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

Факультет Инфокумманикационных технологий

**Лабораторная работа №2**

**Вариант № 6**

**Умножение**

**MD5**

Выполнили:

Колтунова П.В.

Морозова Я.А.

Цветкова Н.В.

Проверил:

Мусаев А.А.

Санкт-Петербург

2023

**ВВЕДЕНИЕ**

Целью данной работы является ознакомление с алгоритмами хэширования. Необходимо изучить методы хэширования: умножение и MD5.

# ГЛАВА 1

## Задание 1

Осуществить хэширование строки методом умножения. Используя выражение *n\*((k\*0,668) mod 1),* где *n* – длинная массива, *k* – ключ для конкретного элемента строки, *0,668* – константа из интервала [0…1].

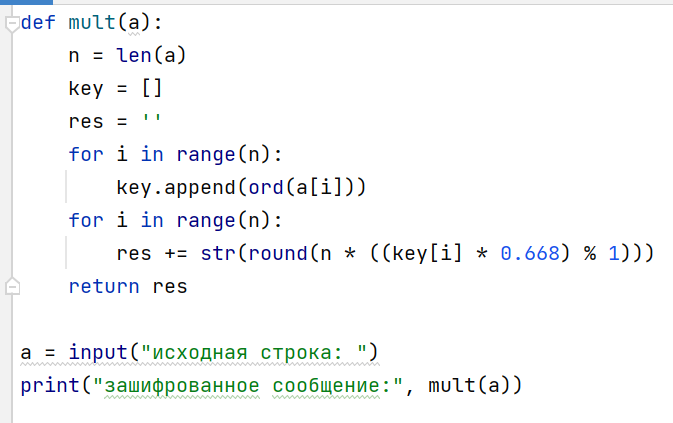


Рисунок 1- Хэширование методом умножения

Вывод программы:

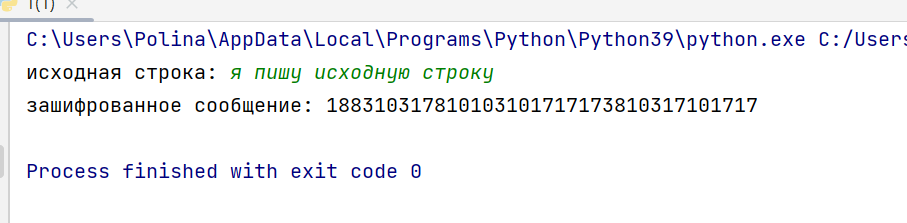


Рисунок 2 - Вывод хэширование методом умножения

## Задание 2

Осуществить хэширование строки посредством метода MD5.

Для работы программы необходимо импортировать модуль math. Ввод исходной строки text осуществляется пользователем.

Для работы со строкой необходимо преобразовать её в бинарную последовательность. Для этого каждый символ строки заменяется на его числовое значение с помощью функции ord(), после чего данное числовое значение переводится в двоичную систему счисления с помощью функции bin(). Полученная строка сохраняется в переменной text\_dv.

Далее происходит выравнивание последовательности. Сохраняется в переменной iznach\_dlina значение длины исходной строки. В конец строки text\_dv добавляется строка “1”, после чего к строке добавляются “0” до тех пор, пока длина числовой последовательности не будет иметь вид: длина = (n \* 512) - 448, где n - наименьшее подходящее натуральное число. Полученная строка записывается в переменную text\_vir.

# Далее проверяется значение длины исходной строки iznach\_dlina - если оно больше или равно 2^64, то берётся его остаток от деления на 2^64. Данное значение переводиться в двоичную СС и приводится к 64-значному виду. Полученная строка добавляется в конец последовательности text\_vir. Итоговая строка записывается в переменную text\_poln.

Далее создается список buffer с буферными переменными:

A: 01 23 45 67

B: 89 ab cd ef

C: fe dc ba 98

D: 76 54 32 10

Каждая переменная переводится в двоичный вид.

Создается список сдвигов S.

Далее объявляются четыре логических функции:F(X,Y,Z) = XY v not(X) Z; G(X,Y,Z) = XZ v Y not(Z); H(X,Y,Z) = X xor Y xor Z; I(X,Y,Z) = Y xor (X v not(Z)). Дополнительно создаётся функция vibor(x, y, z, vibor\_num), которая по входящему значению vibr\_num определяет, какая логическая функция должна использоваться в конкретном этапе вычислений.

Далее создаётся массив из значений T[i]=4,294,967,296\*abs(sin(i)) для всех i = 0..64.

Далее строка text\_poln разбивается на несколько блоков по 512 бит. Каждый блок разбивается на 16 блоков по 32 бита. Инициализируется переменная i\_T, являющаяся счётчиком для массива T, и переменная it, являющаяся счётчиком для списка S.

Для каждого 32-битного блока объявляются четыре переменных, копирующих значение переменных списка buffer. По формуле В = B + (((A + vibor(B,C,D,i) + X[k] + T[i]) << S[it]) в двоичном формате происходит вычисление и перезапись переменной “b”. Этот процесс повторяется четыре раза для четырёх разных логических функций. Полученные значения a, b, c, d суммируются с соответствующими переменными списка buffer и перезаписываются в него.

По завершении вычислений переменные списка buffer переводятся в 16-тиричную СС, после чего последовательно добавляются в строку otvet. Строка otvet является результатом работы программы.

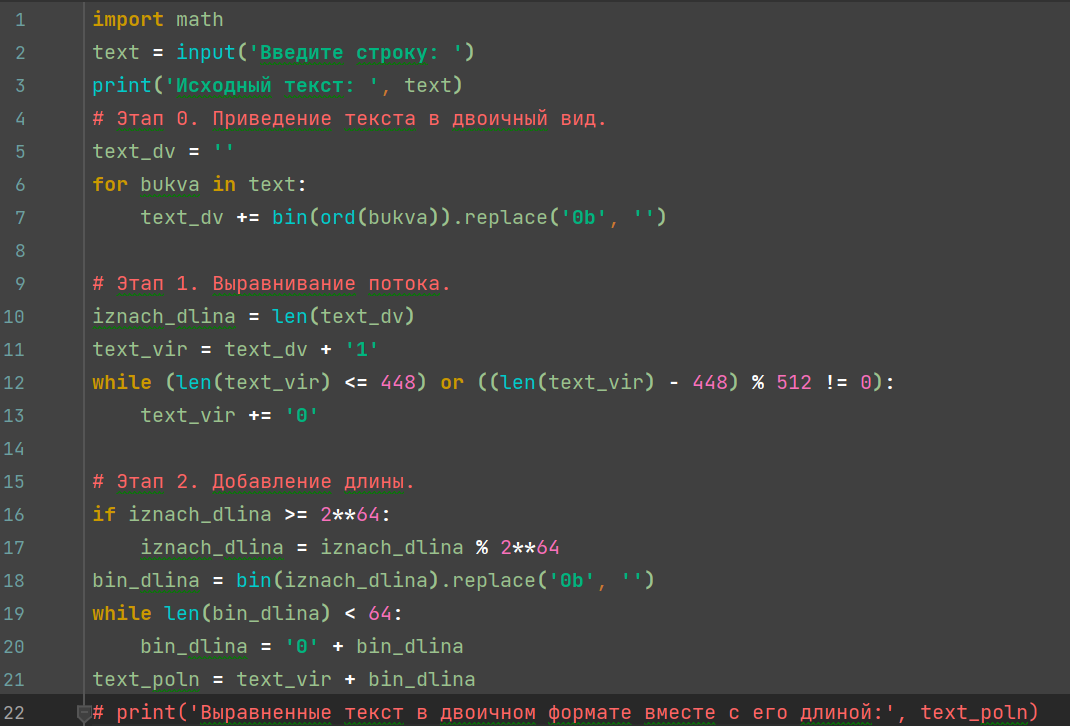


Рисунок 3 - Реализация ввода строки, преобразования в двоичную СС, выравнивания потока и добавления длины.

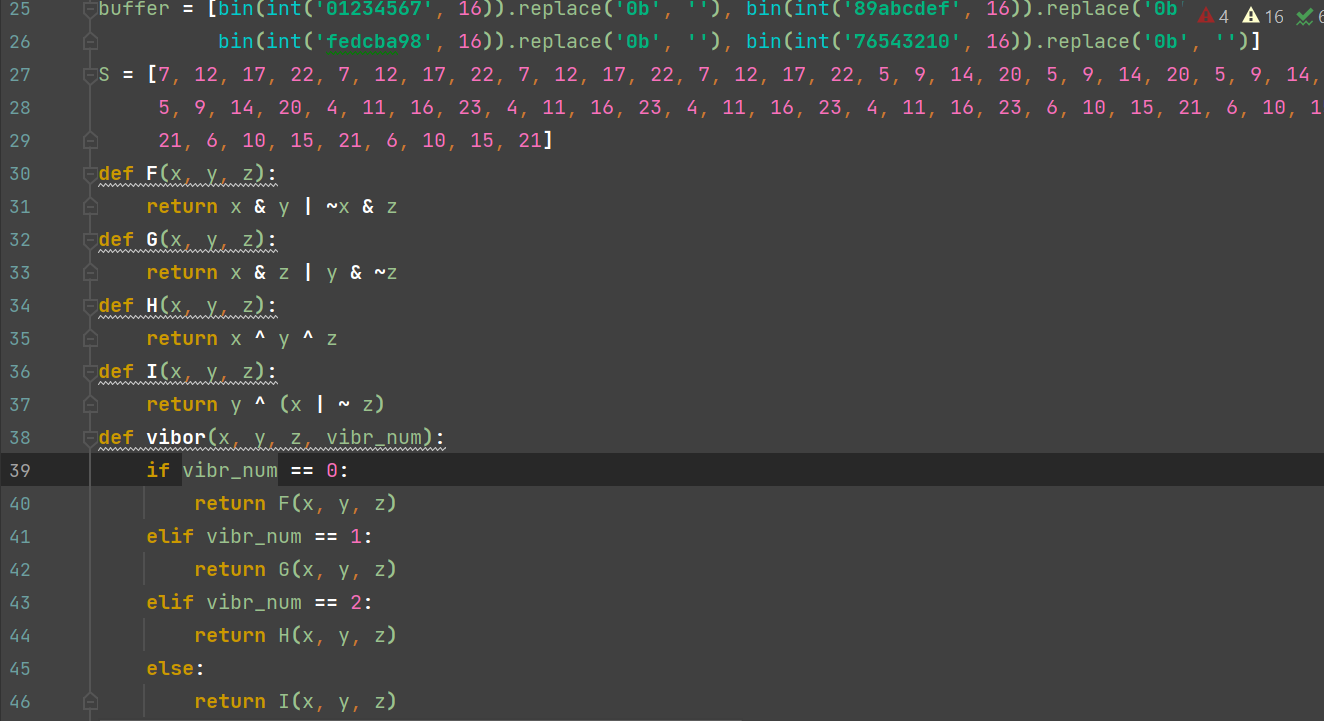


Рисунок 4 - Инициализация буферных переменных, списка сдвигов и логических функций.

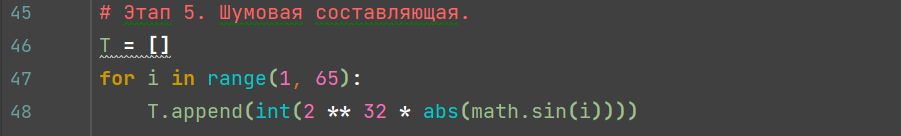


Рисунок 5 - Реализация шумовой составляющей.

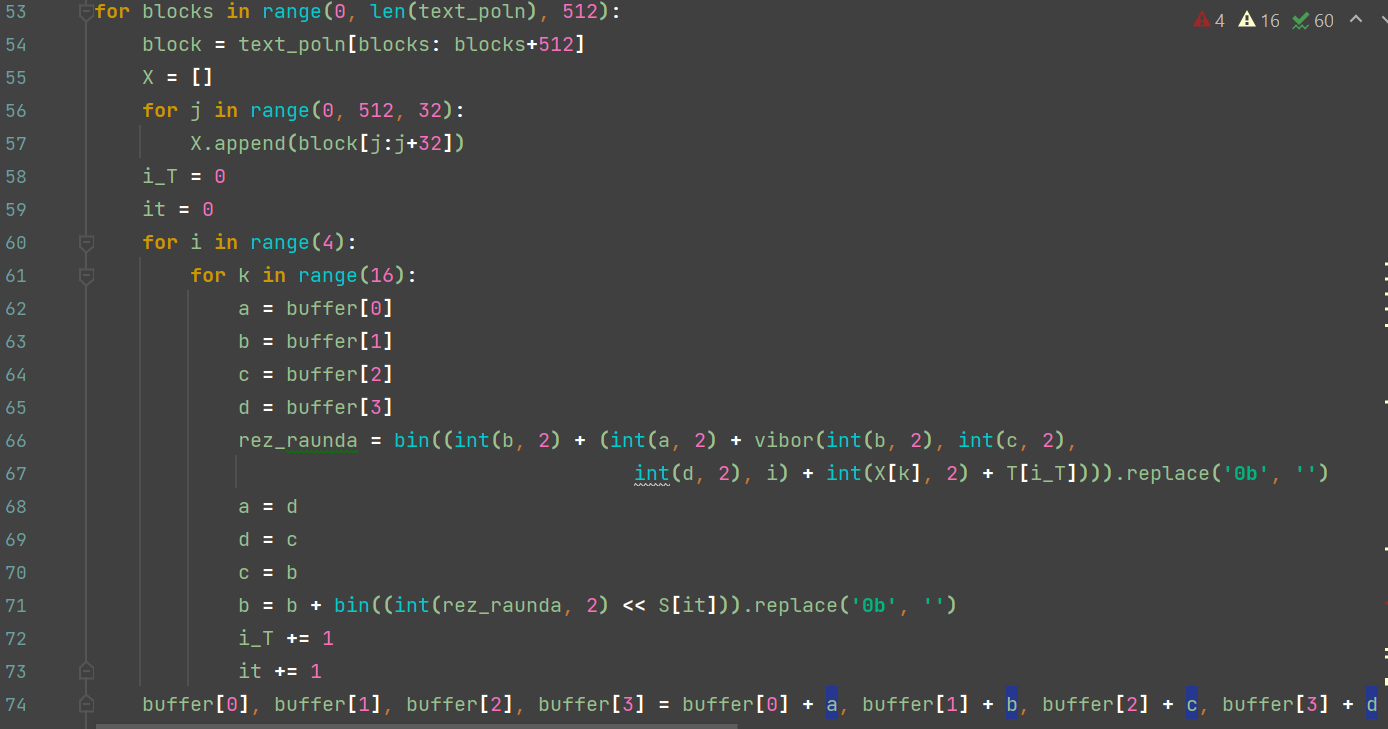
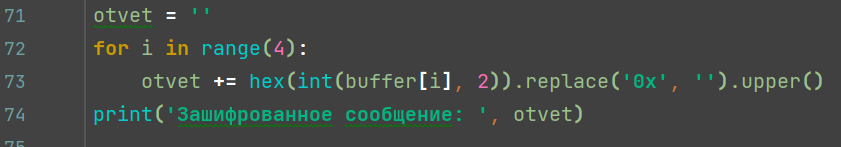


Рисунок 6 - Реализация блока вычислений хэша.



Изображение 1 - Реализация вывода ответа.

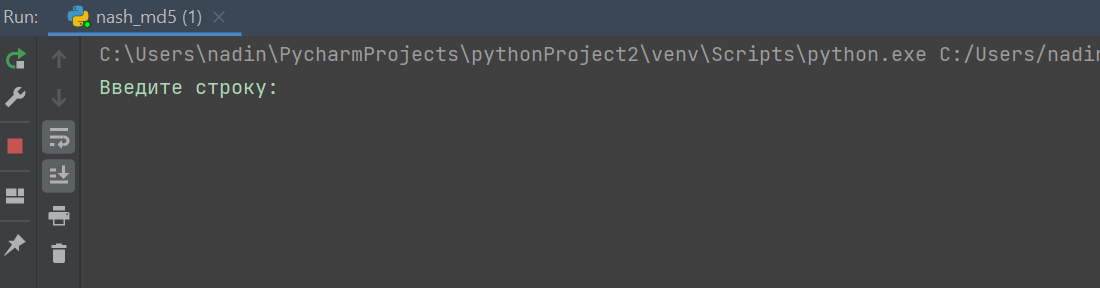


Рисунок 7 - Интерфейс при вызове программы.

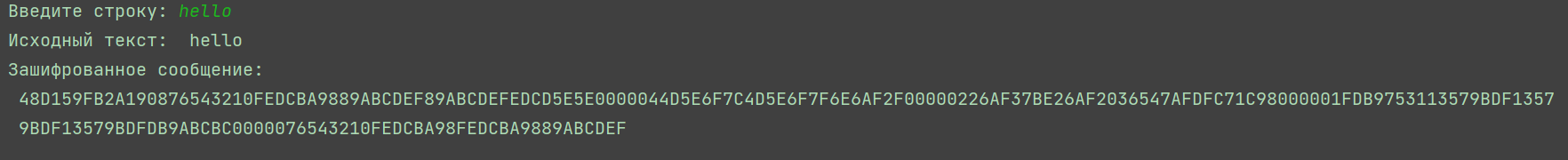


Рисунок 8 - Результат выполнения программы.

**ЗАКЛЮЧЕНИЕ**

Цель была достигнута. Изучены методы хэширования: умножение и MD5.

# СПИСОК ЛИТЕРАТУРЫ

1. Github. [Электронный ресурс] - URL: https://github.com/nadintsvetkova/algoritmi\_str\_dannih.git (Дата последнего обращения — 06.03.2023)