БУ ВО Ханты – Мансийского автономного округа – Югры

«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерный систем

**ЛАБОРАТОРНАЯ РАБОТА №1**

по дисциплине: «Основы защиты информации»

по теме: «Код Хэмминга»

Выполнила: студентка группы №609-11

Пашина Илона Руслановна

Приняла: старший преподаватель кафедры АиКС

Кривицкая Марина Александровна

Сургут

2024 г.

**Тема**: самокорректирующиеся коды.

**Задание на лабораторную работу:**

1. Программно реализовать кодирование по Хэммингу.
2. Внести изменения в случайном бите кодового сообщения.
3. Провести раскодирование сообщения и определить позицию ошибки.
4. Оформить отчет (титульный лист, задание, программный код, результат работы) и разместить его на LMS MOODLE.

**Среда реализации**: любая.

**Листинг**

package main

import (

"fmt"

"math"

"math/rand"

"strconv"

"time")

func stringToBinary(str string) string {

var binary string

for \_, r := range str {

asciiCode := int(r)

binaryCode := strconv.FormatInt(int64(asciiCode), 2)

binaryCode = fmt.Sprintf("%08s", binaryCode)

binary += binaryCode

}

return binary

}

func binaryToString(binary string) string {

var str string

for i := 0; i < len(binary); i += 8 {

binaryCode := binary[i : i+8]

asciiCode, \_ := strconv.ParseInt(binaryCode, 2, 0)

str += string(asciiCode)

}

return str

}

func getExponent(number int) int {

exponent := 0

for number > 1 {

number /= 2

exponent++

}

return exponent

}

func countControlBits(str string, exp int) string {

runes := []rune(str)

for i := 0; i <= exp; i++ {

count := 0

max := int(math.Pow(2, float64(i)))

temp := 1

for j := int(math.Pow(2, float64(i)))-1; j < len(str); j++ {

if temp <= max {

count += int(runes[j]) -'0'

} else if temp == max\*2{

temp = 1

continue

}

temp++

}

runes[max-1] = rune('0' + (count%2)\*1)

}

return string(runes)

}

func codingBinaryString(str string) string {

var binary string

counter := 3

bit\_exp := 2

for \_, r := range str {

if (1 << bit\_exp) == counter {

binary += "0"

bit\_exp++

counter++

}

binary += string(r)

counter++

}

binary = "00" + binary

return binary

}

func makeError(str string) string {

rand.Seed(time.Now().UnixNano())

index := rand.Intn(len(str))

fmt.Println("index position: ", index)

runes := []rune(str)

if runes[index] == '0' {

runes[index] = '1'

} else {

runes[index] = '0'

}

return string(runes)

}

func cleanBinaryString(str string) string { bit\_exp := 2

max := len(str)

runes := []rune(str)

runes[0] = '0'

runes[1] = '0'

for i := 2; i < max; i++ {

if (1 << bit\_exp) == i+1 {

runes[i] = '0'

bit\_exp++

}

}

return string(runes)

}

func findError(str string, input string) string{

runes := []rune(str)

length := len(str)

bit := 0

error\_:= -1

for i:= int(math.Pow(2, float64(bit)))-1; i < length; i = int(math.Pow(2, float64(bit)))-1{

fmt.Print(" ", i)

if str[i] != input[i]{

error\_ += int(math.Pow(2, float64(bit)))

}

bit++

}

fmt.Println("\nerror = ", error\_)

if error\_ >= 0{

runes[error\_] = runes[error\_]^1

}

return string(runes)

}

func deleteControlBits(str string) string{

var latin string

counter := 3

bit\_exp := 2

for \_, r := range str[2:] {

if (1 << bit\_exp) == counter {

bit\_exp++

counter++

continue

}

latin += string(r)

counter++

}

return latin

}

func print(str string){

counter := 2

bit\_exp := 2

fmt.Print(string(str[0]))

fmt.Print(string(str[1]))

fmt.Print(" ")

for \_, r := range str[2:] {

if (1 << bit\_exp) == counter+1{

fmt.Print(" ")

fmt.Print(string(r))

fmt.Print(" ")

bit\_exp++

}else{

fmt.Print(string(r))

}

counter++

}

fmt.Println()

}

func main() {

var input string

var input2 string

fmt.Scanln(&input)

binaryString := stringToBinary(input)

fmt.Println(binaryString)

fmt.Print("дано : ")

fmt.Println(binaryString)

length := len(binaryString)

fmt.Println("Длина строки:", length)

count\_bit := getExponent(length)

fmt.Println("Степень двойки:", count\_bit)

input = codingBinaryString(binaryString)

fmt.Print("к.б. = 0: ")

print(input)

input = countControlBits(input, count\_bit)

fmt.Print("к.б. подсчитн: ")

print(input)

input = makeError(input)

fmt.Print("добав. ошибки: ")

print(input)

input2 = cleanBinaryString(input)

fmt.Print("очистка к.б.: ")

print(input2)

count\_bit = getExponent(length)

input2 = countControlBits(input2, count\_bit)

fmt.Print("к.б. подсчитн: ")

print(input2)

input2 = findError(input2, input)

fmt.Print("исправ. ошиб.: ")

print(input2)

input2 = deleteControlBits(input2)

fmt.Print("удалёны к.б.: ")

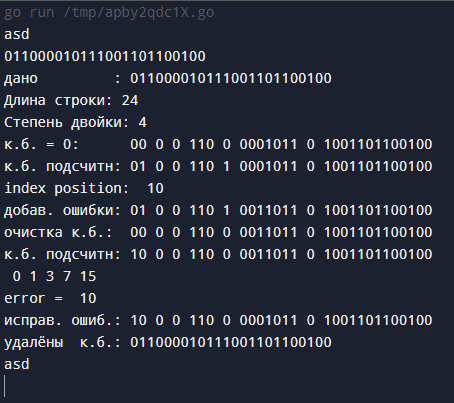
fmt.Println(input2)

input2 = binaryToString(input2)

fmt.Println(input2)

}

**Пример работы программы:**



**Вывод:**

В ходе лабораторной работы был изучен и написан код Хэмминга.