Отчет по лабораторной работе № 23 по курсу Практикум на ЭВМ

Студент группы М8О-110Б-21, № по списку 2 Агеева Алиса

Контакты e-mail: alisa.ageewa2003@gmail.com

Работа выполнена: « » апреля 2022г.

Преподаватель: Доцент каф. 806 Никулин Сергей Петрович

Отчет сдан « » \_\_\_\_\_\_\_\_\_20 \_\_\_ г., итоговая оценка:

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

1. Цель работы: Составить программу на языке Си для построения и обработки дерева общего вида, содержащего узлы типа float
2. Задание (вариант № 5): Найти нетерминальную вершину с самой большой глубиной

Оборудование ПЭВМ студента, если использовалось:

Процессор Intel(R) Core(TM) i5-3450 CPU @ 3.10GHz 3.10 GHz с ОП 56 Гб, НМД 100 Гб. Монитор S23C350 1920x1080

Другие устройства:

5.

Программное обеспечение ЭВМ студента, если использовалось:

Операционная система семейства Windows 10

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.

Теперь опишем используемые в программе функции:

node \*create\_node(float f, node \*par) -Функция создания узла дерева. Создает указатель на узел дерева, помещает в него значение и родителя, а затем возвращает указатель на созданный узел

Tree \*create\_tree(float f) -Функция создания дерева. Создает указатель на дерево, затем создает корень дерева с помощью функции create . Возвращает указатель на созданное дерево.

node \*search\_tree(node \*t, float f) - Функция поиска узла по значению. Работает путем поиска в глубину и возвращает первый найденный узел с значением f

void add\_node\_in\_tree(Tree \*tree, float par\_f, float f) -Функция добавления узла дерева. Для создания узла ищется родитель узла с помощью функции search\_tree. Затем создается узел с помощью функции create\_node. Если у родителя нет сына, то создаваемый узел становится сыном родителя. В противном случае становится самым младшим братом старшего сына родителя.

void delete\_node(Tree \*tree, float f) - Функция удаления узла дерева. С помощью функции search\_tree находится удаляемый узел. Старшим сыном родителя выбранного узла становится следующий брат выбранного узла (если братьев нет, то, как и положено, указатель на сына становится NULL). Указатель на выбранный узел освобождается.

void print\_tree(node \*t, int x) - Функция печати узла дерева. Печать осуществляется в порядке обхода КЛП. Братья находятся слева от родителя на одной вертикальной линии.

int max\_level(node \*t, int deep) – Функция возвращает максимальный уровень, который может быть достигнут из заданной вершины

void noad\_search(node \*t, int start, int n) – сравнивает на каком сейчас уровне обхода и на предпоследнем выводит значение нетерминальной вершины (start = 0, int deep = max\_level(t, 0);

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 (проверка монотонности возрастания ширины уровня дерева)

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].

8. Распечатка протокола

Подклеить листинг окончательного варианта программы с тестовыми примерами,

подписанный преподавателем

**Тест 1**

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

1

Напишите корень дерева

1

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

2

Напишите значение родителя

1

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

3

Напишите значение родителя

1

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

99

Напишите значение родителя

2

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

10

Напишите значение родителя

2

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

7

Напишите значение родителя

99

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

5

1.00

2.00

99.00

7.00

10.00

3.00

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

3

Напишите значение дерева

99

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

5

1.00

2.00

10.00

3.00

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

99

Напишите значение родителя

2

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

7

Напишите значение родителя

99

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

8

Напишите значение родителя

99

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

9

Напишите значение родителя

8

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

13

Напишите значение родителя

3

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

14

Напишите значение родителя

13

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

5

Напишите значение родителя

13

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

11

Напишите значение родителя

14

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

180

Напишите значение родителя

11

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

5

1.00

2.00

10.00

99.00

7.00

8.00

9.00

3.00

13.00

14.00

11.00

180.00

5.00

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

4

Answer: 11 .00

**Тест 2.**

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

1

Напишите корень дерева

1

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

7

Напишите значение родителя

1

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

10

Напишите значение родителя

7

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

12

Напишите значение родителя

7

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

6

Напишите значение родителя

7

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

7

Напишите значение родителя

10

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

9

Напишите значение родителя

1

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

5

1.00

7.00

10.00

7.00

12.00

6.00

9.00

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

4

Answer: 10 .00

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

13

Напишите значение родителя

12

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

14

Напишите значение родителя

12

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

16

Напишите значение родителя

14

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

33

Напишите значение родителя

9

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

2

Напишите значение дерева

88

Напишите значение родителя

33

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

5

1.00

7.00

10.00

7.00

12.00

13.00

14.00

16.00

6.00

9.00

33.00

88.00

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

4

Answer: 14 .00

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

3

Напишите значение дерева

10

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

5

1.00

7.00

12.00

13.00

14.00

16.00

6.00

9.00

33.00

88.00

1. Create tree 2. Add node to tree 3. Delete node from tree 4. Zadacha 5. Print tree 6. Exit

6

Код программы:

#include <stdio.h>

#include <stdlib.h>

#define DEEP 0

#define max(x,y) ((x) > (y) ? (x) : (y))

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 Родитель не содержится в дереве\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 Вершина не содержится в дереве\n", f);

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 max\_level(node \*t, int deep){

if(!t) return deep - 1;

return max(max\_level(t->son, deep + 1), max\_level(t->brother, deep));

}

void noad\_search(node \*t, int start, int n){

if ((start == n) && (t->son!=NULL)){

printf("%-3.f. \n", t->value);

}

else{

if(t->son){

noad\_search(t->son, start + 1, n);

}

if(t->brother){

noad\_search(t->brother, start, n);

}

}

}

int main() {

Tree \*t = NULL;

int a, g = 1;

while (g) {

printf("1. Create tree\t 2. Add node to tree\t 3. Delete node from tree\t 4. Zadacha\t 5. Print tree\t 6. Exit \n");

scanf("%d", &a);

switch (a) {

case 1: {

printf("Напишите корень дерева\n");

float f;

scanf("%f", &f);

t = create\_tree(f);

break;

}

case 2: {

printf("Напишите значение дерева\n");

float f, par\_f;

scanf("%f", &f);

printf("Напишите значение родителя\n");

scanf("%f", &par\_f);

add\_node\_in\_tree(t, par\_f, f);

break;

}

case 3: {

printf("Напишите значение дерева\n");

float f;

scanf("%f", &f);

delete\_node(t, f);

break;

}

case 4: {

int tree\_deep = max\_level(t->root, 0);

printf("Answer: ");

noad\_search(t->root, 1, tree\_deep);

break;

}

case 5: {

print\_tree(t->root, 0);

break;

}

case 6: {

g = 0;

break;

}

default: {

printf("Введите нужную цифру\n");

}

}

}

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. -

10.

Замечания автора по существу:

11.

Выводы: Я научилась работать с динамическими структурами данных на Си и обрабатывать деревья

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_