Ejercicios de Programming set del Libro Sebesta

Jimmy Banchon - Rene Balda

13 de febrero de 2014

1. Introducción

Los capitulos a realizar son:

1. Ch5: 4 - 7
2. Ch6: 1, 2, 7
3. Ch7: 1 - 6, 9
4. Ch8: 3, 4, 5
5. Ch9: 1, 5

2. Preguntas y Respuestas

2.1. Capítulo 5: Nombres, Enlaces y Alcances.

2.1.1. Pregunta 4:

```
def subprograma1(param1):
        a=param1*2
        def subprograma2():
                b=a*3
                def subprograma3():
                        c=b*5
```

En "Python.[el] subprograma presentado arriba corre sin problema.

2.1.2. Pregunta 5:

Para la prueba en C, se genera un error.

```
void main(void)
{
        x=21;
```

```
        printf("%d",x);
        int x;
        x=42;
        printf("%d",x);
}
```

Errores:

```
1.-error C2065  x  identificador no declarado
c:\users\jimmy\documents\visual studio \2011\projects\test\test.c

2.-error C2065  x  identificador no declarado
c:\users\jimmy\documents\visual studio\2011\projects\test\test.c
```

Para la prueba en C++, se dio este error al compilar.

```
void main(void)
{

        cout << "Inserta 21:";
        cin >> x;
        int x;
        x=42;
        cout << "X vale:" << x;
        cin.get();cin.get();

}
```

Errores:

```
1.-error C2065: 'x' : identificador no declarado
c:\users\jimmy\documents\visual studio \2011\projects\test\test.cpp
```

Para la prueba en java, se dio el siguiente error.

```
public class Test {

    static void main(String[] args) {
        x=21;
        int x;
        x=42;
       System.out.println(x);
    }
}
```

Errores:

```
Exception in thread "main" java.lang.RuntimeException:
Uncompilable source code - cannot find symbol
  symbol:   variable x
location: class test.NewMain
at test.NewMain.main(NewMain.java:17)
Java Result: 1
```

2.1.3. Pregunta 6:

Para la prueba en C++

```
using namespace std;

int main(){
  for (int i = 0 ; i<5; i++){
    cout << i;
  }
  cout << i;
  return 0;
}
```

Errores:

```
  Error test.cpp: En la función int main():
test.cpp:8:11: error: la búsqueda de nombre de 'i'
cambió por el nuevo alcance ISO de 'for'
[-fpermissive]
```

Para la prueba en Java

```
public class Test{
    public static void main(String args[]){
        for(int i = 0; i< 5 ; i++)
            {
                System.out.println(i);
            }
        System.out.println(i);
    }
}
```

Errores:

```
Error Test.java:9: error: cannot find symbol
System.out.println(i);
symbol:   variable i
 location: class test
```

Para la prueba en C sharp

```
class c5p6
{
  static void Main()
  {
    for(int i = 0 ; i < 5 ; i++)
      {
        Console.WriteLine(i);
      }
    Console.WriteLine(i);
  }

}
```

Errores:

```
Error test.cs(11,23): error CS0103: The name ‘i’ does
not exist in the current context
```

Como podemos ver tanto en lenguaje C++, Java y C sharp no es posible hacer lo que nos pide el problema y nos genera un error, debido a que la variable solo puede ser accedida dentro del bloque for.

2.1.4. Pregunta 7:

```
#include <iostream>

using namespace std;

void first();
void second();
void third();

int main()
{
   for(int i=0;i<100000;i++){
      third();
   }

   return 0;
}

void first(){
    int i[100000000]={};
```

```
}

void second(){
    int *i= new int(100000000);
}

void third(){
    static int i[100000000];
}
```

Cuando ejecutamos la funcion "first"(memoria de stack) esta ocupa gran cantidad de espacio de memoria virtual lo cual nos dejara con menos memoria para las ultimas iteraciones lo que hace mas lenta la ejecucion del programa.

Cuando ejecutamos la funcion "second"(memoria de heap) esta en cada iteracion crea y elimina referencias, el espacio en momoria no referenciado es reutilizado lo que impide que ocupe menos memoria virtual por ende sera mas rapido.

Finalmente ejecutamos la funcion "third"(memoria estatica) el espacio en memoria al ejecutar esta funcion no se modifica lo que hace que el programa se ejecute mucho mas rapido.

2.2. Capítulo 6: Tipos de Dato.

2.2.1. Pregunta 1:

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

        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();
        }
```

Resultados: El resultado de sum1 es: 46, y el de sum2 es 48. Esto es debido a que cada expresion se evalua de izquierda aderecha, es decir en sum1 primero se realiza la division de i/2 y despues se cambia el valor de dicha variable, en cambio en sum 2 primero se camba el valor de j, asi que teniendo la misma expresion obtenemos resultados diferentes.

2.2.2. Pregunta 2:

Codigo en C++

```
\#include <iostream>

using namespace std;
int  fun(int *val);
int main()
{
    int x=10,y=10;
    int sum1=x/2 + fun(&x);
    int sum2=fun(&y)+ y/2;

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

   return 0;
}

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

Resultado: sum1= 46; sum2=48;

Codigo en C#

```
using System.IO;
using System;

class Suma
{
    int val;
    public Suma(int val1){
        val=val1;

    }
    static void Main()
    {
        int x=10;
        int y=10;
        Suma s=new Suma(x);
        Suma s1=new Suma(y);
        int sum1=s.val/2+s.fun(s.val);
```

```
        int sum2=s1.fun(s1.val)+s1.val/2;

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

    }
}
```

Resultado: sum1= 46; sum2=48;

Codigo en Java

```
public class Suma{

        public int val;

        public Suma(val){
                this.val = val;
        }

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

        public static void main(String args[]){
                int x=10;
                int y=10;
                Suma s=new Suma(x);
        Suma s1=new Suma(y);
        Suma s=new Suma(x);
        Suma s1=new Suma(y);
        int sum1=s.val/2+s.fun(s.val);
        int sum2=s1.fun(s1.val)+s1.val/2;
        System.out.print(sum1 + ";"    +sum2);

        }

}
```

Resultado: sum1= 46; sum2=48;

En todos los lenguajes utilizados para ejecutar el mismo codigo no dio los mismos resulatdos, es decir en estos tres lenguajes se evalua las sentencias de izquierda a derecha.

2.2.3. Pregunta 7:

```
public static void main(String[] args) {

        float val1,val2,val3,val4,val5,result1,result2, result3;
        result1=result2=result3=0;

        val1=8;
        val2=4;
        val3=3;
        val4=5;
        val5=0;
        result1=val1/val2*val3+val4;
        result2=val1*val2/val3+val4;
        if(val5!=0)
                result3=val5*val2*(val3+val4);

        System.out.println("El resultado 1 es: "+result1);
        System.out.println("El resultado 2 es: "+result2);
        System.out.println("El resultado 3 es: "+result3);
}
```

Como sabemos en Java la multiplicacion y la division tienen igual precendecia y asociatividad entre ellas, en cambio la suma y la resta tienen menos precedencia que las 2 anterios; por lo que en el bloque ejecutado siempre se obtendra el resultado de las multiplicacio y divisiones, despues de sumara dicho reslutado con el resto de los valores, en el caso de result3, se hace una evalucion de corto circuito, es decir val5 tiene que tomar un valor diferente de 0 para que result3 tenga un valor diferente de este.

Slaida: El resultado 1 es: 11.0 El resultado 2 es: 15.666667 El resultado 3 es: 0.0

2.3. Capítulo 7: Expressions and Assignment Statements

2.3.1. Pregunta 2:

Traducción al lenguaje C++

```
int fun(int *k); //Prototipo

int main() {
    int i = 10, j = 10, sum1, sum2;
```

```
    sum1 = (i / 2) + fun(&d);
    sum2 = fun(&j) + (j / 2);
    printf("sum1 =  %d \n",sum1);
    printf("sum2 =  %d ",sum2);
    return 0;
}

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

Resultados:

```
sum1 =  46
sum2 =  48
```

Dado los resultados obtenidos podemos concluir que la asociatividad es de izquierda a derecha al igual que C.

Traducción al lenguaje Java

```
public class Test {
    public static void main(String[] args) {
        Test t = new Test();
        int i = 10, j = 10, sum1, sum2;
        sum1 = (i/2) + t.fun(i);
        sum2 = t.fun(j) + (j/2);
        System.out.println("Valor 1: "+sum1);
        System.out.println("Valor 2: "+sum2);
    }

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

Resultados:

```
sum1 =  46
sum2 =  46
```

Traducción al lenguaje C sharp

```
class Test
    {
        static void Main(string[] args)
```

```
        {
            Test t = new Program();
            int i = 10,j=10,sum1,sum2;
            sum1 = (i / 2) + t.fun(ref i);
            sum2 = t.fun(ref j) + (j / 2);
            System.Console.WriteLine(" "+sum1);
            System.Console.WriteLine(" "+sum2);
            Console.Read();
        }
        public int fun(ref int k)
        {
            k = 4 + k;
            return 3*(k) - 1;
        }
    }
```

Resultados:

```
sum1 =  46
sum2 =  48
```

Dado los resultados obtenidos podemos concluir que la asociatividad es de izquierda a derecha al igual que C y C++.

2.3.2. Pregunta 4:

Lenguaje Java.

```
public class Test
    {
    final static int num=5;
    static int x=5;

    public static void main(String[] args){

          x = fun()+x;
          System.out.println(x);
    }

    static int fun() {
            x = 17;
            return 3;
    }

}
```

Resultados:

```
x =  20
```

Primeramente la funcion fun retorna 3 y actualiza la variable global x con 17 para luego sumarla con 3 asignandole 20 a la variable x. En java el operador + tiene asociatividad de izquierda a derecha.

2.3.3. Pregunta 5:

Lenguaje C++

```
int fun();

extern int x = 10;
void main(){

        x = fun()+x;
        a = a+fun();
        printf("\%d", x);
        getch();
}

int fun() {
        x = 17;
        return 3;
}
```

Resultados:

```
x =  20
```

Para este lenguaje podemos ver que siempre en la función al llamar fun(), se va a actualizar la variable global a con 17 y luego se le sumara 3 teniendo asi un valor de 20 en dicha variable.

2.3.4. Pregunta 6:

Lenguaje C Sharp

```
class Test
    {
        static int x = 5;
        static void Main(string[] args)
        {
            x = a +fun();
```

```
            Console.WriteLine(x);
            Console.ReadLine();
        }

        static int fun()
        {
            x = 17;
            return 3;
        }
    }
```

Resultados: OUTPUT

```
x = 8
```

Como podemos ver en C Sharp también se cumple la regla de la asociativad.Es decir la asociatividad va de izquierda a derecha.

2.4. Capítulo 8: Expressions and Assignment Statements

2.4.1. Pregunta 4:

```
void funcion()
{
        int i;
        int op = 0;
        for (i = 0; i < 3; i++)
        {
                switch (op+1)
                {
                        case 3:
                                return;
                        case 2:
                                {op=0;
                                return;}
                        case 1:
                                {op += 2;
                                return;}
                        default:
                                {op= 0;
                                return;}
                }
                if (op > 0) return;
                op = 3 - i;
```

```
            }

}
```

La forma de romper un lazo for y salir de un switch sin usar breaks, es usando return sin ningun valor;