Ejercicios de Programación - Sebesta

Lenguajes de Programación - ESPOL

12 de febrero de 2014

1. Introducción

Las respuestas propuestas en este repositorio son producto del trabajo de los estudiantes de la materia "Lenguajes de Programación" de la ESPOL, 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

2.1.1. Pregunta 4: Escriba un script en Python con subprogramas triplemente anidados, donde cada subprograma referencie a variables que han sido definidas en otros niveles de la jerarquía

Escriba su respuesta con claridad. En las secciones de código utilice listings.

2.1.2. Pregunta 5: Escriba una función en C que incluya la siguiente secuencia de declaraciones:

x = 21;
int x;
x = 42;
Ejecute el programa y explique los resultados. Reescriba el mismo códifgo en C++ y Java y compare los resultados.

Escrito en C:

```
int x = 21;
funcion_Prueba(x);
printf("%d",x);
```

```
        getch();
}


void funcion_Prueba(int numero){
        int x;
        x = 42;
        numero=x;
}
```

RESULTADOS: 21
Como se puede apreciar, el valor de la variable x se mantiene en 21, ser pasada como argumento en la función funciónPrueba no alteró su valor a pesar de que ocurre una igualación a 42 en numero=x; esto se debe a que en C la variable ésta siendo pasada por valor, por lo cual se crea una copia ésta.
Ahora veamos como reaccionán Java y C++:

Escrito en Java:

```
        public static void main(String[] args){
                int x=21;
                funcion_Prueba(x);
                System.out.printf("%d",x);
        }
        public static void funcion_Prueba(int numero){
                int x;
                x = 42;
                numero=x;
        }
}
```

RESULTADOS: 21

Escrito en C++:

```
        int x=21;
        funcion_Prueba(x);
        std::cout<<x;
        getch();
        return 0;
}
```

```
void funcion_Prueba(int numero){
        int x;
        x = 42;
numero=x;
}
RESULTADOS: 21\\\
Ambos resultados son iguales, esto era de esperarse ya que tanto Java y
```

2.1.3. Pregunta 6: Escriba programas de prueba en C + +, Java, C # y para determinar el alcance de una variable declarada en una sentencia for. Específicamente, el código debe determinar si una variable es visible después del cuerpo de la declaración for.

Java: Este ejemplo llena un arreglo de 10 elementos con los números del 1 al 10, usando la variable uno declarada en el cuerpo del for, luego se pretende imprimir dicha variable, en un lugar fuera del for.

```
        public static void main(String[] args){
                int arreglo[];
                arreglo=new int[10];
                for(int i=0;i<10;i++){
                        int uno=1;
                        arreglo[i]=uno+i;
                }
                System.out.printf("%d",uno);
        }
}
}
```

RESULTADOS:
Prueba.java:12: error: cannot find symbo
l symbol: variable uno
location: class Prueba
1 error

Error de compilación, la variable no es visible esta fuera del alcance.

En C++
Usaré el mismo ejemplo anterior, para probar:

```
        int i;
        int arreglo[10];

        for(i=0;i<10;i++){
                int uno=1;
                arreglo[i]=uno+i;
        }
        std::cout<<uno;
        getch();
        return 0;
}
```

RESULTADOS:
error C2065: 'uno' : identificador no declarado
De la misma manera, ocurre un error de compilación porque la variable no es visible.

Ejemplo en C # :

```
{
    class Program
    {
        static void Main(string[] args)
        {
            int i;
            int[] arreglo = new int[10];

                for(i=0;i<10;i++){
                        int uno=1;
                        arreglo[i]=uno+i;
                }
            Console.Write(uno);
            Console.ReadKey();
        }
    }
}
```

RESULTADOS:
Error de compilación: El nombre 'uno' no existe en el contexto actual

En los tres lenguajes sucede que la variable que es declarada dentro de un for no es visible en otro sitio.

2.2. Capítulo 7: Expresiones y asignaciones

2.2.1. Pregunta 2: Reescriba el programa de Programming Exercise 1 en C++, Java, y C #, ejecútelos, y compare los resultados.

Ejemplo escrito en C++:

```
        int i = 10, j = 10, sum1, sum2;
        sum1 = (i / 2) + fun(&i);
        sum2 = fun(&j) + (j / 2);
        std::cout<<sum1;
        std::cout<<sum2;
        getch();
        return 0;
}

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

}
```

RESULTADOS:
sum1 = 46 sum2 = 48
Los operandos son evaluados de izquierda a derecha, en el caso de sum1 el resultados es 46 porque primero de evalúa 'i/2' igual a 5, y luego se evalúa la función la cuál retorna 41 porque el valor de la variable i sigue siendo 10. Por otro lado sum2 toma el valor de 48 porque primero se evalúa 'fun(&j)' esto retorna 41, pero el valor de 'j' cambia a 14, y al efectuar 'j/2' retorna 7, y al sumar da 48.

En java:

```
        public static void main(String[] args){
                int sum1,sum2;
                Integer i,j;
                i=new Integer(10);
                j=new Integer(10);
```

```
                sum1=(i.intValue()/2) + fun(i);
                sum2=fun(j)+ (j.intValue()/2);
                System.out.printf("%d\n%d",sum1,sum2);
                }
        public static int fun(Integer k){
                k=k.intValue()+4;
                return 3*(k.intValue())-1;

        }
}
```

RESULTADOS:
sum1 = 46 sum2 = 46

Con este código no se puede concluir si los operandos son evaluados de izquierda a derecha o viceversa porque sum1 y sum2 toman el mismo valor 46. Pero se puede concluir que una referencia a un objeto es pasado por valor en una función, porque el valor de las variables 'i' y 'j' no se vió afectada.

En C #:

```
namespace PruebaCchar
{
    class Program
    {
        unsafe static void Main(string[] args)
        {
            int i = 10, j = 10, sum1, sum2;
            sum1 = (i / 2) + fun(&i);
            sum2 = fun(&j) + (j / 2);
            Console.Write(sum1);
            Console.Write(sum2);
            Console.ReadKey();
        }

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

RESULTADOS:
sum1 = 46 sum2 = 48
Los operandos se evalúan de izquierda a derecha, tiene el mismo comportamiento que C++

2.3. Capítulo 8: Expresiones y asignaciones

2.3.1. Pregunta 4: Considere el siguiente segmento de un programa C. Reescríbalo sin usar gotos o breaks.

```
j = -3;
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;
}
```

Reescrito:

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


        printf("%d\n%d",j,i);
}
```

Al inicio los valores de 'i' y 'j' fueron 1 y 3 respectivamente, al borrar todos los break los valores cambiaron a 0 y 3 respectivamente, Porque en C, al no incluir un break al terminar un case, continuara con el otro case. Es un muye sencillo darse cuenta de porque sucede esto, basta con hacer una prueba de escritorio.

2.3.2. Pregunta 5: En una carta al editor de CACM, Rubin (1987) se utiliza el siguiente segmento de código como evidencia de que la legibilidad de algún código con gotos es mejor que el código equivalente sin gotos. Este código busca la primera fila de una matriz de enteros de n por n llamada X que tiene nada más que valores cero; Reescribe este codigo sin gotos en uno de los siguientes lenguajes: C, C++, Java, C #, o Ada. Compare la legibilidad de su código con la del código del ejemplo.

```
for (j = 1; j <= n; j++)
if (x[i][j] != 0)
goto reject;
println ('First␣all-zero␣row␣is:', i);
break;
reject:
}
```

Reescrito e Java:

```
for (int j = 1; j <=n; j++){
if( x [ i ] [ j ] !=0 ) {
System.out.println(  i );
break;
}
}
}
```

Comparando la legibilidad, la segunda opción me parece más legible porque en la primera se dificulta un poco la lectura del flujo del programa, en la segunda opción se puede leer más fácilmente.