1---

2title: "Лабораторная работа №2" 3format:

1. html:
2. toc: true
3. code-fold: true
4. code-line-numbers: true
5. code-tools: true

9---

10

11

12## Тема: Шифры перестановки

13

14\*\*Выполнила:\*\* Исламова Сания Маратовна (студ. билет 1132249576) 15

16--17

18```{julia}

19#| label: cipher-program

20#| fig-cap: "Реализация шифров Цезаря и Атбаш"

21#| code-line-numbers: "true"

22#| warning: false

23#| eval: false

24

25

26

27# Маршрутное шифрование

28

29

30function columnar\_main()

31

32 # Бесконечный цикл для работы программы до команды выхода

33

34 while true

35

36 # Выводим меню с доступными командами

37

38 println("Введите Ш для шифрования, Р для расшифрования, В для выхода") 39

1. # Приглашение для ввода команды
2. print(">>> ")
3. # Читаем ввод пользователя, удаляем пробелы и приводим к нижнему регистру43 cmd = lowercase(strip(readline()))

44

1. # Проверяем команду выхода: если "в", то выводим сообщение и прерываем цикл
2. cmd == "в" && (println("Выход"); break)
3. # Проверяем корректность команды: если не "ш" и не "р", выводим ошибку и продолжаем цикл
4. cmd in ["ш", "р"] || (println("Неверная команда"); continue)49
5. # Запрашиваем текст для шифрования/расшифрования
6. print("Введите текст: ")
7. # Читаем введенный текст
8. text = readline()
9. # Запрашиваем пароль для шифрования
10. print("Введите пароль: ")
11. # Читаем введенный пароль
12. password = readline()

58

1. # ПОДГОТОВКА ТЕКСТА:
2. # Удаляем пробелы и приводим к верхнему регистру
3. clean\_text = replace(uppercase(text), " " => "")
4. # Преобразуем пароль в массив символов (для избежания проблем с индексацией русских символов)
5. pass\_chars = collect(uppercase(password))
6. # n - количество столбцов (равно длине пароля)
7. # m - количество строк (вычисляем округлением вверх длины текста / длины пароля)
8. n, m = length(pass\_chars), ceil(Int, length(clean\_text) / length(pass\_chars))

67

1. # СОЗДАНИЕ ТАБЛИЦЫ:
2. # Дополняем текст символами 'А' до полного заполнения таблицы m×n
3. padded = clean\_text \* "А"^(m\*n - length(clean\_text))
4. # Преобразуем строку в массив символов и reshape в матрицу m×n72 table = reshape(collect(padded), (m, n))

73

1. # СОРТИРОВКА СТОЛБЦОВ:
2. # Создаем пары (символ пароля, индекс столбца) для каждого столбца
3. column\_pairs = [(pass\_chars[i], i) for i in 1:length(pass\_chars)]
4. # Сортируем пары по символам пароля (алфавитный порядок)
5. sort!(column\_pairs, by = x -> x[1])
6. # Извлекаем отсортированные индексы столбцов
7. sorted\_cols = [idx for (char, idx) in column\_pairs]

81

1. # ФОРМИРОВАНИЕ РЕЗУЛЬТАТА:
2. # Выписываем символы из таблицы по столбцам в новом порядке
3. # Сначала все строки первого столбца, затем второго и т.д.
4. result = join([table[i,j] for j in sorted\_cols for i in 1:m])

86

1. # Выводим результат шифрования
2. println("Результат: $result")
3. # Разделительная линия для визуального отделения
4. println("-"^50)
5. end

92end

93

94#Запуск основной функции программы

95columnar\_main()

96

97

98

99

100

101# Шифрование с помощью решёток

102

103

104function fleissner\_main()

105

1. # Бесконечный цикл для работы программы до команды выхода
2. while true

108

1. # Выводим меню с доступными командами
2. println("Введите Ш для шифрования, Р для расшифрования, В для выхода")111
3. # Приглашение для ввода команды
4. print(">>> ")

114

115 # Читаем ввод пользователя, удаляем пробелы и приводим к нижнему регистру 116 cmd = lowercase(strip(readline()))

117

1. # Проверяем команду выхода: если "в", то выводим сообщение и прерываем цикл
2. cmd == "в" && (println("Выход"); break)
3. # Проверяем корректность команды: если не "ш" и не "р", выводим ошибку и продолжаем цикл
4. cmd in ["ш", "р"] || (println("Неверная команда"); continue)122
5. # Запрашиваем текст для шифрования
6. print("Введите текст: ")
7. # Читаем введенный текст
8. text = readline()
9. # Запрашиваем пароль (должен содержать 4 символа для решетки 2x2)
10. print("Введите пароль (4 символа): ")
11. # Читаем введенный пароль
12. password = readline()

131

1. # ПОДГОТОВКА ТЕКСТА:
2. # Удаляем пробелы, приводим к верхнему регистру и преобразуем в массив символов
3. clean\_chars = collect(replace(uppercase(text), " " => ""))
4. # Преобразуем пароль в массив символов
5. pass\_chars = collect(uppercase(password))
6. # k = 2 означает решетку 2x2, которая создает маску 4x4
7. k = 2 # размер решетки

139

1. # СОЗДАНИЕ РЕШЕТКИ 4x4:
2. # Размер большой решетки (2k × 2k = 4x4)
3. size\_2k = 2k
4. # Создаем булеву маску (false - закрыто, true - прорезь)
5. grille = falses(size\_2k, size\_2k)
6. # Заполняем маску прорезями в 4 угловых квадратах 2x2
7. for i in 1:k, j in 1:k
8. grille[i, j] = grille[i, k+j] = grille[k+i, j] = grille[k+i, k+j] = true148 end

149

1. # ЗАПОЛНЕНИЕ ТАБЛИЦЫ:
2. # Общее количество ячеек в решетке
3. total = size\_2k^2
4. # Если текст короче, дополняем символами 'А'
5. length(clean\_chars) < total && append!(clean\_chars, fill('А', total - length(clean\_chars)))
6. # Инициализируем таблицу для заполнения, индекс текста и копию маски156 table, idx, mask = fill(' ', size\_2k, size\_2k), 1, copy(grille)

157

1. # ЧЕТЫРЕ ЭТАПА ЗАПОЛНЕНИЯ (0°, 90°, 180°, 270°):
2. for \_ in 1:4
3. # Проходим по всем ячейкам решетки
4. for i in 1:size\_2k, j in 1:size\_2k
5. # Если ячейка - прорезь и есть еще символы для записи
6. mask[i,j] && idx <= length(clean\_chars) && (table[i,j] = clean\_chars[idx]; idx += 1)
7. end
8. # Поворачиваем маску на 90° по часовой стрелке для следующего этапа166 mask = reverse(mask, dims=1)'

167 end

168

1. # ФОРМИРОВАНИЕ РЕЗУЛЬТАТА:
2. # Сортируем индексы столбцов по алфавитному порядку символов пароля
3. sorted\_cols = sort(1:length(pass\_chars), by=i -> pass\_chars[i])
4. # Выписываем символы из таблицы по столбцам в новом порядке
5. result = join([table[i,j] for j in sorted\_cols for i in 1:size\_2k])174
6. # Выводим результат шифрования
7. println("Результат: $result")
8. # Разделительная линия для визуального отделения
9. println("-"^50)
10. end

180end

181

182#Запуск основной функции программы

183fleissner\_main()

184

185

186

187

188

189

190# Таблица Виженера

191

192

193

194function vigenere\_main()

195

196 # Создаем русский алфавит как массив символов для корректной работы с индексами 197

198 alphabet = collect("АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ") 199

200 # Сохраняем длину алфавита для модульных вычислений

201

202 n = length(alphabet)

203

1. # Бесконечный цикл для работы программы до команды выхода
2. while true
3. # Выводим меню с доступными командами
4. println("Введите Ш для шифрования, Р для расшифрования, В для выхода")
5. # Приглашение для ввода команды
6. print(">>> ")
7. # Читаем ввод пользователя, удаляем пробелы и приводим к нижнему регистру211 cmd = lowercase(strip(readline()))

212

1. # Проверяем команду выхода: если "в", то выводим сообщение и прерываем цикл
2. cmd == "в" && (println("Выход"); break)
3. # Проверяем корректность команды: если не "ш" и не "р", выводим ошибку и продолжаем цикл
4. cmd in ["ш", "р"] || (println("Неверная команда"); continue)

217

1. # Запрашиваем текст для шифрования/расшифрования
2. print("Введите текст: ")
3. # Читаем введенный текст
4. text = readline()
5. # Запрашиваем пароль для шифрования
6. print("Введите пароль: ")
7. # Читаем введенный пароль
8. password = readline()

226

1. # ПОДГОТОВКА ТЕКСТА:
2. # Удаляем пробелы, приводим к верхнему регистру и преобразуем в массив символов
3. clean\_chars = collect(replace(uppercase(text), " " => ""))
4. # Аналогично обрабатываем пароль
5. pass\_chars = collect(replace(uppercase(password), " " => ""))

232

1. # СОЗДАНИЕ ПОВТОРЯЮЩЕГОСЯ КЛЮЧА:
2. # Создаем пустой массив символов для ключа
3. key\_chars = Char[]
4. # Для каждого символа текста определяем соответствующий символ ключа
5. for i in 1:length(clean\_chars)
6. # Циклически повторяем пароль: (i-1) % length + 1 дает циклический индекс
7. push!(key\_chars, pass\_chars[(i-1) % length(pass\_chars) + 1])240 end

241

1. # ШИФРОВАНИЕ/ДЕШИФРОВАНИЕ:
2. # Создаем массив для результата
3. result\_chars = Char[]
4. # Обрабатываем каждый символ текста
5. for i in 1:length(clean\_chars)
6. text\_char = clean\_chars[i] # Текущий символ текста
7. key\_char = key\_chars[i] # Соответствующий символ ключа249
8. # Находим позиции символов в алфавите
9. text\_idx = findfirst(==(text\_char), alphabet)
10. key\_idx = findfirst(==(key\_char), alphabet)

253

1. # Если оба символа найдены в алфавите
2. if text\_idx !== nothing && key\_idx !== nothing
3. if cmd == "ш"
4. # ШИФРОВАНИЕ: (текст + ключ) mod n
5. new\_idx = (text\_idx + key\_idx - 1) % n
6. # Обработка случая, когда mod дает 0
7. new\_idx == 0 && (new\_idx = n)
8. else
9. # ДЕШИФРОВАНИЕ: (текст - ключ) mod n
10. new\_idx = (text\_idx - key\_idx) % n
11. # Обработка отрицательных результатов
12. new\_idx <= 0 && (new\_idx += n)
13. end
14. # Добавляем преобразованный символ к результату
15. push!(result\_chars, alphabet[new\_idx])
16. else
17. # Если символ не из алфавита, добавляем как есть (пробелы, знаки препинания)
18. push!(result\_chars, text\_char)
19. end
20. end

274

1. # Преобразуем массив символов обратно в строку
2. result = String(result\_chars)
3. # Выводим результат
4. println("Результат: $result")
5. # Разделительная линия для визуального отделения
6. println("-"^50)
7. end

282end

283

284#Запуск основной функции программы

285vigenere\_main()