|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | ИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий |
|  |
| Кафедра Математического обеспечения и стандартизации информационных технологий |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 8.1** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема:**  «Отдельные вопросы алгоритмизации» | |
|  | |
| Выполнил студент группы ИНБО-12-23 | Албахтин И.В. |
| Принял ассистент | Муравьёва Е.А. |

Москва 2024

**СОДЕРЖАНИЕ**

[1 ЗАДАНИЕ 1 РАБОТЫ 8.1 3](#_Toc184076906)

[1.1 Формулировка задачи 3](#_Toc184076907)

[1.2 Описание алгоритмов 3](#_Toc184076908)

[2 ЗАДАНИЕ 2 РАБОТЫ 8.1 7](#_Toc184076909)

[2.1 Формулировка задачи 7](#_Toc184076910)

[2.2 Описание алгоритма Шеннона – Фано 7](#_Toc184076911)

[2.3 Код программы (алгоритм Шеннона – Фано) 8](#_Toc184076912)

[2.4 Тестирование программы (алгоритм Шеннона – Фано) 11](#_Toc184076913)

[2.5 Описание алгоритма Хаффмана 12](#_Toc184076914)

[2.6 Код программы (алгоритм Хаффмана) 15](#_Toc184076915)

[2.7 Тестирование программы (алгоритм Хаффмана) 18](#_Toc184076916)

[3 ВЫВОД 20](#_Toc184076917)

[4 ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ 21](#_Toc184076918)

**Цель работы:** исследовать алгоритмы сжатия на примерах, разработать программы сжатия и восстановления текста методами Хаффмана и Шеннона-Фано, реализовать алгоритм на основе сокращения числа переборов.

# 1 ЗАДАНИЕ 1 РАБОТЫ 8.1

## Формулировка задачи

Исследование алгоритмов сжатия на примерах

1) Выполнить каждую задачу варианта, представив алгоритм решения в виде таблицы и указав результат сжатия.

2) Описать процесс восстановления сжатого текста.

Индивидуальный вариант 3:

|  |  |  |
| --- | --- | --- |
| Сжатие данных по методу Лемпеля– Зива LZ77  Используя двухсимвольный алфавит (0, 1) закодировать следующую фразу: | Закодировать следующую фразу, используя код LZ78 | Закодировать фразу методами Шеннона– Фано |
| 0100101010010000101 | лорлоралоранранлоран | Эне-бене, рики-таки,  Буль-буль-буль,  Караки-шмаки Эус-  деус-краснодеус бац |

## Описание алгоритмов

Сжатие данных по методу Лемпеля– Зива LZ77 (см. табл. 1).

Используя двухсимвольный алфавит (0, 1) закодировать следующую фразу: 0100101010010000101.

Таблица 1 – Сжатие по методу Лемпеля– Зива LZ77

|  |  |
| --- | --- |
| Исходный  текст | 0.10.01.010.100.1000.0101 |
| LZ-код | 0.10.001.111.0100.1010.1001 |
| R | 2 3 |
| Вводимые  коды | - 10 11 100 101 110 111 |

Таблица 2 – Соответствия комбинаций символов и кодов

|  |  |
| --- | --- |
| Текст | Код |
| 0 | 000 |
| 1 | 001 |
| 10 | 010 |
| 01 | 011 |
| 010 | 100 |
| 100 | 101 |
| 1000 | 110 |
| 0101 | 111 |

Закодированная последовательность: 0.10.001.111.0100.1010.1001

Закодировать следующую фразу, используя код LZ78:

лорлоралоранранлоран (см. табл. 2).

Таблица 2 – Закодирование по коду LZ78

|  |  |  |
| --- | --- | --- |
| Словарь | Считываемое содержимое | Код |
|  | 1. л | <0, л> |
| л = 1 | 1. о | <0, о> |
| л = 1; о = 2 | 1. р | <0, р> |
| л = 1; о = 2; р = 3 | 1. ло | <1, о> |
| л = 1; о = 2; р = 3; ло = 4 | 1. ра | <3, а> |
| л = 1; о = 2; р = 3; ло = 4; ра = 5 | 1. лор | <4, р> |
| л = 1; о = 2; р = 3; ло = 4; ра = 5; лор = 6 | 1. а | <0, а> |
| л = 1; о = 2; р = 3; ло = 4; ра = 5; лор = 6; a = 7 | 1. н | <0, н> |
| л = 1; о = 2; р = 3; ло = 4; ра = 5; лор = 6; а = 7; н = 8 | 1. ран | <5, н> |
| л = 1; о = 2; р = 3; ло = 4; ра = 5; лор = 6; а = 7; н = 8; ран = 9 | 1. ранл | <9, л> |
| л = 1; о = 2; р = 3; ло = 4; ра = 5; лор = 6; а = 7; н = 8; ран = 9; ранл = 10; ор = 11 | 1. ор | <2, р> |
| л = 1; о = 2; р = 3; ло = 4; ра = 5; лор = 6; а = 7; н = 8; ран = 9; ранл = 10; ор = 11; ан = 12 | 1. ан | <7, н> |

Закодированная фраза: 0л0о0р1о3а4р0а0н5н9л2р7н

Закодирование фразы методом Шеннона-Фано (см. табл. 3).

Эне-бене, рики-таки, Буль-буль-буль, Караки-шмаки Эус-деус-краснодеус бац

1. Оформление таблицы метод Шеннона-Фано

Таблица 3 – Закодирование фразы методом Шеннона-Фано

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| - | 7 | 0 | 0 | 0 |  |  |  |  | 000 | 21 |
| к | 6 | 0 | 0 | 1 |  |  |  |  | 001 | 18 |
| а | 6 | 0 | 1 | 0 | 0 |  |  |  | 0100 | 24 |
| у | 6 | 0 | 1 | 0 | 1 |  |  |  | 0101 | 24 |
| пробел | 5 | 0 | 1 | 1 | 0 |  |  |  | 0110 | 20 |
| е | 5 | 0 | 1 | 1 | 1 |  |  |  | 0111 | 20 |
| б | 5 | 1 | 0 | 0 | 0 |  |  |  | 1000 | 20 |
| и | 5 | 1 | 0 | 0 | 1 |  |  |  | 1001 | 20 |
| с | 4 | 1 | 0 | 1 | 0 |  |  |  | 1010 | 16 |
| н | 3 | 1 | 0 | 1 | 1 | 0 |  |  | 10110 | 15 |
| , | 3 | 1 | 0 | 1 | 1 | 1 |  |  | 10111 | 15 |
| р | 3 | 1 | 1 | 0 | 0 |  |  |  | 1100 | 15 |
| л | 3 | 1 | 1 | 0 | 1 | 0 |  |  | 11010 | 15 |
| ь | 3 | 1 | 1 | 0 | 1 | 1 |  |  | 11011 | 15 |
| э | 2 | 1 | 1 | 1 | 0 | 0 |  |  | 11100 | 10 |
| д | 2 | 1 | 1 | 1 | 0 | 1 |  |  | 11101 | 10 |
| т | 1 | 1 | 1 | 1 | 1 | 0 | 0 |  | 111100 | 6 |
| ш | 1 | 1 | 1 | 1 | 1 | 0 | 1 |  | 111101 | 6 |
| м | 1 | 1 | 1 | 1 | 1 | 1 | 0 |  | 111110 | 6 |
| о | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1111110 | 7 |
| ц | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1111111 | 7 |
|  |  |  |  |  |  |  |  |  |  | 310 |

1. Оценка объема и коэфицента сжатия

Незакодированная фраза – 73\*8 = 584 бит

Закодированная фраза – 310 бит

Коэффицент сжатия: 310/584 = 0,53

1. Дерево префиксного кода Фано

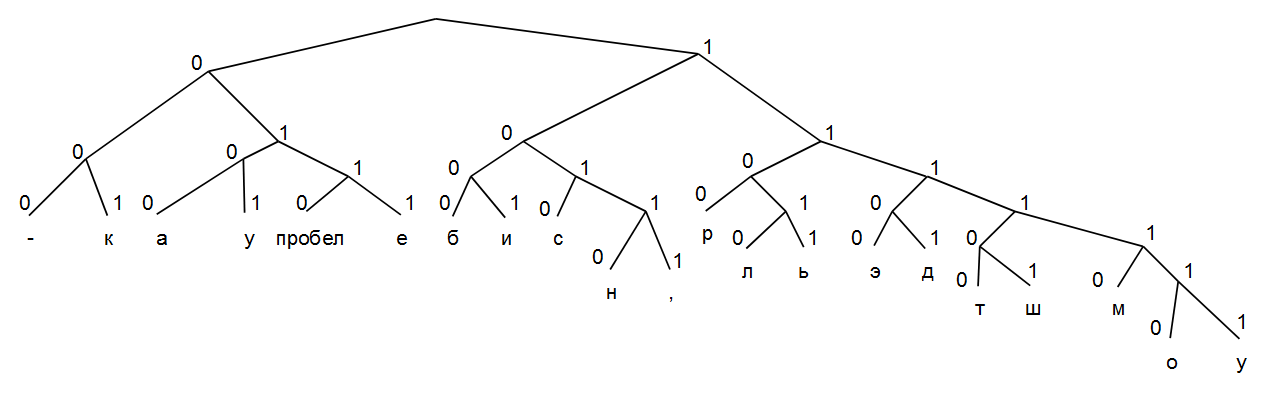


Рисунок 1 – Дерево префиксного кода Фано

# ЗАДАНИЕ 2 РАБОТЫ 8.1

## Формулировка задачи

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.

## Описание алгоритма Шеннона – Фано

Программа реализует алгоритм Шеннона-Фано для сжатия текста. Основные этапы работы программы:

Считывание текста из файла: Программа открывает файл input.txt и считывает его содержимое в строку text.

Расчет вероятностей символов: Программа подсчитывает частоту встречаемости каждого символа в тексте, используя стандартную библиотеку map. Затем вычисляются вероятности появления каждого символа, и они сохраняются в вектор symbols, который содержит структуру Symbol. Каждый элемент этой структуры хранит символ, его вероятность и код, который будет использоваться для его представления.

Сортировка символов по вероятностям: Символы сортируются по убыванию вероятности, чтобы использовать наименьшие коды для наиболее вероятных символов.

Генерация кодов по алгоритму Шеннона-Фано: Программа рекурсивно разбивает символы на две группы, основываясь на их вероятностях, и генерирует для них коды, добавляя "0" для левой группы и "1" для правой группы.

Кодирование текста: Текст кодируется, заменяя каждый символ его соответствующим кодом из словаря, построенного на основе сгенерированных кодов.

Восстановление текста: Кодированный текст декодируется, используя обратный словарь, чтобы вернуть символы из их кодов.

Подсчет процента сжатия: Программа вычисляет процент сжатия, сравнивая размер исходного текста (в битах) с размером закодированного текста.

Сложность алгоритма Шеннона-Фано:

Время: O(n log n), где n - количество символов.

Память: O(n), где n - количество символов.

O(n log n) возникает из-за необходимости сортировки символов по частоте. Алгоритм сортировки, как правило, имеет сложность O(n log n).

O(n) обусловлена необходимостью хранения информации о символах и их кодах.

## Код программы (алгоритм Шеннона – Фано)

На рисунках 1-4 представлен код программы, в котором реализован алгоритм Шеннона-Фано для вычисления кодов символов.

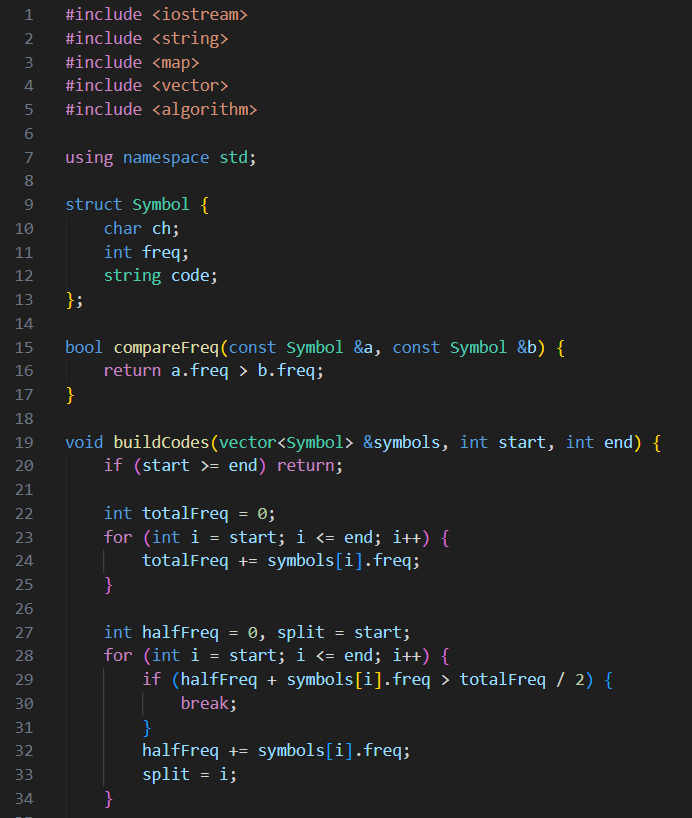


Рисунок 1 – Код программы по методу Шеннона-Фано

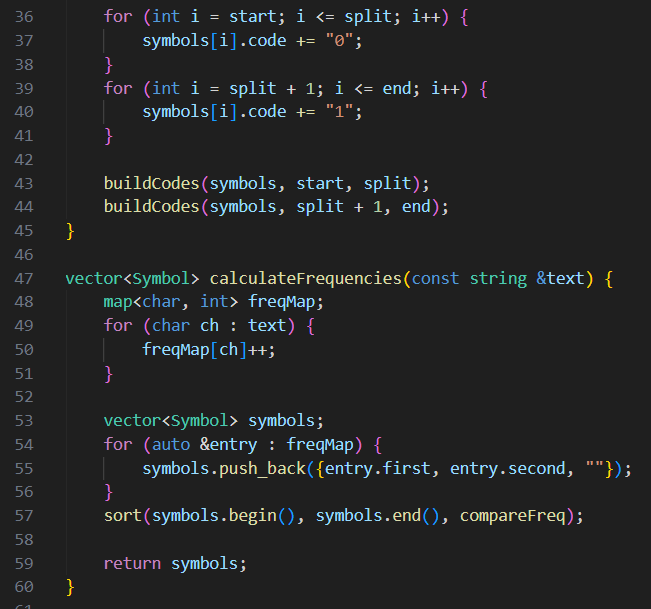


Рисунок 2 – Код программы по методу Шеннона-Фано

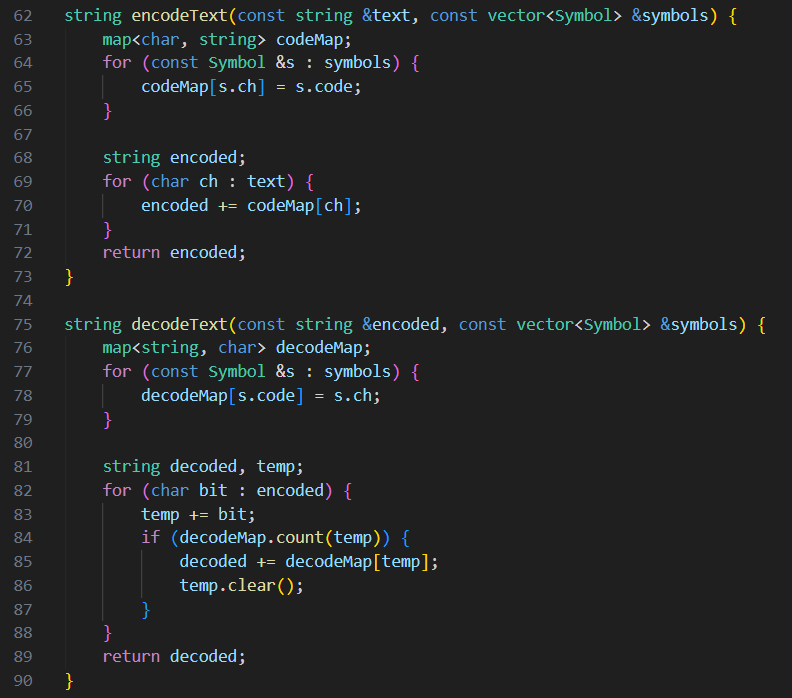


Рисунок 3 – Код программы по методу Шеннона-Фано

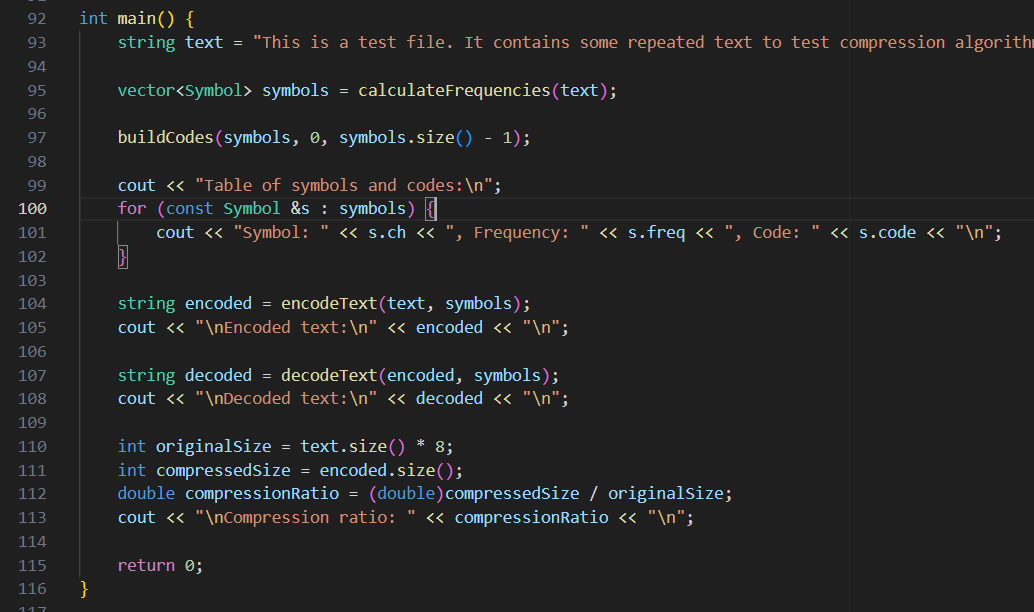


Рисунок 4 – Основной алгоритм программы

## Тестирование программы (алгоритм Шеннона – Фано)

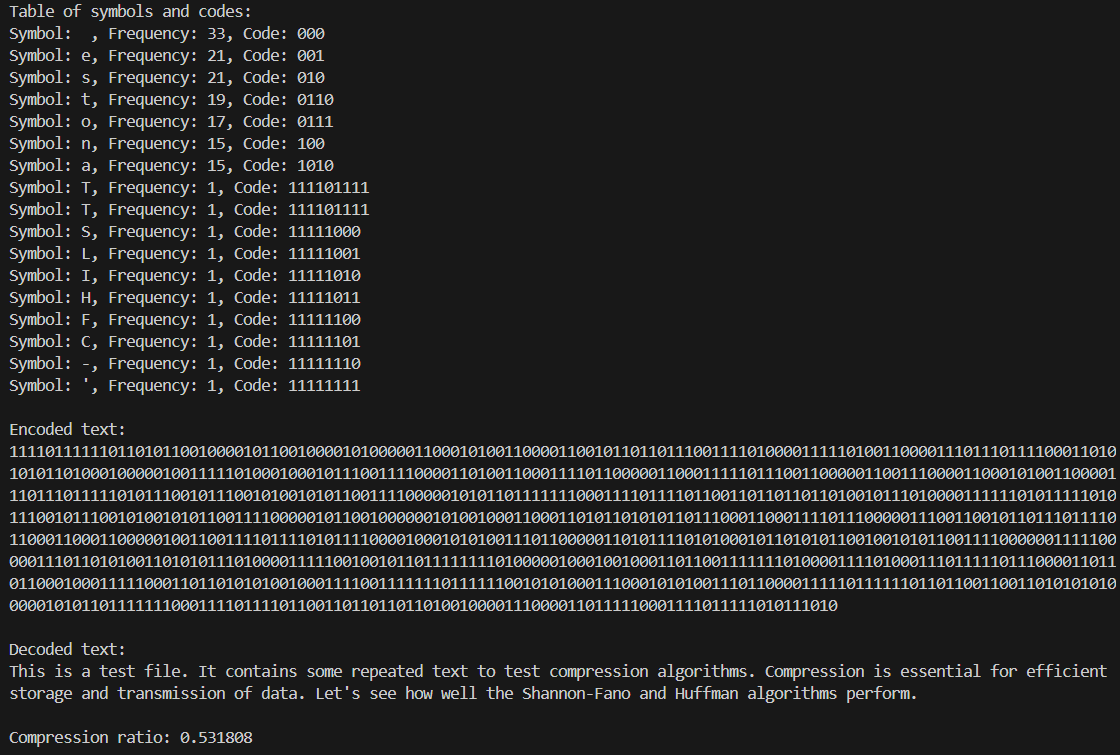


Рисунок 5 – Результат работы программы

Получаем, что для данного текста процент сжатия алгоритмом Шеннона-Фано равен 53,2%. Сравним это значение с сжатием ZIP. Для этого алгоритма процент сжатия будет 58,8%. Можно сделать вывод, что алгоритм Шеннона – Фано эффективен, но менее эффективный, чем ZIP сжатие.

## 2.5 Описание алгоритма Хаффмана

Сначала реализуем применение алгоритма Хаффмана вручную на примере. Проведем кодирование(сжатие) исходной строки символов «Гаврилина Анна Петровна» с использованием алгоритма Хаффмана (см. табл. 4).

Таблица 4 - Таблица частот встречаемости символов

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Алфавит | г | а | в | р | и | л | н | п | е | т | о | « » |
| Кол. вх. | 1 | 5 | 2 | 2 | 2 | 1 | 4 | 1 | 1 | 1 | 1 | 2 |
| Вероятн. | 0.04 | 0.22 | 0.09 | 0.09 | 0.09 | 0.04 | 0.17 | 0.04 | 0.04 | 0.04 | 0.04 | 0.09 |

Отсортируем алфавит в порядке убывания частот появления символов (см. табл. 5).

Таблица 5 – Таблица отсортированных частот

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Алфавит | а | н | в | р | и | « » | г | л | п | е | т | о |
| Кол. вх. | 5 | 4 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| Вероятн. | 0.22 | 0.17 | 0.09 | 0.09 | 0.09 | 0.09 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 |

Построим дерево кодирования Хаффмана (см. рис. 6).

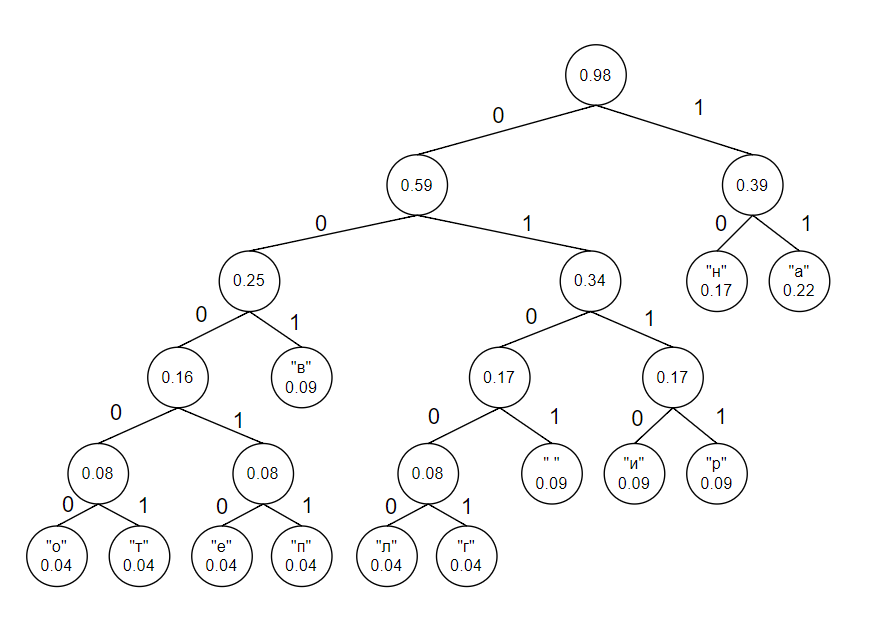


Рисунок 6 – Дерево Хаффмана

Закодируем фразу и получим (см. табл. 6).

Таблица 6 – Закодированная фраза

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| г | а | в | р | и | л | и | н | а |
| 01001 | 11 | 001 | 0111 | 0110 | 01000 | 0110 | 10 | 11 |
| \_ | а | н | н | а | \_ |  |  |  |
| 0101 | 11 | 10 | 10 | 11 | 0101 |  |  |  |
| п | е | т | р | о | в | н | а |  |
| 00011 | 00010 | 00001 | 0111 | 00000 | 001 | 10 | 11 |  |

Получившаяся закодировнная фраза: 78 бит.

Изначальная фраза в кодировке ASCII: 23 символа \* 8 бит = 184 бит.

Длина фразы в относительно равномерном коде: 23 символа \* 5 = 115 бит. (берем 5, так как достаточным будет количество символов для алфавита 2^5=32)

Процент сжатия относительно кодировки ASCII: 57%.

Процент сжатия относительно равномерного кода: 32%.

Рассчитаем среднюю длину полученного кода: 0.04\*5 \* 6 + 0.09\*4 \* 3 + 0.09\*3 + 0.17\*2 + 0.22\*2 = 3.33 бит/символ.

Поскольку при построении дерева кода Хаффмана может возникнуть некоторый произвол, для выбора оптимального варианта кода используют дисперсию. Дисперсия показывает насколько сильно отклоняются длины индивидуальных кодов от их средней величины.

Дисперсия: 0.04\*(5-3.33)^2 \* 6 + 0.09\*(4-3.33)^2 \* 3 + 0.09\*(3-3.33)^2 + 0.17(2-3.33)^2 + 0.22\*(2-3.33)^2 = 1.49.

Теперь опишем алгоритм работы программы.

1. **Определение структуры Node:**  
   Структура Node представляет узел дерева Хаффмана и содержит следующие члены:
   * ch — символ, хранимый в узле.
   * freq — частота встречаемости символа.
   * left и right — указатели на левого и правого потомков.
2. **Определение компаратора:**  
   Для приоритетной очереди используется структура Compare, где сравнение выполняется по частоте символов. Узлы с меньшей частотой имеют более высокий приоритет.
3. **Функция buildCodes:**  
   Это рекурсивная функция, которая обходит дерево Хаффмана и генерирует коды для каждого символа. Если достигается листовой узел (узел, содержащий символ), код для этого символа сохраняется в отображении huffmanCode.
4. **Функция frequencyAnalysis:**  
   Эта функция подсчитывает частоту каждого символа во входной строке и сохраняет данные в отображение freq.
5. **Функция buildHuffmanTree:**
   * Создаются узлы для каждого символа, основываясь на частоте их встречаемости, и добавляются в приоритетную очередь.
   * Узлы с минимальной частотой извлекаются, объединяются в новый родительский узел с частотой, равной сумме частот двух дочерних узлов.
   * Родительский узел добавляется обратно в очередь.
   * Этот процесс повторяется до тех пор, пока в очереди не останется один узел, который становится корнем дерева.
6. **Основная программа main:**
   * В начале программы выполняется анализ частот символов строки.
   * На основе полученных частот строится дерево Хаффмана.
   * Генерируются префиксные коды для символов.
   * Осуществляется кодирование исходного текста с использованием построенных кодов.
   * Выполняется декодирование текста для проверки корректности.
   * Программа выводит на экран частотный анализ, дерево Хаффмана, закодированную строку и раскодированный текст.
7. **Вывод дерева Хаффмана:**  
   Реализованная функция printTree позволяет визуализировать структуру дерева Хаффмана. Листовые узлы отображают символы и их частоту, внутренние узлы (без символов) отображают только суммарную частоту.
8. **Сложность алгоритма Хаффмана:**
   * **Время выполнения:**
     + Построение частотной таблицы имеет сложность O(n), где nn — длина текста.
     + Построение дерева Хаффмана — O(m \log m), где mm — количество уникальных символов (узлов).
   * **Память:**
     + Используется память O(m) для хранения частотной таблицы, дерева и кодов.
9. **Результаты программы:**
   * Частоты символов представлены в табличной форме.
   * Построенное дерево Хаффмана визуализируется в виде иерархии.
   * Исходный текст, закодированная строка и раскодированный текст выводятся для проверки.
   * Закодированная строка демонстрирует снижение объёма данных, что подтверждается процентом сжатия.
10. **Пример запуска:**  
    Программа успешно тестируется на тексте:

This is a test file. It contains some repeated text to test compression algorithms.

Процент сжатия рассчитывается путём сравнения длины оригинального текста в битах с длиной закодированной строки.

## Код программы (алгоритм Хаффмана)

На рисунках 7-10 представлен код программы, реализующий алгоритм Хаффмана для сжатия данных.

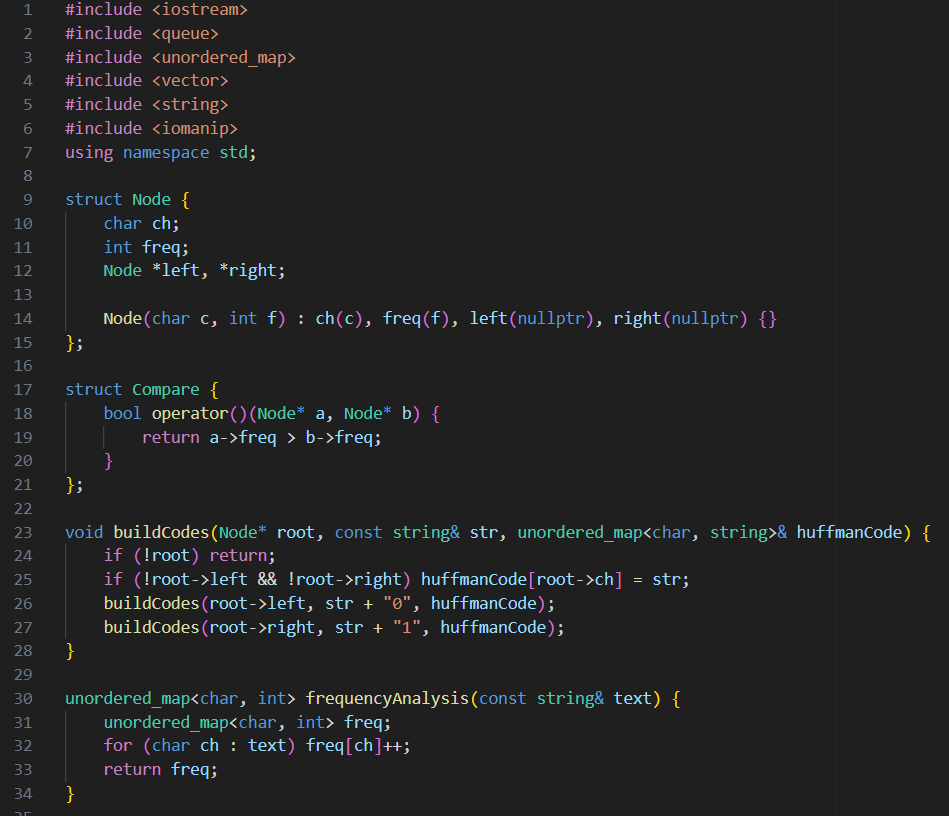


Рисунок 7 – Код программы, реализующий алгоритм Хаффмана для сжатия данных

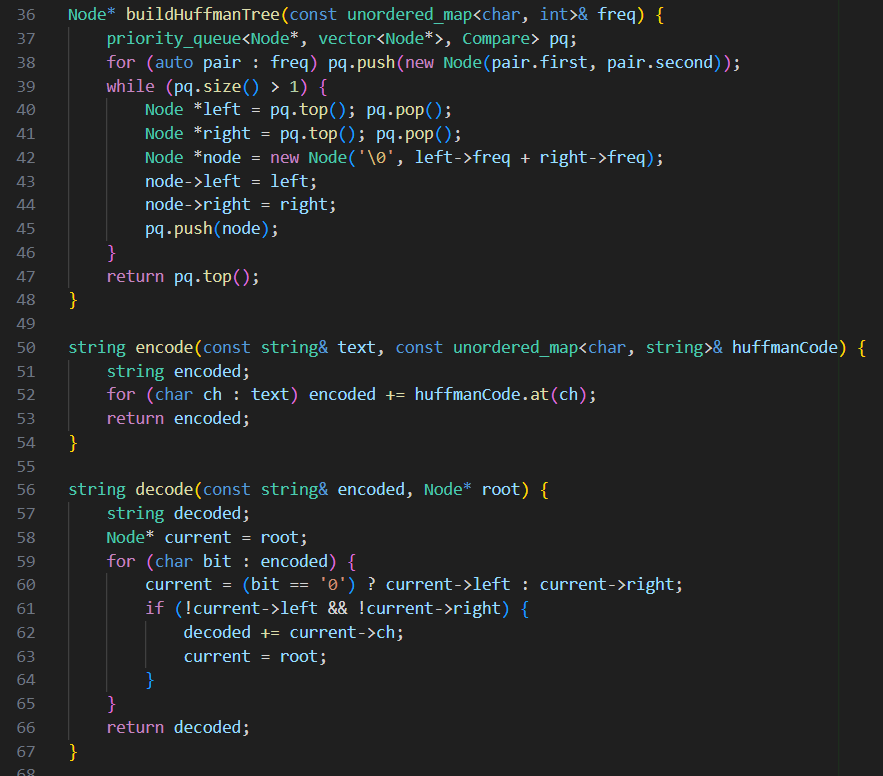


Рисунок 8 – Код программы, реализующий кодировку и декодировку текста

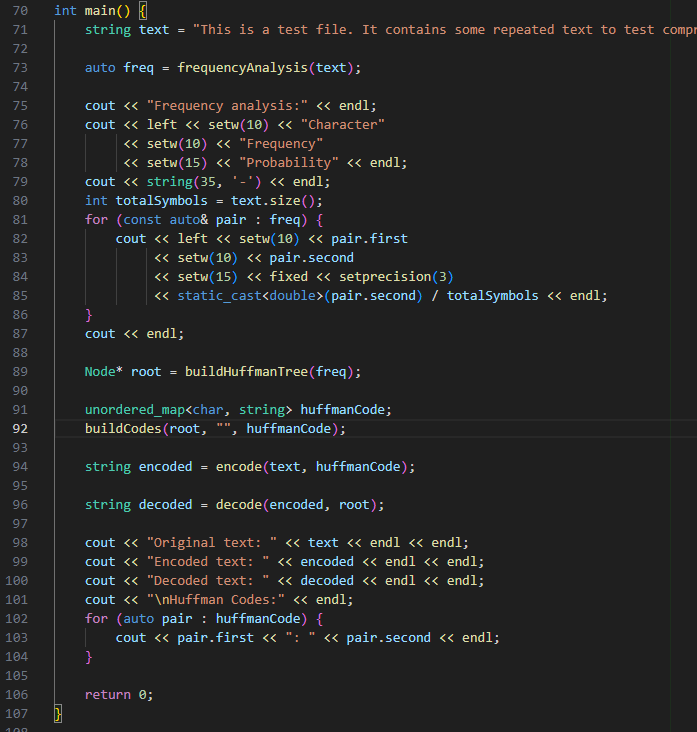


Рисунок 9 – Основной алгоритм программы main

## Тестирование программы (алгоритм Хаффмана)

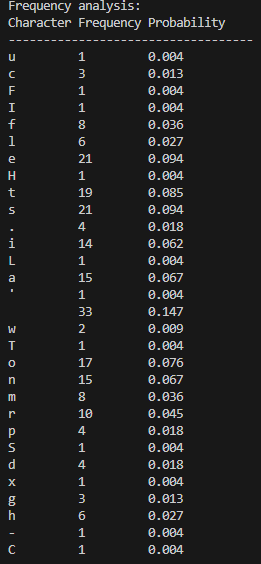


Рисунок 10 – Результат работы программы: Таблица вероятностей

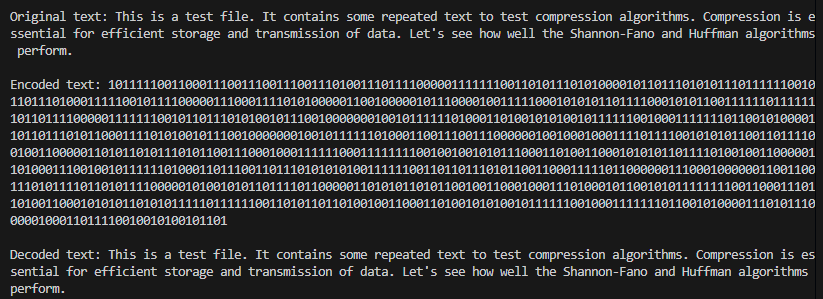


Рисунок 11 – Результат работы программы: Кодировка и декодировка текста

Получаем процент сжатия примерно 52,2%, что меньше, чем при сжатии данных алгоритмом Шеннона-Фано.

# 3 ВЫВОД

В результате выполнения работы были изучены основные алгоритмы сжатия данных, разработаны программы для сжатия и восстановления текста методами Шеннона-Фано и Хаффмана. Проведенный анализ показал эффективность алгоритмов, а также их различия в плане алгоритмической сложности и достигаемого коэффициента сжатия. Можно сделать вывод, что алгоритм Шеннона-Фаноработает эффективней алгоритма Хаффмана. Также в данной работе была решена задача о рюкзаке с использованием метода динамического программирования. Был разработан алгоритм, который позволяет эффективно находить максимальную ценность набора вещей, помещающегося в рюкзак с заданной вместимостью. Для оценки эффективности алгоритма был реализован метод решения данной задачи грубой силой и наглядно продемонстрировано преимущество динамического программирования, особенно на больших данных.

# 4 ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Страуструп Б. Программирование. Принципы и практика с использованием C++. 2-е изд., 2016.

2. Документация по языку С++ [Электронный ресурс]. URL: https://docs.microsoft.com/ruru/cpp/cpp/ (дата обращения 01.09.2021). 3. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. URL: https://online-edu.mirea.ru/course/view.php?id=4020 (дата обращения 01.09.2021).