МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Крымский Федеральный Университет им. В. И. Вернадского»

Физико-технический институт

Кафедра компьютерной инженерии и моделирования

Лабораторная работа № 5

«Количество информации и неопределённость сообщения»

по дисциплине

«КОД ХЕММИНГА»

Выполнил:
Студент 3 курса
Направления подготовки 09.03.04
«Программная инженерия»

ПИ-231(2)
Покидько М.С.
Проверил:
Таран Е.П.
«___» _____ 20__ г.

Оценка: _____

Подпись: _____

Симферополь, 2025

Цель работы: изучить принципы построения циклических кодов, реализовать алгоритмы кодирования и декодирования с возможностью исправления одиночных ошибок и обнаружения двойных.

Техническое задание: на вход передающего устройства поступает информационная последовательность длиной k бит. Необходимо разработать программное обеспечение, которое:

1. Выбирает образующий полином P(x) и формирует циклический код.
2. Добавляет бит общей четности для расширения обнаруживающей способности.
3. Моделирует передачу данных через канал с помехами (внесение 1 или 2 ошибок).
4. На приемной стороне вычисляет синдром, локализует и исправляет одиночную ошибку или сигнализирует о наличии двойной.

Основные формулы:

1. Представление кодовой комбинации в виде полинома: $A(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0$ Формирование циклического кода: $C(x) = A(x) \cdot x^p \oplus R(x)$, где R(x) — остаток от деления $(A(x) \cdot x^p)$ на $P(x)$.
2. Условие отсутствия ошибок (синдром): $S(x) = C'(x) \pmod{P(x)} = 0$
3. Общая проверка на четность: $P = \bigoplus_{i=0}^{n-1} c_i$

Разработанная программа: Реализована на языке **Golang** с использованием стандартных библиотек для работы со строками, математическими операциями и генерацией случайных чисел.

1. **Функции:**
   - `getRemainder` — реализует деление полиномов в поле GF(2) для получения синдрома/остатка.
   - `calculateOverallParity` — вычисляет бит четности для всей комбинации.
   - `syndromeTable` — карта (map), сопоставляющая бинарный вид остатка с номером позиции бита, в котором произошла ошибка.
2. **Проведение экспериментов:** Программа запускает цикл из 6 итераций. В каждом эксперименте:
   - Генерируется случайный вектор данных длины k=4.
   - Выполняется циклическое кодирование и добавление бита P.
   - Вносятся искусственные помехи (1 ошибка для проверки исправления, 2 ошибки для проверки обнаружения).
   - Вычисляется синдром S и общая четность Pdiff.
   - На основе анализа (S,Pdiff) принимается решение о статусе коррекции.

Код программы:

```go
package main

import (
    "fmt"
    "math/rand"
    "strings"
    "time"
)

// Твой параметр k по варианту
const K = 4

// Вспомогательная функция для печати слайса в строку
func sliceToString(s []int) string {
    res := ""
    for _, v := range s {
        res += fmt.Sprintf("%d", v)
    }
    return res
}

// Деление полиномов в GF(2) для получения остатка
func getRemainder(dividend, divisor []int) []int {
    // Исправлено: создаем слайс нужной длины типа []int
    res := make([]int, len(dividend))
    copy(res, dividend)

    for i := 0; i <= len(res)-len(divisor); i++ {
        if res[i] == 1 {
            for j := 0; j < len(divisor); j++ {
                res[i+j] ^= divisor[j]
            }
        }
    }
    // Возвращаем последние p разрядов
    return res[len(res)-(len(divisor)-1):]
}

func copySlice(s []int) []int {
    res := make([]int, len(s))
    copy(res, s)
    return res
}

// Расчет бита общей четности
func calculateOverallParity(data []int) int {
    p := 0
    for _, v := range data {
        p ^= v
    }
    return p
}
```

```
func main() {
    rand.Seed(time.Now().UnixNano())

    // Образующий полином для k=4: x^3 + x + 1
    genPol := []int{1, 0, 1, 1}
    pCount := len(genPol) - 1
    nPrime := K + pCount + 1 // n + 1 (общая четность)

    // Построение таблицы синдромов
    syndromeTable := make(map[string]int)
    for i := 0; i < K+pCount; i++ {
        v := make([]int, K+pCount)
        v[i] = 1
        rem := getRemainder(v, genPol)
        syndromeTable[sliceToString(rem)] = i
    }

    for i := 1; i <= 6; i++ {
        // а) Генерация инфо-кода
        info := make([]int, K)
        for j := range info {
            info[j] = rand.Intn(2)
        }

        // б) Кодирование
        cyclicPart := make([]int, K+pCount)
        copy(cyclicPart, info)
        rem := getRemainder(cyclicPart, genPol)
        for j := 0; j < pCount; j++ {
            cyclicPart[K+j] = rem[j]
        }

        // Добавление общей четности (P) в начало
        overallP := calculateOverallParity(cyclicPart)
        finalEncoded := append([]int{overallP}, cyclicPart...)

        // г) Внесение ошибки (1 или 2 для демонстрации)
        errCount := 1
        if i == 3 || i == 5 || i == 6 {
            errCount = 2
        } // Тестируем двойные ошибки

        received := copySlice(finalEncoded)
        var errPositions []int
        posUsed := make(map[int]bool)
        for len(errPositions) < errCount {
            p := rand.Intn(len(received))
            if !posUsed[p] {
                posUsed[p] = true
                errPositions = append(errPositions, p)
                received[p] ^= 1
            }
        }
```

```
        // --- ДЕКОДИРОВАНИЕ ---
        recP := received[0]
        recData := received[1:]
        calcP := calculateOverallParity(recData)
        syndrome := getRemainder(recData, genPol)

        isSynZero := true
        synVal := 0
        for _, v := range syndrome {
            synVal = (synVal << 1) | v
            if v != 0 {
                isSynZero = false
            }
        }

        pDiff := recP ^ calcP
        var msg, status string
        corrected := copySlice(received)
        detectedCount := errCount

        if isSynZero && pDiff == 0 {
            msg = "Ошибок не обнаружено."
            status = "УСПЕШНО"
            detectedCount = 0
        } else if !isSynZero && pDiff != 0 {
            posInTable, ok := syndromeTable[sliceToString(syndrome)]
            if ok {
                corrected[posInTable+1] ^= 1
                msg = fmt.Sprintf("Обнаружена и исправлена однократная ошибка в позиции
%d.", posInTable+1)
                status = "УСПЕШНО"
                detectedCount = 1
            }
        } else if !isSynZero && pDiff == 0 {
            msg = "Обнаружена двукратная ошибка. Исправление невозможно."
            status = "НЕ ТРЕБУЕТСЯ / НЕВОЗМОЖНО"
            detectedCount = 2
        } else {
            msg = "Ошибка в проверочном бите четности."
            status = "УСПЕШНО"
            corrected[0] ^= 1
            detectedCount = 1
        }

        // ВЫВОД В ФОРМАТЕ ОДНОКУРСНИКА
        fmt.Printf("[ЭКСПЕРИМЕНТ %d]\n", i)
        fmt.Println(strings.Repeat("-", 40))
        fmt.Printf(" Инф. код (k=%d):    %s\n", K, sliceToString(info))
        fmt.Printf(" Переданный код (n'=%d): %s\n", nPrime, sliceToString(finalEncoded))
        fmt.Printf(" Кратность ошибки (ист.): %d\n", errCount)
        fmt.Printf(" Позиции ошибки (ист.):   %v\n", errPositions)
        fmt.Printf(" Принятый код (R):     %s\n\n", sliceToString(received))
        fmt.Println(" *** Результат декодирования ***")
```

```
        fmt.Printf(" Синдром S (dec/bin): %d/%s\n", synVal, sliceToString(syndrome))
        fmt.Printf(" Общая четность P:    %d\n", pDiff)
        fmt.Printf(" Обнаруж. кратность:  %d\n", detectedCount)
        fmt.Printf(" Сообщение:            %s\n", msg)
        fmt.Printf(" Исправленный код (C'): %s\n", sliceToString(corrected))
        fmt.Printf(" Статус коррекции:     %s\n\n", status)
    }
}
```

Результаты выполнения программы: Всего было проведено 6 экспериментов. Использован образующий полином P(x)=x3+x+1. Параметры кода: k=4,n′=8 (с учетом бита четности).

Результаты экспериментов:

```
[ЭКСПЕРИМЕНТ 1]
----------------------------------------
 Инф. код (k=4):    0111
 Переданный код (n'=8): 00111010
 Кратность ошибки (ист.): 1
 Позиции ошибки (ист.):   [6]
 Принятый код (R):     00111000

 *** Результат декодирования ***
 Синдром S (dec/bin): 2/010
 Общая четность P:    1
 Обнаруж. кратность:  1
 Сообщение:           Обнаружена и исправлена однократная ошибка в позиции 6.
 Исправленный код (C'): 00111010
 Статус коррекции:     УСПЕШНО

[ЭКСПЕРИМЕНТ 2]
----------------------------------------
 Инф. код (k=4):    0001
 Переданный код (n'=8): 10001011
 Кратность ошибки (ист.): 1
 Позиции ошибки (ист.):   [7]
 Принятый код (R):     10001010

 *** Результат декодирования ***
 Синдром S (dec/bin): 1/001
 Общая четность P:    1
 Обнаруж. кратность:  1
 Сообщение:           Обнаружена и исправлена однократная ошибка в позиции 7.
 Исправленный код (C'): 10001011
 Статус коррекции:     УСПЕШНО

[ЭКСПЕРИМЕНТ 3]
----------------------------------------
 Инф. код (k=4):    0001
 Переданный код (n'=8): 10001011
 Кратность ошибки (ист.): 2
 Позиции ошибки (ист.):   [6 0]
 Принятый код (R):     00001001

 *** Результат декодирования ***
 Синдром S (dec/bin): 2/010
 Общая четность P:    0
 Обнаруж. кратность:  2
 Сообщение:           Обнаружена двукратная ошибка. Исправление невозможно.
 Исправленный код (C'): 00001001
 Статус коррекции:     НЕ ТРЕБУЕТСЯ / НЕВОЗМОЖНО
```

```
[ЭКСПЕРИМЕНТ 4]
---------------------------------------
 Инф. код (k=4):    0110
 Переданный код (n'=8): 10110001
 Кратность ошибки (ист.): 1
 Позиции ошибки (ист.):   [3]
 Принятый код (R):     10100001

 *** Результат декодирования ***
 Синдром S (dec/bin): 6/110
 Общая четность P:    1
 Обнаруж. кратность:  1
 Сообщение:           Обнаружена и исправлена однократная ошибка в позиции 3.
 Исправленный код (C'): 10110001
 Статус коррекции:     УСПЕШНО

[ЭКСПЕРИМЕНТ 5]
---------------------------------------
 Инф. код (k=4):    0110
 Переданный код (n'=8): 10110001
 Кратность ошибки (ист.): 2
 Позиции ошибки (ист.):   [2 0]
 Принятый код (R):     00010001

 *** Результат декодирования ***
 Синдром S (dec/bin): 7/111
 Общая четность P:    0
 Обнаруж. кратность:  2
 Сообщение:           Обнаружена двукратная ошибка. Исправление невозможно.
 Исправленный код (C'): 00010001
 Статус коррекции:     НЕ ТРЕБУЕТСЯ / НЕВОЗМОЖНО

[ЭКСПЕРИМЕНТ 6]
---------------------------------------
 Инф. код (k=4):    0001
 Переданный код (n'=8): 10001011
 Кратность ошибки (ист.): 2
 Позиции ошибки (ист.):   [7 2]
---------------------------------------
 Инф. код (k=4):    0001
---------------------------------------
---------------------------------------
 Инф. код (k=4):    0001
 Переданный код (n'=8): 10001011
 Кратность ошибки (ист.): 2
 Позиции ошибки (ист.):   [7 2]
 Принятый код (R):     10101010

 *** Результат декодирования ***
 Синдром S (dec/bin): 6/110
 Общая четность P:    0
 Обнаруж. кратность:  2
 Сообщение:           Обнаружена двукратная ошибка. Исправление невозможно.
 Исправленный код (C'): 10101010
 Статус коррекции:     НЕ ТРЕБУЕТСЯ / НЕВОЗМОЖНО
```

Вывод: В ходе лабораторной работы было разработано ПО для моделирования циклического кода. В ходе 6 экспериментов подтверждена способность кода исправлять одиночные инверсии бит и обнаруживать двойные ошибки за счет использования образующего полинома и бита общей четности. Алгоритм декодирования по таблице синдромов показал корректную работу во всех случаях.