|  |  |
| --- | --- |
| http://www.pl130.ru/doc/index/%D0%9B%D0%BE%D0%B3%D0%BE%D1%82%D0%B8%D0%BF22.jpg | Санкт-Петербургское государственное бюджетное  профессиональное образовательное учреждение  "Колледж электроники и приборостроения" |

отчет

**по практической работе №7**

**по дисциплине «Основы алгоритмизации и программирования»**

Тема: Работа со строками

|  |  |  |
| --- | --- | --- |
| Студент гр. 13 ИТ |  | Липинский К.С. |
| Преподаватель |  | Счастливцев А.К. |

Санкт-Петербург

2023

**Цель работы.**

Изучить работу со строками, изучить методы работы со строками.

**Выполнение работы.**

**Задание 1.**

namespace Practice;

using static Console;

class Program

{

static void Main()

{

Write("Enter some phrase: ");

string s = ReadLine()!;

Write("Enter two symbols you want to search for (delimiter is Space): ");

string sbuf = ReadLine()!;

WriteLine();

string[] search = sbuf.Split(' ');

char x = Convert.ToChar(search[0]); char y = Convert.ToChar(search[1]);

WriteLine($@"Количество вхождений заданных символов: {FindSym(s, x, y)}

Самый часто встречающийся символ: {string.Join(", ", FindOft(s, x, y))}

Есть ли два соседствующих символа: {IsNeighbor(s)}");

}

static int FindSym(string s, char x, char y)

{

int count = 0;

foreach (char i in s)

{

if (i == x || i == y)

{

count++;

}

}

return count;

}

static List<char> FindOft(string s, char x, char y)

{

List<char> oftSym = [];

int countX = 0, countY = 0;

foreach (char i in s)

{

if (i == x)

{

countX++;

}

else if (i == y)

{

countY++;

}

}

if (countX > countY)

{

oftSym.Add(x);

}

else if (countX < countY)

{

oftSym.Add(y);

}

else if (countX == countY)

{

oftSym.Add(x);

oftSym.Add(y);

}

return oftSym;

}

static string IsNeighbor(string s)

{

string isNeighborSym = "нет";

for (int i = 0; i < s.Length - 1; i++)

{

if (s[i] == s[i + 1])

{

isNeighborSym = "да";

break;

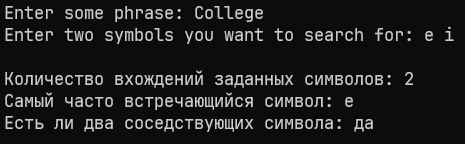
}

}

return isNeighborSym;

}

}



**Вопросы.**

1. string s = new String();
2. Compare, CompareTo, Concat, Join, Split, Insert, Remove, Substring, Replace, IndexOf, LastIndexOf, ToUpper, ToLower, Trim.
3. Особенность операции индексирования для строк в C# заключается в том, что строки являются неизменяемыми (immutable) объектами. Поэтому при доступе к символу строки по индексу создается новая строка, содержащая только этот символ. Изменение значения символа напрямую не разрешено и вызовет ошибку компиляции.
4. Сходства:

* Оба типа представляют последовательность символов.
* Оба типа позволяют обращаться к символам по индексу.
* Оба типа могут быть использованы для хранения и обработки текстовых данных.

Отличия:

* Строки (string) являются неизменяемыми (immutable), тогда как массивы типа char[] изменяемы.
* Строки имеют богатый набор методов и свойств, облегчающих работу с текстом, в то время как массивы типа char[] предоставляют только базовые операции.
* Строки могут быть интернированы (процесс кэширования строк для экономии памяти), в то время как массивы типа char[] не могут.

1. Операция присваивания для строк является присваиванием ссылки. Это означает, что при присваивании одной строки другой, обе переменные будут ссылаться на один и тот же объект в памяти. Если изменить значение одной переменной, это не повлияет на другую переменную.
2. Для сравнения строк используются операции сравнения (==, !=) и методы сравнения (Equals(), Compare()).
3. При конкатенации строк со значениями других типов, эти значения автоматически преобразуются в строки. Наиболее распространенным способом конкатенации строк и значений других типов является использование оператора + или метода Concat().