Міністерство освіти і науки України Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського" Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 6 з дисципліни «Основи програмування-2. Методології програмування» «Структури даних»

Варіант 31 (1)

Виконав студент ІП-13 Сокур Антон Юрійович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вєчерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2022

Мета – вивчити механізми створення бінарних дерев та запису в даних в них.

Завдання:

1. Текстовий файл містить програму мовою С/С++. Надрукувати в алфавітному порядку всі ідентифікатори цієї програми, вказавши для кожного з них число входжень у текст програми. Для збереження ідентифікаторів використати структуру типу дерева, елементами якого є ідентифікатор і число його входжень у текст.

Код програми

C++

Tree.h

```
#pragma once
#include <iostream>
#include <stack>
#include <fstream>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

struct Item {
    string name;
    int number;
};

struct Node {
    Item* data;
    Node* left;
    Node* right;
};

class Tree {
private:
    Node* root;

    void addElement(Node*&, Item*);
    void printTree(Node*, int);
    void printInOrder(Node*);

public:
    Tree();
    void addElement(Item*);
    void printTree();
};
```

Tree.cpp

```
#include "Tree.h"

Tree::Tree() {
    root = NULL;
}

// public addElement
```

```
void Tree::addElement(Item *element) {
    this->addElement(root, element);
}

// private addElement
void Tree::addElement(Node *&node, Item *element) {
    if (!node) {
        node = new Node;
        node->left = NULL;
        node->right = NULL;
        node->data = element;
        return;
    }
    if (element->name == node->data->name){
        node->data->number++;
    }
    if (element->name > node->data->name){
        addElement(node->right, element);
    }
    if (element->name < node->data->name){
        addElement(node->left, element);
    }
}
// public printTree
void Tree::printTree() {
    cout << "Identifiers tree: " << endl;
    printTree(this->root, 0);
    cout << endl << "Identifiers in order:" << endl;
    printInOrder(this->root);
}

//private printTree
void Tree::printTree(Node *node, int a) {
    if (!node) {
        return;
    }
    printTree(node->right, ++a);

    for (int i = 1; i < a; i++) {
        cout << "\t";
    }
    cout << node->data->name << " : " << node->data->number << endl;

    this->printTree(node->left, a);
}

void Tree::printInOrder(Node * parentNode) {
    if (!parentNode) {
        return;
    }
    printInOrder(parentNode->left);
    cout << parentNode->data->name << " : " << parentNode->data->number <<
endl;
    printInOrder(parentNode->right);
}
```

operations.h

```
#include <fstream>
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
```

```
#include "Tree.h"

using namespace std;

void fileOutput(string);
void readFile(string);
vector<string> getKeyWords();
bool isKeyWord(string);
bool isQuote(char);
bool isStillWord(char);
```

operations.cpp

```
#include "operations.h"

void fileOutput(string name) {
    ifstream file(name);
    while (!file.eof()) {
        string str;
        getline(file, str);
        cout << str << endl;
    }
}

void readFile(string name) {
    Tree tree;
    ifstream file(name);
    while (!file.eof()) {
        string str;
        getline(file, str);
        bool isWord = false;
        bool isInsideString = false;
        string newWord = "";
        for (char i : str){
            if (isQuote(i)) {
                isInsideString = !isInsideString;
            }
            if ((isalpha(i) || i == '_') && !isWord && !isInsideString){
                isWord = true;
            }
            if (isWord && !isInsideString){
                if (isStillWord(i)){
                    newWord += i;
                }else{
                    if (!isKeyWord(newWord)){
                        Item* newItem = new Item;
                        newItem->name = newWord;
                        newItem->number = 1;
                        tree.addElement(newItem);
                    }
                    newWord = "";
                    isWord = false;
                }
            }
        }
    }
    tree.printTree();
    file.close();
}

vector<string> getKeyWords(){
    vector<string> keywords;
    ifstream file("keywords.txt");
    while (!file.eof()) {
```

```
        string word;
        getline(file, word);
        word = word.substr(0, word.length()-1);
        keywords.push_back(word);
    }
    file.close();
    return keywords;
}

bool isKeyWord(string word) {
    vector<string> keywords = getKeyWords();
    return find(keywords.begin(), keywords.end(), word) != keywords.end();
}

bool isQuote(char chr) {
    return chr == '"' || chr == char(39); // char(39) - одинарна кавичка
}

bool isStillWord(char chr) {
    return isalpha(chr) || isdigit(chr) || chr == '_';
}
```

main.cpp

```
#include "Tree.h"
#include "operations.h"


int main()
{
    fileOutput("file.txt");
    readFile("file.txt");
    return 0;
}
```

Робота програми на C++

#include <iostream>

#include <string>

void foo(){

int x = 3;

cout << x + 5;

}

class areaClass {

private:

int ter;

public:

```
    areaClass() {
        ter = 8;
    }
}

int main(){
    int ib = 2;
    ib = 2;
    foo();
    foo();
    int ic = 3;
    string bg;
    string bf;
    bf = "afasf";
    bf = "1kljhkal"
    ic = 3;
    ic = 3;
    int id = 4;
    id = 4;
    id = 4;
    id = 4;
    int ir = 5;
    for (int i = 0; i < ir; i++){
        cout << i << "+" << i << "-" << i + i;
    }
    string sc = "thirdstring";
    sc = "33";
}
```

Identifiers tree:

x : 2

ter : 2

sc : 2

main : 1

ir : 2

id : 4

ic : 3

ib : 2

i : 7

foo : 3

bg : 1

bf : 3

areaClass : 2

Identifiers in order:

areaClass : 2

bf : 3

bg : 1

foo : 3

i : 7

ib : 2

ic : 3

id : 4

ir : 2

main : 1

sc : 2

ter : 2

x : 2

Висновок: під час виконання даної лабораторної роботи я вивчив особливості створення бінарних дерев, запису та виведу їх у консоль на прикладі мові C++.