

Отчёт по лабораторной работе №1: Шифры простой замены

**Дисциплина: Математические основы защиты информации и
информационной безопасности**

Савченко Елизавета Николаевна

Содержание

1	Общая информация о задании лабораторной работы	3
1.1	Цель работы	3
1.2	Задание [1]	3
2	Теоретическое введение [2]	3
2.1	Шифры и симметричные шифры	3
2.2	Шифры простой замены	3
2.2.1	Моноалфавитные шифры	3
2.2.2	Шифр Атбаш	4
3	Выполнение лабораторной работы [1]	5
3.1	1. Реализация шифра Цезаря для произвольного ключа k	5
3.2	2. Реализация шифра Атбаш	6
4	Выводы	8
	Список литературы	9

1 Общая информация о задании лабораторной работы

1.1 Цель работы

Ознакомиться с классическими примерами шифров простой замены.

1.2 Задание [1]

1. Реализовать шифр Цезаря с произвольным ключом k ;
2. Реализовать шифр Атбаш.

2 Теоретическое введение [2]

2.1 Шифры и симметричные шифры

Первоначальное сообщение от одного пользователя к другому названо исходным текстом; сообщение, передаваемое через канал, названо зашифрованным текстом. Чтобы создать зашифрованный текст из исходного текста, отправитель использует алгоритм шифрования и совместный ключ засекречивания. Для того чтобы создать обычный текст из зашифрованного текста, получатель использует алгоритм дешифрования и тот же секретный ключ. Мы будем называть совместное действие алгоритмов шифрования и дешифрования шифровкой. Ключ — набор значений (чисел), которыми оперируют алгоритмы шифрования и дешифрования.

Обратите внимание, что шифрование симметричными ключами использует единственный ключ (ключ, содержащий непосредственно набор кодируемых значений) и для кодирования и для дешифрования. Кроме того, алгоритмы шифрования и дешифрования — инверсии друг друга. Если P — обычный текст, C — зашифрованный текст, а K — ключ, алгоритм кодирования $E_k(x)$ создает зашифрованный текст из исходного текста.

Алгоритм же дешифрования $D_k(x)$ создает исходный текст из зашифрованного текста. Мы предполагаем, что $E_k(x)$ и $D_k(x)$ обратны друг другу. Они применяются, последовательно преобразуя информацию из одного вида в другой и обратно.

2.2 Шифры простой замены

Мы можем разделить традиционные шифры с симметричным ключом на две обширные категории: шифры подстановки и шифры перестановки. В шифре подстановки мы заменяем один символ в зашифрованном тексте на другой символ; в шифре перестановки — меняем местами позиции символов в исходном тексте.

Шифры простой замены относятся к шифрам подстановки.

2.2.1 Моноалфавитные шифры

Сначала обсудим шифры подстановки, называемые моноалфавитными шифрами. В такой подстановке буква (или символ) в исходном тексте всегда изменяется на одну и ту же самую букву (или символ) в зашифрованном тексте независимо от его позиции в тексте. Например, если алгоритм определяет, что буква А в исходном тексте меняется

на букву D, то при этом каждая буква A изменяется на букву D. Другими словами, буквы в исходном тексте и зашифрованном тексте находятся в отношении один к одному.

В моноалфавитной подстановке отношения между буквой в исходном тексте и буквой в зашифрованном тексте — один к одному.

2.2.1.1 Аддитивные шифры

Самый простой моноалфавитный шифр — аддитивный шифр, его иногда называют шифром сдвига, а иногда — шифром Цезаря, но термин аддитивный шифр лучше показывает его математический смысл. Предположим, что исходный текст состоит из маленьких букв (от a до z) и зашифрованный текст состоит из заглавных букв (от A до Z). Чтобы обеспечить применение математических операций к исходному и зашифрованному текстам, мы присвоим каждой букве числовое значение (для нижнего и верхнего регистра).

Каждому символу (нижний регистр или верхний регистр) сопоставлено целое число из кольца Z_{26} . Ключ засекречивания между отправителем и получателем — также целое число в кольце Z_n . Алгоритм кодирования прибавляет ключ к символу исходного текста; алгоритм дешифрования вычитает ключ из символа зашифрованного текста. Все операции проводятся в кольце Z_n .

2.2.1.1.1 Шифр сдвига

Исторически аддитивные шифры назывались шифрами сдвига — по той причине, что алгоритм шифрования может интерпретироваться как “клавиша сдвига буквы вниз”, а алгоритм дешифрования может интерпретироваться как “клавиши сдвига буквы вверх”. Например, если ключ = 15, алгоритм кодирования сдвигает букву на 15 букв вниз (к концу алфавита). Алгоритм дешифрования сдвигает букву на 15 букв вверх (к началу алфавита). Конечно, когда мы достигаем конца или начала алфавита, мы двигаемся по кольцу к началу (объявленные свойства операции по модулю 26).

2.2.1.1.2 Шифр Цезаря

Юлий Цезарь использовал аддитивный шифр, чтобы связаться со своими чиновниками. По этой причине аддитивные шифры упоминаются иногда как шифры Цезаря. Цезарь для своей связи использовал цифру 3.

2.2.2 Шифр Атбаш

Шифр Атбаш также является моноалфавитным шифром, однако отличается от шифра Цезаря. Принцип его работы основан на том, что для шифровки сообщения необходимо зеркально отразить алфавит, использующийся при написании этого сообщения. Таким образом, шифр перемещает между собой все символы алфавита (или почти все в алфавитах с нечётным числом символом).

3 Выполнение лабораторной работы [1]

3.1 1. Реализация шифра Цезаря для произвольного ключа k

Для реализации шифра Цезаря необходимо было ограничить алфавит. В тексте лабораторной работы [1] предложен пример использования исключительно латиницы. В своей реализации я предлагаю использовать в качестве алфавита все символы ASCII, которые доступны в Julia [3].

В языке Julia число ASCII символов ограничено 128 [4], в связи с чем при реализации использовалась формула $\text{mod}(\text{Int}(\text{symbol}) + k, 128)$ для преобразования каждого символа symbol , где k – смещение, задаваемое в качестве параметра функции.

```
julia> function Cesar_cipher(text::String, k::Int)
    result = IOBuffer()
    for ch in text
        if 'А' <= ch <= 'Я' # Заглавные русские буквы
            shifted = Char(mod(Int(ch) - Int('А') + k, 32) + Int('А'))
            print(result, shifted)
        elseif 'а' <= ch <= 'я' # Строчные
            русские буквы
            shifted = Char(mod(Int(ch) - Int('а') + k, 32) + Int('а'))
            print(result, shifted)
        else
            # Все другие символы без изменений
            print(result, ch)
        end
    end
end

julia> println(caesar_cipher("Молодец!", 2))
Орнржзш!
```

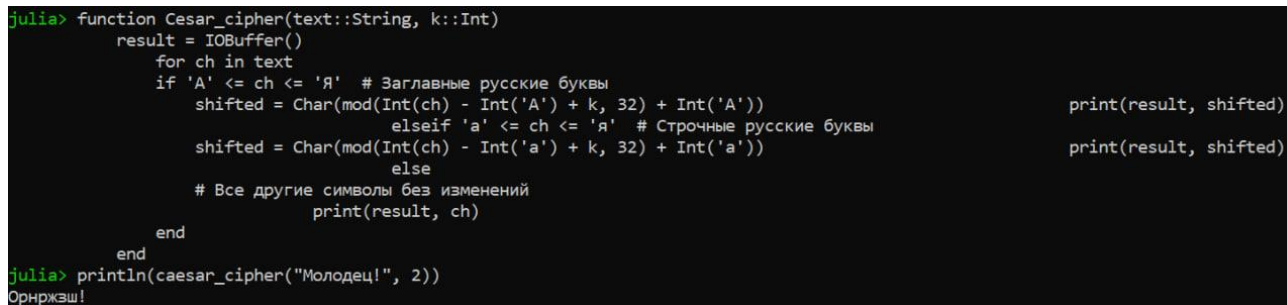
При проверке правильности реализации важно учитывать, что шифры простой замены (а, значит, и шифр Цезаря) относятся к симметричным шифрам. Шифр Цезаря симметричен относительно размера своего алфавита: для расшифровки сообщений необходимо использовать идентичный алфавит, однако при расшифровке необходимо использовать значение параметра $k_{\text{decoder}} = \text{alphabet length} - k_{\text{encoder}}$. Это важно при проверке правильности работы шифра, для чего изначальное сообщение мы пропускаем дважды через функцию, которая зашифровывает сообщение. Так мы должны получить шифрокод после первого запуска функции, и изначальное сообщение после второго запуска функции

```

coded_text = shifrCezarya(3, "TEXT to be coded!!!! αβγ and some innocent letters
")
println("The result of encoding:\n", coded_text, "\n\n")
decoded_text = shifrCezarya(-131, coded_text)
println("The result of decoding:\n", decoded_text)

```

Результат работы кода представлен ниже (рис. 1).



```

julia> function Caesar_cipher(text::String, k::Int)
    result = IOBuffer()
    for ch in text
        if 'А' <= ch <= 'Я' # Заглавные русские буквы
            shifted = Char(mod(Int(ch) - Int('А') + k, 32) + Int('А'))
            print(result, shifted)
        elseif 'а' <= ch <= 'я' # Строчные русские буквы
            shifted = Char(mod(Int(ch) - Int('а') + k, 32) + Int('а'))
            print(result, shifted)
        else
            # Все другие символы без изменений
            print(result, ch)
        end
    end
end
julia> println(caesar_cipher("Молодец!", 2))
Орнржэш!

```

Рис. 1: Результат работы шифра Цезаря

3.2 2. Реализация шифра Атбаш

Для реализации шифра Атбаш необходимо было ограничить алфавит. В тексте лабораторной работы [1] предложен пример на языке Pascal, в котором алфавит символов ограничен 256 символами ASCII.

В языке Julia число ASCII символов ограничено 128 [4], в связи с чем при реализации использовалась формула $127 - \text{Int}(\text{symbol})$ для преобразования каждого символа `symbol`.

```

julia> function atbash_cipher(text::String, k::Int=0)

    # В атбаш обычный ключ не используется, это фиксированное отражение
    # Но если нужен ключ k – реализуем модифицированный атбаш:
    # сначала отражаем буквы, затем сдвигаем результат на k

    function reflect_russian(ch)
        if 'А' <= ch <= 'Я'
            return Char(Int('А') + (Int('Я') - Int(ch)))
        elseif 'а' <= ch <= 'я'
            return Char(Int('а') + (Int('я') - Int(ch)))
        else
            return ch
        end
    end

```

```

end
function shift_russian(ch, k)
    if 'А' <= ch <= 'Я'
        return Char(mod(Int(ch) - Int('А') + k, 32) + Int('А'))
    elseif 'а' <= ch <= 'я'
        return Char(mod(Int(ch) - Int('а') + k, 32) + Int('а'))
    else
        return ch
    end
end
end

```

Для ограничения числа символов, с которыми ведётся работа, до алфавита из 128 ASCII символов, реализованных в Julia, на вводимый текст перед зашифровкой применяется фильтр, который пропускает исключительно символы ASCII и отбрасывает посторонние символы. После этого дополнительно пользователю демонстрируется фраза, которая получилась после фильтрации посторонних символов.

При проверке правильности реализации важно учитывать, что шифры простой замены (а, значит, и шифр Атбаш) относятся к симметричным шифрам. Причём шифр Атбаш, в отличие от уже разобранных шифра Цезаря, симметричен только с учётом идентичности алфавита при шифровании и дешифровании. Это важно при проверке правильности работы шифра, для чего изначальное сообщение мы пропускаем дважды через функцию, которая зашифровывает сообщение. Так мы должны получить шифрокод после первого запуска функции, и изначальное сообщение после второго запуска функции.

```

        result = IOBuffer()
        for ch in text
            reflected = reflect_russian(ch)
            shifted = shift_russian(reflected, k)
            print(result, shifted)
        end
        return String(take!(result))
    end

atbash_cipher (generic function with 2 methods)

```

```

julia> println(atbash_cipher("Молодец!", 2))
Хуцуэъл!

```

julia>

Результат работы кода представлен ниже (рис. 2).

```
julia> function atbash_cipher(text::String, k::Int=0)
    # В атбаш обычный ключ не используется, это фиксированное отражение
    # Но если нужен ключ k – реализуем модифицированный атбаш:
    # сначала отражаем буквы, затем сдвигаем результат на k

    function reflect_russian(ch)
        if 'А' <= ch <= 'Я'
            return Char(Int('А') + (Int('Я') - Int(ch)))
        elseif 'а' <= ch <= 'я'
            return Char(Int('а') + (Int('я') - Int(ch)))
        else
            return ch
        end
    end

    function shift_russian(ch, k)
        if 'А' <= ch <= 'Я'
            return Char(mod(Int(ch) - Int('А') + k, 32) + Int('А'))
        elseif 'а' <= ch <= 'я'
            return Char(mod(Int(ch) - Int('а') + k, 32) + Int('а'))
        else
            return ch
        end
    end

    result = IOBuffer()
    for ch in text
        reflected = reflect_russian(ch)
        shifted = shift_russian(reflected, k)
        print(result, shifted)
    end
    return String(take!(result))
end

atbash_cipher (generic function with 2 methods)

julia> println(atbash_cipher("Молодец!", 2))
Хуцуэъл!
```

Рис. 2: Результат работы шифра Атбаш

4 Выводы

В результате работы мы ознакомились с традиционными моноалфавитными шрифтами простой замены, а именно:

- Шифром Цезаря;
- Шифром Атбаш.

Также были записаны скринкасты:

- Выполнения лабораторной работы
- Запись создания отчёта
- Запись создания презентации
- Защита лабораторной работы

Список литературы

1. Лабораторная работа №1. Шифры простой замены [Электронный ресурс]. RUDN, 2024. URL:
https://esystem.rudn.ru/pluginfile.php/2368506/mod_folder/content/0/lab01.pdf.
2. Математика криптографии и теория шифрования [Электронный ресурс]. URL:
<https://intuit.ru/studies/courses/552/408/info>.
3. Julia 1.10 Documentation [Электронный ресурс]. 2024. URL:
<https://docs.julialang.org/en/v1/>.
4. Julia 1.10 Documentation [Электронный ресурс]. 2024. URL:
<https://docs.julialang.org/en/v1/base/strings/>.