

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №7.1**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

Тема. Нелинейные структуры

Балансировка дерева поиска

|  |  |
| --- | --- |
| Выполнил студент группы ИКБО-43-23 | Кощеев М. И. |
| Принял старший преподаватель | Рысин М.Л. |

Москва 2024

**СОДЕРЖАНИЕ**

[1 ЦЕЛЬ 3](#_Toc182313203)

[2 ХОД РАБОТЫ 4](#_Toc182313204)

[**4** **ВЫВОДЫ** 10](#_Toc182313205)

[**5** **ИСПОЛЬЗУЕМЫЙ ИСТОЧНИК** 11](#_Toc182313206)

# ЦЕЛЬ

Освоить приёмы реализации алгоритмов балансиров

# 2 ХОД РАБОТЫ

Составить программу создания двоичного дерева поиска и реализовать

процедуры для работы с деревом согласно варианту.

Процедуры оформить в виде самостоятельных режимов работы созданного

дерева. Выбор режимов производить с помощью пользовательского

(иерархического ниспадающего) меню.

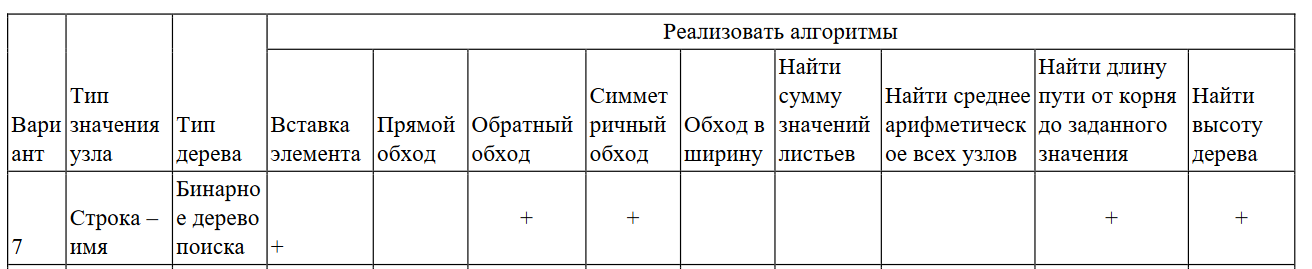
Провести полное тестирование программы на дереве размером n=10

элементов, сформированном вводом с клавиатуры. Тест-примеры определить

самостоятельно. Результаты тестирования в виде скриншотов экранов

включить в отчет по выполненной работе.

Вариант 7:



Алгоритм (menu)

1. Цикл: ввод с клавиатуры числа от 1 до 7

* 1 – вставить элемент
  + Ввод с клавиатуры имени
* 2 – Обратный обход
* 3 – Симметричный обход
* 4 – Нахождение длины пути от корня до заданного значения
  + Ввод имени
  + Если длинна не равна -1, вывод результат работы функции findPathLength
* 5 – Вывод высоты дерева
* 6 – Вывод дерева (данная функция не указана в задаче, на необходима только для тестирования)
* 7 – выход из цикла
* Другое значение – Вывод об ошибки

1. Удаление дерева

Код программы

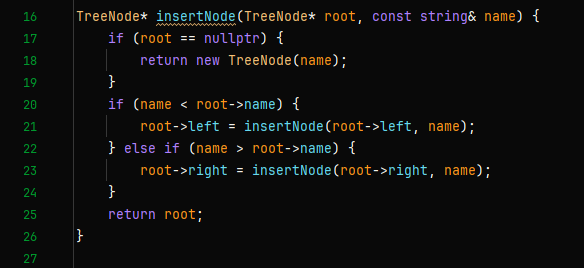




Алгоритм (insertNode):

1. Если дерево пустое, создается новый узел, который станет корнем
2. Если вставляемое значение меньше значения текущего узла, продолжаем искать место для вставки в левом поддереве
3. Если вставляемое значение больше, ищем в правом поддереве
4. Если значение уже существует, ничего не делаем, чтобы избежать дубликатов
5. Когда найдено подходящее место (свободный узел), создаем новый узел с этим значением

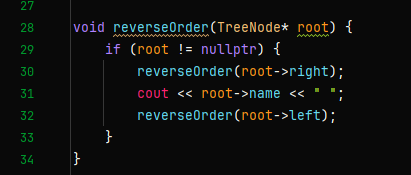
Код



Алгоритм (reverseOrder)

1. Если текущий узел не пустой:
   1. Сначала рекурсивно обходим правое поддерево
   2. Вывод значение текущего узла
   3. рекурсивно обходим левое поддерево

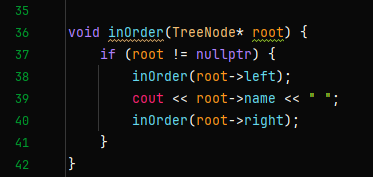
Код



Алгоритм (inOrder)

1. Если текущий узел не пустой:
   1. Сначала рекурсивно обходим левое поддерево
   2. Вывод значение текущего узла
   3. рекурсивно обходим правое поддерево

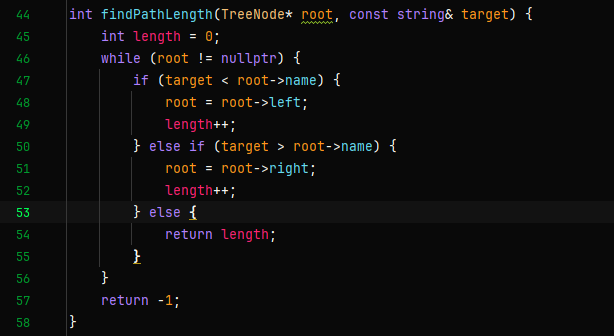
Код



Алгоритм (findPathLength)

1. Начинаем с длины пути, равной нулю
2. Пока текущий узел не пуст
   1. Если искомое значение меньше значения текущего узла, переходим в левое поддерево и увеличиваем длину пути
   2. Если искомое значение больше значения текущего узла, переходим в правое поддерево и снова увеличиваем длину пути
   3. Если значения совпадают, возвращаем текущую длину пути
3. Если прошли по дереву до конца и не нашли значение, возвращаем -1

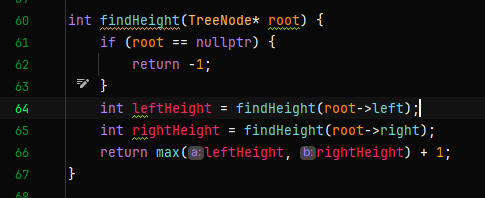
Код



Алгоритм (findHeight)

1. Если текущий узел пуст, возвращаем -1
2. Рекурсивно находим высоту левого поддерева
3. Рекурсивно находим высоту правого поддерева
4. Возвращаем максимальную высоту между левым и правым поддеревом, добавив 1, чтобы учесть текущий узел

Код



Тестирование (reverseOrder)

|  |  |  |
| --- | --- | --- |
| Заданные значения | Ожидаемый результат | Фактический результат |
| Cartman, Stan, Kyle, Kenny, Butters, Randy, Towelie, Mackey, Princess, Jesus | Towelie, Stan, Randy, Princess, Mackey, Kyle, Kenny, Jesus, Cartman, Butters | Towelie Stan Randy Princess Mackey Kyle Kenny Jesus Cartman Butters |



Тестирование (inOrder)

|  |  |  |
| --- | --- | --- |
| Заданные значения | Ожидаемый результат | Фактический результат |
| Cartman, Stan, Kyle, Kenny, Butters, Randy, Towelie, Mackey, Princess, Jesus | Butters, Cartman, Jesus, Kenny, Kyle, Mackey, Princess, Randy, Stan, Towelie | Butters Cartman Jesus Kenny Kyle Mackey Princess Randy Stan Towelie |

****

Тестирование (findPathLength)

|  |  |  |
| --- | --- | --- |
| Заданные значения | Ожидаемый результат | Фактический результат |
| Kenny | 3 | 3 |

****

Тестирование (findPathLength)

|  |  |  |
| --- | --- | --- |
| Заданные значения | Ожидаемый результат | Фактический результат |
| - | 5 | 5 |

****

1. **ВЫВОДЫ**

В результате проделанной работы было разработано двоичное дерево поиска, которое реализует все необходимые операции для работы с дерево

1. **ИСПОЛЬЗУЕМЫЙ ИСТОЧНИК**

1. М.Л. РЫСИН, М.В. САРТАКОВ, М.Б. ТУМАНОВА Учебно-методическое

пособие СиАОД часть 2