```
1 #include "bintree.h"
 2 #include <iostream>
 3
4 #include <cmath>
5 #include <queue>
6 #include <iomanip>
8 using namespace std;
9
10 void Expr_post(string expr, int pos, bintree<char> &A, bintree<char>::node aux_node)
11 | {
12
       if (pos >= 0)
13
14
           A.insert_right(aux_node, expr[pos]);
15
16
           A.insert_left(aux_node, expr[pos]);
17
18
           pos--;
19
           if (*aux_node.right() == '+' || *aux_node.right() == '-' || *aux_node.right()
          || *aux node.right() == '/')
20
21
               Expr_post(expr, pos, A, aux_node.right());
22
               pos--;
23
           if (*aux node.left() == '+' || *aux_node.left() == '-' || *aux_node.left() ==
24
       || *aux_node.left() == '/')
25
26
               Expr_post(expr, pos, A, aux_node.left());
27
           }
28
       }
29 }
30 bintree<char> Expr_post(string expr)
31 | {
32
       bintree<char> resultado(expr[expr.size() - 1]);
33
       int i = expr.size() - 2;
34
       bintree<char>::node aux node = resultado.root();
35
       if (i >= 0)
36
       {
37
           resultado.insert_right(aux_node, expr[i]);
38
39
40
           resultado.insert_left(aux_node, expr[i]);
41
           i--;
42
           if (*aux_node.right() == '+' || *aux_node.right() == '-' || *aux_node.right()
43
          || *aux node.right() == '/')
44
45
               Expr_post(expr, i, resultado, aux_node.right());
46
47
           if (*aux_node.left() == '+' || *aux_node.left() == '-' || *aux_node.left() ==
48
       || *aux node.left() == '/')
49
50
               Expr_post(expr, i, resultado, aux_node.left());
51
52
       }
53
       return resultado;
54 }
55
56 template <class T>
```

localhost:4649/?mode=clike 1/2

```
ejercicio08.cpp
 57 void postorden(const bintree<T> &A, const typename bintree<T>::node &v)
 58 {
        if (!v.null())
 59
 60
        {
 61
            postorden(A, v.left());
 62
            postorden(A, v.right());
 63
            cout << *v;
 64
 65
        }
 66 }
 67
 68
 69 template <class T>
 70 void MostrarArbol(const bintree<T> &A, typename bintree<T>::node root)
 71 {
 72
        queue<typename bintree<T>::node> colaNodos;
 73
        int totalNodos = A.size();
 74
        int techo = log2(totalNodos + 1);
 75
        colaNodos.push(root);
 76
        int pot = 0;
        while (colaNodos.size() > 0)
 77
 78
 79
             int niveles = colaNodos.size();
 80
            while (niveles > 0)
 81
            {
                 typename bintree<T>::node nodoAux = colaNodos.front();
 82
 83
                 colaNodos.pop();
                 cout << setw((niveles == pow(2, pot)) ? pow(2, (techo - pot)) : pow(2,
 84
    (techo - pot + 1)));
 85
                 cout << *nodoAux;</pre>
                 if (!nodoAux.left().null())
 86
                     colaNodos.push(nodoAux.left());
 87
                 if (!nodoAux.right().null())
 88
                     colaNodos.push(nodoAux.right());
 89
 90
                 niveles--;
             }
 91
 92
            pot++;
 93
            cout << endl;</pre>
        }
 94
 95 }
 96
 97 int main()
 98 {
        bintree<char> aux = Expr_post("ab*b*d/e+");
 99
        MostrarArbol(aux, aux.root());
100
        postorden(aux, aux.root());
101
102
        cout << endl;</pre>
103 }
```

2/2 localhost:4649/?mode=clike