**Отчет по лабораторной работе №23** по курсу фундаментальная информатика

Студентка группы М8О-113Б-22, № по списку 1, Астахова Анастасия Сергеевна

Контакты astakhovaanastasia0201@gmail.com

Работа выполнена: « » \_\_\_\_\_\_\_\_ 2023 г.

Преподаватель: доцент каф. 806 Никулин Сергей Петрович

Входной контроль знаний с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отчет сдан « » \_\_\_\_\_\_\_\_\_2023 г., итоговая оценка \_\_\_\_\_

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

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

1. **Цель работы:** Составить программу на языке Си для построения и обработки двоичного дерева, с выполнением следующих функций: добавление нового узла, текстовая визуализация дерева, удаление узла, вычисление функции, заданной в варианте.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Задание (номер варианта 4):** Определить значение листа двоичного дерева, имеющего минимальную глубину.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. **Оборудование** (лабораторное):

ЭВМ \_\_\_\_\_\_\_\_\_, процессор \_\_\_\_ \_\_, имя узла сети \_\_\_\_\_\_ с ОП \_\_\_\_\_\_\_\_ Мб, НМД \_\_\_\_\_\_\_ Мб. Терминал \_\_\_\_\_ адрес \_\_\_\_\_\_\_\_\_\_. Принтер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Другие устройства \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Оборудование ПЭВМ студента, если использовалось:*

Ноутбук: Процессор Intel Core i5-10210U с ОП 16 ГБ, SSD 512 ГБ. Другие устройства: мышь Bloody R80

*Компьютер: Процессор Ryzen 5 3600 с ОП 16 ГБ, SSD 512 ГБ. Другие устройства: мышь* Razer BASILISK X HYPERSPEED, [*Logitech G910 (920-008019) Orion Spectrum RGB Mechanical Gaming*](https://www.komus.ru/katalog/tekhnika/kompyuternaya-tekhnika/klaviatury-i-myshi/klaviatury-provodnye/klaviatura-logitech-g910-920-008019-orion-spectrum-rgb-mechanical-gaming/p/659916/?from=n-13776), монитор [*LG UltraGear 27GP750*](https://www.lg.com/ru/monitors/lg-27gp750-b)

1. **Программное обеспечение (лабораторное):**

Операционная система семейства \_\_\_Unix\_\_\_\_, наименование \_\_\_\_\_Ubuntu\_\_\_\_\_\_\_\_\_\_\_\_ версия \_\_4.15.0\_\_\_\_\_\_

интерпретатор команд \_\_\_\_bash\_\_\_\_\_\_\_\_ версия \_\_\_4.4.20\_\_\_\_

Система программирования \_\_С\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_\_\_\_\_\_\_последняя\_\_\_\_\_\_\_\_\_

Редактор текстов \_\_\_gedit\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_25.2.2 \_\_\_\_\_\_\_\_\_

Утилиты операционной системы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Прикладные системы и программы \_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Местонахождение и имена файлов программ и данных\_\_stud/208104 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**5.** *Программное обеспечение ПЭВМ студента, если использовалось****:***

Ноутбук: Операционная система семейства Ubuntu, наименование версия Ubuntu 20.04.3 LTS,

*интерпретатор команд bash версия 5.0.17. Система программирования C. Редактор текстов gedit версия 28.2. Утилиты: стандартные утилиты UNIX. Программное обеспечение: стандартное программное обеспечение системы UNIX.*

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

Программа будет содержать несколько функций:

1. Инициализация дерева
2. Добавление элемента
3. Распечатка дерева
4. Поиск элемента в дереве
5. Удаление элемента
6. Функция поиска значения листа с минимальной глубиной
7. **Сценарий выполнения работы** [план работы,первоначальный текст программы в черновике(можно на отдельном листе)итесты либо соображения по тестированию].
8. Изучу принцип работы упорядоченного двоичного дерева
9. Придумаю принцип работы функции, заданной в варианте.

Тесты:

PS D:\c\_labs\lab\_23> ./main

Введите одну из комманд:

Инициализация дерева: 1 n

Добавление нового узла: 2 n

Текстовая визуализация дерева: 3

Удаление узла: 4 n

Определить значение листа дерева, имеющего минимальную глубину: 5

Выход из программы: 0

1 4

2 1

2 2

2 3

2 5

2 6

2 7

4 2

4 3

3

7

6

5

4

1

5

1

2 2

3

7

6

5

4

2

1

2 3

3

7

6

5

4

3

2

1

2 0

3

7

6

5

4

3

2

1

0

5

0

0

PS D:\c\_labs\lab\_23> ./main

Введите одну из комманд:

Инициализация дерева: 1 n

Добавление нового узла: 2 n

Текстовая визуализация дерева: 3

Удаление узла: 4 n

Определить значение листа дерева, имеющего минимальную глубину: 5

Выход из программы: 0

1 4

2 2

2 1

2 3

2 6

2 5

2 7

3

7

6

5

4

3

2

1

4 7

4 5

3

6

4

3

2

1

5

6

0

*Пункты 1-7 отчета составляются сторого до начала лабораторной работы.*

*Допущен к выполнению работы.* **Подпись преподавателя****\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. **Распечатка протокола** (подклеить листинг окончательного варианта программы с тестовыми примерами,подписанныйпреподавателем).

Код лабораторной работы:

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

typedef struct tree {

int key;

int lvl;

struct tree \*left;

struct tree \*right;

struct tree \*parent;

} node;

node \*init(node \*root, int key) {

node \*tmp;

tmp = (node\*)malloc(sizeof(node));

tmp->key = key;

tmp->left = NULL;

tmp->right = NULL;

tmp->parent = NULL;

tmp->lvl = 0;

root = tmp;

return(root);

}

void \*add(node \*root, int key) {

node \*elem;

elem = (node\*)malloc(sizeof(node));

elem->left = NULL;

elem->right = NULL;

elem->key = key;

int tmp\_key = root->key;

node \*next\_node;

node \*par;

if (tmp\_key > elem->key) {

par = root;

next\_node = root->left;

}

else if (tmp\_key < elem->key){

par = root;

next\_node = root->right;

}

else {

printf("Такой элемент уже есть в дереве");

free(elem);

}

while (next\_node != NULL) {

if (next\_node->key > elem->key) {

par = next\_node;

next\_node = next\_node->left;

}

else if (next\_node->key < elem->key) {

par = next\_node;

next\_node = next\_node->right;

}

else {

printf("Такой элемент уже есть в дереве");

free(elem);

}

}

next\_node = elem;

next\_node->parent = par;

next\_node->lvl = -1;

if (par->key > next\_node->key) {

par->left = next\_node;

}

else {

par->right = next\_node;

}

}

void print\_tree(node\* root, int n) {

if (root != NULL) {

print\_tree(root->right, n + 1);

for (int i = 0; i < n; i++) printf("\t");

printf("%d\n", root->key);

print\_tree(root->left, n + 1);

}

}

node \*find(node \*root, int n) {

node \*tmp;

tmp = root;

while (1) {

if (tmp == NULL) {

printf("Такого элемента в узле нет\n");

break;

}

if (tmp->key > n) {

tmp = tmp->left;

}

else if (tmp->key < n) {

tmp = tmp->right;

}

else {

return tmp;

}

}

}

node \*succ(node \*root, int n) {

node \*tmp;

tmp = root->right;

while (tmp->left != NULL) {

tmp = tmp->left;

}

return(tmp);

}

void delete(node \*root, int key) {

node \*l = find(root, key);

node \*par;

node \*new\_nd;

if ((l->left) == NULL && (l->right) == NULL) {

par = l->parent;

if (l == par->right) par->right = NULL;

else par->left = NULL;

free(l);

}

else if ((l->left) != NULL && (l->right) == NULL) {

par = l->parent;

if (l == par->right) {

par->right = l->left;

par->right->parent = par;

}

else {

par->left = l->left;

par->left->parent = par;

}

free(l);

}

else if ((l->left) == NULL && (l->right) != NULL) {

par = l->parent;

if (l == par->right) {

par->right = l->right;

par->right->parent = par;

}

else {

par->left = l->right;

par->left->parent = par;

}

free(l);

}

else if ((l->left) != NULL && (l->right) != NULL) {

new\_nd = succ(l, key);

l->key = new\_nd->key;

if (new\_nd->right == NULL) {

if (new\_nd->parent->left == new\_nd) {

new\_nd->parent->left = NULL;

}

else {

new\_nd->parent->right = NULL;

}

}

else {

if (new\_nd->parent->left == new\_nd) {

new\_nd->parent->left = new\_nd->right;

}

else {

new\_nd->parent->left = new\_nd->right;

}

}

free(new\_nd);

}

}

void custom(node \*root, int lvl\_prev, int \*min\_lvl, int \*ans) {

if (root->lvl == -1) {

root->lvl = lvl\_prev + 1;

}

if (root->left != NULL && root->right != NULL) {

custom(root->left, lvl\_prev+1, min\_lvl, ans);

custom(root->right, lvl\_prev+1, min\_lvl, ans);

}

else if (root->left != NULL && root->right == NULL) {

custom(root->left, lvl\_prev+1, min\_lvl, ans);

}

else if (root->left == NULL && root->right != NULL) {

custom(root->right, lvl\_prev+1, min\_lvl, ans);

}

else {

if (root->lvl < \*min\_lvl) {

\*min\_lvl = root->lvl;

\*ans = root->key;

}

}

}

void print\_interface() {

printf("Введите одну из комманд:\n");

printf("Инициализация дерева: 1 n\n");

printf("Добавление нового узла: 2 n\n");

printf("Текстовая визуализация дерева: 3\n");

printf("Удаление узла: 4 n\n");

printf("Определить значение листа дерева, имеющего минимальную глубину: 5\n");

printf("Выход из программы: 0\n");

}

int main() {

//FILE \*f;

//f = fopen("input.txt", "r");

print\_interface();

while (true) {

int c;

node \*root;

//fscanf(f, "%d", &c);

scanf("%d", &c);

if (c == 1) {

int n;

//fscanf(f, "%d", &n);

scanf("%d", &n);

root = init(root, n);

}

else if (c == 2) {

int n;

//fscanf(f, "%d", &n);

scanf("%d", &n);

add(root, n);

}

else if (c == 3) {

print\_tree(root, 0);

}

else if (c == 4) {

int n;

//fscanf(f, "%d", &n);

scanf("%d", &n);

delete(root, n);

}

else if (c == 5) {

int min\_lvl = 1e9;

int ans = root->key;

int \*m\_l = &min\_lvl;

int \*an = &ans;

custom(root, 0, m\_l, an);

printf("%d\n", ans);

}

else if (c == 0) {

return 0;

}

}

}

1. **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. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  | или |  |  |  |  |  |
|  | дом. |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. **Замечания автора** по существу работы:

Замечания отсутствуют.

1. **Выводы**

Я научилась писать бинарное дерево и работать с ним.

Недочёты при выполнении задания могут быть устранены следующим образом:

Недочеты отсутствуют.

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