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 №5 НА ТЕМУ:**

**Избыточное кодирование данных в информационных системах. Итеративные коды**

Ф.И.О.

Трофимчук Михаил Витальевич

Минск 2022

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

**Теоретические сведения**

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.

**Ход работы**

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);
2. Вычислять проверочные биты (биты паритетов) по а) двум, б) трем, в) четырем направлениям (группам паритетов);
3. Формировать кодовое слово Xn, присоединением избыточных символов к информационному слову;
4. Генерировать ошибку произвольной кратности (i, i>0), распределенную случайным образом среди символов слова Xn, в результате чего формируется кодовое слово Yn;
5. Определять местоположение ошибочных символов итеративным кодом в слове Yn в соответствии с используемыми группами паритетов по п. 1.2 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 ошибки этой кратности, которые правильно обнаружены и которые правильно скорректированы.

|  |
| --- |
| static void IterativeMatrix(int height , int width){    Random rand = new Random();  int[] itog = new int[16];  int[] save = new int[16];  int[,] generateMessage = new int[height, width];  int[] sc = new int[width];  int eachRow, i, j;  int[] gorParity = new int[width];  int[] verParity = new int[height];  int result = 0;  for ( i = 0; i < width; i++) sc[i] = 0;  for (i = 0; i < height; i++)  {  eachRow = 0;  for (j = 0; j < width; j++)  {  generateMessage[i,j] = rand.Next(0,2);  Console.Write(generateMessage[i,j] + " ");  eachRow += generateMessage[i,j];  result += generateMessage[i, j];  gorParity[i] = eachRow;  sc[j] += generateMessage[i,j];  }  int l = eachRow % 2;  Console.WriteLine(" |" + l);    }  for (i = 0; i < width; i++)  Console.Write("---");  Console.Write("\n");  for (i = 0; i < width; i++)  {  int z = sc[i] % 2;  verParity[i] = z;  Console.Write( z + " ");  }  Console.WriteLine("\n");    int itogreg = 0;  int itogregd = 0;  Console.Write("Xn= ");  for (i = 0; i < width; i++) sc[i] = 0;  for (i = 0; i < height; i++)  {    for (j = 0; j < width; j++)  {  Console.Write(generateMessage[i, j]);  itog[itogreg++] = generateMessage[i, j];  save[itogregd++] = generateMessage[i, j];  }    }  // вертикальный паритет  int o = 0;  int[] savegor = new int[4];  Console.Write(" ");  foreach (int item in gorParity)  {  int l = item % 2;  savegor[o++] = l;  Console.Write(l);  }  int f = 0;  int[] savever = new int[4];  // горизонтальный паритет  Console.Write(" ");  Array.Reverse(verParity);  foreach (int item in verParity)  {  savever[f++] = item;  Console.Write(item);  }  int res = result % 2;  Console.WriteLine(" " +res);  foreach (int item in itog)  {  Console.Write(item);  }    try  {  Console.WriteLine();  var random = new Random();  int error = random.Next(0, itog.Length);  Console.WriteLine("Позиция ошибки: " + error);  if (itog[error] == 1) itog[error] = 0;  else itog[error] = 1;  }  catch { }  Console.Write("Yn= ");  foreach (int item in itog)  {  Console.Write(item);  }  Console.WriteLine();  int iter = 0;  for (i = 0; i < width; i++) sc[i] = 0;  for (i = 0; i < height; i++)  {  eachRow = 0;  for (j = 0; j < width; j++)  {  generateMessage[i, j] = itog[iter++];  Console.Write(generateMessage[i, j] + " ");  eachRow += generateMessage[i, j];  sc[j] += generateMessage[i, j];  gorParity[i] = eachRow;  }  int l = eachRow % 2;  Console.WriteLine(" |" + l );  }  for (i = 0; i < width; i++)  Console.Write("---");  Console.Write("\n");  for (i = 0; i < width; i++)  {  int z = sc[i] % 2;  verParity[i] = z;  Console.Write(z + " ");  }  Console.WriteLine("\n");  Console.WriteLine("Сравниваем паритеты:");  Console.Write(" ");  foreach (int item in gorParity)  {  int l = item % 2;  Console.Write(l);  }  // горизонтальный паритет  Console.Write(" ");  Array.Reverse(verParity);  foreach (int item in verParity)  {  Console.Write(item);  }  Console.WriteLine();  Console.Write(" ");  foreach (int it in savegor) {  Console.Write(it);  }  Console.Write(" ");  // Array.Reverse(savever);  foreach (int it in savever)  {  Console.Write(it);  }  Console.WriteLine();  Console.WriteLine("Исправляем, строка без ошибки:");  Console.Write("Xn= ");  foreach (int item in save) {  Console.Write(item);  }    } |

Листинг 1 – Функция итеративного кода

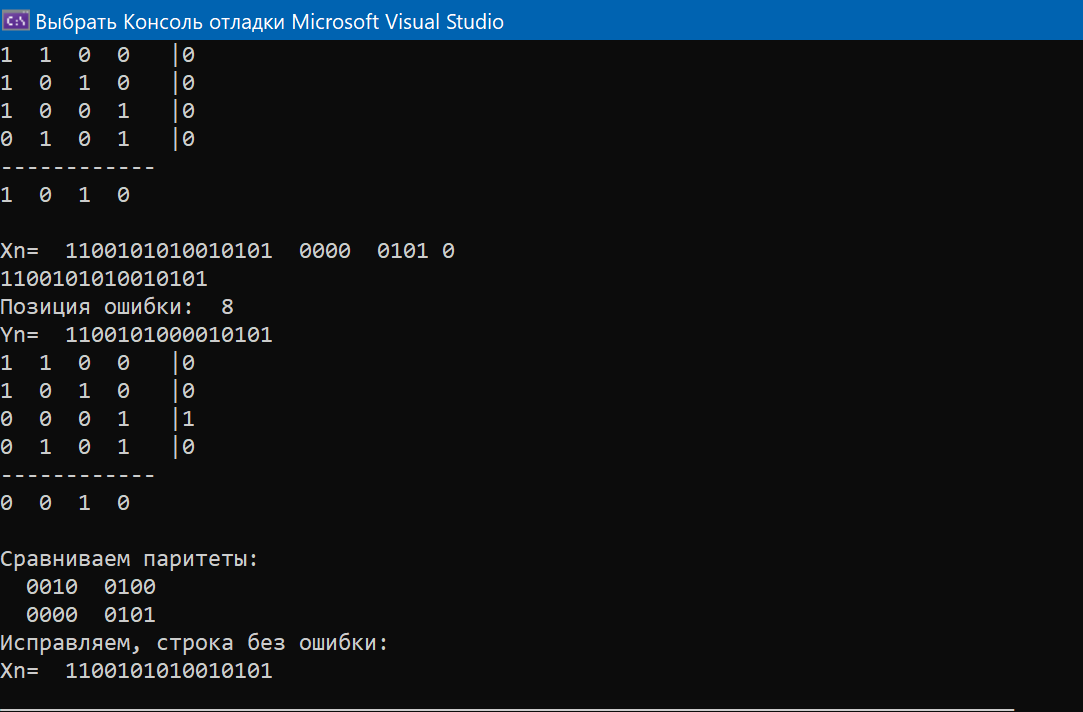


Рисунок 1 – Результат выполнения функции

|  |
| --- |
| static void IterativeMatrixWithDepth(int height, int width, int depth)  {    Random rand = new Random();  int[] itog = new int[16];    int[] sc = new int[width];    int[] result = new int[16];  int itogers = 0;  for (int i = 0; i < depth; i++) sc[i] = 0;    int[,,] array = new int[height, width, depth];  for (int i = 0; i < array.GetLength(0); i++)  {    for (int j = 0; j < array.GetLength(1); j++)  {    int slow = 0;  for (int k = 0; k < array.GetLength(2); k++)  {    array[i, j, k] = rand.Next(0, 2);  slow += array[i, j, k];  result[itogers++] += array[i, j, k];  Console.Write(array[i, j, k]);  sc[k] += array[i, j, k ];  }  int l = slow % 2;  Console.WriteLine(" |" + l);    }  Console.WriteLine("---");  for (int d = 0; d < depth; d++)  {  int z = sc[d] % 2;    Console.Write(z + "");  sc[d] = 0;  }  Console.WriteLine();  Console.WriteLine();  }  Console.WriteLine("Паритет 4 - 5 группы:");  for (int i = 0; i < array.GetLength(0); i++)  {  for (int j = 0; j < array.GetLength(1); j++)  {    for (int k = 0; k < array.GetLength(2); k++)  {  Console.Write(array[i, j, k]);  }  }  Console.WriteLine(" ");  }  int first = result[0] + result[5] + result[10] + result[15];  int firstm = first % 2;  int second = result[1] + result[6] + result[11] + result[12];  int secondm = second % 2;  int third = result[2] + result[7] + result[8] + result[13];  int thirdm = third % 2;  int fourth = result[3] + result[4] + result[9] + result[14];  int fourthm = fourth % 2;  int firstf = result[0] + result[7] + result[10] + result[13];  int firstmf = firstf % 2;  int secondf = result[1] + result[4] + result[11] + result[14];  int secondmf = secondf % 2;  int thirdf = result[2] + result[5] + result[8] + result[15];  int thirdmf = thirdf % 2;  int fourthf = result[3] + result[6] + result[9] + result[12];  int fourthmf= fourthf % 2;  Console.WriteLine("Паритет 5 группы: " + firstm + secondm + thirdm + fourthm);  Console.WriteLine("Паритет 4 группы: " + firstmf + secondmf + thirdmf + fourthmf);  int[] sresult = new int[16];  Console.Write("Xn = ");  int inh = 0;  foreach (var item in result)  {  sresult[inh++] = item;  Console.Write(item);  }    try  {  Console.WriteLine();  var random = new Random();  int error = random.Next(0, itog.Length);  Console.WriteLine("Позиция ошибки: " + error);  if (result[error] == 1) result[error] = 0;  else result[error] = 1;  }  catch { }  Console.Write("Yn = ");  foreach (var item in result)  {  Console.Write(item);  }  Console.WriteLine();  int itogers2 = 0;  int[] result2 = new int[16];  int iter = 0;  for (int i = 0; i < array.GetLength(0); i++)  {  for (int j = 0; j < array.GetLength(1); j++)  {  int slow = 0;  for (int k = 0; k < array.GetLength(2); k++)  {  array[i, j, k] = result[iter++];  slow += array[i, j, k];  result2[itogers2++] += array[i, j, k];  Console.Write(array[i, j, k]);  sc[k] += array[i, j, k];  }  int l = slow % 2;  Console.WriteLine(" |" + l);  }  Console.WriteLine("---");  for (int d = 0; d < depth; d++)  {  int z = sc[d] % 2;  Console.Write(z + "");  sc[d] = 0;  }  Console.WriteLine();  Console.WriteLine();  }  first = result2[0] + result2[5] + result2[10] + result2[15];  firstm = first % 2;  second = result2[1] + result2[6] + result2[11] + result2[12];  secondm = second % 2;  third = result2[2] + result2[7] + result2[8] + result2[13];  thirdm = third % 2;  fourth = result2[3] + result2[4] + result2[9] + result2[14];  fourthm = fourth % 2;  firstf = result2[0] + result2[7] + result2[10] + result2[13];  firstmf = firstf % 2;  secondf = result2[1] + result2[4] + result2[11] + result2[14];  secondmf = secondf % 2;  thirdf = result2[2] + result2[5] + result2[8] + result2[15];  thirdmf = thirdf % 2;  fourthf = result2[3] + result2[6] + result2[9] + result2[12];  fourthmf = fourthf % 2;  Console.WriteLine("Паритет 5 группы: " + firstm + secondm + thirdm + fourthm);  Console.WriteLine("Паритет 4 группы: " + firstmf + secondmf + thirdmf + fourthmf);  Console.WriteLine("Исправленное сообщение:");  Console.Write("Xn = ");  foreach (var item in sresult)  {  Console.Write(item);  }  } |

Листинг 2 – Функция итеративного кода с нахождением паритетов 4 и 5 группы

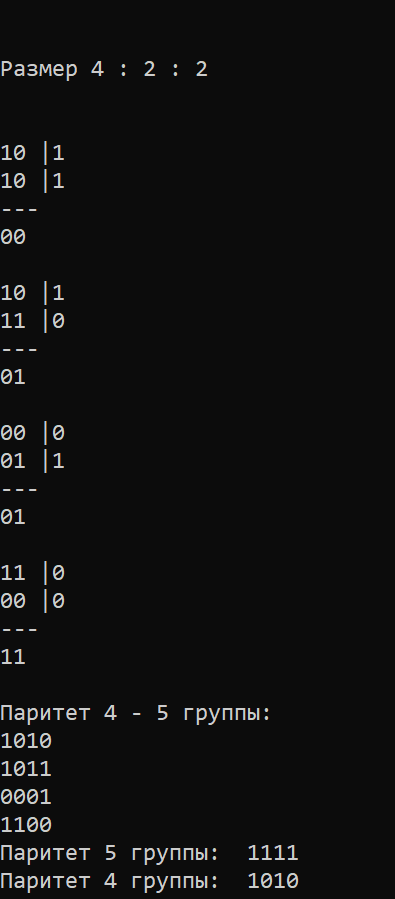


Рисунок 2 – Результат выполнения функции с нахождением паритетов 4 и 5 группы

Анализ корректирующей способности используемого кода (количественная оценка) путем сравнения соответствующих слов Xn и Yn:

В ходе выполнения лабораторной работы, мы можем заметить, что в ситуации, где сообщение Xn отличается на 1 бит от Yn, сообщение подвержено исправлению. Однако в ситуации с двумя ошибками, обнаружить место ошибки невозможно, поскольку при появлении ошибки на одном уровне, сумма по модулю два не будет исправлять сообщение. Следуя от частного к общему, можем прийти к выводу, что итеративный код может исправлять с большей точностью нечетное количество ошибок. Так же при увеличении групп паритетов, шанс обнаружения увеличивается. Наглядный пример:

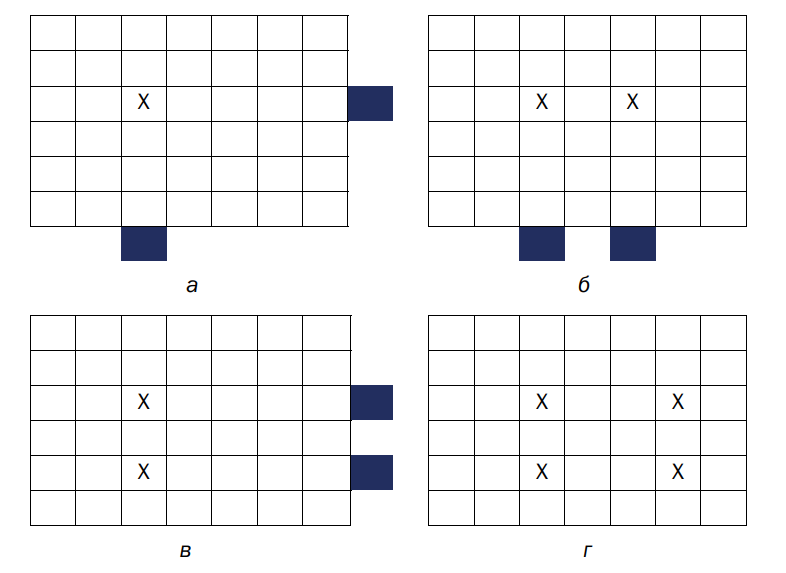


Рисунок 3 – Наглядный пример проблемы обнаружения двух ошибок

**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.