CURSO IBM

PROGRAMAR es fácil

8

Borland Delphi

Programación Visual

Multimedia Ediciones, S.A.

Conceptos de programación (8)

Diagramas de flujo

Los diagramas de flujo son una de las piedras angulares sobre las cuales se ha basado la programación de aplicaciones desde su inicio, aunque actualmente su importancia es mucho menor para los programadores profesionales, sigue siendo de gran utilidad para los que inician sus actividades en este área.

Dar instrucciones precisas para la confección de un programa no es sencillo y menos aún si la documentación que se prepara está destinada a una persona distinta a su autor. Tampoco es tarea fácil, por otra parte, explicar a personas sin experiencia en programación la lógica de un determinado programa o de una sección de éste. Es en estos casos cuando los diagramas de flujo se revelan como herramientas muy útiles. Pasaremos revista en esta unidad a las características generales de estos diagramas.

DIAGRAMAS DE FLUJO

En ocasiones una imagen vale más que mil palabras. Sobre todo al dar los primeros pasos en el área de la programación, cuando los conceptos no están aún del todo claros y todavía nos encontramos en la fase de aprendizaje.

En estas situaciones, un esquema gráfico del flujo de un programa puede ayudarnos mucho. Para esto se emplean los llamados diagramas de flujo u organigramas.

En el contexto de la programación, un organigrama es una representación gráfica de la secuencia en la lógica de un programa, aunque también son utilizados para representar las secuencias de tareas en otras áreas, tales como la organización del trabajo de oficina, árboles genealógicos, organigramas de empresa, etc.

En informática los organigramas fueron uno de los primeros métodos empleados para el diseño de los programas. Con ellos los programadores representaban la lógica y el flujo dibujando una serie de símbolos normalizados, conectados entre sí mediante flechas que indican el paso de uno a otro en función de los resultados intermedios.

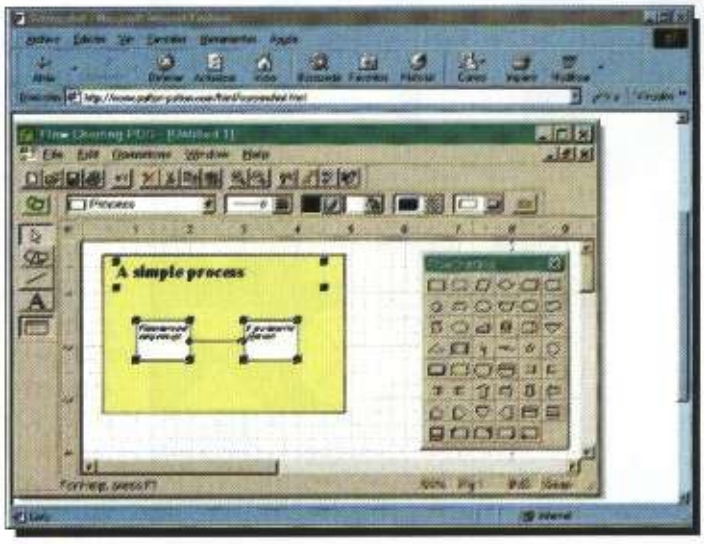

FIG 1. En el mercado se puede encontrar gran cantidad de software de ayuda para la elaboración de organigramas.

Aunque puede parecer un trabajo innecesario, siempre es mucho mejor dibujar este flujo, y a través de él detectar posibles inconsistencias o fallos en los programas, que intentar mantener toda la lógica en la cabeza, ya que con este esquema aumentan las posibilidades de que no se detecten a tiempo errores que con la ayuda del dibujo podrían evitarse. En la actualidad el organigrama se emplea relativamente poco y queda relegado a la tarea de herramienta de aprendizaje o al uso ocasional; es especialmente útil, también, al explicar el funcionamiento de un programa o de una sección del mismo a personas no versadas en programación, aunque siguen siendo utilizados por los jefes de proyectos o los propios directivos. En resumen, actualmente su uso puede considerarse como bastante esporádico por los desarrolladores aunque sigue manteniendo su utilidad general.

Como ya veremos más adelante al tratar el tema, se emplean nuevos tipos de organigrama para la *modelización* de datos (esto es, para diseñar cómo van a ser los datos, estructuras y clases de un programa) al usar métodos de programación orientada a objetos.

LOS SÍMBOLOS

Los organigramas tradicionales emplean símbolos geométricos estandarizados. Es muy importante que, tanto la persona que dibuja el organigrama como cualquiera de los posibles "lectores" del mismo, entiendan lo que significa cada uno de los símbolos utilizados, hasta el punto de que estos están normalizados en acuerdos internacionales. Concretamente, las normas ANSI 3.5 e ISO 1028 recogen la forma y significado de estos símbolos; ANSI e ISO son dos comités de estandarización, americano e internacional, respectivamente.

Los símbolos normalizados son muchos, por lo que vamos tan sólo a mencionar los más importantes; como podrá observar, una buena parte de estos símbolos se corresponde con instrucciones básicas de los lenguajes de programación, como principio y fin de programa, salto condicional, sentencia de programa, etc. La forma más o menos típica de estos símbolos puede apreciarse en las imágenes que ilustran este texto.

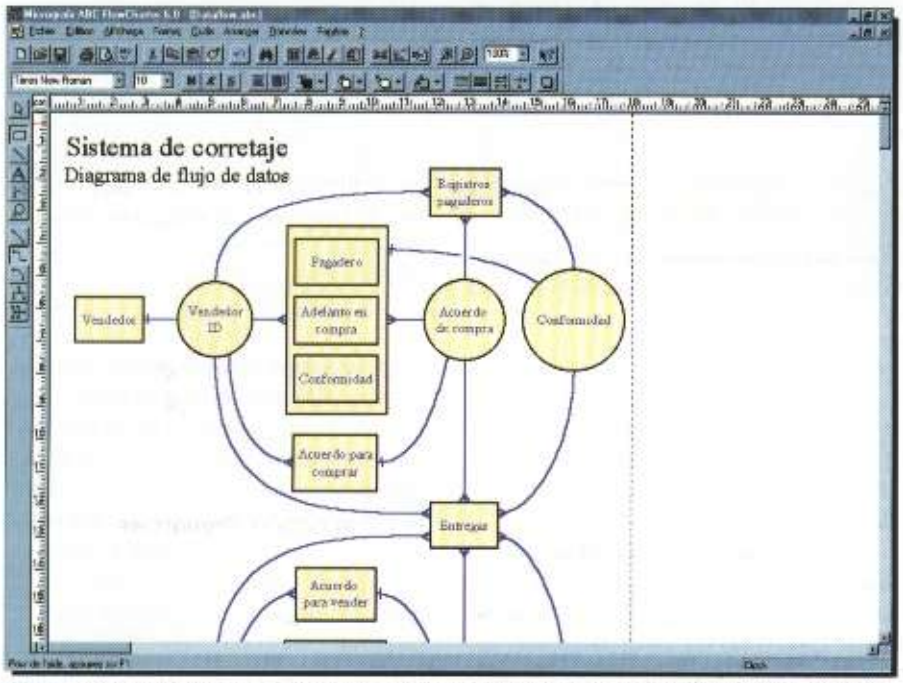

FIG 2. Los organigramas no sólo sirven para la representación de programas, sino para otras muchas aplicaciones.

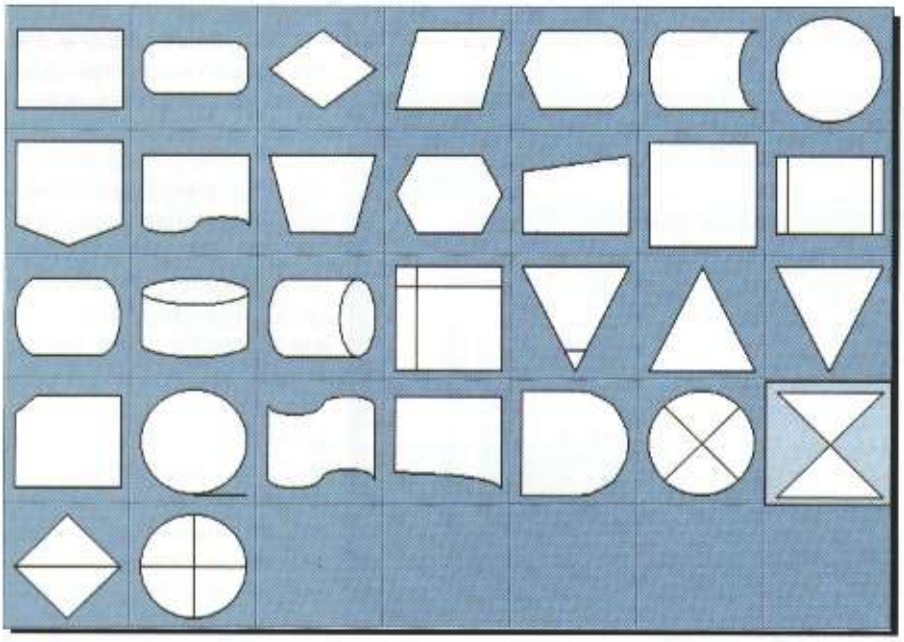

FIG 3. Existen diversos símbolos empleados en la confección de organigramas. Todos ellos están normalizados.

Los símbolos, como ya hemos indicado, se hallan conectados entre sí mediante flechas, que representan el paso de uno a otro. A un símbolo se llega normalmente a través de una flecha que sale de un símbolo anterior, y de él saldrá una o, si se trata de un símbolo de salto condicional, varias flechas en función del resultado que se haya obtenido.

El primer símbolo usado es el de principio o fin de programa. Éste se representa mediante una elipse con la palabra Inicio o Fin en su interior. Del símbolo de inicio sólo puede salir una flecha y al de fin sólo pueden llegar flechas.

Una sentencia de programa, o más bien un proceso o conjunto de sentencias que realizan una labor concreta, se representan mediante un rectángulo, en cuyo interior se coloca un texto descriptivo de la función desarrollada.

Un salto condicional se representa mediante un rombo que, como siempre, contiene en su interior un texto descriptivo de la evaluación que se ejecuta. A este símbolo, normalmente, llega una flecha (aunque no es incorrecto que puedan llegar varias), y salen dos o más en función de los resultados posibles, cada uno de los cuales nos lleva a seguir la lógica que toma el programa en ese momento.

Existen otros símbolos, como por ejemplo los empleados para representar un proceso de entrada o salida (grabación o lectura de disco), impresión, lectura de datos de la pantalla, etc. Estos símbolos, sobre todo en la programación moderna en entornos visuales, se emplean relativamente poco, aunque siguen apareciendo en la bibliografía al respecto o en los símbolos disponibles en cualquier herramienta de software dedicada a la creación de organigramas. En resumen, para poder dibujar el organigrama con una cierta lógica seguiremos las recomendaciones siguientes:

FIG 4. La elaboración y depuración de programas requiere del concurso de todas las personas afectadas por el problema a resolver.

- Decidir el nivel de detalle que vamos a representar mediante nuestro organigrama. Esto depende, sobre todo, de la finalidad con que estamos realizando dicho esquema.
- Listar las funciones y procedimientos que se incluyen en el gráfico y darles nombres comprensibles, aunque no excesivamente largos.
- Dibujar el organigrama. Por supuesto, esta tarea puede hacerse perfectamente con lápiz (sí, lápiz, porque poder borrar para solventar posibles errores nos vendrá bien) y papel; pero existen en el mercado gran variedad de programas diseñados con este fin que permiten dibujar más fácilmente y, lo que es mucho más importante, realizar cambios con facilidad y actualizar el organigrama a medida que sea necesario, sin tener que tirar tanto papel a la basura (además de ahorrarnos trabajo tienen su vertiente ecológica).

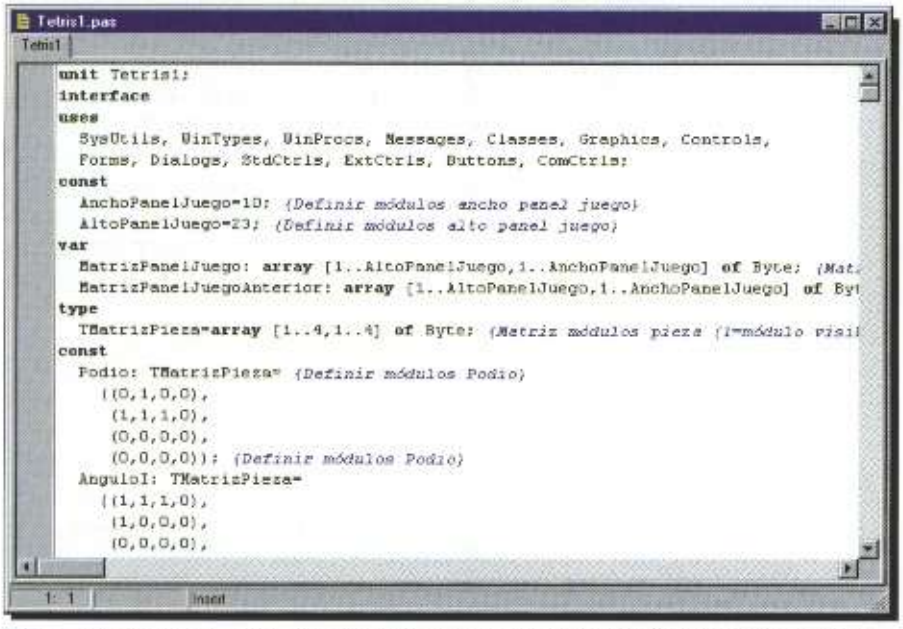

FIG 5. La documentación de los programas es cuestión fundamental. Por ello es recomendable el redactar manuales e incluir comentarios en los programas.

El uso de organigramas es útil sobre todo al principio de nuestra carrera como programadores, como ya hemos dicho. A medida que vaya aumentando nuestra experiencia pronto emplearemos otras herramientas más potentes y cómodas, aunque no siempre tan divertidas de utilizar. La principal de éstas es la que ya conocemos como pseudocódigo.

ELABORACIÓN Y DEPURACIÓN DE PROGRAMAS

El programa es, en resumen, el proceso de planificar una secuencia de instrucciones que se han de seguir para dar solución a un determinado problema. El concepto de programa almacenado en memoria fue concebido por John von Neumann en 1946 y, en esencia, tiene plena vigencia en nuestros días.

Para realizar un programa hay que proponer en primer lugar una solución a un problema, es decir, pensar en una estrategia para solucionarlo para, posteriormente, pasar al análisis en donde se estudiará de qué tipo se problema de trata y cómo se le va a dar solución, así como elegir el lenguaje en el que se va a programar o la herramienta de programación más adecuada para ello. Al menos en teoría, los pasos a seguir en el desarrollo de una aplicación deberían ser los siguientes:

- Análisis y propuesta de solución al problema: se trata de identificar el tipo de problema y el área a la que pertenece. Debe pensarse también en posibles soluciones y valorar los tiempos previstos correspondientes.
- Identificación de variables, constantes y actores involucrados en el problema: se identifican, ya que son los tres factores que pueden modificar el problema o alterar su curso.
- Planificación del programa: se elige el lenguaje de programación o plataforma de desarrollo a utilizar dependiendo de la solución y se crea el plan de trabajo correspondiente, que incluye la búsqueda bibliográfica de los algoritmos de resolución del problema y el cronograma de cada una de las etapas que hay que realizar hasta llegar a dicha solución.
- Algoritmo: desarrollo de la secuencia lógica de pasos para la resolución

del problema (en este paso se ven también involucrados los diagramas de flujo).

- Diagramas de flujo: se deben seguir los pasos del algoritmo comprobando que el problema se resuelva de manera apropiada.
- Desarrollo de las especificaciones: cuando se elige el lenguaje de programación o herramienta de desarrollo existen variables, constantes y otros elementos, que hay que declarar antes de comenzar el programa. Por otro lado, si se van a utilizar ecuaciones matemáticas o funciones, éstas requieren consignarse dentro del programa. Estas especificaciones involucran ciertas características del problema y determinadas funcionalidades del lenguaje seleccionado.
- Codificación (se requiere conocer la sintaxis del lenguaje) y depuración: conversión del algoritmo en programa escribiéndolo en un lenguaje de programación lo más eficientemente posible. Es evidente que nadie programa igual, ya que cada uno razona de forma diferente y el proceso para llegar a una solución depende del propio proceso de razonamiento.
- Ejecución y verificación de errores: cargar el programa en la memoria, ejecutarlo y valorar sus resultados, corrigiendo los errores hasta eliminarlos, de tal forma que se obtenga la solución al problema.
- Prueba final: se tiene la plena seguridad de que el problema quedó resuelto, puesto que se efectuaron todas las pruebas posibles para garantizar que el programa no falle al introducir ciertos valores o rangos de éstos.
- Documentación: mantenimiento y creación de los documentos descriptivos, como el manual del programador y del usuario.

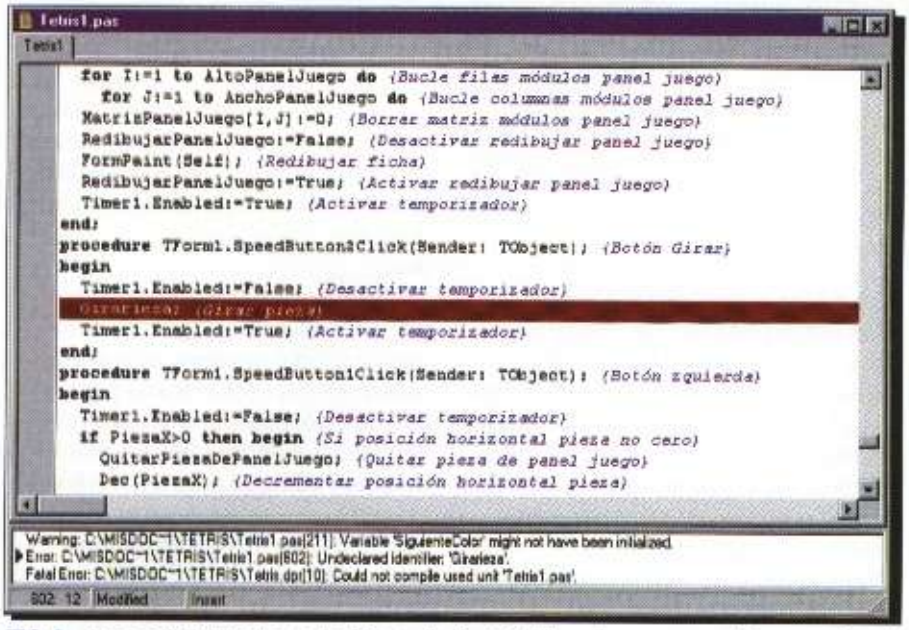

FIG 6. Los compiladores efectúan el diagnóstico de errores, informando al programador sobre su localización y su posible causa.

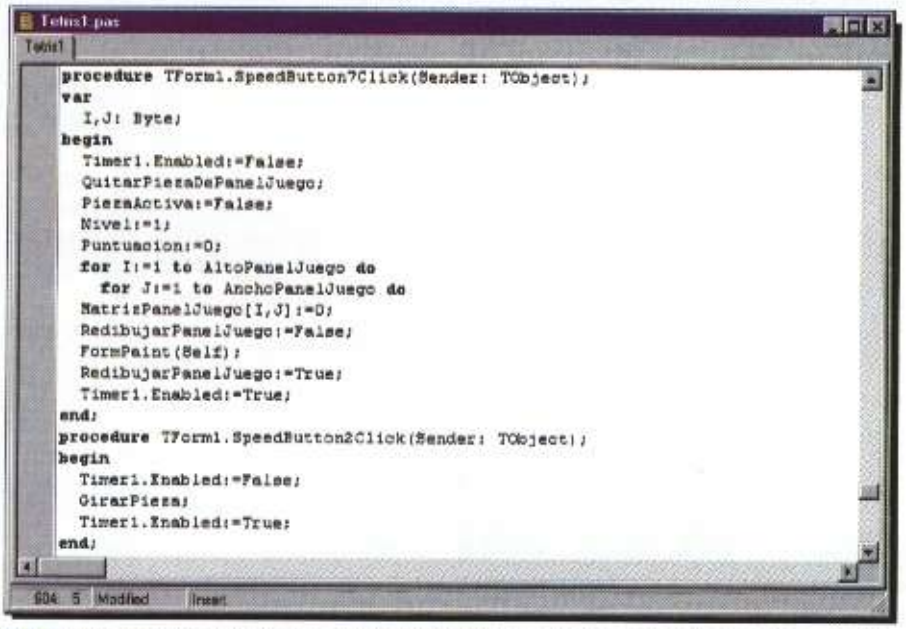

FIG 7. Los programas deben resultar fáciles de leer y comprender para facilitar posibles modificaciones futuras.

DEFINICIONES IMPORTANTES

Los diagramas y organigramas son las representaciones gráficas de un problema dado, para la definición, análisis, o solución, es decir, las representaciones gráficas de un algoritmo. Los algoritmos son los conjuntos de reglas o instrucciones que indican una secuencia lógica de operaciones, que proporcionan la respuesta a un tipo de problema determinado. Los estilos de programación son los distintos métodos que existen para mejorar la calidad de los programas. Los objetivos de un buen programa son estos:

- Debe ser seguro y eficaz.
- No debe plantear dificultades. Es preciso anticipar las posibles situaciones a las que se enfrentarán los usuarios, y garantizar que el programa sea capaz de resolverlas, es decir, que esté libre de errores.
- Debe estar bien documentado. La documentación puede adoptar dos formas: externa, que incluye diagramas de flujo, descripciones de los algoritmos, etc.., e interna, a modo de comentarios en el propio programa, y que van dirigidos exclusivamente a los programadores.
- Debe ser claro. Es muy importante que un programa resulte fácil de leer y de comprender. Ésta es la mejor garantía para asegurar su mantenimiento posterior por cualquier programador.

Soluciones prácticas (8)

Tareas programadas

Windows incluye una herramienta que a buen seguro ya le ha llamado la atención en más de una ocasión. Su nombre es Microsoft Tareas Programadas y permite automatizar fácilmente la ejecución de aplicaciones y procesos. Entre las herramientas que incluye, podemos destacar Microsoft ScanDisk.

No es de extrañar que Microsoft incluya en sus sistemas operativos Windows Microsoft Tareas Programadas, una herramienta específica para realizar automáticamente algunas tareas tales como, por ejemplo, ejecutar Microsoft ScanDisk, la herramienta de diagnóstico y solución de problemas relacionados con los discos. En esta unidad va a poder comprobar por sí mismo lo fácil que es crear tareas programadas a medida de cada necesidad, con la regularidad adecuada y en el momento más apropiado.

Muy probablemente, Microsoft Tareas Programadas ya se encuentre en ejecución, aunque puede que no se haya dado cuenta aún. Puede comprobarlo buscando en la bandeja del sistema (la parte derecha de la barra de tareas, junto al reloj) un icono que representa una ventana y que muestra un pequeño reloj en la parte inferior derecha. Si es así, para abrirlo tan sólo debe hacer doble clic sobre el icono. Si no está en ejecución por cualquier circunstancia (probablemente lo haya deshabilitado del inicio del sistema), podrá ejecutarlo abriendo el menú **Inicio**, accediendo a **Programas**, **Accesorios**, **Herramientas del sistema** y haciendo clic sobre **Tareas programadas**.

FIG. 1 Si Microsoft Tareas programadas no se encuentra en ejecución, puede abrirlo con Inicio | Programas | Accesorios | Herramientas del sistema | Tareas programadas.

La interfaz de esta herramienta es realmente sencilla. La ventana incluirá muy probablemente varios iconos, que representan tareas; su número y naturaleza variará de un sistema a otro (por ejemplo, si dispone de un antivirus, es probable que existan varios iconos relacionados con esta utilidad).

Vamos a centrarnos en la ejecución automática de una tarea concreta. El ejemplo que vamos a desarrollar va a

FIG. 2 La interfaz de Microsoft Tareas programadas es realmente sencilla.

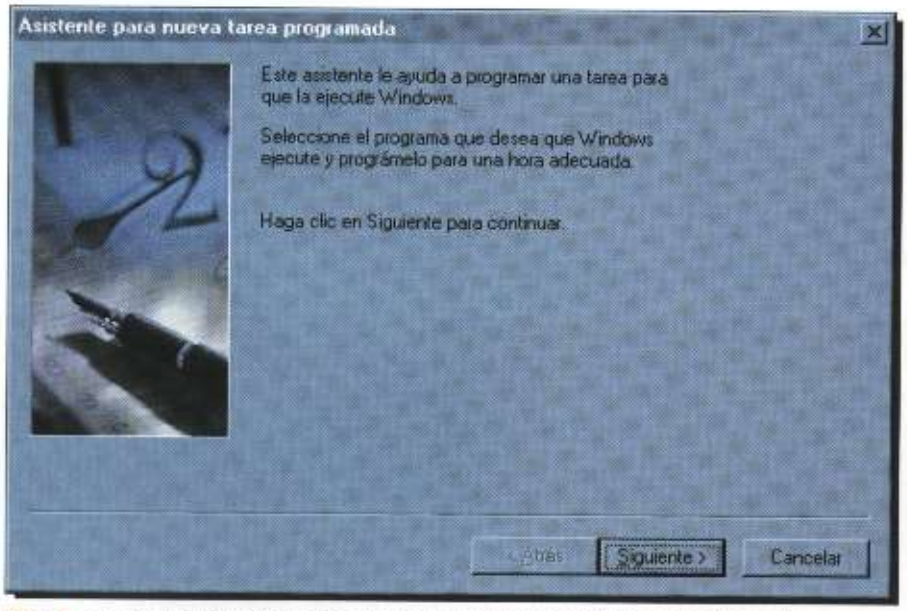

FIG. 3 La primera pantalla del Asistente para nueva tarea programada es meramente informativa.

ser muy sencillo, pero ilustrará el enorme potencial de Microsoft Tareas Programadas. Se va tratar ni más ni menos que de crear una tarea a medida capaz de ejecutar una aplicación; usaremos como ejemplo, Microsoft ScanDisk, pero en su caso puede usar el programa que más le convenga.

EN MARCHA

Seguro que ya ha imaginado cuál es el primer paso a dar en la creación de una tarea. Efectivamente, debe hacer doble clic sobre **Agregar tarea programada**. Verá que se muestra inmediatamente la primera pantalla del **Asistente para nueva tarea programada**. Las instrucciones que proporciona son bastante escuetas pero, en definitiva, crear una tarea es un proceso muy sencillo. Haga clic sobre **Siguiente** cuando haya terminado de leer.

En estos momentos estará contemplando una extensa lista de las aplicaciones instaladas en su sistema operativo. Desplace la lista hacia abajo hasta localizar Microsoft ScanDisk; el nombre del programa que se mostrará será ScanDisk. Haga clic sobre el nombre para seleccionarlo antes de pulsar sobre el botón **Siguiente**.

Si está usando otro programa distinto de Microsoft ScanDisk para crear el ejemplo de tarea y no aparece en la lista, haga clic sobre el botón **Examinar**. El cuadro de diálogo que se mostrará (**Seleccione Programa a organizar**) le permitirá escoger la aplicación navegando por sus unidades de disco. Cuando haya localizado el programa, un doble clic sobre su nombre hará que se seleccione automáticamente (o, tras seleccionarlo, puede hacer clic sobre el botón **Abrir**); en este caso no será necesario usar el botón **Siguiente**.

Tenga en cuenta que no sólo puede seleccionar archivos ejecutables, sino que podrá especificar un archivo cualquiera. En este caso, el programa ejecutará automáticamente la aplicación asociada a la extensión del archivo. De esta forma, si selecciona una hoja de cálculo con extensión XLS, Microsoft Tareas Programadas abrirá automáticamente Microsoft Excel para mostrar el archivo seleccionado.

PERIODICIDAD

La pantalla que se muestra ahora le permite definir dos parámetros: el primero es el nombre con que el que desea identificar la tarea y el segundo su periodicidad, es decir, cuándo se debe ejecutar.

En cuanto a la primera opción, puede escribir por ejemplo **Verificar la integridad de los discos duros** (en plural, ya que vamos a hacer que Microsoft ScanDisk compruebe el estado de cualquier unidad de disco duro del sistema).

En cuanto a la segunda, la periodicidad de la tarea, depende enormemente de la tarea en sí. Veamos algún ejemplo práctico de las interdependencias entre el tipo de tarea y la periodicidad usando las diferentes opciones que ofrece el programa:

- **Diariamente**. Use esta opción para tareas críticas. Si, por ejemplo, estamos hablando de una copia de seguridad de archivos vitales (datos bancarios, facturas, etc.) deberá indicar a Microsoft Tareas Programadas que ha de iniciar la tarea todos los días para realizar una copia de seguridad.

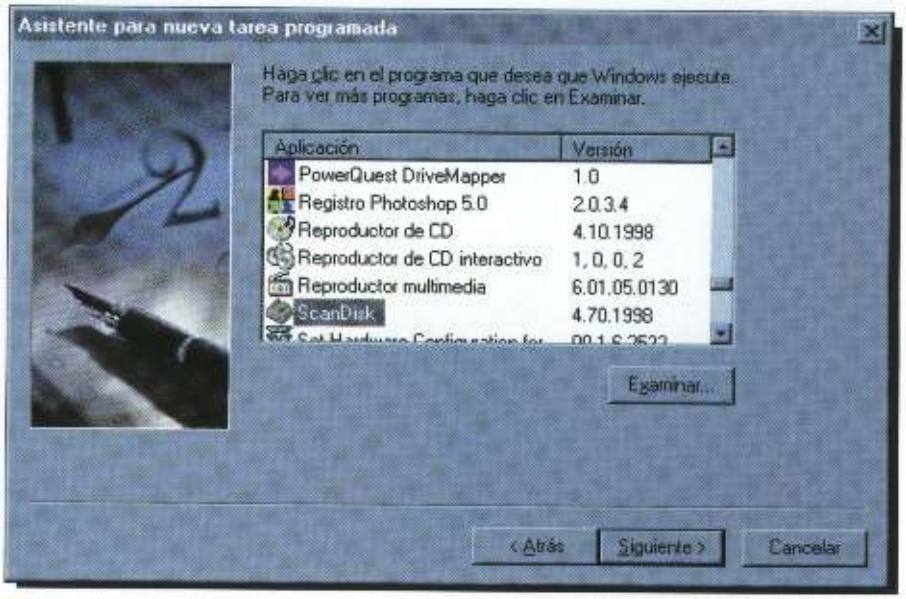

FIG. 4 La lista que se muestra contiene prácticamente la mayoría de las aplicaciones instaladas en Windows.

■ **Semanalmente**. Las tareas importantes, pero no vitales, se deben realizar una vez a la semana. Por ejemplo, si usa el ordenador de forma profesional, puede realizar cada semana una comprobación de los discos duros usando Microsoft ScanDisk, aunque sólo si usa las unidades de disco de forma intensiva. De igual manera, puede realizar copias de seguridad semanales de sus datos, pero no con el fin de mantener una copia de seguridad, sino como archivo.

■ **Mensualmente**. Son tareas que hay que realizar, pero que no son absolutamente vitales para sus intereses. Por ejemplo, en el caso de Microsoft ScanDisk, puede realizar la comprobación una vez al mes si es un usuario doméstico.

■ **Sólo una vez**. Es una tarea concreta que podrá realizar una vez de forma desasistida. Es una opción de poco uso.

■ **Cuando mi PC se inicie**. Use esta opción para realizar tareas tales como la ejecución desasistida de programas al inicio del sistema.

■ **Cuando inicie la sesión**. Es el mismo caso que el anterior, aunque aquí el "activador", es decir, el evento que activa la tarea, es el inicio de sesión en Windows.

En este ejemplo vamos a suponer que el ordenador se usa profesionalmente, aunque no se copian, borran y trasladan archivos con frecuencia, y vamos a optar por hacer una comprobación mensual de las unidades de disco. Para ello es necesario hacer clic sobre **Mensualmente** y, como de costumbre, pulsar sobre el botón **Siguiente**.

En la pantalla que está contemplando en estos momentos podrá definir con mayor precisión cuándo se debe realizar la tarea. La primera opción se refiere a la hora de inicio. Se define según el formato hora:minutos y para establecer una nueva hora basta con hacer clic sobre las horas, usar los botones de las flechas de la parte derecha, y luego hacer lo propio con los minutos. De igual forma, puede hacer clic sobre los números y teclear directamente la hora (o los minutos).

Escoger la hora no es una cuestión baladí en absoluto, ya que tiene que coincidir con algún momento en que esté seguro de que el ordenador estará encendido. Por ejemplo, si normalmente desayuna a las 10:30 de la mañana, puede escoger esta hora como inicio de la comprobación de las unidades.

En lo que se refiere a la elección del día, nuevamente se plantea la cuestión anterior: ¿qué día estará el ordenador encendido? Por ejemplo, si sabe positivamente que el ordenador siempre va a estar encendido los días 15 de cada mes, puede especificar este dato en **Día**. Puede ser, sin embargo, que el día 15 sea un día no laborable, por lo que no tiene la certeza absoluta de que el PC vaya a estar en funcionamiento.

Encontrará de mucha más utilidad la siguiente opción, ya que le permite especificar el nombre del día en que debe realizarse el proceso, así como su orden en el mes. En otras palabras, podrá indicar si lo desea que se ejecute Microsoft ScanDisk el segundo martes de cada mes. Hacerlo es tan sencillo como hacer clic sobre **El** y, por medio de los dos cuadros de lista desplegable siguientes, escoger respectivamente el número de orden del día en el mes (**primera**, **segundo**, **tercero**, **cuarto**, **Última**) y el día concreto (de **lunes** a **domingo**). A efectos de este ejemplo, configuraremos la tarea para que se ejecute el primer lunes de cada mes.

Tras el texto **de los meses** encontrará una serie de cuadros de verificación que le permitirán definir qué meses concretos se realizará la tarea. Por ejemplo, puede desmarcar el mes de agosto si va a disfrutar sus vacaciones en este mes. Para ello tan sólo tiene que hacer clic sobre la marca de verificación.

Pulse sobre **Siguiente** para avanzar en la configuración de la tarea.

Estará contemplando en estos momentos una pantalla en la que se le informa del éxito de la configuración de la tarea. También podrá verificar cuándo se realizará. Si algún detalle de la configuración no le satisface, no olvide que este asistente permite volver atrás usando el botón del mismo nombre.

Microsoft Tareas programadas dispone de más opciones encaminadas a

FIG. 5 Este cuadro de diálogo le permitirá escoger cualquier aplicación que desee ejecutar.

FIG. 6 Use siempre descripciones que permitan identificar inequívocamente cada tarea.

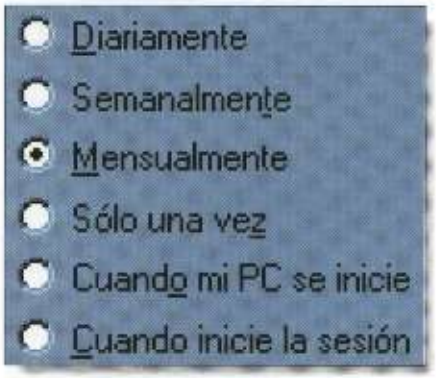

FIG. 7 Existen varias opciones entre las que elegir la periodicidad de la ejecución de la tarea.

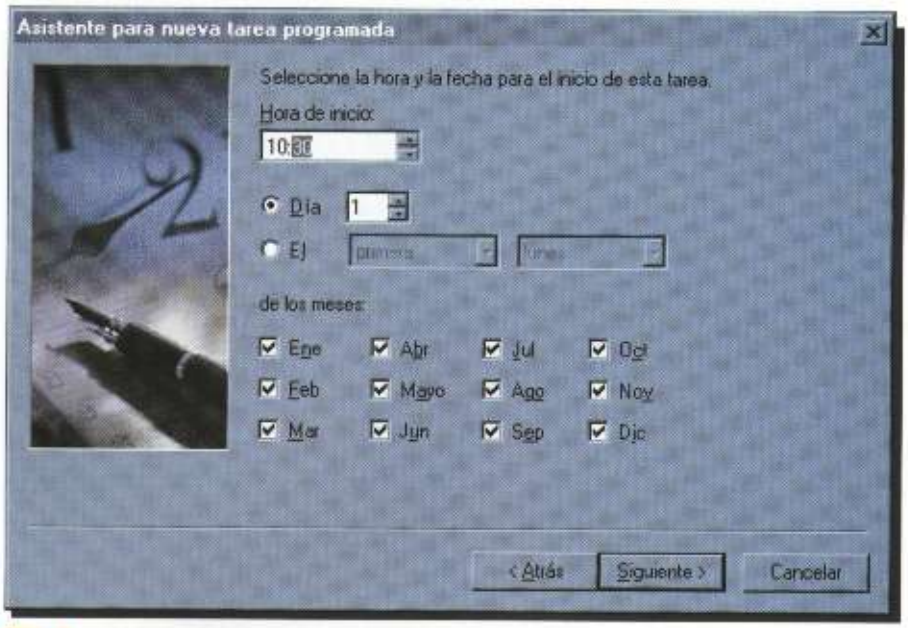

FIG. 8 Debe escoger la hora de comienzo de la tarea con mucho cuidado, ya que debe asegurarse de que el ordenador estará en funcionamiento.

hacer una configuración más detallada y precisa de la ejecución de cada tarea. Para mostrar inmediatamente el cuadro de diálogo de opciones avanzadas, debe marcar la casilla **Abrir propiedades avanzadas para esta tarea cuando haga clic en Finalizar** antes de pulsar sobre el citado botón. Hágalo.

PROPIEDADES AVANZADAS

Microsoft ScanDisk permite el uso de parámetros en la línea de comandos para configurar su ejecución (lea el cuadro *Parámetros de Microsoft* ScanDisk), algo que vamos a aprovechar para definir con precisión la tarea a realizar.

La opción **Ejecutar** debe reflejar **C:\WINDOWS\SCANDSKW.EXE /a /n**, con lo que conseguirá que Microsoft ScanDisk verifique la integridad de todas las unidades locales de disco duro, entrando y saliendo del programa automáticamente. Pulsando el botón **Configuración** podrá ver y cambiar la configuración de la aplicación elegida.

La carpeta de **Inicio** puede ser importante en el caso de algunas aplicaciones, por lo que conviene no olvidar configurarla en consecuencia.

Además, si desea escribir un comentario (por ejemplo, qué significan los parámetros que ha escrito en la línea de comandos), podrá hacerlo en el cuadro de inserción de texto **Comentarios**.

Fíjese por último en la parte inferior de la pantalla. Verá que se encuentra activada la opción **Habilitada (la tarea programada se ejecuta a la hora especificada)**. Como imaginará, podrá deshabilitar esta tarea desmarcando la opción.

Repase ahora la configuración de la tarea haciendo clic sobre la pestaña **Programación**. De esta ficha, el único elemento que no conocerá será el botón **Avanzada**. Pruebe a hacer clic sobre él y se mostrará un cuadro de diálogo con dos opciones: una que le permitirá indicar la fecha final de al tarea y otra en la que podrá especificar que la tarea se debe repetir.

En la última pestaña, **Configuración**, encontrará otras opciones muy interesantes de cara a especificar los parámetros de ejecución de la tarea. Uno especialmente útil en el caso de que el proceso pueda ir mal es **Detener la tarea si se ejecuta durante** X **horas** X **minutos**.

También es especialmente interesante el apartado **Tiempo de inactividad**, ya que le permite ejecutar la tarea en los tiempos en los que Microsoft Tareas programadas detecta que el ordenador no está en uso.

Además, si usa un portátil, le resultarán muy útiles las dos primeras funciones del apartado **Administración de energía**, llamadas **No iniciar la tarea si el equipo funciona con baterías** y **Detener la tarea si se inicia el modo batería**. Finalmente, haga clic sobre **Aceptar** y la tarea quedará perfectamente configurada. A partir de ese momento, Microsoft ScanDisk realizará sus comprobaciones en la fecha y hora señaladas sin necesidad de que usted intervenga en el proceso.

EN DEFINITIVA...

Microsoft Tareas Programadas será un auténtico regalo del cielo para muchos usuarios. Permite ejecutar automáticamente cualquier programa y ofrece un buen control sobre su ejecución. ¿Qué más se le puede pedir a una utilidad tan sencilla como ésta?

PARÁMETROS DE MICROSOFT SCANDISK

Aunque es una aplicación Windows 100%, Microsoft ScanDisk también puede aceptar parámetros de línea de comandos, tales como lo hacían muchos de los programas diseñados para MS-DOS. Esta es una lista de los comandos que se pueden emplear para ejecutar Microsoft ScanDisk:

- **Unidad: : escriba la letra de la unidad seguida de dos puntos para analizarla.**
- **/a: con este parámetro indicará a Microsoft ScanDisk que debe comprobar todos los discos duros locales.**
- **/n: inicia Microsoft ScanDisk y sale del programa automáticamente.**
- **/p: impide que el programa repare los errores que encuentre en el proceso.**

Por ejemplo, para indicar que Microsoft ScanDisk debe comprobar la unidad D: evitando que repare los errores encontrados, la línea de comando quedará como sigue:

```
c:\windows\scandskw.exe d: /p
```

Borland Delphi (2)

Proyectos

Los proyectos son la piedra angular sobre la cual se asientan los desarrollos efectuados en Borland Delphi, estando formados, a su vez, por una serie de archivos de diferentes tipos, generados automáticamente en las distintas fases de la evolución del desarrollo de una aplicación. En esta unidad escribiremos los mecanismos para realizar esta automatización.

El entorno gráfico de Borland Delphi se encarga automáticamente de la gestión de los distintos archivos que componen el proyecto de manera transparente al usuario, de forma que en raras ocasiones se deberá recurrir a la edición manual de los mismos. No obstante, conviene tener una idea de la razón de ser de estos, entre otras cosas, porque ello puede ayudar a comprender mejor el funcionamiento general del entorno. Nos ocuparemos también de las distintas modalidades a la hora de compartir componentes, así como de las opciones de configuración del entorno de trabajo, compilación y generación de proyectos terminados.

ESTRUCTURA DE LOS PROYECTOS

En Borland Delphi se llama proyecto al conjunto de archivos que hacen posible el desarrollo y ejecución de una aplicación, algunos de los cuales se generan automáticamente durante la fase de desarrollo, mientras que otros en la de compilación. La gestión de estos archivos la efectúa automáticamente el propio entorno gráfico de Borland Delphi, por lo que no se tiene que recurrir a su edición manual, que en la mayoría de los casos podría efectuarse mediante cualquier procesador de textos. La espina dorsal del conjunto es el archivo de proyecto, en el cual se consignan las referencias a las fichas y módulos que integran el proyecto, por lo cual representa un papel fundamental en el momento de la compilación de la aplicación. Los archivos de proyecto tienen por omisión la extensión DPR y están escritos en lenguaje Object Pascal, de manera que tras la compilación dan lugar a un nuevo archivo ejecutable de extensión EXE o biblioteca de vínculos dinámicos DLL, según se especifique.

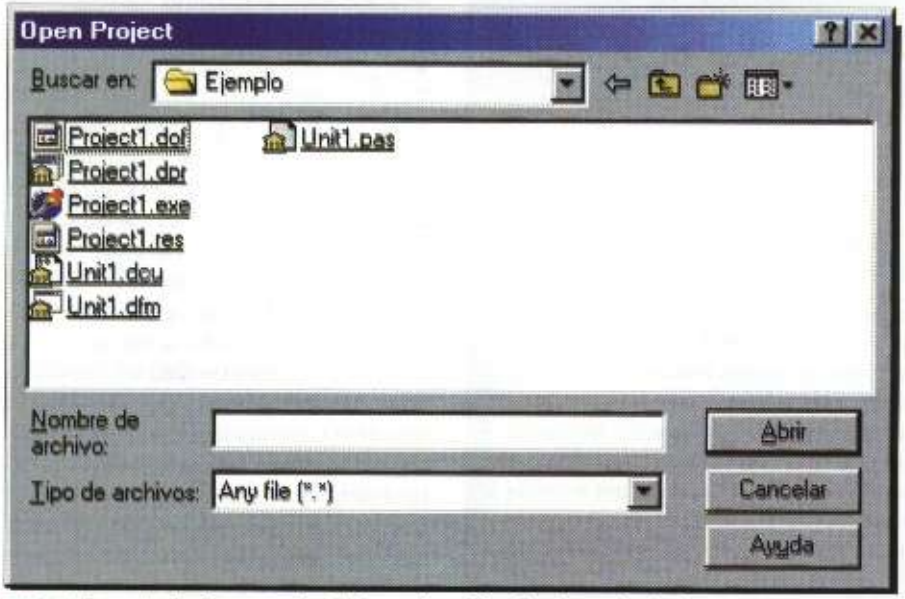

FIG. 1 Aspecto de la carpeta de la aplicación de ejemplo de la entrega anterior con todos los archivos vinculados a ese proyecto.

```
program Project1;
{Declaración del identificador
de proyecto}

uses {Indicación de los
modulos utilizados}
  Forms, {Inclusión de módulos
distintos de fichas}
  Unit1 in 'Unit1.pas' {Form1};
{inclusión de fichas}
{$R *.RES} {Vinculación con el
archivo de recursos}

begin {Inicio del bloque de
programa principal}
```

```
Application.Initialize;
{Inicialización}
Application.CreateForm(TForm1,
Form1); {Creación de la ficha
principal}
Application.Run; {Ejecución
de la aplicación}
end. {Fin del bloque de
programa principal}
```

ARCHIVOS DE PROYECTO

El Project Manager (Gestor de Proyectos) es la herramienta principal de edición de este archivo, por lo que no requiere edición manual. No obstante, pueden utilizarse el propio entorno gráfico o cualquier procesador de textos externo para inspeccionar su contenido y conocer los módulos, ficha principal y secundarias vinculados a un proyecto. La secuencia de menús **View | Project Source** (Ver | Fuente de proyecto) presenta en pantalla la ventana de código. Alternativamente, es posible también obtener los mismos resultados desde el **Project Manager**, que se muestra mediante la secuencia de menús **View | Project Manager** (Ver | Gestor de Proyectos), en este caso mediante la opción **View Project Source** del menú contextual del botón derecho del ratón. Las cadenas encerradas entre llaves son comentarios y, por lo tanto, ignoradas por el compilador, siendo su cometido documentar el código. El nombre de proyecto por omisión es *Project1*, que coincide con el propio nombre del archivo de proyecto. La cláusula **program** (programa) indica al compilador que se desea obtener un programa ejecutable (EXE). Una posibilidad alternativa es utilizar **library** (biblioteca), considerándose en este caso que lo que se desea obtener es una biblioteca de vínculos dinámicos (DLL), o bloque complementario de otro programa ejecutable de extensión EXE. La cláusula **Uses** (utiliza) relaciona los módulos que se deben vincular al proyecto, haciendo referencia **Forms** a las fichas de trabajo, de las cuales **Form1** es el nombre considerado por omisión para la principal.La referencia a **Unit1** indica el nombre del módulo, que corresponde al archivo de código Object Pascal **Unit1.pas** de la ficha **Form1**, según figura en el comentario. Los nombres de las fichas y sus correspondientes archivos de código deben ajustarse a la normativa descrita, ocasionándose en caso contrario un error general de compilación. Así, cada archivo de tipo **Form** debe emparejarse a uno de tipo **Unit**, pudiendo existir tantas fichas y archivos de código como sea necesario. La palabra reservada **in** (en) sólo se muestra en módulos de código asociados a fichas, como es el caso del ejemplo, y establece la relación con la ficha cuyo nombre se muestra en el comentario, según el valor de la propiedad **Name** (Nombre) de la ficha de trabajo. Los nombres de ficha no admiten espacios en blanco ni caracteres

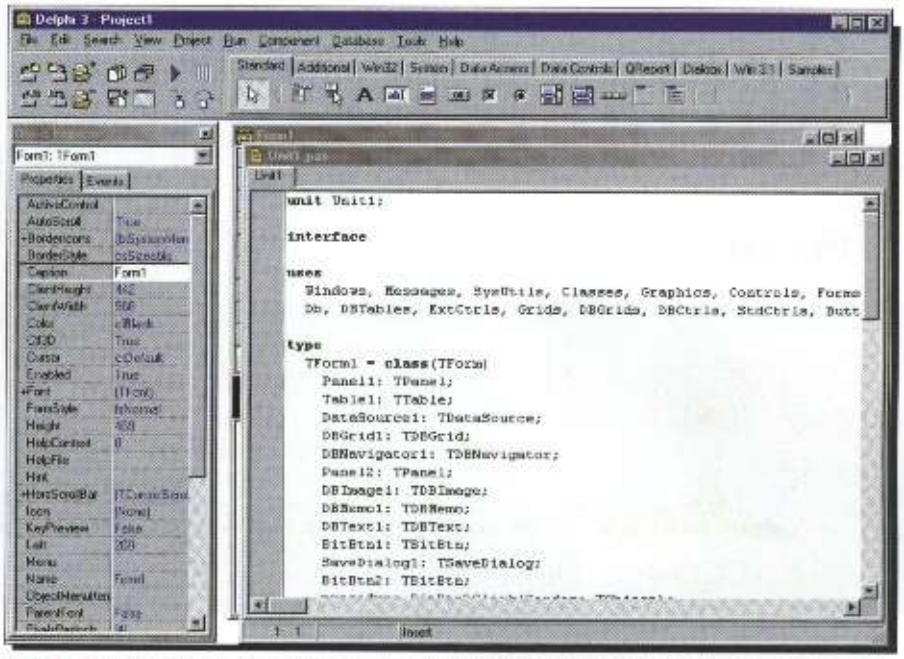

FIG. 2 Contenido de la ventana de código Object Pascal que se muestra mediante la secuencia de menús View | Project Source.

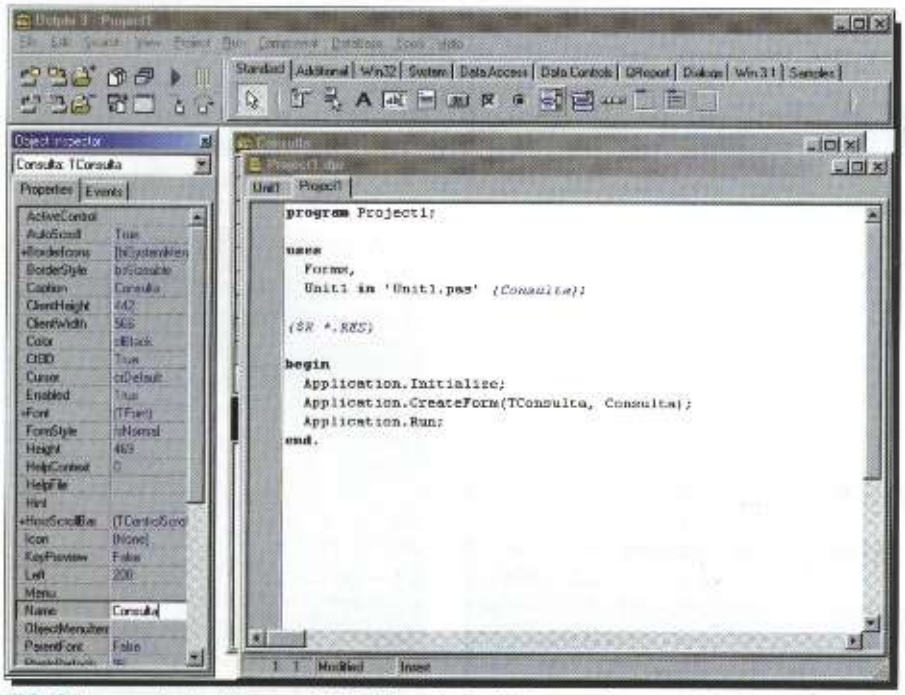

FIG. 3 La propiedad Name de la ficha de trabajo permite cambiar su denominación, modificándose automáticamente el código asociado.

FIG. 4 Puede modificarse directamente el contenido de los archivos de descripción de fichas mediante la opción del menú contextual.

especiales. La directiva **$R**, como ya se ha comentado anteriormente, vincula al proyecto un archivo binario de recursos de extensión RES, en el cual se almacena información como, por ejemplo, el icono del proyecto.

El bloque principal de código de programa es el encerrado entre las cláusulas **begin** (inicio) y **end** (fin). La cláusula **Application.Iniciatilize** (inicializar aplicación) prepara la ejecución. **Application.CreateForm** (crear ficha de aplicación) gestiona la creación de las fichas, siguiendo el orden de definición en el proyecto, que se puede alterar si se desea mediante el cuadro de diálogo que se presenta en pantalla mediante la secuencia de menús **Project | Options** (Proyecto | Opciones). Por último, **Application.Run** (ejecutar aplicación) inicia la aplicación.

ARCHIVOS DE FICHA

Las fichas, como hemos podido comprobar en el ejemplo de la entrega anterior, son un componente básico de los proyectos y sirven como soporte para toda clase de objetos, gestionándose usualmente desde el entorno gráfico integrado. Borland Delphi almacena en disco la información necesaria en archivos binarios de extensión DFM. Estos archivos pueden editarse directamente desde el entorno gráfico mediante el editor de código integrado, con sólo elegir la opción **View as Text** (Ver como Texto) del menú contextual de la ficha, con lo cual pasa a mostrare en la ventana del editor de código el contenido del archivo DFM. A su vez, desde el editor de código puede pasarse al modo de edición de ficha mediante la opción **View as Form** (Ver como Ficha) del menú contextual. Como ya se ha comentado anteriormente, cada ficha tiene asociado un módulo de código, necesario para los manejadores de sucesos, que se almacena en disco con la extensión PAS.

ARCHIVOS DE MÓDULO

Para facilitar la estructuración y reutilización de código, las aplicaciones pueden dividirse en módulos compilables independientemente, que en la mayoría de las ocasiones corresponden a fichas, con su correspondiente descripción de manejadores de sucesos. No obstante, es posible también guardar en disco módulos independientes para alojar DLL, funciones, procedimientos y componentes propios. Los proyectos nuevos, por omisión, están compuestos por un módulo **Unit1** y una ficha **Form1**, que dan origen a los archivos en disco **Unit1.pas** y **Unit1.dfm**. Tras la compilación se genera un nuevo archivo binario de trabajo en disco de extensión DCU por módulo, cuya misión es acelerar los procesos de compilación y enlace de módulos. Alternativamente es posible también generar archivos estándar de objetos de Intel (extensión OBJ). Esto ofrece compatibilidad con otras aplicaciones, pero ralentiza el proceso de compilación, por lo cual no es la opción recomendable a no ser que la compatibilidad sea la cuestión de mayor peso frente a la velocidad. En cualquier caso, el único archivo requerido para distribuir la aplicación es el EXE.

- **Fichas.**

Los módulos correspondientes a fichas, los más habituales, adoptan nada más crearse un formato como el reseñado, numerándose correlativamente de 1 en adelante al crearse cada nueva instancia de objeto. Dentro del bloque **type** (tipo) se define la ficha como una clase de objeto, descendiente de la clase genérica de fichas, razón por la cual heredan automáticamente todas las

DENOMINACIÓN DE ARCHIVOS

Para mejorar la legibilidad del código conviene asignar nombres descriptivos al archivo de proyecto y los de módulos, lo cual se puede hacer desde la primera vez que se almacena en disco un proyecto. Es preciso ajustarse, sin embargo, a las normas generales de denominación de archivos, no pudiendo utilizarse espacios en blanco ni caracteres especiales. Debe tenerse en cuenta también que, aunque los nombres pueden superar los 8 caracteres, a efectos de compilación se consideran los 8 primeros por motivos de compatibilidad. En cuanto a carpetas se refiere, lo más conveniente sería reservar una carpeta para los archivos de cada proyecto, aunque los componentes pueden residir sin restricciones en los trayectos de disco que se considere oportuno.

propiedades de la clase **TForm**. Por otro lado, dentro del apartado **var** (variable) se declara la instancia de objeto ficha como perteneciente a la clase **TForm1**. Finalmente, **$R** es la directiva obligatoria de vinculación para el compilador.

```
unit Unit1; {Identificación de
módulo}
interface
uses {Indicación de
componentes utilizados}
 Windows, Messages, SysUtils,
Classes, Graphics, Controls,
Forms, Dialogs;
type
 TForm1 = class(TForm)
{Declaración de clase}
 private
   {Declaraciones privadas}
 public
   {Declaraciones públicas}
 end;
var
 Form1: TForm1; {Declaración
de instancia}
implementation
{$R *.DFM} {Directiva de
compilador para vinculación
del archivo de ficha}
end.
```

- **Procedimientos.**

Pueden crearse también módulos no asociados a fichas, como por ejemplo, para funciones y procedimientos, que queden a disposición de varios proyectos. Para crear un archivo de módulo no asociado se debe seguir la secuencia de menús **File | New** (Archivo | Nuevo), elegir **Unit** (Módulo) en el cuadro de diálogo de elementos y pulsar **OK** (Aceptar).

ARCHIVOS DE RECURSOS

Los archivos binarios de recursos utilizados por Borland Delphi son estándar de Windows y por lo tanto contienen elementos como iconos, cursores, cadenas de texto, etc, creándose por omisión para cada aplicación uno con el nombre genérico de proyecto.

MANIPULACIÓN DE PROYECTOS

Para gestionar los archivos de un proyecto puede optarse tanto por el **Project Manager** como por el repertorio de opciones del menú desplegable

FIG. 5 El archivo de extensión EXE es el único que es necesario distribuir a los usuarios finales de una aplicación.

File, siendo preferible el primer método, puesto que sus comandos están accesibles tanto desde la barra rápida como desde el menú contextual. Con un proyecto abierto se accede al gestor mediante la secuencia de menús **View | Project Manager**. La barra de botones de su ventana cuenta con las opciones **Add** (Añadir), **Remove** (Eliminar), **Unit** (Ver Módulo), **Form** (Ver Ficha), **Options** (Modificar opciones del proyecto) y **Update** (Actualizar cambios). En el área de trabajo de la ventana se muestran las columnas de identificación de módulos, fichas y trayectos de disco, así como en la barra de estado información estadística sobre el proyecto, como trayecto de disco, número de módulos y fichas. Se accede a los módulos al hacer doble clic sobre la columna **Unit** o un clic para seleccionar y pulsar **Intro**, mientras que a las fichas mediante doble clic o la combinación de teclas **Mayús + Intro**.

A la hora de añadir módulos y fichas compartidos desarrollados por el usuario a un proyecto debe tenerse en cuenta su ubicación física en disco, puesto que el compilador considera los componentes como si pertenecieran al proyecto en curso con independencia de su localización. Por otro lado, para eliminar componentes debe recurrirse a las opciones del programa en lugar de a los procedimientos estándar de Windows, para garantizar así la consistencia de la información de proyecto gestionada automáticamente por el entorno de desarrollo de Borland Delphi.

OTROS ARCHIVOS

Opcionalmente pueden asociarse a los proyectos archivos denominados de paquetes, o DLL, con extensión DPL para la versión ejecutable y DPK para la fuente.
Además, como ya se ha comentado anteriormente, durante el proceso de compilación se crean archivos DOF de opciones de proyecto, cuyo contenido se gestiona habitualmente desde el cuadro de diálogo Project Options (Opciones de Proyecto). Se crean también archivos DSK con información para distribuir el área de trabajo desplegando las ventanas en posiciones y con tamaños determinados.

El menú contextual de la ventana del **Project Manager** aglutina las funciones más habituales, al tiempo que propone una serie de atajos de teclado. Algunas de estas opciones son accesibles también como botones de la barra de botones de la ventana.

DESPLAZAMIENTO ENTRE COMPONENTES

Cuando se trabaja con proyectos es necesario alternar frecuentemente entre los distintos componentes, como módulos, fichas, o las ventanas del gestor de proyectos o la paleta de alineación, por ejemplo, para lo cual puede recurrirse tanto a secuencias de menús como a teclas de función y combinaciones de teclas.

■ **Conmutación entre fichas y módulos.**

Se puede pulsar **F12** o el botón **Toggle Form/Unit** de la **Speedbar** (Barra rápida) del margen izquierdo de la ventana de la aplicación. También se puede seguir la secuencia de menús **View | Toggle Form / Unit** (Ver | Alternar Ficha / Módulo) o hacer doble clic en la columna **Unit** desde la ventana del Project Manager.

■ **Pasar un elemento a primer plano.**

Se puede hacer clic sobre la barra de título de la ventana, o utilizar la combinación de teclas **Alt + 0** para elegir dentro de la lista de ventanas al hacer doble clic o seleccionar una y pulsar **OK** (Aceptar). Es posible también seguir la secuencia de menús **View | Window List** (Ver | Lista de Ventanas). Para seleccionar rápidamente una ficha o módulo determinado se pueden seguir las secuencias de menús **View | Forms** (Ver | Fichas) o **View | Units** (Ver | Módulos), o también utilizar las combinaciones de teclas **Mayús + F12** y **Control + F12**, respectivamente.

ALMACENAMIENTO DE PROYECTOS Y ARCHIVOS

Además de la posibilidad de guardar en cualquier momento el proyecto completo en edición, Borland Delphi permite también hacerlo con un nombre distinto, en otra carpeta, o incluso almacenar archivos individuales como plantillas en el **Object Repository** (Almacén de Objetos).

■ **Guardar el proyecto.**

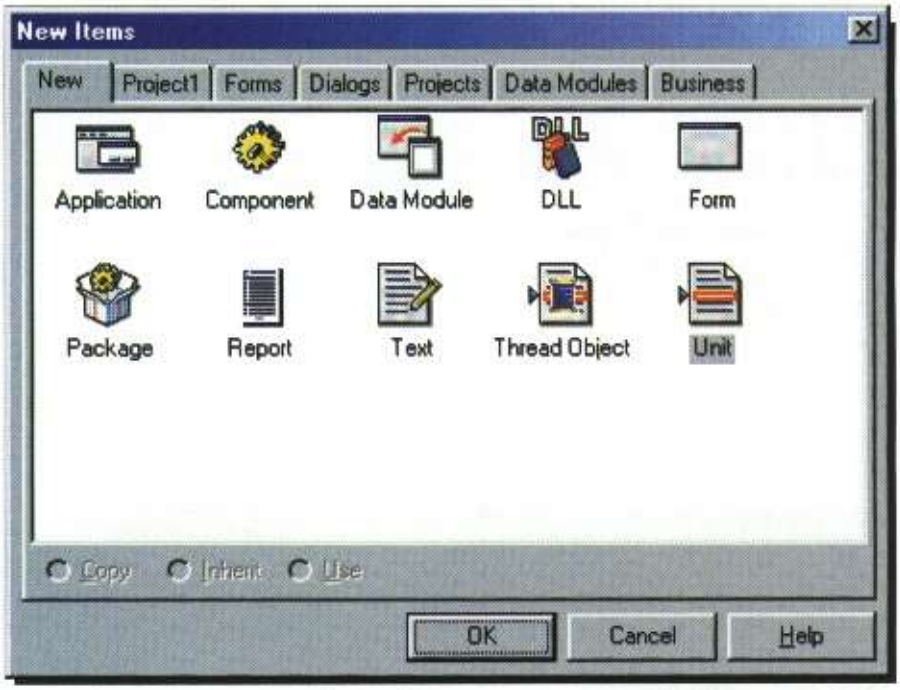

FIG. 6 La secuencia de menús File | New permite crear módulos no asociados a fichas para funciones y procedimientos.

El botón de la Speedbar **Save all** o la secuencia de menús equivalente **File | Save All** (Archivo | Guardar Todo) almacenan en disco el proyecto en edición, manteniendo los nombres y carpetas habituales. Es posible también

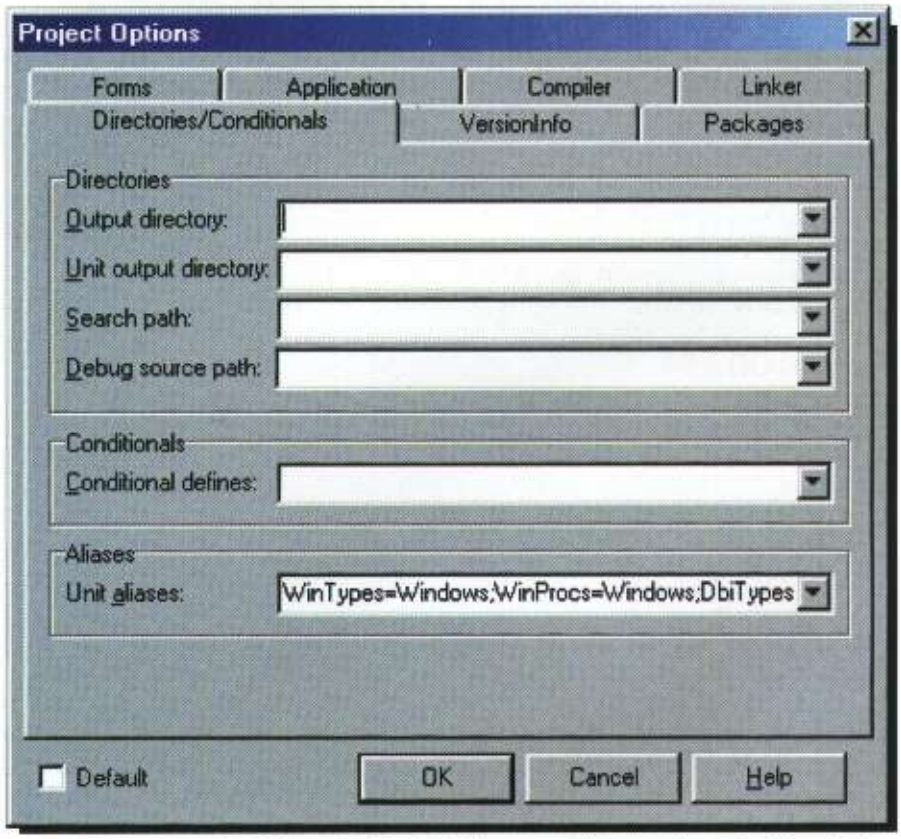

FIG. 7 El cuadro de diálogo Project Options condiciona el contenido de los archivos de opciones de proyecto DOF.

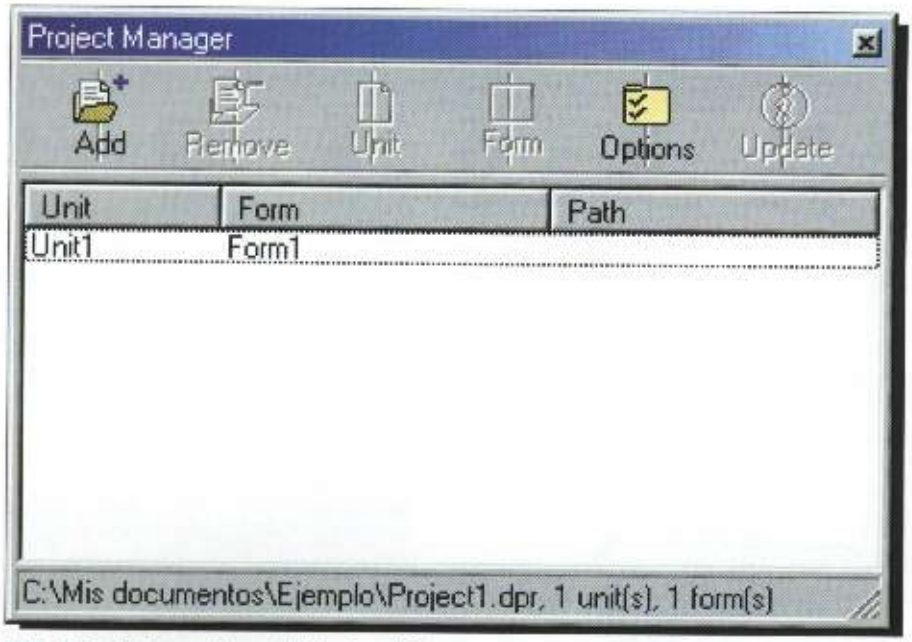

FIG. 8 Desde la ventana del Project Manager se pueden gestionar cómodamente los archivos de proyecto.

hacer lo mismo desde el **Project Manager** eligiendo la opción **Save Project** (Guardar Proyecto) del menú contextual del botón derecho del ratón. Lógicamente, en el caso de que se almacene en disco el proyecto por primera vez, o bien se hayan incorporado nuevos componentes al mismo en la sesión de trabajo, se muestran los oportunos cuadros de diálogo del tipo Guardar como para nombrar los archivos y asignarles ruta de almacenamiento en disco.

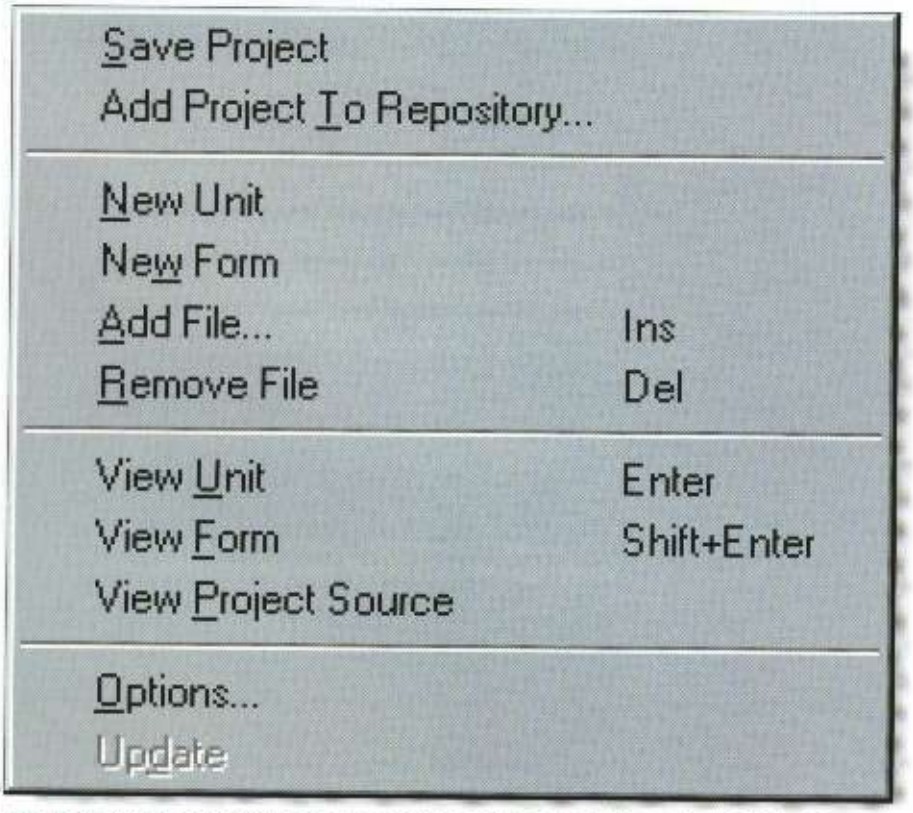

FIG. 9 El menú contextual de la ventana del Project Manager aglutina las opciones más habituales.

- **Guardar copia del proyecto.**

Si se desea se puede almacenar una copia del conjunto de archivos en un trayecto diferente. Para ello se debe seguir la secuencia de menús **File | Save Project As** (Archivo | Guardar Proyecto Como), indicando los nuevos trayectos y nombres de archivos a petición del programa, pulsando **Guardar** para finalizar. Puede recurrirse al **Project Manager** para comprobar que se obtiene de este modo una copia integral del proyecto en desarrollo con el nombre y en la ubicación indicados. Para garantizar la coherencia de los conjuntos de archivos con los registros de Borland Delphi, debe recurrirse siempre a las herramientas del programa en lugar de a los procedimientos estándar de Windows para la copia y traslado de archivos y carpetas.

- **Guardar archivos.**

Es posible almacenar en disco archivos individuales que contengan código, por ejemplo, seleccionando su ventana para convertirla en activa y pulsando el botón **File save** de la **Speedbar**, o siguiendo la secuencia de menús **File | Save As** (Archivo | Guardar como).

- **Copias de seguridad de proyecto.**

El procedimiento a seguir para obtener una copia de respaldo de un proyecto depende de su estructura de archivos. En el caso más sencillo, de que todos los componentes del proyecto se almacenen en una carpeta única, basta con hacer copia de la misma. En caso contrario debe tenerse en cuenta que Borland Delphi registra los trayectos y nombres de todos los archivos componentes del proyecto, aunque la propia carpeta de proyecto no se encuentra registrada en el archivo de proyecto. Una buena forma de verificar que se efectúa correctamente la copia de seguridad es comprobar que se compila el proyecto sin incidencias en la nueva ubicación y se obtiene un ejecutable completamente operativo.

OBJETOS COMPARTIDOS

Como ya se ha comentado anteriormente, el **Object Repository** (Almacén de objetos) es una herramienta que ofrece muchas posibilidades a la hora de compartir objetos entre aplicaciones, como fichas y módulos, proporcionan-

do también plantillas completas de proyecto que sirvan como base para los desarrollos del usuario. El repertorio inicial puede ampliarse en adelante con las aportaciones del usuario, que puede incluso añadir proyectos completos que sirvan como plantilla para el desarrollo de otras aplicaciones similares en el futuro. Se cuenta también con un conjunto de asistentes para la creación de fichas o proyectos completos, que guían al usuario paso a paso. El propio usuario, con los debidos conocimientos, puede también crear nuevos asistentes como ayuda para el desarrollo de fichas y proyectos. El almacén de objetos reside físicamente un archivo de texto, que contiene las referencias a los elementos que aparecen en los cuadros de diálogo **Object Repository** y **New Items** (Elementos Nuevos), localizado en: ***C:\Delphi3m\BIN\DELPHI32.DRO***.

■ **Compartir dentro del proyecto.**

Para compartir elementos dentro de un proyecto no es preciso recurrir al **Object Repository**. En su lugar se puede seguir la secuencia de menús **File | New** (Archivo | Nuevo) para mostrar el cuadro de diálogo **New Items** y seleccionar la pestaña con el nombre del proyecto, mostrándose de este modo todos los componentes del proyecto en forma de iconos. Así se pueden derivar unos elementos de otros y personalizarlos como se desee.

■ **Compartir dentro del grupo de trabajo.**

El procedimiento a seguir en este caso es crear un almacén de objetos compartido, que resida en un trayecto de disco disponible para todos los componentes del grupo de trabajo. Así, al añadir nuevos elementos a éste se actualizará automáticamente el archivo **DELPHI32.DRO** residente en la carpeta, que contiene los punteros a los objetos que se desea compartir. Obviamente, todos los componentes del grupo deben indicar la misma carpeta como localización del almacén de objetos compartido.

Para definir una carpeta compartida se debe seguir la secuencia de menús **Tools | Environment Options** (Herramientas | Opciones de Entorno). A continuación, dentro del apartado **Shared Repository** de la ficha

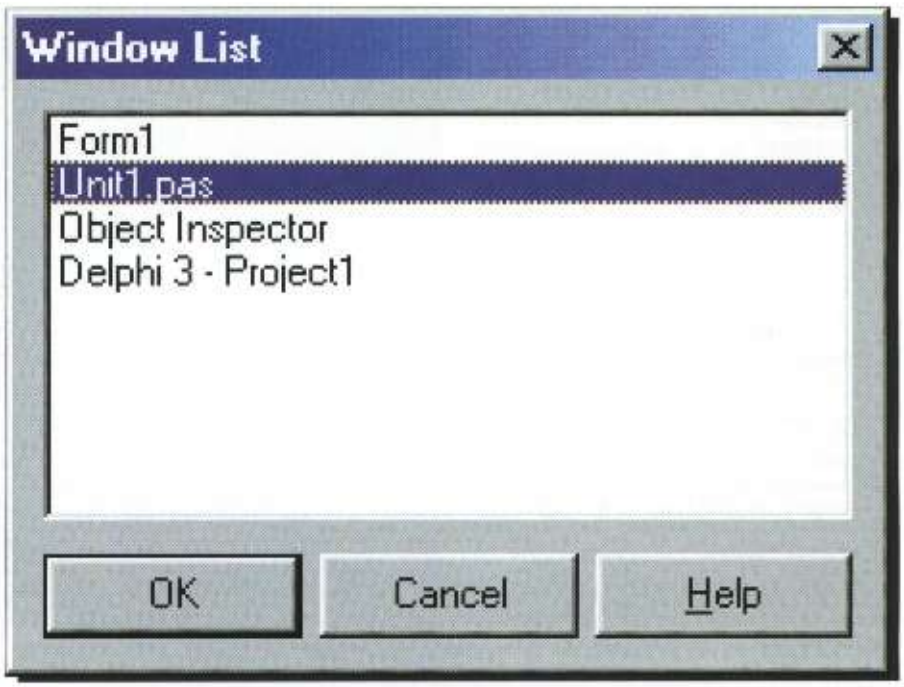

FIG. 10 La Window List permite elegir directamente el elemento que se desea llevar a primer plano del área de trabajo.

Preferences (Preferencias), debe indicarse el trayecto completo de la carpeta, disponiéndose de un botón **Browse** (Mostrar) para inspeccionar cómodamente las unidades de disco en su búsqueda.

■ **Modalidades de adición de elementos.**

Los objetos provenientes del **Object Repository** pueden añadirse a los proyectos en tres modalidades distintas: **Copy** (Copia), **Inherit** (Herencia) y **Use** (Utilización), representadas por casillas de selección en el margen inferior de la ventana **New Items**.

• **Copy.**

Se crea una copia exacta del elemento y se añade al proyecto en edición, quedando desvinculados por completo ambos, de manera que las posi-

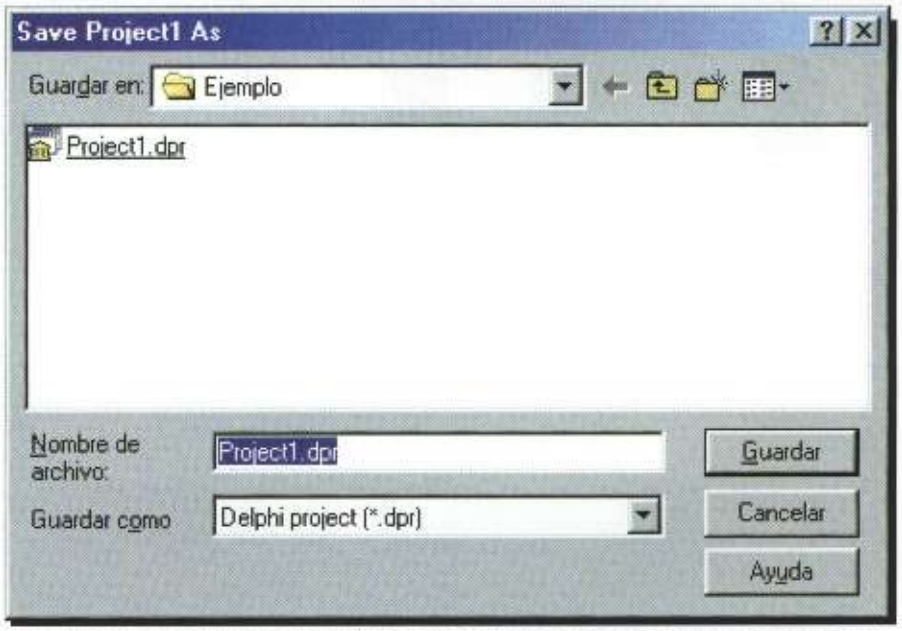

FIG. 11 Este es el cuadro de diálogo que se presenta en pantalla para almacenar en disco el proyecto con nombre o ubicación diferentes.

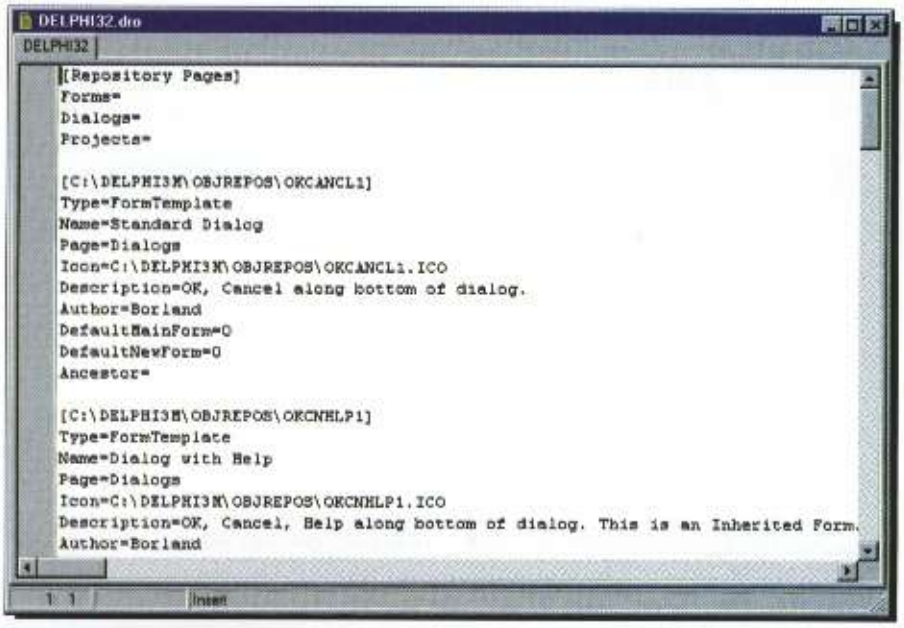

FIG. 12 Este es el aspecto de la ventana del editor de código mostrando el archivo DELPHI32.DRO del Object Repository.

bles modificaciones efectuadas en el proyecto en edición no tienen efecto sobre el elemento del almacén de objetos. Esta es la única opción disponible para plantillas de proyecto.

- **Inherit.**

En este caso se deriva una nueva clase a partir del objeto elegido y se añade al proyecto. En este modo existe vinculación con el objeto del **Object Repository**, de manera que las modificaciones posteriores que puedan efectuarse en la clase afectarán por cuestión de herencia al objeto derivado añadido al proyecto en edición, es decir, que se define un vínculo con el elemento antecesor perteneciente al almacén de objetos.

Obviamente, las posibles modificaciones introducidas en la clase derivada del proyecto en edición no tienen efecto sobre el correspondiente objeto antecesor del **Object Repository**. Esta opción puede aplicarse a fichas, módulos y cuadros de diálogo, aunque no a plantillas, siendo la única posibilidad para reutilizar elementos pertenecientes a un mismo proyecto.

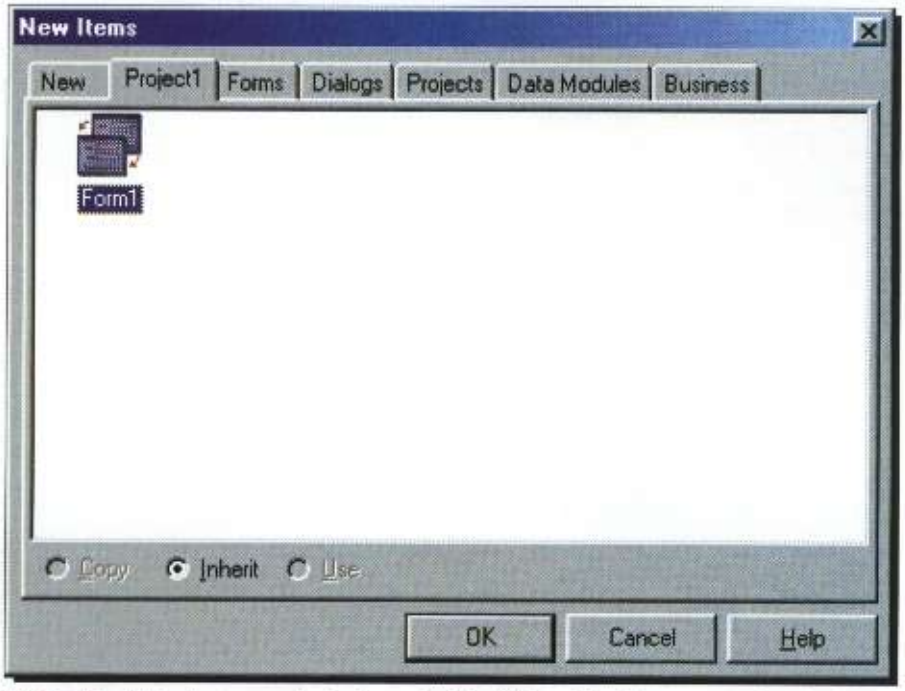

FIG. 13 La ficha de proyecto del cuadro de diálogo New Items permite derivar y personalizar elementos como se desee.

- **Use.**

En este caso se añade directamente al proyecto en edición el propio elemento del **Object Repository**. Las posibles modificaciones introducidas en adelante en el objeto afectan tanto al proyecto en edición como a todos los demás que incluyan el mismo. Aunque de gran potencia, se trata de una opción que entraña cierto peligro, por lo cual se debe tener especial cuidado en su uso. Es aplicable a fichas, módulos y cuadros de diálogo.

■ **Empleo de plantillas.**

Las plantillas de proyecto permiten crear una estructura básica como punto de partida para un proyecto nuevo. Para crear un nuevo proyecto a partir de una plantilla basta con seguir la secuencia de menús **File | New** y seleccionar la pestaña **Projects** (Proyectos). Tras elegir un modelo al hacer doble clic sobre el mismo, o bien seleccionarlo y hacer clic sobre el botón **OK**, se muestra el cuadro de diálogo **Select Directory**, dentro del cual debe elegirse la carpeta de destino, que se crea automáticamente de no existir.

Las posibles modificaciones introducidas en adelante en el nuevo proyecto no afectan a la plantilla empleada como punto de partida.

■ **Incorporaciones al Object Repository.**

Como ya se ha comentado anteriormente, es posible también añadir elementos creados por el usuario al almacén de objetos, como fichas, módulos e incluso proyectos completos. Para añadir un proyecto completo se debe abrir el mismo para seguir la secuencia de menús **Project | Add To Repository** (Proyecto | Añadir Al Almacén). Para añadir sólo una ficha o módulo, en su lugar puede recurrirse a la opción **Add To Repository** del menú contextual. En el cuadro de diálogo que se presenta en pantalla pueden indicarse título, descripción, tipo y autor, disponiéndose de un botón **Browse** para elegir un icono

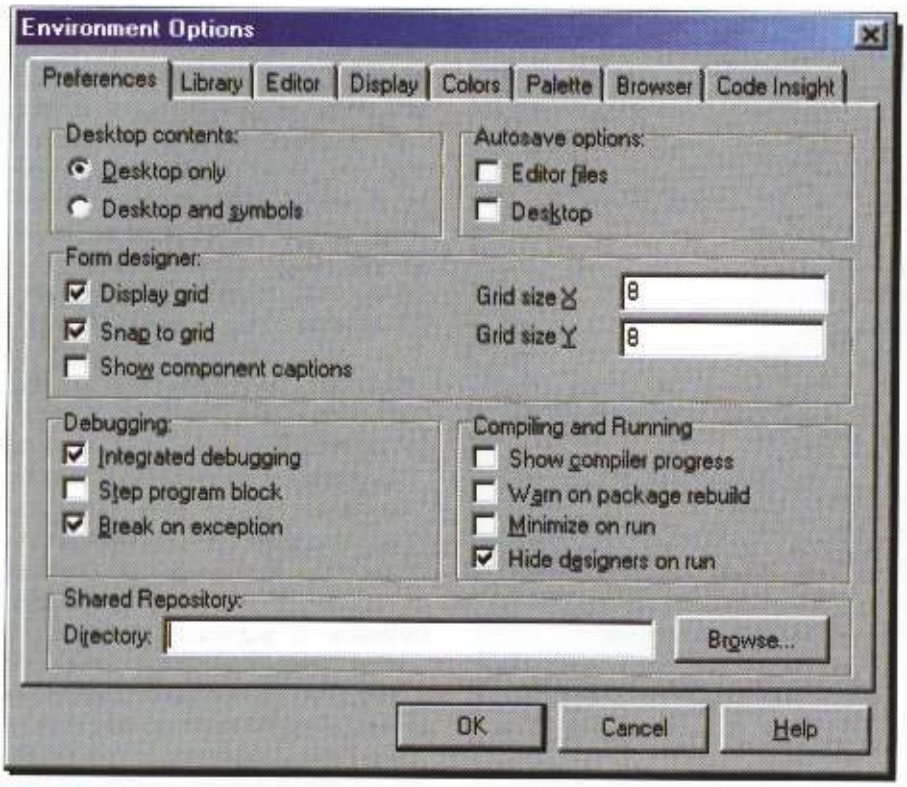

FIG. 14 Cuando se trabaja en grupo puede utilizarse un almacén de objetos compartido, que debe ser el mismo para todos.

representativo del proyecto, validándose por el procedimiento habitual.

■ **Modificación de fichas compartidas.**

Al compartir distintos proyectos un componente del almacén de objetos como una ficha, por ejemplo, las modificaciones introducidas pueden afectar a otros proyectos, dependiendo de la modalidad elegida para la inclusión del componente en cada uno. No existen implicaciones al utilizar la opción de copia pero sí al usar la de herencia, puesto que en este caso se tendrá en cuenta la versión más reciente de la ficha alojada en el almacén. Finalmente, en el caso de que la modalidad elegida sea la utilización directa, las modificaciones introducidas se efectúan siempre con carácter global.

Aún seleccionando la modalidad de herencia es posible evitar que las modificaciones posteriores en el **Object Repository** tengan efecto en la ficha añadida al proyecto en edición. Para ello lo más sencillo es almacenar en disco la ficha con otro nombre y utilizar ésta en lugar de la original.

Es posible también introducir las modificaciones en la ficha en la fase de ejecución en lugar de en la de diseño. Sin embargo, en la práctica esta técnica sólo resulta recomendable si se es usuario único de la misma y no se prevé que sea necesario introducir nuevas modificaciones en la ficha, resultando absolutamente desaconsejable si se piensa reutilizar la misma en otros proyectos. Otra técnica sería convertir la ficha compartida en un componente instalable en la paleta **Component** (Componentes), ya que de este modo se permite la modificación de la misma desde la fase de diseño.

ESPECIFICACIONES POR OMISIÓN PARA PROYECTOS Y FICHAS

A la hora de crear proyectos, fichas principales y secundarias pueden imponerse determinados valores por omisión, lo cual permitirá ahorrar tiempo y errores cuando se efectúan desarrollos con ciertas características similares.

■ **Proyecto.**

La secuencia de menús **File | New Application** (Archivo | Nueva Aplicación) crea un nuevo proyecto en blanco y lo sitúa sobre el área de trabajo, con una ficha principal y su módulo asociado. Alternativamente es posible también elegir una plantilla de proyecto para que se genere como base para cualquier aplicación nueva, o bien que se ejecute automáticamente un experto de proyecto que guíe al usuario paso a paso como asistente basado en cuadros de diálogo.

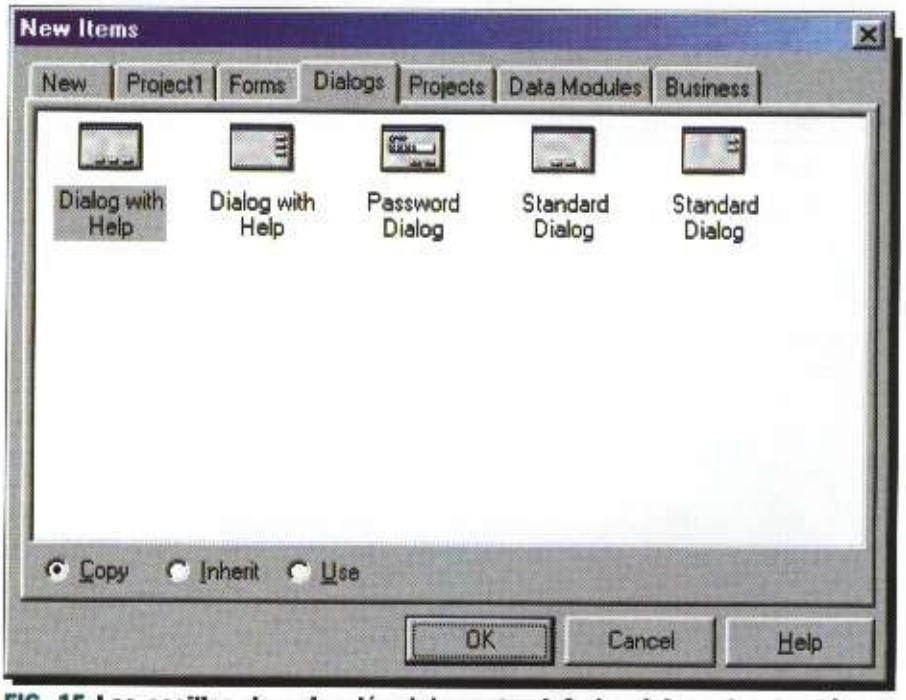

FIG. 15 Las casillas de selección del margen inferior del cuadro de diálogo imponen el tipo de vinculación con el proyecto.

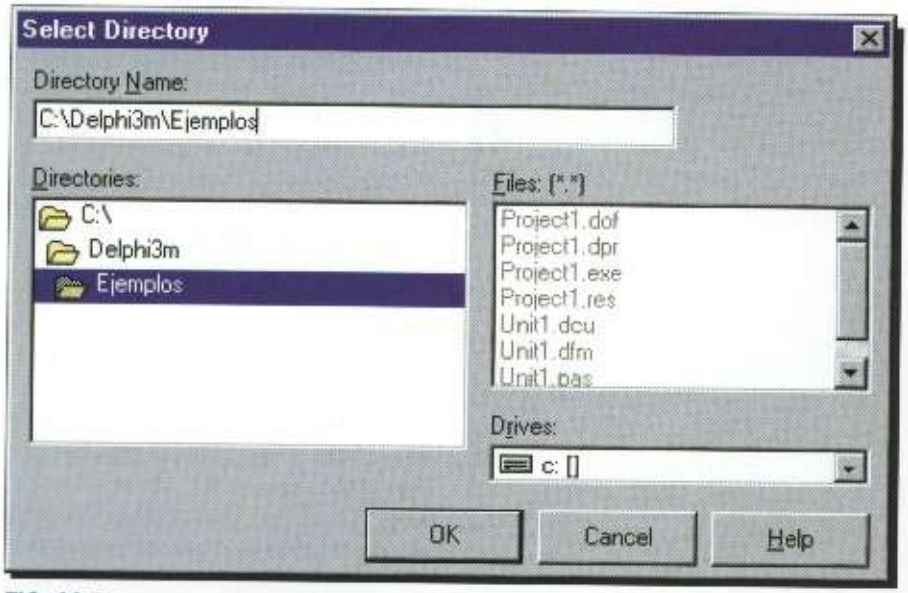

FIG. 16 Para crear un nuevo proyecto a partir de una plantilla basta con elegir una e indicar la carpeta que alojará los archivos para el proyecto.

Para establecer el proyecto en blanco a utilizar se debe seguir la secuencia de menús **Tools | Repository** (Herramientas | Almacén).

Dentro del cuadro de diálogo debe elegirse **Projects** (Proyectos) en el cuadro de lista del margen izquierdo y el proyecto a utilizar como plantilla en el cuadro de lista del margen derecho. A continuación debe marcarse la casilla de verificación **New Project** (Nuevo proyecto) y validar por el procedimiento habitual.

- **Fichas.**

Para definir el modelo de ficha en blanco que se creará mediante la secuencia de menús **File | New Form** (Archivo | Nueva Ficha) se debe seguir el procedimiento anterior, eligiendo en este caso **Forms** (Fichas) en el cuadro de lista del margen izquierdo y la ficha deseada en el derecho, marcando igualmente la casilla de verificación **New Form** y validando por el procedimiento habitual.

Para definir el modelo para la ficha principal debe marcarse en su lugar la casilla de verificación **Main Form** (Ficha Principal). Obviamente, pueden marcarse igualmente ambas casillas de verificación si lo que se desea es utilizar un mismo modelo inicial para ambos tipos de fichas.

FIG. 17 El usuario puede añadir proyectos, fichas y módulos al Object Repository, para su utilización posterior.

ESPECIFICACIONES POR OMISIÓN PARA OPCIONES DE PROYECTOS

Existen una serie de opciones de proyecto para las cuales pueden indicarse valores por omisión diferentes a los estándar, lo cual se lleva a cabo a través de cuadro de diálogo **Project Options**.

Se accede a este cuadro de diálogo tanto a través de la secuencia de menús **Project | Options** como pulsando el botón **Project Options** de la barra de botones de la ventana del **Project Manager**, o seleccionado la opción **Options** del menú contextual de la ventana. Debe tenerse en cuenta que los valores que se modifiquen tendrán efecto exclusivamente sobre el proyecto en edición.

Para que afecten a todos los proyectos que se creen en adelante se debe marcar la casilla de verificación **Default** (Omisión), situada en el margen inferior izquierdo de todas las fichas de este cuadro de diálogo.

Las especificaciones para cada proyecto se almacenan en un archivo de texto de extensión DOF, dentro de su carpeta correspondiente.

- **Opciones generales para proyectos nuevos.**

Las fichas del cuadro de diálogo son: **Forms** (Fichas), **Application** (Aplicación), **Compiler** (Compilador), **Linker** (Montador de enlaces), **Directories/ Conditionals** (Directorios y condicionales), **Version/Info** (Información de versión) y **Packages** (Paquetes).

Cuando se marca la casilla de verificación **Default** para cualquiera de estas fichas se registran las modificaciones en el archivo de texto **DEFPROJ.DOF**, que afecta en adelante a todos los proyectos nuevos. Debe tenerse en cuenta, sin embargo, que es-

tas especificaciones no serán tenidas en cuenta cuando se parte, desde el inicio, de una plantilla de proyecto del **Object Repository** que cuente con archivo de definición de proyecto propio.

- **Forms.**

Esta ficha permite elegir las fichas a utilizar como plantilla, así como determinar el orden en el cual deben crearse sobre el área de trabajo.

- **Application.**

Dentro de esta ficha pueden indicarse valores globales de la aplicación, como su título, trayecto del archivo de ayuda en línea e icono asociado.

El título es el rótulo que mostrará la ventana minimizada de la aplicación, que de no definirse coincidiría con el nombre del archivo ejecutable de la misma.

El trayecto del archivo de ayuda en línea debe establecerse obligatoriamente para vincular éste a la aplicación. Por último, si no se desea utilizar el icono de Delphi por omisión puede utilizarse uno alternativo, que puede tomarse de la propia biblioteca **Image** de Borland Delphi e incluso modificarse mediante el editor de imágenes, o seleccionar un icono estándar de Windows situado en cualquier carpeta.

- **Compiler.**

Se cuenta en esta ficha con una serie de apartados con casillas de verificación, que afectan a la generación de código, sintaxis, tratamiento de errores en tiempo de ejecución, depuración y mensajes.

- **Linker.**

Puede definirse aquí el modo en el cual deben enlazarse los archivos de programa, así como las opciones para archivos EXE y DLL.

La casilla de verificación **Include TD32 debug info** (Incluir información de depuración TD32) permite compilar la información de depuración para que pase a formar parte del archivo ejecutable, aunque a costa de aumentar su tamaño.

- **Directories/Conditionals.**

Esta ficha permite fijar la posición de salida del ejecutable del proyecto, así como establecer los trayectos de disco de localización de los recursos necesarios para las fases de compilación, enlace de módulos y distribución de la aplicación final.

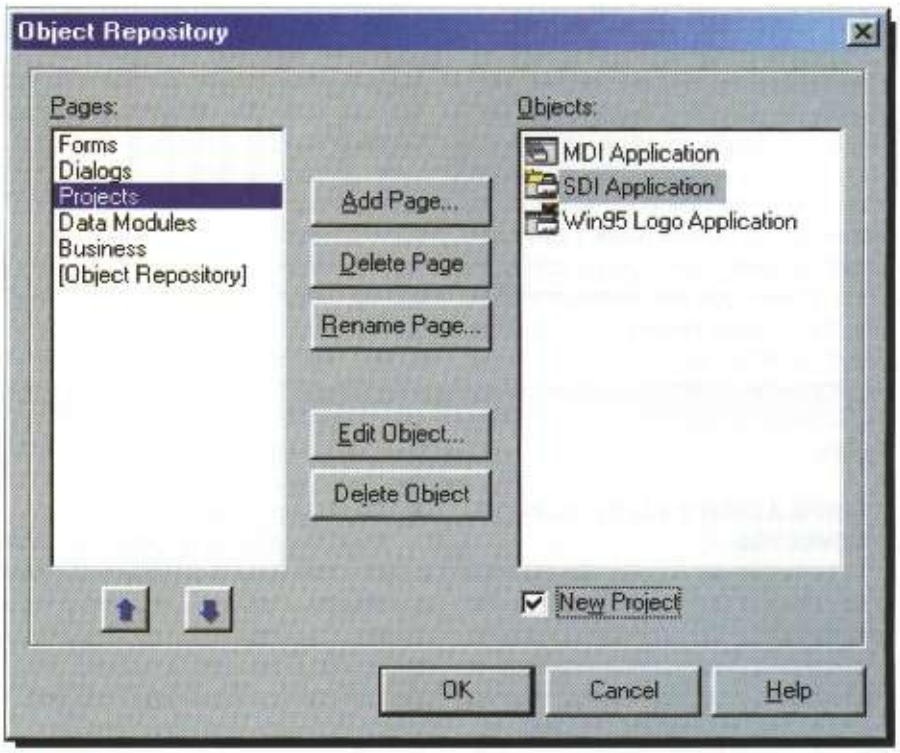

FIG. 18 La casilla de verificación New Project permite establecer un nuevo modelo de proyecto en blanco como punto de partida.

- **Version/Info.**

Puede indicarse aquí si se desea incluir información de versión del proyecto y la manera de hacerlo. Esta información pasa a formar parte de una etiqueta del código y se muestra a través de cuadro de diálogo **Properties** (Propiedades) estándar de Windows al elegir la opción **Properties** del menú contextual del archivo EXE o los posibles DLL al explorar la carpeta de Windows donde se encuentren.

- **Packages.**

Se indican en esta ficha los paquetes en fase de diseño utilizados en el proyecto. Los paquetes incluidos en **Runtime packages** (Paquetes en tiempo de ejecución) se vinculan automáticamente al proyecto en la fase de compilación.

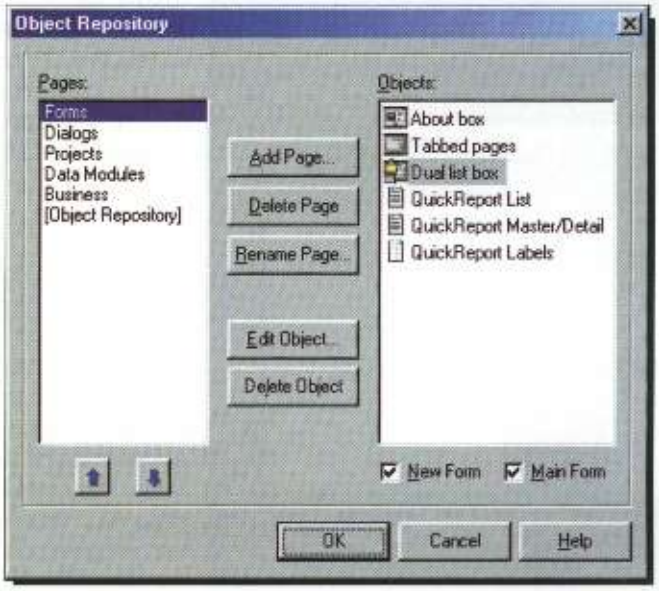

FIG. 19 El procedimiento a seguir para establecer el modelo básico para fichas principales y secundarias es similar al descrito anteriormente.

■ **Restauración de valores por omisión.**

Es posible restaurar globalmente los valores originales considerados por omisión para los parámetros comentados anteriormente con sólo eliminar el archivo **DEFPROJ .DOF** correspondiente de su ubicación en disco, o bien cambiarlo de nombre para hacerlo inaccesible.

PREFERENCIAS DE ENTORNO

El entorno de Borland Delphi es igualmente personalizable por medio del cuadro de diálogo **Environment Options**, afectando en adelante a todos los proyectos editados. Se accede a este cuadro de diálogo a través de la secuencia de menús **Tools | Environment Options**, que cuenta con 8 fichas. Puesto que los valores propuestos por omisión pueden considerarse como óptimos, conviene reservar su modificación a usuarios veteranos en el desarrollo de aplicaciones en Borland Delphi.

COMPILACIÓN Y EJECUCIÓN DE PROYECTOS

El objetivo del entorno de desarrollo de Borland Delphi es la creación de archivos EXE y DLL, mediante los cuales se pueda distribuir una versión final ejecutable del proyecto. Lógicamente, hasta obtener una versión ejecutable sin errores aparentes es preciso normalmente llevar a cabo más de un una compilación para descubrir posibles errores, tanto de desarrollo como mostrados en el momento de la ejecución.

- **Verificación del código.**

Para comprobar la sintaxis del código fuente no es necesario compilar el proyecto, sino que se puede recurrir a un procedimiento específico que no genera archivos de salida y, por lo tanto, resulta mucho más rápido y efectivo, al cual se accede mediante la secuencia de menús **Project | Syntax Check** (Proyecto | Verificación de Sintaxis). Si no se detectan errores el puntero del ratón se muestra simplemente como ocupado durante cierto tiempo y, finalmente, recupera su aspecto habitual no mostrándose ningún mensaje. En caso contrario se muestra la ventana que contiene el código erróneo y la línea resaltada en la cual está localizado, mostrándose el error o errores pertinentes en el margen inferior de la misma.

- **Compilación del proyecto.**

Para compilar el proyecto se debe seguir la secuencia de menús **Project | Compile** (Proyecto Compilar) o bien emplear la combinación de teclas **Control + F9**. Al hacerlo se compila el código fuente de cada módulo, de existir cambios desde la última compilación, generándose un archivo DCU por módulo. A continuación se compila el archivo de proyecto y se crea el ejecutable o biblioteca de vínculos dinámicos, con el mismo nombre que el archivo de proyecto pero con extensión EXE o DLL, según corresponda, quedando el proyecto listo para ejecutarse.

Si se desea se puede obtener información adicional sobre el proceso de compilación a través de cuadro de diálogo **Information** (Información) que se muestra mediante la secuencia de menús **Project | Information**.

Es posible también obtener información adicional durante la compilación si se marca la casilla de verificación **Show Compiler progress** (Mostrar progreso de Compilación) del apartado **Compiling and Running** de la ficha **Preferences**, accesible mediante la secuencia de menús **Tools | Environment Options.**

- **Generación del proyecto.**

Si se desea recompilar íntegramente el proyecto, sin comprobar la necesidad de recompilación de cada módulo en particular en función de las posibles modificaciones introducidas en los mismos, se puede seguir la secuencia de menús **Project | Build All** (Proyecto | Generar Todo).

Esta opción resulta útil para asegurarse de la perfecta sincronización del conjunto de archivos fuente y objeto que componen el proyecto, debiéndose recurrir también a ella cuando se introducen cambios en las directivas generales de compilación.

- **Ejecución del proyecto.**

Tras compilar con éxito el proyecto éste puede ejecutarse tanto desde el propio entorno como explorando la carpeta en que resida el ejecutable y haciendo doble clic sobre el mismo. Es posible también compilar y ejecutar a renglón seguido el proyecto, al pulsar el botón **Run** (Ejecutar) de la **Speedbar**, pulsar **F9**, o bien seguir la secuencia de menús **Run | Run**.

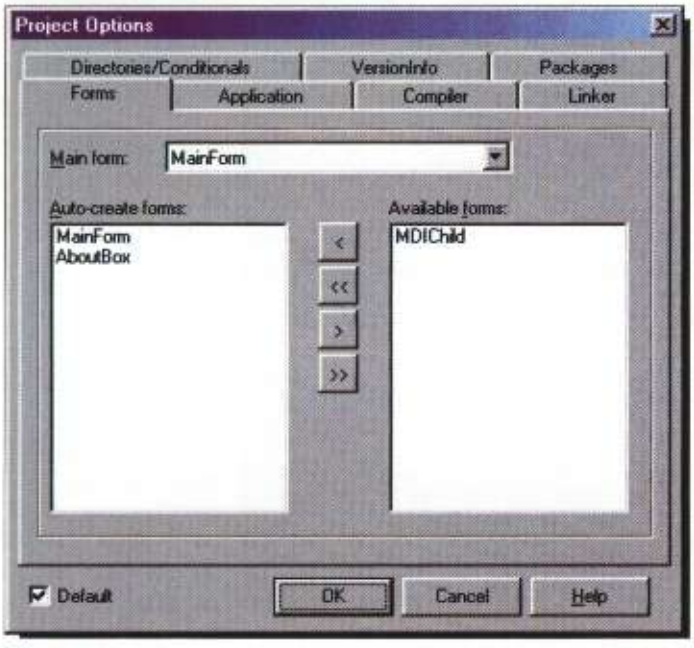

FIG. 20 La ficha Forms del cuadro de diálogo Project Options permite elegir la ficha a utilizar en adelante como plantilla.

RESUMEN

En este capítulo hemos tratado sobre la estructura de los proyectos, indicada principalmente en el archivo DPR de descripción de proyecto, comentando los distintos archivos que componen los proyectos y la distinta funcionalidad de cada uno según sus extensiones. Se ha hablado también sobre el Project Manager como herramienta ideal para la gestión de proyectos, puesto que permite inspeccionar de un vistazo los distintos componentes. Hablamos, finalmente, sobre el Object Repository, como biblioteca de componentes suministrada inicialmente por el fabricante, pero a la que se pueden añadir progresivamente los diseños del propio usuario.