

Федеральное агентство связи

**Государственное бюджетное образовательное учреждение высшего
образования**

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Кафедра «МКиИТ»

дисциплина «СиАОД»

Отчет по Лабораторной работе №3

Подготовил студент

группы БВТ1901: Балдова Татьяна

Проверил: Мелехин А.

Москва 2020

Задание 1

Реализовать методы поиска подстроки в строке. Добавить возможность ввода строки и подстроки с клавиатуры. Предусмотреть возможность существования пробела. Реализовать возможность выбора опции чувствительности или нечувствительности к регистру.

Алгоритмы:

1.Кнута-Морриса-Пратта

2.Упрощенный Бойера-Мура

Метод поиска Кнута-Морриса-Пратта:

```
static int[] GetPrefix(string s)
{
    int[] result = new int[s.Length];
    result[0] = 0;
    int index = 0;

    for (int i = 1; i < s.Length; i++)
    {
        while (index >= 0 && s[index] != s[i]) { index--; }
        index++;
        result[i] = index;
    }

    return result;
}

static bool FindSubstring1(string pattern, string text)
{
    int[] pf = GetPrefix(pattern);
    int index = 0;

    for (int i = 0; i < text.Length; i++)
    {
        while (index > 0 && pattern[index] != text[i]) { index = pf[index - 1]; }
        if (pattern[index] == text[i]) index++;
        if (index == pattern.Length)
        {
            return true;
        }
    }

    return false;
}
```

Метод поиска упрощенный Бойера-Мура:

```
static bool FindSubstring2(string pattern, string MyText)
{
    int MyTextLen = MyText.Length;
    int patternLen = pattern.Length;

    if (patternLen > MyTextLen)
    {
        return false;
    }

    //символ-его удаление от конца
    //все кроме последнего
    Dictionary<char, int> offsetTable = new Dictionary<char, int>();
    for (int s = 0; s < patternLen - 1; s++)
    {
        //если дальше нет этой буквы то добавляем или пропускаем
        if (!(pattern.Substring(s + 1, pattern.Length - 2 - s).Contains(pattern[s])))
        {
            offsetTable.Add(pattern[s], patternLen - 1 - s);
        }
    }
    //для последнего символа
    if (!pattern.Substring(0, patternLen - 1).Contains(pattern[patternLen - 1]))
    {
        offsetTable.Add(pattern[patternLen - 1], patternLen);
    }

    int n = pattern.Length - 1;

    while (n < MyText.Length)
    {
        for (int i = patternLen - 1; i >= 0; i--, n--)
        {
            if (pattern[i] != MyText[n]) {

                if (offsetTable.ContainsKey(MyText[n]))
                {
                    n += offsetTable[MyText[n]];
                }
                else
                {
                    n += patternLen;
                }
                break;
            }
            if (i == 0) return true;
        }
    }
    return false;
}
```

Задание 2

Написать программу, определяющую, является ли данное расположение «решаемым», то есть можно ли из него за конечное число шагов перейти к правильному. Если это возможно, то необходимо найти хотя бы одно решение - последовательность движений, после которой числа будут расположены в правильном порядке.

Код программы:

```
static void Task2()
{
    for (int i = 1; i <= 12; i++)//первые три ряда
    {
        if (i % 4 != 0 && Matrix[i / 4, i % 4 - 1] != i)//если цифра
не на позиции и она не крайняя справа
        {
            RearrangeNumber(i, i / 4, i % 4 - 1);
        }
        else if (i % 4 == 0 && Matrix[i / 4 - 1, 3] != i)//если цифра
не на месте и она крайняя справа
        {
            if (i != 12)
            {
                RearrangeNumber(i, i / 4, 3);
                int[] Oindexs = CreateIndex0(0);
                int[] indexs = CreateIndex0(i);
                if (Oindexs[0] != indexs[0] + 1 && Oindexs[1] !=
indexs[1])
                {
                    Problems1(Oindexs[0], Oindexs[1], i);
                }
                RearrangeRight();
            }
            else
            {

```

```

int[] indexs = CreateIndex0(12);
int[] Oindexs = CreateIndex0(0);

if(indexs[0]==3 && indexs[1]==3 && Oindexs[0]==2 &&
Oindexs[1] == 3)//если один шаг до места
{
    SwapArr(0, Matrix[indexs[0], indexs[1]]);
    Swap(ref Oindexs[0], ref Oindexs[1], indexs[0],
indexs[1]);
}
else if (indexs[0] == 3 && indexs[1] == 3 &&
Oindexs[0]== indexs[0] && Oindexs[1] == indexs[1]-1)//если 12 в углу, а пустота слева
{
    StepsFor12(i);
}
else if (Matrix[2, 3] == 0 && Matrix[Oindexs[0] + 1,
Oindexs[1] - 1] == 12)
{
    Help(1, 0);
    Help(0, -1);
    Help(0, -1);
    Help(-1, 0);
    Help(0, 1);
    Help(0, 1);
    Help(1, 0);
    Help(0, -1);
    Help(-1, 0);
    Help(0, -1);
    Help(1, 0);
    Help(0, 1);
    Help(0, 1);
}
else
{
    if (Oindexs[1] > indexs[1])//если пустота справа
то доходим до 12 и пустота теперь слева
    {

```

```

        while (Oindexes[0] != indexes[0] || Oindexes[1]
!= indexes[1] + 1)
        {
            int[] NewODirection =
Direction2(Oindexes[0], Oindexes[1], indexes[0], indexes[1], i, i);
            SwapArr(0, Matrix[NewODirection[0],
NewODirection[1]]);
            Swap(ref Oindexes[0], ref Oindexes[1],
NewODirection[0], NewODirection[1]);
        }

        SwapArr(0, 12);
        Swap(ref Oindexes[0], ref Oindexes[1],
indexes[0], indexes[1]);
    }

```

```

встало
        if (Array.IndexOf(arr, 12) != 11) //если все еще не
        {
            if (Array.IndexOf(arr, 0) == 13)
            {
                Help(0, -1);
                Help(-1, 0);
                Help(0, 1);
                Help(0, 1);
                Help(1, 0);
                Help(0, 1);
                Help(-1, 0);
                Help(0, -1);
                Help(0, -1);
                Help(0, -1);
                Help(1, 0);
                Help(0, 1);
                Help(0, 1);
                Help(0, 1);
            }
        }

```

```
else if (Array.IndexOf(arr, 0) == 12)
```

```
{
```

```
    Help(-1, 0);
```

```
    Help(0, 1);
```

```
    Help(0, 1);
```

```
    Help(1, 0);
```

```
    Help(0, -1);
```

```
    Help(0, -1);
```

```
    Help(-1, 0);
```

```
    Help(0, 1);
```

```
    Help(0, 1);
```

```
    Help(1, 0);
```

```
    Help(0, 1);
```

```
    Help(-1, 0);
```

```
    Help(0, -1);
```

```
    Help(0, -1);
```

```
    Help(0, -1);
```

```
    Help(1, 0);
```

```
    Help(0, 1);
```

```
    Help(0, 1);
```

```
    Help(-1, 0);
```

```
    Help(0, -1);
```

```
    Help(0, -1);
```

```
    Help(1, 0);
```

```
    Help(0, 1);
```

```
    Help(0, 1);
```

```
    Help(0, 1);
```

```
}
```

```
else
```

```
{
```

```
    StepsFor12(i); //проверка на уже готовый
```

```
}
```

```
}
```

```
}
```

```
}
```

вариант

```

    }
}
if (arr[14] == 0)
{
    SwapArr(0, Matrix[3, 3]);
    int OindexX = 3, OindexY = 2;
    Swap(ref OindexX, ref OindexY, 3, 3);
}
// для расстановки последней строки

if (arr[15] != 0)
{
    while(arr[15] != 0)
    {
        int[] Oind = CreateIndex0(0);
        Help(0, 1);

    }
}
LastLine();
}

```

слева

```

static void StepsFor12(int i)//случай, когда 12 в углу, а пустота
{
    int[] Oindexs = CreateIndex0(0);

    if (!(Oindexs[0] == 3 && Oindexs[1] == 2))
    {
        while (Oindexs[0] != 3 || Oindexs[1] != 2)
        {
            int[] NewODirection = Direction2(Oindexs[0], Oindexs[1],
3, 2, i, i);

            SwapArr(0, Matrix[NewODirection[0], NewODirection[1]]);

```



```

        Swap(ref Oindexs[0], ref Oindexs[1], NewODirection[0],
NewODirection[1]);
    }
}

Help(0, -1);
Help(0, -1);
Help(-1, 0);
for (int z = 0; z < 3; z++) Help(0, 1);
Help(1, 0);
Help(0, -1);
Help(-1, 0);
Help(0, -1);
Help(0, -1);
Help(1, 0);
for (int z = 0; z < 3; z++) Help(0, 1);
}

static void RearrangeNumber(int i,int pos1,int pos2)
{
    while (Matrix[pos1,pos2] != i)//пока она не на позиции
    {
        //индекс цифры
        int index = Array.IndexOf(arr, i);

        int indexX = index / 4;
        int indexY = index % 4;

        //индекс пустоты
        int Oindex = Array.IndexOf(arr, 0);
        int OindexX = Oindex / 4;
        int OindexY = Oindex % 4;

        // новая цель цифры для ходьбы в направлении к своему месту
        int[] NewDirection = Direction(indexX, indexY, pos1, pos2, i);
    }
}

```

```

        //если это место перед ней и там пусто , то сразу переставить
        if (NewDirection[0] == pos1 && NewDirection[1] == pos2 &&
NewDirection[0] == OindexX && NewDirection[1] == OindexY)
        {
            SwapArr(i, Matrix[NewDirection[0], NewDirection[1]]);
            Swap(ref indexX, ref indexY, NewDirection[0],
NewDirection[1]);
        }
        else
        {
            //пока пустота не на нужной позиции, куда нужно встать
цифре
            while (OindexX != NewDirection[0] || OindexY !=
NewDirection[1])
            {
                if (Matrix[pos1, pos2] == i) return;
                //новое направление для пустоты
                int[] NewODirection = Direction2(OindexX, OindexY,
NewDirection[0], NewDirection[1], i, i);
                int[] NewO = CreateIndexO(0);
                OindexX = NewO[0];
                OindexY = NewO[1];
                SwapArr(0, Matrix[NewODirection[0],
NewODirection[1]]);
                Swap(ref OindexX, ref OindexY, NewODirection[0],
NewODirection[1]);
            }

            if (i % 4 == 0 && indexX == NewDirection[0] - 1 && indexY
== NewDirection[1] && OindexX == indexX + 1 && indexY == indexY) return;

            //обмен пустоты и цифры
            SwapArr(i, 0);
            Swap(ref indexX, ref indexY, OindexX, OindexY);

        }
    }
}

```

```

    }
    static void RearrangeRight()
    {
        Help(0, -1);
        Help(-1, 0);
        for (int k = 0; k < 2; k++)
        {
            Help(0, -1);
        }
        Help(-1, 0);
        for (int k = 0; k < 3; k++)
        {
            Help(0, 1);
        }
        Help(1, 0);
        Help(0, -1);
        Help(-1, 0);
        for (int k = 0; k < 2; k++)
        {
            Help(0, -1);
        }
        Help(1, 0);
    } //для перестановки крайних справа
    static void LastLine()
    {
        String MyLastLine = Convert.ToString(Matrix[3, 0]) + " " +
        Convert.ToString(Matrix[3, 1]) + " " + Convert.ToString(Matrix[3, 2]);

        //все вариации позиций
        switch (MyLastLine)
        {
            case "13 14 15":
                result += "\r\n" + "Игра окончена";
                break;
            case "13 15 14":
                result += "пришло к нерешаемому виду" + "\r\n";
            }
    }

```

```

        break;

        case "14 13 15"://??

            result += "пришло к нерешаемому виду" + "\r\n";

            break;

        case "14 15 13":

            result += "поменяем местами 0 и 12" + "\r\n" + "поменяем
местами 0 и 11" + "\r\n" + "поменяем местами 0 и 13" + "\r\n" + "поменяем местами 0 и
15" + "\r\n" + "поменяем местами 0 и 14" + "\r\n" + "поменяем местами 0 и 9" + "\r\n"
+ "поменяем местами 0 и 10" + "\r\n" + "поменяем местами 0 и 13" + "\r\n" + "поменяем
местами 0 и 11" + "\r\n" + "поменяем местами 0 и 12" + "\r\n" + "поменяем местами 0 и
15" + "\r\n" + "поменяем местами 0 и 14" + "\r\n" + "поменяем местами 0 и 13" +
"\r\n" + "поменяем местами 0 и 10" + "\r\n" + "поменяем местами 0 и 9" + "\r\n" +
"поменяем местами 0 и 13" + "\r\n" + "поменяем местами 0 и 14" + "\r\n" + "поменяем
местами 0 и 15" + "\r\n" + "Игра окончена";

            break;

        case "15 14 13"://??

            result += "пришло к нерешаемому виду" + "\r\n";

            break;

        case "15 13 14":

            result += "поменяем местами 0 и 12" + "\r\n" + "поменяем
местами 0 и 11" + "\r\n" + "поменяем местами 0 и 10" + "\r\n" + "поменяем местами 0 и
9" + "\r\n" + "поменяем местами 0 и 15" + "\r\n" + "поменяем местами 0 и 13" + "\r\n"
+ "поменяем местами 0 и 14" + "\r\n" + "поменяем местами 0 и 12" + "\r\n" + "поменяем
местами 0 и 11" + "\r\n" + "поменяем местами 0 и 10" + "\r\n" + "поменяем местами 0 и
9" + "\r\n" + "поменяем местами 0 и 15" + "\r\n" + "поменяем местами 0 и 13" + "\r\n"
+ "поменяем местами 0 и 14" + "\r\n" + "поменяем местами 0 и 15" + "\r\n" + "поменяем
местами 0 и 9" + "\r\n" + "поменяем местами 0 и 10" + "\r\n" + "поменяем местами 0 и
11" + "\r\n" + "поменяем местами 0 и 12" + "\r\n" + "поменяем местами 0 и 15" +
"\r\n" + "поменяем местами 0 и 14" + "\r\n" + "поменяем местами 0 и 13" + "\r\n" +
"поменяем местами 0 и 9" + "\r\n" + "поменяем местами 0 и 10" + "\r\n" + "поменяем
местами 0 и 11" + "\r\n" + "поменяем местами 0 и 12" + "\r\n" + "Игра окончена";

            break;

    }

}

static int[] CreateIndex0(int i)
{
    int Oindex = Array.IndexOf(arr, i);

    int OindexX = Oindex / 4;

    int OindexY = Oindex % 4;

    return new int[] { OindexX, OindexY };
}

```

```

static void Help(int n,int m)
{
    // просто для обмена при становлении крайних правых
    int[] index0 = CreateIndex0(0);
    int indexX = index0[0] + n;
    int IndexY = index0[1] + m;
    SwapArr(Matrix[indexX, IndexY ], 0);
    result += $"поменяли местами {Matrix[indexX, IndexY]} и
{Matrix[index0[0], index0[1]]}"+"\\r\\n";
    //Console.WriteLine($"поменяли местами {Matrix[indexX, IndexY]} и
{Matrix[index0[0], index0[1]]}" + "\\r\\n");
    int temp = Matrix[indexX, IndexY];
    Matrix[indexX, IndexY] = Matrix[index0[0], index0[1]];
    Matrix[index0[0], index0[1]] = temp;
    //printArray();
}

static int[] Direction(int indexX1, int indexY1,int indexX2,int
indexY2,int i)
{
    //новое направление для цифры
    if (indexX1 > indexX2 && (Matrix[indexX1 - 1, indexY1] > i ||
Matrix[indexX1 - 1, indexY1] == 0)) return new int[] { indexX1 - 1, indexY1 };
    if (indexY1 > indexY2 && (Matrix[indexX1, indexY1 - 1]>i ||
Matrix[indexX1, indexY1 - 1] == 0)) return new int[] { indexX1, indexY1 - 1 };
    if (indexY1 < indexY2 && (Matrix[indexX1, indexY1 + 1] >i ||
Matrix[indexX1, indexY1 + 1] == 0)) return new int[] { indexX1, indexY1 + 1 };
    if(Matrix[indexX1 + 1, indexY1] > i) return new int[] { indexX1 +
1, indexY1 };

    return new int[] { -1, -1 };
}

static int[] Direction2(int indexX1, int indexY1, int indexX2, int
indexY2,int num,int i)
{
    //новое направление для пустоты
    int[] result1 = Problems1(indexX1, indexY1, i);
    if (result1[0] != -1)
    {

```

```

        return result1;
    }
    if (indexY1 != indexY2 && indexX1 == indexX2)
    {
        if (indexY1 < indexY2 && Matrix[indexX1, indexY1 + 1] != num
&& Matrix[indexX1, indexY1 + 1] > i) return new int[] { indexX1, indexY1 + 1 };
        if (indexY1 > indexY2 && Matrix[indexX1, indexY1 - 1] != num
&& Matrix[indexX1, indexY1 - 1] > i) return new int[] { indexX1, indexY1 - 1 };
        if (indexX1 - 1 >= 0 && Matrix[indexX1-1, indexY1] > i) return
new int[] { indexX1 - 1, indexY1 };
        if(indexX1 + 1 < 4 && Matrix[indexX1+1, indexY1] > i) return new
int[] { indexX1 + 1, indexY1 };
    }
    else if (indexY1 != indexY2 && indexX1 != indexX2)
    {
        if (indexX1+1 < 4 && ((indexY1-1 >= 0 && Matrix[indexX1 + 1,
indexY1 - 1] == i ) || Matrix[indexX1 + 1, indexY1] == i))
        {
            if (indexY1 - 1 >= 0 && Matrix[indexX1, indexY1 - 1] > i)
return new int[] { indexX1, indexY1 - 1 };
            if (indexY1 + 1 < 4 && Matrix[indexX1, indexY1 + 1] > i)
return new int[] { indexX1, indexY1 + 1 };
        }
        if(Matrix[3, 1] == 0 && indexX1 - 1 >= 0 && indexY1 + 1 < 4 &&
Matrix[indexX1-1, indexY1] < i && Matrix[indexX1, indexY1+1] == i)
        {
            Help(0, -1);
            Help(-1, 0);
            Help(0, 1);
            Help(0, 1);
            Help(1, 0);
            Help(0, -1);
            Help(-1, 0);
            Help(0, -1);
            return new int[] { indexX1, indexY1 - 1 };
        }

        if (indexX1 > indexX2 && Matrix[indexX1 - 1, indexY1] != num
&& Matrix[indexX1 - 1, indexY1] > i) return new int[] { indexX1 - 1, indexY1 };

```

```

        if (indexX1 < indexX2 && Matrix[indexX1 + 1, indexY1] != num
&& Matrix[indexX1 + 1, indexY1] > i) return new int[] { indexX1 + 1, indexY1 };

        if (indexY1 + 1 < 4 && Matrix[indexX1, indexY1 + 1] > i)
return new int[] { indexX1, indexY1 + 1 };

        if (indexY1 - 1 >= 0 && Matrix[indexX1, indexY1 - 1] > i)
return new int[] { indexX1, indexY1 - 1 };

    }

    else if(indexY1 == indexY2 && indexX1 != indexX2)
    {

        if (indexX1 > indexX2 && Matrix[indexX1 - 1, indexY1] != num
&& Matrix[indexX1-1, indexY1] > i) return new int[] { indexX1 - 1, indexY1 };

        if(indexX1 < indexX2 && Matrix[indexX1 + 1, indexY1] != num &&
Matrix[indexX1+1, indexY1] > i ) return new int[] { indexX1 + 1, indexY1 };

        if (indexY1 + 1 < 4 && Matrix[indexX1, indexY1 + 1] > i)
return new int[] { indexX1, indexY1 + 1 };

        if (indexY1 - 1 >= 0 && Matrix[indexX1, indexY1 - 1] > i)
return new int[] { indexX1, indexY1 - 1 };

    }

    else
    {

        int indexNum = Array.IndexOf(arr, num);

        int indexXNum = indexNum / 4;

        if(indexX1==indexXNum || (indexXNum < indexX1 && indexXNum !=
0))

        {

            if (indexX1 > indexX2 && Matrix[indexX1 - 1, indexY1] !=
num && Matrix[indexX1-1, indexY1] > i) return new int[] { indexX1 - 1, indexY1 };

            if (indexX1 < indexX2 && Matrix[indexX1 + 1, indexY1] !=
num && Matrix[indexX1+1, indexY1] > i) return new int[] { indexX1 + 1, indexY1 };

        }

        if (indexY1 < indexY2 && Matrix[indexX1, indexY1 + 1] !=
num)

        {

            if (!(indexX1 == 3 && indexY1 - 1 == 3 &&
Matrix[indexX1 - 1, indexY1 - 1] == num))

```

```

        {
            if (Matrix[indexX1, indexY1 + 1] > i)
                return new int[] { indexX1, indexY1 + 1 };
        }
    }
    if (indexY1 > indexY2 && Matrix[indexX1, indexY1 - 1] !=
num)
    {
        if (!(indexX1 == 3 && indexY1 - 1 == 0 &&
Matrix[indexX1 - 1, indexY1 - 1] == num))
        {
            if(Matrix[indexX1, indexY1 - 1] > i)
                return new int[] { indexX1, indexY1 - 1 };
        }
    }
    else
    {
        if (indexY1 < indexY2 && Matrix[indexX1, indexY1 + 1] !=
num)
        {
            if (!(indexX1 == 3 && indexY1 - 1 == 3 &&
Matrix[indexX1 - 1, indexY1 - 1] == num))
            {
                if (Matrix[indexX1, indexY1 + 1] > i)
                    return new int[] { indexX1, indexY1 + 1 };
            }
        }
        if (indexY1 > indexY2 && Matrix[indexX1, indexY1 - 1] !=
num)
        {
            if (!(indexX1 == 3 && indexY1 - 1 == 0 &&
Matrix[indexX1 - 1, indexY1 - 1] == num))
            {
                if (Matrix[indexX1, indexY1 - 1] > i)
                    return new int[] { indexX1, indexY1 - 1 };
            }
        }
    }
}

```



```

        if (indexX1 > indexX2 && Matrix[indexX1 - 1, indexY1] !=
num && Matrix[indexX1 - 1, indexY1] > i) return new int[] { indexX1 - 1, indexY1 };

        if (indexX1 < indexX2 && Matrix[indexX1 + 1, indexY1] !=
num && Matrix[indexX1 + 1, indexY1] > i) return new int[] { indexX1 + 1, indexY1 };

    }

}

return new int[] { -1,-1};

}

static void SwapArr(int num1,int num2)
{
    int index1 = Array.IndexOf(arr, num1);
    int index2 = Array.IndexOf(arr, num2);
    arr[index1] = num2;
    arr[index2] = num1;

}

static void Swap( ref int I1,ref int J1,int I2,int J2)
{
    result += $"поменяли местами {Matrix[I1, J1]} и {Matrix[I2, J2]}
"+"\\r\\n";

    //Console.WriteLine($"поменяли местами {Matrix[I1, J1]} и
{Matrix[I2, J2]} " + "\\r\\n");
    int temp = Matrix[I1, J1];
    Matrix[I1, J1] = Matrix[I2, J2];
    Matrix[I2, J2] = temp;

    I1 = I2;
    J1 = J2;

    //printArray();

}

static int[] Problems1(int OindexX, int OindexY, int i)
{
    if ((OindexX + 1 < 4 && OindexY + 1 < 4 && Matrix[OindexX, OindexY
+ 1] == i && Matrix[OindexX - 1, OindexY] < i) || (OindexY + 1 < 4 && OindexX == 0 &&
Matrix[OindexX, OindexY + 1] == i ))//ситуация ,когда сверху все на месте, а справа
то, что нужно переставить,но можно пойти вниз

```

```

        {
            result += $"поменяли местами 0 и {Matrix[OindexX + 1,
OindexY]}" + "\r\n";

            result += $"поменяли местами 0 и {Matrix[OindexX + 1, OindexY
+ 1]}" + "\r\n";

            //Console.WriteLine($"поменяли местами 0 и {Matrix[OindexX +
1, OindexY]}");

            //Console.WriteLine($"поменяли местами 0 и {Matrix[OindexX +
1, OindexY + 1]}");

            int temp1 = Matrix[OindexX + 1, OindexY];
            int temp2 = Matrix[OindexX + 1, OindexY + 1];
            SwapArr(0, temp1);
            SwapArr(0, temp2);
            Matrix[OindexX + 1, OindexY + 1] = 0;
            Matrix[OindexX + 1, OindexY] = temp2;
            Matrix[OindexX, OindexY] = temp1;
            OindexX++; OindexY++;
            return new int[] { OindexX, OindexY + 1 };
        }

        else if ((OindexX - 1 >= 0 && OindexX + 1 < 4 && OindexY - 1 >= 0
&& Matrix[OindexX, OindexY - 1] == i && Matrix[OindexX - 1, OindexY] < i) || (OindexY
- 1 >= 0 && OindexX == 0 && Matrix[OindexX, OindexY - 1] == i))//ситуация ,когда
сверху все на месте, а слева то, что нужно переставить,но можно пойти вниз
        {
            result += $"поменяли местами 0 и {Matrix[OindexX + 1,
OindexY]}" + "\r\n";

            result += $"поменяли местами 0 и {Matrix[OindexX + 1, OindexY
- 1]}" + "\r\n";

            //Console.WriteLine($"поменяли местами 0 и {Matrix[OindexX +
1, OindexY]}");

            //Console.WriteLine($"поменяли местами 0 и {Matrix[OindexX +
1, OindexY - 1]}");

            int temp1 = Matrix[OindexX + 1, OindexY];
            int temp2 = Matrix[OindexX + 1, OindexY - 1];
            SwapArr(0, temp1);
            SwapArr(0, temp2);
            Matrix[OindexX + 1, OindexY - 1] = 0;
            Matrix[OindexX + 1, OindexY] = temp2;
            Matrix[OindexX, OindexY] = temp1;

```

```

        OindexX++; OindexY--;
        return new int[] { OindexX, OindexY - 1 };
    }
    else if(OindexX==3 && OindexY==3 && OindexX - 1 >= 0 &&
Matrix[OindexX-1,OindexY]==i)
    {
        if (i / 4 == 2)
        {
            result += $"поменяли местами 0 и {Matrix[OindexX, OindexY
- 1]}" + "\r\n";

            //Console.WriteLine($"поменяли местами 0 и
{Matrix[OindexX, OindexY - 1]}");
            int temp1 = Matrix[OindexX, OindexY - 1];
            SwapArr(0, temp1);
            Matrix[OindexX , OindexY - 1] = 0;
            Matrix[OindexX, OindexY] = temp1;
            OindexY--;
            return new int[] { OindexX - 1, OindexY };
        }
        else
        {
            result += $"поменяли местами 0 и {Matrix[OindexX, OindexY
- 1]}" + "\r\n";

            result += $"поменяли местами 0 и {Matrix[OindexX - 1,
OindexY - 1]}" + "\r\n";

            //Console.WriteLine($"поменяли местами 0 и
{Matrix[OindexX, OindexY - 1]}");
            //Console.WriteLine($"поменяли местами 0 и {Matrix[OindexX
- 1, OindexY - 1]}");

            int temp1 = Matrix[OindexX, OindexY - 1];
            int temp2 = Matrix[OindexX - 1, OindexY - 1];
            SwapArr(0, temp1);
            SwapArr(0, temp2);
            Matrix[OindexX - 1, OindexY - 1] = 0;
            Matrix[OindexX, OindexY - 1] = temp2;
            Matrix[OindexX, OindexY] = temp1;
            OindexX--; OindexY--;
            return new int[] { OindexX - 1, OindexY };
        }
    }
}

```

```

    }
}
else if (OindexX == 3 && OindexY == 0 && OindexX - 1 >= 0 &&
Matrix[OindexX - 1, OindexY] == i)
{
    if (i / 4 == 2 )
    {
        result += $"поменяли местами 0 и {Matrix[OindexX, OindexY
+ 1]}" + "\r\n";

        //Console.WriteLine($"поменяли местами 0 и
{Matrix[OindexX, OindexY + 1]}");
        int temp1 = Matrix[OindexX, OindexY + 1];
        SwapArr(0, temp1);
        Matrix[OindexX, OindexY + 1] = 0;
        Matrix[OindexX, OindexY] = temp1;
        OindexY++;
        return new int[] { OindexX - 1, OindexY };
    }
    else
    {
        result += $"поменяли местами 0 и {Matrix[OindexX, OindexY
+ 1]}" + "\r\n";

        result += $"поменяли местами 0 и {Matrix[OindexX - 1,
OindexY + 1]}" + "\r\n";

        //Console.WriteLine($"поменяли местами 0 и
{Matrix[OindexX, OindexY + 1]}");
        //Console.WriteLine($"поменяли местами 0 и {Matrix[OindexX
- 1, OindexY + 1]}");

        int temp1 = Matrix[OindexX, OindexY + 1];
        int temp2 = Matrix[OindexX - 1, OindexY + 1];
        SwapArr(0, temp1);
        SwapArr(0, temp2);
        Matrix[OindexX - 1, OindexY + 1] = 0;
        Matrix[OindexX, OindexY + 1] = temp2;
        Matrix[OindexX, OindexY] = temp1;
        OindexX--; OindexY++;
        return new int[] { OindexX - 1, OindexY };
    }
}

```

```

    }

    else if(Matrix[3,0]==0 && i==10 && Matrix[3, 1] == 10 && Matrix[2,
0] == 9)

    {

        Help(-1, 0);

        Help(0, 1);

        Help(0, 1);

        Help(1, 0);

        Help(0, -1);

        Help(-1, 0);

        Help(0, -1);

        Help(1, 0);

        Help(0, 1);

        Help(0, 1);

        Help(-1, 0);

        Help(0, -1);

        return new int[] { OindexX , OindexY +1 };

    }

    return new int[] { -1, -1 };

}

```

Результат работы программы:

```

C:\Users\xiaomi\Desktop\СИАОД\лабораторная работа №3\лабораторная работа №3\bin\Debug\лабораторная работа №3.exe
#1 Введите строку:
прогр рпшгвщлг гшрг9рп8РГИИЩД
Введите подстроку:
гшрг9рп8рг
Учитывать регистр?
нет
Учитывать пробел?
нет
Содержится ли подстрока в строке? (алгоритм Кнута-Морриса-Пратта): True
Содержится ли подстрока в строке? (алгоритм Упрощенный Бойера-Мура): True
#2 Введите массив из 16 элементов от 0 до 15:
13 14 1 4 2 10 5 3 8 6 0 15 9 12 11 7
Исходная комбинация
13 14 1 4
2 10 5 3
8 6 0 15
9 12 11 7
Последовательность ходов:
поменяли местами 0 и 5
поменяли местами 0 и 3
поменяли местами 0 и 4
поменяли местами 0 и 4
поменяли местами 0 и 3
поменяли местами 0 и 10
поменяли местами 0 и 14
поменяли местами 1 и 0
поменяли местами 0 и 10
поменяли местами 0 и 14
поменяли местами 0 и 2
поменяли местами 0 и 13
поменяли местами 1 и 0
поменяли местами 2 и 0
поменяли местами 0 и 14
поменяли местами 0 и 10
поменяли местами 0 и 4
поменяли местами 3 и 0
поменяли местами 0 и 10
поменяли местами 0 и 4
поменяли местами 3 и 0
поменяли местами 0 и 10

```

```
C:\Users\xiaomi\Desktop\СИАОД\Лабораторная работа №3\Лабораторная работа №3\bin\Debug\Лабораторная работа №3.exe
поменяли местами 10 и 0
поменяли местами 4 и 0
поменяли местами 14 и 0
поменяли местами 10 и 0
поменяли местами 3 и 0
поменяли местами 2 и 0
поменяли местами 1 и 0
поменяли местами 0 и 13
поменяли местами 0 и 10
поменяли местами 5 и 0
поменяли местами 0 и 15
поменяли местами 0 и 14
поменяли местами 0 и 14
поменяли местами 0 и 15
поменяли местами 0 и 6
поменяли местами 0 и 10
поменяли местами 5 и 0
поменяли местами 0 и 6
поменяли местами 0 и 10
поменяли местами 0 и 8
поменяли местами 0 и 13
поменяли местами 5 и 0
поменяли местами 6 и 0
поменяли местами 0 и 10
поменяли местами 0 и 15
поменяли местами 7 и 0
поменяли местами 0 и 11
поменяли местами 0 и 15
поменяли местами 0 и 10
поменяли местами 0 и 14
поменяли местами 7 и 0
поменяли местами 0 и 10
поменяли местами 0 и 14
поменяли местами 7 и 0
поменяли местами 0 и 10
поменяли местами 0 и 14
поменяли местами 8 и 0
поменяли местами 0 и 12
поменяли местами 0 и 15
поменяли местами 0 и 11
поменяли местами 0 и 14
```

```
C:\Users\xiaomi\Desktop\СИАОД\Лабораторная работа №3\Лабораторная работа №3\bin\Debug\Лабораторная работа №3.exe
поменяли местами 10 и 0
поменяли местами 8 и 0
поменяли местами 12 и 0
поменяли местами 10 и 0
поменяли местами 7 и 0
поменяли местами 6 и 0
поменяли местами 5 и 0
поменяли местами 9 и 0
поменяли местами 0 и 15
поменяли местами 0 и 13
поменяли местами 10 и 0
поменяли местами 11 и 0
поменяли местами 0 и 14
поменяем местами 0 и 12
поменяем местами 0 и 11
поменяем местами 0 и 10
поменяем местами 0 и 9
поменяем местами 0 и 15
поменяем местами 0 и 13
поменяем местами 0 и 14
поменяем местами 0 и 12
поменяем местами 0 и 11
поменяем местами 0 и 10
поменяем местами 0 и 9
поменяем местами 0 и 15
поменяем местами 0 и 13
поменяем местами 0 и 14
поменяем местами 0 и 15
поменяем местами 0 и 9
поменяем местами 0 и 10
поменяем местами 0 и 11
поменяем местами 0 и 12
поменяем местами 0 и 15
поменяем местами 0 и 14
поменяем местами 0 и 13
поменяем местами 0 и 9
поменяем местами 0 и 10
поменяем местами 0 и 11
поменяем местами 0 и 12
Игра окончена
```

Вывод: реализовала методы поиска подстроки в строке. Добавила возможность ввода строки и подстроки с клавиатуры. Предусмотрела возможность существования пробела. Реализовала возможность выбора опции чувствительности или нечувствительности к регистру. Написала программу, определяющую, является ли данное расположение «решаемым», нашла хотя бы одно решение - последовательность движений, после которой числа будут расположены в правильном порядке