Отчет по лабораторной работе №23 по курсу практикум на ЭВМ

Студент группы М8О-102Б-21 Кажекин Денис Андреевич
№ по списку 9

Контакты www, e-mail, icq, skype deniskazhekin@mail.ru

Работа выполнена: « » _____202___г.

Преподаватель: доцент каф. 806 Никулин Сергей Петрович

Входной контроль знаний с оценкой _____

Отчет сдан « » _____202 __ г., итоговая оценка _____

Подпись преподавателя _____

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

2. **Цель работы:** Научиться работать с динамическими структурами данных и обрабатывать деревья

3. **Задание** (*вариант №* 14)**:** Проверить находятся ли все листья дерева на одном уровне.

4. **Оборудование** (лабораторное):
ЭВМ Intel Pentium G2140, процессор 3.30 GHz, имя узла сети Cameron с ОП 8096 Мб,
НМД 7906 Мб. Терминал ASUS адрес dev/pets/3 Принтер HP Laserjet 6P
Другие устройства _____

5. **Программное обеспечение (лабораторное):**
Операционная система семейства Unix, наименование Ubuntu версия 18.15.0
интерпретатор команд bash версия 4.4.20
Система программирования GNU версия 5.8.13
Редактор текстов emacs версия 25.2.2
Утилиты операционной системы cat, ls
Прикладные системы и программы
Местонахождение и имена файлов программ и данных /home/stud

Программное обеспечение ЭВМ студента, если использовалось:
Операционная система семейства Unix, наименование Mint версия 20.1
интерпретатор команд bash версия 5.0.17
Система программирования _____ версия
Редактор текстов emacs версия 25.2.2
Утилиты операционной системы cat, ls
Прикладные системы и программы
Местонахождение и имена файлов программ и данных на домашнем компьютере /home/deniskazhekin

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

Опишем следующие структуры:

- struct tnode {
  float value;
  struct tnode *son;
  struct tnode *brother;
  struct tnode *parent;
  };
  Структура узла дерева. Хранит значение узла, указатель на старшего сына, указатель на следующего брата, указатель на родителя.
- typedef struct {
  node *root;
  } Tree;
  Структура самого дерева. Хранит указатель на корень.

В основной части программы будем использовать меню, в котором есть 6 опций:

1. Создание дерева (Create tree)
   Запрашивает значение корня дерева, а затем создает дерево, вызывая функцию create_tree
2. Добавление узла в дерево (Add node to tree)
   Запрашивает значение добавляемого узла, а затем вызывает функцию add_node
3. Удаление узла дерева (Delete node from tree)
   Вызывает функцию delete_node
4. Выполнение задания (вычисление степени дерева) (Task)
   Вызывает функцию task от корня с максимальным значением 0, а затем выводит ответ.
5. Печать дерева (Print tree)
   Вызывает функцию print_tree
6. Выход (Exit)
   Выходит из меню

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

Тесты:

1) Дерево, состоящее из одного корня (1)

2) a
   b
   c
   d
   f

3) a
   b
   d
   g
   c
   r

4) a
   b
   q
   t
   y
   d
   g
   c
   r

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).

deniskazhekin@MacBook-Air-Denis Desktop % cat tree.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

struct tnode {
    char value;
    struct tnode *son;
    struct tnode *brother;
    struct tnode *parent;
};
typedef struct tnode node;
typedef struct {
    node *root;
} Tree;


node *create_node(char f, node *par) {
    node *t;
    t = (node *) malloc(sizeof(node));
    t->value = f;
    t->son = NULL;
    t->brother = NULL;
    t->parent = par;
    return t;
}

Tree *create_tree(char f) {
    Tree *t;
    t = (Tree *) malloc(sizeof(Tree));
    t->root = create_node(f, NULL);
    return t;
}


node *search_tree(node *t, char f) {
    if (t == NULL){
        return  t;
    }
    node *tree = NULL;
    if (t->value == f) {
        return t;
    }
    if (t->son != NULL) {
        tree = search_tree(t->son, f);
    }
    if (tree == NULL) {
        tree = search_tree(t->brother, f);
    }
    return tree;
}

void add_node_in_tree(Tree *tree, char par_f, char f) {
    node *t = tree->root;
    t = search_tree(t, par_f);
    if (t == NULL) {
        printf("%c not contains in tree\n", par_f);
        return;
    }
    if (t->son == NULL) {
        t->son = create_node(f, t);
    } else {
        t = t->son;
        while (t->brother != NULL) {
            t = t->brother;
        }
```

```
        t->brother = create_node(f, t->parent);
    }
}

void delete_node(Tree *tree, char f) {
    node *t = tree->root;
    t = search_tree(t, f);
    if (t == NULL) {
        printf("%c not contains in tree\n", f);
        return;
    }
    if (t->parent == NULL){
        free(t);
        return;
    }
    if (t->parent->son == t){
        t->parent->son = t->brother;
    }
    else{
        node *tr = t->parent->son;
        while (tr->brother != t){
            tr = tr->brother;
        }
        tr->brother = t->brother;
    }
    free(t);
}

void print_tree(node *t, int x) {
    if (t == NULL) {
        return;
    }
    for (int i = 0; i < x; i++) {
        printf("\t");
    }
    printf("%c\n", t->value);
    print_tree(t->son, x + 1);
    print_tree(t->brother, x);
}

int n = 0;

bool task(node *t, int k) {
    if (t == NULL) {
        return true;
    }
    if (t->son == NULL && t->brother == NULL) {
        if (n == 0) {
            n = k;
        } else {
            return n == k;
        }
    }
    return task(t->son, k + 1) && task(t->brother, k);
}


int main() {
    Tree *t = NULL;
    int choose, g = 1;
    while (g) {
        printf("1. Create tree\t 2. Add node to tree\t 3. Delete node from tree\t 4. Task\t 5. Print tree\t 6. Exit \n");
        scanf("%d", &choose);
        switch (choose) {
            case 1: {
                printf("Write tree's root\n");
                char f;
                scanf(" %c", &f);
```

```
            t = create_tree(f);
            break;
        }
        case 2: {
            printf("Write tree node value\n");
            char f, par_f;
            scanf(" %c", &f);
            printf("Write parent value\n");
            scanf(" %c", &par_f);
            add_node_in_tree(t, par_f, f);
            break;
        }
        case 3: {
            printf("Write tree node value\n");
            char f;
            scanf(" %c", &f);
            delete_node(t, f);
            break;
        }
        case 4: {
            task(t->root, 0) ? printf("All the leaves of the tree are on the same level\n") : printf(
                    "Not all the leaves of the tree are on the same level\n");
            break;
        }

        case 5: {
            print_tree(t->root, 0);
            break;
        }
        case 6: {
            g = 0;
            break;
        }
        default: {
            printf("Wrong answer\n");
        }
      }
   }
   return 0;
}
```

```
deniskazhekin@MacBook-Air-Denis Desktop % gcc tree.c
deniskazhekin@MacBook-Air-Denis Desktop % ./a.out
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
1
Write tree's root
a
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
5
a
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
4
All the leaves of the tree are on the same level
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
b
Write parent value
```

```
a
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
c
Write parent value
a
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
d
Write parent value
a
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
f
Write parent value
a
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit

5
a
    b
    c
    d
    f
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
4
All the leaves of the tree are on the same level
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
3
Write tree node value
c
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
3
Write tree node value
f
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
g
Write parent value
a
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
```

```
c
Write parent value
g
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
r
Write parent value
g
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
5
a
     b
     d
     g
          c
          r
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
4
Not all the leaves of the tree are on the same level
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
q
Write parent value
b
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
t
Write parent value
b
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
y
Write parent value
b
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Task
5. Print tree     6. Exit
5
a
     b
          q
          t
          y
     d
     g
          c
          r
```

```
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
4
Not all the leaves of the tree are on the same level
3
Write tree node value
b
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
3
Write tree node value
d
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
h
Write parent value
r
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
o
Write parent value
h
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
5
a
    g
        c
        r
            h
                o
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
4
Not all the leaves of the tree are on the same level
2
Write tree node value
i
Write parent value
r
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
3
Write tree node value
o
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
5
a
    g
        c
        r
```

```
            h
            i
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
4
Not all the leaves of the tree are on the same level
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
3
Write tree node value
c
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
4
All the leaves of the tree are on the same level
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
5
a
    g
        r
            h
            i
2
Write tree node value
e
Write parent value
a
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
4
Not all the leaves of the tree are on the same level
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
5
a
    g
        r
            h
            i
    e
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
t
Write parent value
e
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
b
Write parent value
t
```

```
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
4
All the leaves of the tree are on the same level
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
5
a
    g
        r
            h
            i
    e
        t
            b
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
3
Write tree node value
i
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
5
a
    g
        r
            h
    e
        t
            b
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
m
Write parent value
h
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
l
Write parent value
h
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
v
Write parent value
b
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
2
Write tree node value
w
```

```
Write parent value
b
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
5
a
    g
        r
            h
                m
                l
    e
        t
            b
                v
                w
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
4
All the leaves of the tree are on the same level
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
3
Write tree node value
h
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
3
Write tree node value
r
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
3
Write tree node value
c
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
3
Write tree node value
g
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
5
a
1. Create tree    2. Add node to tree  3. Delete node from tree   4. Tas
5. Print tree     6. Exit
```

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**

Я научился работать с динамическими структурами и обрабатывать деревья

Недочёты при выполнении задания могут быть устранены следующим образом:

_____

_____

_____

Подпись студента _____