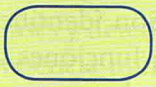
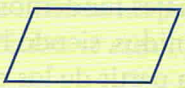
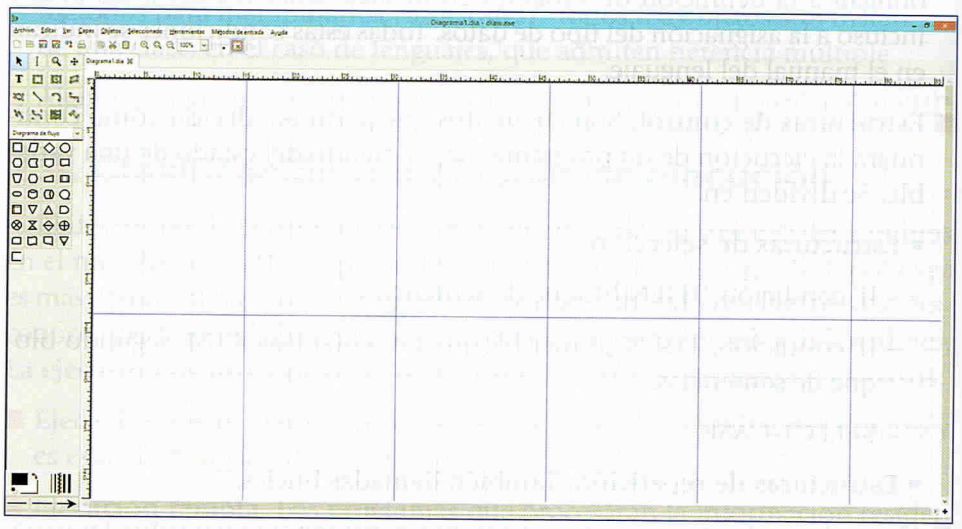
## 4 Diagramas de flujo

Un diagrama de flujo **es un método gráfico de representación de algoritmos.** Existen varios símbolos predefinidos para las distintas acciones que se pueden llevar a cabo, los más empleados son los siguientes:

| Nombre del símbolo | Forma | Imagen | Función |
|---|---|---|---|
| Terminal | Óvalo o elipse |  | Inicio o fin de diagrama |
| Línea de flujo | Línea continua acabada en punta de flecha |  | Indica la dirección que sigue el proceso |
| Proceso | Rectángulo |  | Operación o acción. Se emplea en asignación de valores y operaciones matemáticas |
| Entrada/salida de datos | Romboide |  | Forma general de entrada/salida de datos de un programa |
| Decisión | Rombo |  | Se hace un test sobre una variable y dependiendo del resultado el programa seguirá su curso por un camino u otro |



Dia permite la creación de diagramas de flujo.

Empleando este método el programa puede expresar que lee dos números del teclado, los suma y muestra el resultado en pantalla. Podemos dibujar el gráfico utilizando lápiz y papel, o bien, empleando una aplicación como **Dia**.

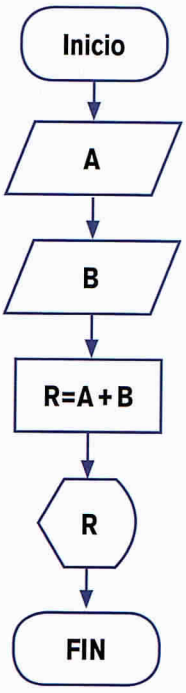
Este sencillo algoritmo produce un diagrama lineal, sin bucles ni toma de decisiones que puedan desviar el flujo de ejecución de instrucciones. Consta de los siguientes elementos:

- **Inicio y fin, representados con sendos** óvalos. Todo programa debe tener un comienzo y, al menos, una instrucción de finalización. En programas complejos es frecuente ver que se llega al fin desde diferentes lugares.

**TICO++**

**Dia**

En Linux y Windows hay varios programas capaces de facilitar la creación de diagramas de flujo. Uno de los más usados es Dia, aunque también se pueden emplear las herramientas de dibujo incluidas en las suites ofimáticas como LibreOffice.



- **Flechas** que indican la dirección que debe seguir el algoritmo en cada momento.
- **Dos instrucciones** consecutivas **de lectura de datos** que serán almacenados en las variables **A y B.** El símbolo de lectura es un **romboide**.
- Una operación de suma que es guardada en una tercera variable R. Esta variable es innecesaria, se ha introducido para ilustrar el empleo de **operaciones de cálculo y asignación** de valores que requieren el uso del símbolo **rectángulo**.
- La **exhibición del resultado** se hace empleando un símbolo que recuerda el **perfil de los antiguos monitores CRT** que precedieron a las pantallas planas. También se podría emplear el **romboide** como símbolo de salida de datos.

**TEN EN CUENTA**

**Pseudocódigo**

Existe una alternativa escrita al uso de los diagramas de flujo: el pseudocódigo. Se trata de un falso lenguaje de programación para describir algoritmos que mezcla lenguaje natural, matemático y una sintaxis de programación similar a la empleada en lenguajes populares como Pascal o C.

**Seguimiento de valores**

Para comprobar si el diagrama de flujo que hemos diseñado resuelve un problema, lo mejor es anotar a un lado las variables y hacer el seguimiento del flujo del programa. Cada vez que una variable cambia de valor tachamos el antiguo y anotamos el nuevo; de ese modo se pueden detectar fallos, especialmente en las estructuras de control.

### Actividades

**18.** Diseña diagramas de flujo que resuelvan los siguientes problemas:

a) Calcular el área de un cuadrado leído el valor del lado por teclado.

b) Calcular el área de un círculo leído el valor del radio por teclado.

c) Calcular el módulo de un vector bidimensional del que proporcionamos por teclado las coordenadas de inicio y fin.

d) Calcular el módulo de un vector tridimensional del que proporcionamos por teclado las coordenadas de inicio y fin.

e) Calcular el ángulo de un vector bidimensional del que proporcionamos por teclado las coordenadas.

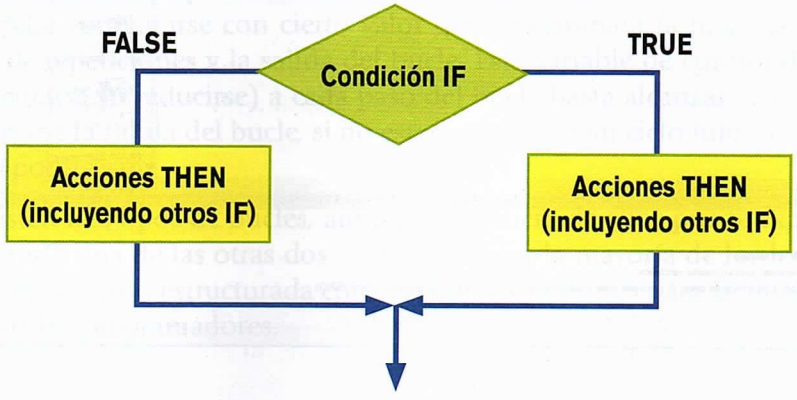
Ten en cuenta que puedes usar la notación matemática convencional dentro de los símbolos de los diagramas de flujo.

**19.** Busca más información sobre qué es un pseudocódigo y compáralo con los diagramas de flujo. ¿Existen puntos en común o equivalencias entre ambos métodos de expresar algoritmos?

## 4.1. Estructuras de control: selección

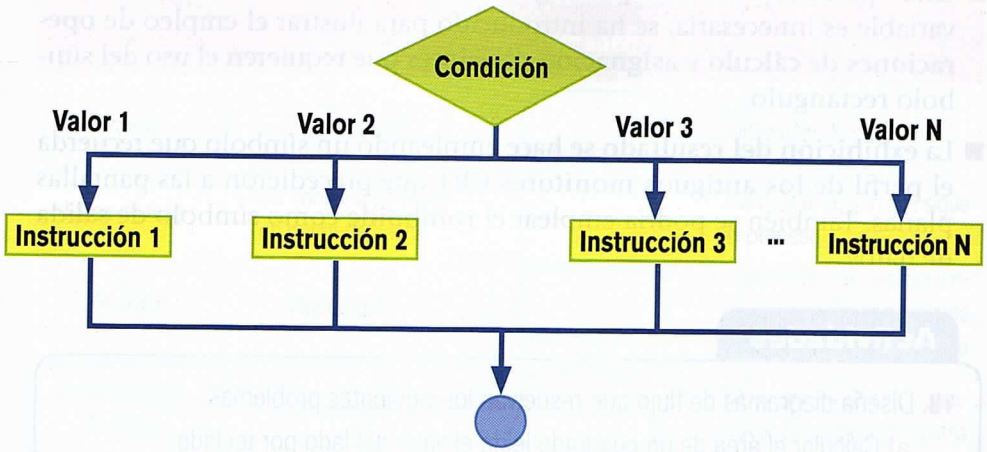
En ocasiones es necesario alterar el flujo del programa y realizar diferentes acciones dependiendo de alguna condición o del cálculo de un resultado. En estos casos se emplea una estructura de control de selección.

**Gráfico 2. Estructura de una situación verdadera o falsa**



Si solo hay dos alternativas posibles, definidas por el cumplimiento o no de una situación (valor ***booleano* verdadero o falso**) se emplea una estructura «**si entonces si no**» (*if then else*, en inglés), también conocida como **selección simple**.

**Gráfico 3. Estructura de selección múltiple**



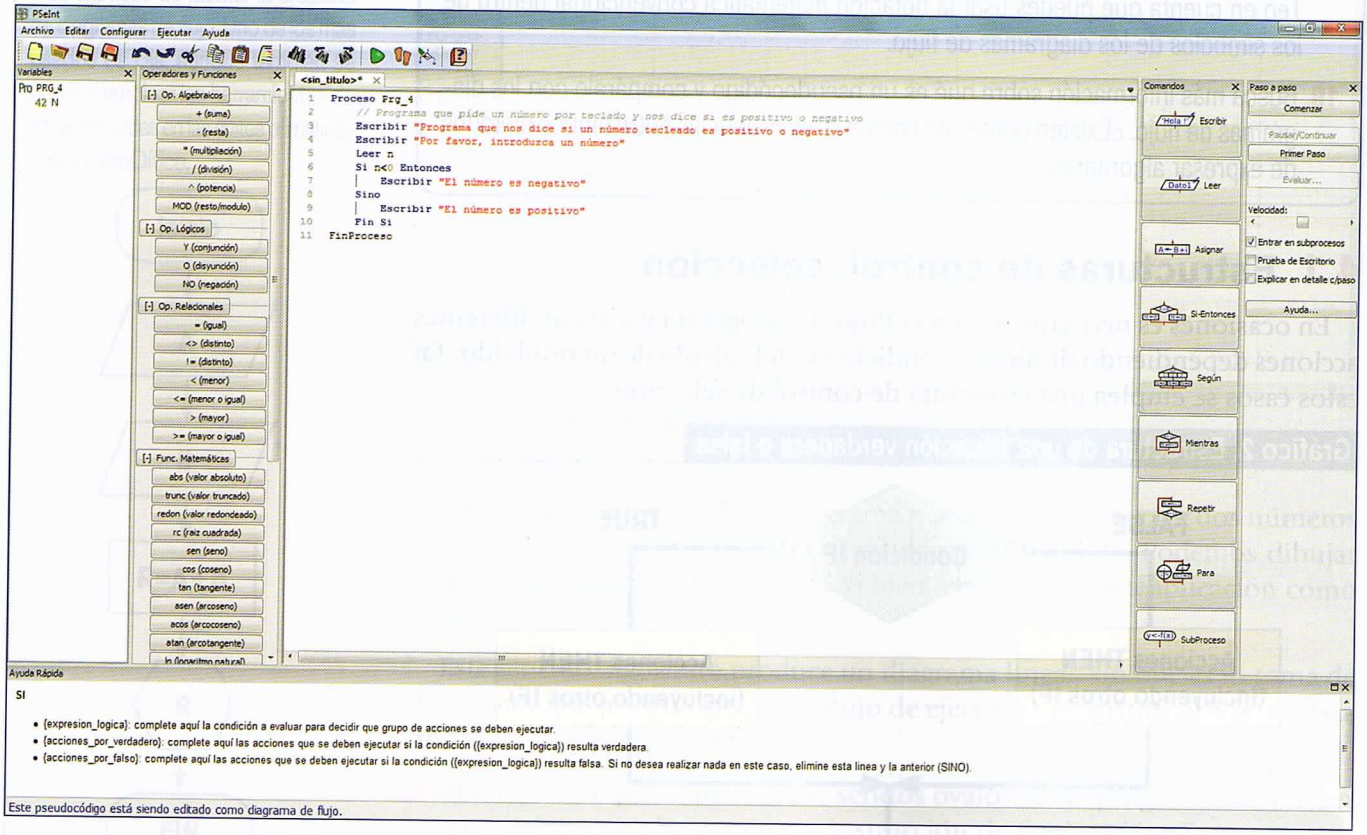

**TICO++**
**PSeInt**

Es posible encontrar una herramienta libre totalmente en castellano para aprender los fundamentos de programación empleando pseudocódigo y diagramas de flujo. Está disponible para distintos sistemas operativos, entre ellos Linux y Windows.

Esta aplicación emplea sus propios símbolos para la entrada por teclado y la salida por pantalla, siendo la entrada un romboide con una flecha en su esquina superior derecha apuntando hacia el interior y la salida una flecha apuntando hacia fuera.

Cuando los valores posibles que puede tomar la variable de control de la estructura son más de dos, por ejemplo, varios valores numéricos enteros, nos encontramos ante una estructura de **selección múltiple** denominada «**según hacer**» (*select case*, en inglés).

Como se observa en los diagramas de ambas estructuras de selección, aunque modifican el flujo de programa, **las estructuras en sí mismas tienen un punto de entrada y un punto de salida en el que confluyen las distintas alternativas.**




PSeInt es un intérprete de pseudocódigo en castellano.

## Actividades

**20.** Diseña diagramas de flujo que resuelvan los siguientes problemas. Para todos ellos, lleva a cabo el seguimiento de valores en tu cuaderno.

a) Leer un número **n** por teclado y escribir un mensaje indicando si es positivo o negativo. La condición de selección debe ser **n < 0.**

a) Modifica el diagrama de flujo anterior para que la condición de selección sea **n > 0.**

b) Pedir al usuario que teclee su edad, la compare con la tuya y escriba en pantalla si es mayor o menor.

a) Hacer una modificación del programa que identifica números positivos y negativos para que nos indique si el número tecleado es **0.**

b) Leer tres números por teclado e imprimir en pantalla el mayor.

**21.** Diseña diagramas de flujo que resuelvan los siguientes problemas:

a) Leer un número natural **n** por teclado, correspondiente a la calificación obtenida en un examen entre **0** y **10.** Debe escribir los siguientes mensajes: (**0 ≤ n ≤ 4,** «Insuficiente»), (**n = 5,** «Suficiente»), (**n = 6,** «Bien»), (**7 ≤ n ≤ 8,** «Notable»), (en cualquier otro caso, «Sobresaliente»).

a) En el anterior problema, ¿cómo contemplarías la posibilidad de que se tecleara un número negativo o superior a **10**?

a) Leer un número natural **n** por teclado comprendido entre **1** y **12,** y que muestre en pantalla los meses del año. Por ejemplo, si tecleo **1** debe escribir «enero». Asegúrate que los números tecleados están entre **1** y **12** para que no se produzcan errores.

b) Leer un carácter de teclado e indicar si se trata de una vocal o una consonante.

c) Leer un carácter de teclado e indicar si es letra, número o signo de puntuación.

## 4.2. Estructuras de control: repetición

Con frecuencia nos encontraremos ante la necesidad de repetir, varias veces, ciertos pasos de un algoritmo para resolver un problema. Por ejemplo, si queremos imprimir la tabla de multiplicar de cierto número, tendremos que calcular el producto de ese número por 1, a continuación por 2 y así sucesivamente. En este ejemplo se ve claramente que hay un elemento que cambia de valor a cada paso del algoritmo, el número por el que multiplicamos, pero el resto no cambia. Este tipo de problemas se resuelven empleando **estructuras repetitivas** llamadas **bucles**.

**Los bucles,** al igual que las estructuras de selección, **solo tienen un punto de entrada y un punto de salida.** Entre ambos extremos se sitúan las instrucciones que deben repetirse.

Las estructuras repetitivas cuentan con una variable de control del bucle que debe compararse con cierto valor que determinará la finalización del ciclo de repeticiones y la salida del bucle. Esta variable de control debe incrementarse (o reducirse) a cada paso del bucle hasta alcanzar el valor que determine la salida del bucle, si no estaríamos ante un ciclo infinito del que no se podría salir.

Existen tres tipos de bucles, aunque cualquiera de ellos puede expresarse utilizando una de las otras dos formas. Si bien, la mayoría de los lenguajes de programación estructurada contemplan los tres tipos para facilitar el trabajo de los programadores.

- Bucle «**repetir hasta**» ***(do until)***. Repite el bloque de instrucciones en su cuerpo **hasta que sea verdadera la condición de salida.** Este tipo de bucle ejecuta las instrucciones en su interior al menos una vez, pues la condición de salida se comprueba al final del bucle.

**Ejemplo:**

**Tabla de multiplicar con bucle «hasta»**

Usando PSeInt el código quedaría así:

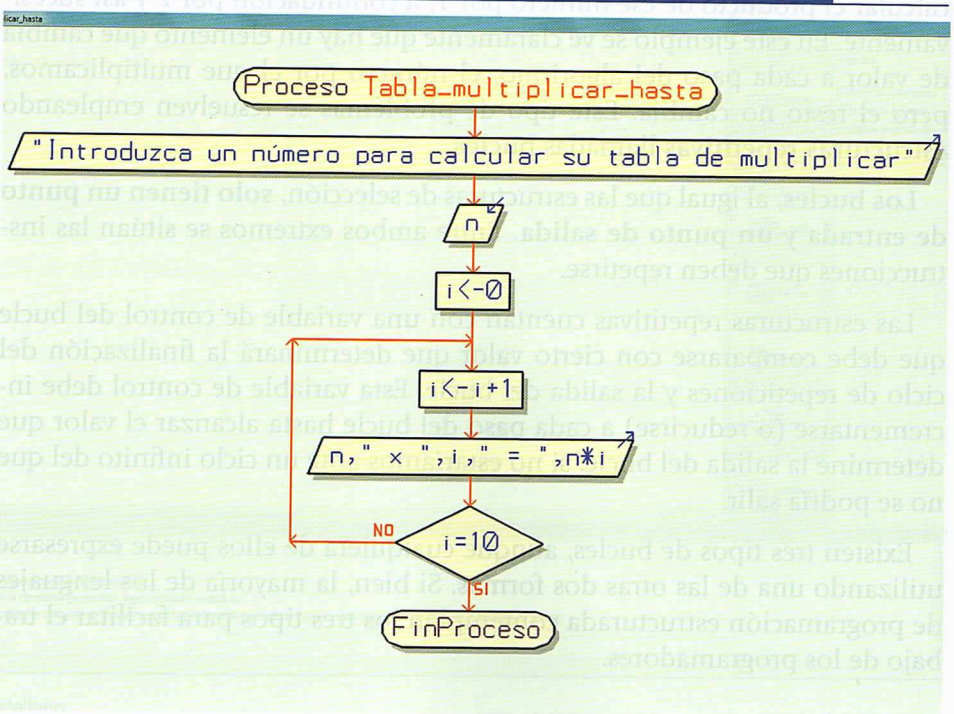
```
1  Proceso Tabla_multiplicar_hasta
2      Escribir "Introduzca un número para calcular su tabla de multiplicar"
3      Leer n
4      i<-0 // i controlará el bucle. Se inicializa su valor a 0
5      Repetir
6          i<-i+1 //Se incrementa la variable de control del bucle
7          Escribir n," x ",i," = ",n*i
8      Hasta Que i=10 //Condición de salida del bucle
9  FinProceso
```

Los números que aparecen a la izquierda del código son números de línea, y son ignorados por el intérprete, igual que los comentarios que empiezan por los símbolos //.

Observamos que la variable de control hay que declararla antes de entrar en el bucle y es muy importante inicializar su valor. Se supone que el intérprete le da el valor 0 al utilizarla por primera vez, pero es preferible asegurarse para evitar errores inesperados.

El diagrama de flujo, en el que se puede observar que la variable de control del bucle se comprueba a la salida del mismo y se modifica en su interior, sería:

**Gráfico 4. Diagrama de flujo de la estructura de control "hacer hasta"**




**Actividad**

**22.** Responde a las siguientes preguntas sobre modificaciones al ejemplo para calcular las tablas de multiplicar con un bucle **«hasta».** Qué pasaría si:

- **a)** El valor de inicialización de **i** fuera **1**.
- **b)** Si se cambia el orden de las dos instrucciones dentro del bucle.
- **c)** Si el incremento de **i** fuera de paso **2** en lugar de paso **1.**
- **d)** Si el incremento de **i** fuera de paso **3** en lugar de paso **1.**

## Actividad

**23.** Los operadores relacionales que podemos usar para controlar un bucle son seis:

| | | | |
|---|---|---|---|
| Menor que | < | Menor o igual que | <= |
| Mayor que | > | Mayor o igual que | >= |
| Igual a | = | Distinto de | != o <> |

En uno de los dos últimos casos del ejercicio anterior el programa no acabaría. ¿Qué operador usarías para que el programa acabe?

- Bucle «**hacer mientras**» ***(do while)***. La condición de entrada en el bucle se comprueba al principio, por lo que, potencialmente, este bucle se ejecuta cero o más veces. Se repite el bloque de instrucciones en su cuerpo **mientras sea verdadera la condición de acceso al bucle.**

**Ejemplo:**

**Tabla de multiplicar con bucle MIENTRAS**

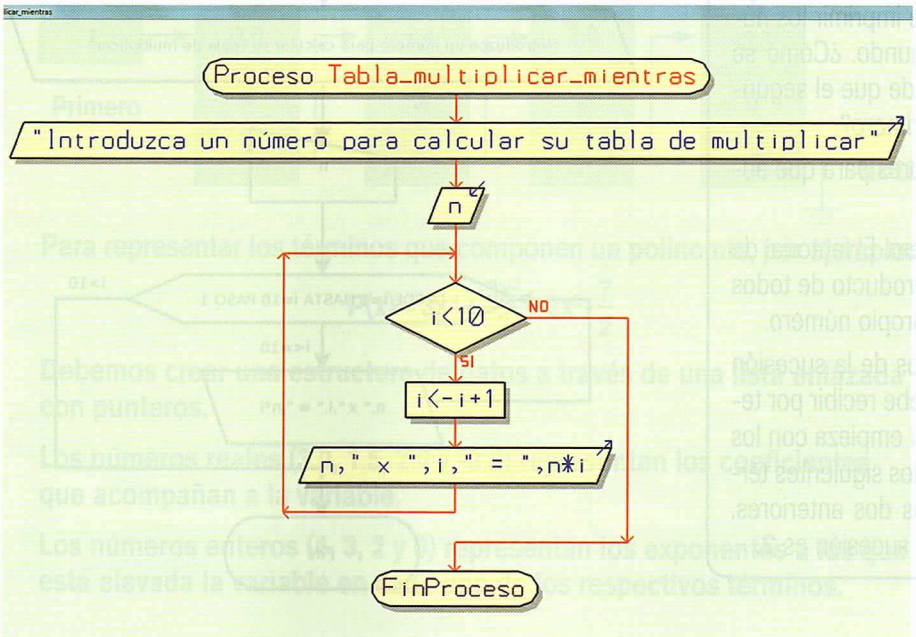
Usando PSeInt el código quedaría así:

```
1  Proceso Tabla_multiplicar_mientras
2      Escribir "Introduzca un número para calcular su tabla de multiplicar"
3      Leer n
4      Mientras i<10 Hacer // Si no se modificó antes, el valor inicial de i es 0
5          i<-i+1  //Se incrementa la variable de control del bucle
6          Escribir n," x ",i," = ",n*i
7      Fin Mientras
8  FinProceso
```

En este caso se declara la variable **i** en la cabecera del bucle y se presume que el intérprete la inicializa automáticamente a **0**, por eso la primera instrucción del cuerpo del bucle consiste en incrementar el valor de **i**.

El diagrama de flujo en el que se puede observar que la variable de control del bucle se comprueba a la entrada del mismo y se modifica en su interior.

**Gráfico 5. Diagrama de flujo de la estructura de control "hacer mientras"**



- Bucle «**para**» *(for)*. Repite el bloque de instrucciones en su interior un número determinado de veces. La sintaxis de este tipo de bucle es un poco más compleja que los dos anteriores. Se asigna a la variable de control un valor inicial, se establece el valor máximo y, opcionalmente, se indica el paso que tendrá el incremento de la variable de control. En algunos lenguajes, si la variable de control es un número entero y el incremento es positivo de una unidad, no es necesario indicarlo. En cambio, si el incremento es negativo, de paso distinto a la unidad, o un número real, sí es necesario indicarlo.

**Ejemplo:**

**Tabla de multiplicar con bucle PARA**

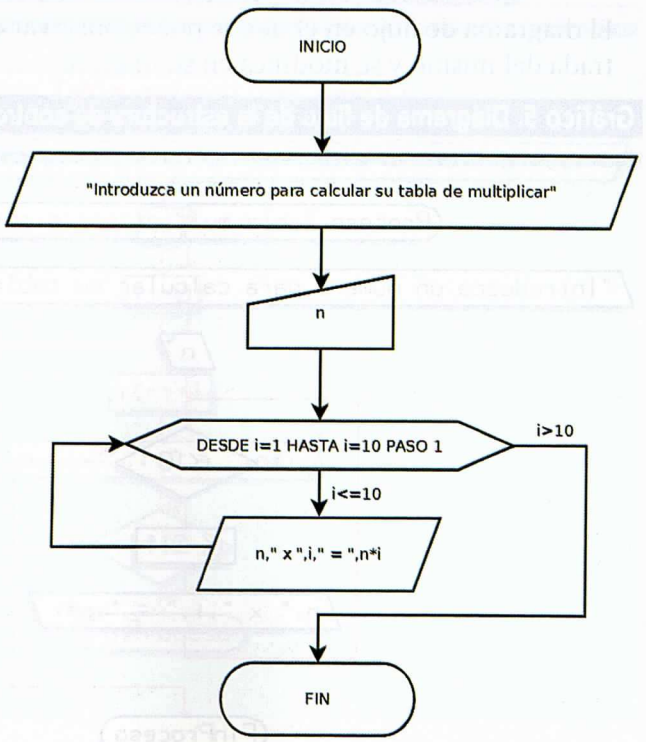
Usando PSeInt el código quedaría así:

```
1  Proceso Tabla_multiplicar_para
2      Escribir "Introduzca un número para calcular su tabla de multiplicar"
3      Leer n
4      Para i<-1 Hasta 10 Con Paso 1 Hacer
5          Escribir n," x ",i," = ",n*i
6      Fin Para
7  FinProceso
```

Como se puede ver, en este bucle se indican los valores de inicio y fin de la variable de control y el paso del incremento.

En el bucle «**para**», PSeInt también se aparta del estándar, por lo que se muestra un diagrama hecho con Dia. Como se observa, el gráfico es muy similar al del bucle «**mientras**»:

**Gráfico 6. Diagrama de flujo de la estructura de control "para"**




## Actividad

**24.** Diseña diagramas de flujo que resuelvan los siguientes problemas empleando los tres tipos de bucle para cada uno:

- **a)** Imprimir una lista de números desde **1** hasta **10.**
- **b)** Leer un número por teclado e imprimir una lista desde **1** hasta ese número.
- **c)** Leer dos números por teclado e imprimir los números desde el primero al segundo. ¿Cómo se debería modificar para el caso de que el segundo número sea menor que el primero?
- **d)** Modificar los algoritmos anteriores para que admitan números negativos.
- **e)** Calcular el factorial de un número. El factorial de un número se define como el producto de todos los números desde **1** hasta el propio número.
- **f)** Calcular los **n** primeros términos de la sucesión de Fibonacci. El valor de **n** se debe recibir por teclado. La sucesión de Fibonacci empieza con los números **1, 1** y a partir de aquí los siguientes términos se calculan sumando los dos anteriores. Es decir, el tercer término de la sucesión es **2.**