**Лабораторная работа №4**

**Дерево поиска. Частотный словарь.**

Составить программу, реализующую частотный словарь текста.

Программа должна производить следующие действия;

1. Выделять в выбранном текстовом файле отдельные слова.

2. Помещать слова в дерево поиска.

3. Выводить из дерева частотный словарь слов в алфавитном порядке.

4. Выводить частотный словарь слов, отсортированный в порядке частоты их употребления.

5. Производить поиск заданного слова.

6. Производить фильтрацию, оставляющую в словаре слова определенной длины.

**Алгоритм**

Функция formTree

1. Функция принимает значения node (указатель на узел дерева) и word

(строку для добавления в дерево).

1. Если node = NULL
   1. Создать новый узел для строки word и k = 1
2. Иначе
   1. Если word < node->word
      1. Вызов функции formTree для левого потомка node  и строки word
   2. Иначе если word > node->word
      1. Вызов функции formTree для правого потомка node  и строки word
   3. Иначе
      1. Увеличить значение k на 1.

Функция printTree

1. Функция принимает значение node (указатель на узел дерева) и файл file
2. Если node != NULL
3. Вызов функции printTree для левого потомка node и файла file
   1. Вывод на экран строки word и значения k
   2. Вызов функции printTree для правого потомка node и файла file

Функция FindMax

1. Функция принимает значение node (указатель на узел дерева) и значение max
2. Если node != NULL
   1. Если node->k > max, то max присвоить значение node->k
   2. Присвоить max результат вызова функции FindMax для левого потомка node и значения max
   3. Присвоить max результат вызова функции FindMax для правого потомка node и значения max
3. Функция возвращает значение max

Функция PrintMax

1. Функция принимает значение node (указатель на узел дерева) и указатели на значения firstmax и secondmax и файл file
2. Если node != NULL
   1. Если node ->k = firstmax то записать в файл file слово word и значение node ->k
   2. Иначе если node ->k < firstmax И node ->k > secondmax присвоить значению на которое указывает secondmax значение node ->k
   3. Вызов функции PrintMax для левого потомка node, значений firstmax и secondmax и файла file
   4. Вызов функции PrintMax для правого потомка node, значений firstmax и secondmax и файла file

Функция printTree2

1. Функция принимает значение node (указатель на узел дерева) и файл file
2. Присвоить firstmax результат вызова функции FindMax для node и 0, secondmax присвоить 0
3. Пока результат вызова функции PrintMax для node, адресов переменных firstmax и secondmax и файла file не равен 0
   1. firstmax присвоить значение secondmax, secondmax присвоить 0

Функция writeBySize

1. Функция принимает значение node (указатель на узел дерева) и значение size и файл file
2. Если node != NULL
   1. Если длинна строки node->word = size, то вывести строку node->word и значение node->k на экран
   2. Вызов функции writeBySize для левого потомка node и значения size и файла file
   3. Вызов функции writeBySize для правого потомка node и значения size и файла file

Функция FindWord

1. Функция принимает значение node (указатель на узел дерева) и строку word
2. Если node != NULL
   1. Если node->word = word, то вывести строку node->word и значение node->k на экран
   2. Вызов функции FindWord для левого потомка node и строки word
   3. Вызов функции FindWord для правого потомка node и строки word

Функция main

1. Открытие файла file с названием, введенным пользователем
2. Пока читается символ c из файла file
   1. Если символ c это не буква и не цифра, то
      1. Если строка word не пустая, то вызов функции formTree для узла root и строки word
      2. Очистить строку word
   2. Иначе добавить символ c к строке word
3. Бесконечный цикл
   1. Выбор режима работы пользователем от 1 до 5
   2. Если выбран режим 1, то
      1. Открытие файла file2 с названием, введенным пользователем
      2. Вызов функции printTree для узла root и файла file2
   3. Иначе если выбран режим 2, то
      1. Открытие файла file2 с названием, введенным пользователем
      2. Вызов функции printTree2 для узла root и файла file
   4. Иначе если выбран режим 3, то
      1. Ввод строки word пользователем
      2. Вызов функции FindWord для узла root и строки word
   5. Иначе если выбран режим 4, то
      1. Открытие файла file2 с названием, введенным пользователем
      2. Вызов функции writeBySize для узла root и файла file
   6. Иначе выйти из цикла

**Код программы**

#include <Windows.h>

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

struct Node {

string word;

int k;

Node\* left, \* right;

};

void formTree(Node\*& node, string word) {

if (node == NULL) {

node = new Node;

node->word = word;

node->k = 1;

node->left = NULL;

node->right = NULL;

}

else

if (word < node->word)

formTree(node->left, word);

else

if (word > node->word)

formTree(node->right, word);

else node->k++;

}

void printTree(Node\* node, ofstream& file) {

if (node != NULL)

{

printTree(node->left,file);

file << node->word << " - " << node->k << endl;

printTree(node->right,file);

}

}

int FindMax(Node\* node, int max) {

if (node != NULL)

{

if (node->k > max) max = node->k;

max = FindMax(node->left, max);

max = FindMax(node->right, max);

}

return max;

}

int PrintMax(Node\* node, int\* firstmax, int\* secondmax, ofstream& file) {

if (node != NULL)

{

if (node->k == \*firstmax) {

file << node->word << " - " << node->k << endl;

}

else if (node->k < \*firstmax && node->k > \*secondmax) {

\*secondmax = node->k;

}

PrintMax(node->left, firstmax, secondmax, file);

PrintMax(node->right, firstmax, secondmax, file);

}

return \*secondmax;

}

void printTree2(Node\* node, ofstream &file) {

int firstmax = FindMax(node, 0);

int secondmax = 0;

while (PrintMax(node, &firstmax, &secondmax, file) != 0) {

firstmax = secondmax;

secondmax = 0;

}

}

void writeBySize(Node\* node, int size, ofstream& file) {

if (node != NULL) {

if (node->word.size() == size)

file << node->word << " - " << node->k << endl;

writeBySize(node->left, size, file);

writeBySize(node->right, size, file);

}

}

void FindWord(Node\* node, string word) {

if (node != NULL) {

if (node->word == word) {

cout << node->word << " - " << node->k << endl;

return;

}

FindWord(node->left, word);

FindWord(node->right, word);

}

}

void main() {

SetConsoleCP(1251);

setlocale(LC\_ALL, "ru");

ifstream file;

string word, filename;

Node\* root = NULL;

char c;

cout << "Введите имя файла" << endl;

cin >> filename;

file.open(filename);

while (file.read((char\*)&c, sizeof(c))) {

if (!((c >= 'А' && c <= 'я') || (c >= 'A' && c <= 'Z') || c=='ё' || c == 'Ё' || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9'))) {

if (word != "") {

formTree(root, word);

word = "";

}

}

else {

word += c;

}

}

if (word != "") formTree(root, word);

file.close();

while (true) {

int choise;

cout << "Выберите действие" << endl;

cout << "1 - Вывести слова по алфавиту" << endl << "2 - Вывести слова по частоте" << endl << "3 - Поиск слова" << endl << "4 - Фильтрация по длине" << endl << "5 - Завершить работу" << endl;

cin >> choise;

if (choise == 1) {

cout << "Введите имя файла" << endl;

cin >> filename;

ofstream file2(filename);

printTree(root, file2);

file2.close();

}

else if (choise == 2) {

cout << "Введите имя файла" << endl;

cin >> filename;

ofstream file2(filename);

printTree2(root, file2);

file2.close();

}

else if (choise == 3) {

cout << "Введите слово" << endl;

cin >> word;

FindWord(root, word);

}

else if (choise == 4) {

cout << "Введите длину" << endl;

int len;

cout << "Введите имя файла" << endl;

cin >> filename;

ofstream file2(filename);

cin >> len;

writeBySize(root, len, file2);

file2.close();

}

else break;

}

system("pause");

}