МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №6

по дисциплине: Информатика
тема: «Обнаружение и исправление однократной ошибки в сообщении»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили: ст. пр.
Бондаренко Т. В.

Белгород 2022 г.

Цель работы: изучить основные принципы помехоустойчивого кодирования; изучить способ определение позиции и значения корректирующих бит кода Хемминга; получить практические навыки построения кода Хемминга, корректирующего однократные ошибки; изучить способ построения линейно-группового кода и возможность коррекции однократной ошибки с помощью линейно-группового кода.

Задания к работе:

1. Выполнить кодирование текстового сообщения M_1 по буквам, используя русский или латинский алфавит, размер сообщения не менее 4 букв. Определить размер n в битах закодированного сообщения M. Например, в качестве кода можно использовать порядковый номер буквы в алфавите. Если M_1 = “АБ”, то M = 000001000010 и размер сообщения n = 12.
2. Определить количество k контрольных разрядов кода Хемминга, необходимых для кодирования сообщения M размер n бит.
3. Определить позиции и значения k контрольных разрядов кода Хемминга двумя способами:
   - подсчёт количества единиц в контролируемых контрольным битом разрядах сообщения;
   - использование двоичного представления номеров разрядов сообщения.
4. Записать полученное сообщение размера (n +k) в коде Хемминга.
5. Смоделировать коррекцию ошибки: внести однократную, двукратную и k-кратную ошибки в произвольные биты сообщения и найти эти ошибки с помощью кода Хемминга, используя:
   - подсчёт количества единиц в контролируемых контрольным битом разрядах сообщения;
   - двоичное представление номеров разрядов сообщения.

Дополнительное задание

Составить программу, выполняющую построение кода Хемминга для произвольного сообщения, состоящего из символов русского и английского алфавита. (Сообщение необходимо закодировать). Смоделировать процесс передачи сообщения, реализовав в программе случайное возникновение однократной, двукратной и k-кратной ошибки в случайно выбранных битах сообщения.

Реализовать в программе проверку сообщения в коде Хемминга на наличие однократной ошибки и поиск позиции бита с ошибкой. Реализовать исправление ошибки и вывод откорректированного сообщения для пользователя.

Задание 1 (M_1 = Цинк**)**

M_1=11000 01010 01111 01100

n = 20

Задание 2

$2^{N-M} - 1 > N$, где M = n = 20

$2^{N-M} - 1 \geq N$

N = 24

$2^4 - 1 \geq 24$, не подходит

N = 25

$2^5 - 1 \geq 25$, подходит

K = N – M = 5, следовательно 5 контрольных разрядов.

Задание 3

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	Чис. ед.	Бит чётности
	1	0	1	1	1	0	0	0	0	0	1	0	1	0	0	0	1	1	1	1	0	1	1	0	0		
K1																										7	1
K2																										6	0
K3																										5	1
K4																										2	0
K5																										6	0

	3	0	0	0	1	1
	5	0	0	1	0	1
	11	0	1	0	1	1
	13	0	1	1	0	1
^	17	1	0	0	0	1
	18	1	0	0	1	0
	19	1	0	0	1	1
	20	1	0	1	0	0
	22	1	0	1	1	0
	23	1	0	1	1	1
		0	0	1	0	1
		K5	K4	K3	K2	K1

Задание 4

10111000001010001111101100 – закодированная последовательность.

Задание 5

10111000001010001011101100

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	Чис. ед.	Бит чётности
	1	0	1	1	1	0	0	0	0	0	1	0	1	0	0	0	1	0	1	1	0	1	1	0	0		
К1																										7	1
К2																										5	1
К3																										5	1
К4																										2	0
К5																										5	1

2+16=18

К1	К2	К3	К4	К5	
1	0	1	1	1	новый
0	0	1	0	1	старый
1	0	0	1	0	= 18

Ошибка допущена в 18 бите, вычисления верны.

10111001001010001011101100

1	0	0	0	0	1
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
8	0	1	0	0	0
11	0	1	0	1	1
13	0	1	1	0	1
17	1	0	0	0	1
19	1	0	0	1	1
20	1	0	1	0	0
22	1	0	1	1	0
23	1	0	1	1	1
	1	1	0	1	0 = 26

Ошибка обнаружена, однако определить её настоящее место невозможно, так как ошибка получена на 26 бите, однако ошибки были внесены в 8 и 18 бит.

11101101001000101011000001

1	0	0	0	0	1

2	0	0	0	1	0
3	0	0	0	1	1
5	0	0	1	0	1
6	0	0	1	1	0
8	0	1	0	0	0
11	0	1	0	1	1
15	0	1	1	1	1
17	1	0	0	0	1
19	1	0	0	1	1
20	1	0	1	0	0
25	1	1	0	0	1
	0	0	0	0	0 = 0

Полученный код не содержит ошибок, однако в оригинальном сообщении было допущено 10 ошибок. Определить ошибку в коде с > 2 ошибками невозможно.

Дополнительное задание

Входные данные	Ожидаемый результат
«Цинк», нет ошибок	«Цинк»
«Цинк», 1 ошибка	«Цинк»
«Цинк», > 1 ошибки	Результат непредсказуем

```
<html lang="ru">
<head>
    <script defer>
        const SenderFieldFormElements = {
            TEXT_MESSAGE: {id: 'textmessage', priority: 2},
            BIN_MESSAGE: {id: 'binmessage', priority: 1},
            HAMMING_MESSAGE: {id: 'hammingmessage', priority: 0},
            ERROR_TYPE: {
                id: "error-type",
                NO_ERROR: "no_error",
                ERROR: "error",
                CUSTOM_ERROR: "custom"
            },
            ERROR_COUNT: {id: 'errorcount'}
        };

        function convertUTF16ToBin(messageData) {
            let modMessage = "";
            for (let val of messageData) {
                var a = val.charCodeAt(0).toString(2);
                a = new Array(17 - a.length).join('0') + a;
                modMessage += a;
            }
            return modMessage;
        }

        function convertBinToUTF16(messageData) {
            let modMessage = "";
            while (messageData !== "") {
                let charCode =
Number.parseInt(messageData.substring(messageData.length - 16), 2);
                modMessage = String.fromCharCode(charCode) + modMessage;
                messageData = messageData.substring(0, messageData.length -
16);
            }
            return modMessage;
        }

        function makeErrors(messageData, errorCount) {
            let errorsMade = Object.entries(messageData);
            let modMessage = messageData;
            while (errorCount > 0) {
                let ind = Math.floor(Math.random() * errorsMade.length);
                let ch = errorsMade[ind][1];
                modMessage = modMessage.split('');
                modMessage[errorsMade[ind][0]] = ch === "1" ? "0" : "1";
                modMessage = modMessage.join('');
                errorsMade.splice(ind, 1);
                errorCount--;
            }
            return modMessage;
        }

        function convertBinToHamming(messageData) {
            let modMessage = "";
            let ind = 1;
```

```
            let nextInd = 1;
            let bits = 0;
            while (messageData !== "") {
                if (ind === nextInd) {
                    modMessage += "0";
                    nextInd *= 2;
                } else {
                    let ch = messageData[0];
                    if (ch == '1' || ch == '0') {
                        modMessage += ch;
                        messageData = messageData.substring(1);
                        if (ch === '1') {
                            bits ^= ind;
                        }
                    } else {
                        throw Error("Ошибка! Вы можете конвертировать в код
Хэмминга только числа в бинарном коде");
                    }
                }
                ind++;
            }
            nextInd = 1;
            while (bits !== 0) {
                modMessage = modMessage.split('');
                modMessage[nextInd - 1] = String(bits % 2);
                modMessage = modMessage.join('');
                bits >>= 1;
                nextInd *= 2;
            }

            return modMessage;
        }

        function getBrokenBit(messageData) {
            let ind = 1;
            let nextInd = 1;
            let oldBits = 0;
            let newBits = 0;
            let bitNum = 0;
            while (ind <= messageData.length) {
                if (ind === nextInd) {
                    let bit = Number.parseInt(messageData.at(ind - 1));
                    oldBits += bit << (bitNum++);
                    nextInd *= 2;
                } else {
                    let ch = messageData[ind - 1];
                    if (ch == '1' || ch == '0') {
                        if (ch === '1') {
                            newBits ^= ind;
                        }
                    } else {
                        throw Error("Ошибка! Вы можете конвертировать в код
Хэмминга только числа в бинарном коде");
                    }
                }
                ind++;
```

```
            }

            return oldBits ^ newBits;
        }

        function decodeHamming(messageData) {
            let ind = 1;
            let nextInd = 1;
            let modMessage = "";
            while (ind <= messageData.length) {
                if (ind === nextInd) {
                    nextInd *= 2;
                } else {
                    modMessage += messageData[ind - 1];
                }
                ind++;
            }

            return modMessage;
        }

        document.addEventListener('DOMContentLoaded', function () {

            function textToBin() {
                try {
                    let messageData = document.querySelector("form
#textmessagebox textarea").value;
                    messageData = convertUTF16ToBin(messageData);
                    document.querySelector("form #binmessagebox
textarea").value = messageData;
                } catch (error) {
                    document.querySelector("form .error-
occured").style.display = 'block';
                }
            }

            function binToHam() {
                try {
                    let messageData = document.querySelector("form
#binmessagebox textarea").value;
                    messageData = convertBinToHamming(messageData);
                    document.querySelector("form #hammingmessagebox
textarea").value = messageData;
                } catch (error) {
                    document.querySelector("form .error-
occured").style.display = 'block';
                }
            }

            document.querySelector("form #error-typebox
select").addEventListener('change', (ev) => {
                if (document.querySelector("form #error-typebox
select").value === SenderFieldFormElements.ERROR_TYPE.CUSTOM_ERROR) {
                    document.querySelector("#errorcountbox").style.display =
'block';
                } else {
```

```
                    document.querySelector("#errorcountbox").style.display =
'none';
                }
            });

            document.querySelector("form #textmessagebox
button").addEventListener('click', (ev) => {
                ev.preventDefault();
                textToBin(ev);
                binToHam(ev);
            });

            document.querySelector("form #binmessagebox
button").addEventListener('click', (ev) => {
                ev.preventDefault();
                binToHam(ev);
            });

            document.querySelector("form #textmessagebox
textarea").addEventListener('input', (ev) => {
                ev.preventDefault();
                textToBin(ev);
                binToHam(ev);
            });

            document.querySelector("form #binmessagebox
textarea").addEventListener('input', (ev) => {
                ev.preventDefault();
                binToHam(ev);
            });

            document.querySelector("form").addEventListener("submit", (ev) =>
{
                ev.preventDefault();

                document.querySelector("form .error-occured").style.display =
'none';

                const eventData = new FormData(ev.target);
                let messageData, messagePriority = Infinity, errorCount = 0,
preventCustomError = false;
                for (let val of [...eventData.entries()]) {
                    switch (val[0]) {
                        case SenderFieldFormElements.TEXT_MESSAGE.id:
                            if (SenderFieldFormElements.TEXT_MESSAGE.priority
< messagePriority && val[1] !== "") {
                                messageData = val[1];
                                messagePriority =
SenderFieldFormElements.TEXT_MESSAGE.priority;
                            }
                            break;
                        case SenderFieldFormElements.BIN_MESSAGE.id:
                            if (SenderFieldFormElements.BIN_MESSAGE.priority
< messagePriority && val[1] !== "") {
                                messageData = val[1];
                                messagePriority =
```

```
SenderFieldFormElements.BIN_MESSAGE.priority;
                            }
                            break;
                        case SenderFieldFormElements.HAMMING_MESSAGE.id:
                            if
(SenderFieldFormElements.HAMMING_MESSAGE.priority < messagePriority && val[1]
!== "") {
                                messageData = val[1];
                                messagePriority =
SenderFieldFormElements.HAMMING_MESSAGE.priority;
                            }
                            break;
                        case SenderFieldFormElements.ERROR_TYPE.id:
                            switch (val[1]) {
                                case
SenderFieldFormElements.ERROR_TYPE.CUSTOM_ERROR:
                                    preventCustomError = false;
                                    errorCount = 0;
                                    break;
                                case
SenderFieldFormElements.ERROR_TYPE.NO_ERROR:
                                    preventCustomError = true;
                                    errorCount = 0;
                                    break;
                                case
SenderFieldFormElements.ERROR_TYPE.ERROR:
                                    preventCustomError = true;
                                    errorCount = 1;
                                    break;
                            }
                            break;
                        case SenderFieldFormElements.ERROR_COUNT.id:
                            if (!preventCustomError) {
                                errorCount = Number.parseInt(val[1]);
                            }
                            break;
                        default:
                            break;
                    }
                }

                if (messagePriority ===
SenderFieldFormElements.TEXT_MESSAGE.priority) {
                    try {
                        messageData = convertUTF16ToBin(messageData);
                        messagePriority =
SenderFieldFormElements.BIN_MESSAGE.priority;
                        document.querySelector("form #binmessagebox
textarea").value = messageData;
                    } catch (error) {
                        document.querySelector("form .error-
occured").style.display = 'block';
                    }
                }

                if (messagePriority ===
```

```
SenderFieldFormElements.BIN_MESSAGE.priority) {
                    try {
                        messageData = convertBinToHamming(messageData);
                        document.querySelector("form #hammingmessagebox
textarea").value = messageData;
                    } catch (error) {
                        document.querySelector("form .error-
occured").style.display = 'block';
                    }
                }

                errorCount = Math.min(errorCount, messageData.length);
                messageData = makeErrors(messageData, errorCount);

                sendData(messageData);
            });
        });

        function sendData(messageData) {
            document.querySelector("#receiver").style.display = 'block';
            document.querySelector("#receiver .received-message").innerHTML =
messageData
            try {
                document.querySelector("#receiver .decode-message .success-
message").style.display = 'none';
                document.querySelector("#receiver .decode-message .fail-
message").style.display = 'none';
                document.querySelector("#receiver .decode-message .total-
fail-message").style.display = 'none';
                let brokenBit = getBrokenBit(messageData);
                if (brokenBit === 0) {
                    document.querySelector("#receiver .decode-message
.success-message").style.display = 'block';
                    messageData = decodeHamming(messageData);
                    document.querySelector("#receiver .decode-message
.success-message .corrected-message")
                        .innerHTML = messageData;
                } else if (brokenBit <= messageData.length) {
                    document.querySelector("#receiver .decode-message .fail-
message").style.display = 'block';
                    document.querySelector("#receiver .decode-message .fail-
message .fail-bit").innerHTML = brokenBit;
                    messageData = messageData.split('');
                    messageData[brokenBit - 1] = messageData[brokenBit - 1]
== "1" ? "0" : "1";
                    messageData = messageData.join('');
                    messageData = decodeHamming(messageData);
                    document.querySelector("#receiver .decode-message .fail-
message .corrected-message")
                        .innerHTML = messageData;
                } else throw new Error("Невозможно декодировать");

                messageData = convertBinToUTF16(messageData);
                document.querySelector("#receiver .final-message").innerHTML
= messageData;
            } catch (e) {
```

```
                document.querySelector("#receiver .decode-message .total-
fail-message").style.display = 'block';
            }
        }
    </script>
    <title>Лаба по инфе №6, доп. задание</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
<div>
    <form style="display: block;" id="sender">
        Отправитель:
        <p id="textmessagebox">
            Сообщение (в виде текста)
            <textarea name="textmessage"></textarea>
            <button type="button">Закодировать сообщение</button>
        </p>
        <p id="binmessagebox">
            Сообщение (в бинарном виде)
            <textarea name="binmessage"></textarea>
            <button>Преобразовать в код Хэмминга</button>
        </p>
        <p id="hammingmessagebox">
            Сообщение (в коде Хэмминга)
            <textarea name="hammingmessage"></textarea>
        </p>
        <p id="error-typebox">
            Качество сигнала
            <select name="error-type">
                <option value="no_error" selected>
                    Без помех
                </option>
                <option value="error">
                    Слабые помехи (1 ошибка)
                </option>
                <option value="custom">
                    Ручка регуляции силы сигнала (выберите количество ошибок)
                </option>
            </select>
        <div id="errorcountbox" style="display: none">
            Количество ошибок:
            <input name="errorcount">
        </div>
        </p>
        <button>Отправить данные</button>
        <p class="error-occured" style="display: none; color: red;">Ошибка
при вычислении!</p>
    </form>
    <hr>
    <div style="display: none;" id="receiver">
        Получатель:
        <div>
            <p>
                Полученное сообщение:
            </p>
            <textarea readonly class="received-message" style="color: black;
```

```
resize: none; width: 100%; height: 150px">
            </textarea>
        </div>
        <div>
            <p>
                Расшифровка сообщения:
            </p>
            <div class="decode-message">
                <div class="success-message" style="color: darkgreen;
display: none">
                    Сообщение передано без ошибок!
                    <textarea readonly class="corrected-message"
style="color: black; resize: none; width: 100%; height: 150px">

                    </textarea>
                </div>
                <div class="fail-message" style="display: none">
                    Обнаружена ошибка в <span class="fail-bit"></span> бите.
Исправлено сообщение:
                    <textarea readonly class="corrected-message"
style="color: black; resize: none; width: 100%; height: 150px">

                    </textarea>
                </div>
                <p class="total-fail-message" style="color: red; display:
none">
                    Сообщение невозможно расшифровать.
                </p>
            </div>
        </div>
        <div>
            <p>
                Текстовое сообщение:
            </p>
            <textarea readonly class="final-message" style="color: black;
resize: none; width: 100%; height: 150px">

            </textarea>
        </div>
    </div>
</div>
</body>
</html>
```

Вывод: в ходе лабораторной работы изучил основные принципы помехоустойчивого кодирования; изучил способ определение позиции и значения корректирующих бит кода Хемминга; получил практические навыки построения кода Хемминга, корректирующего однократные ошибки;