Ejercicios de Programación - Sebesta

Lenguajes de Programación - ESPOL

11 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.

- **Pregunta 5**

  Escriba una función en C que incluya las siguienetes secuencia de sentencias: x=21; int x; x=42; Ejecute el programa ye explique los resultados. Reescriba el código en C++ y Java, compare los resultados. Lenguaje C

```
void main()
 {
        x=21;
        int x;
        x=42;
}
```

  Error: error C2065: 'x' : identificador no declarado

  Lenguaje C++

```
void main(void)
 {
        cout << "Ingresa 21:";
        cin >> x;
```

```
        int x;
       x=42;
       cout << "X vale:" << x;
       cin.get();
}
```

Error: error C2065: 'x' : identificador no declarado

Lenguaje Java

```
public static void main(String[] args) {
   x = 21;
   int x;
   x = 42;
}
```

Error: Exception in thread "main"java.lang.RuntimeException: Uncompilable source code - cannot find symbol-symbol: variable x

- **Pregunta 6**

Escribir un programa en C++, Java, SI SHARP para determinar el alcance de una variable declarada en un For statement. Especialmente el código debe determinar si una variable es visible después del cuerpo del statement FOR.

Lenguaje Java

```
public class JavaApplication2 {
        for(int i = 0; i< 10 ; i++)
       {
               System.out.println(i);
       }

               System.out.println(i);
               //Trata de acceder a la variable i

}
```

En este ejemplo podemos darnos cuenta que el println puede imprimir cada iteración de la variable i; pero hacer println de i fuera del for no es posible. Esta función intenta acceder a 'i' pero no puede ya que 'i' es declarada dentro for, no es una varialbe global.
Cannot find symbol
symbol: variable i

Lenguaje C++

```
void main(){
        for (int i = 0; i<10; i++){
                printf("%d\n", i);

        }

        printf("%d\n", i);
        //Trata de acceder a la variable i

        getch();
}
```

Fuera del FOR, printf intenta acceder a 'i' pero no puede ya que 'i' es declarada dentro for, no es una varialbe global.

Lenguaje SI SHARP

```
namespace ConsoleApplication2
{

    class Program
    {
         static void Main(string[] args)
        {
            for(int i = 0; i< 10 ; i++){
                Console.WriteLine(i);
        }
```

```
            Console.WriteLine(i);
          //Intenta acceder a la variable i
        Console.ReadLine();
        }
        }
}
```

OUTPUT
Error
El nombre i no existe en el actual contexto
Fuera del FOR, Console.WriteLine(i) intenta acceder a 'i' pero no puede ya que 'i' es declarada dentro for, no es una varialbe global.

- **Pregunta 7**

Escriba funciones en C o C++, una que declare un largo arreglo de forma estática, otra que declare el mismo arreglo largo en un stack y otra que declare el mismo arreglo desde el heap. Llame a cada subprograma un gran numero de veces(al menos 100000) e imprime el tiempo de ejecución requerido para cada uno. Lenguaje C

```
void main()
{
        clock_t t1,t2,t3,t4;
        int i;

        t1=clock();
        for(i=0;i<100000;i++)
        {
                funcion1();
                //funcion2();
                //funcion3();
        }
        t2=clock();
        printf("El Tiempo es: %f\n",
        (t2-t1)/(double) CLOCKS_PER_SEC);
}
int funcion1()
{
```

```
        int i;
        int A[256];
        for(i=0;i<256;i++)
                A[i] = i;
        return 0;
}
int funcion2()
{
        int *a;
        int i;
        a = new int[256];
        for(i=0;i<256;i++)
                a[i] = i;
        return 0;
}
```

Output
-El tiempo es:0.156000
-El tiempo es:0.807000
Al observar los tiempos de ejecución, vemos que el primero es menor que el segundo, correspondiente a la función del arreglo estático que nos indica que el tiempo en llevar ese arreglo a la memoria prima es muy corto a diferencia del segundo que se lo realizo en el heap.

2.2. Capítulo 6: 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.