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

ОТЧЕТ

по лабораторная работа №3

по дисциплине «Алгоритмы и структуры данных»

ТЕМА: БИНАРНЫЕ ДЕРЕВЬЯ

Студент гр. 9303 _____ Павлов Д.Р.

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург

2020

Цель работы.

Изучить нелинейную структуру данных бинарное дерево и способы его хранения. Научиться работать с данными, заданными в виде бинарного дерева.

Задание.

Вариант 2д.

Для заданного бинарного дерева *b* типа *BT* с произвольным типом элементов:

- определить максимальную глубину дерева *b*, т. е. число ветвей в самом длинном из путей от корня дерева до листьев;

- вычислить длину внутреннего пути дерева *b*, т. е. сумму по всем узлам длин путей от корня до узла.

Основные теоретические положения.

Дерево – конечное множество T, состоящее из одного или более узлов, таких, что

а) имеется один специально обозначенный узел, называемый корнем данного дерева;

б) остальные узлы (исключая корень) содержатся в m >= 0 попарно не пересекающихся множествах T1, T2, ..., Tm, каждое из которых, в свою очередь, является деревом. Деревья T1, T2, ..., Tm называются поддеревьями данного дерева.

Каждый узел дерева является корнем некоторого поддерева. В том случае, когда множество поддеревьев такого корня пусто, этот узел называется концевым узлом, или листом. Уровень узла определяется рекурсивно следующим образом:

1) корень имеет уровень 1;

2) другие узлы имеют уровень, на единицу больший их уровня в содержащем их поддереве этого корня.

Говорят, что каждый корень является отцом корней своих поддеревьев и что последние являются сыновьями своего отца и братьями между собой.

Говорят также, что узел n – предок узла m (а узел m – потомок узла n), если n – либо отец m, либо отец некоторого предка m.

Если в определении дерева существен порядок перечисления поддеревьев T1, T2, ..., Tm , то дерево называют упорядоченным и говорят о «первом» (T1), «втором» (T2) и т. д. поддеревьях данного корня.

Лес – это множество (обычно упорядоченное), состоящее из некоторого (быть может, равного нулю) числа непересекающихся деревьев.

Используя понятие леса, пункт б в определении дерева можно было бы сформулировать так: узлы дерева, за исключением корня, образуют лес.

Бинарное дерево — конечное множество узлов, которое либо пусто, либо состоит из корня и двух непересекающихся бинарных деревьев, называемых правым поддеревом и левым поддеревом.

Выполнение работы.

Для выполнения работы был реализован класс Node, который имел поля data,

Node* left, Node* right. Также были реализованы методы:

getData – получает данные.

setData – устанавливает данные.

makeRight – создает правое поддерево.

makeLeft – создает левое поддерево.

getRight – получает данные правого поддерева.

getLeft – получает данные левого поддерева.

isRightFree – проверяет свободно ли правое поддерево.

isLeftFree – проверяет свободно ли левое поддерево.

Также были написана функция createForest, которая получает указатель на узел, строку и счетчик узлов n и создает лес.

Далее были реализованы функции getDeep, getLen, которые возвращают максимальную глубину дерева и сумму всех путей.

Вывод

В ходе выполнения работы была изучена структура бинарного дерева, и реализовано две функции, которые считают глубину и сумму путей дерева.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

```
#include <iostream>
#include <fstream>
class Node{
private:
    char data;
    Node* right;
    Node* left;
public:
    Node() {
        data = '#';
        right = nullptr;
        left = nullptr;
    }
    char getData(){
        return data;
    }
    void setData(char x){
        data = x;
    }
    void makeRight(){
        right = new Node;
    }
    void makeLeft(){
        left  = new Node;
    }
    Node* getRight(){
        return right;
    }
    Node* getLeft(){
        return left;
    }
    bool isLeftFree(){
        return left == nullptr;
    }
    bool isRightFree(){
        return right == nullptr;
    }
};

void createForest(Node* a, std::string info, int& n) {

    if(n >= info.length() || info[n]== '\0'){
        n++;
        return;
    }

    if(info[n]=='/')
    {
        a = new Node;
        n++;
    }else {

        a->setData(info[n]);

        n++;
        a->makeLeft();
        createForest(a->getLeft(), info, n );

        a->makeRight();
```

```
        createForest(a->getRight(), info, n );
    }

}

void getLen(Node* head, int s, int& result){
    if(head->getData() == '#'){
        return;
    }
    result += s;
    if(!head->isLeftFree()) {
        getLen(head->getLeft(), s+1, result);
    }
    if(!head->isRightFree()){
        getLen(head->getRight(), s+1, result);
    }
}

int getDeep(Node* head, int s) {
    int x = 1;
    int r;

    if(head->getData() == '#') {
        return x;
    }

    if(!head->isLeftFree()){
        x = getDeep(head->getLeft(), s+1);
    }


    if(!head->isRightFree()){

       r = getDeep(head->getRight(), s+1);
       if (x < r)
           x = r;
    }
    if (s > x) return s;
    else return x;
}


int main(){
    int n = 0;
    int result = 0;
    std::ifstream fin("./IO/input.txt");
    std::ofstream out("./IO/output.txt");
    Node* head = new Node;
    std::string str;
    fin>>std::noskipws;
    if(!fin){
        std::cout<<"Can't open this file!";
    }
    fin>>str;
    createForest(head, str, n);
    result = getDeep(head, 1);
    out << "Result deep = " << result << '\n';
    int a = 0;
    getLen(head, 0, a);
    out << "Len = "<<a<<'\n';
    return 0;
}
```