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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Алгоритмы и структуры данных»

Тема: Деревья

Студент гр. 7383 _____ Власов Р.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург
2018

Содержание

Цель работы

Цель работы: познакомиться с деревьями, создать реализацию бинарного дерева и функцию подсчета его глубины на языке программирования C++.

Формулировка задачи: Вариант 2-а. Для заданного бинарного дерева *b* типа *BT* с произвольным типом элементов определить максимальную глубину дерева *b*, то есть число ветвей в самом длинном из путей от корня дерева до листьев.

Реализация задачи

Для реализации дерева было принято создать следующий класс.

```
template <class T>
class BT{
private:
   BT* left;
   BT* right;
   T value;
public:
   BT(stringstream& s);
   int height();
   ~BT();
};
```

Метод void height() рекурсивно вызывает себя для всех потомков и считает максимальную глубину дерева. Конструктор класса инициализирует дерево строкой.

В программе реализован ввод данных из файла или вручную.

Исходный код программы представлен в приложении Б.

Тестирование

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.

2. Результаты тестирования

По результатам тестирования ошибок в работе программы выявлено не было. Тестовые случаи представлены в приложении А.

Вывод

В ходе выполнения данной работы были изучены деревья. Был создан класс бинарного дерева и написана программа, считающая максимальную глубину дерева.

ПРИЛОЖЕНИЕ А. ТЕСТОВЫЕ СЛУЧАИ

Бинарное дерево	Глубина
(1(2(3(4(5)))))	4
(1#(2#(3#(4#(5)))))	4
(1(2#(3(4))))	3
(1(2(3)(4))(5))	2
(1)	0

ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <cctype>
#include <string>

using namespace std;

template <class T>
class BT{
private:
   BT* left;
   BT* right;
   T value;

public:
   BT(stringstream& s);
   int height();
   ~BT();
};

template <class T>
BT<T>::BT(stringstream& s)
{
   left = NULL;
   right = NULL;
   char ch;
   if (s.peek() == '('
      s >> ch; // remove '('
   s >> this->value;
   switch(s.peek())
   {
      case '(':
         left = new BT(s);
         break;
      case '#':
         s >> ch; // remove '#'
         break;
   }
   if (s.peek() == '('
      {
         right = new BT(s);
         s >> ch; // remove ')'
```

```
        }
    else if (s.peek() == '#')
        s >> ch; // remove '#'
    if (s.peek() == ')')
        s >> ch; // remove ')'
}

template <class T>
int BT<T>::height()
{
    if (!left && !right)
        return 0;
    int left_h = 0, right_h = 0;
    if (left)
        left_h = left->height();
    if (right)
        right_h = right->height();
    return (left_h > right_h ? left_h : right_h) + 1;
}

template <class T>
BT<T>::~BT()
{
    if (left)
        delete left;
    if (right)
        delete right;
}

void run(string str)
{
    stringstream s;
    string str1;
    if (str.size() == 0)
    {
        cout << "The string is empty!" << endl;
        return;
    }
    for (int i = 0; i < str.size(); i++)
    {
        if (str[i] == '(')
        {
            str1.push_back(str[i]);
        }
```

```
        else if (str[i] == ')')
        {
            str1.push_back(str[i]);
        }
        else if (isdigit(str[i]))
        {
            str1.push_back(str[i]);
        }
        else if (str[i] == '#')
        {
            str1.push_back(str[i]);
        }
        else
        {
            cout << "There are some unexpected characters in the string: " << str << endl;
            return;
        }
    }
    cout << str1;
    s << str;
    BT<int> el(s);
    cout << " maximum path length is " << el.height() << endl;
}


int main()
{
    int n, c;
    string inp;
    int *el;
    string str, str1;
    while(true)
    {
        cout << "Press 1 to get input from a file\n" <<
                "Press 2 to enter binary tree by yourself\n" <<
                "Press 3 to exit." << endl;
        cin >> inp;
        if (!isdigit(inp[0]))
            continue;
        c = stoi(inp);
        inp.clear();
        switch (c)
        {
            case 1:
                break;
```

```
            case 2:
                getline(cin, str); // remove '\n'
                getline(cin, str);
                run(str);
                break;
            case 3:
                return 0;
            default:
                cout << "Something went wrong. try again!" << endl;
                continue;
        }
        if(c == 1)
        {
            cout << "Enter file name: ";
            cin >> str;
            ifstream f;
            f.open(str);
            if (!f)
            {
                cout << "Unable to open the file!" << endl;
                continue;
            }
            while(!f.eof())
            {
                getline(f, str);
                if (str.size())
                    run(str);
            }
            f.close();
        }
    }
}
```