Отчёт по лабораторной работе № 23

по курсу «Языки и методы программирования».

Выполнил студент группы М8О-111Б-23: Воробьев Глеб Янович № по списку 5.

Контакты: koshastet13@gmail.com

Работа выполнена: «3» апреля 2024 г.

Преподаватель: каф. 806 Никулин Сергей Петрович

Входной контроль знаний с оценкой: _____

Отчет сдан «4» апреля 2024 г.

Итоговая оценка: _____

Подпись преподавателя: _____

1. **Тема:**
Динамические структуры данных, обработка деревьев.
2. Цель работы:
Составить программу на языке Си для построения и обработки дерева общего вида, содержащее узлы типа float.
3. Задание:
Вариант 8 – определить число вершин дерева, степень которых совпадает со значением элемента.
4. Оборудование:
Процессор AMD Ryzen 5 7640HS.
ОП 16 ГБ.
SSD 512 ГБ.
Монитор 2560x1600~165Hz.
5. Программное обеспечение:
Операционная система семейства Unix.
Наименование Ubuntu версия 22.04.3.
Интерпретатор команд GNU bash версия 6.2.0.
Система программирования -.
Редактор текстов Visual Studio Code.
6. Идея, метод, алгоритм решения задачи:

Описание основных функций скрипта:

1. ***createNode***: Создает новый узел дерева с заданным значением и инициализирует его поля.
2. ***findNode***: Ищет узел по его значению в дереве. Функция рекурсивно обходит дерево в глубину.
3. ***push***: Добавляет новый узел как потомка к указанному родителю. Если у родителя уже есть потомки, новый узел добавляется в конец списка потомков.
4. ***deleteNode***: Удаляет узел из дерева вместе со всеми его потомками. Функция также обновляет связи между родительским и соседними узлами, чтобы удалить ссылки на удаляемый узел.
5. printTree: Выводит структуру дерева в консоль. Для каждого узла выводится его значение и отступ, соответствующий его уровню в дереве.
6. ***getNodeDegree***: Возвращает степень узла, то есть количество его потомков.
7. ***getAns***: Выполняет особый алгоритмический расчет над деревом, суть которого заключается в подсчете количества узлов, значение которых равно их степени (количеству потомков).
8. Основные операции, выполняемые пользователем через консоль (***stateOne***, ***stateTwo***, ***stateThree***, ***stateFour***), представляют собой добавление нового узла, удаление узла, вывод структуры дерева и получение ответа от алгоритма ***getAns*** соответственно.

Скрипт начинается с запроса у пользователя значения для корневого узла, после чего предлагает меню с выбором действий. Пользователь может многократно выбирать действия для взаимодействия с деревом до тех пор, пока не выберет опцию выхода (0).

7. Сценарий выполнения работы:

Файл node.h

```c
#ifndef __NODE_H__
#define __NODEENT_H__

typedef struct TreeNode{
   float key;
   int level;
   struct TreeNode *parent;
   struct TreeNode *firstChild;
   struct TreeNode *nextBrother;
} TreeNode;

TreeNode *createNode(float key);

TreeNode *findNode(TreeNode *root, float key);

void push(TreeNode *parent, float key);

void deleteNode(TreeNode *node);

void printTree(TreeNode *node, int level);

int getNodeDegree(TreeNode *node);

int getAns(TreeNode *node);

void stateOne(TreeNode *root);

void stateTwo(TreeNode *root);

void stateThree(TreeNode *root);

void stateFour(TreeNode *root);

#endif
```

Файл main.c

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

int main()
{
   int state;
   float parent_value;

   printf("Give value of root\n");
```

```
    scanf("%f", &parent_value);

    TreeNode *root = createNode(parent_value);

    printf("Choose option: 1 - add new node; 2 - delete node; 3 - print
tree; 4 - get answer; 0 - exit\n");

    while (scanf("%d", &state))
    {
        if (state == 1) {
            stateOne(root);
        } else if (state == 2) {
            stateTwo(root);
        } else if (state == 3) {
            stateThree(root);
        } else if (state == 4) {
            stateFour(root);
        } else if (state == 0) {
            break;
        } else {
            printf("Choose option from 1 to 4\n");
        }
    }

    deleteNode(root);

    return 0;
}

TreeNode *createNode(float key)
{
    TreeNode *newNode = malloc(sizeof(TreeNode));
    if (newNode == NULL) {
        printf("Malloc returned NULL\n");
        exit(1);
    }

    newNode->key = key;
    newNode->level = 1;
    newNode->firstChild = NULL;
    newNode->nextBrother = NULL;

    return newNode;
}

TreeNode *findNode(TreeNode *root, float key)
{
    if (root == NULL)
```

```
        return NULL;

    if (root->key == key)
        return root;

    TreeNode *child = root->firstChild;

    while (child) {
        TreeNode *found = findNode(child, key);

        if (found)
            return found;

        child = child->nextBrother;
    }

    return NULL;
}

void push(TreeNode *parent, float key)
{
    if (parent == NULL) {
        printf("Error: parent node not founded\n");
        return;
    }

    TreeNode *child = createNode(key);
    child->level = parent->level + 1;
    child->parent = parent;

    if (parent->firstChild) {
        TreeNode *lastChild = parent->firstChild;

        while (lastChild->nextBrother) {
            lastChild = lastChild->nextBrother;
        }

        lastChild->nextBrother = child;
    } else {
        parent->firstChild = child;
    }

}

void deleteNode(TreeNode *node) {

    if (node == NULL) return;
```

```
    if (node->parent) {
        if (node->parent->firstChild == node) {
            node->parent->firstChild = node->nextBrother;
        } else {
            TreeNode *temp = node->parent->firstChild;
            while (temp && temp->nextBrother != node) {
                temp = temp->nextBrother;
            }
            if (temp) {
                temp->nextBrother = node->nextBrother;
            }
        }
    }

    while (node->firstChild) {
        deleteNode(node->firstChild);
    }

    free(node);
}

void printTree(TreeNode *root, int level)
{
    if (root == NULL)
        return;

    for (int i = 0; i < level; i++)
        printf("  ");

    printf("%.2f\n", root->key);

    TreeNode *child = root->firstChild;

    while(child) {
        printTree(child, level + 1);
        child = child->nextBrother;
    }
}

int getNodeDegree(TreeNode *node)
{
    int degree = 0;

    TreeNode *lastChild = node->firstChild;

    while (lastChild) {
        degree++;
        lastChild = lastChild->nextBrother;
```

```
    }

    return degree;
}

int getAns(TreeNode *node)
{
    if (node == NULL)
        return 0;

    int counter = (node->key == (float)getNodeDegree(node)) ? 1 : 0;

    TreeNode *current = node->firstChild;

    while (current) {
        counter += getAns(current);
        current = current->nextBrother;
    }

    return counter;
}

void stateOne(TreeNode *root)
{
    float key, parent_value;

    printf("Print: parent_value child_value\n");
    scanf("%f %f", &parent_value, &key);

    TreeNode *parent = findNode(root, parent_value);
    push(parent, key);
    printf("Choose new option\n");
}

void stateTwo(TreeNode *root)
{
    float keyDeletedNode;

    printf("Print node for deleting:\n");
    scanf("%f", &keyDeletedNode);

    TreeNode *deletedNode = findNode(root, keyDeletedNode);
    deleteNode(deletedNode);

    printf("Choose new option\n");
}

void stateThree(TreeNode *root)
```

```
{
    printf("Your tree:\n");

    printTree(root, 0);

    printf("Choose new option\n");
}

void stateFour(TreeNode *root)
{
    int answer = getAns(root);

    printf("Answer is %d.\nChoose new option:\n", answer);
}
```

Допущен к выполнению работы. **Подпись преподавателя _____**

8. Распечатка протокола:

```
////////////////////////////////////////////////////////////////////////////////

Give value of root
1.4
Choose option: 1 - add new node; 2 - delete node; 3 - print tree; 4 - get
answer; 0 - exit
1
Print: parent_value child_value
1.4 2
Choose new option
1
Print: parent_value child_value
2 6.5
Choose new option
1
Print: parent_value child_value
2 1
Choose new option
1
Print: parent_value child_value
6.5 8.96
Choose new option
1
Print: parent_value child_value
6.5 6.43
Choose new option
1
Print: parent_value child_value
```

```
6.5 54.3
Choose new option
1
Print: parent_value child_value
54.3 23.1
Choose new option
3
Your tree:
1.40
 2.00
   6.50
     8.96
     6.43
     54.30
       23.10
   1.00
Choose new option
1
Print: parent_value child_value
2 3.2
Choose new option
1
Print: parent_value chil  d_value
3.2 3.3
Choose new option
3
Your tree:
1.40
 2.00
   6.50
     8.96
     6.43
     54.30
       23.10
   1.00
   3.20
     3.30
Choose new option
1
Print: parent_value child_value
1 5.32
Choose new option
1
Print: parent_value child_value
8.96 23.67
Choose new option
1
Print: parent_value child_value
```

```
8.96 232
Choose new option
3
Your tree:
1.40
 2.00
   6.50
     8.96
       23.67
       232.00
     6.43
     54.30
       23.10
   1.00
     5.32
   3.20
     3.30
Choose new option
4
Answer is 1.
Choose new option:
2
Print node for deleting:
3.2
Choose new option
3
Your tree:
1.40
 2.00
   6.50
     8.96
       23.67
       232.00
     6.43
     54.30
       23.10
   1.00
     5.32
Choose new option
4
Answer is 2.
Choose new option:
2
Print node for deleting:
8.96
Choose new option
3
Your tree:
```

```
1.40
 2.00
   6.50
     6.43
     54.30
       23.10
   1.00
     5.32
Choose new option
2
Print node for deleting:
1
Choose new option
3
Your tree:
1.40
 2.00
   6.50
     6.43
     54.30
       23.10
Choose new option
4
Answer is 0.
Choose new option:
2
Print node for deleting:
2
Choose new option
3
Your tree:
1.40
Choose new option
0
[1] + Done                       "/usr/bin/gdb" --interpreter=mi
--tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-ba3vqbek.tip"
1>"/tmp/Microsoft-MIEngine-Out-mp1y0cto.2wg"

////////////////////////////////////////////////////////////////////////////
Give value of root
3
Choose option: 1 - add new node; 2 - delete node; 3 - print tree; 4 - get
answer; 0 - exit
1
Print: parent_value child_value
3 3.2
Choose new option
1
```

```
Print: parent_value child_value

3 12.4
Choose new option
1
Print: parent_value child_value
3 14
Choose new option
3
Your tree:
3.00
 3.20
 12.40
 14.00
Choose new option
4
Answer is 1.
Choose new option:
2
Print node for deleting:
14
Choose new option
3
Your tree:
3.00
 3.20
 12.40
Choose new option
2
Print node for deleting:
3.2
Choose new option
33
Choose option from 1 to 4
3
Your tree:
3.00
 12.40
Choose new option
2
Print node for deleting:
12.4
Choose new option
3
Your tree:
3.00
Choose new option
4
```

```
Answer is 0.
Choose new option:
1
Print: parent_value child_value
3 1
Choose new option
3
Your tree:
3.00
 1.00
Choose new option
1
Print: parent_value child_value
1 4.532
Choose new option
3
Your tree:
3.00
 1.00
   4.53
Choose new option
4
Answer is 1.
Choose new option:
1
Print: parent_value child_value
1 43
Choose new option
3
Your tree:
3.00
 1.00
   4.53
   43.00
Choose new option
2
Print node for deleting:
1
Choose new option
3
Your tree:
3.00
Choose new option
0
[1] + Done                       "/usr/bin/gdb" --interpreter=mi
--tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-ba3vqbek.tip"
1>"/tmp/Microsoft-MIEngine-Out-mp1y0cto.2wg"
```

```
////////////////////////////////////////////////////////////////////////////////

Give value of root
4
Choose option: 1 - add new node; 2 - delete node; 3 - print tree; 4 - get
answer; 0 - exit
1
Print: parent_value child_value
4 2
Choose new option
1
Print: parent_value child_value
4 3
Choose new option
1
Print: parent_value child_value
4 3.2
Choose new option
1
Print: parent_value child_value
4 2.8
Choose new option
3
Your tree:
4.00
 2.00
 3.00
 3.20
 2.80
Choose new option
4
Answer is 1.
Choose new option:
1
Print: parent_value child_value
2 5.45
Choose new option
1
Print: parent_value child_value
2 23
Choose new option
3
Your tree:
4.00
 2.00
   5.45
   23.00
 3.00
```

```
 3.20
 2.80
Choose new option
4
Answer is 2.
Choose new option:
1
Print: parent_value child_value
3 2.44
Choose new option
2
Print node for deleting:
2
Choose new option

3
Your tree:
4.00
 3.00
   2.44
 3.20
 2.80
Choose new option
4
Answer is 0.
Choose new option:
1
Print: parent_value child_value
3 12
Choose new option
1
Print: parent_value child_value
3 23.42
Choose new option
3
Your tree:
4.00
 3.00
   2.44
   12.00
   23.42
 3.20
 2.80
Choose new option
4
Answer is 1.
Choose new option:
1
```

```
Print: parent_value child_value
4 22
Choose new option
3
Your tree:
4.00
 3.00
   2.44
   12.00
   23.42
 3.20
 2.80
 22.00
Choose new option
4
Answer is 2.
Choose new option:
1
Print: parent_value child_value
2.44
9.53
Choose new option
1
Print: parent_value child_value
9.53 54
Choose new option
1
Print: parent_value child_value
54 2.22
Choose new option
1
Print: parent_value child_value
2.22 532
Choose new option
3
Your tree:
4.00
 3.00
   2.44
     9.53
       54.00
         2.22
           532.00
   12.00
   23.42
 3.20
 2.80
 22.00
```

```
Choose new option
4
Answer is 2.
Choose new option:
2
Print node for deleting:
2.44
Choose new option
3
Your tree:
4.00
 3.00
   12.00
   23.42
 3.20
 2.80
 22.00
Choose new option
4
Answer is 1.
Choose new option:
2
Print node for deleting:
3
Choose new option
3
Your tree:
4.00
 3.20
 2.80
 22.00
Choose new option
4
Answer is 0.
Choose new option:
0
[1] + Done                       "/usr/bin/gdb" --interpreter=mi
--tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-dhbfezkg.afb"
1>"/tmp/Microsoft-MIEngine-Out-3za44gf5.e2u"

////////////////////////////////////////////////////////////////////////////
```

9. Дневник отладки

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

10. Замечания автора:
По существу работы: замечания отсутствуют.

11. Выводы: научился работать с деревьями общего вида в Си.

Подпись студента _____