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: Деревья

Студент гр. 9384 _____ Гурин С.Н.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить способ реализации бинарного дерева и способы его обхода на языке C++

Задание.

ВАРИАНТ 3.

Для заданного бинарного дерева b типа BT c 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).

Анализ задачи.

Задача заключается в том, что нужно реализовать класс BinaryTree, методы которого служат для создания, заполнения и обработки бинарного дерева.

Выполнение работы.

При запуске программы пользователю необходимо ввести бинарное дерево в скобочном представлении в файл input.txt. Затем выводится введнная строка

Следует учесть, что введенные (/) и ( ) – являются обозначением пустого узла.

Далее вызывается метод ConsBT(), который заполняет бинарное дерево из строки, введенной пользователем в файл. Затем происходит проверка является ли полученная строка пустой, если является, то программа продолжает свою работу, если же нет, то программа прерывается и прекращает свою работу. В свою очередь метод ConsBT() учитывает правильность введенной строки, и так же при обнаруженной ошибки завершает выполнение программы.

Затем с помощью метода PrintBT() пользователю выводятся узлы введенного бинарного дерева и затем с помощью метода PrintL выводятся листься данного дерева.

После вывода листьев пользователю предлагают ввести уровень, с которого будет посчитано количество узлов, находящиеся на данном уровне. Вывод этого количества выполняется благодаря методу PrintNumOfNodes(), который в свою очередь использует метод NumOfNodes(), который считает количество этих узлов.

Тестирование.

№ теста	ввод		вывод
	console	input.txt	
1		(hhhhhh)	String: (hhhhhh) Input Error
2	4	(a(f(l(t)(r)))(e(/)(c)))	String: (a(f(l(t)(r)))(e(/)(c))) Nodes: afltrec Leaves: trc Input level: 4 2
3		(a(v)c)	String: (a(v)c) Input Error
4	2	(a(v))	String: (a(v)) Nodes: av Leaves: v Input level: 2 1
5		(a(v)	String: (a(v) Input Error
6	2	(a(v)(k))	String: (a(v)(k)) Nodes: avk Leaves: vk Input level: 2 2
7	4	(a(e(l(/)(r))(/))(k(f(s)( ))))	String: (a(e(l(/)(r))(/))(k(f(s)( )))) Nodes: aelrkfs Leaves: rs Input level: 4 2
8		(a(e(l(/)(r)))(/))(k(f(s)( ))))	String: (a(e(l(/)(r)))(/))(k(f(s)( )))) Input Error

Выводы.

При выполнении данной лабораторной работы был изучен алгоритм создания бинарных деревьев и работа с ним.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include "BinaryTree.h"

int main() {
    BinaryTree<char> Tree;
    int n, count = 0, amount = 0;
    std::string tmp;
    std::ifstream file;
    file.open("input.txt");
    if (!file.is_open()) {
        std::cout << "Error opening file (input.txt)" << std::endl;
        exit(1);
    }
    getline(file, tmp);
    file.close();
    std::cout << "String: " << tmp << "\n";
    Tree.ConsBT(tmp, count);
    if (!tmp.empty()) {
        std::cout << "Input Error";
        exit(2);
    }
    std::cout << "Nodes: ";
    Tree.PrintBT();
    std::cout << "\nLeaves: ";
    Tree.PrintL();
    std::cout << "\nInput level: ";
    std::cin >> n;
    Tree.PrintNumOfNodes(n);
    return 0;
}
```

Название файла: BinaryTree.h

```
#pragma once
int amount = 0;
template <typename T>
class BinaryTree {
private:
     T root;
     BinaryTree* left;
     BinaryTree* right;
     int level;

     void NumOfNodes(int n) {
```

```
            if (!this->left->isNull()) {
                  this->left->NumOfNodes(n);
            }
            if (!this->right->isNull()) {
                  this->right->NumOfNodes(n);
            }
            if (this->level == n) {
                  amount++;
            }
      }

      bool isNull() {
            return this == nullptr;
      }

public:
      BinaryTree()
      {
            this->left = nullptr;
            this->right = nullptr;
            this->level = 1;
      }

      void PrintBT() {
            if (!this->isNull()) {
                  std::cout << this->root;
                  this->left->PrintBT();
                  this->right->PrintBT();
            }
      }

      void ConsBT(std::string& tmp, int count) {
            if (!tmp.empty()) {
                  if (tmp.front() == '(' && (tmp[1] == '/' || tmp[1] == '
')) {
                        if (tmp[2] != ')') {
                              std::cout << "Input Error";
                              exit(-2);
```

```
                }
                this->level = NULL;
                tmp.erase(0, 3);
                return;
            }
            else if (tmp.front() == '(' && tmp[1]) {
                this->root = tmp[1];
                count += this->level;
                this->level = count;
                tmp.erase(0, 2);
                if (tmp.front() == '(') {
                    this->left = new BinaryTree;
                    this->left->ConsBT(tmp,count);
                }
                if (!tmp.empty() && tmp.front() == '(') {
                    this->right = new BinaryTree;
                    this->right->ConsBT(tmp,count);
                }
                if (!tmp.empty()) {
                    if (tmp.front() == ')'){
                        tmp.erase(0, 1);
                    }
                }
                else {
                    std::cout << "Input Error";
                    exit(-1);
                }
            }
            else {
                std::cout << "Input Error";
                exit(1);
            }
        }
    }

    void PrintL() {
        if (this->left->isNull() && this->right->isNull()) {
            std::cout << this->root;
```

```
            }
            else {
                  if (!this->left->isNull()) {
                        this->left->PrintL();
                  }
                  if (!this->right->isNull()) {
                        this->right->PrintL();
                  }
            }
      }

      void PrintNumOfNodes(int n) {
            NumOfNodes(n);
            std::cout << "Nodes on " << n << " level: " << amount;
      }

};
```

Название файла: Makefile

```
all: goal
      ./start

goal: main.o
      g++ main.o -o start

main.o: main.cpp BinaryTree.h
      g++ -c main.cpp

clean:
      rm -f *.o
```