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 №6 НА ТЕМУ:**

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

Выполнил студент 3 курса 1 группы

Халалеенко Андрей Николаевич

Минск 2024

**Задание 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.

**Задание 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.

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

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

**Задание 5:** Для полученного слова *Yn* 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.

Для выполнения заданий был использован следующий код на языке *typescript*:

import Operations from './operations';

class Program {

    static Main(): void {

        const Xk: string = "10010111000110101110000100";

        const Xr: string = "100101";

        const k: number = Xk.length;

        const n: number = 31;

        const r: number = 5;

        let error: number;

        const masXk: number[] = new Array<number>(k);

        Operations.StringInArray(masXk, Xk);

        const masXr: number[] = new Array<number>(Xr.length);

        Operations.StringInArray(masXr, Xr);

        console.log(`\t\tВходная строка xk: ${Xk}`);

        console.log(`\t\tПорождающий полином xr в двоичном виде: ${Xr}`);

        console.log(`\t\t\t k = ${k}, r = ${r}, n = ${n}`);

        console.log("--------------------------------------------------------------------------------");

        const generationMatrix: number[][] = Array.from({ length: k }, () => Array(n).fill(0));

        Operations.CreateGenerationMatrix(generationMatrix, masXr, k, n);

        console.log("\n\t\t\t\tПорождающая матрица\n");

        Operations.PrintMatrix(generationMatrix, k, n);

        Operations.CreateMatrixCanon(generationMatrix, k, n);

        console.log("--------------------------------------------------------------------------------");

        console.log("\n\t\t\t\tКаноническая матрица\n");

        Operations.PrintMatrix(generationMatrix, k, n);

        const checkMatrix: number[][] = Array.from({ length: n }, () => Array(r).fill(0));

        Operations.CreateMatrixForCheck(checkMatrix, generationMatrix, k, n);

        console.log("--------------------------------------------------------------------------------");

        console.log("\n\t\t\t\tПроверочная матрица канонической формы\n");

        Operations.PrintMatrix(checkMatrix, n, r);

        // Task 1.2

        const masXn: number[] = new Array(n);

        Operations.ShiftR(masXn, masXk, r);

        console.log("--------------------------------------------------------------------------------");

        // Task 2

        console.log("\n\t\t\t\t\tДеление\n");

        Operations.SearchRes(masXn, masXr);

        console.log("Остаток:");

        Operations.PrintArray(masXn);

        console.log("--------------------------------------------------------------------------------");

        console.log("\n\t\t\t\tИтоговая строка\n");

        Operations.ShiftR(masXn, masXk, r);

        Operations.PrintArray(masXn);

        console.log("--------------------------------------------------------------------------------");

        try {

            //console.log("Введите место первой ошибки");

            error = 1

            // error = parseInt(prompt() || "0") - 1;

            if (masXn[error] === 1) masXn[error] = 0;

            else masXn[error] = 1;

        } catch { }

        console.log("Ошибочная строка:");

        Operations.PrintArray(masXn);

        Operations.SearchingMistake(masXn, masXr, checkMatrix, r);

        console.log("--------------------------------------------------------------------------------");

        try {

            error = 2

            console.log(`Место первой ошибки: ${error}`);

            //error = parseInt(prompt() || "0") - 1;

            if (masXn[error] === 1)

                masXn[error] = 0;

            else masXn[error] = 1;

            error = 3

            console.log(`Место второй ошибки: ${error}`);

            //error = parseInt(prompt() || "0") - 1;

            if (masXn[error] === 1)

                masXn[error] = 0;

            else masXn[error] = 1;

        } catch { }

        console.log("Ошибочная строка:");

        Operations.PrintArray(masXn);

        Operations.SearchingMistake(masXn, masXr, checkMatrix, r);

    }

}

Program.Main();

Листинг 1 – задания

Результат работы итогового кода:

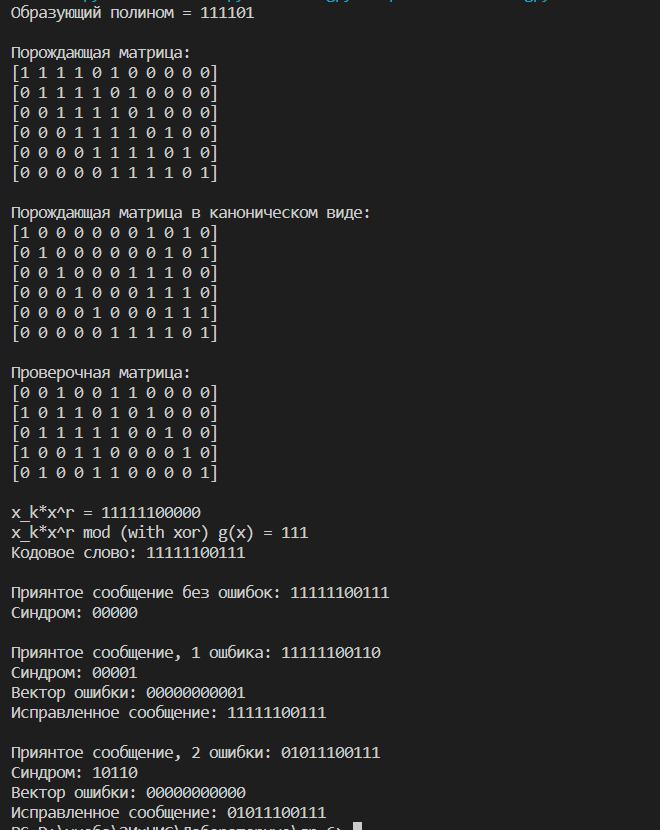


Рисунок 1 ­­­­– Результат работы программы

Данный код гарантирует исправление только 1 ошибки.

**Вывод:** в ходе лабораторной работы были приобретены практические навыки кодирования/декодирования двоичных данных при использовании циклических кодов.