Отчет по лабораторной работе № 23 по курсу Фундаментальная информатика

Студент группы М8О-204Б-22, Филиппов Фёдор Иванович, № по списку 18

Контакты: gooselinjk@yandex.ru

Работа выполнена: “28” сентября 2023 года

Преподаватель: Потенко М.А., каф.806

Входной контроль знаний с оценкой_____

Отчёт сдан “29” сентября 2023 года, ИО ____

Подпись преподавателя _____

1. Тема: Динамические структуры данных. Обработка деревьев

2. Цель работы: Составить программу на языке Си для построения и обработки дерева общего вида, содержащего узлы типа int. Основные функции работы с деревьями реализовать в виде универсальных процедур или функций

3. Задание *(вариант № 19)***:** Определить ширину двоичного дерева

4. Оборудование

ЭВМ — ноутбук HP, процессор — Ryzen 5500U, с ОП 16384 МБ и НМД 1048576 МБ, Терминал Windows Powershell (с возможностью переключения на UNIX)

5. Программное обеспечение

Операционная система семейства Windows, наименование Windows 11 Home, версия 22H2

Редактор текстов — Sublime Text

Утилиты операционной системы — терминал Windows Powershell

Прикладные системы и программы — Visual Studio Code, Visual Studio

6. Идея, метод, алгоритм решения задачи (в формах: словесной, псевдокода, графической или формальные с пред- и постусловиями)

Опишу, что происходит в предоставленных файлах tree.h, tree.c, и main.c, исходя из вашего описания алгоритма:

tree.h:

Включает в себя заголовочные директивы #ifndef TREE и #define TREE, которые предотвращают повторное включение файла.

Определяет структуру Tree, представляющую узел бинарного дерева.

Объявляет прототипы функций, которые будут использоваться для работы с бинарным деревом, такие как create_tree, add_node, delete_tree, min_node, print_tree, free_node, height, level_width, и tree_max_width.

Завершает заголовочный файл директивой #endif.

tree.c:

Включает в себя заголовок "tree.h" и "stdio.h" для доступа к библиотекам и функциям.

Реализует функции, объявленные в "tree.h", включая create_tree, add_node, delete_tree, min_node, print_tree, free_node, height, level_width, и tree_max_width. Эти функции выполняют соответствующие операции с бинарным деревом.

create_tree создает новый узел с заданным значением.

add_node добавляет новый узел в бинарное дерево, учитывая правило, что меньшие значения идут влево, а большие вправо.

delete_tree удаляет узел с заданным значением из бинарного дерева.

min_node находит узел с минимальным значением в дереве.

print_tree выводит дерево на экран в форматированном виде.

free_node освобождает память, выделенную для узлов дерева.

height рекурсивно находит высоту бинарного дерева.

level_width рекурсивно определяет ширину заданного уровня в дереве.

tree_max_width находит максимальную ширину бинарного дерева, используя функции height и level_width.

main.c:

Включает в себя заголовок "tree.h" и стандартные библиотеки "stdio.h" и "stdlib.h".

В main функции создается корень Tree* root, и запускается бесконечный цикл.

Пользователю предлагается выбрать действие (1 - добавление узла, 2 - удаление узла, 3 - вывод дерева, 4 - нахождение ширины, 5 - завершение программы).

В зависимости от выбора пользователя, вызываются соответствующие функции из tree.c, такие как add_node, delete_tree, print_tree, и tree_max_width, чтобы выполнить соответствующие действия с бинарным деревом.

Программа выполняется в диалоговом режиме, и пользователь может взаимодействовать с бинарным деревом, добавлять, удалять узлы и выводить его на экран.

Память, выделенная для узлов, освобождается перед завершением программы.

7. Сценарий выполнения работы (план работы, первоначальный текст программы в черновике и тесты, либо соображения по тестам)

Когда я начал разработку этой программы, первым шагом было определение структуры данных, которую я буду использовать для представления бинарного дерева. Основываясь на моем предварительном понимании задачи и описании алгоритма в тексте, я создал заголовочный файл tree.h, который содержал определение структуры Tree и прототипы всех функций, которые мне потребуются для работы с деревом.

Затем я приступил к реализации функций, начиная с функции create_tree, которая создает новый узел дерева. Далее, я реализовал функцию add_node, которая добавляет узел в дерево с учетом правил добавления (меньшие значения идут влево, большие - вправо). После этого, я реализовал функции min_node и delete_tree для поиска минимального узла в дереве и удаления узла с заданным значением соответственно.

Следующим шагом было реализовать функцию print_tree, чтобы можно было вывести дерево на экран в удобном формате, и free_node для освобождения памяти после завершения работы программы.

Когда все функции для работы с деревом были готовы, я перешел к созданию пользовательского интерфейса в файле main.c. Здесь я реализовал бесконечный цикл, в котором пользователь мог выбирать различные операции: добавление узла, удаление узла, вывод дерева на экран, нахождение ширины дерева и завершение программы. Я также внимательно обрабатывал ввод пользователя, чтобы предотвратить добавление узлов с одинаковыми значениями.

8. Распечатка протокола (подклеить листинг окончательного варианта программы с тестовыми примерами)

main.c

```c
                free_node(root);
                return 0;
                break;
            default:
                printf("INCORRECT INPUT\n");
                break;
        }
    }
    delete_tree(root, root->value);

    return 0;
}

    int value;
    int choice = 0;
    printf("1. Add node\n");
    printf("2. Delete node\n");
    printf("3. Print binary tree on the screen\n");
    printf("4. Find binary tree's width\n");
    printf("5. Exit\n");

    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("Enter the value of the attached node: ");
            scanf("%d", &value);
            root = add_node(root, value);
            break;
        case 2:
            printf("Enter the value of the node to delete: ");
            scanf("%d", &value);
            root = delete_tree(root, value);
            break;
        case 3:
            printf("Tree:\n");
            print_tree(root, 1);
            break;
        case 4:
            printf("Tree's width: %d\n", tree_max_width(root));
            break;
        case 5:
```

tree.c

```c
#include "tree.h"
#include "tree.h"
#include <stdio.h>

// Функция для создания нового узла
Tree* create_tree(int value) {
    Tree* new_tree = (Tree*)malloc(sizeof(Tree));
    new_tree->value = value;
    new_tree->left = NULL;
    new_tree->right = NULL;
    return new_tree;
}

// Функция для добавления нового узла: если элемент меньше корня, добавляем слева, если больше - справа.
Tree* add_node(Tree* root, int value) {
    if(root == NULL) {
        return create_tree(value);
    }
    if(value < root->value) {
```

tree.h

```c
#ifndef TREE
#define TREE

#include <stdlib.h>

typedef struct Tree {
    int value;
    struct Tree* left;
    struct Tree* right;
} Tree;

Tree* create_tree(int value);

Tree* add_node(Tree* root, int value);

Tree* delete_tree(Tree* root, int key);

Tree* min_node(Tree* root);

void print_tree(Tree* root, int n);

void free_node(Tree* root);

int height(Tree* node);

int level_width(Tree* tree, int level);

int tree_max_width(Tree* root);

#endif
```

```
1. Add node
2. Delete node
3. Print binary tree on the screen
4. Find binary tree's width
5. Exit
1
Enter the value of the attached node: 23
1. Add node
2. Delete node
3. Print binary tree on the screen
4. Find binary tree's width
5. Exit
1
Enter the value of the attached node: 3
1. Add node
2. Delete node
3. Print binary tree on the screen
4. Find binary tree's width
5. Exit
1
Enter the value of the attached node: 34
1. Add node
2. Delete node
3. Print binary tree on the screen
4. Find binary tree's width
5. Exit
1
Enter the value of the attached node: 5432
```

```
1. Add node
2. Delete node
3. Print binary tree on the screen
4. Find binary tree's width
5. Exit
3
Tree:
                        5432
                34
        23
                3
1. Add node
2. Delete node
3. Print binary tree on the screen
4. Find binary tree's width
5. Exit
4
Tree's width: 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.	Дата	Время	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

_____

_____

_____

_____

Недочеты при выполнении работы могут быть устранены следующим образом:_____

_____

11. Выводы: Важной частью работы стала разработка структуры дерева и реализация основных функций для работы с ним. Это позволило мне лучше понять принципы работы бинарных деревьев, включая добавление, удаление узлов, поиск минимального значения, определение ширины дерева и вывод его структуры на экран. Также я научился обрабатывать ввод пользователя и взаимодействовать с программой в диалоговом режиме.

Считаю, что данная лабораторная работа дала мне важный опыт в разработке программ на языке C и работе с бинарными деревьями. Этот опыт может быть полезным в дальнейшей работе, особенно если потребуется обработка и анализ данных, организованных в структуру дерева. Кроме того, навыки в обработке пользовательского ввода и интерактивной работы с программой также пригодятся при создании приложений с пользовательским интерфейсом.