 **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технический университет**

**имени Н.Э. Баумана (национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления»**

**Кафедра «Системы обработки информации и управления»**

Домашнее задание №2

по дисциплине «Сети и телекоммуникации»

на тему:

«Разработка программных средств определения обнаруживающей и корректирующей способности кода в линейных протоколах»

Вариант №18

Выполнил:

студент группы ИУ5-53Б

Саргсян А.А.

Проверил:

к.т.н., доц., Галкин В.А.

2023 г.

**Постановка и метод решения задачи:**

Имеется дискретный канал связи, на вход которого подается закодированная в соответствии с вариантом задания кодовая последовательность. В канале возможны ошибки любой кратности. Вектор ошибки может принимать значения от единицы в младшем разряде до единицы во всех разрядах кодового вектора. Для каждого значения вектора ошибки на выходе канала после декодирования определяется факт наличия ошибки и предпринимается попытка ее исправления.

Требуется, используякодирование циклическим кодом Ц [7,4], определить обнаруживающую способность этого кода Со:

|  |  |  |  |
| --- | --- | --- | --- |
| № варианта | Информационный вектор | Код | Способность кода |
| 18 | 1000 | Ц [7,4] | Со |

Обозначения:

Ц [7,4] – Циклический код с образующим полиномом g(x) = x3 + x + 1;

n – число разрядов в закодированной записи: n = 7;

k – число информационных разрядов: k = 4;

r – степень образующего полинома: r = 3.

**Модель канала связи.**

Передатчик имитируется частью программного кода (переменной), содержащей исходное закодированное сообщение.

Трансфер по каналу связи имитируется частью программного кода (циклом), в котором на исходное закодированное сообщение накладываются по очереди вектора ошибок каждого класса.

Приемник имитируется частью программного кода (функцией), которая делит входящий циклический код на образующий полином, и если обнаруживает ненулевой синдром, то инвертирует разряды в полученном сообщении согласно вектору ошибки, соответствующей синдрому по таблице, вычисленной заранее.

**Алгоритмы кодирования, декодирования, вычисления обнаруживающей способности кода для ошибок всех возможных кратностей.**

1. **Алгоритм кодирования циклическим кодом:** 
   1. Осуществить сдвиг информационного вектора на 4 разряда влево, заполнив новые младшие разрядов нулями.
   2. Разделить полученный вектор на вектор порождающего полинома g(x), после получить остаток p(x).
   3. Порождающий полином для циклического кода [7,4] имеет вид g(x) = x3 + x + 1.
   4. Применив операцию конкатенации, объединить вектор остатка p(x) и исходный кодовый вектор для получения закодированной последовательности.
2. **Алгоритм декодирования циклическим кодом.**

После передачи кодового полинома V(x) по каналу связи принятый полином r(x) может содержать ошибки. При декодировании циклического кода можно с большой долей вероятности определить, была ли ошибка или нет. Декодирование выполняется следующим образом:

* 1. Разделить принятый полином r(x) на порождающий полином g(x) и проверить остаток от деления S(x) – он является синдромом ошибки.
  2. Если S(x) = 0, то ошибки нет или она не была обнаружена.
  3. Если S(x) ≠ 0, то ошибка есть. По виду вектора синдрома определить место ошибки и исправить ее.

Осуществить сдвиг в сторону младших разрядов исправленного вектора на z разрядов, чтобы получить первоначальный информационный вектор.

1. **Алгоритм вычисления обнаруживающей способности кода Со для ошибок всех возможных кратностей.**

Обнаруживающая способность кода Cо определяется как отношение числа обнаруженных ошибок No к общему числу ошибок данной кратности, которое определяется как число сочетаний из n (длина кодовой комбинации) по i (кратность ошибки – число единиц в векторе ошибок) - Cin.

Для подсчета обнаруживающей способности нужно перебрать все возможные вектора ошибок. Их число:

Для каждого вектора ошибки текущей кратности вычисляются соответственно принятый на выходе канала связи кодовый вектор и вектор синдрома ошибки, и по синдрому ошибки определяется факт обнаружения ошибки или его отсутствие.

**Реализация алгоритмов в виде приложения, написанного на языке JavaScript:**

*script.js:*

function submitForm() {

    let form = document.getElementById('Form');

    let n = parseInt(form.elements.n.value, 10);

    if (isNaN(n)) {

        alert("Неверное значение для N");

        return false

    }

    let k = parseInt(form.elements.k.value, 10);

    if (isNaN(k)) {

        alert("Неверное значение для K");

        return false

    }

    if (form.elements.m.value.length != k) {

        alert("Длина информационного вектора должна быть равна N")

        return false

    }

    let m = []

    for (let char of form.elements.m.value) {

        if (char !== '0' && char !== '1') {

            alert("Неверное значение для информационного вектора");

            return false

        }

        m.push(+char)

    }

    if (form.elements.g.value.length > (n - k + 1)) {

        alert("Длина порождающего вектора должна быть равна N")

        return false

    }

    let g = []

    for (let char of form.elements.g.value) {

        if (char !== '0' && char !== '1') {

            alert("Неверное значение для порождающего вектора");

            return false

        }

        g.push(+char)

    }

    console.log(n, k, m, g);

    let coded = code(m, g, n, k);

    let cyclic\_code = document.getElementById("cyclic\_code")

    cyclic\_code.textContent = "Циклический [" + n.toString() + ", " + k.toString() + "]-код: " + coded.join('')

    let table = document.getElementById('Table').getElementsByTagName('tbody')[0];

    for (let i = 1; i <= n; i++) {

        let errors = 0

        let detected = 0

        generateErrors(n, i, (e) => {

            errors += 1

            if (checkError(coded, e, g)) {

                detected += 1

            }

        })

        let newRow = table.insertRow();

        let cell1 = newRow.insertCell(0);

        cell1.textContent = i.toString()

        let cell2 = newRow.insertCell(1);

        cell2.textContent = errors.toString()

        let cell3 = newRow.insertCell(2);

        cell3.textContent = detected.toString()

        let cell4 = newRow.insertCell(3);

        cell4.textContent = (detected / errors).toString()

    }

    return false; // Returning false prevents the form from submitting

}

function code(m, g, n, k) {

    let tmp = m.slice().concat(Array(n - k).fill(0));

    let reminder = divide(tmp, g)

    return m.slice().concat(reminder)

}

function divide(a, b) {

    for (let i = 0; i <= (a.length - b.length); i++) {

        if (a[i] === 0) {

            continue

        }

        for (let j = 0; j < b.length; j++) {

            a[i + j] = a[i + j] ^ b[j]

        }

    }

    return a.slice(-(b.length - 1))

}

function checkError(v, e, g) {

    let r = Array.from(v)

    for (let i = e.length; i > 0; i--) {

        r[r.length - i] = r[r.length - i] ^ e[e.length - i]

    }

    let s = divide(r, g)

    for (let i of s) {

        if (i !== 0) {

            return true

        }

    }

    return false

}

function generateErrors(n, k, callback) {

    function backtrack(current, onesCount, remainingZeros) {

        if (onesCount + remainingZeros === 0) {

            callback(current);

            return;

        }

        if (onesCount > 0) {

            backtrack(current.slice().concat([1]), onesCount - 1, remainingZeros);

        }

        if (remainingZeros > 0) {

            backtrack(current.slice().concat([0]), onesCount, remainingZeros - 1);

        }

    }

    backtrack([], k, n - k);

    return;

}

*index.html:*

<!doctype html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>Домашнее задание №2</title>

    <script src="script.js"></script>

    <link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

    <form id="Form" onsubmit="return submitForm()">

        <div>

            <label for="n\_input">N:</label>

            <input type="text" id="n\_input" name="n" value="7">

            <!-- <input type="text" id="n\_input" name="n" required> -->

        </div>

        <div>

            <label for="k\_input">K:</label>

            <input type="text" id="k\_input" name="k" value="4">

        </div>

        <div>

            <label for="m">Информационный вектор:</label>

            <input type="text" id="m" name="m" value="1000">

        </div>

        <div>

            <label for="g">Порождающий вектор:</label>

            <input type="text" id="g" name="g" value="1011">

        </div>

        <button type="submit">Рассчитать</button>

    </form>

    <label id="cyclic\_code"></label>

    <table id="Table">

        <thead>

            <tr>

                <th>Кратность ошибки, i</th>

                <th>Общее число ошибок, Cⁱₙ</th>

                <th>Число обнаруженных ошибок, N₀</th>

                <th>Обнаруживающая способность кода, C₀</th>

            </tr>

        </thead>

        <tbody></tbody>

    </table>

</body>

*style.css:*

form {

    display: flex;

    flex-direction: column;

    max-width: 400px;

    margin: auto;

    padding: 20px;

    background-color: #f4f4f4;

    border-radius: 8px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

label {

    margin-bottom: 8px;

}

input {

    padding: 8px;

    margin-bottom: 16px;

}

button {

    padding: 10px;

    background-color: #914caf;

    color: white;

    border: none;

    border-radius: 4px;

    cursor: pointer;

}

button:hover {

    background-color: #bf2fca;

}

/\* Стили для результатов \*/

#cyclic\_code {

    display: block;

    margin-top: 20px;

    font-size: 18px;

}

/\* Стили для таблицы \*/

table {

    width: 100%;

    border-collapse: collapse;

    margin-top: 20px;

}

th, td {

    border: 1px solid #050505;

    padding: 12px;

    text-align: center;

}

th {

    background-color: #9a42ba;

    color: white;

}

/\* Общие стили для страницы \*/

body {

    font-family: Arial, sans-serif;

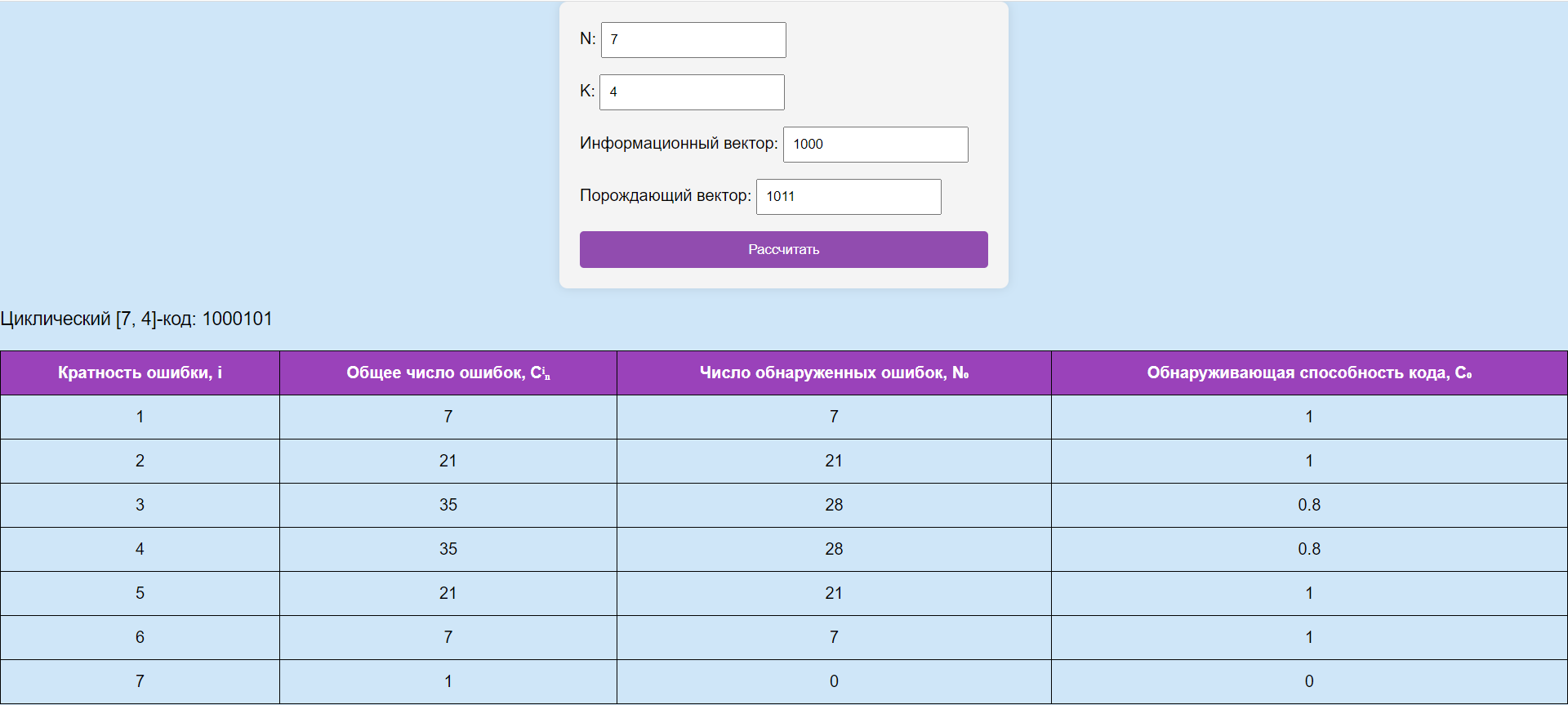
    background-color: #cfe6f8;

    margin: 0;

    padding: 0;

}

**Заполненная программно таблица результатов:**

****

**Таблица результатов:**

|  |  |  |  |
| --- | --- | --- | --- |
| i | Cⁱₙ | N₀ | C₀ |
| 1 | 7 | 7 | 1 |
| 2 | 21 | 21 | 1 |
| 3 | 35 | 28 | 0.8 |
| 4 | 35 | 28 | 0.8 |
| 5 | 21 | 21 | 1 |
| 6 | 7 | 7 | 1 |
| 7 | 1 | 0 | 0 |

# Выводы:

### По таблице результатов видно, что только при кратности ошибки равной 7 обнаруживающая способность циклического [7,4]-кода равна 0 и он не

способен обнаружить ошибку. При кратностях ошибки равными 1, 2, 5 и 6 обнаруживающая способность кода равна 1 и ошибку всегда можно обнаружить. При остальных кратностях ошибки обнаруживающая способность кода близка к 1 и ошибку почти всегда можно обнаружить. Таким образом, циклический [7,4]-код подходит для обнаружения ошибок кратностью до 6.

**Электронная копия отчёта и программы:**

<https://github.com/Blessed011/DZ_ST>

**Список используемой литературы**

1. Телекоммуникации и сети. / В.А. Галкин, Ю.А. Григорьев Учебное пособие для вузов.-М.:Из-во МГТУ им.Н.Э.Баумана 2003 г.

2. Методическое пособие по выполнению домашнего задания по дисциплине «Сети и телекоммуникации» / Галкин В.А. М.: МГТУ им. Н.Э. Баумана. 2018 г.

3. Конспект лекций по дисциплине “Сети и телекоммуникации”. М.: МГТУ им. Н.Э. Баумана. 2023 г. (рукопись)