МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Отчет по лабораторной работе №7**

**ПЕРЕМЕЖЕНИЕ/ДЕПЕРЕМЕЖЕНИЕ ДАННЫХ В ИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ**

Выполнил:

Cтудент 3 курса 1 группы

Парибок И. А.

Вариант 5

Минск 2022

**Цель:** приобретение практических навыков использования  
методов перемежения/деперемежения двоичных данных в информационных системах.

**Практическое задание**

1. Необходимо разработать авторское приложение в соответствии с целью лабораторной работы. По умолчанию используется  
блочный перемежитель/деперемежитель. По желанию студент  
может использовать иной. Задание выполняется по указанию пре-  
подавателя в соответствии с вариантом из таблицы.

За основу разрабатываемого приложения может быть взято  
приложение из выполненной лабораторной работы, соответст-  
вующей заданному корректирующему коду.



Рисунок 1 – Список вариантов

2. Местоположение заданной группы ошибок выбирается (генерируется) случайным образом. Необходимо для группы ошибок  
каждой длины сгенерировать 30−40 случайных ситуаций. После  
деперемежения и исправления ошибок в сообщении сравнить передаваемую последовательность и полученную после исправления ошибок. Проанализировать эффективность перемежения/де-  
перемежения.

3. Результаты оформить в виде отчета по установленным  
правилам.

**Выполнение работы:**

Результаты выполнения задания представлены в рисунках 2-7. Листинг программы изложен в приложении А.



Рисунок 2 – Информационное сообщение

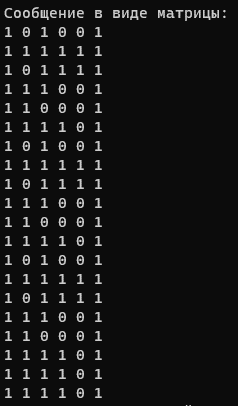


Рисунок 3 – Сообщение в виде матрицы

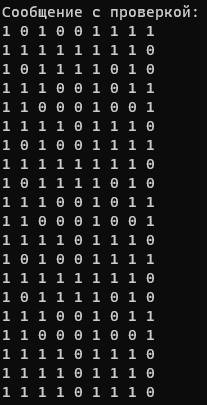


Рисунок 4 – Сообщение с проверочными битами

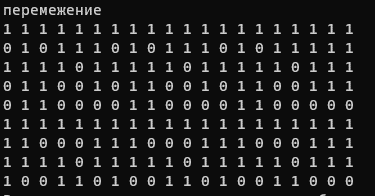


Рисунок 5 – Перемежение таблицы

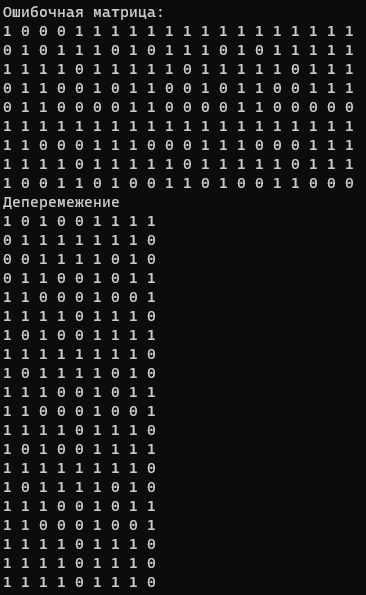


Рисунок 6 – Матрица с блоком ошибок и её деперемежение

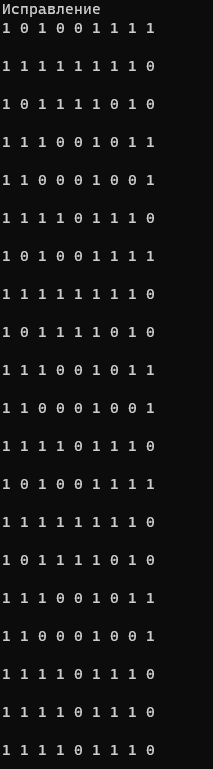


Рисунок 7 – Построчное исправление матрицы

**Вывод:** В результате выполнения работы были изучены принципы работы с двоичными данными в информационных системах и приобретены практические навыки их перемежения/деперемежения.

В ходе 10 итераций проверки исправления ошибок, 8 раз исправление ошибок было успешно. На основе данной выборке шанс полного исправления ошибок при переда сообщения равен 80.

Перемежение и деперемежение являются важными операциями в работе с двоичными данными, которые позволяют хранить и передавать информацию в удобном виде**.**

**Приложение А**

using System;

namespace Lab6

{

class Program

{

static void Main(string[] args)

{

//variant 5

string g = "100101";

int pos, pos2;

int r = 3;

int k = 6;

int n = r + k;

int error;

//полином

int[] masXr = new int[g.Length];

StrInMas(masXr, g);

int[,] generationMatrix = new int[k, n];

CreateGenerationMatrix(generationMatrix, masXr, k, n);

//generation and masxr не меняются

CreateCanonicalMatrix(generationMatrix, k, n);

int[,] checkMatrix = new int[n, r];

CreateCheckMatrix(checkMatrix, generationMatrix, k, n);

string inputString = "101001111111101111111001110001111101101001111111101111111001110001111101101001111111101111111001110001111101111101111101";

int columnCount = 6;

int rowCount = (int)Math.Ceiling((double)inputString.Length / columnCount);

int [,] matrix = new int[rowCount, columnCount];

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

{

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

{

int index = i \* columnCount + j;

if (index < inputString.Length)

{

matrix[i, j] = Convert.ToInt32(inputString[index].ToString());

}

else

{

matrix[i, j] = 0;

}

}

}

Console.WriteLine("Сообщение в виде матрицы: ");

PrintMatrix(matrix);

int[,] totalMatrix = new int[rowCount, n];

int[] masXk = new int[columnCount];

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

{

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

{

masXk[j] = matrix[i, j];

}

int[] masXn = new int[n];

Shift(masXn, masXk, r);

SearchResidue(masXn, masXr);

//Console.WriteLine("Итоговая строка:");

Shift(masXn, masXk, r);

//OutMass(masXn);

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

{

totalMatrix[i, q] = masXn[q];

}

}

Console.WriteLine("Сообщение c проверкой: ");

PrintMatrix(totalMatrix);

int[,] peremelists = new int[n, rowCount];

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

{

for (int o = 0; o < rowCount; o++)

{

peremelists[i, o] = totalMatrix[o,i];

}

}

Console.WriteLine("перемежение");

PrintMatrix(peremelists);

Console.WriteLine("Введите с какого символа начать ошибку");

int ErrorStartPostion = Convert.ToInt32(Console.ReadLine()) - 1;

int st = ErrorStartPostion / n;

int row = ErrorStartPostion % rowCount;

Console.WriteLine("Введите длину");

int ErrorLong = ErrorStartPostion + Convert.ToInt32(Console.ReadLine()) - 1;

int stlong = ErrorLong / n;

int rowlong = ErrorLong % rowCount;

for (; st <= stlong; st++)

{

if (st != stlong)

{

//меняем до конца строки

for (; row < k; row++)

{

if (peremelists[st, row] == 1) peremelists[st, row] = 0;

else peremelists[st, row] = 1;

}

row = 0;

}

else

{

for (; row <= rowlong; row++)

{

if (peremelists[st, row] == 1) peremelists[st, row] = 0;

else peremelists[st, row] = 1;

}

}

}

Console.WriteLine("Ошибочная матрица:");

PrintMatrix(peremelists);

////Console.WriteLine("Ошибочная строка:");

////OutMass(masXnError);

int[,] Deperemelists = new int[rowCount, n];

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

{

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

{

Deperemelists[i, o] = peremelists[o, i];

}

}

Console.WriteLine("Деперемежение");

PrintMatrix(Deperemelists);

Console.WriteLine("Исправление");

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

{

int[] masXn = new int[n];

for (int b = 0; b < 9; b++)

{

masXn[b] = Deperemelists[i, b];

}

OutMass(SearchError(masXn, masXr, checkMatrix, r));

}

Console.ReadKey();

}

#region Functions

public static int[] SearchError(int[] masXn, int[] masXr, int[,] checkMatrix, int r)

{

int n = masXn.Length;

int k = n - r;

int[] masXnSecond = new int[n];

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

{

masXnSecond[i] = masXn[i];

}

SearchResidue(masXnSecond, masXr);

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

{

int coincidence = 0;

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

{

if (checkMatrix[i, j] == masXnSecond[k + j])

{

coincidence++;

}

}

if (coincidence == r)

{

masXn[i] = (masXn[i] + 1) % 2;

break;

}

}

//Console.WriteLine("\nИсправленная строка:");

//OutMass(masXn);

return masXn;

}

public static void PrintMatrix(int[,] matrix)

{

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

{

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

{

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

}

Console.WriteLine();

}

}

public static int[] SearchResidue(int[] masXn, int[] masXr)

{

int end = masXn.Length - masXr.Length + 1;

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

{

if (masXn[i] == 1)

{

AddingMasMod2(masXn, masXr, i);

//OutMass(masXn);

}

}

return masXn;

}

//Сложение массивов по модулю 2 с опр. позиции

public static int[] AddingMasMod2(int[] mas1, int[] mas2, int pos)

{

int end = pos + mas2.Length;

for (int i = pos; i < end; i++)

{

mas1[i] = (mas1[i] + mas2[i - pos]) % 2;

}

return mas1;

}

//Смещение на массива r

public static int[] Shift(int[] shiftMas, int[] mas, int r)

{

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

{

shiftMas[i] = mas[i];

}

return shiftMas;

}

//Преобразование сторки в массив

public static int[] StrInMas(int[] mas, string str)

{

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

{

if (str[i] == 49)

mas[i] = 1;

else mas[i] = 0;

}

return mas;

}

//Создание Порождающей матрицы

static int[,] CreateGenerationMatrix(int[,] generationMatrix, int[] mas, int k, int n)

{

//Заполняем первую строку в проверочной матрице

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

{

if (i < mas.Length)

{

generationMatrix[0, i] = mas[i];

}

else

{

generationMatrix[0, i] = 0;

}

}

//Сдвигаем каждую строки вправо от предыдущей

for (int i = 1; i < k; i++)

{

for (int j = 0; j < n - 1; j++)

{

generationMatrix[i, j + 1] = generationMatrix[i - 1, j];

}

generationMatrix[i, 0] = generationMatrix[i - 1, n - 1];

}

return generationMatrix;

}

//Приведение порождающей матрицы к каноническому виду

static int[,] CreateCanonicalMatrix(int[,] generationMatrix, int k, int n)

{

//Перебираем строки для преведению к каноническому виду

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

{

int i2 = i + 1;

//Перебираем элементы строки, но только до k-элемента

for (int j = i + 1; j < k; j++)

{

//если мы нашли единицу в строке, то...

if (generationMatrix[i, j] == 1)

{

//перебираем этот столбец, пока не найдем единицу

for (; i2 < k; i2++)

{

bool repeat = false;

//Если нашли, то складываем обе строки

if (generationMatrix[i2, j] == 1)

{

for (int j2 = j - 1; j2 > 0; j2--)

{

//Проверяем, есть ли до этой 1 еще 1, если есть то эту строку пропускаем

if (generationMatrix[i2, j2] == 1)

{

repeat = true;

}

}

if (repeat)

continue;

AddingLinesMatrixMod2(generationMatrix, i, i2, n);

i2++;

break;

}

}

}

}

}

return generationMatrix;

}

//Преобразование канонической матрицы в проверочную

static int[,] CreateCheckMatrix(int[,] checkMatrix, int[,] generationMatrix, int k, int n)

{

int r = n - k;

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

{

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

{

checkMatrix[i, j] = generationMatrix[i, k + j];

}

}

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

{

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

{

if (j == i - k)

{

checkMatrix[i, j] = 1;

}

else

{

checkMatrix[i, j] = 0;

}

}

}

return checkMatrix;

}

//Сложение строк матрицы

public static int[,] AddingLinesMatrixMod2(int[,] matrix, int str1, int str2, int lengthString)

{

//Console.WriteLine(str1 + " и " + str2);

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

{

matrix[str1, i] = (matrix[str1, i] + matrix[str2, i]) % 2;

}

return matrix;

}

//вывод матрицы

public static void OutMatrix(int[,] matrix, int k, int n)

{

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

{

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

{

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

//if (j + 1 == k) Console.Write("|");

}

}

}

//вывод одномерного массива

public static void OutMass(int[] mas)

{

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

{

//if (i == k) Console.Write("|");

Console.Write(mas[i] + " ");

}

Console.WriteLine("\n");

}

#endregion

}

}