UNIVERSIDAD NACIONAL DE RÍO CUARTO
FACULTAD DE CS. EXACTAS, FCO-QCAS Y NATURALES - DEPARTAMENTO DE COMPUTACIÓN
ASIGNATURA: **INTRODUCCIÓN A LA ALGORÍTMICA Y PROGRAMACIÓN**
Año: 2020

Práctica Nº 5

Tema: Acciones **Parte**: 1 de 3

Esta práctica tiene como objetivos

- Abstraer la solución de un problema mediante acciones que expresen a través de sus nombres las partes del mismo. A continuación resolver estas acciones.
- Introducir el uso de acciones con parámetros y razonar sobre sus ventajas.
- Emplear los distintos tipos de pasaje de parámetros analizando ventajas y desventajas.

Ejercicios propuestos

I) PREGUNTA: ¿Por qué las funciones deben tener parámetros de tipo dato, es decir, solo parámetros de entrada?

II) PREGUNTA: Dentro del cuerpo de un algoritmo, ¿Dónde se definen las funciones?, ¿Cómo se hace para que una función devuelva un valor?

1) Dada la siguiente acción:

```
Acción SumRes(dato a,b ∈ Z, dato-resultado c ∈ Z, resultado multi
∈ R)
Inicio
 a ← a+1
 b ← b-5
 multi ← a*b
 si multi>=0  entonces
   c ← c + multi
 sino
   c ← c - multi
 fsi
Faccion
```

¿Cuál es la salida de los parámetros actuales después de las siguientes invocaciones a la acción SumRes, en los algoritmos que se muestran abajo?

```
Algoritmo Ejemplo1   //resuelto para que sirva de modelo
Léxico
  x, y, z ∈ Z
  p ∈ R
Acción SumRes(dato a,b ∈ Z, dato-resultado c ∈ Z, resultado multi
∈ R)
Inicio
  a ← a+1
  b ← b-5
  multi ← a*b
  si multi>=0  entonces
    c ← c + multi
  sino
    c ← c - multi
  fsi
Faccion
Inicio
```

```
  x ← 16
  y ← 5
  z ← 20
  p ← 42.0
  {e₀: x=16, y=5, z=20, p= 42 }
  SumRes(x,y,z,p)
  Salida:x y z p
  {eᶠ:x=16, y=5, z= 20, p= 0 }
Fin


Algoritmo Ejemplo2
Léxico
  x, y, z ∈ Z
  p ∈ R
Acción SumRes(dato a,b ∈ Z, dato-resultado c ∈ Z, resultado multi
∈ R)
Inicio
  a ← a+1
  b ← b-5
  multi ← a*b
  si multi>=0  entonces
    c ← c + multi
  sino
    c ← c - multi
  fsi
Faccion
Inicio
  x ← 5
  y ← 4
  z ← 0
  p ← 0.0
  {e₀:                                                  }
  SumRes(x,y,z,p)
  Salida:x y z p
  {eᶠ:                                                   }
Fin


Algoritmo Ejemplo3
Léxico
  a, b, c ∈ Z
  pr ∈ R
Acción SumRes(dato a,b ∈ Z, dato-resultado c ∈ Z, resultado multi
∈ R)
Inicio
  a ← a+1
  b ← b-5
  multi ← a*b
  si multi>=0  entonces
    c ← c + multi
```

```
  sino
    c ← c - multi
  fsi
Faccion
Inicio
  a ← 8
  b ← -6
  c ← 10
  {e₀:                                                    }
  SumRes(b,a,c,pr)
  Salida:b a c pr
  {eᶠ:                                                    }
Fin
```

```
Algoritmo Ejemplo4
Léxico
  x, z ∈ Z
  b ∈ R
Acción SumRes(dato a,b ∈ Z, dato-resultado c ∈ Z, resultado multi
∈ R)
Inicio
  a ← a+1
  b ← b-5
  multi ← a*b
  si multi>=0  entonces
    c ← c + multi
  sino
    c ← c - multi
  fsi
Faccion
Inicio
  x ← 0
  z ←10
  b ← 100.0
  {e₀:                                                    }
  SumRes(1,z,x,b)
  Salida:z x b
  {eᶠ:                                                    }
Fin
```

2) Dado el siguiente algoritmo:

Algoritmo AreaFiguras
Lexico
x, y, z, sup ∈ **R**

Acción Cargar(:a, b ∈ R; :f ∈ Caracter)
Inicio
//ingrese una **t** si es un triángulo y una **r** si es un rectángulo
Entrada:f

```
   si f= 'r'      entonces
      //ingrese el 1er   y 2do lado
      Entrada:a   b
    sino
      // ingrese la base del triángulo y la altura del triángulo
      Entrada:a   b
   fsi
Faccion

Acción Calcular(              a, b∈ R;              :f  ∈ Caracter;              :area ∈ R)
Lexico Local
 s  ∈ R
Inicio
  si f= 'r'    entonces
     area ← a * b
  sino
     area ← a * b /2
  fsi
Faccion

Accion Mostrar (              : a, b ∈ R;              :f  ∈ Caracter; area ∈ R)
Lexico local
  msge ∈ Cadena
Inicio
  Si f= 'r'   Entonces
     msge←"El área del rectángulo dado por los lados"
  Sino
    msge←"El área del triángulo dado por la altura y la base"
  fsi
  Salida:msge a b area
Faccion

Inicio //programa principal
   Cargar (x,y,z)
   Calcular(x,y,z,sup)
   Mostrar (x,y,z,sup)
Fin
```

Determine el tipo de pasaje de parámetros (dato, dato-resultado, resultado) que corresponde dar a las variables declaradas como parámetros formales en las acciones Cargar, Calcular y Mostrar.

3) a) Analiza y describa lo que hace el siguiente algoritmo, y en base a ello, determine el tipo de pasaje de parámetros de cada una de las acciones que componen al mismo.
b) Desarrolla la acción MostrarMonto que corresponde al algoritmo.

```
Algoritmo CalcularPagoFormadePago
Léxico
apellidoNombres ∈ Cadena       // dato para almacenar el nombre del cliente
montoCompra ∈ R        // dato para almacenar la compra $ del cliente
esContado ∈ Lógico      // dato que permite identificar la forma de pago
interesTarjeta ∈ 3..15  // dato para almacenar el interés de la tarjeta
Acción ObtenerDatos (                  apeNombres ∈ Cadena, monto ∈ R, contado ∈ Lógico)
Acción PagoContado (                        monto ∈ R)
Acción IdentificaTarj(                      interTarj ∈ 3..15)
Acción PagoTarjeta (                        monto ∈ R,          interTarj ∈ 3..15)
Acción MostrarMonto (                  apeNombres ∈ Cadena, monto ∈ R)
Inicio  //del algoritmo
```

```
   ObtenerDatos(apellidoNombres, montoCompra, esContado)
   si esContado  entonces //esContado=Verdadero
     PagoContado(montoCompra)
   sino //esContado=Falso
     IdentificaTarj(interesTarjeta)
     PagoTarjeta(montoCompra, interesTarjeta)
   fsi
   MostrarMonto(apellidoNombres, montoCompra)
Fin

Acción IdentificaTarj(          :interTarj ∈ 3..15)
Léxico local
cod ∈ 1..4 //variable para determinar la tarjeta utilizada
Inicio
  //Ingrese el número de la tarjeta utilizada -1,2,3 o 4-
  //1.Master – 2.Visa – 3.Cabal – 4.Cordobesa
  Entrada:cod
  segun
    (cod=1):interTarj←10
    (cod=2):interTarj←5
    (cod=3):interTarj←15
    (cod=4):interTarj←3
  fsegun
Faccion

Acción PagoContado (                monto ∈R)
Inicio
  si monto>=1000   entonces
    monto←monto-(monto*15/100)
  sino
    monto←monto-(monto*10/100)
  fsi
Faccion


Acción PagoTarjeta (                monto ∈R,          interTarj ∈3..15)
Inicio
  monto←monto+monto*interTarj/100
Faccion
```

4) Desarrolla una acción que simule una calculadora. Debe recibir dos números que serán los operandos y un carácter que será el operador. En una variable resultado se almacenará el resultado de aplicar el operador a los operandos. Las operaciones que debe soportar son: '+', '-', '/', '*'. En el caso que se intente la división por cero, la acción emitirá un mensaje 'ERROR' y en la variable resultado almacenará un 999999999.