Практическая работа №6

**Бинарные деревья**

1. **Основные теоретические сведения**

Дерево (tree) - это совокупность узлов и отношений, образующих иерархическую структуру узлов. У А. Ахо дерево определяется следующим образом: а) один узел является деревом. Он же корень (root) данного дерева; б) пусть п - это узел, а Т1, Т2, Тк- деревья с корнями щ, п2, ..., пк соответственно. Можно построить новое дерево, сделав п родителем узлов щ, п2, ..., пк. В этом новом дереве п будет корнем, а Т1, Т2, Тк - поддеревьями (subtrees) данного корня. Узлы щ, п2, называют сыновьями узла п. Число поддеревьев отдельного узла называется его степенью (degree), узел с нулевой степенью - листом (leaf) или концевым узлом, максимальное значение степени всех узлов данного дерева - степенью дерева. Уровень (level) узла по отношению к дереву Т определим рекурсивно: • уровень корня равен нулю; • уровень любого другого узла на единицу выше, чем уровень корня ближайшего поддерева дерева Т, содержащего данный узел. Если в дереве между порожденными узлами, имеющими общий исходный, считается существенным их порядок, то дерево называется упорядоченным (ordered tree). В задачах поиска почти всегда рассматриваются упорядоченные деревья. Если не считать различными два дерева, которые отличаются только относительным порядком поддеревьев узлов, то дерево называется ориентированным (oriented tree). Упорядоченное дерево, степень которого не больше двух, называется бинарным деревом (Binary tree, ВТ). Бинарное дерево особенно часто используется при поиске в оперативной памяти. В таком случае имеют дело с бинарным деревом поиска (binary search tree, BST). Бинарное дерево называется сбалансированным (balanced), если высота левого поддерева каждого узла отличается от высоты правого поддерева не более чем на 1.

Именно бинарные деревья широко используются в программировании. Для хранения и обработки всех прочих деревьев можно использовать те же структуры и алгоритмы, что и для графов, поэтому они отдельно не рассматриваются.

**2. Реализация основных алгоритмов**

Необходимо написать модуль для работы с бинарными деревьями (задания см. ниже), и программу, демонстрирующую работу этого модуля. Программа должна быть выполнена либо в качестве консольного приложения (тогда обязателен командно-текстовый интерфейс), либо иметь графический интерфейс пользователя. Разобранные алгоритмы можно применять непосредственно, однако чаще всего они нуждаются в некоторых изменениях. При необходимости можно использовать вспомогательные функции. В разработанном модуле не должны применяться интерфейсные функции (исключение только для функций вывода дерева). В качестве данных, если не сказано иначе, брать последовательность случайных целых чисел (положительных и, возможно, отрицательных). Желательно также предусмотреть ввод элементов списка с клавиатуры. Для хранения бинарного дерева рекомендуется использовать следующую структуру: struct bintree (int inf; bintree \*left; bintree \*right; );

Каждый элемент дерева связан с двумя потомками, левым и правым. На рис. 2 приведен пример организации бинарного дерева (пустая связь отображается перечеркиванием поля).

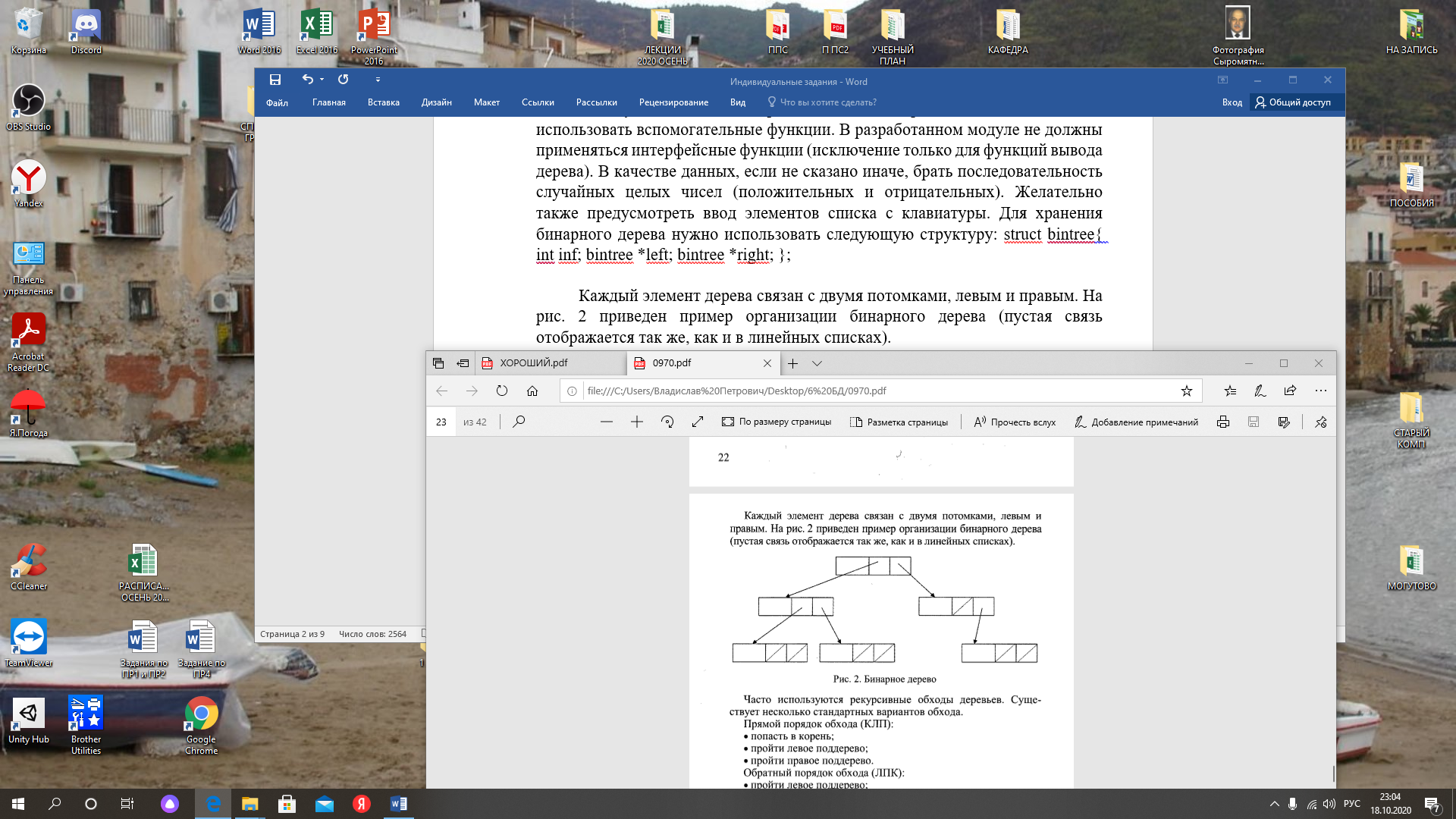


Рис. 2. Бинарное дерево

Часто используются рекурсивные обходы деревьев. Существует несколько стандартных вариантов обхода.

Прямой порядок обхода: • попасть в корень; • пройти левое поддерево; • пройти правое поддерево. Обратный порядок обхода: • пройти левое поддерево; • пройти правое поддерево; • попасть в корень. Центрированный (симметричный) порядок обхода: • пройти левое поддерево; • попасть в корень; • пройти правое поддерево.

От выбора порядка обхода, естественно, зависит порядок вывода узлов дерева, для большинства остальных алгоритмов порядок обхода не имеет особого значения. Рекомендуется реализовать 2-3 разных обхода и использовать их в различных алгоритмах. Для удобства разделим основные алгоритмы работы с деревом на следующие группы: создание дерева и организация интерфейса; алгоритмы работы с деревом; удаление элементов из дерева; сложные алгоритмы работы с деревом.

**Создание дерева и организация интерфейса**

1. Создание бинарного дерева поиска. Необходимо создать бинарное дерево из п элементов (п вводится с клавиатуры), элементы дерева задаются случайным образом и добавляются в дерево по следующему принципу: 1) первый элемент добавляем в корень; 2) если элемент меньше текущего и у текущего элемента нет левого потомка, делаем элемент этим потомком. Если левый потомок есть, то переходим к нему и повторяем пункты 2 и 3; 3) если элемент больше текущего и у текущего элемента нет правого потомка, делаем элемент этим потомком. Если правый потомок есть, то переходим к нему и повторяем пункты 2 и 3. Бинарное дерево поиска не должно содержать повторяющихся элементов, подумайте, как это можно реализовать.

2. Добавление узла в бинарное дерево поиска. Пользователь вводит число, необходимо добавить его в дерево, руководствуясь принципами построения бинарного дерева поиска.

3. Вывод бинарного дерева. Здесь необходимо использовать алгоритм обхода дерева. Принцип рекурсивного обхода дерева рассмотрен выше. Подумайте, какой порядок обхода следует использовать, чтобы вывести элементы дерева в порядке возрастания. Попробуйте реализовать «красивый» вывод дерева, т.е. должна выводиться схема (иерархия) организации дерева, где каждая вершина находится на своем уровне и можно проследить ее связи с родителем и потомками.

**Алгоритмы работы с деревом**

4. Поиск образца в бинарном дереве поиска. Необходимо использовать знания о принципе построения бинарного дерева поиска. Фактически алгоритм поиска будет аналогичен алгоритму бинарного поиска для отсортированного в порядке возрастания массива.

Сначала аргумент поиска сравнивается с ключом, находящимся в корне. Если аргумент совпадает с ключом, поиск закончен, если же не совпадает, то в случае, когда аргумент оказывается меньше ключа, поиск продолжается в левом поддереве, а в случае, когда больше ключа, - в правом поддереве. Увеличив уровень на 1, повторяют сравнение, считая текущий узел корнем. Если дальше идти некуда, аргумента поиска в дереве нет. На рис. 3 показан пример поиска по бинарному дереву поиска, содержащему текстовую информацию.

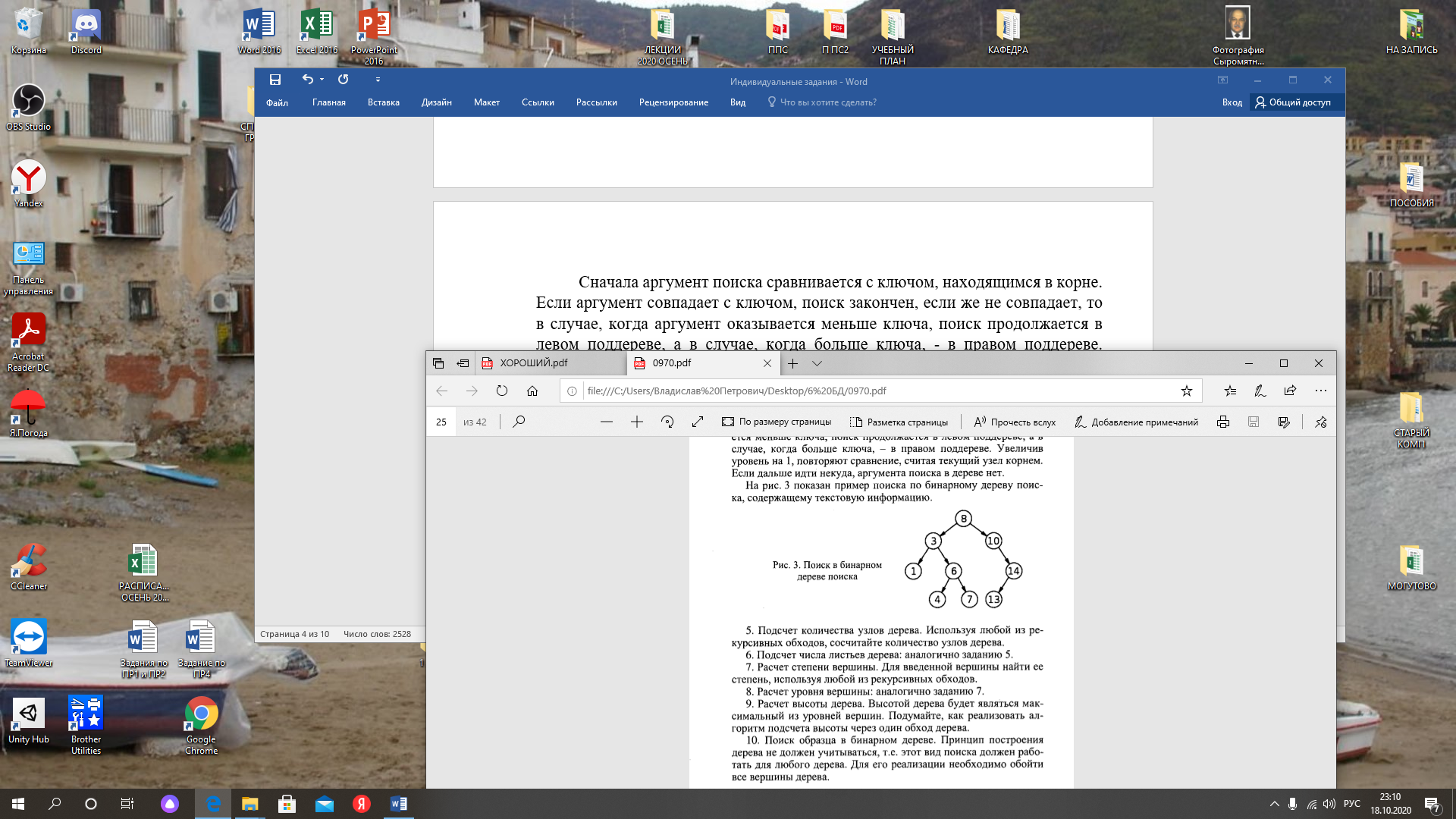


Рис. 3. Поиск в бинарном дереве поиска

5. Подсчет количества узлов дерева. Используя любой из рекурсивных обходов, сосчитайте количество узлов дерева.

6. Подсчет количества листьев дерева: аналогично заданию 5.

7. Расчет степени вершины. Для введенной вершины найти ее степень, используя любой из рекурсивных обходов.

8. Расчет уровня вершины: аналогично заданию 7.

9. Расчет высоты дерева. Высотой дерева будет являться максимальный из уровней вершин. Подумайте, как реализовать алгоритм подсчета высоты через один обход дерева.

10. Поиск образца в бинарном дереве. Принцип построения дерева не должен учитываться, т.е. этот вид поиска должен работать для любого дерева. Для его реализации необходимо обойти все вершины дерева.

**11. Удаление элементов из бинарного дерева**

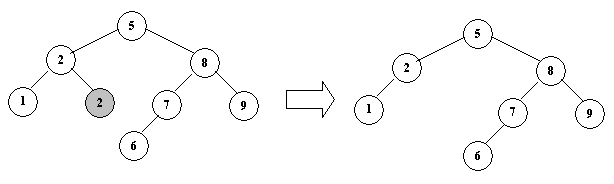
При удалении элемента из бинарного дерева возникает три возможных случая:

* удаление листа (элемента, не содержащего подчиненных элементов);
* удаление узла (элемента), содержащего одно поддерево;
* удаление узла (элемента), содержащего два поддерева.

***11.1 Удаление листа***

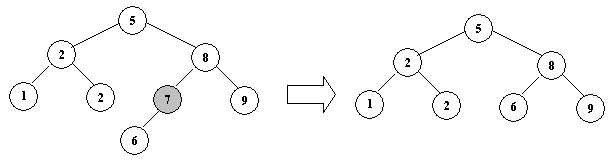
При удалении из бинарного дерева листа адрес на этот лист в соответствующем узле-предке необходимо заменить на константу ***NULL.***

Графически этот случай будет выглядеть следующим образом:



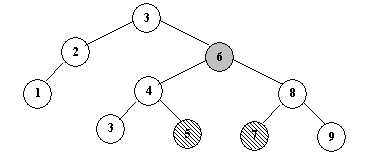
***11.2 Удаление узла, содержащего одно поддерево***

При удалении из бинарного дерева узла, содержащего одно поддерево, необходимо адрес на этот узел в соответствующем узле-предке заменить на адрес нижестоящего узла-потомка.

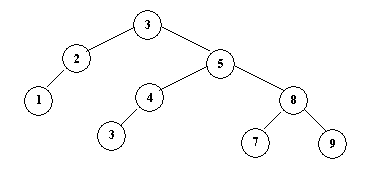


***11.3 Удаление узла, содержащего два поддерева***

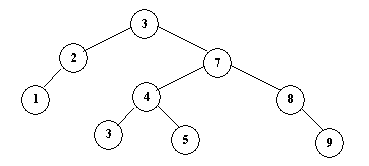
При удалении из бинарного дерева узла, содержащего два поддерева, необходимо найти подходящий элемент, который можно вставить на место удаляемого узла. Этот подходящий элемент, естественно, должен быть «легко» перемещаемым. Такой узел в бинарном дереве всегда существует: ***это либо самый правый лист левого поддерева*** (1-ый вариант замены), ***либо самый левый лист правого поддерева*** (2-ой вариант замены)***.***



С учетом сформулированного правила возможны два варианта удаления узла со значением ***6*** в приведенном бинарном дереве



1-ый вариант замены удаляемого узла



2-ой вариант замены удаляемого узла

*Примечание:*

* Выбор одного из вариантов замены удаляемого узла, содержащего два поддерева, должен быть оговорен до начала программирования.

**Сложные алгоритмы работы с деревом**

12. Балансировка дерева, т.е. приведение дерева к сбалансированному виду. Использование бинарного дерева часто бывает обоснованным, только когда это дерево сбалансировано. Алгоритм балансировки дерева довольно сложен для понимания и реализации, поэтому здесь не рассматривается. Студент может разобрать этот алгоритм самостоятельно, используя дополнительную литературу. Это задание не является обязательным для выполнения.

**Варианты индивидуальных заданий**

1. Составить программу, в которой необходимо построить дерево поиска по введенным целым числам.

Разработать функцию вывода дерева на экран монитора.

Определить:

* количество листьев в данном дереве,
* высоту правого поддерева.
* высоту левого поддерева..

Удалить заданный с клавиатуры:

* узел-лист,
* узел, содержащий одно поддерево,
* узел, содержащий два поддерева.

1. Составить программу, в которой необходимо построить дерево поиска по введенным целым числам. Минимальное значение элемента из каждого уровня заменить среднеарифметическим значением узлов из этого уровня.

Разработать функцию вывода дерева на экран монитора.

Удалить заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.

1. Составить программу, в которой необходимо построить дерево поиска по введенным целым числам. Поменять местами узел, содержащий максимальное число и узел, содержащий минимальное число.

Разработать функцию вывода дерева на экран монитора.

Удалить заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.

4. Составить программу, в которой необходимо построить дерево поиска по введенным целым числам. Определить количество узлов, имеющих одного потомка. После чего удалить все листья данного дерева.

Разработать функцию вывода дерева на экран монитора.

Удалить заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.

5. Составить программу, в которой необходимо из входной последовательности слов построить дерево и удалить слово, которое встречается чаще всего.

Разработать функцию вывода дерева на экран монитора.

Удалить заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.

6. Составить программу, в которой необходимо из входной последовательности слов построить дерево и удалить самое длинное слово.

Разработать функцию вывода дерева на экран монитора.

Удалить заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.

7. Составить программу, в которой необходимо сформировать двоичное дерево. Подсчитать:

* количество узлов-листьев,
* количество узлов, которые имеют одного потомка,
* количество узлов, которые имеют одного правого потомка,
* количество узлов, которые имеют одного левого потомка.

Разработать функцию вывода дерева на экран монитора.

Удалить заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.

8. Составить программу, в которой необходимо сформировать двоичное дерево. Определите, есть ли в данном бинарном дереве два одинаковых элемента (дерево не является бинарным деревом поиска).

Разработать функцию вывода дерева на экран монитора:

* Выведите номера уровней данного бинарного дерева, на которых имеются листья..
* Выведите номера вершин, у которых количество потомков в левом поддереве не равно количеству потомков в правом поддереве.
* Выведите номера вершин, для которых высота левого поддерева не равна высоте правого поддерева.
* Выведите номера вершин, у которых количество потомков в левом поддереве отличается от количества потомков в правом поддереве на 1.

9. Составить программу, в которой необходимо найти высоту дерева h и удалите из него (с перестройкой) все вершины на уровне h/2.

Разработать функцию вывода дерева на экран монитора.

Удалить заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.

10. Составить программу, в которой необходимо найти вершины, у которых высоты поддеревьев не равны, а количество потомков в правом и левом поддеревьях равны.

Разработать функцию вывода дерева на экран монитора.

Удалить заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.

11. Составить программу, в которой необходимо удалить все вершины, у которых высота левого поддерева отличается от высоты правого поддерева на 2.

Разработать функцию вывода дерева на экран монитора.

Удалить заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.

12. Составить программу, которая:

a) присваивает параметру Е запись из самого левого листа непустого дерева Т (лист-вершина, из которого не выходит ни одной ветви);

b) определяет число вхождений записи Е в дерево Т.

c) содержит функцию вывода дерева на экран монитора.

d) удаляет заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.

13. Вершины дерева вещественные числа. Составить программу, которая:

a) вычисляет [среднее арифметическое](https://intellect.icu/srednee-arifmeticheskoe-1793#term-srednee-arifmeticheskoe)всех вершин дерева;

b) добавляет в дерево вершину с вычисленным значением.

c) содержит функцию вывода дерева на экран монитора.

d) удаляет заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.

14. Записи вершин дерева - вещественные числа.

Разработать программу, которая удаляет все вершины с отрицательными записями.

Разработать функцию вывода дерева на экран монитора.

Удалить заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.

15. Записи вершин дерева - вещественные числа. Составить программу, которая:

a) находит максимальное и минимальное значение записей вершин непустого дерева;

b) печатает исходное дерево и записи из всех листьев дерева.

c) содержит функцию вывода дерева на экран монитора.

d) удаляет заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.

16. Составить программу, которая:

a) определяет, входит ли вершина с записью Е в дерево Т;

b) если такая запись не найдена, то она добавляется.

c) содержит функцию вывода дерева на экран монитора.

d) удаляет заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.

17. Составить программу, которая:

a) находит в непустом дереве Т длину (число ветвей) пути от корня до ближайшей вершины с записью Е; если Е не входит в Т, то за ответ принять -1.

b) определяет максимальную глубину непустого дерева Т, т.е. число ветвей в самом длинном из путей от корня дерева до листьев.

дерева).

c) содержит функцию вывода дерева на экран монитора.

d) удаляет заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.  
18. Составить программу, которая содержит функцию вывода дерева на экран монитора и:

1. печатает узлы непустого дерева при обходе слева направо;
2. удаляет все листья исходного дерева;
3. печатает модифицированное дерево.
4. удаляет заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.

19. Составить программу, содержит функцию вывода дерева на экран монитора и:

а) вставляет узел с записью Е в дерево, если ранее такой не было;

b) удаляет ее, если она уже существует.

с) удаляет заданный с клавиатуры:

• узел-лист,

• узел, содержащий одно поддерево,

• узел, содержащий два поддерева.