МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Алгоритмы и структуры данных»

Тема: Деревья

Студент гр. 7383 ___ Сычевский Р.А.

Преподаватель ___ Размочаева Н.В.

Санкт-Петербург

2018

Содержание

1. ЦЕЛЬ РАБОТЫ

Цель работы: Познакомиться со структурой и реализацией бинарного дерева и леса на языке программирования C++.

Формулировка задания (Вариант 6): Для заданного бинарного дерева:

а) получить лес, естественно представленный этим бинарным деревом;

б) вывести изображения бинарного дерева и леса;

в) перечислить элементы леса в горизонтальном порядке (в ширину).

2. РЕАЛИЗАЦИЯ ЗАДАЧИ

В данной работе используется главная функция (main()) и функции pr_menu() full_b_tree(), pr_b_tree(), process(), pr_forest(), pr_elem(), pr_elements(), del_b_tree(), del_forest(), work_with_console(), work_with_file()).

Сначала функция main() вызывает функцию pr_menu(), которая используется для вывода меню. Пользователь выбирает один из пунктов меню и вводит число, которое передается в switch(). В зависимости от выбранного пункта менювыбирается необходимая опция:

1 – ввод бинарного дерева с клавиатуры;

2 – считывание бинарного дерева из файла;

0 – выход из программы;

При вводе единицы, программа вызывает функцию work_with_console(), которая считывает бинарное дерево из консоли и передает его в функцию full_b_tree().

При вводе двойки, программа вызывает функцию work_with_file(), которая считывает бинарное дерево из файла и передает его в функцию full_b_tree().

Функция full_b_tree() получает на вход указатель на корень дерева или поддерева и строку с введенным бинарным деревом и строит бинарное дерево. После этого бинарное дерево выводится при помощи функции pr_b_tree(). Затем при помощи функции process() строится лес и выводится функцией pr_forest(). После этого при помощи функции pr_elements() выводятся элементы леса в горизонтальном порядке.

3. ТЕСТИРОВАНИЕ

3.1 ПРОЦЕСС ТЕСТИРОВАНИЯ

Программа собрана в операционной системе Ubuntu 18.04.1 LTS bionic компилятором g++ (Ubuntu 7.3.0-16ubuntu3) 7.3.0. В других ОС и компиляторах тестирование не проводилось.

3.2 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Тестовые случаи представлены в Приложении А.

Результаты тестирования показали, что программа работает верно, значит поставленная задача выполнена.

4. ВЫВОД

В ходе выполнения данной работы были получены навыки работы с бинарным деревом и лесом. Было построено бинарное дерево и лес на базе указателей. Бинарное дерево является рекурсивной структурой и для неё легко определяются рекурсивные методы.

ПРИЛОЖЕНИЕ А: ТЕСТОВЫЕ СЛУЧАИ

№	Ввод	Вывод
1	(a)	a * a
2	(q(r(t))(i#(o)))	q r t i # o * q r t i # o
3	(o(u(r(e(w(q))))))	o u r e w q * o u r e w q
4	(q#(i#(o#(j(o)(e)))))	q # i # o # j o e * q # i # o # j o e
5	(a(e(r(E)(i(o)))(o(p)(m))))	a e r E i o o p m * a e r E i o o p m

ПРИЛОЖЕНИЕ Б: КОД ПРОГРАММЫ

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <cstring>
using namespace std;

struct b_tree{
    char a;
    b_tree* left;
    b_tree* right;
};

struct forest{
    char a;
    int count;
    forest* branch[25];
};

void pr_menu(){
    cout << "Выберите действие:" << endl;
    cout << "1 - ввод с клавиатуры" << endl;
    cout << "2 - ввод из файла" << endl;
    cout << "0 - выход" << endl;
}

int full_b_tree(b_tree* tree, string str){
    int flag = 0;
    int count = 0;
    int current_count = 0;
    string tmp;
    tree->a = str[1];
    tmp = str.substr(2);
    str = tmp;
    if(str[0] == '('){
        tree->left = new b_tree;
        count += full_b_tree(tree->left, str);
        tmp = str.substr(count);
        str = tmp;
    } else if(str[0] == '#'){
        tree->left = new b_tree;
        tree->left->a = '#';
        tree->left->left = NULL;
        tree->left->right = NULL;
        count++;
        tmp = str.substr(1);
        str = tmp;
    } else if(str[0] == ')'){
        tree->left = NULL;
        tree->right = NULL;
    }
    if(str[0] == '('){
        flag = 1;
```

```
        tree->right = new b_tree;
        count += full_b_tree(tree->right, str);
    }
    if(str[0] == ')')
        tree->right = NULL;
    return count+3;
}

void pr_b_tree(b_tree* tree, int level, int is_right){
    if(is_right){
        for(int i = 0; i < level; i++)
            cout << "    ";
    }
    cout << "   " << tree->a;
    if(tree->left == NULL)
        cout << endl;
    else
        pr_b_tree(tree->left, level+1, 0);
    if(tree->right == NULL)
        return;
    else
        pr_b_tree(tree->right, level+1, 1);
    return;
}

void process(b_tree* tree, forest* f_tree){
    f_tree->branch[f_tree->count] = new forest;
    f_tree->branch[f_tree->count]->a = tree->a;
    f_tree->count++;
    if(tree->left != NULL)
        process(tree->left, f_tree->branch[f_tree->count-1]);
    if(tree->right != NULL)
        process(tree->right, f_tree);
}

int pr_forest(forest* f_tree, int level, int flag){
    int i = 0;
    if(flag)
        for(int k = 0; k < level; k++)
            cout << "    ";
    cout << "   " << f_tree->a;
    if(f_tree->count == 0){
        cout << endl;
    }
    else{
        for(i = 0; i < f_tree->count; i++)
            pr_forest(f_tree->branch[i], level+1, i);
    }
}

int pr_elem(forest* f_tree, int cur_level, int level){
    int flag = 0;
    if(cur_level == level){
        cout << f_tree->a;
        return 1;
    }else
```

```
        if(f_tree->count)
            for(int i = 0; i < f_tree->count; i++)
                flag += pr_elem(f_tree->branch[i], cur_level+1, level);
        else
            return 0;
    return flag;
}
int pr_elements(forest* f_tree){
    int flag = 1;
    int level = 1;
    while(flag){
        flag = 0;
        flag += pr_elem(f_tree, 0, level);
        level++;
    }
}

void del_b_tree(b_tree* tree){
    if(tree->left != NULL){
        del_b_tree(tree->left);
        delete(tree->left);
    }
    if(tree->right != NULL){
        del_b_tree(tree->right);
        delete(tree->right);
    }
}

void del_forest(forest* f_tree){
    if(f_tree->count){
        for(int i = 0; i < f_tree->count; i++){
            del_forest(f_tree->branch[i]);
            delete(f_tree->branch[i]);
        }
    }
}

int work_with_console(){
    cout << "Введите выражение" << endl;
    string str;
    getline(cin, str);
    getline(cin, str);
    int i = 0;
    while((i=str.find(' '))!=std::string::npos)
        str.erase(i, 1);
    b_tree* tree = new b_tree;
    int count = full_b_tree(tree, str);
    cout << "Бинарное дерево:" << endl;
    pr_b_tree(tree, 0, 0);
    forest* f_tree = new forest;
    f_tree->a = '*';
    process(tree, f_tree);
    cout << "Лес:" << endl;
    pr_forest(f_tree, 0, 0);
    cout << "Перечисление элементов в горизонтальном порядке:" << endl;
    pr_elements(f_tree);
```

```
    del_b_tree(tree);
    delete(tree);
    del_forest(f_tree);
    delete(f_tree);
    return 0;
}

int work_with_file(){
    cout << "Введите имя файла, в котором записано выражение и значения
переменных" << endl;
    string file_name;
    cin >> file_name;
    ifstream f;
    f.open(file_name.c_str());
    if (!f){
        cout << "Файл не открыт!" << endl;
        return 0;
    }
    string str;
    getline(f, str);
    cout << str << endl;
    int i = 0;
    while((i=str.find(' '))!=std::string::npos)
        str.erase(i, 1);
    b_tree* tree = new b_tree;
    full_b_tree(tree, str);
    cout << "Бинарное дерево:" << endl;
    pr_b_tree(tree, 0, 0);
    forest* f_tree = new forest;
    f_tree->a = '#';
    process(tree, f_tree);
    cout << "Лес:" << endl;
    pr_forest(f_tree, 0, 0);
    cout << "Перечисление элементов в горизонтальном порядке:" << endl;
    pr_elements(f_tree);
    del_b_tree(tree);
    delete(tree);
    del_forest(f_tree);
    delete(f_tree);
    return 0;
}

int main(){
    pr_menu();
    int way = 0;
    cin >> way;
    while(way){
        switch (way){
            case 1:
                work_with_console();
                cout << endl;
                pr_menu();
                cin >> way;
                break;
            case 2:
                work_with_file();
```

```
                cout << endl;
                pr_menu();
                cin >> way;
                break;
            default:
                cout << "Неверно введены данные!" << endl;
                pr_menu();
                cin >> way;
        }
    }
    return 0;
}
```