Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Отчет по лабораторной работе №5**

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

Выполнил:

студент 3 курса 1 группы

Парибок И.А.

Вариант 5

Минск 2022

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

Задачи:

1. Закрепить теоретические знания по использованию итеративных кодов для повышения надежности передачи и хранения в памяти компьютера двоичных данных.

2. Разработать приложение для кодирования/декодирования двоичной информации итеративным кодом с различной относительной избыточностью кодовых слов.

3. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

Создание произвольного информационного слова *Xk*, длинной *k =* 40в двумерную матрицу размерности *k1* = 5, *k2* = 8 и получение горизонтальных, вертикальных, диагональных паритетов представлено на рисунке 1



Рисунок 1 – Создание слова и получение паритетов

Подсчет паритетов после задания ошибок представлен на рисунке 2.

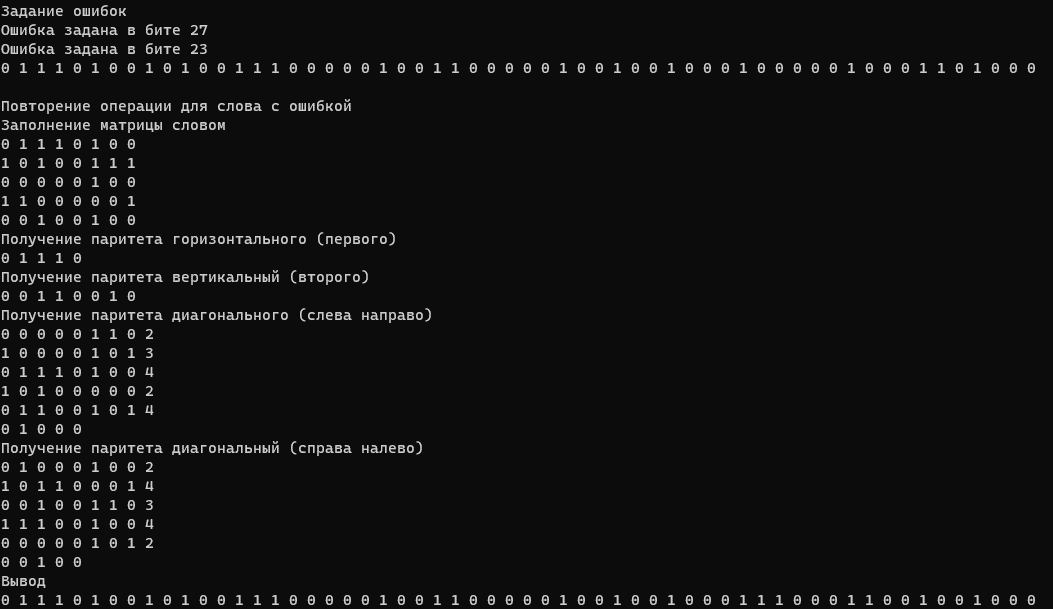


Рисунок 2 – Расчет паритетов после задания ошибок

Поиск ошибок представлен на рисунке 3.

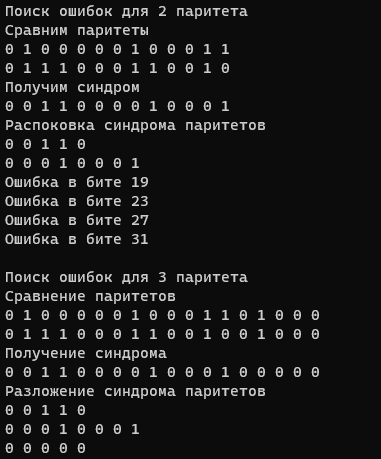


Рисунок 3 – Поиск ошибок

Создание произвольного информационного слова *Xk*, длинной *k =* 40в трехм ерную матрицу размерности *k1* = 2, *k2* = 10, *z* = 2 и получение горизонтального, вертикального, диагональных и глубинного паритетов представлено на рисунке 4.

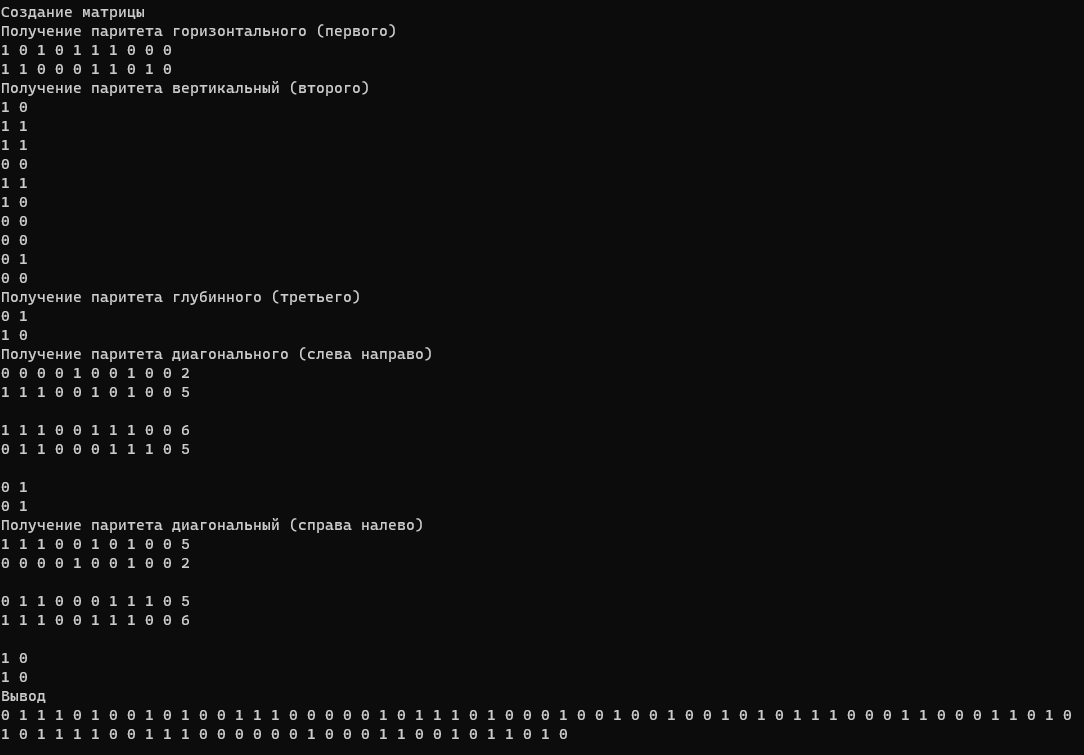


Рисунок 4 – Создание слова и получение паритетов

Подсчет паритетов после задания ошибок представлен на рисунке 5.

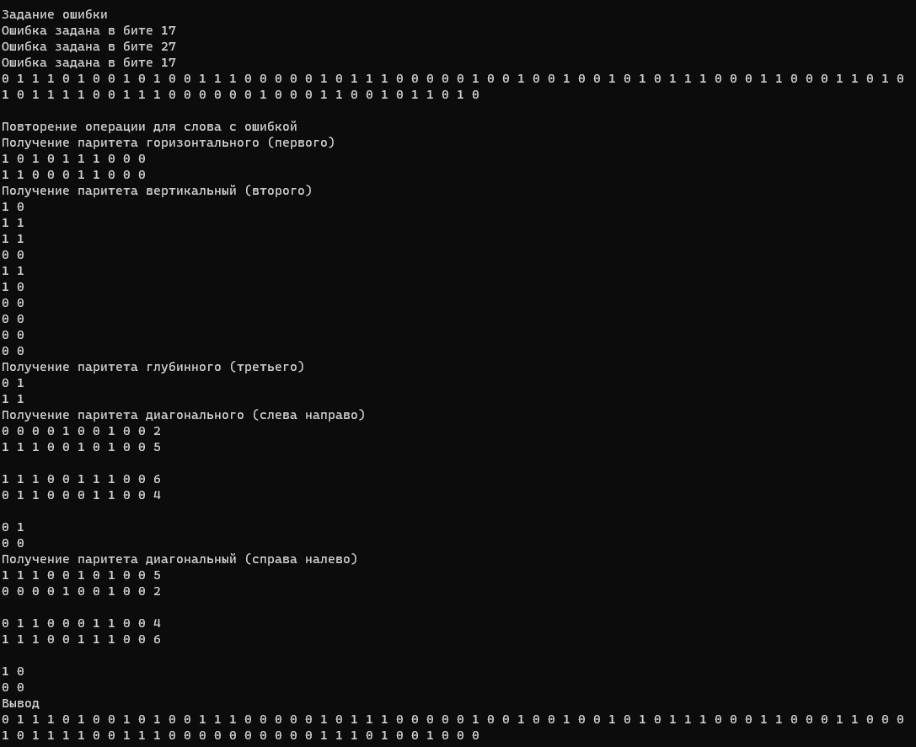


Рисунок 5 – Расчет паритетов после задания ошибок

Поиск ошибок представлен на рисунках 6-8 .



Рисунок 6 – Поиск ошибок



Рисунок 7 – Поиск ошибок

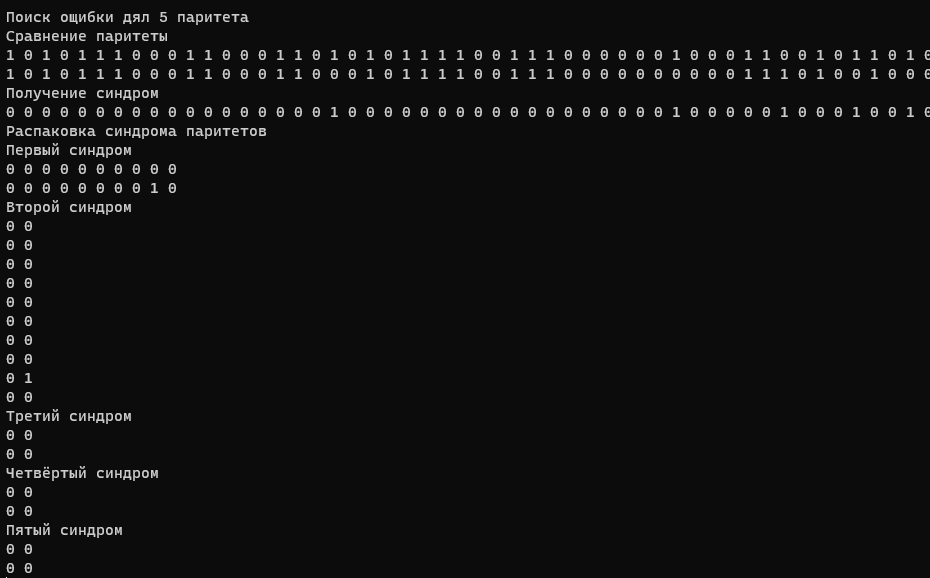


Рисунок 8 – Поиск ошибок

**Вывод**: Был получен опыт приобретения практических навыков кодирования и декодирования двоичных данных при использовании итеративных кодов.

**Приложение А**

static void Main()

{

Console.OutputEncoding = Encoding.UTF8;

//Создание слово

Console.WriteLine("Информационное слово Xk");

int k = 40;

byte[] baseWord = new byte[k];

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

{

baseWord[i] = (byte)(rand.Next(0, 2));

}

ShowWord(baseWord);

//Создание двумерной матрицы

Console.WriteLine("Двумерная матрица");

int k1 = 5;

int k2 = 8;

byte[,] baseMatrix = CreateMatrix(k1, k2);

byte[] fullWord = ActionsForTwoDimensialMatrix(baseWord, baseMatrix);

//Задание ошибок

Console.WriteLine();

Console.WriteLine("Задание ошибок");

int countMistakes = 2;

byte[] fullWordWithMistakes = CreateMistake(fullWord, k, countMistakes);

ShowWord(fullWordWithMistakes);

Console.WriteLine();

//Повторение операции для слова с ошибкой

Console.WriteLine("Повторение операции для слова с ошибкой");

byte[] wordWithMistakes = new byte[k];

Array.Copy(fullWordWithMistakes, wordWithMistakes, k);

byte[] newfullWord = ActionsForTwoDimensialMatrix(wordWithMistakes, baseMatrix);

//Поиск ошибок для 2 паритета

Console.WriteLine();

Console.WriteLine("Поиск ошибок для 2 паритета");

byte[] twoParitiesWithMistakes = new byte[k1 + k2];

Array.Copy(fullWordWithMistakes, k, twoParitiesWithMistakes, 0, k1 + k2);

byte[] newTwoParities = new byte[k1 + k2];

Array.Copy(newfullWord, k, newTwoParities, 0, k1 + k2);

FindMistakesWithTwoParities(k1, k2, twoParitiesWithMistakes, newTwoParities);

//Поиск ошибок 3 паритета

Console.WriteLine();

Console.WriteLine("Поиск ошибок для 3 паритета");

byte[] threeParitiesWithMistakes = new byte[k1 + k2 + k1];

Array.Copy(fullWordWithMistakes, k, threeParitiesWithMistakes, 0, k1 + k2 + k1);

byte[] newThreeParities = new byte[k1 + k2 + k1];

Array.Copy(newfullWord, k, newThreeParities, 0, k1 + k2 + k1);

FindMistakesWithThreeParities(k1, k2, k1, threeParitiesWithMistakes, newThreeParities);

Console.WriteLine("3хмерная матрица");

//создаём матрицу 3-мерную

Console.WriteLine("Создание матрицы");

k1 = 2;

k2 = 10;

int z = 2;

byte[,,] baseMatrixTriple = CreateMatrix(k1, k2, z);

fullWord = ActionsForThreeDimensialMatrix(baseWord, baseMatrixTriple);

//задаём ошибки

Console.WriteLine();

Console.WriteLine("Задание ошибки");

countMistakes = 3;

byte[] fullWordWithMistakesThree = CreateMistake(fullWord, k, countMistakes);

ShowWord(fullWordWithMistakesThree);

Console.WriteLine();

//повторяем операции для слова с ошибкой

Console.WriteLine("Повторение операции для слова с ошибкой");

byte[] wordWithMistakesThree = new byte[k];

Array.Copy(fullWordWithMistakesThree, wordWithMistakesThree, k);

byte[] newfullWordThree = ActionsForThreeDimensialMatrix(wordWithMistakesThree, baseMatrixTriple);

//ищем ошибки 2 паритета

Console.WriteLine();

Console.WriteLine("Поиск ошибки 2 паритета");

byte[] twoParitiesWithMistakesThree = new byte[

(k1 \* k2) + (k2 \* z)];

Array.Copy(fullWordWithMistakesThree, k, twoParitiesWithMistakesThree, 0,

twoParitiesWithMistakesThree.Length);

newTwoParities = new byte[

(k1 \* k2) + (k2 \* z)];

Array.Copy(newfullWordThree, k, newTwoParities, 0,

newTwoParities.Length);

FindMistakesWithTwoParitiesThree(k1, k2, z, twoParitiesWithMistakesThree, newTwoParities);

//ищем ошибки 3 паритета

Console.WriteLine();

Console.WriteLine("Поиск ошибки для 3 паритета");

byte[] threeParitiesWithMistakesThree = new byte[

(k1 \* k2) + (k2 \* z) + (k1 \* z)];

Array.Copy(fullWordWithMistakesThree, k, threeParitiesWithMistakesThree, 0,

threeParitiesWithMistakesThree.Length);

newThreeParities = new byte[

(k1 \* k2) + (k2 \* z) + (k1 \* z)];

Array.Copy(newfullWordThree, k, newThreeParities, 0,

newThreeParities.Length);

FindMistakesWithThreeParitiesThree(k1, k2, z, threeParitiesWithMistakesThree, newThreeParities);

//ищем ошибки 4 паритета

Console.WriteLine();

Console.WriteLine("Поиск ошибки для 4 паритета");

byte[] fourParitiesWithMistakes = new byte[

(k1 \* k2) + (k2 \* z) + (k1 \* z) + (k1 \* z)];

Array.Copy(fullWordWithMistakesThree, k, fourParitiesWithMistakes, 0,

fourParitiesWithMistakes.Length);

byte[] newFourParities = new byte[

(k1 \* k2) + (k2 \* z) + (k1 \* z) + (k1 \* z)];

Array.Copy(newfullWordThree, k, newFourParities, 0,

newFourParities.Length);

FindMistakesWithFourParitiesThree(k1, k2, z, fourParitiesWithMistakes, newFourParities);

//ищем ошибки 5 паритета

Console.WriteLine();

Console.WriteLine("Поиск ощибки дял 5 паритета");

byte[] fiveParitiesWithMistakes = new byte[

(k1 \* k2) + (k2 \* z) + (k1 \* z) + (k1 \* z) + (k1 \* z)];

Array.Copy(fullWordWithMistakesThree, k, fiveParitiesWithMistakes, 0,

fiveParitiesWithMistakes.Length);

byte[] newFiveParities = new byte[

(k1 \* k2) + (k2 \* z) + (k1 \* z) + (k1 \* z) + (k1 \* z)];

Array.Copy(newfullWordThree, k, newFiveParities, 0,

newFiveParities.Length);

FindMistakesWithFiveParitiesThree(k1, k2, z, fiveParitiesWithMistakes, newFiveParities);

Console.ReadLine();

}

public static Random rand = new Random();

public static void ShowWord(byte[] word)

{

foreach (var literal in word)

{

Console.Write(literal + " ");

}

Console.WriteLine();

}

public static byte[,] CreateMatrix(int k1, int k2)

{

return new byte[k1, k2];

}

public static byte[,,] CreateMatrix(int k1, int k2, int z)

{

return new byte[k1, k2, z];

}

public static byte[,] FillMatrix(byte[,] matrix, byte[] word)

{

for (int i = 0; i < matrix.GetLongLength(0); i++)

{

for (int j = 0; j < matrix.GetLongLength(1); j++)

{

matrix[i, j] = (byte)word[i \* matrix.GetLongLength(1) + j];

}

}

return matrix;

}

public static byte[,,] FillMatrix(byte[,,] matrix, byte[] word)

{

for (int i = 0; i < matrix.GetLongLength(0); i++)

{

for (int j = 0; j < matrix.GetLongLength(1); j++)

{

for (int g = 0; g < matrix.GetLongLength(2); g++)

{

matrix[i, j, g] = (byte)word[i \* matrix.GetLongLength(1) + j \* matrix.GetLongLength(2) + g];

}

}

}

return matrix;

}

public static void ShowMatrix(byte[,] matrix)

{

for (int i = 0; i < matrix.GetLongLength(0); i++)

{

for (int j = 0; j < matrix.GetLongLength(1); j++)

{

Console.Write(matrix[i, j] + " ");

}

Console.WriteLine();

}

}

public static byte[] GetFirstParity(byte[,] matrix)

{

byte[] firstParity = new byte[matrix.GetLongLength(0)];

byte sum;

for (int i = 0; i < matrix.GetLongLength(0); i++)

{

sum = (byte)0;

for (int j = 0; j < matrix.GetLongLength(1); j++)

{

sum += (byte)(matrix[i, j]);

}

firstParity[i] = (byte)(sum % 2);

}

return firstParity;

}

public static byte[] GetSecondParity(byte[,] matrix)

{

byte[] secondParity = new byte[matrix.GetLongLength(1)];

byte sum;

for (int i = 0; i < matrix.GetLongLength(1); i++)

{

sum = (byte)0;

for (int j = 0; j < matrix.GetLongLength(0); j++)

{

sum += (byte)(matrix[j, i]);

}

secondParity[i] = (byte)(sum % 2);

}

return secondParity;

}

public static byte[] GetFirstDiagParity(byte[,] matrix)

{

byte[] firstDiagParity = new byte[matrix.GetLongLength(0)];

byte sum;

for (int i = 0; i < matrix.GetLongLength(0); i++)

{

sum = (byte)0;

for (int j = 0; j < matrix.GetLongLength(1); j++)

{

Console.Write(matrix[(i + j) % matrix.GetLongLength(0), j % matrix.GetLongLength(1)] + " ");

sum += (byte)(matrix[(i + j) % matrix.GetLongLength(0), j % matrix.GetLongLength(1)]);

}

Console.WriteLine(sum);

firstDiagParity[i] = (byte)(sum % 2);

}

return firstDiagParity;

}

public static byte[] GetSecondDiagParity(byte[,] matrix)

{

byte[] firstDiagParity = new byte[matrix.GetLongLength(0)];

byte sum;

for (int i = 0; i < matrix.GetLongLength(0); i++)

{

sum = (byte)0;

for (int j = 0; j < matrix.GetLongLength(1); j++)

{

Console.Write(matrix[(matrix.GetLongLength(0) - 1) - ((i + j) % matrix.GetLongLength(0)),

j % matrix.GetLongLength(1)] + " ");

sum += (byte)(matrix[(matrix.GetLongLength(0) - 1) - ((i + j) % matrix.GetLongLength(0)),

j % matrix.GetLongLength(1)]);

}

Console.WriteLine(sum);

firstDiagParity[i] = (byte)(sum % 2);

}

return firstDiagParity;

}

public static byte[] CreateFullWord(byte[] word, byte[] firstParity, byte[] secondParity, byte[] firstDiagParity)

{

return ((word.Concat(firstParity).ToArray())

.Concat(secondParity).ToArray())

.Concat(firstDiagParity).ToArray();

}

public static byte[] CreateMistake(byte[] fullWord, int k, int countMistakes)

{

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

{

int placeOfMistake = rand.Next(0, k);

Console.WriteLine("Ошибка задана в бите " + placeOfMistake);

fullWord[placeOfMistake] = (byte)((fullWord[placeOfMistake] + 1) % 2);

}

return fullWord;

}

public static byte[] GetSyndrom(byte[] paritiesWithMistakes, byte[] newParities)

{

byte[] syndrom = new byte[paritiesWithMistakes.Length];

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

{

syndrom[i] = (byte)((paritiesWithMistakes[i] + newParities[i]) % 2);

}

return syndrom;

}

public static void FindMistakesWithTwoParities(int k1, int k2, byte[] paritiesWithMistakes, byte[] newParities)

{

//сравним паритеты

Console.WriteLine("Сравним паритеты");

ShowWord(paritiesWithMistakes);

ShowWord(newParities);

//получим синдромы

Console.WriteLine("Получим синдром");

byte[] syndrom = GetSyndrom(paritiesWithMistakes, newParities);

ShowWord(syndrom);

//распарсим синдромы паритетов

Console.WriteLine("Распоковка синдрома паритетов");

byte[] firstParitySyndrom = new byte[k1];

Array.Copy(syndrom, 0, firstParitySyndrom, 0, k1);

byte[] secondParitySyndrom = new byte[k2];

Array.Copy(syndrom, k1, secondParitySyndrom, 0, k2);

ShowWord(firstParitySyndrom);

ShowWord(secondParitySyndrom);

//ИСПРАВИМ ОШИБКИ

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

{

if (firstParitySyndrom[i] == (byte)1)

{

for (int j = 0; j < secondParitySyndrom.Length; j++)

{

if (secondParitySyndrom[j] == (byte)1)

{

Console.WriteLine("Ошибка в бите " + ((i \* secondParitySyndrom.Length) + (j + 1) - 1));

}

}

}

}

}

public static void FindMistakesWithThreeParities(int k1, int k2, int k3, byte[] paritiesWithMistakes, byte[] newParities)

{

//сравним паритеты

Console.WriteLine("Сравнение паритетов");

ShowWord(paritiesWithMistakes);

ShowWord(newParities);

//получим синдромы

Console.WriteLine("Получение синдрома");

byte[] syndrom = GetSyndrom(paritiesWithMistakes, newParities);

ShowWord(syndrom);

//распарсим синдромы паритетов

Console.WriteLine("Разложение синдрома паритетов");

byte[] firstParitySyndrom = new byte[k1];

Array.Copy(syndrom, 0, firstParitySyndrom, 0, k1);

byte[] secondParitySyndrom = new byte[k2];

Array.Copy(syndrom, k1, secondParitySyndrom, 0, k2);

byte[] firstDiagParitySyndrom = new byte[k3];

Array.Copy(syndrom, k1 + k2, firstDiagParitySyndrom, 0, k3);

ShowWord(firstParitySyndrom);

ShowWord(secondParitySyndrom);

ShowWord(firstDiagParitySyndrom);

//ИСПРАВИМ ОШИБКИ

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

{

if (firstParitySyndrom[i] == (byte)1)

{

for (int j = 0; j < secondParitySyndrom.Length; j++)

{

if (secondParitySyndrom[j] == (byte)1)

{

for (int g = 0; g < firstDiagParitySyndrom.Length; g++)

{

if (firstDiagParitySyndrom[g] == (byte)1)

{

for (int newI = g, newJ = 0; newJ < secondParitySyndrom.Length; newI++, newJ++)

{

if ((i == newI % firstParitySyndrom.Length) && (j == newJ))

Console.WriteLine("Ошибка в бите " + ((i \* secondParitySyndrom.Length) + (j + 1) - 1));

}

}

}

}

}

}

}

}

public static byte[] ActionsForTwoDimensialMatrix(byte[] baseWord, byte[,] baseMatrix)

{

//заполняем матрицу словом

Console.WriteLine("Заполнение матрицы словом");

byte[,] filledMatrix = FillMatrix(baseMatrix, baseWord);

ShowMatrix(filledMatrix);

//получаем горизонтальный (1) паритет

Console.WriteLine("Получение паритета горизонтального (первого)");

byte[] firstParity = GetFirstParity(filledMatrix);

ShowWord(firstParity);

//получаем вертикальный (2) паритет

Console.WriteLine("Получение паритета вертикальный (второго)");

byte[] secondParity = GetSecondParity(filledMatrix);

ShowWord(secondParity);

//получаем диагональный (слева направо) паритет

Console.WriteLine("Получение паритета диагонального (слева направо)");

byte[] firstDiagParity = GetFirstDiagParity(filledMatrix);

ShowWord(firstDiagParity);

//получаем диагональный (справа налево) паритет

Console.WriteLine("Получение паритета диагональный (справа налево)");

byte[] secondDiagParity = GetSecondDiagParity(filledMatrix);

ShowWord(secondDiagParity);

//выводим всё в 1 строку

Console.WriteLine("Вывод");

byte[] fullWord = CreateFullWord(baseWord, firstParity, secondParity, firstDiagParity);

ShowWord(fullWord);

return fullWord;

}

public static byte[,] GetFirstParity(byte[,,] matrix)

{

byte[,] firstParity = new byte[matrix.GetLongLength(0), matrix.GetLongLength(1)];

byte sum;

for (int i = 0; i < matrix.GetLongLength(0); i++)

{

for (int j = 0; j < matrix.GetLongLength(1); j++)

{

sum = (byte)0;

for (int g = 0; g < matrix.GetLongLength(2); g++)

{

sum += (byte)(matrix[i, j, g]);

}

firstParity[i, j] = (byte)(sum % 2);

}

}

return firstParity;

}

public static byte[,] GetSecondParity(byte[,,] matrix)

{

byte[,] secondParity = new byte[matrix.GetLongLength(1), matrix.GetLongLength(2)];

byte sum;

for (int j = 0; j < matrix.GetLongLength(1); j++)

{

sum = (byte)0;

for (int g = 0; g < matrix.GetLongLength(2); g++)

{

for (int i = 0; i < matrix.GetLongLength(0); i++)

{

sum += (byte)(matrix[i, j, g]);

}

secondParity[j, g] = (byte)(sum % 2);

}

}

return secondParity;

}

public static byte[,] GetThirdParity(byte[,,] matrix)

{

byte[,] thirdParity = new byte[matrix.GetLongLength(2), matrix.GetLongLength(0)];

byte sum;

for (int g = 0; g < matrix.GetLongLength(2); g++)

{

for (int i = 0; i < matrix.GetLongLength(0); i++)

{

sum = (byte)0;

for (int j = 0; j < matrix.GetLongLength(1); j++)

{

sum += (byte)(matrix[i, j, g]);

}

thirdParity[g, i] = (byte)(sum % 2);

}

}

return thirdParity;

}

public static byte[,] GetFirstDiagParity(byte[,,] matrix)

{

byte[,] firstDiagParity = new byte[matrix.GetLongLength(0), matrix.GetLongLength(0)];

byte sum;

for (int g = 0; g < matrix.GetLongLength(2); g++)

{

for (int i = 0; i < matrix.GetLongLength(0); i++)

{

sum = (byte)0;

for (int j = 0; j < matrix.GetLongLength(1); j++)

{

Console.Write(matrix[(i + j) % matrix.GetLongLength(0), j % matrix.GetLongLength(1), g] + " ");

sum += (byte)(matrix[(i + j) % matrix.GetLongLength(0), j % matrix.GetLongLength(1), g]);

}

Console.WriteLine(sum);

firstDiagParity[g, i] = (byte)(sum % 2);

}

Console.WriteLine();

}

return firstDiagParity;

}

public static byte[,] GetSecondDiagParity(byte[,,] matrix)

{

byte[,] secondDiagParity = new byte[matrix.GetLongLength(0), matrix.GetLongLength(0)];

byte sum;

for (int g = 0; g < matrix.GetLongLength(2); g++)

{

for (int i = 0; i < matrix.GetLongLength(0); i++)

{

sum = (byte)0;

for (int j = 0; j < matrix.GetLongLength(1); j++)

{

Console.Write(matrix[(matrix.GetLongLength(0) - 1) - ((i + j) % matrix.GetLongLength(0)),

j % matrix.GetLongLength(1), g] + " ");

sum += (byte)(matrix[(matrix.GetLongLength(0) - 1) - ((i + j) % matrix.GetLongLength(0)),

j % matrix.GetLongLength(1), g]);

}

Console.WriteLine(sum);

secondDiagParity[g, i] = (byte)(sum % 2);

}

Console.WriteLine();

}

return secondDiagParity;

}

public static byte[] ConvertMatrixToString(byte[,] matrix)

{

byte[] str = new byte[matrix.GetLongLength(0) \* matrix.GetLongLength(1)];

for (int i = 0; i < matrix.GetLongLength(0); i++)

{

for (int j = 0; j < matrix.GetLongLength(1); j++)

{

str[i \* matrix.GetLongLength(1) + j] = matrix[i, j];

}

}

return str;

}

public static byte[] CreateFullWord(byte[] word,

byte[,] firstParityMatrix,

byte[,] secondParityMatrix,

byte[,] thirdParityMatrix,

byte[,] firstDiagParityMatrix,

byte[,] secondDiagParityMatrix)

{

byte[] firstParity = new byte[firstParityMatrix.GetLongLength(0) \* firstParityMatrix.GetLongLength(1)];

firstParity = ConvertMatrixToString(firstParityMatrix);

byte[] secondParity = new byte[secondParityMatrix.GetLongLength(0) \* secondParityMatrix.GetLongLength(1)];

secondParity = ConvertMatrixToString(secondParityMatrix);

byte[] thirdParity = new byte[thirdParityMatrix.GetLongLength(0) \* thirdParityMatrix.GetLongLength(1)];

thirdParity = ConvertMatrixToString(thirdParityMatrix);

byte[] firstDiagParity = new byte[firstDiagParityMatrix.GetLongLength(0) \* firstDiagParityMatrix.GetLongLength(1)];

firstDiagParity = ConvertMatrixToString(firstDiagParityMatrix);

byte[] secondDiagParity = new byte[secondDiagParityMatrix.GetLongLength(0) \* secondDiagParityMatrix.GetLongLength(1)];

secondDiagParity = ConvertMatrixToString(secondDiagParityMatrix);

return (((word.Concat(firstParity).ToArray())

.Concat(secondParity).ToArray())

.Concat(thirdParity).ToArray())

.Concat(firstDiagParity).ToArray()

.Concat(secondDiagParity).ToArray();

}

public static byte[] ActionsForThreeDimensialMatrix(byte[] baseWord, byte[,,] baseMatrix)

{

//заполняем матрицу словом

//Console.WriteLine("Заполнение матрицы словом");

byte[,,] filledMatrix = FillMatrix(baseMatrix, baseWord);

//получение горизонтального (1) паритет

Console.WriteLine("Получение паритета горизонтального (первого)");

byte[,] firstParity = GetFirstParity(filledMatrix);

ShowMatrix(firstParity);

//получение вертикального (2) паритет

Console.WriteLine("Получение паритета вертикальный (второго)");

byte[,] secondParity = GetSecondParity(filledMatrix);

ShowMatrix(secondParity);

//получение глубинный (3) паритет

Console.WriteLine("Получение паритета глубинного (третьего)");

byte[,] thirdParity = GetThirdParity(filledMatrix);

ShowMatrix(thirdParity);

Console.WriteLine("Получение паритета диагонального (слева направо)");

byte[,] firstDiagParity = GetFirstDiagParity(filledMatrix);

ShowMatrix(firstDiagParity);

Console.WriteLine("Получение паритета диагональный (справа налево)");

byte[,] secondDiagParity = GetSecondDiagParity(filledMatrix);

ShowMatrix(secondDiagParity);

//выводим всё в 1 строку

Console.WriteLine("Вывод");

byte[] fullWord = CreateFullWord(baseWord, firstParity, secondParity, thirdParity, firstDiagParity, secondDiagParity);

ShowWord(fullWord);

return fullWord;

}

public static void FindMistakesWithTwoParitiesThree(int k1, int k2, int k3, byte[] paritiesWithMistakes, byte[] newParities)

{

//сравним паритеты

Console.WriteLine("Сравнение синдромов");

ShowWord(paritiesWithMistakes);

ShowWord(newParities);

//получим синдромы

Console.WriteLine("Получение синдромов");

byte[] syndrom = GetSyndrom(paritiesWithMistakes, newParities);

ShowWord(syndrom);

//распарсим синдромы паритетов

Console.WriteLine("Распаковка синдром паритетов");

byte[] firstParitySyndrom = new byte[k1 \* k2];

Array.Copy(syndrom, 0, firstParitySyndrom, 0, k1 \* k2);

byte[] secondParitySyndrom = new byte[k2 \* k3];

Array.Copy(syndrom, k1 \* k2, secondParitySyndrom, 0, k2 \* k3);

ShowWord(firstParitySyndrom);

ShowWord(secondParitySyndrom);

//распарсим синдромы паритетов

Console.WriteLine("Распаковка синдром паритетов");

byte[,] firstParitySyndromMatrix = new byte[k1, k2];

for (int i = 0; i < k1; i++)

{

for (int j = 0; j < k2; j++)

{

firstParitySyndromMatrix[i, j] = syndrom[i \* k2 + j];

}

}

byte[,] secondParitySyndromMatrix = new byte[k2, k3];

for (int i = 0; i < k2; i++)

{

for (int j = 0; j < k3; j++)

{

secondParitySyndromMatrix[i, j] = syndrom[(k1 \* k2) + i \* k3 + j];

}

}

Console.WriteLine("Первый синдром");

ShowMatrix(firstParitySyndromMatrix);

Console.WriteLine("Второй синдром");

ShowMatrix(secondParitySyndromMatrix);

//ИСПРАВИМ ОШИБКИ

List<int> errorsNumbers = new List<int>();

for (int fp0 = 0; fp0 < firstParitySyndromMatrix.GetLongLength(0); fp0++)

{

for (int fp1 = 0; fp1 < firstParitySyndromMatrix.GetLongLength(1); fp1++)

{

if (firstParitySyndromMatrix[fp0, fp1] == (byte)1)

{

for (int sp0 = 0; sp0 < secondParitySyndromMatrix.GetLongLength(0); sp0++)

{

for (int sp1 = 0; sp1 < secondParitySyndromMatrix.GetLongLength(1); sp1++)

{

if (secondParitySyndromMatrix[sp0, sp1] == (byte)1)

{

errorsNumbers.Add((int)(fp0 \* firstParitySyndromMatrix.GetLongLength(0)

+ fp1

+ sp0 \* secondParitySyndromMatrix.GetLongLength(0)

+ sp1));

}

}

}

}

}

}

errorsNumbers = errorsNumbers.Distinct().ToList();

errorsNumbers = errorsNumbers.OrderBy(x => x).ToList();

foreach (var item in errorsNumbers)

{

Console.WriteLine("Ошибка в бите " + item);

}

}

public static void FindMistakesWithThreeParitiesThree(int k1, int k2, int k3, byte[] paritiesWithMistakes, byte[] newParities)

{

//сравним паритеты

Console.WriteLine("Сравнение паритетов");

ShowWord(paritiesWithMistakes);

ShowWord(newParities);

//получим синдромы

Console.WriteLine("Получение синдрома");

byte[] syndrom = GetSyndrom(paritiesWithMistakes, newParities);

ShowWord(syndrom);

//распарсим синдромы паритетов

Console.WriteLine("Распаковка синдрома паритетов");

byte[,] firstParitySyndrom = new byte[k1, k2];

for (int i = 0; i < k1; i++)

{

for (int j = 0; j < k2; j++)

{

firstParitySyndrom[i, j] = syndrom[i \* k2 + j];

}

}

byte[,] secondParitySyndrom = new byte[k2, k3];

for (int i = 0; i < k2; i++)

{

for (int j = 0; j < k3; j++)

{

secondParitySyndrom[i, j] = syndrom[(k1 \* k2) + i \* k3 + j];

}

}

byte[,] thirdParitySyndrom = new byte[k3, k1];

for (int i = 0; i < k3; i++)

{

for (int j = 0; j < k1; j++)

{

thirdParitySyndrom[i, j] = syndrom[(k3 \* k1) + (k1 \* k2) + i \* k1 + j];

}

}

Console.WriteLine("Первый синдром");

ShowMatrix(firstParitySyndrom);

Console.WriteLine("Второй синдром");

ShowMatrix(secondParitySyndrom);

Console.WriteLine("Третий синдром");

ShowMatrix(thirdParitySyndrom);

//ИСПРАВИМ ОШИБКИ

List<int> errorsNumbers = new List<int>();

for (int fp0 = 0; fp0 < firstParitySyndrom.GetLongLength(0); fp0++)

{

for (int fp1 = 0; fp1 < firstParitySyndrom.GetLongLength(1); fp1++)

{

if (firstParitySyndrom[fp0, fp1] == (byte)1)

{

for (int sp0 = 0; sp0 < secondParitySyndrom.GetLongLength(0); sp0++)

{

for (int sp1 = 0; sp1 < secondParitySyndrom.GetLongLength(1); sp1++)

{

if (secondParitySyndrom[sp0, sp1] == (byte)1)

{

for (int thp0 = 0; thp0 < thirdParitySyndrom.GetLongLength(0); thp0++)

{

for (int thp1 = 0; thp1 < thirdParitySyndrom.GetLongLength(1); thp1++)

{

if (thirdParitySyndrom[thp0, thp1] == (byte)1)

{

errorsNumbers.Add((int)

(fp0 \* firstParitySyndrom.GetLongLength(1)

+ (fp1 + 1) - 1

+ sp0 \* thirdParitySyndrom.GetLongLength(1)

+ (sp1 + 1) - 1

+ thp0 \* thirdParitySyndrom.GetLongLength(1)

+ (thp1 + 1) - 1)

);

}

}

}

}

}

}

}

}

}

errorsNumbers = errorsNumbers.Distinct().ToList();

errorsNumbers = errorsNumbers.OrderBy(x => x).ToList();

foreach (var item in errorsNumbers)

{

Console.WriteLine("Ошибка в бите " + item);

}

}

public static void FindMistakesWithFourParitiesThree(int k1, int k2, int k3, byte[] paritiesWithMistakes, byte[] newParities)

{

//сравним паритеты

Console.WriteLine("Сравнение паритеты");

ShowWord(paritiesWithMistakes);

ShowWord(newParities);

//получим синдромы

Console.WriteLine("Получение синдром");

byte[] syndrom = GetSyndrom(paritiesWithMistakes, newParities);

ShowWord(syndrom);

//распарсим синдромы паритетов

Console.WriteLine("Распаковка синдрома паритетов");

byte[,] firstParitySyndrom = new byte[k1, k2];

for (int i = 0; i < k1; i++)

{

for (int j = 0; j < k2; j++)

{

firstParitySyndrom[i, j] = syndrom[i \* k2 + j];

}

}

byte[,] secondParitySyndrom = new byte[k2, k3];

for (int i = 0; i < k2; i++)

{

for (int j = 0; j < k3; j++)

{

secondParitySyndrom[i, j] = syndrom[(k1 \* k2) + i \* k3 + j];

}

}

byte[,] thirdParitySyndrom = new byte[k3, k1];

for (int i = 0; i < k3; i++)

{

for (int j = 0; j < k1; j++)

{

thirdParitySyndrom[i, j] = syndrom[(k3 \* k1) + (k1 \* k2) + i \* k1 + j];

}

}

byte[,] fourParitySyndrom = new byte[k1, k3];

for (int i = 0; i < k1; i++)

{

for (int j = 0; j < k3; j++)

{

fourParitySyndrom[i, j] = syndrom[(k3 \* k1) + (k1 \* k2) + (k1 \* k3) + i \* k3 + j];

}

}

byte[,,] syndromMatrixThreeDimensial = new byte[k1, k2, k3];

Console.WriteLine("Первый синдром");

ShowMatrix(firstParitySyndrom);

Console.WriteLine("Второй синдром");

ShowMatrix(secondParitySyndrom);

Console.WriteLine("Третий синдром");

ShowMatrix(thirdParitySyndrom);

Console.WriteLine("Четвёртый синдром");

ShowMatrix(fourParitySyndrom);

//ИСПРАВИМ ОШИБКИ

for (int fp0 = 0; fp0 < k1; fp0++)

{

for (int fp1 = 0; fp1 < k2; fp1++)

{

for (int i = 0; i < k3; i++)

{

syndromMatrixThreeDimensial[fp0, fp1, i] = firstParitySyndrom[fp0, fp1];

}

}

}

for (int sp0 = 0; sp0 < k2; sp0++)

{

for (int sp1 = 0; sp1 < k3; sp1++)

{

for (int i = 0; i < k1; i++)

{

if (syndromMatrixThreeDimensial[i, sp0, sp1] == (byte)1)

syndromMatrixThreeDimensial[i, sp0, sp1] = secondParitySyndrom[sp0, sp1];

}

}

}

for (int thp0 = 0; thp0 < k3; thp0++)

{

for (int thp1 = 0; thp1 < k1; thp1++)

{

for (int i = 0; i < k1; i++)

{

if (syndromMatrixThreeDimensial[thp1, i, thp0] == (byte)1)

syndromMatrixThreeDimensial[thp1, i, thp0] = thirdParitySyndrom[thp0, thp1];

}

}

}

for (int i = 0; i < k3; i++)

{

for (int fd0 = 0; fd0 < k1; fd0++)

{

for (int fd1 = 0; fd1 < k2; fd1++)

{

if (syndromMatrixThreeDimensial[(fd0 + fd1) % k1, (fd1) % k2, i] == (byte)1)

syndromMatrixThreeDimensial[(fd0 + fd1) % k1, (fd1) % k2, i] = fourParitySyndrom[i, fd0];

}

}

}

List<int> errorsNumbers = new List<int>();

for (int i = 0; i < k1; i++)

{

for (int j = 0; j < k2; j++)

{

for (int g = 0; g < k3; g++)

{

if (syndromMatrixThreeDimensial[i, j, g] == (byte)1)

{

errorsNumbers.Add((int)

i \* k2

+ j \* k3

+ g

);

}

}

}

}

errorsNumbers = errorsNumbers.Distinct().ToList();

errorsNumbers = errorsNumbers.OrderBy(x => x).ToList();

foreach (var item in errorsNumbers)

{

Console.WriteLine("Ошибка в бите " + item);

}

}

public static void FindMistakesWithFiveParitiesThree(int k1, int k2, int k3, byte[] paritiesWithMistakes, byte[] newParities)

{

//сравним паритеты

Console.WriteLine("Сравнение паритеты");

ShowWord(paritiesWithMistakes);

ShowWord(newParities);

//получим синдромы

Console.WriteLine("Получение синдром");

byte[] syndrom = GetSyndrom(paritiesWithMistakes, newParities);

ShowWord(syndrom);

//распарсим синдромы паритетов

Console.WriteLine("Распаковка синдрома паритетов");

byte[,] firstParitySyndrom = new byte[k1, k2];

for (int i = 0; i < k1; i++)

{

for (int j = 0; j < k2; j++)

{

firstParitySyndrom[i, j] = syndrom[i \* k2 + j];

}

}

byte[,] secondParitySyndrom = new byte[k2, k3];

for (int i = 0; i < k2; i++)

{

for (int j = 0; j < k3; j++)

{

secondParitySyndrom[i, j] = syndrom[(k1 \* k2) + i \* k3 + j];

}

}

byte[,] thirdParitySyndrom = new byte[k3, k1];

for (int i = 0; i < k3; i++)

{

for (int j = 0; j < k1; j++)

{

thirdParitySyndrom[i, j] = syndrom[(k3 \* k1) + (k1 \* k2) + i \* k1 + j];

}

}

byte[,] fourParitySyndrom = new byte[k1, k3];

for (int i = 0; i < k1; i++)

{

for (int j = 0; j < k3; j++)

{

fourParitySyndrom[i, j] = syndrom[(k3 \* k1) + (k1 \* k2) + (k1 \* k3) + i \* k3 + j];

}

}

byte[,] fifthParitySyndrom = new byte[k1, k3];

for (int i = 0; i < k3; i++)

{

for (int j = 0; j < k1; j++)

{

fifthParitySyndrom[i, j] = syndrom[(k3 \* k1) + (k1 \* k2) + (k1 \* k3) + (k3 \* k1) + i \* k1 + j];

}

}

byte[,,] syndromMatrixThreeDimensial = new byte[k1, k2, k3];

Console.WriteLine("Первый синдром");

ShowMatrix(firstParitySyndrom);

Console.WriteLine("Второй синдром");

ShowMatrix(secondParitySyndrom);

Console.WriteLine("Третий синдром");

ShowMatrix(thirdParitySyndrom);

Console.WriteLine("Четвёртый синдром");

ShowMatrix(fourParitySyndrom);

Console.WriteLine("Пятый синдром");

ShowMatrix(fifthParitySyndrom);

//ИСПРАВИМ ОШИБКИ

for (int fp0 = 0; fp0 < k1; fp0++)

{

for (int fp1 = 0; fp1 < k2; fp1++)

{

for (int i = 0; i < k3; i++)

{

syndromMatrixThreeDimensial[fp0, fp1, i] = firstParitySyndrom[fp0, fp1];

}

}

}

for (int sp0 = 0; sp0 < k2; sp0++)

{

for (int sp1 = 0; sp1 < k3; sp1++)

{

for (int i = 0; i < k1; i++)

{

if (syndromMatrixThreeDimensial[i, sp0, sp1] == (byte)1)

syndromMatrixThreeDimensial[i, sp0, sp1] = secondParitySyndrom[sp0, sp1];

}

}

}

for (int thp0 = 0; thp0 < k3; thp0++)

{

for (int thp1 = 0; thp1 < k1; thp1++)

{

for (int i = 0; i < k1; i++)

{

if (syndromMatrixThreeDimensial[thp1, i, thp0] == (byte)1)

syndromMatrixThreeDimensial[thp1, i, thp0] = thirdParitySyndrom[thp0, thp1];

}

}

}

for (int i = 0; i < k3; i++)

{

for (int fd0 = 0; fd0 < k1; fd0++)

{

for (int fd1 = 0; fd1 < k2; fd1++)

{

if (syndromMatrixThreeDimensial[(fd0 + fd1) % k1, (fd1) % k2, i] == (byte)1)

syndromMatrixThreeDimensial[(fd0 + fd1) % k1, (fd1) % k2, i] = fourParitySyndrom[i, fd0];

}

}

}

for (int i = 0; i < k3; i++)

{

for (int fd0 = 0; fd0 < k1; fd0++)

{

for (int fd1 = 0; fd1 < k2; fd1++)

{

if (syndromMatrixThreeDimensial[(k1 - 1) - (fd0 + fd1) % k1, (fd1) % k2, i] == (byte)1)

syndromMatrixThreeDimensial[(k1 - 1) - (fd0 + fd1) % k1, (fd1) % k2, i] = fifthParitySyndrom[i, fd0];

}

}

}

List<int> errorsNumbers = new List<int>();

for (int i = 0; i < k1; i++)

{

for (int j = 0; j < k2; j++)

{

for (int g = 0; g < k3; g++)

{

if (syndromMatrixThreeDimensial[i, j, g] == (byte)1)

{

errorsNumbers.Add((int)

i \* k2

+ j \* k3

+ g

);

}

}

}

}

errorsNumbers = errorsNumbers.Distinct().ToList();

errorsNumbers = errorsNumbers.OrderBy(x => x).ToList();

foreach (var item in errorsNumbers)

{

Console.WriteLine("Ошибка в бите " + item);

}

}

}