Universidad Nacional de Río Cuarto
Facultad de Ciencias Exactas, Físico-Químicas y Naturales
Departamento de Computación
Asignatura: **ANÁLISIS Y DISEÑO DE SISTEMAS (3303)**
Año 2017

PRÁCTICO Nº 3

TEMA: Prueba

Objetivos

Con este práctico se espera que el estudiante pueda:

- Aplicar los criterios de prueba de caja blanca definiendo el grafo de flujo de control y calculado la complejidad ciclomática en aquellos casos necesarios.
- Aplicar los criterios de prueba de caja negra.
- Corregir los errores detectados en el código.
- Utilizar JUnit para automatizar las pruebas.

1. Prueba Estructural – Prueba de Caja Blanca

Aplicar las técnicas de prueba estructural a los siguientes ejercicios, y analizar los resultados: Ej. ¿Cuáles casos se cubrieron en una técnica y cuáles no?

Para cada inciso se deberán definir los casos de prueba de acuerdo a cada criterio de caja blanca aplicado (cobertura de sentencias, cobertura de condición, cobertura de arcos y cobertura de caminos) e intentar detectar errores en cada programa.

a) Algoritmo de Euclides

```
var
  x,y: Integer;
begin
  read(x); read (y);
  while( x <> y )
     if ( x > y )
          then x := x – y;
          else y := y – x;
     endIf
  endWhile
return x;
end.
```

b) Búsqueda de un elemento en un arreglo

```
Function pertenece( desiredElement:Integer, tabla: array of Integer): boolean
Var
 found:boolean;
 counter, numberOfItem:Integer;
begin
 found:=false;
 counter:=1;
 While ((not found) and (counter < numberOfItem))
   if (tabla[counter] = desiredElement)
       found=true;
   endIf
  counter:=counter+1;
 endWhile
 if (found)
  then
      return:=true
  else
      return:=false
fin
```

c) Mesetas

Dado un arreglo de números enteros ordenados, imprimir la suma de cada una de sus mesetas, donde una meseta es una secuencia de números iguales.

```
Procedure suma_mesetas(arreglo: Arreglo of Integer);
var
 suma, valorOld , x:Integer;
begin
 suma:=0;
 valorOld=arreglo[1];
 for x=1 to arreglo.lenght(arreglo)
          If arreglo[x]<>valorOld then
                print(suma);
                suma:=0;
          EndIf
         suma:=suma+arreglo[x];
         valorOld=arreglo[x];
endfor;
```

d) Cálculo numérico

```
x,y,z: Integer;
if (x ≠ 0) then
        y = 5
else
        z = z – x
endif
if  (z > 1) then
        z = z /x
else
        z = 0
endif.
```

e) Búsqueda de subcadenas

Dado el siguiente fragmento de código:

```
función cantidad_de_subcadenas(W:array[0..n-1]ofchar; S:array[0..m-1]of 1. Char):Int
 var i,j, cant:Int;
 begin
        i:=0;
        j:=0;
        while (i<=length(S)-1)do
                while (W[j]=S[i]  and j<=length(W)-1 )do
                        i := i + 1;
                        j := j + 1;
                end;
                if(j>length(W)-1)
                                then
                                        cant:=cant +1;
                                        i := j ;
                                        j := 0;
                                else
                                        i:= i + 1;
                endif
        end;
        cantidad_de_subcadenas:= cant
end;
```

Ej.

S= [C,A,S,A,D,C,A,S]

W= [C,A,S]

La función debería retornar 2.

Para este último caso analizar si los casos obtenidos con el criterio de "cobertura de caminos y cobertura de arcos" son los mismos. ¿Se detectan los mismos errores?

2. Prueba funcional – Prueba de Caja Negra

a) Un programador ha implementado un programa que realiza el siguiente cálculo matemático $\sqrt{(a+b)}/\sqrt{(c-a)}$, donde a, b y c son números enteros. Diseñe los casos de prueba utilizando el criterio de valor límite.

c) División de números enteros, determine el criterio de caja negra más apropiado.

d) Verificar si una fecha es válida, determine el criterio de caja negra más apropiado.

e) El problema del triángulo (dado los lados de un triángulo, decir si el triángulo es equilátero, Isósceles o escaleno) , determine el criterio de caja negra más apropiado.

f) Un subprograma tiene como entrada 3 parámetros enteros x,y,z. Los valores de x e y representan un intervalo [x, y]. El subprograma tiene como misión estudiar la pertenencia del valor z al intervalo. Las salidas posibles del subprograma son:

- extremo: si z coincide con uno de los extremos del intervalo.
- medio: si z es el punto medio del intervalo, pero no está en la situación anterior.
- interior: si z esta en el intervalo, pero no en las situaciones anteriores.
- exterior: si z no está en el intervalo.
- error: si la entrada es errónea.

Determine el criterio de caja negra más apropiado.

3 . Búsqueda en arreglo

a) Dado el siguiente código java. Diseñe y ejecute los casos de prueba utilizando los criterios de caja blanca estudiados. En caso de encontrar fallas, proponga los cambios apropiados para reparar los errores.

```
Class BúsquedaBin {
   public static int buscar( int [] arreglo, int dato) {
            int inicio;
            int fin = arreglo.length-1;
            int pos;
            while (inicio < fin) {
               pos = (inicio+fin) / 2;
               if ( arreglo[pos] == dato )
                 return pos;
               else if ( arreglo[pos] < dato ) {
                                  inicio = pos+1;
               } else {
                           fin = pos-1;
               }
                }
     return -1;
   }
  }
```

4. MayorCantidad

Dado el siguiente código java:

a) Diseñe y ejecute los casos de prueba aplicando el criterio de "Cobertura de Arco" de la técnica de prueba estructural.

b) En caso de encontrar fallas, proponga los cambios apropiados para reparar el o los errores.

```
/*  El método "mayorCantidad" retorna true si el arreglo de enteros pasado como parámetro contiene mayor cantidad de
elementos mayores que elementos menores al valor del dato pasado como parámetro.
Por ejemplo mayorCantidad(3,[6,6,1]) debería retornar true y
            mayorCantidad(30,[6,6,1]) debería retornar false. */
public static boolean mayorCantidad(int dato, int [] arreglo) {
            int mayores = 0;
            int menores = 0;
            int fin = arreglo.length-1;
            int pos = 0;
            while (pos < fin) {
              if ( arreglo[pos] > dato )
                 mayores = mayores +1;
              if ( arreglo[pos] < dato )
                 menores = menores +1;
            pos = pos+1;
            }
```

```
        if (mayores < menores)
            return true;
    else
            return false;
}
```

5. Cálculo matemático

Dada la función cálculoMatemático(a,b,c,d:Integer):float la cual realiza el siguiente cálculo:

$$\frac{\sqrt{a*(b+\sqrt{c-10})}}{d}$$

Diseñe los casos de prueba utilizando el criterio de caja negra : tabla de decisión.

6. Calculadora

Implemente una calculadora numérica en Java con las siguientes operaciones:

- operaciones básicas (suma, resta multiplicación y división),
- factorial de un número,
- promedio de una lista de números,
- raíz y potencia de un número.

a) Diseñe las pruebas para cada una de las operaciones anteriores utilizando el criterio de prueba de caja negra valor límite.

b) Implemente las pruebas anteriores utilizando JUnit y determine si las pruebas han sido exitosas.

7. Clasificador de triángulos

a) Implemente en java un programa que determine a que clasificación pertenece un un triángulo según sus ángulos (triángulo acutángulo, rectángulo, obtusángulo).

b) Utilice tablas de decisión y clases de equivalencia para diseñar los casos de prueba del programa anterior.

c) Implemente los casos de prueba utilizando JUnit y determine si la prueba a sido exitosa.