

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение
высшего образования «Санкт-Петербургский политехнический
университет Петра Великого»

Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

Направление: 02.03.01 «Математика и компьютерные науки»

Отчет по лабораторной работе №3
по дисциплине
«Функциональное программирование»
Вариант 20

Студент,

группы 5130201/20102

_____ Тищенко А. А.

Преподаватель,

к. т. н., доц.

_____ Моторин Д. Е.

«_____» _____ 2024г.

Санкт-Петербург, 2024

Содержание

1	Постановка задачи	3
2	Математическое описание	4
2.1	Шифр Цезаря	4
3	Особенности реализации	5
3.1	Исходное изображение и текст	5
3.2	Кодирование и декодирование текста с помощью шифра Цезаря	6
3.3	Представление текста в виде последовательности бит	7
3.4	Работа с файлами	8
3.5	Сохранение зашифрованных данных в изображении	8
3.6	Чтение зашифрованных данных из изображения	9
4	Результаты работы программы	10
	Заключение	14
	Список литературы	15

1 Постановка задачи

Для выполнения лабораторной работы необходимо было сделать следующее. Создать проект в `stack`. Все чистые функции записать в библиотеку `Lib.hs` и ограничить доступ к вспомогательным функциям. Использовать `do`-нотацию для работы с внешними файлами. Найти портрет Дэвида Дойча. Перевести изображение в формат `.bmp` (24-разрядный). Сохранить в файл формата `.txt` фрагмент биографии (не менее 1000 символов без пробелов, текст не должен обрываться на середине слова или предложения). Закодировать текст в изображение шифром Цезаря (смещение задается пользователем). Ключ к шифру записывается в имя файла. Написать функцию расшифровывающую текст из изображения используя ключ из имени файла и сохраняющую результат в отдельный текстовый файл. Создать функции шифрующие текст в последний бит каждого байта, последние два бита каждого байта, ..., все биты в байте. В отчете привести примеры искажений изображения.

2 Математическое описание

2.1 Шифр Цезаря

Шифр Цезаря (лат. *Notae Caesarianae*), также известный как шифр сдвига или код Цезаря — разновидность шифра подстановки, в котором каждый символ в открытом тексте заменяется символом, находящимся на некотором постоянном числе позиций левее или правее него в алфавите (так, в шифре со сдвигом вправо на 3, А была бы заменена на Г, Б станет Д, и так далее). Шифр был назван в честь римского полководца Гая Юлия Цезаря, использовавшего его для секретной переписки со своими военачальниками.

Если сопоставить каждому символу алфавита его порядковый номер (нумеруя с 0), то шифрование и дешифрование можно выразить формулами модульной арифметики [1]:

$$\begin{aligned}y &= (x + k) \mod n \\x &= (y - k) \mod n\end{aligned}$$

где:

x — символ открытого текста,

y — символ шифрованного текста,

n — мощность алфавита,

k — ключ.

3 Особенности реализации

3.1 Исходное изображение и текст

Для выполнения лабораторной работы необходимо было найти изображение Дэвида Дойча (см. Рис. 1). Изображение было переведено из формата jpeg в формат bmp с помощью сайта [3].



Рис. 1. Изображение Дэвида Дойча, размещённое на его личном сайте [2].

Отрывок биографии Дэвида Дойча длиной в 1157 символов без учёта пробелов представлен ниже.

David Elieser Deutsch FRS (DOYTCH; born 18 May 1953) is a British physicist at the University of Oxford. He is a visiting professor in the Department of Atomic and Laser Physics at the Centre for Quantum Computation (CQC) in the Clarendon Laboratory of the University of Oxford. He pioneered the field of quantum computation by formulating a description for a quantum Turing machine, as well as specifying an algorithm designed to run on a quantum computer. He is a proponent of the many-worlds interpretation of quantum mechanics. Deutsch was born to a Jewish family in Haifa, Israel on 18 May 1953, the son of Oskar and Tikva Deutsch. In London, David attended Geneva House school in Cricklewood (his parents owned and ran the Alma restaurant on Cricklewood Broadway), followed by William Ellis School in Highgate before reading Natural Sciences at Clare College, Cambridge and taking Part III of the Mathematical Tripos. He went on to Wolfson College, Oxford for his doctorate in theoretical physics, about quantum field theory in curved space-time, supervised by Dennis Sciama and Philip Candelas. His work on quantum algorithms began with a 1985 paper, later

expanded in 1992 along with Richard Jozsa, to produce the DeutschJozsa algorithm, one of the first examples of a quantum algorithm that is exponentially faster than any possible deterministic classical algorithm.

3.2 Кодирование и декодирование текста с помощью шифра Цезаря

Код функций для кодирования и декодирования текста с помощью шифра Цезаря представлен в листинге 1. Функция `encryptCaesar` принимает алфавит в виде списка символов, смещение и сам текст, а возвращает зашифрованный текст. В её коде используется вспомогательная функция `indexOf`. Функция принимает список и элемент списка, а возвращает индекс этого элемента. Для создания алфавита используется функция `createAlphabetFromText`. Она принимает текст, а возвращает алфавит, который в нём используется, в виде списка символов. Для декодирования текста используется функция `decryptCaesar`, которая, по-сути, является лишь обёрткой над функцией `encryptCaesar`, так как процесс кодирования осуществляется почти так же как и декодирования. Функция `decryptCaesar` принимает на вход алфавит, смещение и закодированный текст, а возвращает декодированный текст. Алфавит сохраняется в отдельный файл и должен передаваться вместе с зашифрованным текстом, чтобы этот текст можно было дешифровать.

Листинг 1. Функции для кодирования и декодирования текста с помощью шифра Цезаря.

```
1 encryptCaesar :: [Char] -> Int -> String -> String
2 encryptCaesar alphabet shift text = map caesarChar text
3   where
4     caesarChar c = alphabet !! ((indexOf alphabet c + shift) `mod` length alphabet)
5
6 indexOf :: (Eq t) => [t] -> t -> Int
7 indexOf [] _ = -1
8 indexOf (x : xs) target
9   | x == target = 0
10  | otherwise = 1 + indexOf xs target
11
12 createAlphabetFromText :: String -> [Char]
13 createAlphabetFromText [] = []
14 createAlphabetFromText (x:xs)
15   | x `elem` alphabet = alphabet
16   | otherwise        = x : alphabet
17   where
18     alphabet = createAlphabetFromText xs
19
20 decryptCaesar :: [Char] -> Int -> String -> String
21 decryptCaesar alphabet shift =
22   encryptCaesar alphabet (alphabetLength - (shift `mod` alphabetLength))
23   where
24     alphabetLength = length alphabet
```

Пример закодированного с помощью шифра Цезаря текста биографии Дэвида Дойча для смещения 5 представлен ниже.

```

uiHF rWtFaga.ruabYgo;r3qkrNuCLS59Grcm.lrdwrvnrDJRBTfgrirM.FYFg;rs;ngFo
FgYriYrY;arAlFHa.gFYnrmerCdem. Ur9arFgrirHFgFYFlhrs.meaggm.rFlrY;aruasi.
YQalYrmerIYmQForil r)iga.r8;ngFogriYrY;ar5alY.arem.r(bilYbQr5mQsbYiYFmlr
N5(5TrFlrY;ar5ti.al mlr)icm.iYm.nrmerY;arAlFHa.gFYnrmerCdem. Ur9arsFmlaa
.a rY;areFat rmerpbilYbQromQsbYiYFmlrcnrem.QbtiYFlhrir ago.FsYFmlrem.rir
pbilYbQrSb.FlhrQio;Flayrigr,attrigrsaoFenFlhrilrithm.FY;Qr agFhla rYmr.
blrmlrirpbilYbQromQsbYa.Ur9arFgrirs.msmlalYrmerY;arQiln,m.t grFlYa.s.aYi
YFmlrmerpbilYbQrQao;ilFogUluabYgo;r,igrm.lYmrirxa,Fg;reiQFtnrFlr9iFeiy
r-g.iatrmlrDwrvnrDJRByrY;argmlrmerCg2i.ril rSF2HiruabYgo;Ur-lr)ml mlyru
iHF riYYal a rEalaHir9mbgargo;mmtrFlr5.Fo2ta,mm rN;Fgrsi.alYgrm,la ril r
.ilrY;arItQir.agYib.ilYrmlr5.Fo2ta,mm rM.mi ,inTyremttm,a rcnrPFttFiQrWt
tFgrko;mmtrFlr9Fh;hiYarcaem.ar.ai Flhr0iYb.itrkoFaloagriYr5ti.ar5mttahay
r5iQc.F haril rYi2Flhr8i.Yr--rmerY;arviY;aQiYFoitrS.FsmgUr9ar,alYrmlrYm
rPmtegmlr5mttahayrCdem. rem.r;Fgr moYm.iYarFlrY;am.aYFoitrS;ngFogyricmbY
rpbilYbQreFat rY;am.nrFlrob.Ha rgsioaYFQayrgbsa.HFga rcnrualFlgrkoFiQiri
l r8;FtFsr5il atigU19Fgr,m.2rmlrpbilYbQrithm.FY;Qgrcahllr,FY;rirDJwRrsis
a.yrtiYa.radsil a rFlrDJJzritmlhr,FY;rQFo;i. rxmfgiyrYmrs.m boarY;aruabY
go;xmfgirithm.FY;QyrmlarmerY;areF.gYradiQstagrmerirpbilYbQrithm.FY;QrY;i
YrFgradsmlalYfittnreigYa.rY;ilrilnrsmggFctar aYa.QFlFgYForotiggFoitrithm
.FY;QU

```

3.3 Представление текста в виде последовательности бит

Код функций для преобразования текста в последовательность бит и обратно представлен в листинге 2. Функция `textToBits` принимает текст в виде строки и возвращает его представление в виде вектора бит. Она использует вспомогательную функцию `charToBits`, которая преобразует символ в список бит, представляющих его код ASCII в двоичном виде. Для преобразования последовательности бит обратно в текст используется функция `bitsToText`. Она рекурсивно делит вектор бит на блоки по 8 бит, преобразует каждый блок в символ ASCII и объединяет их в строку. В процессе этого преобразования используется функция `bitsToInt`, которая преобразует вектор бит в целое число, интерпретируя их как двоичное представление этого числа.

Листинг 2. Функции для конвертации текста в последовательность бит и обратно.

```

1 textToBits :: String -> VU.Vector Int
2 textToBits text = VU.fromList $ concatMap charToBits text
3
4 charToBits :: Char -> [Int]
5 charToBits c = [if testBit (ord c) i then 1 else 0 | i <- [7,6..0]]
6
7 bitsToText :: VU.Vector Int -> String
8 bitsToText bits
9   | VU.null bits = []
10  | otherwise = (chr $ bitsToInt (VU.take 8 bits)) : bitsToText (VU.drop 8 bits)
11
12 bitsToInt :: VU.Vector Int -> Int
13 bitsToInt bits =

```

```

14     sum [bit * (2 ^ index) | (bit, index) <- zip (VU.toList bits) [len,(len - 1)..0]]
15     where
16         len = VU.length bits - 1

```

3.4 Работа с файлами

Для работы с текстовыми файлами использовались базовые функции Haskell – `readFile` (читает содержимое файла и возвращает его как строку) и `writeFile` (записывает строку в файл, заменяя его содержимое).

Для работы с изображениями использовалась библиотека `JuicyPixels` [4]. С её помощью можно как прочесть изображение в любом популярном формате, так и сохранить его. В частности в работе использовались функции: `readImage` – для чтения изображения из указанного файла, `saveBmpImage` – для сохранения изображения в формате bmp.

3.5 Сохранение зашифрованных данных в изображении

Код функций для создания изображения с закодированными данными представлен в листинге 3. Функция `encodePixel` отвечает за кодирование последовательности бит в определённый пиксель изображения. Она принимает количество бит данных, которое будет сохранено в каждый байт изображения, исходное изображение, вектор бит зашифрованных данных, координаты пикселя (x, y) и возвращает новый пиксель с закодированными данными. Для этого функция вычисляет индекс пикселя в изображении, извлекает соответствующую часть вектора бит данных, преобразует её в целые числа, накладывает битовую маску, которая соответствует количеству изменяемых бит в байте, и записывает закодированные данные. Для создания маски используется вспомогательная функция `createMask`.

Функция `encodePixel` затем используется вместе с функцией `generateImage` из библиотеки `JuicyPixels` для генерации нового изображения.

При сохранении изображения в файл, в его названии сохраняется смещение шифра Цезаря и количество бит в байте, отведённых для хранения зашифрованных данных. Например, название изображения `david_2_10.bmp` означает, что при кодировании использовался код Цезаря со смещением 10, а для хранения закодированных данных в каждом байте изображения использовалось 2 бита.

Листинг 3. Функции для создания изображения с закодированными данными.

```

1  createMask :: Int -> Int
2  createMask shift = shiftL (complement 0) shift .&. complement 0
3
4  encodePixel :: Int -> Image PixelRGB8 -> VU.Vector Int -> Int -> Int -> PixelRGB8
5  encodePixel bitsPerByte img bits x y = PixelRGB8 newR newG newB
6  where
7      width = imageWidth img
8
9      index = x + y * width

```



```

10     startPos = index * 3 * bitsPerByte
11     pixelBits = VU.slice startPos (3 * bitsPerByte) bits
12
13     bitsIntR = bitsToInt $ VU.slice 0 bitsPerByte pixelBits
14     bitsIntG = bitsToInt $ VU.slice bitsPerByte bitsPerByte pixelBits
15     bitsIntB = bitsToInt $ VU.slice (2 * bitsPerByte) bitsPerByte pixelBits
16
17     mask = createMask bitsPerByte
18
19     PixelRGB8 r g b = pixelAt img x y
20     newR = intToWord8 $ ((word8ToInt r) .&. mask) .|. bitsIntR
21     newG = intToWord8 $ ((word8ToInt g) .&. mask) .|. bitsIntG
22     newB = intToWord8 $ ((word8ToInt b) .&. mask) .|. bitsIntB

```

3.6 Чтение зашифрованных данных из изображения

Код функций для чтения зашифрованных данных из изображения представлен в листинге 4. Функция `extractBits` извлекает заданное количество бит из одного байта пикселя. Она принимает число бит на байт и байт пикселя, возвращая список бит. Функция `extractBitsFromPixel` предназначена для извлечения бит из всех трёх цветовых каналов (R, G, B) пикселя. Она объединяет списки бит из каждого канала в один общий список. Для извлечения бит из всего изображения используется функция `extractBitsFromImage`. Она последовательно обрабатывает все пиксели изображения, извлекая биты с помощью `extractBitsFromPixel`, и объединяет их в общий список.

Функция `extractShift` извлекает смещения для шифра Цезаря из названия файла изображения.

Листинг 4. Функции для чтения зашифрованных данных из изображения.

```

1  extractBits :: Int -> Pixel8 -> [Int]
2  extractBits bitsPerByte pixelByte =
3    [ if testBit pixelByte i then 1 else 0 | i <- [bitsPerByte-1, bitsPerByte-2..0] ]
4
5  extractBitsFromPixel :: Int -> PixelRGB8 -> [Int]
6  extractBitsFromPixel bitsPerByte (PixelRGB8 r g b) =
7    let bitsR = extractBits bitsPerByte r
8        bitsG = extractBits bitsPerByte g
9        bitsB = extractBits bitsPerByte b
10   in bitsR ++ bitsG ++ bitsB
11
12  extractBitsFromImage :: Int -> Image PixelRGB8 -> [Int]
13  extractBitsFromImage bitsPerByte img =
14    let width = imageWidth img
15        height = imageHeight img
16        pixels = [ pixelAt img x y | y <- [0..height - 1], x <- [0..width - 1] ]
17   in concatMap (extractBitsFromPixel bitsPerByte) pixels
18
19  extractShift :: String -> Maybe Int
20  extractShift path =
21    let shift = takeWhile ('0'..'9') (reverse $ takeWhile (/= '_') (reverse path))
22   in readMaybe shift

```

4 Результаты работы программы

При успешном завершении программа создаёт четыре файла: файл изображения с закодированным текстом, текстовый файл с закодированным текстом, текстовый файл с алфавитом и текстовый файл с декодированным текстом.

```
Введите значение сдвига для шифра Цезаря:
10
Введите количество бит для кодирования:
1

Чтение текста из файла "resources/biography.txt"
10 символов текста: "David Elie"

Шифрование текста
Размер алфавита: 61
Алфавит сохранён в файл "tmp/alphabet.txt"
10 символов шифра: "bF93r.PY3i"
Зашифрованный текст сохранён в файл "tmp/biography_encrypted.txt"
10 битов шифра: "[0,1,1,0,0,0,1,0,0,1]"

Кодирование текста в изображение
Изображение сохранено по пути: "tmp/david_1_10.bmp"

Чтение алфавита из файла "tmp/alphabet.txt"

Декодирование текста из изображения
Из названия файла извлечён ключ: 10
10 битов шифра: "[0,1,1,0,0,0,1,0,0,1]"
10 символов шифра: "bF93r.PY3i"
10 символов текста: "David Elie"
Текст сохранён по пути: "tmp/biography.txt"
```

Рис. 2. Результаты работы программы в консоли.

На Рис. 2 представлены результаты работы программы в консоли.



Рис. 3. Изображение с зашифрованными данными (1 бит).



Рис. 4. Изображение с зашифрованными данными (2 бит).



Рис. 5. Изображение с зашифрованными данными (3 бит).



Рис. 6. Изображение с зашифрованными данными (4 бит).



Рис. 7. Изображение с зашифрованными данными (5 бит).



Рис. 8. Изображение с зашифрованными данными (6 бит).



Рис. 9. Изображение с зашифрованными данными (7 бит).

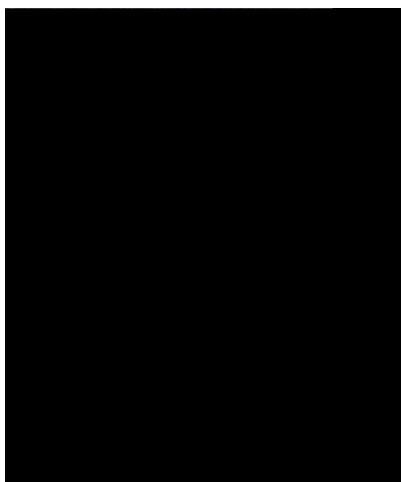


Рис. 10. Изображение с зашифрованными данными (8 бит).

На Рис. 3-10 представлены результирующие изображения с разным количеством бит, отведённых под зашифрованные данные.

Заключение

В результате выполнения лабораторной работы была создана программа на языке Haskell, которая способна кодировать текстовые данные из указанного файла с помощью шифра Цезаря и сохранять эти данные внутри изображения. Причём программа позволяет выбрать как смещение для шифра Цезаря, так и количество бит, которое будет использовано в каждом байте изображения для хранения данных.

Список литературы

- [1] Luciano, D., Prichett, G., Cryptology: From Caesar Ciphers to Public-Key Cryptosystems, The College Mathematics Journal, 1987.
- [2] David Deutsch – personal website, URL: <https://www.daviddeutsch.org.uk/>, Дата обращения: 19.11.2024
- [3] Convertio – BPM to JPG online converter, URL: <https://convertio.co/ru/bmp-jpg/>, Дата обращения: 19.11.2024
- [4] Hackage – JuicyPixels: Picture loading/serialization, URL: <https://hackage.haskell.org/package/JuicyPixels>, Дата обращения: 19.11.2024