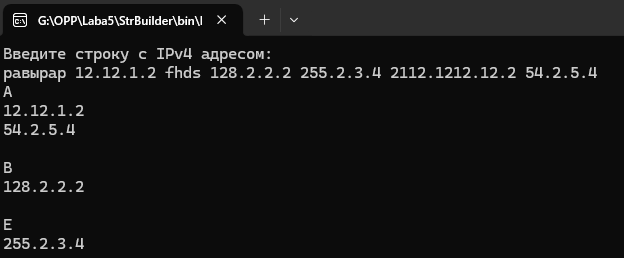


Рисунок 2 – Результат выполнения 1 задания

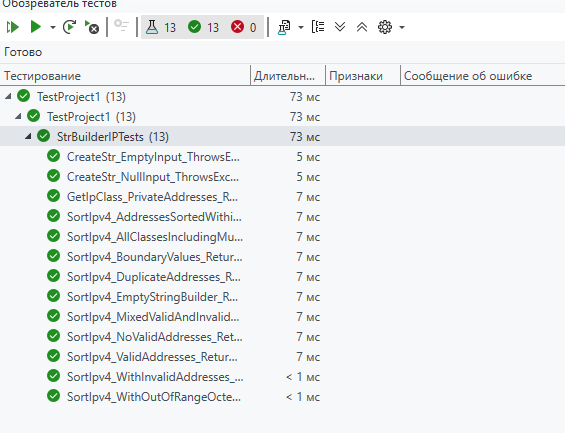


Рисунок 3 – Результат выполнения тестов

Вывод: применение класса stringbuilder позволило повысить эффективность операций со строками и сократить использование памяти. Выполненная лабораторная работа закрепила практические навыки программирования и показала преимущества динамических строк при работе с большими объемами текстовых данных.

**Приложение А**

**Текст программы**

Текст программы консоли

using System.Text;

using System.Text.RegularExpressions;

namespace StrBuilderIPv4

{

/// <summary>

/// Предоставляет методы для работы с IPv4 адресами в строковом представлении

/// </summary>

public class StrBuilderIP

{

/// <summary>

/// Сортирует IPv4 адреса по классам и возвращает словарь с отсортированными списками адресов для каждого класса

/// </summary>

/// <param name="str">StringBuilder содержащий текст с IPv4 адресами</param>

/// <returns>Словарь, где ключ - класс IP-адреса (A, B, C, D, E, U), значение - отсортированный список адресов этого класса</returns>

public Dictionary<char, List<string>> SortIpv4(StringBuilder str)

{

Dictionary<char, List<string>> result = new Dictionary<char, List<string>>

{

{'A', new List<string>()},

{'B', new List<string>()},

{'C', new List<string>()},

{'D', new List<string>()},

{'E', new List<string>() },

{'U', new List<string>()}

};

string Ipv4Pattern = @"^(\d{1,3}\.){3}\d{1,3}$";

string[] parts = str.ToString().Split();

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

{

if (Regex.IsMatch(parts[i], Ipv4Pattern) && IsValid(parts[i]))

{

result[GetIpClass(parts[i])].Add(parts[i]);

}

}

foreach (var kvp in result)

{

kvp.Value.Sort();

}

return result;

}

/// <summary>

/// Проверяет валидность IPv4 адреса

/// </summary>

/// <param name="str">Строка с IPv4 адресом для проверки</param>

/// <returns>True если все октеты адреса находятся в диапазоне от 0 до 255, иначе False</returns>

private bool IsValid(string str)

{

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

foreach (string part in parts)

{

int number = int.Parse(part);

if (number < 0 || number > 255)

return false;

}

return true;

}

/// <summary>

/// Создает StringBuilder из строки, введенной пользователем

/// </summary>

/// <returns>StringBuilder с введенной пользователем строкой</returns>

/// <exception cref="Exception">Выбрасывается когда введена пустая строка или null</exception>

public StringBuilder CreateStr()

{

Console.WriteLine("Введите строку с IPv4 адресом: ");

string? str = Console.ReadLine();

if (string.IsNullOrEmpty(str))

throw new Exception("Строка не может быть пустой или null");

return new StringBuilder(str);

}

/// <summary>

/// Определяет класс IPv4 адреса

/// </summary>

/// <param name="ip">IPv4 адрес для классификации</param>

/// <returns>

/// Символ, обозначающий класс адреса:

/// 'A' - класс A (1-126),

/// 'B' - класс B (128-191),

/// 'C' - класс C (192-223),

/// 'D' - класс D (224-239),

/// 'E' - класс E (240-255),

/// 'U' - неизвестный класс

/// </returns>

private char GetIpClass(string ip)

{

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

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

if (first >= 1 && first <= 126)

return 'A';

if (first >= 128 && first <= 191)

return 'B';

if (first >= 192 && first <= 223)

return 'C';

if (first >= 224 && first <= 239)

return 'D';

if (first >= 240 && first <= 255)

return 'E';

return 'U';

}

}

}

using System.Text;

using System.Text.RegularExpressions;

using StrBuilderIPv4;

namespace StrBuilder

{

internal class Program

{

static void Main(string[] args)

{

StrBuilderIP strb = new StrBuilderIP();

try

{

StringBuilder str = strb.CreateStr();

var list = strb.SortIpv4(str);

foreach (var s in list)

{

if (s.Value.Count > 0)

{

Console.WriteLine(s.Key);

foreach (var ip in s.Value)

System.Console.WriteLine(ip + " ");

System.Console.WriteLine();

}

}

}

catch

{

Console.WriteLine("String is null!");

}

Console.ReadLine();

}

}

}