Departamento de Matemática da Universidade de Coimbra		
2013/2014	**Programação Orientadas para os Objectos**	**Projecto 1**

Arithmetic Expressions Calculator

The reading and calculation of arithmetic expressions using infix notation is difficult to implement because it is an ambiguous notation. For example, how should we read and calculate the expression $x+y\times z$? The problem is the precedence of the operations that is implied but not explicitly written in the expression.

The polish notation[2], is an unambiguous alternative.

prefix notation (polish notation) $+\times yzx$

postfix notation $xyz\times +$

In polish notation we have, always, an operator followed by two operands.

Project of Programação Orientada aos Objectos 2013/2014.

1. Implement, using *C++* a class A.D.T. Binary Trees (BT) creating an instance with Strings:

   BT = ({BTempty,(Elements,BT,BT)}, {getRoot, getBTleft, getBTright, setRoot, setBTleft, setBTright, destroyBTleft, destroyBTright, empty?, inorder, preorder, postorder})

   Consider the error situations.

2. Build a program capable of reading an expression in polish notation, build the binary tree on the process and, at the end, calculate the value of the expression making an inorder traversal of the tree.

- Document your program. Internal and external documentation.
- The report (external documentation, max 5pp) should include the UML diagram of the class structure and a small user's manual. You should identify the group.
- You have to deliver (by electronic mail) one `zip` or `tar.gz` archive containing all the files related to the program (`Makefile, .cpp, .hpp`), and also the report (`PDF` format), up to the 24h00 hours of the project deadline.

p.t.o.

[2]Due to Polish mathematician Jan Łukasiewicz

The "Binary Trees" (BT) algebra (BT) can be specified as:

Attributes

$$\text{BT} = \begin{cases} \text{BTempty}, & \text{empty binary tree} \\ (\text{root}, \text{BTleft}, \text{BTright}), & \text{non-empty binary tree} \end{cases}$$

Methods

$$\begin{array}{rccl}
\text{getRoot}: & BT & \longrightarrow & Elements \cup \{error\} \\
& ab & \longmapsto & \begin{cases} r, & ab = (r, ableft, abright) \\ error, & ab = \text{BTempty} \end{cases} \\
\text{getBTleft}: & BT & \longrightarrow & BT \cup \{error\} \\
& ab & \longmapsto & \begin{cases} ableft, & ab = (root, ableft, abright) \\ error, & ab = \text{BTempty} \end{cases} \\
\text{getBTright}: & BT & \longrightarrow & BT \cup \{error\} \\
& p & \longmapsto & \begin{cases} abright, & ab = (root, ableft, abright) \\ error, & ab = \text{BTempty} \end{cases} \\
\text{setRoot}: & Elements \times BT & \longrightarrow & BT \cup \{error\} \\
& (r, ab) & \longmapsto & \begin{cases} (r, ableft, abright), & ab = (r', ableft, abright) \\ error, & b = \text{BTempty} \end{cases} \\
\text{setBTleft}: & BT \times BT & \longrightarrow & BT \cup \{error\} \\
& (ableft, ab) & \longmapsto & \begin{cases} (r, ableft, abright), & ab = (r, ableft', abright) \\ error, & ab = \text{BTempty} \end{cases} \\
\text{setBTright}: & BT \times BT & \longrightarrow & BT \\
& (abright, ab) & \longmapsto & \begin{cases} (r, ableft, abright), & ab = (r, ableft, abright') \\ error, & ab = \text{BTempty} \end{cases} \\
\text{destroyBTleft}: & BT & \longrightarrow & BT \\
& ab & \longmapsto & \begin{cases} (r, BTempty, abright), & ab = (r, ableft, abright) \\ error, & ab = \text{BTempty} \end{cases} \\
\text{destroyBTright}: & BT & \longrightarrow & BT \\
& ab & \longmapsto & \begin{cases} (r, ableft, BTempty), & ab = (r, ableft, abright) \\ error, & ab = \text{BTempty} \end{cases} \\
\text{empty?}: & BT & \longrightarrow & Bool \\
& ab & \longmapsto & \begin{cases} \mathcal{V}, & ab = (r, ableft, abright) \\ \mathcal{F}, & ab = \text{BTempty} \end{cases} \\
\text{inorder}: & BT & \longrightarrow & Elementslist \\
& ab & \longmapsto & \begin{cases} inorder(ableft) : r : inorder(abright), & \\ & ab = (r, ableft, abright) \\ emptyList, & ab = \text{BTempty} \end{cases}
\end{array}$$

In the *C++* implementation you should consider the constructors and the destructor.
The tree traversal should be adapted to this project having as result the value of the expression.