МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
МО ЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Алгоритмы и структуры данных»
"Алгоритмы кодирования"
Вариант 2

Студент гр. 9302 _____ Кузнецов В.А.
Преподаватель _____ Тутуева А.В.

Санкт-Петербург
2021

Цель работы.

Реализовать кодирование/декодирование по алгоритму Шеннона-Фано.

Постановка задачи.

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
4. Стандартные структуры данных C++ использовать нельзя. Необходимо использовать структуры данных из предыдущих лабораторных работ.

Описание реализуемого алгоритма, класса и методов.

Алгоритму подается на вход текст. Далее считается частота появления каждого символа в тексте, формируется строка из уникальных символов в порядке уменьшения частоты их появления, а далее на каждой итерации кодирования происходит разделение строки на две части так, чтобы разница сумм частот символов в левой части н правой части была минимальной. Левая часть строки маркируется 1, а правая 0, и так до тех пор, пока все строки не разобьются на отдельные символы. Получившееся последовательность из нулей и единиц каждого символа и есть его код, который записывается в ассоциированный массив на основе обычного массива, у которого индекс – это ASCII код символа, а значение – это его шифр.

Декодирование реализовано таким образом: на вход подается зашифрованная строка и ассоциированный массив, где его индексы – это ASCII код символа, а значение – его шифр. Алгоритм проходится по индексам массива и находит первое совпадение с началом кодируемой строки и заменяет начало на соответствующий символ, далее начинает проход со

следующего недекодированного символа, и так до конца закодированной строки.

Для этого был реализован класс FanoAlgorithm со следующими полями:

1. string str – исходная строка, которую нужно закодировать;
2. string encodedStr – закодированная строка;
3. string decodedStr – декодированная строка;
4. size_t charsFreq[SIZE] – ассоциированный массив, таблица частот каждого символа, где SIZE = 256;
5. string charsCode[SIZE] – таблица шифров каждого символа.

Методы класса:

1. size_t getCharFreq(char c) – получить частоту появления символа c в строке str. Возвращает charsFreq[(size_t)c];
2. string getCharCode(char c) – получить шифр символа c. Возвращает charsCode[(size_t)c];
3. int getStrFreq(string str) – возвращает сумму частот появления символов в исходной строке из строки str;
4. void encodeCharsIteration(string str, string code) – одна итерация алгоритма, где строка str разбивается на две части с примерно равными суммами частот символов и функция рекурсивно вызывается уже для левой и правой части строки, с добавлением ‘1’ или ‘0’ к code соответственно, пока str не станет символом.
5. void sortByFreq(string& str) – сортировка символов строки str в порядке убывания их частот;
6. void encode() – реализация алгоритма кодирования;
7. void decode() – реализация алгоритма декодирования;
8. string getEncodedStr() – возвращает закодированную строку;
9. string getDecodedStr() – возвращает декодированную строку;
10. void printCharsEncoding() – печатает в консоли уникальные символы текста, их частоту и шифр;

11. size_t initStrSizeInBites() – возвращает объем памяти, занимаемый исходным текстом (в битах)
12. size_t encodeStrSizeInBites() – возвращает объем памяти, занимаемый закодированным текстом (в битах)
13. double compressionRatio() – возвращает коэффициент сжатия.

Оценка временной сложности алгоритмов.

Алгоритм Шеннона-Фано	Оценка
Кодирование	O(n)
Декодирование	$O(n^2)$

Описание реализованных unit-тестов.

В классе тестирования UnitTest1 были реализованы следующие объекты, на которых проводилось тестирование методов:

1. FanoAlgorithm* f1 = new FanoAlgorithm("it is test string");
2. FanoAlgorithm* f2 = new FanoAlgorithm("Algorithms and data structures");
3. FanoAlgorithm* f3 = new FanoAlgorithm("ab");

Реализованные тесты:

1. TestEncoding – тестирование функций кодирования;
2. TestDecoding – тестирование функции декодирования;
3. TestSize – тестирование функций вычисления объема памяти;

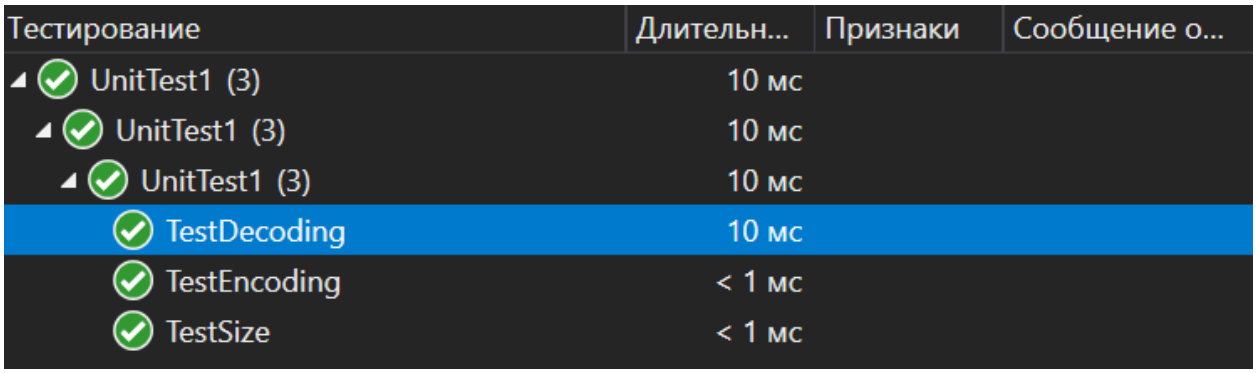

Рисунок 1 – Результаты тестирования

Пример работы программы.

Пример 1.

Make a lot of walks to get healthy and don't read that much but save yourself some until you're grown up.

```
#include <iostream>
#include "FanoAlgorithm.h"

int main()
{
    string str = "Make a lot of walks to get healthy and don't read that much but save yourself some until you're grown up.";
    FanoAlgorithm* f = new FanoAlgorithm(str);
    cout << "Initial string: " << str << endl;
    f->printCharsEncoding();
    cout << "Encoded string: " << f->getEncodedStr() << endl;
    cout << "Initial string memory: " << f->initStrSizeInBites() << " bites" << endl;
    cout << "Encoded string memory: " << f->encodedStrSizeInBites() << " bites" << endl;
    cout << "Compression ratio: " << f->compressionRatio() << endl;
    cout << "Decoded string: " << f->getDecodedStr() << endl;
    return 0;
}
```

```
Консоль отладки Microsoft Visual Studio
Initial string: Make a lot of walks to get healthy and don't read that much but save yourself some until you're grown up.
Char    Freq    Code
        20      111
'       2       00100
.       1       000000
M       1       000100
a       8       1011
b       1       0000110
c       1       0000111
d       3       00110
e       8       100
f       2       000111
g       2       001010
h       4       01100
i       1       0000011
k       2       001011
l       5       01101
m       2       000110
n       4       01011
o       8       1010
p       1       0000010
r       4       01010
s       4       0100
t       9       110
u       6       0111
v       1       000010
w       2       000101
y       3       00111
Encoded string: 000100101100101110011110111011010110111010000111111000101101101101001011010011111010101110010101001101110110010010110111011001100001111
1110110101100110111001101010010110010011011101010100101100111011111001100101011110110001100111000011101100111000011001111101110010010110000101001110011110100111
0101001001000110100011111101001010000110100111011101011100000011011011110011101001110010100101010100001010101111101110000010000000
Initial string memory: 840 bites
Encoded string memory: 446 bites
Compression ratio: 1.88341
Decoded string: Make a lot of walks to get healthy and don't read that much but save yourself some until you're grown up.
```

Рисунки 2-3 – Демонстрация работы

Пример 2.

The supreme task of the physicist is to arrive at those universal elementary laws from which the cosmos can be built up by pure deduction. There is no logical path to these laws; only intuition, resting on sympathetic understanding of experience, can reach them.

Рисунок 4 – Демонстрация работы

Пример 3.

If A is success in life, then A = x + y + z. Work is x, play is y and z is keeping your mouth shut.

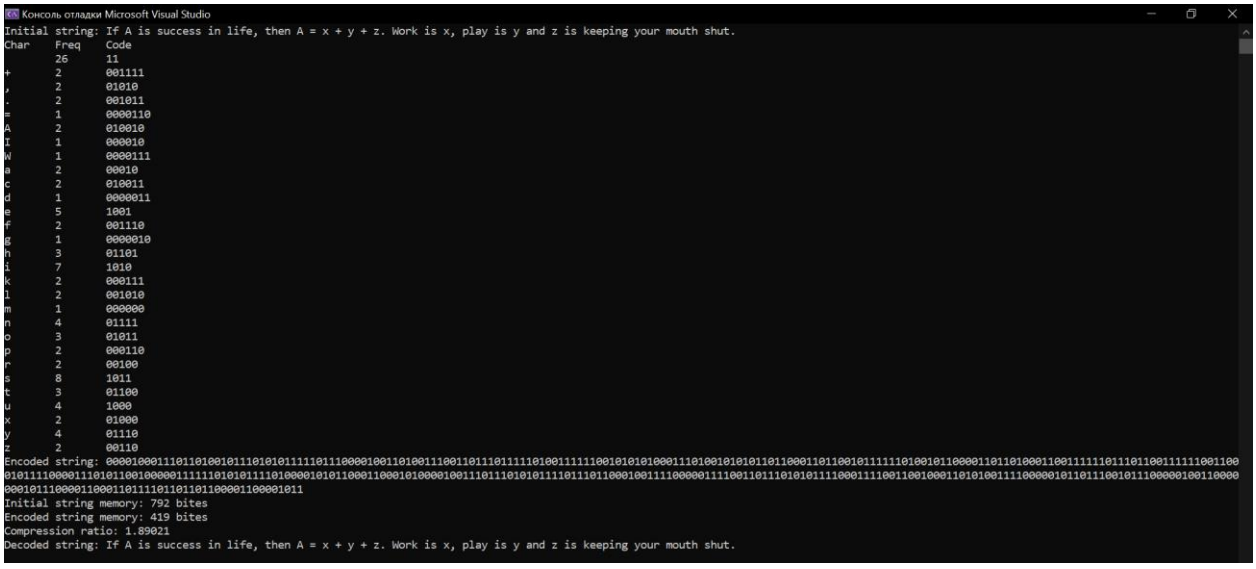

Рисунок 5 – Демонстрация работы

ЛИСТИНГ

FanoAlgorithm.h

```
#pragma once
#include <iostream>
#include <string>
#include <math.h>

using namespace std;

constexpr auto SIZE = 256;

class FanoAlgorithm
{
	string str;
	string encodedStr;
	string decodedStr;
	size_t charsFreq[SIZE] = { 0 };
	string charsCode[SIZE] = { "" };

	size_t getCharFreq(char c) {
		return this->charsFreq[(size_t)c];
	}

	string getCharCode(char c) {
		return this->charsCode[(size_t)c];
	}

	int getStrFreq(string str) {
		int freq = 0;

		for (size_t i = 0; i < str.length(); i++) {
			freq += getCharFreq(str[i]);
		}
		return freq;
	}

	void encodeCharsIteration(string str, string code) {
		if (str.length() > 1) {
			string leftStr = "";
			int leftFreq = 0;
			int strFreq = getStrFreq(str);
			size_t i;

			for (i = 0; i < str.length() - 1; i++) {
				int freq = getCharFreq(str[i]);
				int nextFreq = getCharFreq(str[i + 1]);
				leftFreq += freq;
				leftStr += str[i];
				if (abs(strFreq - 2 * leftFreq) < abs(strFreq - 2 * (leftFreq
+ nextFreq))) {
					break;
				}
			}

			encodeCharsIteration(leftStr, code + '1');
			encodeCharsIteration(str.substr(i + 1, str.length() - i), code +
'0');
		}
		else {
			this->charsCode[(size_t)str[0]] = code;
		}
	}
```

```
        void sortByFreq(string& str) {
                for (size_t i = 0; i < str.length(); i++) {
                        for (size_t j = i + 1; j < str.length(); j++) {
                                if (this->getCharFreq(str[i]) < this->getCharFreq(str[j])) {
                                        char t = str[i];
                                        str[i] = str[j];
                                        str[j] = t;
                                }
                        }
                }
        }

        void encode() {
                string uniq = "";
                for (size_t i = 0; i < str.length(); i++) {
                        if (getCharFreq(str[i]) == 0) {
                                uniq += str[i];
                        }
                        this->charsFreq[(size_t)str[i]] += 1;
                }
                this->sortByFreq(uniq);
                encodeCharsIteration(uniq, "");
                for (size_t i = 0; i < this->str.length(); i++) {
                        this->encodedStr += this->getCharCode(str[i]);
                }
        }

        void decode() {
                this->decodedStr = this->encodedStr;
                size_t length = this->decodedStr.length();
                for (size_t pos = 0; pos < length; pos++) {
                        for (size_t i = 0; i < 256; i++) {
                                char c = i;
                                string code = this->getCharCode(c);
                                if (!code.empty()) {
                                        size_t currPos = this->decodedStr.find(code, pos);
                                        if (currPos == pos) {
                                                string s = string(1, c);
                                                this->decodedStr.replace(pos, code.length(), s);
                                                length = this->decodedStr.length();
                                                break;
                                        }
                                }
                        }
                }
        }

public:
        FanoAlgorithm(string str) {
                this->str = str;
                this->encodedStr = "";
                this->decodedStr = "";
        }

        string getEncodedStr() {
                if (this->encodedStr.empty()) {
                        this->encode();
                }
                return this->encodedStr;
        }

        string getDecodedStr() {
                if (this->encodedStr.empty()) {
                        this->encode();
                }
```

```
		if (this->decodedStr.empty()) {
			this->decode();
		}
		return this->decodedStr;
	}

	void printCharsEncoding() {
		if (this->encodedStr.empty()) {
			this->encode();
		}
		cout << "Char\tFreq\tCode" << endl;
		for (size_t i = 0; i < 256; i++) {
			if (!this->charsCode[i].empty()) {
				char c = i;
				cout << c << '\t' << this->getCharFreq(c) << '\t' << this-
>getCharCode(c) << endl;
			}
		}
	}

	size_t initStrSizeInBites() {
		return this->str.length() * 8;
	}

	size_t encodedStrSizeInBites() {
		if (this->encodedStr.empty()) {
			this->encode();
		}
		size_t size = 0;
		for (size_t i = 0; i < this->str.length(); i++) {
			size += this->getCharCode(str[i]).length();
		}
		return size;
	}

	double compressionRatio() {
		if (this->encodedStr.empty()) {
			this->encode();
		}
		return  (double)this->initStrSizeInBites() / (double)this-
>encodedStrSizeInBites();
	}
};
```

UnitTest1.cpp

```
#include "pch.h"
#include "CppUnitTest.h"
#include "../Lab2_Kuznetsov/FanoAlgorithm.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest1
{
	TEST_CLASS(UnitTest1)
	{
	public:
		FanoAlgorithm* f1 = new FanoAlgorithm("it is test string");
		FanoAlgorithm* f2 = new FanoAlgorithm("Algorithms and data structures");
		FanoAlgorithm* f3 = new FanoAlgorithm("ab");
		TEST_METHOD(TestEncoding)
		{
```

```
	Assert::AreEqual((string)"10111100101011100110100111110001111001110100100000", f1->getEncodedStr());

	Assert::AreEqual((string)"00110000110100001111100010011101011010101011101010000010101111010011110011110010101011111110011000010111011011000001011", f2->getEncodedStr());
		Assert::AreEqual((string)"10", f3->getEncodedStr());
	}
	TEST_METHOD(TestDecoding) {
		Assert::AreEqual((string)"it is test string", f1->getDecodedStr());
		Assert::AreEqual((string)"Algorithms and data structures", f2->getDecodedStr());
		Assert::AreEqual((string)"ab", f3->getDecodedStr());
	}
	TEST_METHOD(TestSize) {
		Assert::AreEqual(136, (int)f1->initStrSizeInBites());
		Assert::AreEqual(240, (int)f2->initStrSizeInBites());
		Assert::AreEqual(16, (int)f3->initStrSizeInBites());
		Assert::AreEqual(49, (int)f1->encodedStrSizeInBites());
		Assert::AreEqual(118, (int)f2->encodedStrSizeInBites());
		Assert::AreEqual(2, (int)f3->encodedStrSizeInBites());
	}
};
}
```