

Introducción a la Algorítmica y Programación (3300)

Prof. Ariel Ferreira Szpiniak - aferreira@exa.unrc.edu.ar
Departamento de Computación
Facultad de Cs. Exactas, Fco-Qcas y Naturales
Universidad Nacional de Río Cuarto

Teoría 5

Modularización, Abstracción Acciones



2017 Lic. Ariel Ferreira Szpiniak

Noticias

2 de abril: Día del Veterano y de los Caídos en la Guerra de Malvinas – 35 años (1982-2017)

Con orgullo y sin olvidar a los gauchos, con Rivero a la cabeza, y los pibes del '82 que defendieron las islas de los imperialistas. Sin olvidar a los genocidas y torturadores del 76 al 83 que, acostumbrados a matar gente indefensa, hablaban con valentía frente a la cámaras de televisión, pero no fueron al frente de batalla sino que mandaron a nuestros pibes.

Dice Eduardo Galeano: “*la Guerra de las Malvinas, guerra patria que por un rato unió a los argentinos pisadores y a los argentinos pisados, culmina con la victoria del ejército colonialista de Gran Bretaña. No se han hecho ni un tajito los generales y coroneles argentinos que habían prometido derramar hasta la última gota de sangre. Quienes declararon la guerra no estuvieron en ella ni de visita. Para que la bandera argentina flameara en estos hielos, causa justa en manos injustas, los altos mandos enviaron al matadero a los muchachitos enganchados por el servicio militar obligatorio, que más murieron de frío que de bala. No les tiembla el pulso: con mano segura firman la rendición los violadores de mujeres atadas, los verdugos de obreros desarmados.*”



2017 Lic. Ariel Ferreira Szpiniak

2

Modularización

- Las acciones son otra forma de ayudar a resolver problemas complejos del mundo real.
- Las acciones posibilitan construir módulos, para obtener soluciones independientes, y así aprovechar todas las ventajas de la modularización: productividad, reusabilidad, mantenimiento correctivo, facilidad de crecimiento, y mejor legibilidad.

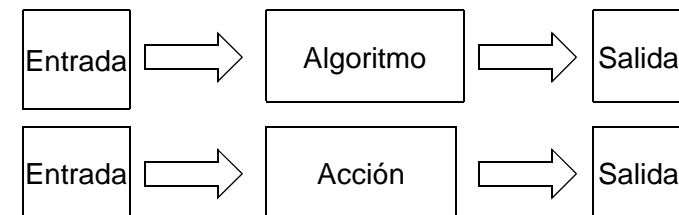


2017 Lic. Ariel Ferreira Szpiniak

3

Acciones como Algoritmos

- Las acciones son mini-algoritmos.
- Un algoritmo puede ser entendido como un procesador de información, con datos de entrada y de salida.
- Las acciones son similares a los algoritmos.
- Una acción puede transformar un estado en otro distinto.



2017 Lic. Ariel Ferreira Szpiniak

4

Acciones con parámetros

- Las acciones son más efectivas cuando son módulos **autocontenidos**.
- Cuando un problema es muy complicado los programas escritos para resolverlo serán a su vez también complejos.
- Entonces, para poder encontrar una buena solución al problema recurrimos al diseño descendente y dividimos el problema en subproblemas.



Acciones con parámetros

- Si las acciones que dan solución a los subproblemas son módulos **autocontenidos**, uno puede resolver y testear cada acción **independientemente** de los otras acciones.
- Para que las acciones sean **autocontenidas** no **deben** hacer **referencia** **declaraciones** (variables, constantes, tipos) que estén **fuera de dicha acción**, como por ejemplo el léxico del algoritmo principal.



Acciones con parámetros


- Para lograr que una acción pueda ser considerada un **algoritmo autocontenido**, la información debe poder ser transferida entre las acciones y el resto del algoritmo principal a través de lo que llamaremos **parámetros**.
- Los parámetros permiten que una acción pueda manipular diferentes valores y por lo tanto la misma acción puede ser usada tantas veces como sea necesario en un mismo algoritmo.



Acciones Estructura

- Se componen de:
 1. Un encabezamiento: el cual tiene la palabra reservada **Acción** seguida de un identificador.

Este identificador puede o no ser seguido de parámetros.


 2. Declaraciones locales: esto es un léxico local donde se declaran las variables locales a la acción.
 3. Bloque o Cuerpo de acciones ejecutables: encerradas entre **Inicio** y **Fin** se desarrollan el conjunto de acciones o composiciones (secuenciales, condicionales, etc.) que resuelven la especificación de la acción.



Acciones

Estructura

1. *Cabecera* **Acción** <identificador> (<lista de parámetros>)
2. *Declaraciones* **Lexico local** (si es necesario)
variables, constantes, acciones y funciones propias de la acción
3. *Sentencias ejecutables* **Inicio**
<acción más simple>
...
<acción más simple>
Fin
4. *Ubicación*
Las declaraciones de acciones se hacen en el léxico del algoritmo principal, después de las declaraciones de los identificadores del mismo (variables, etc.).



Acciones

Invocación y ejecución

- Una acción se ejecuta indicando su nombre y los parámetros. Esto se conoce como “llamado” o “invocación” de la acción. El resultado es la ejecución de las acciones contenidas en esa acción.
- Luego de ejecutar una acción el algoritmo continúa ejecutando las acciones siguientes a la llamada o invocación.



Acciones con parámetros

Motivación

Ejemplo: Intercambio de variables

Supongamos que deseamos resolver el siguiente problema:

Dadas 3 variables **a**, **b** y **c** con algún valor, todos distintos entre sí, colocar en las variables **p**, **s** y **t** los mismos valores contenidos en **a**, **b** y **c** pero en **orden creciente**.

Pensemos una solución.....

Un algoritmo que solucione el problema planteado puede ser:



Intercambio de variables

Acción clasificarTresValores

Lexico local ←

aux, p, s, t ∈ Z

Inicio

```
p ← a
s ← b
t ← c
según
  p < s y p < t: nada
  s < p y s < t: aux ← s
                 s ← p
                 p ← aux
  t < p y t < s: aux ← t
                 t ← p
                 p ← aux
```

fsegun

según

```
s < t: nada
t < s: aux ← s
       s ← t
       t ← aux
```

fsegun

Para cuando necesitamos usar variables que solo participan dentro de ésta acción



Intercambio de variables

Reescribimos el algoritmo anterior de la siguiente forma:

Acción clasificarTresValores

Lexico local

aux, p, s, t ∈ Z

Inicio

p ← a

s ← b

t ← c

según

p < s y p < t: nada

s < p y s < t: intercambia-ps

t < p y t < s: intercambia-pt

fsegún

según

s < t: nada

t < s: intercambia-st

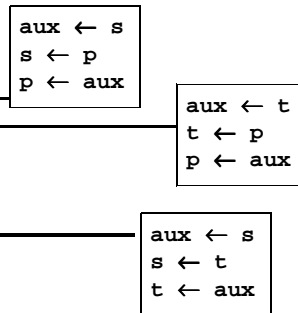
fsegún

Fin



2017 Lic. Ariel Ferreira Szpiniak

13



Intercambio de variables

Acción intercambia-ps

Lexico local

aux ∈ Z

Inicio

aux ← p

p ← s

s ← aux

Fin

Acción intercambia-pt

Lexico local

aux ∈ Z

Inicio

aux ← p

p ← t

t ← aux

Fin

Acción intercambia-st

Lexico local

aux ∈ Z

Inicio

aux ← s

s ← t

t ← aux

Fin



2017 Lic. Ariel Ferreira Szpiniak

14

*¿Qué cosas
tienen en
común éstas 3
acciones?*

Intercambio de variables

Todas tienen el mismo formato:

Acción Intercambiar-♣♦

Lexico local

aux ∈ Z

Inicio

aux ← ♣

♣ ← ♦

♦ ← aux

Fin



2017 Lic. Ariel Ferreira Szpiniak

15

Intercambio de variables

Podemos definir una única acción utilizando *parámetros*:

Acción Intercambiar(x, y ∈ Z)

Lexico local

aux ∈ Z

Inicio

aux ← x

x ← y

y ← aux

Fin

El algoritmo quedaría: (ver siguiente diapositiva)



2017 Lic. Ariel Ferreira Szpiniak

16

Intercambio de variables

Entonces clasificarTresValores quedaría:

Acción clasificarTresValores

Lexico local

Acción Intercambiar($x, y \in \mathbb{Z}$)

Inicio

```
p ← a
s ← b
t ← c
según
  p < s ∧ p < t: nada
  s < p ∧ s < t: Intercambiar(p, s)
  t < p ∧ t < s: Intercambiar(p, t)
fsegún
según
  s < t: nada
  t < s: Intercambiar(s, t)
fsegún
```

Fin



2017 Lic. Ariel Ferreira Szpiniak

17

Acciones con parámetros

Ejemplo

- Veamos el problema del área del triángulo.
- Supongamos que queremos escribir una **acción autocontenida** para hallar el área de un triángulo.
- Ya sabemos que necesitamos conocer la altura y la base como datos de entrada. El resultado o salida es el mostrar el área del triángulo por pantalla.
- Esta información puede ser incluida en la cabecera o encabezamiento de la acción, en forma de parámetros. De esta manera lograremos obtener una **acción autocontenida**.



2017 Lic. Ariel Ferreira Szpiniak

18

Acciones con parámetros

Ejemplo

Desarrollar una **acción** que dada la base y la altura de un triángulo, calcule y muestre el área del mismo.

• Análisis

Datos: base y altura son números reales positivos dados (no hay que ingresarlos).

Resultado: área es un número real positivo.

Relación entre e/s: $\text{área} = (\text{base} * \text{altura}) / 2$



2017 Lic. Ariel Ferreira Szpiniak

19

Acciones con parámetros

Ejemplo

• Diseño 1

Los datos de entrada pueden ser parámetros de entrada de la acción.

Parámetros de entrada: altura y base.

Parámetro de salida: no tiene.

Acción CalcularMostrarAreaTriángulo($\text{baseTri} \in \mathbb{R}, \text{alturaTri} \in \mathbb{R}$)

Lexico local

areaTri $\in \mathbb{R}$

Inicio

```
areaTri ← (baseTri*alturaTri)/2
Escribir(areaTri)
```

Fin



2017 Lic. Ariel Ferreira Szpiniak

20

Aún está
incompleto

Acciones con parámetros

Invocación o llamado

Ejemplo

Supongamos ahora que deseamos solucionar el problema: Desarrollar un **algoritmo** que solicite al usuario la base y la altura de un triángulo y calcule y muestre el área del mismo. Como vemos, este problema es muy similar al anterior solo que debemos ingresar la base y la altura. Entonces podemos **reutilizar** la acción **CalcyMostrarAreaTriángulo** definida anteriormente. Luego del análisis y el diseño correspondiente obtendríamos el siguiente algoritmo.



Acciones con parámetros

Invocación o llamado - Ejemplo

Algoritmo CalcularAreaTriángulo

Lexico

base, altura $\in \mathbb{R}$ {base y altura del triángulo}

Acción LeerDatos($x \in \mathbb{R}, y \in \mathbb{R}$)

Acción CalcyMostrarAreaTriángulo($baseTri \in \mathbb{R}, alturaTri \in \mathbb{R}$)

Inicio

LeerDatos(base, altura)

CalcyMostrarAreaTriángulo(base, altura)

Fin

Acción CalcAreaTriángulo($baseTri \in \mathbb{R}, alturaTri \in \mathbb{R}$)

Inicio

Escribir($baseTri * alturaTri / 2$)

Fin

Acción LeerDatos($x \in \mathbb{R}, y \in \mathbb{R}$)

Inicio

Leer(x, y)

Fin



Acciones con parámetros

Invocación o llamado

- Observar que el tipo de las variables pasadas en la invocación son correspondientes con los tipos de las variables que se ponen en la definición de la acción.

...
base, altura $\in \mathbb{R}$

Acción LeerDatos($x \in \mathbb{R}, y \in \mathbb{R}$)

Acción CalcyMostrarAreaTriángulo(baseTri $\in \mathbb{R},$ alturaTri $\in \mathbb{R}$)

...

CalcyMostrarAreaTriángulo(base, altura)

...



Acciones con parámetros

Ejemplo

¿Y si en lugar de solo mostrar el área por pantalla deseamos dejar almacenado dicho valor en una variable?

Entonces sería más claro utilizar la partición original del problemas en 3 subproblemas (mostrar el resultado aparte) y agregar a la acción **CalcAreaTriángulo** una parámetro más en la lista de parámetros para almacenar el área.

Parámetros de entrada: altura y base.

Parámetro de salida: área

Acción CalcAreaTriángulo2(baseTri $\in \mathbb{R},$ alturaTri $\in \mathbb{R},$ areaTri $\in \mathbb{R}$)

Inicio

areaTri $\leftarrow (baseTri * alturaTri) / 2$

Fin



Acciones con parámetros

Invocación o llamado - Ejemplo

También podríamos parametrizar la acción

MostrarResultados

Algoritmo CalcularAreaTriángulo

Lexico

base, altura $\in \mathbb{R}$ {base y altura del triángulo}

area $\in \mathbb{R}$ {área del triángulo}

Acción LeerDatos($x \in \mathbb{R}, y \in \mathbb{R}$)

Acción CalcAreaTriángulo2($baseTri \in \mathbb{R}, alturaTri \in \mathbb{R}, areaTri \in \mathbb{R}$)

Acción MostrarResultados($a \in \mathbb{R}$)

Inicio

LeerDatos(base, altura)

CalcAreaTriángulo2(base, altura, area)

MostrarResultados(area)

Fin

Acción MostrarResultados($a \in \mathbb{R}$)
Inicio
Escribir(a)
Fin

Aún está incompleto



2017 Lic. Ariel Ferreira Szpiniak

25

Acciones con parámetros

Invocación o llamado

La invocación a una acción con parámetros se realiza de la misma manera que hemos visto que se invoca a una acción sin parámetros, agregando, a continuación del nombre de la acción, la lista de valores o variables a utilizar (parámetros actuales).

• Invocación con valores:

CalcAreaTriángulo(2,9, 6,4)

CalcAreaTriángulo2(2,9, 6,4, area)

• Invocación con variables:

CalcAreaTriángulo(base, altura)

CalcAreaTriángulo2(base, altura, area)

• Invocación mixta (con valores y variables):

CalcAreaTriángulo(4,5, altura)

CalcAreaTriángulo2(4,5, altura, area)

No puede ir un valor aquí.



2017 Lic. Ariel Ferreira Szpiniak

26

Tipos de parámetros

Parámetros Formales

Nombre asignado en la cabecera de la acción a los objetos que serán manipulados en la acción.

Parámetros Actuales

Objetos que son pasados como datos en la invocación (o llamada) de una acción. También llamados efectivos o reales.

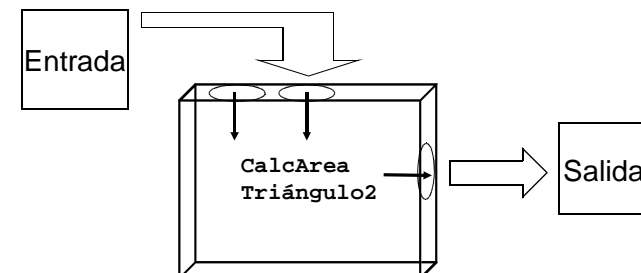


2017 Lic. Ariel Ferreira Szpiniak

27

Tipos de parámetros

- En la acción del triángulo, por ejemplo, los parámetros **x** e **y** no tienen valor asignado hasta que la acción **LeerDatos** es invocada. Recién en ese momento se le pasará el valor real de la **base** al parámetro **x**, y **altura** al parámetros **y**.



2017 Lic. Ariel Ferreira Szpiniak

28

Acciones con parámetros

Invocación o llamado - Ejemplo

Algoritmo CalcularAreaTriángulo

Lexico

base, altura $\in \mathbb{R}$ {base y altura del triángulo}
 area $\in \mathbb{R}$ {área del triángulo}

Acción LeerDatos(x, y $\in \mathbb{R}$)

Acción CalcAreaTriángulo2(baseTri, alturaTri, areaTri $\in \mathbb{R}$)

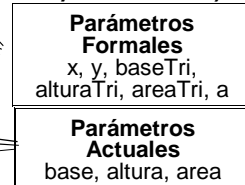
Acción MostrarResultados(a $\in \mathbb{R}$)

Inicio

LeerDatos(base, altura)

CalcAreaTriángulo2(base, altura, area)

MostrarResultados(area)



Fin

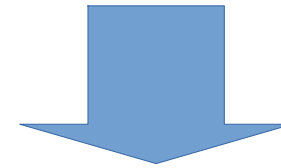
¿Cómo hacemos para distinguir cuáles son los parámetros de entrada y cuáles los de salida?



2017 Lic. Ariel Ferreira Szpiniak

29

Tipos de pasaje de parámetros por copia



- Por Valor
- Por Resultado
- Por Valor/Resultado



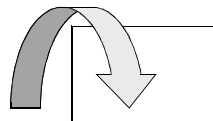
2017 Lic. Ariel Ferreira Szpiniak

30

Tipos de pasaje de parámetros

Pasaje por Valor

Cláusula: dato



Los parámetros que poseen este tipo de pasaje de parámetro se lo conoce como **parámetros de entrada**.

Cuando un parámetro es pasado por valor, el valor del parámetro actual es utilizado para inicializar el valor del parámetro formal.

Al asociarse el parámetro formal (puede ser una variable o una constante) sólo al valor inicial del parámetro actual, las modificaciones en el parámetro formal no afectan al parámetro actual.



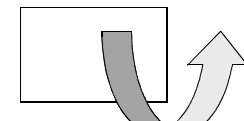
2017 Lic. Ariel Ferreira Szpiniak

31

Tipos de pasaje de parámetros

Pasaje por Resultado

Cláusula: resultado



Los parámetros que poseen este tipo de pasaje de parámetro se lo conoce como **parámetros de salida**.

Cuando un parámetro es pasado por resultado, no se transmite ningún valor durante la invocación. El valor inicial del parámetro formal es *indeterminado*.

Cuando el módulo finaliza su ejecución, el valor final del parámetro formal se asocia al parámetro actual, es decir se le asigna un resultado a la variable utilizada durante la invocación.



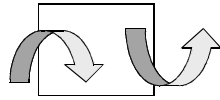
2017 Lic. Ariel Ferreira Szpiniak

32

Tipos de pasaje de parámetros

Pasaje por Valor / Resultado

Cláusula: dato-resultado



Los parámetros que poseen este tipo de pasaje de parámetro se lo conoce como **parámetros de entrada/salida**.

Es una combinación del pasaje por valor y por resultado.

El valor del parámetro actual es utilizado para inicializar el parámetro formal.

Cuando el módulo finaliza su ejecución, el valor final del parámetro formal se asocia al parámetro actual, es decir se actualiza el valor del parámetro actual.



2017 Lic. Ariel Ferreira Szpiniak

33

Acciones con parámetros

Tipos de pasaje de Parámetros - Ejemplo

Algoritmo CalcularAreaTriángulo

Lexico

base, altura $\in \mathbb{R}$ {base y altura del triángulo}
area $\in \mathbb{R}$ {área del triángulo}

Acción LeerDatos(resultado $x \in \mathbb{R}$, $y \in \mathbb{R}$)

Acción CalcAreaTriángulo2(dato baseTri, alturaTri $\in \mathbb{R}$,
resultado areaTri $\in \mathbb{R}$)

Acción MostrarResultados(dato a $\in \mathbb{R}$) \longrightarrow
Inicio

LeerDatos(base, altura)
CalcAreaTriángulo2(base, altura, area)
MostrarResultados(area)

Aunque siendo dato-resultado funciona bien en este caso porque no se modifica la variable a dentro de la acción, es muy peligroso para modificaciones futuras.

No debe hacerse!!!

Fin

No hace falta que el nombre de los parámetros formales sea distinto que el de los actuales. Pero es **aconsejable** hacerlo hasta tanto se adquiera más práctica. Esto tiene sus pro y contras. ¿cuáles?



2017 Lic. Ariel Ferreira Szpiniak

34

Acciones con parámetros

Tipos de Parámetros

Ejemplo III

Acción ProSum(dato $x, z \in \mathbb{R}$, dato-resultado $w \in \mathbb{R}$, resultado $prom \in \mathbb{R}$)

Lexico local

a $\in \mathbb{R}$

Inicio

{eo: $x=X_0$, $z=Z_0$, $w=W_0$ }

$x \leftarrow x+1$

$a \leftarrow x*2$

$w \leftarrow x+z+w$

$z \leftarrow z / 2$

$prom \leftarrow w / 4$

{ef: como deber}

Fin

¿Qué valores tienen las variables utilizadas en Prosum luego de la invocación de dicha acción?

Algoritmo Ejemplo1

lexico local

b, c, d, p $\in \mathbb{R}$

Inicio

d $\leftarrow 80$

c $\leftarrow 3$

b $\leftarrow 0$

ProSum(d,c,b,p)

Escribir(d,c,b,p)

Fin

Algoritmo Ejemplo3

lexico local

x, z, b $\in \mathbb{R}$

Inicio

x $\leftarrow 2$

z $\leftarrow 1$

b $\leftarrow 100$

ProSum(1,z,x,b)

Escribir(z,x,b)

Fin

Algoritmo Ejemplo2

lexico local

a, b, c, pr $\in \mathbb{R}$

Inicio

a $\leftarrow 8$

b $\leftarrow -6$

c $\leftarrow 10$

pr $\leftarrow 10$

ProSum(a,b,c,pr)

Escribir(a,b,c,pr)

Fin

Algoritmo Ejemplo4

lexico local

x, z, b $\in \mathbb{R}$

Inicio

x $\leftarrow 2$

z $\leftarrow 1$

b $\leftarrow 100$

ProSum(z,1,x,b)

Escribir(z,x,b)

Fin



2017 Lic. Ariel Ferreira Szpiniak

35

Procedimientos con parámetros

Estructura general

- Las acciones vistas se pueden traducir a procedimientos Pascal teniendo en consideración determinados detalles.

- En **Pascal** los tipos de pasaje de parámetros son solamente dos: **por valor** y **por referencia**.

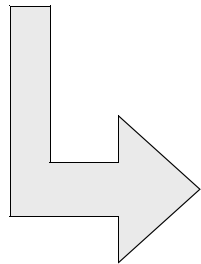
- Pascal no** tiene pasaje de parámetro **por resultado** ni **por valor/resultado**. Por lo tanto analizaremos como traducir o simular los tres tipos de pasaje de parámetros vistos anteriormente.



2017 Lic. Ariel Ferreira Szpiniak

36

Tipos de pasaje de parámetros en Pascal

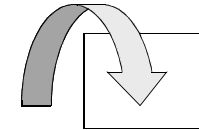


- Por Valor
- Por Referencia



Tipos de pasaje de parámetros en Pascal - Pasaje por Valor

Cláusula:



El pasaje de parámetros por **valor** se *traduce a Pascal* definiendo el parámetro formal solamente con su nombre y tipo, sin ninguna otra consideración adicional, es decir con la cláusula vacía (no se escribe nada) seguida del nombre y tipo del parámetro.

Ejemplo:

```
PROCEDURE LeerDatos(x: Integer, y: Real);
```



Tipos de pasaje de parámetros en Pascal - Pasaje por Referencia

Cláusula: **VAR**



Asocia el parámetro formal a la variable computacional que será pasada como parámetro actual. Cualquier modificación en el parámetro formal es almacenada en el lugar de memoria de dicha variable computacional. No se utiliza otra variable.

El pasaje de parámetros por **referencia** permite *simular en Pascal* los tipos de pasaje de parámetro por **resultado** y por **valor/resultado**. Para ello se define el parámetro formal con la cláusula **VAR** seguida del nombre y tipo del parámetro.

Ejemplo: `PROCEDURE Suma(x,y: Real, VAR result: Real)`



Procedimientos con parámetros Tipos de pasaje de parámetros

Notación Algorítmica	Pascal
dato	
dato-resultado	VAR
resultado	VAR

Es nada más que una forma de simularlo!!!

Ver ejemplos animados en Pascal de pasaje por valor y por referencia (sección "Materiales" del aula virtual)



Acciones En Pascal

- Se identifican los mismos componentes que en notación algorítmica.
- Le declaración de una acción se realiza mediante la palabra reservada **PROCEDURE**
- El **INICIO** y **FIN** se indica con **BEGIN** y **END**, donde, a diferencia que en un programa, el **END** es seguido de un punto un coma (;)



Procedimientos con parámetros Estructura general

1. *Cabecera* **PROCEDURE** <identificador> (lista de parámetros) ;
2. *Declaraciones* variables, constantes, acciones y funciones propias de la acción
3. *Sentencias ejecutables* **BEGIN** <sentencia>;
...
<sentencia>
END;

4. Ubicación

Deben aparecer antes del **BEGIN** del programa y después de las declaraciones de identificadores del mismo (variables, etc).



Acciones con parámetros Ejemplo

Notación Algorítmica

Acción CalcAreaTriángulo2(dato baseTri ∈ R, dato alturaTri ∈ R, resultado areaTri ∈ R)

Inicio

areaTri ← (baseTri*alturaTri)/2

Fin

Pascal

```
PROCEDURE CalcAreaTriángulo2(baseTri: Real, alturaTri: Real;
    VAR areaTri: Real)
```

BEGIN

```
    areaTri := (baseTri*alturaTri)/2
```

END;



Acciones con parámetros Ejemplo

```
PROGRAM CalcularAreaTriangulo;
```

```
VAR
```

```
base, altura: Real;           {base y altura del triángulo}
```

```
area: Real;                   {área del triángulo}
```

```
PROCEDURE LeerDatos(VAR x,y: Real);
```

```
BEGIN
```

```
    WRITE('Ingrese la base: ');
```

```
    READ(x);
```

```
    WRITE('Ingrese el altura: ');
```

```
    READ(y)
```

```
END;
```

```
PROCEDURE CalcAreaTriangulo2(baseTri, alturaTri: Real;
    VAR areaTri: Real);
```

```
BEGIN
```

```
    areaTri := (baseTri*alturaTri)/2
```

```
END;
```

```
PROCEDURE MostrarResultados(a: Real);
```

```
BEGIN
```

```
    Writeln('El área del triángulo es: ',a)
```

```
END;
```

```
{SIGUE EN LA PRÓXIMA DIAPOSITIVA}
```

¿qué sucede si no pongo VAR?

x: Real; y: Real

Ojo

Total de dígitos del número real contando parte entera, punto y dígitos decimales (alineado a derecha)

a:5:3

dígitos decimales de un número real



Acciones con parámetros

Ejemplo

```
{CONTINÚA DE LA DIAPOSITIVA ANTERIOR}
{ESTE SERÍA EL PROGRAMA PRINCIPAL}
BEGIN

  LeerDatos(base, altura);
  CalcAreaTriangulo2(base, altura, area);
  MostrarResultados(area)

END.
```



Acciones con parámetros

Ejemplo II

Notación Algorítmica

Acción ProSum(dato $x, z \in \mathbb{Z}$, dato-resultado $w \in \mathbb{Z}$, resultado $prom \in \mathbb{R}$)

Lexico local

$a \in \mathbb{Z}$

Inicio

$x \leftarrow x+1$

$a \leftarrow x*2$

$w \leftarrow x+z+w$

$z \leftarrow z / 2$

$prom \leftarrow w / 4$

Fin

Pascal

PROCEDURE ProSum(x, z : Integer, VAR w : Integer, VAR $prom$: Real)

VAR

a : Integer

BEGIN

$x := x+1$;

$a := x*2$;

$w := x+z+w$;

$z := z / 2$;

$prom := w / 4$

END;



Bibliografía

- Watt, David: Programming Language Concepts and Paradigms, Prentice-Hall International Series in Computer Science (1990). Cap. 5
- Biondi, J. y Clavel, G. “Introducción a la Programación. Tomo 1: Algorítmica y Lenguajes”: (pags. 181 - 190)
- Scholl, P. y Peyrin, J.-P. “Esquemas Algorítmicos Fundamentales: Secuencias e iteración”. (pags. 71 - 87)
- Quetglás, Toledo, Cerverón. “Fundamentos de Informática y Programación”. Capítulo 3. <http://robotica.uv.es/Libro/Indice.html>
 - Programación Modular (pags 110 - 111)
- Joyanes Aguilar, L., “Programación en Turbo Pascal”. Mc Graw Hill, 1993.
- Grogono, P., “Programación en Pascal”, Wilmington, Adisson-Wesley, 1996.
- Wirth, N. and K. Jensen, “Pascal: User Manual and Report”, 4ª ed., New York, Springer-Verlag, 1991 (traducción de la primera edición “Pascal: Manual del Usuario e Informe”, Buenos Aires, El Ateneo, 1984).



Introducción a la

Algorítmica y Programación

(3300)

Prof. Ariel Ferreira Szpiniak - aferreira@exa.unrc.edu.ar

Departamento de Computación

Facultad de Cs. Exactas, Fco-Qcas y Naturales

Universidad Nacional de Río Cuarto

Teoría 5 (continuación)

Modularización, Abstracción

Acciones



Pasaje de parámetros en C

```
// C solo tiene pasaje de parámetro por dato (copia). El pasaje por referencia, como tal, tampoco es
// soportado, sino que se imita utilizando las direcciones de memoria de las variables (*).
#include <stdio.h>
#include <stdlib.h>

float base, altura; //base y altura del triangulo
float area;          //area del triangulo

void LeerDatos(float *x, float *y){
    printf("Ingrese la base: \n ");
    scanf("%f",&(*x));
    printf("Ingrese el altura: \n");
    scanf("%f",&(*y));
}

void CalcAreaTriangulo2(float baseTri, float alturaTri, float *areaTri){
    *areaTri = (baseTri*alturaTri)/2;
}

void MostrarResultados(float a){
    printf("El area del triangulo es: %f \n",a);
}

void main(){
    LeerDatos(&base, &altura);
    CalcAreaTriangulo2(base, altura, &area);
    MostrarResultados(area);
}
```



Pasaje de parámetros en C

```
// C solo tiene pasaje de parámetro por dato (copia). El pasaje por
// referencia, como tal, tampoco es
// soportado, sino que se imita utilizando las direcciones de memoria
// de las variables (*).

#include <stdio.h>
#include <stdlib.h>

float base, altura; //base y altura del triangulo
float area;          //area del triangulo

void LeerDatos(float *x, float *y){
    printf("Ingrese la base: \n ");
    scanf("%f",&(*x));
    printf("Ingrese el altura: \n");
    scanf("%f",&(*y));
}

}
```



Pasaje de parámetros en C

```
void CalcAreaTriangulo2(float baseTri, float alturaTri, float *areaTri){
    *areaTri = (baseTri*alturaTri)/2;
}

void MostrarResultados(float a){
    printf("El area del triangulo es: %f \n",a);
}

void main(){
    LeerDatos(&base, &altura);
    CalcAreaTriangulo2(base, altura, &area);
    MostrarResultados(area);
}
```



Introducción a la Algorítmica y Programación (3300)

Prof. Ariel Ferreira Szpiniak - aferreira@exa.unrc.edu.ar
Departamento de Computación
Facultad de Cs. Exactas, Fco-Qcas y Naturales
Universidad Nacional de Río Cuarto

Teoría 5 - Anexo

Ejemplo de Modularización usando Acciones

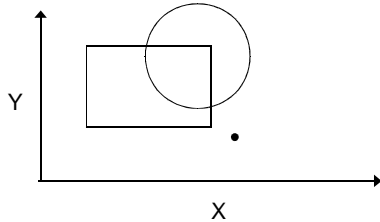


Ejemplo

Dados un círculo, un rectángulo y un punto, determinar la posición del punto respecto a las dos figuras.

Análisis

Para comprender el problema debo contextualizarlo en el Plano, esto es:



¿cómo los represento?
el cuadrado con 2 puntos
el círculo con el punto central y radio

- ¿Qué resultados son posibles?

Las 4 posibilidades de posición del punto respecto de las 2 figuras.

- ¿Cómo puedo obtener los posibles resultados?

Puedo averiguar la pertenencia (estar dentro de) a cada una de las figuras y luego determinar la solución. Para el cuadrado puedo lograrlo comparando con los ejes de abscisas y ordenadas. Para el caso del punto con la distancia del mismo al centro respecto del radio.



2017 Lic. Ariel Ferreira Szpiniak

53

Modularizar

Acciones

- Obtener los Datos
- Pertenencia al Cuadrado
- Pertenencia al Círculo

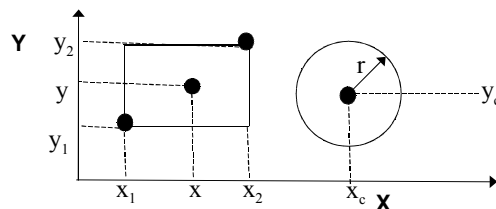


2017 Lic. Ariel Ferreira Szpiniak

54

Modularizar

Obtener los Datos: Modularizado como tres Acciones.



Como acotación del problema asumimos que los puntos que representan el cuadrado son el inferior izquierdo y el superior derecho (en ese orden), y que el radio es positivo.



2017 Lic. Ariel Ferreira Szpiniak

55

Obtener los Datos

Parámetros formales de salida

```

Acción LeerPunto(resultado xp, yp ∈ R)
Inicio
  Escribir('Ingrese la coordenada x e y del punto')
  Leer(xp,yp)
Fin

Acción ObtenerCuadrado(resultado x1,y1,x2,y2 ∈ R)
Inicio
  Repetir
    Escribir('Coordenada del punto inf izq del cuadrado')
    LeerPunto(x1,y1) ← Parámetros actuales
    Escribir('Coordenada del punto sup der del cuadrado')
    LeerPunto(x2,y2) ← Parámetros actuales
  Hasta que (x1 < x2 y y1 < y2)
Fin

Acción ObtenerCírculo(resultado xc,yc,r ∈ R)
Inicio
  Escribir('Coordenada del punto central del círculo')
  LeerPunto(xc,yc) ← Parámetros actuales
  Repetir
    Escribir('Radio del círculo')
    Leer(r)
  Hasta que r > 0
Fin
  
```

Parámetros formales de salida

Parámetros formales de salida

Parámetros formales de salida

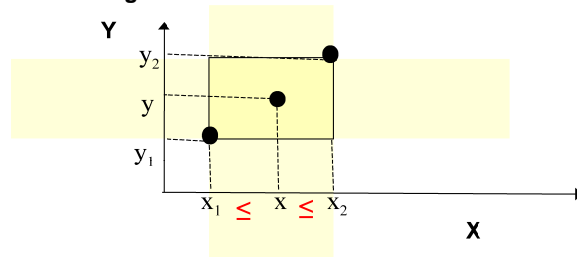


2017 Lic. Ariel Ferreira Szpiniak

56

Modularizar

- **Pertenece Rectángulo:** Modularizado como una Acción.



Para que el punto este dentro del rectángulo, debe estar en la intersección de las dos franjas, es decir, debe cumplir con las 2 condiciones (comparaciones) de su x e y.

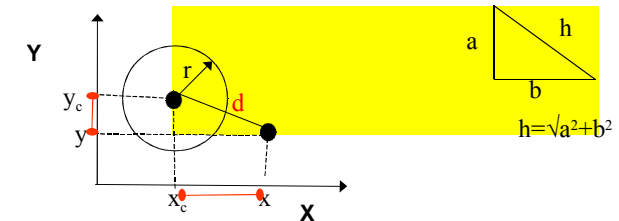


2017 Lic. Ariel Ferreira Szpiniak

57

Modularizar

- **Pertenece a Círculo:** Modularizado como una Acción.



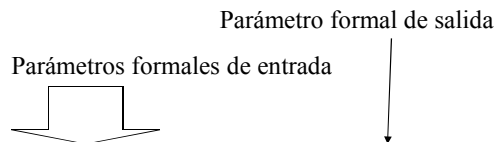
Para determinar la pertenencia del punto al círculo, podemos comprobar si la distancia del mismo a centro es menor o igual que el radio. Para ello podemos determinar la distancia con Pitágoras, donde la distancia a averiguar es la hipotenusa.



2017 Lic. Ariel Ferreira Szpiniak

58

- **Pertenece Cuadrado:**

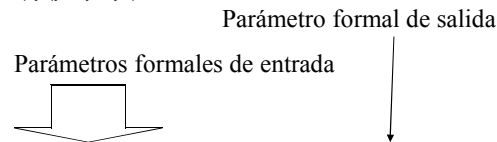


Acción PertAlCuadrado(dato x,y,x1,y1,x2,y2 ∈ R, resultado perteneceCuad ∈ Lógico)

Inicio
perteneceCuad <- (x1 ≤ x ≤ x2) y (y1 ≤ y ≤ y2)

Fin

- **Pertenece Círculo:**



Acción PertAlCírculo(dato x,y,xc,yc,r ∈ R, resultado perteneceCirc ∈ Lógico)

Inicio
perteneceCirc <- (r ≥ √((x-xc)² + (y-yc)²))

Fin



2017 Lic. Ariel Ferreira Szpiniak

59

Algoritmo FigurasYPunto

Léxico

a1, a2, b1, b2, c1, c2, d1, d2, radio ∈ R {Datos entrada}
estaCuad, estaCirc ∈ Lógico {Soluciones Parciales}

Acción LeerPunto(resultado xp, yp ∈ R)

Acción ObtenerCuadrado(resultado x1,y1,x2,y2 ∈ R)

Acción ObtenerCírculo(resultado xc,yc,r ∈ R)

Acción PertAlCuadrado(dato x,y,x1,y1,x2,y2 ∈ R, resultado perteneceCuad ∈ Lógico)

Acción PertAlCírculo(dato x,y,x1,y1,x2,y2 ∈ R, resultado perteneceCirc ∈ Lógico)

Inicio

ObtenerCuadrado(b1,b2,c1,c2)

ObtenerCírculo(d1,d2,radio)

Escribir('Coordenada del punto a determinar su posición')

LeerPunto(a1,a2)

PertAlCuadrado(a1,a2,b1,b2,c1,c2,estaCuad)

PertAlCírculo(a1,a2,d1,d2,radio,estaCirc)

según

estaCuad y estaCirc: Escribir('Está en ambos')

¬estaCuad y estaCirc: Escribir('Está sólo en el Círculo')

estaCuad y ¬estaCirc: Escribir('Está sólo en el Cuadrado')

¬estaCuad y ¬estaCirc: Escribir('Está fuera de ambos')

fsegún

Fin



2017 Lic. Ariel Ferreira Szpiniak

60

Citar/Atribuir: Ferreira, Szpiniak, A. (2017). Teoría 5: Modularización, Abstracción. Acciones. Introducción a la Algorítmica y Programación (3300). Departamento de Computación. Facultad de Cs. Exactas, Fco-Qcas y Naturales. Universidad Nacional de Río Cuarto.

Usted es libre para:

Compartir: copiar y redistribuir el material en cualquier medio o formato.

Adaptar: remezclar, transformar y crear a partir del material.

El licenciente no puede revocar estas libertades en tanto usted siga los términos de la licencia.

Bajo los siguientes términos:



Atribución: Usted debe darle crédito a esta obra de manera adecuada, proporcionando un enlace a la licencia, e indicando si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo del licenciente.



Compartir Igual: Si usted mezcla, transforma o crea nuevo material a partir de esta obra, usted podrá distribuir su contribución siempre que utilice la misma licencia que la obra original.

<https://creativecommons.org/licenses/by-sa/2.5/ar/>

