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 2
ТЕМА: АЛГОРИТМЫ КОДИРОВАНИЯ

Студент гр. 0309 _____ Головатюк К.А.

Преподаватель _____ Тутуева А.В

Санкт-Петербург
2022

Постановка задачи

1. Реализовать кодирование и декодирование по алгоритму Шеннона-Фано (2 вариант) 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.

Наличие unit-тестов является обязательным требованием

Описание реализованных методов и оценка их временной сложности

Class ShenonFano
n – количество уникальных символов в строке, m – длинна строки.

Название метода	Описание	Оценка временной сложности
void create_table()	Создание таблицы	O(m + 2^(n-1))
string code()	Кодирование строки	O(m * n)
string decode()	Декодирование строки	O(m * n)
void print_table()	Вывод в консоль	O(n)

Class List
n – размер списка.

Название метода	Описание	Оценка временной сложности
void push(char c)	Добавление элемента в список	O(n)
void sort()	Сортировка списка	O(n^2)
string search(char c)	Поиск элемента по символу, возвращает код элемента	O(n)

Описание алгоритма Шеннона — Фано и структур данных

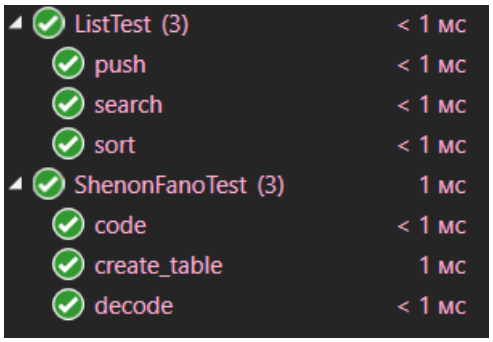
1. Символы сортируются по убыванию частоты.
2. Символы делятся на 2 группы, с примерной одинаковой частотой.
3. 1 – группе присваивается ‘0’, 2 – группе присваивается ‘1’.
4. Группы рекурсивно делятся, и им присваиваются двоичные цифры
5. Если группа состоит из 1 элемента, этот элемент становится листом бинарного дерева, который хранит в себе код, символ и частоту символа.

Алгоритм является не оптимальным, так для одной строки можно построить несколько кодов, и они будут отличаться по сжатию, но если построить все возможные коды, то среде них будут оптимальные.

Были использованы такие структуры как, бинарное дерево и список. Список хранит в себе набор уникальных символов строки и их частоту. Бинарное дерево хранит в себе закодированные символы, код элемента является путем до этого элемента в бинарном дереве, такая структура позволяет эффективно раскодировать строку.

Описание реализованных unit-тестов

Список тестов:



```
[==========] Running 6 tests from 2 test cases.
[----------] Global test environment set-up.
[----------] 3 tests from ShenonFanoTest
[ RUN      ] ShenonFanoTest.create_table
[    String] asdfsagasdklasfasaasfdaafas
[      Code] 000111010010011100001110111101111100011000010000011011000001000001
[       OK ] ShenonFanoTest.create_table (0 ms)
[ RUN      ] ShenonFanoTest.code
[    String] asdfsagasdklasfasaasfdaafas
[      Code] 000111010010011100001110111101111100011000010000011011000001000001
[       OK ] ShenonFanoTest.code (0 ms)
[ RUN      ] ShenonFanoTest.decode
[    String] asdfsagasdklasfasaasfdaafas
[      Code] 000111010010011100001110111101111100011000010000011011000001000001
[       OK ] ShenonFanoTest.decode (1 ms)
[----------] 3 tests from ShenonFanoTest (1 ms total)

[----------] 3 tests from ListTest
[ RUN      ] ListTest.push
[    Symbols] 1 2 3 4 5 6
[      Freq] 1 1 1 5 1 1
[       OK ] ListTest.push (0 ms)
[ RUN      ] ListTest.sort
[    Symbols] 4 1 2 3 5 6
[      Freq] 5 1 1 1 1 1
[       OK ] ListTest.sort (0 ms)
[ RUN      ] ListTest.search
[    Symbols] 4 1 2 3 5 6
[      Freq] 5 1 1 1 1 1
[       OK ] ListTest.search (1 ms)
[----------] 3 tests from ListTest (2 ms total)

[----------] Global test environment tear-down
[==========] 6 tests from 2 test cases ran. (5 ms total)
[  PASSED  ] 6 tests.
```

Пример работы

```
Научить человека быть счастливым – нельзя, но воспитать его так, чтобы он был счастливым, можно.
  – 000 frequency = 14
т – 0010 frequency = 8
о – 0011 frequency = 8
а – 0100 frequency = 6
ч – 0101 frequency = 5
л – 0110 frequency = 5
ы – 0111 frequency = 5
с – 1000 frequency = 5
и – 10010 frequency = 4
ь – 10011 frequency = 4
е – 1010 frequency = 4
е – 1010 frequency = 4
в – 10110 frequency = 4
е – 1010 frequency = 4
в – 10110 frequency = 4
н – 10111 frequency = 4
б – 1100 frequency = 3
м – 11010 frequency = 3
, – 11011 frequency = 3
к – 11100 frequency = 2
Н – 111010 frequency = 1
у – 1110110 frequency = 1
– – 1110111 frequency = 1
з – 111100 frequency = 1
я – 111101 frequency = 1
п – 111110 frequency = 1
г – 1111110 frequency = 1
ж – 11111110 frequency = 1
. – 11111111 frequency = 1
code – 1110100100111011001011001000101001100001011010011000111011010101110001000001100011100101001100010000
1010100100000100110100101011001111101000011101110001011110100110100111111001111011101100010111001100010110
0111000111110100100010010000101001100010101111110001100000100100111001101100001010010001111000111000001110
1100011000111011000010000101010010000010011010010101100111110101101100011010001111111110101110011111111111
decode – Научить человека быть счастливым – нельзя, но воспитать его так, чтобы он был счастливым, можно.
size code string = 419
size string = 768
compression ratio = 1.83294
```

```
Времени нет. Серьезно? Это желания нет, а время есть всегда.
е – 000 frequency = 10
  – 001 frequency = 9
н – 010 frequency = 5
т – 0110 frequency = 4
р – 0111 frequency = 3
а – 1000 frequency = 3
м – 10010 frequency = 2
и – 10011 frequency = 2
. – 10100 frequency = 2
ь – 10101 frequency = 2
о – 10110 frequency = 2
я – 10111 frequency = 2
в – 1100 frequency = 2
с – 11010 frequency = 2
В – 110110 frequency = 1
С – 110111 frequency = 1
з – 11100 frequency = 1
? – 111010 frequency = 1
Э – 111011 frequency = 1
ж – 111100 frequency = 1
л – 111101 frequency = 1
, – 111110 frequency = 1
г – 1111110 frequency = 1
д – 1111111 frequency = 1
code – 1101100111000100100000101001100101000001101010000111011100001111010100011100010101
0110111010001111011011010110001111100000111101100001010011101110010100000110111110001100
0000111000111000100101011100100011010011010101001110011010000111111011111111100010100
decode – Времени нет. Серьезно? Это желания нет, а время есть всегда.
size code string = 251
size string = 480
compression ratio = 1.91235
```

```
В природе противоположные причины часто производят одинаковые действия: лошадь равно па
дает на ноги от застоя и от излишней езды.
  - 000 frequency = 18
о - 001 frequency = 16
и - 0100 frequency = 11
т - 0101 frequency = 8
а - 0110 frequency = 8
д - 0111 frequency = 7
е - 1000 frequency = 7
н - 1001 frequency = 7
п - 1010 frequency = 6
р - 10110 frequency = 6
в - 10111 frequency = 5
ы - 11000 frequency = 4
з - 11001 frequency = 4
л - 11010 frequency = 3
с - 11011 frequency = 3
я - 11100 frequency = 3
ч - 111010 frequency = 2
й - 111011 frequency = 2
ш - 111100 frequency = 2
В - 1111010 frequency = 1
ж - 1111011 frequency = 1
к - 1111100 frequency = 1
: - 1111101 frequency = 1
ь - 1111110 frequency = 1
г - 11111110 frequency = 1
. - 11111111 frequency = 1
code - 1111010000101010110010010110001011110000001010101100010101010010111001101000111
1000111110111001110001000000101010110010011101001001001110000001110100110110110101001000
0101010110001010011001101110010111111000101000001011101001001011011111000011011111000100
0000001111000111011110110101101110100111001111101000110100011111000110011111111110000101
1001101011110010010001010011001110110100001010001001011000010010011111111001000000001010
1000110010110110110101001111000000100000001010100001001100111010010011110010011000111010
1000100011001011111100011111111
decode - В природе противоположные причины часто производят одинаковые действия: лошадь
 равно падает на ноги от застоя и от излишней езды.
size code string = 545
size string = 1032
compression ratio = 1.89358
```

Вывод

При реализации были улучшены навыки создания таких структур данных как, списки и бинарные деревья. Получены навыки кодирования и декодирования. Изучен алгоритм Шеннона-Фано.

Листинг

Файл main.cpp:

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

using namespace std;

int main()
{
	string str = "";
	getline(cin, str);

	ShenonFano test(str);
	test.create_table();
	test.print_table();

	cout << "code - " << test.code() << endl;
	cout << "decode - " << test.decode() << endl;

	cout << "size code string = " << test.take_code().size() << endl;
	cout << "size string = " << test.take_decode().size() * 8 << endl;

	cout << "compression ratio = " << float(test.take_decode().size() * 8) /
float(test.take_code().size()) << endl;
}
```

Файл ShenonFano.h:

```
#pragma once
#include "List.h"
#include <iostream>

using namespace std;

class NodeTree {
public:
	string symb;
	string code;
	int freq;

	NodeTree* left;
	NodeTree* right;

	NodeTree() : symb(""), code(""), freq(0), left(nullptr), right(nullptr) {};
};

class ShenonFano {
public:
	ShenonFano(string s) : str(s), decode_str(""), root(nullptr), code_str(""),
table(nullptr) {};
	void create_table();

	void print_table();

	string code();
	string decode();
```

```
	string take_decode() { return decode_str; };
	string take_code() { return code_str; };
	List* take_table() { return table; };

private:
	void create_code(string code, NodeTree* node, int start, int end);
	string str;
	string decode_str;
	string code_str;
	List* table;
	NodeTree* root;
};
```

Файл List.h

```
#pragma once
#include <string>

using namespace std;

class Node {
public:
	int freq;
	char symb;
	string code;

	Node* next;
	Node* prev;

	Node(char c) : freq(1), code(""), symb(c), next(nullptr), prev(nullptr) {};
};

class List{
public:
	List() : size(0), head(nullptr), tail(nullptr) {};

	void push(char c);
	void sort();
	Node* take_head() { return this->head; };
	Node* take_tail() { return this->tail; };
	int take_size() { return this->size; };
	string search(char c);

private:
	void swap(Node* l, Node* r);
	int size;
	Node* head;
	Node* tail;
};
```

Файл ShenonFano.cpp:

```
#include "ShenonFano.h"

void ShenonFano::create_table()
{
	this->table = new List();
	for (int i = 0; i < str.size(); i++) {
		this->table->push(str[i]);
	}
	this->table->sort();
	this->root = new NodeTree();
```

```
	if (this->table->take_size() == 1) {
		this->root->left = new NodeTree();
		this->root->left->code = "0";
		this->root->left->symb = table->take_head()->symb;
		this->root->left->freq = table->take_head()->freq;
		table->take_head()->code = "0";
		return;
	}
	create_code("", this->root, 1, this->table->take_size());
}

void ShenonFano::create_code(string code, NodeTree* node, int start, int end)
{
	if (node == nullptr) {
		node = new NodeTree();
	}


	Node* buff = this->table->take_head();
	for (int i = 1; i < start; i++) {
		buff = buff->next;
	}

	if (start == end) {
		node->code = code;
		node->symb = buff->symb;
		node->freq = buff->freq;
		buff->code = code;
		return;
	}

	Node* cur = buff;
	string str = "";
	int ds = 0;

	for (int i = start; i < end; i++) {
		ds += buff->freq;
		str = str + buff->symb;
		buff = buff->next;
	}

	ds /= 2;

	int S = 0;
	int it = start;
	int m = start;
	buff = cur;
	while ((S + buff->freq < ds) && (it < end)) {
		S += buff->freq;

		it++; m = it;
		buff = buff->next;
	}

	node->symb = str;
	node->left = new NodeTree();
	node->right = new NodeTree();
```

```
	create_code(code + '0', node->left, start, m);
	create_code(code + '1', node->right, m+1 , end);
}

void ShenonFano::print_table()
{
	Node* buff = this->table->take_head();
	while (buff)
	{
		cout << buff->symb << " - " << buff->code
			<< " frequency = " << buff->freq << endl;
		buff = buff->next;
	}
}

string ShenonFano::code()
{
	this->code_str = "";
	for (int i = 0; i < this->str.size(); i++) {
		this->code_str = this->code_str + this->table->search(str[i]);
	}
	return this->code_str;
}

string ShenonFano::decode()
{
	this->decode_str = "";
	if (root == nullptr)
		return decode_str;
	NodeTree* buff = this->root;
	for (int i = 0; i < this->code_str.size(); i++) {
		if (this->code_str[i] == '0') {
			buff = buff->left;
		}
		else {
			buff = buff->right;
		}
		if ((buff->left == nullptr) && (buff->right == nullptr)) {
			this->decode_str = this->decode_str + buff->symb;
			buff = this->root;
		}
	}

	return decode_str;
}
```

Файл List.cpp

```
#include "List.h"

void List::push(char c)
{
	if (this->head == nullptr) {
		this->head = new Node(c);
		this->tail = this->head;
		this->size++;
```

```
			return;
		}

		Node* buff = this->head;
		while (buff)
		{
			if (buff->symb == c) {
				buff->freq++;
				return;
			}
			buff = buff->next;
		}

		buff = this->tail;
		Node* add = new Node(c);
		add->prev = buff;
		buff->next = add;
		this->tail = add;
		this->size++;
}

void List::sort()
{
		for (int i = 0; i < this->size; i++) {
			Node* left = this->head;
			Node* right = this->head->next;
			for (int j = 0; j < this->size - 1; j++) {
				if (left->freq < right->freq) {
					swap(left, right);
					left = right->next;
					right = left->next;
					continue;
				}
				left = left->next;
				right = right->next;
			}
		}
}

void List::swap(Node* l, Node* r)
{
		Node* buff = r->next;
		r->next = l;
		l->next = buff;

		buff = l->prev;
		l->prev = r;
		r->prev = buff;

		if (r->prev)
			r->prev->next = r;
		else
			this->head = r;

		if (l->next)
			l->next->prev = l;
		else
			this->tail = l;
}

string List::search(char c)
{
		Node* buff = this->head;
		while (buff)
		{
```

```
			if (buff->symb == c) {
				return buff->code;
			}
			buff = buff->next;
		}
		return "_";
}
```

Файл test.cpp

```
#include "pch.h"
#include "ShenonFano.h"
#include <iostream>
#include <windows.h>

void PrintTabs(HANDLE hStdOut) {
		SetConsoleTextAttribute(hStdOut, FOREGROUND_GREEN | FOREGROUND_INTENSITY);
		cout << "[          ] ";
		SetConsoleTextAttribute(hStdOut, FOREGROUND_GREEN | FOREGROUND_BLUE
			| FOREGROUND_RED | FOREGROUND_INTENSITY);
}

void TreeOut(ShenonFano test, HANDLE hStdOut) {
		SetConsoleTextAttribute(hStdOut, FOREGROUND_GREEN | FOREGROUND_INTENSITY);
		cout << "[    String] ";
		SetConsoleTextAttribute(hStdOut, FOREGROUND_GREEN | FOREGROUND_BLUE
			| FOREGROUND_RED | FOREGROUND_INTENSITY);

		cout << test.take_decode() << endl;

		SetConsoleTextAttribute(hStdOut, FOREGROUND_GREEN | FOREGROUND_INTENSITY);
		cout << "[      Code] ";
		SetConsoleTextAttribute(hStdOut, FOREGROUND_GREEN | FOREGROUND_BLUE
			| FOREGROUND_RED | FOREGROUND_INTENSITY);

		cout << test.take_code() << endl;
}

void ListOut(List test, HANDLE hStdOut) {
		SetConsoleTextAttribute(hStdOut, FOREGROUND_GREEN | FOREGROUND_INTENSITY);
		cout << "[    Symbols] ";
		SetConsoleTextAttribute(hStdOut, FOREGROUND_GREEN | FOREGROUND_BLUE
			| FOREGROUND_RED | FOREGROUND_INTENSITY);

		Node* buff = test.take_head();
		while (buff)
		{
			cout << buff->symb << " ";
			buff = buff->next;
		}
		cout << endl;

		SetConsoleTextAttribute(hStdOut, FOREGROUND_GREEN | FOREGROUND_INTENSITY);
		cout << "[      Freq] ";
		SetConsoleTextAttribute(hStdOut, FOREGROUND_GREEN | FOREGROUND_BLUE
			| FOREGROUND_RED | FOREGROUND_INTENSITY);

		buff = test.take_head();
		while (buff)
		{
			cout << buff->freq << " ";
			buff = buff->next;
		}
```

```
	cout << endl;
}


TEST(ShenonFanoTest, create_table) {
	HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
	SetConsoleTextAttribute(hStdOut, FOREGROUND_GREEN | FOREGROUND_BLUE
		| FOREGROUND_RED | FOREGROUND_INTENSITY);

	ShenonFano test("asdfsagasdklasfasaasfdaafas");
	char symb[] = { 'a', 's', 'f', 'd', 'g', 'k', 'l' };
	string code[] = { "00", "01", "10", "110", "1110", "11110", "11111" };
	int freq[] = { 10,7,4,3,1,1,1 };
	test.create_table();
	test.code();
	test.decode();

	TreeOut(test, hStdOut);

	List* table = test.take_table();
	Node* buff = table->take_head();

	for (int i = 0; i < 7; i++) {
		ASSERT_EQ(buff->symb, symb[i]);
		ASSERT_EQ(buff->freq, freq[i]);
		ASSERT_STREQ(buff->code.c_str(), code[i].c_str());
		buff = buff->next;
	}

}


TEST(ShenonFanoTest, code) {
	HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
	SetConsoleTextAttribute(hStdOut, FOREGROUND_GREEN | FOREGROUND_BLUE
		| FOREGROUND_RED | FOREGROUND_INTENSITY);

	ShenonFano test("asdfsagasdklasfasaasfdaafas");
	char str[] = "000111010010011100001110111101111100011000010000011011000000100001";
	test.create_table();
	test.code();
	test.decode();

	TreeOut(test, hStdOut);

	ASSERT_STREQ(test.take_code().c_str(), str);
}

TEST(ShenonFanoTest, decode) {
	HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
	SetConsoleTextAttribute(hStdOut, FOREGROUND_GREEN | FOREGROUND_BLUE
		| FOREGROUND_RED | FOREGROUND_INTENSITY);

	ShenonFano test("asdfsagasdklasfasaasfdaafas");
	char str[] = "asdfsagasdklasfasaasfdaafas";
	test.create_table();
	test.code();
	test.decode();

	TreeOut(test, hStdOut);

	ASSERT_STREQ(test.take_decode().c_str(), str);
}

TEST(ListTest, push) {
```

```
	HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
	SetConsoleTextAttribute(hStdOut, FOREGROUND_GREEN | FOREGROUND_BLUE
		| FOREGROUND_RED | FOREGROUND_INTENSITY);

	List test;
	string str = "1234444456";
	char arr[] = { '1','2','3','4','5','6' };
	int freq[] = { 1,1,1,5,1,1 };
	for (int i = 0; i < str.size(); i++) {
		test.push(str[i]);
	}

	ListOut(test, hStdOut);

	Node* buff = test.take_head();
	for (int i = 0; i < 6; i++) {
		ASSERT_EQ(buff->symb, arr[i]);
		ASSERT_EQ(buff->freq, freq[i]);
		buff = buff->next;
	}
}

TEST(ListTest, sort) {
	HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
	SetConsoleTextAttribute(hStdOut, FOREGROUND_GREEN | FOREGROUND_BLUE
		| FOREGROUND_RED | FOREGROUND_INTENSITY);

	List test;
	string str = "1234444456";
	char arr[] = { '4','1','2','3','5','6' };
	int freq[] = { 5,1,1,1,1,1 };
	for (int i = 0; i < str.size(); i++) {
		test.push(str[i]);
	}

	test.sort();

	ListOut(test, hStdOut);



	Node* buff = test.take_head();
	for (int i = 0; i < 6; i++) {
		ASSERT_EQ(buff->symb, arr[i]);
		ASSERT_EQ(buff->freq, freq[i]);
		buff = buff->next;
	}
}

TEST(ListTest, search) {
	HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
	SetConsoleTextAttribute(hStdOut, FOREGROUND_GREEN | FOREGROUND_BLUE
		| FOREGROUND_RED | FOREGROUND_INTENSITY);

	ShenonFano table("1234444456");
	table.create_table();
	List* test = table.take_table();
	string code[] = { "0","100","101","110","1110","1111" };

	ListOut(*test, hStdOut);

	Node* buff = test->take_head();
	for (int i = 0; i < 6; i++) {
		ASSERT_STREQ(buff->code.c_str(), code[i].c_str());
		buff = buff->next;
```

```
        }
    }
```