Министерство образования Российской Федерации

**КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ**

**Кафедра** ВТ и АСУ

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

**по дисциплине ‘‘**Теория информации и сигналов’’

**на тему** **‘‘**Коды БЧХ: обнаружение и исправление ошибок’’

**Выполнил студент \_Байдин Константин Николаевич\_\_\_\_\_\_\_**

**группы \_\_\_\_\_\_13-КБ-ПР1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Допущен к защите\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Нормоконтролер\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Защищен**\_\_\_\_\_\_\_\_\_\_\_\_\_ **Оценка\_\_\_\_\_\_\_\_\_\_**\_\_\_\_\_

(дата)

**Члены комиссии\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись,дата,расшифровка подписи)

Министерство образования Российской Федерации

**КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ**

**Кафедра** ВТ и АСУ

**Утверждаю:**

**Зав. кафедрой ВТ и АСУ**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ЗАДАНИЕ**

**на курсовое проектирование**

**Студенту: Байдину Константину Николаевичу**

**группы 13-КБ-ПР1**

**2** **курса**

**института КСиИБ**

**специальности программная инженерия (231000)**

**Тема проекта:** Коды БЧХ: обнаружение и исправление ошибок

**Содержание задания:** Реализация алгоритмов кодирования и декодирования информации кодами БЧХ и алгоритма обнаружения и исправления ошибок в кодовых комбинациях.

**Объем работы:**.

а) пояснительная записка к проекту 12 стр.

б) 1 программа .

**Рекомендуемая литература:** Прокис Дж. Цифровая связь. Пер. с англ./ Под ред. Д.Д. Кловского. – М.: Радио и связь. 2000. - 800 с.

**Срок выполнения проекта: с “\_\_\_”\_\_\_\_\_\_\_\_ по “\_\_\_”\_\_\_\_\_\_20\_\_\_ г.**

**Срок защиты: “\_\_\_”\_\_\_\_\_\_\_20\_\_\_ г.**

**Дата выдачи задания: “\_\_\_”\_\_\_\_\_\_\_20\_\_\_ г.**

**Дата сдачи проекта на кафедру: “\_\_\_”\_\_\_\_\_\_\_20\_\_\_ г.**

**Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(подпись,ф.и.о.,звание,степень)

**Задание принял студент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

**РЕФЕРАТ**

Пояснительная записка данной работы содержит 12 страниц и 3 рисунка.

Ключевые слова: БЧХ, циклические коды, информационные разряды, дополнительные разряды, обнаружение ошибок, искажение кодовой комбинации, кратность гарантийно обнаруживаемых ошибок, образующий полином, действия над полиномами, исправление ошибок.

Для поиска ошибок при передаче сигналов, каждую кодовую комбинацию k-элементного безызбыточного кода, выраженной в виде многочлена, перемножаем на образующий полином с приведением подобных членов по модулю два. Следовательно, в случае отсутствия ошибок любая разрешенная кодовая комбинация циклического кода должна разделиться на образующий полином без остатка. Задачей данной работы является реализация обнаружения и исправления ошибок с помощью кодов БЧХ.

# 

# **Введение**

В настоящее объемы передаваемой информации постоянно растут. Растут и требования к скорости передачи информации. С увеличением скорости передачи информации увеличивается вероятность появления ошибок при передаче. Стали разрабатываться универсальные методы обнаружения ошибок, одним из которых является метод циклических кодов, который обеспечивает надежность передачи данных.

В работе рассматриваются циклические коды и, конкретно, коды Боуза - Чоудхури – Хоквингема. Реализованы алгоритмы кодирования и декодирования кодами БЧХ, и алгоритм исправления ошибок.

Оглавление

[Введение 3](#_Toc418727765)

[1 Коды Боуза - Чоудхури – Хоквингема 5](#_Toc418727766)

[1.1 Формирование кодового слова 5](#_Toc418727767)

[1.2 Выбор образующего полинома 6](#_Toc418727768)

[1.3 Обнаружение и исправление ошибок 6](#_Toc418727769)

[2 Практическая часть 7](#_Toc418727770)

[2.1 Задаваемые для расчета технические параметры 7](#_Toc418727771)

[2.2 Описание алгоритма 7](#_Toc418727772)

[2.3 Метод реализации алгоритма 7](#_Toc418727773)

[3 Требование к программному изделию 8](#_Toc418727774)

[3.1 Требование к функциональным характеристикам 8](#_Toc418727775)

[3.2 Контроль входной и выходной информации 8](#_Toc418727776)

[4 Руководство пользователя 8](#_Toc418727777)

[4.1 Интерфейс пользователя 8](#_Toc418727778)

[4.2 Вывод результатов 9](#_Toc418727779)

[5 Тестирование программы 10](#_Toc418727780)

[Заключение 11](#_Toc418727781)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 12](#_Toc418727782)

Коды Боуза - Чоудхури – Хоквингема

  Коды Боуза-Чоудхури-Хоквингема (БЧХ) относятся к циклическим помехоустойчивым кодам. Циклические коды широко применяются при построении устройств вычислительной техники и вычислительных сетей. БЧХ коды особенны тем, что позволяют точно задать кратность распознаваемой или исправляемой ошибки и длину кодового слова.

* 1. Формирование кодового слова

Для того, чтобы закодировать сообщение, его представляют в двоичном виде, из которого переводят сообщение в полиномиальную форму. Делается это следующим образом: биты сообщения являются коэффициентами при членах полинома. Например, сообщение 10011 представляется многочленом . При этом все действия над этим полиномом нужно проводить по модулю два. Например,

Для кодирования сообщений выбирается образующий полином G(x). Образующий полином равен неприводимому минимальному полиному или произведению нескольких неприводимых минимальных полиномов. Неприводимый минимальный полином – это полином, делящийся без остатка на себя и на единицу.

Далее код может формироваться двумя способами: с образованием систематического кода и с образованием несистематического кода.

В систематическом коде первые k бит являются копией кодируемого сообщения. А остальные (n-k) бит являются проверочными. Образуется этот код следующим образом: кодируемый полином P(x) умножается на единичный полином степени, равной степени образующего полинома и получается полином Q(x). После этого вычисляется остаток R(x) от деления получившегося полинома Q(x) на образующий полином G(x). Итоговый код равен Q(x) + R(x). Для декодирования просто извлекаются первые k бит.

В несистематическом коде проверочные биты смешаны с информационными. Для получения несистематического кода кодируемый полином P(x) умножается на образующий полином G(x). Для декодирования код делится на образующий полином G(x).

* 1. Выбор образующего полинома

Пусть требуется исправлять t ошибок и длина кодовой комбинации должна быть n. Тогда образующий полином должен иметь степень k=[log2(n+1)]\*S и быть произведением t минимальных неприводимых полиномов. При этом, старший из них должен иметь порядок p = 2 \* S – 1, и все полиномы должны быть нечётными. Минимальные неприводимые полиномы определены и сведены в таблицу минимальных неприводимых полиномов. Также существуют уже рассчитанные таблицы образующих полиномов, в которых можно выбрать образующий полином по длине кодовой комбинации n, количеству исправляемых ошибок t и количеству информационных бит k.

* 1. Обнаружение и исправление ошибок

Обнаружение ошибки всегда происходит одинаково: если остаток R(x) от деления кодовой комбинации C(x) на образующий полином G(x) не равен нулю, то в кодовой комбинации появилась ошибка. Остаток R(x) также называется синдромом ошибки.

Существует несколько методов исправления ошибок. При несистематическом коде строится таблица из векторов ошибок, в которых единицами обозначаются ошибочные разряды, а нулями – верные. Каждому вектору ставится в соответствие определённый синдром. Для исправления ошибки нужно по синдрому найти в таблице нужный вектор ошибок, затем сложить его по модулю два с кодовой комбинацией С(х). Для нахождения векторов ошибок существует несколько алгоритмов, например, алгоритм Берлекэмпа — Мэсси.

При систематическом коде может применяться следующий алгоритм, который является упрощением предыдущего алгоритма с учётом строения систематических кодов:

1. Получить остаток R(x) при делении кодовой комбинации C(x) на образующий полином G(x).
2. Подсчитать количество единиц в остатке.
3. Если количество единиц в остатке больше, чем количество t исправляемых данным образующим многочленом ошибок, то произвести циклический сдвиг кодовой комбинации C(x) влево на один бит. Перейти на шаг 1.
4. Прибавить по модулю два к кодовой комбинации C(x) остаток R(x).
5. Выполнить циклический сдвиг кодовой комбинации C(x) вправо на столько же бит, на сколько она перед этим была сдвинута влево.
6. Практическая часть
   1. Задаваемые для расчета технические параметры

Исходными данными для кодирования информации БЧХ-кодом и обнаружения и исправления ошибок являются:

1. Количество исправляемых ошибок
2. Длина кодовой комбинации
3. Информационный полином P(x)
   1. Описание алгоритма

Пусть информационный полином P(x) был закодирован образующим полиномом G(x) и была получена кодовая комбинация C(x). Если при делении кодовой комбинации на образующий полином получился ненулевой остаток, то значит существует ошибка, которую можно исправить следующим алгоритмом:

1. Получить остаток R(x) при делении кодовой комбинации C(x) на образующий полином G(x).
2. Подсчитать количество единиц в остатке.
3. Если количество единиц в остатке больше, чем количество t исправляемых данным образующим многочленом ошибок, то произвести циклический сдвиг кодовой комбинации C(x) влево на один бит. Перейти на шаг 1.
4. Прибавить по модулю два к кодовой комбинации C(x) остаток R(x).
5. Выполнить циклический сдвиг кодовой комбинации C(x) вправо на столько же бит, на сколько она перед этим была сдвинута влево.
   1. Метод реализации алгоритма

Проблема реализации алгоритма заключается в реализации математических операций над многочленами, которые являются очень объёмными. В данной работе это было учтено и был создан отдельный класс для многочленов. Таким образом, реализация алгоритма была отделена от визуализации WindowsForms. И в дальнейшем может быть организована в виде отдельной библиотеки для кодирования и декодирования информации. При разработке использовался принцип объектно-ориентированного программирования - инкапсуляция.

1. Требование к программному изделию
   1. Требование к функциональным характеристикам

Разработанная программа пригодна для кодирования информации одним из трёх предлагаемых образующих многочленов с разными показателями исправления ошибок и длиной кодовой комбинации. Программа обнаруживает и исправляет ошибки, вводимые пользователем в кодовую комбинацию.

* 1. Контроль входной и выходной информации

Принимая входную информацию от пользователя, программа анализирует вводимые пользователем данные. При поступлении ошибочной информации или информации, неудовлетворяющей ограничительным критериям, выводится диалоговое окно с сообщением об ошибке и информацией о том, каким критериям должны соответствовать входные данные. После этого, программа снова готова к вводу пользователя.

1. Руководство пользователя
   1. Интерфейс пользователя

На рисунке 1 показан внешний вид программы при запуске.

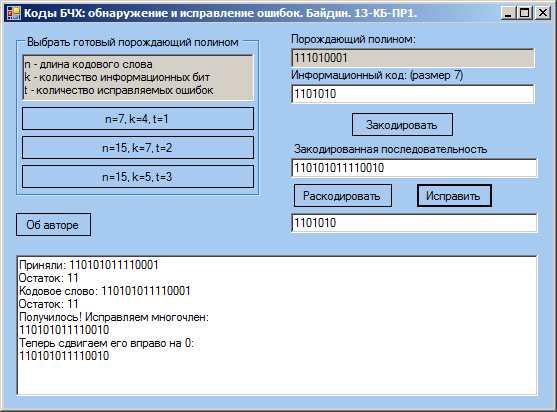


Рисунок 1. Внешний вид рабочего окна программы

В верхнем левом углу пользователю предлагается выбрать один из трёх образующих полиномов. Над списком полиномов располагается расшифровка символов n, k и t. После выбора полинома, справа изменится поле с порождающим полиномом, а поле с информационным кодом заполнится тестовой последовательностью необходимой длины. При желании, пользователь может сам написать информационный код. По нажатию кнопки поле закодированной последовательности заполняется получившейся кодовой комбинацией. Если информационный код был не той длины, которую требует порождающий многочлен, то пользователю будет выведено сообщение, показанное на рисунке 2.

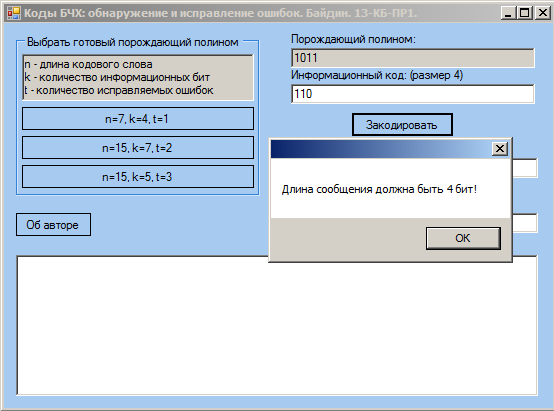


Рисунок 2. Вид окна с предупреждением об ошибке ввода.

* 1. Вывод результатов

После нажатия кнопки «Раскодировать», если кодовая комбинация не была изменена, то в поле ниже появится раскодированное информационное сообщение. Но пользователь может сам внести ошибки в кодовую комбинацию. Тогда появится искажённая информация красным цветом. В таком случае можно нажать кнопку «Исправить» и алгоритм обнаружения и исправления ошибок исправит кодовую комбинацию и раскодирует её. Так же алгоритм обнаружения и исправления ошибок будет выводить этапы алгоритма в нижнее текстовое поле. Пример показан на рисунке 3.

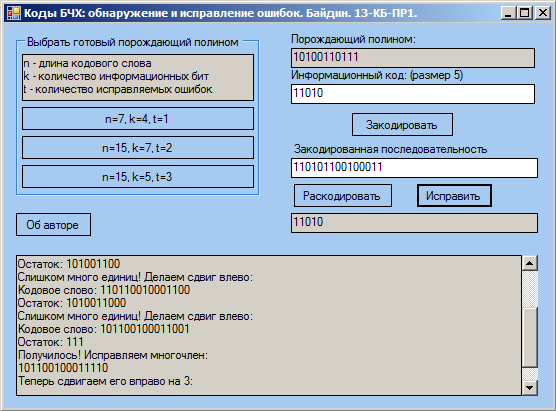


Рисунок 3. Вывод сообщений на каждом шаге алгоритма исправления ошибок.

1. Тестирование программы

Для тестирования разработанного программного продукта возьмем значения, введенные по умолчанию и произведём ручной просчёт.

Порождающий полином G(x) = 1011. Информационный код P(x) = 1010. Параметры порождающего полинома: n = 7, k = 4, t = 1.

Построим систематизированный код:

1. Степень G(x) равна 3, поэтому P(x) \* 1000 = 1010 \* 1000 = 1010000
2. Остаток R(x) = 1010000 mod G(x) = 1010000 mod 1011 = 11
3. Кодовая комбинация C(x) = 1010000 + R(x) = 1010011

Теперь для проверки нахождения и исправления ошибок внесём в кодовую комбинацию ошибку: С(x) = 0010011.

Производим декодирование:

1. R(x) := C(x) mod G(x) = 101
2. Количество единиц в остатке R(x) больше, чем t. Производим циклический сдвиг влево:
3. C(x) := 0100110
4. R(x) := C(x) mod G(x) = 1
5. Количество единиц в остатке R(x) равно t. Произведём сложение по модулю 2:
6. C(x) := C(x) + 1 = 0100111
7. Производим циклический сдвиг C(x) обратно вправо:
8. C(x) := 1010011

# **Заключение**

Нужно понимать, что кодирование позволяет обнаруживать или исправлять только определённое количество ошибок. Количество обнаруживаемых или исправляемых ошибок можно постоянно повышать, но всегда будет вероятность того, что произойдет ошибка большей кратности, чем может обнаружить или исправить данный метод шифрования.

В данной работе были детально разобраны принципы образования кодовых комбинаций циклических кодов и выбора образующего полинома, алгоритм обнаружения и исправления ошибок разной кратности с различными образующими полиномами.

Разработанная программа предоставляет пользователю возможность опробовать возможности БЧХ кодов и позволяет пользователю самому вносить помехи в кодовую комбинацию, чтобы можно было лучше понять особенности и ограничения этих алгоритмов.

Разработанное приложение может быть сжато в отдельную независимую библиотеку и предоставлять возможности по кодированию и декодированию информации любыми пользовательскими образующими полиномами и по обнаружению и исправлению ошибок.

# 

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Власенко А.В., Ключко В.И. Теория информации и сигналов. Учебное

пособие / Краснодар: Изд-во КубГТУ, 2003.- 97 с.

2. Теория передачи сигналов: Учебник для вузов / А.Г.Зюко, Д.Д.Кловский,

М.В.Назаров, Л.М.Финк. - 2-е изд., перераб. и доп. - М.: Радио и связь, 1986. - 304 с.: ил.

3. Прокис Дж. Цифровая связь. Пер. с англ./ Под ред. Д.Д. Кловского. – М.: Радио и связь. 2000. - 800 с.

4. Hank Wallace. Error Detection and Correction Using the BCH Code / Virginia: Hank Wallace, 2001.- 21 c.

5. Shu Lin and Daniel J. Costello, Jr. Error Control Coding -2nd ed. / Englewood Cliffs, New Jersey: Prentice Hall, 1983.- 618 c.

# **ПРИЛОЖЕНИЕ А**

Исходный код программы.

class Polynom

{

string p;

public Polynom(string p)

{

this.p = p;

}

public static Polynom operator \* (Polynom poly1, Polynom poly2)

{

string p1 = poly1, p2 = poly2;

int s1 = p1.Count();

int s2 = p2.Count();

int sr = p1.Count() + p2.Count() - 1;

StringBuilder result = new StringBuilder();

result.Length = sr;

result.Replace('\0', '0');

for (int i = 0; i < s1; i++)

{

for (int j = 0; j < s2; j++)

{

char mul = Mul(p1[s1-i-1], p2[s2-j-1]);

int f = i + j;

result[sr-f-1] = Plus(result[sr-f-1], mul);

}

}

return new Polynom(result.ToString());

}

public static Polynom operator / (Polynom poly1, Polynom poly2)

{

string p1 = poly1, p2 = poly2;

int s1 = p1.Count();

int s2 = p2.Count();

StringBuilder result = new StringBuilder(p1);

Polynom realResult = new Polynom("");

while (result.Length >= s2)

{

int f = result.Length - (s2-1);

StringBuilder a = new StringBuilder();

a.Length = f;

a.Replace('\0', '0');

a[0] = '1';

Polynom ap = new Polynom(a.ToString());

realResult = realResult + ap;

Polynom m = poly2 \* ap;

result = new StringBuilder(new Polynom(result.ToString()) + m);

}

return realResult;

}

public static Polynom Mod (Polynom poly1, Polynom poly2)

{

string p1 = poly1, p2 = poly2;

int s1 = p1.Count();

int s2 = p2.Count();

StringBuilder result = new StringBuilder(p1);

Polynom realResult = new Polynom("");

while (result.Length >= s2)

{

int f = result.Length - (s2 - 1);

StringBuilder a = new StringBuilder();

a.Length = f;

a.Replace('\0', '0');

a[0] = '1';

Polynom ap = new Polynom(a.ToString());

realResult = realResult + ap;

Polynom m = poly2 \* ap;

result = new StringBuilder(new Polynom(result.ToString()) + m);

}

string sresult = result.ToString();

if (sresult.Length == 0) sresult = "0";

return new Polynom(result.ToString());

}

public static Polynom operator + (Polynom poly1, Polynom poly2)

{

string p1 = poly1, p2 = poly2;

int s1 = p1.Count();

int s2 = p2.Count();

if (s1 > s2)

{

p2 = MakeString(s1 - s2) + p2;

s2 = s1;

}

if (s2 > s1)

{

p1 = MakeString(s2 - s1) + p1;

s1 = s2;

}

StringBuilder r = new StringBuilder();

r.Length = s1;

for (int i = 0; i < s1; i++)

{

r[i] = Plus(p1[i], p2[i]);

}

int index = 0;

for (; index < s1; index++)

{

if (r[index] != '0') break;

}

r.Remove(0, index);

return new Polynom(r.ToString());

}

public static string MakeString(int len)

{

StringBuilder r = new StringBuilder();

r.Length = len;

r.Replace('\0', '0');

return r.ToString();

}

static char Plus(char x, char y)

{

if (x != y) return '1';

else return '0';

}

static char Mul(char x, char y)

{

if (x == '0' || y == '0') return '0';

else return '1';

}

public static implicit operator string(Polynom poly)

{

return poly.p;

}

public static int C(int n, int k)

{

return Enumerable.Range(1, k).Aggregate(n, (acc, next) => acc \*= n - k);

}

}

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

Dictionary<string, Polynom> errMap;

errMap = new Dictionary<string, Polynom>();

errMap.Add("0", new Polynom("0000000"));

errMap.Add("101", new Polynom("1000000"));

errMap.Add("111", new Polynom("0100000"));

errMap.Add("110", new Polynom("0010000"));

errMap.Add("11", new Polynom("0001000"));

errMap.Add("100", new Polynom("0000100"));

errMap.Add("10", new Polynom("0000010"));

errMap.Add("1", new Polynom("0000001"));

maps.Add(errMap);

}

List<Dictionary<string, Polynom>> maps = new List<Dictionary<string, Polynom>>();

int mapIndex = 0;

int t = 1;

int k = 4;

int n = 7;

private void bStart\_Click(object sender, EventArgs e)

{

Polynom p1 = new Polynom(tbP1.Text);

Polynom p2 = new Polynom(tbP2.Text);

string s1 = p1;

string s2 = p2;

StringBuilder s3 = new StringBuilder(Polynom.MakeString(s2.Length));

s3[0] = '1';

p1 = p1 \* new Polynom(s3.ToString());

p1 += Polynom.Mod(p1, p2);

tbCoded.Text = p1;

if (tbP1.Text.Length != k)

{

MessageBox.Show("Длина сообщения должна быть " + k + " бит!");

return;

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (tbCoded.Text.Length != n)

{

MessageBox.Show("Кодовое слово должны быть длиной " + n + " бит!");

return;

}

tbLog.Clear();

Polynom p1 = new Polynom(tbCoded.Text);

Polynom p2 = new Polynom(tbP2.Text);

Polynom remainder = Polynom.Mod(p1, p2);

if (remainder == "")

{

tbResult.ForeColor = Color.Black;

}

else

{

tbResult.ForeColor = Color.Red;

}

string s1 = p1;

tbResult.Text = s1.Substring(0, k);

}

int OnesCount(string s)

{

int sum = 0;

foreach (char c in s)

{

if (c == '1') sum++;

}

return sum;

}

private void bCorrect\_Click(object sender, EventArgs e)

{

if (tbCoded.Text.Length != n)

{

MessageBox.Show("Кодовое слово должны быть длиной " + n + " бит!");

return;

}

tbLog.Clear();

Polynom p1 = new Polynom(tbCoded.Text);

int n1 = ((string)p1).Length;

Polynom p2 = new Polynom(tbP2.Text);

tbResult.ForeColor = Color.Black;

{

string remainder = Polynom.Mod(p1, p2);

if (remainder == "") remainder = "0";

tbLog.AppendText("Приняли: " + p1 + "\r\n");

tbLog.AppendText("Остаток: " + remainder + "\r\n");

}

int shifts = 0;

while (true)

{

Polynom polyRemainder = Polynom.Mod(p1, p2);;

string remainder = polyRemainder;

tbLog.AppendText("Кодовое слово: " + p1 + "\r\n");

tbLog.AppendText("Остаток: " + remainder + "\r\n");

if (OnesCount(remainder) > t)

{

tbLog.AppendText("Слишком много единиц! Делаем сдвиг влево:\r\n");

string s = p1;

s = s.Substring(1) + s[0];

p1 = new Polynom(s);

shifts++;

}

else

{

tbLog.AppendText("Получилось! Исправляем многочлен:\r\n");

p1 = p1 + polyRemainder;

tbLog.AppendText(p1 + "\r\n");

break;

}

}

tbLog.AppendText("Теперь сдвигаем его вправо на "+shifts+":\r\n");

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

{

string s = p1;

if (s.Length < n1)

{

while (s.Length < n1)

{

s = '0' + s;

}

}

s = s[s.Length-1] + s.Substring(0, s.Length-1);

p1 = new Polynom(s);

}

tbLog.AppendText(p1 + "\r\n");

tbCoded.Text = p1;

tbResult.Text = ((string)p1).Substring(0, k);

/\*foreach (var elem in maps[mapIndex])

{

tbLog.AppendText(elem.Value + " " + elem.Key + "\r\n");

}

Polynom result = (p1 + maps[mapIndex][remainder]) / p2;

tbResult.Text = result;\*/

}

private void bChoose1\_Click(object sender, EventArgs e)

{

tbP2.Text = "1011";

label1.Text = "Информационный код: (размер 4)";

tbP1.Text = "1101";

mapIndex = 0;

t = 1;

k = 4;

n = 7;

}

private void bChoose2\_Click(object sender, EventArgs e)

{

tbP2.Text = "111010001";

label1.Text = "Информационный код: (размер 7)";

tbP1.Text = "1101010";

mapIndex = 1;

t = 2;

k = 7;

n = 15;

}

private void bChoose3\_Click(object sender, EventArgs e)

{

tbP2.Text = "10100110111";

label1.Text = "Информационный код: (размер 5)";

tbP1.Text = "11010";

mapIndex = 2;

t = 3;

k = 5;

n = 15;

}

private void bAbout\_Click(object sender, EventArgs e)

{

MessageBox.Show("Эта программа разработана в ходе курсовой работы по теме"+

"\r\n\"Коды БЧХ: обнаружение и исправление ошибок\"."

+"\r\nРаботу выполнил: Байдин Константин,\r\nстудентом группы 13-КБ-ПР1");

}

}