Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный технический университет”
Кафедра интеллектуально-информационных технологий

# Лабораторная работа №1

По дисциплине «Современные методы защиты компьютерных систем»

Выполнил:
студент 4 курса
группы ИИ-22
Копанчук Е. Р.
Проверил:
Хацкевич А. С.

Брест-2024

## Ход работы:

**Задание:** закрепить теоретические знания по использованию методов помехоустойчивого кодирования для повышения надежности передачи и хранения в памяти компьютера двоичных данных. Разработать приложение для кодирования/декодирования двоичной информации кодом Хемминга с минимальным кодовым расстоянием 3 или 4.

## Код программы:

```
class HammingCode {
  private r: number;
  private poses: number[];

  constructor() {
    this.r = 5;
    this.poses = [0, 1, 3, 7, 15];
  }

  private toBinaryArray(message: string): number[] {
    const binArr = [];
    for (let char of message) {
      const bin = char.charCodeAt(0).toString(2).padStart(8, '0');
      for (let bit of bin) {
        binArr.push(parseInt(bit));
      }
    }
    return binArr;
  }

  encode(message: string): number[] {
    let data = this.toBinaryArray(message);
    const encoded = [];
    let dataPos = 0;
    for (let i = 0; i < data.length + this.r; i++) {
      if (this.poses.includes(i)) {
        encoded.push(0);
      } else {
        encoded.push(data[dataPos]);
        dataPos++;
      }
    }
    for (let pos of this.poses) {
      let parity = 0;
      for (let i = 0; i < encoded.length; i++) {
        if (i !== pos && (i + 1) & (pos + 1)) {
          parity ^= encoded[i];
        }
      }
      encoded[pos] = parity;
    }
    return encoded;
  }

  validate(encoded: number[]): { corrected: number[], errorPos: number | null } {
    let errorPos = 0;
    for (let pos of this.poses) {
      let parity = 0;
      for (let i = 0; i < encoded.length; i++) {
        if ((i + 1) & (pos + 1)) {
```

```
          parity ^= encoded[i];
        }
      }
      if (parity !== 0) {
        errorPos += (pos + 1);
      }
    }
    if (errorPos !== 0) {
      encoded[errorPos - 1] ^= 1;
    }
    return {
      corrected: encoded,
      errorPos: errorPos === 0 ? null : errorPos - 1
    };
  }

  decode(encoded: number[]): string {
    const decoded = [];
    for (let i = 0; i < encoded.length; i++) {
      if (!this.poses.includes(i)) {
        decoded.push(encoded[i]);
      }
    }
    const chars = [];
    for (let i = 0; i < decoded.length; i += 8) {
      const byte = decoded.slice(i, i + 8).join('');
      chars.push(String.fromCharCode(parseInt(byte, 2)));
    }
    return chars.join('');
  }
}
```

## Пример:

```
const hamming = new HammingCode();
const message = 'hello world';

let encoded = hamming.encode(message);
console.log('Original:', encoded.join(""));

let { corrected, errorPos } = hamming.validate(encoded);
console.log('Corrected:', corrected.join(""));
console.log('errPos: ', errorPos !== null ? errorPos : 'No error');

let decoded = hamming.decode(corrected);
console.log('Decoded:', decoded);

if (decoded === message) {
  console.log('Passed');
} else {
  console.log('Failed');
}

encoded[10] ^= 1;
console.log('Encoded message with error:', encoded.join(""));

({ corrected, errorPos } = hamming.validate(encoded));
console.log('Corrected (error):', corrected.join(""));
console.log('errPos: ', errorPos !== null ? errorPos : 'No error');

decoded = hamming.decode(corrected);
```

```
console.log('Decoded (correction):', decoded);

if (decoded === message) {
  console.log('Passed');
} else {
  console.log('Failed');
}
```

```
PS C:\Users\kopan\OneDrive\Desktop\CM3KC\lab1> ts-node index.ts
Original: 1000110010000111001010110110001101100011011111001000000111011101110111011100100110110001100100
Corrected: 1000110010000111001010110110001101100011011111001000000111011101110111011100100110110001100100
errPos:  No error
Decoded: hello world
Passed
Encoded message with error: 1000110010100111001010110110001101100011011111001000000111011101110111011100100110110001100100
Corrected (error): 1000110010000111001010110110001101100011011111001000000111011101110111011100100110110001100100
errPos:  10
Decoded (correction): hello world
Passed
```