Отчёт по лабораторной работе № 23 по курсу Практикум на ЭВМ

Студент группы М8О-111Б Тимирчев Янис Федорович, № по списку 23

Контакты www, e-mail, icq, skype yanis.timirchev@yandex.ru

Работа выполнена: «26» апреля 2022 г.

Преподаватель: доцент каф.806 Никулин С.П.

Входной контроль знаний с оценкой _____

Отчёт сдан « » _____201 __ г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема:** Динамические структуры. Обработка деревьев

2. **Цель работы:** Составить программу на языке си для обработки деревьев.

3. **Задание** (*вариант №* 25): Определить глубину двоичного дерева

4. **Оборудование(лабораторное)**:
ЭВМ _____, процессор _____, имя узла сети_____ с ОП _____ Мб,
НМД_____Мб. Терминал_____адрес _____. Принтер _____
Другие устройства _____

Оборудование ПЭВМ студента, если использовалось:
Процессор Intel Core i5-7300HQ с ОП 16Гб Мб, НМД 1 ТБ Мб. Монитор 1920x1080 Full HD
Другие устройства _____

5. **Программное обеспечение(лабораторное):**
Операционная система семейства _____, наименование _____ версия _____
интерпретатор команд _____ версия _____
Система программирования _____ версия _____
Редактор текстов _____ версия _____
Утилиты операционной системы_____
Прикладные системы и программы _____
Местонахождение и имена файлов программ и данных _____

Программное обеспечение ЭВМ студента, если использовалось:
Операционная система семейства Linux, наименование Ubuntu версия 20.04.3
интерпретатор команд bash версия 5.0.17 (1).
Система программирования _____версия _____
Редактор текстов _____версия 26.2
Утилиты операционной системы_____
Прикладные системы и программы emacs
Местонахождение и имена файлов программ и данных на домашнем компьютере_____

6. **Идея, метод, алгоритм** решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями)

Для выполнения работы, нам надо реализовать двоичное дерево. Основный функции работы с бинарным деревом будем реализовывать с помощью функций. А именно, создание узла, вставку, вывод, удаление. Для вычисления глубины используем следующую функцию:

```
int depth(Node* tree) {
    int val = 0;
    if (tree != NULL) {
        int lDepth = depth(tree->leftChild);
        int rDepth = depth(tree->rightChild);
        val = lDepth + 1 > rDepth + 1 ? lDepth + 1 : rDepth + 1;
    }
    return val;
}
```

Алгоритм для вычисления максимальной глубины двоичного дерева следующий:
- считаем высоту левого под-дерева от корня.
- считаем высоту правого под-дерева от корня.
- выбираем максимальную высоту между правым и левым поддеревом и добавляем к ней 1 (учитываем сам корневой узел).

7. **Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты либо соображения по тестированию].

```
Commands:
        1. Вставка
        2. Удаление
        3. Вывод
        4. Глубина
        5. Выход
Введите номер команды: 1
Enter num: 2
Введите номер команды: 1
Enter num: 5
Введите номер команды: 1
Enter num: 7
Введите номер команды: 3

Tree:
        7.000000
    5.000000
 2.000000

Введите номер команды: 4
Depth: 3
Введите номер команды: 2
Enter num: 2
Введите номер команды: 3

Tree:
    7.000000
 5.000000

Введите номер команды: 5


...Program finished with exit code
Press ENTER to exit console.
```

Пункты 1-7 отчета составляются ***строго до*** *начала лабораторной работы.*

Допущен к выполнению работы. Подпись преподавателя _____

8. **Распечатка протокола** (подклеить листинг окончательного варианта программы с тестовыми примерами, подписанный преподавателем).

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node Node;
struct Node
{
    double value;
    Node* leftChild;
    Node* rightChild;
};
Node* makeNode(double val)
{
    Node* node = (Node*)malloc(sizeof(Node));
    node->value = val;
    node->leftChild = node->rightChild = NULL;
    return node;
}
void insert(Node* current, Node* val)
{
    if (val->value < current->value) {
        if (current->leftChild == NULL) {
            current->leftChild = val;
            return;
        }
        insert(current->leftChild, val);

    } else {
        if (current->rightChild == NULL) {
            current->rightChild = val;
            return;
        }
        insert(current->rightChild, val);
    }
}
void print(Node* current, int d)
{
    if (current->rightChild != NULL) {
        print(current->rightChild, d + 1);
    }
    printf("%*s%lf\n", 4 * d, " ", current->value);
    if (current->leftChild != NULL) {
        print(current->leftChild, d + 1);
    }
}
void destroy(Node* leaf)
{
    if (leaf != NULL) {
        destroy(leaf->leftChild);
        destroy(leaf->rightChild);
        free(leaf);
    }
}
Node* deleteNode(Node* root, double val)
{
    if (root == NULL)
    return root;
```

```
if (root->value > val) {
        root->leftChild = deleteNode(root->leftChild, val);
        return root;
    }

    if (root->value < val) {
        root->rightChild = deleteNode(root->rightChild, val);
        return root;
    }

    if (root->leftChild == NULL) {
        Node* temp = root->rightChild;
        free(root);
        return temp;
    }

    if (root->rightChild == NULL) {
        Node* temp = root->leftChild;
        free(root);
        return temp;
    }

    Node* tmpParent = root;
    Node* tmp = root->rightChild;
    while (tmp->leftChild != NULL) {
        tmpParent = tmp;
        tmp = tmp->leftChild;
    }

    if (tmpParent != root)
    tmpParent->leftChild = tmp->rightChild;
    else
    tmpParent->rightChild = tmp->rightChild;

    root->value = tmp->value;
    free(tmp);
    return root;
}


int depth(Node* tree) {
    int val = 0;
    if (tree != NULL) {
        int lDepth = depth(tree->leftChild);
        int rDepth = depth(tree->rightChild);
        val = lDepth + 1 > rDepth + 1 ? lDepth + 1 : rDepth + 1;
    }
    return val;
}



int main() {
    Node* root = NULL;
    int comm;
    printf("Commands:\n\t1. Вставка\n\t2. Удаление\n\t3. Вывод\n\t4. Глубина\n\t5. Выход\n");

    do {
```

```
  printf("Введите номер команды: ");
        if (scanf("%i", &comm) != 1)
        getchar();
        double val;

        switch (comm) {

            case 1:
            printf("Enter num: ");
            scanf("%lf", &val);
            if (root == NULL) {
                root = makeNode(val);
            } else {
                insert(root, makeNode(val));
            }
            break;

            case 2:
            printf("Enter num: ");
            scanf("%lf", &val);
            root = deleteNode(root, val);
            break;

            case 3:
            if (root != NULL) {
                printf("\nTree:\n");
                print(root, 0);
                printf("\n");
            } else {
                printf("Empty tree!\n");
            }
            break;

            case 4:
            printf("Depth: %d\n", depth(root));
            break;
            default:
            break;
        }

    } while (comm != 5);
    destroy(root);
    return 0;
}
```

9. **Дневник отладки** должен содержать дату и время сеансов отладки и основные события (ошибки в сценарии и программе, нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

№	Лаб. или дом.	Дата	Время	Событие	Действие по исправлению	Примечание

10. **Замечания автора** по существу работы _____

11. **Выводы**

Таким образом, я реализовал двоичное дерево на языке Си и применил алгоритм вычисления его максимальной глубины.

Недочёты при выполнении задания могут быть устранены следующим образом: _____

Подпись студента _____