```cpp
#include "bintree.h"
#include <iostream>

#include <queue>
#include <cmath>
#include <iomanip>

using namespace std;

template <class T>
void MostrarArbol(const bintree<T> &A, typename bintree<T>::node root)
{
    queue<typename bintree<T>::node> colaNodos;
    int totalNodos = A.size();
    int techo = log2(totalNodos + 1);
    colaNodos.push(root);
    int pot = 0;
    while (colaNodos.size() > 0)
    {
        int niveles = colaNodos.size();
        while (niveles > 0)
        {
            typename bintree<T>::node nodoAux = colaNodos.front();
            colaNodos.pop();
            cout << setw((niveles == pow(2, pot)) ? pow(2, (techo - pot)) : pow(2,
(techo - pot + 1)));
            cout << *nodoAux;
            if (!nodoAux.left().null())
                colaNodos.push(nodoAux.left());
            if (!nodoAux.right().null())
                colaNodos.push(nodoAux.right());
            niveles--;
        }
        pot++;
        cout << endl;
    }
}

double valor(char a)
{
    return (double)a;
}

double evaluacion(bintree<char> arb, bintree<char>::node v)
{
    if (!v.null())
    {
        switch (*v)
        {
        case '+':
            return evaluacion(arb,v.left()) + evaluacion(arb,v.right());
            break;
        case '*':
            return evaluacion(arb,v.left()) * evaluacion(arb,v.right());
            break;
        case '-':
            return evaluacion(arb,v.left()) - evaluacion(arb,v.right());
            break;
        case '/':
            return evaluacion(arb,v.left()) / evaluacion(arb,v.right());
```

```
                break;
            default:
                return valor(*v);
                break;
            }
        }
}

int main()
{
    bintree<char> arb('*');

    bintree<char>::node aux;

    //RAMA IZQUIERDA

    arb.insert_left(arb.root(), '-');

    aux = arb.root().left();

    arb.insert_left(aux, '/');

    arb.insert_right(aux, '+');

    aux = aux.left();

    arb.insert_left(aux, 'A');

    arb.insert_right(aux, 'C');

    aux = aux.parent().right();

    arb.insert_left(aux, 'D');

    arb.insert_right(aux, 'Z');

    //RAMA DERECHA

    arb.insert_right(arb.root(), '*');

    aux = arb.root().right();

    arb.insert_left(aux, '/');

    arb.insert_right(aux, '+');

    aux = aux.left();

    arb.insert_left(aux, 'Y');

    arb.insert_right(aux, 'H');

    aux = aux.parent().right();

    arb.insert_left(aux, 'F');

    arb.insert_right(aux, 'F');

    cout << "---------------ARBOL ORIGINAL------------------" << endl;

    MostrarArbol(arb, arb.root());

    cout << "El valor del arbol es --> " << evaluacion(arb, arb.root()) << endl;
}
```