Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Дисциплина «Защита информации и надёжность информационных систем»

**Лабораторная работа №4**

**Тема «ИЗБЫТОЧНОЕ КОДИРОВАНИЕ ДАННЫХ В ИНФОРМАЦИОННЫХ СИСТЕМАХ. ИТЕРАТИВНЫЕ КОДЫ»**

Выполнил:

Студент 3 курса 7 группы ФИТ

Тимошенко Д. В.   
 Проверила:   
 асс. Николайчук А. С.

Минск 2024

**Цель работы:** приобретение практических навыков кодирования/декодирования двоичных данных при использовании кода Хемминга.

**Задание на лабораторную работу**

1. Разработать собственное приложение, которое позволяет выполнять следующие операции:

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):

а) по двум;

б) по трем;

в) по 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 Y n;

5) определять местоположение ошибочных символов итеративным кодом в слове Y n в соответствии с используемыми группами паритетов по пункту (2) и исправлять ошибочные символы

(результат исправления – слово Y n’);

6) выполнять анализ корректирующей способности используемого кода (количественная оценка) путем сравнения соответствующих слов Xn и Y n’; 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.

Вариант

Приложение написано на языке программирования JavaScript. Ниже представлен листинг функций, используемых во всех задания, листинг 1.1.

|  |
| --- |
| const fs = require('fs');  function binary(text) {  let binaryStr = '';  for (let c of text) {  binaryStr += String(c.charCodeAt(0).toString(2)).padStart(8, '0');  }  return binaryStr;  }  function generateHammingMatrix(infoWord) {  let k = infoWord.length;  let r = 0;  while (Math.pow(2, r) < k + r + 1) {  r++;  }  let n = k + r;  console.log("k=", k, ", r=", r, ", n=", n);  let checkMatrix = Array.from({ length: r }, () => Array(n).fill(0));  for (let i = 0; i < r; i++) {  for (let j = 0; j < n; j++) {  checkMatrix[i][j] = ((j + 1) & (1 << i)) !== 0 ? 1 : 0;  }  }  return checkMatrix;  }  function computeRedundantBits(checkMatrix, infoWord) {  let redundantBits = '';  for (let i = 0; i < checkMatrix.length; i++) {  let sum = 0;  for (let j = 0; j < infoWord.length; j++) {  sum += checkMatrix[i][j] \* parseInt(infoWord[j], 2);  }  redundantBits += sum % 2;  }  return redundantBits;  }  function addErrors(word, numErrors) {  let wordArray = word.split('');  for (let i = 0; i < numErrors; i++) {  let errorPos = Math.floor(Math.random() \* word.length);  wordArray[errorPos] = wordArray[errorPos] === '0' ? '1' : '0';  }  return wordArray.join('');  }  function correctError(encodedWord, encodedWordErrors) {  let errorVector = '';  for (let i = 0; i < encodedWord.length; i++) {  errorVector += encodedWord[i] ^ encodedWordErrors[i];  }  console.log("Вектор ошибок: " + errorVector);  let correctedWord = '';  for (let i = 0; i < encodedWordErrors.length; i++) {  correctedWord += encodedWordErrors[i] ^ errorVector[i];  }  return correctedWord;  }  function computeSyndrome(originalWord, receivedWord) {  let syndrome = '';  for (let i = 0; i < receivedWord.length; i++) {  syndrome += receivedWord[i] ^ originalWord[i];  }  return syndrome;  } |

Листинг 1.1 – используемые функции

А также код выполнения самих заданий лабораторной работы, листинг 1.2.

|  |
| --- |
| (async () => {  let givenText = '';  try {  givenText = fs.readFileSync(\_\_dirname+'/input.txt', 'utf8');  } catch (e) {  console.error(e);  }  givenText = binary(givenText);  console.log("Xk:", givenText);  console.log("\n Матрица Хемминга(Hn,k)");  let checkMatrix = generateHammingMatrix(givenText);  checkMatrix.forEach(row => console.log(row.join(' ')));  let redunBits = computeRedundantBits(checkMatrix, givenText);  console.log("Xr: ", redunBits);  let encodedWord = givenText + redunBits;  console.log("Xn: " + encodedWord);  let T1E = addErrors(givenText, 1);  let redBits1Err = computeRedundantBits(checkMatrix, T1E);  console.log("Syndrom(1): ", computeSyndrome(givenText, T1E));  console.log("Yn(1): ", T1E);  console.log("Yr(1): ", redBits1Err);  let T2E = addErrors(givenText, 2);  let redBits2Err = computeRedundantBits(checkMatrix, T2E);  console.log("Syndrom(2): ", computeSyndrome(givenText, T2E));  console.log("Yn(2): ", T2E);  console.log("Yr(2): ", redBits2Err);  let cor1Err = correctError(givenText, T1E);  console.log("Correct(1): ", cor1Err.match(/.{1,8}/g).map(byte => String.fromCharCode(parseInt(byte, 2))).join(''));  console.log("Correct(1) + Yr(1): ", (cor1Err + redBits1Err));  let cor2Err = correctError(givenText, T2E);  console.log("Correct(2): ", cor2Err.match(/.{1,8}/g).map(byte => String.fromCharCode(parseInt(byte, 2))).join(''));  console.log("Correct(2) + Yr(2): ", (cor1Err + redBits2Err));  })(); |

Листинг 1.2 – выполнение задания лабораторной работы

Вывод представленного кода, рисунок 1.1.

Рисунок 1.1

**Вывод:** в ходе выполнения лабораторной работы было выполнено преобразование сообщения в бинарный вид, построена проверочная матрица Хемминга. С использованием матрицы было вычислено избыточное слово. Используя матрицу Хемминга, был вычислен синдром и по синдрому было установлено количество ошибок, также была предпринята попытка их исправить.