



**«Московский государственный технический университет
имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Системы обработки информации и управления (ИУ5)

О т ч е т

по домашнему заданию №2

**Разработка программных средств определения обнаруживающей и
корректирующей способности кода в линейных протоколах**

Дисциплина: Сети и телекоммуникации

Студент гр. ИУ5-53Б

(Подпись, дата)

Терентьев В.О.

(Фамилия И.О.)

Преподаватель

(Подпись, дата)

Галкин В.А.

(Фамилия И.О.)

Москва, 2020

1. Метод решения задачи для варианта задания

Задача решена в виде приложения Windows Presentation Foundation на языке C#.

Вариант задания:

№ варианта	Информационный вектор	Код	Способность кода
20	11111010001	Ц [15,11]	C_0

2. Реализация алгоритмов кодирования, декодирования, модели канала связи и вычисления обнаруживающей способности кода для ошибок всех возможных кратностей

Реализация алгоритма кодирования:

Функция **Coding** реализует алгоритм кодирования циклическим [15,11]-кодом. Она принимает на вход информационный вектор. Результатом функции является циклический [15,11]-код.

```
public int[] Coding(int[] vec)
{
    int[] c_vec = new int[15];
    for (int i = 4; i < 15; ++i)
        c_vec[i] = vec[i - 4];
    for (int i = 14; i > 3; --i)
    {
        if (c_vec[i] == 0)
            continue;
        for (int k = 0; k < 5; ++k)
            c_vec[i - k] = c_vec[i - k] ^ g_pol[4 - k];
    }
    for (int i = 4; i < 15; ++i)
        c_vec[i] = vec[i - 4];
    return c_vec;
}
```

Реализация алгоритма декодирования:

Функция **Decoding** реализует алгоритм декодирования циклического [15,11]-кода. Она принимает на вход циклический [15,11]-код. Результатом функции является вектор синдрома ошибки.

```
public int[] Decoding(int[] vec)
{
    int[] tmp = (int[])vec.Clone();
    for (int i = 14; i > 3; --i)
```

```

    {
        if (tmp[i] == 0)
            continue;
        for (int k = 0; k < 5; ++k)
            tmp[i - k] = tmp[i - k] ^ g_pol[4 - k];
    }
    int[] synd = new int[4];
    for (int i = 0; i < 4; ++i)
        synd[i] = tmp[i];
    return synd;
}

```

Реализация модели канала связи:

Функция **Error** реализует модель канала связи. Она принимает на вход два параметра: кодовый вектор и вектор ошибки. Результатом функции является кодовый вектор, принятый на выходе канала связи.

```

public int[] Error(int[] vec, int[] err_vec)
{
    int[] a_vec = new int[15];
    for (int i = 0; i < 15; ++i)
        a_vec[i] = vec[i] ^ err_vec[i];
    return a_vec;
}

```

Реализация вычисления обнаруживающей способности кода для ошибок всех возможных кратностей:

Вычисление обнаруживающей способности кода для ошибок всех возможных кратностей выполняется в цикле **for**. Для каждого вектора ошибки текущей кратности с помощью функций **Error** и **Decoding** вычисляются соответственно принятый на выходе канала связи кодовый вектор и вектор синдрома ошибки и по синдрому ошибки определяется факт обнаружения ошибки или его отсутствие. Результаты вычислений записываются в таблицу данных **result**.

```

for (int i = 1; i < 16; ++i)
{
    int C = 0;
    int N = 0;
    int[] err_vec = new int[15];
    int[] res = new int[i];
    Stack<int> stack = new Stack<int>();
    stack.Push(0);
    while (stack.Count > 0)
    {
        int value = stack.Pop();
    }
}

```

```

        int index = stack.Count;
        if (value - 1 >= 0)
            err_vec[value - 1] = 0;
        while (value < 15)
        {
            err_vec[value] = 1;
            res[index++] = value++;
            stack.Push(value);

            if (index == i)
            {
                int[] synd_vec = Decoding(Error(c_vec, err_vec));
                ++C;
                if ((synd_vec[0] != 0) || (synd_vec[1] != 0) || (synd
_vec[2] != 0) || (synd_vec[3] != 0))
                    ++N;
                break;
            }
        }
        DataRow dr = results.NewRow();
        dr[0] = i;
        dr[1] = C;
        dr[2] = N;
        dr[3] = (double)N / C;
        results.Rows.Add(dr);
    }
}

```

3. Полный текст программы

MainWindow.xaml.cs:

```

using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.Data;
using System.Windows;

namespace DZ_ST
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        int[] g_pol = new int[5] { 1, 1, 0, 0, 1 };
        DataTable results;

        public MainWindow()
        {

```

```

        InitializeComponent();
    }

    public int[] Coding(int[] vec)
    {
        int[] c_vec = new int[15];
        for (int i = 4; i < 15; ++i)
            c_vec[i] = vec[i - 4];
        for (int i = 14; i > 3; --i)
        {
            if (c_vec[i] == 0)
                continue;
            for (int k = 0; k < 5; ++k)
                c_vec[i - k] = c_vec[i - k] ^ g_pol[4 - k];
        }
        for (int i = 4; i < 15; ++i)
            c_vec[i] = vec[i - 4];
        return c_vec;
    }

    public int[] Error(int[] vec, int[] err_vec)
    {
        int[] a_vec = new int[15];
        for (int i = 0; i < 15; ++i)
            a_vec[i] = vec[i] ^ err_vec[i];
        return a_vec;
    }

    public int[] Decoding(int[] vec)
    {
        int[] tmp = (int[])vec.Clone();
        for (int i = 14; i > 3; --i)
        {
            if (tmp[i] == 0)
                continue;
            for (int k = 0; k < 5; ++k)
                tmp[i - k] = tmp[i - k] ^ g_pol[4 - k];
        }
        int[] synd = new int[4];
        for (int i = 0; i < 4; ++i)
            synd[i] = tmp[i];
        return synd;
    }

    private void Button_Click_1(object sender, RoutedEventArgs e)
    {
        if (textbox1.Text.Length != 11)
        {
            MessageBox.Show("Длина информационного вектора должна быть равна
11", "Ошибка");
        }
    }

```

```

        button3.RaiseEvent(new RoutedEventArgs(System.Windows.Controls.Primitives.ButtonBase.ClickEvent));
        textbox1.Focus();
        return;
    }

    int[] i_vec = new int[11];

    bool flg = false;
    for (int i = 0; i < 11; ++i)
    {
        if (textbox1.Text[10 - i] != '1' && textbox1.Text[10 - i] != '0')
            flg = true;
        i_vec[i] = Convert.ToInt32(textbox1.Text[10 - i]) - 48;
    }
    if (flg)
    {
        MessageBox.Show("Проверьте правильность ввода информационного вектора (информационный вектор содержит только 1 и 0)", "Ошибка");
        button3.RaiseEvent(new RoutedEventArgs(System.Windows.Controls.Primitives.ButtonBase.ClickEvent));
        textbox1.Focus();
        return;
    }

    int[] c_vec = Coding(i_vec);

    label1.Content = null;
    for (int i = 14; i >= 0; --i)
        label1.Content += c_vec[i].ToString();

    results = new DataTable { TableName = "Результаты вычисления обнаруживающей способности циклического кода" };
    results.Columns.Add("Кратность ошибки, i", Type.GetType("System.Int32"));
    results.Columns.Add("Общее число ошибок, C\u2071\u2099", Type.GetType("System.Int32"));
    results.Columns.Add("Число обнаруженных ошибок, N\u2080", Type.GetType("System.Int32"));
    results.Columns.Add("Обнаруживающая способность кода, C\u2080", Type.GetType("System.Double"));
    results.Columns[0].ReadOnly = true;
    results.Columns[1].ReadOnly = true;
    results.Columns[2].ReadOnly = true;
    results.Columns[3].ReadOnly = true;
    for (int i = 1; i < 16; ++i)
    {
        int C = 0;
        int N = 0;
        int[] err_vec = new int[15];
        int[] res = new int[i];
    }

```

```

Stack<int> stack = new Stack<int>();
stack.Push(0);
while (stack.Count > 0)
{
    int value = stack.Pop();
    int index = stack.Count;
    if (value - 1 >= 0)
        err_vec[value - 1] = 0;
    while (value < 15)
    {
        err_vec[value] = 1;
        res[index++] = value++;
        stack.Push(value);

        if (index == i)
        {
            int[] synd_vec = Decoding(Error(c_vec, err_vec));
            ++C;
            if ((synd_vec[0] != 0) || (synd_vec[1] != 0) || (synd
_vec[2] != 0) || (synd_vec[3] != 0))
                ++N;
            break;
        }
    }
    DataRow dr = results.NewRow();
    dr[0] = i;
    dr[1] = C;
    dr[2] = N;
    dr[3] = (double)N / C;
    results.Rows.Add(dr);
}
resultsGrid.ItemsSource = results.AsDataView();
}

private void Button_Click_2(object sender, RoutedEventArgs e)
{
    SaveFileDialog sfd = new SaveFileDialog
    {
        FileName = "Результаты",
        DefaultExt = ".xml",
        Filter = "XML Files|*.xml"
    };
    if (sfd.ShowDialog() == true)
        results.WriteXml(sfd.FileName);
}

private void textbox1_PreviewTextInput(object sender, System.Windows.Input
t.TextCompositionEventArgs e)
{
    if ((e.Text != "0") && (e.Text != "1"))

```

```

        e.Handled = true;
    }

    private void Button_Click_3(object sender, RoutedEventArgs e)
    {
        label1.Content = null;
        resultsGrid.ItemsSource = null;
    }
}

```

MainWindow.xaml:

```

<Window x:Class="DZ_ST.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Title="MainWindow" Height="540" Width="745" MinHeight="305" MinWidth="315"
">
    <Grid>
        <Grid HorizontalAlignment="Left" Height="26" Margin="10,10,0,0" VerticalAlignment="Top" Width="282">
            <TextBox x:Name="textbox1" MaxLength="11" HorizontalAlignment="Left" Margin="162,0,0,0" Text="11111010001" TextWrapping="Wrap" VerticalAlignment="Center" Width="120" PreviewTextInput="textbox1_PreviewTextInput"/>
            <Label Content="Информационный вектор:" HorizontalAlignment="Left" VerticalAlignment="Center"/>
        </Grid>
        <Button HorizontalAlignment="Left" Margin="10,41,0,0" VerticalAlignment="Top" Click="Button_Click_1" Height="39" Width="282">
            <TextBlock TextAlignment="Center">Вычисление обнаруживающей<LineBreak/>способности циклического кода</TextBlock>
        </Button>
        <Grid Margin="10,116,10,42">
            <DataGrid x:Name="resultsGrid" CanUserAddRows="False" CanUserDeleteRows="False" Margin="0,31,0,0"/>
            <Label Content="Результаты:" HorizontalAlignment="Left" VerticalAlignment="Top"/>
        </Grid>
        <Grid HorizontalAlignment="Left" Height="26" VerticalAlignment="Top" Width="282" Margin="10,85,0,0">
            <Label Content="Циклический [15,11]-код:" HorizontalAlignment="Left" VerticalAlignment="Center"/>
            <Label x:Name="label1" HorizontalAlignment="Left" Margin="153,0,0,0" VerticalAlignment="Center"/>
        </Grid>
        <Button Content="Сохранить результаты" HorizontalAlignment="Right" Margin="0,0,10,10" VerticalAlignment="Bottom" Height="27" Click="Button_Click_2" Width="135"/>
    </Grid>
</Window>

```



```

        <Button x:Name="button3" Content="Очистить" HorizontalAlignment="Left" Margin="10,0,0,10" VerticalAlignment="Bottom" Height="27" Width="66" Click="Button_Click_3"/>
    </Grid>
</Window>

```

4. Заполненная программно таблица результатов

Скриншот результата выполнения программы:

MainWindow

Информационный вектор: 11111010001

Вычисление обнаруживающей способности циклического кода

Циклический [15,11]-код: 111110100010100

Результаты:

Кратность ошибки, i	Общее число ошибок, C_n^i	Число обнаруженных ошибок, N_o	Обнаруживающая способность кода, C_o
1	15	15	1
2	105	105	1
3	455	420	0.9230769230769231
4	1365	1260	0.9230769230769231
5	3003	2835	0.9440559440559441
6	5005	4725	0.9440559440559441
7	6435	6000	0.9324009324009324
8	6435	6000	0.9324009324009324
9	5005	4725	0.9440559440559441
10	3003	2835	0.9440559440559441
11	1365	1260	0.9230769230769231
12	455	420	0.9230769230769231
13	105	105	1
14	15	15	1
15	1	0	0

Очистить

Сохранить результаты

Таблица результатов:

i	C_n^i	N_0	C_0
1	15	15	1
2	105	105	1
3	455	420	0.923076923
4	1365	1260	0.923076923
5	3003	2835	0.944055944
6	5005	4725	0.944055944
7	6435	6000	0.932400932
8	6435	6000	0.932400932
9	5005	4725	0.944055944
10	3003	2835	0.944055944
11	1365	1260	0.923076923
12	455	420	0.923076923
13	105	105	1
14	15	15	1
15	1	0	0

5. Выводы

По таблице результатов видно, что только при кратности ошибки равной 15 обнаруживающая способность циклического [15,11]-кода равна 0 и он не способен обнаружить ошибку. При кратностях ошибки равными 1, 2, 13 и 14 обнаруживающая способность кода равна 1 и ошибку всегда можно обнаружить. При остальных кратностях ошибки обнаруживающая способность кода близка к 1 и ошибку почти всегда можно обнаружить. Таким образом, циклический [15,11]-код подходит для обнаружения ошибок кратностью до 14.

6. Список используемой литературы

1. Телекоммуникации и сети. / В.А.Галкин, Ю.А.Григорьев Учебное пособие для вузов.-М.:Из-во МГТУ им.Н.Э.Баумана 2003 г.
2. Методическое пособие по выполнению домашнего задания по дисциплине «Сети и телекоммуникации» / Галкин В.А. М.: МГТУ им. Н.Э.Баумана. 2018 г.
3. Конспект лекций по дисциплине “Сети и телекоммуникации”. М.: МГТУ им. Н.Э.Баумана. 2020 г. (рукопись)