**1. ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ**

        Для всех вариантов заданий:

* определить типы и функции в соответствии с вариантом задания;
* в функции **main()** реализовать демонстрацию работы созданных функций;

**ЗАМЕЧАНИЯ:**

        Весь ввод/вывод должен осуществляться только внутри функции **main()**. Если в функции необходимо передавать объекты типа структура, то передавать их следует через указатель либо ссылку на объект.

**4)** Имеется узел бинарного дерева:

struct Node

{

char name[10]; //имя узла

Node \* left; //левая ветвь

Node \* right; //правая ветвь

};

        Определите функцию добавления новых узлов в дерево **AddNode()** и функцию удаления дерева **DelTree()**.Определите рекурсивную функцию вывода такого дерева **PrintTree()** на экран.

**2. ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ**

        Для всех вариантов заданий:

* определить классы их свойства и методы в соответствии с вариантом задания;
* в функции **main()** реализовать демонстрацию работы с объектами созданных классов;

**ЗАМЕЧАНИЯ:**

**Варианты заданий**

        Весь ввод/вывод должен осуществляться только внутри функции **main()**.

        Передача в функции объектов типа класс должна осуществляться через указатель либо ссылку на объект.

        Все другие нестатические данные должны передаваться функциям через параметры.

        Все нестатические элементы класса должны иметь закрытый атрибут доступа. Доступ извне к таким элементам должен осуществляться через функции-элементы класса.

**4)** Определите узел бинарного дерева следующим образом:

class Node

{

char name[10]; //имя узла

Node \* left; //левая ветвь

Node \* right; //правая ветвь

...

};

        Определите в классе следующие функции:

**Init()** – инициализация узла. Функция должна установить указатели на левый и правый узел в ноль;

**AddNode()** – добавление узла в левую или правую ветви. Если слева нет узла, то добавить слева, если справа нет узла, то добавить справа, иначе ничего не добавлять;

**DelTree()** – удаление поддеревьев;

**Print()** – рекурсивная функция вывода дерева на экран.

**3. ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ**

        Для всех вариантов заданий:

* определить классы их свойства и методы в соответствии с вариантом задания;
* в функции **main()** реализовать демонстрацию работы с объектами созданных классов;

**ЗАМЕЧАНИЯ:**

* Весь ввод/вывод должен осуществляться только внутри функции **main()**.
* Передача в функции объектов типа класс должна осуществляться через указатель либо ссылку на объект.
* Все другие нестатические данные должны передаваться функциям через параметры.
* Все нестатические элементы класса должны иметь закрытый атрибут доступа. Доступ извне к таким элементам должен осуществляться через функции-элементы класса.

**4)** Определите узел бинарного дерева следующим образом:

class Node

{

char name[10]; //имя узла

Node \* left; //левая ветвь

Node \* right; //правая ветвь

...

};

        Определите в классе следующие функции:

**Init()** – инициализация узла. Функция должна установить указатели на левый и правый узел в ноль;

**AddNode()** – добавление узла в левую или правую ветви. Если слева нет узла, то добавить слева, если справа нет узла, то добавить справа, иначе ничего не добавлять;

**DelTree()** – удаление поддеревьев;

**Print()** – рекурсивная функция вывода дерева на экран.

**Определить в классе конструкторы: по умолчанию, с параметрами, копирования. Определить деструктор.**

**4. ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ**

        Для всех вариантов необходимо выполнить следующее:

* определить классы их свойства и методы в соответствии с вариантом задания;
* в функции main() реализовать демонстрацию работы операций над объектами созданных классов;

**ЗАМЕЧАНИЯ:**

* Весь ввод/вывод должен осуществляться внутри функции main() если только это не оговорено для отдельных функций.
* Передача в функции объектов типа класс должна осуществляться через указатель либо ссылку на объект.
* Все другие нестатические данные должны передаваться функциям через параметры.
* Все нестатические элементы класса должны иметь закрытый атрибут доступа. Доступ извне к таким элементам должен осуществляться через функции-элементы класса.

**4)** Определите в классе **Node** следующие операции:

Node& operator=(const Node&);//копирование узла

Node& operator<<(const char\*);//добавить узел влево

Node& operator>(const char\*);//добавить узел вправо

Node& operator[](const char\*);//вернуть узел по его имени

friend ostream& operator<<(ostream& os, const Node&);//вывод