CAPITULO 5

- **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.

Código 0.1: Código en lenguaje C.

```
1
2 int main(){
3     x = 21;
4     int x;
5     x = 42;
6
7     return 0;
8 }
```

cap5.c: En la función 'main':
cap5.c:2:2: error: 'x' no se declaró aquí (primer uso en esta función)
cap5.c:2:2: nota: cada identificador sin declarar se reporta sólo una vez para cada función en el que aparece

Código 0.2: Código en lenguaje C++.

```
1
2 int main(){
3     x = 21;
4     int x;
5     x = 42;
6
7     return 0;
8 }
```

cap5.cpp: En la función 'int main()':
cap5.cpp:2:2: error: 'x' no se declaró en este ámbito

Código 0.3: Código en lenguaje Java.

```
1
2 public class cap5{
3     public static void main(string args[]){
4         x = 21;
5         int x;
```

```
6           x = 42;
7       }
8   }
```

cap5.java:3: error: cannot find symbol x=21;
symbol: variable x
location: class cap5
1 error

- **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.

Código 0.4: Código en lenguaje C++.

```
1
2   #include <iostream>
3
4   using namespace std;
5
6   void main(){
7       for(int i=1; i<=5; i++)
8           cout << i;
9
10      // Intento acceder a la variable "i"
11      cout << i;
12  }
```

Aqui se intenta imprimir el valor de la variable i, pero nos da un error de compilacion, ya que la variable solo existe dentro del lazo "for".

Código 0.5: Código en lenguaje Java.

```
1
2   public class cap6{
3       public static void main(String args[]){
4           for (int i=1; i<=5; i++)
5               System.out.println(i);
6
7           // Intento acceder a la variable "i"
8           System.out.println(i);
9       }
10  }
```

cap6.java:7: error: cannot find symbol System.out.println(i); symbol: variable i location: class cap6

Aqui nos dice que no se encuentra la variable i, esto se debe a que solo existe dentro del lazo "for".

Código 0.6: Código en lenguaje C#.

```
1
2  using System;
3
4  namespace cap6
5  {
6      class cap6
7      {
8          static void Main(string[] arg)
9          {
10             for(int i=1; i<=5; i++)
11                 Console.WriteLine(i);
12
13             // Intento acceder a "i"
14             Console.WriteLine(i);
15         }
16     }
17 }
```

cap6.cs(13,43): error CS0103: The name ‘i’ does not exist in the current context
Este mensaje nos dice que la variable ï"no existe fuera del lazo "for", ya que no es una variable global

- **Pregunta 7**

Write three functions in C or C++: one that declares a large array stati- cally, 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.

Código 0.7: Código en lenguaje C#.

```
1
2  #include <iostream>
3  #include <stdio.h>
4  #include <time.h>
5
6  int stack();
7  int heap();
8
9  int main()
10 {
11     int i;
12     clock_t time1,time2;
```

```
13
14      time1=clock();
15      for(i=0;i<100000;i++)
16          stack();
17
18      time2=clock();
19      printf("El tiempo en el stack es: %f\n",
           (time2-time1)/(double) CLOCKS_PER_SEC);
20
21      time1=clock();
22      for(i=0;i<100000;i++)
23          heap();
24
25      time2=clock();
26      printf("El tiempo en el heap es: %f\n",
           (time2-time1)/(double) CLOCKS_PER_SEC);
27
28      return 0;
29  }
30
31  int stack(){
32      int i;
33      int M[1000];
34      for(i=0;i<1000;i++)
35              M[i] = i;
36      return 0;
37  }
38
39  int heap(){
40      int *M;
41      int i;
42      M = new int[1000];
43      for(i=0;i<1000;i++)
44              M[i] = i;
45
46      delete M;
47      return 0;
48  }
```

Hacemos la prueba con un arreglo de 1000 elementos, tanto en el stack como en el heap y calculamos sus tiempos de ejecucion y obtenemos los siguiente resultados:
El tiempo en el stack es: 0.370000
El tiempo en el heap es: 0.390000

Nos podemos dar cuenta que la asignacion en el stack(estatica) toma menos tiempo que la asignacion en el heap(dinamica).