

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

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)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 8.1**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

Тема: «Алгоритмы кодирования и сжатия данных»

|  |  |  |
| --- | --- | --- |
| Выполнил студент: Лазаренко С.А.  Группа: ИКБО-10-23 |  |  |

Москва 2024

**СОДЕРЖАНИЕ**

[ЦЕЛЬ РАБОТЫ 3](#_Toc178190517)

[ЗАДАНИЕ 1 4](#_Toc178190518)

[Формулировка задачи 1.1 4](#_Toc178190519)

[Описание подхода к решению 4](#_Toc178190520)

[Ход работы 4](#_Toc178190521)

[Формулировка задачи 1.2 5](#_Toc178190519)

[Ход работы 5](#_Toc178190521)

[Формулировка задачи 1.3 6](#_Toc178190519)

[Ход работы 6](#_Toc178190521)

[ЗАДАНИЕ 2 7](#_Toc178190518)

[Формулировка задачи 2.1 7](#_Toc178190519)

[Математическая модель решения 2.1 7](#_Toc178190521)

[Код программы 2.1 9](#_Toc178190521)

[Тестирование программы 2.1 10](#_Toc178190521)

[Формулировка задачи 2.2 11](#_Toc178190519)

[Код программы 2.2 11](#_Toc178190521)

[Тестирование программы 2.2 13](#_Toc178190521)

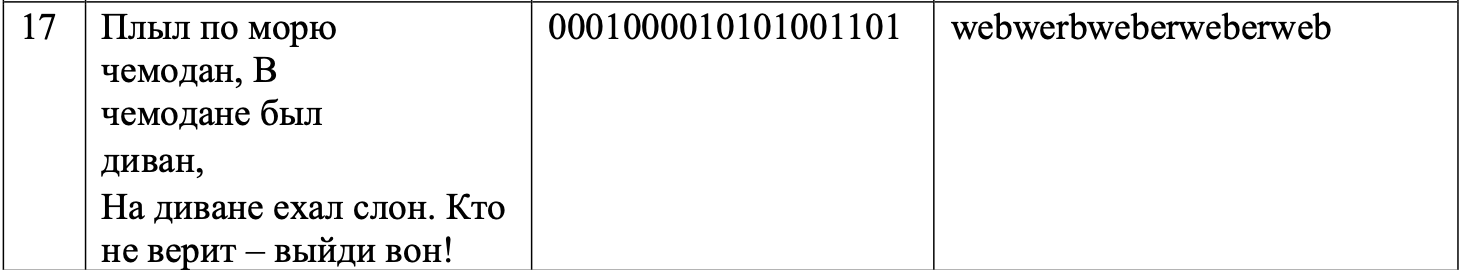
[ВЫВОД 14](#_Toc178190523)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 15](#_Toc178190524)

# ЦЕЛЬ РАБОТЫ

Исследовать и реализовать алгоритмы сжатия данных, а также сравнить их эффективности. В первой части задачи мы рассматриваем сжатие текста с использованием различных методов, таких как алгоритмы Хаффмана и Шеннона-Фано. В ходе выполнения задания будет проанализирован процесс сжатия данных на примерах, для чего каждый алгоритм будет представлен в виде таблицы с указанием результатов сжатия. Также будет описан процесс восстановления сжатого текста, что позволит понять, как происходит декодирование данных и восстановление исходной информации.

В рамках второй части работы планируется разработка программ для сжатия и восстановления текста с использованием методов Хаффмана и Шеннона-Фано.



Оформить отчет в соответствии с требованиями документирования.

# ЗАДАНИЕ 1

### Формулировка задачи 1.1

Разработать решения задач с использованием заданных алгоритмов сжатия и кодирования.

Индивидуальный вариант работы – 17.

### Описание подхода к решению

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

### Ход работы

### 

Рисунок 1 – Таблица решения задачи 1.1

Незакодированная фраза: 89 байта

Закодированная фраза: 392 бит

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

### Формулировка задачи 1.2

Сжатие данных по методу Лемпеля– Зива LZ77.

### Ход работы

### *Таблица 1. - Решение задачи 1.2*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Итерация | Словарь | Буфер | Совпадение | Код |
| 1 |  | 0 | (0, 0, 0) | (0, 0, 0) |
| 2 | 0 | 0 | (0, 0, 0) | (0, 0, 0) |
| 3 | 0, 0 | 1 | (0, 0, 1) | (0, 0, 1) |
| 4 | 0, 0, 1 | 0 | (2, 1, 0) | (2, 1, 0) |
| 5 | 0, 0, 1, 0 | 0 | (1, 0, 0) | (1, 0, 0) |
| 6 | 0, 0, 1, 0, 0 | 1 | (0, 0, 1) | (0, 0, 1) |
| 7 | 0, 0, 1, 0, 0, 1 | 0 | (3, 1, 0) | (3, 1, 0) |
| 8 | 0, 0, 1, 0, 0, 1, 0, | 10 | (4, 1, 0) | (4, 1, 0) |
| 9 | 0, 0, 1, 0, 0, 1, 0, 1 | 0 | (5, 1, 0) | (5, 1, 0) |
| 10 | 0, 0, 1, 0, 0, 1, 0, 1, 0 | 1 | (6, 1, 1) | (6, 1, 1) |
| 11 | 0, 0, 1, 0, 0, 1, 0, 1, 0,1 | 1 | (0, 0, 1) | (0, 0, 1) |

### После завершения процесса кодирования, получаем следующие коды:

1. (0, 0, 0)
2. (0, 0, 0)
3. (0, 0, 1)
4. (2, 1, 0)
5. (1, 0, 0)
6. (0, 0, 1)
7. (3, 1, 0)
8. (4, 1, 0)
9. (5, 1, 0)
10. (6, 1, 1)
11. (0, 0, 1)

### Формулировка задачи 1.3

Закодировать следующую фразу (webwerbweberweberweb), используя код LZ78.

**Ход решения**

### *Таблица 2. - Решение задачи 1.2*

|  |  |  |
| --- | --- | --- |
| Индекс | Строка | Код |
| 1 | w | (0, w) |
| 2 | e | (0, e) |
| 3 | b | (0, b) |
| 4 | we | (1, e) |
| 5 | wer | (2, r) |
| 6 | r | (0, r) |
| 7 | be | (3, e) |
| 8 | we | (1, e) |
| 9 | web | (2, b) |
| 10 | weber | (6, w) |
| 11 | werw | (1, e) |
| 12 | webwer | (2, b) |
| 13 | webwerb | (2, e) |

### После завершения процесса кодирования, получаем следующие коды:

1. (0, 0, 0)
2. (0, 0, 1)
3. (2, 1, 0)
4. (1, 0, 0)
5. (0, 0, 1)
6. (3, 1, 0)
7. (4, 1, 0)
8. (5, 1, 0)
9. (6, 1, 1)
10. (0, 0, 1)

**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.

### Формулировка задачи 2.1

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.

### Математическая модель решения 2.1

Сжатие осуществляется за счет кодирования каждого символа исходного текста уникальным двоичным кодом, сформированным на основе частоты символов. Логика программы начинается с анализа частоты символов в исходном тексте. Сначала функция compressText загружает текст из файла inputFile, подсчитывает частоты каждого символа и сохраняет их в словаре frequencyMap, где ключом является символ, а значением — его частота. Затем все уникальные символы с их частотами добавляются в вектор symbols, где для каждого символа создается объект структуры Symbol, который хранит сам символ, его частоту и код, который будет присвоен в дальнейшем. Далее вектор symbols сортируется по убыванию частот с помощью функции compareByFrequency, которая обеспечивает правильный порядок символов для кодирования.

Процесс кодирования выполняется рекурсивной функцией shannonFanoCoding, которая делит отсортированный массив символов на две группы с примерно равной суммарной частотой символов. Для этого в функции сначала подсчитывается общая частота всех символов в пределах индексов start и end, затем определяется точка разбиения splitPoint, где кумулятивная частота приближается к половине общей. Символам из первой группы присваивается код «0», а символам из второй — «1». Функция рекурсивно вызывает саму себя для каждой группы, пока не останется по одному символу в каждой группе, что завершает процесс кодирования для всех символов. В результате каждого символу в массиве symbols назначается уникальный код.

Затем программа сохраняет сжатый текст в новый файл outputFile. В этом процессе формируется строка compressedText, в которой каждый символ исходного текста заменяется его двоичным кодом из словаря encodingMap, сформированного по завершении shannonFanoCoding. Полученный сжатый текст сохраняется в outputFile в бинарном формате. После этого программа выводит процент сжатия, который вычисляется как отношение сжатого размера файла compressedSize к исходному originalSize, умноженное на 100, и вычтенное из 100%.

Для восстановления исходного текста из сжатого файла используется функция decompressText, которая считывает двоичный текст из файла outputFile и восстанавливает исходные символы, используя словарь decodingMap, построенный на основе encodingMap. Программа считывает сжатый текст, накапливая биты в строку codeBuffer до тех пор, пока эта последовательность битов не будет найдена в decodingMap как соответствующий код какого-либо символа. Когда совпадение найдено, соответствующий символ добавляется в decompressedText, и codeBuffer очищается для накопления следующего кода. В конце восстановленный текст сохраняется в файл output2.txt, что завершает процесс восстановления.

### Код программы 2.1

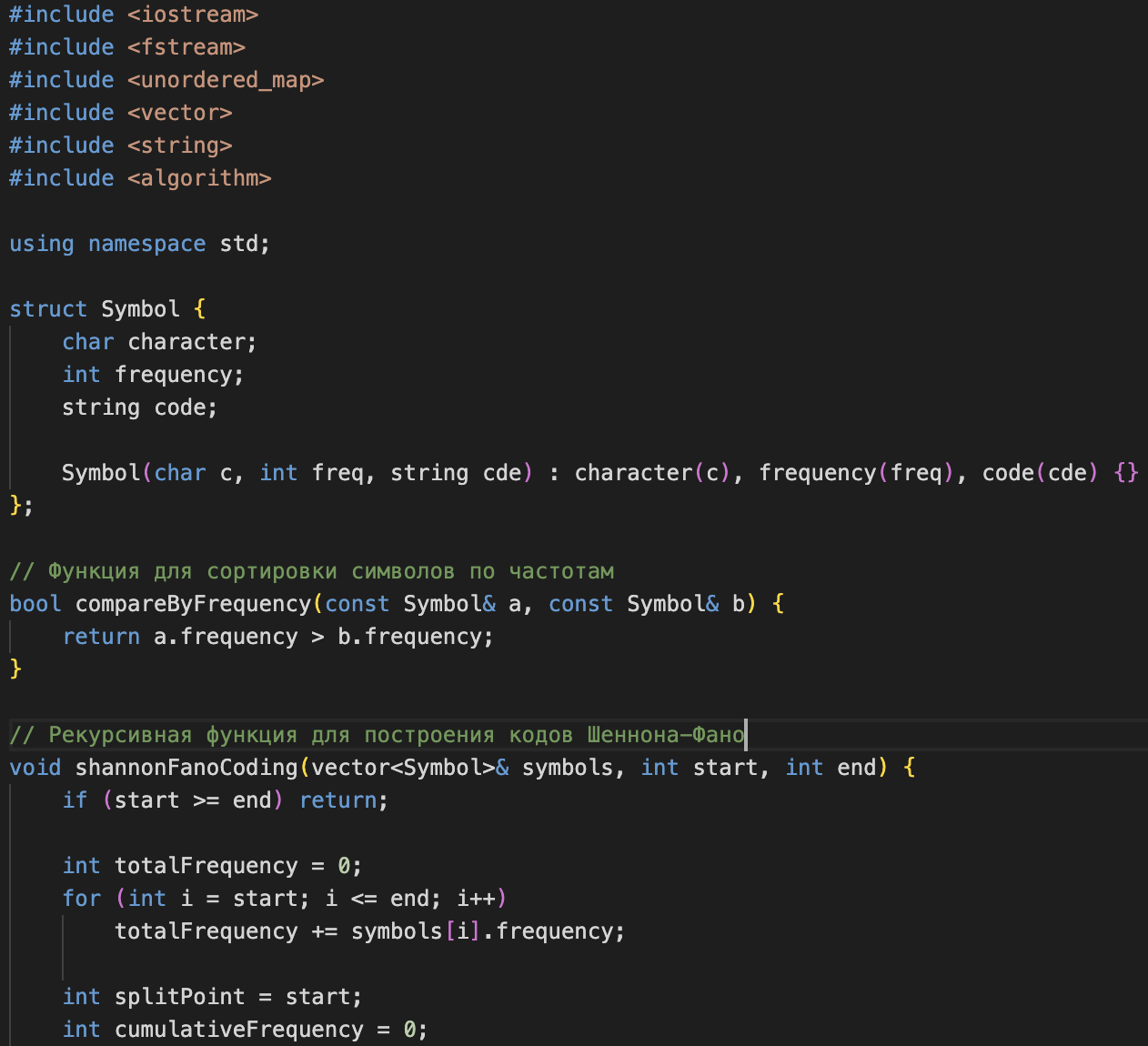


Рисунок 2 – Код программы 2.1 (1 часть)

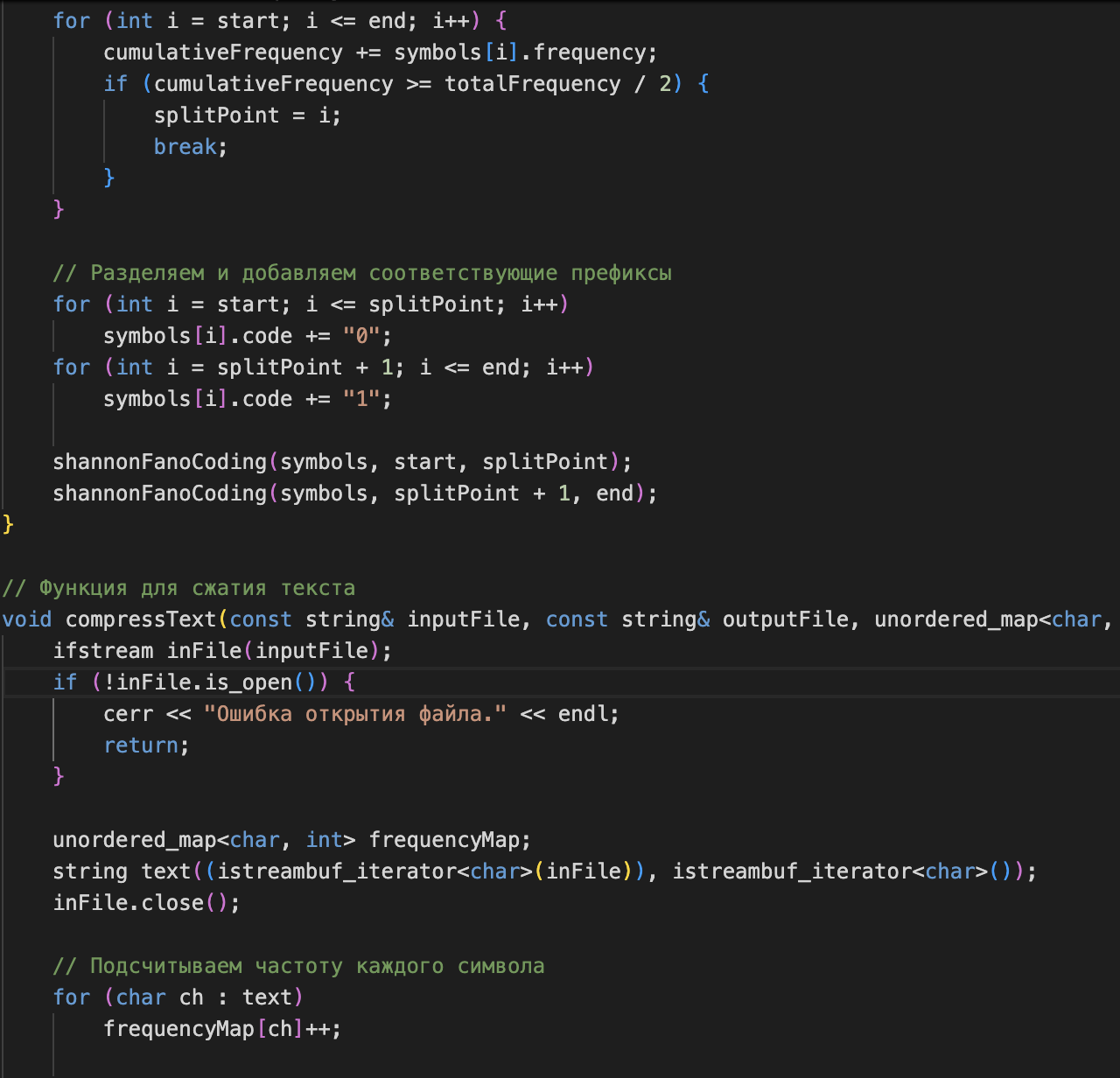


Рисунок 3 – Код программы 2.1 (2 часть)

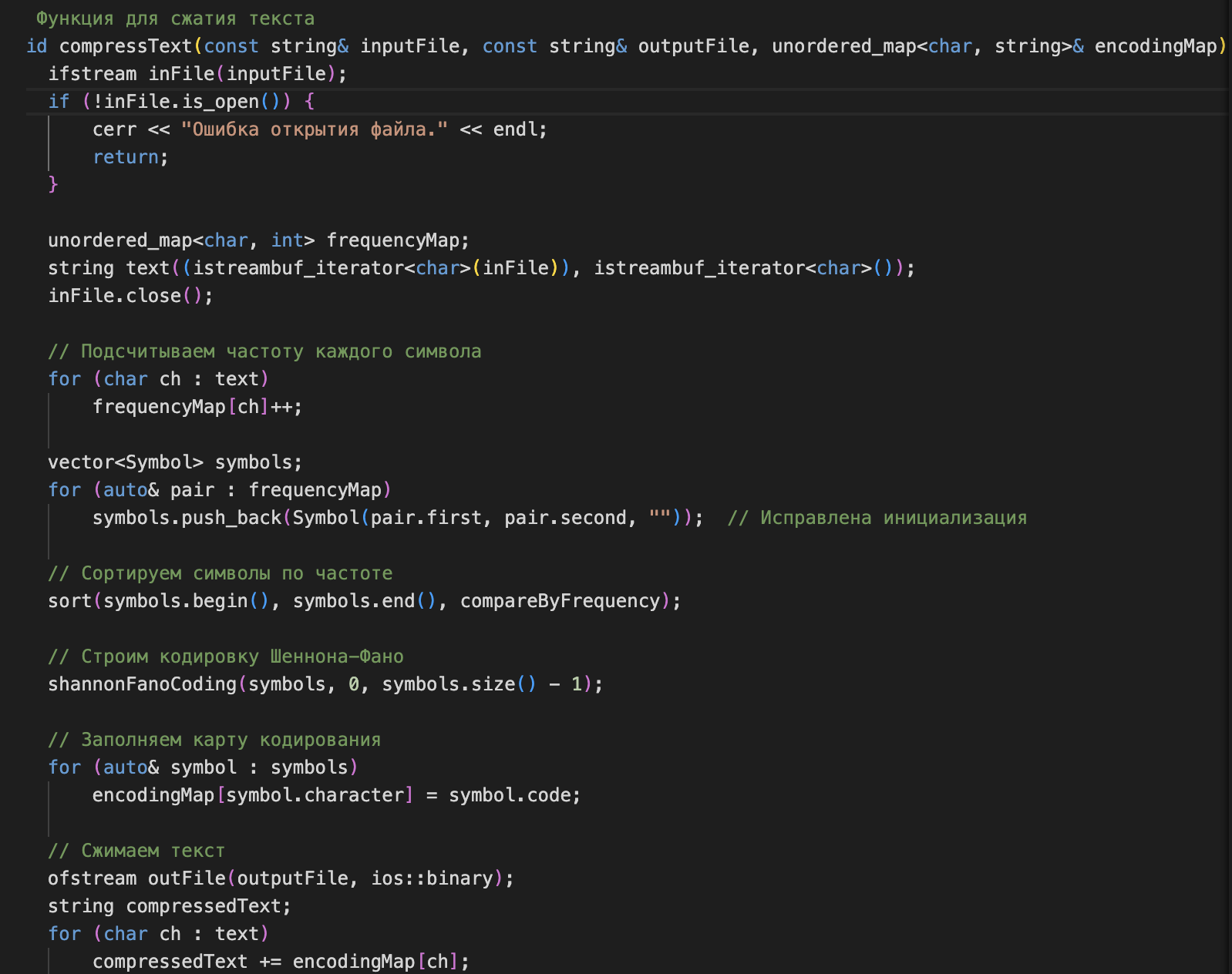


Рисунок 4 – Код программы 2.1 (3 часть)

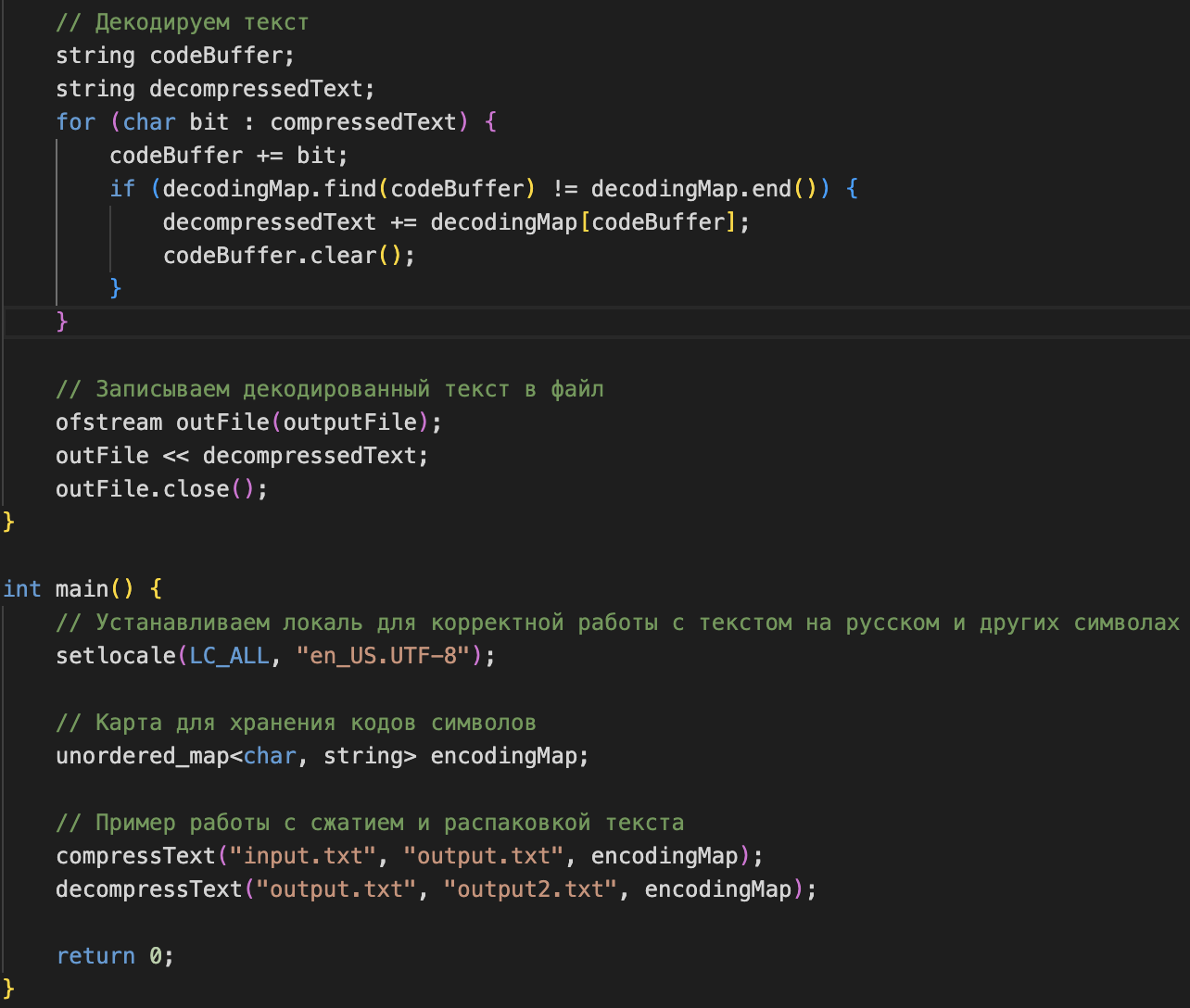
****

Рисунок 5 – Код программы 2.1 (4 часть)

### Тестирование программы 2.1

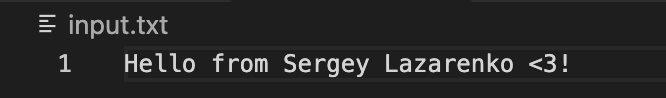
****

Рисунок 6 – Входной файл с текстом

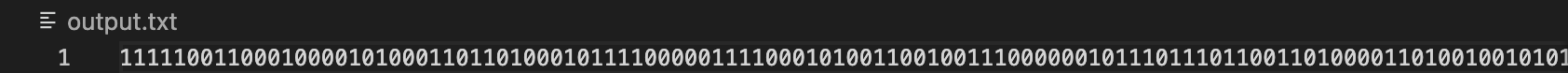
****

Рисунок 7 – Выходной файл с сжатым текстом

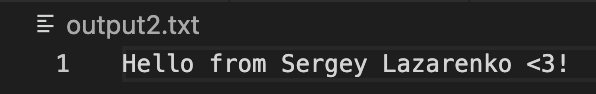


Рисунок 8 – Файл с восстановленным сжатым текстом



Рисунок 9 – Процент сжатия

### Формулировка задачи 2.2

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

### Код программы 2.2

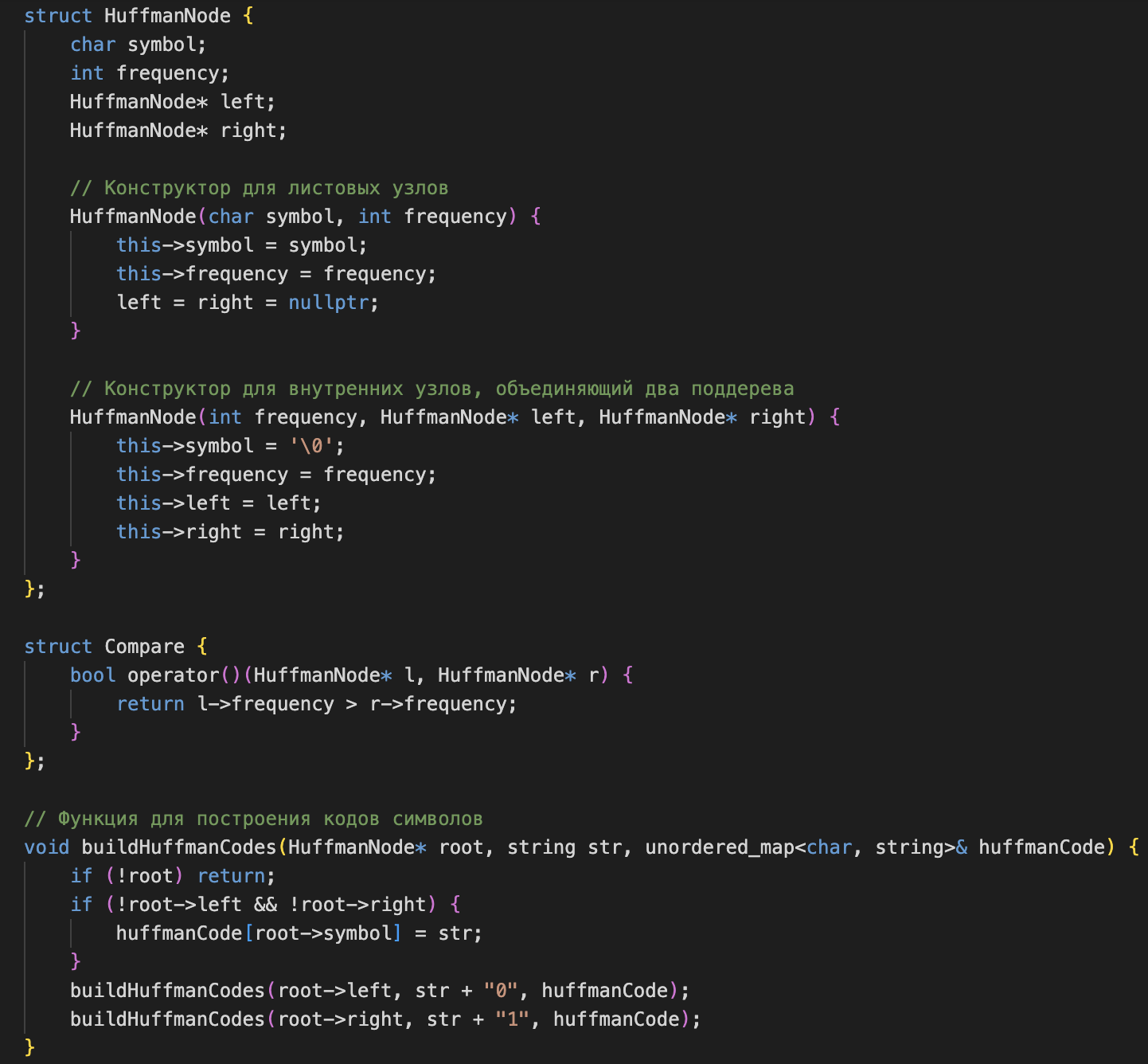


Рисунок 10 – Код программы 2.2 (1 часть)

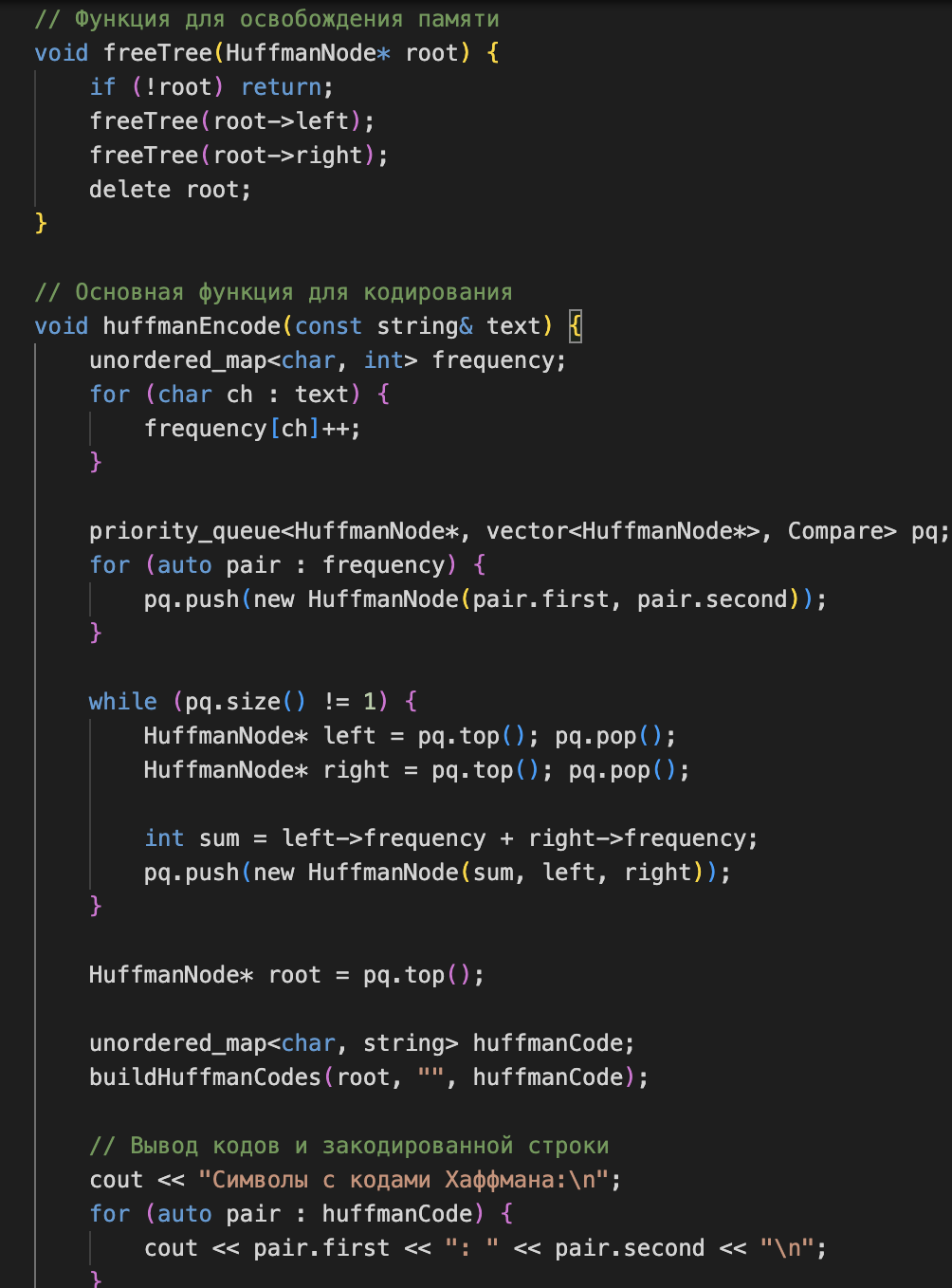


Рисунок 11 – Код программы 2.2 (2 часть)

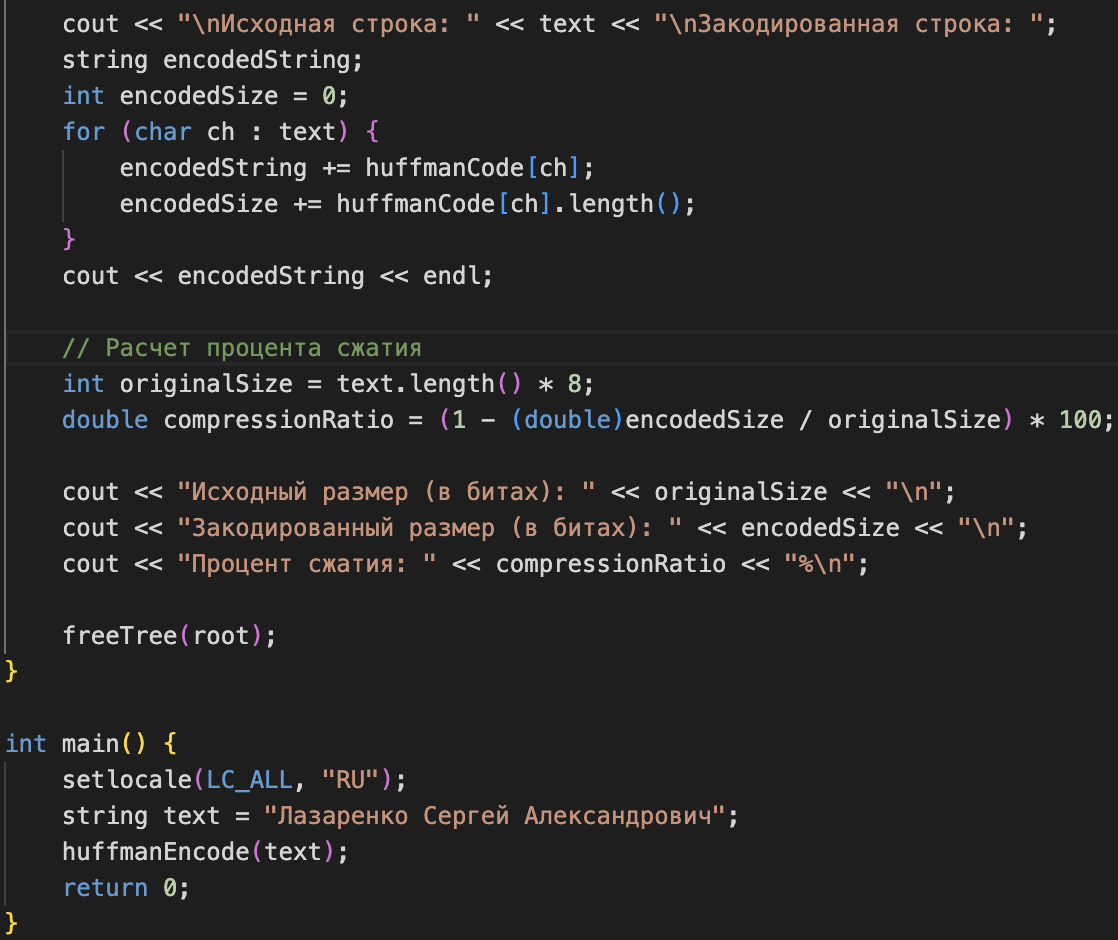


Рисунок 12 – Код программы 2.2 (3 часть)

### Тестирование программы 2.2

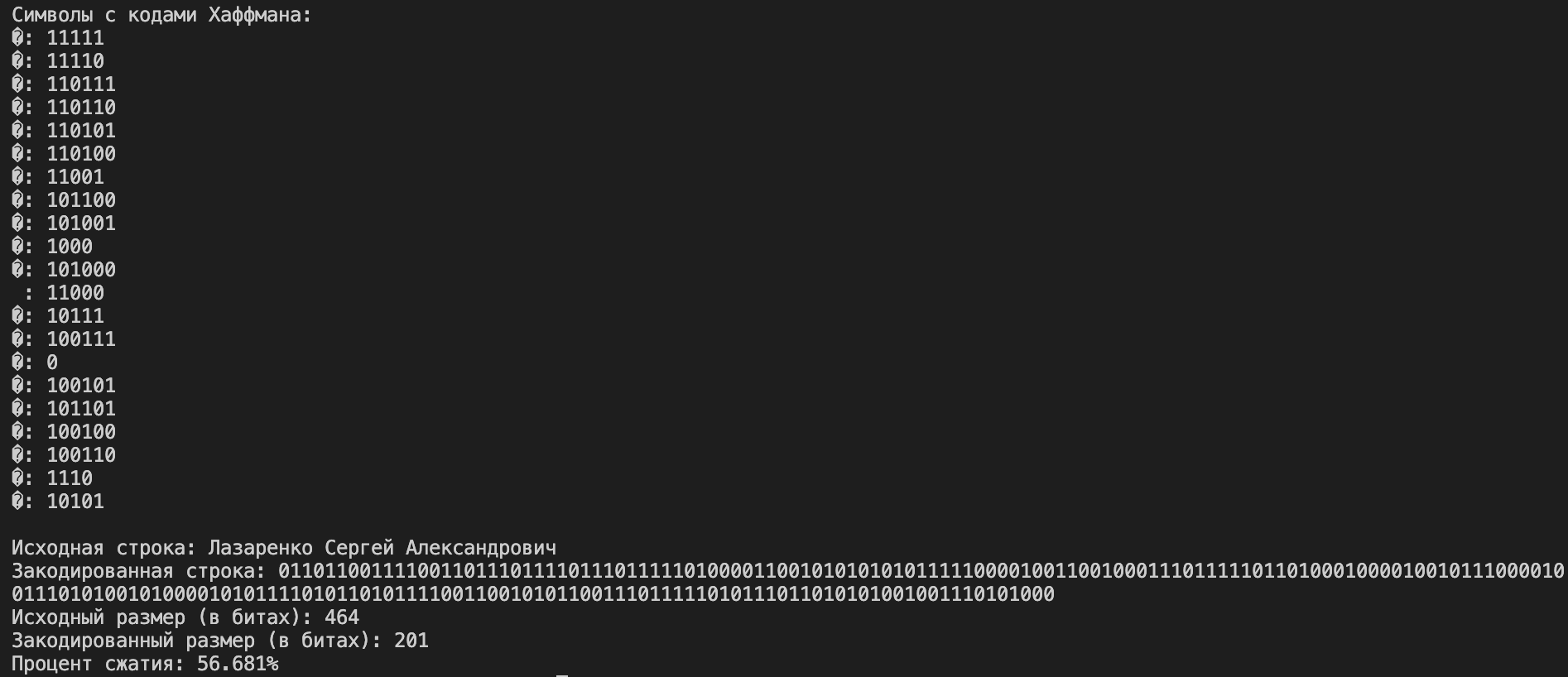
****

Рисунок 13 – Результат тестирования

# ВЫВОД

В результате выполнения работы были освоены навыки по реализации алгоритмов сжатия и восстановления текста методами Хаффмана и Шеннона – Фано, сжатие данных по методу Лемпеля– Зива LZ77.

# СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

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/ru-ru/cpp/cpp/ (дата обращения 08.09.2024).

3. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. https://online-edu.mirea.ru/course/view.php?id=4020 (дата обращения 04.09.2024)