**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

ОТЧЁТ

**по лабораторной работе №2**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: «Алгоритмы кодирования»**

**Вариант 2.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9302 |  | Бабкин И.А. |
| Преподаватель |  | Тутуева А.В. |

Санкт-Петербург

2021

**Постановка задачи**

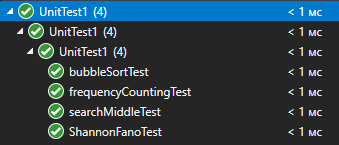
Реализовать кодирование и декодирование по алгоритму Шеннона-Фано входной строки, вводимой через консоль. Посчитать объем памяти, который занимает исходная и закодированная строки. Выводить на экран таблицу частот и кодов, результат кодирования и декодирования, коэффициент сжатия

## Описание и оценка временной сложности функций

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Временная сложность |
| void bubbleSort | Сортировка пузырьком | O(n^2) |
| void frequencyCounting | Подсчет частоты встречаемости символов | O(n) |
| int searchMiddle | Поиск середины | O(n) |
| void ShannonFanoAlgorithm | Алгоритм Шеннона-Фано | O(n^2) |
| void result | Кодирование и декодирование входной строки | O(n^2) |

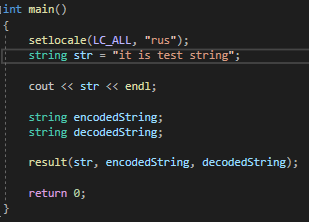
## Описание реализованный unit-тестов

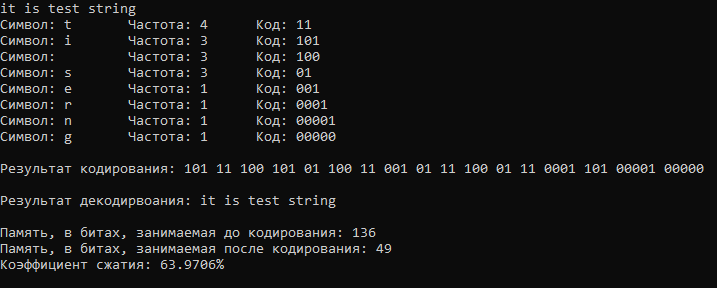
|  |
| --- |
| Название теста |
| bubbleSortTest |
| searchMiddleTest |
| frequencyCountingTest |
| ShannonFanoTest |



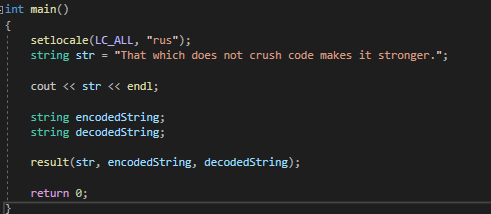
## Пример работы

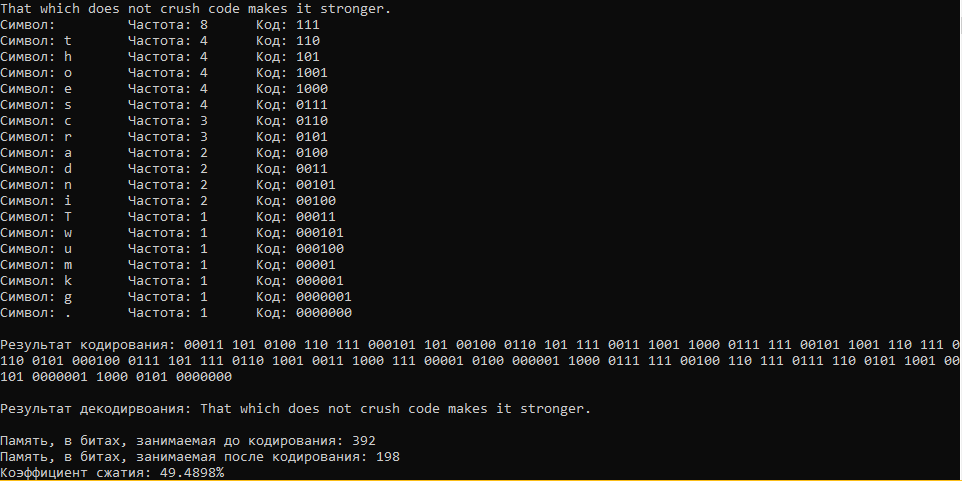
Пример №1:



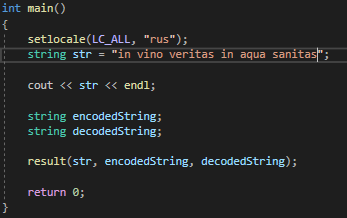


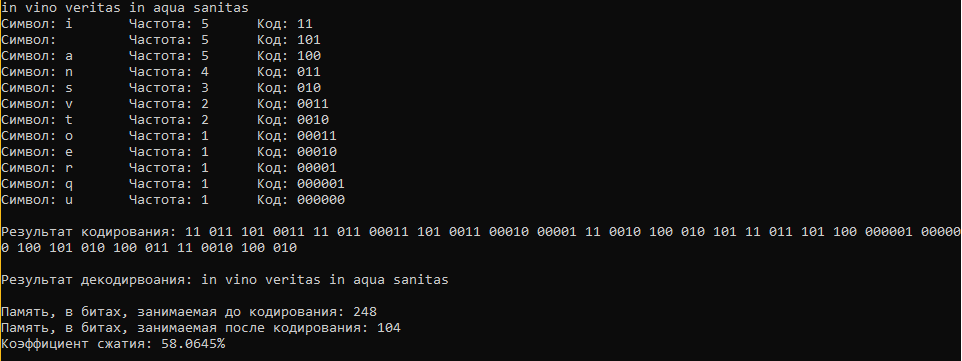
Пример №2:





Пример №3:





## Листинг

#pragma once

#include "RBTree.h"

#include "list.h"

#include <string>

#include <iostream>

using namespace std;

// Сортировка пузырьком

void bubbleSort(LinkedList<char>\* listOfSymbols, LinkedList<int>\* listOfRates, size\_t size)

{

if (size == 0) {

throw("There is no elements");

}

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

{

for (size\_t j = i + 1; j < size; ++j)

{

if (listOfRates->at(i)->data < listOfRates->at(j)->data)

{

listOfRates->swap(i, j);

listOfSymbols->swap(i, j);

}

}

}

}

// Подсчет частоты встречаемости символов

void frequencyCounting(LinkedList<char>\* listOfSymbols, LinkedList<int>\* listOfRates, string str)

{

int flag;

int index;

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

flag = listOfSymbols->find(str[i]);

if (flag == -1) {

listOfSymbols->push\_back(str[i]);

listOfRates->push\_back(1);

}

else {

index = listOfSymbols->find(str[i]);

listOfRates->at(index)->data++;

}

}

}

// Поиск середины

int searchMiddle(LinkedList<int>\* listOfRates, int start, int end)

{

int average = 0;

for (int i = start; i < end; i++) {

average = average + listOfRates->at(i)->data;

}

average = average / 2;

int sum = 0, i = start, middle = i;

while ((sum + listOfRates->at(i)->data < average) && (i < end))

{

sum = sum + listOfRates->at(i)->data;

i++;

middle++;

}

return middle;

}

// Алгоритм Шенона-Фано

void ShannonFanoAlgorithm(LinkedList<char>\* listOfSymbols, LinkedList<int>\* listOfRates, LinkedList<string>\* listOfCodes, int start, int end, RBTree<char, char>\* ShannonFano, string fullCode, string code, unsigned& memoryAfter)

{

int middle;

string tempStr;

tempStr = fullCode + code;

if (start == end) {

memoryAfter += tempStr.size() \* listOfRates->at(start)->data;

ShannonFano->insert(listOfSymbols->at(start)->data, ' ');

cout << "Символ: " << listOfSymbols->at(start)->data << "\tЧастота: " << listOfRates->at(start)->data << "\tКод: " << tempStr << endl;

listOfCodes->push\_back(tempStr);

return;

}

middle = searchMiddle(listOfRates, start, end);

string zero = "0", one = "1";

ShannonFanoAlgorithm(listOfSymbols, listOfRates, listOfCodes, start, middle, ShannonFano, tempStr, one, memoryAfter);

ShannonFanoAlgorithm(listOfSymbols, listOfRates, listOfCodes, middle + 1, end, ShannonFano, tempStr, zero, memoryAfter);

}

void result(string& str, string& encodedString, string& decodedString)

{

LinkedList<char>\* listOfSymbols = new LinkedList<char>;

LinkedList<int>\* listOfRates = new LinkedList<int>;

LinkedList<string>\* listOfCodes = new LinkedList<string>;

unsigned memoryBefore = str.size() \* 8, memoryAfter = 0;

// Кодирование

frequencyCounting(listOfSymbols, listOfRates, str);

bubbleSort(listOfSymbols, listOfRates, listOfSymbols->get\_size());

RBTree<char, char> ShannonFano;

string temp = "";

ShannonFanoAlgorithm(listOfSymbols, listOfRates, listOfCodes, 0, listOfRates->get\_size() - 1, &ShannonFano, temp, temp, memoryAfter);

int index;

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

index = listOfSymbols->find(str[i]);

encodedString += listOfCodes->at(index)->data + " ";

}

// Декодирование

string code;

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

code = "";

while (encodedString[i] != ' ') {

code += encodedString[i++];

}

index = listOfCodes->find(code);

decodedString += listOfSymbols->at(index)->data;

}

cout << endl;

cout << "Результат кодирования: " << encodedString << endl << endl;

cout << "Результат декодирования: " << decodedString << endl << endl;

cout << "Память, в битах, занимаемая до кодирования: " << memoryBefore << endl;

cout << "Память, в битах, занимаемая после кодирования: " << memoryAfter << endl;

cout << "Коэффициент сжатия: " << (1 - (float)memoryAfter / (float)memoryBefore) \* 100 << "%" << endl;

}