«Код Хемминга»

Листинг программы

```
// get mess
Console.WriteLine("Введите кодируемое сообщение");
string str = Console.ReadLine();
//to bits
byte[] str_bytes = Encoding.ASCII.GetBytes(str);
BitArray str_bits = new BitArray(str_bytes);
//print
Console.Write("Сообщение в байтах: ");
PrintBytes(str_bytes);
Console.Write("Сообщение в битах: ");
PrintBits(str_bits, 8);
PrintBits(str_bits,4);
```

Программа начинает работу с вводом пользователя кодируемого сообщения. Это собщение представляется в байт формате, а потом и в бит формате, паралельно выводя результаты на экран с помощью методов для вывода, что бы нагляднее выглядел процесс.

```
static void PrintBits(BitArray bits, int n)
{
    for(int i=0;i<bits.Length/n;i++)
    {
        for(int j=0;j<n;j++)
        {
            if(bits[n * i + j])
                Console.Write('1');
            else
                Console.Write('0');
        }
        Console.Write(' ');
    }
    Console.WriteLine();
}
Ссылок: 2
static void PrintBytes(byte[] bytes)
{
    for(int i=0;i<bytes.Length;i++)
    {
        Console.Write(bytes[i]);
        Console.Write(' ');
    }
    Console.WriteLine();
}
```

Это сами методы для вывода байтов и битов в удобном, читаемом формате.

```
//code
int n = 4, nr = 7;
BitArray bits4 = new BitArray(n);
BitArray bits7 = new BitArray(nr);
BitArray str_bits_code = new BitArray(str_bits.Length / n * nr);
for(int i=0;i<str_bits.Length/n;i++)
{
    bits4.SetAll(false);
    for(int j=0;j<n;j++)
    {
        bits4[j] = str_bits[i * n + j];
    }
    bits7 = code_block(bits4);
    for (int j = 0; j < nr; j++)
    {
        str_bits_code[i * nr + j] = bits7[j];
    }
}
Console.WriteLine("Закодированное");
Console.Write("Сообщение в битах: ");
PrintBits(str_bits_code, nr);
```

Далее программа кодирует сообщение по Хеммингу. Она делить массив исходных битов на блоки по 4 бита, и кодирует их, на выходе получая блоки по 7 битов и сращивая их в один итоговый массив. Результат выводиться на экран. Кодирование происходит с помощью метода code_block.

```
static BitArray code_block(BitArray bits)
{
    //create generate matrix
    byte[,] generate_matrix = new byte[,]{{1,0,0,0,1,0,1 },
                                          {0,1,0,0,1,1,1 },
                                          {0,0,1,0,1,1,0 },
                                          {0,0,0,1,0,1,1 }};
    //create bits value
    byte[] bits_v = new byte[4];
    for(int i=0;i<4;i++)
    {
        if (bits[i])
        {
            bits_v[i] = 1;
        }
        else
        {
            bits_v[i] = 0;
        }
    }
    //code
    BitArray code_bits = new BitArray(7);
    code_bits.SetAll(false);
    for(int i=0;i<7;i++)
    {
        if((bits_v[0] * generate_matrix[0,i] + bits_v[1] * generate_matrix[1, i] + bits_v[2] * generate_matrix[2, i] + bits_v[3] * generate_matrix[3, i])%2 == 1)
        {
            code_bits[i] = true;
        }
        else
        {
            code_bits[i] = false;
        }
    }
    return code_bits;
}
```

Метод code_block принимает массив из 4 бит. Он генерирует массив bits_v в формате байт с 1 и 0, а не с true false, как в формате BitArray. Это необходимо для удобного перемножения матриц. Далее метод перемножает матрицу bits_v с матрицей generate_matrix, которая дана по заданию и является порождающей. В итоге получается 7 бит, 4 первых - это исходные биты, 3 последние - это контрольные биты. Результат записывается уже в формате BitArray.

```
//read
Console.WriteLine("Введите ваш код:");
str_bits_code = StringToBits(Console.ReadLine());
//decode
bits4.SetAll(false);
bits7.SetAll(false);
str_bits.SetAll(false);
for (int i=0;i< str_bits_code.Length/nr; i++)
{
    bits7.SetAll(false);
    for (int j = 0; j < nr; j++)
    {
        bits7[j] = str_bits_code[i * nr + j];
    }
    bits4 = dcode_block(bits7);
    for (int j = 0; j < n; j++)
    {
        str_bits[i * n + j] = bits4[j];
    }
}
Console.WriteLine("Декодированное");
Console.Write("Сообщение в битах: ");
PrintBits(str_bits, n);
//to bytes
for(int i=0;i<str_bits.Length/8;i++)
{
    str_bits.CopyTo(str_bytes, 0);
}
Console.Write("Сообщение в байтах: ");
PrintBytes(str_bytes);
//to string
Console.Write("Сообщение: ");
Console.WriteLine(System.Text.Encoding.ASCII.GetString(str_bytes));
Console.ReadKey();
```

Далее программа просит у пользователя код. Этот код должен симулировать полученный код с возможной ошибкой, так как в условиях идеальной работы компьютера ошибку естественным путём получить почти невозможно. Введёный новый код программа подразделяет на блоки по 7 бит, декодирует их методом dcode_block и выводит результат в блок по 4 бита. За тем сращивает блоки в итоговый массив бит и выводит его на экран. Этот массив программа также преобразовывает в массив байт и символы.

```
static BitArray dcode_block(BitArray bits)
{
    byte[,] generate_matrix = new byte[,]{{1,0,1},
                                          {1,1,1},
                                          {1,1,0},
                                          {0,1,1},
                                          {1,0,0},
                                          {0,1,0},
                                          {0,0,1}};
    //create bits value
    byte[] bits_v = new byte[7];
    for (int i = 0; i < 7; i++)
    {
        if (bits[i])
        {
            bits_v[i] = 1;
        }
        else
        {
            bits_v[i] = 0;
        }
    }
```

```
//dcode
bool o = true;
BitArray code_bits = new BitArray(4);
byte[] S_bits = new byte[3];
code_bits.SetAll(false);
for (int i=0;i<4;i++)
{
    code_bits[i] = bits[i];
}
int sum = 0;
while (o)
{
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 7; j++)
        {
            sum += bits_v[j] * generate_matrix[j, i];
        }
        if (sum % 2 == 1)
        {
            S_bits[i] = 1;
        }
        else
        {
            S_bits[i] = 0;
        }
        sum = 0;
    }
```

```
        //check
        o = false;
        for (int i = 0; i < 3; i++)
        {
            if (S_bits[i] != 0)
            {
                o = true;
            }
        }
        int num = 0;
        if (o)
        {
            for (int i = 0; i < 7; i++)
            {
                if ((S_bits[0] == generate_matrix[i, 0]) && (S_bits[1] == generate_matrix[i, 1]) && (S_bits[2] == generate_matrix[i, 2]))
                {
                    num = i;
                    break;
                }
            }
            if (num < 4)
            {
                code_bits[num] = !code_bits[num];
                o = false;
            }
            else
            {
                if (bits_v[num] == 1)
                {
                    bits_v[num] = 0;
                }
                else
                {
                    bits_v[num] = 1;
                }
            }
        }
    }
    return code_bits;
}
```

Метод dcode_block получает на вход массив из 7 бит. Он так же преобразовывает массив в тип байт для удобства. Далее он записывает первые биты в массив для вывода. Потом наступает этап проверки. Он происходит циклом, так как если ошибка была в проверочных битах, то необходимо пересчитать результат. Происходит перемножение матриц с порождающей, заданной уже для декодирования. Так как число сумм при перемножении элементов матрицы большое, итоговая сумма считается циклом, а уже потом расчитывается результат по остатку от 2. В итоге получается синдром последовательности - массив с 3 битам (типом байт для удобства). Далее наступает этап проверки. Если все биты синдрома равны 0, то все биты и так переданны правильно. Если есть где то ненулевой элемент, начинается сравнение комбинаций синдрома с строками порождающей матрицы. Найдя нужную строку получаем номер ошибочного символа. Если он относится к первым 4 битам, то мы просто инвертируем нужный, если - к проверочным, то мы инвертируем

его и идём слежующий круг цикла, уже с правильным проверочным битом. Хотя это и не обязательно, так как одну ошибку программа нашла, а если их больше, то ошибки изначально неправильно исправяться.

Пример работы

```
Введите кодируемое сообщение
w
Сообщение в байтах: 119
Сообщение в битах: 11101110
1110 1110
Закодированное
Сообщение в битах: 1110100 1110100
Введите ваш код:
10101000110100
Декодированное
Сообщение в битах: 1110 1110
Сообщение в байтах: 119
Сообщение: w
```

Для начала рассмотрим простейший пример с одним символом. Я намеренно совершил ошибки в 2 и 8 бите, так как ошибки находятся в разных блоках, они исправляются.

```
Введите кодируемое сообщение
word
Сообщение в байтах: 119 111 114 100
Сообщение в битах: 11101110 11110110 01001110 00100110
1110 1110 1111 0110 0100 1110 0010 0110
Закодированное
Сообщение в битах: 1110100 1110100 1111111 0110001 0100111 1110100 0010110 0110001
Введите ваш код:
11101001110100111111110110001010011111101000010110011000 1
Декодированное
Сообщение в битах: 1110 1110 1111 0110 0100 1110 0010 0110
Сообщение в байтах: 119 111 114 100
Сообщение: word
```

Вот пример с более сложным сообщением. Нужно учитывать, что при вводе кода, его необходимо вводить слитно. Выводится код раздельно для удобства чтения.

```
Введите кодируемое сообщение
Hello, word!!!
Сообщение в байтах: 72 101 108 108 111 44 32 119 111 114 100 33 33 33
Сообщение в битах: 00010010 10100110 00110110 00110110 11110110 00110100 00000100 11101110 11110110 01001110 00100110 10
000100 10000100 10000100
0001 0010 1010 0110 0011 0110 0011 0110 1111 0110 0011 0100 0000 0100 1110 1110 1111 0110 0100 1110 0010 0110 1000 0100
1000 0100 1000 0100
Закодированное
Сообщение в битах: 0001011 0010110 1010011 0110001 0011101 0110001 0011101 0110001 1111111 0110001 0011101 0100111 00000
00 0100111 1110100 1110100 1111111 0110001 0100111 1110100 0010110 0110001 1000101 0100111 1000101 0100111 1000101 01001
11
Введите ваш код:
0001011001011010010011011000100111010110001001110101110011111111101101010011101010011100010000100111111010011101001111111
1100010100111111010000101100110001100010101001111000101010011110001010100111
Декодированное
Сообщение в битах: 0001 0010 1010 0110 0011 0110 0011 0110 1111 0110 0011 0100 0000 0100 1110 1110 1111 0110 0100 1110 0
010 0110 1000 0100 1000 0100 1000 0100
Сообщение в байтах: 72 101 108 108 111 44 32 119 111 114 100 33 33 33
Сообщение: Hello, word!!!
```

Ещё пример работы с намеренными ошибками.