Отчет по лабораторной работе №23 по курсу практикум на ЭВМ

Студент группы М8О-107Б-22 Брюханов Захар Дмитриевич, № по списку 5

Контакты e-mail: br_zahar@mail.ru; telegram: @br_zahar

Работа выполнена: «25» марта 2023 г.

Преподаватель: Аносов Наталья Павловна

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 № 5*)**:** Определить значение нетерминальной вершины дерева с максимальной глубиной.

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 _____

Оборудование ПЭВМ студента, если использовалось:
Процессор M1 Pro с 10-ядерным процессором и 14-ядерным графическим процессором с ОП 16 Гб, НМД 512 Гб. Дисплей Liquid Retina XDR
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
Прикладные системы и программы _____
Местонахождение и имена файлов программ и данных stud/208104

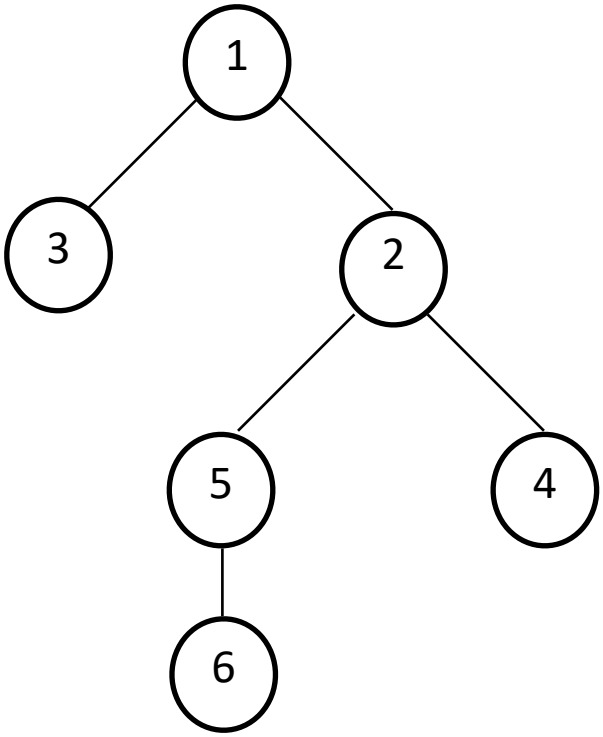
Программное обеспечение ЭВМ студента, если использовалось:
Операционная система Mac OS версия 13.2.1
интерпретатор команд bash версия 5.0.17
Система программирования Clion версия 2022.3.3
Редактор текстов emacs версия 25.2.2
Утилиты операционной системы cat
Прикладные системы и программы _____
Местонахождение и имена файлов программ и данных на домашнем компьютере /Users/br_zahar

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

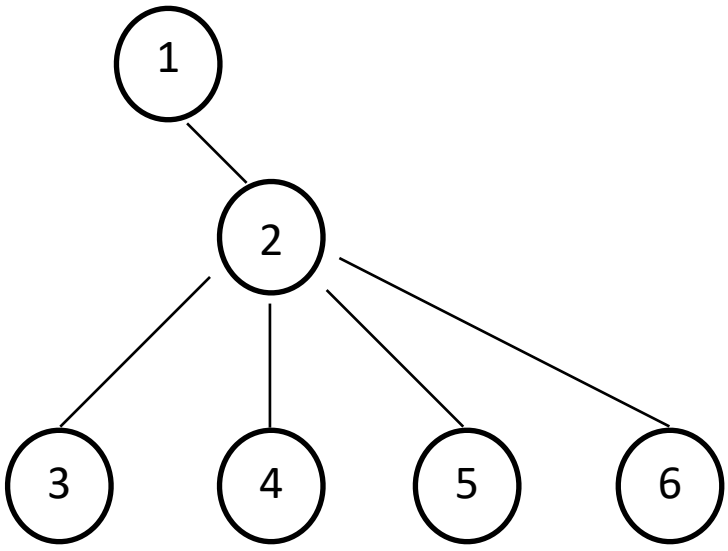
- Изучить материалы по использованию LATEX
- Переписать 2 страницы учебника, стараясь повторять все их содержимое

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

Тесты:



Ответ - 5



Ответ - 2

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).

Программа:

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

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

node *create_node(float 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(float f) {
```

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

```
node *search_tree(node *t, float 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, float par_f, float f) {
    node *t = tree->root;
    t = search_tree(t, par_f);
    if (t == NULL) {
```

```
        printf("%-.2f 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, float f) {
    node *t = tree->root;
    t = search_tree(t, f);
    if (t == NULL) {
        printf("%-.2f 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("%-.2f\n", t->value);
    print_tree(t->son, x + 1);
    print_tree(t->brother, x);
}

int node_degree(node *t) {
    if (t == NULL || t->son == NULL) {
        return 0;
    }
    int n = 1;
    t = t->son;
```

```
    while (t->brother != NULL) {
        t = t->brother;
        n++;
    }
    return n;
}

int max(int a, int b) {
    return a > b ? a : b;
}

node *max_depth_node;
int max_depth;

void dfs(node *t, int depth) {
    if (t == NULL) {
        return;
    }
    if (depth > max_depth) {
        max_depth = depth;
        max_depth_node = t;
    }
    dfs(t->son, depth + 1);
    dfs(t->brother, depth);
}

float task(node *t) {
    max_depth_node = NULL;
```

```
    max_depth = 0;
    dfs(t, 1);
    return max_depth_node->parent->value;
    //    return max_depth_node->value;
}


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");
                float f;
                scanf("%f", &f);
                t = create_tree(f);
                break;
            }
            case 2: {
                printf("Write tree node value\n");
                float f, par_f;
                scanf("%f", &f);
```

```
                printf("Write parent value\n");
                scanf("%f", &par_f);
                add_node_in_tree(t, par_f, f);
                break;
            }
            case 3: {
                printf("Write tree node value\n");
                float f;
                scanf("%f", &f);
                delete_node(t, f);
                break;
            }
            case 4: {
                int ans = task(t->root);
                printf("the inner vertex with the maximum depth
is %d\n", ans);
                break;
            }
            case 5: {
                print_tree(t->root, 0);
                break;
            }
            case 6: {
                g = 0;
                break;
            }
            default: {
                printf("Wrong answer\n");
```

```
            }
        }
    }
    return 0;
}
```

Результат:

Тест 1:

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Task 5. Print tree 6. Exit
1
Write tree's root
1
1. Create tree 2. Add node to tree 3. Delete node from tree 4. Task 5. Print tree 6. Exit
2
Write tree node value
2
Write parent value
1
1. Create tree 2. Add node to tree 3. Delete node from tree 4. Task 5. Print tree 6. Exit
2
Write tree node value
3
Write parent value
1
1. Create tree 2. Add node to tree 3. Delete node from tree 4. Task 5. Print tree 6. Exit
2
Write tree node value
4
Write parent value
2
1. Create tree 2. Add node to tree 3. Delete node from tree 4. Task 5. Print tree 6. Exit
2
Write tree node value
5
Write parent value
2
1. Create tree 2. Add node to tree 3. Delete node from tree 4. Task 5. Print tree 6. Exit
2
Write tree node value
6
Write parent value
5
1. Create tree 2. Add node to tree 3. Delete node from tree 4. Task 5. Print tree 6. Exit
5
1.00
        2.00
                4.00
                5.00
                        6.00
        3.00
1. Create tree 2. Add node to tree 3. Delete node from tree 4. Task 5. Print tree 6. Exit
4
the inner vertex with the maximum depth is 5

Тест 2:

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Task 5. Print tree 6. Exit
1
Write tree's root
1

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Task 5. Print tree 6. Exit
2
Write tree node value
2
Write parent value
1
1. Create tree 2. Add node to tree 3. Delete node from tree 4. Task 5. Print tree 6. Exit
2
Write tree node value
3
Write parent value
2
1. Create tree 2. Add node to tree 3. Delete node from tree 4. Task 5. Print tree 6. Exit
2
Write tree node value
4
Write parent value
2
1. Create tree 2. Add node to tree 3. Delete node from tree 4. Task 5. Print tree 6. Exit
2
Write tree node value
5
Write parent value
2
1. Create tree 2. Add node to tree 3. Delete node from tree 4. Task 5. Print tree 6. Exit
2
Write tree node value
6
Write parent value
2
1. Create tree 2. Add node to tree 3. Delete node from tree 4. Task 5. Print tree 6. Exit
5
1.00
2.00
3.00
4.00
5.00
6.00
1. Create tree 2. Add node to tree 3. Delete node from tree 4. Task 5. Print tree 6. Exit
4
the inner vertex with the maximum depth is 2

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.	Дат	Врем	Событие	Действие по	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 _____