PRÁCTICA 6: 1 sesión (S8: del 16 al 20 de mayo)

EVALUACIÓN DE EXPRESIONES ALGEBRAICAS CON VARIABLES

En esta práctica procederemos a evaluar expresiones escritas en forma 'infija' (el operador está situado entre los operandos) con paréntesis y con operandos que serán 'variables' (cada vez que deseemos evaluar la expresión pediremos al usuario que nos dé los valores concretos de las variables para esa evaluación). Las operaciones válidas que evaluaremos serán las operaciones binarias '+', '-', '*', '/'.

Expresiones válidas a evaluar serán, por ejemplo: `a + (b - c) / d, (a+b) / (c-d) * e,...`

0. OBJETIVOS

a. Utilización del tipo abstracto de datos 'Arbol binario'.

b. Reutilización de código (reutilización de la clase 'Pila').

c. Utilización de código no desarrollado por el usuario (función 'infija_a_posfija' y clase 'TablaSimbolos").

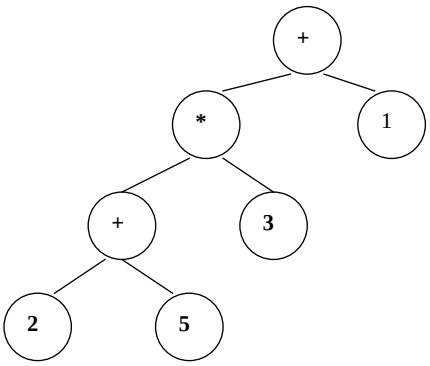
d. Trabajo con funciones recursivas.

1. INTRODUCCIÓN

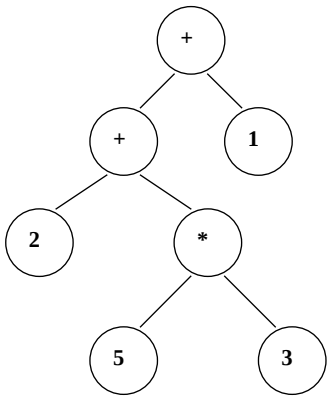
Una manera adecuada de representar expresiones aritméticas es a través de los árboles binarios de expresiones. Esta representación retiene de manera natural la precedencia y la asociatividad de los operadores aritméticos.

En un árbol binario de expresiones cada nodo contiene la información de un elemento de la expresión (un operando o un operador) y la propia estructura del árbol viene determinada por la forma de la expresión aritmética.

Un ejemplo de esto lo tenemos en los siguientes árboles de expresiones:



(2+5)*3+1



2+5*3+1

Se puede ver que dependiendo de la colocación de los paréntesis se fuerza el cambio de precedencia de los operadores y las expresiones generan árboles distintos.

También es importante resaltar que en el árbol claramente no son necesarios los paréntesis, ya que la precedencia en las operaciones viene dada por la estructura del árbol.

La evaluación de las expresiones, es decir, la obtención de su resultado, se consigue recorriendo el árbol de expresiones en forma **postfija** y realizando para cada nodo que contiene un operador dicha operación sobre sus hijos. De hecho, si examinamos el árbol en recorrido post-orden imprimiendo en pantalla el

contenido de cada nodo, obtendremos lo que se denomina la **expresión postfija** de la expresión. Las expresiones postfijas de los árboles del ejemplo son las siguientes:

Para la **expresión infija (2+5)*3+1,** la expresión postfija es **2 5 + 3 * 1 +**

Para la **expresión infija 2+5*3+1,** la expresión postfija es **2 5 3 * + 1 +**

Nótese que la expresión postfija, al igual que ocurría en el árbol, no necesita utilizar paréntesis.

De la misma forma que podemos obtener la expresión postfija a partir del árbol de expresiones, también podemos hacer lo contrario, es decir, obtener el árbol de expresiones a partir de una expresión postfija. Este hecho va a ser utilizado en la presente práctica.

2. REALIZACIÓN DE LA PRÁCTICA

a.- Obtención de la expresión postfija y la tabla de variables.

El programa deberá empezar pidiendo la expresión aritmética en forma infija que deseamos evaluar.

Una vez guardada la expresión en una variable de tipo *string* pasaremos la expresión a su forma postfija y extraeremos de la expresión una tabla con las variables utilizadas en la expresión Utilizaremos para ello una función proporcionada por el profesor de prácticas '**`infijo_a_posfijo`**'. El prototipo de esta función es:

```
bool infijo_a_posfijo (string , string &, TablaSimbolos &);
```

Esta función tiene una entrada que es la cadena de la expresión infija.

Como salidas de la función tenemos tres valores (dos devueltos por referencia y uno como resultado de la función):

La primera salida (segundo parámetro de la función) es la cadena con la expresión postfija.

La segunda salida (tercer parámetro de la función) es un objeto de tipo `TablaSimbolos` que es la tabla donde se almacenan los identificadores correspondientes a los nombres de cada variable.

La tercera salida (valor devuelto por la función) es un valor booleano. La función devuelve '`true`' si la expresión esta sintácticamente bien formada o '`false`' si había algún error sintáctico en ella de estos dos tipos:

- Nos hemos dejado algún paréntesis sin cerrar
- El número de operadores o de operandos no es el correcto

La función no detecta errores como que el orden entre operandos y operadores sea incorrecto.

b. Paso de la expresión postfija a su representación mediante un árbol de expresión.

El árbol que vamos a utilizar para guardar la expresión tendrá como información en cada uno de sus nodos un carácter, tal y como hemos visto en la introducción.

Para realizar el paso de la expresión postfija a su representación en el árbol vamos a necesitar una pila (de árboles binarios) para construir el árbol binario que guardará la expresión.

Para conseguirlo realizaremos el siguiente proceso:

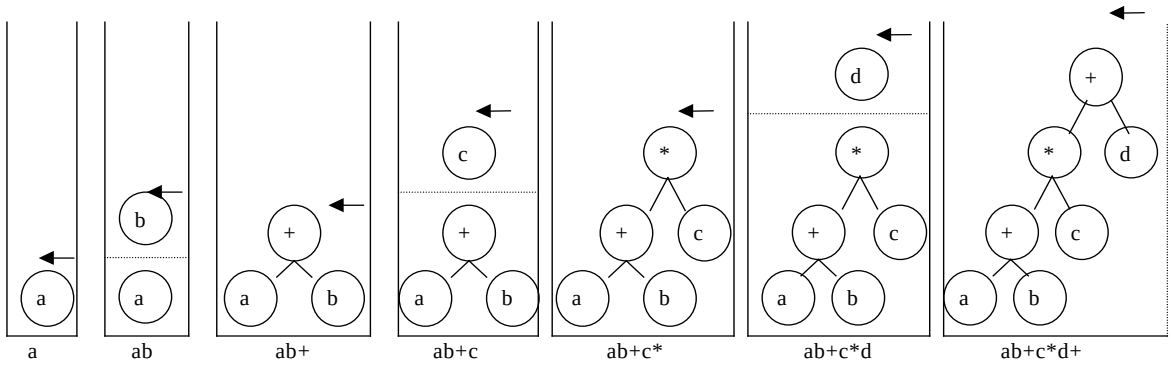
Para cada uno de los elementos de la expresión postfija tenemos que hacer lo siguiente:

- Si el elemento analizado es un operando (variable), construimos un árbol binario de un único nodo que contiene como información el carácter que representa la variable y como hijos dos subárboles vacíos. Este árbol lo apilamos en la pila.
- Si el elemento analizado es un operador, construiremos un nuevo árbol de la siguiente manera:

- Extraemos dos elementos de la pila (cada elemento puede ser un árbol complejo si este procedimiento se ha hecho ya anteriormente).
- El **primer elemento extraído** de la pila será el **hijo derecho** del nuevo árbol.
- El **segundo elemento extraído** será el **hijo izquierdo** del nuevo árbol
- La información del nodo raíz del nuevo árbol es el carácter que represente la operación.

Una vez construido el árbol lo apilamos en la pila.

En el dibujo se detalla el proceso seguido para la obtención del árbol de expresión para la expresión algebraica postfija **`a b + c * d +`**



Una vez analizados todos los elementos de la expresión en la pila se encontrará un único árbol que es el correspondiente árbol de expresión.

Se puede comprobar la correcta ejecución de este paso si recorriendo de forma postfija el árbol obtenemos la expresión postfija original.

c. Evaluación de la expresión.

La evaluación consistirá básicamente en dos tareas:

c.1. Pedir los valores de las variables que contiene la expresión.

Para realizar esta tarea nos serviremos de los métodos públicos de la clase **`TablaSimbolos`**. La interfaz de la clase es la siguiente:

```
class TablaSimbolos
{
   public:
        //Constructor
        TablaSimbolos ();
        //Anyadir un simbolo a la tabla
        //devuelve false si la tabla esta llena o si el simbolo ya esta en la tabla
        bool NuevoSimbolo (char);
        //Devuelve el valor numerico asociado con un simbolo
        //devuelve NaN si el simbolo no esta en la tabla
        float ValorSimbolo (char);
        //Modifica el valor numerico asociado con un simbolo
        //devuelve false si el simbolo no esta en la tabla
        bool CambiarValorSimbolo (char, float);
        //Devuelve todos los simbolos almacenados en la tabla
        //en forma de cadena de caracteres
        string TodosLosSimbolos ();
    private:
        .??.
};
```

A partir de la interfaz vemos que la tarea que tenemos que realizar consistirá en pedir al objeto que nos diga que elementos contiene utilizando para ello el método **`string TodosLosSimbolos ()`**. A partir del **`string`** que nos devuelve el método, iremos preguntando al usuario cuál es el valor que quiere asignar al símbolo y a continuación lo guardamos con el método **`bool CambiarValorSimbolo (char, float)`**.

c.2. A partir de estos valores obtener el valor numérico de operar las variables con las operaciones indicadas en el árbol.

Para evaluar la expresión a través del árbol sólo **necesitamos recorrer el árbol en orden postfijo haciendo lo siguiente**:

- Si el nodo a evaluar es un operando (variable), la función que recorre el árbol debe devolver el valor de la variable correspondiente consultando el valor en la tabla de variables (utilizando el método **`float ValorSimbolo (char)`**).
- Si el nodo a evaluar es un operador, se aplica este operador sobre los valores numéricos obtenidos de evaluar a sus hijos y la función devolverá dicho valor.

Cuando termine el recorrido, el valor final obtenido corresponderá a la evaluación de la expresión completa.

Esta evaluación podrá repetirse tantas veces como desee el usuario con valores distintos de las variables.

3. REQUISITOS PARA PODER ENTRAR A REALIZAR LA PRÁCTICA

Antes de poder empezar a realizar cualquiera de las prácticas *es necesario* presentar las hojas de especificación de programas (formulario de documentación de programas) **con las tareas que se van a realizar en la práctica, explicando brevemente como se van a solucionarse los problemas que se plantean.**

En este formulario irán reflejados los siguientes detalles del programa:

- **las tres clases que se van a utilizar en la práctica:**

  Clase Pila / Clase Arbol / Clase TablaSimbolos (aunque esta última no la hayamos desarrollado nosotros vamos a utilizar sus métodos públicos y debemos documentarlos).

- **Las funciones en que vamos a dividir el programa principal y que vamos a utilizar a lo largo de nuestro programa, especificando claramente las entradas y salidas de las mismas, así como la tarea que desarrolla contada muy brevemente.**

  En esta práctica se recomienda dividir el programa principal en fundamentalmente tres funciones que atiendan cada una de las tareas especificadas en el punto 2 “realización de la práctica” (la primera función es la que corresponde a la que os proporcionará el profesor de prácticas).

- **Los diagramas de flujo tanto del programa principal como de aquellas funciones que consideremos más interesantes o complejas.**

  En esta práctica se recomienda la realización del diagrama de flujo de la función principal, así como de la función que construye el árbol binario. También se recomienda la realización en pseudocódigo de la función que evalúe el árbol binario para obtener el valor final de la expresión.

4. ENTREGA DE PROGRAMAS

Cinco días después de realizada la sesión de prácticas se entregará al profesor la siguiente información:

1) Archivo con el programa principal (**`##evaluacion.cpp`**)
2) Archivos con la clase *`Pila`* (**`##Pila.h`** y **`##Pila.cpp`**)
3) Archivos con la clase *`Arbol`* (**`##Arbol.h`** y **`##Arbol.cpp`**)

dónde ## es el número asignado a la pareja (01, 02,...) en cada grupo de prácticas.

Al comenzar la siguiente sesión de prácticas se entregará al profesor el formulario con la documentación definitiva de la práctica reflejando todo el trabajo realizado realmente en la práctica.

ENTREGA DE PROGRAMAS:

Los días 21, 22, 23, 24 y 25 de mayo de 2005, correspondientes con las fechas de realización de prácticas 16, 17, 18, 19 y 20 de mayo respectivamente.

ENTREGA DE FORMULARIOS DEFINITIVOS DE DOCUMENTACIÓN:

Los días 23, 24, 25, 26 y 27 de Mayo de 2005, correspondientes con las fechas de realización de prácticas 16, 17, 18, 19 y 20 de mayo respectivamente