**Binary Tree in C**

This is a header file for a binary tree implementation in C. It defines a structure called TreeBase, which contains basic information about the tree such as the element pointer, element size, logical length, and allocated length. It also contains function pointers for data comparison and memory freeing. Additionally, there are a number of macros defined to make using the binary tree more convenient.

**Macros:**

The given macros are for creating and manipulating a binary tree template for a specific data type T.

* **TreeT(T)** macro creates a struct with a TreeBase as a member and an additional member with the name data\_ of the type T.

**Functions:**

* **void TreeNew\_(TreeBase\* t, unsigned int elemsize, int(\*DataCmp)(const void \*key1, const void \*key2, unsigned int keysize), void(\*FreeFunc)(void\* elems))**: This function creates a new binary tree with a given comparison function and memory freeing function. It sets up the TreeBase structure with the appropriate values and initializes the binary tree. The function takes in a pointer to the TreeBase structure, the size of an element, a pointer to a function that compares data, and a pointer to a function that frees memory as arguments.
* **TreeBase TreeInsert\_(TreeBase\* t, void\* data)**: This function inserts an element into the binary tree. It takes in a pointer to the TreeBase structure and a pointer to the element to be inserted as arguments.
* **bool TreeContains\_(TreeBase\* t, void\* data)**: This function checks if an element is present in the binary tree. It takes in a pointer to the TreeBase structure and a pointer to the element to be checked as arguments.
* **TreeBase TreeRemove\_(TreeBase\* t, void\* data)**: This function removes an element from the binary tree. It takes in a pointer to the TreeBase structure and a pointer to the element to be removed as arguments.
* **void TreeClear\_(TreeBase\* t)**: This function clears all elements from the binary tree. It takes in a pointer to the TreeBase structure as an argument.
* **void TreeDelete\_(TreeBase\* t)**: This function deletes the binary tree and frees the memory associated with it. It takes in a pointer to the TreeBase structure as an argument.
* **unsigned int TreeSize\_(TreeBase\* t)**: This function returns the number of elements in the binary tree. It takes in a pointer to the TreeBase structure as an argument.
* **bool TreeEmpty\_(TreeBase\* t)**: This function checks if the binary tree is empty. It takes in a pointer to the TreeBase structure as an argument.
* **TreeBase\* TreeMin\_(TreeBase\* t)**: This function returns a pointer to the smallest element in the binary tree. It takes in a pointer to the TreeBase structure as an argument.
* **TreeBase\* TreeMax\_(TreeBase\* t)**: This function returns a pointer to the largest element in the binary tree. It takes in a pointer to the TreeBase structure as an argument.
* **void TreeInOrder\_(TreeBase\* t, const char\* data\_format)**: This function performs an in-order traversal of the binary tree and prints the data in the specified format. It takes in a pointer to the TreeBase structure and a string specifying the format of the data as arguments.
* **void TreePreOrder\_(TreeBase\* t, const char\* data\_format)**: This function performs a pre-order traversal of the tree and prints the data of each node in the specified format. The function takes in a pointer to the TreeBase structure and a pointer to a string specifying the format of the data to print as arguments.
* **void TreePostOrder\_(TreeBase\* t, const char\* data\_format)**: This function performs a post-order traversal of the tree and prints the data of each node in the specified format. The function takes in a pointer to the TreeBase structure and a pointer to a string specifying the format of the data to print as arguments.
* **Templates:**
* **Tree** is a binary tree template for any data type.
* **TreeInt** is a binary tree template specifically for integers.
* **TreeFloat** is a binary tree template specifically for single-precision floating-point numbers.
* **TreeDouble** is a binary tree template specifically for double-precision floating-point numbers.
* **TreeChar** is a binary tree template specifically for characters.
* **TreeString** is a binary tree template specifically for null-terminated character strings.