**Отчет по лабораторной работе №23** по курсу Практикум программирования

Студент группы М8О-104Б-22 Тесля Данила Сергеевич, № по списку 16

Контакты www, e-mail, icq, skype tesla\_2002@mail.ru

Работа выполнена: «30» мая 2023г.

Преподаватель: \_\_\_\_\_\_ каф. 806 Потенко Максим Алексеевич

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

Отчет сдан «3» октября 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 деревьев, реализовать основные функции работы с деревьями.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

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

1. **Задание** (*вариант №* **31**)**:** Определить число нетерминальных вершин двоичного дерева.

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

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

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

1. **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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

*Оборудование ПЭВМ студента, если использовалось:*

Процессор \_\_Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz с ОП\_\_8192\_\_Мб, НМД 524 288 Мб. Монитор \_\_\_-\_\_\_\_\_\_\_\_\_

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

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

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

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

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

Система программирования \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

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

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

Местонахождение и имена файлов программ и данных\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства \_\_\_Unix\_\_\_\_, наименование \_\_Ubuntu\_\_\_\_\_\_\_\_ версия \_\_\_22.04.3 LTS\_\_\_\_\_\_

интерпретатор команд \_\_\_\_bash\_\_\_\_\_\_\_\_ версия \_\_\_\_\_\_.

Система программирования \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_версия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Редактор текстов \_\_\_\_\_Sublime\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия 25.2.2\_\_\_\_\_\_\_\_\_\_\_\_

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

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

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

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

Местонахождение и имена файлов программ и данных на домашнем компьютере \_\_\_\_\_\_/Documents\_\_\_\_\_\_\_\_

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

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

Перед выполнением работы необходимо изучить материалы по созданию и обработке структур данных – деревьев на языке Си, а также изучить основы модульного программирование на Си и автоматизацию сборки программ модульной структуры на Си с использованием утилиты make. Создадим программу из трех файлов, один из которых будет главным файлом программы, в нем будет функция main, во втором файле будет реализация всех необходимых функций программы, и третий – заголовочный файл с определениями структур и функций. В заголовочном файле определим структуру узла дерева как абстрактного типа данных, так как по условию задание дерево – двоичное, то структура каждого узла дерева будет содержать поля: указатель на своего родителя, на левого потомка, на правого потомка и значение самого узла. Далее в файле для реализации функций необходимо будет реализовать функции: создания корня дерева, обхода и поиска значения в дереве, добавления узла в дерево, удаления узла из дерева, функции печати узла дерева, печати всего дерева, удаления и очистки памяти, выделенной под дерево, и функцию для выполнения задания по варианту – подсчёту нетерминальных вершин дерева. Когда все эти функции будут реализованы – перейдем главному файлу программы – в функции main сделаем контекстное меню, где пользователь сможет произвести действия с деревом по выбору – добавления узла, удаления узла, печати дерева и подсчёт нетерминальных вершин дерева, сделать это можно например конструкцией switch case в цикле while, и считывать выбор пользователя, пока он не захочет завершить работу с деревом, и после этого вывести окончательное дерево и количество нетерминальных вершин в стандартный поток вывода.

1. **Сценарий выполнения работы** [план работы,первоначальный текст программы в черновике(можно на отдельном листе)итесты либо соображения по тестированию].

Создадим программу из трех файлов, один из которых будет главным файлом программы – 23main.c, в нем будет функция main, во втором файле tree.c будет реализация всех необходимых функций программы, и третий – заголовочный файл tree.h с определениями структур и функций. В заголовочном файле для начала воспользуемя директивами #ifndef \_TREE\_H / #define \_TREE\_H /#endif методом условной компиляции, для предотвращения повторной компиляции кода, подключим стандартные библиотеки языка Си. Далее в этом же файле введем тип данных typedef int TreeItem, для переменных, которые будут хранить значения узлов дерева, затем определим структуру узла дерева как структуры данных, - typedef struct \_tree\* Tree; - указатель на структуру типа \_tree, затем определим поля структуры узла дерева \_tree, так как по условию задание дерево – двоичное, то структура каждого узла дерева будет содержать поля: указатель на своего родителя parent – типа Tree, на левого потомка – left – типа Tree, на правого потомка – right – типа Tree, и значение самого узла – data – типа TreeItem. Также в заголовочном файле определим конструкцию enum для ее дальнейшего использования в опреторе switch в основном файле программы. Далее в файле для реализации функций необходимо будет реализовать функции. Создания корня дерева – tree-create, которая будет принимать введенное пользователем значение корня дерева, и возвращать указатель на созданный узел (корень) дерева (типа Tree), в теле функции выделяется память под структуру узла Tree, полю структуры data присваивается введенное пользователем значение, остальным полям присваиваются пустые указатели, так как ни потомков, ни родителя у данного узла пока что нет.

Обхода и поиска значения в дереве – search-in-tree, которая принимает указатель на корень дерева и значение узла, которое необходимо найти и возвращает указатель на найденный узел типа Tree, а если он не был найден – то пустой указатель, данная функция разбита в моей программе на 2 функции – search-right и search\_left, которые рекурсивно совершают обход правой и левой ветвей дерева соответственно, а в основной функции поиска search-in-tree происходит их вызов, сначала вызов поиска в левой ветви дерева, если узел не находится в ней, то вызывается функция поиска в правой ветви дерева.

Добавления узла в дерево – add\_node, принимающая указатель на узел, к которому нужно добавить новый, переменную содержащую введенную пользователем сторону (лево или право) – каким потомком станет новый узел – левым или правым, и значение нового узла, также введенного пользователем, в теле функции в операторе switch case в case 1: добавляется левый потомок (пользователь ввел 1 – левый потомок), производится проверка, не существует ли уже такой узел, и если не существует, то выделяется память под новый узел, полю его значения data присваивается переданное в функцию значение нового узла, полю parent присваивается указатель на узел, к которому добавляется новый, и затем полю левого потомка узла к которому добавляется узел присваивается указатель на новый добавленный узел, если узел уже существует на этом месте, выводится сообщение пользователю об этом и ничего не происходит, пользователь должен выбрать новое место для узла. Аналогично с добавлением узла вправо – case 2. Далее функция delete\_node

удаления узла из дерева – принимает указатель на узел, который пользователь хочет удалить, возвращает 1 если удаление узла произошло, а если был удален корень дерева и узлов больше не осталось, то возвращает -1, в теле функции есть отдельная ветвь оператора if для последнего случая, когда у удаляемого узла нет ни потомков ни родителя – это и есть корень и он единственный узел в дереве, затем ветви, когда у узла есть родитель, но нет потомков, ветви – где есть только один из потомков, и ветвь где у узла есть оба потомка, в каждой из ветвей, кроме ветви, где у узла есть родитель, но нет потомков, происходит следующее – через временную переменную temp меняются местами именно значения удаляемого узла со значением потомка (левого в случае если есть только левый или оба, и правого в случае если есть только правый), и затем происходит рекурсивный вызов уже для узла-потомка, так как теперь он содержит значение, которое нужно убрать из дерева, таким образом происходит перекидывание удаляемого значения узла вниз по уровням дерева, пока это значение не станет значением терминального узла дерева – в этом случае произойдет вход в условие отсутствия потомков, и очистка этого узла командой free, и выход из рекурсии функцией return 1; , все остальные узлы сдвинутся к началу дерева по правилу удаления узлов из двоичного дерева общего вида. Далее - функция печати узла дерева print\_node и печати всего дерева print\_tree, в функции печати дерева происходит сначала рекурсивный проход по правой ветви дерева, начиная с самого правого листа и вызов функции печати узла для каждого из пройденных узлов, затем выводится значение корня дерева и затем аналогично обход левой ветви дерева и печать. Далее функция count для выполнения задания по варианту – подсчёту нетерминальных вершин дерева – в ней происходит рекурсивный обход всех узлов дерева в глубину и пока у узлов есть потомки, ведется подсчёт. В конце функция удаления и очистки памяти, выделенной под дерево – delete\_tree – в ней происходит обход (рекурсивными вызовами) правой ветви, начиная с листьев, затем аналогично левой – и очистка памяти из под каждого узла.

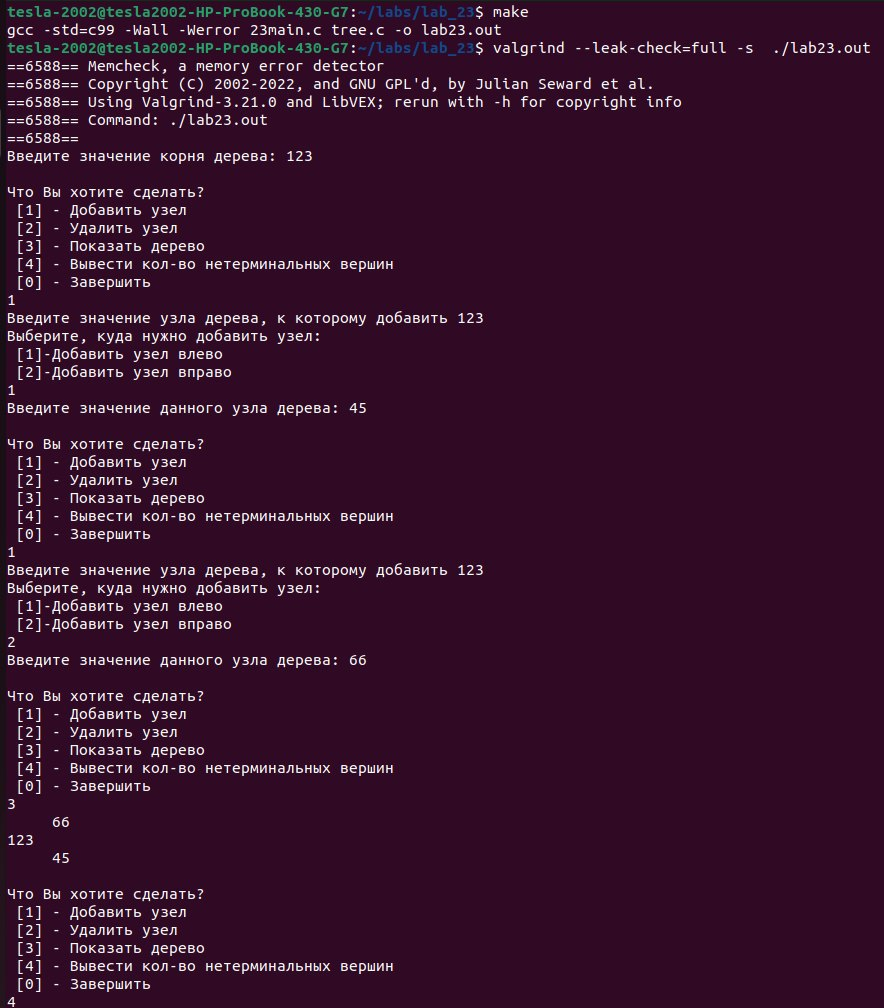
Переходим к главному файлу программы – в функции main сделаем контекстное меню, где пользователь сможет произвести действия с деревом по выбору – добавления узла, удаления узла, печати дерева и подсчёт нетерминальных вершин дерева, это реализовано конструкцией switch case в цикле while, с использованием конструкции enum. Считывается выбор пользователя, пока он не захочет завершить работу с деревом, и после этого выводится окончательное дерево и количество нетерминальных вершин в стандартный поток вывода.

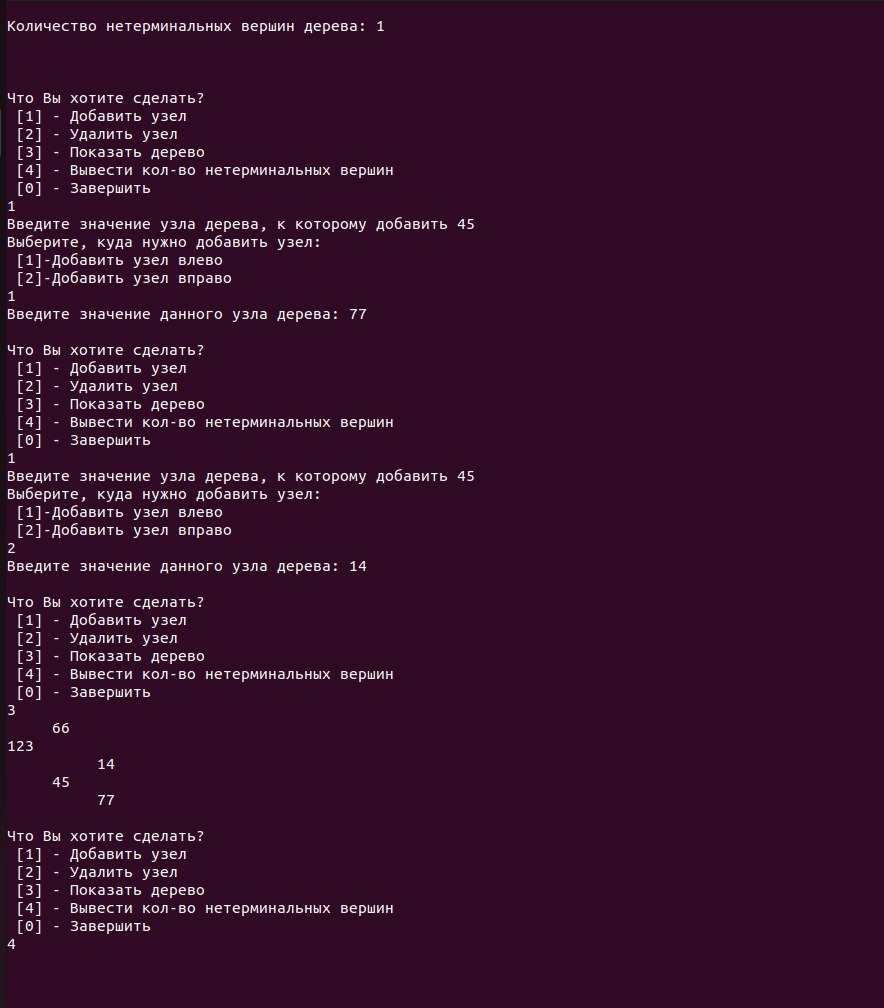
*Пункты 1-7 отчета составляются строго до начала лабораторной работы.*

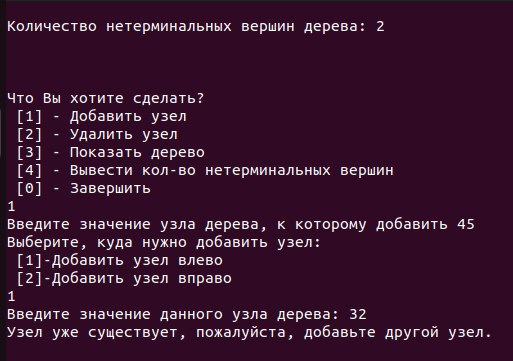
*Допущен к выполнению работы.* **Подпись преподавателя****\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

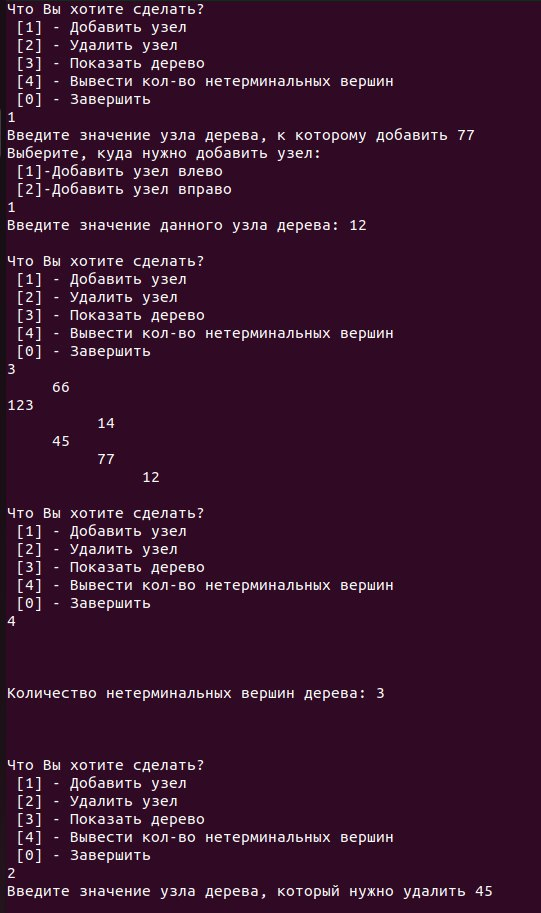
1. **Распечатка протокола** (подклеить листинг окончательного варианта программы с тестовыми примерами,подписанныйпреподавателем).

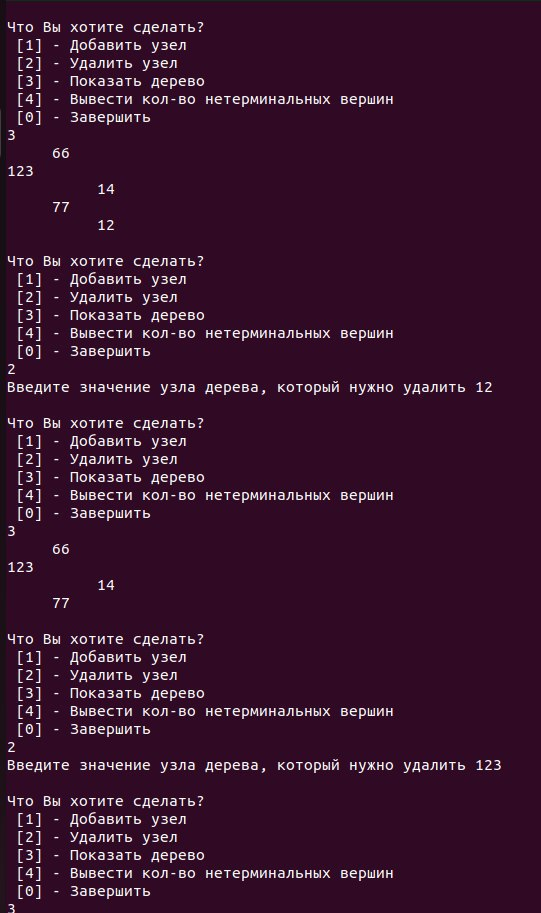
**Ссылка для просмотра кода программы:** https://github.com/tesla-2002/lab\_23

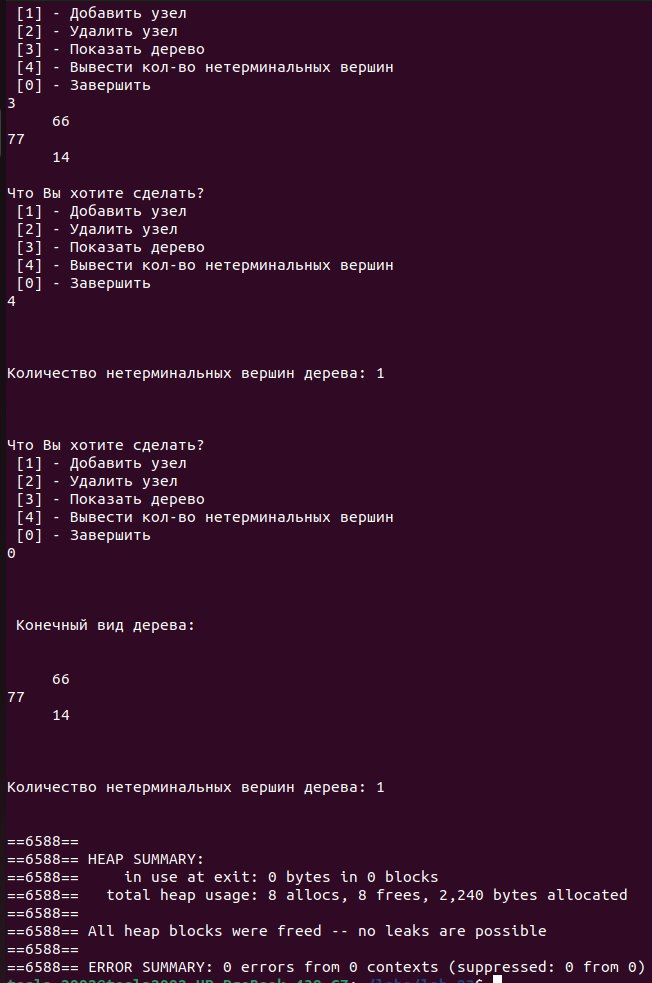












1. **Дневник отладки** должен содержать дату и время сеансов отладки, и основные события(ошибки в сценарии и программе,нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

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

1. **Замечания автора** по существу работы

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

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

1. **Выводы**

\_В ходе выполнения данной лабораторной работы я изучил и освоил на практике такую структуру данных на языке Си , как бинарное дерево, научился выполнять основные методы обработки дерева на Си .Научился реализовывать рекурсивные функции для обхода деревьев , вывода дерева и поиска в дереве узла с заданным значением. Все эти знания и умения пригодятся мне в дальнейшем для выполнения лабораторных и курсовых работ на языке Си. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Недочёты при выполнении задания могут быть устранены следующим образом: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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