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

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Алгоритмы и структуры данных»
Тема: Бинарные деревья

Студентка гр. 9303 _____ Булыно Д. А.

Преподаватель _____ Филатов А. Ю.

Санкт-Петербург
2020

Цель работы.

Формирование практических навыков работы с бинарными деревьями на языке программирования C++ путём решения поставленной задачи.

Основные теоретические положения.

Дерево – структура данных, представляющая собой древовидную структуру в виде набора связанных узлов.

Бинарное дерево — это конечное множество элементов, которое либо пусто, либо содержит элемент (корень), связанный с двумя различными бинарными деревьями, называемыми левым и правым поддеревьями. Каждый элемент бинарного дерева называется узлом. Связи между узлами дерева называются его ветвями.

Бинарное дерево применяется в тех случаях, когда в каждой точке вычислительного процесса должно быть принято одно из двух возможных решений.

Бинарное дерево является рекурсивной структурой, поскольку каждое его поддерево само является бинарным деревом и, следовательно, каждый его узел в свою очередь является корнем дерева.

Задание.

Вариант 9д

Рассматриваются бинарные деревья с элементами типа Elem (в качестве Elem использовать char). Заданы перечисления узлов некоторого дерева b в порядке КЛП и ЛКП. Требуется:

- восстановить дерево b и вывести его изображение;

- перечислить узлы дерева b в порядке ЛПК.

Выполнение работы.

Бинарное дерево было реализовано через динамическую память, для этого был создан шаблон класса `Node` при помощи `template`, где

`Node* left` – поле, которое хранит ссылку на левый объект класса `Node`;

`Node* right` – поле, которое хранит ссылку на правый объект класса `Node`;

`T value` – поле, которое хранит значение узла дерева.

Методы класса `Node`:

`T getValue()` – метод, который возвращает значение текущего элемента.

`void setValue(T x)` – метод, который принимает значение текущего элемента.

`Node*& Left()` – метод, который возвращает ссылку на указатель на левый элемент.

`Node*& Right()` – метод, который возвращает ссылку на указатель на правый элемент.

С помощью функции `void LeftRightRoot(Node<T>* elem)` происходит рекурсивное перечисление узлов исходного дерева в порядке ЛПК.

С помощью функции `void Print(Node<T>* elem, int count)` рекурсивно выводится изображение дерева на консоль, с поворотом в 90 градусов против часовой стрелки.

С помощью функций `string toLeft(string &str, int to)` и `string toRight(string &str, int from)` программа получает строки из значений узлов левого и правого поддеревьев соответственно.

С помощью функций `void Creater_LTR(Node<T>* elem, string str)` и `void Creater_TLR(Node<T>* elem, string str, int& n)` рекурсивно восстанавливается дерево с помощью перечисления узлов в порядке ЛКП и КЛП соответственно.

Исходный код программы см. в приложении А.

Результаты работы программы см. в приложении Б.

Выводы.

В ходе проведения лабораторной работы были получены навыки работы с бинарными деревьями. В результате проведения лабораторной работы была написана программа, которая из заданных перечислений узлов некоторого дерева b в порядке КЛП и ЛКП восстанавливает дерево, выводит его изображение и перечисляет узлы дерева b в порядке ЛПК.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <string>

using namespace std;

template <typename T>
class Node{
private:
    Node* left;
    Node* right;
    T value;
public:
    Node(){
        right = left = nullptr;
    }

    T getValue(){
        return value;
    }

    void setValue(T x){
        value = x;
    }

    Node*& Left(){
        return left;
    }

    Node*& Right(){
        return right;
    }
    ~Node(){
        if(left) delete left;
        if(right) delete right;
    }
};

template <typename T>
void LeftRightRoot(Node<T>* elem){
    if(elem == nullptr) return;
```

```
    LeftRightRoot(elem->Left());
    LeftRightRoot(elem->Right());

    cout<< elem->getValue()<< "";
}

template <typename T>
void Print(Node<T>* elem, int count){
    if(elem == nullptr) return;

    Print(elem->Right(), count+1);

    for(int i=0; i < count; i++){
        cout<<"    ";
    }
    cout<< elem->getValue()<< "\n";

    Print(elem->Left(), count+1);
}

string toLeft(string &str, int to){
    string out = "";
    for (size_t i = 0; i < to; i++){
        out += str[i];
    }
    return out;
}

string toRight(string &str, int from){
    string out = "";
    for (size_t i = from+1; i < str.length(); i++){
        out += str[i];
    }
    return out;
}


template <typename T>
void Creater_LTR(Node<T>* elem, string str){

    if (str == "") return;
    int rootPos = (str.length()/2);

    if (str == "#"){
```

```
        elem->setValue(str[rootPos]);
        return;
    }
    else{
        elem->setValue(str[rootPos]);

        elem->Left() = new Node<T>();
        Creater_LTR(elem->Left(), toLeft(str, rootPos));

        elem->Right() = new Node<T>();
        Creater_LTR(elem->Right(), toRight(str, rootPos));
    }
}

template <typename T>
void Creater_TLR(Node<T>* elem, string str, int& n){
    if (str[n] == '#' || n >= str.length()){
        elem->setValue(str[n]);

        n++;
        return;
    }
    else{
        elem->setValue(str[n]);
        n++;

        elem->Left() = new Node<T>();
        Creater_TLR(elem->Left(), str, n);

        elem->Right() = new Node<T>();
        Creater_TLR(elem->Right(), str, n);
    }
}

int main(){
     cout << "Выберите порядок перечисления узлов (TLR или LTR): ";
     string type = "";
     cin >> type;
     if (type == "TLR"){
          cout << "Введите перечисление узлов, обозначив за # листья: ";
     string str = "";
     cin >> str;
     int count = 0;
```

```
        Node<char> * head = new Node<char>();
        Creater_TLR(head, str, count);
        cout << endl;
        cout << "Узлы дерева b в порядке КЛП:\n" << str <<"\n\n";
        cout << "Изображение дерева b:\n";
        Print(head, 0);
        cout << endl;
        cout << "Узлы дерева b в порядке ЛПК:\n";
        LeftRightRoot(head);
        delete head;
        cout << endl;
    }else if (type == "LTR"){
        cout << "Введите перечисление узлов, обозначив за # листья: ";
        string str = "";
        cin >> str;
        int count = 0;
        Node<char> * head = new Node<char>();
        Creater_LTR(head, str);
        cout << endl;
        cout << "Узлы дерева b в порядке ЛКП:\n" << str <<"\n\n";
        cout << "Изображение дерева b:\n";
        Print(head, 0);
        cout << endl;
        cout << "Узлы дерева b в порядке ЛПК:\n";
        LeftRightRoot(head);
        delete head;
        cout << endl;
    }
    else{
        cout << "Порядок не определён" << endl;
        return 0;
    }
    return 0;
}
```

ПРИЛОЖЕНИЕ Б

РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

Таблица 1. Результаты работы программы

№	Входные данные	Результат
1.	TLR abd##g##ce##fi##jk###	Изображение дерева b: # j # k # f # i # c # e # a # g # b # d # Узлы дерева b в порядке ЛПК: ##d##gb##e##i##k#jfca

Продолжение таблицы 1.

2.	LTR #f#d#b#a#c#e###	Изображение дерева b: # # # e # c # a # b # d # f # Узлы дерева b в порядке ЛПК: ##f##bd##c###ea
3.	LTP	Порядок не определён