

# Ejercicios de Programación - Sebesta

## Lenguajes de Programación - ESPOL

13 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

- **Pregunta 4:** Write a Python function that has subprograms nested three deep and in which each nested subprogram references variables defined in all of its enclosing subprogram

```
def funcion0():  
    def funcion1():  
        nombre1="hola soy 1"  
        def funcion2():  
            nombre2="hola soy 2"  
            def funcion3():  
                nombre3="hola soy 3"  
                funcion3()  
            print("%s en f2"%(nombre3))  
        funcion2()  
        print("%s en f1"%(nombre2))  
        print("%s en f1"%(nombre3))  
    funcion1()  
    print("%s en f0"%(nombre1))  
    print("%s en f0"%(nombre2))  
    print("%s en f0"%(nombre3))  
funcion0()
```

```
C:\Users\NiAS>C:\Users\NiAS\Documents\Python\test3.py  
Traceback (most recent call last):  
  File "C:\Users\NiAS\Documents\Python\test3.py", line 17, in <module>  
    funcion0()  
  File "C:\Users\NiAS\Documents\Python\test3.py", line 13, in funcion0  
    funcion1()  
  File "C:\Users\NiAS\Documents\Python\test3.py", line 10, in funcion1  
    funcion2()  
  File "C:\Users\NiAS\Documents\Python\test3.py", line 9, in funcion2  
    print("%s en f2"%(nombre3))  
NameError: global name 'nombre3' is not defined  
C:\Users\NiAS>
```

NO EJECUTADO EN PYTHON: Variables no son declaradas globales

- **Pregunta 5:** Write a C function that includes the following sequence of statements: `x = 21;int x;x = 42;` Run the program and explain the results. Rewrite the same code in C++ and Java and compare the results.

- Función en C

```
#include<conio.h>
#include<stdlib.h>

int main(){

    x = 21;
    int x;
    x = 42;

}
```

NO COMPILA: error C2065: 'x' : identificador no declarado

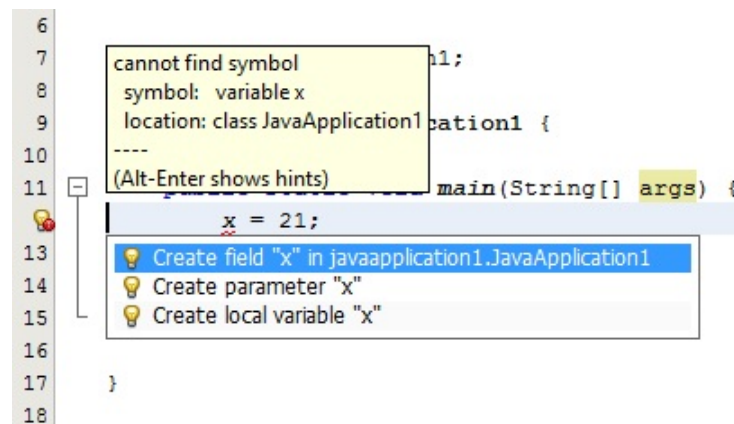
ARGUMENTO: En el lenguaje C es necesario crear la variable o instanciarla de las siguientes formas antes de poder usarla o asignarle un valor:

- (tipo) (nombreVariable);  
Ejemplo: int x;
- (tipo) (nombreVariable)=(valor Inicial);  
Ejemplo: int x=10;

- Función en C++

Sucede lo mismo que en C, el compilador de visual presenta los mismos errores, como si las tres líneas estuvieran mal escritas, aunque realmente es la primera.

- Función en Java



NO COMPILA: error: cannot find symbol

ARGUMENTO: En el lenguaje Java, el intérprete antes de compilar sugiere al programador crear la variable X como una variable de clase, y una vez compilado mostrar el error de que no puede encontrar el simbolo que se esta intentado usar, en este caso X.

- **Pregunta 6: Write test programs in C++, Java, and C# to determine the scope of a variable declared in a for statement. Specifically, the code must determine whether such a variable is visible after the body of the for statement.**

- Función en C++

```
#include<stdio.h>

int main(){
    int i=5;
    int length=0;
    for (i = 0; i < 5; i++)
    {
        int cont;
        length++;
        cont=length+5;
    }
    printf("%i",cont);
}
```

Error: el identificador "cont" no está definido

NO COMPILA: No es visible despues de la sentencia for.

- Función en C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            int i=100;
            for (int j = 0; j < 5; j++)
            {
                int holamundo= 5;
                holamundo++;
            }
            Console.WriteLine("Hello World! {1}",holamundo,i);
            Console.WriteLine("Press any key to e");
            Console.ReadKey();
        }
    }
}
```

El nombre 'holamundo' no existe en el contexto actual

NO COMPILA: No es visible despues de la sentencia for.

Es de notar que los avisos que muestra el intérprete son diferentes en los dos casos anteriores, en C++ nos dice que no existe definición de la

variable que se quiere usar y por tanto no hay ningún valor que mostrar. En la siguiente, en C# en cambio nos informa que no estamos en el mismo contexto y que por tanto la variable no es visible ni alcanzable.

- **Pregunta 7: Write three functions in C or C++: one that declares a large array statically, one that declares the same large array on the stack, and one that creates the same large array from the heap. Call each of the subprograms a large number of times (at least 100,000) and output the time required by each. Explain the results.**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include <time.h>

void funcion1();
void funcion2();
void funcion3();

int main(){
    clock_t start = clock();
    int i;
    for(i=0;i<100000;i++){
        funcion1();
    }
    printf("Tiempo transcurrido: %f", ((double)clock() - start) / CLOCKS_PER_SEC);
    getch();
}

void funcion1(){
    static int array1[100]={0};
}

void funcion2(){
    int array2[100]={0};
}

void funcion3(){
    int* array3;
    array3 = (int*)malloc (100*sizeof(int));
    free(array3);
}
```

Tiempo transcurrido: 0.002000

Tiempo transcurrido: 0.005000

Tiempo transcurrido: 0.214000

EN C: Diferentes tiempos para diferentes funciones

ARGUMENTO: Las variables y vectores en C ocupan un tamaño fijo, el cuál es establecido antes de ejecutar el programa, por tanto no pueden variarlo durante esta ejecución. Por medio de punteros se puede reservar o liberar memoria dinámicamente, es decir, según se necesite. La Función malloc sirve para solicitar un bloque de memoria del tamaño suministrado como parámetro. Es notable que la petición MALLOC al querer reservar memoria toma más tiempo en su ejecución, pero también es importante entender que es una forma segura de programar, ya que estaremos seguros de que obtendremos la memoria que necesitemos para las variables que usemos, sin tener errores de memoria en tiempo de ejecución. Así que aunque en la imagen veamos que toma más tiempo ejecutar MALLOC es preferible su uso, para no arriesgarnos a errores de memoria.

## 2.2. Capítulo 6: Tipos de Datos

- **Pregunta 1** OJ000000000000 faltaaaaaaaaaaaaaaaaaaaaaa
- **Pregunta 2** OJ000000000000 faltaaaaaaaaaaaaaaaaaaaaaa
- **Pregunta 7** OJ000000000000 faltaaaaaaaaaaaaaaa

## 2.3. Capítulo 7: Expresiones e Instrucciones de asignación

- **Pregunta 1** OJ000000000000 faltaaaaaaaaaaaaaaaaaaaaaa
- **Pregunta 2** OJ000000000000 faltaaaaaaaaaaaaaaaaaaaaaa
- **Pregunta 3** OJ000000000000 faltaaaaaaaaaaaaaaaaaaaaaa
- **Pregunta 4** OJ000000000000 faltaaaaaaaaaaaaaaaaaaaaaa
- **Pregunta 5** OJ000000000000 faltaaaaaaaaaaaaaaaaaaaaaa
- **Pregunta 6** OJ000000000000 faltaaaaaaaaaaaaaaaaaaaaaa
- **Pregunta 9** OJ000000000000 faltaaaaaaaaaaaaaaaaaaaaaa

## 2.4. Capítulo 8: Estructuras de Control

- **Pregunta 3** OJ000000000000 faltaaaaaaaaaaaaaaaaaaaaaa
- **Pregunta 4** OJ000000000000 faltaaaaaaaaaaaaaaaaaaaaaa
- **Pregunta 5** OJ000000000000 faltaaaaaaaaaaaaaaaaaaaaaa

## 2.5. Capítulo 9: SubProgramas

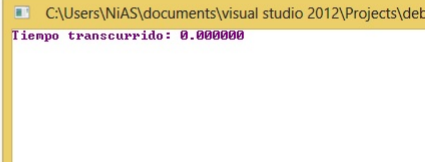
- **Pregunta 1:** Write a program in a language that you know to determine the ratio of the time required to pass a large array by reference and the time required to pass the same array by value. Make the array as large as possible on the machine and implementation you use. Pass the array as many times as necessary to get reasonably accurate timings of the passing operations.

```

int main(){
    clock_t start = clock();
    int array[100000]={1};
    Referencia(array);
    printf("Tiempo transcurrido: %f", ((double)clock() - start) / CLOCKS_PER_SEC);
    getch();
}

int Referencia(int *r)
{
    int i;
    for(i=0; i<100000; i++){
        r[i]=5;
    }
}

```



POR REFERENCIA: Es notable que es más eficiente la ejecución al pasar un arreglo por referencia.


```

int main(){
    clock_t start = clock();
    int array[100000]={1},i;

    for(i=0; i<100000; i++){
        array[i]=Valor(array[i]);
    }
    printf("Tiempo transcurrido: %f", ((double)clock() - start) / CLOCKS_PER_SEC);
    getch();
}

int Valor(int v)
{
    v=5;
    return v;
}

```



POR VALOR: Se modificó la función para que hiciera lo mismo que la anterior, y notamos un cambio en los tiempos pero pasando cada elemento del arreglo por valor.