2. ESTRUCTURA DE UN PROGRAMA EN TURBOPASCAL

Conceptos: *Programación estructurada, Estructura de un programa, Cabecera, Declaración, Cuerpo Principal, Comentario, Metacomando, Identificador.*

Resumen: Uno de los objetivos de la programación estructurada es el incremento de la productividad en el desarrollo de programas, reduciendo de forma notable el tiempo requerido para escribir, verificar, depurar y mantener los programas. A continuación se introduce el lenguaje y el entorno integrado de TurboPascal como idóneo para la implementación de programas que siguen la programación estructurada. TurboPascal no es un solo un lenguaje derivado del Pascal estándar con nuevas y potentes características, sino que es un entorno en el que se integra un editor de texto para escribir programas fuente sin necesidad de salir del entorno, un compilador muy rápido que detecta los errores sintácticos en tiempo de compilación y un conjunto de herramientas para depurar y verificar la corrección del programa. Por otro lado se introduce la estructura de un programa en TurboPascal con la estructura fija que le caracteriza: un encabezamiento, una zona de declaraciones y el cuerpo principal del programa. La última parte del tema se focaliza en el concepto de identificador.

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

a) Describir la estructura del código fuente de un programa en TurboPascal (*Conocimiento*)
b) Interpretar la estructura del código fuente de un programa en TurboPascal (*Comprensión*)
c) Definir los conceptos de edición, compilación y ejecución (*Conocimiento*)
d) Realizar la codificación, compilación y ejecución de un programa dado en TurboPascal (*Aplicación*)
e) Definir el concepto de identificador (*Conocimiento*)
f) Interpretar la estructura del código fuente de un programa en TurboPascal (*Aplicación*)

2.1. INTRODUCCIÓN

En este capítulo se verá cuál es la estructura de un programa en TurboPascal, pero antes es conveniente aclarar algunos términos:

- **Pascal** es un lenguaje de programación de ordenadores de propósito general.
- **TurboPascal** es la versión mas extendida del lenguaje Pascal. Desarrollado por la compañía Borland más tarde denominada Inprise.
- **Delphi** es una herramienta visual para Windows desarrollada por Borland basada en Pascal.

TurboPascal es una versión actualizada y ampliada de Pascal, el lenguaje de programación creado por Niklaus Wirth con fines académicos a principios de los 70. Entre otras extensiones, TurboPascal amplía la capacidad de manejar tipos de datos numéricos, introduce el tipo cadena (`string`), facilita y amplía el uso de los archivos y punteros y potencia el uso de las unidades. Estas modificaciones confieren a este lenguaje una mayor potencia a costa de restarle posibilidades a sus programas fuente de ser migrados a otros sistemas que no admitan tales extensiones de TurboPascal.

Los programas se diseñan para resolver un problema determinado utilizando diversos datos en una serie de etapas: entrada de datos, almacenamiento de éstos, operaciones, obtención de resultados y salida de los mismos. Para mantener un orden, la **Programación Estructurada** organiza el programa mediante una secuencia de pasos a seguir evitando saltos o laberintos. Dentro de esta secuencia de pasos pueden utilizarse acciones alternativas bajo árboles de condiciones y acciones repetitivas o bucles.

2.2. ESTRUCTURA DE UN PROGRAMA

Un programa desarrollado en TurboPascal consta en general de una cabecera, una sección o zona de declaraciones y un cuerpo principal.

Tabla 8. Esquema de la estructura de un programa en TurboPascal 7.0

Sección		*Palabras Clave asociadas*
CABECERA		`PROGRAM`
SECCIÓN o ZONA DE DECLARACIONES	Unidades	`USES`
	Etiquetas	`LABEL`
	Constantes	`CONST`
	Tipos	`TYPE`
	Variables	`VAR`
	Funciones y Procedimientos	`FUNCTION` `PROCEDURE`

CUERPO PRINCIPAL DEL PROGRAMA		`BEGIN`
		`...`
		`END.`

2.3. CABECERA

La cabecera de un programa es opcional y puramente informativa. Si existe se compone de una única sentencia que sirve para asociar un nombre o *identificador* al programa. Más adelante, en la sección 2.10 de este capítulo, se indicarán las secuencias de caracteres que

pueden utilizarse como identificadores. Este identificador no tiene por qué ser necesariamente el mismo que el del archivo fuente que lo contiene, ni que el del archivo ejecutable una vez compilado. La cabecera, como, en general, cualquier otra sentencia de un programa, se separa de las demás sentencias con un carácter de punto y coma.

Sintaxis: `PROGRAM Nombre_Programa;`

En el siguiente ejemplo se muestra la cabecera de un programa que gestiona un listado de números de teléfono:

Ej.: `Program Listin;`

2.4. SECCIÓN O ZONA DE DECLARACIONES

En todo programa de TurboPascal es necesario declarar o definir previamente todo lo que se vaya a utilizar y que no tenga un significado específico o *a priori* para este lenguaje de programación. En esta sección se realizan estas definiciones o declaraciones del programa. Exceptuando la declaración de utilización de unidades que, si existe, es única y deberá incluirse al principio, el número y orden de las demás declaraciones no es rígido. A este respecto, la única norma general que es necesario respetar es que *cualquier elemento que se utilice en un punto determinado del programa deberá haber sido declarado previamente*.

2.4.1. Declaración de utilización de unidades

La sentencia de declaración de unidades especifica el nombre o identificador de las unidades que se van a utilizar en el programa. Como se verá más adelante detenidamente, una unidad es una colección de declaraciones de constantes, tipos de datos, variables, funciones y procedimientos que pueden emplearse en un programa de TurboPascal. Si son varias unidades se podrán declarar en la misma sentencia separándolas por comas.

Sintaxis: `USES` *`Unidad1, Unidad2, Unidad_n`*`;`

Si existe una sentencia de declaración de unidades en un programa deberá colocarse al principio de la sección de declaraciones de dicho programa, es decir, antes de cualquier otra declaración. En el siguiente ejemplo se declara el uso en el programa de dos unidades denominadas `Crt` y `Dos`:

Ej.: `Uses Crt, Dos;`

2.4.2. Declaración de etiquetas

Permiten realizar saltos incondicionales en la secuencia de instrucciones de un programa. Su utilización va unida a la sentencia `goto` y, aunque es un elemento incluido en la sintaxis de Pascal estándar, no se recomienda por la filosofía de la programación estructurada (que evita los saltos incondicionales).

Sintaxis: `LABEL` *`Etiqueta1, Etiqueta2, Etiqueta_n`*`;`

Una etiqueta es un *identificador* o una secuencia de cuatro dígitos decimales (entre 0 y 9999). Si las etiquetas son varias se podrán declarar en la misma sentencia separándolas por comas.

Ej.: `Label 100, 200;`

2.4.3. Declaración de constantes

Las constantes son datos que no cambian durante la ejecución del programa y que se definen durante el tiempo de compilación.

Sintaxis:
```
CONST Nombre_Constante = Expresion_1;
      Nombre_Constante_2 = Expresión_2;
```

```
                    Nombre_Constante_3 = Expresión_3;...
```

Si se declaran varias constantes en un programa podrán incluirse en una única sentencia `CONST` separando cada declaración de las demás con caracteres de punto y coma, aunque también puede haber varias sentencias `CONST` en la sección de declaraciones de un programa.

Ej.:
```
Const Pi = 3.1415;     { constante numerica real }
      Limite = 325;     { constante numerica entera }
      Saludo = '¡Hola!';      { cadena de caracteres }
```

2.4.4. Declaración de tipos de dato

Un tipo de dato es un conjunto de valores de datos. En el lenguaje de programación TurboPascal todo dato ha de pertenecer a algún *tipo* determinado. Esta especificación determinará cómo se almacenará el dato correspondiente y qué operaciones se podrán realizar con dicho dato durante la ejecución del programa. En TurboPascal hay tipos predefinidos que no es necesario declarar (tipos de datos numéricos enteros, numéricos reales, lógicos o booleanos, caracteres...) y otros que no lo están y que el programador deberá declarar.

La declaración de un tipo de dato consta del nombre o identificador del tipo de dato seguido de los valores que pueden tomar los datos de ese tipo. Por otro lado, existe la posibilidad de que algunos tipos puedan ser subconjuntos o *subrangos* de otros tipos. También es necesario declarar estos tipos de datos.

Sintaxis:
```
TYPE Nombre_Tipo_1 = Definicion_1;
     Nombre_Tipo_2 = Definicion_2;
     Nombre_Tipo_3 = Definicion_3;...
```

`Definicion_n` puede ser una lista de valores que van entre paréntesis (tipo de dato enumerado), un subconjunto de otro tipo ya definido o tipo subrango (en este caso se indica el valor inicial y final que define el subconjunto) o la especificación en cuanto a tamaño o estructura de un tipo de dato más complejo o *estructurado*.

Si se declaran varios tipos de dato en un programa podrán incluirse en una única sentencia TYPE separando cada declaración de las demás con caracteres de punto y coma. En cualquier caso, también puede haber varias sentencias TYPE en la sección de declaraciones de un programa.

En el siguiente ejemplo se incluyen en una misma declaración, los tres primeros son tipos de datos enumerados, los dos siguientes son de tipo subrango (numérico entero y de caracteres, respectivamente) y el último es de tipo estructurado como cadena de veinte caracteres.

Ej.:
```
type palo      = (bastos, oros, copas, espadas);
     Estado    = (soltero, casado, viudo);
     dia       = (lu,ma,mi,ju,vi,sa,dm);
     digito    = 0..9;
     minuscula = 'a'..'z';
     nombre    = string[20];
```

2.4.5. Declaración de variables

Una variable es un espacio de la memoria reservado durante la ejecución del programa a la que se le asocia un nombre o identificador y en la que se puede almacenar un valor que puede cambiar durante dicha ejecución. La declaración consta de la palabra VAR seguida del identificador de cada variable y su tipo, que puede ser predefinido o estar definido previamente en la sección anterior.

Sintaxis:
```
VAR Variable: Tipo;
```

Si se declaran varias variables del mismo tipo pueden incluirse en la misma sentencia de declaración separadas por comas.

Sintaxis: `VAR` *`Variable_1,Var_2,...,Var_n`*`: Tipo;`

Si se declaran varias variables en un programa podrán incluirse en una única sentencia VAR separando cada declaración de las demás con caracteres de punto y coma, aunque también puede haber varias sentencias VAR en la sección de declaraciones de un programa.

Ej.:
```
VAR   x,y,z        : Real;
      i,j          : Integer;
      condicion    : estado;
      nota         : digito;
      libra        : dia;
```

En el ejemplo anterior se declaran ocho variables de las cuales las cinco primeras son de tipos predefinidos por TurboPascal (tres de tipo `Real` y dos de tipo `Integer`, respectivamente) y las tres últimas aprovechan las declaraciones de tipos de dato del ejemplo de apartado anterior.

Al declarar una variable se reserva espacio en memoria para almacenar los valores que va tomando dicha variable durante la ejecución del programa. La cantidad de memoria reservada dependerá del tipo de variable. Una variable de tipo `Integer` es una variable numérica entera que ocupa 2 bytes (16 bits) de memoria, mientras que una de tipo `Real`, es una variable numérica real que necesita 6 bytes (48 bits).

2.4.6. Declaración de funciones y procedimientos

Las funciones y procedimientos son las rutinas, subrutinas o subprogramas de Pascal. Una rutina es un conjunto de instrucciones que pueden ejecutarse en cualquier lugar del programa principal o, dentro de otras subrutinas, sólo referenciando su nombre o identificador. Como se verá más adelante, existen rutinas ya predefinidas o estándar en TurboPascal. Se tendrán que declarar obligatoriamente las subrutinas no predefinidas que vayan a utilizarse en el programa o que no estén incluidas en unidades cuyo uso se declare en el programa.

Las subrutinas tienen una estructura muy parecida a los programas con las excepciones de que su cabecera empieza por la palabra `FUNCTION` o `PROCEDURE` y su cuerpo no acaba en un punto sino en un carácter de punto y coma. Si bien tanto las funciones como los procedimientos pueden ejecutar una serie de sentencias, las funciones se diferencian de los procedimientos en que, una vez finalizada su ejecución, devuelven un valor, cuyo tipo de dato se especifica al final de la cabecera.

Sintaxis de la declaración de una función:
```
FUNCTION Nombre_Funcion (parametros): Tipo;
       Begin
       { sentencias ... }
       End;
Sintaxis de la declaración de un procedimiento:
PROCEDURE Nombre_Procedimiento (parametros);
       Begin
       { sentencias ... }
        End;
```

A continuación se presenta un ejemplo de declaración de una función que devuelve la media de dos números reales y de un procedimiento que visualiza un cuadro de caracteres por la pantalla.

Ej.:
```
Function Media(x1,x2:Real):Real;
  BEGIN
  Media:=(x1+x2)/2
  END;
Procedure Cuadro;
  BEGIN
  WriteLn('********************');
  WriteLn('*                  *');
  WriteLn('********************')
  END;
```

Como se ha comentado anteriormente, en general, las secciones de declaraciones de etiquetas, constantes, tipos de dato, variables, funciones y procedimientos pueden incluirse en cualquier orden en el programa y repetirse cualquier número de veces. Lo único que hay que respetar es que, cualquier identificador que se utilice en un punto determinado de un programa debe haber sido declarado previamente.

2.5. CUERPO PRINCIPAL DEL PROGRAMA

Es la parte final del archivo fuente, situado a continuación de la sección de declaraciones y delimitado por la pareja de palabras `BEGIN` y `END`. Incluye la secuencia de sentencias que se van llevando a cabo cuando se ejecuta el programa. En TurboPascal, cada sentencia se separa de la siguiente con un carácter de punto y coma. Después del `END` del cuerpo principal del programa siempre hay que poner un punto que indica al compilador que ha llegado el final del programa.

Sintaxis:
```
BEGIN
{ Sentencias del cuerpo del programa }
END.
```

2.6. COMENTARIOS

Los comentarios son sentencias que pueden incluirse en, prácticamente, cualquier parte del programa y que son ignorados por el compilador (¡el traductor del código del programa fuente!). Sólo sirven para dar información o explicaciones sobre el conjunto o alguna parte del programa a la persona que lea el código fuente. Esto cobra especial importancia cuando se trata de un código más o menos complejo o desarrollado por un equipo de programadores. Los comentarios suelen ir delimitados entre caracteres de llaves: `{` y `}` o entre las parejas de caracteres `(*` y `*)`.

Ej.:
```
{ Esto es un comentario }
(* Esto tambien es un comentario *)
(* Este otro comentario ocupa
   mas de una linea *)
```

Se recomienda especialmente a los programadores noveles que se acostumbren a *comentar* el código fuente de los programas que vayan creando.

2.7. METACOMANDOS O DIRECTIVAS DEL COMPILADOR

Los metacomandos son órdenes para el compilador insertadas en el código fuente de un programa. Estas órdenes para el compilador también llamadas, en general, directivas del compilador, pueden darse a su vez, a través del menú `Options` del entorno de TurboPascal. Los metacomandos se insertan como los comentarios: entre llaves `{ }` o paréntesis `(* *)` y comienzan con el signo `$`. Hay directivas globales que afectan a toda la compilación y que se insertan antes de las declaraciones y directivas locales que pueden aparecer en "casi" cualquier sitio y que afectan sólo a parte de la compilación. Algunas directivas pueden llevar parámetros o argumentos. Por ejemplo, existe una directiva de compilación que controla la generación de errores en las operaciones de Entrada y Salida de datos (por ejemplo, en la escritura de datos en un archivo) en tiempo de ejecución. Por defecto, su estado es activa `{$I+}`.

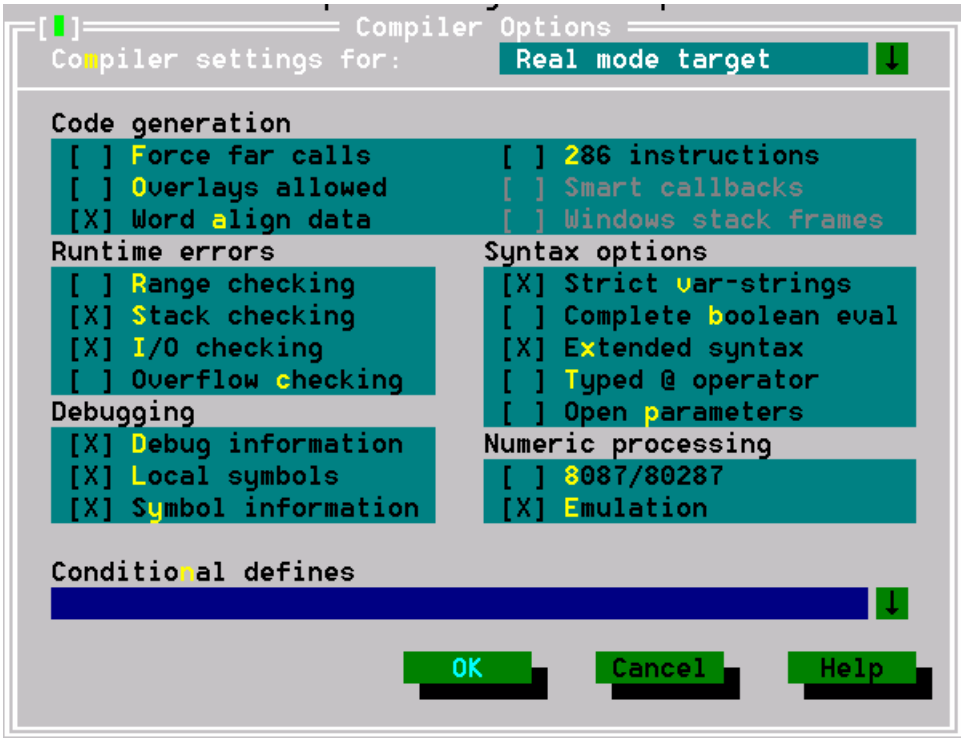

Figura 4. Menú de Options del entorno de TurboPascal 7.0

2.8. EJEMPLO DE UN PROGRAMA DE TURBOPASCAL

A continuación se muestra un ejemplo sencillo de programa con sus partes típicas: cabecera, sección de declaraciones y cuerpo del programa principal. Se aconseja al lector que pruebe a teclearlo, compilarlo y ejecutarlo en su entorno de TurboPascal. Para llevar a cabo esta tarea podrán resultar de ayuda la secuencia de pasos mostrada en las Figuras 1, 2, 3, 4 y 5, así como la información facilitada en el apéndice *El entorno de programación*.

```
(*******************************************************)
{ El programa Adicion devuelve el resultado de sumar dos }
{ numeros enteros                                       }
{ Autor              :    Angel Garcia y Beltran        }
{ Ultima revision    :    30 de junio de 2007           }
(*******************************************************)

PROGRAM Adicion;
{ Comienza la seccion de declaraciones del programa }

USES Crt;
VAR a,b,c :Integer;
{ El procedimiento Linea escribe una secuencia de asteriscos
{ en pantalla }
PROCEDURE Linea;
  BEGIN
  WriteLn('****************************')
  END;
{ Fin de la seccion de declaraciones del programa }
{ Comienza el cuerpo con las sentencias del programa }

BEGIN
{ El procedimiento ClrScr esta declarado en la Unidad Crt, }
{ limpia toda la pantalla y situa el cursor en la }
{ parte superior izquierda de la pantalla }
ClrScr;
```

```
{ Llamada al procedimiento Linea declarado anteriormente }
{ en la seccion de declaraciones del programa }
Linea;
{ Las siguientes sentencias asignan valores a las }
{ variables numericas enteras a y b, respectivamente }
a:=3;
b:=4;
{ La siguiente sentencia calcula la expresion a+b }
{ y posteriormente asigna el resultado a la variable entera c }
c:=a+b;
{ El procedimiento WriteLn visualiza el dato }
{ almacenado en la variable c }
WriteLn(' La suma es ',c);
{ Y otra vez llamada al procedimiento Linea }
Linea

END. { Fin del codigo del programa }
```

2.9. PASOS PARA LA CONSTRUCCIÓN DE UN PROGRAMA EN EL ENTORNO DE PROGRAMACIÓN

Se parte de la base de que el entorno de programación de TurboPascal está previamente instalado y configurado en un ordenador con sistema operativo Windows 95 o superior. El icono de acceso al entorno de programación puede aparecer como muestra la Figura 5.

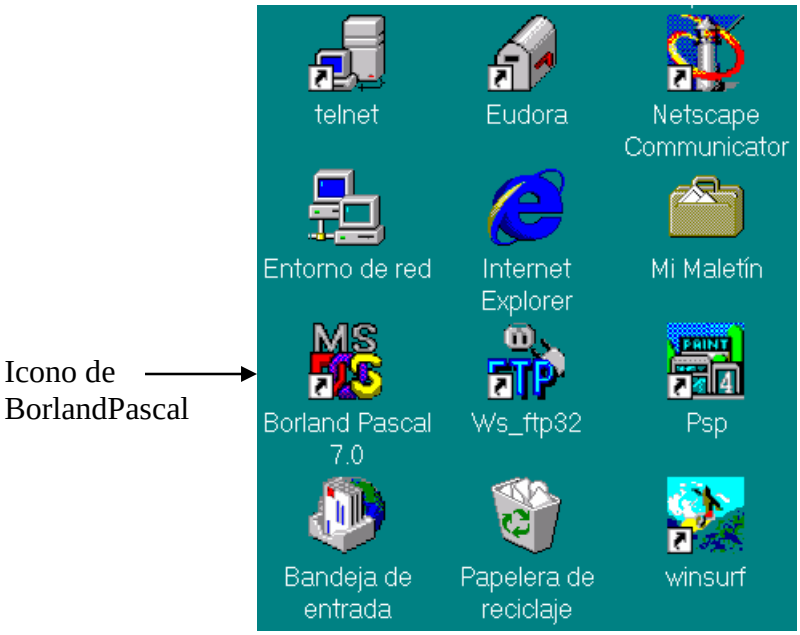

Figura 5. Icono de acceso directo al entorno de programación de Borland Pascal en el sistema operativo Windows

En primer lugar, se utiliza el editor de textos integrado en el entorno de programación para construir el archivo de texto que contiene el **código fuente** del programa o programa fuente (Figura 6). En principio, podría emplearse cualquier otro editor o procesador de textos que permita manipular y almacenar archivos en formato ASCII.

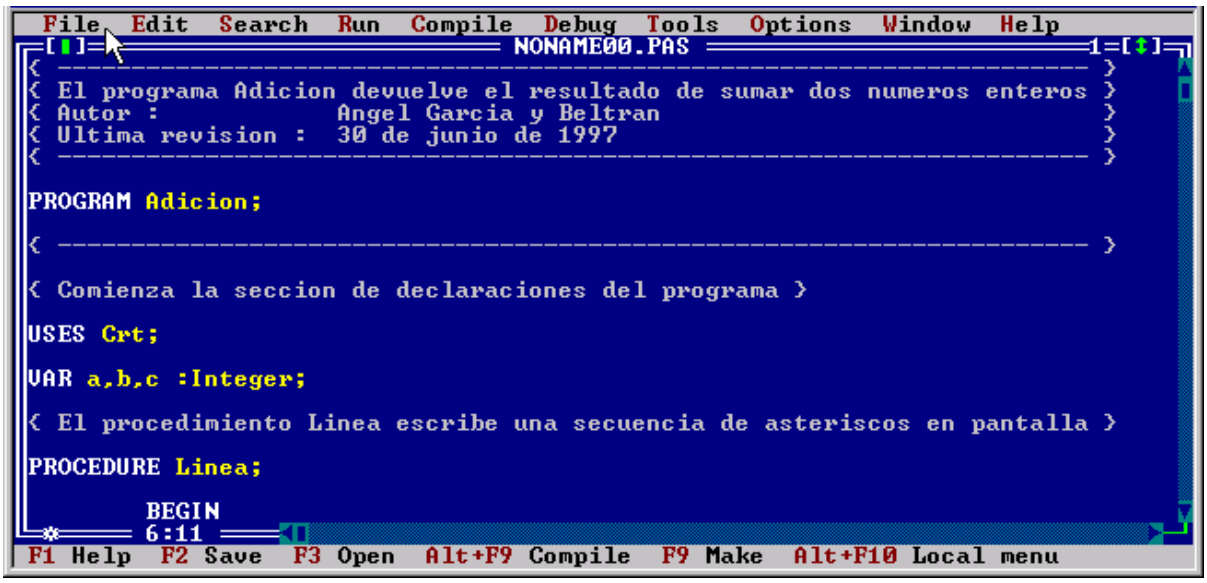

Figura 6. Edición del programa fuente en el entorno de Borland Pascal 7.0 (modo protegido)

Una vez escrito el texto del programa fuente se aconseja almacenarlo con un nombre determinado, por ejemplo `adicion`, en el disco. Para realizar dicha acción, se selecciona la opción File/Save as y se escribe el nombre en la ventana de diálogo que aparece (Figura 7). Al pulsar la opción OK, el archivo es almacenado en disco con el nombre `adicion.pas`. Por defecto, los archivos que almacenan programas fuente tienen extensión `.pas`.

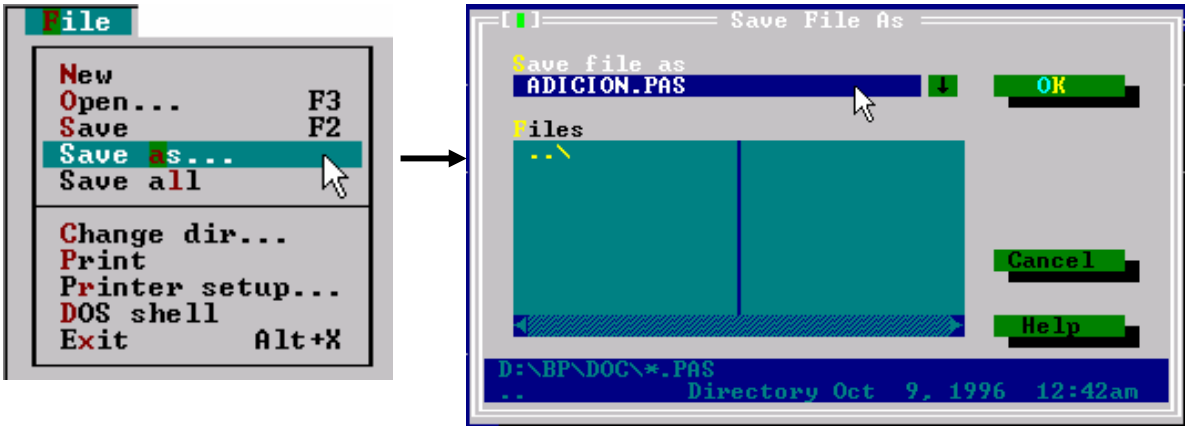

Figura 7. Almacenamiento en disco del archivo `adicion.pas` que contiene el código fuente del programa

El siguiente paso es la compilación o traducción del programa fuente a código máquina lo que se puede realizar con la opción Compile/Compile (Figura 8). Si la compilación ha tenido éxito, incluida el análisis sintáctico previo del código fuente, se muestra la ventana de la derecha. En caso, contrario, debe corregirse el error (de compilación) en el código fuente y repetirse el proceso hasta que no haya ningún error (Figura 2.5).

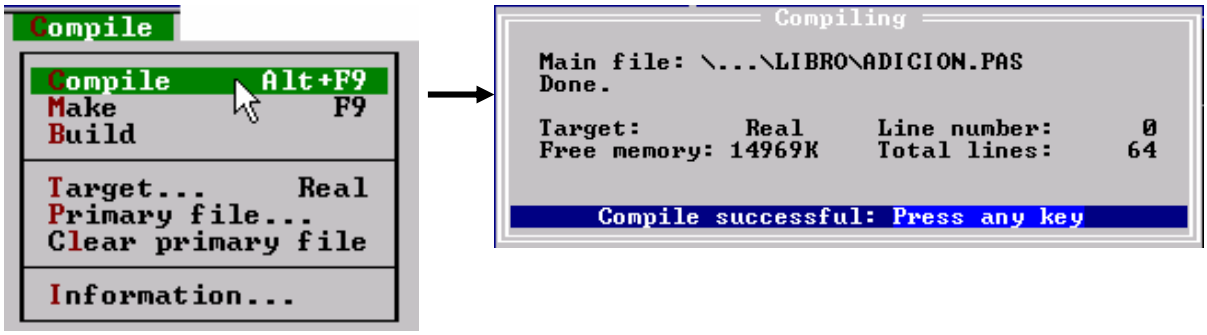

Figura 8. Compilación del programa fuente `adicion.pas` y generación del ejecutable `adicion.exe`

La compilación tiene como resultado la generación de un archivo que contiene el código traducido, objeto o ejecutable. En este caso, al archivo creado se le da el nombre `adicion.exe`. Finalmente, puede ejecutarse este programa ya compilado mediante la opción Run/Run (Figura 9) o desde el terminal DOS del sistema operativo. En el primer caso, puede observarse en todo momento la salida por pantalla del programa mediante la opción Debug/User screen.

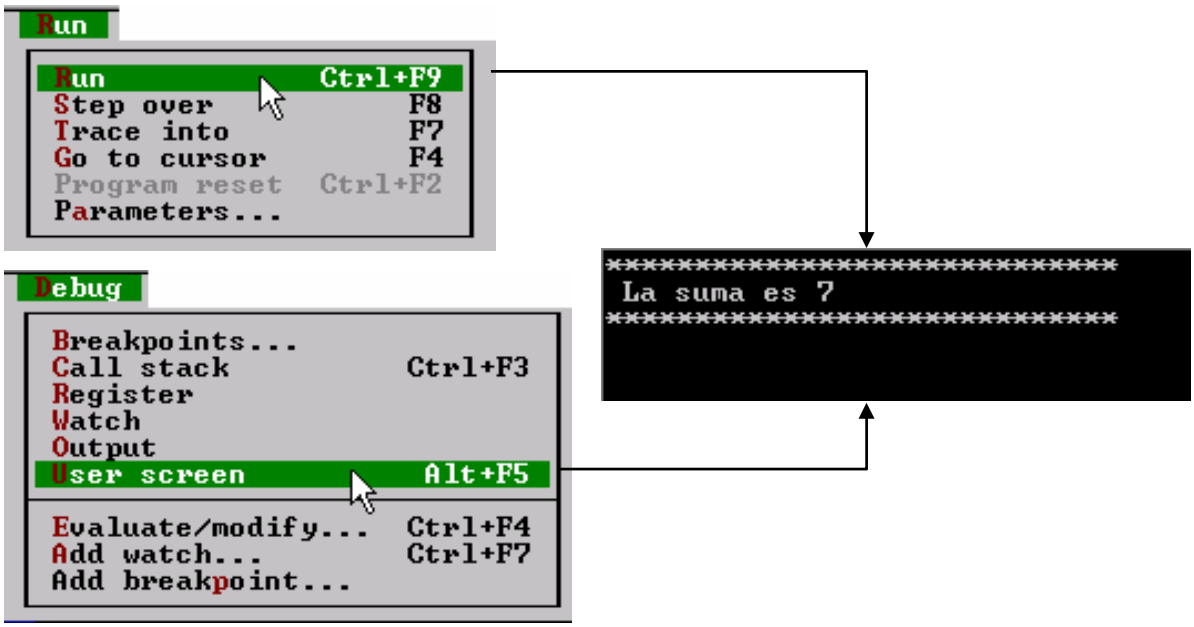

Figura 9. Ejecución del programa `adicion.exe` desde el entorno de programación y observación de la salida por pantalla

El programa también puede ser ejecutado desde el terminal DOS escribiendo el nombre del archivo ejecutable en la línea de comandos (Figura 10).

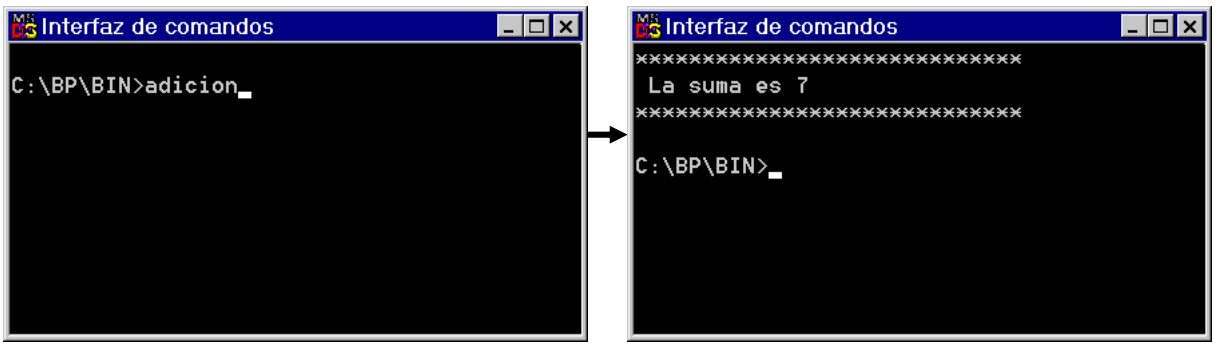

Figura 10. Ejecución del programa adicion.exe desde el terminal DOS de Windows

2.10. IDENTIFICADORES

Un **identificador** es un nombre de un programa, una variable, una constante, un tipo de dato, una función, un procedimiento, una etiqueta, una unidad o un campo de una estructura en un programa de TurboPascal. Existen identificadores que tienen ya un significado definido en TurboPascal y otros nuevos cuyo significado el programador puede definir. En el programa ejemplo anterior, `program`, `Adicion`, `Crt`, `a`, `b`, `c`, `Integer`, `begin`, `Linea`,... son identificadores. Los identificadores sólo existen en el código del programa fuente y no en el programa objeto (que es el resultado de la compilación del programa fuente correspondiente). En TurboPascal todo nuevo identificador se debe definir unívocamente con anterioridad a su utilización.

2.10.1. Normas para la construcción de identificadores

Para construir un identificador es necesario seguir una serie de normas que se dan a continuación:

1. Un identificador se compone de una sucesión de letras, dígitos decimales y caracteres de subrayado. No son válidos otros caracteres, por ejemplo, caracteres de espacio en blanco o guiones. Pueden tener cualquier longitud pero sólo los primeros 63 caracteres son significativos.
2. Dicha sucesión debe empezar por una letra o un carácter de subrayado.
3. En un programa no puede declararse un mismo identificador más de una vez. Con las excepciones que se verán en los capítulos de *Rutinas* y *Unidades*.
4. En los identificadores **no** se diferencian los caracteres en mayúsculas de las minúsculas (Por ejemplo: `DiaMes` = `Diames` = `diames` = `DIAMES`).
5. Existen una serie de palabras reservadas que no se pueden utilizar como identificadores (ver en el siguiente apartado del capítulo, por ejemplo, `BEGIN`) y otras que tienen un significado definido en TurboPascal (por ejemplo, `WRITELN`) y que, en este último caso, podrían redefinirse (se les puede dar otro empleo) lo que no es nada recomendable por su habitual utilización.
6. Aunque no es una norma de obligado cumplimiento, es conveniente utilizar identificadores significativos para orientar al usuario o a cualquier otra persona que accede al programa fuente sobre lo que representan.

Algunos ejemplos de identificadores no válidos son: `Dia-semana`, `Dia semana`, `2mes` o `Const`. En el primer ejemplo se ha utilizado un carácter no válido (`-`); en el segundo identificador, también (el espacio en blanco); el tercero, empieza por un carácter numérico y el cuarto es una palabra reservada del lenguaje de programación TurboPascal. Algunos ejemplos

de identificadores válidos son: `Dia_semana`, `Mes2` o `Constante`. Por otro lado y como se verá más adelante, en TurboPascal no está prohibido emplear el mismo identificador en distintas unidades. En este caso, para determinar la referencia en el programa es necesario anteponer al identificador de lo que se quiera referenciar el nombre de la unidad correspondiente seguido de un punto: `unidad.identificador`. Esto es lo que se denomina un *identificador cualificado.*

Para su diferenciación de los demás elementos en los ejemplos incluidos en esta publicación los identificadores se escribirán en el tipo de letra `Courier`.

2.10.2. Directivas e identificadores o palabras reservadas

Existe otro grupo de identificadores correspondientes a *directivas estándar* y *directivas de procedimiento* en TurboPascal. A diferencia de las palabras reservadas, estos identificadores pueden redefinirse, pero no es aconsejable.

```
absolute
assembler               (directiva de procedimiento)
export                  (directiva de procedimiento)
external                (directiva de procedimiento)
far                     (directiva de procedimiento)
forward                 (directiva de procedimiento)
index
interrupt               (directiva de procedimiento)
near                    (directiva de procedimiento)
private
public
resident
virtual                 (directiva de procedimiento)
```

Existen identificadores o palabras reservadas que tienen un significado específico y fijo dentro del lenguaje TurboPascal 7.0 y que, en ningún caso, pueden ser redefinidas. Son los siguientes:

```
and                     asm                     array
begin                   case                    const
constructor             destructor              div
do                      downto                  else
end                     exports                 file
for                     function                goto
if                      implementation          in
inherited               inline                  interface
label                   library                 mod
nil                     not                     object
of                      or                      packed
procedure               program                 record
repeat                  set                     shl
shr                     string                  then
to                      type                    unit
until                   uses                    var
while                   with                    xor
```

Finalmente, existen otros identificadores definidos en las unidades *estándar* de TurboPascal, que se verán posteriormente: tipos de dato, funciones, procedimientos,... que, por su uso habitual, tampoco se aconsejan redefinir (`Integer`, `Write`,...).

Bibliografía básica

- Borland Pascal with Objects - Language Guide, Editorial Borland, 1992
- Borland Pascal with Objects - Programmer's Reference, Editorial Borland, 1992
- **Joyanes, L**. Fundamentos de programación, Algoritmos y Estructuras de Datos, Editorial McGraw-Hill, 1996