Ejercicios de Programación - Sebesta

Lenguajes de Programación - ESPOL

10 de febrero de 2014

1. Introducción

Las respuestas propuestas en este repositorio son correspondientes a las preguntas del libro de Robert Sebesta, Concepts of Programming Languages.

2. Preguntas y Respuestas

2.1. Capítulo 5: Nombres, Enlaces y Alcances.

2.2. Capítulo 7: Expressions and Assignment Statements

- **Pregunta 3**

  Escribe un programa en tu lenguaje favorito que determine y muestre la precedencia y asociatividad de sus operadores aritmeticos y booleanos.
  Lenguaje Java

```
public static void main(String[] args) {

        float a,b,c,d,e,res1,res2, res3;
        res1=res2=res3=0;

        a=8;
        b=4;
        c=3;
        d=5;
        e=0;
        res1=a/b*c+d;
        res2=a*b/c+d;
```

```
        if(e!=0){
            res3=e*b*(c+d);
        }
        System.out.println("El resultado 1 es: "+res1);
        System.out.println("El resultado 2 es: "+res2);
        System.out.println("El resultado 3 es: "+res3);
}
```

Se sabe que en Java: La Exponenciación tiene precedencia 1(mayor), cambio de signo(-) e identidad(+) tienen precedencia 2, division y multiplicacion tienen precedencia 3, la suma y resta tienen precedencia 4 ,etc.
Los operadores también pueden tener la misma precedencia. En este caso, la asociatividad determina el orden en que deben actuar los operadores.
La asociatividad puede ser de izquierda a derecha o de derecha a izquierda.
En el resultado 1(res1), primero se resuelve los operadores de mayor precedencia. En este caso, tenemos la division y multiplicacion de igual precedencia. Como ambas tienen la misma precedencia y asociatividad desde la izquierda, lo que significa que los operadores de la izquierda se procesan (division)antes que los operadores de la derecha(multiplicacion), quiere decir:
Primero, realiza a/b, luego ese resultado * c, y finalmente el resultado de la multiplición se le suma d.

En el resultado 2 (res2), como (/,*) tienen misma precedencia, primero se resuelve el operador de la izquierda (*), luego la división. Finalmente la suma de menor precedencia

En el resultado 3, se hace una evaluación de corto circuito. Cada vez que e sea diferente de 0, realiza la operación; caso contrario es 0

OUTPUT
El resultado 1 es: 11.0
El resultado 2 es: 15.666667
El resultado 3 es: 0.0

- **Pregunta 4**

Escribir un programa en Java que exponga la regla para el orden de evaluación de operando cuando uno de los operandos es un método call.

Lenguaje Java

```
public class JavaApplication2 {
    final static int num=5;
    static int a=5;
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

          a = fun1()+a;
          System.out.println(a);
    }

    static int fun1() {
           a = 17;
           return 3;
       }

}
```

OUTPUT
20
Como podemos ver en el main en la 1era linea, fun1() retorna 3 y actualiza la variable global 'a'=17 y luego suma 17+3 asignando 20 a 'a'. Esto se debe a que en JAVA el operador '+' tiene asociatividad desde la izquierda.
En cambio si fuese: a = a + fun1();. En este caso primero 'a'=5 + fun1() que retorne 3; dando una asignación de 8 a la variable a.

- **Pregunta 5**

Lenguaje C++

```
int fun1();

extern int a = 10;
void main(){

         a = fun1()+a;   // a = a+fun1();
         printf("%d", a);
         getch();
```

```
}

int fun1() {
        a = 17;
        return 3;
}
```

OUTPUT
20
Como podemos ver en el main en la 1era linea, comparado con JAVA Y Si Shard en vez de que fun1() retorne 3 y actualize la variable global 'a' a 17,y luego sumarlos y asignarle 20 a 'a'. En C++, sea a = fun1()+a ó a = a+fun1(), llamado de función en la izq o derecha del operador en este caso '+'; siempre al llamar fun1(), este va actualizar la variable global a =17 y luego va sumarle 3, asignando un valor de 20 a la variable 'a'.

- **Pregunta 6**

Lenguaje Si Sharp

```
class Program
    {
        static int a = 5;
        static void Main(string[] args)
        {
            a = a +fun1();
            Console.WriteLine(a);
            Console.ReadLine();
       }

        static int fun1()
        {
            a = 17;
            return 3;
        }
    }
```

OUTPUT
8
Como podemos ver en Si Sharp también se cumple la regla de la asociativad. En este caso el operador '+' asoicativdad desde la izquierda. Primero a=5,

luego le suma 3; cuyo resultado que es 8 se le asigna a la variable a.
En cambio si fuese: a = fun1()+a. En ese caso fun1() retorna 3 y actualiza la variable global 'a'=17 y luego suma 17+3 asignando 20 a la varaible a.

2.3. Capítulo 9: SubProgramas.