Отчет по лабораторной работе № 25 по курсу «Практикум на ЭВМ»

Студент группы Алапанова Эльза Халилевна, № по списку 3

Контакты e-mail : alapanowa02@yandex.ru

Работа выполнена: «21» мая 2021г.

Преподаватель: каф. 806 Найденов Иван Евгеньевич

Отчет сдан « » _____20 ___ г.,
итоговая оценка _____

Подпись преподавателя
_____

1. Тема: Динамические структуры данных.
2. Цель работы: Составить программу на Си построения и обработки дерева.
3. Задание (*вариант № 9*)**:** Определить число вершин двоичного дерева, степень которых равна значению.
4. Оборудование (студенческое)

Процессор Intel® Core™ i5-10210 @ 1.60 GHz с ОП 8192 Мб, НМД 512 Uб.
Монитор 1920 x 1080

5. Программное обеспечение (студенческое):
Операционная система семейства Ubuntu, наименование_Ubuntu 20.04.2 LTS_ версия
интерпретатор команд _____ версия _____.
Система программирования
_____версия
_____
Редактор текстов
_____ версия
_____
Утилиты операционной системы
_____
_____
_____
Прикладные системы и программы Sublime Text

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

- Составить элемент дерева, в котором есть значение, ссылка на левый элемент и на правый.
- Составить структуру дерева, с указателем на корень
- Составить функции инициализации элементов и самого дерева(с динамическим выделением памяти)
- Составить функции заполнения, удаления и печати дерева
- Составить функцию нахождения подсчёта числа вершин

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

-Изучение теории

-Написание программы

-Отладка

-Создание протокола

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

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

elza@elza-NBLB-WAX9N:~/23$ cat lab23.c

```
#include <stdio.h>

#include <stdlib.h>


typedef struct TreeNode TreeNode;

typedef double T;


struct TreeNode

{

   T data;

   TreeNode* l;

   TreeNode* r;

};


TreeNode* makeNode(double val) {

   TreeNode* node = (TreeNode*)malloc(sizeof(TreeNode));

   node->data = val;

   node->l = NULL;

   node->r = NULL;

   return node;

}


void add(TreeNode* cur, TreeNode* newel) {

   if (newel->data < cur->data) {

      if (cur->l == NULL) {

         cur->l = newel;

         return;

      }

      add(cur->l, newel);

   }

   else {

      if (cur->r == NULL) {

         cur->r = newel;
```

```
            return;
        }
        add(cur->r, newel);
    }
}

TreeNode** find(TreeNode** cur, double val) {
    if (*cur == NULL) {
        return cur;
    }
    if (val < (*cur)->data) {
        return find(&((*cur)->l), val);
    } else if (val > (*cur)->data) {
        return find(&((*cur)->r), val);
    } else {
        return cur;
    }
}

void clearTree(TreeNode* cur) {
    if (cur->l != NULL) {
        clearTree(cur->l);
        free(cur->l);
    }
    if (cur->r != NULL) {
        clearTree(cur->r);
        free(cur->r);
    }
}

void printTree(TreeNode* cur, int d) {
    if (cur->r != NULL) {
        printTree(cur->r, d + 1);
    }
```

```
    printf("%*s%lf\n", 4 * d, " ", cur->data);
    if (cur->l != NULL) {
        printTree(cur->l, d + 1);
    }
}

TreeNode* Min(TreeNode* cur){
    if (cur->l->l == NULL){
        return cur;
    }
    return Min(cur->l);
}

void DeleteElem(TreeNode** cur, double val, TreeNode* par){
    if (val < (*cur)->data){
        DeleteElem(&((*cur)->l), val, *cur);
    }
    else if (val > (*cur)->data){
        DeleteElem(&((*cur)->r), val, *cur);
    }
    else {
        if (((*cur)->l == NULL) && ((*cur)->r == NULL)) {
            if (par) {
                if (par->l) {
                    if (par->l->data == (*cur)->data) {
                        par->l = NULL;
                    }
                }
                if (par->r) {
                    if (par->r->data == (*cur)->data) {
                        par->r = NULL;
                    }
                }
            }
```

```
            free(*cur);
            *cur = NULL;
        } else if (((*cur)->l == NULL) || ((*cur)->r == NULL)) {
            TreeNode* Help = NULL;
            if ((*cur)->l) {
                Help = (*cur)->l;
            } else {
                Help = (*cur)->r;
            }
            (*cur)->l = Help->l;
            (*cur)->r = Help->r;
            (*cur)->data = Help->data;
            free(Help);
        } else {
            if ((*cur)->r->l == NULL) {
                (*cur)->data = (*cur)->r->data;
                TreeNode* rr = (*cur)->r->r;
                free((*cur)->r);
                (*cur)->r = rr;
            } else {
                TreeNode* Minpar = Min((*cur)->r);
                free(Minpar->l);
                Minpar->l = NULL;
            }
        }
    }
}

TreeNode* func(TreeNode* cur, double d){
    if(cur->data > d){
        return func(cur->l, d);
    }
    if(cur->data < d){
        return func(cur->r, d);
```

```
    }
    if(cur->data == d){
        if(d=2 && cur->l && cur->r){
            return cur;
        }
        if(d=1 && ((cur->l && !cur->r)||(cur->r && !cur->l))){
            return cur;
        }
        if(d=0 && cur->l==NULL && cur->r==NULL){
            return cur;
        } else{
        return NULL;
    }
    } else{
        return NULL;
    }
}

T main() {
    int s = 1, n, d = 0;
    double c=0;
    TreeNode* root = NULL;
    double value;
    while (s != 0){
        printf("Menu option:\n");
        printf("1 - to stop\n");
        printf("2 - to add element in tree\n");
        printf("3 - to print tree\n");
        printf("4 - to delete element in tree\n");
        printf("5 - to do task #9\n");
        scanf("%d", &s);
        switch (s){
            case 1:
                if (root != NULL){
```

```
        clearTree(root);
    }
    return 0;
case 2:
    printf("Cardinality:");
    scanf("%d", &n);
    for (n; n > 0; n--) {
        scanf("%lf", &value);
        if (root == NULL) {
            root = makeNode(value);
        }else {
            add(root, makeNode(value));
        }
    }
    break;
case 3:
    printf("Tree:\n");
    printTree(root, 0);
    break;
case 4:
    if (root == NULL){
        printf("Mistake!!! You must create the tree\n");
    }else {
    printf("Element:");
    scanf("%lf", &value);
    DeleteElem(&root, value, NULL);
    printf("Successfully deleted\n");
    }
    break;
case 5:
    if (root == NULL){
        printf("Mistake!!! You must create the tree\n");
    }else {
        for(int f=0; f<=2; f++){
```

```
                    if(func(root, f)){
                        c++;
                    }
                }
            }
            printf("Quantity:%lf\n", c);
            c=0;
            break;
        }
    }
    return 0;
}
```

```
elza@elza-NBLB-WAX9N:~$ gcc lab23.c
elza@elza-NBLB-WAX9N:~$ ./a.out
Menu option:
1 - to stop
2 - to add element in tree
3 - to print tree
4 - to delete element in tree
5 - to do task #9
2
Cardinality:5
2
3
1
0
4
Menu option:
1 - to stop
2 - to add element in tree
3 - to print tree
4 - to delete element in tree
5 - to do task #9
3
```

Tree:

4.000000

3.000000

2.000000

1.000000

0.000000

Menu option:

1 - to stop

2 - to add element in tree

3 - to print tree

4 - to delete element in tree

5 - to do task #9

4

Element:4

Successfully deleted

Menu option:

1 - to stop

2 - to add element in tree

3 - to print tree

4 - to delete element in tree

5 - to do task #9

3

Tree:

3.000000

2.000000

1.000000

0.000000

Menu option:

1 - to stop

2 - to add element in tree

3 - to print tree

4 - to delete element in tree

5 - to do task #9

5

Quantity:3.000000

Menu option:

1 - to stop

2 - to add element in tree

3 - to print tree

4 - to delete element in tree

5 - to do task #9

4

Element:1

Successfully deleted

Menu option:

1 - to stop

2 - to add element in tree

3 - to print tree

4 - to delete element in tree

5 - to do task #9

3

Tree:

3.000000

2.000000

0.000000

Menu option:

1 - to stop

2 - to add element in tree

3 - to print tree

4 - to delete element in tree

5 - to do task #9

5

Quantity:2.000000

Menu option:

1 - to stop

2 - to add element in tree

3 - to print tree

4 - to delete element in tree

5 - to do task #9

2

Cardinality:3

1.5

10.7

0.4

Menu option:

1 - to stop

2 - to add element in tree

3 - to print tree

4 - to delete element in tree

5 - to do task #9

3

Tree:

10.700000

3.000000

2.000000

1.500000

0.400000

0.000000

Menu option:

1 - to stop

2 - to add element in tree

3 - to print tree

4 - to delete element in tree

5 - to do task #9

1

elza@elza-NBLB-WAX9N:~$ ./a.out

Menu option:

1 - to stop

2 - to add element in tree

3 - to print tree

4 - to delete element in tree

5 - to do task #9

2

Cardinality:1

0

Menu option:

1 - to stop

2 - to add element in tree

3 - to print tree

4 - to delete element in tree

5 - to do task #9

5

Quantity:1.000000

Menu option:

1 - to stop

2 - to add element in tree

3 - to print tree

4 - to delete element in tree

5 - to do task #9

4

Element:0

Successfully deleted

Menu option:

1 - to stop

2 - to add element in tree

3 - to print tree

4 - to delete element in tree

5 - to do task #9

2

Cardinality:1

1

Menu option:

1 - to stop

2 - to add element in tree

3 - to print tree

4 - to delete element in tree

5 - to do task #9

5

Quantity:0.000000

Menu option:

1 - to stop

2 - to add element in tree

3 - to print tree

4 - to delete element in tree

5 - to do task #9

1

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
1						
2						

10. Замечания автора по существу работы : замечаний нет.
11. Выводы : Мной была изучена такая структура, как бинарное дерево. Составлена программа реализации такого дерева. Бинарное дерево – очень полезная структура, которая может решать различные задачи
Недочёты при выполнении задания могут быть устранены следующим образом:
_____

_____Подпись студента _____