Capítulo 6

Parámetros de entrada

Objetivos

El objetivo de este capítulo es extender la sintaxis de definición de procedimientos a fin de permitir que se comparta información entre el módulo que llama y el módulo que es llamado.

Esto brindará la posibilidad de flexibilizar el comportamiento del procedimiento obteniendo, de esta forma, mejores resultados.

Temas a tratar

- ✓ Comunicación entre módulos
- ✓ Declaración de parámetros
- ✓ Un ejemplo sencillo
- ✓ Ejemplos
- ✓ Restricción en el uso de los parámetros de entrada
- ✓ Conclusiones

6.1 Comunicación entre módulos

La metodología Top-Down se basa en la descomposición del problema original en partes más simples. Esto facilita su resolución dando origen a diversos módulos, cada uno de ellos con una función bien definida. Por otro lado, si es posible contar con un conjunto de subproblemas ya resueltos correctamente, estos podrán ser combinados para expresar soluciones más complejas.

Por ejemplo, podría ser útil contar con un procedimiento que permitiera conocer la cantidad de flores que el robot lleva en su bolsa, o tal vez podría desarrollarse un módulo que le permitiera al robot realizar un rectángulo cuyo alto y ancho se indicara durante la ejecución del programa.

Situaciones como las anteriores requieren que los procedimientos compartan información con el módulo que los invoca.

Los módulos desarrollados en el capítulo 5 no cuentan con esta posibilidad y su comportamiento es muy limitado ya que hacen siempre lo mismo. Por ejemplo, el procedimiento JuntarPapeles definido en 5.2 o el procedimiento Cuadrado definido en 5.6. Cada uno de estos procedimientos, para funcionar, sólo requiere que el robot esté posicionado en la esquina donde deben comenzar a ejecutarse. Al terminar, volverán a dejar al robot posicionado en ese mismo lugar.

Este tipo de comportamiento resulta muy acotado. Por ejemplo, ¿Qué pasaría si ahora hubiera que pedirle al procedimiento JuntarPapeles que retorne la cantidad de papeles que recogió? o ¿Qué pasaría si se quisiera realizar un cuadrado de lado 2 y otro de lado 5 utilizando el mismo procedimiento?

Cuando se quiere que el procedimiento interactúe con el módulo que lo llama es preciso compartir información.

La información es compartida entre módulos a través de los parámetros.

Se puede decir entonces que se denomina parámetro a la información que se intercambia entre módulos.

En general, existen tres tipos de parámetros que interesan considerar:

Parámetro de entrada: a través de este tipo de parámetro un procedimiento puede recibir información del módulo que lo llama. Por ejemplo, podría modificarse el procedimiento Cuadrado definido en el ejemplo 5.6 para que reciba información acerca del tamaño del cuadrado a realizar. De esta forma, el mismo procedimiento podría ser utilizado para realizar cuadrados de diferentes tamaños.

Parámetro de salida: este tipo de parámetro permite que el procedimiento llamado genere información y pueda enviarla al módulo que lo llamó. Note que en el caso anterior, la información era generada por el módulo que llamaba, en cambio ahora, la información la genera el módulo llamado. Por ejemplo, podría ser muy útil contar con un procedimiento que permita conocer la cantidad de papeles que hay en una esquina. Este procedimiento tendría que contar los papeles y a través de un parámetro de salida, permitir que el módulo que lo llamó pueda tener acceso a este valor.

Parámetro de entrada/salida: este tipo de parámetro permite una comunicación más estrecha entre los módulos ya que la información viaja en ambos sentidos. A través de él, un dato, enviado por el módulo que llama, puede ser utilizado y modificado por el módulo llamado. Por ejemplo, el módulo JuntarPapeles podría recibir la cantidad de papeles juntados hasta el momento y actualizarla con el total de papeles de la esquina donde se encuentra parado.

6.2 Declaración de parámetros

La sintaxis a utilizar para definir un procedimiento con parámetros es la siguiente:

```
procedimiento nombre ( lista de parámetros )
variables
```

```
  {declare aquí las variables de este módulo}
comenzar
  {acciones a realizar dentro del módulo}
fin
```

La lista de parámetros, indicada a continuación del nombre del procedimiento, define cada uno de los parámetros a utilizar; es decir, la información a compartir entre el módulo llamado y el módulo que llama. Esta lista es opcional. Por ejemplo, los procedimientos definidos en el capítulo 5 no la utilizaban. En caso de utilizarse, **la declaración de cada uno de los parámetros que la componen tiene tres partes**:

1. En primer lugar debe indicarse la **clase de parámetro** a utilizar. Los tres tipos de parámetros que utilizaremos en la sintaxis del robot son los descriptos en la sección 6.1. Se utilizará las letras **en** para indicar un parámetro de entrada, las letras **sa** para indicar un parámetro de salida y las letras **es** para indicar un parámetro de entrada/salida.

2. Luego de definir la clase de parámetro a utilizar debe especificarse **su nombre**. Este identificador será utilizado por el procedimiento para recibir y/o enviar información.

3. Finalmente, a continuación del nombre del parámetro y precedido por ":" debe indicarse **el tipo** de dato al cual pertenece el parámetro. En la sintaxis del robot, las opciones aquí son numero, lógico o texto.

Los parámetros se separan dentro de esta lista por ";". Por ejemplo, a continuación puede verse un módulo con su lista de parámetros:

```
procedimiento ProcedimientoDePrueba (en dato:numero; es TodoBien:logico)
variables
      {declare aquí las variables de este módulo}
comenzar
      {acciones a realizar dentro del módulo}
fin
```

dónde dato es un parámetro de entrada numérico, y TodoBien es un parámetro de entrada/salida lógico ó booleano. Estos parámetros indicados en el encabezado del procedimiento son denominados parámetros formales.

Se denominan parámetros formales a los indicados en el encabezado del procedimiento y se denominan parámetros actuales o reales a los utilizados en la invocación del procedimiento.

Notemos la importancia de la letra que precede al nombre del parámetro. Por su intermedio puede conocerse la manera en que se compartirá la información, es decir, si es de entrada o de entrada\salida.

6.3 Un ejemplo sencillo

Analicemos la figura 6.1. En ella aparecen varios cuadrados de diferente tamaño. Básicamente, hay tres formas de resolver este tipo de problemas:

1. Sin utilizar modularización. Esta es la forma en que han sido resueltos los problemas en los capítulos 2 y 3. Sin embargo, hemos analizado en el capítulo 5 las ventajas de aplicar la modularización en el diseño de soluciones, por lo que resulta recomendable su utilización.

2. Utilizando procedimientos sin parámetros. Si se utilizara una idea similar al procedimiento Cuadrado definido en 5.6, debería haber tantos procedimientos diferentes como cuadrados de distinto tamaño de lado aparezcan en el recorrido.

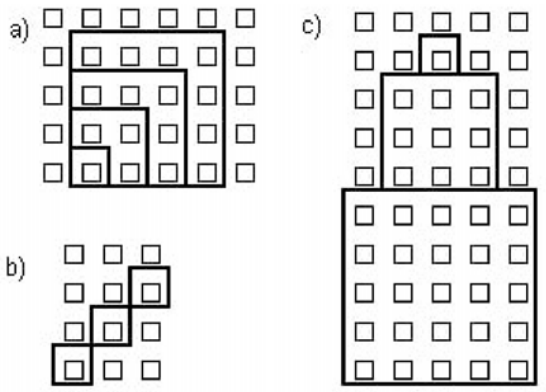


Figura 6.1: ¿Cómo se hace para que un mismo procedimiento sirva para realizar todos los cuadrados?

3. Utilizando un único procedimiento cuadrado al cual pueda decírsele la longitud del lado a realizar en cada caso.

Ejemplo 6.1: Programe al robot para que realice el recorrido a) de la figura 6.1.

Estos son los programas correspondientes al recorrido a) según las opciones 1 y 2 indicadas anteriormente:

```
programa MuchosCuadradosV1
{Sin módulos}
comenzar
  iniciar
  {Cuadrado de lado 1}
  repetir 4
    comenzar
    mover
    derecha
    fin

{Cuadrado de lado 2}
repetir 4
  comenzar
  repetir 2
    mover
  derecha
  fin

{Cuadrado de lado 3}
repetir 4
  comenzar
  repetir 3
    mover
  derecha
  fin

{Cuadrado de lado 4}
repetir 4
  comenzar
  repetir 4
    mover
  derecha
  fin
fin


Programa MuchosCuadradosV2
{Con módulos pero SIN parámetros}
Subprogramas
  procedimiento cuadrado1
  comenzar
    repetir 4
      comenzar
      mover
      derecha
      fin
  fin

  procedimiento cuadrado2
  comenzar
    repetir 4
      comenzar
      repetir 2
```

```
      mover
    derecha
    fin
fin

procedimiento cuadrado3
comenzar
  repetir 4
    comenzar
    repetir 3
      mover
    derecha
    fin
fin
```

```
  procedimiento cuadrado4
  comenzar
    repetir 4
      comenzar
      repetir 4
        mover
      derecha
      fin
  fin

comenzar
  iniciar
  Cuadrado1
  Cuadrado2
  Cuadrado3
  Cuadrado4
fin
```

Ambas soluciones presentan los siguientes inconvenientes:

• Son difíciles de generalizar: en el programa MuchosCuadradosV1 se ha realizado todo el recorrido sin descomponer el problema. Es decir, se ha analizado e implementado cada uno de los cuadrados secuencialmente. Además podemos observar que es bastante costosa su lectura.

En el programa MuchosCuadradosV2 se han utilizado cuatro procedimientos (que pudieron haber sido desarrollados previamente) pero si para cada tamaño de cuadrado a realizar es necesario escribir un procedimiento específico, se debe conocer de antemano el tamaño del cuadrado para poder escribir el procedimiento correspondiente. Además al aumentar la cantidad de cuadrados de diferente tamaño de lado, tendríamos que aumentar también la cantidad de procedimientos a escribir.

• No se está aprovechando la idea de que todos los cuadrados requieren del mismo recorrido, sólo sería preciso cambiar la longitud del lado. Es decir, todos se basan en repetir 4 veces: hacer el lado y girar a la derecha. Lo único que cambia entre un cuadrado y otro es la cantidad de cuadras que se deben recorrer en línea recta para hacer el lado correspondiente.

En realidad, lo adecuado sería poder indicarle al procedimiento Cuadrado la longitud del lado que debe realizar. Para esto, el procedimiento cuadrado debería recibir un valor que represente la longitud del lado a través de un parámetro de entrada.

La solución toma la siguiente forma:

```
procedimiento cuadrado (en lado:numero)   (1)
comenzar
  repetir 4
    comenzar
    repetir lado  (2)
      mover
    derecha
    fin
fin
```

En este caso, la lista de parámetros (lo que aparece entre paréntesis en la línea (1)) está formada por un único parámetro llamado lado. Según esta declaración, se trata de un parámetro de entrada pues su nombre se encuentra precedido por la letra E y corresponde al tipo numero. El identificador lado será utilizado por el procedimiento para recibir la información.

Esto quiere decir que el procedimiento cuadrado podrá utilizar el parámetro lado en el cuerpo del procedimiento. Pero por tratarse de un parámetro de entrada, el módulo que llamó al procedimiento cuadrado no podrá recibir, a través de dicho parámetro, ninguna respuesta.

Cuando el procedimiento cuadrado sea invocado y reciba en lado la longitud del lado que debe realizar, este valor será utilizado por la línea (2) para efectuar el recorrido pedido.

La resolución del recorrido a) de la figura 6.1 utilizando este procedimiento es la siguiente:

```
programa MuchosCuadradosV3
subprogramas
  procedimiento cuadrado(en lado : numero)    (3)
  comenzar
    repetir 4
      comenzar
      repetir lado                    (4)
        mover
      derecha
      fin
  fin

comenzar
  iniciar                             (1)
  cuadrado(1)                         (5)
  cuadrado(2)
  cuadrado(3)
  cuadrado(4)
fin
```

La ejecución del programa MuchosCuadradosV3 comienza por la línea (1). La primera invocación al procedimiento se realiza en (2). Notemos que ahora, a continuación del nombre del procedimiento se indica, en la invocación, el valor que recibirá el procedimiento cuadrado en el parámetro lado. Cuando el procedimiento cuadrado recibe el control, en la línea (3), lado tendrá el valor 1 y por lo tanto al ejecutarse realizará un cuadrado de lado 1. Es decir, en la línea (4) la instrucción repetir lado se reemplaza por repetir 1 ya que el parámetro lado ha tomado el valor 1.

Una vez que el procedimiento termina, la ejecución continúa en la línea (5) donde se vuelve a invocar al procedimiento cuadrado pero ahora se le pasa el valor 2 como parámetro. Esto es recibido por el procedimiento, en la línea (3), por lo que realizará un cuadrado de lado 2. Esto se repite dos veces más invocando al procedimiento cuadrado con los valores 3 y 4 como parámetro.

En la solución anterior puede verse que:

- El código es más claro que en las dos versiones anteriores, facilitando de esta forma su lectura y comprensión. En el programa principal se nota claramente que se realizan cuatro cuadrados donde el primero tiene lado 1, el segundo lado 2, el tercero lado 3 y el cuarto lado 4.

- El procedimiento cuadrado, a través de su parámetro de entrada, puede ser utilizado para realizar un cuadrado de cualquier tamaño. Esto resuelve el problema de generalización presentado en MuchosCuadradosV2.

- ¿Explique por qué si se cambia el nombre del parámetro formal, lado, el programa principal no cambia?

6.4 Ejemplos

Ejemplo 6.2: Programe al robot para que realice el recorrido c) de la figura 6.1.

Retomando el análisis realizado en el ejemplo anterior, se comenzará resolviendo este problema sin utilizar modularización. La modularización tiene que ver con el estilo de programación. Un programa que no utilice módulos tendrá algunas desventajas con respecto al que si los utiliza.

```
programa TorreSinMódulos
comenzar
  iniciar
  repetir 4
    comenzar
    repetir 5
      mover
    derecha
    fin
  Pos(2,6)
  repetir 4
    comenzar
    repetir 3
      mover
    derecha
    fin
  Pos(3,9)
  repetir 4
    comenzar
    repetir 1
      mover
    derecha
    fin
fin
```

Una de las principales desventajas de esta solución es la falta de claridad en el programa ya que para comprender lo que hace es necesario analizar cada una de las líneas que lo componen. Sería más fácil de comprender si se escribiera el cuerpo del programa de la siguiente forma:

```
comenzar
  iniciar
  {realizar un cuadrado de lado 5}
  Pos (2,6)
  {realizar un cuadrado de lado 3}
```

```
  Pos (3,9)
  {realizar un cuadrado de lado 1}
fin
```

Si consideramos esta solución, vemos que sería conveniente utilizar el módulo cuadrado con el parámetro de entrada lado visto anteriormente. El programa completo modularizado puede escribirse como sigue:

```
programa TorreConTresMódulos
subprogramas
  procedimiento cuadrado(en lado : numero)
  comenzar
    repetir 4
      comenzar
      repetir lado
        mover
      derecha
      fin
  fin
comenzar
  iniciar
  cuadrado(5)
  Pos(2,6)
  cuadrado(3)
  Pos(3,9)
  cuadrado(1)
fin
```

Como podemos observar, la primera vez que se invoca al módulo cuadrado desde el programa principal, el parámetro lado recibe el valor 5, luego será invocado con valor 3 y finalmente con el valor 1.

A continuación se presenta otra opción de solución que muestra un programa principal que utiliza una variable llamada long (para representar el lado), el módulo cuadrado y el parámetro lado:

```
programa cuadrados
subprogramas
  procedimiento cuadrado (en lado : numero)
  comenzar
    repetir 4
      comenzar
      repetir lado
        mover
      derecha
      fin
  fin

variables
long : numero

comenzar
  iniciar
```

```
  long:=5
  cuadrado(long)                                    (1)
  Pos(2,6)
  long := long – 2
  cuadrado(long)                                    (2)
  Pos(3,9)
  long := long – 2
  cuadrado(long)                                    (3)
fin
```

A partir de la solución presentada se pueden analizar algunos aspectos:

- Las invocaciones (1), (2) y (3) son idénticas, sólo debemos notar que cada una envía al módulo un valor de long actualizado. Inicialmente le enviará el valor 5, luego el valor 3 y finalmente el valor 1, como resultado de la actualización de la variable long. Siguiendo este razonamiento entonces ¿podríamos utilizar una estructura repetir 3 para que la legibilidad del cuerpo del programa mejore? La respuesta es SÍ pero debemos analizar algunos aspectos adicionales para poder lograrlo. En párrafos posteriores trataremos esto.

- Existe una relación entre la esquina donde comienza el recorrido del segundo cuadrado y el tamaño del lado del primer cuadrado. Lo mismo ocurre entre el tercer cuadrado y el segundo cuadrado.

La siguiente solución contempla los aspectos analizados anteriormente:

```
programa cuadrados
subprogramas
  procedimiento cuadrado (en lado : numero)  (3)
  comenzar
    repetir 4
      comenzar
      repetir lado
        mover
      derecha
      fin
  fin

variables
long : numero

comenzar
  iniciar
  long := 5                              (1)
  {realizar los tres cuadrados }
  repetir 3
    comenzar
    cuadrado(long)                       (2)
    Pos(PosAv + 1, PosCa + long)         (4)
    long := long - 2                     (5)
    fin
fin
```

Sigamos la solución: en (1) se asigna el valor inicial 5 a long ya que el primer cuadrado de la figura es de este tamaño. Este valor es utilizado en (2) para llamar por primera vez al procedimiento cuadrado. El parámetro formal lado recibe en (3) el valor 5 y lo utiliza dentro del procedimiento para realizar el cuadrado correspondiente.

Cuando el procedimiento termina, retorna al programa principal el cual se encarga de hacer las modificaciones necesarias para realizar el próximo cuadrado, esto es: posicionar al robot (4) y decrementar la longitud del lado del cuadrado (5). En la línea (4) estamos reposicionando al robot, esto significará ubicarlo en la esquina formada por: la avenida en la que se encontraba posicionado aumentada en 1 y en la calle en la que se encontraba posicionado aumentada en el tamaño del lado del cuadrado anterior.

La próxima invocación al procedimiento se hará con long valiendo 3, de donde en (3) lado recibirá este valor y hará el cuadrado de lado 3.

Finalmente, esto se repite una vez más realizando el cuadrado de lado 1.

Notemos que al terminar el programa, long vale -1 y el robot, a diferencia de las soluciones anteriores, está posicionado en (4,10).

En consecuencia, podemos afirmar que esta última solución resuelve el problema de generalidad de las dos soluciones anteriores.

Ejemplo 6.3: Programar al robot para que realice el recorrido a) de la figura 6.1 utilizando una variable para indicar la longitud del lado de cada cuadrado.

Puede observarse que el procedimiento principal consiste en realizar un cuadrado, el cual va cambiando su tamaño, comenzando por un cuadrado de lado 1 hasta uno de lado 4.

El primer análisis del algoritmo es:

```
{recorrido de cuadrados con igual origen}
{realizar los cuatro cuadrados, donde para cada uno es necesario …}
{hacer un cuadrado del lado correspondiente}
{incrementar el lado del cuadrado}
```

El cuadrado, como puede observarse, debe ir modificando su tamaño. Por lo tanto, es necesario definir un atributo que represente esta condición. Se utilizará para ello la variable largo.

Detallando lo anterior se puede escribir el siguiente programa en DaVinci:

```
programa cuadrados
subprogramas
  procedimiento cuadrado (en lado:numero)
  comenzar
    repetir 4
       comenzar
      repetir lado
        mover
      derecha
      fin
  fin

variables
largo : numero

comenzar
  iniciar
  {valor inicial, el primer cuadrado es de lado 1}
  largo := 1
  repetir 4
    comenzar
    cuadrado (largo)
    {se incrementa el valor que indica el largo del lado}
    largo := largo + 1
    fin
fin
```

A continuación se describe la ejecución del algoritmo. La segunda instrucción asigna el valor 1 a la variable largo, el cual se corresponde con el tamaño del lado del primer cuadrado. Cuando se llama al procedimiento cuadrado, se le manda al mismo la información correspondiente a la medida del lado. En esta primera invocación se le pasa el valor 1. Notemos que en el encabezado del procedimiento se encuentra definido un dato, lado, el cual se corresponde con la variable largo utilizada en la invocación; pero que, como se observa, se lo llama con nombre diferente. El valor de la variable largo se copia en el dato lado. A partir de este momento, lado puede ser utilizado en el procedimiento cuadrado con el valor que recibió. Observemos también que cuando se indica que el valor de largo se copia en lado es equivalente a asignar a lado el valor de largo.

El procedimiento realiza un cuadrado de lado 1. Cuando el mismo finaliza, la instrucción siguiente a cuadrado consiste en aumentar en uno el valor de la variable largo, teniendo ahora el valor 2. Esto se repite tres veces más, completando el recorrido.

Ejemplo 6.4: Se desea programar al robot para que realice el recorrido de la figura 6.2

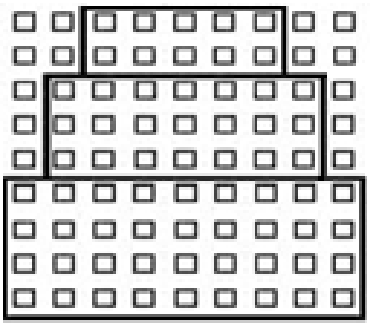


Figura 6.2: Torre de Rectángulos

Puede verse que la figura presenta diferentes rectángulos. El primero (el de más abajo) de 9 cuadras de base por 4 de alto, y el último (el de más arriba) de 5 por 2, o sea que cada rectángulo difiere con su superior en 2 cuadras de base y 1 cuadra en su altura.

El diseño Top-Down correspondiente al problema se muestra en la figura 6.3. El esquema del algoritmo quedará como:

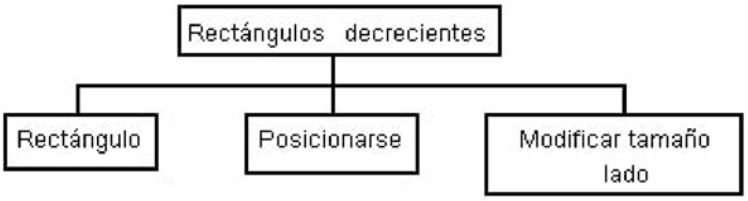


Figura 6.3: Descomposición Top-Down del ejemplo 6.4

```
{ torre de rectángulos }
{ realizar los cinco  rectángulos, como sigue ... }
{ hacer un rectángulo }
{ posicionarse para el siguiente }
{ modificar las dimensiones }
```

El rectángulo, a diferencia del cuadrado del ejemplo anterior, necesita dos elementos para indicar su tamaño. Por lo tanto, es necesario definir dos variables que representen esta condición. Para ello se utilizarán las variables ancho y alto.

Detallando lo anterior, la implementación de esta solución en DaVinci es:

```
programa rectangulos
subprogramas
  procedimiento Rectangulo (en base : numero; en altura : numero)
  comenzar
    repetir 2
      comenzar
      repetir altura
        mover
      derecha
      repetir base
        mover
      derecha
      fin
  fin

variables
ancho : numero
alto : numero

comenzar
  iniciar
  ancho := 9                          (1)
  alto := 4                           (2)
  repetir 3
    comenzar
    Rectangulo (ancho, alto)          (3)
    Pos (PosAv+1, PosCa + alto)       (4)
    ancho := ancho -  2               (5)
    alto := alto - 1                  (6)
    fin
fin
```

A continuación se describe la ejecución del algoritmo. En (1) y (2) se define el tamaño del primer rectángulo (el de más abajo). Cuando se llama al procedimiento Rectangulo, se le manda al mismo información correspondiente a la medida de sus lados. En esta primera invocación se le pasan los valores 9 y 4 respectivamente.

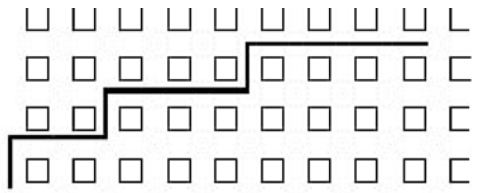
El procedimiento realiza un rectángulo de 9 por 4. Cuando finaliza, se devuelve el control al programa principal en la línea siguiente a (3) donde se posiciona al robot para realizar el siguiente rectángulo (4). Luego en (5) y (6) se modifican adecuadamente los valores que definen alto y ancho del próximo rectángulo.

En el problema planteado se observa que es necesario que el módulo reciba información. En este caso, el procedimiento Rectangulo recibe dos parámetros de entrada, base y altura, que son enviados por el programa principal a través de las variables ancho y alto, respectivamente.

Ejemplo 6.5: Supongamos que se dispone del siguiente procedimiento:

```
procedimiento escalon(en entra1 : numero, en entra2 : numero)
```

donde entra1 corresponde a la altura y entra2 corresponde al ancho de un escalón. Utilice el procedimiento anterior para resolver el recorrido de la figura 6.4.



6.5 Restricción en el uso de los parámetros de entrada

En los ejemplos anteriores se han desarrollado procedimientos con parámetros de entrada y en todos los casos, la información recibida de esta forma ha sido utilizada como "de lectura". En otras palabras, en ninguno de los ejemplos se ha asignado un valor sobre un parámetro de entrada.

Esta restricción es conceptual. Desde el punto de vista conceptual, no tiene sentido modificar el valor de un parámetro de entrada ya que es información recibida desde el módulo que realizó la invocación, el cual no espera recibir ninguna respuesta a cambio.

6.6 Conclusiones

En este capítulo se han presentado los parámetros de entrada como medio de comunicación entre módulos.

Este tipo de parámetros permite que el procedimiento llamado reciba información de quien lo llama. Dicha información podrá ser utilizada internamente por el procedimiento para adaptar su comportamiento.

Utilizando parámetros de entrada, la información viaja en un único sentido, desde el módulo que llama hacia el módulo llamado.

Nota: para realizar este apunte se utilizó (con el permiso correspondiente) el material del curso de Ingreso a la Facultad de Informática de la UNLP. En el mismo se han realizado algunos agregados y modificaciones.