Оглавление

1. Постановка задачи.

1.1. Цель курсовой работы

Задачей данной курсовой работы является разработка программы, которая должна архивировать и разархивировать одиночные файлы, используя алгоритм Фано.

1.2 Информация, позволяющая понять постановку задачи.

При запуске вводится имя кодируемого или декодируемого файла, в зависимости от действия, мы либо кодируем символы, те что встречаются чаще коротким кодом, те что реже более длинным, это помогает на несколько процентов сжать исходный файл, а при большой вероятности некоторых элементов, мы можем сжать файл чуть больше чем в 2 раза. Это поможет нам выделять для него меньше памяти и сократить время передачи. Либо мы декодируем полученный файл и получаем данные практически без потерь.

2. Алгоритм.

2.1 Алгоритм Фано.

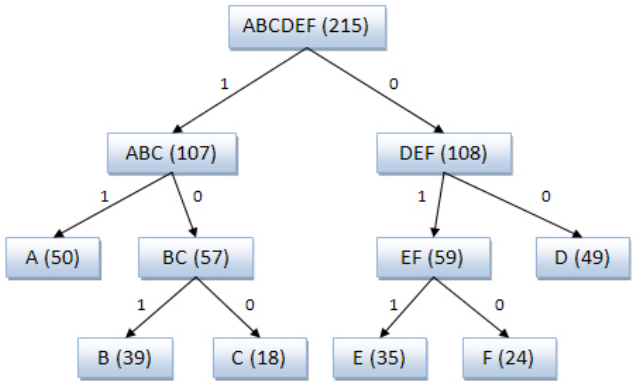
Код Шеннона — Фано строится с помощью дерева. Построение этого дерева начинается от корня. Всё множество кодируемых элементов соответствует корню дерева (вершине первого уровня). Оно разбивается на два подмножества с примерно одинаковыми суммарными вероятностями. Эти подмножества соответствуют двум вершинам второго уровня, которые соединяются с корнем. Далее каждое из этих подмножеств разбивается на два подмножества с примерно одинаковыми суммарными вероятностями. Им соответствуют вершины третьего уровня. Если подмножество содержит единственный элемент, то ему соответствует концевая вершина кодового дерева; такое подмножество разбиению не подлежит. Подобным образом поступаем до тех пор, пока не получим все концевые вершины. Ветви кодового дерева размечаем символами 1 и 0, как в случае кода Хаффмана.

При построении кода Шеннона — Фано разбиение множества элементов может быть произведено, вообще говоря, несколькими способами. Выбор разбиения на уровне n может ухудшить варианты разбиения на следующем уровне (n + 1) и привести к неоптимальности кода в целом. Другими словами, оптимальное поведение на каждом шаге пути ещё не гарантирует оптимальности всей совокупности действий. Поэтому код Шеннона — Фано не является оптимальным в

общем смысле, хотя и дает оптимальные результаты при некоторых распределениях вероятностей. Для одного и того же распределения вероятностей можно построить, вообще говоря, несколько кодов Шеннона — Фано, и все они могут дать различные результаты. Если построить все возможные коды Шеннона — Фано для данного распределения вероятностей, то среди них будут находиться и все коды Хаффмана, то есть оптимальные коды.

Исходные символы:

- A (частота встречаемости 50)
- B (частота встречаемости 39)
- C (частота встречаемости 18)
- D (частота встречаемости 49)
- E (частота встречаемости 35)
- F (частота встречаемости 24)



Полученный код: A — 11, B — 101, C — 100, D — 00, E — 011, F — 010.

2.2 Псевдокод

```
#ifndef BYTE_CODE
#define BYTE_CODE

#include <vector>
#include <string>

class ByteCode
{
private:
	char byte;
	int freq;
	std::string code;
public:
	ByteCode(char b, int f) : byte(b), freq(f)
	{}
	char getByte()
	{
		return byte;
	}
	int getFrequency()
	{
		return freq;
	}
```

```
	std::string getCode()
	{
		return code;
	}
	void addCode(int c)
	{
		code.push_back(c);
	}
	~ByteCode()
	{}
};

#endif

#pragma once

#include <vector>
#include <string>
#include <fstream>
#include <map>
#include <utility>
#include "ByteCode.h"
using namespace std;

class FanoEncoding
{
private:
	ifstream in;
	multimap<int, unsigned char> countFrequencies();
	void makeCodes(int first, int last, std::vector<ByteCode>& byteCodes);
	void writeToFile(ofstream &out, std::vector<ByteCode>& byteCodes, string& in_str);
	unsigned char stringToByte(int first, int last, string& code);
public:
	FanoEncoding();
	~FanoEncoding();
	void fanoEncoding(string in_str);
};

#include "FanoEncoding.h"
#include <iostream>
#include <math.h>

FanoEncoding::FanoEncoding()
{}

FanoEncoding::~FanoEncoding()
{}

void FanoEncoding::fanoEncoding(string in_str)
{

	in.open(in_str, ios::binary);
	if (!in.is_open())
	{
		cout << endl << "Невозможно открыть файл";
		exit(1);
	}
	multimap<int, unsigned char> frequencies = countFrequencies();
```

```
	std::vector<ByteCode> byteCodes;
	for (std::multimap<int, unsigned char>::iterator it = frequencies.begin(); it !=
frequencies.end(); ++it)
	{
		byteCodes.push_back(ByteCode(it->second, it->first));
	}
	makeCodes(0, byteCodes.size() - 1, byteCodes);
	ofstream out;
	out.open("encoded", ios::binary);
	writeToFile(out, byteCodes, in_str);
	in.close();
	out.close();
}

multimap<int, unsigned char> FanoEncoding::countFrequencies()
{
	multimap<int, unsigned char> frequencies;
	vector<int> alf(256, 0);
	unsigned char a = in.get();
	while (!in.eof())
	{
		alf[a]++;
		a = in.get();
	}
	for (int i = 0; i < 256; i++)
	{
		if (alf[i] != 0)
		{
			frequencies.insert(make_pair(alf[i], i));
		}
	}
	return frequencies;
}

void FanoEncoding::makeCodes(int first, int last, std::vector<ByteCode>& byteCodes)
{
	if (first >= last)
	{
		return;
	}
	if (first == last - 1)
	{
		byteCodes[first].addCode('0');
		byteCodes[last].addCode('1');
	}
	int right = last;
	int left = first;
	int rightSum = byteCodes[right].getFrequency();
	int leftSum = byteCodes[left].getFrequency();
	while (left != right - 1)
	{
		if (leftSum > rightSum)
		{
			right--;
			rightSum += byteCodes[right].getFrequency();
		}
		else
		{
```

```
                left++;
                leftSum += byteCodes[left].getFrequency();
            }
        }
        for (int i = first; i <= left; i++)
        {
            byteCodes[i].addCode('0');
        }
        for (int i = right; i <= last; i++)
        {
            byteCodes[i].addCode('1');
        }
        makeCodes(first, left, byteCodes);
        makeCodes(right, last, byteCodes);
}

void FanoEncoding::writeToFile(ofstream &out, std::vector<ByteCode>& byteCodes, string&
in_str)
{
        map<unsigned char, string> codesMap;
        for (std::vector<ByteCode>::iterator i = byteCodes.begin(); i != byteCodes.end();
++i)
        {
            codesMap.insert(std::make_pair(i->getByte(), i->getCode()));
        }
        string encoded;
        in.clear();
        in.seekg(0);
        unsigned char a = in.get();
        while (!in.eof())
        {
            std::map<unsigned char, string>::iterator it = codesMap.find(a);
            encoded += it->second;
            a = in.get();
        }
        for (int i = 0; i < 256; i++)
        {
            std::map<unsigned char, string>::iterator it = codesMap.find(i);
            if (it != codesMap.end())
            {
                out << it->second;
            }
            else
            {
                out << " ";
            }
            out << "\n";
        }
        out << in_str;
        out << "\n";
        int no_read_bits = 0;
        while ((no_read_bits + encoded.size()) % 8 != 0)
            no_read_bits++;
        out << no_read_bits << "\n";
        int index = 0;
        for (size_t i = 0; i < (encoded.size() - (encoded.size() % 8)) / 8; i++, index +=
8)
        {
```

```
			out << stringToByte(index, index + 7, encoded);
		}
		if (no_read_bits != 0)
		{
			out << stringToByte(index, encoded.size() - 1, encoded);
		}
}

unsigned char FanoEncoding::stringToByte(int first, int last, string& code)
{
		unsigned char res = 0;
		for (int i = first, j = 7; i <= last; i++, j--)
		{
			if (code[i] == '1')
			{
				res += pow(2, j);
			}
		}
		return res;
}
```

```
#pragma once

#include <string>
#include <fstream>
#include <map>
using namespace std;

class FanoDecoding
{
private:
		ifstream in;
		map<string, unsigned char> getTable();
		void writeToFile(ofstream& out, map<string, unsigned char>& table);
		string byteToStr(unsigned char byte);
public:
		FanoDecoding();
		~FanoDecoding();
		void fanoDecoding(string s);
};
```

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

FanoDecoding::FanoDecoding()
{}

FanoDecoding::~FanoDecoding()
{}

void FanoDecoding::fanoDecoding(string s)
{
		in.open(s, ios::binary);
		map<string, unsigned char> table = getTable();
		ofstream out;
		writeToFile(out, table);
		in.close();
		out.close();
```

```
}

map<string, unsigned char> FanoDecoding::getTable()
{
	map<string, unsigned char> table;
	for (int i = 0; i < 256; i++)
	{
		string s;
		getline(in, s, '\n');
		if (s != " ")
		{
			table.insert(make_pair(s, i));
		}
	}
	return table;
}

void FanoDecoding::writeToFile(ofstream& out, map<string, unsigned char>& table)
{
	string uncoded, in_str, s;
	getline(in, in_str, '\n');
	int noReadBits = in.get() - '0';
	in.get();
	while (!in.eof())
	{
		unsigned char const a = in.get();
		if (!in.eof())
		{
			s = byteToStr(a);
			for (size_t i = 0; i < s.size(); i++)
			{
				uncoded.push_back(s[i]);
			}
			s.clear();
		}
	}
	s.clear();
	out.open(in_str, ios::binary);
	std::map<std::string, unsigned char>::iterator it;
	for (size_t i = 0; i < uncoded.size() - noReadBits; i++)
	{
		s.push_back(uncoded[i]);
		it = table.find(s);
		if (it != table.end())
		{
			out << it->second;
			s.clear();
		}
	}
}

string FanoDecoding::byteToStr(unsigned char byte)
{
	std::string res(8, '0');
	unsigned char r;
	int i = 0;
	while (byte >= 2)
	{
```

```
		r = byte % 2;
		byte = (byte - r) / 2;
		if (r == 1)
			res[7 - i] = '1';
		else
			res[7 - i] = '0';
		i++;
	}
	if (byte == 1)
		res[7 - i] = '1';
	else
		res[7 - i] = '0';
	return res;
}

#include <iostream>
#include <fstream>
#include <string>
#include "FanoEncoding.h"
#include "FanoDecoding.h"
#include <locale.h>
using namespace std;

int main()
{
	setlocale(LC_ALL, "RUS");
	int input;
	cout << "Кодирование и декодирование с помощью алгоритма Фано" << endl;
	cout << "Введите 0, если хотите закодировать файл; введите 1, если хотите
раскодировать файл" << endl;
	cin >> input;
	if (input == 0)
	{
		cout << "Введите имя файла: " << endl;
		string s;
		cin >> s;
		FanoEncoding f;
		f.fanoEncoding(s);
		cout << "Файл закодирован. Название файла encoded." << endl;
	}
	else if (input == 1)
	{
		cout << "Введите имя файла: " << endl;
		string s;
		cin >> s;
		FanoDecoding f2;
		f2.fanoDecoding(s);
		cout << "Файл раскодирован." << endl;
	}
	else
	{
		cout << "Неверный ввод" << endl;
	}
	system("pause");
	return 0;
}
```

2.2. Сложность:

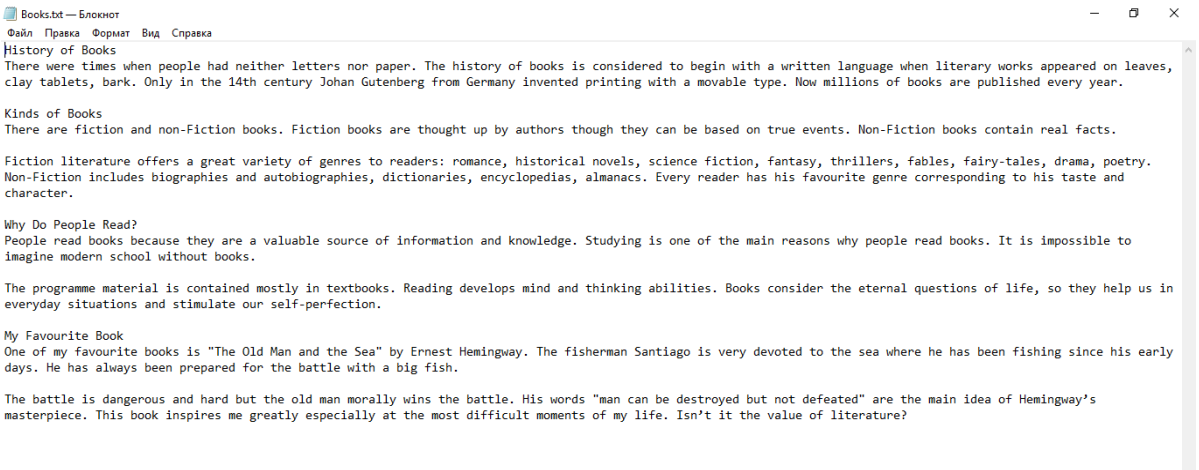
Рекурсивно вызывается функция построения «дерева» нашего алфавита, то есть массив делится примерно пополам на каждом шаге, таким образом, можно предположить, что сложность логарифмическая и равна O(n*log(n)).

3. Инструкция пользователя.

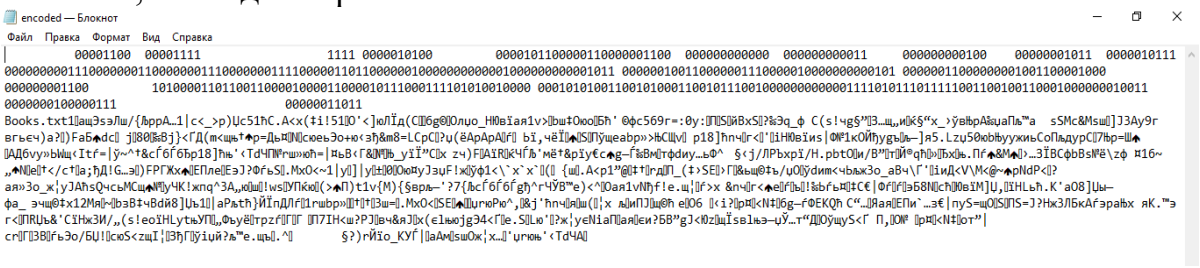
После запуска программы, пользователь может выбрать закодировать или раскодировать файл и ввести 0 или 1 соответственно. После выбора 0: Будет предложено ввести имя и формат файла, который пользователь хочет закодировать. Программа произведет кодировку и выведет на экран результат «Файл закодирован. Название файла encoded.». После выбора 1: Будет предложено ввести имя файла, который пользователь хочет раскодировать, encoded. Программа выведет результат «Файл раскодирован.». Файл будет храниться в папке программы, под именем файла, который был закодирован в исходном файле.

4. Текстовые примеры.

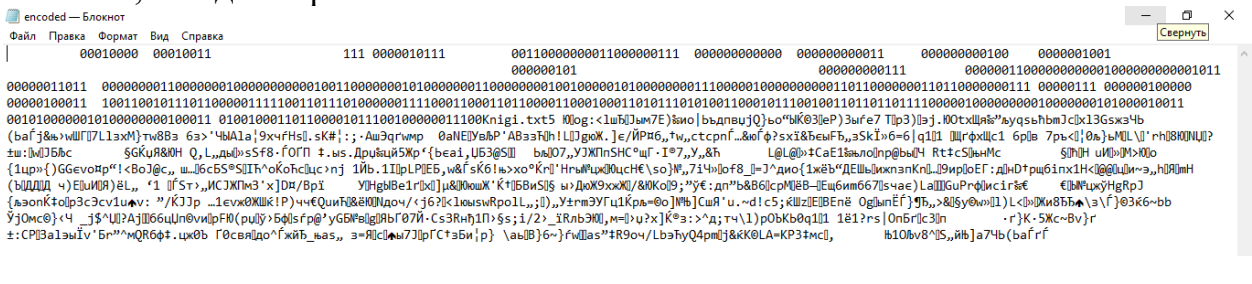
Пример 1:
Кодирование и декодирование с помощью алгоритма Фано
Введите 0, если хотите закодировать файл; введите 1, если хотите раскодировать файл
0
Введите имя файла:
Books.txt
Файл закодирован. Название файла encoded.
Books.txt, Входной файл:



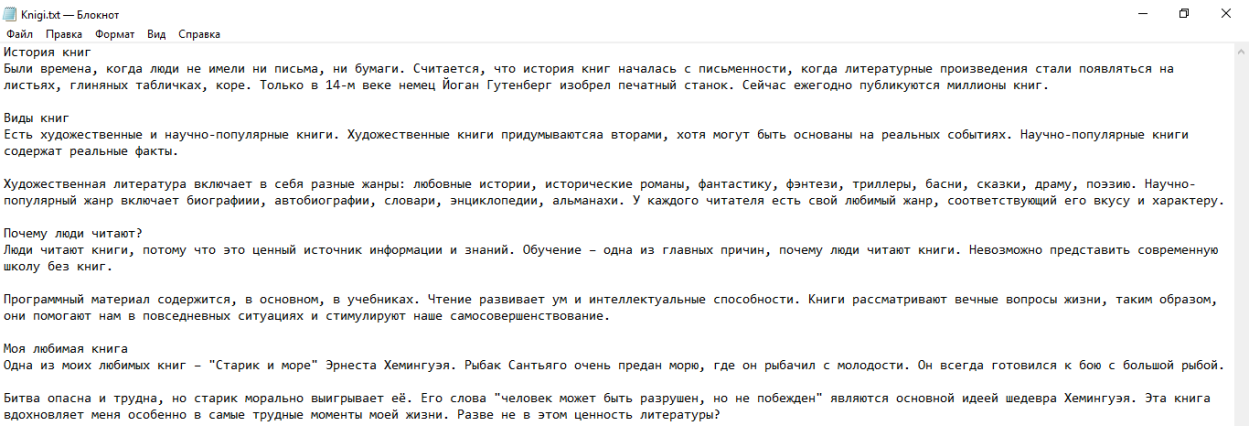
encoded, Выходной файл:



Пример 2:
Кодирование и декодирование с помощью алгоритма Фано
Введите 0, если хотите закодировать файл; введите 1, если хотите раскодировать файл
1
Введите имя файла:
encoded
Файл раскодирован.
Encoded, Входной файл:



Knigi.txt, Выходной файл



Пример 3:
Кодирование и декодирование с помощью алгоритма Фано
Введите 0, если хотите закодировать файл; введите 1, если хотите раскодировать файл
0
Введите имя файла:
Knigi.txt
Файл закодирован. Название файла encoded.
Knigi.txt, Входной файл:

Книги.txt — Блокнот
Файл Правка Формат Вид Справка

История книг
Были времена, когда люди не имели ни письма, ни бумаги. Считается, что история книг началась с письменности, когда литературные произведения стали появляться на листьях, глиняных табличках, коре. Только в 14-м веке немец Йоган Гутенберг изобрел печатный станок. Сейчас ежегодно публикуются миллионы книг.

Виды книг
Есть художественные и научно-популярные книги. Художественные книги придумываются вторами, хотя могут быть основаны на реальных событиях. Научно-популярные книги содержат реальные факты.

Художественная литература включает в себя разные жанры: любовные истории, исторические романы, фантастику, фэнтези, триллеры, басни, сказки, драму, поэзию. Научно-популярный жанр включает биографии, автобиографии, словари, энциклопедии, альманахи. У каждого читателя есть свой любимый жанр, соответствующий его вкусу и характеру.

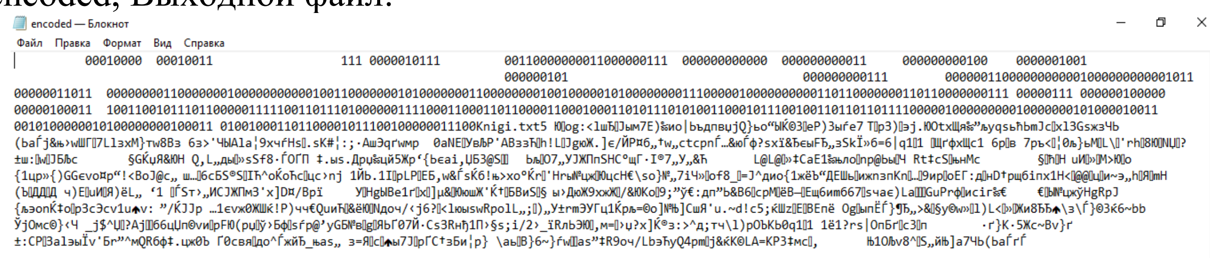
Почему люди читают?
Люди читают книги, потому что это ценный источник информации и знаний. Обучение – одна из главных причин, почему люди читают книги. Невозможно представить современную школу без книг.

Программный материал содержится, в основном, в учебниках. Чтение развивает ум и интеллектуальные способности. Книги рассматривают вечные вопросы жизни, таким образом, они помогают нам в повседневных ситуациях и стимулируют наше самосовершенствование.

Моя любимая книга
Одна из моих любимых книг – "Старик и море" Эрнеста Хемингуэя. Рыбак Сантьяго очень предан морю, где он рыбачил с молодости. Он всегда готовился к бою с большой рыбой.

Битва опасна и трудна, но старик морально выигрывает её. Его слова "человек может быть разрушен, но не побежден" являются основной идеей шедевра Хемингуэя. Эта книга вдохновляет меня особенно в самые трудные моменты моей жизни. Разве не в этом ценность литературы?

encoded, Выходной файл:



encoded — Блокнот
Файл Правка Формат Вид Справка

00010000 00010011 111 0000010111 0011000000001100000111 000000000000 000000000011 000000000100 0000001001
000000101 000000000111 0000001100000000000100000000001011
00000011011 000000001100000001000000000010011000000010100000001100000000100100000101000000001110000010000000001101100000001101100000000111 00000111 000000100000
00000100011 1001100101110110000011111001101110100000011110001100011011000011000100011010111010100110001011100100110110110111100000100000000010000000101000010011
001010000001010000000100011 01001000110110000101110010000011100Knigi.txt5

5. Список литературы

А. М. Яглом, И. М. Яглом. Вероятность и информация

М.Н. Аршинов, Л.Е. Садовский, Коды и математика

https://ru.wikipedia.org/wiki/Алгоритм_Шенона_—_Фано