# МИНОБРНАУКИ РОССИИ

# САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

# ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

# «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

# Кафедра САПР

# Отчет

# по лабораторной работе №2

# по дисциплине «Алгоритмы и структуры данных»

# Вариант: 1

|  |  |  |
| --- | --- | --- |
| Студентка гр. 9301 |  | Служевская А.С. |
| Преподаватель |  | Тутуева А.В. |

Санкт-Петербург

2021

**Постановка задачи.**

Задача:

1. Реализовать кодирование и декодирование по алгоритму Хаффмана / Шеннона-Фано (1 и 2 вариант соответственно) входной строки, вводимой через консоль
2. Посчитать объем памяти, который занимает исходная и закодированная строки
3. Выводить на экран таблицу частот и кодов, результат кодирования и декодирования, коэффициент сжатия

**Описание реализуемого класса и методов**

Для реализации кодирования по алгоритму Хаффмана используется класс Haffman. В этом классе есть методы Code и Decode.

Code основан на приоритетной очереди, в которой приоритетом является частота встречаемости символа в строке. После построения очереди символы заносятся в красно-черное дерево и получает свой код.

Decode проходит по дереву и на основе стэка осуществляет декодирование строки.

**Оценка временной сложности алгоритмов:**

Таблица 1. Временные затраты

|  |  |
| --- | --- |
| Huffman() | O(n2) |
| Code() | O(n\*log(n)) |
| Decode() | O(n) |

**Реализованные Unit-тесты:**

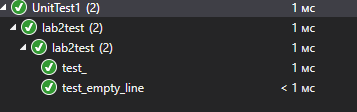


Рисунок 1 — Тесты

1. test\_ — проверяет работу Code и Decode
2. test\_empty\_line — проверяет ошибку при попытке кодирования пустой строки

**Пример работы**

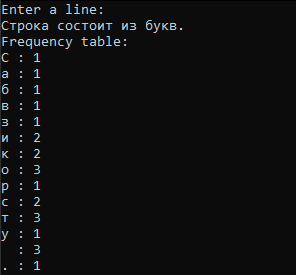
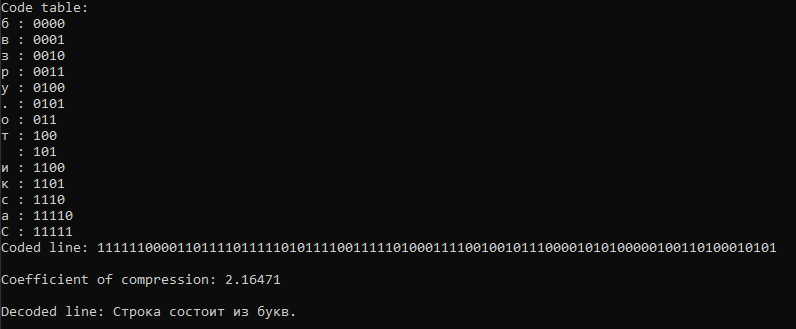
**** ****

Рисунок 2, 3 — Пример работы

**Листинг**

#include <iostream>

#include "Huffman.h"

int main()

{

int line\_size = 400; //max quantity of symbols in input line

char\* str = new char[line\_size]; //array for input\_line

int size; //real size of input line

size = 0;

char symbol;

cout << "Enter a line:" << endl;

do //reading the input line

{

cin >> noskipws >> symbol; //to read with spaces

str[size] = symbol;

size++;

} while ((symbol != '.') && ((size + 1) < line\_size));

str[size] = '\n';

double coefficient = 1.0 \* sizeof(char) \* size \* 8; //volume of the line

Huffman ToCode(str);

int size\_coded; //size of the coded line

bool\* code = ToCode.Code(size); //coded line

char\* decode = ToCode.Decode(size\_coded); //decoded line

cout << "Coded line: ";//output of the coded line

for (size\_t i = 0; i < size; i++)

{

cout << code[i];

}

cout << endl << endl;

coefficient = coefficient / (1 \* size); //calculate compression ratio

cout << "Coefficient of compression: " << coefficient << endl << endl;

cout << "Decoded line: ";

for (size\_t i = 0; i < size\_coded; i++)

{

cout << decode[i];

}

cout << endl;

}