МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Алгоритмы и структуры данных»

Тема: Кодирование и декодирование: Фано-Шеннона

Студентка гр. 7383 _____ Чемова К.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2018

Содержание

Цель работы

Цель работы: познакомиться с алгоритмом декодирования Фано-Шеннона и его реализацией на языке программирования C++.

Формулировка задачи:

На вход подаётся файл, содержимое которого требуется закодировать алгоритмом Фано-Шеннона, после этого файл нужно раскодировать обратно.

Реализация задачи

Структура struct Elem представляет собой элемент бинарного дерева.

```
struct Elem {
   char c;
   string value;
   Elem* left;
   Elem* right;
       };
```

Структура struct Node представляет собой элемент массива структур, содержащий в себе букву, число ее повторений и указатель на соответствующий элемент бинарного дерева.

```
struct Node {
   int num;
   Elem* ptr;
   char value;
       };
```

Функция void SearchTree кодирует дерево и создает таблицу кодирования.

Функция void Encode кодирует дерево.

Функция void Decode декодирует дерево.

Функция MakeList печатает дерево.

Тестирование

Программа была собрана в компиляторе g++ в OS Linux Ubuntu 16.04 при помощи g++. В других системах тестирование не проводилось. Результаты тестирования приведены в приложении А.

Выводы

В ходе выполнения работы были изучены алгоритмы кодирования и декодирования. Была написана программа, кодирующая и декодирующая алгоритм Фано-Шеннона.

ПРИЛОЖЕНИЕ А

ТЕСТОВЫЕ СЛУЧАИ

Данные файла	Результат
EFRDGFCVHJBKLHGF	[F] - 00 [G] - 010 [H] - 0110 [E] - 0111 [R] - 1000 [D] - 1001 [C] - 1010 [V] - 1011 [J] - 1100 [B] - 1101 [K] - 1110 [L] - 11110 [\N] - 11111 01110010001001010001010101101101100110111101111001100100011111 efrdgfcvhjbklhgf
AAAAAAAAAAAAAABAAAAAAAAAAAAAAAAAAAAABBBBBAAAAAAAAAB	[A] - 0 [B] - 10 [\N] - 11 00000000000000100000000000000000000000000001010101010000000000001011 AAAAAAAAAAAAAABAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBBAAAAAAAAAB
ASDZXASDZXASDZXA	[A] - 00 [S] - 010 [D] - 011 [X] - 10 [Z] - 110 [C] - 1110 [\N] - 1111 00010011110100001001111010000100111101000010011110100101111010111000010011101101110010000111011101111 ASDZXASDZXASDZXA

ПРИЛОЖЕНИЕ Б

КОД ПРОГРАММЫ

```
#include <iostream>
#include <string>
#include <fstream>
#include <cstdlib>
#include <cmath>
using namespace std;

struct Elem {
   char c;
   string value;
   Elem* left;
   Elem* right;
};

struct Node {
   int num;
   Elem* ptr;
   char value;
};


int cmp(const void* a,const void* b) {
   return (*(Node*)b).num-(*(Node*)a).num;
}

void SearchTree(Elem** t,Node* el, string &branch, string &fullBranch, int start,
int end, string &symbols, int &index) {

   cout << "Step " << index++ << "   ";
   for (int i=start;i<=end;i++){
     if (el[i].value == '\n')
        cout << "\\n";
     else cout << el[i].value;
   }
   cout << endl;
   *t = new Elem;
   (*t)->left = NULL;
   (*t)->right = NULL;
   (*t)->value = "";
   (*t)->c = '.';
```

```
    double dS=0;
    int i, S=0;
    string cBranch = "";
    cBranch = fullBranch + branch;
    if (start==end)
    {
        string p = "[";
        if (el[start].value != '\n')
            symbols += p + el[start].value + "] - " + cBranch + "\n";
        else symbols += p + "\\n] - " + cBranch + "\n";
        (*t)->value += cBranch;
        (*t)->c = el[start].value;
        el[start].ptr = *t;
        return;
    }
    for (i=start;i<=end;i++)
        dS+=el[i].num;
    dS /= 2.;
    i=start+1;
    S+=el[start].num;
    while (fabs(dS-(S+el[i].num))<fabs(dS-S) && (i<end)) {
      S += el[i].num;
      i++;
    }
    string zero = "0";
    string one = "1";
    SearchTree(&(*t)->left,el,zero,cBranch,start,i-1,symbols,index);
    SearchTree(&(*t)->right,el,one,cBranch,i,end,symbols,index);
}

void Encode(Node* arr,int index,string copy, string &binary){//кодирует
исходный код в другую строку
  for (int i=0;copy[i]!='\0';i++)
    for (int j = 0; j<index; j++)
      if (copy[i] == arr[j].value)
        binary += arr[j].ptr->value;
}

void print(Elem *t, string &out, int count){
   if (t != NULL) {
       print(t->left,out,count+1);
       for (int i = 0;i < count;i++)
           out = out + "   ";
       if (t->c == '\n')
           out = out + "\\n" + "\n";
```

```
      else out = out + t->c + "\n";
      print(t->right,out,count+1);
   }
}

void MakeList(Elem* t){
   if (t == NULL)
     cout<<"Tree is empty";
   else {
      string tr;
      print(t,tr,0);
      cout << tr;
   }
}

void Decode(Node* arr, string &answer, string binary, int index){
   int len = 0;
   string ptr = "";
   while (len <= binary.length()){
      ptr += binary[len++];
      for (int i = 0; i < index;i++){
         if(ptr == arr[i].ptr->value) {
            answer += arr[i].value;
            ptr.clear();
         }
      }
   }
}

int main() {
   int k;
   string str = "", copy = "", binary;
   ifstream in("text.txt");
   if (!in)
      cout<<"File can not be opened" << endl;
   else {
      Elem* Tree = NULL;
      Node* arr = (Node*) malloc(sizeof(Node));
      int index = 0;
      if (in.peek() == EOF){
         cout << "File is empty" << endl;
         return 0;
      }

      char all[100];
```

```
while (in.getline(all,100)){
   str += string(all) + "\n";
}
in.close();
cout<<str;
copy += str;
for (int j=0;str[j] != '\0';j++){
   k = 0;
   for (int i=j+1;str[i] != '\0';i++) {
      if (str[j] == str[i]){
         while (str[j] == str[i]) {
            str.erase(i,1);
            k++; //счетсчик повторения
         }
      }
   }
   k++;
   index++;
   arr = (Node*) realloc (arr, index*sizeof(Node));
   if (arr != NULL){
      arr[index-1].num = k;
      arr[index-1].value = str[j];
      arr[index-1].ptr = NULL;
   }
   else {
      free(arr);
      cout<<"Error with allocation";
      break;
   }
}

in.close();
qsort(arr,index,sizeof(Node),cmp);
string a = "", b = "", symbols = "";
int stet = 1;
for (int i=0;i<index;i++){
   cout << arr[i].value << " " << arr[i].num << endl;
}
SearchTree(&Tree,arr,a,b,0,index-1,symbols,stet);
cout<<symbols;
Encode(arr,index,copy,binary);
cout<<binary << endl;
string answer = "";
Decode(arr,answer,binary,index);
MakeList(Tree);//печать дерева
```

```
        cout<<answer;
        free(arr);
        free(Tree);
    }
    return 0;
}
```