Traversieren durch Bäume

9. Aufgabe *3 Punkte*

Implementieren Sie für den binären Baum des letzten Übungsblattes noch folgende Funktion:

int **bitree_merge**(BiTree *merge, BiTree *left, BiTree *right, int count);

Zur Funktion *bitree_merge*:
Der binäre Baum *merge* soll als Wurzelknoten einen Knoten mit dem Wert *count* erhalten. Als linken beziehungsweise rechten Nachfolger erhält der neue Baum die Bäume *left* und *right*.

10. Aufgabe *7 Punkte*

Nach der gelungen Konstruktion des binären Baumes möchten Sie nun auch Ihren Baum durchqueren. Für Bäume gibt es verschiedene Vorgehensweisen wie systematisch jeder Knoten aufgesucht werden kann. Diese Methoden unterscheiden sich vor allem hinsichtlich der Reihenfolge, in welcher die Knoten besucht werden.

Preorder-Traversierung:
„Besuche die Wurzel, besuche dann den linken Unterbaum, besuche dann den rechten Unterbaum."

Inorder-Traversierung:
„Besuche den linken Unterbaum, besuche dann die Wurzel, besuche dann den rechten Unterbaum."

Postorder-Traversierung:
„Besuche den linken Unterbaum, besuche dann den rechten Unterbaum, besuche dann die Wurzel."

Implementieren Sie dazu folgende Funktionen:

int **preorder**(const BiTreeNode *node);

int **inorder**(const BiTreeNode *node);

int **postorder**(const BiTreeNode *node);

Die Funktionen sollen die Werte der einzelnen Knoten ausgeben. Im Fehlerfall wird *-1* zurückgeliefert und ansonsten *0*.