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 вывода его элементов по уровням на языке программирования C++.

Формулировка задачи: Вариант 4. Задано бинарное дерево *b* типа *BT* с произвольным типом элементов. Используя очередь и операции над ней, напечатать все элементы дерева *b* по уровням: сначала – из корня дерева, затем (слева направо) – из узлов, сыновних по отношению к корню, затем (также слева направо) – из узлов, сыновних по отношению к этим узлам, и т. д.

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);
   void print (Queue<T> &q);
   ~BT();
};
```

Метод void print (Queue<T> &q) добавляет в очередь сыновние элементы и выводит значение текущего элемента. Конструктор класса инициализирует дерево строкой.

Очередь была основана на основе списка. Структура класса очереди и хранящихся в ней элементов приведена ниже:

```
template <class T>
class Queue{
private:
   El<T>* out;
   El<T>* in;
public:
   Queue();
   bool isEmpty();
   void push(BT<T>* el);
   BT<T>* pop();
};
template <class T>
struct El{
   BT<T>* elem;
   El* next;
   El(BT<T>* elem);
};
```

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, выводящая значения всех элементов дерева по уровням от корня к листьям.

ПРИЛОЖЕНИЕ А. ТЕСТОВЫЕ СЛУЧАИ

Бинарное дерево	Вывод
(1(2(3(4(5)))))	12345
(1#(2#(3#(4#(5)))))	12345
(1(2#(3(4))))	1234
(1(2(3)(4))(5))	12534
(1)	1
(1(2(4)(5(6)))(3))	123456
(1(2(4)(5(6)))(3(7)))	1234576

ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД

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

using namespace std;

template <class T>
class Queue;

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

public:
    BT(stringstream& s);
    void print (Queue<T> &q);
    ~BT();
};

template <class T>
struct El{
    BT<T>* elem;
    El* next;
    El(BT<T>* elem)
    {
        this->elem = elem;
        next = NULL;
    };
};

template <class T>
class Queue{
private:
    El<T>* out;
    El<T>* in;
public:
    Queue();
    bool isEmpty();
```

```
    void push(BT<T>* el);
    BT<T>* pop();
};

template <class T>
Queue<T>::Queue()
{
    out = NULL;
    in = NULL;
}

template <class T>
bool Queue<T>::isEmpty()
{
    return (out ? false : true);
}

template <class T>
void Queue<T>::push(BT<T>* el)
{
    if (in)
    {
        in->next = new El<T>(el);
        in = in->next;
    }
    else
    {
        in = new El<T>(el);
    }
    if (!out)
        out = in;
}

template <class T>
BT<T>* Queue<T>::pop()
{
    El<T>* tmp = out;
    BT<T>* ans = out->elem;
    if (out == in)
        in = NULL;
    out = out->next;
    delete tmp;
    return ans;
}
```

```
template <class T>
void BT<T>::print (Queue<T> &q)
{
   if (left)
      q.push(left);
   if (right)
      q.push(right);
   cout << value;
}

template <class T>
BT<T>::BT(stringstream& s)
{
   left = NULL;
   right = NULL;
   char ch;
   if (s.peek() == '(')
      s >> ch; // remove '('
   s >> 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>
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 << endl;
    s << str;
    BT<int>* el = new BT<int>(s);
    Queue<int> q;
    q.push(el);
    while (!q.isEmpty())
    {
        BT<int>* tmp = q.pop();
```

```
        tmp->print(q);
    }
    cout << 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();
    }
  }
}
```