Lab 10: Binary Trees

Aim

This lab class gives you an opportunity to:

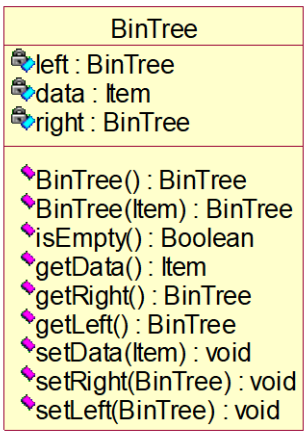
- explore the Binary Tree ADT;
- practise writing recursive functions; and
- learn about different approaches for traversing non-linear (tree-based) data structures.

Context

A *Binary Tree* is a tree of degree 2 and is either:

- empty; or
- a node with a value (*data*), and two branches (*left* and *right*) each of which is also a binary tree.

The UML diagram for the ADT is:



The first 'constructor' creates an empty binary tree while the second creates a leaf node with the given value within the node. `isEmpty()` examines the binary queue and indicates whether there are any items present (returning `true` if not, and `false` if so), `get???()` returns the relevant component of the current tree (with an error occurring if the binary tree is empty), and `set???()` stores the given value in the relevant component of the current tree (again, with an error occurring if the binary tree is empty).

Displaying the contents of a tree-based data structure is a much more complex problem than displaying the contents of a linked-list. A linked-list is a *linear* data structure and only a single 'path' exists from the first node to the last. In a tree, there is still a single starting point but possibly many end points (the leaf nodes).

The purpose of this tutorial is to add a function to 'diagrammatically' display

the contents of a binary tree. To assist, the nodes in the binary tree have been augmented to include a 'label' which can hold a number. The tree nodes can be 'decorated' so that the left-most node has label 1, the second from left node has label 2, and so on.

Two functions need to be written:

- `height()` — which should calculate the number of 'levels' in the tree. It has function heading (from `BinTree.h`):

  ```
  int height(BinTree t);
  ```

  and has the following algorithm:

  if the tree is empty
  height=0
  otherwise
  if the height of the left sub-tree > the height of the right sub-tree
  height=1+height of the left sub-tree
  otherwise
  height=1+height of the right sub-tree

- `decorate()` — which should set the label field of each note to its left-to-right position (a number from 1 to *n* where *n* is the total number of nodes in the tree). It has function heading (from `BinTree.h`):

  ```
  int decorate(BinTree t,int w);
  ```

  and has the following algorithm:

  if the tree is not empty
  decorate the left sub-tree updating the latest label (w)
  set the label of the current node to the latest label
  increment the latest label
  decorate the right sub-tree updating the latest label
  last label=latest label

Tasks

1. The compressed project folder (`Lab10.zip`) should be obtained from MyLO and all contents extracted to your home. Open the project folder and open the project file (`Lab10.sln`).
2. Complete the implementation of the `height()` function from `bin_tree.c`.
3. Compile and execute the project to check your implementation. Correct any errors.
4. Complete the implementation of the `decorate()` function from `bin_tree.c`.
5. Compile and execute the project to check your implementation. Correct any errors.