|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |  |
|  | |  |
| Кафедра инструментального и прикладного программного обеспечения | |  |
|  |  | |
|  |  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 8.1** | |
| **по дисциплине** | |
| **«**Графы: создание, алгоритмы обхода**»**  **Тема: «Алгоритмы кодирования и сжатия данных»** | |
|  | |
| Выполнил студент группы ИКБО-10-23 | Харитонов А.Н. |
| Принял преподаватель | Макеева О.В. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_ \_ \_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2024

# **Цель работы**

Разработать решения задач с использованием заданных алгоритмов сжатия и кодирования.

# **Ход работы**

**Задание 1.1**

Алгоритм Шеннона-Фано основан на частоте символов, встречающихся в фразе. Более частым символам присваиваются более короткие коды, более редким — более длинные.

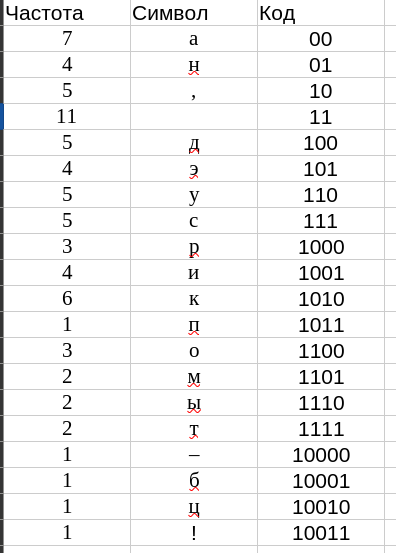


Рисунок 1.

Итоговая фраза: 0001001011100101110111101110001001101010011011101100101010011011100110010001101111011101011001000110111101110101100011111111110011111001010100110111001011101111110010111011111101000010010010111011111100001110001001001010011

Для восстановления текста необходимо посимвольно сравнивать закодированную строку с кодами до нахождения совпадения. В случае совпадения обнулять буфер сравнения, а найденное совпадение сохранять.

**Задание 1.2**

LZ77 — это алгоритм сжатия данных, который использует скользящее окно. Он ищет повторы подстрок и заменяет их на ссылки.

Алгоритм использует тройки (смещение, длина, следующий символ) для кодирования данных:

* **Смещение** — указывает на положение совпадения в окне поиска. (сколько символов надо перекинуть, чтобы найти исходную подстроку)
* **Длина** — длина совпадающей подстроки.
* **Следующий символ** — первый символ, не входящий в совпадающую подстроку.

Исходная строка: 0001010010101001101.

Рисунок 2

Закодированная строка:

**(0, 0, 0) (0, 0, 0) (2, 1, 1) (3, 1, 0) (2, 2, 0) (3, 1, 1) (3, 1, 0) (2, 2, 1) (5, 1, 1)**

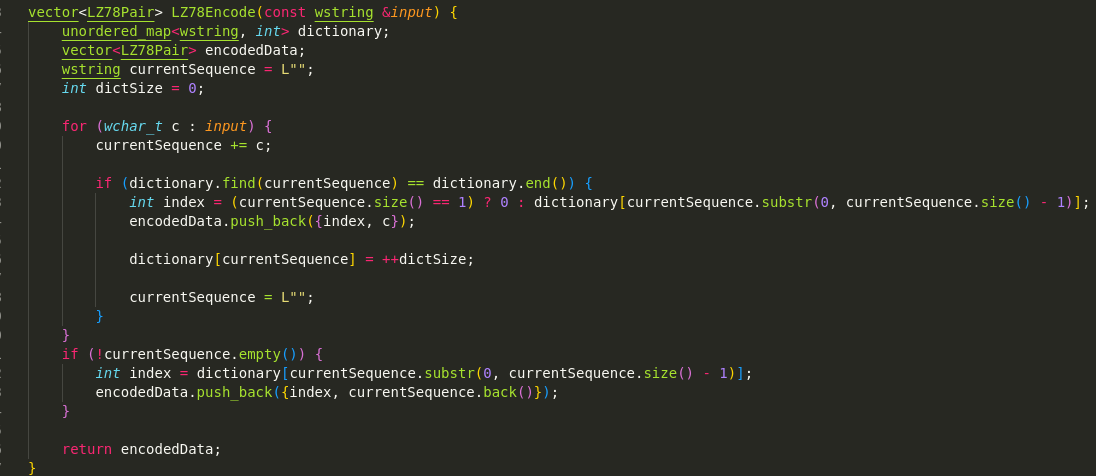
**Задание 1.3**

Алгоритм LZ78 кодирует строки, представляя их в виде пар (индекс, символ), где индекс указывает на позицию предыдущего совпавшего подстрока, а символ — это первый символ, который не был найден в предыдущем коде.

Закодируем строку **"кукуркукурекурекун"** с помощью LZ78:

1. Инициализируем пустой словарь и начинаем считывать строку символ за символом.
2. Каждый раз, когда не удается найти текущую последовательность символов в словаре, записываем пару (индекс последнего найденного элемента, новый символ) и добавляем последовательность в словарь.

Для этого была написана функция LZ78Encode.

Рисунок 3

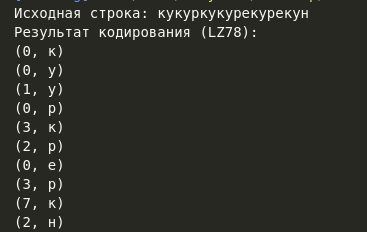
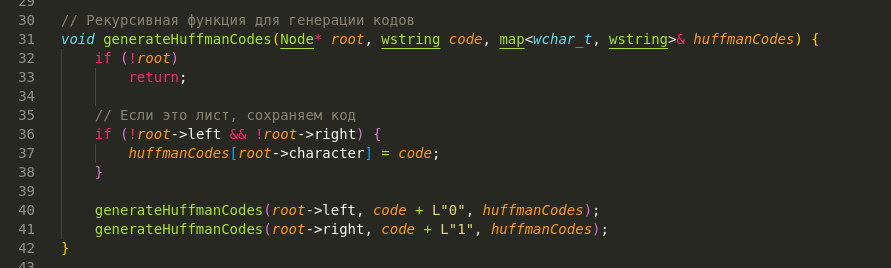


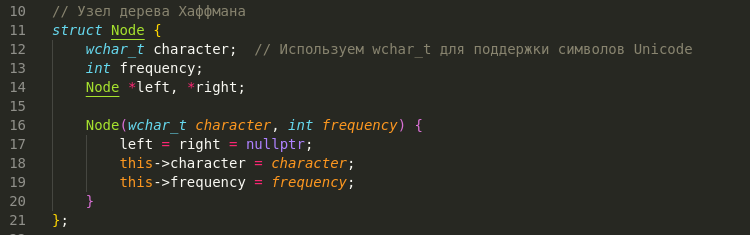
Рисунок 4

**Задание 2**

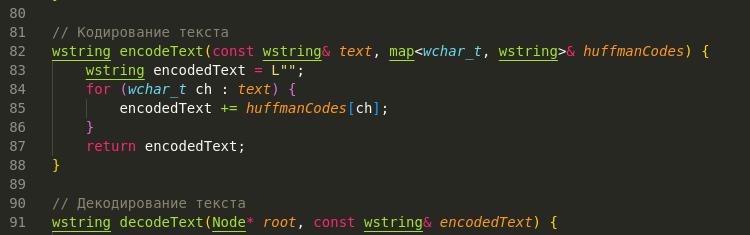
Для выполнения первого пункта была написана функция buildHuffmanTree.



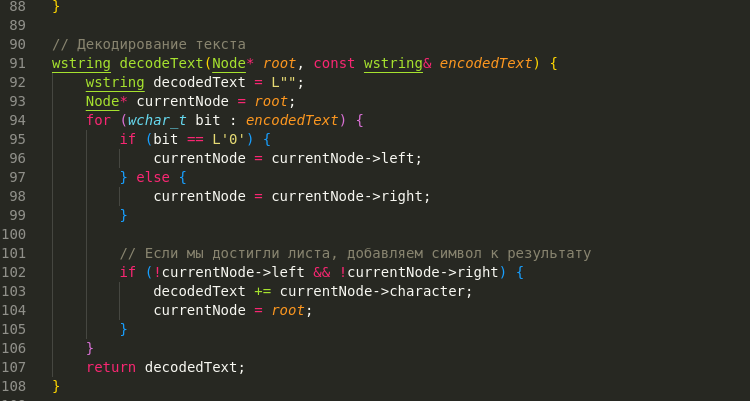
Рисунок 5

Рисунок 6

Для решения 2 подпункта была написана функция encodeText

Рисунок 7

Для декодирования: decodeText

Рисунок 8

**3. Вывод**

В ходе работы я составил использовал принципы динамического программирования для решения поставленной задачи.