

Санкт-Петербургский Национальный Исследовательский
Университет Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий

Лабораторная работа №6 "Хэширование"
Вариант 2

Выполнили:
Голованов Д.И.,
Шарыпов Е.А.,
Стафеев И.А.
Проверил
Мусаев А.А.

Санкт-Петербург,
2024

СОДЕРЖАНИЕ

Стр.

ВВЕДЕНИЕ	3
1 Задача 1	4
2 Задача 2	5
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	7

ВВЕДЕНИЕ

В данной лабораторной работе необходимо решить следующие задачи:

1. Реализовать алгоритм хэширования по методу умножения и проверить работоспособность на основе строки, вводимой пользователь;
2. Реализовать алгоритм нахождения контрольной суммы CRC-16, и проверить работоспособность на основе строки, вводимой пользователь.

1 Задача 1

Код алгоритма для хэширования через умножения представлен на рисунке 1.1. В качестве константы выбран золотое сечение, поскольку оно обеспечивает более-менее равномерное распределение хэш-кодов [1].

```
import math

def mult_hash(text, m=2**10):
    """Хэширование через умножение"""
    const = (math.sqrt(5) - 1) / 2
    codes = [int(m * ((ord(letter) * const) % 1)) for letter in text]
    return codes
```

Рисунок 1.1 — Алгоритм хэширования через умножение

На рисунке 1.2 представлен результат выполнения функции для исходной строки *hello world!* и $m = 2^{10}$.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\ProgrammingProjects\itmo_algos_labs> & "D:/Program Files/Python37/Python.exe" D:/ProgrammingProjects/itmo_algos_labs/mult_hash.py
Введите строку для хэширования:
hello world!
[282, 431, 765, 765, 616, 795, 559, 616, 466, 765, 822, 404]
PS D:\ProgrammingProjects\itmo_algos_labs> █
```

Рисунок 1.2 — Результат выполнения функции хэширования

2 Задача 2

Код алгоритма вычисления контрольной суммы CRC-16 представлен на рисунке 2.1.

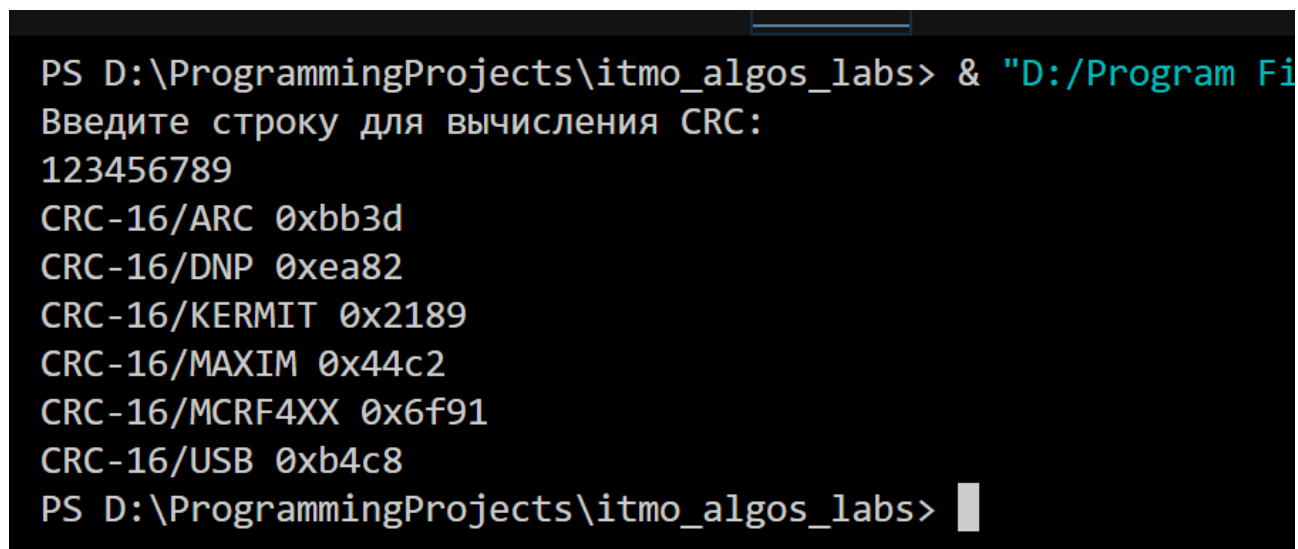
```
12 def reverse_poly(poly):
13     return int(f"{poly:016b}"[::-1], 2)
14
15
16 def crc_16(text, start_crc, polynom, xor_out):
17     """Хэширование методом CRC-16 [c] LSB"""
18     data = bytearray(bytes(text, "utf-8"))
19     crc = reverse_poly(start_crc)
20     polynom = reverse_poly(polynom)
21     for elem in data:
22         crc ^= elem
23         for _ in range(8):
24             if crc & 0x0001:
25                 crc = (crc >> 1) ^ polynom
26             else:
27                 crc = (crc >> 1)
28     return crc ^ xor_out
29
```

Рисунок 2.1 — Алгоритм вычисления контрольной суммы CRC-16

Функция *reverse_poly* переворачивает бинарное число с добавленными слева нулями (чтобы длина была равна 16). Эта функция используется в самом алгоритме нахождения контрольной суммы.

Здесь реализован вариант алгоритма, который вычисляет контрольную сумму, начиная с младшего бита, поэтому в начале стартовое значения *crc* и производящий полином необходимо перевернуть. В цикле находится хог с текущим битовым значением символа заданной строки, затем во вложенном цикле для младших 8 бит происходит либо сдвиг, либо сдвиг и хог с производящим полиномом.

На рисунке 2.2 представлен результат выполнения программы для строки *123456789* в нескольких реализациях CRC-16, в которых меняются стартовое значение, полином и XorOut. Вариации алгоритма взяты с Википедии [2] и там же сравнены результаты выполнения с написанной программой.



```
PS D:\ProgrammingProjects\itmo_algos_labs> & "D:/Program Fi
Введите строку для вычисления CRC:
123456789
CRC-16/ARC 0xbb3d
CRC-16/DNP 0xea82
CRC-16/KERMIT 0x2189
CRC-16/MAXIM 0x44c2
CRC-16/MCRF4XX 0x6f91
CRC-16/USB 0xb4c8
PS D:\ProgrammingProjects\itmo_algos_labs>
```

Рисунок 2.2 — Результат нахождения контрольной суммы

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Хэш-функции. Универсальное хэширование [Электронный ресурс]: [сайт]. - URL: https://proproprogs.ru/structure_data/std-hash-funkcii-universalnoe-heshirovanie (дата обращения: 22.03.2024).
2. Википедия. Циклический избыточный код [Электронный ресурс]: [сайт]. - URL: https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D0%BA%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%B7%D0%B1%D1%8B%D1%82%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4 (дата обращения: 22.03.2024).