|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | | |  |
|  | Институт информационных технологий (ИТ) | |
|  | Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ) | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**  **«Отдельные вопросы алгоритмизации»** | | | |  |
| **по дисциплине «Структуры и алгоритмы обработки данных (часть 2/2)»** | | | |  |
|  | | | |  |
| Выполнил студент группы ИКБО-41-23 | | Попов А.В. | |  |
|  | |  | |  |
| Принял  *Ассистент* | | Рысин М.Л. | |  |
| Практические работы выполнены | «\_\_\_»\_\_\_\_\_\_\_2024 г. | | (подпись студента) | |
| «Зачтено» | «\_\_\_»\_\_\_\_\_\_\_2024 г. | | (подпись преподавателя) | |
|  |  | |  | |

СОДЕРЖАНИЕ

[1 ЦЕЛЬ РАБОТЫ 3](#__RefHeading___Toc815_1995867043)

[2 ПОСТАНОВКА ЗАДАЧИ 4](#__RefHeading___Toc29578_1445094056)

[3 ЗАДАНИЕ 1.1 6](#__RefHeading___Toc29580_1445094056)

[3.1 Алгоритм решения 6](#__RefHeading___Toc29582_1445094056)

[3.2 Реализация задачи и тестирование 9](#__RefHeading___Toc29584_1445094056)

[4 ЗАДАНИЕ 1.2 12](#__RefHeading___Toc29586_1445094056)

[4.1 Алгоритм решения 12](#__RefHeading___Toc29588_1445094056)

[4.2 Реализация задачи и тестирование 14](#__RefHeading___Toc29590_1445094056)

[5 ЗАДАНИЕ 1.3 16](#__RefHeading___Toc29592_1445094056)

[5.1 Алгоритм решения 16](#__RefHeading___Toc29594_1445094056)

[5.2 Реализация задачи и тестирование 18](#__RefHeading___Toc29596_1445094056)

[6 ЗАДАНИЕ 2.1 19](#__RefHeading___Toc29598_1445094056)

[6.1 Требования к выполнению задания 19](#__RefHeading___Toc29600_1445094056)

[6.2 Реализация задачи и тестирование 19](#__RefHeading___Toc29602_1445094056)

[6.3 Анализ коэффициента сжатия 23](#__RefHeading___Toc29604_1445094056)

[7 ЗАДАНИЕ 2.2 24](#__RefHeading___Toc29606_1445094056)

[7.1 Требования к выполнению задания 24](#__RefHeading___Toc29608_1445094056)

[7.2 Алгоритм решения 24](#__RefHeading___Toc29610_1445094056)

[7.3 Реализация задачи 27](#__RefHeading___Toc29612_1445094056)

[8 ЗАДАНИЕ 2.3 31](#__RefHeading___Toc29614_1445094056)

[8.1 Требования к выполнению задания 31](#__RefHeading___Toc29616_1445094056)

[8.2 Реализация задачи 31](#__RefHeading___Toc29618_1445094056)

[8.3 Анализ коэффициента сжатия 32](#__RefHeading___Toc29620_1445094056)

[9 ВЫВОД 34](#__RefHeading___Toc29622_1445094056)

# 1 ЦЕЛЬ РАБОТЫ

Получение практических навыков и знаний по выполнению сжатия данных рассматриваемыми методами.

# 2 ПОСТАНОВКА ЗАДАЧИ

Задание 1 Исследование алгоритмов сжатия на примерах

1) Выполнить каждую задачу варианта, представив алгоритм решения в

виде таблицы и указав результат сжатия. Примеры оформления решения

представлены в Приложении1 этого документа.

2) Описать процесс восстановления сжатого текста.

3) Сформировать отчет, включив задание, вариант задания, результаты

выполнения задания варианта.

Задание 2 Разработать программы сжатия и восстановления текста методами Хаффмана и Шеннона – Фано.

1) Реализовать и отладить программы.

2) Сформировать отчет по разработке каждой программы в соответствии с требованиями.

- По методу Шеннона-Фано привести: постановку задачи, описать алгоритм формирования префиксного дерева и алгоритм кодирования, декодирования, код и результаты тестирования. Рассчитать

коэффициент сжатия. Сравнить с результат сжатия вашим алгоритмом с

результатом любого архиватора.

- по методу Хаффмана выполнить и отобразить результаты выполнения всех требований, предъявленных в задании и оформить разработку программы: постановка, подход к решению, код, результаты тестирования.

Вариант №17. Условие задания:

|  |  |  |
| --- | --- | --- |
| Закодировать фразу методами Шеннона– Фано | Сжатие данных по методу Лемпеля–Зива LZ77 Используя двух символьный алфавит (0, 1) закодировать следующую фразу | Закодировать следующую фразу, используя код LZ78 |
| Плыл по морю чемодан, В чемодане был диван, На диване ехал слон. Кто не верит – выйди вон! | 0001000010101001101 | webwerbweberweberweb |

# 3 ЗАДАНИЕ 1.1

## 3.1 Алгоритм решения

Метод Шеннона-Фано — это алгоритм сжатия данных, который используется для кодирования символов на основе их частоты. Суть метода заключается в том, что каждый символ получает уникальный код, и чем чаще символ встречается в тексте, тем короче будет его код.

Процесс работает следующим образом: сначала для каждого символа вычисляется его частота появления в тексте, затем символы сортируются по убыванию частоты. После этого весь список символов делится на две группы таким образом, чтобы суммы частот в этих группах были как можно более равными. Этот процесс деления повторяется рекурсивно, пока не останется по одному символу в каждой группе. Символы из левой группы получают код с префиксом "0", а символы из правой группы — с префиксом "1".

После того как процесс завершен, получается кодировка для каждого символа, которая используется для сжатия данных.

Процесс восстановления сжатого текста методом Шеннона–Фано заключается в использовании ранее созданного словаря кодов для преобразования закодированной последовательности обратно в исходный текст. Каждый символ в словаре имеет уникальный префиксный код, поэтому декодирование выполняется посимвольно. Этот метод восстановления гарантирует точное восстановление текста, так как коды уникальны и не пересекаются.

Для варианта №17 кодировка фразы «Плыл по морю чемодан, В чемодане был диван, На диване ехал слон. Кто не верит – выйди вон!» представлена в Таблице 1.

| **Символ** | **Кол-во** | **1-я цифра** | **2-я цифра** | **3-я цифра** | **4-я цифра** | **5-я цифра** | **6-я цифра** | **7-я цифра** | **Код** | **Кол-во бит** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **пробел** | **17** | 0 | 0 | 0 |  |  |  |  | 000 | 51 |
| **о** | **7** | 0 | 0 | 1 |  |  |  |  | 001 | 21 |
| **ее** | **7** | 0 | 1 | 0 |  |  |  |  | 010 | 21 |
| **нн** | **7** | 0 | 1 | 1 | 0 |  |  |  | 0110 | 28 |
| **а** | **6** | 0 | 1 | 1 | 1 |  |  |  | 0111 | 24 |
| **лл** | **5** | 1 | 0 | 0 | 0 |  |  |  | 1000 | 20 |
| **дд** | **5** | 1 | 0 | 0 | 1 |  |  |  | 1001 | 20 |
| **в** | **5** | 1 | 0 | 1 | 0 |  |  |  | 1010 | 20 |
| **и** | **4** | 1 | 0 | 1 | 1 | 0 |  |  | 10110 | 20 |
| **ыы** | **3** | 1 | 0 | 1 | 1 | 1 |  |  | 10111 | 15 |
| **мм** | **3** | 1 | 1 | 0 | 0 | 0 |  |  | 11000 | 15 |
| **рр** | **2** | 1 | 1 | 0 | 0 | 1 | 0 |  | 110010 | 12 |
| **,** | **2** | 1 | 1 | 0 | 0 | 1 | 1 |  | 110011 | 12 |
| **чч** | **2** | 1 | 1 | 0 | 1 | 0 |  |  | 11010 | 10 |
| **тт** | **2** | 1 | 1 | 0 | 1 | 1 | 0 |  | 110110 | 12 |
| **пп** | **1** | 1 | 1 | 0 | 1 | 1 | 1 |  | 110111 | 6 |
| **ПП** | **1** | 1 | 1 | 1 | 0 | 0 | 0 |  | 111000 | 6 |
| **.** | **1** | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1110010 | 7 |
| **юю** | **1** | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1110011 | 7 |
| **В** | **1** | 1 | 1 | 1 | 0 | 1 | 0 |  | 111010 | 6 |
| **!** | **1** | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1110110 | 7 |
| **бб** | **1** | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1110111 | 7 |
| **НН** | **1** | 1 | 1 | 1 | 1 | 0 | 0 |  | 111100 | 6 |
| **хх** | **1** | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1111010 | 7 |
| **с** | **1** | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1111011 | 7 |
| **К** | **1** | 1 | 1 | 1 | 1 | 1 | 0 |  | 111110 | 6 |
| **-** | **1** | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1111110 | 7 |
| **йй** | **1** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1111111 | 7 |
|  |  |  |  |  |  |  |  |  |  | 387 |

Таблица 1 — Сжатие методом Шеннона-Фано

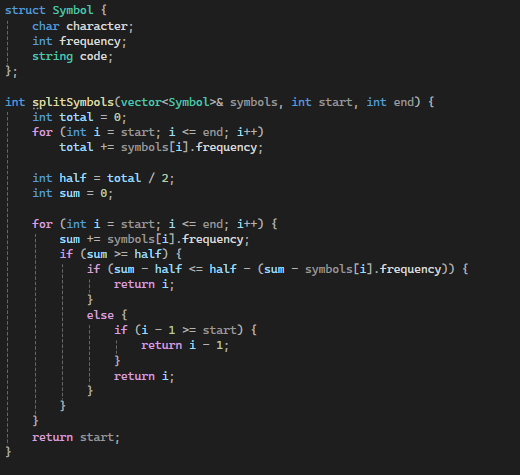
Не закодированная фраза — 91 \* 8 = 728 бит

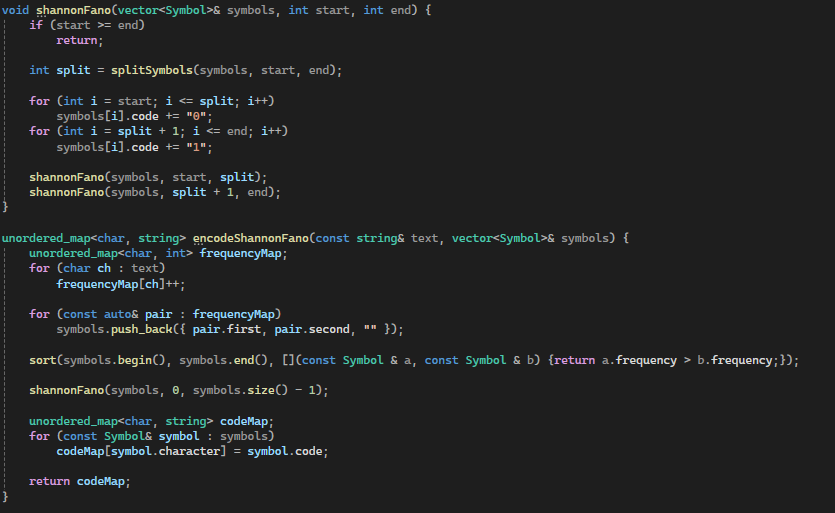
Закодированная фраза — 387 бит

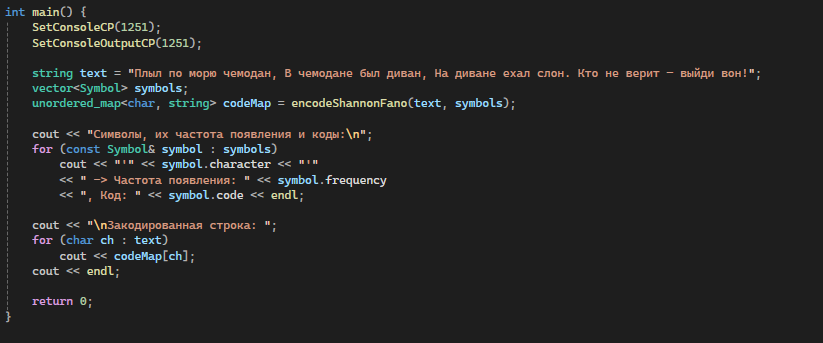
Закодированная строка:

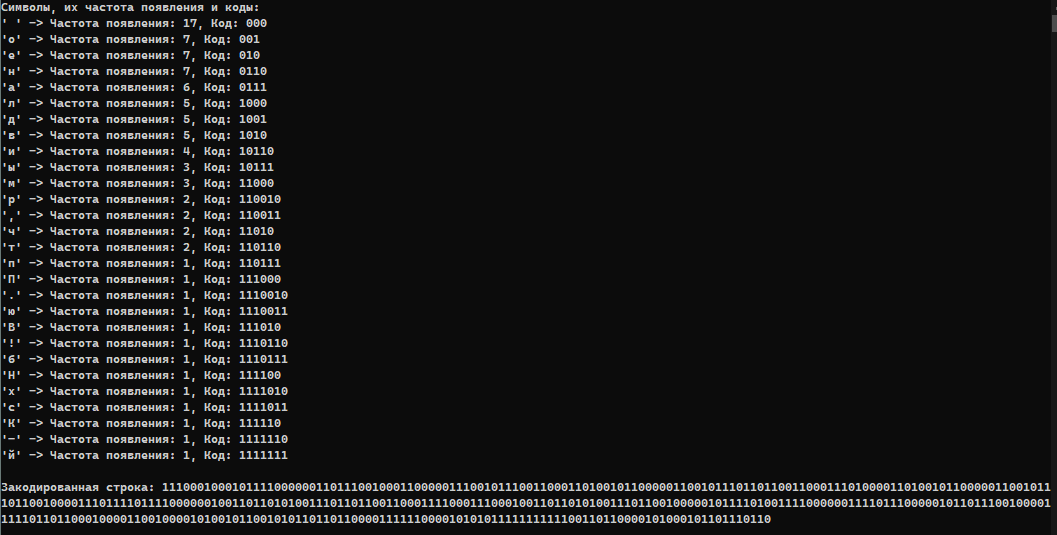
111000100010111100000011011100100011000001110010111001100011010010110000011001011101101100110001110100001101001011000001100101110110010000111011110111100000010011011010100111011011001100011110001110001001101101010011101100100000101111010011110000001111011100000101101110010000111110110110001000011001000010100101100101011011011000011111100001010101111111111100110110000101000101101110110

## 3.2 Реализация задачи и тестирование

Рисунок 1 — Стукрура символа и функция разделения списка символов

Рисунок 2 — Рекурсивная функция кодирования символов и основная функция сжатия методом Шеннона-Фано

Рисунок 3 — Основная функция программы

Рисунок 4 — Вывод программы для строки согласно варианту

# 4 ЗАДАНИЕ 1.2

## 4.1 Алгоритм решения

Метод Лемпеля-Зива (LZ77) — это алгоритм сжатия данных, который заменяет повторяющиеся фрагменты на пары (смещение, длина совпадения, следующий символ после совпадения). Для каждого символа алгоритм ищет ранее встречавшийся фрагмент в скользящем окне и заменяет его на ссылку на этот фрагмент. Если фрагмент не найден, символ записывается как есть. LZ77 эффективен для текстов с повторяющимися подстроками, но требует памяти для хранения буфера.

Принцип восстановления сжатой информации в LZ77 заключается в следующем: сжатые данные содержат пары (позиция, длина) и символ. При восстановлении, для каждой пары (позиция, длина), алгоритм копирует из уже восстановленных данных фрагмент длины, указанной в паре, начиная с позиции, и добавляет к этому следующий символ.

Для варианта №17 кодировка фразы состоящей из двух символьного алфавита (0, 1) представлена в Таблице 2.

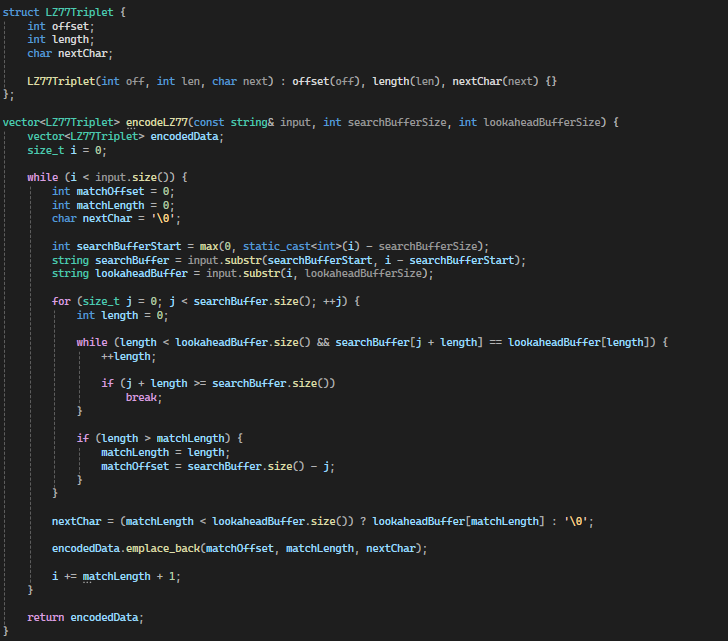
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Шаг** | **Скользящее окно** | | **Совпадающая фраза** | **Закодированные данные** | | |
| **Окно поиска (5 символов)** | **Буфер (5 символов)** | **offset** | **length** | **nextChar** |
| **1** | - | 00010 | - | 0 | 0 | 0 |
| **2** | 0 | 00100 | 0 | 1 | 1 | 0 |
| **3** | 000 | 10000 | - | 0 | 0 | 1 |
| **4** | 0001 | 00001 | 000 | 4 | 3 | 0 |
| **5** | 10000 | 10101 | 10 | 5 | 2 | 1 |
| **6** | 00101 | 01001 | 010 | 4 | 3 | 0 |
| **7** | 10100 | 1101 | 1 | 5 | 1 | 1 |
| **8** | 100111 | 01 | 01 | 3 | 2 | - |

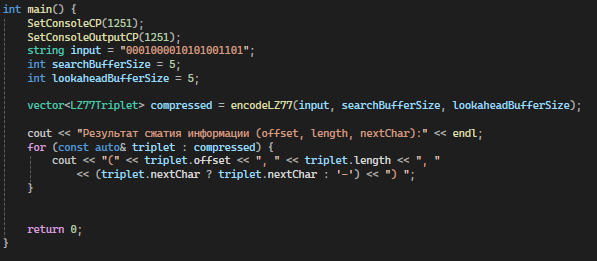
Таблица 2 - Сжатие методом Лемпеля –Зива LZ77

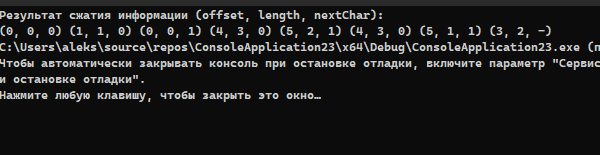
Сжатая фраза:

(0, 0, 0) (1, 1, 0) (0, 0, 1) (4, 3, 0) (5, 2, 1) (4, 3, 0) (5, 1, 1) (3, 2, -)

## 4.2 Реализация задачи и тестирование

Рисунок 5 — Структура триплета и функция сжатия методом Лемпеля –Зива LZ77

Рисунок 6 — Основная функция программы

Рисунок 7 - Вывод программы для строки согласно варианту

# 5 ЗАДАНИЕ 1.3

## 5.1 Алгоритм решения

LZ78 — это один из алгоритмов сжатия данных. Он работает на основе построения словаря, в котором хранятся уникальные подстроки данных, встречающиеся в процессе их обработки.

Алгоритм работы LZ78 можно описать следующим образом. На этапе кодирования входная строка читается посимвольно. На каждом шаге алгоритм ищет самую длинную подстроку, которая уже есть в словаре. Если такая подстрока найдена, алгоритм добавляет в словарь новую запись, состоящую из индекса этой подстроки в словаре, дополненного следующим символом из входной строки. Если же подстрока отсутствует в словаре, она сразу добавляется в словарь, а для её кодирования записывается пара с индексом 0 и первым символом подстроки. Таким образом, LZ78 кодирует данные, динамически формируя словарь без необходимости полного анализа входного потока заранее.

Процесс декодирования является обратным. Декодер начинает с пустого словаря, идентичного тому, что создавался при сжатии. Для каждой пары (индекс, символ) из сжатых данных декодер извлекает подстроку из словаря по заданному индексу и добавляет к ней символ. Затем эта новая строка добавляется в словарь, а результат декодирования объединяется с ранее восстановленными данными.

Таким образом, алгоритм LZ78 эффективен для сжатия данных, содержащих повторяющиеся структуры, поскольку он позволяет сократить размер представления за счёт построения и использования компактного словаря.

Для варианта №17 кодировка фразы представлена в Таблице 3.

|  |  |  |
| --- | --- | --- |
| Словарь | Считываемое содержимое | Код |
| - | w | (o, w) |
| w = 1 | e | (o, e) |
| w = 1, e = 2 | b | (o, b) |
| w = 1, e = 2, b = 3 | we | (1, e) |
| w = 1, e = 2, b = 3  we = 4 | r | (0, r) |
| w = 1, e = 2, b = 3, r = 5  we = 4 | bw | (3, w) |
| w = 1, e = 2, b = 3, r = 5  we = 4, bw = 6 | eb | (2, b) |
| w = 1, e = 2, b = 3, r = 5  we = 4, bw = 6, eb = 7 | er | (2, r) |
| w = 1, e = 2, b = 3, r = 5  we = 4, bw = 6, eb = 7, er = 8 | web | (4, b) |
| w = 1, e = 2, b = 3, r = 5  we = 4, bw = 6, eb = 7, er = 8  web = 9 | erw | (8, w) |
| w = 1, e = 2, b = 3, r = 5  we = 4, bw = 6, eb = 7, er = 8  web = 9, erw = 10 |  | (7, -) |

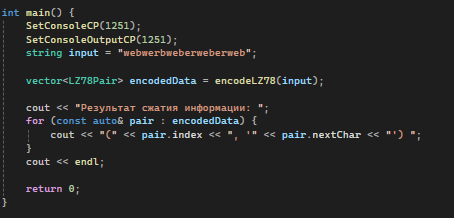
Таблица 3 - Сжатие методом LZ78

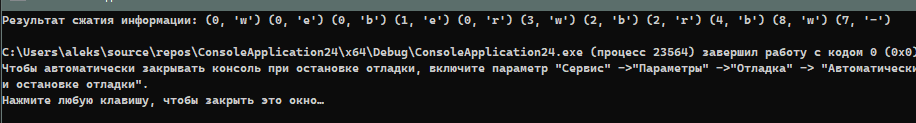
Сжатая фраза:

(0, 'w') (0, 'e') (0, 'b') (1, 'e') (0, 'r') (3, 'w') (2, 'b') (2, 'r') (4, 'b') (8, 'w') (7, '-')

## 5.2 Реализация задачи и тестирование

Рисунок 8 - Структура пары и функция сжатия методом LZ78

Рисунок 9 - Основная функция программы

Рисунок 10 - Вывод программы для строки согласно варианту

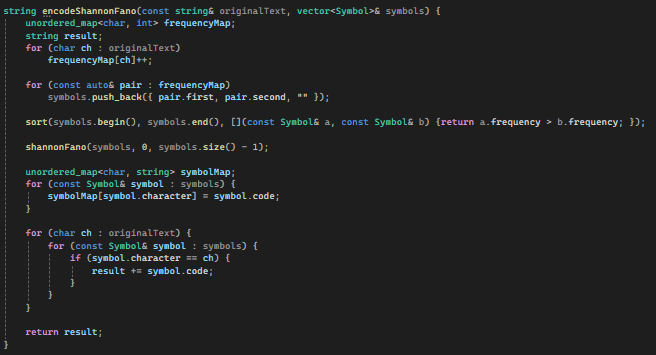
# 6 ЗАДАНИЕ 2.1

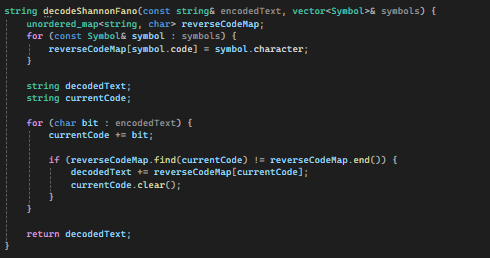
## 6.1 Требования к выполнению задания

Разработать алгоритм и реализовать программу сжатия текста алгоритмом Шеннона – Фано. Разработать алгоритм и программу восстановления сжатого текста. Выполнить тестирование программы на текстовом файле. Определить процент сжатия.

## 6.2 Реализация задачи и тестирование

Некоторые функции для реализации задания были взяты из задания 1.1, и их функциональность осталась без изменений. Однако в процессе работы были внесены следующие изменения:

Рисунок 11 — Переработанная функция сжатия текстам

Рисунок 12 — Функция восстановления сжатого файла

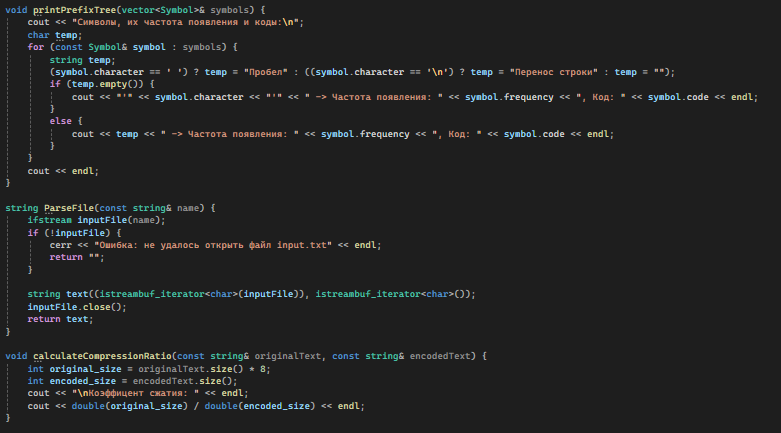
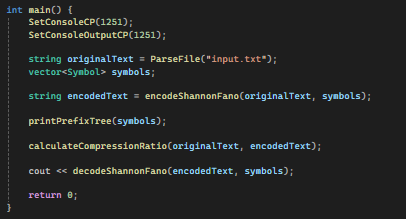
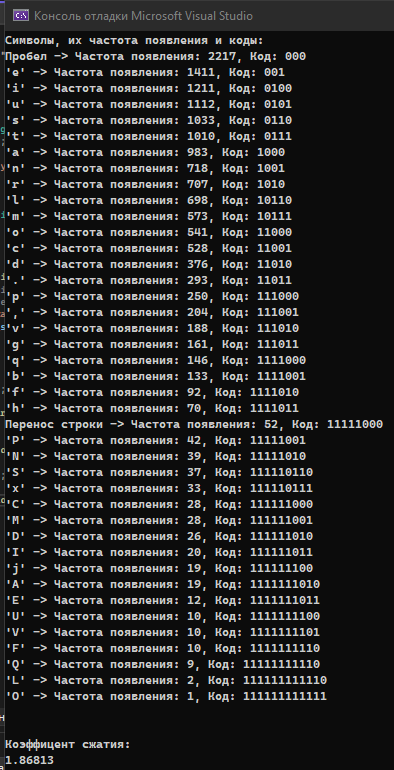
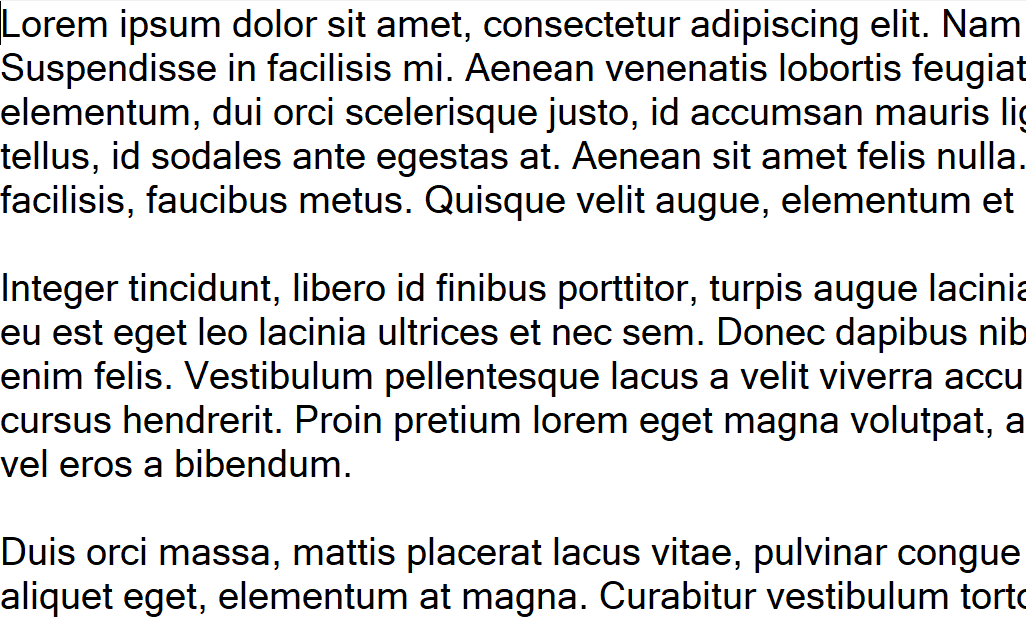
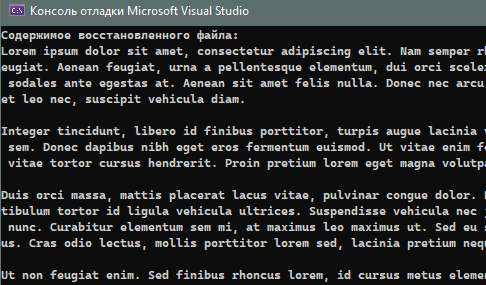
Рисунок 13 — Функции вывода префиксного дерева, чтения файла с текстом и вычисления коэффициента сжатия

Рисунок 14 — Основная функция программы

Рисунок 15 — Вывод дерева префиксов и коэффициента сжатия

Рисунок 16 — Содержание тестового файла

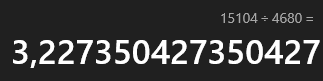
Рисунок 17 — Тестирование функции восстановления сжатого файла

## 6.3 Анализ коэффициента сжатия

Коэффициент сжатия методом Шеннона – Фано: 1.86813

Рисунок 18 — Вывод функции вычисления коэффициента сжатия

Коэффициент сжатия архиватором WinRAR: 3.22735

  
  
Рисунок 19 — Вычисленный коэффициент сжатия архиватором WinRAR

Из результатов сжатия видно, что полноценное приложение производит сжатие лучше, чем реализованная программа.

# 7 ЗАДАНИЕ 2.2

## 7.1 Требования к выполнению задания

Провести кодирование(сжатие) исходной строки символов «Фамилия Имя Отчество» с использованием алгоритма Хаффмана. Исходная строка символов, таким образом, определяет индивидуальный вариант задания для каждого студента.

## 7.2 Алгоритм решения

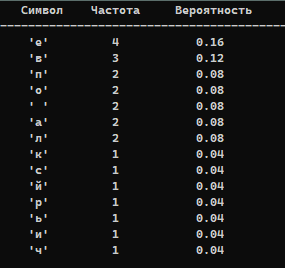
Алгоритм Хаффмана — это метод построения оптимального префиксного кода, который минимизирует среднюю длину кодируемого сообщения.

Работа алгоритма начинается с анализа входных данных: каждому символу присваивается его частота появления. Эти частоты используются для построения дерева Хаффмана, где каждый символ представляет собой лист, а его частота становится весом узла. На первом этапе все символы рассматриваются как отдельные узлы. Далее начинается процесс построения дерева. Алгоритм выбирает два узла с наименьшими весами и объединяет их в один новый узел, вес которого равен сумме весов объединяемых узлов. Этот новый узел становится родительским для выбранных узлов, а старые узлы удаляются из набора. Процесс продолжается до тех пор, пока не останется только один узел, представляющий корень дерева.

После построения дерева каждому символу присваивается уникальный бинарный код. Это делается путем прохождения по дереву от корня до листа: каждой ветви присваивается 0 или 1, в зависимости от направления. В итоге получается префиксный код, где ни один код не является началом другого.

Индивидуальный вариант (ФИО студента):

Попов Алексей Валерьевич

Рисунок 20 - Таблица отсортированных частот встречаемости символов в исходной строке

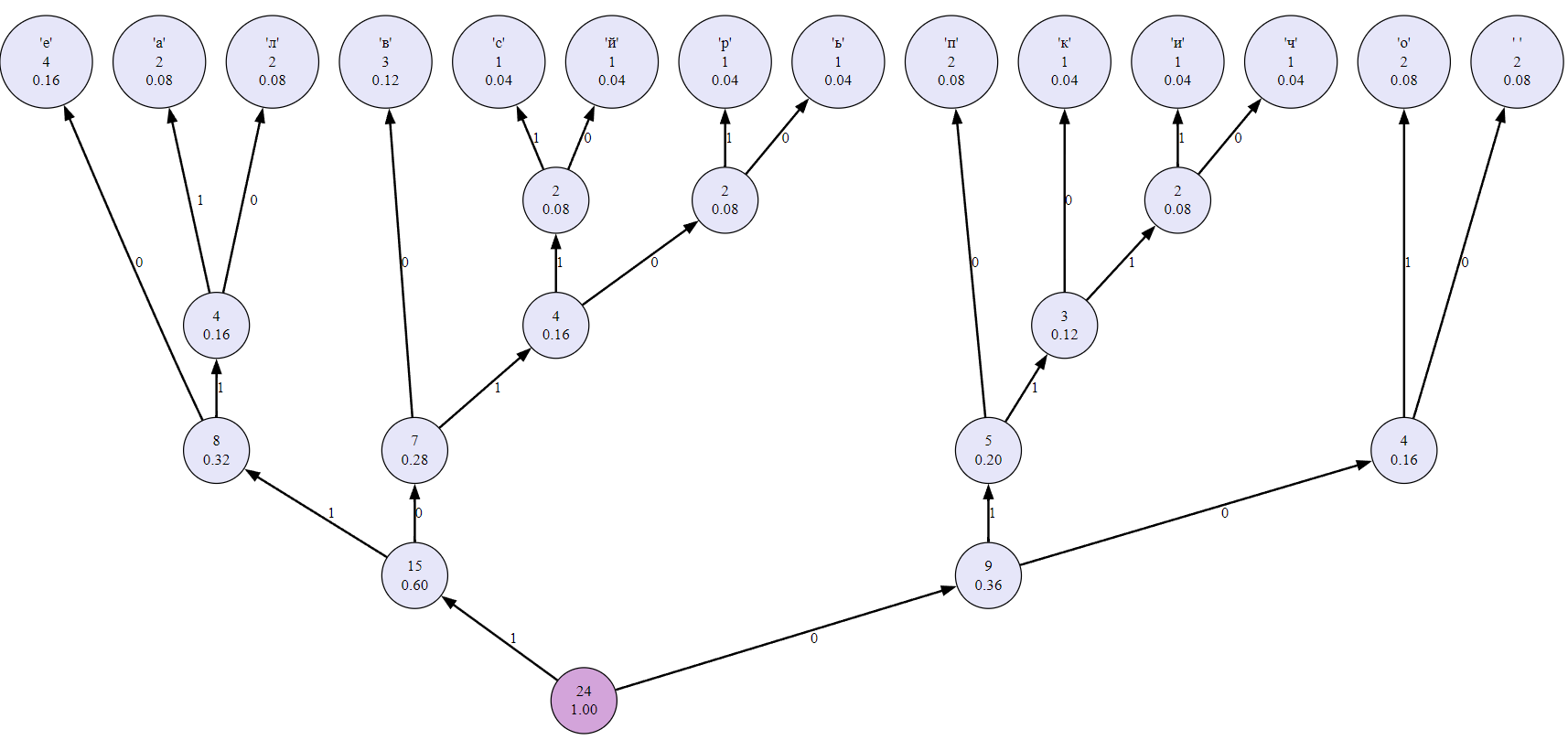
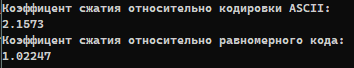
Рисунок 21 - Дерево кодирования Хаффмана

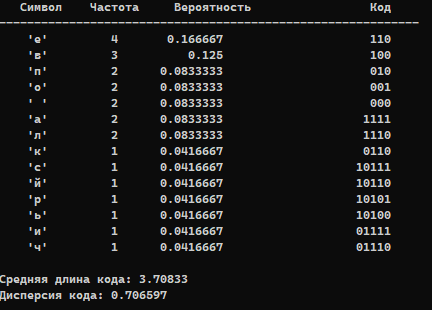
Рисунок 22 — Сжатая строка

Рисунок 23 - Коэффициенты сжатия относительно кодировки ASCII и

относительно равномерного кода

Коэффициент сжатия относительно ASCII больше, потому что ASCII кодирует каждый символ фиксированным количеством бит (8 бит), что избыточно для многих символов.

Коэффициент сжатия относительно **равномерного кода** всегда будет ближе к 1, так как равномерный код уже оптимален для равномерного распределения символов. Но если распределение сильно неравномерное, этот коэффициент тоже может значительно увеличиваться.

Рисунок 24 - Рассчет средней длины полученного кода и его дисперсии

## 7.3 Реализация задачи

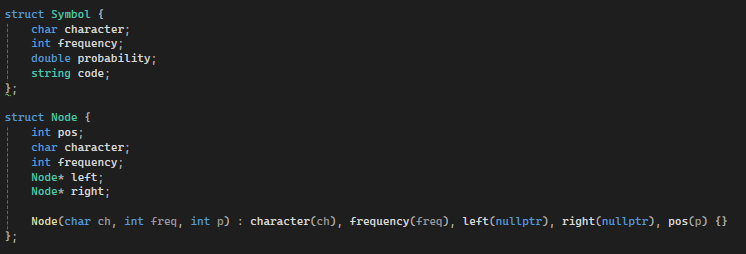
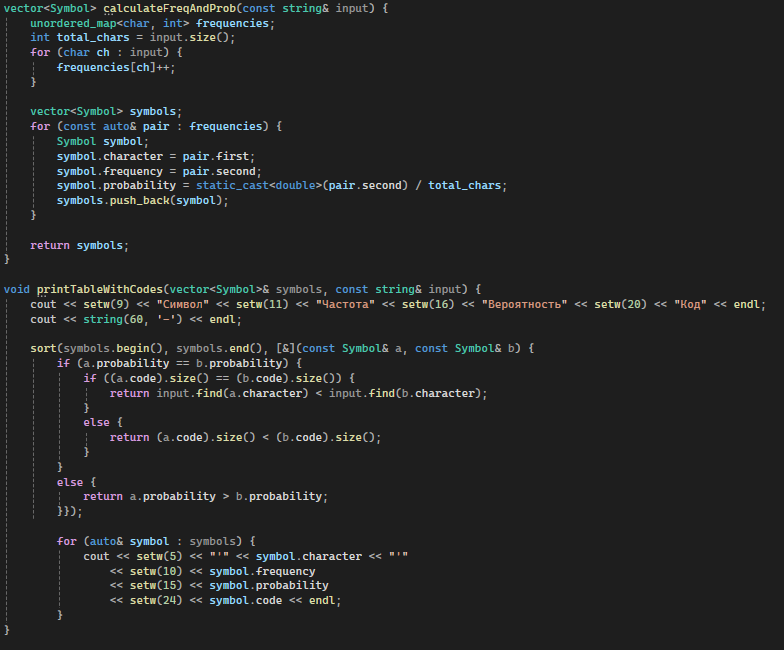
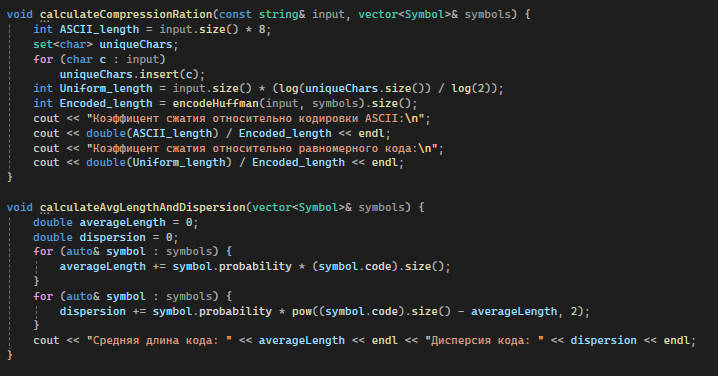
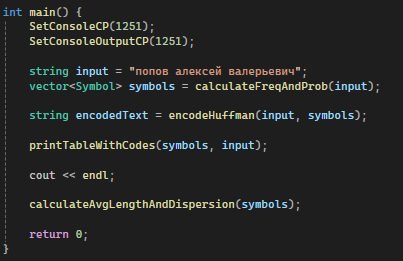
Рисунок 25 — Структуры для записи символа и узла дерева

Рисунок 26 — Функции реализации сжатия методом Хаффмана

Рисунок 27 — Функция рассчета частот встречаемости символов и их вероятности появления, функция вывода таблицы кодов

Рисунок 28 — Функция вычисления коэффициента сжатия относительно кодировки ASCII и равномерного кода, функция вычисления средней длины кода и его дисперсии

Рисунок 29 — Основная функция программы

# 8 ЗАДАНИЕ 2.3

## 8.1 Требования к выполнению задания

Применить алгоритм Хаффмана для архивации данных текстового файла. Выполнить практическую оценку сложности алгоритма Хаффмана. Провести архивацию этого же файла любым архиватором. Сравнить коэффициенты сжатия разработанного алгоритма и архиватора.

## 8.2 Реализация задачи

Большая часть функций для реализации задания были взяты из задания 2.2, и их функциональность осталась без изменений. Однако в процессе работы были внесены следующие изменения:

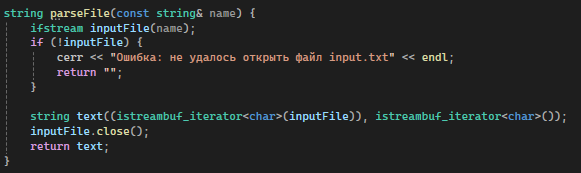
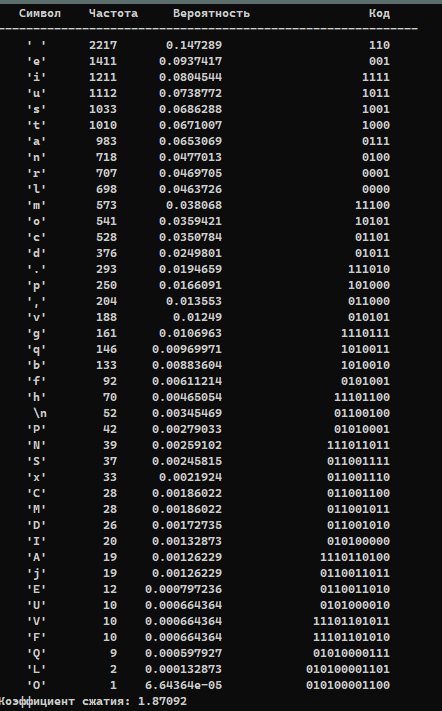
Рисунок 30 — Функция чтения текста из файла

Рисунок 31 — Функция вывода коэффициента сжатия алгоритмом Хаффмана

## 8.3 Анализ коэффициента сжатия

Рисунок 32 — Вывод таблицы кодов для текстового файла и коэффициента сжатия

Коэффициент сжатия алгоритмом Хаффмана: 1.87092

Рисунок 33 — Рассчет коэффициента сжатия архиватором WinRAR

Коэффициент сжатия архиватором WinRAR: 3.22735

Из результатов сжатия видно, что полноценное приложение производит сжатие лучше, чем реализованная программа.

# 9 ВЫВОД

В рамках работы были изучены и реализованы различные алгоритмы сжатия данных, включая LZ77, LZ78, Шеннона-Фано и Хаффмана. Выполнен расчет коэффициентов сжатия, построены таблицы частот и коды для префиксных деревьев, а также реализованы методы восстановления данных. Алгоритмы были протестированы на реальных данных, включая текстовые файлы, что позволило оценить их эффективность в зависимости от структуры данных. По итогам работы определены сильные и слабые стороны каждого метода, а также их применимость для разных типов данных.