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 №5 НА ТЕМУ:**

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

Выполнил студент 3 курса 1 группы

Халалеенко Андрей Николаевич

Минск 2024

**Задание 1:** Разработать собственное приложение, которое позволяет выполнять следующие операции: вписывать произвольное двоичное представление информационного слова *Xk* (кодируемой информации) длиной *k* битов в двумерную матрицу размерностью в соответствии с вариантом либо в трехмерную матрицу в соответствии с вариантом

**Задание 2:** вычислять проверочные биты (биты паритетов): а) по двум; б) по трем; в) по четырем направлениям (группам паритетов).

**Задание 3:** Формировать кодовое слово *Xn* присоединением избыточных символов к информационному слову.

Дополним разработанный в предыдущем задании код следующим образом:

**Задание 4:** Генерировать ошибку произвольной кратности (*i, i* > 0), распределенную случайным образом среди символов слова *Xn*, в результате чего формируется кодовое слово *Yn.*

**Задание 5:** Определять местоположение ошибочных символов итеративным кодом в слове *Yn* в соответствии с используемыми группами паритетов по пункту (2) и исправлять ошибочные символы (результат исправления – слово *Yn’*);

**Задание 6:** Выполнять анализ корректирующей способности используемого кода (количественная оценка) путем сравнения соответствующих слов *Xn* и *Yn’*; 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.

Для выполнения заданий был написан код на языке Typescript:

function iterativeMatrix(height: number, width: number): void {

    const generateMessage: number[][] = [];

    const gorParity: number[] = [];

    const verParity: number[] = [];

    const result: number[] = [];

    const sc: number[] = [];

    // Генерация сообщения

    for (let i = 0; i < height; i++) {

        generateMessage[i] = [];

        let eachRow = 0;

        for (let j = 0; j < width; j++) {

            generateMessage[i][j] = Math.floor(Math.random() \* 2);

            eachRow += generateMessage[i][j];

            result.push(generateMessage[i][j]);

            sc[j] = (sc[j] || 0) + generateMessage[i][j];

            process.stdout.write(`${generateMessage[i][j]}  `);

        }

        gorParity.push(eachRow % 2);

        process.stdout.write(` |${eachRow % 2}\n`);

    }

    // Вывод горизонтального паритета

    process.stdout.write("---".repeat(width) + "\n");

    for (const parity of sc) {

        process.stdout.write(`${parity % 2}  `);

    }

    process.stdout.write("\n\n");

    // Вывод информационного слова

    process.stdout.write("Xn=  ");

    for (const value of result) {

        process.stdout.write(`${value}`);

    }

    // Генерация ошибки

    try {

        const error = Math.floor(Math.random() \* result.length);

        process.stdout.write(`\nПозиция ошибки:  ${error}\n`);

        result[error] = result[error] === 1 ? 0 : 1;

    } catch (error) {}

    // Вывод исправленного сообщения

    process.stdout.write("Yn=  ");

    for (const value of result) {

        process.stdout.write(`${value}`);

    }

    // Вывод вертикального паритета

    process.stdout.write("\n\n");

    process.stdout.write("Сравниваем паритеты:\n");

    process.stdout.write("  ");

    for (const parity of gorParity) {

        process.stdout.write(`${parity}`);

    }

    process.stdout.write("  ");

    for (const parity of sc) {

        process.stdout.write(`${parity % 2}`);

    }

    process.stdout.write("\n");

}

function IterativeMatrix2(height: number, width: number): void {

    const a: number[][] = [];

    const sc: number[] = [];

    const result: number[] = [];

    // Генерация сообщения

    for (let i = 0; i < height; i++) {

        a[i] = [];

        let s = 0;

        for (let j = 0; j < width; j++) {

            a[i][j] = Math.floor(Math.random() \* 2);

            s += a[i][j];

            result.push(a[i][j]);

            sc[j] = (sc[j] || 0) + a[i][j];

            process.stdout.write(`${a[i][j]}  `);

        }

        process.stdout.write(` |${s % 2}\n`);

    }

    // Вывод горизонтального паритета

    process.stdout.write("---".repeat(width) + "\n");

    for (const parity of sc) {

        process.stdout.write(`${parity % 2}  `);

    }

    process.stdout.write("\n\n");

    // Вывод информационного слова

    process.stdout.write("Xn=  ");

    for (const value of result) {

        process.stdout.write(`${value}`);

    }

    // Генерация ошибки

    try {

        const error = Math.floor(Math.random() \* result.length);

        process.stdout.write(`\nПозиция ошибки:  ${error}\n`);

        result[error] = result[error] === 1 ? 0 : 1;

    } catch (error) {}

    // Вывод исправленного сообщения

    process.stdout.write("Yn=  ");

    for (const value of result) {

        process.stdout.write(`${value}`);

    }

}

function IterativeMatrixWithDepth(height: number, width: number, depth: number): void {

    const array: number[][][] = [];

    const result: number[] = [];

    const sc: number[] = [];

    // Генерация сообщения

    for (let i = 0; i < height; i++) {

        array[i] = [];

        for (let j = 0; j < width; j++) {

            array[i][j] = [];

            let slow = 0;

            for (let k = 0; k < depth; k++) {

                array[i][j][k] = Math.floor(Math.random() \* 2);

                slow += array[i][j][k];

                result.push(array[i][j][k]);

                sc[k] = (sc[k] || 0) + array[i][j][k];

                process.stdout.write(`${array[i][j][k]}  `);

            }

            process.stdout.write(` |${slow % 2}\n`);

        }

        process.stdout.write("---\n");

        for (const parity of sc) {

            process.stdout.write(`${parity % 2}  `);

        }

        process.stdout.write("\n\n");

    }

    // Вычисление и вывод паритетов 4 и 5 групп

    const parity4: number[] = [];

    const parity5: number[] = [];

    for (let i = 0; i < width; i++) {

        let sum4 = 0;

        let sum5 = 0;

        for (let j = 0; j < height; j++) {

            for (let k = 0; k < depth; k++) {

                if ((j === 3 && k === 1) || (j === 3 && k === 0) || (j === 2 && k === 1) || (j === 1 && k === 0)) {

                    sum5 += array[j][i][k];

                } else {

                    sum4 += array[j][i][k];

                }

            }

        }

        parity4.push(sum4 % 2);

        parity5.push(sum5 % 2);

    }

    // Вывод паритетов 4 и 5 групп

    process.stdout.write("Паритет 4 - 5 группы:\n");

    for (const p5 of parity5) {

        process.stdout.write(`${p5}`);

    }

    process.stdout.write("\n");

    for (const p4 of parity4) {

        process.stdout.write(`${p4}`);

    }

    process.stdout.write("\n");

    // Подсчет и вывод общего паритета

    const totalParity5 = parity5.join('');

    const totalParity4 = parity4.join('');

    process.stdout.write(`Паритет 5 группы:  ${totalParity5}\n`);

    process.stdout.write(`Паритет 4 группы:  ${totalParity4}\n`);

    // Вывод информационного слова

    process.stdout.write("Xn =  ");

    for (const value of result) {

        process.stdout.write(`${value}`);

    }

    // Генерация ошибки

    try {

        const error = Math.floor(Math.random() \* result.length);

        process.stdout.write(`\nПозиция ошибки:  ${error}\n`);

        result[error] = result[error] === 1 ? 0 : 1;

    } catch (error) {}

    // Вывод исправленного сообщения

    process.stdout.write("Yn =  ");

    for (const value of result) {

        process.stdout.write(`${value}`);

    }

}

function main(): void {

    iterativeMatrix(4, 4);

    console.log();

    console.log("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

    console.log();

    IterativeMatrix2(8, 2);

    console.log();

    console.log("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

    console.log();

    console.log();

    console.log();

    console.log("Размер 4 : 2 : 2");

    console.log();

    console.log();

    IterativeMatrixWithDepth(4, 2, 2);

    console.log();

    console.log("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

    console.log();

    IterativeMatrixWithDepth(2, 4, 2);

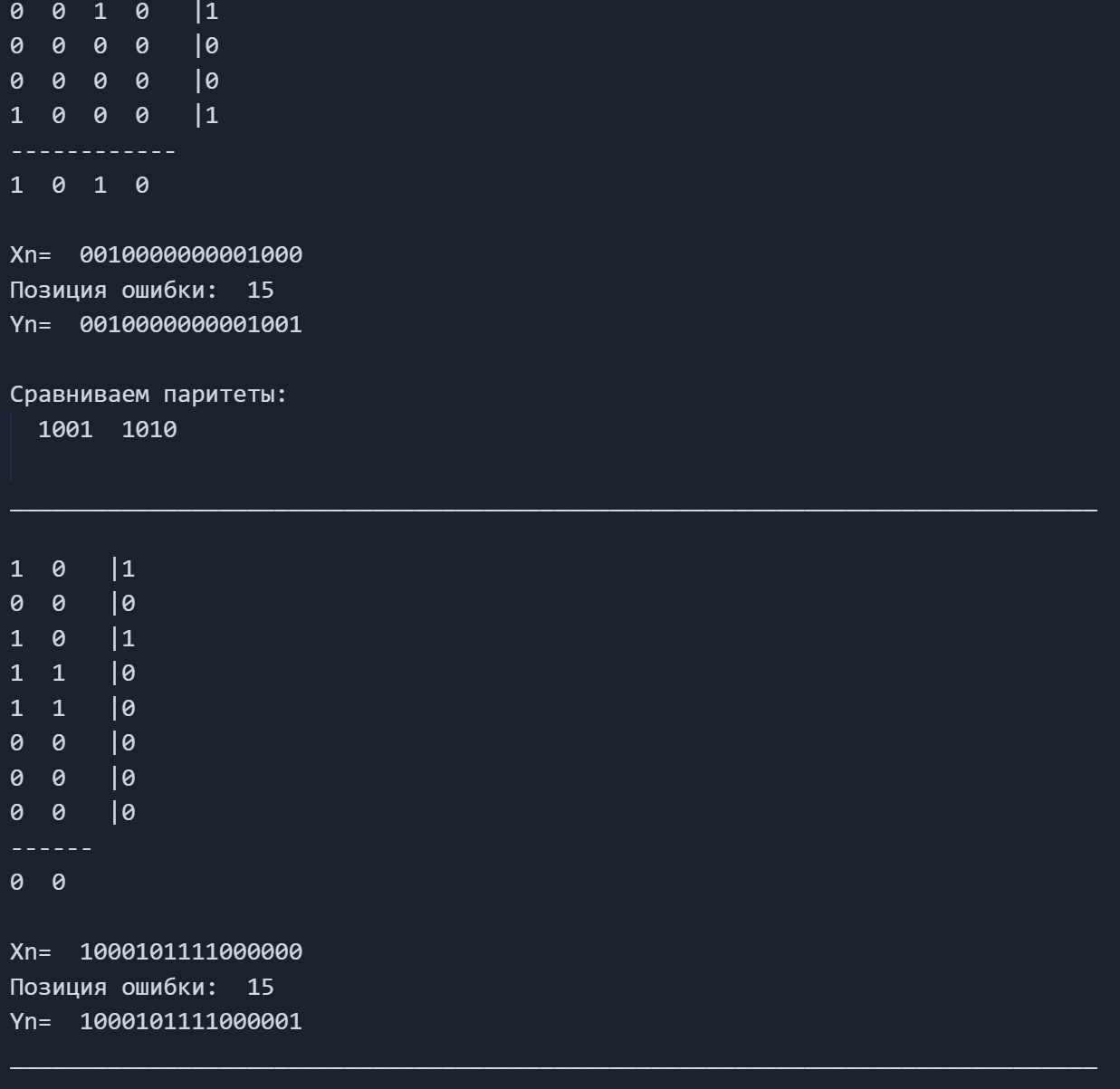
    console.log();

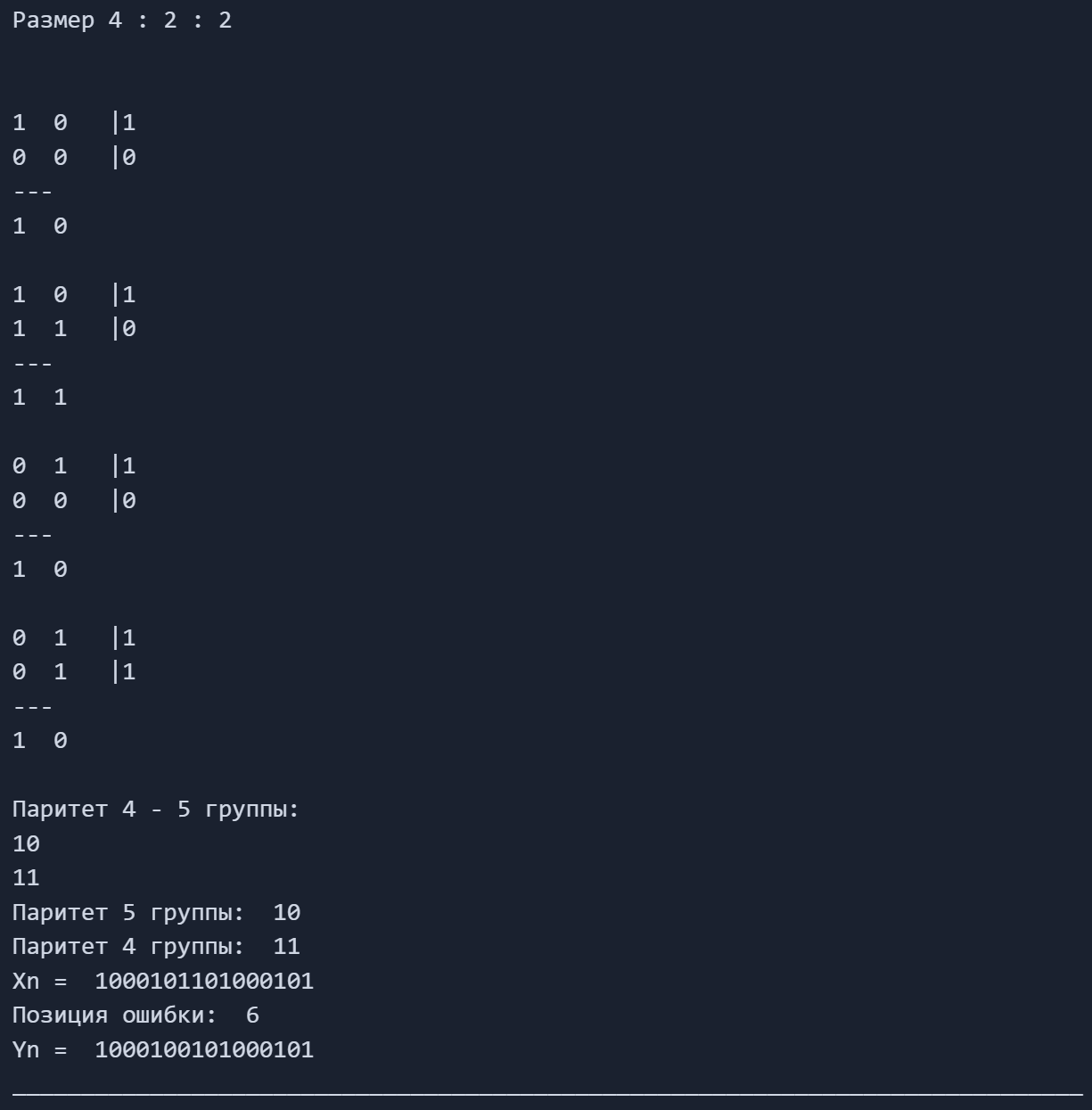
    console.log("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

main();

Результат работы итогового кода:





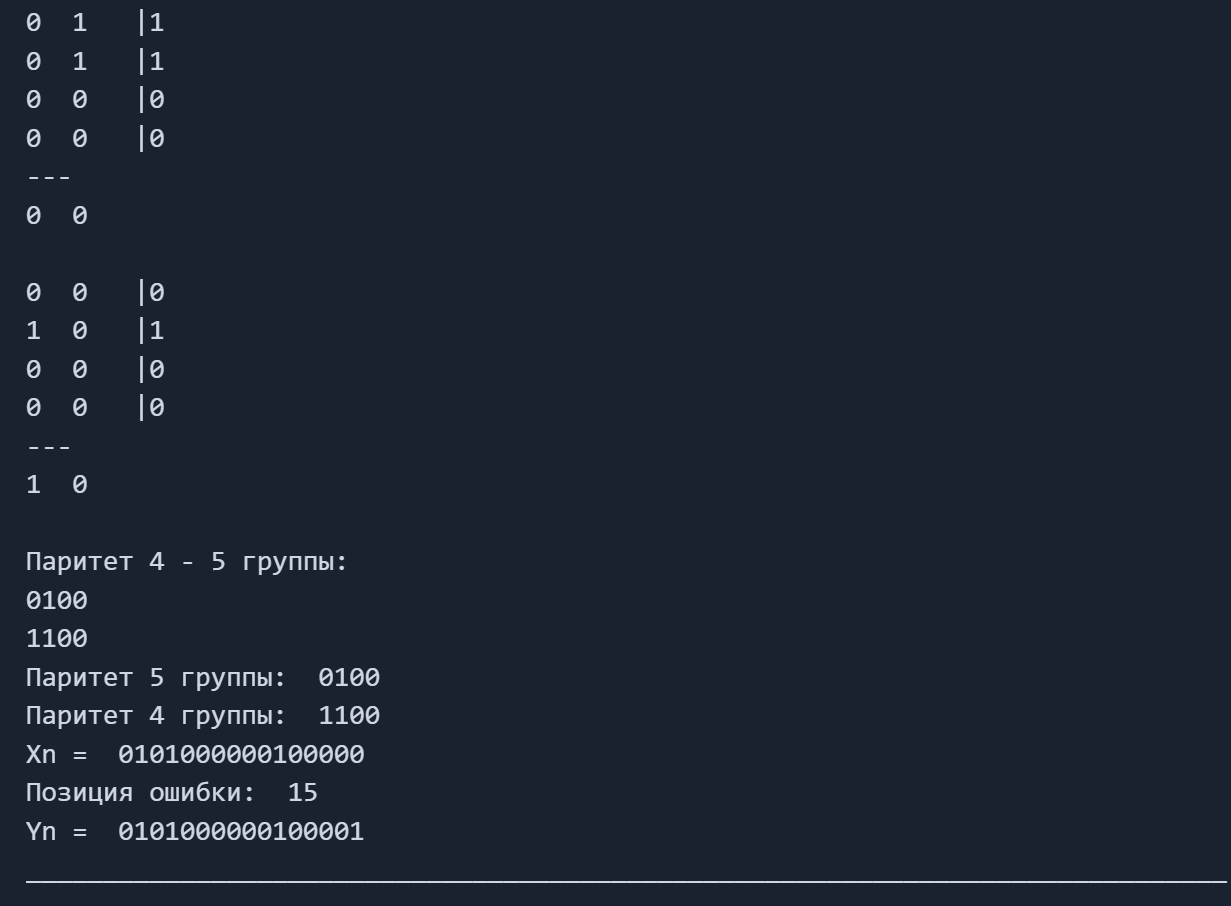


Рисунок 1 ­­­­– Результат работы программы

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