Ejercicios de Programación - Sebesta

Lenguajes de Programación - ESPOL

13 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 y 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, C share 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 C share

```
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();
        }
        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 7: Expressions and Assignment Statements

- **Pregunta 1**

Ejecute el siguiente código en algún sistema que soporte C para determinar los valores de sum1 y Sum2. Explicar la resultados..
Codigo ya pasado a C

```
        int fun(int *k) {
        *k += 4;
        return 3 * (*k) - 1;
```

```
        }
//Suppose fun is used in a program as follows:
        void main() {
        int i = 10, j = 10, sum1, sum2;
        sum1 = (i / 2) + fun(&i);
        sum2 = fun(&j) + (j / 2);
        getch();
        }
```

OUTPUT
El resultado 1de sum1 es: 46, y el de sum2 es 48
Esto se da a que se evaluan las expresiones de izquierad a derecha , por lo que en sum1 primero se realiza i/2 y despues la funcion fun modifica el valor de la variable i y devuelve 41 , caso contrario en sum2 que primero modifica fun el valor de j y despues a este valor modificado se lo divide para 2 , de ahy que siendo las mismas expresiones pero con orden diferente devulevan valores distintos

- **Pregunta 2**

  Modificar el programa del Ejercicio 1 en C + +, Java y C #, ejecutarlos, y comparar los resultados.

  Codigo en C++

```
\#include <iostream>

using namespace std;
int  fun(int *a);
int main()
{
    int a=10,b=10,c=10;
    int sum1=a/2 + fun(&a);
    int sum2=fun(&b)+ b/2;

   cout << sum1 << endl;
   cout << sum2 << endl;

   return 0;
}
```

```
int fun(int *a){
*a +=4;
return 3 * (*a) - 1;
}
```

OUTPUT
El resultado de sum1 es: 46, y el de sum2 es 48

Codigo en C#

```
using System.IO;
using System;

class Program
{
    int a;
    public Program(int a1){
        a=a1;

    }
    static void Main()
    {
        int a=10;
        int b=10;
        Program P=new Program(a);
        Program P1=new Program(b);
        int sum1=P.a/2+P.fun(P);
        int sum2=P1.fun(P1)+P1.a/2;

        Console.WriteLine(sum1);
        Console.WriteLine(sum2);
    }
    public int fun(Program a){
        a.a+=4;
        return 3*a.a-1;

    }
}
```

OUTPUT
El resultado de sum1 es: 46, y el de sum2 es 48

Codigo en Java

```
public class NewClass {

 public int a=10,b=10,c=10;
    public static void main(String[] args) {
        NewCla numeroa=new NewCla(10);
        NewCla numerob=new NewCla(10);
        int sum1,sum2;
        sum1=numeroa.a/2+numeroa.fun(numeroa);
        sum2=numerob.fun(numerob)+numerob.a/2;

        System.out.print(sum1 + ";"    +sum2);
    }

}
class NewCla {
int a;
public NewCla(int a1){
    a=a1;
}
int fun(NewCla a){
a.a+=4;
return 3 * (a.a) - 1;
}
}
```

OUTPUT
El resultado de sum1 es: 46, y el de sum2 es 48

Como podemos observar en todos los casos dieron los mismos resultaos , quier decir que los tres programas dan prioridad a evaluar primero de izquierda a derecha.

- **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 C Share

```
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.

- **Pregunta 9**

  Escriba un programa, ya sea en Java, C + +, C # que lleve a cabo un gran número de operaciones de punto flotante y un número igual de operaciones con enteros y comparar el tiempo requerido..

  Lenguaje Java

```
public class Ejercicio9 {
   void enteros(){
        int a = 10;
        int b=10;
        int resultado=0;
        for (double i=0;i<900000000;i++){
        resultado +=a+b;
        resultado -=a;
        resultado -=b;
        resultado -=b+a;
```

```
        resultado +=a+b;
        resultado +=a+(4*b);

        }
        System.out.print(resultado);
     }
    void flotantes(){
    float a = (float) 10.0;
        float b=(float) 10.0;
        float resultado=0;
        for (double i=0;i<900000000;i++){
        resultado +=a+b;
        resultado -=a;
        resultado -=b;
        resultado -=b+a;
        resultado +=a+b;
        resultado +=a+(4*b);

        }
        System.out.print(resultado+"\n");
    }
    public static void main(String[] args) {
     Ejercicio9 e=new Ejercicio9();
     e.flotantes();
     e.enteros();


     }
}
```

El tiempo realizando las operaaciones con int fue :6 segundos , mientras que con los flotantes fue : 10 segundos. Se llega a la conclusion que por ser los foltantes de mayor precion que los int a la maquina le toam mayor trabajo realizar esas opraciones , debidoa eso la diferencia de tiempo.

2.3. Capítulo 8: Statement-Level Control Structures.

- **Pregunta 4**

Considere el sgte segmento de un programa en C. Rescribalo sin usar go-

tos o breaks

Lenguaje C

```
for (i = 0; i < 3; i++) {
switch (j + 2) {
case 3:
case 2: j--; break;
case 0: j += 2; break;
default: j = 0;
}
if (j > 0) break;
j = 3 - i
}
```

Resolución:

```
void funcion()
{
        int i;
        int j = -3;
        for (i = 0; i < 3; i++)
        {
                switch (j + 2)
                {
                        case 3:
                                return;
                        case 2:
                                {j--;
                                return;}
                        case 0:
                                {j += 2;
                                return;}
                        default:
                                {j = 0;
                                return;}
                }
                if (j > 0) return;
                j = 3 - i;
        }
```

```
}
```

Otra forma de romper un lazo for y un switch es con una sentencia de return.

2.4. Capítulo 9: SubProgramas.

- **Pregunta 5**

  Write a program in some language that has both static and stackdynamic local variables in subprograms. Create six large (at least 100 * 100) matrices in the subprogram—three static and three stack dynamic. Fill two of the static matrices and two of the stack-dynamic matrices with random numbers in the range of 1 to 100. The code in the subprogram must perform a large number of matrix multiplication operations on the static matrices and time the process. Then it must repeat this with the stack-dynamic matrices. Compare and explain the results.

  Lenguaje Java

```
public class Lenguajes {
        int tam =100;

        public static int[][] m1 = new int[tam][tam];
        public static int[][] m2 = new int[tam][tam];
        public static int[][] mr = new int[tam][tam];

        public static void main(String args[])
        {

                long ini = System.nanoTime();
                this.ejecutarRandom();
                long fin = System.nanoTime();
                long t1 = fin-ini;
                System.out.println(t1);


                        long ini2 = System.nanoTime();
                this.operacionesD();
                long fin2 = System.nanoTime();
```

```
            long t2 =fin2-ini2;
                    System.out.println(t2);
        }


        void ejecutarRandom(int[][] m1, int[][] m2){
            for(int i=0;i<tam;i++){
                for(int j=0;jl<tam;jl++){
                        m[i][j] = (int)(Math.random()*tam)+1;
                        m[i][j] = (int)(Math.random()*tam)+1;

                }
            }
        }

        void operacionesDinamic(){
            int[][] dinamic_m1 = new int[tam][tam];
            int[][] dinamic_m2 = new int[tam][tam];
            int[][] dinamic_ml = new int[tam][tam];

            this.matricesRandom(dinamic_m1, dinamic_m2);

             for(int i = 0; i <tam; i++){
                for (int j = 0; j < tam; j++){
                    for (int k = 0; k <tam; k++){
                            dinamic_ml[i][j] +=
                              dinamic_m1[i][k] * dinamic_m2[k][j];
                    }
                }
            }

        }


 }
```

Output:
T1: 0.155000 T2: 0.140000 El tiempo obtenido de forma dinamica es más bajo que el obtenido de forma estática.