Факультет компьютерных технологий

Лабораторная работа №9

«Разработка программ с использованием управляющих инструкций»

Вариант №10

Выполнил: студент гр. 981063 Ефименко Павел Викторович

Проверил: Яковлев А.В.

Минск 2021

Задание №1 ­Обработка исключительных ситуаций

Условие:

Шифр «двойной квадрат» Уинстона

Код:

namespace Laba\_1

{

    class WheatstoneСipher

    {

        private readonly char[,] \_firstMatrix = new char[7, 5]

        {

          {'ж', 'щ', 'н', 'ю', 'р'},

          {'и', 'т', 'ь', 'ц', 'б'},

          {'я', 'м', 'е', '.', 'с'},

          {'в', 'ы', 'п', 'ч', ' '},

          {'й', 'д', 'у', 'о', 'к'},

          {'з', 'э', 'ф', 'г', 'ш'},

          {'х', 'а', ',', 'л', 'ъ'}

        };

        private readonly char[,] \_secondMatrix = new char[7, 5]

        {

          {'и', 'ч', 'г', 'я', 'т'},

          {',', 'ж', 'м', 'ь', 'о'},

          {'з', 'ю', 'р', 'в', 'щ'},

          {'ц', 'й', 'п', 'е', 'л'},

          {'ъ', 'а', 'н', '.', 'х'},

          {'э', 'к', 'с', 'ш', 'д'},

          {'б', 'ф', 'у', 'ы', ' '}

        };

        public string Encrypt(string text)

        {

            text = text.ToLower();

            string result\_text = "";

            if (text.Length % 2 != 0)

            {

                text += ' ';

            }

            int length = text.Length / 2;

            int k = 0;

            char[,] bigram = new char[length, 2];

            char[,] kripto\_bigram = new char[length, 2];

            for (int i = 0; i < length; i++)

            {

                for (int j = 0; j < 2; j++)

                {

                    bigram[i, j] = text[k];

                    k++;

                }

            }

            int step = 0;

            while (step < length)

            {

                var cortege1 = FindIndexes(bigram[step, 0], \_firstMatrix);

                var cortege2 = FindIndexes(bigram[step, 1], \_secondMatrix);

                kripto\_bigram[step, 0] = \_secondMatrix[(int)cortege1.i, (int)cortege2.j];

                kripto\_bigram[step, 1] = \_firstMatrix[(int)cortege2.i, (int)cortege1.j];

                step++;

            }

            for (int i = 0; i < length; i++)

            {

                for (int j = 0; j < 2; j++)

                {

                    result\_text += kripto\_bigram[i, j].ToString();

                }

            }

            return result\_text;

        }

        public string Decrypt(string text)

        {

            string result\_text = "";

            int length = text.Length / 2;

            int k = 0;

            char[,] bigram = new char[length, 2];

            char[,] kripto\_bigram = new char[length, 2];

            for (int i = 0; i < length; i++)

            {

                for (int j = 0; j < 2; j++)

                {

                    bigram[i, j] = text[k];

                    k++;

                }

            }

            int step = 0;

            while (step < length)

            {

                var cortege1 = FindIndexes(bigram[step, 0], \_secondMatrix);

                var cortege2 = FindIndexes(bigram[step, 1], \_firstMatrix);

                kripto\_bigram[step, 0] = \_firstMatrix[(int)cortege1.i, (int)cortege2.j];

                kripto\_bigram[step, 1] = \_secondMatrix[(int)cortege2.i, (int)cortege1.j];

                step++;

            }

            for (int i = 0; i < length; i++)

            {

                for (int j = 0; j < 2; j++)

                {

                    result\_text += kripto\_bigram[i, j].ToString();

                }

            }

            return result\_text;

        }

        private (int? i, int? j) FindIndexes(char symbol, char[,] matrix)

        {

            for (int i = 0; i < 7; i++)

            {

                for (int j = 0; j < 5; j++)

                {

                    if (symbol == matrix[i, j])

                    {

                        return (i, j);

                    }

                }

            }

            return (i: null, j: null);

        }

    }

}

using System;

namespace Laba\_1

{

    class Program

    {

        static void Main(string[] args)

        {

            var wheatstoneСipher = new WheatstoneСipher();

            var programParam = args[0].ToLower() ?? throw new ArgumentNullException("args");

            var userString = args[1] ?? throw new ArgumentNullException("args");

            try

            {

                MethodHandle(wheatstoneСipher, programParam, userString);

            }

            catch (Exception ex)

            {

                Console.WriteLine(ex.Message);

                Console.WriteLine("[1]: decrypt or encrypt\n[2]:Your string");

            }

            Console.WriteLine("Press any key for close.");

            Console.ReadKey();

        }

        private static void MethodHandle(WheatstoneСipher wheatstoneСipher, string programParam, string userString)

        {

            switch (programParam)

            {

                case "decrypt":

                    var resutlDecrypt = wheatstoneСipher.Decrypt(userString);

                    Console.WriteLine(resutlDecrypt);

                    break;

                case "encrypt":

                    var resultEncrypt = wheatstoneСipher.Encrypt(userString);

                    Console.WriteLine(resultEncrypt);

                    break;

                default:

                    throw new ArgumentWrongException("Невеный порядок аргументов");

            }

        }

    }

}

using System;

namespace Laba\_1

{

    [Serializable]

    internal class ArgumentWrongException : Exception

    {

        public ArgumentWrongException()

        {

        }

        public ArgumentWrongException(string message) : base(message)

        {

        }

    }

}