Grupo 13 - Laboratorio 3

Estudiantes:

- FERNANDO HERMOSO CARA (C40)
- IGNACIO PALLÁS GOZÁLVEZ (C62)

10

Fichero `2020_05_04/C40/11-C09/14330_AC/Prac9.cut.cpp`

```
/*@ <answer> */

/*
 * Comienza poniendo el nombre de los/as componentes del grupo:
 *
 * Estudiante 1: Ignacio Palls Gozlvez
 * Estudiante 2: Fernando Hermoso Cara
 */

 /*@ </answer> */
/*@ <answer> */

using namespace std;


void  procesaArbol(const BinTree<string>& arbol, int& result, const MapTree<string, int>& dic
    ) {
    //caso base es que el root no se un operando

    if (arbol.empty()) {
        result = 0;
        return;
    }
    else {
        // caso base
        if (arbol.root() != "+" && arbol.root() != "-" && arbol.root() != "*") {

            //mirar si es una variable y ver si esta en el diccionario
            if (dic.contains(arbol.root())) {
                result = dic.at(arbol.root());
            }
            //si no es una variable que este ya en el dicionario => que es un numero
            else {
                result = stoi(arbol.root());
            }
            return;
        }

        int  izq, der;

        procesaArbol(arbol.left(), izq, dic);
```

Podríais haber devuelto un int, en lugar de tener un parámetro de salida.

```
        procesaArbol(arbol.right(), der, dic);

        if (arbol.root() == "+") {
            result = izq + der;
        }
        else if (arbol.root() == "-") {
            result = izq - der;
        }
        else if (arbol.root() == "*") {
            result = izq * der;
        }
    }
}


bool tratar_caso() {
    int numInstruciones;
    MapTree<string, int> dic;
    cin >> numInstruciones;

    if (numInstruciones == 0) {
        return false;
    }
    else {
        BinTree<std::string> instruccion;
        for (int i = 0; i < numInstruciones; i++) {
            MapTree<string, int>::MapEntry map("␣", 0);
            instruccion = parse(cin);

           map.key = instruccion.left().root();

            int valor;
            procesaArbol(instruccion.right(), valor, dic);
            map.value = valor;
            if (dic.contains(map.key)) {
                dic.erase(map.key);
            }
            dic.insert(map);

        }

        for (auto it = dic.cbegin(); it != dic.cend(); it++) {
            MapTree<string, int>::MapEntry aux = *it;
            cout << aux.key << "␣=␣" << aux.value << endl;
        }

        cout << "---" << endl;
        return true;
    }
}
```

map[key] = value
hace lo mismo

```
/*@ </answer> */
```