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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: Программирование алгоритмов с бинарными деревьями

Студент гр. 9381 _____ Андрух И.А.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2020

Цель работы.

Познакомиться со структурой данных бинарного дерева, реализовать его рекурсивную обработку на языке программирования C++.

Задание.
Вариант 3д.

В заданиях 1 - 5, в зависимости от варианта, предлагается реализовать рекурсивные или нерекурсивные процедуры (функции); в последнем случае следует использовать стек и операции над ним.

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

- напечатать элементы из всех листьев дерева b;
- подсчитать число узлов на заданном уровне n дерева b (корень считать узлом 1-го уровня).

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 T 1 , T 2 , ..., T m , каждое из которых, в свою очередь, является деревом. Деревья T 1 , T 2 , ..., T m называются поддеревьями данного дерева.

При программировании и разработке вычислительных алгоритмов удобно использовать именно такое рекурсивное определение, поскольку рекурсивность является естественной характеристикой этой структуры данных.

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.

Наиболее важным типом деревьев являются бинарные деревья. Удобно дать следующее формальное определение. Бинарное дерево конечное множество узлов, которое либо пусто, либо состоит из корня и двух непересекающихся бинарных деревьев, называемых правым поддеревом и левым поддеревом.

Функции и структуры данных.

Бинарное дерево реализовано при помощи элементов класса Elem.

Приватные поля класса:

- Elem* left - указатель на левое поддерево;
- Elem* right - указатель на правое поддерево;
- T data - значение элемента типа T;

Также для работы с элементом класса есть следующие функции:

- Elem():let(nullptr, right(nullptr), data('\0') - конструктор класса;
- Elem* getLeft() - возвращает значение левого поддерева;
- Elem* getGight() - возвращает значение правого поддерева;
- void setLeft(Elem* l) - устанавливает левое поддерево;
- void setRight(Elem* r) - устанавливает правое поддерево;
- T getData() - возвращает значение элемента;
- T setData(T t) - устанавливает значение элемента;

Для обработки дерева были написаны следующие рекурсивные функции:

- void recTreePrint(Tree node) — в качестве параметра принимает указатель на элемент дерева, выводит бинарное дерево;

- void recLeafPrint(Tree root) — в качестве параметра принимает указатель на элемент дерева, выводит листья бинарного дерева;
- int getMaxDepth(Tree root, int depth) — в качестве параметров принимает указатель на элемент дерева и уровень дерева, который сейчас обрабатывается, возвращает количество уровней дерева;
- int countelem(Tree root, int level, int i) — в качестве параметров принимает указатель на элемент дерева, введенный номер уровня и уровень дерева, который сейчас обрабатывается, возвращает количество узлов дерева, расположенных на уровне level;
- Tree readBT(string input) — в качестве параметра принимает введенную строку, создает бинарное дерево;

Описание алгоритма.

Программа считывает строку данных с консоли или из файла, после чего создает бинарное дерево. Дерево реализуется на базе указателей, в полях каждого узла должен храниться указатель на левый и правый элементы узла (если узел пустой — хранится указатель nullptr).

Считанная строка обрабатывается посимвольно. Сначала заполняется левое поддерево узла. Если встречается символ «/», начинает заполняться правое поддерево. Если оба поддерева заполнены, то происходит возврат на узел выше и рекурсивное заполнение оставшихся узлов дерева. После считывания строка проверяется. Она считается корректной, если пустых элементов на 1 больше, чем непустых.

Если строка введена корректно, то в консоль выводится считанное дерево для проверки (функция recTreePrint(root)), листья дерева (функция recLeafPrint(root)), количество уровней дерева (функция getMaxDepth(root,0)). После этого программа запрашивает у пользователя номер уровня, проверяет, что введенное число не превышает максимальное количество уровней в дереве,

выводит все узлы заданного уровня и их количество (функция countelem(root,cur,1).

Тестирование.

№	Входные данные	Вывод
1	abc//d//bc//ef/// 3	Введенное дерево: abc//d//bc//ef/// Листья дерева: c d c f Максимальная глубина дерева 4 Подсчёт количества узлов заданного уровня: Количество узлов на данном уровне: c d c e 4
2	ab//c// 2	Введенное дерево: ab//c// Листья дерева: b c Максимальная глубина дерева 2 Подсчёт количества узлов заданного уровня: Количество узлов на данном уровне: b c 2
3	////	Введенные данные некорректны
4	abcd//e///fg/// 3	Введенное дерево: abcd//e///fg/// Листья дерева: d e g

		Максимальная глубина дерева 4 Подсчёт количества узлов заданного уровня: Количество узлов на данном уровне: c g 2
5	abcdefgh//////// 8	Введенное дерево: abcdefgh//////// Листья дерева: h Максимальная глубина дерева 8 Подсчёт количества узлов заданного уровня: e 1
6	abdh//k//el//m//cfn//o//gp//q// 8	Введенное дерево: abdh//k//el//m//cfn//o//gp//q// Листья дерева: h k l m n o p q Максимальная глубина дерева 4 Подсчёт количества узлов заданного уровня: Количество узлов на данном уровне: d e f g 4
7	ab/c//	Введенные данные некорректны
8	a//	Введенное дерево: a//

	1	Листья дерева: a Максимальная глубина дерева 1 Подсчёт количества узлов заданного уровня: Количество узлов на данном уровне: a 1
9	fff//// 1	Введенное дерево: fff//// Листья дерева: f Максимальная глубина дерева 3 Подсчёт количества узлов заданного уровня: Количество узлов на данном уровне: f 1

Демонстрация работы программы.

```
Выберите:
1. Ввод строки с консоли
2. Чтение строки из файла
1
Введите дерево в виде строки, начиная с корня, пустой элемент обозначать: '/'
abdh//k//el//m//cfn//o//gp//q//
Введенное дерево: abdh//k//el//m//cfn//o//gp//q//
Листья дерева:  h k l m n o p q
Максимальная глубина дерева
4
Подсчёт количества узлов заданного уровня:
Введите число не более 4
2

Количество узлов на данном уровне:
b
c
2
```

Выводы.

Были изучены метод реализации бинарного дерева и принцип работы с ним на языке C++.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <fstream>
#include <iostream>
using namespace std;
static int readIndex;     //переменная, отвечающая за индекс элемента в строке
typedef char T;
class Elem {
    Elem* left;
    Elem* right;
    T data;
public:
    Elem():left(nullptr), right(nullptr), data('\0') {};
    Elem* getLeft() {
        return left;        //возвращает значение левого поддерева
    }
    void setLeft(Elem* l) {  //устанавливает левое поддерево
        left = l;
    }
    void setRight(Elem* r) {     //устанавливает правое поддерево
        right = r;
    }
    Elem* getRight() {           //возвращает значение правого поддерева
        return right;
    }
    T getData() const {         //возвращает значение элемента
        return data;
    }
    void setData(T t) {        //устанавливает значение элемента
        data = t;
    }
};
typedef Elem* Tree;
void recTreePrint(Tree node) {
    if (!node) {
        cout << '/';
        return;
    }
    cout << node->getData();
    recTreePrint(node->getLeft());      //печать левого
    recTreePrint(node->getRight());     //печать правого
}
void recLeafPrint(Tree root){
    if(!root)
        return;
    if((root->getLeft()== nullptr)&&(root->getRight()== nullptr))
        cout << " " << root->getData();
    recLeafPrint(root->getLeft());
    recLeafPrint(root->getRight());
}
int getMaxDepth(Tree root, int depth) {
    if (!root)
        return depth;
    return max(getMaxDepth(root->getLeft(), depth + 1),
getMaxDepth(root->getRight(), depth + 1));
}
int countelem(Tree root, int level, int i)
{
    static int cnt = 0;
    if(!root)
        return 0;
```

```
    else if(i == level) {
        cout<<root->getData()<<"\n";
        return 1;
    }
    else {
        cnt += countelem(root->getLeft(), level,i+1);
        cnt += countelem(root->getRight(), level,i+1);
    }
    return (i > 1) ? 0 : cnt;
}
Tree readBT(string input){
    T sign = input[readIndex];
    readIndex++;
    if (sign == '/'){        //если элемент пустой
        return nullptr;
    }
    else{
        Tree buf = new Elem();   //если нет, создаем листок
        buf->setData(sign);
        buf->setLeft(readBT(input));
        buf->setRight(readBT(input));
        return buf;
    }
}
int countSymbols(string str, char c){     //подсчет конкретного символа в строке
    int count = 0;
    for (char symbol : str){
        if (symbol == c)
            count++;
    }
    return count;
}
int main() {
    string input;
    ifstream file;
    string name;
    cout << "Выберите:\n1. Ввод строки с консоли\n2. Чтение строки из файла\n";
    int choice = 0;
    cin >> choice;
    switch(choice){
        case 1:
            cout << "Введите дерево в виде строки, начиная с корня, пустой
элемент обозначать: '/' \n";
            cin >> input;
            break;
        case 2:
            cout << "Введите полный путь до файла: \n";
            cin >> name;
            file.open(name);
            if (!file.is_open()){
                cout << "Файл не может быть открыт!\n";
                exit(1);
            }
            getline(file, input);    //считывание из файла строки с данными
            file.close();  //закрытие файла
            break;
        default:
            cout << "Вы должны ввести 1 или 2";
            return 0;
    }
    if ((2*countSymbols(input, '/') - input.length() != 1) || (input[0] ==
'/')){ //проверка на то, что '/' на одну больше, чем остальных символов
        cout << "Введенные данные некорректны";
        return 0;
```

```
    }
    Tree root = readBT(input);
    cout << "Введенное дерево: ";
    recTreePrint(root);
    cout << "\n";
    cout << "Листья дерева: ";
    recLeafPrint(root);
    cout << "\nМаксимальная глубина дерева\n";
    cout << getMaxDepth(root,0);
    cout<<"\nПодсчёт количества узлов заданного уровня:";
    cout << "\nВведите число не более " << getMaxDepth(root,0) << "\n";
    int cur;
    cin >> cur;
    if(cur>getMaxDepth(root,0))
        cout << "\nЧисло превышает количество узлов в дереве\n";
    else {
        cout << "\nКоличество узлов на данном уровне:\n" <<
countelem(root,cur,1);
    }
    return 0;
}
```