Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Дисциплина «Защита информации и надёжность информационных систем»

**Лабораторная работа №5**

**Тема «ИЗБЫТОЧНОЕ КОДИРОВАНИЕ ДАННЫХ В ИНФОРМАЦИОННЫХ СИСТЕМАХ. ЦИКЛИЧЕСКИЕ 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 циклических кодов (ЦК).

**Задание на лабораторную работу**

1. Задание выполняется по указанию преподавателя в соответствии с вариантом из табл. 6.2, из которого выбирается порождающий полином ЦК, а по значению соответствующего ему значения r – длина k информационного слова Xk. Полагаем, что каждый полином соответствует коду, обнаруживающему и исправляющему одиночные ошибки в кодовых словах. Определить параметры (n, k)-кода для своего варианта. Основой задания является разработка приложения.

2. Составить порождающую матрицу (n, k)-кода в соответствии с формулой (6.7), трансформировать ее в каноническую форму и далее – в проверочную матрицу канонической формы.

3. Используя порождающую матрицу ЦК, вычислить избыточные символы (слово Xr) кодового слова Xn и сформировать это кодовое слово.

4. Принять кодовое слово Y n со следующим числом ошибок: 0; 1; 2. Позиция ошибки определяется (генерируется) случайным образом.

5. Для полученного слова Y n вычислить и проанализировать синдром. В случае, если анализ синдрома показал, что информационное сообщение было передано с ошибкой (или 2 ошибками), сгенерировать унарный вектор ошибки Еn = е1, е2, …, еn и исправить одиночную ошибку, используя выражение (6.5); проанализировать ситуацию при возникновении ошибки в 2 битах.

6. Результаты оформить в виде отчета по установленным правилам.

Вариант 11



Циклические коды − это семейство помехоустойчивых кодов, одной из разновидностей которых являются коды Хемминга.

Основные свойства ЦК:

• относятся к классу линейных, систематических;

• сумма по модулю 2 двух разрешенных кодовых комбинаций дает также разрешенную кодовую комбинацию;

• каждый вектор (кодовое слово), получаемый из исходного кодового вектора путем циклической перестановки его символов, также является разрешенным кодовым вектором; к примеру, если кодовое слово имеет следующий вид: 1101100, то разрешенной кодовой комбинацией будет и такая: 0110110;

• при простейшей циклической перестановке символы кодового слова перемещаются слева направо на одну позицию, как в приведенном примере;

• поскольку к числу разрешенных кодовых комбинаций ЦК относится нулевая комбинация 000...00, то минимальное кодовое расстояние dmin для ЦК определяется минимальным весом разрешенной кодовой комбинации;

• циклический код не обнаруживает только такие искаженные помехами кодовые комбинации, которые приводят к появлению на стороне приема других разрешенных комбинаций этого кода;

• в основе описания и использования ЦК лежит полином или многочлен некоторой переменной (обычно Х).

Формирование разрешенных кодовых комбинаций ЦК Bj(X) основано на предварительном выборе порождающего (генераторного или образующего) полинома G(X), который обладает важным отличительным признаком: все комбинации Bj(X) делятся на порождающий полином G(X) без остатка:

Bj(X) / G(X) = Aj(X), (6.2)

здесь Bj(X) = Xn – кодовое слово; Aj(X) = Xk – информационное слово.

Степень порождающего полинома определяет число проверочных символов: r = n – k.

Порождающими могут быть только такие полиномы, которые являются делителями двучлена (бинома) Х^z + 1:

(Х^z + 1) / G(X) = H(X), (6.4)

при нулевом остатке: R(X) = 0.

Деление полиномов позволяет представить кодовые слова в виде блочного кода, т. е. информационных Хk (Аi(Х)) и проверочных Хr (Ri(X)) символов.

Поскольку число последних равно r, то для компактной их записи в младшие разряды кодового слова надо предварительно к кодируемому (информационному) слову Аi(Х) справа дописать r нулевых символов.

Основная операция: принятое кодовое слово (Yn) нужно поделить на по-  
рождающий полином, который использовался при кодировании.

Деление принятого кодового слова на G(X) формально запишем в следующем виде:

Yn / G(X) = U, Sr, (6.6)

где Sr − остаток от деления (Yn) / (G(X)), или синдром.

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

|  |
| --- |
| function encodingToBytes(message: string): string {  let bin: string = '';  for (let ch of message) {  bin += ch.charCodeAt(0).toString(2);  }  return bin;  }  function encodingFromBytes(binary: number[], r : number): string {  let binaryString = binary.join('').slice(0, binary.length - r);  let text = '';  for (let i = 0; i < binaryString.length; i += 7) {  let byte = binaryString.slice(i, i + 7);  let charCode = parseInt(byte, 2);  text += String.fromCharCode(charCode);  }  return text;  }  function hemingLength(k: number): number {  let r: number = Math.ceil(Math.log2(k));  return r;  }  function searchError(masXn: number[], masG: number[], checkMatrix: number[][], r: number): void {  let n = masXn.length;  let k = n - r;  let masXnSecond = [...masXn];  console.log("\nДеление");  searchResidue(masXnSecond, masG);  console.log("\nОстаток:");  printArr(masXnSecond);  console.log("Синдром: ", masXnSecond.join('').slice(masXnSecond.indexOf(1), masXnSecond.length));  for (let i = 0; i < n; i++) {  let coincidence = 0;  for (let j = 0; j < r; j++) {  if (checkMatrix[i][j] === masXnSecond[k + j]) {  coincidence++;  }  }  if (coincidence === r) {  masXn[i] = (masXn[i] + 1) % 2;  break;  }  }  console.log("\nИсправленная строка:");  printArr(masXn);  console.log(encodingFromBytes(masXn, r));  }  function addLineMatrixMod2(matrix: number[][], str1: number, str2: number, lengthString: number): number[][] {  for (let i = 0; i < lengthString; i++) {  matrix[str1][i] = (matrix[str1][i] + matrix[str2][i]) % 2;  }  return matrix;  }  function searchResidue(masXn: number[], masXr: number[]): number[] {  let end = masXn.length - masXr.length + 1;  for (let i = 0; i < end; i++) {  if (masXn[i] === 1) {  addArrMod2(masXn, masXr, i);  printArr(masXn);  }  }  return masXn;  }  function addArrMod2(mas1: number[], mas2: number[], pos: number): number[] {  let end = pos + mas2.length;  for (let i = pos; i < end; i++) {  mas1[i] = (mas1[i] + mas2[i - pos]) % 2;  }  return mas1;  }  function shift(shiftMas: number[], mas: number[]): number[] {  for (let i = 0; i < mas.length; i++) {  shiftMas[i] = mas[i];  }  return shiftMas;  }  function createGenerationMatrix(mas: number[], k: number, n: number): number[][] {  let generationMatrix: number[][] = Array(k).fill(0).map(() => Array(n).fill(0));  for (let i = 0; i < n; i++) {  if (i < mas.length) {  generationMatrix[0][i] = mas[i];  } else {  generationMatrix[0][i] = 0;  }  }  for (let i = 1; i < k; i++) {  for (let j = 0; j < n - 1; j++) {  generationMatrix[i][j + 1] = generationMatrix[i - 1][j];  }  generationMatrix[i][0] = generationMatrix[i - 1][n - 1];  }  return generationMatrix;  }  function createCanonicalMatrix(generationMatrix: number[][], k: number, n: number): number[][] {  for (let i = 0; i < k; i++) {  let i2 = i + 1;  for (let j = i + 1; j < k; j++) {  if (generationMatrix[i][j] === 1) {  for (; i2 < k; i2++) {  let repeat = false;  if (generationMatrix[i2][j] === 1) {  for (let j2 = j - 1; j2 > 0; j2--) {  if (generationMatrix[i2][j2] === 1) {  repeat = true;  }  }  if (repeat) continue;  addLineMatrixMod2(generationMatrix, i, i2, n);  i2++;  break;  }  }  }  }  }  return generationMatrix;  }  function createCheckMatrix(generationMatrix: number[][], k: number, n: number): number[][] {  let r = n - k;  let checkMatrix: number[][] = Array(n).fill(0).map(() => Array(r).fill(0));  for (let i = 0; i < k; i++) {  for (let j = 0; j < r; j++) {  checkMatrix[i][j] = generationMatrix[i][k + j];  }  }  for (let i = k; i < n; i++) {  for (let j = 0; j < r; j++) {  if (j === i - k) {  checkMatrix[i][j] = 1;  } else {  checkMatrix[i][j] = 0;  }  }  }  return checkMatrix;  }  function outMatrix(matrix: number[][], k: number, n: number): void {  for (let i = 0; i < k; i++) {  console.log(matrix[i].join(''));  }  console.log();  }  function outMatrixProv(matrix: number[][], r: number, n: number): void {  for (let i = 0; i < n; i++) {  console.log(matrix[i].join(''));  }  console.log();  }  function strInMas(mas: number[], str: string): number[] {  for (let i = 0; i < str.length; i++) {  if (str[i] === '1') {  mas[i] = 1;  } else {  mas[i] = 0;  }  }  return mas;  }  function printArr(mas: number[]): void {  console.log(mas.join(''));  } |

Листинг 1.1 – используемые функции

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

|  |
| --- |
| const message = 'hillellr';  let encodingMessage: string = encodingToBytes(message);    let G: string = "1100001";  let k: number = encodingMessage.length;  let r: number = hemingLength(k);  let n: number = k + r;    let masXk: number[] = new Array(k).fill(0);  strInMas(masXk, encodingMessage);  let masG: number[] = new Array(G.length).fill(0);  strInMas(masG, G);    console.log("Входная строка: " + message);  console.log(`Введенное сообщение в двоичном виде: ${encodingMessage}`);  console.log("Порождающий полином: " + G);  console.log(`k = ${k}, r = ${r}, n = ${n}`);    let generationMatrix: number[][] = createGenerationMatrix(masG, k, n);  console.log("\nПорождающая матрица:");  outMatrix(generationMatrix, k, n);    console.log();  console.log("Сложение следующих строк: ");  createCanonicalMatrix(generationMatrix, k, n);  console.log("\nКаноническая матрица: ");  outMatrix(generationMatrix, k, n);    let checkMatrix: number[][] = createCheckMatrix(generationMatrix, k, n);  console.log("\nПроверочная матрица в канонической форме: ");  outMatrixProv(checkMatrix, r, n);    let masXn: number[] = new Array(n).fill(0);  shift(masXn, masXk);    searchResidue(masXn, masG);  console.log("\nОстаток:");  const sindrom : number[] = [];  shift(sindrom, masXn);  printArr(masXn);    console.log("Кодовое слово:");  shift(masXn, masXk);  printArr(masXn);  console.log(encodingFromBytes(masXn, r));  let numberOfErrors: number = Math.round(Math.random() + 1);  let vector : number[] = new Array(n).fill(0);  console.log("Колво ошибок: ", numberOfErrors);  switch (numberOfErrors) {  case 1:  let errorPosition: number = Math.round(Math.random()\*k);  console.log("Позиция ошибки: " + errorPosition);  masXn[errorPosition] = masXn[errorPosition] === 1 ? 0 : 1;  vector[errorPosition] = 1;  console.log();  break;  case 2:  let errorPosition1: number = Math.round(Math.random()\*k);  let errorPosition2: number;  do {  errorPosition2 = Math.round(Math.random()\*k);  } while (errorPosition1 === errorPosition2);  console.log(`Позиция первой ошибки: ${errorPosition1}, позиция второй: ${errorPosition2}`);  masXn[errorPosition1] = masXn[errorPosition1] === 1 ? 0 : 1;  masXn[errorPosition2] = masXn[errorPosition2] === 1 ? 0 : 1;  vector[errorPosition1] = 1;  vector[errorPosition2] = 1;  console.log();  break;  default:  console.log("Ошибочка");  console.log();  break;  }  console.log("Ошибочная строка:");  printArr(masXn);  console.log("Вектор ошибки: ", vector.join(''));  console.log("Синдром: ", sindrom.join('').slice(sindrom.indexOf(1), sindrom.length));    searchError(masXn, masG, checkMatrix, r); |

Листинг 1.2 – выполнение задания лабораторной работы

Вывод представленного кода, рисунок 1.1.

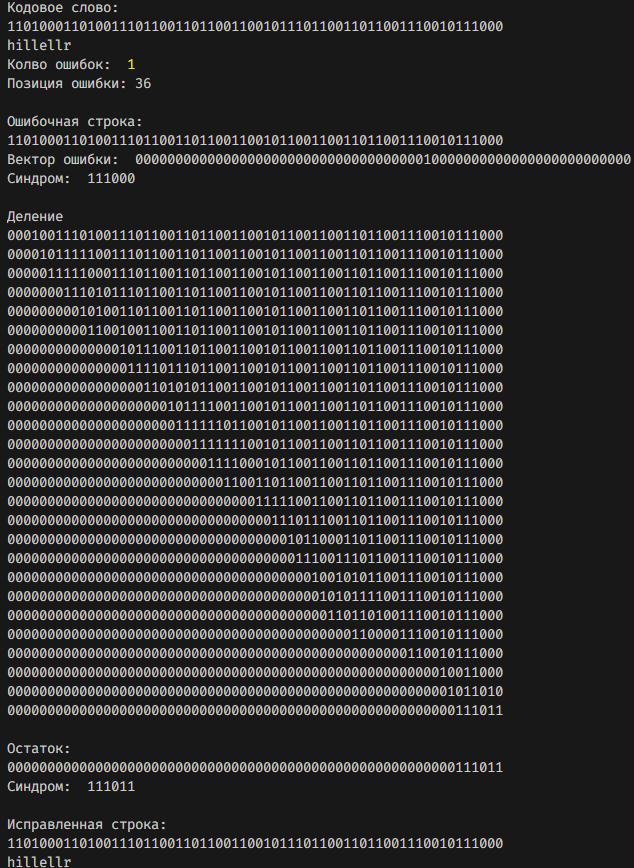


Рисунок 1.1

**Вывод:** в ходе выполнения лабораторной работы были определены (n, k) соответствующие r=6, было выполнено построение матриц (порождающей и проверочной), матрицы были приведены к канонической форме, были вычислены проверочные биты и сформировано кодовое слово, были созданы, обнаружены и исправлены ошибки (при возможности их исправления).