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. 9304 Сорин А.В.

Преподаватель Филатов А.Ю.

Санкт-Петербург

2020

Цель работы.

Узнать о деревьях и их использовании в практике.

Задание.

Для заданного бинарного дерева b типа BT с произвольным типом элементов определить, есть ли в дереве b хотя бы два одинаковых элемента.

Вариант 4м – реализация бинарного дерева на массиве.

Формат входных и выходных данных.

На вход подается скобочное представление бинарного дерева. Например:

(a^(b(c(d^^)^)(f^^)))

На выходе результат – есть одинаковые элементы, или нет.

Выполнение работы.

Для выполнения задания было создано 2 класса – bin_tree_node и bin_tree.

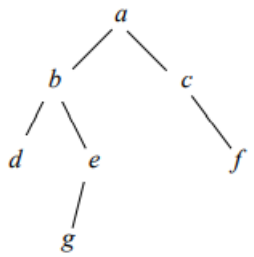
В bin_tree_node содержатся Value – хранимое значение, LeftElemNum и RightElemNum – номера левого и правого элементов. Также есть дефолтные конструктор и деструктор.

В bin_tree хранится head – умный указатель на массив узлов, size – размер массива. Также есть следующие методы bin_tree() – конструктор, ~bin_tree() – дефолтный деструктор, bin_tree(bin_tree_node<base> BTnode) – конструктор, bin_tree(bin_tree<base>& tree) – копирующий конструктор, bin_tree<base>& operator=(bin_tree<base>& tree) – оператор копирования, bin_tree(bin_tree<base>&& tree) – конструктор перемещения, bin_tree<base>& operator=(bin_tree<base>&& tree) – оператор перемещения, void AddToLeftOfPos(size_t pos) – добавление элемента слева, void AddToRightOfPos(size_t pos) – добавления элемента справа, void ReadBT(std::stringstream& Stream) – чтение дерева, bool IsIdenticalNodes(void) –

определение, есть ли в дереве одинаковые элементы, void AddElem(void) добавление элемента, void ReadBTRec(int Ind, std::stringstream& Stream) – рекурсия считывания дерева.

Пример дерева:

Для выражения (a(b(d^^)(e(g^^)^))(c^(f^^)))



Тестирование.

Тестирование проводится при помощи скрипта на python. При этом текст из входного файла подаётся в поток на вход, а получившийся на выходе результат сравнивается в правильным и выводится, пройден тест, или нет. В таблице приведены результаты тестирования.

Таблица Б.1 – Результаты тестирования

№	Входные данные	Выходные данные	Комментарии

1	(a^^)	answer: All nodes are different	Один узел в дереве
2	(a(b^^)(c^^))	All nodes are different	Узел с сыновьями
3	(a(n^(l^^))(k(l^^)^))	There are identical nodes	Дерево с одинаковыми элементами
4	a^(b^^))	Error while entering expression	Ошибка в записи дерева
5	(a(b^^)(c(a^())(n^^)))	Error while entering expression	Ошибка в записи дерева

Выводы.

Стало известно о деревьях и их использовании в практике.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <string>

#include "BinTree.h"


int main() {

	try

	{

		bin_tree<char> BT;


		bool Res = 0;

		std::string Str;

		if (!std::getline(std::cin, Str))

			throw std::runtime_error("Error while reading from
stream");

		std::stringstream Stream(Str);

		BT.ReadBT(Stream);

		Res = BT.IsIdenticalNodes();

		if (Res == true)

			std::cout << "There are identical nodes" << std::endl;

		else

			std::cout << "All nodes are different" << std::endl;

		system("pause");
```

```
	}
	catch (const std::exception& Error)
	{
		std::cout << Error.what();
	}
	return 0;
}
```

Название файла: BinTree.h

```
#ifndef __BIN_TREE_H
#define __BIN_TREE_H

#include <iostream>
#include <stdexcept>
#include <sstream>
#include <memory>
#include <vector>

template <typename base>
class bin_tree_node {
public:
base Value;
int LeftElemNum = -1;
int RightElemNum = -1;
bin_tree_node() = default;
~bin_tree_node() = default;
};

template <typename base>
class bin_tree {
public:
bin_tree() : size(0), head(nullptr) {
```

```
}
~bin_tree() = default;

bin_tree(bin_tree_node<base> BTnode) : size(1) {
	head(new bin_tree_node<base>[1]);
	head[0] = BTnode;
}

bin_tree(bin_tree<base>& tree) {
	size = tree.size;
	head(new bin_tree_node<base>[size]);
	for (int i = 0; i < size; i++) {
		head[i] = tree.head[i];
	}
}
bin_tree<base>& operator=(bin_tree<base>& tree) {
	size = tree.size;
	head(new bin_tree_node<base>[size]);
	for (int i = 0; i < size; i++) {
		head[i] = tree.head[i];
	}
	return *this;
}

bin_tree(bin_tree<base>&& tree) {
	head = std::move(tree.head);
}
bin_tree<base>& operator=(bin_tree<base>&& tree) {
	head = std::move(tree.head);
	return *this;
}

void AddToLeftOfPos(size_t pos) {
	if (pos >= size)
```

```
            throw std::invalid_argument("Error of position");
      if (head[pos].LeftElemNum != -1)
            throw std::invalid_argument("The node is already on the left");
      else {
            head[pos].LeftElemNum = size;
            AddElem();
      }
}

void AddToRightOfPos(size_t pos) {
      if (pos >= size)
            throw std::invalid_argument("Error of position");
      if (head[pos].RightElemNum != -1)
            throw std::invalid_argument("The node is already on the left");
      else {
            head[pos].RightElemNum = size;
            AddElem();
      }
}
void ReadBT(std::stringstream& Stream) {
      if (head == nullptr)
            AddElem();
      char c = 0;
      if (!Stream.get(c))
            throw std::invalid_argument("Error while entering expression");
      if (c != '(')
            throw std::invalid_argument("Error while entering expression");
      ReadBTRec(0, Stream);
}

bool IsIdenticalNodes(void) {
      std::vector<base> Save;

      for (size_t k = 0; k < size; k++) {
```

```
            for (size_t i = 0; i < Save.size(); i++)
                  if (head[k].Value == Save[i])
                        return true;
            Save.push_back(head[k].Value);
      }
      return false;
}
private:
std::shared_ptr<bin_tree_node<base>[]> head;
size_t size;

void AddElem(void) {
      std::shared_ptr<bin_tree_node<base>[]> tmp(new
bin_tree_node<base>[size + 1]);
      for (size_t i = 0; i < size; i++) {
            tmp[i] = head[i];
      }
      head = std::move(tmp);
      size++;
}

void ReadBTRec(int Ind, std::stringstream& Stream) {
      int IsO = 0;
      char c = 0;

      if (!Stream.get(c))
            throw std::invalid_argument("Error while entering expression");
      if (c < 'a' || c > 'z')
            throw std::invalid_argument("Error while entering expression");
      head[Ind].Value = (base)c;

      if (Stream.get(c)) {
            if (c == '^')
                  IsO++;
```

```
            else if (c == '(') {
                  AddToLeftOfPos(Ind);
                  ReadBTRec(size - 1, Stream);
                  IsO++;
            }
            else
                  throw std::invalid_argument("Error while entering
expression");
      }
      else
            throw std::invalid_argument("Error while entering expression");

      if (Stream.get(c)) {
            if (c == '^')
                  IsO++;
            else if (c == '(') {
                  AddToRightOfPos(Ind);
                  ReadBTRec(size - 1, Stream);
                  IsO++;
            }
            else
                  throw std::invalid_argument("Error while entering
expression");
      }
      else
            throw std::invalid_argument("Error while entering expression");

      if (!Stream.get(c))
            throw std::invalid_argument("Error while entering expression");
      if (c != ')')
            throw std::invalid_argument("Error while entering expression");

      return;
}
```

```
};


#endif // __BIN_TREE_H
```

Название файла тестирующей программы: lab3_test.py

```
import os


number_of_tests = 5

exec_file = './lab3'

path_to_tests = 'Tests/tests/test_'

path_to_answers = 'Tests/answers/answers_'

path_to_correct_answers = 'Tests/correct_answers/answers_'

exp = '.txt'


for i in range(number_of_tests):

    num = i + 1

    os.system(

        f'{exec_file} {path_to_tests}{num}{exp} >
{path_to_answers}{num}{exp}')


    ans_1 =
open(f'{path_to_correct_answers}{num}{exp}').readline().rstrip('\n')

    ans_2 = open(f'{path_to_answers}{num}{exp}').readline().rstrip('\n')

    str_test = open(f'{path_to_tests}{num}{exp}').readline()


    print(f'test_{num}:\n  string: {str_test}\n  answer: {ans_1}\n
result: ', end='')

    if ans_1 == ans_2:

        print(f'correct')

    else:

        print(f'incorrect')
```

Название файла: Makefile

```
lab2: ./src/main.cpp
     g++ -std=c++17 ./Src/main.cpp -o lab2

tests: ./lab2_tests/main.cpp
     g++ -std=c++17 ./lab2_tests/main.cpp -o tests
```