МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 25

ОТЧЁТ ЗАЩИЩЕН С ОЦЕНКОЙ_____

РУКОВОДИТЕЛЬ

ассистент		А. А. Бурков
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

ИСПОЛЬЗОВАНИЕ ЦИКЛИЧЕСКИХ КОДОВ ДЛЯ ОБНАРУЖЕНИЯ ОШИБОК В СЕТЯХ ПЕРЕДАЧИ ДАННЫХ.

по дисциплине: Устройства и системы беспроводной связи.

СТУДЕНТ ГР. № 3035		С.Е.Попов
номер группы	подпись, дата	инициалы, фамилия

Санкт-Петербург 2022

Цель работы

Разработка программы, наглядно демонстрирующую работу кодера и декодера для типового алгоритма формирования циклических кодов.

Описание моделируемой системы

В большинстве современных систем передачи данных для обнаружения ошибок применяется следующий подход. К передаваемым данным добавляют контрольную сумму, которая вычисляется на основе этих же данных. По каналу передается сообщение, состоящее из данных и контрольной суммы. Использование контрольной суммы позволяет определить, по принятому сообщению, возникли ли ошибки при передаче данного сообщения по каналу.

На рис. 1 изображена структурная схема рассматриваемой в лабораторной работе системы передачи данных.

Рисунок 1 – Структурная схема системы передачи данных

На вход кодера поступает некоторое информационное сообщение m, состоящее из нулей и единиц. Кодер по некоторому алгоритму вычисляет контрольную сумму, дописывает ее к передаваемому сообщению и таким образом формирует закодированное сообщение а так же состоящее из 0 и 1. В канале могут произойти ошибки, в результате которых некоторые биты сообщения инвертируются (0 становится 1 или 1 становится 0). Вектор ошибок показывает на каких позициях произошла ошибка, при этом канал может быть описан как операция XOR передаваемого сообщения и вектора ошибок. Пример: $\bar{a}$ = [101], $\bar{e}$ = [001], $\bar{b}$ = [100]. Декодер по некоторому

алгоритму проверяет контрольную сумму в принятом сообщении и принимает одно из следующих решений:

$$E = \begin{cases} 1, \text{если были ошибки} \\ 0, \text{если не было ошибок} \end{cases}$$

Рассматривается модель двоично-симметричного канала (ДСК) без памяти представленного на рис. 2. Как видно из рисунка 2 с вероятностью p происходит ошибка (0 становится 1 или 1 становиться 0). Канал является двоичным, поэтому возможны только два значения битов на входе и выходе канала: {0,1}. Канал называется симметричным ввиду того, что вероятность ошибки для обоих значений битов одинакова. Модель ДСК приведена на рис. 2. Канал без памяти характеризуется тем, что случайные события, связанные с ошибками в канале независимы для разных моментов времени.

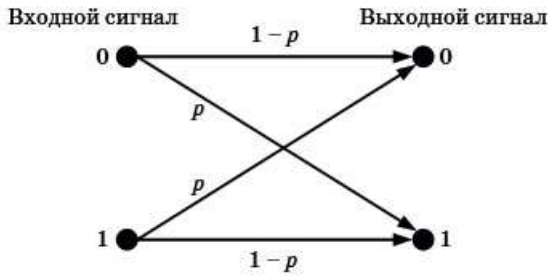

Рисунок 2 – Модель двоично -симметричного канала

Описание задания для варианта

На вход программы подается порождающий многочлен g x( ), число k, вектор ошибки e и информационная последовательность из l бит (l может быть как меньше, так и больше k). Программа формирует кодовое слово. На основе него и вектора ошибки формируется последовательность на выходе канала. По принятой последовательности принимается решение о наличии ошибок в канале. В программе должна быть предусмотрена возможность вывода всех промежуточных значений, которые формируются как при работе кодера, так и декодера.

Алгоритм работы Кодера:

Передаваемое сообщение рассматривается как вектор длины k. Для каждого сообщения ($\overline{m}$) кодер выполняет следующие действия:

1. На основе вектора m формируется многочлен m(x). Степень многочлена m(x) при этом может быть, как равна k – 1, так и превышать это значение;

2. Вычисляется многочлен $c(x) = m(x)x^r \bmod g(x)$;

3. Вычисляется многочлен $a(x) = m(x)x^r + c(x)$;

4. На основе многочлена a(x) формируется вектор a, длина которого n бит, где n = k + r.

Алгоритм работы Декодера:

1. Принятое сообщение $\bar{b} = \bar{a} + \bar{e}$ переводится в многочлен b(x);

2. Вычисляется синдром: $s(x) = b(x) \bmod g(x)$;

3. Если $s(x) \neq 0$, то декодер выносит решение, что произошли ошибки (E = 1), иначе декодер выносит решение, что ошибки не произошли (E = 0).

Блок схема алгоритма

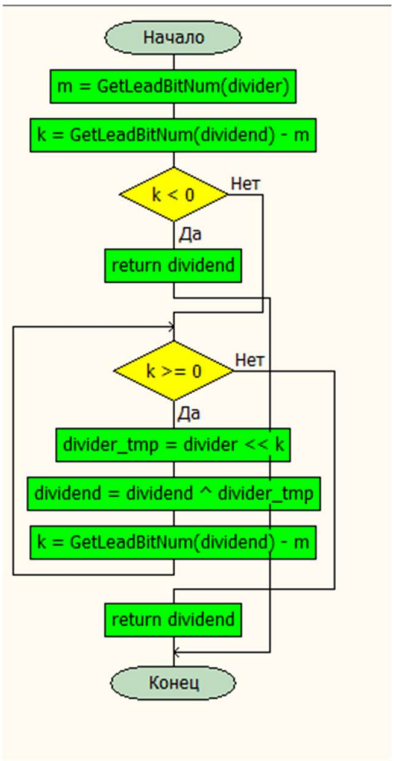

Рис. 3. – Блок схема

Контрольный пример (Visual Studio – C#)

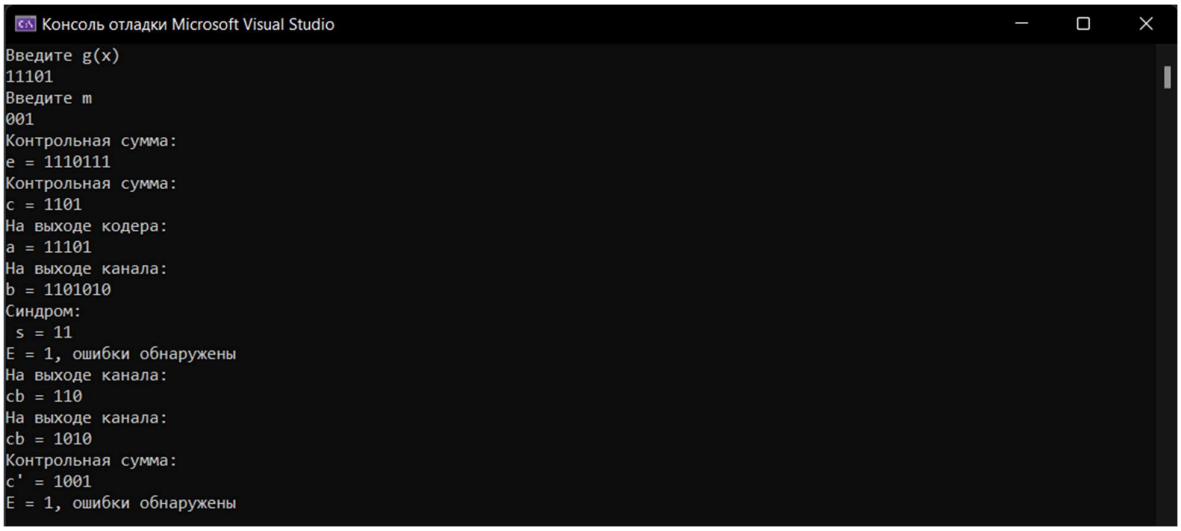

Рис. 4. – Пример 1.

Дополнительное задание пример (Visual Studio – C#)

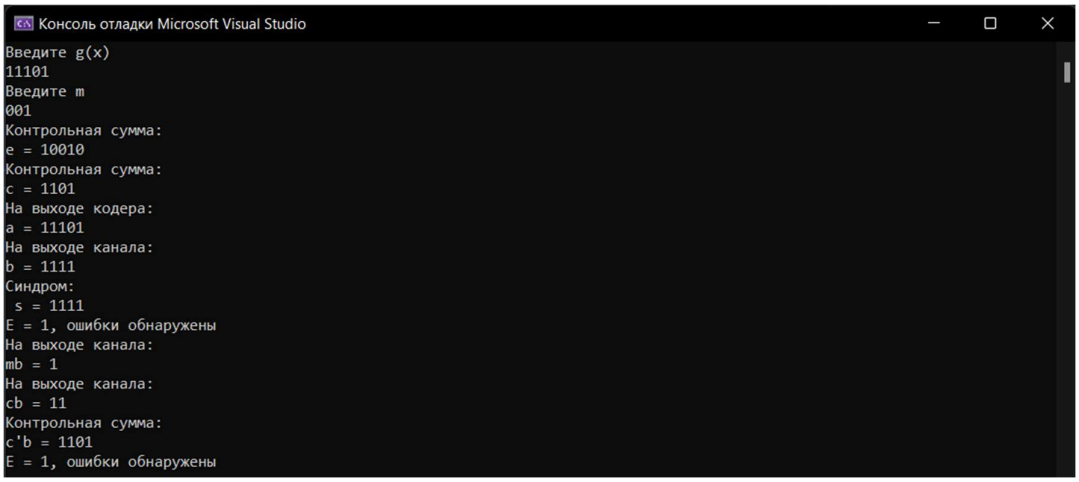

Рис. 5. – Пример 1.

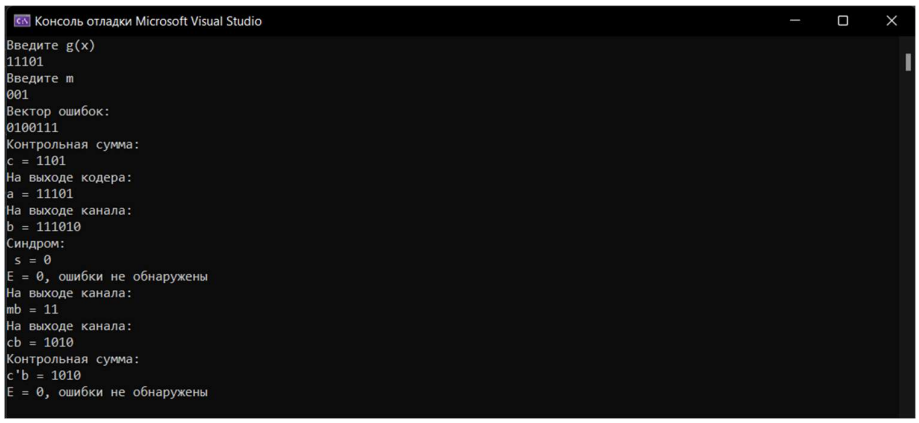

Рис. 6. – Пример 2.

Выводы

Были выполнены цели работы: рассмотрен алгоритм построения контрольной суммы (кодирование и декодирование), обнаружения ошибки; разработана программа, кодирующая и декодирующая сообщение, и выявляющая наличие ошибки. Также было сделано дополнительное задание.

Листинг программы

```
class Program
{
    static int GetLeadBitNum(int Val)
    {
        int BitNum = 31;
        uint CmpVal = 1u << BitNum;
        while (Val < CmpVal)
        {
            CmpVal >>= 1;
            BitNum--;
        }
        return BitNum;
    }

    static int DividePolynomMod(int dividend, int divider)
    {
        int m = GetLeadBitNum(divider);
        int k = GetLeadBitNum(dividend) - m;
        int remainder;
        if (k < 0)
        {
            return dividend;
        }
        while (k >= 0)
        {
            int divider_tmp = divider << k;
            dividend = dividend ^ divider_tmp;
            k = GetLeadBitNum(dividend) - m;
        }
        return dividend;
    }

    static void Main()
    {
        int g, e, m, c, s, tlen, tc_, tm_, c_;
        string tb, tc, tm;
        Console.WriteLine("Введите g(x)");
        g = Convert.ToInt32(Console.ReadLine(), 2);
        Console.WriteLine("Введите m");
        m = Convert.ToInt32(Console.ReadLine(), 2);

        // Генерация вектора ошибки
        Random rnd = new Random();
        e = rnd.Next(0, 127);
        Console.WriteLine("Контрольная сумма:\ne = " + Convert.ToString(e, 2));
        /* Console.WriteLine("Вектор ошибок: ");
         e = Convert.ToInt32(Console.ReadLine(), 2);*/
        //  Console.WriteLine("Вектор ошибок:\ne = " + e);

        m = m << GetLeadBitNum(g);
```

```
            c = DividePolynomMod(m, g);
            Console.WriteLine("Контрольная сумма:\nc = " + Convert.ToString(c, 2));
            m = m | c;
            Console.WriteLine("На выходе кодера:\na = " + Convert.ToString(m, 2));
            m = m ^ e;

            tb = Convert.ToString(m, 2);
            tlen = (tb.Length / 2) - 1;
            tm = new string(tb.Take(tlen).ToArray());
            tc = new string(tb.Take(2..^0).ToArray());
            Console.WriteLine("На выходе канала:\nb = " + Convert.ToString(m, 2));

            s = DividePolynomMod(m, g);
            Console.WriteLine("Синдром:\n s = " + Convert.ToString(s, 2));
            if (s == 0)
            {
                Console.WriteLine("E = 0, ошибки не обнаружены");
            }
            else
            {
                Console.WriteLine("E = 1, ошибки обнаружены");
            }
            //Доп задание
            Console.WriteLine("На выходе канала:\nmb = " + Convert.ToString(tm));
            Console.WriteLine("На выходе канала:\ncb = " + Convert.ToString(tc));
            tc_ = Convert.ToInt32(tc, 2);
            tm_ = Convert.ToInt32(tm, 2);

            tm_ = tm_ << GetLeadBitNum(g);
            c_ = DividePolynomMod(tm_, g);
            Console.WriteLine("Контрольная сумма:\nc'b = " + Convert.ToString(c_, 2));

            if (tc_ == c_)
            {
                Console.WriteLine("E = 0, ошибки не обнаружены");
            }
            else
            {
                Console.WriteLine("E = 1, ошибки обнаружены");
            }
        }
    }
```