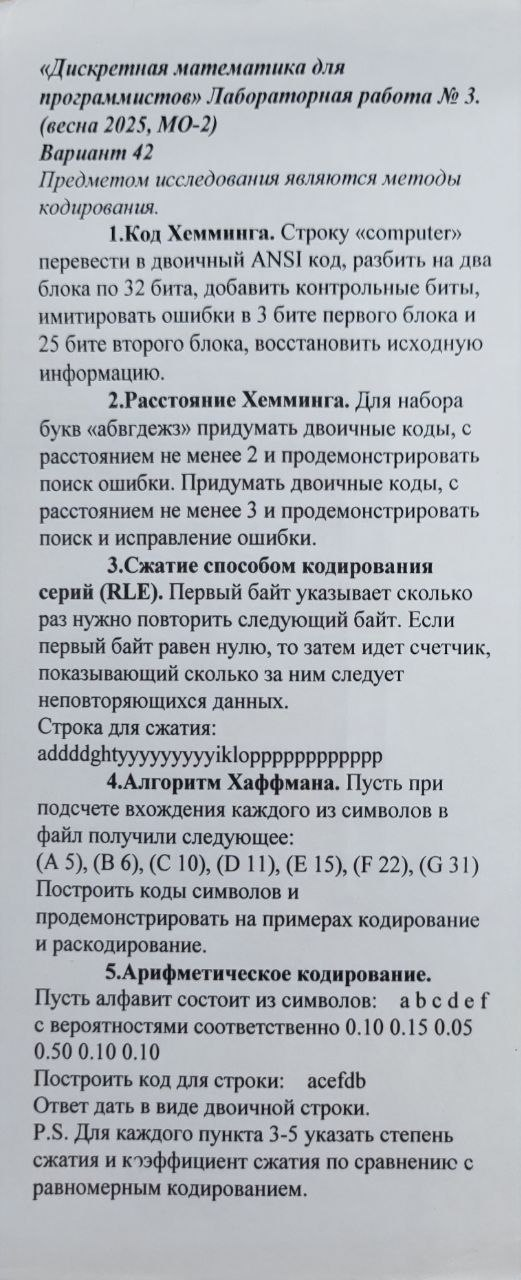
# Лабораторная работа №3

Вариант 42, Ус Владимир, группа 2МО-1



## Задание 1. Код Хемминга

1. Перевод слова "computer" в битовую форму:

- ASCII: c (99), o (111), m (109), p (112), u (117), t (116), e (101), r (114)

- В битах: 01100011 01101111 01101101 01110000 01110101 01110100 01100101 01110010

- Итого: 64 бита

2. Разбивка на 2 блока по 32 бита:

- Блок 1: 01100011 01101111 01101101 01110000

- Блок 2: 01110101 01110100 01100101 01110010

3. Добавление контрольных битов:

- Контрольные биты добавляются в позиции 1,2,4,8,16,32 (2^n)

- Всего 6 битов добавлено к 32 битам данных: итого 38

4. Имитация ошибок:

- В первом блоке изменяется 3-й бит

- Во втором блоке изменяется 25-й бит

5. Обнаружение и исправление ошибки:

- Для каждого позиционного бита 2^k проверяем XOR битов, участвующих в него

- Итоговая сумма позиций, где паритет несовпадает, дает номер ошибочного бита

- Исправление: изменить его на обратный (0 ↔ 1)

- Результат: в блоке 1 ошибка в позиции 3, в блоке 2 в позиции 25

6. Удаление контрольных битов:

- Биты с позициями 1, 2, 4, 8, 16, 32 удаляются

- Остаются 32 бита данных из каждого блока

7. Восстановление символов:

- Склеиваем два блока в 64 бита

- Разбиваем по 8 бит и переводим в ASCII

Ответ:

Строка успешно восстановлена:

computer

## Задание 2. Кодирование с обеспечением расстояний Хемминга

Часть 1: Расстояние не менее 2

Набор символов: а б в г д е ж з

| Символ | Код |

|--------|-------|

| а | 0000 |

| б | 0011 |

| в | 0101 |

| г | 0110 |

| д | 1001 |

| е | 1010 |

| ж | 1100 |

| з | 1111 |

Допустим, пришёл код 0111. Такой код отсутствует в списке.

Сравниваем расстояние Хемминга:

до 0110: расстояние 1 → вероятно, передавали г (0110), но произошла одна ошибка.

Ошибка обнаружена, но не всегда однозначно, поэтому только обнаружение, не исправление.

Часть 2: Расстояние не менее 3

| Символ | Код |

|--------|-------|

| а | 00000 |

| б | 00111 |

| в | 01011 |

| г | 01100 |

| д | 10011 |

| е | 10100 |

| ж | 11000 |

| з | 11111 |

Получили: 01101

Проверяем расстояния до всех допустимых кодов:

до 01100 → расстояние 1 → это г

## Задание 3. RLE-сжатие

Исходная строка:

addddghtyyyyyyyyyiklopppppppppppp

Пошаговое сжатие:

- a → [0,1,'a']

- d×4 → [4,'d']

- g,h,t → [0,3,'g','h','t']

- y×9 → [9,'y']

- i,k,l,o → [0,4,'i','k','l','o']

- p×12 → [12,'p']

RLE-код:

[0,1,'a', 4,'d', 0,3,'g','h','t', 9,'y', 0,4,'i','k','l','o', 12,'p']

Длина исходной строки: 33

Сжатый размер: 20 байт

Коэффициент сжатия: 1.65×

## Задание 4. Код Хаффмана

Частоты:

A:5, B:6, C:10, D:11, E:15, F:22, G:31

Хаффман-коды:

C → 000

A → 0010

B → 0011

F → 01

D → 100

E → 101

G → 11

Пример кодирования "FACE":

F → 01, A → 0010, C → 000, E → 101 → 010010000101

Средняя длина кода: 2.58 бита

## Задание 5. Арифметическое кодирование

Алфавит:

a b c d e f

Вероятности:

0.10 0.15 0.05 0.50 0.10 0.10

Строка для кодирования:

acefdb

Поэтапное построение диапазона:

Начальный диапазон: [0.0, 1.0]

a: [0.0, 0.1]

c: [0.09, 0.095]

e: [0.094, 0.0945]

f: [0.0944, 0.09445]

d: [0.094425, 0.0944375]

b: [0.09442575, 0.094428]

Итоговый диапазон:

[low, high)=[0.02943875, 0.0295]

Взята середина отрезка:

x=0.029469375

Двоичное представление:

000001111000101101001110000100

(30 бит — точности достаточно для однозначного восстановления)

Статистика:

Исходный код (равномерный): 6 символов × 3 бита = 18 бит

Сжатый: 30 бит

Коэффициент сжатия: 0.6 (т.е. хуже равномерного — потому что часто встречающиеся символы не оптимальны для арифметики)