9. UNIDADES

Conceptos: *Unidad, Librería, Unidad estándar, Interfaz, Implementación, Inicialización*

Resumen: Una de las características más interesantes de las últimas versiones de TurboPascal es la posibilidad de descomponer un programa grande en módulos más pequeños que se pueden compilar de forma independiente. A estos módulos se les denomina unidades en TurboPascal y se definen como un conjunto de declaraciones de constantes, tipos de dato, variables, procedimientos y funciones. El concepto de unidad acentúa el carácter estructurado y modular de TurboPascal para el diseño de grandes programas. Las unidades tienen una estructura similar a la de un programa y consta de cuatro secciones: la cabecera, la sección de interfaz, la sección de implementación y la sección de inicialización. A continuación se detalla las características específicas de los procesos de creación, compilación y utilización de unidades.

Objetivos específicos. Al finalizar el tema, el alumno deberá ser capaz de:

a) Definir el concepto de unidad y describir la estructura de su código fuente y el mecanismo de creación y funcionamiento (*Conocimiento*)
b) Interpretar el código fuente de una unidad (*Comprensión*)
c) Codificar una unidad que incluya la declaración de elementos de distintos tipos (*Aplicación*)

9.1. INTRODUCCIÓN

El concepto de *unidad* en TurboPascal es equivalente al concepto de *librería* en otros lenguajes de programación como Fortran o C.

Una UNIDAD es, básicamente, una **colección de declaraciones** de constantes, tipos de dato, variables, funciones y procedimientos. Su estructura es bastante rígida y parecida a la de un programa. Al igual que un programa fuente, una unidad fuente se puede compilar y traducirse a código máquina, generándose un archivo que contiene la unidad compilada. Pero, a diferencia del programa compilado, una unidad compilada sigue siendo sólo un conjunto de declaraciones, un archivo que no es posible ejecutar directamente (Figura 48).

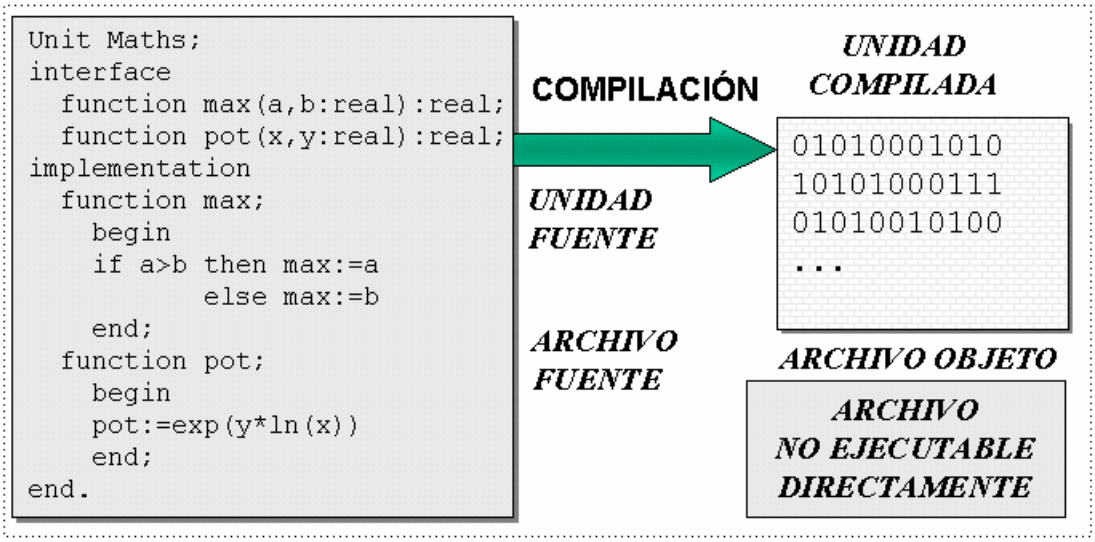

Figura 48. Compilación de una unidad de TurboPascal

9.2. EMPLEO DE LAS UNIDADES

Si una unidad no puede ejecutarse como un programa, entonces, ¿cuál es la utilidad de las unidades? Una unidad permite generar *librerías* o conjuntos de declaraciones que pueden utilizarse en otros programas sin necesidad de volver a incluir en éstos las declaraciones correspondientes (Figura 49). Esto permite, por un lado, que lo declarado en una unidad pueda ser utilizado en uno o más programas (Figura 50) y, por otro, que todo lo utilizado en un programa pueda dividirse en módulos (un conjunto de unidades y el programa en sí) y compilarse independientemente en distintos archivos fuente (Figura 51). En cualquiera de los casos, el archivo resultante al compilar el programa sí es ejecutable y, además, de forma independiente (incluye todo el código necesario para su ejecución).

Una vez escrita una unidad (`unidad.pas`), ésta se compila creándose un nuevo archivo de código objeto (`unidad.tpu`). Cualquier programa puede ahora utilizar lo incluido en dicha unidad. Por otro lado, una unidad puede a su vez utilizar lo declarado en otras unidades,...

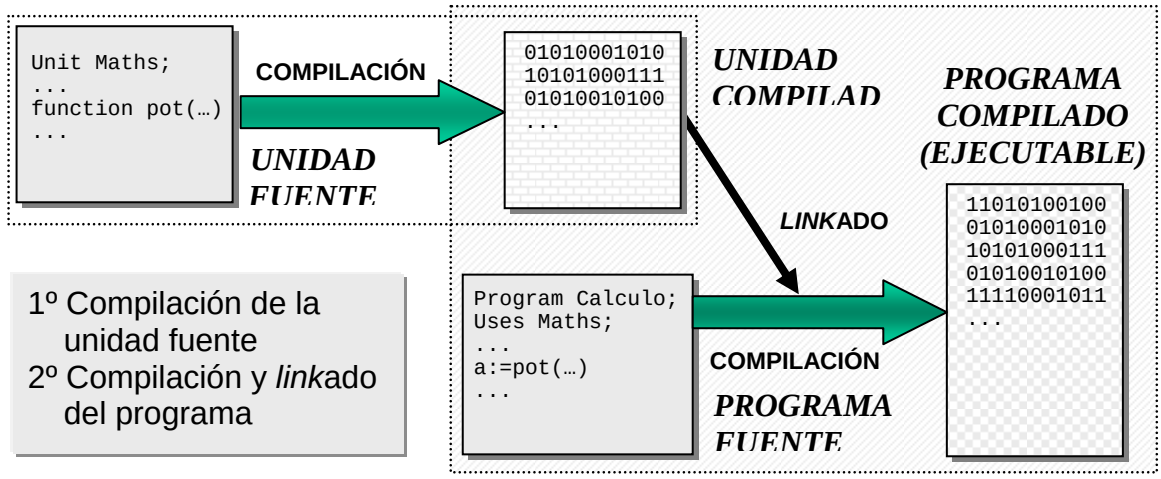

Figura 49. Esquema gráfico de la utilización de una unidad en un programa de Pascal

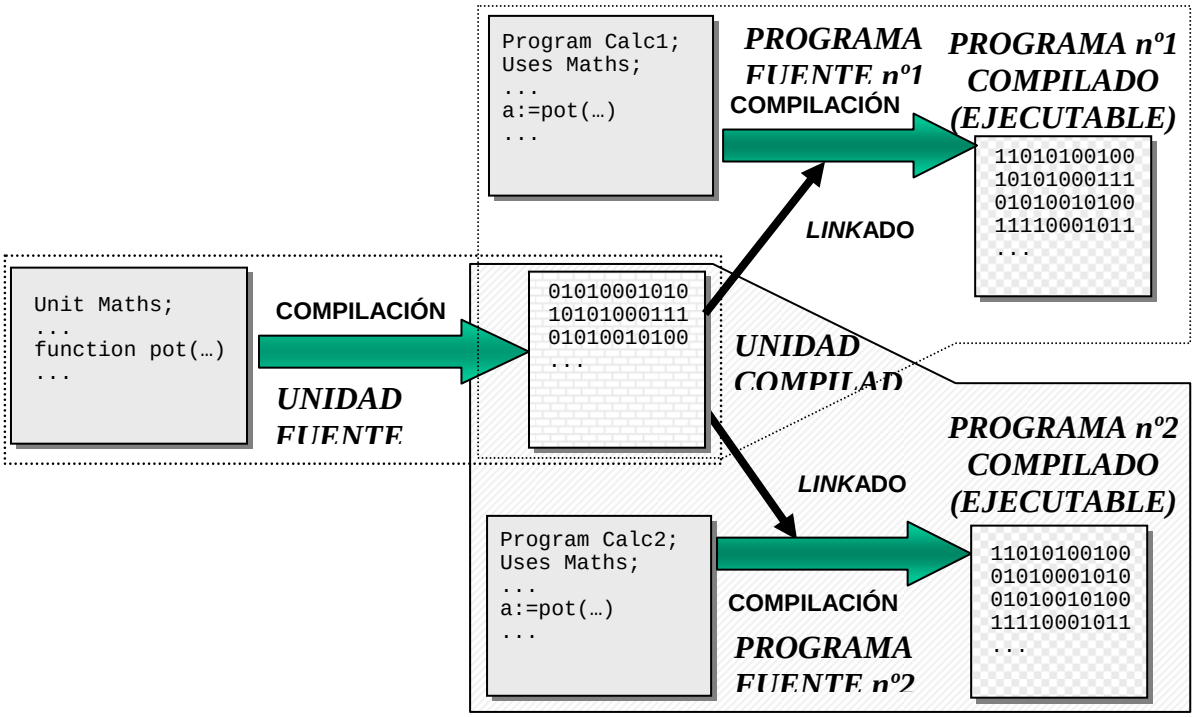

Figura 50. Esquema gráfico de la utilización de una unidad en dos programas de TurboPascal

9.3. VENTAJAS DEL EMPLEO DE UNIDADES

Las ventajas de utilizar unidades en programación son las siguientes:

1. Posibilita la creación de librerías de declaraciones personalizadas de cada usuario.
2. Facilita el diseño de grandes programas. Se sigue el lema "divide y vencerás". El programador incluye en su programa fuente la declaración de las unidades que contienen lo que va a utilizar. Cuando se compila el programa fuente, que en TurboPascal 7.0 trabajando en modo real puede tener un tamaño máximo de 64Kb, se incluye en el nuevo archivo objeto creado todo el código necesario para ser ejecutado de forma independiente.

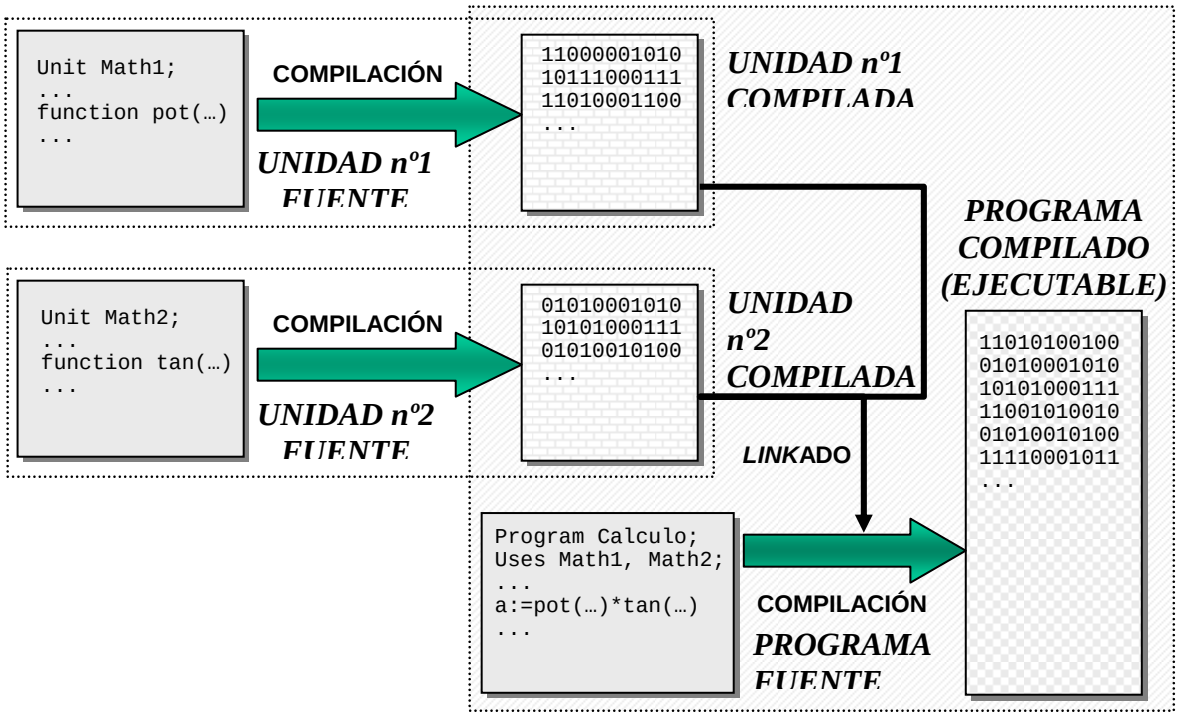

Figura 51. Esquema gráfico de la utilización de dos unidades en un programa de TurboPascal

3. La existencia de unidades *estándar* de Turbo Pascal: `System`, `Crt`,... que incluyen una gran cantidad de declaraciones de constantes, tipos de dato, variables, funciones y procedimientos muy útiles que el programador puede aprovechar para la construcción de sus programas.

9.4. UNIDADES ESTÁNDAR DE TURBOPASCAL 7.0

El archivo de disco `turbo.tpl` (`tpl` ≡ **T**urbo **P**ascal **L**ibrary o librería de unidades de Turbo Pascal) contiene, en principio, las unidades estándar `System`, `Overlay`, `Crt`, `Dos` y `Printer`. Las demás unidades estándar: `Graph`, `Graph3`, `Turbo3`,... se encuentran almacenadas como archivos independientes con extensión `.TPU` (**T**urbo **P**ascal **U**nit) para la versión del entorno de Turbo Pascal que trabaja en modo real y `.TPP` para la versión del entorno de BorlandPascal que trabaja en modo protegido. El contenido principal de cada una de estas unidades es el siguiente:

`System`: Contiene las funciones y los procedimientos estándar más usuales. Como se explica en la sección de uso y manejo de unidades, esta unidad tiene como característica particular el ser la única que no necesita ser declarada en un programa para utilizar en éste lo declarado dentro de ella.

`Crt`: Contiene todas las funciones y procedimientos para conseguir un control de las operaciones de entrada/salida por teclado y pantalla.

`Dos`: Contiene funciones y procedimientos para la manipulación de archivos de disco y del sistema operativo DOS.

`Overlay`: Incluye soporte para el manejo de *overlays* (recubrimientos de memoria).

`Printer`: Declara la variable archivo `Lst` de tipo `text`, que facilita la salida de datos por la impresora.

`Graph`: Contiene subrutinas para manipular la pantalla gráfica.

Graph3: Permite obtener la compatibilidad con los procedimientos gráficos de la versión 3.0 de TurboPascal.

Turbo3: Permite obtener la compatibilidad con los procedimientos (no gráficos) y funciones de la versión 3.0.

9.5. ESTRUCTURA DE UNA UNIDAD

La estructura de una unidad sigue el esquema que se da a continuación:

Cabecera	`Unit n_unidad;`

Interfaz	`Interface`
	`uses ... { unidades que emplea }`
	`{ declaraciones públicas o visibles de objetos exportables a otros programas }`
	`const ...`
	`type ...`
	`var ...`
	`procedure ... { sólo la cabecera }`
	`function ... { sólo la cabecera }`

Implementación	`Implementation`
	`{ declaraciones privadas:locales a la unidad }`
	`uses ...`
	`const ...`
	`type ...`
	`var ...`
	`procedure ... { cabecera y cuerpo }`
	`function ... { cabecera y cuerpo }`

Inicialización	`[begin` `... { código de inicialización } ]`
Final de la unidad	`end.`

Las características principales de cada una de las partes de una unidad son las que se comentan a continuación.

9.5.1. Cabecera

La cabecera de la unidad es obligatoria. El identificador asociado a la unidad, por ejemplo `n_unidad`, debe coincidir con el nombre del archivo que contiene el código fuente: `n_unidad.pas`, y también con el nombre del archivo que contiene el código ya compilado: `n_unidad.tpu`. Este nombre o identificador es el que luego se emplea en la declaración de uso de la unidad dentro de la instrucción `uses`. La única excepción a esta última circunstancia la dan los archivos de unidades incluidos, a su vez, en la librería de unidades de Turbo Pascal `turbo.tpl`).

9.5.2. Interfaz

La sección de interfaz *abre* la unidad a otros programas y unidades. Lo que se declare en esta sección es lo "exportable" a (utilizable en) otras unidades y programas. Esta parte comienza con la palabra reservada `interface` e incluye una lista de declaraciones de unidades, constantes, tipos de dato, variables y cabeceras de funciones y procedimientos y finaliza con la

palabra reservada `implementation`. Cuando un programa u otra unidad declara la utilización de una unidad (`uses nombre_unidad;`) todas las declaraciones que ésta contenga en la parte de interfaz están disponibles como si se hubieran definido dentro del programa mismo. Es la parte pública o visible de la unidad. En esta sección no están permitidas las declaraciones `forward`.

9.5.3. Implementación

La sección de implementación comienza con la palabra reservada `implementation` y contiene el cuerpo de las funciones y procedimientos declarados en la sección anterior (lo que es obligatorio) y además la parte "no exportable" de la unidad (utilizable sólo y exclusivamente dentro de la unidad). A la sección de implementación también se le denomina parte privada de la unidad.

Los cuerpos de las rutinas declaradas en la sección de interfaz deben incluirse en la sección de implementación pero la condición inversa no es necesaria, es decir, no todas las rutinas incluidas en la implementación tienen por qué incluirse en la sección de interfaz.

9.5.4. Inicialización

La sección de inicialización pertenece a la parte privada o no visible de la unidad, empieza con la palabra reservada `begin` y permite ejecutar sentencias antes de la ejecución de la primera instrucción del cuerpo del programa que utilice dicha unidad. No es obligatoria (en la estructura mostrada anteriormente aparece entre corchetes). No todas las unidades incluyen esta sección, que suele usarse para dar valores a cualquier estructura de datos (*inicializar* variables) declarada en la unidad ...

9.6. CREACIÓN Y USO DE LAS UNIDADES

A continuación se muestra un ejemplo de una unidad escrita en TurboPascal:

```
unit matemat;
interface
             uses crt;
             const e = 2.71828182;
             type cadena = string[10];
             var clave : cadena;
             function factorial(n:integer):real;
             function exponencial(a,b:real):real;
             function arcsen(x:real):real;
             function arcos(x:real):real;
             function mayor(a,b:integer):integer;
             procedure pausa;
implementation
       const mensaje = 'Pulsa INTRO para continuar';
       function factorial;
           begin
           if n=0 then factorial:=1
                  else factorial:=n*factorial(n-1)
           end;
       function exponencial;
           begin
           exponencial:=exp(a*ln(b))
           end;
       function arcsen;
           begin
           arcsen:=arctan(x/sqrt(1-x*x))
           end;
       function arcos;
           begin
           arcos:=arctan(sqrt(1-x*x)/x)
```

```
        end;
    procedure cambio(var x,y:integer);
        var aux:integer;
        begin
        aux:=x;
         x:=y;
         y:=aux
        end;
    function mayor;
        begin
        if a<b then cambio(a,b);
        mayor:=a
        end;
    procedure pausa;
        begin
        clrscr;
        write(mensaje);
        readln;
        clrscr
        end;
begin
clave := 'inicial'
end.
```

Una vez escrito el código fuente de la unidad y almacenado en un archivo con extensión `.pas`, se compila igual que un programa pero dando origen a un archivo de extensión `.tpu` **no ejecutable directamente**, según muestra la figura 9.5.

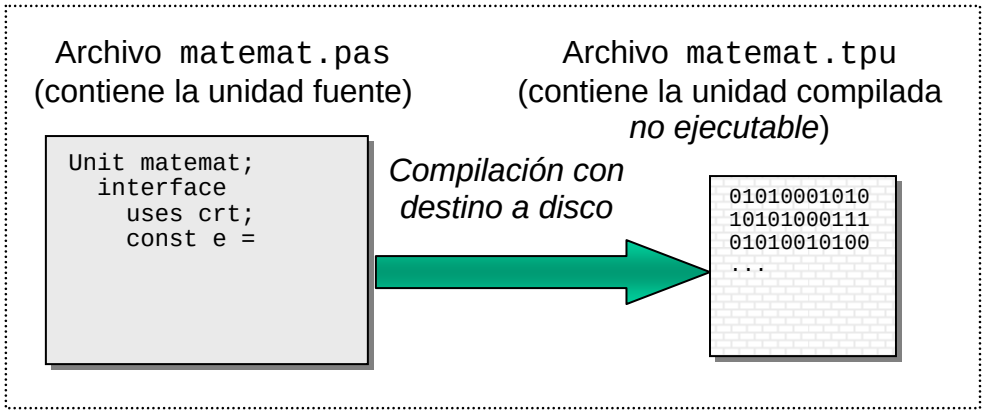

Figura 52. Esquema de la compilación de una unidad en Turbo Pascal

Para utilizar lo declarado dentro de una unidad en un programa es necesaria la existencia del archivo que contiene la unidad ya compilada con extensión `.tpu` en el disco si se trabaja en la versión del entorno de Turbo Pascal que funciona en modo real, `.tpp` para la versión del entorno de Borland Pascal que funciona en modo protegido y , `.tpw` para la versión del entorno de Windows.

En el primer caso es necesario seleccionar explícitamente como destino del código generado, al realizar la compilación de la unidad, la opción *Disco* (*Disk*) en lugar de la opción *Memoria* (*Memory*). En el segundo caso, no hay opción y el resultado de la compilación será siempre un archivo de disco.

Una vez creado este archivo ya puede emplearse en cualquier programa escrito en Turbo Pascal; sólo hay que incluir la correspondiente declaración de utilización de unidades justo después de la cabecera del programa:

Ej.:
```
program ejemplo;
uses matemat;
{ resto del codigo del programa }
```

Si van a utilizarse varias unidades en un mismo programa bastará con utilizar una única sentencia `uses` seguida de los identificadores de cada una de las unidades separadas por comas.

Ej.:
```
program ejemplo;
uses unidad_A, unidad_B;
{ resto de declaraciones del programa }
begin
...
end.
```

Una vez se haya construido el archivo `ejemplo.pas` que contiene el programa en código fuente, se podrá compilar como de costumbre utilizando la opción *Compile* del entorno de TurboPascal obteniéndose un único archivo `ejemplo.exe` que no necesita de ningún otro archivo para poder ser ejecutado directamente: en este archivo se tiene **todo** el código necesario para su ejecución, con la excepción de la unidad `graph` (no se incluyen los archivos `.bgi` y `.chr`) . Un ejemplo de un programa que utiliza la unidad anterior sería el siguiente:

```
program calculo;
uses crt, matemat;
var a : integer;
begin
clrscr;
write('Introduce un numero: ');
readln(a);
writeln('El factorial es:',factorial(a):5:0)
end.
```

Para localizar la unidad, el compilador primero la busca en la librería de unidades (`turbo.tpl, tpp.tpl` ó `tpw.tpl`). Si no la encuentra la busca en disco. Primero en el directorio actual y luego en los directorios especificados como Directorios de Unidades en (*Options/Directories*). El compilador supone que el nombre del archivo coincide con el de la unidad mas la extensión `.tpu, .tpp` ó `.tpw`, dependiendo de la plataforma.

9.7. IDENTIFICADORES REPETIDOS EN UNIDADES Y PROGRAMAS

Si en un programa que utiliza una unidad se declara un identificador ya empleado en la unidad, no se producirá, por este motivo, un error al compilar el programa. Sencillamente, cualquier referencia en el programa a dicho identificador se entenderá al declarado en el programa.

Si se desea referenciar dentro del programa al elemento de la unidad que comparte el mismo identificador, éste tendrá que venir precedido del nombre de la unidad seguido de un punto. Por ejemplo, si se ha declarado **`e`** como una **variable** de tipo `integer` en un programa que emplea la unidad `matemat`, para utilizar la **constante** `e` declarada en la unidad se hará referencia al identificador *cualificado* **`matemat.e`**

9.8. UTILIZACIÓN DE UNIDADES CON REFERENCIA CIRCULAR

En un programa puede declararse el uso de dos unidades A y B con *referencia circular*: la unidad A declara el uso de la unidad B y viceversa, la unidad B declara el uso de la unidad A. Para que esto sea posible, se debe incluir al menos una de las respectivas declaraciones de utilización de unidad (`uses`...) en las sección de implementación correspondiente. Podrá emplearse la opción *Compile* para compilar dicho programa. Un ejemplo de utilización de unidades con referencia circular sería el mostrado a continuación:

```
unit uniA;
   ...
implementation
  uses uniB;
  ...
```

```
unit uniB;
...
implementation
  uses uniA;
  ...
```

```
program programaC;
uses uniA, uniB;
...
```

9.9. OTRAS OPCIONES DE COMPILACIÓN EN EL ENTORNO DE TURBOPASCAL

Ademas de la opción *Compile* pueden emplearse otras opciones para compilar unidades y programas en el entorno de TurboPascal. La opción *Make* además de compilar un programa, gestiona el uso de las unidades utilizadas en dicho programa. Si se compila algún programa fuente con esta orden no sólo compila dicho programa fuente, también vuelve a compilar automáticamente cualesquiera unidades empleadas por el programa sólo si ha habido alguna variación en el archivo fuente de la unidad desde su última compilación.

9.9.1. La opción MAKE

Empleando esta opción, el compilador *detecta* si ha habido alguna variación en el archivo fuente de una unidad si la hora/fecha de generación/modificación de éste (`.pas`) es posterior a la hora/fecha de generación/modificación del archivo que contiene la unidad compilada (`.tpu` o `.tpp`). En este caso, el compilador procesa en primer lugar la unidad fuente generando un nuevo archivo que contiene la unidad compilada (con la hora y fecha actual, posterior a la hora/fecha del archivo que contiene la unidad fuente) y después compila el programa.

9.9.2. La opción BUILD

La opción *Build* es semejante a *Make* salvo que la compilación de las unidades declaradas en un programa es ahora incondicional (vuelve a compilar las unidades utilizadas aunque no haya habido variación), es decir, no comprueba las hora/fecha de generación/modificación de los archivos correspondientes.

9.9.3. Declaraciones encadenadas de unidades

Es importante tener en cuenta que el compilador de TurboPascal no acepta declaraciones "anidadas" o "encadenadas" de unidades. Es decir, si se declara el uso de una unidad A dentro de una unidad B (ya sea en la sección de interfaz o en la sección de implementación), y un programa C declara el uso de la unidad B ¿puede entenderse que el programa C podrá utilizar lo declarado dentro de la unidad A?. La respuesta es no. Para que el programa C pueda emplear lo declarado en la unidad A, es necesario declarar explícitamente el uso de la unidad A en el programa C.

9.10. LA UTILIDAD TPUMOVER

La utilidad `TPUMOVER` es un programa que sirve para conseguir información de la librería de unidades estándar `turbo.tpl` así como para integrar, eliminar y separar unidades de la misma. En principio, en el archivo `turbo.tpl` se encuentran las unidades estándar `System`, `Crt`, `Overlay`, `Dos` y `Printer`. Este archivo se almacena siempre en memoria cuando se carga el TurboPascal al ejecutar el programa `turbo.exe`. El contenido de

`turbo.tpl` puede modificarse añadiendo nuevas unidades o eliminando alguna de las ya existentes con la utilidad `TPUMOVER.EXE`.

9.10.1. Visualización del contenido de la librería de unidades estándar

Para emplear esta utilidad con el fin de conocer el contenido de la librería de unidades estándar `turbo.tpl` hay que ir al directorio de instalación donde se encuentre este archivo ejecutable y teclear `tpumover turbo.tpl<Intro>` según muestra la Figura 53.

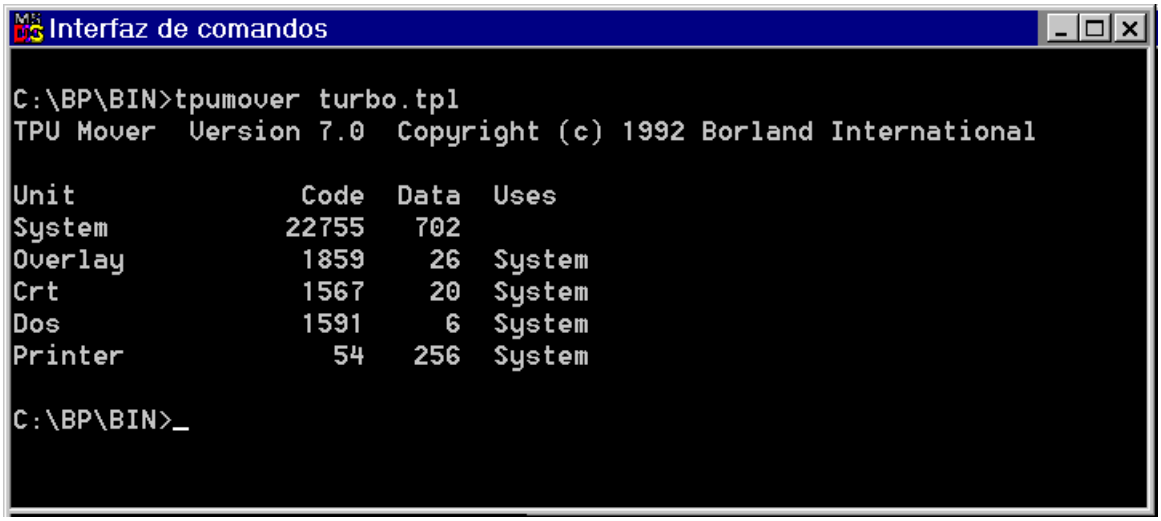

Figura 53. Ejemplo de uso de `tpumover` para ver el contenido de la librería `turbo.tpl`

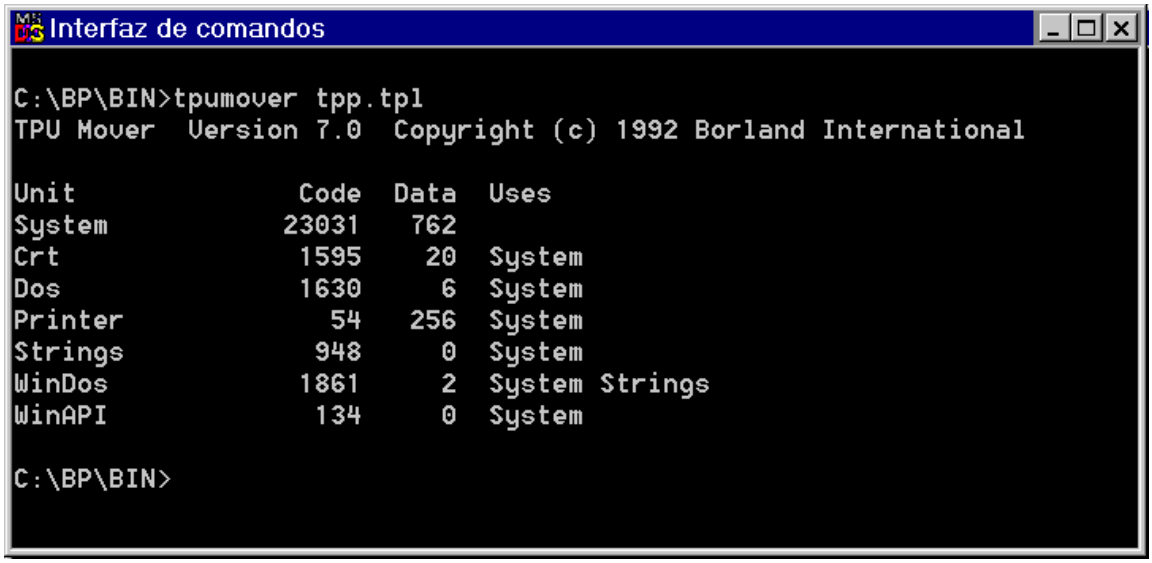

Figura 54. Ejemplo de uso de `tpumover` para ver el contenido de la librería `tpp.tpl`

9.10.2. Manipulación del contenido de la librería de unidades estándar

Para manipular su contenido se utiliza la línea de comandos en el sistema operativo D.O.S.:

`$\>tpumover turbo.tpl <signo>`*`nombre_unidad`*`<Intro>`

donde `<signo>` puede ser:

`+` para añadir la unidad *`nombre_unidad`* a `turbo.tpl`

- para eliminar la unidad *nombre_unidad* de `turbo.tpl`
- * para extraer la unidad *nombre_unidad* de `turbo.tpl` creando el archivo independiente *nombre_unidad*`.tpu`

Por ejemplo:

```
$\>tpumover turbo.tpl -overlay<Intro>
TPU Mover  Version 7.0  Copyright (c) 1992 Borland International

$\>tpumover turbo.tpl<Intro>
TPU Mover  Version 7.0  Copyright (c) 1992 Borland International
Unit              Code  Data  Uses
System           22755   702
Crt               1567    20  System
Dos               1591     6  System
Printer             54   256  System

$\>_
```

La principal ventaja de tener una unidad en este archivo consiste en que, al estar ya almacenado en memoria, su acceso es más cómodo y rápido ya que no será necesario que la unidad esté almacenada en el disco como un archivo independiente: el acceso a disco es siempre más lento que el acceso a memoria. El inconveniente es que cuantas más unidades se incluyan dentro de `turbo.tpl` menos espacio quedará disponible en memoria para otras necesidades.

Bibliografía básica

- **García-Beltrán, A., Martínez, R.** y **Jaén, J.A.** *Métodos Informáticos en TurboPascal*, Ed. Bellisco, 2ª edición, Madrid, 2002
- Borland Pascal with Objects - Language Guide, Editorial Borland, 1992
- Borland Pascal with Objects - Programmer's Reference, Editorial Borland, 1992
- **Duntemann, J.** *La Biblia de TurboPascal*, Anaya Multimedia, Madrid, 1991