

Figura 1.43. DFD de nivel 3, *Generar Pedido a Proveedor*.

El proceso de recibir pedido de proveedor lo dividimos en dos subprocesos: *Comprobar Albarán* (2.2.1) y *Actualizar stock* (2.2.2). El primero recibe los datos del albarán que se comprobarán con el pedido registrado en el almacén. Después de la comprobación se realizará el proceso de actualización del stock de los productos cuyas unidades se han recibido, Figura 1.44.

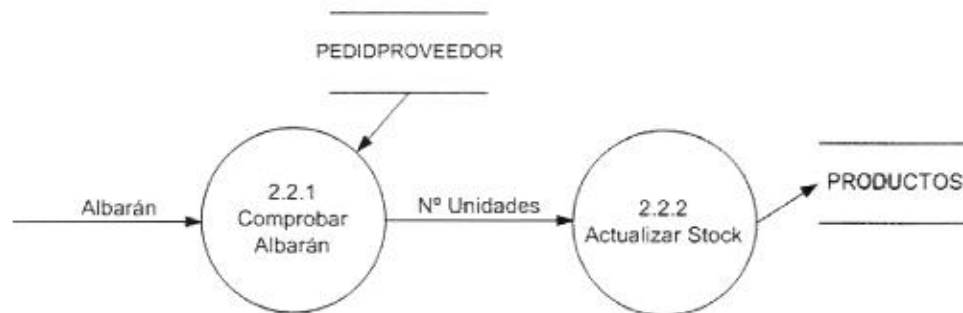



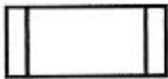



Figura 1.44. DFD de nivel 3, *Recibir Pedido de Proveedor*.

1.10.2. Pseudocódigo y diagramas de flujo

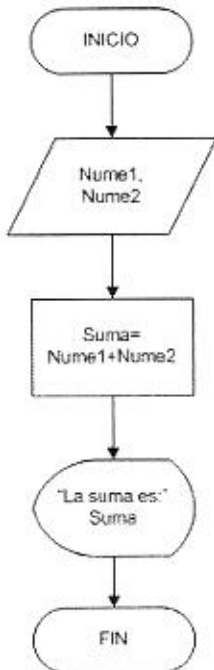
En el siguiente apartado se muestran algunos ejemplos de pseudocódigos y el correspondiente diagrama de flujo. Además de los símbolos vistos anteriormente utilizaremos los siguientes símbolos:

Símbolo <i>Terminador</i> , representa el inicio y final de un programa.	
Símbolo de <i>Entrada/Salida</i> , representa entrada y salida de datos.	
<i>Pantalla</i> , se utiliza para representar salida por pantalla.	

<i>Impresora</i> , se utiliza como símbolo de entrada (documento de entrada) y salida (impresora).	
<i>Teclado</i> , se utiliza como símbolo de entrada por teclado.	
<i>Conector</i> , se utiliza para unir una parte del diagrama con otra.	
Llamada a subrutina o procedimiento.	
Disco magnético, representa una función de entrada/salida para soporte en un disco magnético.	

Ejemplo 1. Programa que lee dos números y muestra la suma en pantalla.

Se utilizan estructuras secuenciales básicas.

Pseudocódigo	Diagrama de flujo
<p>Inicio</p> <p>Leer Num1, Num2</p> <p>Suma=Num1+Num2</p> <p>Visualizar "La suma es:" Suma</p> <p>Fin</p>	 <pre> graph TD INICIO([INICIO]) --> INPUT[/Num1, Num2/] INPUT --> PROCESS[Suma = Num1 + Num2] PROCESS --> OUTPUT([La suma es: Suma]) OUTPUT --> FIN([FIN]) </pre>

Ejemplo 2. Programa que lee dos números y muestra el mayor en pantalla, si son iguales deberá mostrar un mensaje indicándolo. Se utiliza la estructura condicional para comprobar los valores: **Si** <condición> **Entonces** <Instrucciones> **Si no** <Instrucciones> **Fin si**.

Además se muestra una estructura condicional dentro de otra.

Pseudocódigo	Diagrama de flujo
<p>Inicio</p> <p>Leer A, B</p> <p>Si $A > B$ Entonces</p> <p>Visualizar "El mayor es:" A</p> <p>Si no</p> <p>Si $A = B$ Entonces</p> <p>Visualizar "Son iguales"</p> <p>Si no</p> <p>Visualizar "El mayor es:" B</p> <p>Fin si</p> <p>Fin si</p> <p>Fin</p>	<pre> graph TD INICIO([INICIO]) --> Input[/A, B/] Input --> Cond1{A > B} Cond1 -- SI --> OutA([El mayor es: A]) Cond1 -- NO --> Cond2{A = B} Cond2 -- SI --> OutEq([Son iguales]) Cond2 -- NO --> OutB([El mayor es: B]) OutA --> Join(()) OutEq --> Join OutB --> Join Join --> FIN([FIN]) </pre>

Ejemplo 3. Programa que lee dos números en un proceso repetitivo. Este proceso terminará cuando los números leídos sean iguales.

Se utiliza la estructura repetitiva **Repetir** <instrucciones> **Hasta que** <condición>.

Pseudocódigo	Diagrama de flujo
<p>Inicio</p> <p>Repetir</p> <p>Visualizar "Escribe dos números"</p> <p>Leer A, B</p> <p>Hasta $A = B$</p> <p>Fin</p>	<pre> graph TD INICIO([INICIO]) --> Out([Escribe dos números]) Out --> Input[/A, B/] Input --> Cond{A = B} Cond -- SI --> FIN([FIN]) Cond -- NO --> Input </pre>

Ejemplo 4. Programa que lee 10 números en un proceso repetitivo y muestra la suma.

Es necesario declarar variables para contar los números que se van leyendo y para ir guardando la suma. Se deben inicializar a 0.

Se utiliza la estructura repetitiva **Repetir** <instrucciones> **Hasta que** <condición>.

Pseudocódigo	Diagrama de flujo
<p>Inicio</p> <p>Declarar Cuenta=0</p> <p>Declarar Suma =0</p> <p>Repetir</p> <p>Visualizar "Escribe un número"</p> <p>Leer A</p> <p>Cuenta = Cuenta +1</p> <p>Suma = Suma + A</p> <p>Hasta Cuenta = 10</p> <p>Visualizar "La suma es:" Suma</p> <p>Fin</p>	<pre> graph TD INICIO([INICIO]) --> Init[Cuenta = 0
Suma = 0] Init --> Input[/Escribe un número/] Input --> A[/A/] A --> Process[Cuenta = Cuenta + 1
Suma = Suma + A] Process --> Decision{Cuenta = 10} Decision -- NO --> Input Decision -- SI --> Output[/La suma es: Suma/] Output --> FIN([FIN]) </pre>

Ejemplo 5. Programa que lee registros de un fichero secuencial. Cada registro contiene información de un alumno: Nombre, Curso, Nota.

El programa debe mostrar por cada registro leído el Nombre, el Curso y la Nota.

Al final del proceso de lectura debe mostrar la nota media. Esta se calcula sumando las notas de todos los alumnos y dividiéndola por el número de alumnos que hay.

Se utiliza la estructura repetitiva **Mientras** <condición> **Hacer** <instrucciones> **Fin mientras**.

Se declaran tres variables, una para contar alumnos, otra para sumar notas y una tercera para guardar la nota media.

Pseudocódigo	Diagrama de flujo
<p>Inicio</p> <p>Declarar Cuenta=0</p> <p>Declarar Suma =0</p> <p>Declarar Media</p> <p>Abrir Fichero Notas</p> <p>Leer Registro (Nombre, Curso, Nota)</p> <p>Mientras NO sea Final de Fichero Hacer</p> <p>Visualizar Nombre, Curso, Nota</p> <p>Cuenta = Cuenta +1</p> <p>Suma = Suma + Nota</p> <p>Leer Registro (Nombre, Curso, Nota)</p> <p>Fin mientras</p> <p>Media= Suma/Cuenta</p> <p>Visualizar "Nota media:" Media</p> <p>Cerrar Fichero Notas</p> <p>Fin</p>	<pre> graph TD INICIO([INICIO]) --> Init[Cuenta = 0 Suma = 0] Init --> Abrir[Abrir fichero] Abrir --> Read[/Nombre, Curso, Nota/] Read --> Decision{¿NO es Fin de Fichero?} Decision -- Si --> LoopStart((Nombre, Curso, Nota)) LoopStart --> LoopBody[Cuenta=Cuenta + 1 Suma = Suma + Nota] LoopBody --> Read Decision -- NO --> CalcMedia[Media=Suma/ Cuenta] CalcMedia --> DisplayMedia[/"Nota media:" Media/] DisplayMedia --> Cerrar[Cerrar fichero] Cerrar --> FIN([FIN]) </pre>

Ejemplo 6. Programa que lee un número de teclado y muestra a qué día de la semana se corresponde.

Si el número leído es 1 visualizará Lunes, si es 2 visualizará Martes, si es 3 visualizará Miércoles, si es 4 Jueves, si es 5 Viernes, si es 6 Sábado y si es 7 Domingo.

Para cualquier otro valor visualizará *No válido*.

Se utilizará la estructura **Según sea**:

Según sea <variable> **Hacer**

Caso valor 1: Instrucciones

Caso valor 2: Instrucciones

Otro caso: Instrucciones

Fin según

