Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

Кафедра интеллектуально-информационных технологий

Лабораторная работа №2
“Избыточное кодирование данных в информационных системах. Итеративные коды”

По дисциплине “Современные методы защиты компьютерных систем”

Выполнила:
студентка 4 курса
группы ИИ-22
Леваневская Н.И.
Проверил:
Хацкевич А.С.

Брест-2024

Вариант 5

Цель: приобретение практических навыков кодирования/декодирования двоичных данных при использовании итеративных кодов.

Ход работы:

Задача. Разработать приложение для кодирования/декодирования двоичной информации итеративным кодом различной относительной избыточностью кодовых слов.

Длина информационного слова (бит), k	k1	k2	z	Количество групп паритетов
40	5 4 5 2	8 10 4 10	- - 2 2	2;3 2;3 2;3;4;5 2;3;4;5

Код программы:

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace lab_2
{
    class MatrixCode
    {
        private int k, k1, k2, z, size, pSize;
        private List<int> parity;
        private bool isPpBits;

        public MatrixCode(int k, int k1, int k2, int z = 1, List<int> parity = null, bool isPpBits =
true)
        {
            this.k = k;
            this.k1 = k1;
            this.k2 = k2;
            this.z = z;
            this.parity = parity ?? new List<int> { 2, 3 };
            this.isPpBits = isPpBits;
            this.size = k / z;
            pSize = 0;

            if (this.parity.Contains(2)) pSize += k1;
            if (this.parity.Contains(3)) pSize += k2;
            if (this.parity.Contains(4)) pSize += 1;
            if (this.parity.Contains(5)) pSize += 1;
        }


        private List<List<int>> GetMatrix2D(string message)
        {
            var matrix2D = new List<int>();
            for (int i = 0; i < k1; i++)
            {
                for (int j = 0; j < k2; j++)
```

```
            {
                matrix2D.Add(int.Parse(message[i * k2 + j].ToString()));
            }
        }
        return new List<List<int>> { matrix2D };
    }

    private List<int> GetParityBits(List<List<int>> matrix2D)
    {
        var h = new List<int>();

        if (parity.Contains(2))
        {
            for (int i = 0; i < k1; i++)
            {
                int bit = 0;
                for (int j = 0; j < k2; j++)
                {
                    bit ^= matrix2D[0][i * k2 + j];
                }
                h.Add(bit);
            }
        }

        var v = new List<int>();

        if (parity.Contains(3))
        {
            for (int j = 0; j < k2; j++)
            {
                int bit = 0;
                for (int i = 0; i < k1; i++)
                {
                    bit ^= matrix2D[0][i * k2 + j];
                }
                v.Add(bit);
            }
        }

        var d = new List<int>();

        if (parity.Contains(4) || parity.Contains(5))
        {
            int group4Parity = 0;
            for (int i = 0; i < k1; i++)
            {
                for (int j = 0; j < k2; j++)
                {
                    group4Parity ^= matrix2D[0][i * k2 + j];
                }
            }
            d.Add(group4Parity);

            if (parity.Contains(5))
            {
                int group5Parity = 0;
                for (int i = 0; i < Math.Min(k1, k2); i++)
                {
                    group5Parity ^= matrix2D[0][i * k2 + i];
                }
                d.Add(group5Parity);
            }
        }

        if (isPpBits)
        {
            int ppBit = h.Aggregate(0, (current, item) => current ^ item) ^ v.Aggregate(0,
(current, item) => current ^ item);
```

```
                return h.Concat(v).Concat(d).Append(ppBit).ToList();
            }
            return h.Concat(v).Concat(d).ToList();
        }

        public string Encode(string message)
        {
            var matrix2D = GetMatrix2D(message);
            var pBits = GetParityBits(matrix2D);
            return message + string.Join("", pBits);
        }

        public (int? errPos, string message) Decode(string encoded)
        {
            var pBits = encoded.Substring(k);
            var message = encoded.Substring(0, k);
            var encodeTest = Encode(message);

            if (encoded == encodeTest)
            {
                return (null, message);
            }
            else
            {
                var errPos = GetErrPos(pBits, encodeTest.Substring(k));
                var messageCorrected = message.Substring(0, errPos) + (message[errPos] == '0' ? '1'
: '0') + message.Substring(errPos + 1);
                return (errPos, messageCorrected);
            }
        }

        private int GetErrPos(string pBits, string pBitsTest)
        {
            if (pBits.Length != pBitsTest.Length)
            {
                throw new ArgumentException("Длины битов четности не совпадают.");
            }

            int errorPositionX = -1;
            int errorPositionY = -1;

            for (int i = 0; i < k1; i++)
            {
                if (pBits[i] != pBitsTest[i])
                {
                    errorPositionX = i;
                    break;
                }
            }

            for (int i = k1; i < pBits.Length; i++)
            {
                if (pBits[i] != pBitsTest[i])
                {
                    errorPositionY = i - k1;
                    break;
                }
            }

            return errorPositionX * k2 + errorPositionY;
        }


    }

    class Program
    {
        static void Main()
```

```
        {
            int k = 40;
            int k1 = 5;
            int k2 = 8;
            int z = 1;

            List<int> customParity = new List<int> { 2, 3};

            MatrixCode matrixCode = new MatrixCode(k, k1, k2, z, customParity);

            string message = "1101010101010101010101010101010101010101";

            string encodedMessage = matrixCode.Encode(message);
            Console.WriteLine("Encoded Message: " + encodedMessage);

            string erroneousMessage = encodedMessage.Substring(0, 5) + (encodedMessage[5] == '0' ?
'1' : '0') + encodedMessage.Substring(6);
            Console.WriteLine("Erroneous Message: " + erroneousMessage);

            var (errPos, decodedMessage) = matrixCode.Decode(erroneousMessage);
            Console.WriteLine("Decoded Message: " + decodedMessage);
            if (errPos.HasValue)
            {
                Console.WriteLine($"Error Position: {errPos.Value}");
            }
            else
            {
                Console.WriteLine("No errors detected.");
            }
        }
    }

}
```

Результат работы программы:

```
Encoded Message: 1101010101010101010101010101010101010110000110101010
Erroneous Message: 1101000101010101010101010101010101010110000110101010
Decoded Message: 1101010101010101010101010101010101010101
Error Position: 5
```

Вывод: приобрела практические навыки кодирования/декодирования двоичных данных при использовании итеративных кодов**.**