МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность Информационные системы и технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4 НА ТЕМУ:**

**Избыточное кодирование данных в информационных системах. Код Хемминга**

Выполнила студентка 3 курса 1 группы

Пригодич Вера Валерьевна

Минск 2022

**Задание 1:** На основе информационного сообщения, представленного символами русского/английского алфавитов, служебными символами и цифрами, содержащегося в некотором текстовом файле сформировать информационное сообщение в двоичном виде; длина сообщения в бинарном виде должна быть не менее 16 символов.

Для выполнения задания был использован следующий код на языке C#:

public static string TextReader(string path)

{

string str;

using (StreamReader reader = new StreamReader(path))

{

str = reader.ReadToEnd().ToLower();

}

return str;

}

public static string ConvertToBinary(string str)

{

string binStr = null;

foreach (char c in str)

{

if (c != ' ')

{

binStr += Convert.ToString(c, 2).PadLeft(8, '0');

}

}

return binStr;

}

Листинг 1 – задание 1

**Задание 2:** Для полученного информационного слова построить проверочную матрицу Хемминга.

Для выполнения этого задания код, разработанный в предыдущем задании был дополнен кодом из предыдущей лабораторной работы, а также следующим кодом:

public static int Factorial(int n)

{

if (n == 1)

return 1;

else

return n \* Factorial(n - 1);

}

public static int NewtonBinom(int wt, int r)

{

return Factorial(r) / (Factorial(wt) \* Factorial(r - wt));

}

static public char[,] MatrixA(int r, int k)

{

int countOfColomn = 0;

int wt = 2;

char[,] matrixA = new char[r, k];

while (true)

{

int amountAllCombination = NewtonBinom(wt, r);

int rightCombination = 0;

int number = 3;

while (rightCombination != amountAllCombination && countOfColomn != k)

{

string binaryNumber = Convert.ToString(number, 2);

int binaryNumberLength = binaryNumber.Length;

//Счетчик единиц в бинарной строке

int unitCounter = 0;

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

{

if (binaryNumber[i] == '1')

unitCounter++;

}

if (unitCounter == wt)

{

countOfColomn++;

if (binaryNumberLength < r)

{

for (int j = 0; j < (r - binaryNumberLength); j++)

{

binaryNumber = "0" + binaryNumber;

}

}

for (int i1 = 0; i1 < r; i1++)

{

matrixA[i1, countOfColomn - 1] = binaryNumber[i1];

}

rightCombination++;

}

number++;

}

if (countOfColomn != k)

wt++;

else

break;

}

return matrixA;

}

static public char[,] MatrixI(int r)

{

char[,] matrixI = new char[r, r];

int k = 0;

for (int i = k; i < r; i++)

{

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

{

if (j == i - k)

matrixI[i, j] = '1';

else

matrixI[i, j] = '0';

}

}

return matrixI;

}

static public char[,] HammingMatrix(int r, int k)

{

char[,] hammingMatrix = new char[r, r + k];

char[,] matrixA = MatrixA(r, k);

char[,] matrixI = MatrixI(r);

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

{

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

{

hammingMatrix[j, i] = matrixA[j, i];

}

}

int i1 = 1;

for (int i = k; i < (k + r); i++)

{

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

{

hammingMatrix[j, i] = matrixI[j, i1 - 1];

}

i1++;

}

return hammingMatrix;

}

Листинг 2 – задание 2

**Задание 3:** Используя построенную матрицу вычислить избыточные символы (слово Xr).

Дополним разработанный в предыдущем задании код следующим методом:

static public string RedundancySymbols(string Xk, char[,] hammingMatrix, int r, int k)

{

string Xr = null;

int pairs = 0;

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

{

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

{

if (hammingMatrix[i, j] == '1' && Xk[j] == '1')

pairs++;

}

if (pairs % 2 == 0)

Xr += '0';

else

Xr += '1';

pairs = 0;

}

return Xr;

}

Листинг 3 – задание 3

**Задание 4:** Принять исходное слово со следующим числом ошибок: 0, 1, 2. Позиция ошибки определяется случайным образом.

Дополним разработанный в предыдущем задании код следующим методом:

static public string RandomError(string Xk, int amount)

{

char[] Yk = new char[Xk.Length];

Yk = Xk.ToArray<char>();

int previosPosition = 100;

int currentPosition;

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

{

Random random = new Random();

currentPosition = random.Next(0, Xk.Length);

while (currentPosition == previosPosition)

{

currentPosition = random.Next(0, Xk.Length);

}

Yk[currentPosition] = char.Parse(XOR(Yk[currentPosition].ToString(), '1'.ToString()));

Thread.Sleep(100);

previosPosition = currentPosition;

}

return new string(Yk);

}

Листинг 4 – задание 4

**Задание 5:** Для полученного слова Yn = Yk, Yr, используя уже известную проверочную матрицу Хемминга, вновь вычислить избыточные символы.

Для выполнения данного задания используются методы, разработанные в предыдущих заданиях.

**Задание 6:** В случае, если анализ синдрома показал, что информационное сообщение было передано с ошибкой (или 2 ошибками), сгенерировать унарный вектор ошибки n и исправить одиночную ошибку, используя формулу (4.7).

Дополним разработанный в предыдущем задании код следующим методом:

static public string ErrorVector(int position, int n)

{

string errorVector = null;

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

{

if (i != position)

errorVector += '0';

else

errorVector += '1';

}

return errorVector;

}

static public int ErrorPosition(char[,] hammingMatrix, string syndrome, int r, int k)

{

int amountCoincidences = 0;

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

{

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

{

if (hammingMatrix[j, i] == syndrome[j])

amountCoincidences++;

}

if (amountCoincidences == r)

return i;

else

amountCoincidences = 0;

}

return -1;

}

Листинг 5 – задание 6

Результат работы итогового кода:

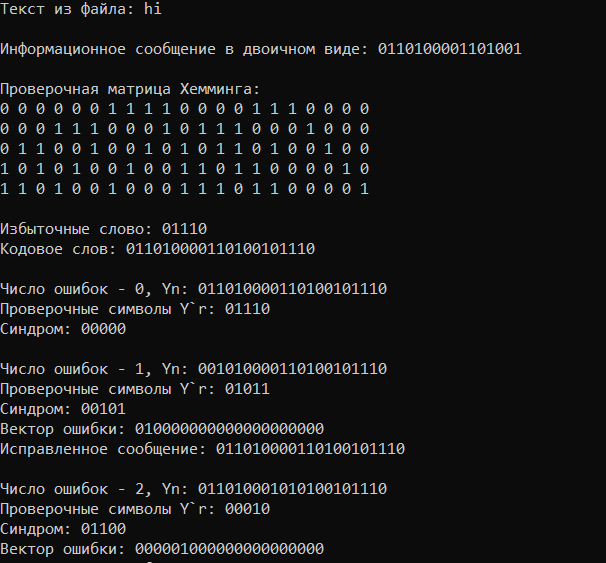


Рисунок 1 ­­­­– Результат работы программы

**Вывод:** в ходе лабораторной работы были приобретены практические навыки кодирования/декодирования двоичных данных при использовании кода Хемминга.