Учреждение образования

«Белорусский государственный технологический университет»

**Лабораторная работа №5**

**ИТЕРАТИВНЫЙ КОД**

Выполнил:

Студент 3 курса 3 группы ФИТ

Калоша И.В.

**2020 г**

**Цель:** приобретение практических навыков кодирования/декодирования двоичных данных при использовании итеративных кодов.

**Теория**

*Итеративные коды* относятся к классу кодов произведения. Кодом произведения двух исходных (базовых) помехоустойчивых кодов называется такой многомерный помехоустойчивый код, кодовыми последовательностями которого являются все двумерные таблицы со строками кода (k1) и столбцами кода (k2). Итеративные коды могут строиться на основе использования дву-, трехмерных матриц (таблиц) и более высоких размерностей. Каждая из отдельных последовательностей информационных символов кодируется определенным линейным кодом (групповым или циклическим). Получаемый таким образом итеративный код также является линейным. Простейшим из итеративных кодов является двумерный код с проверкой на четность по строкам и столбцам.

Запись матрицы в таком виде не считается канонической, так как подматрица I не является единичной диагональной матрицей. Для преобразования такой записи к каноническому виду воспользуемся свойствами линейного кода: в дополнительную строку необходимо записать сумму по модулю 2 соответствующих символов матрицы кода с dmin = 3.

В общем случае любая проверочная матрица кода Хемминга с dmin = 4 имеет нечетный вес столбцов, т. е. вес любого из столбцов подматрицы А′ может быть равен 3, 5, 7, … .

Как и в предыдущем случае (при dmin = 3), равенство нулю синдрома означает отсутствие ошибок. Если же синдром не равен нулю и имеет нечетный вес, то это говорит о том, что произошла одиночная ошибка. Если же синдром не равен нулю и его вес четный, то произошла двойная ошибка, так как вес суммы любых двух столбцов всегда четный.

**Практическое задание**

1. вписывать произвольное двоичное представление информационного слова Хk (кодируемой информации) длиной k битов в двумерную матрицу размерностью в соответствии с вариантом либо в трехмерную матрицу в соответствии с вариантом (указаны в табл. 5.2);

static void ArrayOut (int [] arr)

{

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

{

Console.Write(arr[i]);

}

Console.WriteLine();

}

1. вычислять проверочные биты (биты паритетов): а) по двум; б) по трем; в) по четырем направлениям (группам паритетов);

static int[] GettingCheckBits(int[] arr, int[] fullArr)

{

int n = arr.Length;

int nSqrt = (int)Math.Sqrt(n);

int lenght = fullArr.Length;

int l = nSqrt + 1;

int[,] matrix = new int[l, l];

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

{

int parity = 0;

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

{

matrix[i, j] = arr[i \* nSqrt + j];

parity += matrix[i, j];

fullArr[i \* nSqrt + j] = matrix[i, j];

}

matrix[i, nSqrt] = parity % 2;

fullArr[n + i] = matrix[i, nSqrt];

}

for (int i = nSqrt -1 ; i >= 0; i--)

{

int parity = 0;

for (int j = nSqrt - 1; j >= 0; j--)

{

parity += matrix[j, i];

}

matrix[nSqrt, i] = parity % 2;

fullArr[n + nSqrt + (nSqrt - i- 1)] = matrix[nSqrt, i];

}

MatrixOut(matrix, l);

ArrayOut(fullArr);

return fullArr;

}

3) формировать кодовое слово Xn присоединением избыточных символов к информационному слову;

1. генерировать ошибку произвольной кратности (i, i > 0), распределенную случайным образом среди символов слова Xn, в результате чего формируется кодовое слово Yn;

static int[] GettingError(int [] fullArr)

{

int error;

try

{

Console.WriteLine("Введите место первой ошибки");

error = Convert.ToInt32(Console.ReadLine()) - 1;

if (fullArr[error] == 1) fullArr[error] = 0;

else fullArr[error] = 1;

}

catch { }

try

{

Console.WriteLine("Введите место второй ошибки");

error = Convert.ToInt32(Console.ReadLine()) - 1;

if (fullArr[error] == 1) fullArr[error] = 0;

else fullArr[error] = 1;

}

catch { }

ArrayOut(fullArr);

return fullArr;

}

1. определять местоположение ошибочных символов итеративным кодом в слове Yn в соответствии с используемыми группами паритетов по пункту (2) и исправлять ошибочные символы (результат исправления – слово Yn’); 6) выполнять анализ корректирующей способности используемого кода (количественная оценка) путем сравнения соответствующих слов Xn и Yn’; результат анализа может быть представлен в виде отношения общего числа сгенерированных кодовых слов с ошибками определенной одинаковой кратности (с одной ошибкой, с двумя ошибками и т. д.) к числу кодовых слов, содержащих Избыточное кодирование данных в информационных системах. Итеративные коды 55 ошибки этой кратности, которые правильно обнаружены и которые правильно скорректированы.

static int[] CorrectionError(int[] newArr, int[] fullArr)

{

int n = newArr.Length;

int nSqrt = (int)Math.Sqrt(n);

int lenght = fullArr.Length;

int l = nSqrt + 1;

int ErrorStr = -1;

int ErrorSlb = -1;

bool ErrorDoubleStr = false;

bool ErrorDoubleSlb = false;

int[,] matrix = new int[l, l];

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

{

int parity = 0;

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

{

matrix[i, j] = fullArr[i \* nSqrt + j];

parity += matrix[i, j];

newArr[i \* nSqrt + j] = matrix[i, j];

}

matrix[i, nSqrt] = parity % 2;

if (matrix[i, nSqrt] != fullArr[n + i])

{

if(ErrorStr != -1)

{

ErrorDoubleStr = true;

}

ErrorStr = i;

}

}

if (ErrorStr == -1)

{

Console.WriteLine("Обшибок не обнаружено");

return newArr;

}

for (int i = nSqrt - 1; i >= 0; i--)

{

int parity = 0;

for (int j = nSqrt - 1; j >= 0; j--)

{

parity += matrix[j, i];

}

matrix[nSqrt, i] = parity % 2;

if (matrix[nSqrt, i] != fullArr[n + nSqrt + (nSqrt - i - 1)])

{

if (ErrorSlb != -1)

{

ErrorDoubleSlb = true;

if (ErrorStr != -1)

{

Console.WriteLine("Внимание! Обнаружено больше одной ошибки, исправление невозможно.");

return newArr;

}

}

ErrorSlb = i;

if (ErrorDoubleStr)

{

Console.WriteLine("Внимание! Обнаружено больше одной ошибки, исправление невозможно.");

return newArr;

}

}

}

if (ErrorStr != -1 && ErrorSlb != -1)

{

newArr[ErrorStr \* nSqrt + ErrorSlb] = (matrix[ErrorStr, ErrorSlb] + 1) % 2;

}

ArrayOut(newArr);

return newArr;

}

}

**Вывод:** в данной работе был рассмотрен вариант итеративного кодирования информации, рассмотрены его плюсы и минусы. Было установлено, что итеративные коды могут корректировать несколько ошибок одинаковой четности.