

Федеральное государственное бюджетное образовательное учреждение
высшего образования Нижегородский государственный технический
университет им. Р.Е. Алексеева

Лабораторная работа №4
«Алгоритм Хаффмана»

Выполнил:
студент 2-го курса
кафедры «Прикладная математика»
Поляков Кирилл

2020г.

Оглавление

Введение.....	3
Документация.....	4
Файл Huffman.cpp.....	4
Содержание Huffman.cpp.....	5
Содержание huffmanTree.cpp.....	6
Файл huffmanTree.h.....	8
Класс huffmanTree.....	9
Структура huffmanTree::Compare.....	13
Содержание Node.cpp.....	14
Класс Node.....	15
Содержание Node.h.....	19
Файл Structs.h.....	20
Структура Arguments.....	21
Структура Data.....	23
Содержание Structs.h.....	25
Содержание WorkWithFiles.cpp.....	26
Файл WorkWithFiles.h.....	28
Класс WorkWithFiles.....	29
Содержание WorkWithFiles.h.....	33
Тестирование.....	34

Введение

Цель работы - написать программу-архиватор на основе статического алгоритма Хаффмана. Архиватор должен уметь сжимать файлы с любым содержанием (пустой, русский текст, английский текст и.т.д) Также разархивированный файл должен быть полностью идентичен файлу, который подавался на вход архиватору

Файл Huffman.cpp

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

Функции

```
int main (int argc, char *argv[])
```

Функции

◆ main()

```
int main ( int    argc,
            char * argv[]
        )
```

См. определение в файле [Huffman.cpp](#) строка 4

Huffman.cpp

```
1 #include <iostream>
2 #include "huffmanTree.h"
3
4 int main(int argc, char *argv[])
5 {
6     huffmanTree huf(argc, argv);
7     huf.print(huf.getRoot());
8     huf.printTable();
9 }
```

huffmanTree.cpp

```
1 #include "huffmanTree.h"
2
3 Arguments huffmanTree::ArgumentsReading(int argc, char* argv[])
4 {
5     Arguments args;
6     for (int i = 0; i < argc; i++)
7     {
8         if (string(argv[i]).find("decompress") != string::npos)
9             args.mode = args.decompress;
10        else if (string(argv[i]).find("compress") != string::npos)
11            args.mode = args.compress;
12        else if (string(argv[i]).find("input_filename") != string::npos ||
13                string(argv[i]).find("inputFilename") != string::npos)
14        {
15            if (string(argv[i]).find("=") != string::npos)
16                args.inputFilename = string(argv[i]).substr(string(argv[i]).find('=') +
171, string(argv[i]).length() - 1);
18            else if (i < argc)
19                args.inputFilename = argv[++i];
20        }
21        else if (string(argv[i]).find("output_filename") != string::npos ||
22                string(argv[i]).find("outputFilename") != string::npos)
23        {
24            if (string(argv[i]).find("=") != string::npos)
25                args.outputFilename = string(argv[i]).substr(string(argv[i]).find('=') +
261, string(argv[i]).length() - 1);
27            if (i < argc)
28                args.outputFilename = argv[++i];
29        }
30        else if (string(argv[i]).find("prefix_phrase") != string::npos ||
31                string(argv[i]).find("prefixPhrase") != string::npos)
32        {
33            if (string(argv[i]).find("=") != string::npos)
34                args.prefix = string(argv[i]).substr(string(argv[i]).find('=') + 1,
35string(argv[i]).length() - 1);
36            if (i < argc)
37                args.prefix = argv[++i];
38        }
39    }
40    return args;
41 }
42
43 huffmanTree::huffmanTree(int argc, char* argv[])
44 {
45     Arguments args = ArgumentsReading(argc, argv);
46     if (WorkWithFiles::FileReading(args, data))
47     {
48         list<Node*> tree;
49
50         map<char, int>::iterator iter;
51         for (iter = data.map.begin(); iter != data.map.end(); ++iter)
52         {
53             Node* temp = new Node();
54             temp->setValue(iter->second, iter->first);
55             tree.push_back(temp);
56         }
57
58         if (tree.size() == 0)
59             return;
60
61         while (tree.size() != 1)
62         {
63             tree.sort(Compare());
64
65             Node* sonLeft = tree.front();
66             tree.pop_front();
67             Node* sonRight = tree.front();
68             tree.pop_front();
69
70             Node* parent = new Node(sonLeft, sonRight);
71             tree.push_front(parent);
72         }
73     }
74 }
```

```

71     }
72
73     root = tree.front();
74     buildTable(root);
75     WorkWithFiles::SaveInFile(args, data);
76 }
77 }
78
79 Node* huffmanTree::getRoot()
80 {
81     return root;
82 }
83
84 void huffmanTree::buildTable(Node* root)
85 {
86     if (root->getLeftChild() != NULL)
87     {
88         code.push_back(0);
89         buildTable(root->getLeftChild());
90     }
91     if (root->getRightChild() != NULL)
92     {
93         code.push_back(1);
94         buildTable(root->getRightChild());
95     }
96     if (root->getLetter())
97         data.table[root->getLetter()] = code;
98
99     if (code.size() != 0)
100         code.pop_back();
101 }
102
103 void huffmanTree::print(Node* root, unsigned k)
104 {
105     if (root != NULL)
106     {
107         print(root->getLeftChild(), k + 3);
108
109         for (unsigned i = 0; i < k; i++)
110             cout << " ";
111
112         if (root->getLetter())
113             if (root->getLetter() == '\n')
114                 cout << root->getFrequency() << "\\n" << endl;
115             else
116                 cout << root->getFrequency() << "(" << root->getLetter() << ")" << endl;
117         else
118             cout << root->getFrequency() << endl;
119         print(root->getRightChild(), k + 3);
120     }
121 }
122
123 void huffmanTree::printTable()
124 {
125     for (map<char, vector<bool>>::iterator iter = data.table.begin(); iter !=
126         data.table.end(); ++iter)
127     {
128         if (iter->first == '\n')
129             cout << "\\n" << " : ";
130         else
131             cout << iter->first << " : ";
132         for (int i = 0; i < iter->second.size(); i++)
133             cout << iter->second[i];
134         cout << " (" << data.map[iter->first] << ")" << endl;
135     }
136     for (int i = 0; i < data.str.length(); i++)
137         for (int j = 0; j < data.table[data.str[i]].size(); j++)
138             cout << data.table[data.str[i]][j];
139 }

```

Файл huffmanTree.h

```
#include <map>
#include <list>
#include <string>
#include <vector>
#include <fstream>
#include <iostream>
#include "Node.h"
#include "Structs.h"
#include "WorkWithFiles.h"
```

Классы

class	huffmanTree
struct	huffmanTree::Compare

Класс `huffmanTree`

```
#include <huffmanTree.h>
```

Классы

```
struct Compare
```

Открытые члены

```
huffmanTree (int argc, char *argv[])
```

```
Node * getRoot ()
```

```
void buildTable (Node *root)
```

```
void print (Node *root, unsigned k=0)
```

```
void printTable ()
```

Закрытые члены

```
Arguments ArgumentsReading (int argc, char *argv[])
```

Закрытые данные

```
Node * root
```

```
vector< bool > code
```

```
Data data
```

Подробное описание

Класс, реализующий дерево Хаффмана

См. определение в файле `huffmanTree.h` строка **18**

Конструктор(ы)

◆ `huffmanTree()`

```
huffmanTree::huffmanTree ( int    argc,  
                           char * argv[]  
                           )
```

Конструктор класса с аргументами

Аргументы

argc Количество аргументов, переданных в консоль

argv Аргументы, переданные в консоль

См. определение в файле [huffmanTree.cpp](#) строка **41**

Методы

◆ ArgumentsReading()

```
Arguments huffmanTree::ArgumentsReading ( int    argc,  
                                           char * argv[]  
                                           )
```

private

Функция, для чтения аргументов, переданных в консоль

Аргументы

argc Количество аргументов, переданных в консоль

argv Аргументы, переданные в консоль

Возвращает

Структура **Arguments**

См. определение в файле [huffmanTree.cpp](#) строка **3**

◆ buildTable()

```
void huffmanTree::buildTable ( Node * root )
```

Функция для построения таблицы кодировки символов

Аргументы

root Корневой элемент дерева

См. определение в файле [huffmanTree.cpp](#) строка **84**

◆ getRoot()

```
Node * huffmanTree::getRoot ( )
```

Функция для получения корневого элемента дерева

Возвращает

Корневой элемент дерева

См. определение в файле [huffmanTree.cpp](#) строка **79**

◆ print()

```
void huffmanTree::print ( Node *   root,  
                        unsigned k = 0  
                        )
```

Функция вывода дерева в консоль

Аргументы

root Корневой элемент дерева

k Разрыв между элементами дерева

См. определение в файле [huffmanTree.cpp](#) строка **103**

◆ printTable()

```
void huffmanTree::printTable ( )
```

Функция вывода таблицы кодировки символов

См. определение в файле [huffmanTree.cpp](#) строка **123**

Данные класса

◆ code

`vector<bool> huffmanTree::code`

private

См. определение в файле [huffmanTree.h](#) строка 21

◆ data

Data `huffmanTree::data`

private

См. определение в файле [huffmanTree.h](#) строка 22

◆ root

Node* `huffmanTree::root`

private

См. определение в файле [huffmanTree.h](#) строка 20

Объявления и описания членов классов находятся в файлах:

- [huffmanTree.h](#)
- [huffmanTree.cpp](#)

Структура `huffmanTree::Compare`

Открытые члены

```
bool operator() (Node *left, Node *right) const
```

Подробное описание

Структура, реализующая сравнение двух элементов дерева

См. определение в файле [huffmanTree.h](#) строка 27

Методы

◆ `operator()()`

```
bool huffmanTree::Compare::operator() ( Node * left,  
                                         Node * right  
                                         )      const
```

inline

Переопределенный оператор скобок

Аргументы

left Левый потомок

right Правый потомок

Возвращает

true - левый меньше правого (по frequency величине), false - в другом случае

См. определение в файле [huffmanTree.h](#) строка 36

Объявления и описания членов структуры находятся в файле:

- [huffmanTree.h](#)

Node.cpp

```
1  #include "Node.h"
2
3  Node::Node()
4  {
5  }
6
7  Node::Node(Node* left, Node* right)
8  {
9      this->childLeft = left;
10     this->childRight = right;
11     this->frequency = left->getFrequency() + right->getFrequency();
12 }
13
14 void Node::setValue(int frequency, char letter)
15 {
16     this->frequency = frequency;
17     this->letter = letter;
18 }
19
20 int Node::getFrequency()
21 {
22     return frequency;
23 }
24
25 char Node::getLetter()
26 {
27     return letter;
28 }
29
30 Node* Node::getLeftChild()
31 {
32     return childLeft;
33 }
34
35 Node* Node::getRightChild()
36 {
37     return childRight;
38 }
```

Класс Node

```
#include <Node.h>
```

Открытые члены

Node ()
Node (Node *left, Node *right)
void setValue (int frequency , char letter)
int getFrequency ()
char getLetter ()
Node * getLeftChild ()
Node * getRightChild ()

Закрытые данные

int frequency
char letter
Node * childLeft
Node * childRight

Подробное описание

Класс реализующий элемент дерева Хаффмана

См. определение в файле **Node.h** строка **6**

Конструктор(ы)

◆ Node() [1/2]

Node::Node ()

Конструктор класса по умолчанию

См. определение в файле **Node.cpp** строка **3**

◆ Node() [2/2]

```
Node::Node ( Node * left,  
            Node * right  
            )
```

Конструктор класса с аргументами

Аргументы

left Указатель на левого ребёнка

right Указатель на правого ребёнка

См. определение в файле **Node.cpp** строка 7

Методы

◆ getFrequency()

```
int Node::getFrequency ( )
```

Функция для получения частотности символа

Возвращает

Частотность символа

См. определение в файле **Node.cpp** строка 20

◆ getLeftChild()

```
Node * Node::getLeftChild ( )
```

Функция для получения указателя на левого ребёнка

Возвращает

Указатель на левого ребёнка

См. определение в файле **Node.cpp** строка 30

◆ getLetter()


```
char Node::getLetter ( )
```

Функция для получения символа

Возвращает

Символ

См. определение в файле **Node.cpp** строка **25**

◆ getRightChild()

```
Node * Node::getRightChild ( )
```

Функция для получения указателя на правого ребёнка

Возвращает

Указатель на правого ребёнка

См. определение в файле **Node.cpp** строка **35**

◆ setValue()

```
void Node::setValue ( int    frequency,  
                     char letter  
                     )
```

Функция для установки значений элемента

Аргументы

frequency Частотность символа

letter Символ

См. определение в файле **Node.cpp** строка **14**

Данные класса

◆ childLeft

Node* Node::childLeft

private

См. определение в файле **Node.h** строка **10**

◆ childRight

Node * Node::childRight

private

См. определение в файле **Node.h** строка **10**

◆ frequency

int Node::frequency

private

См. определение в файле **Node.h** строка **8**

◆ letter

char Node::letter

private

См. определение в файле **Node.h** строка **9**

Объявления и описания членов классов находятся в файлах:

- **Node.h**
- **Node.cpp**

Node.h

```
1  #pragma once
2
6  class Node
7  {
8      int frequency;           // Частотность символа
9      char letter;             // Символ
10     Node *childLeft, *childRight; // Указатель на левого ребёнка и указатель на
    правого ребёнка
11 public:
12     Node();
13     Node(Node* left, Node* right);
14
15     void setValue(int frequency, char letter);
16     int getFrequency();
17     char getLetter();
18     Node* getLeftChild();
19     Node* getRightChild();
20 };
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
```

Файл Structs.h

```
#include <string>
#include <map>
#include <vector>
```

[См. исходные тексты.](#)

Классы

struct	Arguments
--------	------------------

struct	Data
--------	-------------

Структура Arguments

```
#include <Structs.h>
```

Открытые типы

```
enum modes { compress = 1, decompress }
```

Открытые атрибуты

```
enum Arguments::modes mode  
    string inputFilename  
    string outputFilename  
    string prefix
```

Подробное описание

Структура, хранящая данные аргументов, переданных через консоль

См. определение в файле **Structs.h** строка **10**

Перечисления

◆ modes

```
enum Arguments::modes
```

Элементы перечислений

compress	
decompress	

См. определение в файле **Structs.h** строка **12**

Данные класса

◆ inputFilename

string Arguments::inputFilename

См. определение в файле **Structs.h** строка 13

◆ mode

enum **Arguments::modes** Arguments::mode

◆ outputFilename

string Arguments::outputFilename

См. определение в файле **Structs.h** строка 14

◆ prefix

string Arguments::prefix

См. определение в файле **Structs.h** строка 15

Объявления и описания членов структуры находятся в файле:

- **Structs.h**

Структура Data

```
#include <Structs.h>
```

Открытые атрибуты

string **str**

map< char, vector< bool > > **table**

map< char, int > **map**

Подробное описание

Структура, хранящая промежуточные данные

См. определение в файле **Structs.h** строка **21**

Данные класса

◆ map

map<char, int> Data::map

См. определение в файле **Structs.h** строка **25**

◆ str

string Data::str

См. определение в файле **Structs.h** строка **23**

◆ table

map<char, vector<bool> > Data::table

См. определение в файле **Structs.h** строка **24**

Объявления и описания членов структуры находятся в файле:

- **Structs.h**

Structs.h

```
1  #pragma once
2  #include <string>
3  #include <map>
4  #include <vector>
5
6  using namespace std;
10 struct Arguments
11 {
12     enum modes{compress = 1, decompress} mode; // Режим работы
13     string inputFilename; // Имя входного файла
14     string outputFilename; // Имя выходного файла
15     string prefix; // Префикс (для идентификации файла)
16 };
17
21 struct Data
22 {
23     string str; // Исходный текст
24     map <char, vector<bool>> table; // Таблица кодировки (символ, двоичная запись)
25     map <char, int> map; // Таблица частотности (символ, количество
    повторения символа в тексте)
26 };
```

WorkWithFiles.cpp

```
1  #include "WorkWithFiles.h"
2
3  bool WorkWithFiles::ReadStandartFile(string filename, Data& data)
4  {
5      ifstream file(filename);
6      if (file.is_open())
7      {
8          string str;
9          while (getline(file, str))
10         {
11             data.str += str + '\n';
12             for (int i = 0; i < str.length(); i++)
13                 data.map[str[i]]++;
14             data.map['\n']++;
15         }
16     }
17     else
18     {
19         cout << "File opening error." << endl;
20         return false;
21     }
22
23     file.close();
24     return true;
25 }
26
27 bool WorkWithFiles::ReadBinaryFile(Arguments args, Data& data)
28 {
29     ifstream in(args.inputFilename, ios::binary | ios::in);
30     string prefix;
31
32     if (!in.fail())
33     {
34         string str;
35         map<vector<bool>, char> reverseTable;
36         vector<bool> code;
37         int numEl = 1;
38         int sizeTable;
39         int sizeText;
40         int tempInt = 0;
41         char c;
42
43         while (!in.eof())
44         {
45             in.get(c);
46
47             if (c != ' ' && numEl < 4)
48                 str += c;
49             else
50             {
51                 if (numEl == 1)
52                 {
53                     prefix = str;
54                     str.clear();
55                     numEl++;
56                     if (prefix != args.prefix)
57                     {
58                         cout << "The discrepancy between the prefixes" << endl;
59                         return false;
60                     }
61                 }
62                 else if (numEl == 2)
63                 {
64                     sizeTable = atoi(str.c_str());
65                     str.clear();
66                     numEl++;
67                 }
68                 else if (numEl == 3)
69                 {
70                     sizeText = atoi(str.c_str());
71                     str.clear();
72                     numEl++;
73                 }
74             }
75         }
76     }
77 }
```

```

74         else if (numEl == 4)
75         {
76             int num = 0;
77             char temp = c;
78             string tem;
79             while (num < sizeTable)
80             {
81                 in.get(c);
82                 if (c == '1' || c == '0')
83                     tem += c;
84                 else
85                 {
86                     num++;
87                     vector<bool> code;
88                     for (int i = 0; i < tem.length(); i++)
89                         (tem[i] == '1') ? (code.push_back(1)) : (code.push_back(0));
90                     data.table.emplace(tem, code);
91                     temp = c;
92                     tem.clear();
93                 }
94             }
95             numEl++;
96             for (map<char, vector<bool>>::iterator iter = data.table.begin();
iter != data.table.end(); iter++)
                reverseTable.emplace(iter->second, iter->first);
97         }
98     }
99     else
100     {
101         for (int num = 0; num <= 7; num++)
102         {
103             if (c & (1 << (7 - num)))
104                 code.push_back(1);
105             else
106                 code.push_back(0);
107
108             if (reverseTable.count(code) > 0 && tempInt < sizeText - 1)
109             {
110                 tempInt++;
111                 str += reverseTable[code];
112                 code.clear();
113             }
114         }
115     }
116 }
117 }
118 StandartSaveInFile(args.outputFilename, str);
119 }
120 else cout << "File not found" << endl;
121 in.close();
122 return false;
123 }
124
125 bool WorkWithFiles::StandartSaveInFile(string filename, string strSave)
126 {
127     ofstream out(filename);
128     if (out.is_open())
129     {
130         out << strSave;
131         out.close();
132         return true;
133     }
134     return false;
135 }
136
137
138
139 bool WorkWithFiles::FileReading(Arguments args, Data& data)
140 {
141     if (args.mode == args.compress)
142         return ReadStandartFile(args.inputFilename, data);
143     if (args.mode == args.decompress)
144         return ReadBinaryFile(args, data);
145 }
146
147 bool WorkWithFiles::SaveInFile(Arguments args, Data& data)
148 {
149     if (args.mode == args.compress)
150         return BinarySaveInFile(args, data);
151     else if (args.mode == args.decompress)
152         return StandartSaveInFile(args.outputFilename, data.str);

```

Файл WorkWithFiles.h

```
#include <string>
#include <fstream>
#include <iostream>
#include <iterator>
#include "Structs.h"
```

Классы

```
class WorkWithFiles
```

Класс WorkWithFiles

```
#include <WorkWithFiles.h>
```

Открытые статические члены

```
static bool FileReading (Arguments args, Data &data)
```

```
static bool SaveInFile (Arguments args, Data &data)
```

Закрытые статические члены

```
static bool ReadStandartFile (string filename, Data &data)
```

```
static bool ReadBinaryFile (Arguments args, Data &data)
```

```
static bool StandartSaveInFile (string filename, string strSave)
```

```
static bool BinarySaveInFile (Arguments args, Data &data)
```

Подробное описание

Класс, реализующий работу с файлом

См. определение в файле [WorkWithFiles.h](#) строка 14

Методы

◆ BinarySaveInFile()

```
static bool WorkWithFiles::BinarySaveInFile ( Arguments args,
                                              Data & data
                                              )
```

[inline](#) [static](#) [private](#)

Функция, для сохранения заархивированных данных в файл

Аргументы

filename Имя выходного файла

strSave Текст для сохранения в файл

Возвращает

true - запись прошла успешно, false - при записи произошла ошибка

См. определение в файле [WorkWithFiles.h](#) строка 47

◆ FileReading()

```
bool WorkWithFiles::FileReading ( Arguments args,  
                                Data & data  
                                )
```

static

Функция для чтения файла

Аргументы

args Данные аргументов, переданных через консоль

data Данные, в которые будет произведена запись

Возвращает

true - чтение прошло успешно, false - при чтении произошла ошибка

См. определение в файле [WorkWithFiles.cpp](#) строка **139**

◆ ReadBinaryFile()

```
bool WorkWithFiles::ReadBinaryFile ( Arguments args,  
                                     Data & data  
                                     )
```

static

private

Функция, для чтения бинарного(заархивированного) файла

Аргументы

filename Имя входного файла

data Данные, в которые будет произведена запись

Возвращает

true - чтение прошло успешно, false - при чтении произошла ошибка

См. определение в файле [WorkWithFiles.cpp](#) строка **27**

◆ ReadStandartFile()

```
bool WorkWithFiles::ReadStandartFile ( string  filename,  
                                         Data & data  
                                         )
```

static private

Функция, для чтения стандартного файла

Аргументы

filename Имя входного файла

data Данные, в которые будет произведена запись

Возвращает

true - чтение прошло успешно, false - при чтении произошла ошибка

См. определение в файле [WorkWithFiles.cpp](#) строка 3

◆ SaveInFile()

```
bool WorkWithFiles::SaveInFile ( Arguments args,  
                                 Data & data  
                                 )
```

static

Функция для записи данных в файл

Аргументы

args Данные аргументов, переданных через консоль

data Данные, из которых будет произведена запись

Возвращает

true - запись прошла успешно, false - при записи произошла ошибка

См. определение в файле [WorkWithFiles.cpp](#) строка 147

◆ StandartSaveInFile()

```
bool WorkWithFiles::StandartSaveInFile ( string filename,  
                                         string strSave  
                                         )
```

static private

Функция, для стандартного сохранения данных в файл

Аргументы

filename Имя выходного файла

strSave Текст для сохранения в файл

Возвращает

true - запись прошла успешно, false - при записи произошла ошибка

См. определение в файле [WorkWithFiles.cpp](#) строка 125

Объявления и описания членов классов находятся в файлах:

- [WorkWithFiles.h](#)
- [WorkWithFiles.cpp](#)

WorkWithFiles.h

```
1  #pragma once
2  #include <string>
3  #include <fstream>
4  #include <iostream>
5  #include <iterator>
6
7  #include "Structs.h"
8
9  using namespace std;
10
11 static class WorkWithFiles
12 {
13     static bool ReadStandartFile(string filename, Data& data);
14     static bool ReadBinaryFile(Arguments args, Data& data);
15     static bool StandartSaveInFile(string filename, string strSave);
16     static bool BinarySaveInFile(Arguments args, Data& data)
17     {
18         ofstream out(args.outputFilename, ios::binary | ios::out);
19         char space = ' ';
20
21         out << args.prefix << space;
22
23         out << data.table.size() << space;
24         out << data.str.length() << space;
25
26         for (map<char, vector<bool>>::iterator iter = data.table.begin(); iter !=
27 data.table.end(); iter++)
28         {
29             out << iter->first;
30             for (int i = 0; i < iter->second.size(); i++)
31                 out << iter->second[i];
32         }
33         out << space;
34
35         char buf = 0;
36         int count = 0;
37         for (int i = 0; i < data.str.length(); i++)
38             for (int j = 0; j < data.table[data.str[i]].size(); j++)
39             {
40                 buf = buf | data.table[data.str[i]][j] << (7 - count);
41                 count++;
42                 if (count == 8)
43                 {
44                     out << buf;
45                     count = buf = 0;
46                 }
47             }
48         out.close();
49         return true;
50     }
51 public:
52     static bool FileReading(Arguments args, Data& data);
53     static bool SaveInFile(Arguments args, Data& data);
54 };
```

Тестирование

--compress

--output_filename output.txt

--input_filename input.txt

--prefix_phrase HF11

Файл input.txt (2,91 Кб)

Таким образом, рамки и место обучения кадров играет важную роль в формировании существующих финансовых и административных условий. Повседневная практика показывает, что выбранный нами инновационный путь в значительной степени обуславливает создание соответствующих условий активизации. Задача организации, в особенности же выбранный нами инновационный путь создаёт предпосылки качественно новых шагов для модели развития? Соображения высшего порядка, а также повышение уровня гражданского сознания представляет собой интересный эксперимент проверки системы масштабного изменения ряда параметров.

Повседневная практика показывает, что сложившаяся структура организации требует от нас системного анализа существующих финансовых и административных условий. Задача организации, в особенности же постоянное информационно-техническое обеспечение нашей деятельности обеспечивает актуальность существующих финансовых и административных условий. Повседневная практика показывает, что начало повседневной работы по формированию позиции влечет за собой процесс внедрения и модернизации системы обучения кадров, соответствующей насущным потребностям. Дорогие друзья, курс на социально-ориентированный национальный проект влечет за собой процесс внедрения и модернизации системы обучения кадров, соответствующей насущным потребностям. Задача организации, в особенности же социально-экономическое развитие создаёт предпосылки качественно новых шагов для модели развития.

Дорогие друзья, дальнейшее развитие различных форм деятельности играет важную роль в формировании экономической целесообразности принимаемых решений. Практический опыт показывает, что сложившаяся структура организации обеспечивает широкому кругу специалистов участие в формировании направлений прогрессивного развития! Задача организации, в особенности же рамки и место обучения кадров требует определения и уточнения системы масштабного изменения ряда параметров. С другой стороны начало повседневной работы по формированию позиции способствует подготовке и реализации модели развития. Разнообразный и богатый опыт новая модель организационной деятельности играет важную роль в формировании новых предложений.

Задача организации, в особенности же начало повседневной работы по формированию позиции играет важную роль в формировании экономической целесообразности принимаемых решений? Равным образом сложившаяся структура организации напрямую зависит от ключевых компонентов планируемого обновления! Равным образом постоянный количественный рост и сфера нашей активности представляет собой интересный эксперимент проверки ключевых компонентов планируемого обновления? С другой стороны рамки и место обучения кадров обеспечивает актуальность соответствующих условий активизации. Повседневная практика показывает, что новая модель организационной деятельности влечет за собой процесс внедрения и модернизации соответствующих условий активизации.

Повседневная практика показывает, что консультация с профессионалами из IT...

Файл output.txt(2,06Кб)

HF11 47 2983

ë11000110100д110001101013001110110п110001100P1100010010C1100010011T110001101110a10116100000b11111г1101101д110011
e1001ж11011001з110010и1110100001к110111л110101м111100н10100000п111101р0010с0100т0101у110100ф11000111х1000100ц001
100с0001111ш11011000щ11000101ы100011ь1000101э001110111ю00111100я110000
00111010 011!11000110110,0011010-1100010000.0011011?1100010001110001101111T110001101110 Жхпю1!«CГЛiншутQC|h|
Ч'чжи>цтмъшц-!2!E!хажеoi!,'уясC!гх] ьq;по°Е+«ЭGилюф&ю0}':ухаэ+{~{кп(яг!™кгб°е8]_ньё~мбу/Э|
+ык-юуц-!2,ы5МАуk!чiр-!уЭтг! *«э!бфGүj?Р»э{э•™ьмкi;+ш-·]•™ьiэ,!МРЕэi/сАWU.İO}т?фРе!ю<л;ж!э)пР'лi=мщЕьХ!ССд{!г}
э°!БьлМi/тц!<ДАБi!fk°!К6'h~,Г=му!•sM!Ггшльz!э'ц•мэт>КУ К°»!йНh•эр•h !сг'°E6g}уш.тi_н?
%i4дYт7e!Bb!6!щy55ш2г;~щ_ 'м>лил!>«Kокp~oSSiFч°!NSLx
5!|=э<Б0hг!лY[!Q!лиi;ль'бм«
кC\$°"п°6!uzцVYLYЕю!1z#3кЕашв>wцk\лbCът#C5иM!мзгб!Q!лиi;K?°, €!jцeь!F!B_v8/,O!C!Y%|!|vx,шц>kГХмл/9бGхг!уе
%="ыч°пуТх<А!+M1e!ьЕиЦЕ!ггЕщю~ur)Zчк8UDф!Y6сfй9фю!i!й^лыбфюуGы•4ПРКПСф}':ур...•А!7ихаjвегЦC—!бя-ц*т!Q,!!
фГ'°ЕяS°Mv!уf9°м-Ожъ<>F!D!5ш=п24P!ьЕ/ф9<0ерY!Jсб!|
!ëхмгPCoKMгK.(!ль!):!бoSбPД!тC!Этреi8Ул!J...ьBw!wэдщW•r!Q_м!«ГььKk°!|ь!Ywefw4дYт6 Pщ°Бпу!щ!б/еQ!YMY...
X!|м!PГ°PРЕГГ!нушцБтем-огом ,j,/=™h!лц-!«@нëYз)ë-шцZ!ajh!тiтэ »уюп°_Hпт?G±{Gюzаь!Ojктя/'!ил!,!з!e!Qp4ysбф
±2КтЯ/к+л+>JсБхун!и<А!цм+•!S!ьPг!-!wч!O!ы°жwбPуц!°бВМке бi°!jэ.°тэив2к%°,ь!ж!oj!|!э!ж!iCSeoтУ=сMFP!bpl!|!иe!Втем-Ож
°цопь«шв!ц4{->тiм°чi°!цS!гг-ч!O!ы°эw!й_хлы_мб°ywm!«i!тц!шкi;+ш-·]•™ьiэ,!МРЕэi,İншутQC|h|ч'чжи>ц!Q!
±!Ns-ч'щи ьлм!г,щ!тУан
!Cмj!льм!«кi!i!YR6с7°!гBCA
K!хuз!иьНу?Y!+б,орсБуд?узJ.3№S!Y!i!тУзь! y|Ко°Кмо~«з°ey!-°бс+Кбi!°т!O!Q!8Xц5т?х?!s~V!мЧeфP!j!э!кбPЕзыJe_пfг8Ик!эг,у€~л
NРьqГ°9к6fе!gGgKON!мЧeфw! a!T!Yмл«>x!Cфьк!B!W XЯYг,у€~лб?F!gшГiГiчiзд5люсБуд?убOы!qт>€Y°~!«еА!цГХЮ!щFS
±5Pв/!iф~!|!«сЖк!хЮЕа4RYУP-,ьшцYKÖ!TГH!<i!ь~+яS!тPгчz°!igЕаэq!>
Y€(!Y°Г!<JяQшд!y!y!au/q!i(i!ьёбd!_43'•w±0»э{ф!уьSь-я°Z!|/oдr°8Yтцк!ж!ль!й!юп!N?т#ьуиoГ~лиЕяшб† Yэф°«Ево2Уh...Y,!!+y~|
ь°PД!5ш=п2b Odюэк°этсbPЕ!ьP/ф9<C!ëЯвнуцVgq•ль!O\$хх?Jч°ЮЯJ{K?бo}z!счгpg:Е°!ЭвVvg
Y!OpYц-!«!цO•y[A ...!«!D!«!м!>!пбъ•vVgs@!тE!Pж/Dz&Jэ!сYчiщYKжOб!фiй9фю!i!й^лыбфюуGы•4ПРЧ!ьк!|!ОБJф†<Yр«цэзцogqifo

--decompress

--output_filename output.txt

--input_filename input.txt

--prefix_phrase HF11

Файл input.txt(2,06Кб)

```
HF11 47 2983
ë11000110100д110001101013001110110P110001100P1100010010C1100010011T110001101110a10116100000B11111г1101101д110011
e1001ж11011001з110010и11101100001к110111л110101м111100н1010000п111101р0010с0100т0101у110100ф11000111х1000100ц001
100ч001111ш1101100щ1100010ы100011ь1000101э001110111ю0011110я110000
00111010 011!11000110110,0011010-1100010000.0011011?1100010001I110001101111T110001101110 ЖхпЮ1П^СГЛІНщутQC|h|
Ч^чji}цТМъыщ-П2Е|хажвеoi!,уяCS|гх] Ыq;ПО^Е+^эG|ильЦф&Ю0}' :цХаэ+э{ {~Кп(яг!™кгЪ°Е8]_ньЕ~МбУ/Э|
+ыК~ЮУЦ-П2,ы5МАЎк|Чіг-ПУэт| *<э|бфGүj?Р»э{э•™ьмкІ; >ш-• ]•™ьіэ,,МРЕзі/сАWU.İO}т?фрЕѢ|о<| Л;Жьэ|ПР'лі=мщЕьХ|сд{Пг}
э°|БьлМІ/тц|ДАБІfk°|К6'h~,Г=мУЉ•smіГгШьїз|э`ц•мэт>КЎ К°»|ЙNh•эр•h |сГ'”Ебg}Ущ.ті_м?
%І4дУт7е!Bb|6Іщу55ш2Г;~щ_’>лил|”“КОкр~•oSSİfч”|MSLx
5|~э<Б0hг!|лҮ[|ЉлIi;|ь”Бм«
кС$”П”6!uzцVУYЕю†1z#зкеАшв>wцк\ьVСъѣС5иMіμзgb|ЉлIi;К?°, €|јцєѢF|B_V8/,O|C|Y%| |;вX,Щ>кГХмL/9БгХг|ує
%=”ыч”пуТх<А|+М1е|ьЕиЄЙ|г€щю~ur)Зчк8ЦDф|Ў6сғй9фю|Ій^лыбыфоуГы•4ПркПСсф}' :уР...А|7ихајзвегUC–|бя-ц*тЉ,|
фГ’€ЯS™Mv|уф9’м-Ожњ<>Fд|5ш=п24Р|ь€/ф9<0егУ|]сь|!
|ӘхмгРСокМгК.(Ѣы):ЉBOSbPд|тТс|ЭТреі8Uл|]...ьBwwэдщW•г|Љ_м«Гьькк°| ЫЙWefw4дУтБ Рщ°Бпућщ|б/eq|ЎМУ...
Х!μЦРГ°РЕГГ|нущЦБтем-ОГОм ,j,/=™һлЦ-▲@пёЎэ)ё-ш-щзZ▲іjhїтњэ »уЮП”_НРт?G±{ГЮзaby|ојктя/`лил|,|з^е!Qp4ysбфг
±2Ктя/К+л^>ЈєсБхуы|ц<А|ум+•|СьРг!-WчГ|О!ы°жwьРуц”ьВМке ЪІ*|јэ.°тзИв2к%°,ьж|ој| |э|ЖІCSeoТУ=CMFP|bpL|)ієиVБтем-Ож
”цОп«шв|Ц4{->тім™чі”|С|Гг^~чГ|О!ы°эWй_хли_мб”ўмт▲І/тц|цкІ; >ш-• ]•™ьіэ,,МРЕзі,ІНщутQC|h|Ч^чji}©L|]
±ЙNS-Ч’щ| ълм|г,щ|туАп
|См|ьлм|аКі|ІЙR6с7”ігВсА
КХузІъНu?У|+ѣ,орсБуд?узJ.3N$Ў|ІТУьЗь| у|Ко^Кмо~▲з^ey|~°6с+КѢБІ*Ѣ|O|Љц8Хц5т?х?|s-V|мчєфр€|јз|кбРЕзыJe_пфг8иК|эг,у€~л
№РьqГ°9к6fє|lgGGkOП|мчєfw| а|Т|умл<>х|сфьк|В|W ХЯҮг,у€~лѢ?F\gsнГгІчізд5юсБуд?убьOыьqт>€Y“~ |еА|уГХЮ|щFS
±5Рв/|іФ~1|▲СжьКҺюЕа4RYуР-,ьшщYКOІТГН|<Іъь~+ЯS†гєчз*іGЯєqі>
Ў€(ПY°Г|КЯqшд|у|Ў|ау/qІ(іъЕ6d|_43’•W±0»э{ф|уьСь-я\%Z|/|dГ*8Yятцк|жьлЙ|юП)N?Ѣ#ьуиОГ~луЕмшб† Үәф”▲Ево2ућ...ў,|+у~|
ь”Рд|5ш=п2b Одюэк»этSbPEльР|ь€/ф9<с%ЕявпупцVgq•њIO$xx?Јч^ЮЯЯ{К?ь0|зІсчҫrg:Е’Љэvvg
Ү|орYЦ-П^|цО•у|А ...ь|ЙD|ом”|»пбь•vVgs@|тЕіРж/Dz&зэ|с|щц|ж|оѢф±fй9фю|Ій^лыбыфоуГы•4ПркЦьк|л|обзф†<үр<цъзщ<orqifo
```

Файл output.txt (2,91 Кб)

Таким образом, рамки и место обучения кадров играет важную роль в формировании существующих финансовых и административных условий. Повседневная практика показывает, что выбранный нами инновационный путь в значительной степени обуславливает создание соответствующих условий активизации. Задача организации, в особенности же выбранный нами инновационный путь создаёт предпосылки качественно новых шагов для модели развития? Соображения высшего порядка, а также повышение уровня гражданского сознания представляет собой интересный эксперимент проверки системы масштабного изменения ряда параметров.

Повседневная практика показывает, что сложившаяся структура организации требует от нас системного анализа существующих финансовых и административных условий. Задача организации, в особенности же постоянное информационно-техническое обеспечение нашей деятельности обеспечивает актуальность существующих финансовых и административных условий. Повседневная практика показывает, что начало повседневной работы по формированию позиции влечет за собой процесс внедрения и модернизации системы обучения кадров, соответствующей насущным потребностям. Дорогие друзья, курс на социально-ориентированный национальный проект влечет за собой процесс внедрения и модернизации системы обучения кадров, соответствующей насущным потребностям. Задача организации, в особенности же социально-экономическое развитие создаёт предпосылки качественно новых шагов для модели развития.

Дорогие друзья, дальнейшее развитие различных форм деятельности играет важную роль в формировании экономической целесообразности принимаемых решений. Практический опыт показывает, что сложившаяся структура организации обеспечивает широкому кругу специалистов участие в формировании направлений прогрессивного развития! Задача организации, в особенности же рамки и место обучения кадров требует определения и уточнения системы масштабного изменения ряда параметров. С другой стороны начало повседневной работы по формированию позиции способствует подготовке и реализации модели развития. Разнообразный и богатый опыт новая модель организационной деятельности играет важную роль в формировании новых предложений.

Задача организации, в особенности же начало повседневной работы по формированию позиции играет важную роль в формировании экономической целесообразности принимаемых решений? Равным образом сложившаяся структура организации напрямую зависит от ключевых компонентов планируемого обновления! Равным образом постоянный количественный рост и сфера нашей активности представляет собой интересный эксперимент проверки ключевых компонентов планируемого обновления? С другой стороны рамки и место обучения кадров обеспечивает актуальность соответствующих условий активизации. Повседневная практика показывает, что новая модель организационной деятельности влечет за собой процесс внедрения и модернизации соответствующих условий активизации.

Повседневная практика показывает, что консультация с профессионалами из IT...