МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МОЭВМ

отчет

по лабораторной работе №4

по дисциплине «Алгоритмы и структуры данных»

Тема: Деревья

Студент гр.8304 _____ Холковский К.В.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2019

Задание.

Вариант 3-в

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

Цель работы.

Решить полученную задачу, используя бинарное дерево на векторе. Получения навыков работы с нелинейными структурами данных.

Описание алгоритма.

В начале, программа получает строку – варажение формата (<имя>(<выражение>)(<выражение>)) и записывает из нее данные в дерево функцией `bool mkBinTree(bin_tree<std::string>& abc, std::string& str, size_t& ind)`. После отработки функции, имеем готовое бинарное дерево, остается только проверить на наличие одинаковых элементов. Так как данные дерева хранятся в векторе, проблем проверки не возникает.

Описание функций и структур данных.

- `Struct bin_tree;` - хранит массив и данные о нынешнем индексе элемента, глубине на которой он находится и максимальную глубину.
- `bin_tree::bin_tree(const Elem& val = "")`- конструктор, записывает в корневой элемент данные и инициализирует все поля в bin_tree.
- `void bin_tree::set(const Elem val)` записывает в соответствующий элемент переданные данные.
- `bin_tree& bin_tree::left/right/back()` изменяет cur_ind в соответствии с методом, для left увеличивает в 2 раза и на 1, для right увеличивает в 2 раза и на 2, для back уменьшает на 1 и в 2 раза.
- `const Elem bin_tree::get()` возвращает значение по нынешнему индексу.
- `bin_tree& bin_tree::cl_root()` приводит дерево к девственному состоянию.

Выводы.

Для решения полученной задачи нецелесообразно было использовать бинарное дерево, выгоднее было бы записать все данные в массив и сравнить.

Тестирование

```
Тест 1
Для выражения: (as(df(ef)(bv(tr)))(tr))
Есть одинаковые

Тест 2
Для выражения: (yt(e)(cxc))
Нет одинаковых

Тест 3
Для выражения: dsadasdaa
Некорректный формат ввода

Тест 4
Для выражения: ((((((
Некорректный формат ввода
```

Пример вывода программы

a(b)(c)	**Некорректный формат ввода**
(a(b(a)(a2))(c(a)(a)))	**Есть одинаковые**
((b)(c))	**Некорректный формат ввода**
(yt(e)(cxc))	**Нет одинаковых**

Исходный код

```
#include <iostream>
#include <fstream>
#include <vector>

template <typename Elem>
struct bin_tree {
bin_tree(const Elem& val = ""): cur_deep(1), deep(1), cur_ind(0) {
bt.resize(1);
bt[0] = val;
}

void set(const Elem val) {
bt[cur_ind] = val;
}

bin_tree& left() {
cur_ind = (cur_ind << 1) + 1;
if(++cur_deep > deep) {
```

```
      deep = cur_deep;
      bt.resize((1 << deep) - 1,"");
}
return *this;
}

bin_tree& right() {
cur_ind = (cur_ind << 1) + 2;
if(++cur_deep > deep) {
      deep = cur_deep;
      bt.resize((1 << deep) - 1,"");
}
return *this;
}

bin_tree& back() {
if(cur_ind != 0) {
      cur_ind = (cur_ind - 1) >> 1;
}
return *this;
}

const Elem get() {
return bt[cur_ind];
}

bin_tree& cl_root() {
cur_ind = 0;
deep = 1;
cur_deep = 1;
bt.resize(1);
return *this;
}

bool check() {
for(int i = 0; i < (1 << deep) - 1 ; ++i)
      for(int j = 0; j < (1 << deep) - 1; ++j)
            if((i != j) && (bt[i] != "") && (bt[i] == bt[j]))
                  return true;
return false;
}
private:
int cur_deep;
```

```
int deep;
int cur_ind;
std::vector<Elem> bt;
};

bool mkBinTree(bin_tree<std::string>& abc, std::string& str, size_t& ind) {
if(str[ind] != '(') {
std::cout << "Некорректный формат ввода" << std::endl;
return false;
}
std::string a = "";
while((str[++ind] != '(') && (str[ind] != ')') && (ind != str.size()))
a += str[ind];
if((a == "") && (str[ind] == '(')) {
std::cout << "Некорректный формат ввода" << std::endl;
return false;
}
abc.set(a);
if(str[ind] == ')') {
abc.back();
ind++;
return true;
}
if(mkBinTree(abc.left(), str, ind) == false) {
return false;
}
if(str[ind] == ')') {
abc.back();
ind++;
return true;
}
if(mkBinTree(abc.right(), str, ind) == false) {
return false;
}
ind++;
abc.back();
return true;
}

int main(int argc, char* argv[]) {
bin_tree<std::string> abc;
std::string str;
size_t pos, ind = 0, k = 0;
```

```
if(argc == 1) {
std::getline(std::cin, str);
}
else {
std::cout << std::endl << "Для файла: " << argv[1] << std::endl;
std::ifstream in(argv[1]);
if (!in.is_open()) {
	std::cout << "Не могу открыть файл((" << std::endl;
	return 0;
}
if (in.eof()) {
	std::cout << argv[1] << "Как-то пусто" << std::endl;
	return 0;
}
while (std::getline(in, str)) {
	std::cout << std::endl << "Тест " << ++k << std::endl << "Для выражения: " << str
<< std::endl;
	while((pos = str.find(' ')) != std::string::npos) {
		str.erase(pos,1);
	}
	if(mkBinTree(abc, str, ind)) {
		abc.check() ? std::cout << "Есть одинаковые" << std::endl: std::cout <<
"Нет одинаковых" << std::endl;
	}
	ind = 0;
	abc.cl_root();
}
return 0;
}
while((pos = str.find(' ')) != std::string::npos) {
str.erase(pos,1);
}
if(mkBinTree(abc, str, ind)) {
abc.check() ? std::cout << "Есть одинаковые" << std::endl: std::cout << "Нет
одинаковых" << std::endl;
}
return 0;
}
```