Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Lenguaje Natural

Dado un árbol binario lleno, recorrer el mismo, me posiciono en su raíz y reviso si sus hijos son distintos de nulo. Cuando tiene los 2 hijos tienen valores, indica que ese nodo es un operador, entonces ejecuta la operación correspondiente. Las operaciones se definen de forma similar, solo que evalúa si cada hijo para saber si es hoja, en caso de serlo lo asigna como valor y realiza la operación correspondiente y devuelve el valor de esta.

Precondiciones:

Dado un AB lleno.

El valor inicial debe ser 0

No puede un nodo numérico ser hijo de otro nodo numérico.

Postcondiciones:

El resultado de la expresión debe ser el valor de las operaciones realizadas según el árbol dado.

El árbol original no debe ser modificado

Pseudo:

calcularOperaciones() devuelve: resultado entero

Comienza

Si hijoIzquierdo <> nulo ENTONCES

hijoIzquierdo.calcularOperaciones()

FINSI

Si hijoDerecho <> nulo ENTONCES

hijoDerecho.calcularOperaciones()

FINSI

SI hijoIzquierdo <> nulo y hijoDerecho <> nulo ENTONCES

SI este.dato comparo con Operadores (+, -, /, \*)

Operación()

FINSI

Devuelvo resultado

FIN

Operación()

Entero a 🡨 0

Entero b 🡨 0

Si hijoIzquierdo <> nulo ENTONCES

Si hijoIzquierdo.esHoja() ENTONCES

a 🡨 este.dato

FINSI

FINSI

Si hijoDerecho<> nulo ENTONCES

Si hijoDerecho.esHoja() ENTONCES

b 🡨 este.dato

FINSI

FINSI

Entero operación 🡨 a (+, -, /, \*) b

hijoDerecho 🡨 nulo

hijoIzquierdo 🡨 nulo

este.dato 🡨 operacion

FIN

Texto

Descripción generada automáticamente

Lenguaje Natural

Dada una expresión aritmética compuesta por valores enteros, operadores y paréntesis, pasada como una cadena de caracteres.

Iniciar un contador que sume cada vez que encuentre un paréntesis abierto y reste cuando encuentre uno cerrado. Recorro la cadena y mientras el valor del contador sea 0 evaluó los caracteres hasta encontrar un operador, entonces divido la cadena desde el inicio hasta el operador y la asigno a una variable de cadena llamada izquierdo y los valores de la cadena después del operador encontrado los asigno a otra variable de cadena llamada derecho.

Ya con izquierdo, operador y derecho lo que hago es asignar al nodo actual el valor de operador y a sus hijos respectivos.

Luego cada hijo si la expresión que tiene asignada es distinta de nula realiza la llamada recursiva.

Precondiciones.

El árbol debe estar vacío.

La entrada puede debe ser aritmética y correcta.

postcondiciones.

Se deben insertar la expresión acorde la asociación de los paréntesis.

Pseudo:

insertarExpresion(String expresión) Devuelve: Árbol Binario

Comienza

String op = "";

pasa 🡨 false

String derecho 🡨 ""

String izquierdo 🡨 ""

entero cont 🡨 0

PARA CADA elemento de la expresion HACER

Si contador = 0 y esOperador() entonces

Op 🡨 elemento

Izquierodo 🡨 expresion desde 0 hasta elemento -1

Derecho 🡨 expresion desde elemento +1 hasta final

SALIR

FINSI

Este.dato 🡨op

Este.etiqueta 🡨op

hijoDerecho 🡨 derecho

hijoIzquierdo 🡨 izquierdo

FIN PARA

SI hijoDerecho <> nulo ENTONCES

hijoDerecho.insertarExpresion()

FINSI

SI hijoIzquierdo <> nulo ENTONCES

hijoIzquierdo.insertarExpresion()

FINSI

FIN

esOperador(carácter carácter) devuelve verdadero o falso

Comienza

Comprueba carácter con +, -, /, \*

Devuelve verdadero si es operador

Devuelve falso si no es operador

FIN