

**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: Бинарные деревья поиска и алгоритмы сжатия**  
**Вариант 2**

Студент гр. 8304

\_\_\_\_\_

Николаева М. А.

Преподаватель

\_\_\_\_\_

Фирсов М.А.

Санкт-Петербург

2019

### **Цель работы.**

Познакомиться с методикой решения задачи декодирования закодированной последовательности.

### **Постановка задачи.**

Вариант 2: реализовать декодирование: Фано-Шеннона

### **Описание алгоритма.**

Декодирование каждого символа будет сводиться к чтению кода и переводу его в символ, основываясь на алфавит, введенный пользователем вместе с закодированным сообщением. Для декодирования введенный алфавит сохраним в виде словаря, который содержит код и символ, соответствующий коду. Последовательно двигаясь по закодированному сообщению, будем «собирать» код очередного символа, как только такой код обнаружится в словаре, символ будет сохранен в результирующее декодированное сообщение и код начнет «собираться» заново. Это возможно, так как каждый код уникально определяет один закодированный символ и не является префиксом любого другого кода.

### **Спецификация программы.**

Программа предназначена для декодирования введенного пользователем сообщения, которое закодировано с помощью алгоритма Фано-Шеннона.

Программа написана на языке C++. Входными данными являются таблица кодов символов и закодированное сообщение, считываемые из файла или вводимые с клавиатуры. Выходными данными являются декодированное сообщение. Данные выводятся на экран монитора. Результат работы программы представлен на рисунке 1.

## Описание функций.

```
std::map<std::string, std::string> readTable(std::istream& in);  
std::map<std::string, std::string> readTable(std::ifstream& in);
```

Функции для считывания таблицы кодов символов из исходных данных. Принимают на вход поток данных, результатом является словарь, состоящий из ключей — кодов символов и значений — символов. Возвращаемое значение `std::map<std::string, std::string>`.

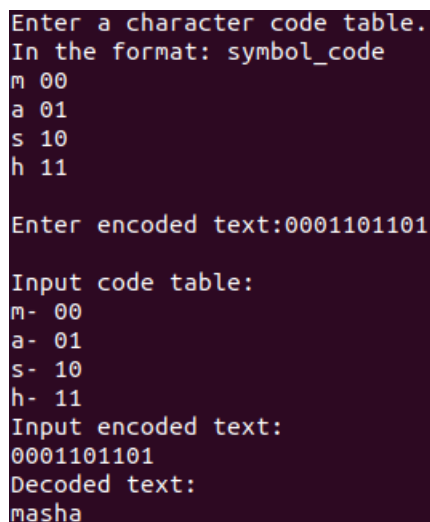
```
std::string readText(std::istream& in);  
std::string readText(std::ifstream& in);
```

Функции для считывания исходного закодированного сообщения. Возвращают значение типа `string`.

```
std::string decodingFanoShannon(std::string &encodedText,  
std::map<std::string, std::string> &codeTable)
```

Функция для декодирования сообщения. Принимает на вход закодированное сообщение и таблицу кодов, реализованную в виде словаря. Результатом работы функции является строка, содержащая декодированное сообщение.

## Тестирование.



```
Enter a character code table.  
In the format: symbol_code  
m 00  
a 01  
s 10  
h 11  
  
Enter encoded text:0001101101  
  
Input code table:  
m- 00  
a- 01  
s- 10  
h- 11  
Input encoded text:  
0001101101  
Decoded text:  
masha
```

Рисунок 1 – Результат работы программы.

Результаты тестирования программы приведены в табл. 1.

Таблица 1 – Результаты тестирования

Ввод	Вывод
a 0 b 1 0110	abba
H 00 a 01 p 10 y 11 0001101011	Happy
N 0 e 10 w 11 0011	New
Y 00 e 01 a 10 r 110 ! 111 000110110111000110110111	Year!Year!

i 00 + 01 # 10 ‘ ‘ 110 ^ 111 0001011011011011011111111000101	i++# #^^i++
N 0 e 10 w 11 111111111110000000000	wwwwwwwwwNNNNNNNN

### Выводы.

В ходе работы были приобретены навыки работы с закодированным методом Фано-Шеннона сообщениями. А конкретно, с декодированием таких сообщений.

## Приложение А.

### Main.cpp

```
#include <iostream>
#include <string>
#include <fstream>
#include <iostream>
#include <map>
#include <string>

using std::cout;
using std::cin;
using std::cerr;

std::string decodingFanoShannon(std::string &encodedText, std::map<std::string, std::string> &codeTable);

std::string readText(std::istream& in);

std::string readText(std::ifstream& in);

std::map<std::string, std::string> readTable(std::istream& in);

int main(int argc, char* argv[]) {
    setlocale(LC_ALL, "");

    if (argc < 3) {
        std::map<std::string, std::string> codeTable;
        std::string encodedText;
        std::string result;

        if (argc == 2) {
            std::string fileName = argv[1];
            result += "Read encoded text from file " + fileName + "\n";

            std::ifstream inputFile(fileName, std::ios::in);
            codeTable = readTable(inputFile);
            encodedText = readText(inputFile);
            inputFile.close();
        }
        else {
            cout << "Enter a character code table. \nIn the format: symbol_code\n";
            codeTable = readTable(cin);

            cout << "Enter encoded text:";
            encodedText = readText(cin);
        }

        result += "\nInput code table:\n";
        std::map<std::string, std::string>::iterator it = codeTable.begin();
        while (it != codeTable.end()) {
            result += it->second + "- " + it->first + "\n";
            ++it;
        }

        result += "Input encoded text:\n" + encodedText + "\n";
        result += "Decoded text:\n" + decodingFanoShannon(encodedText, codeTable) + "\n\n";

        cout << result;

        std::ofstream resultFile("result.txt", std::ios::app);
        resultFile << result;
        resultFile.close();
    }
}
```

```

        else {
            cerr << "Error: incorrect console's arguments\n";
        }

        return 0;
    }

    std::string readText(std::istream& in) {
        std::string result;
        getline(in, result);

        return result;
    }

    std::map<std::string, std::string> readTable(std::istream& in) {
        std::string oneLineStr;
        std::map<std::string, std::string> codeTable;
        std::string symbol;
        std::string code;

        getline(in, oneLineStr);
        while (oneLineStr != "") {
            symbol = oneLineStr.substr(0, 1);
            code = oneLineStr.substr(2);
            codeTable.insert(std::make_pair(code, symbol));
            getline(in, oneLineStr);
        }

        return codeTable;
    }

    std::string readText(std::ifstream& in) {
        if (!in.is_open()) {
            cerr << "Error: incorrect file name!\n";
            return "";
        }

        std::string result;
        getline(in, result);

        return result;
    }

    std::map<std::string, std::string> readTable(std::ifstream& in) {
        std::string oneLineStr;
        std::map<std::string, std::string> codeTable;
        std::string symbol;
        std::string code;

        getline(in, oneLineStr);
        while (oneLineStr != "") {
            symbol = oneLineStr.substr(0, 1);
            code = oneLineStr.substr(2);
            codeTable.insert(std::make_pair(code, symbol));
            getline(in, oneLineStr);
        }

        return codeTable;
    }

    std::string decodingFanoShannon(std::string &encodedText, std::map<std::string, std::string> &codeTable) {
        std::string code;
        std::string decodedText;

        for (auto elem : encodedText) {

```

```
        code += elem;
        if (codeTable.count(code)) {
            decodedText += codeTable[code];
            code = "";
        }
    }
    return decodedText;
}
```