Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Направление 09.03.01 – «Информатика и вычислительная техника»

Дисциплина: «Защита информации»

Профиль: «Автоматизированные системы обработки информации и управления»

Семестр 7

ОТЧЕТ

по лабораторной работе №5

Тема: «Шифры обнаружения и коррекции ошибок»

Выполнил: студент группы АСУ-17-1б

Хохряков Денис

Проверил: доцент кафедры ИТАС

Шереметьев В.Г.\_\_\_\_\_\_\_\_\_\_

Дата\_\_\_\_\_\_

# ЦЕЛЬ РАБОТЫ

Получить практические навыки по коррекции ошибок при помощи кода Хэмминга

# ЗАДАНИЕ

Реализовать коррекцию ошибки в двоичной кодовой последовательности, используя метод Хэмминга.

# КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ

Код Хэмминга — самоконтролирующийся и самокорректирующийся код. Построен применительно к двоичной системе счисления.

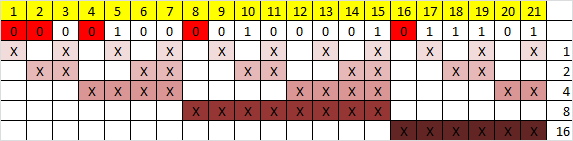
Позволяет исправлять одиночную ошибку (ошибка в одном бите слова) и находить двойную.

Он широко использовался в Dial Up сетях за счет его простоты и скорости работы. Хотя в настоящее время куда чаще используются циклические коды (например, Рида-Соломона)

# ХОД РАБОТЫ

Программа выполняет следующую последовательность действий:

1. Преобразуем строку в ее бинарное представление
2. Вставим контрольные биты (нули) в места строки, индекс которых делится на 2. Т.е. из слова 11111 получится 0010111011
3. Дальше для каждого N контрольного бита посчитаем значение, сложив по модулю все последующие N бит через каждые N бит (см. рисунок для примера)



1. Передадим информацию по каналу связи
2. Приняв информацию, пересчитаем все контрольные биты заново, и запомним те биты, значение которых не сошлось
3. Далее складываем индекс ошибочных контрольных битов и получаем индекс бита с ошибкой.
4. Далее просто инвертируем ошибочный бит, и все!

# РАБОТА ПРОГРАММЫ

Программа на лету генерирует бинарное представление строки, и, сразу же преобразует его по коду Хэмминга, вставляя контрольные биты.

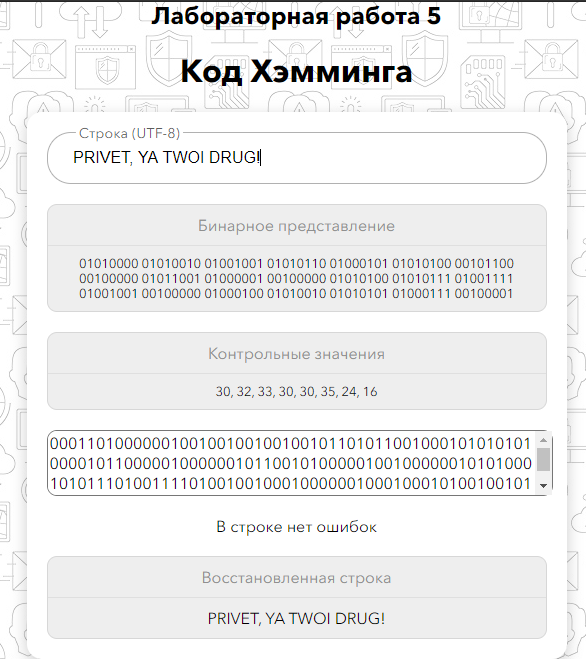


Рисунок 1. Основное окно программы

Как видно на рисунке 1, ошибок в строке нет. Тогда давайте изменим бинарную строку, изменив какой-нибудь бит. И, словно это какая-то магия, программа определяет точное место ошибки и восстанавливает исходную строку

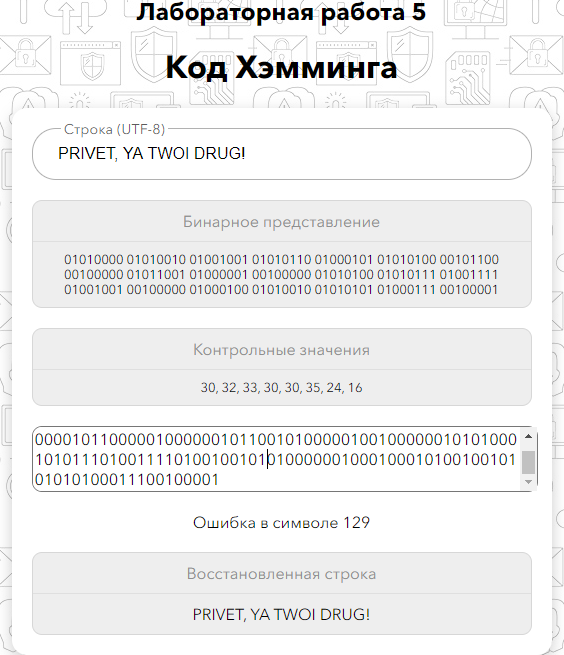


Рисунок 2. Вычисление и коррекция ошибки

# КОД ПРОГРАММЫ (ФУНКЦИЯ ПРЕОБРАЗОВАНИЯ)

# (Язык программирования – JavaScript)

//Преобразование в двоичную строку

const binary = [];

for (var i = 0; i < obj.input.length; i++)

binary.push(zeroPad(obj.input[i].charCodeAt(0).toString(2)));

const str = binary.join("");

let k = 0;

let n = 1;

let hamming = [];

while(k < str.length){

if(hamming.length < n-1){

const end = k+n-1-hamming.length;

for(let i = k; i < Math.min(end, str.length); i++){

hamming.push(str[i]);

}

k = end;

}

if(k < str.length)

hamming.push(0);

n \*= 2;

}

//Вставка контрольных битов

n = 1;

const controlBits = [];

while(n < hamming.length){

const a = 0;

for(let i = n-1; i < hamming.length; i+=n\*2)

for(let j = 0; j < n; j++)

a += hamming[i+j] === '1';

controlBits.push(a);

n \*= 2;

}

obj.controlBits = controlBits;

//Вычисление контрольных битов

n = 1;

for(let i = 0; i < controlBits.length; i++){

hamming[n-1] = controlBits[i]%2;

n \*= 2;

}

obj.hamming = hamming.join("");

obj.binary = binary;

obj.error = 0;

}

//Вычисление ошибки

let error = 0;

if(obj.hamming){

let n = 1;

while(n < obj.hamming.length){

const a = 0;

for(let i = n-1; i < obj.hamming.length; i+=n\*2)

for(let j = 0; j < n; j++)

a += obj.hamming[i+j] === '1';

if(a%2 === 1)

error += n;

n\*= 2;

}

//Коррекция ошибки

let str = "";

let array = [];

n = 1;

for(let i = 1; i < obj.hamming.length+1; i++){

if(i === n){ n\*=2; continue; }

str += (i !== error)? obj.hamming[i-1]: (obj.hamming[i-1]==='1'?'0': '1'); //Инвертируем ошибочный бит

if(str.length === 8){

array.push(parseInt(str, 2));

str ="";

}

}

obj.error = error;

obj.str = String.fromCharCode(...array);