

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

**ОТЧЕТ**  
**по лабораторной работе №5**  
**по дисциплине «Алгоритмы и структуры данных»**  
**Тема: Метод Шеннона-Фано**

Студент гр. 9304

\_\_\_\_\_

Атаманов С.Д.

Преподаватель

\_\_\_\_\_

Филатов Ар. Ю.

Санкт-Петербург

2020

### **Цель работы.**

Ознакомиться с алгоритмом шифрования Шеннона-Фано. Реализовать алгоритм, используя язык программирования C++.

### **Задание.**

Вариант 1.

Метод Шеннона-Фано

Программа должна иметь следующие ключи запуска: `encode` (включить режим «кодирование»), `decode` (включить режим «декодирование»), `file` (указать название входного файла, иначе ожидается ввод с консоли), `o` (указать название выходного файла, иначе на консоль), `debug` (включить режим отладки). Файл, полученный кодированием, должен быть расшифрован программой обратно.

### **Выполнение работы.**

#### **Описание алгоритма.**

Работа алгоритма основывается на Бинарном дереве. Алгоритм получает таблицу символов с частотой их появления в тексте(далее — весом). Все символы складываются в единую строку, их веса складываются. В узлы дерева записываются отдельные символы или строка состоящая из них. Затем, если это строка, она разделяется по весу напополам, в левое поддерево записывается строка с меньшим весом, в правое с большим. Так продолжается до тех пор, пока в листьях не будут отдельные символы. После по дереву проходятся, добавляя к каждому коду левого поддерева каждого узла 0, а к правому 1. В результате у каждого символа получается уникальный код который тем короче, чем чаще встречается символ и наоборот, тем длинее, чем символ встречается реже.

#### **Формат входных и выходных данных.**

Программа имеет 4 ключа запуска: `encode` (включить режим «кодирование»), `decode` (включить режим «декодирование»), `file` (указать название входного файла, иначе ожидается ввод с консоли), `o` (указать название выходного файла, иначе на консоль). При указании только режима

«кодирование» программа считывает строку из `stdin`, выводит закодированное сообщение в `stdout` и сохраняет файл с кодом в папке с исполняемым файлом. В режиме только «декодирование» программе нужно скормить путь к файлу с кодом символов, ввести закодированное сообщение в `stdin`, раскодированное сообщение будет выведено в `stdout`. Ключи `-f` и `-o` требуют пути к файлам, из которых будет происходить чтение и запись соответственно.

Формат входных данных ограничивается символами из таблицы ASCII английского языка.

### **Используемые структуры данных и реализованные функции.**

`class BinTreeNode` — класс, описывающий узлы бинарного дерева. Состоит из 3 полей: 2 `std::shared_ptr<BinTreeNode>` указателя на левое и правое поддерево, `std::pair<std::string, int>` - значение узла, где `std::string` — строка или символ, `int` — вес.

`std::shared_ptr<BinTreeNode> getShannonFanoTree()` - метод класса `BinTreeNode`, которое возвращает дерево Шеннона-Фано с узлами, заполненными строками и символами.

`void getListOfElem()` - функция заполняет словарь уникальными символами, параллельно считая вес каждого символа в кодируемой строке.

`void getStringWithWeigh()` - функция складывает все символы словаря в одну строку, вместе с их весами.

`std::string encode()` - функция кодирует исходную строку. Функция проходит по строке, при встрече каждого символа, ищет в словаре его код и записывает его в выходной файл или `stdout`, параллельно формируя файл с кодами символов.

`std::string getStringFromFile()` - преобразует содержимое файла в одну строку символов.

`std::map<std::string, char> getCodesFromFile()` - заполняет словарь для декодирования из файла с кодами символов.

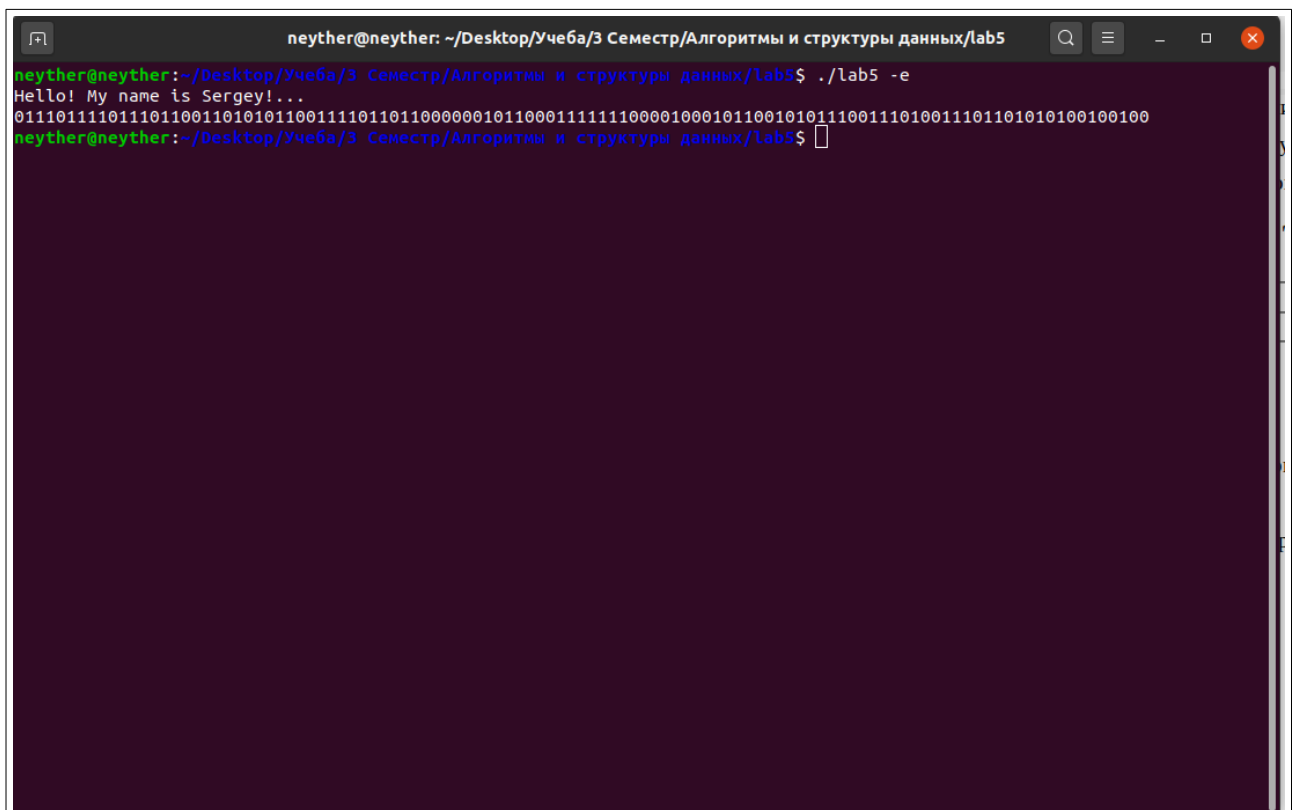
`void decode()` - происходит декодирование файла. Происходит проход по строке, запись символов во временную строку и сравнение последней со

словарем кодов, после в выходной файл или же stdout записывается декодированный символ за символом.

Разработанный программный код смотри в приложении А.

### **Тестирование.**

Для тестирования был написан bash-скрипт, который считывает аргументы командной строки и тестовые данные из текстовых документов папки Tests и передает их программе. Результаты работы программы сравниваются с заведомо правильными данными из папки Test\_results. Данные о результатах тестирования выводятся в stdout.



```
neyther@neyther: ~/Desktop/Учеба/3 Семестр/Алгоритмы и структуры данных/lab5$ ./lab5 -e
Hello! My name is Sergey!...
011101111011101100110101011001111011011000000101100011111100001000101100101011100111010011101101010100100100
neyther@neyther: ~/Desktop/Учеба/3 Семестр/Алгоритмы и структуры данных/lab5$
```

Рисунок 1 — Пример работы программы

Результаты тестирования смотри в приложении Б.

### **Выводы.**

В ходе выполнения работы были получены навыки работы с алгоритмом Шеннона-Фано.

Была написана программа на языке C++, которая выполняет кодирование и декодирование текста методом Шеннона-Фано.



## ПРИЛОЖЕНИЕ А.

### ИСХОДНЫЙ КОД ПРОГРАММЫ.

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <memory>
#include <map>
#include <algorithm>
#include <getopt.h>

class BinTreeNode{
public:
    std::shared_ptr<BinTreeNode> left {nullptr};
    std::shared_ptr<BinTreeNode> right {nullptr};
    std::pair<std::string, int> data;

    std::shared_ptr<BinTreeNode> getShannonFanoTree(std::pair<std::string, int>
stringWithWeight, std::map<char, int> map, std::map<char, std::string>& codes, std::string
code) {
    std::shared_ptr<BinTreeNode> tree = std::make_shared<BinTreeNode>();
    std::pair<std::string, int> left;
    std::pair<std::string, int> right;
    tree->data = stringWithWeight;
    if(stringWithWeight.first.length() == 1) {
        codes.insert({tree->data.first[0], code});
        return tree;
    }
    while (true) {
        if ((left.second + map[stringWithWeight.first[0]]) > (stringWithWeight.second) &&
left.second == 0) {
            right.first += stringWithWeight.first[0];
            right.second += map[stringWithWeight.first[0]];
            stringWithWeight.second -= map[stringWithWeight.first[0]];
            stringWithWeight.first.erase(0, 1);
            left.first = stringWithWeight.first;
            left.second = stringWithWeight.second;
            break;
        }
        else if((left.second + map[stringWithWeight.first[0]]) > (stringWithWeight.second))
            break;
        else {
            left.first += stringWithWeight.first[0];
            left.second += map[stringWithWeight.first[0]];
            stringWithWeight.second -= map[stringWithWeight.first[0]];
            stringWithWeight.first.erase(0, 1);
        }
    }
    if(right.second == 0) {
        right.first = stringWithWeight.first;
        right.second = stringWithWeight.second;
    }

    if(left.second > right.second)
        std::swap(left, right);
    if (left.second != 0) {
        tree->left = std::make_shared<BinTreeNode>();
```

```

        tree->left = getShannonFanoTree(left, map, codes, code + '0');
    }
    if (right.second != 0) {
        tree->right = std::make_shared<BinTreeNode>();
        tree->right = getShannonFanoTree(right, map, codes, code + '1');
    }
    return tree;
}
};

void getListOfElem(std::map<char, int>& map, std::string::iterator iterator){
    if(*iterator == '\0')
        return;
    while(*iterator != '\0') {
        if (map.find(*iterator) != map.end()) {
            map[*iterator]++;
        } else {
            map.insert({*iterator, 1});
        }
        iterator++;
    }
}

void getStringWithWeigh(std::map<char, int> stringMap, std::pair<std::string, int>&
weightString){
    auto iterBeg = stringMap.begin();
    auto iterEnd = stringMap.end();
    std::pair<char, int> max;
    std::map<char, int> finder;
    int flag;
    while(1){
        flag = 0;
        for(;iterBeg != iterEnd;iterBeg++){
            if(iterBeg->second > max.second && finder.find((*iterBeg).first) == finder.end()){
                max.first = (*iterBeg).first;
                max.second = (*iterBeg).second;
                flag = 1;
            }
        }
        iterBeg = stringMap.begin();
        if(flag == 1){
            weightString.first += max.first;
            weightString.second += max.second;
            finder.insert({max.first, max.second});
            max.second = 0;
            continue;
        }
        break;
    }
}

std::string encode(std::string::iterator encodeString){
    std::map<char, int> map;
    std::map<char, std::string> codes;
    std::pair<std::string, int> symbols;
    std::shared_ptr<BinTreeNode> shannonTree;
    std::string code;
    std::string output;

    getListOfElem(map, encodeString);
    getStringWithWeigh(map, symbols);

```

```

shannonTree = shannonTree->getShannonFanoTree(symbols, map, codes, code);

for(*encodeString != '\0'; encodeString++)
    output += codes[*encodeString];

std::ofstream outputFile;
outputFile.open("./Codes.txt");
auto iterBeg = codes.begin();
for(iterBeg; iterBeg != codes.end(); iterBeg++)
    outputFile << iterBeg->first << ":" << iterBeg->second << "; \n";
outputFile.close();

return output;
}

std::string getStringFromFile(const std::string& fileName) {
    std::string stringFile;
    std::ifstream encodeFile;
    encodeFile.open(fileName);
    if (!encodeFile.is_open()) {
        std::cout << "Error! \nUndeclared <fileName>.txt \n";
        exit(EXIT_FAILURE);
    }

    std::getline(encodeFile, stringFile);
    encodeFile.close();
    if (stringFile.empty()) {
        std::cout << "Error: Empty encode \n";
        exit(EXIT_FAILURE);
    }
    return stringFile;
}

std::map<std::string, char> getCodesFromFile(const std::string& fileName) {
    std::map<std::string, char> codes;
    char symbol;
    std::string temp;
    std::string code;
    auto iterBeg = temp.begin();
    std::ifstream file;
    file.open(fileName);
    if (!file.is_open()) {
        std::cout << "Error: Undeclared codeFile";
        exit(EXIT_FAILURE);
    }
    while(!file.eof()) {
        std::getline(file, temp);
        iterBeg = temp.begin();
        symbol = *iterBeg;
        iterBeg += 2;
        while(*iterBeg != ';') {
            code += *iterBeg;
            iterBeg++;
        }
        codes.insert({code, symbol});
        code.clear();
        temp.clear();
    }
    file.close();
    return codes;
}

```



```

void decode(const std::string& encode, std::map<std::string, char> codes, int output){
    std::string code;
    std::string fileDescript;
    if(output) {
        std::ofstream decodeFile;
        decodeFile.open("./Output.txt");
        auto iterBeg = encode.begin();
        if(encode.empty()){
            std::cout << "Error!\nUndeclared <fileName>.txt\n";
            exit(EXIT_FAILURE);
        }
        for (; iterBeg != encode.end(); iterBeg++) {
            while (true) {
                code += *iterBeg;
                if (codes.find(code) != codes.end())
                    break;
                else if((iterBeg+1) == encode.end() && codes.find(code) == codes.end()){
                    decodeFile << "Error: Wrong codeFile\n";
                    exit(EXIT_FAILURE);
                }
                else{
                    iterBeg++;
                    continue;
                }
            }
            fileDescript += codes[code];
            code.clear();
        }
        decodeFile << fileDescript << "\n";
        decodeFile.close();
    }
    else{
        auto iterBeg = encode.begin();
        for (iterBeg; iterBeg != encode.end(); iterBeg++) {
            while (true) {
                code += *iterBeg;
                if (codes.find(code) != codes.end())
                    break;
                else if(iterBeg == encode.end() && codes.find(code) ==
codes.end()){
                    std::cout << "Error: Wrong codeFile\n";
                    exit(EXIT_FAILURE);
                }
                else {
                    iterBeg++;
                    continue;
                }
            }
            fileDescript += codes[code];
            code.clear();
        }
        std::cout << fileDescript << "\n";
    }
}

struct Configuration{
    int encode = 0;
    int decode = 0;
    int output = 0;
    std::string codeFile;
}

```

```

    std::string inputFile;
};

int main(int argc, char* argv[]){
    setlocale(LC_ALL, "ru_RU");
    Configuration config;
    char* opts = "e?:d:f:o?:";
    struct option longOpts[]{
        {"encode", no_argument, nullptr, 'e'},
        {"decode", required_argument, nullptr, 'd'},
        {"file", required_argument, nullptr, 'f'},
        {"output", no_argument, nullptr, 'o'},
        {nullptr, 0, nullptr, 0}
    };

    int opt;
    int longIndex;
    opt = getopt_long(argc, argv, opts, longOpts, &longIndex);
    while(opt != -1){
        switch (opt) {
            case 'e':
                config.encode = 1;
                break;
            case 'd':
                config.decode = 1;
                config.codeFile = optarg;
                break;
            case 'f':
                config.inputFile = optarg;
                break;
            case 'o':
                config.output = 1;
                break;
        }
        opt = getopt_long(argc, argv, opts, longOpts, &longIndex);
    }

    if(config.encode == config.decode){
        std::cout << "Error: Must be or 'encode' or 'decode'\n";
        exit(EXIT_FAILURE);
    }

    if(config.encode){
        std::string encodeString;
        std::string encoded;
        if(!config.inputFile.empty()){
            encodeString = getStringFromFile(config.inputFile);
        }
        else
            std::getline(std::cin, encodeString);

        auto iterBeg = encodeString.begin();
        encoded = encode(iterBeg);

        if(config.output){
            std::ofstream output;
            output.open("./Output.txt");
            output << encoded << "\n";
            output.close();
        }
        else
            std::cout << encoded << "\n";
    }
}

```

```

    }

    if(config.decode){
        std::map<std::string, char> codes = getCodesFromFile(config.codeFile);
        std::string encode;
        if(!config.inputFile.empty()) {
            encode = getStringFromFile(config.inputFile);
        }
        else
            std::getline(std::cin, encode);

        decode(encode, codes, config.output);
    }

    return 0;
}

```

**ПРИЛОЖЕНИЕ Б.**  
**РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ.**

Входные данные	Выходные данные	Комментарий
-e Hello! My name is Sergey!	1111001100100110110000010111 1111100101000110111001101101 0100101010010110100111011111 000111110000	Кодирование из stdin в stdout
-d ./Tests/Tests/Test2/Codes.txt 111100110010011011000001 011111111001010001101110 011011010100101010010110 100111011111000111110000	Hello! My name is Sergey!	Декодирование 1 текста из stdin в stdout
-e -f ./Tests/Tests/Test3/Enter.txt I show you, who boss of this gym!!!	0100011110100111100010111000 1110011110000111001000111101 1101001110101101111110010101 1101110001101100101111010110 00101101100100100	Кодирование из файла в stdout
-d ./Tests/Tests/Test4/Codes.txt -f ./Tests/Tests/Test4/Enter.txt 010001111010011110001011 100011100111100001110010 001111011101001110101101 111110010101110111000110 110010111101011000101101 100100100	I show you, who boss of this gym!!!	Декодирование 3 текста из файла в stdout
-e -o Hey, buddy, you choose the wrong door. The leather club two box down...	1000100001011110110011111101 1011001100111011110110011101 1101011011011101001110010010 1001100000111101011100000111 1001111101010000100101011001 1010010111110010011100011111 0000011101010001001001101011 1000011111011101001010111011 1111001111010110001001111110 0101001110110011010110010000 001000100010	Кодирование из stdin в файл
-d ./Tests/Tests/Test6/Codes.txt	Hey, buddy, you choose the wrong door. The leather club two box	Декодирование 5 текста из stdin

<p>-o  100010000101111011001111  110110110011001110111101  100111011101011011011101  001110010010100110000011  110101110000011110011111  010100001001010110011010  010111110010011100011111  000001110101000100100110  101110000111110111010010  101110111111001111010110  001001111110010100111011  001101011001000000100010  0010</p>	<p>down...</p>	<p>в файл</p>
<p>-e -f  ./Tests/Tests/Test7/Enter.txt -  o  Oh, s##t, I'm sorry! Sorry for  what? Our daddy taught us  not to be ashamed of our...</p>	<p>1001011111100111011111010011  0100110001110011101110111010  1111010110101111100100000001  1011101100011101011001000000  0110110111000101000001110100  1111110010001110111110111001  0110000001111110001011110111  1011011011001100101100101010  1111100110111100111001110100  0010001101100110100111010111  1000001100101110111110010101  1011000011110011010100010110  101100000110101101011010</p>	<p>Кодирование из  файла в файл</p>
<p>-d  ./Tests/Tests/Test8/Codes.txt  -f  ./Tests/Tests/Test8/Enter.txt -  o  1001011111100111011111101  001101001100011100111011  101110101111010110101111  100100000001101110110001  110101100100000001101101  110001010000011101001111  110010001110111110111001  011000000111111000101111  011110110110110011001011  001010101111100110111100  111001110100001000110110</p>	<p>Oh, s##t, I'm sorry! Sorry for  what? Our daddy taught us not to  be ashamed of our...</p>	<p>Декодирование  7 текста из  файла в файл</p>

011010011101011110000011 001011101111100101011011 000011110011010100010110 101100000110101101011010		
-e -d ./Codes.txt Proverka №9	Error: Must be or 'encode' or 'decode'	Неправильный ввод аргументов
-e -f ./HA-HA	Error! Undeclared <fileName>.txt	Не существует файла, который нужно закодировать
-d ./Tests/Tests/Test11/Codes.txt -f ./Ha-Ha	Error! Undeclared <fileName>.txt	Не существует файла, который можно декодировать
-e -f ./Tests/Tests/Test12/Enter.txt	Error: Empty encode	Файл, который нужно закодировать пустой
-d ./MustBeCodes.txt ....:	Error: Undeclared codeFile	Файл с кодом для декодирования не существует