Отчет

По специальности Информационная Безопасность

Подготовил

Студент группы 22206

Меньшаков Артём

Лабораторная работа №2

Защитное кодирование

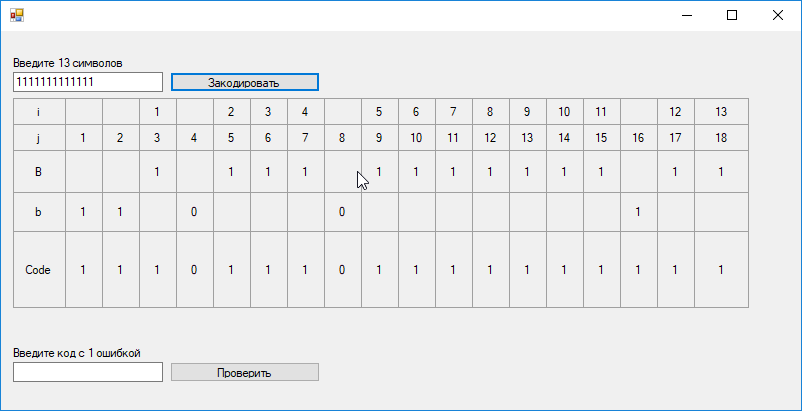
по методу Хэмминга

Описание метода

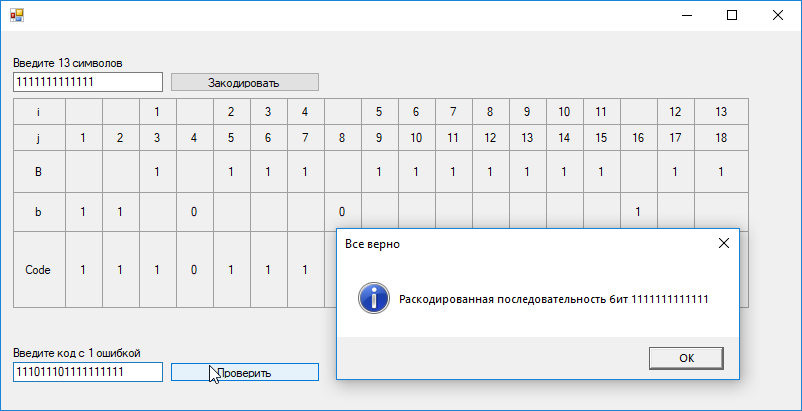
Суть метода рассечения-разнесения состоит в том, при использовании этого метода вся последовательность данных разбивается на блоки фиксированной длины n. При кодировании длина фиксированного блока увеличивается до такой длины N, чтобы дополнительные биты позволяли сформировать число от нуля до N – для фиксации места ошибки (позиции ошибочного бита блока).

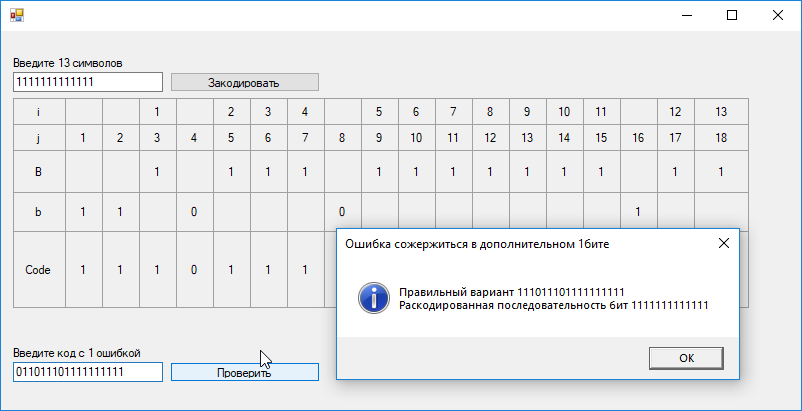
По своему заданию я реализовал кодирование блока длиной 13. После кодирования длина составляет 18.

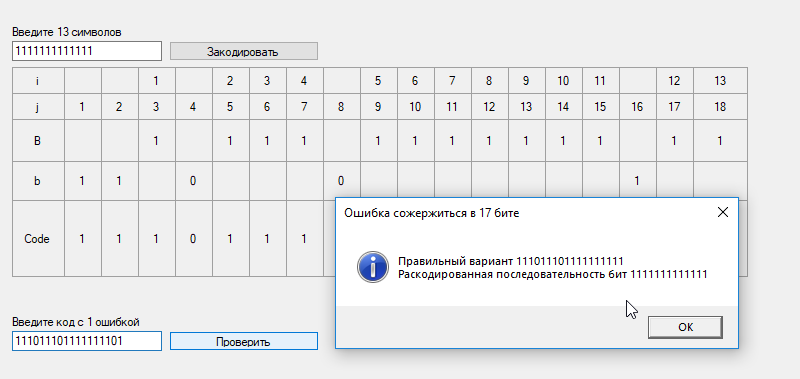
Пример



И расшифровка с проверкой







Сам код

// метод возвращает закодированную массив

private int[] encode(string text)

{

char[] ch = text.ToCharArray();

int[] full\_array = new int[18];

// заполняем массив не трогая 0,1,3,7,15 места(биты)

int k = 0;

for (int i = 0; i < full\_array.Length; i++)

{

if ((i + 1) != 1 && (i + 1) != 2 && (i + 1) != 4 && (i + 1) != 8 && (i + 1) != 16)

{

full\_array[i] = int.Parse(ch[k].ToString());

k++;

}

}

// вычисляем сколько едениц контролирует 1 бит

// исходя из этого определяем чему равен этот бит 0 или 1

int sum = 0;

for (int i = 0; i < full\_array.Length; i += 2)

{

if (full\_array[i] == 1)

{

sum++;

}

}

if (sum % 2 == 0 && sum != 0) full\_array[0] = 1;

else full\_array[0] = 0;

// вычисляем сколько едениц контролирует 2 бит

// исходя из этого определяем чему равен этот бит 0 или 1

sum = 0;

for (int i = 1; i < full\_array.Length; i += 4)

{

if (full\_array[i] == 1)

{

sum++;

}

if (i + 1 < full\_array.Length && full\_array[i + 1] == 1)

{

sum++;

}

}

if (sum % 2 == 0 && sum != 0) full\_array[1] = 1;

else full\_array[1] = 0;

// вычисляем сколько едениц контролирует 4 бит

// исходя из этого определяем чему равен этот бит 0 или 1

sum = 0;

for (int i = 3; i < full\_array.Length; i += 8)

{

for (int j = i; j < i + 4 && j < full\_array.Length; j++)

{

if (full\_array[j] == 1)

{

sum++;

}

}

}

if (sum % 2 == 0 && sum != 0) full\_array[3] = 1;

else full\_array[3] = 0;

// вычисляем сколько едениц контролирует 8 бит

// исходя из этого определяем чему равен этот бит 0 или 1

sum = 0;

for (int i = 7; i < full\_array.Length; i += 16)

{

for (int j = i; j < i + 8 && j < full\_array.Length; j++)

{

if (full\_array[j] == 1)

{

sum++;

}

}

}

if (sum % 2 == 0 && sum != 0) full\_array[7] = 1;

else full\_array[7] = 0;

// вычисляем сколько едениц контролирует 32 бит

// исходя из этого определяем чему равен этот бит 0 или 1

sum = 0;

for (int i = 15; i < full\_array.Length; i += 32)

{

for (int j = i; (j < i + 16) && j < full\_array.Length; j++)

{

if (full\_array[j] == 1)

{

sum++;

}

}

}

if (sum % 2 == 0 && sum != 0) full\_array[15] = 1;

else full\_array[15] = 0;

// возвращаем закодированный массив

return full\_array;

}

И функция декодирования

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

{

int k = 0;

int[] ar = new int[13];

// разделяем биты переданного сообщения и котрольные биты

for (int i = 0; i < textBox2.Text.Length; i++)

{

if ((i + 1) != 1 && (i + 1) != 2 && (i + 1) != 4 && (i + 1) != 8 && (i + 1) != 16)

{

ar[k] = int.Parse(textBox2.Text[i].ToString());

k++;

}

}

string s = "";

foreach (int i in ar)

{

s += i.ToString();

}

// расчиываем контрольные биты

int[] array\_rith = encode(s);

int[] array\_wrong = new int[18];

for (int i = 0; i < 18; i++)

{

array\_wrong[i] = int.Parse(textBox2.Text[i].ToString());

}

// находим сумму индексов несовпадающих контрольных бит

int sum = sum\_wrong\_bite(array\_rith, array\_wrong);

// если совпадений не найденно

if (sum == 0)

{

MessageBox.Show(

"Раскодированная последовательность бит " + rith(array\_wrong),

"Все верно",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

// если ошибка не в контрольных битах

else if (sum - 1 != 0 && sum - 1 != 1 && sum - 1 != 3 && sum - 1 != 7 && sum - 1 != 15)

{

array\_wrong[sum - 1] = invers(array\_wrong[sum - 1]);

s = "";

foreach (int i in array\_wrong)

{

s += i.ToString();

}

MessageBox.Show(

"Правильный вариант " + s + "\n" +

"Раскодированная последовательность бит " + rith(array\_wrong),

"Ошибка сожержиться в " + sum + " бите",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

else

// если ошибка в контрольных битах

{

array\_wrong[sum - 1] = invers(array\_wrong[sum - 1]);

s = "";

foreach (int i in array\_wrong)

{

s += i.ToString();

}

MessageBox.Show(

"Правильный вариант " + s + "\n" + "Раскодированная последовательность бит " + rith(array\_wrong),

"Ошибка сожержиться в дополнительном " + sum + "бите",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

private string rith(int[] array\_wrong)

{

// разделяем биты переданного сообщения и котрольные биты

int k = 0;

int[] decode = new int[13];

for (int i = 0; i < array\_wrong.Length; i++)

{

if ((i + 1) != 1 && (i + 1) != 2 && (i + 1) != 4 && (i + 1) != 8 && (i + 1) != 16)

{

decode[k] = array\_wrong[i];

k++;

}

}

string s = "";

foreach (int i in decode)

{

s += i.ToString();

}

return s;

}

private int invers(int v)

{

return (v == 1) ? 0 : 1;

}

private int sum\_wrong\_bite(int[] array\_rith, int[] array\_wrong)

{

// сравниваем контрольные биты в полученном сообщении и вычесленном

// по сумме индексов несовпадающих контрольных бит

// можно определить место ошибки

int sum = 0;

for (int i = 0; i < 5; i++)

{

if (array\_rith[(int)Math.Pow(2, i) - 1] != array\_wrong[(int)Math.Pow(2, i) - 1])

sum += (int)Math.Pow(2, i);

}

return sum;

}

Все пункты для самопроверки программа прошла успешно.