Práctica Nro. 8
Excepciones

Objetivo: Conocer e interpretar los distintos modelos de excepciones que implementan los lenguajes de programación.

Ejercicio 1: ¿Explique claramente a qué se denomina excepción?
Ejercicio 2: ¿Qué debería proveer un lenguaje para el manejo de las excepciones? ¿Todos los lenguajes lo proveen?
Ejercicio 3: ¿Qué ocurre cuando un lenguaje no provee manejo de excepciones? ¿Se podría simular? Explique cómo lo haría
Ejercicio 4: Cuando se termina de manejar la excepción, la acción que se toma luego es importante. Indique

01. ¿Qué modelos diferentes existen en este aspecto?
02. Dé ejemplos de lenguajes que utilizan cada uno de los modelos presentados anteriormente. Por cada uno responda respecto de la forma en que trabaja las excepciones.
    a. ¿Cómo se define?
    b. ¿Cómo se lanza?
    c. ¿Cómo se maneja?
    d. ¿Cuál es su criterio de continuación?
03. ¿Cuál de esos modelos es más inseguro y por qué?

Ejercicio 5: La propagación de los errores, cuando no se encuentra ningún manejador asociado, no se implementa igual en todos los lenguajes. Realice la comparación entre CLU, PL/1, ADA, respecto a este tema. Defina la forma en que se implementa en un lenguaje conocido por Ud.
Ejercicio 6: Sea el siguiente programa escrito en Pascal

```
...
Procedure Manejador;
  Begin ...   end;
Procedure P(X:Proc);
  begin
    ....
    if Error then X;
    ....
  end;
Procedure A;
  begin
          ....
          P(Manejador);
          ….
  end;
....
```

¿Qué modelo de manejo de excepciones está simulando? ¿Qué necesitaría el programa para que encuadre con los lenguajes que no utilizan este modelo? Justifique la respuesta.

Ejercicio 7: Sea el siguiente programa escrito en Pascal:

```
Program Principal;
var x:int;  b1,b2:boolean;
 Procedure P (b1:boolean);
      var x:int;
      Procedure Manejador1
                        begin
                 x:=x + 1;
                end;
        begin
        x:=1;
        if b1=true then Manejador1;
        x:=x+4;
        end;
        Procedure Manejador2;
          begin
                x:=x * 100;
        end;
```

```
Begin
        x:=4;
        b:=true;
        b1:=false;
        if b1=false then Manejador2;
        P(b);
        write (x);
End.
```

a) Implemente este ejercicio en PL/1 utilizando manejo de excepciones
b) ¿Podría implementarlo en ADA utilizando manejo de excepciones? En caso afirmativo, realícelo.

Ejercicio 8: Sean los siguientes, procedimientos de un programa escrito en CLU:

```
Proc_1=proc(m:int)returns(int)
signals(tipo1)
.....
  if m=0 then signal tipo1
            Except
          when tipo1: write("Se produjo un
error de tipo1 en Proc_1"); signal tipo2;
          others: write("Se produjo otro tipo
de error en Proc_1")
                  end
    m:=m * 10
end Proc_1
Proc_2=proc()returns(int)
z=0;
While z>=0
  if z=0 then Proc_1(z)
            Except
          when tipo2: write("Se produjo un
error de tipo2 en Proc_2")
          end
end while; Except
          when tipo1: write("Se produjo un
error de tipo2 en Proc_2")
          end
.....
end Proc_2
Proc_ppal=proc()returns(int)
.....
  Proc_2
          Except
          when tipo1: write("Se produjo un
error de tipo1 en Proc_ppal")
          end
.....
end Proc_ppal
```

a) Analizar el ejemplo y decir qué manejadores ejecuta y en qué valores quedan las variables. JUSTIFIQUE LA RESPUESTA.

b) Podría simular un efecto parecido en ADA? En caso de poder, explique cómo.

Ejercicio 9: Indique diferencias y similitudes entre Phyton y Java con respecto al manejo de excepciones.
Ejercicio 10: Qué modelo de excepciones implementa Ruby?. Qué instrucciones específicas provee el lenguaje para manejo de excepciones y cómo se comportan cada una de ellas?
Ejercicio 11: Indique el mecanismo de excepciones de javascript.
Ejercicio 12: Sea el siguiente programa escrito en ADA:

```
Procedure Principal;
        x:exception;
        y:integer;      b:boolean;
Procedure Prueba1 (out m:integer);
        begin
        m:=20;
        if (b=true) then raise X;
        m:=m + 2;
        end;
        Procedure Prueba2;
        a:int:=0;   b:=4/a;
        begin
                y:=y+8;
        exception
        when constraint-error => y:=y+10;
end;
```

```
begin
                Read(b);
                 y:=1;
                 Prueba1(y);
                 Prueba2;
                 write(y);
                exception
                 when constraint-error => begin
                                        y:=y+4;
                                         write(Y);
                                         end;
                 when X => begin  y:= y*30;
                                        write(Y);
                                         end;
end;
```

a) Indique el camino de ejecución.
b) Agregar el uso de una excepción anónima

Ejercicio 13: Sea el siguiente código escrito en CLU

```
Procedure Main
    Error1 : exception;
    x, y: integer;
    Procedure UNO () signals error1;
            x:integer
            Begin
            x:=2; .....
              While y < x Do
                    If y=0 Then
                            signal error1;
                  end if; exception
                            when error1 ->
                   y:=y+7;
                   x:=x+2; resignal;
                   end;
                    Dos();
                   y:=y+1;
            Wend; exception
                    when error1 -> y:=x+3;
                    x:=x+3; Resignal; .End;
    End; //UNO
```

```
Procedure Dos() signals error1;
            m:integer;
            Begin
              ...
              if m=0 then  signal error1;
            End;

Begin //MAIN
    x:=1; y:=0;
    Uno(); exception when error1 -> x:=x+1;
                                    y:=y+1;
end;
    ...
    Dos(): exception when error1 -> resignal;
end;
    ...
End; //MAIN
```

a) Indique cómo se ejecuta el código. Debe quedar en claro los caminos posibles de ejecución, cuales son los manejadores que se ejecutan y cómo se buscan los mismos y si en algún caso se produce algún error.
b) La ejecución del manejador para error1 modifica siempre la variable x de UNO? En caso negativo indique cómo haría para lograrlo. Justifique la respuesta.

Ejercicio 14. Dado el siguiente código en Ada. Marque con una cruz sólo los caminos de ejecución correctos en los casos en que se produzca o levante una excepción.

```
Program Main
var x,y,i:integer;
e,e2:exception
Procedure A
   var y,b:integer;
   e:exception;
Begin
  x=x=+1;
  y=0;
  b=x+i+3;
  B();
 (6)Exception when e Begin
       x:=x + 5; raise;
      End;
End;
```

```
Procedure B
 var z:integer;
Begin
   z=3;
   raise e;(7)
   (4)Exception when e Begin
         x:=x + 1; raise;
       End;
  (9) when e2 Begin
        x:=x + 6;
      End;
    (5)when others Begin
         x:=x + 1;
        End;
End;
```

```
Procedure C
Begin
If (x=1) then begin
   raise e2;(8)
 end;
else A();
End;
```

```
BEGIN //MAIN
 x=1; y=1; i=2;
 C();
 write(x);
 (1)Exception when e Begin
         x:=x + 5;
       End;
(2) when e2 Begin
      x:=x + 6;
End;
(3) when others Begin
      x:=x + 7;
     End;
END.
```

En **(7)** se levanta e y se maneja en 4, luego se relanza la excepción que es	En **(7)** se levanta e y se maneja en 4, luego se relanza la excepción que	En **(8)** se levanta e2 y se maneja en 4, luego se relanza la excepción que es

manejada en **(6)** y termina manejándose nuevamente en **(1)**.	es manejada en **(1)** porque e pertenece a main.	manejada en **(1)** porque e pertenece a main.
En **(7)** se levanta e y se maneja en 4, luego se vuelve a levantar la excepción de forma anónima por lo que B termina y se busca el manejador en A para menejarse finalmente en (1) de Main porque C y A no tiene manejadores definidos para esa variable.	En **(7)** se levanta e y se maneja en 4, luego se relanza la excepción que es manejada en **(3)** porque e pertenece a main pero pierde el alcance en A.	En **(8)** se levanta e2 y se maneja en 2, y el programa termina.