

Федеральное агентство связи
ордена Трудового Красного Знамени
федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра МКиТ

Курсовая работа

по дисциплине «СиАОД»

Выполнил: студент

Группы БСТ1902

Козлов М. С.

Вариант №7

Москва 2021

Оглавление

Задача «Треугольник с максимальным периметром»	4
Описание	4
Код программы.....	4
Результат	4
Задача «Максимальное число»	5
Описание	5
Код программы.....	5
Результат	5
Задача «Сортировка диагоналей в матрице»	5
Описание	5
Код программы.....	5
Результат	6
Задача «Объединение отрезков»	7
Описание	7
Код программы.....	7
Результат	8
Задача «Стопки монет»	8
Описание	8
Код программы.....	9
Результат	9
Задача «Шарики и стрелы»	9
Описание	9
Код программы.....	10
Результат	10

Задача №1 со строками.....	11
Описание	11
Код программы.....	11
Результат	11
Задача №2 со строками.....	11
Описание	11
Код программы.....	11
Результат	12
Задача №3 со строками.....	12
Описание	12
Код программы.....	12
Результат	13
Вывод.....	13

Задача «Треугольник с максимальным периметром»

Описание

Массив A состоит из целых положительных чисел - длин отрезков. Составьте из трех отрезков такой треугольник, чтобы его периметр был максимально возможным. Если невозможно составить треугольник с положительной площадью - функция возвращает 0.

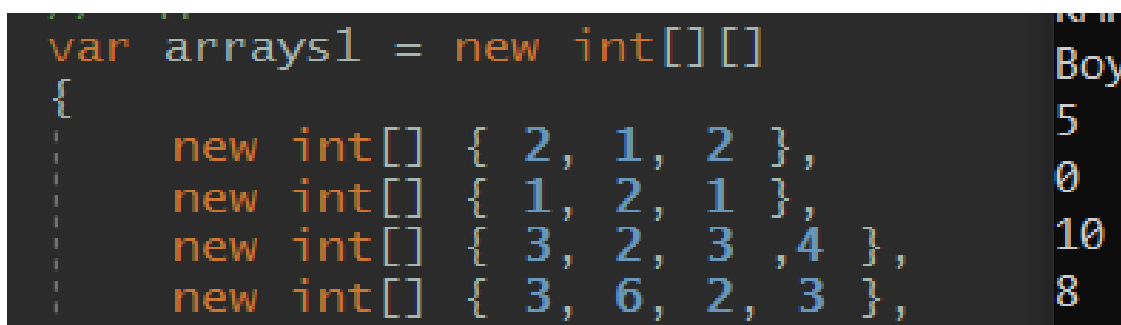
Код программы

```
public static int MaxPerimeter(int[] digits)
{
    int max = 0;
    Array.Sort(digits);
    Array.Reverse(digits);

    for (int i = 0; i < digits.Length - 2; i++)
    {
        if (digits[i] < digits[i + 1] + digits[i + 2])
        {
            max = Math.Max(max, digits[i] + digits[i + 1] + digits[i +
2]);
            break;
        }
    }

    return max;
}
```

Результат



```
var arrays1 = new int[][]
{
    new int[] { 2, 1, 2 },
    new int[] { 1, 2, 1 },
    new int[] { 3, 2, 3, 4 },
    new int[] { 3, 6, 2, 3 },
}
```

Triangle Sides	Perimeter
{ 2, 1, 2 }	5
{ 1, 2, 1 }	0
{ 3, 2, 3, 4 }	10
{ 3, 6, 2, 3 }	8

Рисунок 1 – Результат работы программы

Задача «Максимальное число»

Описание

Дан массив неотрицательных целых чисел `pums`. Расположите их в таком порядке, чтобы вместе они образовали максимально возможное число.

Код программы

```
public static string MaxDigit(int[] digits)
{
    var arr = digits.Select(x => x.ToString()).ToArray();
    for (var i = 1; i < arr.Length; i++)
        for (var j = 0; j < arr.Length - i; j++)
            if (Compare(arr[j], arr[j + 1]) < 0)
                Swap(ref arr[j], ref arr[j + 1]);

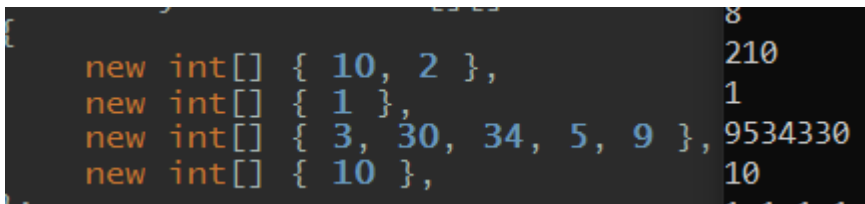
    var result = string.Empty;
    foreach (var str in arr) result += str;

    return result;
}

private static void Swap<T>(ref T x, ref T y)
{
    var temp = x;
    x = y;
    y = temp;
}

private static int Compare(string s1, string s2)
{
    while (s1.Length != s2.Length)
    {
        if (s1.Length > s2.Length) s2 += "0";
        else s1 += "0";
    }
    return s1.CompareTo(s2);
}
```

Результат



The screenshot shows a code editor with the following content:

```
[
    new int[] { 10, 2 },
    new int[] { 1 },
    new int[] { 3, 30, 34, 5, 9 },
    new int[] { 10 },
    ]
```

To the right of the code, the corresponding string outputs are displayed:

```
8
210
1
9534330
10
1 1 1 1
```

Рисунок 2 – Результат работы программы

Задача «Сортировка диагоналей в матрице»

Описание

Дана матрица `mat` размером $m * n$, значения - целочисленные. Напишите функцию, сортирующую каждую диагональ матрицы по возрастанию и возвращающую получившуюся матрицу.

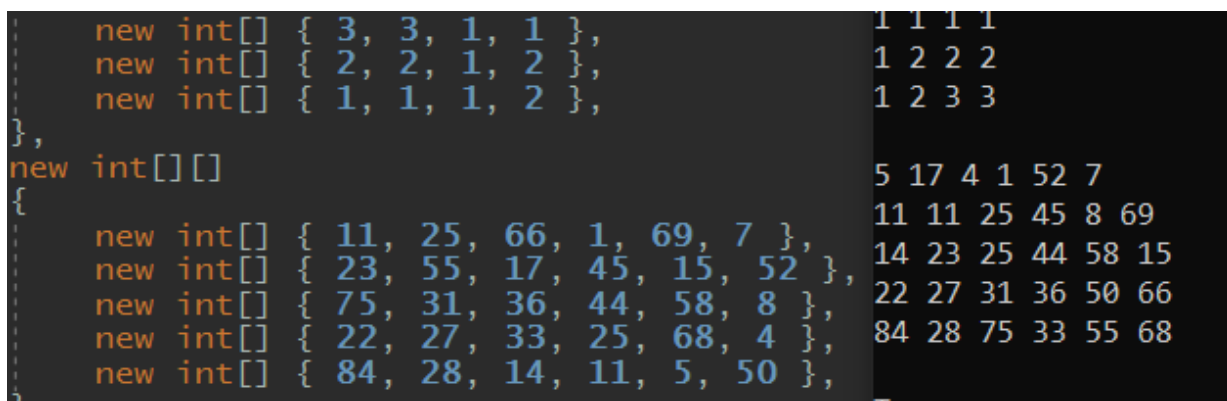
Код программы

```
public static void SortMatrixDiagonals(int[][] matrix)
{
    var m = matrix.Length;
    var n = matrix[0].Length;
    var c = 1;

    while (c != m)
    {
        Sorting(m, n, matrix);
        c++;
    }
}

private static void Sorting(int m, int n, int[][] matrix)
{
    for (var i = 0; i < m; i++)
    {
        for (var j = 0; j < n; j++)
        {
            if (i + 1 < m && j + 1 < n && matrix[i + 1][j + 1] <
matrix[i][j])
            {
                var temp = matrix[i + 1][j + 1];
                matrix[i + 1][j + 1] = matrix[i][j];
                matrix[i][j] = temp;
            }
        }
    }
}
```

Результат



```
new int[] { 3, 3, 1, 1 },
new int[] { 2, 2, 1, 2 },
new int[] { 1, 1, 1, 2 },
},
new int[][]
{
    new int[] { 11, 25, 66, 1, 69, 7 },
    new int[] { 23, 55, 17, 45, 15, 52 },
    new int[] { 75, 31, 36, 44, 58, 8 },
    new int[] { 22, 27, 33, 25, 68, 4 },
    new int[] { 84, 28, 14, 11, 5, 50 },
}
```

1	1	1	1		
1	2	2	2		
1	2	3	3		
5	17	4	1	52	7
11	11	25	45	8	69
14	23	25	44	58	15
22	27	31	36	50	66
84	28	75	33	55	68

Рисунок 3 – Результат работы програм

Задача «Объединение отрезков»

Описание

Дан массив отрезков `intervals`, в котором `intervals[i] = [start i, end i]`, некоторые отрезки могут пересекаться. Напишите функцию, которая объединяет все пересекающиеся отрезки в один и возвращает новый массив непересекающихся отрезков.

Код программы

```
public static int[][] Merge(int[][] intervals)
{
    if (intervals == null || intervals.Length == 0)
        return new int[][] { };

    List<int[]> result = new List<int[]>();

    Array.Sort(intervals, (x, y) => x[0].CompareTo(y[0]));

    int s = intervals[0][0],
        e = intervals[0][1];

    for (int i = 1; i < intervals.Length; i++)
        if (e < intervals[i][0])
        {
            result.Add(new int[] { s, e });

            s = intervals[i][0];
            e = intervals[i][1];
        }
        else
            e = Math.Max(e, intervals[i][1]);

    result.Add(new int[] { s, e });

    return result.ToArray();
}
```

Результат

```
new int[][] { new int[] { 1, 3 }, new int[] { 2, 6 }, new int[] { 8, 10 }, new int[] { 15, 18 },
new int[] { 1, 4 }, new int[] { 4, 5 } }, [ ,1 ,6 ], [ ,8 ,10 ], [ ,15 ,18 ],
[ ,1 ,5 ],
```

Рисунок 4 – Результат работы программы

Задача «Стопки монет»

Описание

На столе стоят $3n$ стопок монет. Вы и ваши друзья Алиса и Боб забираете стопки монет

по следующему алгоритму:

1. Вы выбираете 3 стопки монет из оставшихся на столе.
2. Алиса забирает себе стопку с максимальным количеством монет.
3. Вы забираете одну из двух оставшихся стопок

4. Боб забирает последнюю стопку.

5. Если еще остались стопки, то действия повторяются с первого шага.

Дан массив целых положительных чисел `piles`. Напишите функцию, возвращающую максимальное число монет, которое вы можете получить.

Код программы

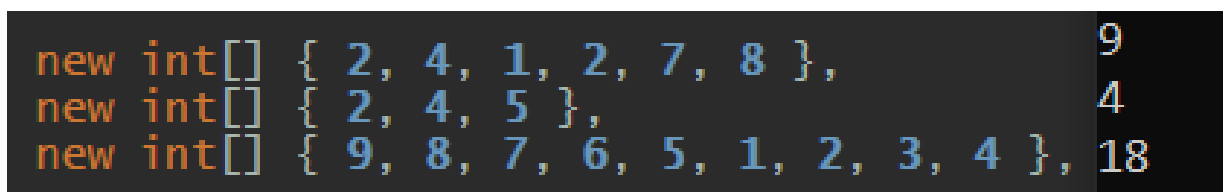
```
public static int GetMax(int[] piles)
{
    int maxSum = 0;

    Array.Sort(piles);

    for (int i = piles.Length / 3; i < piles.Length; i += 2)
    {
        maxSum += piles[i];
    }

    return maxSum;
}
```

Результат



```
new int[] { 2, 4, 1, 2, 7, 8 }, 9
new int[] { 2, 4, 5 }, 4
new int[] { 9, 8, 7, 6, 5, 1, 2, 3, 4 }, 18
```

Рисунок 5 – Результат работы программы

Задача «Шарики и стрелы»

Описание

Некоторые сферические шарики распределены по двумерному пространству. Для каждого шарика даны x -координаты начала и конца его горизонтального диаметра. Так как пространство двумерно, то y -координаты не имеют значения в данной задаче.

Координата x `start` всегда меньше x `end`. Стрелу можно выстрелить строго вертикально (вдоль y -оси) из разных точек x -оси. Шарик с

координатами x start и x end уничтожается стрелой, если она была выпущена из такой позиции x , что x start $\leq x \leq x$ end. Когда стрела выпущена, она летит в пространстве бесконечное время (уничтожая все шарики на пути).

Дан массив points, где points[i] = [x start, x end]. Напишите функцию, возвращающую минимальное количество стрел, которые нужно выпустить, чтобы уничтожить все шарики.

Код программы

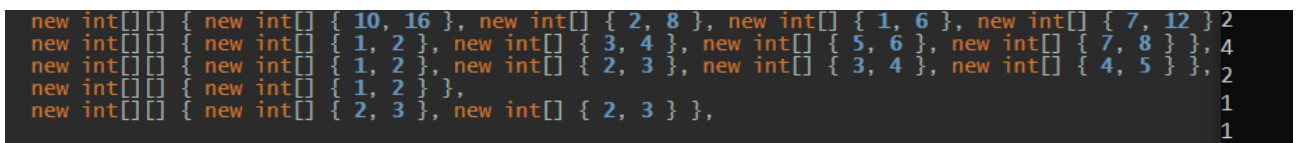
```
public static int FindMinArrowShots(int[][] points)
{
    if (points == null || points.Length == 0)
    {
        return 0;
    }

    Array.Sort(points, (x, y) => x[1].CompareTo(y[1]));

    int end = points[0][1], count = 1;
    for (int i = 1; i < points.Length; i++)
    {
        if (!(points[i][0] <= end))
        {
            end = points[i][1];
            count++;
        }
    }

    return count;
}
```

Результат



```
new int[][] { new int[] { 10, 16 }, new int[] { 2, 8 }, new int[] { 1, 6 }, new int[] { 7, 12 } } 2
new int[][] { new int[] { 1, 2 }, new int[] { 3, 4 }, new int[] { 5, 6 }, new int[] { 7, 8 } }, 4
new int[][] { new int[] { 1, 2 }, new int[] { 2, 3 }, new int[] { 3, 4 }, new int[] { 4, 5 } }, 2
new int[][] { new int[] { 1, 2 } }, 1
new int[][] { new int[] { 2, 3 }, new int[] { 2, 3 } }, 1
```

Рисунок 6 – Результат работы программы

Задача №1 со строками

Описание

Даны две строки: s1 и s2 с одинаковым размером, проверьте, может ли некоторая перестановка строки s1 “победить” некоторую перестановку строки s2 или наоборот.

Строка x может “победить” строку y (обе имеют размер n), если $x[i] \geq y[i]$ (в алфавитном порядке) для всех i от 0 до n-1.

Код программы

```
public static bool StringBattle(string s1, string s2)
{
    if (s1.Length != s2.Length)
    {
        throw new ArgumentException();
    }

    bool result = false;

    ShowAllCombinations(s2.ToCharArray(), s1, ref result);

    return result;
}

private static bool CheckBattle(string s1, string s2)
{
    for (int i = 0; i < s1.Length; i++)
    {
        if (s2[i] < s1[i])
        {
            return false;
        }
    }

    return true;
}

private static void ShowAllCombinations
    (IList<char> arr, string s1, ref bool result, string current = "")
{
    if (result) return;

    if (arr.Count == 0)
    {
        result = CheckBattle(s1, current);

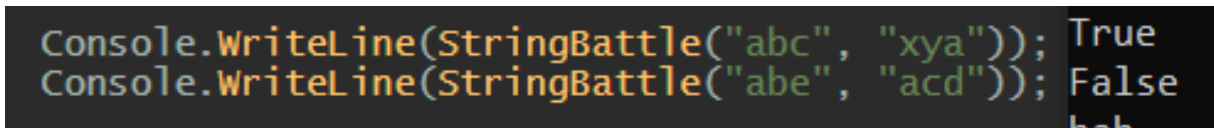
        return;
    }
    for (int i = 0; i < arr.Count; i++)
    {
        List<char> lst = new List<char>(arr);
        lst.RemoveAt(i);
```

```

        ShowAllCombinations(lst, sl, ref result, current +
arr[i].ToString());
    }
}

```

Результат



```

Console.WriteLine(StringBattle("abc", "xya")); True
Console.WriteLine(StringBattle("abe", "acd")); False

```

Рисунок 7 – Результат работы программы

Задача №2 со строками

Описание

Дана строка s, вернуть самую длинную полиндромную подстроку в s

Код программы

```

public static string MaxPalindrome(string word)
{
    int count = word.Length;

    while (count != 0)
    {
        for (int i = 0; i < word.Length - count; i++)
        {
            int secondIndex = i + count;

            if (PalindromeCheck(word, i, secondIndex))
            {
                StringBuilder builder = new StringBuilder();

                for (int k = i; k <= secondIndex; k++)
                {
                    builder.Append(word[k]);
                }

                return builder.ToString();
            }
        }

        count--;
    }

    return string.Empty;
}

private static bool PalindromeCheck(string word, int minIndex, int
maxIndex)
{
    for (int i = 0; i < maxIndex-minIndex; i++)
    {

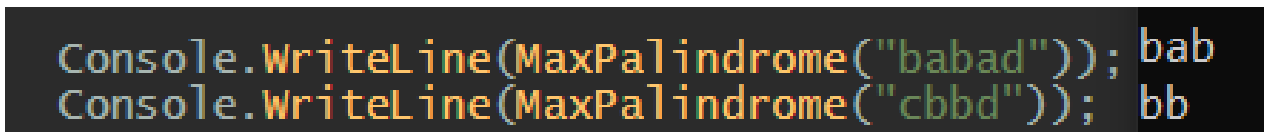
```

```

        if (word[minIndex+i] != (word[maxIndex-i]))
        {
            return false;
        }
    }
    return true;
}

```

Результат



```

Console.WriteLine(MaxPalindrome("babad")); bab
Console.WriteLine(MaxPalindrome("cbbd")); bb

```

Рисунок 8 – Результат работы программы

Задача №3 со строками

Описание

Вернуть количество отдельных непустых подстрок текста, которые могут быть записаны как конкатенация некоторой строки с самой собой (т.е. она может быть записана, как $a + a$, где a - некоторая строка).

Код программы

```

public static int DistinctEchoSubstrings(string text)
{
    int mod = (int)Math.Pow(10, 9) + 7;
    int[,] dp = new int[text.Length + 1, text.Length + 1];
    HashSet<int> res = new HashSet<int>();

    for (int i = 1; i <= text.Length; i++)
    {
        for (int j = i; j <= text.Length; j++)
        {
            dp[i, j] = dp[i, j - 1] * 26 % mod + text[j - 1] - 'a' + 1;

            if (i * 2 - j - 1 >= 0 && dp[i * 2 - j - 1, i - 1] == dp[i,
j])
            {
                res.Add(dp[i, j]);
            }
        }
    }

    return res.Count;
}

```

Результат

```
Console.WriteLine(DistinctEchoSubstrings("abcbabcabc")); 3
```

Рисунок 9 – Результат работы программы

Вывод

В ходе выполнения курсовой работы, был решен ряд задач, в которых были использованы необходимые знания по алгоритмам и структурам данных.