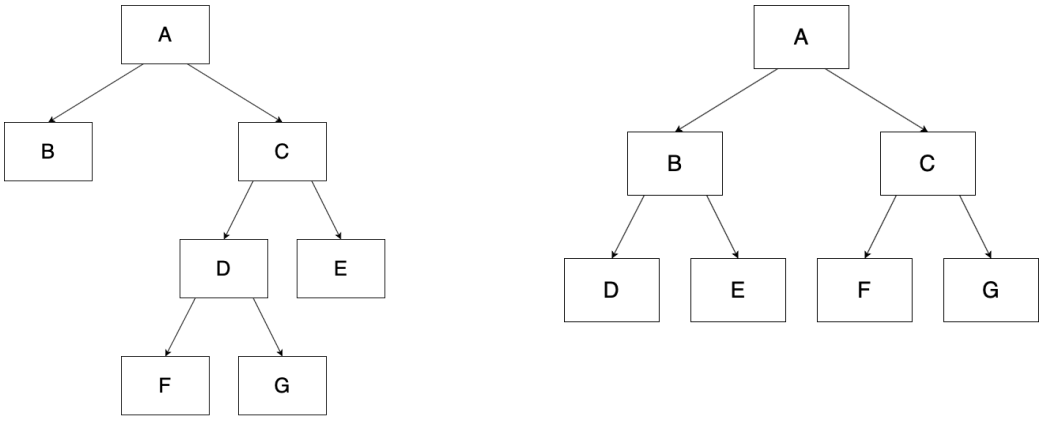
Лабораторная работа №7
Тема: Деревья.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Дерево - непустое конечное множество элементов, один из которых называется ***корнем***, а остальные делятся на несколько непересекающихся подмножеств, каждое из которых также является деревом. Одной из разновидностей деревьев являются **бинарные деревья**. Бинарное дерево имеет один корень и два непересекающихся подмножества, каждое из которых само является бинарным деревом. Эти подмножества называются **левым** и **правым поддеревьями** исходного дерева. Ниже, на рисунке 1, приведены графические изображения бинарных деревьев. Если один или более узлов дерева имеют более двух ссылок, то такое дерево не является бинарным.



а) б)

Рисунок 1 - Бинарные деревья

На рисунке 1,а A - корень дерева, B (C) - корень левого (правого) поддерева. Говорят, что A - родительский узел, B (C) - левый (правый) дочерний узел.

Узел, не имеющий дочерних: B, F, G и E - **лист**. Узел i - **предок** узла j (а j - потомок i), если i – либо родительский узел j, либо родительский узел некоторого предка j. Если каждый узел бинарного дерева, не являющийся листом, имеет непустые правые и левые поддеревья, то дерево называется **строго бинарным деревом (**рисунок 1,б). Строго бинарное дерево с n листами всегда содержит 2n-1 узлов. Уровень узла в бинарном дереве может быть определен следующим образом.

Корень дерева имеет **уровень** 0, уровень любого другого узла дерева на 1 больше уровня его отца.

Глубина бинарного дерева – максимальный уровень листа дерева (длина максимального пути от корня к листу).

Полное бинарное дерево уровня n - дерево, в котором каждый узел уровня n является листом, и каждый узел уровня меньше n имеет непустые левое и правое поддеревья.

Почти полное бинарное дерево - бинарное дерево, для которого существует неотрицательное целое k такое, что:

- каждый лист в дереве имеет уровень k или k+1;
- если узел дерева имеет правого потомка уровня k+1, то все его левые потомки, также листья, имеют уровень k+1.

Бинарные деревья с успехом могут быть использованы, например, при сравнении и организации хранения очередной порции входной информации с информацией, введенной ранее, и при этом в каждой точке сравнения может быть принято одно из двух возможных решений. Так как информация вводится в произвольном порядке, то нет возможности предварительно упорядочить ее и применить бинарный поиск. При использовании линейного поиска время пропорционально квадрату количества анализируемых слов. Каким же образом, не затрачивая большое количество времени, организовать эффективное хранение и обработку входной информации. Один из способов постоянно поддерживать упорядоченность имеющейся информации: это перестановка ее при каждом новом вводе информации, что требует существенных временных затрат. Построение бинарного дерева осуществляется на основе *лексикографического упорядочивания* входной информации. Различным элементам входной информации ставятся в соответствие различные узлы бинарного дерева, каждый из которых является структурой, содержащей:

- информационную часть;
- указатель на левый сыновний узел;
- указатель на правый сыновний узел.

Лексикографическое упорядочивание информации в бинарном дереве заключается в следующем. Считывается первая информация и помещается в узел который становится корнем бинарного дерева с пустыми левым и правым поддеревьями. Затем каждая вводимая порция информации сравнивается с информацией содержащейся в корне. Если значения совпадают, то, например, наращиваем число повторов и переходим к новому вводу информации. Если же введенная информация меньше значения в корне, то процесс повторяется для левого поддерева, иначе для правого. Так продолжается до тех пор, пока не встретится дубликат или пока не будет достигнуто пустое поддерево. В этом случае число помещается в новый узел данного места дерева.

Следующая операция **прохождение** бинарного дерева. Существуют три метода обхода бинарного дерева: в **прямом** порядке, в **симметричном** порядке и в **обратном** порядке.

Чтобы пройти непустое бинарное дерево в *прямом* порядке надо:

1. Попасть в корень.
2. Пройти в прямом порядке левое поддерево.
3. Пройти в прямом порядке правое поддерево.

4. Для прохождения в *симметричном* порядке:
5. Пройти в симметричном порядке левое поддерево.
6. Попасть в корень.
7. Пройти в симметричном порядке правое поддерево.

Для прохождения в *обратном* порядке:

1. Пройти в обратном порядке левое поддерево.
2. Пройти в обратном порядке правое поддерево.
3. Попасть в корень.

Ниже приведены некоторые функции, иллюстрирующие работу с бинарным деревом.

```
typedef struct Node {
  int value;
  struct Node *left;
  struct Node *right;
} Node;

Node *create_node(int value) {
  Node *new_node = (Node *)malloc(sizeof(Node));
  new_node->value = value;
  new_node->left = NULL;
  new_node->right = NULL;
  return new_node;
}

Node *find_parent(Node *root, int value) {
  Node *current = root;
  Node *previous = NULL;
  while (current != NULL) {
    previous = current;
    if (value < current->value) {
      current = current->left;
    }
    else {
      current = current->right;
    }
  }
  return previous;
}

// Добавление с учетом того, что добавляемого ключа нет в дереве
void add_node(Node **root, int value) {
  Node *new_node = create_node(value);
  Node *parent = find_parent(*root, value);

  if (parent == NULL) {
    *root = new_node;
  }
  else if (value < parent->value) {
```

```
    parent->left = new_node;
  }
  else {
    parent->right = new_node;
  }
}

void print(Node *node) {
  if (node == NULL) {
    return;
  }

  if (node->left != NULL) {
    print(node->left);
  }

  printf("%d ", node->value);

  if (node->right != NULL) {
    print(node->right);
  }
}
```

Кроме рассмотренного выше узлового представления, бинарное дерево (почти полное) может быть представлено с помощью массива (рисунок 2). Узлы почти полного бинарного дерева могут быть занумерованы таким образом, что номер, назначенный левому сыну, равен 2k, где k - номер, присвоенный узлу отца, а номер, присвоенный правому сыну, 2k+1. Дерево с n листами представляется массивом размером 2n-1, если дерево строго бинарное, и 2n иначе. Корень индекс в массиве

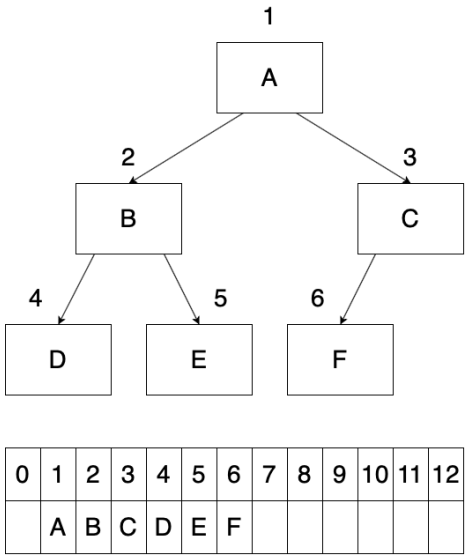


Рисунок 2 - Бинарное дерево в виде массива

Представление бинарного дерева в виде массива дерева содержится в элементе с номером 1.

Таким образом левый сын узла, расположенного в позиции k массива, находится в позиции 2k этого массива, а правый сын - в позиции 2k+1 соответственно. Аналогично позиция отца в массиве может быть получена путем округления до целого значения n/2 или (n+1)/2 (где n - позиция левого сына). Такое представление почти полного бинарного дерева в виде массива может быть расширено до общего представления любого бинарного дерева. При этом элемент массива выделяется вне зависимости от того, будет ли он содержать узел дерева. Следовательно, необходимо отметить элементы массива, соответствующие пустым (несуществующим) узлам дерева. Это может быть выполнено либо, например, занесением в эти элементы значений, не допустимых для узлов дерева, либо дополнительным массивом с информацией о таких узлах.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

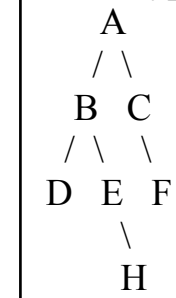
Задача 1 - максимум 4 балла

Задача 1 + 2 - максимум 9 баллов.

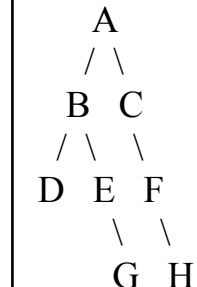
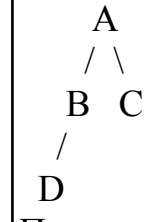
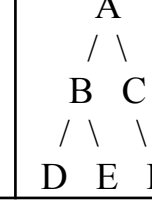
Если в условии задачи говорится о бинарном дереве, то имеется в виду произвольное бинарное дерево: добавлять и удалять элементы можно по любым правилам (если определенные правила не указаны в задаче). Если в условии задачи говорится о бинарном дереве **поиска**, то имеется в виду, что при добавлении и удалении элементов должна поддерживаться корректная организация бинарного дерева **поиска**.

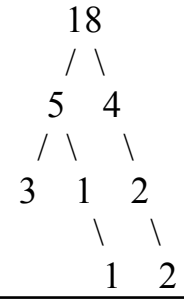
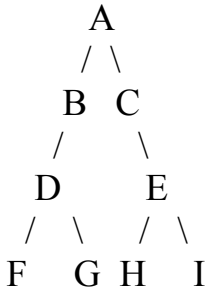
Варианты заданий

Вариант	Задание
1	1. Реализовать бинарное дерево с целочисленными ключами, вывод элементов дерева в центрированном (он же симметричный, он же in-order) порядке, добавление нового элемента с клавиатуры, задавая ключ родительского элемента и сторону (левый или правый потомок). Вывести глубину дерева (кол-во уровней). Порядок in-order для дерева A / \ B C / \ \ D E F будет D B E A C F. 2. Реализовать бинарное дерево **поиска**, добавление нового элемента с клавиатуры. Вывести бинарное дерево по уровням, разрешается использовать дополнительные коллекции. Пример. Дано дерево: A / \ B C / \ \ D E F Вывод: A B C D E F

2	1. Реализовать бинарное дерево с целочисленными ключами, вывод элементов дерева, добавление нового элемента с клавиатуры, задавая ключ родительского элемента и сторону (левый или правый потомок). Вывести количество листьев в дереве. 2. Реализовать структуру “Абонент”, которая хранит фамилию и номер телефона. Хранение организовать в бинарном дереве **поиска**. Ввести в коде программы большое количество записей с фамилией в качестве ключа. Реализовать поиск телефона по фамилии абонента, вводимой с клавиатуры. Сравнить время поиска со случаем хранения записей в массиве или списке (для этого можно повторить поиск большое количество раз).
3	1. Реализовать бинарное дерево с целочисленными ключами, вывод элементов дерева, добавление нового элемента с клавиатуры, задавая ключ родительского элемента и сторону (левый или правый потомок). По запросу выводить минимальное и максимальное значение, хранящееся в дереве. 2. Реализовать бинарное дерево **поиска**, добавление нового элемента с клавиатуры. Реализовать удаление элемента с определенным ключом из бинарного дерева **поиска**. Ключ задать с клавиатуры. Предусмотреть все случаи удаления: удаление узла с одним потомком, удаление узла с двумя потомками и реорганизацией дерева.
4	1. Реализовать бинарное дерево с целочисленными ключами, вывод элементов дерева, добавление нового элемента с клавиатуры, задавая ключ родительского элемента и сторону (левый или правый потомок). Высчитать сумму ключей листьев этого дерева. 2. Реализовать бинарное дерево **поиска** с целочисленными ключами, функцию добавления элементов с клавиатуры. Найти ближайшего общего предка для двух элементов бинарного дерева **поиска**, заданных при помощи их ключей с клавиатуры. Пример: для дерева  элементов D и H ближайший общий предок - B, для H и F - A.

5	1. Реализовать бинарное дерево **поиска** с целочисленными ключами, функцию добавления элементов с клавиатуры и вывода в прямом (pre-order) порядке. Вывести количество узлов с только одним потомком. Порядок pre-order для дерева A / \ B C / \ \ D E F будет A B D E C F. 2. Реализовать бинарное дерево с целочисленными ключами, вывод элементов дерева, добавление нового элемента с клавиатуры, задавая ключ родительского элемента и сторону (левый или правый потомок). Вывести количество элементов на каждом уровне бинарного дерева, где корень - 0-й уровень, его потомки - 1-й уровень, их потомки - 2-й уровень и т.д.
6	1. Реализовать бинарное дерево с целочисленными ключами, добавление нового элемента с клавиатуры, задавая ключ родительского элемента и сторону (левый или правый потомок). Проверить, является ли дерево строго бинарным деревом. Строго бинарное дерево - бинарное дерево, у которого каждый узел, не являющийся листом, имеет непустые правые и левые поддеревья. 2. Реализовать бинарное дерево **поиска** с целочисленными ключами, функцию добавления элементов с клавиатуры. Задать два бинарных дерева. Проверить, является ли одно дерево копией какого-либо **поддерева** другого дерева.
7	1. Реализовать бинарное дерево с целочисленными ключами, добавление нового элемента с клавиатуры, задавая ключ родительского элемента и сторону (левый или правый потомок). С клавиатуры задать ключ M. Вывести кол-во узлов от корня до узла с ключом M. 2. Реализовать бинарное дерево с целочисленными ключами и строчными значениями, функцию добавления элементов. С клавиатуры ввести узлы дерева. Добавить счётчик обращений к узлу и возможность поиска значения по ключу. При поиске у искомого узла инкрементируется счетчик обращений. По запросу с клавиатуры построить на основе этого дерева бинарное дерево **поиска**, где ключами является кол-во обращений к узлу, а значениями - структуры, содержащие ключ и значение элементов предыдущего дерева.

8	1. Реализовать бинарное дерево с целочисленными ключами, добавление нового элемента с клавиатуры, задавая ключ родительского элемента и сторону (левый или правый потомок). Высчитать сумму ключей всех узлов этого дерева. 2. Реализовать бинарное дерево **поиска** с целочисленными ключами, функцию добавления элементов с клавиатуры. Распечатать левый вид бинарного дерева. Левый вид дерева - последовательность самых левых элементов дерева на каждом уровне от корня до последнего уровня. Пример. Для дерева  левый вид будет следующим: A B D G.
9	1. Реализовать бинарное дерево **поиска** с целочисленными ключами, функцию добавления элементов с клавиатуры. Ввести элементы бинарного дерева поиска и преобразовать его в односвязный список, отсортированный по убыванию. Дополнительных коллекций и сортировок не использовать. 2. Реализовать бинарное дерево с целочисленными ключами, добавление нового элемента с клавиатуры, задавая ключ родительского элемента и сторону (левый или правый потомок). Проверить, является ли бинарное дерево завершенным бинарным деревом. Завершенным бинарным деревом будем считать дерево, в котором каждый узел каждого уровня является полным за исключением, возможно, последнего, в котором все узлы сдвинуты влево. Пример завершенного бинарного дерева:  Пример незавершенного бинарного дерева: 

10	1. Реализовать бинарное дерево **поиска** с целочисленными ключами, функцию добавления элементов с клавиатуры. Ввести элементы бинарного дерева. Проверить, является ли бинарное дерево вырожденным до списка, т.е. все узлы имеют потомка только с одной стороны. 2. Реализовать бинарное дерево с целочисленными ключами, добавление нового элемента с клавиатуры, задавая ключ родительского элемента и сторону (левый или правый потомок). Проверить, является ли бинарное дерево бинарным деревом суммы, то есть каждый узел кроме листьев является суммой всех элементов своего поддерева. Пример дерева суммы: 
11	1. Реализовать бинарное дерево **поиска** с целочисленными ключами, функцию добавления элементов с клавиатуры. Ввести элементы бинарного дерева поиска. Найти минимальный элемент в правой от корня части бинарного дерева поиска, максимальный элемент в левой от корня части бинарного дерева поиска. 2. Реализовать бинарное дерево с целочисленными ключами, добавление нового элемента с клавиатуры, задавая ключ родительского элемента и сторону (левый или правый потомок). Проверить, симметрична ли структура бинарного дерева (левая часть является отражением правой, не учитывая значения ключей). Пример дерева с симметричной структурой: 

12	1. Реализовать бинарное дерево с целочисленными ключами, добавление нового элемента с клавиатуры, задавая ключ родительского элемента и сторону (левый или правый потомок). Ввести элементы бинарного дерева. Вывести количество узлов дерева, у которых есть оба дочерних узла. 2. Реализовать бинарное дерево **поиска** с целочисленными ключами, функцию добавления элементов с клавиатуры. Проверить, являются ли два узла бинарного дерева двоюродными братьями (у их предков общий предок). Ключи узлов задать с клавиатуры.
13	1. Реализовать бинарное дерево **поиска** с целочисленными ключами, функцию добавления элементов с клавиатуры. Ввести элементы бинарного дерева поиска. Найти сумму N максимальных ключей элементов дерева. N ввести с клавиатуры. 2. Реализовать бинарное дерево с целочисленными ключами, добавление нового элемента с клавиатуры, задавая ключ родительского элемента и сторону (левый или правый потомок). Сделать зеркальное отражение бинарного дерева.
14	1. Реализовать бинарное дерево с целочисленными ключами, добавление нового элемента с клавиатуры, задавая ключ родительского элемента и сторону (левый или правый потомок). Ввести элементы бинарного дерева. Удалить все листья дерева с четными ключами. 2. Реализовать бинарное дерево **поиска** с целочисленными ключами, функцию добавления элементов с клавиатуры. Объединить два бинарных дерева поиска. Можно использовать дополнительные коллекции.

15	1. Реализовать бинарное дерево с целочисленными ключами, добавление нового элемента с клавиатуры, задавая ключ родительского элемента и сторону (левый или правый потомок). Ввести элементы бинарного дерева. Вывести все элементы дерева в обратном (post-order) порядке так, что для каждого из них выводится ключ самого элемента, ключ родителя и сторона по отношению к родителю (левая или правая). Порядок post-order для дерева будет D E B F C A. 2. Дано бинарное дерево **поиска**. В дереве произвольные два элемента поменяли местами. Найти эти два элемента и скорректировать дерево, вернуть эти элементы на изначальные места. Пример. Для дерева эти элементы - это 12 и 4.