

using System;

using System.Collections;

using System.Text;

using System.Windows.Forms;

namespace Hamming

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

public int ERROR\_VAL = -1;

public int EMERGENCY = -1;

#region HAMMING\_CODE

// Кодирование Хэмминга с выводом дополнительных данных

public BitArray Code(string inMessage)

{

if (!IsValidBinaryString(inMessage))

throw new ArgumentException("Входное сообщение должно содержать только 0 и 1.");

outputTextBox.AppendText($"Исходные данные: {inMessage}\n");

var messageArray = new BitArray(inMessage.Length, false);

for (int i = 0; i < inMessage.Length; i++)

messageArray[i] = inMessage[i] == '1';

int messageInd = 0;

int retInd = 0;

int controlIndex = 1;

var retArray = new BitArray(messageArray.Length + 1 + (int)Math.Ceiling(Math.Log(messageArray.Length, 2)));

outputTextBox.AppendText("Создаем слово данных, оставляя места для битов четности: ");

for (int i = 0; i < retArray.Length; i++)

{

if (i + 1 == controlIndex)

{

outputTextBox.AppendText("\_ ");

controlIndex \*= 2;

}

else

{

outputTextBox.AppendText(messageArray[messageInd] ? "1 " : "0 ");

messageInd++;

}

}

outputTextBox.AppendText("\n");

messageInd = 0;

retInd = 0;

controlIndex = 1;

while (messageInd < messageArray.Length)

{

if (retInd + 1 == controlIndex)

{

retInd++;

controlIndex \*= 2;

continue;

}

retArray.Set(retInd, messageArray.Get(messageInd));

messageInd++;

retInd++;

}

retInd = 0;

controlIndex = 1 << (int)Math.Log(retArray.Length, 2);

outputTextBox.AppendText("Рассчитываем четность для каждого бита четности:\n");

while (controlIndex > 0)

{

int c = controlIndex - 1;

int counter = 0;

outputTextBox.AppendText($"Позиция {controlIndex} проверяет биты: ");

while (c < retArray.Length)

{

for (int i = 0; i < controlIndex && c < retArray.Length; i++)

{

outputTextBox.AppendText($"{c + 1} ");

if (retArray.Get(c))

counter++;

c++;

}

c += controlIndex;

}

bool parity = counter % 2 != 0;

outputTextBox.AppendText($". Четность: {(parity ? "нечётное" : "чётное")}.\n");

retArray.Set(controlIndex - 1, parity);

outputTextBox.AppendText($"Устанавливаем бит четности {controlIndex} в {(parity ? "1" : "0")}.\n");

controlIndex /= 2;

}

outputTextBox.AppendText("Закодированное слово: ");

for (int i = 0; i < retArray.Length; i++)

outputTextBox.AppendText(retArray[i] ? "1" : "0");

outputTextBox.AppendText("\n");

return retArray;

}

// Декодирование Хэмминга с выводом дополнительных данных

public BitArray Decode(string inMessage)

{

if (!IsValidBinaryString(inMessage))

throw new ArgumentException("Входное сообщение должно содержать только 0 и 1.");

outputTextBox.AppendText($"Получено кодовое слово: {inMessage}\n");

var codedArray = new BitArray(inMessage.Length, false);

for (int i = 0; i < codedArray.Length; i++)

codedArray[i] = inMessage[i] == '1';

outputTextBox.AppendText("Извлекаем информационные биты, пропуская контрольные биты:\n");

var decodedArray = new BitArray((int)(codedArray.Count - Math.Ceiling(Math.Log(codedArray.Count, 2))), false);

int count = 0;

for (int i = 0; i < codedArray.Length; i++)

{

bool isControlBit = false;

for (int j = 0; j < Math.Ceiling(Math.Log(codedArray.Count, 2)); j++)

{

if (i == Math.Pow(2, j) - 1)

{

isControlBit = true;

break;

}

}

if (!isControlBit)

{

decodedArray[count++] = codedArray[i];

outputTextBox.AppendText(codedArray[i] ? "1 " : "0 ");

}

else

{

outputTextBox.AppendText($"\_ ");

}

}

outputTextBox.AppendText("\n");

outputTextBox.AppendText("Сравниваем контрольные биты для поиска ошибок:\n");

var checkArray = Code(BitArrayToString(decodedArray)); // Рекодируем, чтобы получить контрольные биты

byte[] failBits = new byte[checkArray.Length - decodedArray.Length];

count = 0;

bool isMistake = false;

for (int i = 0; i < checkArray.Length - decodedArray.Length; i++)

{

int controlBitPos = (int)Math.Pow(2, i) - 1;

if (codedArray[controlBitPos] != checkArray[controlBitPos])

{

outputTextBox.AppendText($"Контрольный бит на позиции {controlBitPos + 1} не совпадает.\n");

failBits[count++] = (byte)(Math.Pow(2, i));

isMistake = true;

}

else

{

outputTextBox.AppendText($"Контрольный бит на позиции {controlBitPos + 1} совпадает.\n");

}

}

if (isMistake)

{

int mistakeIndex = 0;

for (int i = 0; i < failBits.Length; i++)

mistakeIndex += failBits[i];

mistakeIndex--;

outputTextBox.AppendText($"Ошибка найдена в бите {mistakeIndex + 1}.\n");

if (mistakeIndex >= 0 && mistakeIndex < codedArray.Length)

{

codedArray.Set(mistakeIndex, !codedArray[mistakeIndex]);

outputTextBox.AppendText($"Исправляем бит на позиции {mistakeIndex + 1}.\n");

ERROR\_VAL = mistakeIndex;

}

else

{

throw new InvalidOperationException("Ошибка за пределами диапазона закодированного сообщения.");

}

outputTextBox.AppendText("Исправленное кодовое слово: ");

for (int i = 0; i < codedArray.Length; i++)

outputTextBox.AppendText(codedArray[i] ? "1" : "0");

outputTextBox.AppendText("\n");

}

else

{

outputTextBox.AppendText("Ошибок не обнаружено в кодовом слове.\n");

}

outputTextBox.AppendText("Извлекаем окончательное сообщение (информационные биты):\n");

count = 0;

for (int i = 0; i < codedArray.Length; i++)

{

bool isControlBit = false;

for (int j = 0; j < Math.Ceiling(Math.Log(codedArray.Count, 2)); j++)

{

if (i == Math.Pow(2, j) - 1)

{

isControlBit = true;

break;

}

}

if (!isControlBit)

{

decodedArray[count++] = codedArray[i];

outputTextBox.AppendText(codedArray[i] ? "1 " : "0 ");

}

}

outputTextBox.AppendText("\n");

return decodedArray;

}

// Метод для преобразования BitArray в строку

private string BitArrayToString(BitArray bits)

{

StringBuilder sb = new StringBuilder(bits.Length);

foreach (bool bit in bits)

{

sb.Append(bit ? '1' : '0');

}

return sb.ToString();

}

#endregion

private void button1\_Click(object sender, EventArgs e)

{

try

{

outputTextBox.Clear();

BitArray code = Decode(inputTextBox.Text);

if (ERROR\_VAL == -1)

{

MessageBox.Show("В коде ошибки нет.");

return;

}

//outputTextBox.AppendText($"Ошибка в {ERROR\_VAL + 1} бите\nИсправленный вариант:\n");

char[] textArray = inputTextBox.Text.ToCharArray();

textArray[ERROR\_VAL] = textArray[ERROR\_VAL] == '0' ? '1' : '0';

//outputTextBox.AppendText(new string(textArray));

}

catch (ArgumentException ex)

{

MessageBox.Show($"Ошибка ввода: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Непредвиденная ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button2\_Click(object sender, EventArgs e)

{

try

{

outputTextBox.Clear();

if (string.IsNullOrWhiteSpace(inputTextBox.Text))

{

MessageBox.Show("Введите сообщение для кодирования.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

outputTextBox.AppendText("Закодированная строка:\n");

BitArray code = Code(inputTextBox.Text);

//for (int i = 0; i < code.Length; i++)

//outputTextBox.AppendText(code[i] ? "1" : "0");

}

catch (ArgumentException ex)

{

MessageBox.Show($"Ошибка ввода: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Непредвиденная ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private bool IsValidBinaryString(string input)

{

foreach (char c in input)

if (c != '0' && c != '1')

return false;

return true;

}

private void outputTextBox\_TextChanged(object sender, EventArgs e)

{

}

}

}