CorelDRAW® GRAPHICS SUITE X5

Guía de Programación de Macros

Guía traducida al español por el Corel-Experto: Felipe de Jesús Guzmán Llamas, Guadalajara, México.
Publicada con autorización en el WWW.CORELCLUB.ORG - Club Internacional de Usuarios de Corel.

Contenido

Introducción

Bienvenido a la Guía de Programación de Macros.

Este recurso puede ayudarlo a explorar las características y funciones relacionadas con las macros de CorelDRAW® y Corel® PHOTO-PAINT™. La comprensión de estas características y funciones puede ayudarlo a automatizar tareas o desarrollar soluciones comerciales que se integren con el producto.

Esta guía contiene las siguientes secciones:
- **Entendiendo la automatización** — lo introduce a los conceptos de automatización y macros, y a los formatos de programación de macros que son soportados por CorelDRAW y Corel PHOTO-PAINT.
- **Iniciando con las macros** — proporciona un vistazo general de las herramientas y características relacionadas con las macros en CorelDRAW y Corel PHOTO-PAINT.
- **Creación de macros** — describe cómo escribir, grabar, ejecutar y depurar macros.
- **Confección de una interfaz amigable de macros** — demuestra cómo resaltar la funcionalidad de las macros mediante cuadros de diálogo, botones de barras de herramientas, interacción con el usuario y documentación.
- **Organización y puesta en marcha de macros** — muestra cómo organizar y desplegar las macros creadas por usted.
- **Entendimiento del modelo de objeto CorelDRAW** — explica las características y funciones más importantes del modelo de objeto CorelDRAW.

También se incluye un glosario, el cual define muchos de los términos clave utilizados en esta documentación.

Esta sección contiene los siguientes temas:
- Acerca de esta documentación.
- Acerca de los recursos adicionales.
- Acerca de Corel.

La mayoría de los ejemplos de código proporcionados en esta documentación están escritos en VBA.

Acerca de esta documentación

Esta documentación da por hecho que el lector tiene experiencia en, por lo menos, un lenguaje procesal de programación, tal como BASIC, Microsoft® Visual Basic® (VB), C, C + +, Java™, Pascal, Cobol o Fortran.
Esta documentación no describe las bases de programación procesal (tales como funciones, condicionales, bifurcaciones y bucles). Por consiguiente, antes de utilizar esta documentación, a los no-programadores se les recomienda encarecidamente aprender programación básica en un lenguaje tal como Microsoft® Visual Basic® for Applications (VBA).

La mayoría de los ejemplos de código proporcionados en esta documentación están escritos en VBA.

Para información más detallada sobre el ambiente de programación en VB y VBA, vea la ayuda de Microsoft Visual Basic, la cual está disponible en el Menú **Ayuda** del Editor de Macros.
Para una introducción más básica a las macros, por favor vea el tema "Operaciones con macros" en el archivo principal de Ayuda para CorelDRAW o Corel PHOTO-PAINT. Puede acceder al archivo principal de ayuda desde la aplicación haciendo click en **Ayuda** » **Temas de Ayuda...**

Las convenciones de documentación llevadas en esta guía son explicadas en la siguiente tabla:

Cuando vea esto:	**Encontrará:**
	Una nota — describe las condiciones requeridas para ejecutar un procedimiento o presenta otra información esencial.
	Un consejo — describe información útil tal como atajos, métodos alternos o ventajas relacionadas con el procedimiento.
Texto en negrita	El nombre de un control u otro elemento en la interfaz del usuario.
<texto en cursiva y entre corchetes angulares>	Un contenedor para información específica del usuario, tal como una ruta o nombre de archivo.
`Texto tipo monospace`	Una referencia para codificación.

Para más información acerca de las macros

CorelDRAW Graphics Suite X5 le proporciona recursos adicionales con información útil acerca de las macros.
La documentación de las macros para CorelDRAW Graphics Suite X5 está instalada en la siguiente ruta (donde **X:** es la unidad en la que el programa está instalado):
X:\Archivos de programa\Corel\CorelDRAW Graphics Suite\Data

Estos recursos adicionales están descritos en la tabla siguiente:

Recurso	**Descripción y nombre de archivo**
Ayuda de Macros para CorelDRAW	Proporciona información extensa acerca de los modelos de objeto de CorelDRAW, así como características y funciones de la aplicación relacionadas con macros. **draw_om.chm**
Ayuda de Macros para Corel PHOTO-PAINT	Proporciona información extensa acerca de los modelos de objeto de Corel PHOTO-PAINT, así como características y funciones de la aplicación relacionadas con macros. **pp_om.chm**
Diagrama de modelos de objeto para CorelDRAW	Proporciona una representación jerárquica de los modelos de objeto en CorelDRAW. **CorelDRAW Object Model Diagram.pdf**

Recurso	Descripción y nombre de archivo
Diagrama de modelos de objeto para Corel PHOTO-PAINT	Proporciona una representación jerárquica de los modelos de objeto en Corel PHOTO-PAINT. **Corel PHOTO-PAINT Object Model Diagram.pdf**

Para una introducción más básica a las macros, por favor vea el tema "Operaciones con macros" en el archivo principal de Ayuda de CorelDRAW o Corel PHOTO-PAINT. Puede acceder al archivo principal de ayuda desde la aplicación haciendo click en **Ayuda** » **Temas de Ayuda...**

Puede proponer cualquier comentario o sugerencia acerca de la documentación de macros utilizando la información de contacto localizada en www.corel.com/contact.

Acerca de recursos adicionales

CorelDRAW Graphics Suite X5 le proporciona recursos con información útil acerca del software.

Para mayor información acerca de las características en CorelDRAW Graphics Suite X5, puede consultar la Guía del Usuario en la Ayuda.

Por defecto, la Guía del Usuario está instalada en la siguiente ruta (donde **X:** es la unidad en la que se instaló el programa):

X:\Archivos de programa\Corel\CorelDRAW Graphics Suite X5\Languages\es\Help

También puede acceder a la Ayuda desde CorelDRAW o Corel PHOTO-PAINT haciendo click en **Ayuda** » **Temas de Ayuda...**

Para más información acerca del software

Para más información acerca de CorelDRAW Graphics Suite X5, vea los siguientes recursos en línea:

Recurso	Descripción y nombre de archivo
Sitio Web de CorelDRAW Graphics X5	Proporciona las últimas noticias, consejos y trucos, así como información acerca de actualizaciones. **www.corel.com/coreldraw**
Sitio Web de Servicios de Soporte de Corel	Proporciona información dinámica y precisa acerca de las características del producto, especificaciones, precios, disponibilidad, servicios y soporte técnico. **www.corel.com/support**
Base de Conocimiento de Corel®	Proporciona un repertorio de artículos escritos por el equipo de Servicio de Soporte Técnico de Corel en respuesta a dudas de los usuarios. **www.corel.com/knowledgebase**

Recurso	Descripción y nombre de archivo
Comunidad CorelDRAW en línea	Proporciona interacción con otros usuarios a través de experiencias compartidas, respuestas a preguntas y recibiendo ayuda y sugerencias. **www.coreldraw.com**

Puede proponer cualquier comentario o sugerencia acerca de la documentación de macros utilizando la información de contacto localizada en www.corel.com/contact.

Acerca de Corel

Corel es una de las más sobresalientes compañías de software del mundo, con más de 100 millones de usuarios activos en más de 75 países. Desarrollamos software que ayuda a la gente a expresar sus ideas y compartir sus historias de una manera más excitante, creativa y convincente. A lo largo de los años, nos hemos forjado una reputación al entregar productos verdaderamente innovadores y fáciles de aprender y utilizar, ayudando con esto a que la gente alcance nuevos niveles de productividad. La industria nos ha respondido con cientos de premios por la innovación, diseño y valor del software.

Nuestro catálogo de premios ganados por productos incluye algunas de las marcas más ampliamente reconocidas y populares a nivel mundial, en las que se incluyen CorelDRAW® Graphics Suite, Corel® Painter™, Corel DESIGNER® Technical Suite, Corel® PaintShop Photo™ Pro, Corel® VideoStudio®, Corel® WinDVD®, Corel® WordPerfect® Office, WinZip®, así como el recientemente lanzado Corel® Digital Studio™ 2010. Nuestro principal centro de operaciones está en Ottawa, Canadá, además de importantes oficinas en los Estados Unidos, Reino Unido, Alemania, China, Taiwán y Japón.

Entendiendo la automatización

Antes de comenzar a trabajar con macros, necesitará entender el concepto de automatización. Esta sección proporciona información básica acerca de la automatización y acerca de los formatos de programación de macros que son soportados por CorelDRAW y Corel PHOTO-PAINT.

Esta sección contiene los siguientes temas:
- ¿Qué es la automatización?
- ¿Cuáles ambientes de automatización son soportados?
- ¿Cuáles son los principales elementos de automatización?
- ¿Cómo está estructurado el código de automatización?

¿Qué es la automatización?

Muchas acciones que usted ejecuta en CorelDRAW y Corel PHOTO-PAINT pueden estar combinadas con otras acciones ligadas dentro de una simple solución automatizada. La automatización de tareas repetitivas ahorra tiempo, reduce esfuerzo y le permite ejecutar operaciones que sería bastante complejo el ejecutarlas manualmente.

La automatización puede ser utilizada tanto por programadores como por no-programadores.

Esta documentación no enseña habilidades de programación a los no-programadores; más bien le ayuda a programadores experimentados a desarrollar soluciones útiles dentro de CorelDRAW y Corel PHOTO-PAINT. Si usted no es un programador, sería buena idea referirse a otros recursos relacionados con la programación antes de continuar leyendo esta documentación.

¿Qué es una macro?

La mayoría de las macros son creadas para automatizar una serie de tareas dentro de la aplicación. El significado más simple del término "macro" es la grabación de un grupo de acciones relacionadas que pueden ser luego ejecutadas automáticamente en secuencia en cualquier momento que se necesite. Las macros constan de instrucciones que son escritas en un lenguaje de programación. Algunos lenguajes de programación proporcionan acceso a acciones adicionales más avanzadas que no pueden ser grabadas.

Para propósitos de esta documentación, una macro se refiere a una solución codificada que ejecuta tareas en la aplicación mediante funciones y subrutinas automatizadas (ver "¿Cómo se construyen las funciones y subrutinas?" en la página 16).

Aunque usted puede grabar una secuencia de acciones en CorelDRAW y Corel PHOTO-PAINT, el verdadero poder de la automatización será cuando pueda añadir condiciones y mecanismos de bucleo (looping) a una grabación. Por ejemplo, consideremos una simple macro que aplica un relleno rojo y un contorno con grosor de 1 punto a una forma seleccionada. Al añadirle una condición y un mecanismo de bucleo al código, se puede producir una macro que busque cada forma seleccionada y aplique solamente el relleno a objetos de texto y únicamente el contorno a todos los demás tipos de objetos.

Después de crear una macro, asegúrese que ésta produce los resultados deseados ajustando su código línea por línea, o "depurándolo". Cuando esté satisfecho con la macro, puede guardarla para usos futuros e incluso compartirla con otros.

¿Cuáles macros de muestra están disponibles?

CorelDRAW Graphics Suite X5 incluye muestras de macros VBA, las cuales proporcionan funcionalidad adicional, demuestran la automatización en la suite y le proporcionan el ejemplo de código.

Las siguientes muestras de macros VBA están incluidas en CorelDRAW:

- Asistente de calendario (**CalendarWizard.gms**) — genera calendarios personalizados. Usted puede elegir el rango de fecha, diseño, fuente, color, lenguaje y muchas otras opciones. También puede añadir días festivos y fases lunares.
- Convertidor de archivos (**FileConverter.gms**) — convierte un vector o mapa de bits a un formato específico de vector o mapa de bits. Usted puede elegir exportar parámetros al utilizar los cuadros de diálogo asociados con filtros particulares. También puede guardar cada página como un archivo separado y ajustar varias propiedades de página, tales como tamaño, orientación y color de fondo. Los siguientes formatos de archivo son soportados: AI, BMP, CDR, CGM, CMX, CPT, DSF, EPS, GIF, JPEG, PCT, PNG, PPF, SVG, SWF, TIF, WMF y WPG.

Las siguientes muestras de macros VBA están incluidas en Corel PHOTO-PAINT:

- Creador de diapositivas HTML (**Slideshow.gms**) — genera una diapositiva HTML de las imágenes que usted especifique. Abre archivos que pueden ser añadidos, ordenados y publicados como una serie de archivos HTML, cada uno de los cuales muestra una imagen y proporciona botones de navegación. Usted también puede crear un título, alternar texto y nombrar cada diapositiva. Además puede elegir una ruta y un nombre para la descripción de la carpeta, seleccionar la imagen de la carpeta y especificar notas, un título, un URL y más.

¿Cuáles ambientes de automatización son soportados?

Para las versiones 6 a 9 de CorelDRAW, el único método de automatizar tareas era utilizando el lenguaje Corel SCRIPT™. Los desarrolladores de soluciones utilizaban Corel SCRIPT para crear mini-aplicaciones inteligentes para el dibujo de formas, reposicionamiento y cambio de tamaño de formas, apertura y cierre de documentos, y ajuste de estilos dentro de CorelDRAW.

Aunque el lenguaje Corel SCRIPT fue útil para automatizar tareas básicas, una solución más flexible y poderosa estaba siendo necesaria. Para la versión 10, CorelDRAW se había reforzado con el motor Microsoft Visual Basic for Applications (VBA), el cual manejaba la automatización "detrás de escena". La adición de VBA hizo a CorelDRAW inmediatamente accesible para millones de desarrolladores de VBA y de Microsoft Visual Basic (VB) alrededor del mundo. Desde entonces, VBA se ha incluido en cada versión de CorelDRAW Graphics Suite. Más recientemente, CorelDRAW Graphics Suite ha incluido Microsoft® Visual Studio® Tools for Applications (VSTA), el sucesor de VBA. CorelDRAW X5 y Corel PHOTO-PAINT X5 soportan VBA versión 6.4 y VSTA versión 2.0.

Aunque CorelDRAW ya no incluye el editor Corel SCRIPT, todavía incluye el motor en tiempo de ejecución Corel SCRIPT. Por consiguiente, usted fácilmente puede migrar scripts que fueron escritos en versiones

anteriores de CorelDRAW a otras versiones más recientes del software. Para más información sobre el uso de Corel SCRIPT con CorelDRAW vea "Trabajando con scripts" en el archivo principal de Ayuda de CorelDRAW (**draw.chm**).

En Corel PHOTO-PAINT usted puede automatizar tareas utilizando VBA o VSTA para crear una macro, o utilizando Corel SCRIPT para crear un script. Una macro es la mejor elección si usted quiere escribir el código que sea requerido para llevar a cabo la tarea (utilizando VBA o VSTA), mientras un script es la mejor elección si lo que usted quiere es grabar los pasos que se requieran para llevar a cabo la tarea (utilizando Corel SCRIPT). Para más información sobre el uso de Corel SCRIPT en Corel PHOTO-PAINT vea "Trabajando con scripts" en el archivo principal de Ayuda de Corel PHOTO-PAINT (**corelpp.chm**).

La combinación de VBA y VSTA con CorelDRAW Graphics Suite proporciona una plataforma para el desarrollo de poderosas soluciones gráficas corporativas, tales como generadores automatizados de boletos, calendarios personalizados y procesadores de archivos por lotes. Estas aplicaciones también pueden ser utilizadas para optimizar el flujo de trabajo. Por ejemplo, cuando se crea la hoja membretada de una compañía, usted puede añadir una página-esquema al vuelo. También puede personalizar alguna funcionalidad integrada del software, tal como la creación, alineación o transformación de objetos.
VBA y VSTA proporcionan su propio ambiente de desarrollo totalmente integrado (IDE), con listas de mensajes automáticos (pop-ups), sintaxis rasaltada, depuración línea por línea y ventanas visuales de diseñador. Estas características son particularmente útiles para desarrolladores inexpertos.

Para más información sobre VBA y VSTA, vea las siguientes secciones:
- ¿Qué es VBA?
- ¿Qué es VSTA?

¿Qué es VBA?

Microsoft Visual Basic for Applications (VBA) es un ambiente de programación incorporado que se puede utilizar para automatizar funciones repetitivas y crear soluciones inteligentes en CorelDRAW y Corel PHOTO-PAINT. VBA es un subconjunto del ambiente de programación orientado a objetos de Microsoft Visual Basic (VB). Comúnmente, VBA está integrado dentro de otra aplicación para personalizar la funcionalidad dentro de esa aplicación.

VBA es tanto un lenguaje como un editor. El lenguaje VBA no puede ser utilizado sin su editor, y el editor VBA es la única utilidad en la cual el código VBA puede ser editado o los programas VBA pueden ser ejecutados.

El lenguaje VBA es un lenguaje de programación orientado a eventos. En otras palabras, es utilizado para escribir código que produzca una respuesta a determinada acción, tal como el hacer click sobre un botón o el elegir una opción de un cuadro de lista. Cuando la acción ocurre, el evento apropiado es llamado, y el código para ese evento es ejecutado. Los eventos pueden ser simples o complejos. Por ejemplo, se puede codificar una simple línea que muestre una caja de mensaje o escribir un procedimiento entero que interactúe con una base de datos.

Con la programación tradicional procesal (o "programación orientada a objetos"), el programa inicia en la primer línea y continúa ejecutándose línea por línea. VB proporciona un ejemplo de un ambiente de programación orientado a objetos.

La mayoría de los ejemplos de código proporcionados en esta documentación están escritos en VBA

El editor VBA, llamado el "Editor de Macros" (formalmente el "Editor de Visual Basic") en CorelDRAW y Corel PHOTO-PAINT, es un ambiente de desarrollo integrado (IDE, por sus siglas en inglés) que le permite manipular los objetos que están expuestos por el modelo de objeto de la aplicación. Para ayudarlo a codificar sus macros, el Editor de Macros le proporciona Ayuda Sensible al Contexto para todos los elementos de modelo de objeto que estén disponibles .

VBA es un controlador de automatización en proceso. En otras palabras, VBA puede ser utilizado para controlar las características de Corel DRAW y Corel PHOTO-PAINT que puedan ser automatizadas, y VBA se ejecuta eficientemente bordeando los mecanismos de sincronización del interproceso. Sin embargo, la automatización en proceso a la que VBA puede acceder también puede ser accesada por lo siguiente:

- Controladores de automatización externos fuera de proceso (clientes OLE).
- Aplicaciones que están desarrolladas en lenguajes de programación (tales como VB, Visual C++, Windows® Script Host y C++) que pueden ser utilizados para desarrollar clientes OLE.
- El motor VBA de otras aplicaciones.

VBA proporciona un conjunto de herramientas para personalizar la interfaz gráfica del usuario de CorelDRAW y Corel PHOTO-PAINT. Estas herramientas le permiten procesar y presentar datos eficiente y efectivamente. Las ventajas del uso de VBA con CorelDRAW y Corel PHOTO-PAINT incluyen las siguientes:

- Familiaridad del lenguaje VB.
- Desarrollo de aplicaciones rápidas (RAD) IDE.
- Ejecución rápida del tiempo de ejecución en las soluciones integradas resultantes.
- Paquete de formas extendibles que soportan controles ActiveX® para la creación de interfaces de usuario.
- Acceso total a la Interfaz de Programación de la Aplicación (API) de Windows® y al archivo de sistema subyacente.
- Conectividad con datos empresariales
- Integración con otro software que esté basado en modelos de componentes de objeto (COMs)

VBA le permite personalizar una aplicación a la medida de sus necesidades, o incluso integrarla con otra aplicación habilitada para VBA al hacer referencia a los componentes del modelo de objeto de la segunda aplicación. Aunque VBA fue desarrollado por Microsoft y está incorporado en casi todas sus aplicaciones de escritorio (incluyendo Microsoft Office), Microsoft otorga licencias de esta tecnología a otras compañías, incluyendo Corel Corporation (en CorelDRAW Graphics Suite, Corel DESIGNER® Technical Suite y Corel® WordPerfect® Office), Autodesk, Inc. (en AutoCAD®), y el Consorcio en Tecnología IntelliCAD (en IntelliCAD®). Por esta razón, CorelDRAW Graphics Suite es compatible con una amplia gama de aplicaciones que soportan VBA. Sin embargo, el motor de CorelDRAW Graphics Suite también puede controlar aplicaciones que no soporten VBA. Consecuentemente, usted puede construir soluciones en CorelDRAW y Corel PHOTO-PAINT que accedan a bases de datos, procesadores de texto, editores de contenido especializado, documentos XML y más.

Para una lista completa de las aplicaciones que soportan VBA, vea el sitio Web de Microsoft.

CorelDRAW Graphics Suite X5 incluye la versión 6.4 de VBA.

¿En qué difiere VBA de VB y VBScript?

El sistema de programación VB es un conjunto avanzado de herramientas de programación que proporciona funcionalidad y componentes avanzados para el sistema operativo Windows y otros programas basados en Windows. Por ejemplo, a diferencia de VBA o VBScript, VB le permite crear extensiones de aplicación (archivos DLL) y archivos ejecutables independientes (archivos EXE). (Los programas que usted construya con VBA deben correr dentro de la aplicación anfitriona).

VB es una versión "visual" del lenguaje de programación BASIC —esto es, proporciona indicios visuales dentro del editor. Como resultado, VB es un lenguaje fácil de aprender. Por otro lado, Microsoft ha enriquecido grandemente el lenguaje original BASIC, de modo que VB es tanto poderoso como rápido (aunque no tan poderoso como Java o C++, ni tan rápido como C).

VBA es un subconjunto del lenguaje de programación VB, y utiliza la estructura de programación de VB para manipular los elementos del modelo de objeto que están expuestos por una aplicación. La manipulación de estos

objetos da como resultado pequeños paquetes de procedimientos de código dentro de la aplicación. Estos procedimientos de código y proyectos resultantes son llamados adiciones ("add-ins").

VBScript (algunas veces referido como Microsoft Visual Basic, Scripting Edition) es también un subconjunto del lenguaje de programación VB. VBScript es un lenguaje de programación basado en documentos Web HTML.

¿En qué difiere VBA de Java y JavaScript?

VBA es similar a Java y JavaScript® en que es un lenguaje de programación procesal de alto nivel, con total eliminación de residuos e indicador de soporte de memoria muy baja. (Para más información, vea ¿Cómo se utilizan los indicadores de memoria, y cómo está alojada la memoria?" en la página 17). Por otro lado, el código que es desarrollado en VBA, así como el código desarrollado en Java y JavaScript, soportan compilación a petición y pueden ser ejecutados sin ser compilados.

VBA es también similar a JavaScript en que no puede ser ejecutado como una aplicación independiente. JavaScript está incorporado dentro de páginas web como un mecanismo para manipular el modelo de objeto del documento (DOM) del explorador web. De igual forma, los programas VBA son ejecutados dentro de un ambiente anfitrión (en este caso, CorelDRAW o Corel PHOTO-PAINT) para manipular el modelo de objeto del anfitrión.

La mayoría de las aplicaciones VBA pueden ser compiladas a código-p para hacerlas correr más rápidamente, aunque la diferencia es apenas notable debido a la actual sofisticación del hardware de computadora. Una compilación similar es posible en Java, pero no en JavaScript.

Finalmente, mientras que VBA utiliza sólo un signo de igual (=) tanto para comparación como para asignación, Java y JavaScript utilizan un signo de igual (=) para asignación y dos signos de igual (==) para comparaciones Booleanas. (Para más información, vea "¿Cómo se utilizan las comparaciones Booleanas y de asignación?" en la página 18).

¿En qué difiere VBA de C y C++?

Al igual que C y C++, VB utiliza funciones. En VB las funciones pueden ser utilizadas para devolver un valor, pero las subrutinas no se utilizan de ese modo. Sin embargo, las funciones son utilizadas en C y C++ independientemente de si se quiere devolver un valor. (Para más información, vea "¿Cómo se construyen las funciones y subrutinas?" en la página 16).

VBA aloja y libera memoria transparentemente. En C y C++, sin embargo, el desarrollador es responsable de la mayoría de la administración de memoria. Como resultado, la utilización de cadenas en VBA es incluso más simple que el uso de clases **Cstring** en C++.

Finalmente, mientras que VBA utiliza sólo un signo de igual (=) tanto para comparación como para asignación, C y C++ utilizan un signo de igual (=) para asignación y dos signos de igual (==) para comparaciones Booleanas. (Para más información, vea "¿Cómo se utilizan las comparaciones Booleanas y de asignación?" en la página 18).

¿En qué difiere VBA de WHS?

Windows Script Host (WHS) es un controlador de automatización "fuera de proceso" que le permite hacer programación ocasional y automatización de las tareas de Windows y puede ser utilizado para controlar CorelDRAW Graphics Suite. Aunque WSH es una útil extensión del sistema operativo Windows, los scripts de WSH tienden a ser lentos porque deben correr fuera de proceso, y no pueden ser compilados (y deben ser interpretados así como son ejecutados).

WSH es un anfitrión para varios lenguajes de programación, cada uno de los cuales tiene su propia sintaxis. Sin embargo, el lenguaje estándar que utiliza WSH es un lenguaje de macro que se asemeja a VB, de modo que para scripts estándar, la sintaxis es la misma que la de VBA.

¿Qué es VSTA?

El sucesor de VBA, Microsoft Visual Studio Tools for Applications (VSTA) está basado en Microsoft Visual Studio 2008. En CorelDRAW Graphics Suite X5, la característica VSTA soporta la plataforma .NET y permite el desarrollo en dos lenguajes de programación: Visual Basic .NET y Visual C#.

La mayor parte de los ejemplos de código proporcionados en esta documentación están escritos en VBA.

El editor VSTA en CorelDRAW y Corel PHOTO-PAINT es un ambiente de desarrollo integrado (IDE, por sus siglas en inglés) que le permite crear soluciones VSTA para el software.

CorelDRAW Graphics Suite X5 incluye la versión 2.0 de VSTA.

¿En qué se asemeja VSTA con VBA?

Tanto VSTA como VBA le permiten crear potentes soluciones de macro. Con VSTA, se utiliza el Editor VSTA como un IDE, y se utiliza Visual Basic .NET o Visual C# como un lenguaje de programación. Con VBA, se utiliza el Editor de Macros como un IDE, y se utiliza VBA como un lenguaje de programación.

Si se quiere ejecutar alguna de las siguientes tareas, se puede utilizar ya sea VSTA o VBA:

- Personalizar o extender las características de CorelDRAW Graphics Suite X5.
- Interactuar con otras aplicaciones que utilicen Visual Basic 6 — o con otros componentes compatibles que son externos a CorelDRAW Graphics Suite X5.
- Interactuar con servicios basados en Web.
- Personalizar el IDE con add-ins.
- Crear proyectos de macro — con soporte multi-enlazado, si se desea.
- Acceder al código de los proyectos de macro.
- Generar código de macro dinámicamente.
- Almacenar código de macro en formato pre-compilado.
- Ocultar código de macro de otros autores de macros.
- Depurar proyectos de macro.
- Crear interfaces de usuario personalizadas para proyectos de macro.

Sin embargo, si se quiere ejecutar alguna de las tareas siguientes, se debe utilizar VSTA:

- Acceder nativamente a la plataforma .NET — para dar soporte utilizando Managed Add-in Framework (MAF), referenciando directamente ensamblajes .NET, ejecutando código personalizado de Common Language Runtime (CLR), implementando políticas de seguridad .NET o creando interfaces de usuario al utilizar .NET WinForms.
- Personalizar totalmente el IDE.
- Crear proyectos de macro que estén certificados para correr en Windows Vista.
- Crear proyectos de macro que soporten procesadores de 64 bits.
- Crear proyectos de macro que soporten adecuaciones lado servidor.
- Crear proyectos de macro que soporten todo tipo de datos, incluyendo Int64 y BigDecimal.
- Crear proyectos de macro y ensamblajes que persistan sin el uso de almacén estructurado.
- Abrir y modificar proyectos de macro en Visual Studio.
- Compilar proyectos de macro para ensamblaje DLL.
- Ejecutar proyectos de macro fuera de proceso.

- Ejecutar proyectos de macro sin ocasionar que la aplicación anfitrión detenga la ejecución por errores o puntos de interrupción.
- Aislar proyectos de macro entre uno y otro; ejecutar independientemente proyectos de macro, y detenerlos durante tiempo de ejecución sin afectar otros proyectos activos.
- Publicar macros dentro del código dirigido.
- Evitar que corran servidores de interfaces de usuario personalizadas para proyectos de macro.

¿Cuáles son los elementos principales de automatización?

Si alguna vez ha desarrollado código orientado a objetos en C++, Borland Delphi o Java, está familiarizado con conceptos relacionados con la programación tales como "objetos", "clases", "propiedades" y "métodos". Sin embargo, vamos re-examinando estos términos tal y como se aplican a la automatización en CorelDRAW y Corel PHOTO-PAINT.

- ¿Qué es un modelo de objeto?
- ¿Qué es una clase?
- ¿Qué es una colección?
- ¿Qué es una propiedad?
- ¿Qué es un método?
- ¿Qué es un evento?
- ¿Qué es una enumeración?
- ¿Qué es una constante?

¿Qué es un modelo de objeto?

Un modelo de objeto representa la jerarquía de elementos (u "objetos") que forman una aplicación y define las interrelaciones de dichos objetos dentro de esa jerarquía. En un modelo de objeto, cada objeto es un hijo de otro objeto, el cual es, a su vez, hijo de otro objeto y así sucesivamente. Más aún, cada objeto en un modelo de objeto está definido por una propiedad, un método o un evento, o una combinación de estos elementos.

Además de proporcionar un alto nivel de estructura, un modelo de objeto también le permite utilizar tipos de objetos (o "clases") de varias maneras. Por ejemplo, un objeto **Shape** de tipo "grupo" es utilizado para contener otros objetos **Shape**, cada uno de los cuales es de tipo "grupo" o de algún otro tipo, tal como "rectángulo", "curva" o "texto".

Este alto nivel de organización hace que el modelo de objeto sea fácil de usar, a pesar de lo potente que es.

¿Cómo se utiliza un modelo de objeto en la automatización?

La automatización de CorelDRAW o Corel PHOTO-PAINT se realiza al utilizar el modelo de objeto de la aplicación para acceder a los objetos en un documento y hacer cambios en esos objetos.

En CorelDRAW y Corel PHOTO-PAINT, el objeto **Application** representa la parte superior en la jerarquía de objetos: el programa en sí. Todos los objetos son hijos o nietos (o bisnietos, y así sucesivamente) de la aplicación.

Iniciando con el objeto **Application**, usted puede "escudriñar" a través de las capas de jerarquía en el modelo de objeto hasta encontrar el deseado, y usualmente el más específico, objeto. Para referenciar el objeto deseado, se debe utilizar una notación estándar para separar cada nivel de la jerarquía de objetos.

Como en muchos lenguajes orientados a objetos, el ambiente de automatización de CorelDRAW y Corel PHOTO-PAINT requiere el uso de un punto u "operador punto" (.) para indicar que el objeto sobre la derecha es un miembro (o hijo) del objeto sobre la izquierda.

```
Application.Documents(1).Pages(1).Layers(1).Shapes(1).Name = "ABC"
```

Un objeto requiere su referencia jerárquica completa (es decir, que esté "totalmente titulado") a menos que esté disponible un atajo para él (o a menos que éste tenga un significado implícito o sobreentendido). Un atajo a un objeto es meramente un reemplazo sintáctico de la versión extendida de un objeto. Por ejemplo, el atajo del Objeto `ActiveLayer` reemplaza a la versión extendida `Application.ActiveDocument.ActivePage.ActiveLayer`, mientras que el atajo del objeto `ActiveSelection` reemplaza a la versión extendida `Application.ActiveDocument.Selection`

Para más información de atajos de objeto, vea "¿Cómo se utilizan los atajos de objeto?" en la página 22.

¿Qué es una clase?

Una clase es la definición o descripción de un objeto. Una clase esboza las propiedades, métodos y eventos que se aplican a un tipo de objeto en una aplicación; ésta actúa como una plantilla para todos los objetos de ese tipo de clase. Por utilizar una metáfora, la clase "carro" es un vehículo pequeño con un motor y cuatro ruedas.

Un objeto es una instancia de una clase. Para extender la metáfora del carro, el carro físico real (comprado para los propósitos de conducirlo) es un objeto (es decir, una instancia de la clase "carro").

En el contexto de CorelDRAW y Corel PHOTO-PAINT, cada documento abierto es una instancia de la clase **Document**, cada página en un documento es una instancia de la clase **Page**, y cada capa (y cada forma en cada capa) son más instancias de más clases. Por ejemplo, `Document` representa la clase **Document** en CorelDRAW y Corel PHOTO-PAINT. Sin embargo, `ActiveDocument` representa un objeto dentro de esa clase porque éste hace referencia específica a un objeto.

Como se discutió previamente, a menudo los objetos están conformados por otros objetos más pequeños. Por ejemplo, un carro contiene cuatro objetos de la clase "rueda", dos objetos de la clase "luz delantera", y así sucesivamente. Cada uno de estos objetos hijo tiene las mismas propiedades y métodos de su tipo de clase. Esta relación de objetos padre/hijo es un tema importante a reconocer, particularmente cuando se hace referencia a un objeto individual.

Algunas clases "heredan" características de sus padres. Por ejemplo, en el contexto de CorelDRAW y Corel PHOTO-PAINT, el tipo **Shape** tiene muchos subtipos (o "tipos heredados"), incluyendo **Rectangle**, **Ellipse**, **Curve** y **Text**. Todos estos subtipos pueden hacer uso de los miembros básicos del tipo **Shape**, incluyendo métodos para movimiento o transformación de la forma y ajuste de su color. Sin embargo, los subtipos también tienen sus propios miembros especializados; por ejemplo, un **Rectangle** puede tener esquinas redondeadas, mientras que un **Text** tiene asociada una propiedad **Font**.

¿Qué es una colección?

Una colección es similar a un conjunto de objetos; es un objeto que contiene un grupo de objetos que son similares en tipo. Estos objetos comparten las misma propiedades, métodos y eventos, y son únicamente identificados dentro de la colección por su número indicativo o su nombre. Los objetos de la colección actúan de la misma manera y siempre son plural.

Por ejemplo, `Documents` representa la clase colección **Documents** en CorelDRAW y Corel PHOTO-PAINT. Sin embargo, `Documents.Item (1)` se refiere al primer objeto **Document** en esa colección.

¿Qué es una propiedad?

Una propiedad es como un adjetivo en el que se representa un atributo o característica de un objeto. Las propiedades pueden ser devueltas o estar fijas, o pueden ser de sólo lectura.

La mayoría de las clases tienen propiedades. A manera de ilustración, las propiedades de la clase "carro" son que éste es pequeño, que tiene un motor y cuatro ruedas. Cada instancia de la clase "carro" (es decir, cada objeto en esa clase) tiene también propiedades tales como color, velocidad y número de asientos. Las propiedades de sólo lectura están ajustadas por el diseño de la clase; por ejemplo, el número de ruedas o asientos no varía (normalmente) de carro a carro. Sin embargo, otras propiedades pueden ser cambiadas luego de que el objeto ha sido creado; por ejemplo, la velocidad del carro puede subir y bajar, y, con un poco de ayuda, su color puede ser cambiado.

En el contexto de CorelDRAW y Corel PHOTO-PAINT, los objetos **Document** tienen un nombre, una resolución y unidades de medida horizontal y vertical; las formas individuales tienen propiedades de contorno y propiedades de relleno, así como una posición y un factor de rotación; y los objetos de texto tienen propiedades de texto, las cuales pueden incluir el texto en sí. Por ejemplo, `ActiveDocument.Name` representa la propiedad **Name** de un objeto **Document**; es decir, especifica el nombre del documento activo.

¿Qué es un método?

Un método es como un verbo en el que se representa una acción que puede ser ejecutada por o sobre un objeto. En el ejemplo de una clase "carro", el carro puede ser hecho para ir más rápido y más lento, así que dos métodos para la clase son "acelerar" y "desacelerar".

En el contexto de CorelDRAW y Corel PHOTO-PAINT, los documentos tienen métodos para crear nuevas páginas, las capas tienen métodos para crear nuevas formas, y las formas tienen métodos para aplicar transformaciones y efectos. Por ejemplo, `ActiveDocument.Close` representa el método **Close** de un objeto **Document**; es decir, cierra el documento activo.

¿Qué es un evento?

Un evento es como un sustantivo en el que se representa una acción que toma lugar dentro de un objeto. Un evento es activado por una acción, tal como un click de ratón, una tecla presionada o un temporizador de sistema. Un evento puede ser codificado para activar apropiadamente una respuesta en su objeto.

Por ejemplo, el evento `ActiveDocument.AfterSave` activa una acción en el objeto **Document** después de que éste ha sido guardado.

¿Qué es una enumeración?

Una enumeración (llamada también "tipo enumerado") representa un valor fijo en los procedimientos y funciones de la codificación de una macro. Mientras que una variable almacena temporalmente un valor cambiante de datos, el valor de una enumeración no cambia.

¿Qué es una constante?

Una constante es una instancia de una enumeración, y una enumeración agrupa constantes similares.

Por ejemplo, `AddinFilter` es una enumeración, aún cuando ésta contenga varias constantes, incluyendo `AddinFilterNone` y `AddinFilterNew`.

¿Cómo está estructurada la automatización de la codificación?

Sus conocimientos de programación deberán ayudarle a aprender a automatizar CorelDRAW y Corel PHOTO-PAINT, independientemente de su nivel de experiencia con Microsoft Visual Basic for Applications (VBA) o Microsoft Visual Studio Tools for Applications (VSTA).

Esta sección examina los siguientes temas sobre la estructura y sintaxis del lenguaje VBA:

- ¿Cómo se declaran las variables?
- ¿Cómo se construyen las funciones y subrutinas?
- ¿Cómo se finalizan las líneas?
- ¿Cómo se incluyen comentarios?
- ¿Cómo se utilizan los indicadores de memoria y cómo está asignada la memoria?
- ¿Cómo está definido el alcance?
- ¿Cómo son utilizadas las comparaciones y las asignaciones Booleanas ?
- ¿Cómo son utilizados los operadores lógicos y "bitwise"?
- ¿Cómo se proporcionan los cuadros de mensaje y los cuadros de entrada (input box)?
- ¿Cómo se establecen referencias a objetos?
- ¿Cómo se establecen referencias a colecciones?
- ¿Cómo se utilizan atajos a objetos?
- ¿Cómo se proporcionan los manejadores de eventos?

El Editor de Macros le da formato a todo el código VBA por usted (como se discute en "Formateando automáticamente el código" en la página 32). La única forma de personalizar el formateo es cambiar el tamaño de las sangrías.

VBA puede ser utilizado para crear clases orientadas a objetos. Sin embargo, esta función es una característica del lenguaje de programación y por lo tanto no es discutida en detalle en esta documentación,

¿Cómo se declaran las variables?

En VBA, la construcción para la declaración de las variables es como sigue:

```
Dim foobar As Integer
```

Los tipos de datos incorporados son `Byte`, `Boolean`, `Integer`, `Long`, `Single`, `Double`, `String`, `Variant`, y varios otros tipos menos utilizados incluyendo `Date`, `Decimal` y `Object`.

Las variables pueden ser declaradas en cualquier lugar dentro del cuerpo de una función, o en la parte superior del módulo actual. Sin embargo, generalmente es una buena idea declarar una variable antes de que ésta sea utilizada; de lo contrario, el compilador la interpretará como un `Variant`, y pueden ocurrir deficiencias en tiempo de ejecución.

Los Booleanos toman `False` para ser `0` y `True` para ser cualquier otro valor, aunque la conversión de un `Boolean` a un `Long` resulta en `True` siendo convertido a un valor de `-1`.

Para obtener más información acerca de alguno de los tipos de datos incorporados, tecléelo en la ventana **Código** del Editor de Macros, selecciónelo y luego presione **F1**.

Las estructuras de datos pueden ser construidas utilizando la siguiente sintaxis VBA:

```
Public Type fooType
   item1 As Integer
   item2 As String
End Type
Dim myTypedItem As fooType
```

Los elementos dentro de una variable declarada como tipo `fooType` son accesados al utilizar la notación punto:

```
myTypedItem.item1 = 5
```

¿Cómo se declaran las cadenas?

El uso de cadenas es mucho más simple en VBA que en C. En VBA, las cadenas pueden ser sumadas, truncadas, buscadas adelante y atrás, y pasadas como simples argumentos a funciones.

Para sumar dos cadenas en VBA, simplemente utilice el operador de concatenación (`&`) o el operador de adición (`+`):

```
Dim string1 As String, string2 As String
string2 = string1 & " más texto" + " incluso más texto"
```

En VBA hay muchas funciones para la manipulación de cadenas, incluyendo `InStr()`, `Left()`, `Mid()`, `Right()`, `Len()` y `Trim()`.

¿Cómo se declaran las enumeraciones?

Para declarar una enumeración en VBA, utilice la siguiente construcción:

```
Public Enum fooEnum
   ItemOne
   ItemTwo
   ItemThree
End Enum
```

Por defecto, al primer elemento en un tipo enumerado le es asignado un valor de `0`.

¿Cómo se declaran los arreglos?

Para declarar un arreglo en VBA, utilice paréntesis — es decir, los símbolos `(` y `)`:

```
Dim barArray (4) As Integer
```

El valor define el índice del último elemento en el arreglo. Debido a que los índices son base-cero por defecto, hay cinco elementos en la muestra de arreglo anterior (es decir, elementos desde el `0` hasta el `4`, inclusive).

Los arreglos pueden ser redimensionados utilizando `ReDim`. Por ejemplo, el siguiente código VBA añade un elemento extra a `barArray`, pero preserva el contenido existente de los cinco elementos originales:

```
ReDim Preserve barArray (6)
```

Los límites superior e inferior de un arreglo pueden ser determinados en tiempo de ejecución utilizando las funciones `Ubound()` y `LBound()`.

Los arreglos multidimensionales pueden ser declarados separando los índices de dimensión con comas, como en el siguiente ejemplo VBA:

```
Dim barArray (4, 3)
```

¿Cómo se construyen las funciones y subrutinas?

VBA utiliza tanto funciones como subrutinas (o "subs"). Las funciones pueden ser utilizadas para devolver un valor, pero las subrutinas no. En VBA, las funciones y las subs no necesitan ser declaradas antes de ser utilizadas, ni antes de ser definidas. De hecho, las funciones y subs necesitan ser declaradas sólo si realmente existen en sistemas externos DLLs.

Las funciones típicas en un lenguaje tal como Java o C++ pueden estar estructuradas como sigue:

```
void foo( string stringItem ) {
   //El cuerpo de la función va aquí
}
double bar( int numItem ) [ return 23.2; }
```

En VBA, sin embargo, las funciones están estructuradas como en el siguiente ejemplo:

```
Public Sub foo (stringItem As String)
   ´El cuerpo de la subrutina va aquí
End Sub
Public Function bar (numItem As Integer) As Double bar = 23.2
End Function
```

Para forzar a una función o a una sub a salir inmediatamente, se puede utilizar `Exit Function` o `Exit Sub` (respectivamente).

¿Cómo se finalizan las líneas?

En VBA, cada declaración debe existir en su propia línea, pero no se requiere ningún carácter especial para denotar el final de cada línea. (En contraste, muchos otros lenguajes de programación utilizan un punto y coma para separar declaraciones individuales).

Para dividir una declaración VBA larga en dos o más líneas, cada una de las líneas (a excepción de la última línea) debe finalizar con un guión bajo (_) precedido de por lo menos un espacio:

```
newString = fooFunction ("Esta es una cadena", _
                         5, 10, 2)
```

Se pueden combinar varias declaraciones en una sola línea VBA al separarlas con dos puntos:

```
a = 1 : b = 2 : c = a + b
```

Una línea VBA no puede finalizar con dos puntos. Las líneas VBA que finalizan con dos puntos son etiquetas utilizadas por la declaración `Goto`.

¿Cómo se incluyen comentarios?

Los comentarios en VBA — similarmente como en ANSI, C++ y Java — pueden ser creados sólo al final de una línea. Los comentarios inician con un apóstrofe (') y terminan al final de la línea.

Cada línea de un comentario multilínea debe iniciar con su propio apóstrofe en VBA:

```
a = b   ' Esta es una pieza de código realmente interesante
        ' que requiere tanta explicación que se necesitará
        ' dividir el comentario en varias líneas
```

Para poner entre comentarios grandes secciones de código VBA, se utiliza la siguiente sintaxis (similarmente como en C o C++):

```
#If 0 Then     ' Esto es un cero, no es la letra O.
   ' Todo este código será ignorado
   ' por el compilador en tiempo de ejecución.
#End If
```

¿Cómo se utilizan los indicadores de memoria y cómo está asignada la memoria?

VBA no soporta indicadores de memoria estilo C. La asignación de la memoria y la recolección de desechos son automáticas y transparentes, justo como en Java y JavaScript (y algo de código C++).

¿Cómo son pasados los argumentos?

La mayoría de los lenguajes, incluyendo C++ y Java, pasan un argumento a un procedimiento como una copia del original. Si el original debe ser pasado, entonces una de dos cosas puede suceder:

- Un indicador de memoria es pasado dirigiendo el procedimiento al argumento original en memoria.
- Una referencia al argumento original es pasada.

Microsoft Visual Basic (VB) tiene los mismos requerimientos para pasar argumentos. En VB, el pasar una copia del argumento original se le llama "pasada por valor" y el pasar una referencia al original se le llama "pasada por referencia".

Por defecto, los parámetros de funciones y subrutinas son pasados por referencia. Una referencia a la variable original es pasada en el argumento del procedimiento; el cambio del valor de ese argumento, de hecho, cambia también el valor de la variable original. En este sentido, más de un valor puede ser devuelto de una función o subrutina. Para acotar explícitamente el código para indicar que un argumento está siendo pasado por referencia, se puede prefijar el argumento con `ByRef`.

Si se quiere prevenir que un procedimiento cambie el valor de la variable original, se puede forzar el copiado de un argumento. Para hacer esto en VBA, se prefija el argumento con `ByVal`, como se muestra en el ejemplo siguiente. La funcionalidad de `ByRef` y `ByVal` es similar a la habilidad de C y C++ para pasar una copia de una variable, o pasar un indicador a la variable original.

```
Private Sub fooFunc (ByVal int1 As Integer, _
                     ByRef long1 As Long, _
                     long2 As Long) ' Pasada ByRef por defecto
```

En el ejemplo VBA anterior, los argumentos `long1` y `long2` son ambos, por defecto, pasados por referencia. La modificación de uno u otro argumento dentro del cuerpo de la función modifica la variable original; sin embargo, la modificación de `int1` no modifica el original porque éste es una copia del original.

¿Cómo está definido el alcance?

Se puede definir el alcance de un tipo de datos o procedimiento (o incluso de un objeto). Los tipos de datos, las funciones y las subrutinas (y los miembros de clases) que son declarados como `private` son visibles únicamente dentro de ese módulo (o archivo). En cambio, las funciones que son declaradas como `public` son visibles completamente en todos los módulos; sin embargo, puede ser necesario utilizar referencias tituladas totalmente si los módulos están casi fuera de alcance — por ejemplo, si usted está referenciando una función en un proyecto diferente.

A diferencia de C, VBA no utiliza llaves — es decir, los símbolos `{` y `}` — para definir alcance local. El alcance local en VBA está definido por una declaración de definición de apertura de función o subrutina (esto es, `Function` o `Sub`) y una declaración `End` congruente (es decir, `End Function` o `End Sub`). Cualquier variable declarada dentro de la función está disponible únicamente dentro del alcance de la función misma.

¿Cómo son utilizadas las comparaciones y las asignaciones Booleanas?

En Microsoft Visual Basic (VB), las comparaciones Booleanas y las asignaciones Booleanas son ejecutadas al utilizar solo un signo de igual (=):

```
If a = b Then c = d
```

En cambio, muchos otros lenguajes utilizan un doble signo de igual para comparaciones Booleanas y un solo signo de igual para asignaciones Booleanas:

```
if( a == b ) c = d:
```

El siguiente código, el cual es válido en C, C++, Java y JavaScript, es inválido en VBA:

```
if( ( result = fooBar( ) ) == true )
```

El ejemplo anterior sería escrito en VBA como sigue:

```
result = fooBar( )
If result = True Then
```

Para otras comparaciones Booleanas, VBA utiliza los mismos operadores que otros lenguajes (excepto los operadores para "es igual a" y "no es igual a"). Todos los operadores Booleanos de comparación se proporcionan en la siguiente tabla:

Comparación	Operador VBA	Operador estilo C
Es igual a	=	==
No es igual a	<>	!=
Es mayor que	>	>
Es menor que	<	<
Es mayor o igual a	>=	>=
Es menor o igual a	<=	<=

El resultado del uso de un operador Booleano es siempre `True` o `False`.

¿Cómo son utilizados los operadores lógicos y "bitwise"?

En VBA, las operaciones lógicas son ejecutadas al utilizar las palabras clave `And`, `Not`, `Or`, `Xor`, `Imp` y `Eqv`, las cuales ejecutan las operaciones lógicas AND, NOT, OR, Exclusive-OR, implicación lógica y equivalencia lógica (respectivamente). Estos operadores también ejecutan comparaciones Booleanas.

El siguiente código muestra una comparación escrita en C o en un lenguaje similar:

```
if( ( a && b ) || ( c && d ) )
```

Este ejemplo sería escrito como sigue en VBA:

```
If ( a And b ) Or ( c And d ) Then
```

Alternativamente, el código VBA anterior podría ser escrito de la siguiente forma extendida:

```
If ( a And b = True ) Or ( c And d = True ) = True Then
```

La siguiente tabla proporciona una comparación de los cuatro operadores VBA lógicos y bitwise, y los operadores lógicos y bitwise estilo C que son utilizados por C, C++, Java y JavaScript.

Operador VBA	Operador bitwise estilo C	Operador Booleano estilo C
`And`	`&`	`&&`
`Not`	`~`	`!`
`Or`	`|`	`||`
`Xor`	`^`	

¿Cómo se proporcionan los cuadros de mensaje y los cuadros de entrada (input box)?

En VBA, usted puede presentar al usuario mensajes simples utilizando la función `MsgBox`:

```
Dim retval As Long
retval = MsgBox("Click OK si está de acuerdo.", _
                vbOKCancel, "Mensaje Sencillo")
If retval = vbOK Then
  MsgBox "Ha hecho click en  OK.", vbOK, "Afirmativo"
End If
```

También puede obtener cadenas del usuario al utilizar la función `InputBox`:

```
Dim inText As String
inText = InputBox("Ingrese algún texto:", "tecleé aquí")
If Len(inText) > 0 Then
  MsgBox "Usted escribió lo siguiente: " & inText & "."
End If
```

Si el usuario hace click en **Cancelar**, la longitud de la cadena devuelta en `inText` es cero.

Para más información sobre la creación de interfaces de usuario más complejas, vea "Construcción de macros con interfaz amigable" en la página 55.

¿Cómo son referenciados los objetos?

Si quiere crear una referencia a un objeto de modo que pueda tratar esa referencia como una variable (`sh`, en el ejemplo VBA siguiente), usted puede utilizar la palabra clave `Set`.

```
Dim sh As Shape
Set sh = ActiveSelection.Shapes.Item(1)
```

Después de crear esta referencia, usted puede tratarla como si fuera el objeto mismo.

```
sh.Outline.Color.GrayAssign 35
```

Si la selección es cambiada mientras `sh` está aún al alcance, `sh` hace referencia a la forma original de la antigua selección y no es afectada por la nueva selección. No se puede simplemente asignar el objeto a la variable como en el siguiente ejemplo:

```
Dim sh As Shape
sh = ActiveSelection.Shapes.Item(1)
```

Para liberar un objeto, se debe ajustar su valor de referencia a `Nothing`.

```
Set sh = Nothing
```

También puede probar si una variable hace referencia a un objeto válido utilizando la palabra clave `Nothing`.

```
If sh Is Nothing Then MsgBox "sh está des-referenciado."
```

Los objetos no necesitan ser explícitamente liberados. En la mayoría de los casos, VB libera el objeto al momento de la eliminación de la variable cuando usted sale de la función o subrutina.

¿Cómo son referenciadas las colecciones?

Muchos objetos son miembros de colecciones. Una colección es similar a un arreglo, excepto que ésta contiene objetos en lugar de valores. Sin embargo, los miembros de las colecciones pueden ser accesados de la misma forma que en los arreglos. Por ejemplo, una colección que es utilizada frecuentemente en CorelDRAW es la colección de formas en una capa: El objeto `ActiveLayer` hace referencia ya sea a la capa actual o a la capa que está seleccionada en la ventana acoplable **Administrador de Objetos** de CorelDRAW.

CorelDRAW contiene muchas colecciones. Un documento contiene páginas, una página contiene capas, una capa contiene formas, una curva contiene subtrayectos, un subtrayecto contiene segmentos y nodos, un rango de texto contiene líneas y palabras, un grupo contiene formas, y la aplicación contiene ventanas. Todas estas colecciones son manejadas en la misma forma por VBA.

¿Cómo son referenciados los elementos en una colección?

Para referenciar las formas en una capa, la colección de formas para esa capa es utilizada: `ActiveLayer.Shapes`. Para referenciar formas individuales en la colección, se utiliza la propiedad `Item()`. Aquí está un ejemplo VBA para CorelDRAW:

```
Dim sh As Shape
Set sh = ActiveLayer.Shapes.Item(1)
```

La mayoría de los elementos de una colección inician en `1` y se van incrementando. Para la colección CorelDRAW `ActiveLayer.Shapes`, `Item(1)` es el elemento en la "cima" o "al frente" de la capa —en otras palabras, es el elemento que está adelante de todas las otras formas. Además, puesto que cada elemento en la colección `ActiveLayer` es un objeto de tipo `Shape`, se puede referenciar cualquier elemento en VBA simplemente por incorporación del miembro apropiado con notación punto:

```
ActiveLayer.Shapes.Item(1).Outline.ConvertToObject
```

Algunas veces, los elementos individuales tienen nombres. Si el elemento que está buscando tiene un nombre asociado (y usted sabe qué nombre es y en cuál colección está el elemento), usted puede utilizar ese nombre para referenciar el elemento directamente, como en el siguiente ejemplo VBA para CorelDRAW:

```
Dim sh1 As Shape, sh2 As Shape
Set sh1 = ActiveLayer.CreateRectangle(0, 5, 7, 0)
sh1.Name = "myShape"
Set sh2 = ActiveLayer.Shapes.Item("myShape")
```

Además, puesto que un elemento es usualmente el miembro implícito o por defecto de una colección, no es estrictamente requerido. Por esta razón, la última línea del código VBA anterior puede ser reescrita como sigue:

```
Set sh2 = ActiveLayer.Shapes("myShape")
```

¿Cómo son contados los elementos en una colección?

Todas las colecciones tienen una propiedad llamada `Count`. Esta propiedad de sólo lectura nos da el número de miembros en la colección, como en el siguiente ejemplo VBA para CorelDRAW:

```
Dim count As Long
count = ActiveLayer.Shapes.Count
```

El valor devuelto no sólo es el número de elementos en la colección: Puesto que la colección inicia desde `1`, éste es también el índice del último elemento.

¿Cómo son analizados gramaticalmente los elementos en una colección?

Con frecuencia es necesario analizar gramaticalmente todos los miembros de una colección para verificar o cambiar las propiedades de cada elemento.

Al utilizar los miembros `Item()` y `Count`, es sencillo recorrer una colección de elementos. Con cada iteración es posible probar las propiedades del elemento actual, o llamar a sus métodos. El siguiente código VBA para CorelDRAW restringe todas las formas en la capa a que no sean más anchas de diez unidades:

```
Dim I As Long, count As Long
count = ActiveLayer.Shapes.Count
For I = 1 to count
   If ActiveLayer.Shapes.Item(i).SizeWidth > 10 Then
      ActiveLayer.Shapes.Item(i).SizeWidth = 10
   End If
Next I
```

Sin embargo, hay una forma más conveniente de analizar gramaticalmente una colección en VBA. En lugar de utilizar la propiedad `Count` y un bucle **For-Next**, esta técnica utiliza un bucle **For-Each-In**:

```
Dim sh As Shape
For Each sh In ActiveLayer.Shapes
   If sh.SizeWidth > 10 Then
      sh.SizeWidth = 10
   End If
Next sh
```

Si quiere copiar la selección y luego analizarla gramaticalmente cuando ya no está seleccionada, copie la selección dentro de un objeto `ShapeRange`:

```
Dim sr As ShapeRange
Dim sh As Shape
Set sr = ActiveSelectionRange
For Each sh In sr
  ' Hacer algo con cada forma
Next sh
```

¿Cómo se utilizan los atajos de objetos?

Los atajos son proporcionados para algunos objetos frecuentemente accesados. El uso de atajos requiere menos escritura, así que los atajos son más fáciles de utilizar que sus versiones largas. (Además, el uso de atajos puede mejorar la ejecución en tiempo de ejecución debido a que el compilador no necesita determinar cada objeto en una referencia larga de punto-separado).

Para CorelDRAW, un atajo puede ser utilizado por sí solo como una propiedad del objeto `Application` de CorelDRAW. La siguiente tabla proporciona los atajos y sus formas largas para CorelDRAW. (Para una descripción de cualquier elemento, por favor vea la sección "Referencia del Modelo de Objeto" del archivo Ayuda de Macros de CorelDRAW [**draw_om.chm**].)

Atajo	Forma larga
`ActiveLayer`	`ActivePage.ActiveLayer`
`ActivePage`	`ActiveDocument.ActivePage`
`ActiveSelection`	`ActiveDocument.Selection`
`ActiveSelectionRange`	`ActiveDocument.SelectionRange`
`ActiveShape`	`ActiveDocument.Selection.Shapes(1)`
`ActiveView`	`ActiveWindow.ActiveView`
`ActiveWindow`	`ActiveDocument.ActiveWindow`

Para Corel PHOTO-PAINT, un atajo puede ser utilizado por sí solo como una propiedad del objeto `Application` de Corel PHOTO-PAINT. La siguiente tabla proporciona los atajos y sus formas largas para Corel PHOTO-PAINT. (Para una descripción de cualquier elemento, por favor vea la sección "Referencia del Modelo de Objeto" del archivo Ayuda de Macros de Corel PHOTO-PAINT [**pp_om.chm**].)

Atajo	Forma larga
`ActiveLayer`	`ActivePage.ActiveLayer`
`ActiveWindow`	`ActiveDocument.ActiveWindow`

Para CorelDRAW, los siguientes atajos también pueden ser utilizados como miembros de un determinado objeto `Document`:

- `ActiveLayer`
- `ActivePage`
- `ActiveShape`
- `ActiveWindow`

Para Corel PHOTO-PAINT, los siguientes atajos también pueden ser utilizados como miembros de un determinado objeto `Document`:

- `ActiveLayer`
- `ActiveWindow`

Para CorelDRAW, el objeto `Document` también tiene las propiedades `Selection` y `SelectionRange`, las cuales le permiten obtener la selección y el rango de selección (respectivamente) de un documento específico independientemente si ese documento está activo.

¿Cómo se proporcionan los manejadores de evento?

Mientras se están ejecutando, CorelDRAW y Corel PHOTO-PAINT plantean varios eventos a los que las macros pueden responder por medio de manejadores en uno de los siguientes módulos de código:

- **ThisMacroStorage** — para proyectos de macro que están almacenados en archivos Global Macro Storage (GMS).
- **ThisDocument** — para proyectos de macro que están almacenados en documentos.

El objeto **GlobalMacroStorage** es un objeto virtual que representa todos y cada uno de los documentos abiertos. El objeto **GlobalMacroStorage** tiene varios eventos que son activados en el momento que suceda cualquier evento, tal como la apertura, impresión, guardado o cierre del documento (aunque el rango de eventos es de hecho más grande que éste porque cada uno tiene un evento "antes y "después").

Para responder a un evento, se debe proporcionar un manejador de evento — una subrutina en cualquier módulo **ThisMacroStorage** con un nombre específico el cual la aplicación está pre-programada a buscar. Sin embargo, la aplicación verifica todos los módulos **ThisMacroStorage** en todos los proyectos instalados; por esta razón, usted puede crear una solución de evento-dado y distribuirla como un simple archivo de proyecto justo como proporcionaría cualquier otra solución. Cada proyecto puede tener sólo un módulo **ThisMacroStorage**, y éste es creado automáticamente cuando primero se crea el proyecto.

En VBA, se pueden añadir manejadores de evento al módulo **ThisMacroStorage** utilizando el Editor de Macros. Por ejemplo, una solución de macro CorelDRAW puede necesitar responder al cierre de un documento al registrar el cierre en un archivo como parte de un flujo de trabajo-administración de sistema. Para responder a la apertura de un documento, la solución debe responder al evento `OpenDocument` para la clase `GlobalMacroStorage`. Para crear este manejador de evento en VBA, se hace lo siguiente:

- Abrir un módulo **ThisMacroStorage** para editarlo en el Editor de Macros.
- Enseguida, elegir **GlobalMacroStorage** del cuadro de lista **Object** en la parte superior de la ventana de **Código**, y luego elegir **DocumentOpen** del cuadro de lista **Procedure**.

El Editor de Macros crea una nueva subrutina vacía llamada `GlobalMacroStorage_DocumentOpen()` — o, si la subrutina ya existe, el Editor de Macros coloca el cursor dentro de ella. Para luego añadir el nombre del archivo abierto al registro, sólo se necesita escribir algo de código. Para reducir el tamaño del módulo **ThisMacroStorage** se puede asignar esta tarea de evento registrado a una subrutina pública en otro módulo. Esta técnica le permite al intérprete de tiempo de ejecución analizar gramaticalmente más fácil todos los módulos **ThisMacroStorage** cada vez que un evento es activado. El siguiente código VBA ilustra este ejemplo para CorelDRAW:

```
Private Sub GlobalMacroStorage_OpenDocument(ByVal Doc As Document, _
ByVal FileName As String)
Call LogFileOpen(FileName)
End Sub
```

Aquí está una pequeña muestra de los eventos disponibles en CorelDRAW:

Evento	Descripción
`Start`	Se activa cuando el usuario inicia la aplicación.
`DocumentNew`	Se activa cuando un documento es creado; pasa una referencia al documento
`DocumentOpen`	Se activa cuando un documento es abierto; pasa una referencia al documento
`DocumentBeforeSave`	Se activa antes de que un documento sea guardado; pasa el nombre del archivo del documento como un parámetro.
`DocumentAfterSave`	Se activa después de que un documento es guardado; pasa el nombre del archivo del documento como un parámetro.
`DocumentBeforePrint`	Se activa antes de que el cuadro de diálogo **Imprimir** se muestre.
`DocumentAfterPrint`	Se activa después de que un documento se imprime.
`SelectionChange`	Se activa cuando una selección cambia.
`DocumentClose`	Se activa cuando se cierra un documento.
`Quit`	Se activa cuando el usuario sale de la aplicación.

Los manejadores de evento para eventos frecuentes — tales como eventos relacionados con la clase **Shape** — deberán ser tan eficientes como sea posible, para mantener la aplicación en ejecución tan rápido como sea posible.

Iniciando con las macros

Ahora que entiende un poco acerca de la automatización, está listo para iniciar con las macros. Esta sección proporciona un vistazo a las herramientas y funcionalidades de CorelDRAW y Corel PHOTO-PAINT relacionadas con las macros.

Esta sección contiene los siguientes temas:

- Configurando las características de automatización.
- Utilizando las barra de herramientas Macros.
- Utilizando la ventana acoplable Administrador de Macros.
- Utilizando el Administrador de Adiciones.
- Utilizando el Editor de Macros.
- Utilizando el Editor VSTA.

Configuración de las características de automatización

Antes de que usted pueda desarrollar y ejecutar macros en CorelDRAW Graphics Suite X5, necesitará configurar las características de automatización para VBA y VSTA.

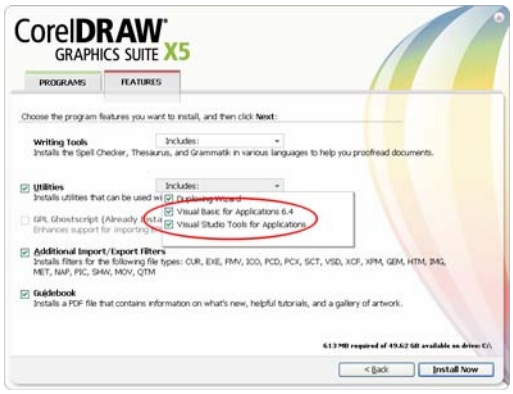
Cuando usted ejecuta una "instalación típica" de CorelDRAW Graphics Suite X5, las características VBA y VSTA son instaladas por defecto. Sin embargo, se pueden instalar manualmente si es necesario. Usted puede especificar varias opciones para VBA.

- Instalar las características VBA y VSTA.
- Especificar opciones VBA.

Para instalar las características VBA y VSTA

1. Inserte el disco de instalación en su computadora.

   Si el asistente de instalación no se inicia automáticamente, localice y ejecute el archivo **Setup.exe** en el disco de instalación.
2. Siga las instrucciones en pantalla para la modificación del software.
3. En la página **Características** del instalador, habilite las siguientes casillas de verificación de la ficha **Utilidades**:
   - **Visual Basic for Applications 6.4**
   - **Visual Studio Tools for Applications**



En el apartado ***Utilidades****, habilite las casillas de verificación* ***Visual Basic for Applications 6.4 y Visual Studio Tools for Applications***

Para especificar opciones VBA

1. Haga click en **Herramientas** » **Opciones**.
2. En la lista de categorías **Espacio de Trabajo**, haga click en **VBA**.
3. En el área **Seguridad**, especifique cómo controlar el riesgo de ejecutar macros maliciosas haciendo click en **Opciones de seguridad...**

   Si quiere pasar por alto esta característica de seguridad, habilite la casilla de verificación **Confiar en todos los módulos GMS instalados** y luego continúe en el paso 6.
4. En la pestaña **Nivel de seguridad**, habilite una de las siguientes opciones:
   - **Muy alto** — permite ejecutar las macros instaladas en ubicaciones de confianza. Las demás macros firmadas y sin firmar se deshabilitarán.
   - **Alto** — sólo se ejecutarán las macros firmadas que procedan de fuentes de confianza. Las macros sin firmar se deshabilitan automáticamente.
   - **Medio** — puede elegir ejecutar o no macros, incluso si son potencialmente peligrosas.
   - **Bajo (no recomendado)** — permite ejecutar todas las macros potencialmente inseguras. habilite esta opción si tiene un antivirus instalado o si ha comprobado la seguridad de todos los documentos que abre.
5. En la pestaña **Editores de confianza**, revise cuáles editores de macros son confiables. Haga click en **Ver** para mostrar detalles del editor de la macro, o haga click en **Quitar** para borrar de la lista el editor de macro seleccionado.

   Si lo desea, puede habilitar o deshabilitar la casilla de verificación **Confiar en el acceso a proyectos de visual basic** del editor de macros seleccionado.
6. Deshabilite la casilla de verificación **Demorar carga de VBA** si quiere cargar la característica VBA al inicio.

Utilizando la barra de herramientas Macros

CorelDRAW y Corel PHOTO-PAINT presentan una barra de herramientas **Macros** que le permite acceder a las funcionalidades comunes de las macros.

Utilizando la barra de herramientas Macros en CorelDRAW

La barra de herramientas **Macros** en CorelDRAW proporciona fácil acceso a varias características relacionadas con las macros, tal como el Editor de Macros

*La barra de herramientas **Macros** en CorelDRAW*

La barra de herramientas **Macros** presenta los siguientes botones:

- Botón **Administrador de Macros** — abre la ventana acoplable **Administrador de Macros**.
- Botón **Ejecutar Macro** — ejecuta una macro.
- Botón **Editor de Macros** — abre el Editor de Macros.
- Botón **Desactivar acciones de la aplicación** — alterna el Editor de Macros entre su modo de diseño y ejecución de macros.
- Botón **Iniciar Grabación** — graba una macro.
- Botón **Pausar Grabación** — pausa la grabación de una macro.
- Botón **Detener Grabación** — detiene la grabación de una macro.

Para mostrar la barra de herramientas **Macros** en CorelDRAW, haga click en **Ventana** » **Barras de herramientas** » **Macros**. Una marca de verificación junto al comando indica que la barra está mostrada.

Utilizando la barra de herramientas Macros en Corel PHOTO-PAINT

La barra de herramientas **Macros** en Corel PHOTO-PAINT proporciona fácil acceso a algunas características relacionadas con las macros, tal como el Editor de Macros

*La barra de herramientas **Macros** en Corel PHOTO-PAINT*

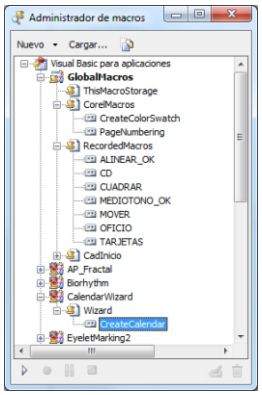
La barra de herramientas **Macros** presenta los siguientes botones:

- Botón **Administrador de Macros** — abre la ventana acoplable Administrador de Macros.
- Botón **Ejecutar Macro** — ejecuta una macro.
- Botón **Editor de Macros** — abre el Editor de Macros.
- Botón **Desactivar acciones de la aplicación** — alterna el Editor de Macros entre su modo de diseño y ejecución de macros.

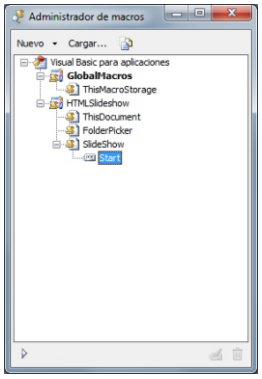
Para mostrar la barra de herramientas **Macros** en Corel PHOTO-PAINT, haga click en **Ventana** » **Barras de herramientas** » **Macros**. Una marca de verificación junto al comando indica que la barra ya está mostrada.

Utilizando la ventana acoplable Administrador de Macros

En CorelDRAW y Corel PHOTO-PAINT, las macros son almacenadas en módulos de código, los cuales son almacenados en proyectos de macro. La ventana acoplable **Administrador de Macros** en CorelDRAW y Corel PHOTO-PAINT proporciona una lista de todos los proyectos de macro existentes, más los módulos de código y macros almacenados en ellos. Se puede utilizar la ventana acoplable **Administrador de Macros** para ejecutar varias tareas relacionadas con proyectos de macro, módulos de código y macros (como se explica en "Creación de macros" en la página 43).



*La ventana acoplable **Administrador de Macros** en CorelDRAW, con una macro seleccionada*



*La ventana acoplable **Administrador de Macros** en Corel PHOTO-PAINT, con una macro seleccionada*

Para mostrar la ventana acoplable **Administrador de Macros**, opte por una de las siguientes opciones:

- Click en **Herramientas** » **Macros** » **Administrador de macros**.
- Click en el botón **Administrador de macros** de la barra de herramientas **Macros**.

Utilizando el Administrador de Adiciones (Add-ins)

Un add-in es un módulo separado que extiende la funcionalidad de una aplicación. El Administrador de Adiciones para CorelDRAW y Corel PHOTO-PAINT muestra una lista de todos los add-ins registrados por la aplicación.

Para abrir el Administrador de Adiciones, haga click en **Herramientas** » **Macros** » **Administrador de Adiciones**. Usted puede utilizar el Administrador de Adiciones para ejecutar varias tareas relacionadas con los add-ins.

Se puede utilizar un add-in registrado haciendo click en **Herramientas** » **Macros** » **Adiciones** y luego eligiendo el add-in.

Utilizando el Editor de Macros

CorelDRAW y Corel PHOTO-PAINT proporcionan un ambiente de desarrollo integrado (IDE, por sus siglas en inglés) para la creación de proyectos de macro VBA. Llamado el Editor de Macros, este IDE es similar al que se incluye en la versión completa de Visual Basic. Se puede utilizar el Editor de Macros para ejecutar varias tareas relacionadas con las macros VBA, tales como las siguientes:

- Explorar el contenido de proyectos de macro VBA.
- Desarrollar y depurar código de macro VBA.
- Ajustar propiedades de objetos para macros VBA.
- Crear cuadros de diálogo o "formas" para macros VBA.


No se puede utilizar el Editor de Macros para compilar archivos de programas ejecutables (EXE).

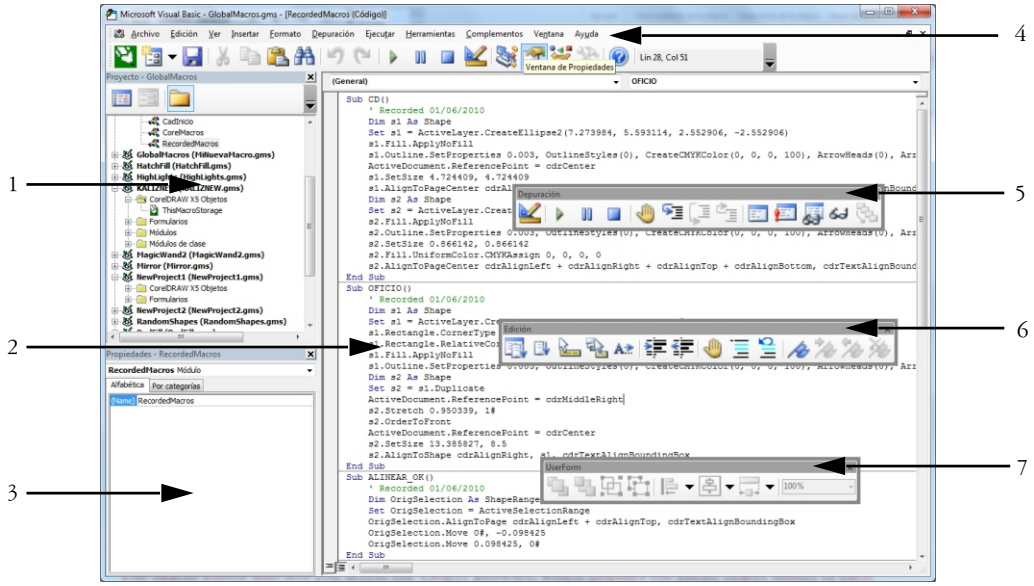
El Editor de Macros presenta tres áreas principales:

- Explorador de Proyectos — permite explorar los proyectos de macro y su contenido. Para más información, vea "Utilizando el Explorador de Proyectos" en la página 30.
- Ventana **Código** — permite trabajar con código de macro. Para más información, vea "Utilizando la ventana Código" en la página 31.
- Ventana **Propiedades** — muestra un listado de todas las propiedades editables de un objeto seleccionado. Para más información, vea "Utilizando la ventana Propiedades" en la página 35.

El Editor de Macros también destaca cuatro barras de herramientas principales:

- Barra de Herramientas **Estándar** — es la barra de herramientas predeterminada.
- Barra de Herramientas **Depurar** — contiene botones para tareas comunes de depuración.
- Barra de Herramientas **Editar** — contiene botones para tareas comunes de edición de código.
- Barra de Herramientas **UserForm** —contiene botones específicos para diseñar cuadros de diálogo.

Para más información sobre estas barras de herramientas, vea "Utilizando las barras de herramientas del Editor de Macros" en la página 35.



El Editor de Macros presenta lo siguiente: 1) Explorador de Proyectos; 2) Ventana de ***Código****; 3) Ventana* ***Propiedades****; 4) Barra de Herramientas* ***Estándar****; 5) Barra de Herramientas* ***Depuración****; 6) Barra de Herramientas* ***Edición****; 7) Barra de Herramientas* ***UserForm***

El Editor de Macros también le permite acceder al Explorador de Objetos, el cual muestra el modelo de objeto completo de cada componente referenciado y de CorelDRAW o Corel PHOTO-PAINT. Para más información vea "Utilizando el Explorador de Objetos" en la página 36.

Puede acceder al Editor de Macros desde Corel DRAW o Corel PHOTO-PAINT. Aunque el Editor de Macros se abre en una ventana separada, éste corre dentro del proceso de su aplicación fuente. Para mostrar el Editor de Macros, haga una de estas cosas:

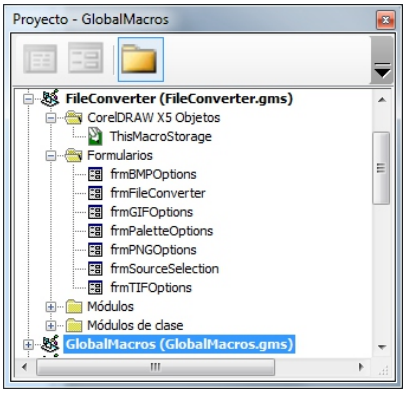
- Click en **Herramientas** » **Macros** » **Editor de Macros** en el menú principal de la aplicación.
- Click en el botón **Editor de Macros** en la barra de herramientas **Macros**.
- Click derecho en **Visual Basic para Aplicaciones** en la ventana acoplable **Administrador de Macros** y luego click en **Mostrar IDE**.
- Presionar **Alt** + **F11**.

Para alternar entre el Editor de Macros y la aplicación, utilice la barra de tareas de Windows o presione **Alt** + **F11** o **Alt** + **Tab**.

Para obtener información más detallada sobre la construcción de procedimientos de código, por favor vea el archivo de Ayuda de Microsoft Visual Basic for Applications, el cual está disponible en el menú **Ayuda** del Editor de Macros.

Utilizando en Explorador de Proyectos

El Explorador de Proyectos es esencial para navegar por los proyectos de macros y su contenido: documentos y objetos, formas, módulos y módulos de clase (o "clases").



El Explorador de Proyectos

Cada tipo de componente en el Explorador de Proyectos tiene un icono asignado a él:

Icono	Elemento
	proyecto de macro
	carpeta
	documento u objecto (CorelDRAW)
	documento u objecto (Corel PHOTO-PAINT)
	forma
	módulo
	módulo de clase (o “clase”)

Para mostrar u ocultar el Explorador de Proyectos, haga una de las siguientes cosas:

- Click en **Ver** » **Explorador de Proyectos**.
- Click en el botón **Explorador de Proyectos**  en la barra de herramientas **Estándar**.
- Presione **Ctrl + R**.

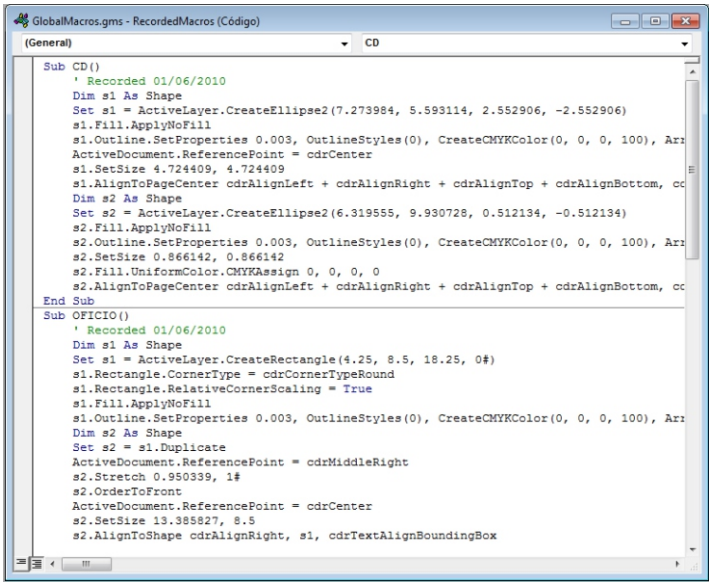
Utilizando la ventana Código

La ventana **Código** es donde usted pasará la mayoría de su tiempo cuando trabaje con macros. Un editor de código estándar al estilo de Microsoft Visual Studio, la ventana **Código** le permite hacer lo siguiente:

- Dar formato a código automáticamente.
- Colorear sintaxis automáticamente.

- Verificar automáticamente la sintaxis.
- Saltar a definiciones.
- Utilizar listas contextuales automáticas y completamiento automático.

Si ya está familiarizado con algún editor de Microsoft Visual Studio, la ventana **Código** le será completamente familiar.



*La ventana **Código***

Para mostrar la ventana **Código**, haga una de las siguientes cosas:

- Click en **Ver** » **Código**.
- Presione **F7**.

Formateo automático de código

El Editor de Macros formatea automáticamente el código por usted. Incluso la capitalización de palabras clave, funciones, subrutinas y variables es realizada por el Editor de Macros, independientemente de lo que usted escriba. No se puede personalizar el formateo del código, aunque se puede ajustar el sangrado de cada línea, así como la colocación de líneas de quiebre.

Si utiliza el valor devuelto cuando se llama a una función, los paréntesis alrededor de los parámetros son obligatorios (igual que en la mayoría de los lenguajes de programación modernos):

```
a = fooFunc (b, c)
```

Sin embargo, si el valor devuelto de una llamada de función está siendo descartado, o si se llama a una subrutina, los paréntesis deben descartarse (a diferencia de muchos otros lenguajes):

```
barFunc d, e
fooBarSub f
```

Si prefiere ver siempre los paréntesis, utilice la palabra clave `Call` antes de llamar a la función o a la subrutina:

```
Call barFunc (d, e)
Call fooBarSub (f)
```

Coloración automática de sintaxis

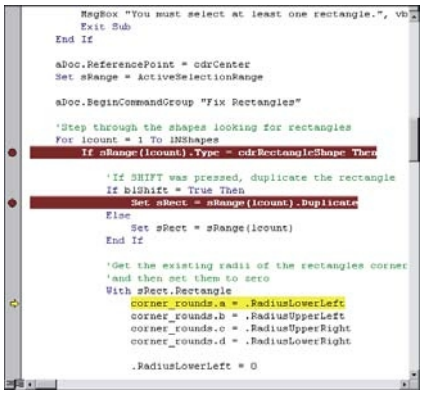
Al escribir código en la ventana **Código**, el Editor de Macros colorea cada palabra de acuerdo a su clasificación.

Color de palabra	Clasificación
Azul	Automatización de palabra clave o declaración de programación
Verde	Comentario
Negra	Todo el texto restante

La ventana **Código** también utiliza las siguientes técnicas de colorización:

Técnica de colorización	Clasificación
Texto rojo	Línea de código conteniendo errores
Texto blanco sobre fondo azul	Texto seleccionado
Texto resaltado en amarillo	Línea donde la ejecución está pausada para depuración
Texto blanco sobre fondo rojo y un punto rojo en el margen izquierdo	Punto de ruptura establecido para propósitos de depuración Para más información, vea "Ajuste de puntos de ruptura" en la página 53
Punto azul en el margen izquierdo	Establece marcador de libro en el código

Estas técnicas de colorización de sintaxis hacen que el código sea mucho más fácil de leer.



Coloreo y resalte de sintaxis

Los puntos de ruptura y los marcadores de libro se pierden cuando se sale de la aplicación.

El Editor de Macros le permite modificar los colores predeterminados para resaltar la sintaxis. Haga click en **Herramientas** » **Opciones** y elija sus ajustes en la pestaña **Formato del Editor**.

Verificación automática de sintaxis

Cada vez que usted mueve el cursor fuera de una línea de código, el Editor de Macros verifica la sintaxis del código en esa línea; si se encuentra algún error, la línea es coloreada y aparece un mensaje de advertencia. Esta comprobación en

tiempo real es útil (particularmente cuando se está aprendiendo a programar macros) porque nos revela muchos posibles errores en el código antes de tener que ejecutarlo.

El Editor de Macros le permite deshabilitar los mensajes automáticos de advertencia. Haga click en **Herramientas** » **Opciones**, click en la pestaña **Editor** y luego deshabilite la casilla de verificación **Comprobación de sintaxis automática**. A pesar de que el Editor de Macros aún verifica la sintaxis y colorea de rojo las líneas erróneas, éste deja de mostrar una advertencia cuando usted pega texto desde otra línea de código.

Saltando a definiciones

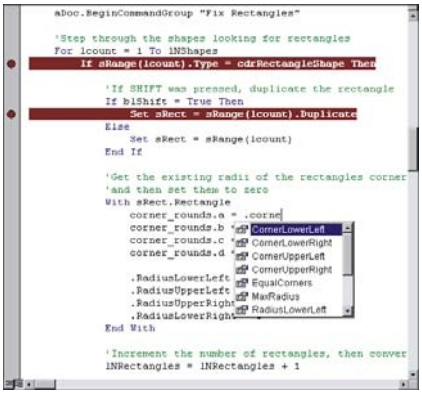
El Editor de Macros le permite saltar directamente a la definición de una variable, función u objeto.

Definición deseada	Procedimiento	Destino
Variable	Click derecho en la variable en la ventana **Código** y luego click en **Definición**.	La definición de la variable en el Código.
Función	Click derecho en la función en la ventana **Código** y luego click en **Definición**.	La definición de la función en el Código.
Objecto	Click derecho en el objeto en la ventana **Código** y luego click en **Definición**.	La definición del objeto en el Explorador de Objetos.

Para regresar a donde usted solicitó la definición, haga click derecho en cualquier parte de la ventana **Código** y luego haga click en **Última posición**.

Utilizando listas contextuales automáticas para auto-completado

El Editor de Macros añade las funciones que usted escribe y las variables que usted define a una lista interna que contiene todas las palabras clave incorporadas y los valores enumerados. Al escribir, el Editor de Macros muestra una lista contextual de palabras que son candidatos válidos para insertar en la posición actual. Esta característica de auto-completado hace más rápido y más conveniente el desarrollo del código.



Una lista automática de auto-completado

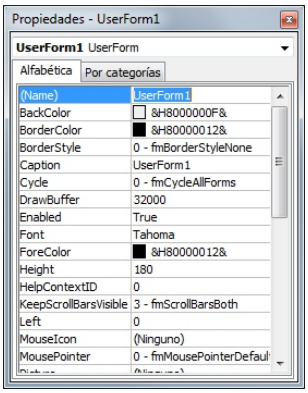
Si usted escribe los primeros caracteres de la palabra que desea utilizar, la lista automática avanza al candidato más cercano que concuerde con esos caracteres. Seleccione la palabra deseada y luego opte por una de estas opciones:

- Escriba el carácter para continuar la palabra (usualmente un espacio, salto de línea, paréntesis, punto o coma).
- Ingrese sólo la palabra presionando **Tab** o **Ctrl + Enter**.

Para forzar a que aparezca el menú emergente, presione **Ctrl + barra espaciadora**; el menú se desplaza a la palabra que más se asemeje a los caracteres que usted ha tecleado. Esta técnica es particularmente útil para llenado de listas de parámetros cuando se llama a una función o subrutina. Si sólo hay una comparación exacta. el Editor de Macros inserta la palabra sin mostrar la lista; para mostrar la lista emergente para la palabra clave seleccionada en cualquier momento sin auto-completarla, presione **Ctrl + J**.

Utilizando la ventana Propiedades

La ventana **Propiedades** muestra una lista de todas las propiedades editables para el objeto seleccionado. Muchos objetos de macro — incluyendo proyectos, módulos y formas (y sus controles) — tienen hojas de propiedades que pueden ser modificadas.



*La ventana **Propiedades**, con las propiedades de una forma mostrada*

La ventana **Propiedades** se actualiza automáticamente cuando se selecciona un objeto, o cuando se cambian las propiedades del objeto seleccionado utilizando otros métodos (por ejemplo, utilizando el ratón para mover y redimensionar controles de formulario).

Para mostrar u ocultar la ventana **Propiedades**, haga una de estas cosas:

- Click en **Ver** » **Ventana Propiedades**.
- Click en el botón **Ventana de Propiedades** en la barra de herramientas **Estándar**.
- Presionar **F4**.

Utilizando las barras de herramientas del Editor de Macros

El Editor de Macros presenta cuatro barras de herramientas — **Estándar**, **Depuración**, **Edición** y **UserForm** — que se pueden utilizar para ejecutar tareas relacionadas con las macros.

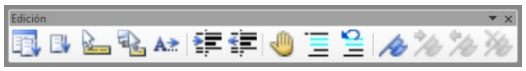
La barra de herramientas **Estándar** es la barra de herramientas predeterminada.

*La barra de herramientas **Estándar***

La barra de herramientas **Depuración** contiene botones para tareas comunes de depuración (como se discute en "Depuración de macros" en la página 51).

*La barra de herramientas **Depuración***

La barra de herramientas **Edición** contiene botones para tareas comunes de edición.



*La barra de herramientas **Edición***

La barra de herramientas **UserForm** contiene botones específicos para diseñar cuadros de diálogo (como se discute en "Diseño de cuadros de diálogo" en la página 59).

*La barra de herramientas **UserForm***

Para mostrar u ocultar una barra de herramientas, haga click en **Ver** » **Barras de herramientas** y luego haga click en el comando correspondiente. Una marca de verificación junto al comando indica que esa barra de herramientas ya está siendo mostrada

Se puede hacer "flotar" una barra de herramientas al arrastrarla de la barra de menú.

Se puede acoplar una barra de herramientas al arrastrarla a la barra de menú

Utilizando el Examinador de Objetos

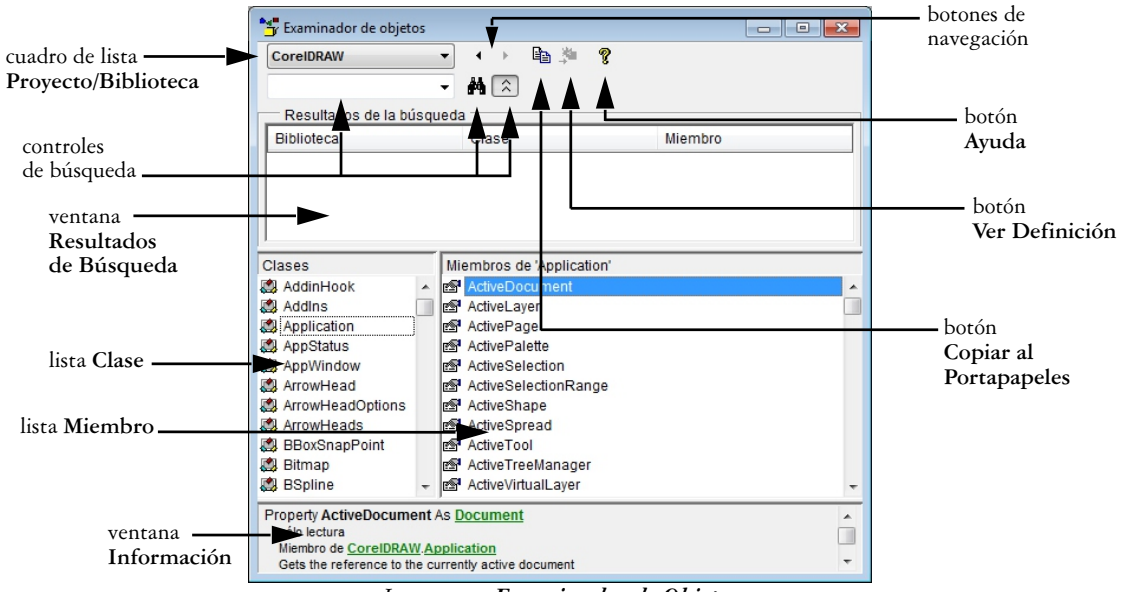
El Examinador de Objetos es una de las herramientas más poderosas proporcionadas por el Editor de Macros. En un formato estructurado fácil de usar, el Examinador de Objetos muestra el modelo de objetos completo de cada componente referenciado y, más importante, de CorelDRAW o Corel PHOTO-PAINT.

Los componentes referenciados incluyen todos los objetos ActiveX u OLE (Object Linking and Embedding) que son utilizados por el proyecto.

La ventana del **Examinador de Objetos** presenta los siguientes elementos:

- Cuadro de lista **Proyecto/Biblioteca** — muestra una lista de todos los componentes referenciados (proyectos y bibliotecas). Al elegir un proyecto o biblioteca de esta lista, se actualiza el Examinador de Objetos para mostrar sólo los elementos de ese proyecto o biblioteca. Generalmente, al mostrarse sólo un proyecto o biblioteca a la vez, se hace más fácil utilizar el Examinador de Objetos.
- Botones de navegación — le permite circular a través de sus selecciones desde el Examinador de Objetos.
- Botón **Copiar al Portapapeles** — copia la selección actual al Portapapeles.
- Botón **Ver Definición** — muestra dónde está definida la selección actual en la ventana **Código**.
- Botón **Ayuda** — muestra un tema de Ayuda para la selección actual. También se puede acceder a este tema de Ayuda presionando **F1**.
- Controles de búsqueda — le permite buscar el proyecto o biblioteca seleccionada por una cadena dada. Para más información, vea "Utilizando controles de búsqueda" en la página 41.
- Ventana **Resultados de Búsqueda** — muestra los resultados de una búsqueda. Para más información, vea "Utilizando controles de búsqueda" en la página 41.

- Lista **Clase** — muestra todos los elementos relacionados con clases para el proyecto o biblioteca seleccionados. Para más información, vea "Utilizando la lista Clase" en la página 37.
- Lista **Miembro** — muestra los miembros de una clase seleccionada. Para más información, vea "Utilizando la lista Miembro" en la página 39.
- Ventana **Información** — muestra información acerca de la clase seleccionada o el miembro clase. Para más información, vea "Utilizando la ventana Información" en la página 40.



*La ventana **Examinador de Objetos***

Para abrir el Examinador de Objetos desde el Editor de Macros, opte por una de las siguientes opciones:

- Click en **Ver** » **Examinador de Objetos**.
- Click en el botón **Examinador de Objetos** en la barra de herramientas **Estándar**.
- Presione **F2**.

Para referenciar el modelo de objeto de otra aplicación, haga click en **Herramientas** » **Referencias**. Los componentes referenciados pueden ser accesados por el código de macro.

Utilizando la lista Clases

Cada proyecto y biblioteca tiene un modelo de objeto que contiene los siguientes elementos relacionados con clases, los cuales son mostrados en la lista **Clase**:

- Valores globales.
- Clases.
- Módulos.
- Tipos.
- Enumeraciones.

Los valores globales se aplican a un proyecto o biblioteca completos, e incluyen miembros individuales de enumeraciones (tales como alineaciones de texto de párrafo, tipos de formas y filtros de importación y exportación).

Las Clases contienen propiedades, métodos y eventos. Para más información, vea "¿Qué es una clase?" en la página 12.

Para documentación de todas las clases disponibles para CorelDRAW o Corel PHOTO-PAINT, vea la sección "Referencia/Clases del Modelo de Objeto" del archivo de Ayuda de Macros para la aplicación.

Los Módulos contienen código de macro.

Para documentación de todos los módulos disponibles para CorelDRAW, vea la sección "Referencia/Módulos del Modelo de Objeto" del archivo de Ayuda de Macros para la aplicación.

Los Tipos son tipos de datos personalizados que complementan los tipos de datos incorporados que son proporcionados por el ambiente de automatización (como se describe en "¿Cómo se declaran las variables?" en la página 14.

Para documentación de todos los tipos disponibles para CorelDRAW, vea la sección "Referencia/Tipos del Modelo de Objeto" del archivo de Ayuda de Macros para la aplicación.

Las Enumeraciones representan valores fijos en los procedimientos y funciones de la codificación para una macro. Para más información, vea "¿Qué es una enumeración?" en la página 13.

Para documentación de todas las enumeraciones disponibles para CorelDRAW o Corel PHOTO-PAINT, vea la sección "Referencia/Enumeraciones del Modelo de Objeto" del archivo de Ayuda de Macros para la aplicación.

Cada tipo de elemento en la lista **Clase** tiene un icono asignado a él:

Icono	Elemento
	valor global
	clase
	módulo
	tipo
	enumeración

Para acceder al tema de Ayuda de un elemento seleccionado, haga click en el botón **Ayuda** o presione **F1**.

Utilizando la lista Miembro

Cuando usted elige un elemento de la lista **Clase**, los miembros de ese elemento aparecen en la lista **Miembro**. Las clases de miembros incluyen las siguientes:

- propiedades
- métodos
- eventos
- constantes

Una propiedad puede ser un tipo simple (tal como un Booleano, integer o string), o puede ser una clase o enumeración de una lista **Clase**. Una propiedad que está basada en una clase de la lista **Clase** hereda todos los miembros de esa clase.

Muchas clases tienen una propiedad predeterminada. La propiedad predeterminada es implícita si no es dado el nombre de la propiedad cuando se obtiene o se ajusta el valor del objeto-padre. Por ejemplo, los tipos colección tienen la propiedad predeterminada **Item**, la cual puede ser indexada; en tales casos, no es necesario especificar la propiedad **Item**. Es decir:

```
ActiveSelection.Shapes.Item(1).Selected = False
```

es lo mismo que su forma corta:

```
ActiveSelection.Shapes(1).Selected = False
```

porque **Item** es la propiedad predeterminada o implícita de una colección de formas.

Para documentación de todas las propiedades disponibles para CorelDRAW o Corel PHOTO-PAINT, vea la sección "Referencia/Propiedades del Modelo de Objeto" del archivo de Ayuda de Macros para la aplicación.

Los Métodos son comúnmente conocidos como "funciones de miembro" — funciones que una clase puede ejecutar sobre sí misma. Un buen ejemplo es el método **Move** de una clase **Shape** en CorelDRAW, el cual mueve una forma utilizando un vector [x, y].

El siguiente código mueve las formas seleccionadas 2 unidades de medida hacia la derecha y 3 unidades de medida hacia arriba:

```
ActiveSelection.Move 2, 3
```

Si el valor devuelto de una función no es utilizado, la llamada a la función no lleva paréntesis alrededor de la lista de argumentos a menos que sea utilizada la palabra clave `Call`.

Para documentación de todos los métodos disponibles para CorelDRAW o Corel PHOTO-PAINT, vea la sección "Referencia/Métodos del Modelo de Objeto" del archivo de Ayuda de Macros para la aplicación.

Los Eventos están asociados con algunas clases. Se puede configurar que un manejador de evento sea llamado cuando el evento ocurra en la aplicación; esta funcionalidad le permite desarrollar sofisticadas aplicaciones que respondan automáticamente a lo que está sucediendo dentro de la aplicación. Comúnmente, los manejadores de evento incluyen los eventos **BeforePrint**, **BeforeSave**, **PageActivate**, **SelectionChange** y **ShapeMove** de la clase **Document**.

Para documentación de todos los eventos disponibles para CorelDRAW o Corel PHOTO-PAINT, vea la sección "Referencia/Eventos del Modelo de Objeto" del archivo de Ayuda de Macros para la aplicación.

Las constantes mostradas en la lista **Miembro** son miembros de enumeraciones o están definidas como públicas (`public`) en un módulo. Las enumeraciones agrupan elementos relacionados de una lista cerrada — tal como los tipos de formas CorelDRAW, filtros de importación/exportación y alineaciones — para utilizarse donde quiera que un valor entero sea requerido.

Muchas constantes CorelDRAW y Corel PHOTO-PAINT comienzan con **cdr** (por ejemplo, **cdrEPS** y **cdrLeftAligment**), mientras que otras comienzan con **clr**, **cui**, **pdf**, **pnt** o **prn**. Visual Basic también tiene sus propias constantes, incluyendo aquellas para teclas pulsadas (tal como **vbKeyEnter**) y para botones de cuadros de diálogo (tal como **vbOK**).

Para documentación de todas las constantes disponibles para CorelDRAW o Corel PHOTO-PAINT, vea la sección "Referencia/Constantes del Modelo de Objeto" del archivo de Ayuda de Macros para la aplicación.

Cada tipo de elemento en la lista **Miembro** tiene un icono asignado a él:

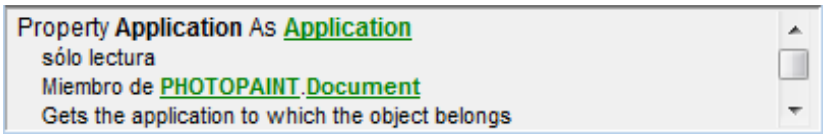
Icono	Elemento
	propiedad
	propiedad por defecto
	método
	evento
	constante

Para acceder al tema de Ayuda de un elemento seleccionado, haga click en el botón **Ayuda** o presione **F1**.

Utilizando la ventana Información

La ventana **Información** proporciona información acerca de la clase seleccionada o miembro clase. Esta información incluye lo siguiente:

- un prototipo del elemento.
- una indicación de si el elemento es una propiedad de sólo lectura.
- el padre del elemento.
- una descripción corta del elemento.



La ventana Información para la propiedad ***Document.Application*** *en Corel PHOTO-PAINT*

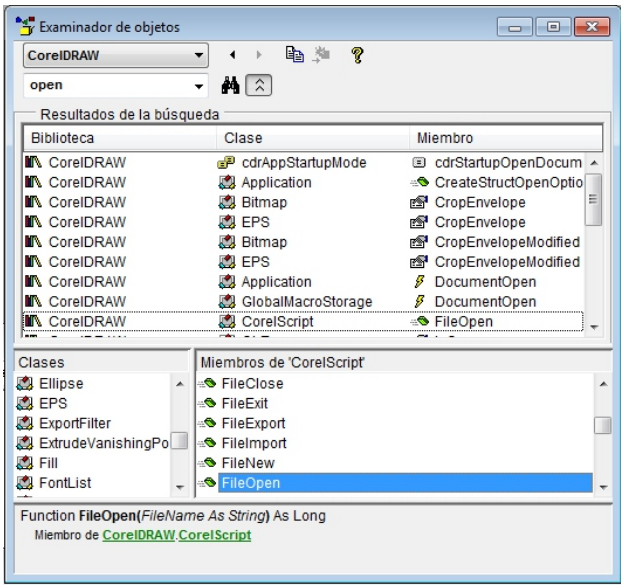
La ventana **Información** proporciona hiperenlaces para todos los tipos y clases referenciados que están definidos dentro del modelo de objeto actual. Por ejemplo, la información para la propiedad **Document.Application** en Corel PHOTO-PAINT (ver la figura precedente) incluye los siguientes hiperenlaces:

- **Application** — accede a la clase **Application**, ya que **Application** es tanto el tipo de la propiedad **Document.Application** como una clase en la biblioteca **PHOTOPAINT**.
- **PHOTOPAINT** — accede a la clase de la biblioteca **PHOTOPAINT**, la cual contiene todas las clases en el modelo de objeto Corel PHOTO-PAINT.
- **Document** — accede a la clase **Document**, la cual es el padre de la propiedad **Application**.

Cuando la ventana **Información** no es lo suficientemente alta para mostrar todo su contenido, se proporciona una barra de desplazamiento. Para incrementar la altura de la ventana **Información**, arrastre el borde superior de la ventana hacia arriba.

Utilizando los controles de búsqueda

Los controles de búsqueda le permiten buscar el proyecto o biblioteca seleccionada para una cadena dada. La búsqueda es útil cuando usted puede recordar sólo parcialmente el nombre de una clase o miembro de clase, o cuando quiere encontrar clases y miembros que tengan nombres similares (tales como aquellos que contengan la palabra "open").



Búsqueda de un modelo de objeto

Para buscar las clases y los miembros del modelo de objeto seleccionado, escriba algo dentro de la caja **Búsqueda** y luego haga click en el botón **Búsqueda** . La ventana **Resultados de la búsqueda** muestra, en orden alfabético, todas las correspondencias. Al hacer click en una de las correspondencias avanza la lista **Clase** y la lista **Miembro** a ese elemento y muestra la ventana **Información** para ese elemento.

Los nombres de clases concordantes tienen una columna **Miembro** vacía en la ventana **Resultados de la búsqueda**

Para ocultar la ventana **Resultados de la búsqueda**, haga click en el botón **Ocultar resultados de la búsqueda** .

Utilizando el Editor VSTA

CorelDRAW y Corel PHOTO-PAINT proporcionan un ambiente de desarrollo integrado (IDE, por sus siglas en inglés) para la creación de proyectos de macro VSTA. Llamado el Editor VSTA, este IDE es similar al Editor de Macros (el cual es el IDE para proyectos de macro VBA en CorelDRAW y Corel PHOTO-PAINT). Se puede utilizar el Editor VSTA para ejecutar varias tareas relacionadas con macros VSTA.

Por defecto, CorelDRAW Graphics Suite X5 crea carpetas VSTA en la siguiente ruta:

- para CorelDRAW: **Mis Documentos\Corel\VSTA\CorelDRAW**
- para Corel PHOTO-PAINT: **Mis Documentos\Corel\VSTA\Corel PHOTO-PAINT**

Asegúrese cargar las adiciones VSTA de la siguiente ubicación:

- para CorelDRAW: **Mis Documentos\Corel\VSTA\CorelDRAW\Addins**
- para Corel PHOTO-PAINT: **Mis Documentos\Corel\VSTA\Corel PHOTO-PAINT\Addins**

Se puede abrir el Editor VSTA desde CorelDRAW o Corel PHOTO-PAINT. Aunque el Editor VSTA se abre en una ventana separada, éste se ejecuta dentro del proceso de su aplicación fuente. Para abrir el Editor VSTA, haga una de las siguientes cosas:

- Click en **Herramientas** » **Macros** » **Editor VSTA** en el menú principal de la aplicación.
- Presione **Alt** + **Shift** + **F12**.

Para alternar entre el Editor VSTA y la aplicación, utilice la barra de tareas de Windows, o presione **Alt** + **Shift F12** o **Alt** + **Tab**.

Para información más detallada sobre VSTA y su ambiente de programación, consulte el menú **Ayuda** en el Editor VSTA.

Creación de macros

Ahora que está familiarizado con el concepto de automatización y con las herramientas y funcionalidades relacionadas con las macros de CorelDRAW y Corel PHOTO-PAINT, está listo para crear macros.

Esta sección contiene los siguientes temas:

- Creación de proyectos de macro.
- Escritura de macros.
- Grabación de macros.
- Ejecución de macros.
- Depuración de macros.

Creación de proyectos de macro

El proceso de creación de macros inicia con la creación de un proyecto de macro. Un proyecto de macro puede ser creado mediante una de las siguientes maneras:

- como un archivo Global Macro Storage (GMS) o archivo de "proyecto"
- en un documento

Para mejores resultados en el almacenamiento y distribución de un proyecto de macro, es muy recomendable que utilice un archivo GMS. Los archivos GMS se almacenan en la carpeta **GMS** de la aplicación, la cual está instalada en la siguiente ruta:

- para proyectos de macro que se remesan con CorelDRAW Graphics Suite X5: X**Archivos de programa\Corel\CorelDRAW Graphics Suite X5\<*aplicación*>** (donde **X:** es la unidad y **Archivos de programa\Corel\CorelDRAW Graphics Suite X5** es la ruta donde se instaló el software, y donde ***<aplicación>*** es la subcarpeta de la aplicación).
- para proyectos de macro creados por el usuario en Windows 7 y Windows Vista®: **X:\Usuarios*<nombre de usuario>*\AppData\Roaming\Corel*<aplicación>*** (donde **X:** es la unidad donde se instaló el software, ***<nombre de usuario>*** es el nombre del usuario, y ***<aplicación>*** es la subcarpeta de la aplicación).
- para proyectos de macro creados por el usuario en Windows XP: **X:\Documents and Settings*<nombre de usuario>*\Application Data\Corel\CorelDRAW Graphics Suite X5*<aplicación>*** (donde **X:** es la unidad donde se instaló el software, ***<nombre de usuario>*** es el nombre del usuario, y ***<aplicación>*** es la subcarpeta de la aplicación).

La ventana acoplable **Administrador de Macros** proporciona las herramientas básicas para trabajar con proyectos de macro. Para acceder a herramientas más avanzadas, se utiliza el Editor de Macros (para proyectos de macro VBA) o el Editor VSTA (para proyectos de macro VSTA).

En el Editor de Macros, un proyecto de macro VBA está dividido en cuatro tipos de componentes, los cuales son mostrados como las siguientes carpetas en el Explorador de Proyectos (vea "Utilizando el Explorador de Proyectos" en la página 30):

- **Objetos CorelDRAW X5 u objetos Corel PHOTO-PAINT X5** — contienen un elemento único que se utiliza comúnmente para manejo de evento: **ThisMacroStorage** para proyectos de macro basados en GMS, o **ThisDocument** para proyectos de macro basados en documentos. Para código normal, este módulo no se utiliza.
- **Formas** — contiene cuadros de diálogo e interfaces de usuario personalizados, además del código para controlarlos.
- **Módulos** — contiene módulos de código, para almacenamiento general de código y macros.
- **Módulos de Clase** — contiene módulos de Clase de Visual Basic orientados a objetos (los cuales no son discutidos en esta documentación)

En el Editor de Macros no se puede mover un componente de una carpeta a otra dentro del mismo proyecto. Sin embargo, se puede arrastrar un componente a otro proyecto para hacer una copia de él ahí.

Para procedimientos detallados sobre la creación de proyectos de macros, vea los siguientes temas:

- Crear un proyecto de macro.
- Añadir una cuadro de diálogo a un proyecto de macro.
- Añadir un módulo de código a un proyecto de macro.
- Añadir un módulo de Clase a un proyecto de macro.

Para crear un proyecto de macro

En la ventana acoplable **Administrador de Macros**, hacer una de las siguientes cosas:

- Hacer click en **Nuevo** y luego click en **Nuevo proyecto de macro**.
- Hacer click derecho en **Visual Basic para Aplicaciones** en la lista, luego hacer click en **Nuevo proyecto de macro**.

Los nombres de los proyectos deben seguir las convenciones normales de nombramiento de variables: Deben comenzar con un carácter alfabético y no deben contener espacios ni caracteres especiales, a menos que sea un guión bajo (_).

También se puede

Cargar un proyecto de macro	Opte por una de las siguientes cosas: • Click en **Cargar...** y luego elegir el proyecto • Click derecho en **Visual Basic para Aplicaciones** en la lista, luego click en **Cargar macro de proyecto** y por último elegir el proyecto.
Renombrar un proyecto de macro	Click derecho en el proyecto de la lista y luego click en **Cambiar nombre**. También se puede renombrar una macro dentro del Editor de Macros. Elegir la macro en el Explorador de Proyectos y luego editar el valor (**Name**) en la ventana **Propiedades** y presionar **Enter** para confirmas los cambios.

También se puede

Copiar un proyecto de macro basado en GMS.	Click derecho en algún proyecto de la lista, click en **Copiar en...** y luego elegir la ubicación destino para la copia. **NOTA:** No se puede copiar un proyecto de macro basado en documento. Tales proyectos son almacenados dentro de un documento y no pueden ser administrados separadamente de ese documento.
Descargar un proyecto de macro basado en GMS.	Click derecho en algún proyecto de la lista y luego click en **Descargar proyecto de macro**. **NOTA:** No se puede cerrar un proyecto de macro basado en documento simplemente cerrando el documento en el cual está almacenado.


Algunos proyectos de macro están bloqueados y no pueden ser modificados.

Para añadir un cuadro de diálogo a un proyecto de macro

1 En el Explorador de Proyectos del Editor de Macros, hacer click derecho en el proyecto.
2 Hacer click en **Insertar** » **UserForm**.
Se añade una forma a la carpeta **Formularios** del proyecto.


Algunos proyectos de macro están bloqueados y no pueden ser modificados.


Para más información, vea "Proporcionando cuadros de diálogo para macros" en la página 57.

Para añadir un módulo de código a un proyecto de macro

- Hacer una de las cosas siguientes:
  - En la ventana acoplable **Administrador de Macros**, click en algún proyecto de la lista, click en **Nuevo** y luego click en **Nuevo Módulo**.
  - En la ventana acoplable **Administrador de Macros**, click derecho en algún proyecto de la lista y luego click en **Nuevo Módulo**.
  - En el Explorador de Proyectos del Editor de Macros, click derecho en el proyecto y luego click en **Insertar** » **Módulo**.

También se puede

Mostrar u ocultar todos los módulos de código en la ventana acoplable **Administrador de Macros**.	En la ventana acoplable **Administrador de Macros**, hacer click en el botón Modo sencillo .
Editar un módulo de código.	En la ventana acoplable **Administrador de Macros**, hacer una de las siguientes cosas: • Click en algún módulo de la lista y luego click en el botón **Editar** . • Click derecho en algún módulo de la lista y luego click en **Editar**. El módulo de código se abre en el Editor de Macros.
Renombrar un módulo de código.	En la ventana acoplable **Administrador de Macros**, hacer click derecho en algún módulo de la lista y luego click en **Renombrar**.

También se puede	
Eliminar un módulo de código	En la ventana acoplable **Administrador de Macros**, hacer una de las siguientes cosas: • Click en algún módulo de la lista y luego click en el Botón **Eliminar** . • Click derecho en algún módulo de la lista y luego click en **Eliminar**.

Algunos proyectos de macro están bloqueados y no pueden ser modificados.

Para añadir un módulo de clase a un proyecto de macro

1 En el Explorador de Proyectos del Editor de Macros, hacer click derecho en el proyecto.
2 Hacer click en **Insertar** » **Módulo de clase**.
Se añade un nuevo módulo de clase en la carpeta **Módulos de clase** del proyecto.

Algunos proyectos de macro están bloqueados y no pueden ser modificados.

Documentación más detallada sobre la creación de módulos de clase está fuera del alcance de esta documentación.

Escritura de macros

Se puede codificar manualmente una macro escribiéndola en el Editor de Macros o en el Editor VSTA. (Alternativamente, en CorelDRAW, se puede crear una macro VBA grabándola. Para más información, vea "Grabación de macros" en la página 47). Las macros que son desarrolladas en el Editor de Macros o en el Editor VSTA pueden beneficiarse del control total de programación, incluyendo ejecución de condicionales, bucles y bifurcaciones. De hecho, se pueden escribir macros que son programas en toda la extensión de la palabra.

En esta documentación, todo el código de macro es referido como una macro, aún cuando, en algunos contextos, una macro es sólo aquella parte de código que puede ser ejecutada por CorelDRAW o Corel PHOTO-PAINT.

Para escribir una macro, primero debe añadírse ésta a un módulo de código para el proyecto de macro deseado. Se pueden editar, renombrar o incluso eliminar macros.

Para procedimientos detallados, vea los siguientes temas:

- Añadir una macro a un proyecto de macro.
- Editar una macro VBA.
- Eliminar una macro VBA.

Añadir una macro a un proyecto de macro

1 En la ventana acoplable **Administrador de Macros**, hacer una de las siguientes cosas:
   - Click en el módulo de contenedor deseado en el proyecto de macro, click en **Nuevo** y luego click en **Nueva macro**.
   - Click derecho en el módulo de contenedor deseado en el proyecto de macro y luego click en **Nueva macro**.

Algunos proyectos de macro están bloqueados y no pueden ser modificados.

Editar una macro VBA

- En la ventana acoplable **Administrador de Macros**, haga una de las siguientes cosas:
  - Click en alguna macro de la lista y luego click en el botón **Editar**.
  - Click derecho en una macro de la lista y luego click en **Editar**.

  La macro se abre en el Editor de Macros.

Algunos proyectos de macro están bloqueados y no pueden ser modificados.

Para información detallada acerca de codificación manual de macros, vea "¿Cómo está estructurada la automatización del código?" en la página 14.

Eliminar una macro VBA

- En la ventana acoplable **Administrador de Macros**, haga una de las siguientes cosas:
  - Click en alguna macro de la lista y luego click en el botón **Eliminar**.
  - Click derecho en alguna macro de la lista y luego click en **Eliminar**.

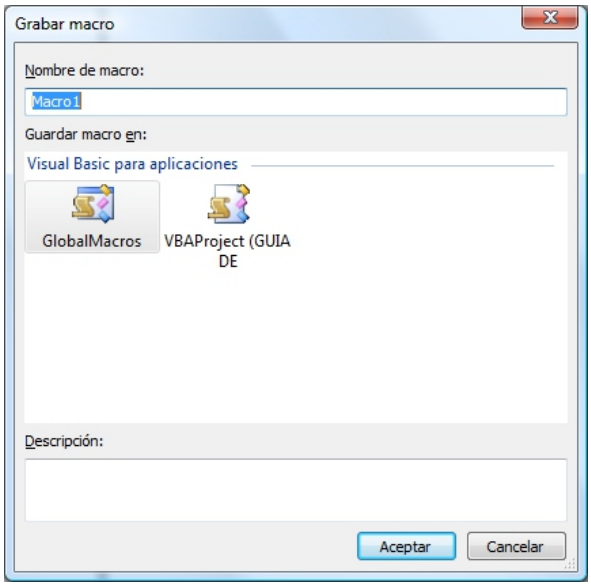
Algunos proyectos de macro están bloqueados y no pueden ser modificados.

Grabación de macros

CorelDRAW ofrece una característica de grabación que le permite crear una macro sin necesidad de codificarla manualmente. Para muchas tareas simples y repetitivas, las macros grabadas son una rápida y eficiente solución: Estas almacenan la secuencia de teclas que usted presiona y las acciones del ratón que usted ejecuta dentro de la aplicación. Tal vez prefiera crear macros grabándolas si no está familiarizado con el modelo de objeto para la aplicación, o si no está seguro cuáles objetos y métodos utilizar.

En Corel PHOTO-PAINT, las acciones pueden ser grabadas como scripts de Corel SCRIPT pero no como macros VBA o VSTA. Para más información sobre grabación de scripts, vea "Trabajando con scripts" en el archivo principal de Ayuda de Corel PHOTO-PAINT (corelpp.chm).

Si quiere almacenar una macro grabada para uso futuro, puede guardarla utilizando el cuadro de diálogo **Grabación de Macros**. Guardar una macro grabada es particularmente útil si usted quiere extender o personalizar su funcionalidad al editarla en el Editor de Macros.



El cuadro de diálogo ***Grabar Macro***

Sin embargo, si quiere utilizar una macro grabada sólo durante la sesión actual, se puede grabar una macro temporal.

Para procedimientos detallados, vea los siguientes temas:

- Grabar y guardar una macro.
- Grabar una macro temporal.

Grabar y guardar una macro

1 Hacer una de las siguientes cosas:

- Click en **Herramientas** » **Macros** » **Iniciar grabación**, o click en el botón **Iniciar grabación** en la barra de herramientas **Macros** para almacenar la macro en el proyecto de macro para grabación por defecto.
- En la ventana acoplable **Administrador de macros**, hacer click en el proyecto en el cual se almacenará la macro y luego click en el botón **Grabar** .

Aparece el cuadro de diálogo **Grabar macro**.

2 En el cuadro **Nombre de macro**, escriba un nombre para la macro.

Los nombres de macro pueden contener números, pero deben comenzar con una letra. Los nombres de macro no pueden contener espacios o caracteres no alfanuméricos a menos que sea el guión bajo (_).

3 Escriba una descripción de la macro en el cuadro **Descripción** y luego haga click en **Aceptar**.

4 Ejecute las acciones que quiera grabar.

La aplicación comienza a grabar sus acciones. Si quiere pausar la grabación, haga una de las siguientes cosas:

- Click en **Herramientas** » **Macros** » **Pausar grabación**. Repita este paso para reanudar la grabación.
- Click en el botón **Pausar grabación** en la barra de herramientas **Macros** o en la ventana acoplable **Administrador de macros**. Repita este paso para reanudar la grabación.

5 Para detener la grabación, haga una de las siguientes cosas:

- Click en **Herramientas** » **Macros** » **Detener grabación**.
- Click en el botón **Detener grabación** en la barra de herramientas **Macros** o en la ventana acoplable **Administrador de macros**.
- Presionar **Ctrl** + **Shift** + **O**.

No se puede grabar una macro si todos los proyectos de macro disponibles están bloqueados.

No todas las acciones pueden ser grabadas — algunas debido a su complejidad (aunque muchas de esas acciones pueden ser codificadas manualmente en el Editor de Macros). Cuando una acción no puede ser grabada, aparece un comentario en el código de la macro (`"La grabación de este comando no está soportada."`), pero el proceso de grabación continúa hasta detenerse. Se puede ver cualquier comentario en código abriendo la macro en el Editor de Macros.

Se puede especificar el proyecto de macro predeterminado para grabación haciendo click derecho en el proyecto de la ventana acoplable **Administrador de macros** y luego haciendo click en **Establecer como proyecto de grabación**. Sin embargo, no se puede especificar un proyecto de macro bloqueado.

Se puede cancelar la grabación de una macro y descartar cualquier comando grabado hasta entonces haciendo click en **Herramientas** » **Macros** » **Cancelar grabación**.

También se puede

Grabar las acciones de la lista **Deshacer** como una macro VBA	Hacer click en **Herramientas** » **Deshacer**, ejecutar las acciones que quiere grabar y luego hacer click en el botón **Guardar lista en una macro de VBA** de la ventana acoplable **Deshacer**.

Grabar una macro temporal

1 Hacer una de las siguientes cosas:

- Click en **Herramientas** » **Macros** » **Grabar macro temporal**.
- Presionar **Ctrl** + **Shift** + **R**.

2 Ejecutar las acciones que quiere grabar.

La aplicación comienza a grabar sus acciones. Si quiere pausar la grabación, haga una de las siguientes cosas:

- Click en **Herramientas** » **Macros** » **Pausar grabación**. Repita este paso para reanudar la grabación.
- Click en el botón **Pausar grabación** en la barra de herramientas **Macros** o en la ventana acoplable **Administrador de macros**. Repita este paso para reanudar la grabación.

3 Para detener la grabación, haga una de las siguientes cosas:

- Click en **Herramientas** » **Macros** » **Detener grabación**.
- Click en el botón **Detener grabación** en la barra de herramientas **Macros** o en la ventana acoplable **Administrador de macros**.
- Presionar **Ctrl** + **Shift** + **O**.

La macro es temporalmente guardada en el proyecto de grabación por defecto. Cuando se finaliza la sesión actual, la macro es eliminada del proyecto

No se puede grabar una macro si todos los proyectos de macro disponibles están bloqueados.

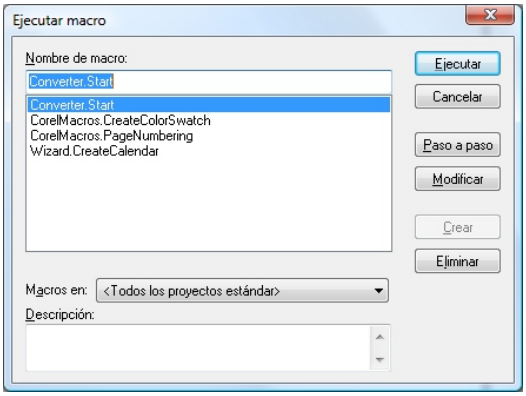
No todas las acciones pueden ser grabadas .

Se puede especificar el proyecto de grabación de macro predeterminado haciendo click derecho en la ventana acoplable **Administrador de Macros** y luego clickeando en **Establecer como proyecto de grabación**. Sin embargo, no se puede especificar un proyecto de macro bloqueado.

Se puede cancelar la grabación de una macro y descartar los comandos grabados hasta entonces haciendo click en **Herramientas** » **Macros** » **Cancelar grabación**.

Ejecución de macros

Usted puede ejecutar las macros guardadas ya sea directamente desde dentro de CorelDRAW o Corel PHOTO-PAINT o desde el Editor de Macros.



*El cuadro de diálogo **Ejecutar Macro***

También se puede ejecutar cualquier macro temporal grabada en CorelDRAW.

Para procedimientos detallados, vea los siguientes temas:

- Ejecutar una macro grabada.
- Ejecutar una macro temporal.

Ejecutar una macro grabada

- Hacer una de las siguientes cosas:
  - Click en **Herramientas** » **Macros** » **Ejecutar macro**, o click en el botón **Ejecutar macro** de la barra de herramientas **Macros**. Desde el cuadro de lista **Macros en** elija el proyecto o archivo en el cual está almacenada la macro. Desde la lista **Nombre de macro** elija la macro. Haga click en **Ejecutar**.
  - En la ventana acoplable **Administrador de Macros**, haga doble click en una macro de la lista.
  - En la ventana acoplable **Administrador de Macros**, haga click en una macro de la lista y luego click en el botón **Ejecutar**.
  - En la ventana acoplable **Administrador de Macros**, haga click derecho en alguna macro de la lista y luego click en **Ejecutar**.
  - En el Editor de Macros, haga click en cualquier parte de la subrutina que conforma la macro y luego haga click en **Ejecutar** » **Ejecutar macro**.

Ejecutar una macro temporal

- Hacer una de las siguientes cosas:
  - Click en **Herramientas** » **Macros** » **Ejecutar macro temporal**.
  - Presionar **Ctrl** + **Shift** + **P**.

Esta opción está habilitada sólo después de que ha grabado una macro temporal.

Si ha creado múltiples macros temporales, debe especificar cuál proyecto de macro contiene el que quiere ejecutar. Haga click derecho en el proyecto desde la ventana acoplable **Administrador de Macros** y luego click en **Establecer como proyecto de grabación**.

Depuración de macros

Para asegurarse que sus macros se ejecuten como lo espera, es importante depurarlas.

El Editor de Macros proporciona cuatro ventanas para depuración de código VBA. El Editor de Macros también cuenta con dos potentes prestaciones de depuramiento que son comunes a los editores de lenguaje: ajuste de puntos de interrupción y transferencia de código paso a paso.

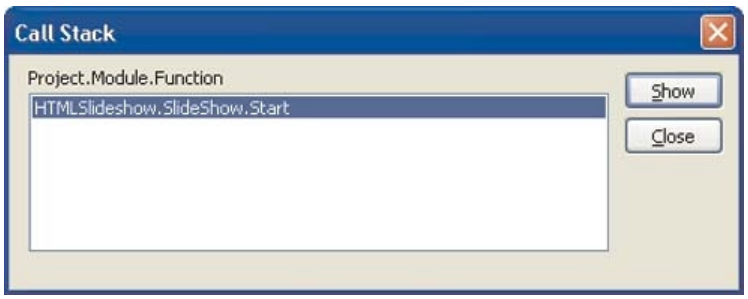
El Editor de Macros también soporta dos técnicas avanzadas de depuración que no se discuten en esta documentación: Hacer cambios al código mientras se está ejecutando, y observación y cambio de variables.

Utilizando las ventanas de depuración

El Editor de Macros proporciona cuatro ventanas para depuración de código VBA: la ventana **Pila de llamadas**, la ventana **Inmediato**, la ventana **Locales** y la ventana **Inspección**. Todas estas ventanas proporcionan información importante acerca del estado de las funciones y variables mientras una aplicación se ejecuta.

La ventana **Pila de llamadas** es un cuadro de diálogo modal que enlista cuál función llama a cual función. En aplicaciones grandes y complicadas, esta información es útil para trazar los pasos de una función particular que está siendo llamada. Para visitar una función listada en la ventana, seleccione el nombre de la función y luego haga click en **Mostrar**, o si no cierre la ventana.

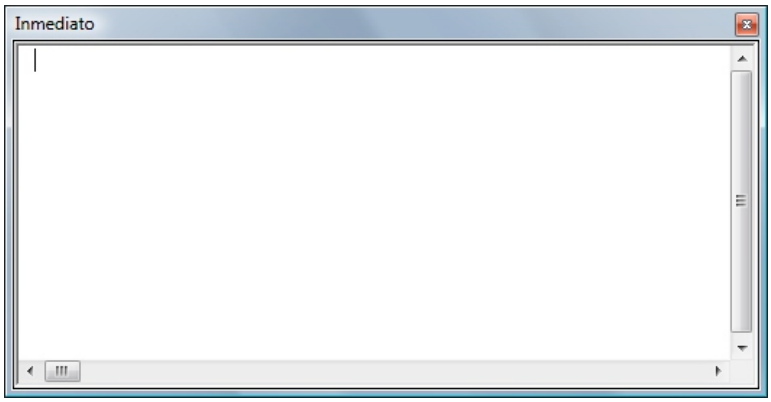
Para mostrar la ventana **Pila de llamadas**, haga click en **Ver** » **Pila de llamadas**.



La ventana ***Pila de llamadas***

La ventana **Inmediato** le permite escribir y ejecutar líneas arbitrarias de código mientras una macro está pausada. Esta funcionalidad es útil para obtener o ajustar la propiedad de un objeto en un documento, o para ajustar el valor de una variable en el código. Para ejecutar una pieza de código, escríbala en la ventana **Inmediato** y luego presione **Enter**; el código se ejecutará inmediatamente.

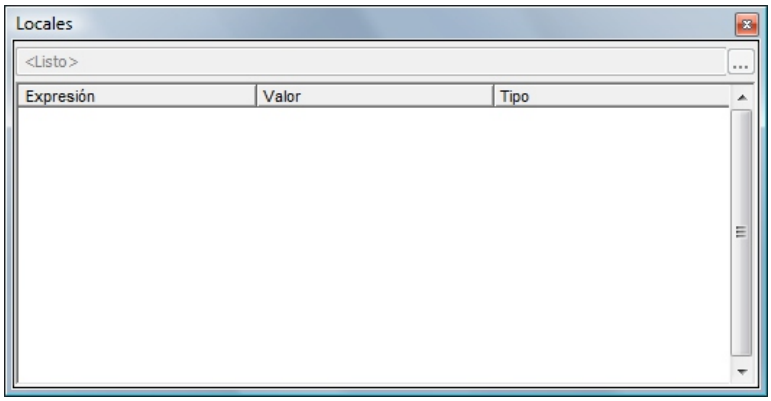
Para mostrar la ventana **Inmediato**, haga click en **Ver** » **Ventana Inmediato**.



*La ventana **Inmediato***

La ventana **Locales** muestra todas las variables y objetos que existen dentro del campo de acción actual. El tipo y valor para cada variable está enlistado en las columnas junto al nombre de la variable. Algunas variables y objetos tienen varios hijos, los cuales pueden ser mostrados haciendo click en el botón junto al padre. Muchas variables le permiten editar su valor haciendo click en ellas.

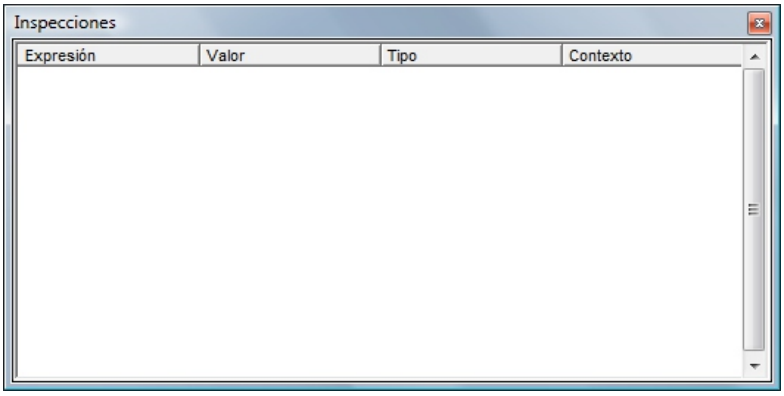
Para mostrar la ventana **Locales**, haga click en **Ver** » **Ventana Locales**.



*La ventana **Locales***

La ventana **Inspección** es utilizada para observar variables específicas o propiedades de objetos. Esta funcionalidad es útil para observación de sólo uno o dos valores con preferencia de búsqueda entre todos los valores en la ventana **Locales**.

Para mostrar la ventana **Inspección**, haga click en **Ver** » **Ventana Inspección**.

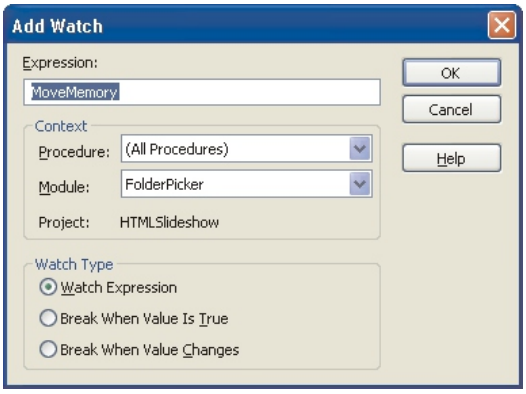


La ventana ***Inspección***

Para añadir un valor a la ventana **Inspección**, haga una de las siguientes cosas:

- Seleccione la variable u objeto y su propiedad y luego arrastre la selección hacia la ventana **Inspección**.
- Haga click en el elemento y luego click en **Depuración** » **Inspección rápida**.

Aparece el cuadro de diálogo **Añadir Inspección**.



El cuadro de diálogo ***Añadir Inspección***

Seleccione el elemento que quiere inspeccionar, seleccione alguna condición para esta inspección y luego haga click en **Aceptar**. Si la condición pasa a ser verdadera, la aplicación se pausa para permitirle examinar el código.

Ajuste de puntos de interrupción

Un punto de interrupción es un marcador en una línea de código que provoca una pausa. Para continuar, se debe o bien reiniciar la ejecución o bien dar un paso a través de líneas de código subsecuentes.

Para ajustar o quitar un punto de interrupción, haga click en la línea y luego click en **Depuración** » **Alternar punto de interrupción**. Por defecto, la línea está resaltada en rojo y un punto rojo se coloca en el margen.

Para restablecer el código después de pausarlo en un punto de interrupción, haga click en **Ejecutar** » **Continuar**. Para pausar la ejecución del código (inmediatamente saliendo de todas las funciones y descartando todos los valores de retorno), haga click en **Ejecutar** » **Restablecer**.

También puede "ejecutar hasta el cursor" — es decir, ejecutar el código hasta que éste llegue a la línea en donde está el cursor, y luego pausarlo en esa línea. Para hacer esto, haga click en la línea en donde quiere que se pause la ejecución y luego haga click en **Depuración** » **Ejecutar hasta el cursor**.

Para limpiar todos los puntos de interrupción, haga click en **Depuración** » **Borrar todos los puntos de interrupción**.

Si la línea con el punto de interrupción (o el cursor, cuando se activa "Ejecutar hasta el cursor") no es ejecutada porque está en un bloque condicional (if-then-else), el código no se detiene en esa línea.

Los puntos de interrupción no se guardan. Estos se pierden cuando se cierra el Editor de Macros.

Paso a paso a través del código

Cuando la ejecución se pausa en un punto de interrupción, se puede continuar a través del código una línea a la vez. Esta funcionalidad, llamada "paso a paso a través del código", le permite hacer lo siguiente:

- Examinar los valores de variables individuales después de cada línea.
- Determinar cómo afecta el código a los valores.
- Determinar cómo afectan los valores al código.

Para pasear a través del código, haga click en **Depuración** » **Paso a paso por instrucciones**. La ejecución avanza a cada línea en todas la llamadas de función y subrutinas.

Para ir paso a paso a través de cada línea de la función o subrutina actual pero no a través de las líneas de cada llamada de función o subrutina, haga click en **Depuración** » **Paso a paso por procedimientos**. Las llamadas de funciones y subrutinas son ejecutadas, pero no línea por línea.

Para ejecutar el resto de la función o subrutina actual pero pausar cuando la función o subrutina regresa al punto donde fue llamada, haga click en **Depuración** » **Paso a paso para salir**. Esta técnica es una forma rápida de regresar al punto de entrada de una función, para continuar paso a paso a través del código de la llamada de función.

Construcción de macros con interfaz amigable

Una parte importante de muchas soluciones de macro es la interfaz de usuario. Una interfaz bien diseñada mejora la facilidad de uso, poder y aceptación de una solución de macro. Interfaces de usuario simples pueden ser creadas con barras de herramientas, mientras que interfaces más complejas pueden ser creadas con cuadros de diálogo o ventanas acoplables — y pueden incluso permitir al usuario interactuar con el ratón.

Sin embargo, para algunas soluciones de macro, una interfaz de usuario aislada no es suficiente. Para hacer una solución de macro tan amigable al usuario como sea posible, usted puede proporcionar documentación para ello.

Esta sección contiene los siguientes temas:

- Proporcionar barras de herramientas a las macros.
- Proporcionar cuadros de diálogo a las macros.
- Proporcionar interacción con el usuario a las macros.
- Proporcionar documentación a las macros.

Proporcionar barras de herramientas a las macros

Una barra de herramientas proporciona una interfaz básica que realza la experiencia del usuario con la solución de macro. Las barras de herramientas son útiles porque sus botones son fáciles de recordar incluso si son pequeños, y porque estos botones pueden ser configurados para mostrar significativas leyendas y útiles ventanas de ayuda emergente.

Diseñando barras de herramientas para macros

Cuando se crean barras de herramientas éstas se deben planear cuidadosamente. Tener múltiples barras de herramientas pequeñas conteniendo unos cuantos botones relacionados es mejor que tener una gran barra de herramienta conteniendo todos los botones para todas las macros. Al separar los botones dentro de pequeños grupos será mucho más fácil destacarlos de los proyectos a los cuales pertenecen.

Para más información, vea los siguientes temas:

- Crear una barra de herramientas de macro.
- Añadir botones a una barra de herramientas de macro.

Asociando imágenes o iconos con macros

Los comandos de macro pueden tener una imagen o icono asociado a ellos. Esta imagen o icono puede estar visible u oculta en las barras de herramientas y menús, y puede ser de tamaño pequeño (16 X 16 pixeles), mediano (32 X 32 pixeles) o grande (48 X 48 pixeles).

Para más información, vea el siguiente procedimiento:

- Asociar una imagen o icono con una macro.

Para crear una barra de herramientas de macro

Cada macro puede tener tanto una leyenda como una ayuda contextual o cartel de información (tooltip). La leyenda se muestra cuando el comando de menú es utilizado y puede ser mostrado como parte de un botón, mientras que el cartel de información aparece cuando el puntero se sitúa sobre el botón o elemento de menú.

Para más información, vea los siguientes procedimientos:

- Configurar una leyenda para una macro.
- Configurar una ayuda contextual para una macro.

Para crear una barra de herramientas de macro

1 Click en **Herramientas** » **Opciones**.

2 Click en **Espacio de trabajo** » **Personalización** » **Barras de comandos**.

3 Click en Nuevo.

4 Escriba un nombre para la barra de herramientas.

5 Habilite la casilla de verificación junto al nombre de la barra de herramientas.

Para añadir botones a la barra de herramientas de macro

1 Click en **Espacio de trabajo** » **Personalización** » **Comandos**.

2 Elija **Macros** desde la lista **Comandos**.

La lista muestra los nombres totalmente aptos de todas las subrutinas públicas de parámetro libre de todos los archivos de proyecto instalados (GMS).

3 Arrastre una macro de la lista a la barra de herramientas.

La macro aparece en la barra de herramientas con el icono de macro predeterminado.

Para asociar una imagen o icono con una macro

1 Click en **Espacio de trabajo** » **Personalización** » **Comandos**.

2 Elija **Macros** desde la lista **Comandos**.

La lista muestra los nombres totalmente aptos de todas las subrutinas públicas de parámetro libre de todos los archivos de proyecto instalados (GMS).

3 Seleccione una macro en la lista **Comandos**.

4 Click en la pestaña **Aspecto** y luego haga una de las cosas siguientes:

- Aplicar una imagen bitmap de Windows (BMP) a la macro, click en **Importar**, navegar a donde la imagen está almacenada y seleccionarla. Los colores en la imagen son mapeados a su más cercana comparación disponible.
- Crear un icono personalizado para la macro, editar los pixeles mostrados en el área **Imagen**.

Para colocarle una leyenda a una macro

1 Click en **Espacio de trabajo** » **Personalización** » **Comandos**.

2 Elija **Macros** desde la lista **Comandos**.

La lista muestra los nombres totalmente aptos de todas las subrutinas públicas de parámetro libre de todos los archivos de proyecto instalados (GMS).

3 Seleccione una macro en la lista **Comandos**.

4 Click en la pestaña **Aspecto** y luego escribir la leyenda en el cuadro **Información**.

Para especificar un carácter en la leyenda como un acelerador que puede ser activado en combinación con la tecla **Alt**, escriba un símbolo de unión (**&**) antes del carácter. Esta tecla aceleradora se aplica sólo a comandos de menú, los cuales muestran caracteres aceleradores con un guión bajo (**_**).

Para colocar un cartel de información (tooltip) en una macro

1 Click en **Espacio de trabajo** » **Personalización** » **Comandos**.

2 Elija **Macros** desde la lista **Comandos**.

La lista muestra los nombres totalmente aptos de todas las subrutinas públicas de parámetro libre de todos los archivos de proyecto instalados (GMS).

3 Seleccione una macro de la lista **Comandos**.

4 Click en la pestaña **General** y luego escriba una descripción en el cuadro **Ayuda emergente**.

Proporcionar cuadros de diálogo a las macros

Un cuadro de diálogo proporciona una interfaz amigable al usuario para soluciones de macro más complejas.

Para mejores resultados, todos los cuadros de diálogo deben proporcionar lo siguiente:

- Un título significativo.
- Una función obvia para cancelar o cerrar el cuadro de diálogo.
- Un diseño de fácil uso.
- Un botón de Ayuda desde el cual los usuarios puedan acceder a documentación de ayuda.
- Una ayuda emergente (es decir, una cadena **ControlTipText**) para cada control.

Hay dos tipos de cuadros de diálogo: modal y no modal.

Entendiendo los cuadros de diálogo tipo modal

Un cuadro de diálogo modal bloquea la aplicación hasta que el usuario decide algo y luego cierra el cuadro de diálogo. La mayoría de los cuadros de diálogo incorporados para soluciones de macro son de tipo modal, y la mayoría de los cuadros de diálogo de este tipo proporcionan los siguientes botones:

- **Aceptar** — ejecuta una acción y luego cierra el cuadro de diálogo. Este botón es el predeterminado.
- **Cancelar** — cierra el cuadro de diálogo sin ejecutar ninguna acción. Este botón proporciona la misma funcionalidad que el botón **Cerrar** de la esquina superior derecha del cuadro de diálogo.

Por otro lado, algunos cuadros de diálogo tipo modal proporcionan el siguiente botón:

- **Aplicar** — ejecuta una acción que puede ser confirmada haciendo click en el botón **Aceptar** o cancelada haciendo click en el botón **Cancelar**.

Finalmente, la mayoría de los cuadros de diálogo estilo asistente proporcionan los siguientes botones:

- **Atrás** — regresa a la página previa. Este botón puede estar deshabilitado en la primer página del cuadro de diálogo.
- **Siguiente** — avanza a la siguiente página. Este botón puede ser reemplazado por un botón Finalizar en la última página del cuadro de diálogo.
- **Finalizar** — ejecuta la acción para el cuadro de diálogo y luego lo cierra.

Entendiendo los cuadros de diálogo tipo no modal

Un cuadro de diálogo no modal no bloquea la aplicación, de modo que el usuario puede mantener abierto el cuadro de diálogo y continuar trabajando en la aplicación. De esta forma, los cuadros de diálogo se comportan como ventanas acoplables. La mayoría de los cuadros de diálogo de tipo no modal proporcionan los siguientes botones:

- **Aplicar** o **Crear** — ejecuta una acción (y puede, de hecho, estar especialmente etiquetado para describir esa acción). Este botón es normalmente el predeterminado.
- **Cerrar** — cierra el cuadro de diálogo. Este botón se utiliza después de que la acción es aplicada

Eligiendo entre cuadros de diálogo modales y no modales

Antes de que usted empiece a crear un cuadro de diálogo para su solución de macro, debe decidir si lo hace modal o no modal considerando lo que quiera que el cuadro de diálogo lleve a cabo.

Por ejemplo, digamos que usted está creando una solución en sucesión instantánea tal como un cuadro de diálogo **Imprimir** o un cuadro de diálogo **Guardar**. En este caso, usted proporcionaría un cuadro de diálogo modal porque es poco probable que el usuario quiera aplicar repetidamente los ajustes especificados.

Por otro lado, digamos que desea crear una solución para configurar un efecto y aplicarlo a una(s) forma(s) seleccionada(s). Para permitir que el usuario especifique los ajustes deseados y luego los aplique repetidamente, usted proporcionaría un cuadro de diálogo no modal.

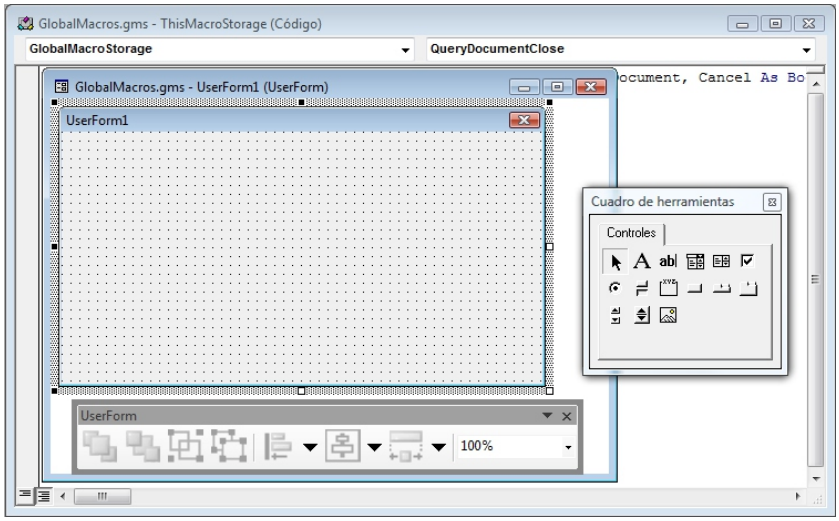
Después de elegir cuál tipo de cuadro de diálogo va a proporcionar, está listo para configurarlo. Para más información, vea "Configurando los cuadros de diálogo" en la página 58.

Después de configurar un cuadro de diálogo, ahora necesitará codificarlo. Para más información, vea "Codificando cuadros de diálogo" en la página 61.

Configuración de cuadros de diálogo

El Diseñador de Formas en el Editor de Macros le proporciona fácil acceso a las herramientas para configurar un cuadro de diálogo.

Se puede acceder al Diseñador de Formas al crear un nuevo cuadro de diálogo vacío. En el Explorador de Proyectos haga click derecho en el proyecto al que quiera añadirle un cuadro de diálogo y luego haga click en **Insertar** » **UserForm**.



Una forma en blanco en el Diseñador de Formas

El Diseñador de Formas proporciona dos características principales para diseñar cuadros de diálogo:

- El cuadro de herramientas **Diseñador de Formas**.
- La barra de herramientas **UserForm**.

El Diseñador de Formas también proporciona funciones para nombramiento y prueba de cuadros de diálogo.

Diseñando cuadros de diálogo

El cuadro de herramientas **Diseñador de Formas** es la utilidad principal para el diseño de cuadros de diálogo. Este le permite añadir controles a un cuadro de diálogo al arrastrarlos desde el cuadro de herramientas.

El cuadro de herramientas ***Diseñador de Formas***

El cuadro de herramientas **Diseñador de Formas** le permite añadir los siguientes controles a los cuadros de diálogo:

Icono	Nombre de control	Función
	Label	Proporciona texto estático, tal como instrucciones o títulos.
	TextBox	Proporciona un área para escribir texto. Para información sobre la codificación de este control, vea "Proporcionando cuadros de texto en cuadros de diálogo" en la página 61
	ComboBox	Proporciona una lista desde la cual puede ser seleccionado un solo elemento; opcionalmente, también proporciona un área para escribir texto. Para información sobre la codificación de este control, vea "Proporcionando cuadros combinados y cuadros de lista en los cuadros de diálogo".
	ListBox	Proporciona una lista en la cual pueden ser seleccionados múltiples elementos. Para información sobre la codificación de este control, vea "Proporcionando cuadros combinados y cuadros de lista en los cuadros de diálogo".
	CheckBox	Proporciona una casilla de verificación que puede estar habilitada (al clickearla para mostrar una marca de verificación), deshabilitada (al clickearla para eliminar la marca de verificación), o atenuada (es decir, que no está disponible).
	OptionButton	Proporciona un "radio botón" que está enlazado a otros radio botones con la misma propiedad **GroupName**, de modo que sólo uno de los botones puede estar habilitado a la vez.
	ToggleButton	Proporciona un botón que puede ser alternado (para verse presionado o no presionado)

Icono	Nombre de control	Función
	**Frame**	Agrupa elementos de tal manera que éstos se muevan junto con el marco.
	**CommandButton**	Proporciona un botón que puede ser clickeado para realizar una acción asignada. Para información sobre la codificación de este control, vea "Proporcionando botones en cuadros de diálogo" en la página 62.
	**TabStrip**	Proporciona vistas separadas de controles relacionados.
	**MultiPage**	Proporciona páginas múltiples de controles.
	**ScrollBar**	Proporciona acceso inmediato por desplazamiento a un rango de valores.
	**SpinButton**	Realza otro control (tal como un control **TextBox**), de tal manera que el valor para ese control pueda ser ajustado más rápidamente.
	**Image**	Proporciona una imagen. Para información sobre la codificación de este control, vea "Proporcionando imágenes en cuadros de diálogo" en la página 64.

El cuadro de herramientas **Diseñador de Formas** también presenta una herramienta **Selección**  , la cual le permite seleccionar y mover los controles del cuadro de diálogo.

Para mostrar un tema de Ayuda que contenga información acerca de un control seleccionado del cuadro de diálogo en el Diseñador de Formas, presione **F1**.

El Diseñador de Formas también proporciona acceso a la barra de herramientas **UserForm**, la cual puede ser utilizada cuando se diseña un cuadro de diálogo. Para información sobre esta barra de herramientas, vea "Utilizando las barras de herramientas del Editor de Macros" en la página 35.

Nombramiento de cuadros de diálogo

Después de que haya terminado de diseñar su cuadro de diálogo, tal vez quiera cambiar su título. Haga click en el cuadro de diálogo para seleccionarlo y luego en la ventana **Propiedades** cambie la propiedad **Caption**.

Para mejor comprensión, se le puede dar a cada cuadro de diálogo un nombre único y descriptivo utilizando la ventana **Propiedades**. Sin embargo, recuerde seguir las convenciones estándar de programación para nombramiento de variables.

Prueba de cuadros de diálogo

En cualquier momento usted puede probar su cuadro de diálogo presionando **F5** para ejecutarlo.

Después de que termine de configurar su cuadro de diálogo, puede empezar a codificarlo. Para más información, vea "Codificación de cuadros de diálogo" en la página 61.

Codificación de cuadros de diálogo

Después de configurar un cuadro de diálogo, se puede desarrollar el código VBA para presentarlo. Usted puede desarrollar el código para suministrar sus cuadros de texto, cuadros combinados y cuadros de lista, botones e imágenes.

Exposición de cuadros de diálogo

El método **Show** para un cuadro de diálogo le permite determinar como es presentado un cuadro de diálogo.

Por ejemplo, el siguiente código utiliza el método **Show** para mostrar el cuadro de diálogo **frmFooForm**:

```
frmFooForm.Show
```

Por otro lado, el método **Show** proporciona un parámetro **Modal**, el cual le permite especificar si el cuadro de diálogo es modal o no modal. Un valor de **vbModal** (o **1**) para este parámetro crea un cuadro de diálogo modal, mientras que un valor de **vbModeless** (o **0**) crea un cuadro de diálogo no modal. El siguiente ejemplo VBA crea un cuadro de diálogo no modal:

```
frmFooForm.Show vbModeless
```

Para abrir un cuadro de diálogo de una macro que esté disponible dentro de la aplicación misma, se debe crear una subrutina pública dentro del módulo de código. Sin embargo, una subrutina no puede ser facilitada desde el interior de la aplicación si la subrutina existe ya sea dentro del código de un cuadro de diálogo o ya sea dentro de un módulo de clase. Además, la subrutina no puede tomar ningún parámetro.

El siguiente ejemplo de subrutina VBA abre **frmFooForm** como un cuadro de diálogo no modal:

```
Public Sub showFooForm()
   frmFooForm.Show vbModeless
End Sub
```

Cuando se carga un cuadro de diálogo, éste activa su propio evento **UserForm_Initialize**. Desde este manejador de evento, usted debe inicializar todos los controles relevantes en el cuadro de diálogo. Para más información, vea "¿Cómo se proporcionan los manejadores de evento?" en la página 23.

Por último, también se puede utilizar el método **Show** para abrir cuadros de diálogo adicionales desde dentro del actual, como en el siguiente ejemplo VBA:

```
UserForm2.Show vbModal
```

Sin embargo, el control no es regresado a usted hasta que todos los cuadros de diálogo estén cerrados.

Exposición de cuadros de texto en cuadros de diálogo

Los cuadros de texto (es decir, los controles **TextBox**) son el soporte principal de ingreso de datos del usuario. Son fáciles de utilizar y rápidos de programar y son apropiados para un buena cantidad de propósitos.

Para ajustar el texto en un cuadro de texto cuando éste se inicialice, configure la propiedad **Text** (por defecto o implícita) para el control **TextBox** como en el siguiente ejemplo VBA:

```
txtWidth.Text = "3"
txtHeight = "1"
```

Para obtener el valor de un control **TextBox**, obtenga su propiedad **Text** en la ventana **Propiedades**, como en el siguiente ejemplo VBA:

```
Call SetSize(txtWidth.Text, txtHeight.Text)
```

Exposición de cuadros combinados y cuadros de lista en cuadros de diálogo

En un cuadro combinado (es decir, un control **ComboBox**), el usuario puede o bien elegir un elemento de la lista o bien escribir un valor dentro del cuadro de texto. Usted puede impedir que los usuarios sean capaces de escribir dentro de un control **ComboBox** configurando su propiedad **Style** (en la ventana **Propiedades**) a **fmStyleDrpDownList**.

En un cuadro de lista (es decir, un control **ListBox**), el usuario puede elegir uno o más elementos (normalmente, de entre tres y diez elementos) de la lista.

Para llenar una lista de cualquier tipo, se debe llamar al miembro de la función **AddItem** de la lista. Esta función toma dos parámetros: la cadena o valor numérico, y la posición en la lista. La posición del parámetro es opcional, de modo que si se omite se inserta el elemento en la última posición de la lista. Por ejemplo, el siguiente código VBA llena la lista **ComboBox1** con cuatro elementos:

```
ComboBox1.AddItem 1
ComboBox1.AddItem 2
ComboBox1.AddItem 3
ComboBox1.AddItem 0, 0
```

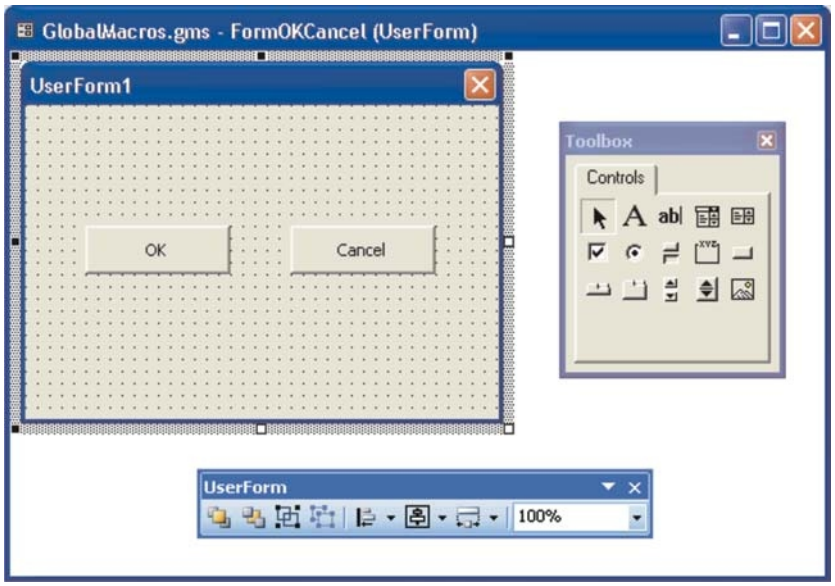
Para verificar cuál elemento está seleccionado cuando se haga click en el botón **Aceptar**, pruebe la propiedad **ListIndex** de la lista.

Para obtener el valor del título de un cuadro combinado o un cuadro de lista seleccionado, pruebe la propiedad **Text** del elemento, como en el siguiente código VBA:

```
Dim retList As String
retList = ComboBox1.Text
```

Exposición de botones en cuadros de diálogo

Se le puede añadir un botón a un cuadro de diálogo utilizando el control **CommandButton**. Haga click en el cuadro de diálogo para añadir un botón de tamaño predeterminado, o arrastre para crear uno de tamaño personalizado. Haga click en su título para editarlo o seleccione el botón y edite su propiedad **Caption** en la ventana **Propiedades**. Tal vez también quiera cambiar el nombre del botón a algo más descriptivo, tal como **buttonOK** o **buttonCancelar**.



Diseñando botones en la Forma Diseñador

La mayoría de los cuadros de diálogo tienen un botón **Aceptar** y un botón **Cancelar**. Sin embargo, ningún botón funciona hasta que su cuadro de diálogo tiene código para manejar el evento click del botón. (Esto se debe a que los cuadros de diálogo en VBA y VSTA son eventos dados).

Para un botón **Aceptar**, se puede ajustar su propiedad **Default** a **True** de modo que el manejador de evento para el botón sea llamado cuando el usuario presione la tecla **Enter** para activar el cuadro de diálogo. De este modo, el manejador de evento click para el botón ejecuta la funcionalidad del cuadro de diálogo y luego descarga (cierra) ese cuadro de diálogo.

Si el cuadro de diálogo es utilizado para configurar el tamaño de las formas CorelDRAW seleccionadas para ajustar su anchura y altura, entonces el manejador de evento click para el botón **Aceptar** podría ser parecido a la siguiente muestra de código VBA (la cual asume que usted ya ha creado dos cuadros de texto llamados **txtWidth** y **txtHeight**):

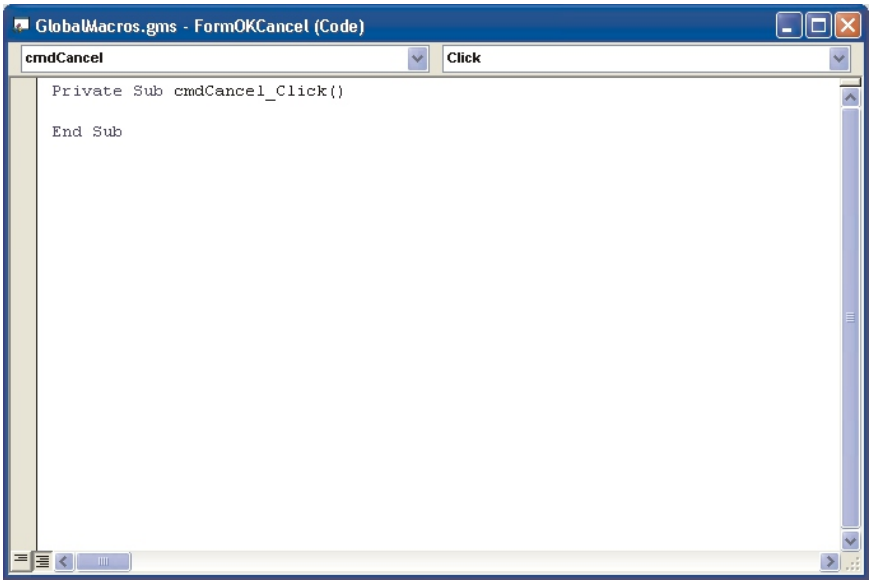
```
Private Sub buttonOK_Click()
  Me.Hide
  Call SetSize(txtWidth.Text, txtHeight.Text)
  Unload Me
End Sub
```

Similarmente, la subrutina CorelDRAW de ajuste de tamaño, podría parecerse a la siguiente:

```
Private Sub SetSize(width As String, height As String)
  ActiveDocument.Unit = cdrInch
  ActiveSelection.SetSize CDbl(width), CDbl(height)
End Sub
```

Desde el interior del módulo de código del cuadro de diálogo, el objeto cuadro de diálogo está implícito, de modo que todos los controles pueden ser accesados simplemente por nombre. Desde otros módulos, los controles deben ser accesados a través de sus nombres completos (como en **UserForm1.buttonOK**).

El botón **Cancelar** es el control más simple: éste debe dar por terminada la forma sin hacer ninguna otra cosa. Para añadir una acción de cancelar al botón **Cancelar**, haga doble click en el botón desde el interior del Diseñador de Forma para mostrar su código en la ventana **Código**. Esto crea una nueva subrutina llamada **cmdCancel–Click**:



*La ventana **Código** con el código para un botón **Cancelar***

El siguiente código VBA, si se aplica al botón **Cancelar**, da por terminado el cuadro de diálogo cuando el botón es clickeado:

```
Private Sub cmdCancel_Click()
    Unload Me
End Sub
```

Si usted continúa y configura la propiedad **Cancel** del cuadro de diálogo a **True**, entonces cuando el usuario presione la tecla **Escape**, el evento **cmdCancel_Click** es accionado y el código proporcionado descarga (cierra) la forma.

Exposición de imágenes en cuadros de diálogo

El control **Imagen** es utilizado para colocar gráficos en un cuadro de diálogo. La imagen (un bitmap) es contenida en la propiedad **Picture**, de modo que usted puede o bien cargar una imagen RGB desde un archivo (tal como un archivo GIF, JPEG o Windows Bitmap) o bien pegar una dentro de la propiedad.

En tiempo de ejecución, usted puede cambiar la propiedad **Picture** si quiere cargar una nueva imagen dentro del control **Imagen**. Para cambiar la propiedad **Picture**, utilice la función **LoadPicture** y proporcione una ruta para el nuevo archivo de imagen, como en el siguiente ejemplo VBA:

```
Image1.Picture = LoadPicture("C:\Images\NewImage.gif")
```

Proporcionar interacción con el usuario en las macros

Una forma de hacer sus soluciones de macro más amigables con el usuario es optimizarlas para que interactúen con el usuario, tal como una acción del ratón. Una macro que captura las acciones del ratón le da a los usuarios influencia en tiempo real sobre el resultado de esa macro.

El modelo de objeto CorelDRAW proporciona tres principales formas de recibir acciones de ratón de los usuarios, como es explicado en los siguientes temas:

- Captura de clicks de ratón.
- Captura de arrastres de ratón.
- Captura de coordenadas.

Captura de clicks de ratón

Para obtener la posición de un simple click de ratón, se puede utilizar el método **GetUserClick** de la clase **Document**. Este método pausa la macro hasta que transcurre el periodo de tiempo especificado, o hasta que el usuario hace click en el documento o presiona la tecla **Escape**. Aquí se muestra un ejemplo VBA que utiliza el método **Document.GetUserClick**:

```
Dim doc As Document, retval As Long
Dim x As Double, y As Double, shift As Long
Set doc = ActiveDocument
doc.Unit = cdrCentimeter
retval = doc.GetUserClick(x, y, shift, 10, True, cdrCursorPick)
```

Los siguientes parámetros para el método **Document.GetUserClick** están codificados dentro del ejemplo anterior:

- La variable `x` devuelve la posición horizontal del click de ratón.
- La variable `y` devuelve la posición vertical del click de ratón.
- El parámetro `shift` devuelve la combinación de las teclas **Shift**, **Ctrl** y **Alt** que se mantengan presionadas por el usuario cuando se haga el click de ratón. Las teclas **Shift**, **Ctrl** y **Alt** tienen valores asignados de `1`, `2` y `4` (respectivamente), la suma de los cuales es el valor devuelto.
- El valor `10` especifica el número de segundos para que el usuario haga click en el documento.
- El valor `True` especifica que el parámetro `SnapToObjects` está habilitado.
- El valor `cdrCursorPick` especifica que el icono de la herramienta **Selección** es utilizado por el icono del cursor. (No se puede utilizar un icono personalizado).

Uno de los siguientes valores es devuelto:

- **0** — El usuario completó con éxito el click.
- **1** — El usuario canceló al presionar la tecla **Escape**.
- **2** — La operación demoró y se agotó el tiempo.

Las coordenadas devueltas son relativas al origen de la página y, a menos que se especifique explícitamente, están en las mismas unidades que las del documento.

Para obtener las formas bajo el punto de click devuelto, se puede utilizar el método **SelectShapesAtPoint** (el cual es un miembro de **Page**), como en el siguiente ejemplo VBA:

```
doc.ActivePage.SelectShapesAtPoint x, y, True
```

Un valor de `True` selecciona objetos sin relleno, mientras que un valor `False` no selecciona objetos sin relleno.

Captura de arrastres de ratón

Para obtener la posición de un arrastre de ratón (o un área o rectángulo), se puede utilizar el método **GetUserArea** de la clase **Document**. Este método pausa la macro hasta que transcurre el periodo de tiempo especificado, o hasta que el usuario hace click, arrastra y libera el ratón en el documento o presiona la tecla **Escape**.

Aquí se muestra un ejemplo VBA que utiliza el método **Document.GetUserArea**:

```
Dim doc As Document, retval As Long, shift As Long
Dim x1 As Double, y1 As Double, x2 As Double, y2 As Double
Set doc = ActiveDocument
doc.Unit = cdrCentimeter
retval = doc.GetUserArea(x1, y1, x2, y2, shift, 10, True, cdrCursorExtPick)
ActivePage.SelectShapesFromRectangle x1, y1, x2, y2, False
```

Los siguientes parámetros para el método **Document.GetUserArea** están codificados dentro del ejemplo anterior:

- Las variables `x1` y `y1` devuelven las posiciones horizontal y vertical (respectivamente) de la esquina superior izquierda del área.
- Las variables `x2` y `y2` devuelven las posiciones horizontal y vertical (respectivamente) de la esquina inferior derecha del área.
- El parámetro `shift` devuelve la combinación de las teclas **Shift**, **Ctrl** y **Alt** que se mantengan presionadas por el usuario cuando se arrastra el ratón. Las teclas **Shift**, **Ctrl** y **Alt** tienen valores asignados de 1, 2 y 4 (respectivamente), la suma de los cuales es el valor devuelto.
- El valor `10` especifica el número de segundos para que el usuario haga click en el documento.
- El valor `True` especifica que el parámetro `SnapToObjects` está habilitado.
- El valor `cdrCursorExtPick` especifica el icono a utilizar para el cursor.

En el ejemplo anterior, el código finaliza al seleccionar las formas que caigan completamente dentro del área al utilizar el método **Page.SelectShapesFromRectangle**.

Uno de los siguientes valores es devuelto:

- **0** — El usuario completó con éxito la selección.
- **1** — El usuario canceló al presionar la tecla **Escape**.
- **2** — La operación demoró y se agotó el tiempo.

Este método devuelve dos puntos que son interpretados como las esquinas de un rectángulo. Sin embargo, los dos puntos pueden también ser utilizados como el punto inicial y el punto final de un arrastre de ratón.

Las coordenadas devueltas son relativas al origen de la página y, a menos que sea explícitamente especificado, están en las mismas unidades que las del documento.

Captura de coordenadas

Cuando se capturan acciones de ratón, o cuando se desarrolla una compleja solución de macro, tal vez quiera transmutar entre coordenadas de pantalla y coordenadas del documento. Esta conversión se hace con los métodos **ScreenToDocument** y **DocumentToScreen** de la clase **Window.**

El siguiente ejemplo VBA convierte un conjunto de coordenadas de pantalla en un punto en el documento que está visible en la ventana activa:

```
Dim docX As Double, docY As Double
ActiveDocument.Unit = cdrMillimeter
ActiveWindow.ScreenToDocument 440, 500, docX, docY
```

El siguiente ejemplo VBA devuelve las coordenadas de pantalla de un punto en el documento como si éste apareciera en la pantalla:

```
Dim screenX As Long, screenY As Long
ActiveDocument.Unit = cdrMillimeter
ActiveWindow.DocumentToScreen 40, 60, screenX, screenY
```

En ambos ejemplos, las coordenadas convertidas son devueltas en los últimos dos parámetros.

Las coordenadas de pantalla inician en la esquina superior izquierda de la pantalla, de modo que los valores positivos de y están pantalla abajo, mientras que los valores negativos de y están pantalla arriba.

Se puede probar si un conjunto de coordenadas (es decir, un punto) está dentro, fuera o sobre el contorno de una curva al utilizar el método **Shape.IsOnShape**. Para un conjunto de coordenadas de documento, este método devuelve una de las siguientes cosas:

- **cdrInsideShape** — si la coordenada está dentro de la forma.
- **cdrOutsideShape** — si la coordenada está fuera de la forma.
- **cdrOnMarginOfShape** — si la coordenada está sobre o junto al contorno de la forma.

Por ejemplo, el siguiente código VBA prueba dónde está el punto **(4, 6)** en relación a la forma activa:

```
Dim onShape As Long
ActiveDocument.Unit = cdrInch
onShape = ActiveShape.IsOnShape(4, 6)
```

Proporcionar documentación para las macros

Para hacer una macro tan amigable al usuario como sea posible, se puede proporcionar documentación para ella.

Una solución es crear un archivo Léame o un manual impreso. Otra solución es incorporar la documentación directamente dentro de la interfaz del usuario para la macro, pero este método consume "bienes raíces" en pantalla. No obstante, otra solución es crear un sistema de Ayuda en línea, pero este método requiere herramientas especiales y una considerable cantidad de trabajo adicional.

Tal vez la manera más simple de proporcionar documentación para la macro es en forma de un archivo de texto puro. De hecho, en el momento de la instalación, un proyecto de macro puede crear un valor de registro que señale la localización de este archivo. En VBA, la siguiente función puede ser utilizada para abrir un archivo de texto puro (donde el parámetro `file` proporciona la ruta completa del archivo, tal como **C:\ReadMe.txt**):

```
Public Sub launchNotepad(file As String)
  Shell "Notepad.exe" & " " & file, vbNormalFocus
End Sub
```

Una solución mucho más poderosa es proporcionar documentación en formato HTML. HTML proporciona numerosos beneficios sobre texto puro, incluyendo soporte para gráficos y enlaces de hipertexto (tal como especificar localizaciones en el documento — por ejemplo, **index.html#middle**).

En VBA, la siguiente función puede ser utilizada para abrir un archivo HTML (donde el parámetro `url` proporciona la ruta completa del archivo — tal como **C:\ReadMe.txt** — o un URL para el archivo):

```
' Colocar esta enunciación Declare antes de todas las Subrutinas y Funciones!
Declare Function ShellExecute Lib "shell32.dll" _
    Alias "ShellExecuteA" (ByVal hwnd As Long, _
    ByVal lpOperation As String, ByVal lpFile As String, _
    ByVal lpParameters As String, ByVal lpDirectory As String, _
    ByVal nShowCmd As Long) As Long
Public Sub launchBrowser(url As String)
    ShellExecute 0, vbNullString, url, vbNullString, vbNullString, 5
End Sub
```

Organización y Puesta en Uso de Macros

Cuando haya terminado de desarrollar su solución macro, puede hacerla disponible para otros usuarios.

Esta sección contiene los siguientes temas:

- Organización de macros.
- Puesta en uso de macros.

Organización de Macros

Para hacer que sus soluciones de macro sean fáciles de utilizar, usted puede organizarlas. Aquí tiene algunos consejos:

- Para ordenar sus macros, utilice un módulo de código separado para cada macro y luego agrupe las macros relacionadas dentro de un solo archivo GMS.
- Para ayudar a los usuarios a encontrar el punto de entrada a una macro, coloque todas las subrutinas públicas dentro de un solo módulo de código de modo que la macro pueda ser llamada desde el interior de CorelDRAW o Corel PHOTO-PAINT.

Puesta en Uso de Macros

Puede dejar listas para instalar las soluciones de macro para los usuarios. Puede alistar archivos GMS o espacios de trabajo, o ambos.

Preparando archivos GMS

Cada documento CorelDRAW o Corel PHOTO-PAINT tiene un archivo GMS intrínseco. Por esta razón, usted puede distribuir explícitamente una macro como parte de un documento porque cuando el documento es abierto, el usuario tiene acceso inmediato a sus macros. Esta técnica de implementación le permite, por ejemplo, configurar una macro para monitorear cuánto tiempo ha empleado el usuario editando un documento.

Alternativamente, se puede distribuir el módulo de código que contiene la macro. Sin embargo, este método de implementación requiere que los usuarios integren manualmente el módulo de código dentro de un archivo de proyecto existente.

La forma más simple y confiable de poner en marcha una macro es utilizar su archivo GMS. Para comenzar, usted debe exportar el archivo GMS desde su computadora. Luego, cada usuario debe importar el archivo GMS utilizando la ventana acoplable **Administrador de Macros**.

- Exportar un archivo GMS.
- Importar un archivo GMS.

Preparando espacios de trabajo

Algunas soluciones de macro incluyen un espacio de trabajo personalizado que contiene apropiadas barras de herramientas, menús y atajos de teclado. Usted puede preparar las características de un espacio de trabajo personalizado para los usuarios al crear un archivo Corel de espacio de trabajo (XSLT). Se puede exportar un subconjunto de características de espacio de trabajo — tal como menús individuales, barras de herramientas individuales, o grupos completos de atajos de teclado — si desea que los usuarios instalen sólo aquellas características, o puede exportar el espacio de trabajo completo si lo prefiere.

- Exportar características de espacio de trabajo

Los usuarios pueden instalar características de espacios de trabajo personalizados al importar archivos XSLT que usted les facilite.

- Importar características de espacio de trabajo

En CorelDRAW, los usuarios pueden importar características de espacio de trabajo utilizando el método **Application.ImportWorkspace**.

Para exportar un archivo GMS

- Localice el archivo GMS en su computadora y déjelo disponible para los usuarios.

  Los proyectos de macro creados por el usuario son almacenados normalmente en las siguientes rutas:

  - En Windows 7 y Windows Vista: **X:\Usuarios\<*nombre de usuario*>\AppData\Roaming\Corel\CorelDRAW Graphics Suite X5\<*aplicación*>** (donde **X:** es la unidad donde se instaló el programa, **<*nombre de usuario*>** es el nombre del usuario, y **<*aplicación*>** es la subcarpeta de la aplicación).
  - En Windows XP: **X:\Documents and Settings\<*nombre de usuario*>\Application Data\Corel\CorelDRAW Graphics Suite X5\<*aplicación*>** (donde **X:** es la unidad donde se instaló el programa, **<*nombre de usuario*>** es el nombre del usuario, y **<*aplicación*>** es la subcarpeta de la aplicación).

Para importar un archivo GMS

1 Guardar el archivo GMS en su computadora.

  Los proyectos de macro creados por el usuario son almacenados normalmente en las siguientes rutas:

  - En Windows 7 y Windows Vista: **X:Usuarios\<*nombre de usuario*>\AppData\Roaming\Corel\CorelDRAW Graphics Suite X5\<*aplicación*>** (donde **X:** es la unidad donde se instaló el programa, **<*nombre de usuario*>** es el nombre del usuario, y **<*aplicación*>** es la subcarpeta de la aplicación).
  - En Windows XP: **X:\Documents and Settings\<*nombre de usuario*>\Application Data\Corel\CorelDRAW Graphics Suite X5\<*aplicación*>** (donde **X:** es la unidad donde se instaló el programa, **<*nombre de usuario*>** es el nombre del usuario, y **<*aplicación*>** es la subcarpeta de la aplicación).

2 En la ventana acoplable **Administrador de Macros**, haga una de las siguientes cosas:

  - Haga click en la lista de **Visual Basic para aplicaciones**, click en **Cargar...** y luego elija el proyecto.
  - Haga click derecho en la lista de **Visual Basic para aplicaciones** y luego click en **Cargar macro de proyecto**.

Para exportar características de espacio de trabajo

1 Haga click derecho en la barra de menús y luego click en **Personalizar** » **Espacio de Trabajo** » **Exportar espacios de trabajo**.

2 En la lista, habilite las casillas de verificación junto a las características de espacio de trabajo que quiera exportar:

  - **Ventanas acoplables** — incluye los tamaños y posiciones de las ventanas acoplables.
  - **Menús** — le permite elegir cuáles menús se incluirán.

- **Atajos de Teclado** — incluye todos los atajos de teclado disponibles.
- **Barra de Estado** — incluye la barra de estado.
- **Barras de Herramientas** — le permite elegir cuáles barras de herramientas incluir.

Habilite todas las casillas de verificación si quiere exportar el espacio de trabajo completo.

3 Haga click en **Guardar**.

4 En el cuadro **Nombre de archivo**, escriba un nombre para el archivo.

Las características del espacio de trabajo especificado son guardadas en un archivo de espacio de trabajo Corel simple (XSLT) con el nombre de archivo especificado.

Si lo desea, puede exportar cada característica de espacio de trabajo a un archivo separado. Simplemente exporte un elemento a la vez para crear series de archivos XSLT.

Cuando exporta atajos de teclado, se exportan todos los atajos de teclado. Si quiere distribuir sólo unos cuantos atajos, cree un nuevo espacio de trabajo, elimine todos los atajos de teclado en él y luego agregue solamente los atajos deseados.

Importar características de espacio de trabajo

1 Haga click derecho en la barra de herramientas y luego click en **Personalizar** » **Espacio de trabajo** » **Importar espacios de trabajo**.

2 Haga click en **Explorar**.

3 Seleccione el archivo de espacio de trabajo Corel (XSLT) deseado y luego haga click en **Siguiente**.

4 En la lista, habilite las casillas de verificación de las características del espacio de trabajo que quiera importar:

- **Ventanas Acoplables** — incluye los tamaños y posiciones de las ventanas acoplables.
- **Menús** — le permite elegir cuáles menús incluir.
- **Atajos de Teclado** — incluye todos los atajos de teclado disponibles.
- **Barra de Estado** —incluye la barra de estado.
- **Barras de herramientas** — le permite elegir cuáles barras de herramientas incluir.

Habilite todas las casillas de verificación si desea importar el espacio de trabajo completo.

5 Haga click en **Siguiente**.

6 Elija un destino para las características del espacio de trabajo haciendo una de las siguientes cosas:

- Habilite la opción **Espacio de trabajo actual** para importar las características del espacio de trabajo especificado dentro del espacio de trabajo actual, y luego haga click en **Siguiente**.
- Habilite la opción **Nuevo espacio de trabajo** para importar las características del espacio de trabajo especificado dentro de un nuevo espacio de trabajo. Haga click en **Siguiente** y proporcione detalles acerca del espacio de trabajo. Haga click en **Siguiente**.

7 Confirme los detalles de lo importado y luego haga click en **Finalizar**.

Las características del espacio de trabajo especificado son importadas dentro del espacio de trabajo que eligió.

Si el nombre de una barra de herramientas entrante es el mismo que el de una barra de herramientas existente, la barra de herramientas entrante es renombrada.

Si un comando importado llama a una macro desinstalada, ésta no funcionará.

Entendiendo el modelo de objeto CorelDRAW

En CorelDRAW, el objeto **Application** es la raíz de todos los demás objetos y se utiliza si ningún otro objeto raíz se ha especificado. Se puede utilizar el objeto **Application** de CorelDRAW para referenciar al modelo de objeto CorelDRAW desde un controlador fuera de proceso, como en el siguiente ejemplo Visual Basic:

```
Dim cdr As CorelDRAW.Application
Set cdr = CreateObject("CorelDRAW.Application")
```

Si lo desea, puede evitar utilizar la palabra clave **CreateObject** en el ejemplo anterior al importar el tipo de biblioteca a enfocar y utilizando directamente los tipos de datos.

El objeto **Application** contiene una colección **Documents** de todos los objetos abiertos **Document** (o "documentos") en la aplicación. Cuando un documento CorelDRAW es creado o abierto, un objeto correspondiente **Document** se añade a la colección **Documents** para el objeto **Application**. Cada objeto **Document** contiene una colección **Pages** de todos los objetos **Page** (o "páginas") en ese documento. Cada objeto **Page** contiene una colección **Layers** de todos los objetos **Layer** (o "capas") sobre esa página. Finalmente, cada objeto **Layer** contiene una colección **Shapes** de todos los objetos **Shape** (o "formas") sobre esa capa.

Por otro lado, el modelo de objeto contiene un conjunto de objetos filtro, los cuales proporcionan soporte para archivos de otras aplicaciones gráficas técnicas. Los filtros de importación están gobernados por la clase **Layer**, mientras que los filtros de exportación están gobernados por la clase **Document**.

Los documentos, páginas, capas, formas y filtros están entre los objetos más importantes en el modelo de objeto CorelDRAW. El entendimiento de estos objetos — y sus relaciones entre sí — es la clave para la comprensión del modelo de objeto CorelDRAW.

Esta sección contiene los siguientes temas:

- Trabajo con documentos.
- Trabajo con páginas.
- Trabajo con capas.
- Trabajo con formas.
- Trabajo con filtros de importación y filtros de exportación.

Todos los ejemplos de código en esta sección están escritos en VBA.

Para una representación visual del modelo de objeto CorelDRAW completo, por favor vea el diagrama del modelo de objeto CorelDRAW en la siguiente ubicación (donde **X:** es la unidad donde se instaló el programa):

- **X:\Archivos de programa \Corel\CorelDRAW Graphics Suite X5\ Data\CorelDRAW Object Model Diagram.pdf**

Trabajo con documentos

Cada documento abierto CorelDRAW, u objeto **Document**, es un miembro de la colección **Application.Documents**. Los documentos en esa colección aparecen en el orden en que fueron creados o abiertos.

CorelDRAW proporciona un número de propiedades, métodos y eventos para trabajar con documentos, la mayoría muy útiles, los cuales son listados en la siguiente tabla.

Clase	Miembro	Descripción
ActiveView	Propiedad **OriginX** y Propiedad **OriginY**	Se combina para especificar el origen de la vista activa. Para más información, vea "Paneo" en la página 84.
ActiveView	Método **SetViewPoint**	Especifica el origen de la vista activa. Para más información, vea "Paneo" en la página 84.
ActiveView	Propiedad **Zoom**	Especifica el factor zoom de la vista activa. Para más información, vea "Zooming" en la página 84.
AddInHook	Evento **New**	Es activado cuando se crea un documento. Para más información, vea "Creación de documentos" en la página 81.
Application	Propiedad **ActiveDocument**	Proporciona acceso directo al documento activo. Para más información, vea "Activación de documentos" en la página 81.
Application	Método **CreateDocument** o Método **CreateDocumentFromTemplate**	Crea un documento Para más información, vea "Creación de documentos" en la página 81.
Application	Evento **DocumentAfterExport**	Es activado al exportar un documento (es decir, cuando se cierra el cuadro de diálogo **Exportar**). Para más información, vea "Exportación de documentos" en la página 86.
Application	Evento **DocumentAfterPrint**	Es activado al imprimir un documento (es decir, cuando se cierra el cuadro de diálogo **Imprimir**). Para más información, vea "Impresión de documentos" en la página 89.
Application	Evento **DocumentAfterSave**	Es activado al guardar un documento (es decir, cuando se cierra el cuadro de diálogo **Guardar**). Para más información, vea "Guardado de documentos" en la página 86.

Clase	Miembro	Descripción
Application	Evento **DocumentBeforeExport**	Es activado al abrir el cuadro de diálogo **Exportar**. Para más información, vea "Exportación de archivos desde documentos" en la página 86.
Application	Evento **DocumentBeforePrint**	Es activado al abrir el cuadro de diálogo **Imprimir**. Para más información, vea "Impresión de documentos" en la página 89.
Application	Evento **DocumentBeforeSave**	Es activado al abrir el cuadro de diálogo **Guardar**. Para más información, vea "Guardado de documentos" en la página 86.
Application	Evento **DocumentClose**	Es activado cuando se cierra un documento. Para más información, vea "Cierre de documentos" en la página 91.
Application	Evento **DocumentNew**	Es activado cuando se crea un documento. Para más información, vea "Creación de documentos" en la página 81.
Application	Evento **DocumentOpen**	Es activado cuando se abre un documento. Para más información, vea "Apertura de documentos" en la página 81.
Application	Propiedad **Documents**	Contiene la colección de documentos abiertos. Para más información, vea "Activación de documentos" en la página 81.
Application	Método **OpenDocument**	Abre un documento. Para más información, vea "Apertura de documentos" en la página 81.
Application	Evento **QueryDocumentClose**	Es activado cuando el usuario responde a una solicitud de cierre de un documento. Para más información, vea "Cierre de documentos" en la página 91.
Application	Evento **QueryDocumentExport**	Es activado cuando el usuario responde a una solicitud de exportación de un documento. Para más información, vea "Exportación de archivos desde documentos" en la página 86.

Clase	Miembro	Descripción
Application	Evento **QueryDocumentPrint**	Es activado cuando el usuario responde a una solicitud de impresión de un documento. Para más información, vea "Impresión de documentos" en la página 89.
Application	Evento **QueryDocumentSave**	Es activado cuando el usuario responde a una solicitud de guardado de un documento. Para más información, vea "Guardado de documentos" en la página 86.
Application	Evento **WindowActivate**	Es activado cuando se activa una ventana. Para más información, vea "Activación de documentos" en la página 81.
Application	Evento **WindowDeactivate**	Es activado cuando se desactiva una ventana. Para más información, vea "Activación de documentos" en la página 81.
Document	Método **Activate**	Activa un documento. Para más información, vea "Activación de documentos" en la página 81.
Document	Propiedad **ActiveWindow**	Proporciona acceso directo a la ventana activa para un documento. Para más información, vea "Trabajo con ventanas" en la página 83.
Document	Evento **AfterExport**	Es activado al exportar un documento (es decir, cuando se cierra el cuadro de diálogo **Exportar**). Para más información, vea "Exportación de documentos" en la página 86.
Document	Evento **AfterPrint**	Es activado al imprimir un documento (es decir, cuando se cierra el cuadro de diálogo **Imprimir**). Para más información, vea "Impresión de documentos" en la página 89.
Document	Evento **AfterSave**	Es activado al guardar un documento (es decir, cuando se cierra el cuadro de diálogo **Guardar**). Para más información, vea "Guardado de documentos" en la página 86.
Document	Evento **BeforeExport**	Es activado al abrir el cuadro de diálogo **Exportar**. Para más información, vea "Exportación de archivos desde documentos" en la página 86.

Clase	Miembro	Descripción
Document	Evento **BeforePrint**	Es activado al abrir el cuadro de diálogo **Imprimir**. Para más información, vea "Impresión de documentos" en la página 89.
Document	Evento **BeforeSave**	Es activado al abrir el cuadro de diálogo **Guardar**. Para más información, vea "Guardado de documentos" en la página 86.
Document	Método **BeginCommandGroup** y Método **EndCommandGroup**	Se combina para crear un "grupo comando" que reduce una serie de acciones programadas relacionadas con el documento, a un simple paso que se pueda deshacer. Para más información, vea "Creación de grupos de comandos para documentos" en la página 85.
Document	Evento **Close**	Es activado al cerrar un documento. Para más información, vea "Cierre de documentos" en la página 91.
Document	Método **Close**	Cierra un documento. Para más información, vea "Cierre de documentos" en la página 91.
Document	Método **CreateView**	Crea una vista de documento. Para más información, vea "Trabajo con vistas" en la página 84.
Document	Método **Export**, Método **ExportEx** o Método **ExportBitmap**	Exporta un archivo desde un documento. Para más información, vea "Exportación de archivos desde documentos" en la página 86.
Document	Propiedad **FilePath**, Propiedad **FileName** o Propiedad **FullFileName**	Especifica la ruta o nombre de archivo (o ambas) de un documento guardado. Para más información, vea "Activación de documentos" en la página 81.
Document	Método **GetUserArea**	Devuelve información acerca de un área del documento que el usuario arrastra con el ratón. Para más información, vea "Captura de arrastres de ratón" en la página 66.
Document	Método **GetUserClick**	Devuelve información acerca de una posición del documento en la que el usuario hace click con el ratón. Para más información, vea "Captura de clicks de ratón" en la página 65.

Clase	Miembro	Descripción
Document	Evento **Open**	Es activado cuando se abre un documento. Para más información, vea "Apertura de documentos" en la página 81.
Document	Método **PrintOut** y Propiedad **PrintSettings**	Se combina para imprimir un documento utilizando los ajustes especificados. Para más información, vea "Impresión de documentos" en la página 89.
Document	Método **PublishToPDF** y Propiedad **PDFSettings**	Se combina para publicar un documento como PDF utilizando los ajustes especificados. Para más información, vea "Publicación de documentos como PDF" en la página 88.
Document	Evento **QueryClose**	Es activado cuando el usuario responde a una solicitud de cierre de un documento. Para más información, vea "Cierre de documentos" en la página 91.
Document	Evento **QueryExport**	Es activado cuando el usuario responde a una solicitud para exportar un documento. Para más información, vea "Exportación de archivos desde documentos" en la página 86.
Document	Evento **QueryPrint**	Es activado cuando el usuario responde a una solicitud para imprimir un documento. Para más información, vea "Impresión de documentos" en la página 89.
Document	Evento **QuerySave**	Es activado cuando el usuario responde a una solicitud para guardar un documento. Para más información, vea "Guardado de documentos" en la página 86.
Document	Propiedad **ReferencePoint**	Especifica el punto de referencia del documento. Para más información, vea "Ajuste de las propiedades del documento" en la página 82.
Document	Método **SaveAs** o Método **Save**	Guarda un documento. Para más información, vea "Guardado de documentos" en la página 86.

Clase	Miembro	Descripción
Document	Propiedad **Unit**	Especifica la unidad de medida para el documento. Para más información, vea "Ajuste de las propiedades del documento" en la página 82.
Document	Propiedad **WorldScale**	Especifica la escala de dibujo para el documento. Para más información, vea "Ajuste de las propiedades del documento" en la página 82.
Document	Propiedad **Views**	Contiene la colección de vistas para el documento. Para más información, vea "Trabajo con vistas" en la página 84.
GlobalMacroStorage	Evento **DocumentAfterExport**	Es activado cuando se exporta un documento (es decir, cuando se cierra el cuadro de diálogo **Exportar**). Para más información, vea "Exportación de archivos desde documentos" en la página 86.
GlobalMacroStorage	Evento **DocumentAfterPrint**	Es activado cuando se imprime un documento (es decir, cuando se cierra el cuadro de diálogo **Imprimir**). Para más información, vea "Impresión de documentos" en la página 89.
GlobalMacroStorage	Evento **DocumentAfterSave**	Es activado cuando se guarda un documento (es decir, cuando se cierra el cuadro de diálogo **Guardar**). Para más información, vea "Guardado de documentos" en la página 86.
GlobalMacroStorage	Evento **DocumentBeforeExport**	Se activa cuando se abre el cuadro de diálogo **Exportar**. Para más información, vea "Exportación de archivos desde documentos" en la página 86.
GlobalMacroStorage	Evento **DocumentBeforePrint**	Se activa cuando se abre el cuadro de diálogo **Imprimir**. Para más información, vea "Impresión de documentos" en la página 89.
GlobalMacroStorage	Evento **DocumentBeforeSave**	Se activa cuando se abre el cuadro de diálogo **Guardar**. Para más información, vea "Guardado de documentos" en la página 86.
GlobalMacroStorage	Evento **DocumentClose**	Se activa cuando se cierra un documento. Para más información, vea "Cierre de documentos" en la página 91.

Clase	Miembro	Descripción
GlobalMacroStorage	Evento **DocumentNew**	Es activado cuando se crea un documento. Para más información, vea "Creación de documentos" en la página 81.
GlobalMacroStorage	Evento **DocumentOpen**	Es activado cuando se crea un documento. Para más información, vea "Apertura de documentos" en la página 81.
GlobalMacroStorage	Evento **QueryDocumentClose**	Es activado cuando el usuario responde a una solicitud de cierre de un documento. Para más información, vea "Cierre de documentos" en la página 91.
GlobalMacroStorage	Evento **QueryDocumentExport**	Es activado cuando el usuario responde a una solicitud de exportación de un documento. Para más información, vea "Exportación de archivos desde documentos" en la página 86.
GlobalMacroStorage	Evento **QueryDocumentPrint**	Es activado cuando el usuario responde a una solicitud de impresión de un documento. Para más información, vea "Impresión de documentos" en la página 89.
GlobalMacroStorage	Evento **QueryDocumentSave**	Es activado cuando el usuario responde a una solicitud de guardado de un documento. Para más información, vea "Guardado de documentos" en la página 86.
GlobalMacroStorage	Evento **WindowActivate**	Es activado cuando se activa una ventana. Para más información, vea "Activación de documentos" en la página 81.
GlobalMacroStorage	Evento **WindowDeactivate**	Es activado cuando se desactiva una ventana. Para más información, vea "Activación de documentos" en la página 81.
View	Método **Activate**	Aplica una vista guardada en la ventana del documento. Para más información, vea "Trabajo con vistas" en la página 84.
Window	Método **Activate**	Activa una ventana del documento. Para más información, vea "Trabajo con ventanas" en la página 83.

Clase	Miembro	Descripción
Window	Propiedad **ActiveView**	Proporciona acceso directo a la vista activa de una ventana del documento. Para más información, vea "Trabajo con vistas" en la página 84.
Window	Método **Close**	Cierra una ventana de documento. Para más información, vea "Trabajo con ventanas" en la página 83.
Window	Método **NewWindow**	Crea una ventana de documento. Para más información, vea "Trabajo con ventanas" en la página 83.
Window	Propiedad **Next**	Accede a la ventana siguiente de un documento. Para más información, vea "Trabajo con ventanas" en la página 83.
Window	Propiedad **Previous**	Accede a la ventana previa de un documento. Para más información, vea "Trabajo con ventanas" en la página 83.

Para información detallada sobre cualquier propiedad, método o evento, vea la sección "Referencia del Modelo de Objeto" en el archivo de Ayuda de Macros de CorelDRAW.

Para más información sobre actividades relacionadas con documentos, vea los siguientes temas:

- Creación de documentos.
- Apertura de documentos.
- Activación de documentos.
- Ajuste de las propiedades en documentos.
- Presentación de documentos.
- Modificación de documentos.
- Creación de grupos de comando para documentos.
- Guardado de documentos.
- Exportación de archivos desde documentos.
- Publicación de documentos como PDF.
- Impresión de documentos.
- Cierre de documentos.

Archivos de todos los formatos soportados pueden ser importados a CorelDRAW. Los archivos importados son colocados en capas del documento, así que la información sobre archivos importados es proporcionada en la sección sobre trabajo con capas (ver "Importación de archivos dentro de capas" en la página 101) en lugar de estar en esta sección de Trabajo con documentos.

Creación de documentos

El objeto **Application** tiene dos métodos para creación de documentos: **CreateDocument** y **CreateDocumentFromTemplate**.

El método **Application.CreateDocument** crea un documento vacío basado en el tamaño, orientación y estilo predeterminados de página :

```
Application.CreateDocument() As Document
```

El método **Application.CreateDocumentFromTemplate** crea un documento sin título desde una plantilla específica CorelDRAW (CDT):

```
Application.CreateDocumentFromTemplate(Template As String, _
                    [IncludeGraphics As Boolean = True])As Document
```

Ambas funciones devuelven una referencia al nuevo documento, de modo que son normalmente utilizadas de la siguiente manera:

```
Dim newDoc as Document
Set newDoc = CreateDocument
```

El nuevo documento se convierte inmediatamente en el documento activo y puede ser referenciado utilizando la propiedad **Application.ActiveDocument**. Para más información sobre esta propiedad, vea "Activación de documentos" en la página 81.

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sean activados al crear un documento:

- **AddinHook.New**
- **Application.DocumentNew**
- **GlobalMacroStorage.DocumentNew**

Apertura de documentos

Para abrir un documento se puede utilizar el método **Application.OpenDocument**:

```
Dim doc As Document
Set doc = OpenDocument("C:\graphic1.cdr")
```

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sean activados al crear un documento:

- **AddinHook.New**
- **Application.DocumentNew**
- **GlobalMacroStorage.DocumentNew**

Activación de documentos

Cada documento abierto en CorelDRAW es un miembro de la colección **Application.Documents**. Los documentos en esa colección aparecen en el orden en el cual fueron creados o abiertos.

Para reflejar el orden actual de apilamiento de los documentos en CorelDRAW, se debe utilizar la colección **Application.Windows**.

La propiedad **Application.ActiveDocument** proporciona acceso directo al documento activo — es decir, el documento que está al frente de todos los demás documentos en la ventana CorelDRAW. **ActiveDocument** es un objeto de tipo **Document** y, por consiguiente, tiene los mismos miembros — propiedades, métodos y objetos — que la clase **Document**.

Si ningún documento está abierto, **ActiveDocument** no devuelve nada. Se puede verificar si hay documentos abiertos utilizando el siguiente código VBA:

```
If Documents.Count = 0 Then
  MsgBox "No hay ningún documento abierto.", vbOK, "Sin Documentos"
  Exit Sub
End If
```

El método **Document.Activate** activa un documento de modo que éste puede ser referenciado por **ActiveDocument**. El siguiente código VBA activa el tercer documento abierto (si tres o más documentos están abiertos en CorelDRAW):

```
Documents(3).Activate
```

El uso del método **Document.Activate** en la propiedad **Application.ActiveDocument** no tiene efecto.

Si lo desea, puede especificar cuál documento abierto activar al referenciar una de las siguientes propiedades:

- **Document.FilePath** — revisa sólo la ruta de archivo (por ejemplo, **C:\Mis Documentos**).
- **Document.FileName** — revisa sólo el nombre de archivo (por ejemplo, **Graphic1.cdr**).
- **Document.FullFileName** — revisa tanto la ruta de archivo como el nombre de archivo (por ejemplo, **C:\Mis Documentos\Graphic1.cdr**).

Puede revisar el nombre de archivo de cada documento abierto utilizando el siguiente código VBA:

```
Public Function findDocument(filename As String) As Document
  Dim doc As Document
  For Each doc In Documents
    If doc.FileName = filename Then Exit For
    Set doc = Nothing
  Next doc
  Set findDocument = doc
End Function
```

Luego puede activar el documento devuelto al utilizar el método **Document.Activate**.

Configuración de las propiedades del documento

Puede especificar el punto de referencia, unidad de medida y escala de dibujo para un documento utilizando las propiedades correspondientes de la clase **Document**.

La propiedad **Document.ReferencePoint** especifica el punto de referencia para un documento. Este punto es referenciado cuando se posicionan los objetos en ese documento.

La propiedad **Document.Unit** especifica la unidad de medida para un documento. Esta unidad es utilizada para la posición y el tamaño de los objetos en ese documento.

La propiedad **Document.WorldScale** especifica la escala de dibujo para un documento. La escala de dibujo le permite hacer las distancias en un dibujo proporcionado a distancias del mundo real; por ejemplo, se puede especificar que 1 pulgada en el dibujo corresponda a 1 metro en el mundo físico.

Estas propiedades afectan a todos los elementos en su documento, tal como los objetos que usted dibuje. Para resultados óptimos, elija los ajustes que mejor se adapten a su solución de macro.

Presentación de documentos

CorelDRAW le permite mostrar simultáneamente múltiples ventanas en un solo documento. Por ejemplo, un documento grande puede ser mostrado con una ventana "zoomeada" en la esquina superior derecha del documento y otra "zoomeada" en la esquina inferior derecha. Aunque las ventanas individuales pueden ser "zoomeadas" y paneadas independientemente, el poner la página en una ventana afecta a todas las ventanas.

Al utilizar el Administrador de Vistas, se pueden crear vistas que tengan ajustes individuales de presentación. Al elegir una vista guardada se muestra la página que concuerde con los ajustes de esa vista.

En VBA, el objeto **Window** proporciona acceso a las ventanas que contienen cada objeto **View** (o vista de) para un documento dado. El objeto **Window** representa un bastidor, mientras que el objeto **View** muestra el documento en el interior de ese bastidor.

Además, al permitirle trabajar con ventanas y vistas, CorelDRAW le permite mostrar documentos al "zoomearlos" y panearlos.

Trabajando con ventanas

Cada objeto **Document** tiene una colección **Windows** para mostrar ese documento. Para alternar entre ventanas, utilice el método **Window.Activate**:

```
ActiveDocument.Windows(2).Activate
```

La propiedad **Document.ActiveWindow** proporciona acceso directo a la ventana activa — es decir, la ventana del documento que está al frente de todas las otras ventanas en CorelDRAW.

La ventana siguiente y la ventana previa para el documento activo son referenciadas en las propiedades **Window.Next** y **Window.Preview**:

```
ActiveWindow.Next.Activate
```

Para crear una nueva ventana, utilice el método **Window.NewWindow**:

```
ActiveWindow.NewWindow
```

Para cerrar una ventana (y el documento, si éste tiene sólo una ventana abierta), utilice el método **Window.Close**:

```
ActiveWindow.Close
```

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sean accionados al activar una ventana:

- **Application.WindowActivate**
- **GlobalMacroStorage.WindowActivate**

También puede utilizar manejadores de evento para responder a eventos que sean accionados por la desactivación de una ventana:

- **Application.WindowDeactivate**
- **GlobalMacroStorage.WindowDeactivate**

Trabajando con vistas

Tanto la propiedad **Window.ActiveView** como la propiedad **Document.Views** representan vistas de documentos. Cada objeto **Window** tiene un objeto **ActiveView**, el cual representa la vista actual del documento; al guardar los ajustes de despliegue en pantalla para un objeto **ActiveView** se crea una vista. En contraste, cada objeto **Document** tiene una colección de objetos **View** en su propiedad **Views**; al elegir un objeto **View** se activa la vista guardada correspondiente, la cual contiene los ajustes de despliegue de pantalla para el objeto **ActiveView** correspondiente.

La única forma de acceder a un objeto **ActiveView** es desde la propiedad **Window.ActiveView**.

Se puede crear un objeto **View** y añadirlo a una colección **Document.Views**. El siguiente código VBA añade los ajustes **ActiveView** actuales a la colección **Views**:

```
ActiveDocument.Views.AddActiveView "New View"
```

También se puede crear una vista con ajustes específicos utilizando el método **Document.CreateView**. El siguiente código VBA crea un nuevo objeto **View** que accede a la posición (3, 4) en pulgadas, utiliza un factor de zoom del 95% y muestra la página 6:

```
ActiveDocument.Unit = cdrInch
ActiveDocument.CreateView "New View 2", 3, 4, 95, 6
```

Para aplicar una vista guardada a la ventana activa, se llama al método **View.Activate**:

```
ActiveDocument.Views("New View").Activate
```

Zooming

Para "zoomear" un objeto **ActiveView** a una cantidad fija, ajuste la propiedad **ActiveView.Zoom** especificando un valor doble en porcentaje. Por ejemplo, el siguiente código VBA ajusta el factor de zoom a 200%:

```
ActiveWindow.ActiveView.Zoom = 200.0
```

También puede "zoomear" utilizando los siguientes métodos de la clase **ActiveView**:

- **SetActualSize**
- **ToFitAllObjects**
- **ToFitArea**
- **ToFitPage**
- **ToFitPageHeight**
- **ToFitPageWidth**
- **ToFitSelection**
- **ToFitShape**
- **ToFitShapeRange**

Paneo

Para "panear" un objeto **ActiveView**, se puede mover su origen modificando las propiedades **ActiveView.OriginX** y **ActiveView.OriginY**.

El siguiente código VBA panea el documento 5 pulgadas a la izquierda y 3 pulgadas arriba:

```
Dim av As ActiveView
ActiveDocument.Unit = cdrInch
Set av = ActiveWindow.ActiveView
av.OriginX = av.OriginX - 5
av.OriginY = av.OriginY + 3
```

Alternativamente, se puede utilizar el método **ActiveView.SetViewPoint**:

```
Dim av As ActiveView
ActiveDocument.Unit = cdrInch
Set av = ActiveWindow.ActiveView
av.SetViewPoint av.OriginX - 5, av.OriginY + 3
```

Modificación de documentos

Se puede modificar un documento independientemente si está activo o no.

El modificar un documento inactivo no activa ese documento. Para activar un documento, se debe utilizar su método **Activate**, como se explica en "Activación de documentos" en la página 81.

El siguiente código VBA añade una capa llamada "fooLayer" al tercer documento abierto:

```
Dim doc As Document
Set doc = Documents(3)
doc.ActivePage.CreateLayer "fooLayer"
```

El siguiente código VBA utiliza la función **findDocument()** para añadir una capa llamada "fooLayer" al documento inactivo llamado **barDoc.cdr**:

```
Dim doc As Document
Set doc = findDocument("barDoc.cdr")
If Not doc Is Nothing Then doc.ActivePage.CreateLayer "fooLayer"
```

Creación de grupos de comando para documentos

Dos métodos muy útiles de la clase **Document** se combinan para crear un "grupo de comando", el cual puede reducir una serie de acciones relacionadas con la programación del documento a un simple paso (y que se puede deshacer). Estos métodos — **BeginCommandGroup** y **EndCommandGroup** — son demostrados en el siguiente ejemplo VBA:

```
Dim sh As Shape
ActiveDocument.BeginCommandGroup "CreateCurveEllipse"
    Set sh = ActiveLayer.CreateEllipse(0, 1, 1, 0)
    sh.ConvertToCurves
ActiveDocument.EndCommandGroup
```

El código anterior establece la cadena **Undo** en el menú **Editar** como **Undo CreateCurveEllipse**. Al hacer click en este comando se deshace no sólo la operación **ConvertToCurves** sino también la operación **CreateEllipse**.

Un grupo de comando puede contener cientos de comandos si así se requiere. La creación de grupos de comando puede hacer que sus macros parezcan estar totalmente integradas dentro de CorelDRAW.

Guardado de documentos

CorelDRAW ofrece dos métodos para guardar documentos: **Document.SaveAs** y **Document.Save**.

El método **Document.SaveAs** guarda un documento utilizando la ruta y nombre de archivo especificados. Se puede utilizar este método para guardar un documento no guardado anteriormente o para guardar un documento existente en un archivo diferente.

El método **Document.SaveAs** proporciona un parámetro opcional que le permite acceder a la clase **StructSaveAsOptions** para especificar ajustes adicionales.

El método **Document.Save** guarda sobre un archivo de documento existente — es decir, utiliza la ruta y el nombre de archivo existente para el documento.

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sean activados al abrir el cuadro de diálogo **Guardar**:

- **Application.DocumentBeforeSave**
- **Document.BeforeSave**
- **GlobalMacroStorage.DocumentBeforeSave**

También puede utilizar manejadores de evento para responder a eventos que sean activados al guardar un documento y cerrar el cuadro de diálogo **Guardar**:

- **Application.DocumentAfterSave**
- **Document.AfterSave**
- **GlobalMacroStorage.DocumentAfterSave**

Finalmente, también se pueden utilizar manejadores de evento para responder a eventos que sean activados cuando el usuario responda a la solicitud de guardar un documento:

- **Application.QueryDocumentSave**
- **Document.QuerySave**
- **GlobalMacroStorage.QueryDocumentSave**

Exportación de archivos desde documentos

Archivos de todos los formatos soportados pueden ser exportados desde CorelDRAW.

Los archivos son exportados en el nivel **Document** porque el rango de objetos exportados puede extenderse sobre múltiples capas y múltiples páginas. Sin embargo, los archivos son importados en el nivel **Layer** porque cada objeto importado es asignado a una capa específica (vea "Importación de archivos dentro de capas" en la página 101).

La clase **Document** tiene tres métodos de exportación de archivos — **Export**, **ExportEx** y **ExportBitmap** — cada uno de los cuales puede exportar a formatos de bitmap y de vector.

La amplia selección de formatos de archivo que son soportados por CorelDRAW se debe al vasto número de filtros que están disponibles en la aplicación. Cada filtro le permite trabajar con los archivos de otras aplicaciones gráficas. Para aprender más acerca del trabajo con estos filtros, vea "Trabajando con filtros de importación y filtros de exportación" en la página 138.

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sean activados al abrir el cuadro de diálogo **Exportar**:

- **Application.DocumentBeforeExport**
- **Document.BeforeExport**
- **GlobalMacroStorage.DocumentBeforeExport**

También puede utilizar manejadores de evento para responder a eventos que sean activados al exportar un documento y cerrar el cuadro de diálogo **Exportar**:

- **Application.DocumentAfterExport**
- **Document.AfterExport**
- **GlobalMacroStorage.DocumentAfterExport**

Finalmente, también se pueden utilizar manejadores de evento para responder a eventos que sean activados cuando el usuario responda a la solicitud de exportar un documento:

- **Application.QueryDocumentExport**
- **Document.QueryExport**
- **GlobalMacroStorage.QueryDocumentExport**

Entendiendo el método Document.Export

Para exportar una página, se requiere sólo el método **Document.Export**, un nombre de archivo y un tipo de filtro. El siguiente código VBA exporta la página actual a un archivo de bitmap TIFF:

```
ActiveDocument.Export "C:\ThisPage.eps", cdrTIFF
```

Sin embargo, el código anterior otorga poco control sobre la salida de la imagen. Se obtiene más control al incluir el objeto **StructExportOptions**, como en el siguiente código VBA:

```
Dim expOpts As New StructExportOptions
expOpts.ImageType = cdrCMYKColorImage
expOpts.AntiAliasingType = cdrNormalAntiAliasing
expOpts.AntiAliasingType = cdrNormalAntiAliasing
expOpts.ResolutionX = 72
expOpts.ResolutionY = 72
expOpts.SizeX = 210
expOpts.SizeY = 297
ActiveDocument.Export "C:\ThisPage.eps", cdrTIFF, cdrCurrentPage, expOpts
```

Un objeto **StructPaletteOptions** también puede ser incluido en la llamada de la función para formatos de imagen basados en paleta si usted quiere proporcionar la configuración para generar automáticamente la paleta.

Entendiendo el método Document.ExportEx

El método **Document.ExportEx** es el mismo que el método **Document.Export**, excepto que **ExportEx** puede retomar las configuraciones del cuadro de diálogo para un filtro y aplicarlas a la exportación:

```
Dim eFilt As ExportFilter
Set eFilt = ActiveDocument.ExportEx("C:\ThisPage.eps", cdrEPS)
If eFilt.HasDialog = True Then
  If eFilt.ShowDialog = True Then
      eFilt.Finish
    End If
Else
  eFilt.Finish
End If
```

Entendiendo el método Document.ExportBitmap

El método **Document.ExportBitmap** es similar al método **Document.ExportEx** en el sentido de que devuelve un objeto **ExportFilter** que puede ser utilizado para mostrar el cuadro de diálogo **Exportar**. Sin embargo, el método **ExportBitmap** simplifica la codificación al tomar los miembros individuales del objeto **StructExportOptions** como parámetros:

```
Dim eFilt As ExportFilter
Set eFilt = ActiveDocument.ExportBitmap("C:\Selection.eps", _
              cdrTIFF, cdrSelection, cdrCMYKColorImage, _
              210, 297, 72, 72, cdrNormalAntiAliasing, _
              False, True, False, cdrCompressionLZW)
eFilt.Finish
```

Publicación de documentos como PDF

La publicación de documentos como PDF es un proceso de dos pasos. El primer paso es para especificar la configuración del PDF (aunque este paso puede ser saltado al especificar esos ajustes desde el interior de CorelDRAW o utilizando la configuración predeterminada). El segundo paso es para exportar el archivo.

Para especificar la configuración del PDF, se puede utilizar la propiedad **Document.PDFSettings**. Esta propiedad es un objeto de tipo **PDFVBASettings** y contiene propiedades para todas las configuraciones de PDF que pueden ser ajustadas en el cuadro de diálogo **Publicar como PDF**.

El siguiente código VBA exporta las páginas 2, 3 y 5 a un archivo PDF llamado **MyPDF.pdf**:

```
Dim doc As Document
Set doc = ActiveDocument
With doc.PDFSettings
  .Author = "Corel Corporation”
  .Bookmarks = True
  .ColorMode = pdfRGB
  .ComplexFillsAsBitmaps = False
  .CompressText = True
  .DownsampleGray = True
  .EmbedBaseFonts = True
  .EmbedFonts = True
  .Hyperlinks = True
  .Keywords = "Test, Example, Corel, CorelDRAW, PublishToPDF”
  .Linearize = True
  .PageRange = "2-3, 5"
  .pdfVersion = pdfVersion13
  .PublishRange = pdfPageRange
  .TrueTypeToType1 = True
End With
doc.PublishToPDF "C:\MyPDF.pdf”
```

El siguiente ejemplo VBA otorga más control al usuario al mostrar el cuadro de diálogo **Configuración para Publicar como PDF**:

```
Dim doc As Document
Set doc = ActiveDocument
If doc.PDFSettings.ShowDialog = True Then
  doc.PublishToPDF "C:\MyPDF.pdf”
End If
```

Los perfiles de configuración de PDF pueden ser guardados y cargados utilizando el método **PDFVBASettings.Save** y el método **PDFVBASettings.Load** (respectivamente)

Impresión de documentos

El uso de VBA para imprimir documentos es simple: casi todas las configuraciones que están disponibles en el cuadro de diálogo **Imprimir** de CorelDRAW están disponibles en la propiedad **Document.PrintSettings**. Cuando estas propiedades son ajustadas, la impresión del documento es simplemente cuestión de llamar al método **Document.PrintOut**.

Por ejemplo, el siguiente código VBA imprime tres copias de las páginas 1, 3 y 4 en una impresora PostScript a un nivel 3:

```
With ActiveDocument.PrintSettings
  .Copies = 3
  .PrintRange = prnPageRange
  .PageRange = "1, 3-4"
  .Options.PrintJobInfo = True
  With .PostScript
    .DownloadType1 = True
    .Level = prnPSLevel3
  End With
End With
ActiveDocument.PrintOut
```

Cada página en el cuadro de diálogo **Imprimir** tiene una clase correspondiente de modelo de objeto que contiene todas las configuraciones para esa página. La siguiente tabla enlista estas clases.

Página en el Cuadro de Diálogo Imprimir	Clase en el Modelo de Objeto
General	PrintSettings
Diseño	PrintSettings y PrintLayout
Separaciones	PrintSeparations y PrintTrapping
Preimpresión	PrintPrepress
PostScript	PrintPostScript
Compuesta	PrintOptions

No se pueden ajustar las opciones de diseño de página en VBA. Sin embargo, de ser necesario, se puede abrir el cuadro de diálogo **Imprimir** utilizando el método **PrintSettings.ShowDialog**.

Se pueden imprimir sólo los objetos seleccionados en un documento ajustando la propiedad **PrintSettings.PrintRange** a **prnSelection**.

Se puede utilizar una impresora específica en la colección **Application.Printers** especificándola en la propiedad **PrintSettings.Printer**.

Se puede guardar un perfil de impresión utilizando el método **PrintSettings.Save**.

Se puede acceder a un perfil de impresión guardado utilizando el método **PrintSettings.Load**, pero asegurándose de especificar la ruta completa al perfil.

Se pueden restaurar los ajustes de impresión utilizando el método **PrintSettings.Reset**.

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sean activados al abrir el cuadro de diálogo **Imprimir**:

- **Application.DocumentBeforePrint**
- **Document.BeforePrint**
- **GlobalMacroStorage.DocumentBeforePrint**

También puede utilizar manejadores de evento para responder a eventos que sean activados al imprimir un documento y cerrar el cuadro de diálogo **Imprimir**:

- **Application.DocumentAfterPrint**
- **Document.AfterPrint**
- **GlobalMacroStorage.DocumentAfterPrint**

Finalmente, también puede utilizar manejadores de evento para responder a eventos que sean activados cuando el usuario responda a la solicitud para imprimir un documento:

- **Application.QueryDocumentPrint**
- **Document.QueryPrint**
- **GlobalMacroStorage.QueryDocumentPrint**

Cierre de documentos

Se puede cerrar un documento llamando al método **Document.Close**.

El siguiente código VBA cierra el documento activo y activa el documento detrás de él en CorelDRAW:

```
ActiveDocument.Close
```

Si el código cierra un documento que no esté activo, el documento referenciado por la propiedad **Application.ActiveDocument** no cambia.

Se debe probar explícitamente la propiedad **Dirty** para un documento y tomar una acción apropiada si ese documento ha sido modificado.

También se puede cerrar un documento utilizando el método **Close** del objeto **Document** en sí (como en `doc.Close`).

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sean activados al cerrar un documento:

- **Application.DocumentClose**
- **Document.Close**
- **GlobalMacroStorage.DocumentClose**

También puede utilizar manejadores de evento para responder a eventos que sean activados cuando el usuario responda a la solicitud para cerrar un documento:

- **Document.QueryClose**
- **GlobalMacroStorage.QueryDocumentClose**
- **GlobalMacroStorage.QueryDocumentClose**

Trabajo con páginas

Cada página, u objeto **Page**, es un miembro de la colección **Document.Pages** para el documento en el cual ésta aparece. Las páginas en una colección **Document.Pages** aparecen en el mismo orden en el que aparecen en ese documento — por ejemplo, la quinta página en el documento activo es ActiveDocument.Pages.Item(5). Si se añaden, reordenan o borran páginas, la colección afectada **Pages** es inmediatamente actualizada para reflejar el nuevo orden de páginas de ese documento.

CorelDRAW proporciona un número de propiedades, métodos y eventos para trabajar con páginas, de los cuales se presentan los más útiles en la siguiente tabla:

Clase	Miembro	Descripción
Application	Propiedad **ActivePage**	Proporciona acceso directo a la página activa del documento activo. Para más información, vea "Activando páginas" en la página 94.
Application	Propiedad **PageSizes**	Contiene la colección de tamaños de página definidos para la aplicación. Para más información, vea "Uso de los tamaños de página definidos" en la página 96.
Document	Propiedad **ActivePage**	Proporciona acceso directo a la página activa de un documento. Para más información, vea "Activando páginas" en la página 94.
Document	Método **AddPages** o Método **AddPagesEx**	Añade páginas en blanco al final de un documento. Para más información, vea "Creación de páginas" en la página 94.
Document	Método **InsertPages** o Método **InsertPagesEx**	Inserta páginas en la localización especificada en un documento. Para más información, vea "Creación de páginas" en la página 94.
Document	Propiedad **MasterPage**	Especifica el tamaño de página por defecto. Para más información, vea "Especificando el tamaño de página por defecto" en la página 96.
Document	Evento **PageActivate**	Se acciona cuando una página es activada. Para más información, vea "Activando páginas" en la página 94.
Document	Evento **PageChange**	Se acciona cuando una página es desactivada. Para más información, vea "Activando páginas" en la página 94.
Document	Evento **PageCreate**	Se acciona cuando una página es creada. Para más información, vea "Creación de páginas" en la página 94.
Document	Evento **PageDelete**	Se acciona cuando una página es eliminada. Para más información, vea "Eliminación de páginas" en la página 94.
Document	Propiedad **Pages**	Contiene la colección de páginas para un documento. Para más información, vea "Activando páginas" en la página 94.

Clase	Miembro	Descripción
Page	Método **Activate**	Activa una página. Para más información, vea "Activando páginas" en la página 94.
Page	Método **Delete**	Elimina una página. Para más información, vea "Eliminación de páginas" en la página 97.
Page	Método **MoveTo**	Mueve una página a la localización especificada en un documento. Para más información, vea "Reordenamiento de páginas" en la página 95.
Page	Método **SetSize**	Ajusta el tamaño de una página. Para más información, vea "Especificando el tamaño y orientación de páginas" en la página 95.
PageSize	Propiedad **BuiltIn**	Devuelve **True** si el tamaño de página está incorporado (en vez de estar definido por el usuario). Para más información, vea "Uso de tamaños de página definidos" en la página 96.
PageSize	Método **Delete**	Elimina un tamaño de página definido por el usuario. Para más información, vea "Uso de tamaños de página definidos" en la página 96.
PageSize	Propiedad **Height**	Especifica la altura de un tamaño de página definido. Para más información, vea "Uso de tamaños de página definidos" en la página 96.
PageSize	Propiedad **Name**	Especifica el nombre de un tamaño de página definido. Para más información, vea "Uso de tamaños de página definidos" en la página 96.
PageSize	Propiedad **Width**	Especifica la anchura de un tamaño de página definido. Para más información, vea "Uso de tamaños de página definidos" en la página 96.

Para información detallada sobre cualquier propiedad, método o evento, vea la sección "Referencia del Modelo de Objeto" en el archivo de Ayuda de Macros en CorelDRAW.

Para más información sobre actividades relacionadas con páginas, vea los siguientes temas:

- Creación de páginas
- Activación de páginas
- Reordenamiento de páginas
- Asignación de tamaño de páginas
- Modificación de páginas
- Eliminación de páginas

Creación de páginas

Los métodos para la creación de páginas pertenecen a la clase **Document**.

Tanto el método **Document.AddPages** como el método **Document.AddPagesEx** añaden el número especificado de páginas al final de un documento. La diferencia entre estos métodos es que **AddPages** utiliza el tamaño de página por defecto, mientras que **AddPagesEx** utiliza un tamaño especificado.

Similarmente, tanto el método **Document.InsertPages** como el método **Document.InsertPagesEx** insertan el número de páginas especificado en la localización especificada en un documento. La diferencia entre estos métodos es que **InsertPages** utiliza el tamaño de página por defecto, mientras que **InsertPagesEx** utiliza un tamaño especificado.

A manera de ejemplo, el siguiente código VBA utiliza el método AddPages para añadir tres páginas de tamaño predeterminado (por defecto) al final de un documento:

```
Public Function AddSomeSimplePages() as Page
    Set AddSomeSimplePages = ActiveDocument.AddPages(3)
End Function
```

El siguiente ejemplo de código VBA utiliza el método AddPagesEx para añadir al final del documento tres páginas de 8.5 pulgadas de ancho y11 pulgadas de alto:

```
Public Function AddSomeSpecifiedPages() as Page
    Dim doc as Document
    Set doc = ActiveDocument
    doc.Unit = cdrInch
    Set AddSomeSpecifiedPages = doc.AddPagesEx(3, 8.5, 11)
End Function
```

Los ejemplos anteriores afectan a la primer página que fue añadida; todas las otras páginas añadidas emulan a esta página. Por consiguiente, se puede hacer referencia a cualquiera de las páginas añadidas incrementando la propiedad **Index** de la página afectada:

```
Dim firstNewPage As Page, secondNewPage As Page
Set firstNewPage = AddSomeSimplePages
Set secondNewPage = ActiveDocument.Pages(firstNewPage.Index + 1)
```

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sean activados al crear una página:

- **Document.PageCreate**

Activación de páginas

Cada página es un miembro de la colección **Document.Pages** para el documento en el cual ésta aparece. Las páginas en una colección **Document.Pages** aparecen en el mismo orden en el que aparecen en ese documento — por ejemplo, la quinta página en el documento activo es `ActiveDocument.Pages.Item(5)`. Si se añaden, reordenan o borran página, la colección **Pages** afectada es actualizada inmediatamente para reflejar el nuevo orden de páginas en ese documento.

Se puede acceder a la página activa del documento activo utilizando la propiedad **Application.ActivePage** (o `ActiveDocument.ActivePage`, o simplemente `ActivePage`). Una referencia a la página activa del documento activo, de tipo **Page**, es devuelta.

```
Dim pg As Page
Set pg = ActivePage
```

Se puede acceder a la página activa de un documento, independientemente si ese documento está activo o no, utilizando la propiedad **ActivePage** para ese documento:

```
Public Function getDocsActivePage(doc As Document) As Page
    Set getDocsActivePage = doc.ActivePage
End Function
```

Se pueden conmutar páginas encontrando la página deseada y luego invocando su método **Activate**. El siguiente código VBA activa la página 3 en un documento CorelDRAW:

```
ActiveDocument.Pages(3).Activate
```

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sean accionados al activar una página

- **Document.PageActivate**

También puede utilizar manejadores de evento para responder a eventos que sean accionados al desactivar una página.

- **Document.PageChange**

Reordenamiento de páginas

Una página puede ser movida a otra localización en un documento utilizando su método **MoveTo**. El siguiente código VBA mueve la página 2 a la posición de la página 4:

```
ActiveDocument.Pages(2).MoveTo 4
```

La activación de una página en un documento inactivo no activa ese documento. Para activar un documento, se debe utilizar su método **Activate**, como se explica en "Activación de documentos" en la página 81.

Dimensionamiento de páginas

Se puede especificar el tamaño y orientación de las páginas al especificar el tamaño de página por defecto, y así utilizar tamaños de página definidos.

Especificando el tamaño y orientación de páginas

Se puede dimensionar una página utilizando su método **SetSize**, el cual aplica dos valores de tamaño (anchura y altura) a la página. El siguiente código VBA cambia el tamaño de la página activa en el documento activo a A4:

```
ActiveDocument.Unit = cdrMillimeter
ActivePage.SetSize 210, 297
ActivePage.Orientation = cdrLandscape
```

Para el método **SetSize**, el primer número es siempre la anchura de la página y el segundo número es siempre la altura. Al cambiar el orden de los dos números se alterna la orientación de la página.

Especificando el tamaño de página por defecto

El tamaño de página por defecto para un documento está determinado por el valor del elemento que tiene un índice de 0 en la colección **Document.Pages**. Se puede especificar el tamaño de página por defecto al cambiar el valor de este elemento:

```
Dim doc As Document
Set doc = ActiveDocument
doc.Unit = cdrMillimeter
doc.Pages(0).SetSize 297, 210
```

Alternativamente, se puede utilizar la propiedad **Document.MasterPage** para especificar el tamaño de página por defecto:

```
Dim doc As Document
Set doc = ActiveDocument
doc.Unit = cdrMillimeter
doc.MasterPage.SetSize 297, 210
```

Uso de tamaños de página definidos

Los tamaños de página pueden ser definidos tanto por CorelDRAW como por el usuario. Todos los tamaños de página definidos son almacenados en la colección **Application.PageSizes**, y el nombre de cada objeto **PageSize** en esa colección está definido por su propiedad **Name**:

```
Dim pageSizeName As String
pageSizeName = Application.PageSizes(3).Name
```

Los tamaños de página pueden ser especificados por nombre. Por ejemplo, el siguiente código VBA obtiene el objeto **PageName** llamado "Business Card":

```
Dim thisSize As PageSize
Set thisSize = Application.PageSizes("Business Card")
```

Se pueden obtener las dimensiones actuales de un objeto **PageSize** utilizando sus propiedades **Width** y **Height**. El siguiente código VBA recupera la anchura y altura (en milímetros) del tercer objeto **PageSize**:

```
Dim pageWidth As Double, pageHeight As Double
Application.Unit = cdrMillimeter
pageWidth = Application.PageSizes(3).Width
pageHeight = Application.PageSizes(3).Height
```

Aunque cada objeto **PageSize** proporciona un método **Delete**, este método puede ser utilizado sólo en tamaños de página definidos por el usuario. Se puede determinar si un objeto **PageSize** está definido por el usuario probando su propiedad Booleana **BuiltIn**:

```
Public Sub deletePageSize(thisSize As PageSize)
    If Not thisSize.BuiltIn Then thisSize.Delete
End Sub
```

Se puede especificar una unidad de medida particular para un tamaño de página al ajustar las unidades del documento antes de obtener su anchura y altura.

Modificación de páginas

Se puede modificar una página independientemente de si está activa o no.

La activación de una página en un documento inactivo no activa ese documento. Para activar un documento, se debe utilizar su método **Activate**, como se explica en "Activación de documentos" en la página 81.

Al referenciar explícitamente la página que se quiera modificar, se pueden hacer esos cambios sin activar la página. El siguiente código VBA elimina todas las formas de la página 3 del documento activo sin activar esa página:

```
Public Sub DeleteShapesFromPage3()
  Dim doc As Document
  Set doc = ActiveDocument
  doc.Pages(3).Shapes.All.Delete
End Sub
```

Eliminación de páginas

Se puede eliminar una página utilizando su método **Delete**, como en el siguiente ejemplo VBA:

```
ActivePage.Delete
```

El método **Page.Delete** elimina todas las formas de esa página, elimina la página de la colección **Pages** para ese documento y luego actualiza esa colección para reflejar el cambio.

Si se quiere eliminar más de una página, se debe utilizar el método **Delete** para cada página no deseada. Sin embargo, no se pueden borrar todas las páginas en un documento. Se puede evitar tratar de borrar la última página restante en un documento utilizando el siguiente código VBA:

```
If ActiveDocument.Pages.Count > 1 Then ActivePage.Delete
```

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sean activados al borrar una página:

- **Document.PageDelete**

Trabajo con capas

Las capas son planos invisibles que le permiten organizar los objetos sobre una página. Se pueden agrupar objetos relacionados dentro de las capas, y se puede cambiar el orden vertical (u "orden de apilamiento") de esas capas para cambiar la apariencia de la página. Las capas maestras se aplican a todas las capas en un documento, mientras que las capas locales se aplican a una sola página.

Cada capa, u objeto **Layer**, es un miembro de la colección **Page.Layers** para la página en la cual ésta aparece. Las capas en la colección **Page.Layers** aparecen en el mismo orden en el que aparecen en esa página — la primer capa es la que está en la parte superior de la "pila", y la última capa es la que está en la parte inferior. Si se añaden, reordenan o borran capas, la colección **Page.Layers** afectada se actualiza inmediatamente para reflejar el nuevo orden de las capas en esa página.

CorelDRAW proporciona un número de propiedades, métodos y eventos para trabajar con capas, entre los cuales se enlistan los más útiles en la siguiente tabla.

Clase	Miembro	Descripción
Document	Propiedad **ActiveLayer**	Proporciona acceso directo a la capa activa del documento. Para más información, vea "Activación de capas" en la página 99.
Document	Evento **LayerActivate**	Se acciona cuando una capa es activada. Para más información, vea "Activación de capas" en la página 99.
Document	Evento **LayerChange**	Se acciona cuando una capa es desactivada. Para más información, vea "Activación de capas" en la página 99.
Document	Evento **LayerCreate**	Se acciona cuando una capa es creada. Para más información, vea "Creación de capas" en la página 99.
Document	Evento **LayerDelete**	Se acciona cuando una capa es eliminada. Para más información, vea "Eliminación de capas" en la página 102.
Layer	Método **Activate**	Activa una capa. Para más información, vea "Activación de capas" en la página 99.
Layer	Método **Delete**	Elimina una capa. Para más información, vea "Eliminación de capas" en la página 102.
Layer	Propiedad **Editable**	Controla si una capa es editable. Para más información, vea "Bloqueo y ocultamiento de capas" en la página 100.
Layer	Método **Import** o Método **ImportEx**	Importa un archivo dentro de una capa. Para más información, vea "Importación de archivos dentro de capas" en la página 101.
Layer	Método **MoveAbove** o Método **MoveBelow**	Mueve una capa. Para más información, vea "Reordenamiento de capas" en la página 100.
Layer	Propiedad **Name**	Especifica el nombre de una capa. Para más información, vea "Renombramiento de capas" en la página 100.
Layer	Propiedad **Visible**	Controla si el contenido de una capa está visible. Para más información, vea "Bloqueo y ocultamiento de capas" en la página 100.
Page	Propiedad **ActiveLayer**	Proporciona acceso directo a la capa activa de una página. Para más información, vea "Activación de capas" en la página 99.
Page	Método **CreateLayer**	Inserta una nueva capa en la parte superior de la lista de capas no maestras. Para más información, vea "Creación de capas" en la página 99.

Clase	Miembro	Descripción
Page	Propiedad **Layers**	Contiene la colección de capas de una página. Para más información, vea "Activación de capas" en la página 99.

Para información detallada sobre cualquier propiedad, método o evento, vea la sección "Referencia del Modelo de Objeto" en el archivo de Ayuda de Macros en CorelDRAW.

Para más información sobre actividades relacionadas con capas, vea los siguientes temas:

- Creación de capas.
- Activación de capas.
- Bloqueo y ocultamiento de capas.
- Reordenamiento de capas.
- Renombramiento de capas.
- Importación de archivos dentro de capas.
- Eliminación de capas.

Creación de capas

Se puede crear una capa utilizando el método **Page.CreateLayer**. Al crear una capa se inserta una nueva capa en la parte superior de la lista de capas no maestras.

El siguiente código VBA crea una nueva capa llamada "My New Layer":

```
ActivePage.CreateLayer "My New Layer"
```

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sean accionados al activar una capa:

- **Document.LayerCreate**

Activación de capas

Cada capa es un miembro de la colección **Page.Layers** para la página en la cual aparece. Las capas en una colección **Page.Layers** aparecen en el mismo orden en el que aparecen en esa página — la primer capa es la que está en la cima de la "pila", y la última capa es la que está en el fondo. Si se añaden, reordenan o eliminan capas, la colección **Page.Layers** afectada se actualiza inmediatamente para reflejar el nuevo orden de capas en esa página.

La propiedad **Document.ActiveLayer** proporciona acceso directo a la capa activa de un documento, mientras que la propiedad **Page.ActiveLayer** proporciona acceso directo a la capa activa de una página.

Se puede activar una capa utilizando el método **Layer.Activate**:

```
ActivePage.Layers("Layer 1").Activate
```

El activar una capa bloqueada no la desbloquea. Similarmente, si se activa una capa oculta, ésta no se hace visible. Para más información sobre desbloqueo y exposición de capas, vea "Bloqueo y ocultamiento de capas" en la página 100.

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sean accionados al activar una capa:

- **Document.LayerActivate**

También puede utilizar manejadores de evento para responder a eventos que sean accionados al desactivar una capa:

- **Document.LayerChange**

Bloqueo y ocultamiento de capas

Los objetos **Layer** presentan las propiedades **Editable** y **Visible**, las cuales controlan (respectivamente) si la capa es editable y si su contenido está visible. Ambas propiedades son Booleanas. Al configurar ambas propiedades a **True**, se desbloquea y muestra la capa para editarla. Si se configura una de las propiedades a **False**, no importa cuál, se bloquea la capa de modo que no puede ser editada.

La siguiente muestra de código VBA bloquea, pero muestra, la capa en la página activa:

```
ActivePage.Layers("Layer 1").Visible = True
ActivePage.Layers("Layer 1").Editable = False
```

El resultado de cualquier cambio para estas propiedades se muestra inmediatamente en el Administrador de Objetos.

El ejemplo anterior afecta sólo a la capa activa. Se puede acceder a las configuraciones de capa para una página dada al especificar una página de la colección **Document.Pages**, o al hacer referencia a la propiedad **Document.ActivePage**. Para hacer los cambios en todas las páginas de un documento, se utiliza la propiedad **Document.MasterPage**:

```
ActiveDocument.MasterPage.Layers("Layer 1").Visible = True
```

Para más información sobre trabajo con páginas, vea "Trabajando con páginas" en la página 92.

Reordenamiento de capas

Se pueden reordenar las capas utilizando los siguientes dos métodos de la clase **Layer**: **MoveAbove** y **MoveBelow**. Ambos métodos mueven la capa especificada hacia arriba o hacia abajo de la capa que esté referenciada como un parámetro.

El siguiente código VBA mueve la capa llamada "Layer 1" hacia abajo de la capa "Guides":

```
Dim pageLayers As Layers
Set pageLayers = ActivePage.Layers
pageLayers("Layer 1").MoveBelow pageLayers("Guides")
```

El cambio se refleja inmediatamente en el Administrador de Objetos (aunque puede estar aparentemente sólo en el Administrador de Capas).

Renombramiento de capas

Se puede renombrar una capa al editar su propiedad **Name**.

El siguiente código VBA renombra "Layer 1" como "Layer with a New Name":

```
ActivePage.Layers("Layer 1").Name = "Layer with a New Name"
```

Importación de archivos dentro de capas

Los archivos de todos los formatos soportados pueden ser importados a CorelDRAW.

Los archivos son importados en el nivel **Layer** porque cada objeto importado es asignado a una capa especificada de una página especificada. Sin embargo, los archivos son exportados en el nivel **Document** porque el rango de objetos exportados puede extenderse sobre múltiples capas y múltiples páginas (vea "Exportación de archivos desde documentos" en la página 86).

La clase **Layer** tiene dos métodos para importar archivos: **Import** e **ImportEx**.

La amplia selección de formatos de archivo que son soportados por CorelDRAW se debe al vasto número de filtros que están disponibles para la aplicación. Cada filtro le permite trabajar con los archivos de otras aplicaciones gráficas. Para aprender más sobre el trabajo con estos filtros, vea "Trabajo con filtros de importación y exportación" en la página 138.

Entendiendo el método Layer.Import

El método **Layer.Import** proporciona funcionalidad básica para importación de archivos.

El siguiente código VBA importa el archivo **C:\logotype.gif** dentro de la capa activa en el centro de la página:

```
ActiveLayer.Import "C:\logotype.gif"
```

Al importar un archivo se selecciona el contenido de ese archivo y se deselecciona cualquier otro objeto seleccionado en el documento. Por consiguiente se pueden reposicionar o redimensionar los objetos importados obteniendo la selección del documento:

```
ActiveDocument.Unit = cdrInch
ActiveSelection.SetSize 3, 2
```

Algunos formatos de archivo pueden ser importados utilizando uno de varios filtros, así que es importante entender los beneficios de cada filtro disponible. Por ejemplo, cuando se importa un archivo Encapsulated PostScript (EPS), se puede elegir entre el filtro EPS y el filtro PDF. El filtro EPS le permite hacer lo siguiente:

- importar un archivo EPS como un objeto ubicable que se puede imprimir mas no modificar.
- interpretar la porción PostScript del archivo, de modo que se pueda importar la ilustración original desde el interior del archivo en vez de su encabezado de baja resolución.

Para especificar cuál filtro se va a utilizar, se puede incluir el parámetro opcional **Filter**, como en el siguiente ejemplo VBA:

```
ActiveLayer.Import "C:\map.eps", cdrPSInterpreted
```

Entendiendo el método Layer.ImportEx

El método **Layer.ImportEx** proporciona mucho mejor control sobre los filtros de importación a través del uso opcional de un objeto **StructImportOptions**. El siguiente código VBA importa el archivo especificado como un archivo vinculado:

```
Dim iFilt As ImportFilter
Dim importProps As New StructImportOptions
importProps.LinkBitmapExternally = True
Set iFilt = ActiveLayer.ImportEx("C:\world-map.epsf", cdrAutoSense, importProps)
iFilt.Finish
```

Eliminación de capas

Como se discutió previamente, cada capa es un miembro de la colección **Page.Layers** para la página en la cual aparece.

Se puede eliminar una capa llamando a su método **Delete**. La eliminación de una capa remueve esa capa del documento, llevándose con ella todas las formas en esa capa en todas las páginas del documento.

El siguiente código VBA elimina la capa llamada "Layer 1":

```
ActivePage.Layers("Layer 1").Delete
```

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sean accionados al eliminar una capa.

- **Document.LayerDelete**

Trabajo con formas

Cada documento CorelDRAW está constituido por formas, u objetos **Shape**, los cuales son creados utilizando las herramientas de dibujo. Cualquier cambio hecho en las propiedades de una forma — tal como mover la forma, cambiar su tamaño o aplicarle un nuevo relleno — se visualiza inmediatamente en el modelo de objeto.

Las formas en una página de un documento son almacenadas en capas. Cada capa es un miembro de la colección **Layer.Shapes** para la página en la cual aparece. Las formas en una colección **Layer.Shapes** aparecen en el mismo orden en el que aparecen en la capa — la primer forma es la que está en la cima de la "pila" y la última forma es la que está en el fondo. Si se añaden, reordenan o eliminan formas, la colección **Layer.Shapes** afectada se actualiza inmediatamente para reflejar el nuevo orden de formas en esa capa.

Además, cada página del documento tiene una colección **Shapes**, la cual contiene todas las colecciones **Layer.Shapes** para esa página. La primer forma en la colección **Page.Shapes** es la que está hasta el tope de esa página, y la última forma es la que está hasta el fondo.

CorelDRAW proporciona un número de propiedades, métodos y eventos para trabajar con formas, entre los cuales se enlistan los más útiles en la siguiente tabla:

Clase	Miembro	Descripción
AddinHook	Evento **ShapeCreated**	Se activa cuando una forma es creada. Para más información vea "Creación de formas" en la página 112.
Application	Método **CreateCurve**	Crea una línea o una curva "en memoria". Para más información vea "Creación de líneas y curvas" en la página 114.
Application	Evento **SelectionChange**	Se activa cuando una selección es desactivada. Para más información vea "Deselección de formas" en la página 124.

Clase	Miembro	Descripción
Application	Propiedad **SymbolLibraries**	Contiene la colección de todas las bibliotecas de símbolos externos para la aplicación. Para más información vea "Creación de símbolos" en la página 118.
Color	Método **CopyAssign**	Copia un color del relleno o contorno de una forma a otra. Para más información vea "Trabajo con color" en la página 134.
Color	Método **Type**	Especifica el modelo de color para un color de forma. Para más información vea "Trabajo con color" en la página 134.
Curve	Método **CreateSubPath**	Añade un subtrayecto a una línea o a una curva. Para más información vea "Creación de líneas y curvas" en la página 114.
Document	Método **ClearSelection**	Deselecciona todos los objetos en un documento. Para más información vea "Deselección de formas" en la página 124.
Document	Método **Selection**	Devuelve, como un solo objeto **Shape**, todos los objetos seleccionados en un documento. Para más información vea "Accesando directamente a las selecciones" en la página 122.
Document	Evento **SelectionChange**	Se activa cuando una selección es desactivada. Para más información vea "Deselección de formas" en la página 124.
Document	Propiedad **SelectionRange**	Devuelve, como un objeto **ShapeRange**, todos los objetos seleccionados en un documento. Para más información vea "Accesando a copias de selecciones" en la página 123.
Document	Evento **ShapeChange**	Se activa cuando una forma es desactivada. Para más información vea "Deselección de formas" en la página 124.
Document	Evento **ShapeCreate**	Se activa cuando una forma es creada. Para más información vea "Creación de formas" en la página 112.

Clase	Miembro	Descripción
Document	Evento **ShapeDelete**	Se activa cuando una forma es eliminada. Para más información vea "Eliminación de formas" en la página 137.
Document	Evento **ShapeDistort**	Se activa cuando una forma es distorsionada. Para más información vea "Aplicando distorsiones" en la página 136.
Document	Evento **ShapeMove**	Se activa cuando una forma es posicionada. Para más información vea "Posicionando de formas" en la página 129.
Document	Evento **ShapeTransform**	Se activa cuando una forma es transformada. Para más información vea "Transformación de formas" en la página 125.
Document	Propiedad **SymbolLibrary**	Devuelve la biblioteca de símbolos internos para un documento. Para más información vea "Creación de símbolos" en la página 118.
Fill	Método **ApplyFountainFill**	Aplica un relleno degradado a una forma. Para más información vea "Aplicación de rellenos degradados" en la página 130.
Fill	Método **ApplyHatchFill**	Aplica un relleno de malla a una forma. Para más información vea "Aplicación de rellenos de tramado" en la página 132.
Fill	Método **ApplyPatternFill**	Aplica un relleno de patrón a una forma. Para más información vea "Aplicación de rellenos de patrón" en la página 131.
Fill	Método **ApplyPostScriptFill**	Aplica un relleno PostScript a una forma. Para más información vea "Aplicación de rellenos PostScript" en la página 132.
Fill	Método **ApplyTextureFill**	Aplica un relleno de textura a una forma. Para más información vea "Aplicación de rellenos de textura" en la página 132.
Fill	Método **ApplyUniformFill**	Aplica un relleno uniforme a una forma. Para más información vea "Aplicación de rellenos uniformes" en la página 130.

Clase	Miembro	Descripción
Fill	Propiedad **Fountain**	Especifica las propiedades de relleno degradado de una forma. Para más información vea "Aplicación de rellenos degradados" en la página 130.
Fill	Propiedad **Hatch**	Especifica las propiedades de relleno de malla de una forma. Para más información vea "Aplicación de rellenos de malla" en la página 132.
Fill	Propiedad **Pattern**	Especifica las propiedades de relleno de patrón de una forma. Para más información vea "Aplicación de rellenos de patrón" en la página 131.
Fill	Propiedad **PostScriptFill**	Especifica las propiedades de relleno PostScript de una forma. Para más información vea "Aplicación de rellenos PostScript" en la página 132.
Fill	Propiedad **Texture**	Especifica las propiedades de relleno de textura de una forma. Para más información vea "Aplicación de rellenos de textura" en la página 132.
Fill	Propiedad **Type**	Especifica el tipo de relleno que se aplica a una forma. Para más información vea "Coloreando formas" en la página 129.
Fill	Propiedad **UniformColor**	Especifica las propiedades de relleno uniforme de una forma. Para más información vea "Aplicación de rellenos uniformes" en la página 130.
FountainColor	Método **Move**	Mueve un color en el relleno degradado de una forma. Para más información vea "Aplicación de rellenos degradados" en la página 130.
FountainColors	Método **Add**	Añade un color al relleno degradado de una forma. Para más información vea "Aplicación de rellenos degradados" en la página 130.
FountainColors	Propiedad **Count**	Cuenta el número de colores entre el color inicial y el color final en el relleno degradado de una forma. Para más información vea "Aplicación de rellenos degradados" en la página 130.
GlobalMacroStorage	Evento **SelectionChange**	Es accionado cuando una selección es desactivada. Para más información vea "Deselección de formas" en la página 124.

Clase	Miembro	Descripción
Layer	Método **CreateArtisticText** o Método **CreateArtisticTextWide**	Crea un objeto de texto artístico en la capa especificada. Para más información vea "Creación de objetos de texto" en la página 116.
Layer	Método **CreateCurve**	Crea, en la capa especificada, una línea o una curva que es creada "en memoria" utilizando el método **Application.CreateCurve** Para más información vea "Creación de líneas y curvas" en la página 114.
Layer	Método **CreateCurveSegment** o Método **CreateCurveSegment2**	Crea una curva básica en la capa especificada. Para más información vea "Creación de líneas y curvas" en la página 114.
Layer	Método **CreateEllipse**, Método **CreateEllipse2** o Método **CreateEllipseRect**	Crea una elipse en la capa especificada. Para más información vea "Creación de elipses" en la página 113.
Layer	Método **CreateLineSegment**	Crea una línea básica en la capa especificada. Para más información vea "Creación de líneas y curvas" en la página 114.
Layer	Método **CreateParagraphText** o Método **CreateParagraphTextWide**	Crea un objeto de texto de párrafo en la capa especificada. Para más información vea "Creación de objetos de texto" en la página 116.
Layer	Método **CreateRectangle**, Método **CreateRectangle2** o Método **CreateRectangleRect**	Crea un rectángulo en la capa especificada. Para más información vea "Creación de rectángulos" en la página 112.
Layer	Propiedad **Shapes**	Contiene la colección de formas de una capa. Para más información vea "Selección de formas" en la página 121.
Outline	Propiedad **Color**	Especifica el color de contorno de una forma. Para más información vea "Aplicación de contornos" en la página 132.
Outline	Propiedad **Style**	Especifica la configuración de guiones (es decir, las propiedades de estilo) del contorno de una forma. Para más información vea "Aplicación de contornos" en la página 132.

Clase	Miembro	Descripción
Outline	Propiedad **Type**	Especifica si un contorno es aplicado a una forma. Para más información vea "Aplicación de contornos" en la página 132.
Outline	Propiedad **Width**	Especifica, en unidades del documento, la anchura del contorno de una forma. Para más información vea "Aplicación de contornos" en la página 132.
Segment	Método **AddNodeAt**	Añade un nodo a un segmento de línea o de curva. Para más información vea "Creación de líneas y curvas" en la página 114.
Shape	Método **CreateBlend**	Aplica un efecto de mezcla a una forma. Para más información vea "Aplicación de mezclas" en la página 135.
Shape	Método **CreateContour**	Aplica un efecto de silueta a una forma. Para más información vea "Aplicación de siluetas" en la página 135.
Shape	Método **CreateCustomDistortion**	Aplica un efecto de distorsión personalizada a una forma. Para más información vea "Aplicación de distorsiones" en la página 136.
Shape	Método **CreateCustomEffect**	Aplica un efecto personalizado a una forma. Para más información vea "Aplicación de efectos personalizados" en la página 136.
Shape	Método **CreateDropShadow**	Aplica un efecto de sombra a una forma. Para más información vea "Aplicación de sombras" en la página 136.
Shape	Método **CreateEnvelope**, Método **CreateEnvelopeFromCurve** o Método **CreateEnvelopeFromShape**	Aplica un efecto de envoltura a una forma. Para más información vea "Aplicación de envolturas" en la página 136.
Shape	Método **CreateExtrude**	Aplica un efecto de extrusión a una forma. Para más información vea "Aplicación de extrusiones" en la página 136.
Shape	Método **CreateLens**	Aplica un efecto de transparencia a una forma. Para más información vea "Aplicación de transparencias" en la página 137.

Clase	Miembro	Descripción
Shape	Método **CreatePerspective**	Aplica un efecto de perspectiva a una forma. Para más información vea "Aplicación de perspectivas" en la página 137.
Shape	Método **CreatePushPullDistortion**	Aplica un efecto de distorsión Push-and-pull a una forma. Para más información vea "Aplicación de distorsiones" en la página 136.
Shape	Método **CreateSelection**	Crea una selección de una sola forma. Para más información vea "Selección de formas" en la página 121.
Shape	Método **CreateTwisterDistortion**	Aplica un efecto de distorsión de Pincel Deformador a una forma. Para más información vea "Aplicación de distorsiones" en la página 136.
Shape	Método **CreateZipperDistortion**	Aplica un efecto de distorsión de Pincel Agreste a una forma. Para más información vea "Aplicación de distorsiones" en la página 136.
Shape	Método **Duplicate**	Duplica una forma. Para más información vea "Duplicación de formas" en la página 125.
Shape	Método **Evaluate**	Devuelve el resultado de una expresión dada que evalúa las propiedades de la forma actual. Para más información vea "Búsqueda de formas" en la página 137.
Shape	Método **GetBoundingBox**	Devuelve el tamaño de una forma basándose en el tamaño de su caja delimitadora. Para más información vea "Dimensionamiento de formas" en la página 125.
Shape	Método **GetPosition**	Devuelve la posición horizontal y vertical de una forma. Para más información vea "Posicionamiento de formas" en la página 129.
Shape	Método **GetSize**	Devuelve el tamaño de una forma. Para más información vea "Dimensionamiento de formas" en la página 125.

Clase	Miembro	Descripción
Shape	Método **PlaceTextInside**	Coloca un texto seleccionado dentro de una forma especificada. Para más información vea "Creación de objetos de texto" en la página 116.
Shape	Propiedad **PositionX**	Devuelve, o ajusta, la posición horizontal de una forma. Para más información vea "Posicionamiento de formas" en la página 129.
Shape	Propiedad **PositionY**	Devuelve, o ajusta, la posición Vertical de una forma. Para más información vea "Posicionamiento de formas" en la página 129.
Shape	Método **Rotate** o Método **RotateEx**	Rota una forma. Para más información vea "Rotación de formas" en la página 128.
Shape	Método **Selected**	Especifica si una forma está seleccionada. Para más información vea "Selección de formas" en la página 121.
Shape	Método **SetBoundingBox**	Ajusta el tamaño de una forma basándose en el tamaño de su caja delimitadora. Para más información vea "Dimensionamiento de formas" en la página 125.
Shape	Método **SetPosition** o Método **SetPositionEx**	Ajusta la posición de una forma. Para más información vea "Posicionamiento de formas" en la página 129.
Shape	Método **SetSize** o Método **SetSizeEx**	Ajusta el tamaño de una forma. Para más información vea "Dimensionamiento de formas" en la página 125.
Shape	Propiedad **SizeHeight**	Devuelve, o ajusta, la altura de una forma. Para más información vea "Dimensionamiento de formas" en la página 125.
Shape	Propiedad **SizeWidth**	Devuelve, o ajusta, la anchura de una forma. Para más información vea "Dimensionamiento de formas" en la página 125.
Shape	Método **Skew** o Método **SkewEx**	Sesga una forma. Para más información vea "Sesgado de formas" en la página 128.

Clase	Miembro	Descripción
Shape	Método **Stretch** o Método **StretchEx**	Estira (o escala) una forma. Para más información vea "Estiramiento de formas" en la página 127.
Shape	Propiedad **Type**	Devuelve el tipo de una forma. Para más información vea "Determinación del tipo de forma" en la página 121.
ShapeRange	Método **CreateSelection**	Crea una selección de un rango de formas. Para más información vea "Selección de formas" en la página 121.
ShapeRange	Método **Duplicate**	Duplica un rango de formas. Para más información vea "Duplicación de formas" en la página 125.
Shapes	Método **All**	Devuelve todas las formas de la colección especificada de formas. Para más información vea "Selección de formas" en la página 121.
Shapes	Método **FindShape**	Devuelve una sola forma que tiene las propiedades especificadas. Para más información vea "Búsqueda de formas" en la página 137.
Shapes	Método **FindShapes**	Devuelve, como un rango de formas, todas las formas que tienen las propiedades especificadas. Para más información vea "Búsqueda de formas" en la página 137.
SubPath	Método **AppendCurveSegment** o Método **AppendCurveSegment2**	Añade un segmento tipo curva a un subtrayecto. Para más información vea "Creación de líneas y curvas" en la página 114.
SubPath	Método **AppendLineSegment**	Añade un segmento tipo línea a un subtrayecto. Para más información vea "Creación de líneas y curvas" en la página 114.
Symbol	Propiedad **Definition**	Devuelve la definición de un símbolo. Para más información vea "Creación de símbolos" en la página 118.

Clase	Miembro	Descripción
SymbolDefinition	Propiedad **NestedSymbols**	Contiene la colección de todos los símbolos anidados para una definición de símbolo. Para más información vea "Creación de símbolos" en la página 118.
SymbolLibrary	Propiedad **Symbols**	Contiene la colección de todas las definiciones de símbolos para una biblioteca de símbolos. Para más información vea "Creación de símbolos" en la página 118.
Text	Método **FitTextToPath**	Agrega el texto artístico especificado al contorno de una forma. Para más información vea "Creación de objetos de texto" en la página 116.
Text	Propiedad **Frames**	Representa una serie de cajas de texto (u objetos **TextFrame**), cada una de las cuales tiene su propio rango de texto (u objeto **TextRange**). Para más información vea "Creación de objetos de texto" en la página 116.
Text	Propiedad **Story**	Representa un rango de texto (u objeto **TextRange**) que incluye todo el texto en una serie de cajas de texto (u objetos **TextFrame**). Para más información vea "Creación de objetos de texto" en la página 116.

Para información detallada sobre cualquier propiedad, método o evento, vea la sección "Referencia del Modelo de Objeto" en el archivo de Ayuda de Macros en CorelDRAW.

Para más información sobre actividades relacionadas con formas, vea los siguientes temas:

- Creación de formas.
- Determinación del tipo de forma.
- Selección de formas.
- Duplicación de formas.
- Transformación de formas.
- Coloración de formas.
- Aplicación de efectos a formas.
- Búsqueda de formas.
- Eliminación de formas.

Creación de formas

Cada documento CorelDRAW está compuesto de formas, u objetos **Shape**, las cuales son creadas utilizando las herramientas de dibujo. Las formas en una página del documento están almacenadas en capas, de modo que los distintos métodos de creación de formas pertenecen a la clase **Layer**.

Esta sección cubre los siguientes temas:

- Creación de rectángulos.
- Creación de elipses.
- Creación de líneas y curvas.
- Creación de objetos de texto.
- Creación de símbolos.

CorelDRAW soporta formas adicionales que no se discuten en esta sección, tales como polígonos (u objetos **Polygon**) y formas personalizadas (u objetos **CustomShape**).

Las formas personalizadas que están soportadas en CorelDRAW incluyen tablas (u objetos **TableShape**).

Las formas se miden en unidades del documento. Se puede especificar la unidad de medida de un documento utilizando la propiedad **Document.Unit**, como se explica en "Configuración de las propiedades del documento" en la página 82.

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sean activados al crear una forma:

- **AddinHook.ShapeCreated**
- **Document.ShapeCreate**

Creación de rectángulos

Se pueden añadir rectángulos (u objetos **Rectangle**) a sus documentos. CorelDRAW ofrece tres métodos para crear rectángulos — **Layer.CreateRectangle**, **Layer.CreateRectangle2** y **Layer.CreateRectangleRect** — cada uno de los cuales devuelve una referencia al nuevo objeto **Shape**. Estos métodos difieren únicamente en los parámetros que toman, de modo que se puede elegir el método que mejor se ajuste a la solución de macro.

También se pueden utilizar estos métodos de creación de rectángulos para crear cuadrados.

El método **CreateRectangle** crea un rectángulo utilizando cuatro parámetros que especifican lo siguiente:

- La distancia entre los lados izquierdo, superior, derecho e inferior del rectángulo (en ese orden).
- Los bordes correspondientes del marco de la página.

El siguiente ejemplo VBA utiliza el método **CreateRectangle** para crear un rectángulo de 2" × 1" que está posicionado 6" arriba del fondo de la página y 3" a la derecha del lado izquierdo de la página:

```
Dim sh As Shape
ActiveDocument.Unit = cdrInch
Set sh = ActiveLayer.CreateRectangle(3, 7, 6, 5)
```

El método **CreateRectangle2** crea un rectángulo basado en las coordenadas de su esquina inferior izquierda, su anchura y su altura.

El siguiente ejemplo VBA utiliza el método **CreateRectangle2** para crear el mismo rectángulo que el ejemplo previo:

```
Dim sh As Shape
ActiveDocument.Unit = cdrInch
Set sh = ActiveLayer.CreateRectangle2(3, 6, 2, 1)
```

Finalmente, el método **CreateRectangleRect** crea un rectángulo basado en su caja delimitadora (u objeto **Rect**).

Estos tres métodos de creación de rectángulos proporcionan parámetros opcionales para especificar la redondez de esquinas.

El método **CreateRectangle** especifica la redondez de esquinas utilizando parámetros para las esquinas superior izquierda, superior derecha, inferior izquierda e inferior derecha (en ese orden). Estos parámetros toman valores enteros (los cuales por defecto van desde 0 hasta 100) que definen el radio de las cuatro esquinas como un porcentaje del número total de la mitad de la longitud del lado más corto.

El siguiente ejemplo VBA recrea el rectángulo de 2" × 1" de los ejemplos previos. Sin embargo, esta vez, el radio de las cuatro esquinas está ajustado a 100%, 75%, 50% y 0% de la mitad de la longitud del lado más corto (es decir, a 0.5", 0.375", 0.25" y 0", y un vértice):

```
Dim sh As Shape
ActiveDocument.Unit = cdrInch
Set sh = ActiveLayer.CreateRectangle(3, 7, 6, 5, 100, 75, 50, 0)
```

El método **CreateRectangle2** y el método **CreateRectangleRect** definen el radio de las esquinas en el mismo orden que el método **CreateRectangle** (es decir, superior izquierda, superior derecha, inferior izquierda e inferior derecha). Sin embargo, **CreateRectangle2** y **CreateRectangleRect** toman valores dobles (de punto flotante) que miden el radio de las esquinas en unidades del documento.

Cuando se utilice **CreateRectangle2** o **CreateRectangl**e, se debe limitar el tamaño del radio de la esquina a menos de la mitad de la longitud del lado más corto del rectángulo.

El siguiente ejemplo VBA utiliza el método **CreateRectangle2** para crear el mismo rectángulo de esquinas redondeadas del ejemplo anterior:

```
Dim sh As Shape
ActiveDocument.Unit = cdrInch
ActiveDocument.ReferencePoint = cdrBottomLeft
Set sh = ActiveLayer.CreateRectangle2(3, 6, 2, 1, 0.5, 0.375, 0.25, _
0)
```

Creación de elipses

Se pueden añadir elipses (u objetos **Ellipse**) a sus documentos. CorelDRAW ofrece tres métodos para la creación de elipses — **Layer.CreateEllipse**, **Layer.CreateEllipse2** y **Layer.CreateEllipseRect** — cada uno de los cuales devuelve una referencia al nuevo objeto **Shape**. Estos métodos difieren únicamente en los parámetros que toman, de modo que se puede elegir el método que mejor se ajuste a la solución de macro.

También se pueden utilizar estos métodos de creación de elipses para crear círculos, arcos y rebanadas de pastel.

El método **CreateEllipse** crea una elipse utilizando cuatro parámetros que especifican lo siguiente:

- La distancia entre los lados izquierdo, superior, derecho e inferior de la elipse (en ese orden).
- Los bordes correspondientes del marco de la página.

El siguiente ejemplo VBA crea un círculo de 50 milímetros:

```
Dim sh As Shape
ActiveDocument.Unit = cdrMillimeter
Set sh = ActiveLayer.CreateEllipse(75, 150, 125, 100)
```

El método **CreateEllipse2** crea una elipse basada en su punto central, su radio horizontal y su radio vertical. (Si sólo se especifica un radio, se crea un círculo).

El siguiente ejemplo VBA utiliza el método **CreateEllipse2** para crear una elipse:

```
Dim sh As Shape
ActiveDocument.Unit = cdrMillimeter
Set sh = ActiveLayer.CreateEllipse2(100, 125, 50, 25)
```

El siguiente ejemplo VBA utiliza el método **CreateEllipse2** para crear el mismo círculo de 50 milímetros que el ejemplo **CreateEllipse**:

```
Dim sh As Shape
ActiveDocument.Unit = cdrMillimeter
Set sh = ActiveLayer.CreateEllipse2(100, 125, 25)
```

Finalmente, el método **CreateEllipseRect** crea una elipse basada en su caja delimitadora (u objeto **Rect**).

Estos tres métodos para la creación de elipses proporcionan parámetros opcionales para crear arcos o rebanadas de pastel. Los parámetros **StartAngle** y **EndAngle —** los cuales son valores dobles que se miden con cero estando horizontalmente a la derecha de la página y con valores positivos siendo graduados desde cero y moviéndose en contra del sentido del reloj — son utilizados para definir el ángulo inicial y el ángulo final de la forma (respectivamente). Por otro lado, el parámetro **Pie** — el cual es un valor Booleano — define si la forma es un arco (**False**) o una rebanada de pastel (**True**).

El siguiente código VBA utiliza el método **CreateEllipse** para crear una figura en forma de "C":

```
Dim sh As Shape
ActiveDocument.Unit = cdrMillimeter
Set sh = ActiveLayer.CreateEllipse(75, 150, 125, 100, 60, 290, False)
```

Creación de líneas y curvas

Se pueden añadir líneas y curvas (u objetos **Curve**) a sus documentos. Para crear una línea o una curva, primero se debe crear un objeto **Curve** "en memoria" utilizando el método **Application.CreateCurve**.

Cada objeto **Curve** tiene por lo menos un subtrayecto (u objeto **SubPath**). Se puede añadir un subtrayecto a una línea o a una curva utilizando el método **Curve.CreateSubPath**.

Cada objeto **SubPath** tiene por lo menos un segmento (u objeto **Segment**), el cual puede ser tipo línea o tipo curva. Se puede añadir un segmento tipo línea al final del subtrayecto utilizando el método **SubPath.AppendLineSegment**; se puede añadir un segmento tipo curva utilizando el método **SubPath.AppendCurveSegment** o el método **SubPath.AppendCurveSegment2**.

El método **SubPath.AppendLineSegment** requiere un grupo de coordenadas Cartesianas, las cuales definen el final de un nuevo segmento.

El método **SubPath.AppendCurveSegment** requiere un grupo de coordenadas Cartesianas, las cuales definen el final del nuevo segmento. Opcionalmente, se pueden especificar dos grupos de coordenadas polares si se desea definir las longitudes y ángulos del inicio y final de los manejadores de control para el segmento.

El método **SubPath.AppendCurveSegment2** requiere tres grupos de coordenadas Cartesianas: una para definir el final del nuevo segmento, y dos para definir las posiciones de manejadores de control inicial y final del segmento.

Se puede añadir un segmento al comienzo del subtrayecto configurando a **True** el parámetro **AppendAtBeginning** para el método de creación de segmento.

Finalmente, cada objeto **Segment** tiene por lo menos un nodo (u objeto **Node**). Se puede añadir un nodo a un segmento utilizando el método **Segment.AddNodeAt**.

Se puede cerrar un objeto **Curve** configurando a **True** su propiedad **Closed**.

Después de crear una curva "en memoria", usted puede aplicarla a una capa utilizando el método **Layer.CreateCurve**. Una referencia al nuevo objeto **Shape** es devuelta.

El siguiente código VBA crea una curva cerrada en forma de "D":

```
Dim sh As Shape, spath As SubPath, crv As Curve
ActiveDocument.Unit = cdrCentimeter
Set crv = Application.CreateCurve(ActiveDocument)
'Crea un objeto Curve
Set spath = crv.CreateSubPath(6, 6) ' Crea un subtrayecto
spath.AppendLineSegment 6, 3 ' Añade el segmento vertical corto
spath.AppendCurveSegment 3, 0, 2, 270, 2, 0 ' Curva inferior
spath.AppendLineSegment 0, 0 ' Esquina recta inferior
spath.AppendLineSegment 0, 9 ' Esquina recta de la izquierda
spath.AppendLineSegment 3, 9 ' Esquina recta superior
spath.AppendCurveSegment 6, 6, 2, 0, 2, 90 ' Curva superior
spath.Closed = True ' Cierra la curva
Set sh = ActiveLayer.CreateCurve(crv) ' Crea la forma curva
```

La clase **Layer** proporciona tres métodos adicionales que actúan como atajos para crear una línea básica o una curva básica que tenga un solo segmento o un solo subtrayecto.

- **Layer.CreateLineSegment** — crea una línea básica basada en el punto inicial y el punto final dados.
- **Layer.CreateCurveSegment** — crea una línea básica basada en el punto inicial y el punto final dados y, opcionalmente, en las longitudes y ángulos de los manejadores de control inicial y final de la curva.
- **Layer.CreateCurveSegment2** — crea una curva básica basada en el punto inicial y el punto final dados y en las posiciones dadas inicial y final de los manejadores de control de la curva.

Estos tres métodos devuelven una referencia al nuevo objeto **Shape**.

Creación de objetos de texto

Se puede añadir texto (u objetos **Text**) a los documentos. CorelDRAW soporta dos tipos de texto: texto artístico y texto de párrafo. Un objeto de texto artístico es una pequeña línea de texto en la cual se pueden aplicar efectos gráficos. En contraste, un objeto de texto de párrafo es un gran bloque de texto — almacenado en un contenedor rectangular llamado "marco" — al cual se le puede aplicar un formateo más complejo.

Para crear un objeto de texto artístico, se puede utilizar uno de los siguientes métodos:

- **Layer.CreateArtisticText** — crea texto artístico básico.
- **Layer.CreateArtisticTextWide** — crea texto artístico que está en formato Unicode.

Ambos métodos requieren especificar la posición y el contenido del objeto de texto artístico. Opcionalmente, ambos métodos le permiten ajustar ciertos atributos de texto tales como estilo de fuente, tamaño de fuente, formato y alineación. Además, ambos métodos devuelven una referencia al nuevo objeto **Shape**.

El siguiente código VBA utiliza el método **CreateArtisticText** para crear un objeto básico de texto artístico que coloca las palabras "Hola Mundo" en la posición especificada:

```
Dim sh As Shape
ActiveDocument.Unit = cdrInch
Set sh = ActiveLayer.CreateArtisticText(1, 4, "Hola Mundo")
```

Se puede ajustar el texto artístico a un trayecto utilizando el método **Text.FitTextToPath**, el cual simplemente adhiere el texto al contorno de la forma de manera que el texto fluye a lo largo del trayecto de esa forma.

El siguiente código VBA crea un nuevo objeto de texto artístico y lo adhiere a la forma seleccionada:

```
Dim sh As Shape, sPath As Shape
ActiveDocument.Unit = cdrInch
Set sPath = ActiveShape
Set sh = ActiveLayer.CreateArtisticText(1, 4, "Hola Mundo")
sh.Text.FitToPath sPath
```

Para crear un objeto de texto de párrafo, se puede utilizar uno de los siguientes métodos:

- **Layer.CreateParagraphText** — crea texto de párrafo básico.
- **Layer.CreateParagraphTextWide** — crea texto de párrafo que está en formato Unicode.

Ambos métodos requieren especificar el tamaño del marco de texto de párrafo ajustando la posición de sus lados izquierdo, superior, derecho e inferior (en ese orden) con respecto al borde de la página. Opcionalmente, ambos métodos le permiten especificar el texto deseado y ajustar ciertos atributos de texto tales como estilo de fuente, tamaño de fuente, formato y alineación. Además, ambos métodos devuelven una referencia al nuevo objeto **Shape**.

El siguiente código VBA utiliza el método **CreateParagraphText** para crear un objeto básico de texto de párrafo que centra las palabras "Hola Mundo" en un marco de tamaño especificado:

```
Dim sh As Shape
ActiveDocument.Unit = cdrInch
Set sh = ActiveLayer.CreateParagraphText(1, 4, 5, 2, "Hola Mundo", _
  Alignment := cdrCenterAlignment)
```

Se puede dar formato a texto de párrafo existente utilizando rangos de texto (u objetos **TextRange**). CorelDRAW manipula los rangos de texto en dos formas, en las cuales se involucran marcos (u objetos **TextFrame**):

- Método "marcos" — La propiedad **TextFrames** representa una serie de marcos de texto, cada uno de los cuales tiene su propio rango de texto.
- Método "historia" — La propiedad **Text.Story** representa un rango de texto que incluye todo el texto en una serie de marcos de texto.

Un rango de texto puede ser tratado como un único bloque de texto, de tal manera que cualquier cambio a las propiedades de texto (tal como estilo de fuente y tamaño de fuente) son aplicadas a todo el texto en ese rango de texto. Alternativamente, un rango de texto puede ser descompuesto en los siguientes rangos de texto más pequeños:

- Columnas (u objetos **TextColumns**).
- Párrafos (u objetos **TextParagraphs**).

  Líneas (u objetos **TextLines**).
- Palabras (u objetos **TextWords**).
- Caracteres (u objetos **TextCharacters**).

El modelo de objeto CorelDRAW soporta todas las opciones de formato de párrafo y formato de caracteres que se proporcionan con la aplicación.

El siguiente código VBA da formato al rango de texto especificado utilizando la propiedad **TextStory**. El primer párrafo de la historia es cambiado a un estilo "título" y el segundo y tercer párrafos a un estilo "cuerpo del texto":

```
Dim txt As TextRange
' Dar formato al primer párrafo
  Set txt = ActiveShape.Text.Story.Paragraphs(1)
  txt.ChangeCase cdrTextUpperCase
  txt.Font = "Verdana"
  txt.Size = 18
  txt.Bold = True
' Dar formato al segundo y tercer párrafos
  Set txt = ActiveShape.Text.Story.Paragraphs(2, 2)
  txt.Font = "Times New Roman"
  txt.Size = 12
  txt.Style = cdrNormalFontStyle
```

Se puede colocar el texto seleccionado dentro de formas cerradas utilizando el método **Shape.PlaceTextInside**.

El siguiente código VBA crea una elipse y coloca en su interior el texto seleccionado:

```
Dim txt As Shape, sh As Shape
ActiveDocument.Unit = cdrInch
Set txt = ActiveShape
Set sh = ActiveLayer.CreateEllipse(0, 2, 5, 0)
sh.PlaceTextInside txt
```

Creación de símbolos

Un símbolo (u objeto **Symbol**) es un elemento gráfico reutilizable que está definido en una biblioteca de símbolos. El uso de símbolos en los documentos proporciona los siguientes beneficios:

- Tamaño de archivo más pequeño — Cada símbolo se define sólo una vez, independientemente de cuántas instancias actuales de ese símbolo aparezcan en el documento.
- Incremento de la productividad — Cualquier cambio hecho a la definición de un símbolo se propaga automáticamente a todas las instancias de ese símbolo en el documento.
- Mejoramiento del flujo de trabajo — Las bibliotecas de símbolos son una manera conveniente de almacenar y reutilizar elementos gráficos comunes.

Las bibliotecas de símbolos vienen en dos variedades: externas e internas.

Las bibliotecas externas de símbolos utilizan la extensión de nombre de archivo CSL y contienen definiciones de símbolos que deben ser añadidas manualmente al espacio de trabajo en el nivel de la aplicación. No se puede modificar un símbolo que está definido en una biblioteca externa a menos que se abra el archivo de la biblioteca externa asociada (CSL); no es suficiente importar el archivo como una biblioteca para poder permitirle modificar su contenido

Las bibliotecas externas de símbolos deben ser publicadas en una ubicación en la que todos los usuarios puedan acceder. Una unidad vinculada en común es una buena solución, pero una red interna empresarial es aún mejor. Sin embargo, si la seguridad de los símbolos no es importante, la mejor solución es un sitio de Internet de la empresa.

Las bibliotecas externas de símbolos existen en el nivel documento. La definición de un nuevo símbolo en un documento — o la adición de una instancia de un símbolo de biblioteca externa al documento — automáticamente agrega ese símbolo a la biblioteca interna de ese documento. Por esta razón, cada documento tiene su propia biblioteca interna de símbolos única.

Al insertar una instancia de un símbolo desde una biblioteca externa de símbolos se crea un vínculo a la definición de ese símbolo en la biblioteca externa de símbolos. Si, en cualquier punto, el documento no puede tener acceso a la biblioteca externa de símbolos, la definición del símbolo en la biblioteca interna de símbolos para ese documento es utilizada en su lugar.

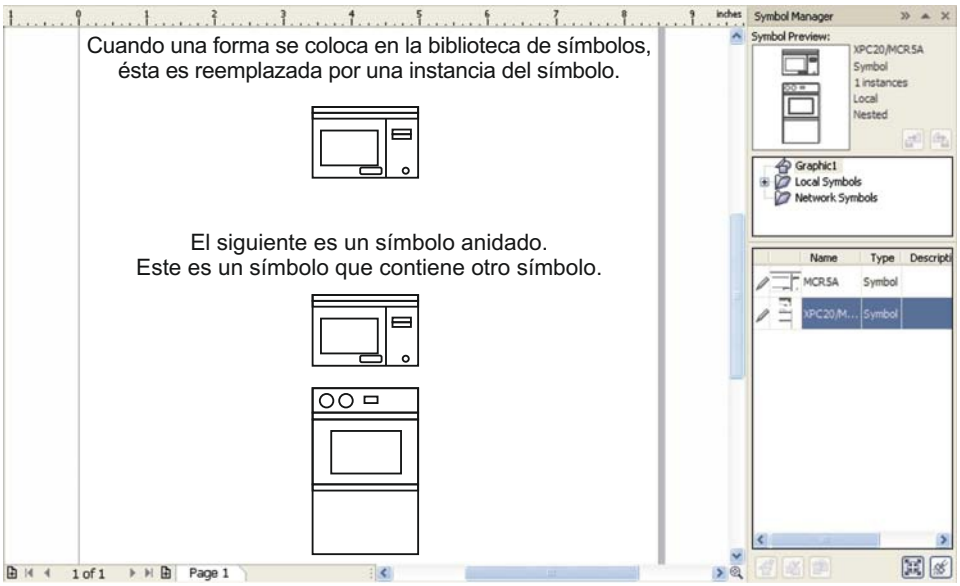
La propiedad **Application.SymbolLibraries** contiene la colección de todas las bibliotecas externas de símbolos (u objetos **SymbolLibrary)** que están disponibles para la aplicación; la propiedad **DocumentSymbolLibrary** devuelve sólo la biblioteca interna de símbolos para ese documento. La propiedad **SymbolLibrary.Symbols** contiene la colección de todas las definiciones de símbolo (u objetos **SymbolDefinition**) en esa biblioteca de símbolos. Un objeto **SymbolDefinition** es también devuelto por la propiedad **Symbol.Definition**; por ello, se puede modificar la definición de un símbolo utilizando las variadas propiedades y métodos de la clase **SymbolDefinition**.

Para eliminar una definición de símbolo de la biblioteca interna de símbolos de un documento, se deben eliminar todas las instancias del símbolo desde el documento y luego ejecutar el método **SymbolLibrary.PurgeUnusedSymbols**. El simple hecho de eliminar todas las instancias de un símbolo de un documento no implica que se elimine automáticamente su definición de símbolo de la biblioteca interna de símbolos de ese documento.

El siguiente código VBA muestra los fundamentos del uso de símbolos:

```
Sub AddRemoveSymbols()
    Dim objSymLibSwitchA As SymbolLibrary
    Dim shpSymBreaker2 As Shape, shpSymBreaker2A As Shape
    ActiveDocument.Unit = cdrMillimeter
    'Agrega las bibliotecas externas de símbolos switchesA al espacio
    'de trabajo global.
      Set objSymLibSwitchA = SymbolLibraries.Add _
        ("C:\libs\switches\switchesA.csl")
    'Agrega el símbolo breaker2 a la capa activa.
    'NOTA: Esto agrega automáticamente la definición del símbolo a la
    'biblioteca interna de símbolos del documento.
      Set shpSymBreaker2 = ActiveLayer.CreateSymbol(15, 20, _
        "breaker2", SymbolLibraries("switchesA"))
    'Agrega otra instancia del símbolo breaker2, esta vez desde la
    'biblioteca interna de símbolos. NOTA: Nosotros no especificamos una biblioteca,
    'así que la biblioteca para el documento local es utilizada por defecto.
      Set shpSymBreaker2A = ActiveLayer.CreateSymbol(30, 20, _
        "breaker2")
    'Elimina la biblioteca switchesA del espacio de trabajo global.
      SymbolLibraries("switchesA").Delete
    'Elimina los dos símbolos breaker2.
      shpSymBreaker2.Delete
      shpSymBreaker2A.Delete
    'En este punto, la biblioteca interna de símbolos del documento
    'aún tiene almacenada la definición de breaker2. Para eliminar esta
    'definición, debemos purgar los símbolos sin utilizar de la biblioteca.
    'La definición está sin utilizar porque no hay instancias que
    'hagan referencia a ella.
      ActiveDocument.SymbolLibrary.PurgeUnusedSymbols
End Sub
```

Un símbolo puede contener (o "anidar") otros símbolos. Un símbolo en la cima del nivel puede contener símbolos, y cada uno de esos símbolos pueden contener un símbolo, y así sucesivamente. En el modelo de objeto CorelDRAW, la propiedad **SymbolDefinition.NestedSymbols** devuelve (como un objeto **SymbolDefinitions**) la colección de símbolos anidados de una definición de símbolo. Mientras que no haya restricción de cuántos niveles de anidación puedan ser creados, el símbolo no puede ser suministrado sin acceder a su definición de símbolo (ya sea externo o interno). Por otro lado, incluso si el primer y segundo niveles de anidación de un símbolo están suministrados correctamente, un símbolo en el tercer nivel de anidación puede no ser suministrado correctamente sin acceder a su definición de símbolo requerida.



Símbolos y símbolos anidados

El siguiente código VBA muestra los fundamentos del uso de símbolos anidados:

```
Sub MakeNestedSymbol()
    Dim shp1 As Shape, shp2 As Shape, shp3 As Shape, shpSym As Shape
    Dim shpRng As New ShapeRange
    'Crea un par de rectángulos y un círculo.
      Set shp1 = ActividadiveLayer.CreateRectangle2(0, 0, 10, 20)
      Set shp2 = ActividadiveLayer.CreateRectangle2(50, 50, 20 ,10)
      Set shp3 = ActividadiveLayer.CreateEllipse(10, 10, 20)
    'Elabora un símbolo del círculo. NOTA: Este círculo se
    'añade automáticamente a la biblioteca interna de símbolos
    'del documento.
      Set shpSym = shp3.ConvertToSymbol("circle")
    'Agrega los rectángulos y el símbolo del círculo a un rango de forma.
      shpRng.Add shp1
      shpRng.Add shp2
      shpRng.Add shpSym
    'Convierte el rango de forma a un símbolo. NOTA: Este símbolo se
    'añade a la biblioteca interna de símbolos del documento. Este es
    'también un símbolo anidado porque contiene el símbolo "circle".
      shpRng.ConvertToSymbol "shapes”
End Sub
```

Determinación del tipo de forma

Cada objeto **Shape** tiene una propiedad **Type** de sólo lectura, la cual devuelve el tipo de forma (por ejemplo, rectángulo, elipse, curva, texto o grupo). Las propiedades y métodos que están disponibles para una forma varían con el tipo de forma; por lo tanto, es una buena idea determinar el tipo de forma antes de aplicarle cualquier propiedad o método a esa forma.

La siguiente muestra de código VBA determina si una forma es texto. Si la forma es texto, el código determina si es texto artístico o texto de párrafo. Si la forma es texto artístico, éste es rotado 10 grados.

```
Dim sh As Shape
Set sh = ActiveShape
If sh.Type = cdrTextShape Then
  If sh.Text.IsArtisticText = True Then
    sh.Rotate 10
  End If
End If
```

Selección de formas

En CorelDRAW, cada forma es un miembro de la colección **Layer.Shapes** de la capa en la que aparece. Las formas en una colección **Layer.Shapes** aparecen en el mismo orden en el que aparecen sobre la capa — la primer forma es la que está en la cima de la "pila", y la última forma es la que está en el fondo. Si se añaden, reordenan o eliminan formas, la colección **Layer.Shapes** afectada se actualiza inmediatamente para reflejar el nuevo orden de formas de esa capa.

Además, cada página del documento tiene una colección **Shapes**, la cual contiene todas las colecciones **Layer.Shapes** de esa página. La primer forma en una colección **Page.Shapes** es la que está hasta la cima de esa página, y la última forma en la que está hasta el fondo.

Si se desea acceder a formas individuales, pueden seleccionarse. Cuando se seleccionan formas, se crea una "selección" que contiene sólo esas formas.

La propiedad **Shape.Selected** toma un valor Booleano que indica si está seleccionada una forma: **True** si la forma está seleccionada, **False** si no lo está. Se puede seleccionar una forma cambiando el valor de su propiedad **Selected** a **True**; esta técnica agrega la forma a la selección actual — es decir, en lugar de que se cree una nueva selección que contenga sólo esa forma.

Si desea crear una nueva selección de una forma — es decir, seleccionando una forma especificada y deseleccionando cualquier otra forma(s) seleccionada(s) — se puede utilizar el método **Shape.CreateSelection**, como el en siguiente ejemplo VBA:

```
Dim sh As Shape
Set sh = ActivePage.Shapes(1)
If sh.Selected = False Then sh.CreateSelection
```

Se pueden seleccionar múltiples formas utilizando el método **ShapeRange.CreateSelection**. El siguiente código VBA utiliza este método — en combinación con el método **Shapes.All** — para seleccionar todas las formas de la página activa (excepto aquellas que están en capas bloqueadas u ocultas):

```
ActivePage.Shapes.All.CreateSelection
```

Se puede acceder a una selección mediante una de las siguientes técnicas:

- Utilizando el método **Document.Selection** para devolver un objeto **Shape** especial que contenga la selección actual. Este objeto **Shape** se renueva automáticamente cuando se actualiza la selección.
- Utilizando la propiedad **Document.SelectionRange** para devolver un objeto **ShapeRange** que contenga una copia de la selección. Este objeto **ShapeRange** representa una "foto instantánea" de la selección (en el momento que el objeto **ShapeRange** sea creado), de modo que no se renueva automáticamente cuando se actualiza la selección.

Para resumir, se puede acceder directamente a una selección, o se puede acceder a una copia de esa selección; alternativamente, se puede acceder a un subconjunto de formas en una selección. También se pueden reordenar las formas en una selección. Cuando ya no se requiera la selección, se pueden deseleccionar una o todas las formas.

- Accesando directamente a las selecciones.
- Accesando a copias de selecciones.
- Accesando a las formas en una selección.
- Reordenando las formas en una selección.
- Deseleccionando formas.

Accesando directamente a selecciones

Como se discutió previamente, se puede utilizar el método **Document.Selection** para acceder directamente al contenido de una selección. Un objeto **Shape** es devuelto, y este objeto **Shape** se actualiza para reflejar cualquier cambio hecho en la selección.

El siguiente código VBA devuelve la selección del documento activo:

```
Dim sel As Shape
Set sel = ActiveDocument.Selection
```

El atajo para `ActiveDocument.Selection` es `ActiveSelection`, el cual devuelve un objeto **Shape** de subtipo **cdrSelectionShape**. El subtipo **Shape** tiene una colección de miembros llamada **Shapes**, la cual representa una colección de todos los objetos seleccionados en el documento. Se puede acceder a las formas en la colección `ActiveSelection.Shapes` mediante el siguiente ejemplo VBA:

```
Dim sh As Shape, shs As Shapes
Set shs = ActiveSelection.Shapes
For Each sh In shs
  sh.Rotate 15 'Rotate each shape by 15 degrees counterclockwise
Next sh
```

Después de utilizar el comando `ActiveSelection` para seleccionar formas, no se puede utilizar el comando subsecuentemente para tener acceso a esas formas. En lugar de eso, se debe crear una copia de la selección utilizando uno de los siguientes métodos:

- Volver a crear la selección como un arreglo de objetos **Shape**.
- Volver a crear la selección como una colección **Shapes**.
- Crear una "foto instantánea" de la selección como un objeto **ShapeRange** (como se describe en "Accesando a copias de selecciones" en la página 123.

Accesando a copias de selecciones

Como se discutió previamente, se puede utilizar la propiedad **Document.SelectionRange** para hacer una copia de las formas en una selección. Sin embargo, el objeto devuelto **ShapeRange** no es renovado cuando la selección se actualiza porque éste representa una "foto instantánea" de la selección en el momento en que el objeto **ShapeRange** fue creado.

El siguiente código VBA devuelve una copia de la selección para el documento activo:

```
Dim selRange As ShapeRange
Set selRange = ActiveDocument.SelectionRange
```

El atajo para el comando `ActiveDocument.SelectionRange` es `ActiveSelectionRange`, el cual devuelve un objeto **ShapeRange**. Este objeto contiene una colección de referencias a las formas que estaban seleccionadas en el momento en que la propiedad fue invocada. Se puede acceder a las formas en la colección `ActiveSelectionRange` mediante el siguiente ejemplo VBA:

```
Dim sh As Shape, shRange As ShapeRange
Set shRange = ActiveSelectionRange
For Each sh In shRange
  sh.Skew 15 ' Skew each shape thru 15° counterclockwise
Next sh
```

Después de que se utilice el comando `ActiveSelectionRange` para crear una copia de la selección del documento actual, se puede acceder subsecuentemente al objeto **ShapeRange** devuelto para acceder a cualquiera de sus formas. Incluso se pueden agregar formas a o eliminar formas del objeto **ShapeRange** devuelto. Luego se puede utilizar el método **ShapeRange.CreateSelection** si se desea reemplazar la selección actual con el objeto **ShapeRange** modificado.

El siguiente código VBA crea un objeto **ShapeRange** de la selección del documento actual, elimina la primera y segunda formas de ese rango de formas y luego reemplaza la selección original con este objeto **ShapeRange** modificado:

```
Dim shRange As ShapeRange
Set shRange = ActiveSelectionRange
shRange.Remove 1
shRange.Remove 2
shRange.CreateSelection
```

Si se quiere agregar un objeto **ShapeRange** especificado a la selección actual (en lugar de utilizarlo para reemplazar la selección actual), se puede utilizar el método **ShapeRange.AddToSelection**.

Accesando a formas en una selección

Se puede utilizar un proceso similar para acceder a las formas en una selección así como acceder a las formas en un rango de selección.

Aquí tenemos una muestra de código VBA para acceder a las formas en una selección:

```
Dim shs As Shapes, sh As Shape
Set shs = ActiveSelection.Shapes
For Each sh In shs
  ' Hacer algo con la forma, sh
Next sh
```

Aquí tenemos una muestra de código VBA para acceder a las formas en un rango de selección:

```
Dim sRange As ShapeRange, sh As Shape
Set sRange = ActiveSelectionRange
For Each sh In sRange
  ' Hacer algo con la forma, sh
Next sh
```

Recuerde que el comando `ActiveSelection.Shapes` proporciona acceso directo a la selección actual, mientras que el comando `ActiveSelectionRange` proporciona una copia de la selección actual. Utilice `ActiveSelection.Shapes` si desea acceder a la selección actual; utilice `ActiveSelectionRange` si desea crear una "foto instantánea" de la selección actual a la que pueda acceder más tarde.

Reordenando las formas en una selección

El comando `ActiveSelection.Shapes` y el comando `ActiveSelectionRange` devuelven formas en el orden inverso en el cual fueron seleccionadas: la primer forma es la última que fue seleccionada, y la última forma es la primera que fue seleccionada. Por favor tenga en cuenta este hecho cuando reordene las formas en una selección.

Deseleccionando formas

Se puede deseleccionar cualquier forma cambiando el valor de su propiedad **Shape.Selected** a **False**.

Se pueden deseleccionar todas las formas utilizando el método **Document.ClearSelection**. El siguiente código VBA utiliza el método **ClearSelection** para deseleccionar todas las formas en el documento activo:

```
ActiveDocument.ClearSelection
```

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sea accionados al deseleccionar una forma:

- **Document.ShapeChange**

También puede utilizar manejadores de evento para responder a eventos que sea accionados al desactivar una selección:

- **Application.SelectionChange**
- **Document.SelectionChange**
- **GlobalMacroStorage.SelectionChange**

Duplicación de formas

Se puede utilizar el método **Shape.Duplicate** para duplicar una forma, y se puede utilizar el método **ShapeRange.Duplicate** para duplicar un rango de formas.

```
ActiveSelection.Duplicate
```

El método **Duplicate** proporciona dos parámetros opcionales, **OffsetX** y **OffsetY**, los cuales contrabalancean al duplicado del original (horizontal y verticalmente, respectivamente). El siguiente código VBA posiciona el duplicado dos pulgadas a la derecha y una pulgada hacia abajo del original:

```
ActiveDocument.Unit = cdrInch
ActiveSelection.Duplicate 2, 1
```

Transformación de formas

Se pueden transformar formas de diversas maneras, tal como se explica en los siguientes temas:

- Dimensionamiento de formas.
- Estiramiento de formas.
- Sesgamiento de formas.
- Rotación de formas.
- Posicionamiento de formas.

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sean accionados al transformar una forma:

- **Document.ShapeTransform**

Dimensionamiento de formas

Se puede devolver la anchura y altura de una forma (en unidades del documento) utilizando las propiedades **Shape.SizeWidth** y **Shape.SizeHeight**, como en el siguiente ejemplo VBA:

```
Dim width As Double, height As Double
ActiveDocument.Unit = cdrMillimeter
width = ActiveShape.SizeWidth
height = ActiveShape.SizeHeight
```

También se pueden utilizar las propiedades **Shape.SizeWidth** y **Shape.SizeHeight** para redimensionar una forma existente especificando nuevos valores para esas propiedades. El siguiente ejemplo VBA utiliza estas propiedades para ajustar el tamaño de la forma activa a una anchura de 50 milímetros y a una altura de 70 milímetros:

```
ActiveDocument.Unit = cdrMillimeter
ActiveShape.SizeWidth = 50
ActiveShape.SizeHeight = 70
```

Se puede devolver tanto la anchura como la altura de una forma (en unidades del documento) utilizando el método **Shape.GetSize**, como en el siguiente ejemplo VBA:

```
Dim width As Double, height As Double
ActiveDocument.Unit = cdrMillimeter
ActiveShape.GetSize width, height
```

Se puede redimensionar una forma utilizando el método **Shape.SetSize** para especificarle una nueva anchura y una nueva altura, como en el siguiente ejemplo VBA:

```
ActiveDocument.Unit = cdrMillimeter
ActiveShape.SetSize 50, 70
```

También se puede redimensionar una forma utilizando el método **Shape.SetSizeEx**. Además de la nueva anchura y la nueva altura de la forma, este método toma un punto de referencia para el redimensionamiento (en lugar de utilizar el punto central de la forma). El siguiente código VBA utiliza el método **SetSizeEx** para redimensionar la selección actual a 10 pulgadas de ancho por 8 pulgadas de alto tomando como referencia el punto (6, 5) en el documento:

```
ActiveDocument.Unit = cdrInch
ActiveSelection.SetSizeEx 6, 5, 10, 8
```

Si desea considerar el contorno de una forma dentro de la cantidad cuando se devuelva el tamaño de esa forma, se debe utilizar el método **Shape.GetBoundingBox**. La caja delimitadora de una forma circunda tanto a la forma como a su contorno: sin embargo, las dimensiones actuales de una forma especifican su anchura y altura independientemente del tamaño de su contorno. El siguiente ejemplo VBA utiliza el método **GetBoundingBox** para devolver el tamaño de la forma activa:

```
Dim width As Double, height As Double
Dim posX As Double, posY As Double
ActiveDocument.Unit = cdrInch
ActiveDocument.ReferencePoint = cdrBottomLeft
ActiveShape.GetBoundingBox posX, posY, width, height, True
```

El método **Shape.GetBoundingBox** toma parámetros que especifican la posición de la esquina inferior izquierda de la forma, la anchura de la forma y la altura de la forma. El parámetro final es un valor Booleano que indica si se incluye (**True**) o se excluye (**False**) el contorno de la forma. El método **Shape.SetBoundingBox** le permite ajustar el tamaño de una forma especificando el tamaño de su caja delimitadora; sin embargo, este método carece del parámetro para especificar si se incluye el contorno en el nuevo tamaño.

Si desea calcular el tamaño y posición de la caja delimitadora de una forma sin incluir su contorno, puede utilizar el método **GetBoundingBox** dos veces (una incluyendo el contorno y otra sin incluirlo), como en el siguiente ejemplo VBA:

```
Public Sub SetBoundingBoxEx(X As Double, Y As Double, _
    Width As Double, Height As Double)
  Dim sh As Shape
  Dim nowX As Double, nowY As Double
  Dim nowWidth As Double, nowHeight As Double
  Dim nowXol As Double, nowYol As Double
  Dim nowWidthol As Double, nowHeightol As Double
  Dim newX As Double, newY As Double
  Dim newWidth As Double, newHeight As Double
  Dim ratioWidth As Double, ratioHeight As Double
  Set sh = ActiveSelection
  sh.GetBoundingBox nowX, nowY, nowWidth, nowHeight, False
  sh.GetBoundingBox nowXol, nowYol, nowWidthol, nowHeightol, True
  ratioWidth = Width / nowWidthol
  ratioHeight = Height / nowHeightol
  newWidth = nowWidth * ratioWidth
  newHeight = nowHeight * ratioHeight
  newX = X + (nowX - nowXol)
  newY = Y + (nowY - nowYol)
  sh.SetBoundingBox newX, newY, newWidth, newHeight, False, _
    cdrBottomLeft
End Sub
```

Estiramiento de formas

Se puede estirar una forma (o escalarla estirándola proporcionalmente) al utilizar el método **Shape.Stretch** o el método **Shape.StretchEx**. Ambos métodos toman un valor decimal del estiramiento, donde 1 es 100% (o sin cambio); no se puede utilizar cero, más bien se debe utilizar un valor muy pequeño en su lugar.

El siguiente código VBA utiliza el método **Shape.Stretch** para estirar la selección a la mitad de su altura actual y el doble de su anchura, alrededor del centro del borde inferior de su caja delimitadora:

```
ActiveDocument.ReferencePoint = cdrBottomMiddle
ActiveSelection.Stretch 2, 0.5
```

Si desea especificar el punto de referencia en torno al cual se ejecute el estiramiento, se puede utilizar el método **Shape.StretchEx**. El siguiente código VBA ejecuta el mismo estiramiento que el código anterior, pero en torno al punto (4, 5) de la página (en pulgadas):

```
ActiveDocument.Unit = cdrInch
ActiveSelection.StretchEx 4, 5, 2, 0.5
```

Los métodos **Shape.Stretch** y **Shape.StretchEx** proporcionan un parámetro opcional Booleano que determina qué tanto se estira el texto de párrafo. Un valor de **True** estira los caracteres, mientras que **False** estira la caja delimitadora y reacomoda el texto dentro de ella.

Sesgado de formas

Se puede sesgar (torcer hacia un lado) una forma utilizando el método **Shape.Skew** o el método **Shape.SkewEx**. Estos métodos le permiten especificar el ángulo horizontal de sesgado (en grados, donde los valores positivos mueven el borde superior a la izquierda y el borde inferior a la derecha) y el ángulo vertical de sesgado (en grados, donde los valores positivos mueven el borde derecho hacia arriba y el borde izquierdo hacia abajo).

El sesgado con ángulos iguales o mayores a 90° no está permitido.

El sesgado horizontal se aplica entes que el sesgado vertical.

La diferencia entre los métodos **Shape.Skew** y **Shape.SkewEx** es el punto en torno al cual se ejecuta el sesgado: **Skew** utiliza el centro de rotación de la forma, mientras que **SkewEx** utiliza un punto de referencia especificado.

Se puede determinar el centro de rotación de una forma devolviendo los valores de sus propiedades **Shape.RotationCenterX** y **Shape.RotationCenterY**. Al cambiar estos valores se mueve el centro de rotación de esa forma.

El siguiente código VBA utiliza el método **Shape.Skew** para sesgar la selección (en torno a su centro de rotación) 30° horizontalmente y 15° verticalmente:

```
ActiveSelection.Skew 30, 15
```

Rotación de formas

Se puede rotar una forma utilizando el método **Shape.Rotate** o el método **Shape.RotateEx**. Estos métodos rotan la forma de acuerdo al ángulo dado (en grados). Sin embargo, la diferencia entre estos métodos es el punto en torno al cual se ejecuta la rotación: **Rotate** utiliza el centro de rotación de la forma, mientras que **RotateEx** utiliza un punto de referencia especificado.

Se puede determinar el centro de rotación de una forma devolviendo los valores de sus propiedades **Shape.RotationCenterX** y **Shape.RotationCenterY**. Al cambiar estos valores se mueve el centro de rotación de esa forma.

El siguiente código VBA utiliza el método **Shape.Rotate** para rotar la selección (en torno a su centro de rotación) 30°:

```
ActiveSelection.Rotate 30
```

El siguiente código VBA utiliza el método **Shape.RotateEx** para rotar cada forma seleccionada 15° en sentido de las manecillas del reloj en torno a su esquina inferior derecha:

```
Dim sh As Shape
ActiveDocument.ReferencePoint = cdrBottomRight
For Each sh In ActiveSelection.Shapes
  sh.RotateEx -15, sh.PositionX, sh.PositionY
Next sh
```

Posicionamiento de formas

Se puede devolver la posición horizontal y vertical de una forma utilizando las propiedades **Shape.PositionX** y **Shape.PositionY** (respectivamente). Alternativamente, se puede utilizar el método **Shape.GetPosition** para devolver tanto la posición horizontal como la vertical de una forma.

Se puede utilizar el método **Shape.GetBoundingBox** si se quiere devolver la posición de una forma, incluyendo su contorno. Para más información sobre este método, vea "Dimensionamiento de formas" en la página 125.

El siguiente código VBA utiliza el método **Shape.GetPosition** para devolver la posición de la selección correspondiente al punto de referencia actual del documento activo, cuyo código explícitamente la ajusta a la esquina inferior izquierda:

```
Dim posX As Double, posY As Double
ActiveDocument.ReferencePoint = cdrBottomLeft
ActiveSelection.GetPosition posX, posY
```

También se pueden utilizar las propiedades **Shape.PositionX** y **Shape.PositionY** para ajustar la posición horizontal y vertical de una forma (respectivamente), en consecuencia moviendo esa forma a la posición especificada. Alternativamente, se puede utilizar el método **Shape.SetPosition** para mover una forma a la posición horizontal y vertical específica, o se puede utilizar el método **Shape.SetPositionEx** para mover la forma a un punto especificado.

Se pueden utilizar los métodos **Shape.SetSizeEx** y **Shape.SetBoundingBox** para ajustar la posición de una forma. Para más información sobre estos métodos, vea "Dimensionamiento de formas" en la página 125.

El siguiente código VBA utiliza el método **Shape.SetPosition** para ajustar la posición de la esquina inferior derecha de cada forma seleccionada en el documento activo a (3, 2) en pulgadas:

```
Dim sh As Shape
ActiveDocument.Unit = cdrInch
ActiveDocument.ReferencePoint = cdrBottomRight
For Each sh In ActiveSelection.Shapes
  sh.SetPosition 3, 2
Next sh
```

Si lo desea, puede utilizar manejadores de evento para responder a eventos que sean accionados al posicionar una forma:

- **Document.ShapeMove**

Coloración de formas

Se le puede añadir color a una forma aplicándole un relleno (u objeto **Fill**). El tipo relleno de una forma es registrado por la propiedad **Fill.Type** como una de las siguientes constantes para la enumeración **cdrFillType**:

- **cdrUniformFill** — relleno uniforme.
- **cdrFountainFill** — relleno degradado.
- **cdrPatternFill** — relleno de patrón.
- **cdrTextureFill** — relleno de textura.
- **cdrPostScriptFill** — relleno PostScript.
- **cdrHatchFill** — relleno de malla.
- **cdrNoFill** — sin relleno

El siguiente código VBA devuelve el tipo de relleno que es aplicado a la forma activa:

```
Dim fillType As cdrFillType
fillType = ActiveShape.Fill.Type
```

No se puede cambiar el tipo de relleno de una forma modificando su propiedad **Fill.Type**. En vez de eso, se debe utilizar el método **Fill.Apply...Fill** apropiado, como se describe en las subsecciones siguientes.

También se le puede añadir color a una forma aplicándole un contorno (u objeto **Outline**).

Por otro lado, el modelo de objeto CorelDRAW proporciona una variedad de propiedades y métodos para trabajar con los colores (u objetos **Color**) que se aplican a las formas.

Para información sobre aplicación de rellenos y contornos y sobre el trabajo con colores, vea las siguientes subsecciones:

- Aplicación de rellenos uniformes.
- Aplicación de rellenos degradados.
- Aplicación de rellenos de patrón.
- Aplicación de rellenos de textura.
- Aplicación de rellenos PostScript.
- Aplicación de rellenos de malla.
- Aplicación de contornos.
- Trabajo con color.

En sus macros CorelDRAW, usted puede incluir rastreos que busquen formas que tengan propiedades de relleno, contorno o color especificas. Para información, vea "Incluyendo rastreos en macros" en el archivo de Ayuda de Macros de CorelDRAW.

Aplicación de rellenos uniformes

Los rellenos uniformes constan de un único color sólido. Un relleno uniforme está representado por la propiedad **Fill.UniformColor** como un objeto **Color**.

Se puede aplicar un relleno uniforme a una forma utilizando el método **Fill.ApplyUniformFill**. El siguiente ejemplo VBA aplica un relleno uniforme rojo a la forma activa:

```
ActiveShape.Fill.ApplyUniformFill CreateRGBColor(255, 0, 0)
```

Se puede cambiar el color de un relleno uniforme modificando su propiedad **Fill.UniformColor**. El siguiente ejemplo VBA cambia el relleno uniforme de la forma activa a azul marino profundo:

```
ActiveShape.Fill.UniformColor.RGBAssign 0, 0, 102
```

Se puede eliminar el relleno uniforme de una forma utilizando el método **Fill.ApplyNoFill**.

Aplicación de rellenos degradados

Los rellenos degradados muestran una progresión entre dos colores. Un relleno degradado está representado por la propiedad **Fill.Fountain** como un objeto **Fountain.Fill**, la cual especifica las distintas propiedades del relleno degradado: color inicial, color final, ángulo, tipo de mezcla y demás. Los colores en un relleno degradado están representados por una colección **FountainColors**.

Se puede aplicar un relleno degradado a una forma utilizando el método **Fill.ApplyFountainFill**. Este método proporciona parámetros opcionales para varios ajustes de relleno degradado, tal como el punto medio y el desajuste de la mezcla. El siguiente ejemplo VBA crea un simple relleno degradado lineal, de rojo a amarillo, a 30 grados del horizontal:

```
Dim startCol As New Color, endCol As New Color
startCol.RGBAssign 255, 0, 0
endCol.RGBAssign 255, 255, 0
ActiveShape.Fill.ApplyFountainFill startCol, endCol, cdrLinearFountainFill, 30
```

Se le puede añadir un color a un relleno degradado utilizando el método **FountainColors.Add**. Las posiciones del color son valores enteros en porcentaje, donde 0% es la posición del color inicial y 100% es la posición del color final. El siguiente ejemplo VBA agrega un color verde al relleno degradado en la posición cercana a un tercio (33%) desde el color rojo ya existente:

```
Dim fFill As FountainFill
Set fFill = ActiveShape.Fill.Fountain
fFill.Colors.Add CreateRGBColor(0, 102, 0), 33
```

Es posible mover un color en un relleno degradado utilizando el método **FountainColor.Move**. El siguiente código VBA mueve el color verde del ejemplo anterior a una posición que está a 60% desde el rojo (es decir, más hacia el amarillo):

```
ActiveShape.Fill.Fountain.Colors(1).Move 60
```

Se puede utilizar la propiedad **FountainColors.Count** para determinar el número de colores entre el color inicial y el color final de un relleno degradado. (Para el ejemplo anterior, este valor es 1). El primer color en la colección es el color inicial, y su número de índice es **0**; este color no puede ser movido, pero su color puede ser cambiado. El último color en la colección es el color final, y su número de índice es (**Count** + 1); este color no puede ser movido, pero su color puede ser cambiado. El siguiente código VBA cambia el color final de amarillo a azul:

```
Dim cols As FountainColors
Set cols = ActiveShape.Fill.Fountain.Colors
cols(cols.Count + 1).Color.RGBAssign 0, 0, 102
```

Se puede eliminar un relleno degradado de una forma utilizando el método **Fill.ApplyNoFill**.

Aplicación de rellenos de patrón

Los rellenos de patrón muestran una serie de objetos de vector repetitivos o imágenes de mapas de bit. Un relleno de patrón está representado por la propiedad **Fill.Pattern** como un objeto **PatternFill**, el cual especifica las distintas propiedades del relleno de patrón: color de primer plano, color de fondo, descentrado de mosaico y demás.

CorelDRAW almacena la colección de rellenos de patrón disponibles en la colección **PatternCanvases**.

Se le puede aplicar un relleno de patrón a una forma utilizando el método **Fill.ApplyPatternFill**.

Se puede eliminar un relleno degradado de una forma utilizando el método **Fill.ApplyNoFill**.

Aplicación de rellenos de textura

Los rellenos de textura son generados fractalmente y llenan una forma con una imagen en vez de una serie de imágenes repetitivas. Un relleno de textura está representado por la propiedad **Fill.Texture** como un objeto **TextureFill**, el cual especifica las distintas propiedades para el relleno de textura: origen, resolución, descentrado de mosaico y demás.

CorelDRAW almacena las propiedades de un relleno de textura en una colección **TextureFillProperties**.

Es posible aplicar un relleno de textura a una forma utilizando el método **Fill.ApplyTextureFill**.

Se puede eliminar el relleno de textura de una forma utilizando el método **Fill.ApplyNoFill**.

Aplicación de rellenos PostScript

Los rellenos PostScript son rellenos de textura que están diseñados para utilizarse en lenguaje PostScript. Un relleno PostScript está representado por la propiedad **Fill.PostScript** como un objeto **PostScriptFill**, el cual especifica las distintas propiedades para el relleno PostScript.

Se puede aplicar un relleno PostScript a una forma utilizando el método **Fill.ApplyPostScriptFill**.

Se puede eliminar el relleno PostScript de una forma utilizando el método **Fill.ApplyNoFill**.

Aplicación de rellenos de malla

Los rellenos de malla están compuestos de líneas basadas en vectores y pueden ser utilizados para distinguir claramente los materiales o relaciones de los objetos en un dibujo. Un relleno de malla está representado por la propiedad **Fill.Hatch** como un objeto **HatchFill**, el cual especifica las distintas propiedades para el relleno de malla.

CorelDRAW almacena la colección de patrones de rellenos de malla disponibles en la colección **HatchPatterns**, y cada documento CorelDRAW almacena su propia biblioteca de patrones de relleno de malla en una colección **HatchLibraries**.

Se puede aplicar un relleno de malla a una forma utilizando el método **Fill.ApplyHatchFill**.

Se puede eliminar el relleno de malla de una forma utilizando el método **Fill.ApplyNoFill**.

Aplicación de contornos

Se pueden utilizar las distintas propiedades y métodos de la clase **Outline** para definir el contorno de una forma.

La propiedad **Outline.Type** utiliza las siguientes constantes de la enumeración **cdrOutlineType** para registrar si la forma especificada tiene un contorno:

- **cdrOutline** — indica que la forma tiene un contorno.
- **cdrNoOutline** — indica que la forma no tiene un contorno.

Si una forma no tiene contorno, al ajustar su propiedad **Outline.Type** a **cdrOutline** se aplica el estilo de contorno por defecto del documento.

Si una forma tiene contorno, al ajustar su propiedad **Outline.Type** a **cdrNoOutline** se elimina ese contorno.

La propiedad **Outline.Width** para un contorno ajusta su anchura en unidades del documento. En el siguiente ejemplo VBA, el contorno de las formas seleccionadas es ajustado a 1 milímetro:

```
ActiveDocument.Unit = cdrMillimeter
ActiveSelection.Outline.Width = 1
```

Si una forma no tiene contorno, su valor **Outline.Width** es **0**. El cambio de este valor aplica un contorno y automáticamente cambia el valor de la propiedad **Outline.Type** de **cdrNoOutline** a **cdrOutline**.

De manera similar, si una forma tiene contorno, su valor **Outline.Width** es más grande que **0**. El cambio de este valor a **0** elimina el contorno y automáticamente cambia el valor de la propiedad **Outline.Type** de **cdrOutline** a **cdrNoOutline**.

La propiedad **Outline.Color** para un contorno define su color, como en el siguiente ejemplo VBA:

```
ActiveSelection.Outline.Color.GrayAssign 0 ' Se ajusta a negro
```

Al ajustar el color de un contorno se ajusta automáticamente la propiedad **Outline.Type** de ese contorno a **cdrOutline** y se aplica la anchura predeterminada de contorno.

La propiedad **Outline.Style** para un contorno especifica los ajustes de guión de ese contorno. Estos ajustes de guión están definidos por las siguientes propiedades de la clase **OutlineStyle**:

- **DashCount** — representa el número de pares de guiones y espacios intermedios en un contorno. Este valor va de 1 a 5.
- **DashLength** — representa la longitud de cada guión en un contorno. Este valor se calcula como un múltiplo de la anchura del contorno, el cual se mide en unidades del documento. Por ejemplo, si `DashLength(1)` es `5` y el contorno es 0.2" de ancho, la longitud del guión es de 1"; sin embargo, si la anchura de la línea se cambia a 0.1", la longitud del guión pasa a 0.5".
- **GapLength** — representa la longitud de cada espacio en blanco en un contorno. Este valor se calcula como un múltiplo de la anchura del contorno, el cual es medido en unidades del documento.
- **Index** — representa el número indicativo de un estilo de contorno predefinido en la colección **OutlineStyles** de la aplicación. La colección **OutlinesStyles** es personalizable, de modo que el número indicativo que está asociado con cada estilo de contorno en la colección puede variar de usuario en usuario; sin embargo, la expresión `OutlineStyles.Item(0)` siempre especifica una línea sólida.

Los objetos **Outline** tienen muchas otras propiedades, incluyendo las siguientes:

- **StartArrow** y **EndArrow** — especifica la punta de flecha de cada final e inicio de curva.
- **LineCaps** y **LineJoin** — respectivamente, especifica el tipo de finales de línea (cabezada, redondeada o cuadrada) y uniones de línea (biselada, mitrada o redondeada).
- **NibAngle** y **NibStretch** — especifica la forma de la punta utilizada para dibujar el contorno.
- **BehindFill** y **ScaleWithShape** — respectivamente, dibuja la línea detrás del relleno y escala el contorno con la forma.

Los objetos **Outline** también tienen métodos, incluyendo los siguientes:

- **ConvertToObject** — convierte el contorno en un objeto.
- **SetProperties** — ajusta la mayoría de las propiedades de contorno disponibles en una sola llamada.

Trabajando con color

La clase **Color** define los colores de relleno y de contorno que se aplican a las formas. Esta clase proporciona una variedad de propiedades y métodos para el trabajo con color.

Se puede determinar el modelo de color de un color accesando a su propiedad **Color.Type**, como en el siguiente ejemplo VBA:

```
Dim colType As cdrColorType
colType = ActiveShape.Outline.Color.Type
```

La propiedad **Color.Type** está definida por la enumeración **cdrColorType**, la cual proporciona las siguientes constantes (entre muchas otras) para modelos de color soportados:

- **cdrColorCMYK** — especifica el modelo de color CMYK.
- **cdrColorRGB** — especifica el modelo de color RGB.
- **cdrColorGray** — especifica el modelo de color de escala de grises.

Los componentes de color de cada modelo de color soportado están definidos por propiedades adicionales de la clase **Color**, como se demuestra en los siguientes ejemplos VBA:

- Modelo de color CMYK — está definido por las propiedades **Color.CMYKCyan**, **Color.CMYKMagenta**, **Color.CMYKYellow** y **Color.CMYKBlack**.
- Modelo de color RGB — está definido por las propiedades **Color.RGBRed**, **Color.RGB** y **Color.RGBBlue**.
- Modelo de color de escala de grises — está definido por la propiedad **Color.Gray**.

El rango de valores que están soportados por un componente de color depende del modelo de color de ese componente.

Para crear un color, se puede utilizar la palabra clave de automatización **New**, como en `Dim col As New Color`.

Para asignarle un modelo de color a un nuevo color, se puede utilizar el método deseado **...Assign** (tal como **Color.CMYKAssign**, **Color.RGBAssign** o **Color.GrayAssign**). Cada uno de estos métodos proporciona un parámetro para cada componente de color en su respectivo modelo de color. Por ejemplo, `col.RGBAssign 0, 0, 102` asigna un color azul profundo RGB al nuevo color que fue creado en el consejo anterior.

Para utilizar los ajustes de administración de color de CorelDRAW para cambiar el modelo de color que está asignado a un color, se puede emplear el método deseado **ConvertTo...** (tal como **Color.ConvertToCMYK**, **Color.ConvertToRGB** o **Color.ConvertToGray**). Por ejemplo, `ActiveShape.Fill.UniformColor.ConvertToRGB` convierte el relleno de la forma activa al modelo de color RGB.

Se pueden copiar las propiedades de un color a otro utilizando el método Color.CopyAssign, como en el siguiente ejemplo VBA:

```
Dim sh As Shape
Set sh = ActiveShape
sh.Outline.Color.CopyAssign sh.Fill.UniformColor
```

El color "ninguno" no existe. Para ajustar un color de relleno o de contorno a "ninguno", se debe ajustar más bien el tipo de relleno o el tipo de contorno a "ninguno".

Aplicación de efectos a formas

El modelo de objeto CorelDRAW proporciona una variedad de métodos para aplicarle efectos a las formas. Para información sobre estos métodos, vea las siguientes subsecciones:

- Aplicación de mezclas.
- Aplicación de siluetas.
- Aplicación de efectos personalizados.
- Aplicación de distorsiones.
- Aplicación de sombras.
- Aplicación de envolturas.
- Aplicación de extrusiones.
- Aplicación de transparencias.
- Aplicación de perspectivas.

La aplicación de un efecto devuelve un objeto **Effect**, el cual le permite acceder a distintas propiedades y métodos para el efecto creado. Por ejemplo, se puede utilizar el método **Effect.Separate** para separar las formas que sean generadas por un efecto de la forma en la cual ese efecto es aplicado. Además, se puede utilizar el método **Effect.Clear** para eliminar un efecto de una forma.

Aplicación de mezclas

El método **Shape.CreateBlend** crea una mezcla entre la forma actual y la forma que se especifique como un parámetro. Este método proporciona parámetros opcionales para varios ajustes de mezcla, tales como la aceleración de la mezcla y la trayectoria a lo largo de la cual la mezcla es creada.

El siguiente código VBA crea una mezcla básica de 10 etapas:

```
Dim sh As Shapes, eff As Effect
Set sh = ActiveSelection.Shapes
Set eff = sh(1).CreateBlend(sh(2), 10)
```

En el ejemplo anterior, el número de formas en la mezcla es doce: las formas inicial y final, más las diez etapas de mezcla que son creadas.

El método **Shape.CreateBlend** devuelve un objeto **Effect**, en el que la propiedad **Effect.Blend** se puede utilizar para modificar la mezcla creada.

Aplicación de siluetas

El método **Shape.CreateContour** le aplica una silueta a una forma. Este método proporciona parámetros opcionales para varios ajustes de silueta, tales como los colores y la aceleración de la silueta.

El siguiente código VBA crea una silueta de tres etapas en un espaciado de cinco milímetros:

```
Dim eff As Effect
ActiveDocument.Unit = cdrMillimeter
Set eff = ActiveShape.CreateContour(cdrContourOutside, 5, 3)
```

El método **Shape.CreateContour** devuelve un objeto **Effect**, en el que la propiedad **Effect.Contour** se puede utilizar para modificar la silueta creada.

Aplicación de efectos personalizados

El método **Shape.CreateCustomEffect** le aplica un efecto personalizado a una forma. Este método proporciona parámetros para varios ajustes en los efectos.

El método **Shape.CreateCustomEffect** devuelve un objeto **Effect**, en el que la propiedad **Effect.Custom** se puede utilizar para modificar el efecto creado.

Aplicación de distorsiones

Los siguientes métodos le aplican una distorsión a una forma:

- **Shape.CreatePushPullDistortion** — le aplica una distorsión Push-and-pull.
- **Shape.CreateTwisterDistortion** — le aplica una distorsión de Pincel Deformador.
- **Shape.CreateZipperDistortion** — le aplica una distorsión de Pincel Agreste.
- **Shape.CreateCustomDistortion** — le aplica una distorsión personalizada.

Estos métodos proporcionan parámetros para varios ajustes de distorsión.

Los métodos de distorsión-creación devuelven un objeto **Effect**, en el que la propiedad **Effect.Distortion** se puede utilizar para modificar las distorsiones creadas.

Si lo desea, puede utilizar manejadores de evento que respondan a eventos que sean accionados al distorsionar una forma:

- **Document.ShapeDistort**

Aplicación de sombras

El método **Shape.CreateDropShadow** le aplica una sombra a una forma. Este método proporciona parámetros para varios ajustes de sombra, tales como el fundido y el desajuste de la sombra.

El método **Shape.CreateDropShadow** devuelve un objeto **Effect**, en el que la propiedad **Effect.Custom** se puede utilizar para modificar la sombra creada.

Aplicación de envolturas

Los siguientes métodos le aplican una envoltura a una forma:

- **Shape.CreateEnvelope** — le aplica una envoltura básica.
- **Shape.CreateEnvelopeFromCurve** — le aplica una envoltura utilizando como plantilla la curva especificada.
- **Shape.CreateEnvelopeFromShape** — le aplica una envoltura utilizando como plantilla la forma especificada.

Estos métodos proporcionan parámetros para varios ajustes de envoltura.

Los métodos de creación de envolturas devuelven un objeto **Effect**, en el que la propiedad **Effect.Envelope** se puede utilizar para modificar las envolturas creadas.

Aplicación de extrusiones

El método **Shape.CreateExtrude** le aplica una extrusión a una forma. Este método proporciona parámetros para varios ajustes de extrusión, tales como el ángulo y el color de la extrusión.

El método **Shape.CreateExtrude** devuelve un objeto **Effect**, en el que la propiedad **Effect.Extrude** se puede utilizar para modificar la extrusión creada.

Aplicación de transparencias

El método **Shape.CreateLens** le aplica un efecto de transparencia a una forma. Este método proporciona parámetros para varios ajustes de transparencia, tales como el color y la cantidad de transparencia.

El método **Shape.CreateLense** devuelve un objeto **Effect**, en el que la propiedad **Effect.Lens** se puede utilizar para modificar la transparencia creada.

Aplicación de perspectiva

El método **Shape.CreatePerspective** le aplica un efecto de perspectiva a una forma. Este método proporciona parámetros para especificar puntos de desvanecimiento horizontales y verticales.

El método **Shape.CreatePerspective** devuelve un objeto **Effect**, en el que la propiedad **Effect.Perspective** se puede utilizar para modificar la perspectiva creada.

Búsqueda de formas

En sus macros CorelDRAW, usted puede incluir rastreos que busquen las formas que tengan propiedades específicas de forma, tales como propiedades de relleno, de contorno o de color. Para hacer esto, utilice Corel Query Language (CQI) junto con uno de los siguientes métodos:

- **Shape.Evaluate** — devuelve el resultado de una expresión dada que evalúa las propiedades de la forma actual.
- **Shape.FindShape** — devuelve una única forma que tenga las propiedades especificadas.
- **Shape.FindShapes** — devuelve, como un rango de formas, todas las formas que tengan las propiedades especificadas.

Se pueden especificar las propiedades de las formas que sean buscadas. Por ejemplo, la expresión `ActiveShape.Evaluate("@name")` busca la propiedad **Name** de todas las formas seleccionadas.

Considere el siguiente ejemplo VBA, en el cual la propiedad **Type** y la propiedad **Width** son utilizadas para seleccionar todos los rectángulos que sean más anchos de dos pulgadas:

```
ActivePage.Shapes.FindShapes(Query := "@type = 'rectangle' and _
@width > {2 in}").CreateSelection
```

Para información extensa sobre el uso de CQL, vea "Incluyendo rastreos en macros" en el archivo de Ayuda de Macros de CorelDRAW.

Eliminación de formas

Si lo desea, puede utilizar manejadores de evento que respondan a eventos que sean accionados al eliminar una forma:

- **Document.ShapeDelete**

Trabajo con filtros de importación y filtros de exportación

Como se discutió previamente, CorelDRAW proporciona métodos para importación de archivos (vea "Importación de archivos dentro de capas" en la página 101) y para exportación de archivos (vea "Exportación de archivos desde documentos" en la página 86).

Estos métodos de importación y exportación de archivos también pueden ser utilizados para ejecutar conversiones en lote o para modificar archivos depositarios.

La amplia selección de formatos de archivo que soporta CorelDRAW se debe al vasto número de filtros que están disponibles en la aplicación. Cada filtro le permite trabajar con archivos de otras aplicaciones gráficas. Para aprender más acerca del trabajo con estos filtros, vea los siguientes temas:

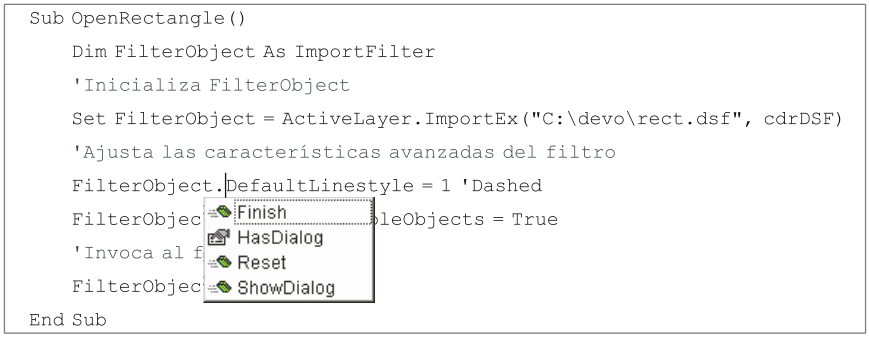
- Trabajo con filtros de importación.
- Trabajo con filtros de exportación.

Trabajo con filtros de importación

Para asegurar la portabilidad de un script de archivo de importación, se debe utilizar el objeto **ImportFilter** predeterminado (en lugar del objeto **DSFImport** específico del filtro), como en el siguiente ejemplo VBA:

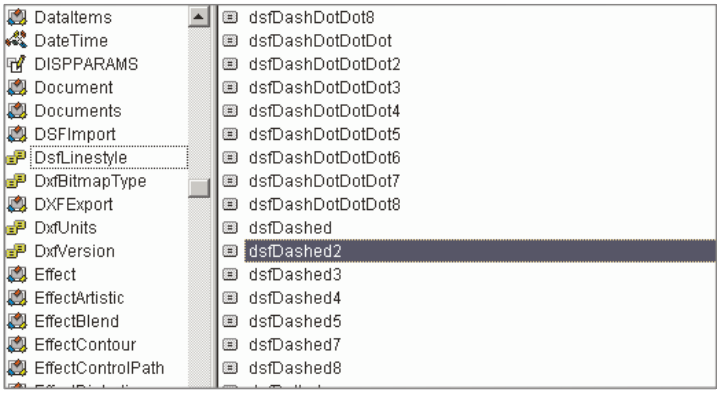
```
Sub OpenRectangle()
    Dim FilterObject As ImportFilter
    'Inicializa FilterObject
    Set FilterObject = ActiveLayer.ImportEx("C:\devo\rect.dsf", _
cdrDSF)
    'Ajusta las características avanzadas del filtro
    FilterObject.DefaultLinestyle = 1 'Dashed
    FilterObject.DeleteInvisibleObjects = True
    'Invoca al filtro
    FilterObject.Finish
End Sub
```

Para mejores resultados, utilice el objeto **DSFImport** del filtro específico sólo para aprender las interfaces específicas que estén soportadas por un filtro. Por ejemplo, la siguiente captura de pantalla ilustra que el objeto **ImportFilter** expone sólo interfaces genéricas en Microsoft® IntelliSense® debido a que la interfaz **ImportFilter** es genérica (y no un filtro específico).



El objeto **ImportFilter** no contiene las propiedades **DefaultLinestyle** ni **DeleteInvisibleObjects**. Sin embargo, aún se pueden ajustar estas propiedades en la interfaz **ImportFilter** si son soportadas por el filtro de importación específico.

Como se discutió previamente, el uso del objeto **ImportFilter** (en lugar del objeto **DSFImport** del filtro específico) asegura que un script de archivo de importación pueda ser utilizado en cualquier otra estación de trabajo que ejecute la misma versión de CorelDRAW. Para referenciar las propiedades, métodos y enumeraciones de un filtro específico, localice ese filtro en el Administrador de Objetos. Por ejemplo, la siguiente captura de pantalla ilustra que el estilo de línea **dsfDashed2** puede ser especificado al asignarle un valor de 7 a la propiedad **DefaultLinestyle**.



Para acceder al modelo de objeto de un filtro, haga click en **Herramientas** » **Referencias** desde el Editor de macros. En el cuadro de diálogo **Referencias** que aparezca, haga click en **Examinar...** y navegue a la carpeta **Filters** de la ruta de instalación de CorelDRAW Graphics Suite X5. Seleccione el archivo de enlace dinámico de biblioteca (DLL) del filtro deseado y luego haga click en **Aceptar**. Cuando vuelva a aparecer el cuadro de diálogo **Referencias**, habilite la casilla de verificación que corresponda al filtro deseado y luego haga click en **Aceptar**. Ahora puede acceder al modelo de objeto del filtro, como en el siguiente ejemplo VBA:

```
Sub OpenRectangleDSF()
    Dim FilterObject As DSFImport
    Dim Style As DsfLinestyle
    'Inicializa FilterObject
    Set FilterObject = ActiveLayer.ImportEx("C:\devo\rect.dsf", cdrDSF)
    'Ajusta las características avanzadas del filtro
    Style = dsfDashed
    FilterObject.DefaultLinestyle = Style
    FilterObject.DeleteInvisibleObjects = True
    'Invoca al filtro
    FilterObject.Finish
End Sub
```

```
Sub OpenRectangleDSF()
    Dim FilterObject As DSFImport
    Dim Style As DsfLinestyle

    'Inicializa FilterObject
    Set FilterObject = ActiveLayer.ImportEx("C:\devo\rect.dsf", cdrDSF)

    'Ajusta las características avanzadas del filtro
    Style = dsfDashed
    FilterObject.DefaultLinestyle = Style
    FilterObjec                                  ts = True
                DefaultLinestyle
                DeleteInvisibleObjects
    'Invoca al  Finish
    FilterObjec HasDialog
End Sub         Reset
                ShowDialog
```

El trabajar con un filtro de importación es mucho más fácil si se tiene acceso al script del modelo de objeto de ese filtro; sin embargo, como se discutió, esta técnica reduce la portabilidad del script. Cuando se utilice en otra estación de trabajo, el script primero debe ser actualizado con la correcta localización del archivo DLL para el filtro.

Trabajo con filtros de exportación

El siguiente ejemplo VBA demuestra cómo guardar un documento como un archivo AutoCAD DXF utilizando un filtro de exportación:

```
Sub SaveRectangleDXF()
    Dim FilterObject As DXFExport
    Dim BitmapType As DxfBitmapType
    Dim TextAsCurves As Boolean
    Dim Units As DxfUnits
    Dim Version As DxfVersion
    'Inicializa FilterObject
    Set FilterObject = ActiveDocument.ExportEx("C:\devo\rect.dxf", cdrDXF)
    'Ajusta las características avanzadas del filtro
    BitmapType = dxfBitmapGIF
    FilterObject.BitmapType = BitmapType
    Units = dxfInches
    FilterObject.Units = Units
    TextAsCurves = False
    FilterObject.TextAsCurves = TextAsCurves
    Version = dxfVersion2000
    FilterObject.Version = Version
    'Invoca al filtro
    FilterObject.Finish
End Sub
```

En el ejemplo anterior, se llamó al método **ActiveDocument.ExportEx** y la interfaz del filtro de exportación (**DXFExport**) fue invocada. Sin embargo, se puede utilizar la interfaz genérica de exportación (**ExportFilter**) en lugar de la interfaz específica del filtro (**DXFExport**), como en el siguiente ejemplo VBA:

```
Sub SaveRectangle()
    Dim FilterObject As ExportFilter
    'Inicializa FilterObject
    Set FilterObject = ActiveDocument.ExportEx("C:\devo\rect.dxf", cdrDXF)
    'Ajusta las características avanzadas del filtro
    FilterObject.BitmapType = 1 'GIF
    FilterObject.Units = 0 'Inches
    FilterObject.TextAsCurves = False
    FilterObject.Version = 1 'AutoCAD 2000
    'Invoca al filtro
    FilterObject.Finish
End Sub
```

El siguiente ejemplo VBA demuestra cómo invocar al cuadro de diálogo **Exportar**:

```
Sub ShowExportDialog()
  Dim FilterObject As ExportFilter
  Dim vntReturn As Variant
  'Inicializa FilterObject
  Set FilterObject = ActiveDocument.ExportEx("C:\devo\rect.dxf", cdrDXF)
  'Si FilterObject soporta a diálogo, lo invoca
  If (FilterObject.HasDialog = True) Then
    vntReturn = FilterObject.ShowDialog
    'Verica que el usuario hizo click en "Aceptar" y no en "Cancelar"
    If (vntReturn = True) Then
      'Invoca al filtro
      FilterObject.Finish
    End If
  End If
End Sub
```

El ejemplo anterior requiere que se verifiquen los valores devueltos del cuadro de diálogo y que se invoque al método **Finish** para cuando el usuario haga click en **Aceptar**.

Glosario

alcance

La visibilidad de un tipo de dato, procedimiento u objeto.

archivo GMS

También llamado un "archivo de proyecto" (y abreviatura de archivo "Global Macro Storage"), lugar en el cual el Editor de Macros almacena todos los módulos de un proyecto.

argumento

Vea "parámetro".

arreglo

Un conjunto de objetos indizados secuencialmente del mismo tipo de datos (o "arreglo de elementos").

Cada arreglo de elementos tiene el mismo tipo de datos (aunque los elementos pueden tener valores diferentes), y el arreglo completo se almacena contiguamente en memoria (sin huecos entre elementos). Por ejemplo, se podría tener un arreglo de enteros o un arreglo de caracteres o un arreglo de lo que sea que esté definido como un tipo de datos.

Por defecto, los arreglos se indizan con base cero.

Los arreglos pueden tener más de una dimensión. Un arreglo unidimensional es llamado un "vector", mientras que un arreglo bidimensional es llamado una "matriz".

atajo de objeto

Un reemplazo sintáctico de la versión de escritura normal de un objeto.

automatización

El proceso de grabación o escritura de una macro.

clase

La definición de cada propiedad, método y evento que se aplica a un tipo de objeto en la aplicación.

colección

Un grupo de objetos que tienen características similares y acciones similares pero que están identificados exclusivamente por nombres indizados o números indizados.

Las colecciones son siempre plurales. Por ejemplo, **Documents** es una colección de objetos **Document**.

constante

Un valor en una estructura de automatización programada que permanece fija mientras se ejecuta la macro.

A diferencia de una variable, la cual temporalmente almacena un valor cambiante de datos en un procedimiento de código o función de código, los valores constantes no cambian.

Una constante es una instancia de una enumeración.

cuadro de diálogo modal

Un cuadro de diálogo que bloquea la aplicación y sobre el que se debe actuar (es decir, ya sea aceptándolo o cancelándolo) antes de que la macro pueda reanudarse.

La mayoría de los cuadros de diálogo incorporados que pueden controlarse por código de automatización son de tipo modal.

cuadro de diálogo no modal

Un cuadro de diálogo que no bloquea la aplicación y puede dejarse abierto mientras el usuario continúa trabajando en la aplicación.

Los cuadros de diálogo de tipo no modal se comportan como ventanas acoplables.

enumeración

También llamada un "tipo enumerado", un tipo de datos que lista todos los valores posibles de las variables que lo utilizan.

A diferencia de una variable, la cual temporalmente almacena un valor cambiante de datos, una enumeración almacena valores fijos.

Una constante es una instancia de una enumeración.

espacio intermedio

Un espacio entre guiones en un estilo de contorno.

evento

Una acción que se lleva a cabo en un objeto y que se reconoce por una forma o control.

Cada objeto dentro de un modelo de objeto está definido por una propiedad, método o evento, o una combinación de ellos. Un evento es activado por una acción — tal como un click de ratón, una tecla presionada o un temporizador de sistema — y se puede escribir código que provoque que un objeto responda a ese evento.

forma

Un tipo de módulo que se utiliza para cuadros de diálogo e interfaces de usuario personalizadas, y que incluye el código que los controla.

función

Un procedimiento que ejecuta una tarea dada en una macro y que se puede utilizar para devolver un valor.

Un procedimiento de función inicia con una declaración `Function` y finaliza con una declaración `End Function`. En VBA y VSTA, las funciones no necesitan ser declaradas antes de utilizarse, ni antes de definirse.

macro

Un conjunto de tareas grabadas o escritas que se pueden invocar repetidamente dentro de una aplicación.

Una macro es un símbolo, nombre o clave que representa una lista de comandos.

manejador de evento

Una subrutina que se programa para provocar que la aplicación responda a un evento específico.

método

Una operación que un objeto puede haber ejecutado sobre sí mismo.

mezcla de colores

El proceso de simulación de color al colocar muy cercanos los puntos de otro color.

El sistema operativo Windows utiliza mezcla de colores para mostrar los colores que la tarjeta gráfica no puede mostrar.

modelo de objeto

Una estructura de alto nivel de las relaciones entre los objetos padre y los objetos hijo en una aplicación.

Por ejemplo, el objeto **Application** representa el inicio en la jerarquía de objetos. A partir del objeto **Application**, se puede "escudriñar hasta el fondo" y navegar a través del modelo de objeto hasta encontrar el objeto deseado. Para referenciar un objeto con código Visual Basic, se separa cada nivel de la jerarquía del objeto con el operador punto (.).

módulo

Un contenedor que es utilizado por un archivo GMS para almacenar componentes del proyecto.

Los módulos genéricos son utilizados por código general y por macros. Otros tipos de módulos incluyen módulos de formas y de clase.

módulo de clase

Un tipo de módulo que contiene la definición de una clase Visual Basic orientada a objetos, incluyendo las definiciones de las propiedades y métodos de esa clase.

número indicativo

Una referencia a un objeto en una colección que contiene más de un objeto.

Un número indicativo se utiliza para identificar cada objeto en una colección. El número indicativo puede extenderse desde 1 hasta el número de objetos disponibles dentro de la colección.

objeto

Cuando se hace referencia a un modelo de objeto, una instancia de una clase.

parámetro

Sinónimo de "argumento", un valor que es pasado a una rutina y que define una característica de un objeto en el ambiente de programación Visual Basic.

Los parámetros son atributos que aparecen después de un comando grabado en la ventana acoplable **Recorder**. Por ejemplo, las opciones de cuadro de diálogo no son grabadas como comandos separados en la ventana acoplable **Recorder**, éstas se graban como atributos del comando que inicialmente invocó al cuadro de diálogo.

pasada por referencia

El acto de pasar un argumento a una función o subrutina utilizando una referencia del original.

Por defecto, los parámetros de función y los parámetros de subrutina son pasados por referencia. Para indicar expresamente que se quiere pasar un argumento por referencia, se prefija el argumento con `ByRef`.

pasada por valor

El acto de pasar un argumento a una función o subrutina utilizando una copia del original.

Para indicar expresamente que se quiere pasar un argumento por valor, se prefija el argumento con `ByVal`.

programación de evento dado

Un estilo de programación, a diferencia de un procedimiento tradicional de programación (en el cual los programas inician en la línea 1 y se ejecutan línea por línea), que ejecuta código en respuesta a eventos.

Visual Basic for Applications es un lenguaje de programación de evento dado. La mayoría de los códigos que se crean son escritos para responder a un evento.

Compare con "programación orientada a objetos".

programación orientada a objetos

Un estilo de programación que le da énfasis a la creación y el uso de objetos.

Compare con "programación de evento dado".

propiedad

Una característica de una clase.

Las propiedades pueden ser devueltas o establecidas. Además, las propiedades pueden ser designadas como de sólo lectura (para indicar que están fijadas por el diseño de la clase).

rango

Una serie de objetos similares.

String (cadena)

Un tipo de dato que consiste en una secuencia de caracteres contiguos que representan los caracteres mismos en lugar de sus valores numéricos.

Una cadena puede incluir letras, números, espacios y puntuaciones. El tipo de dato `String` puede almacenar cadenas de longitud fija que van en un rango de caracteres de 0 a 63K y cadenas dinámicas que van en un rango de 0 a 2 billones de caracteres aproximadamente. El carácter de signo de dólar (**$**) de tipo declaración representa una cadena en Visual Basic.

sub

Vea "subrutina".

subrutina

Algunas veces llamada una "sub", un procedimiento que ejecuta una tarea dada en una macro pero que no puede ser utilizada para devolver un valor.

Un procedimiento de subrutina inicia con una declaración `Sub` y finaliza con una declaración `End Sub`. En VBA y VSTA, las subrutinas no necesitan ser declaras antes de utilizarse, ni antes de ser definidas.

tipo enumerado

Vea "enumeración".

valor global

Un valor que se aplica a un proyecto dado en su totalidad.

variable

Un elemento que puede ser creado (o "declarado") para los propósitos de almacenamiento de datos.

Los tipos de datos incorporados son `Boolean`, `Double`, `Integer`, `Long`, `Single`, `String`, `Variant` y varios otros tipos menos utilizados incluyendo `Date`, `Decimal` y `Object`. Si una variable no es declarada antes de ser utilizada, el compilador la interpreta como de tipo `Variant`.

Variant

El tipo de dato para todas las variables que no son declaradas como otro tipo, tal como `Dim`, `Private`, `Public` o `Static`.

El tipo de dato `Variant` no tiene carácter de tipo de declaración.

VBA

Un lenguaje de programación incorporado que puede automatizar funciones repetitivas y crear soluciones inteligentes en CorelDRAW y Corel PHOTO-PAINT.

VBA es un subconjunto del ambiente de programación Microsoft Visual Basic (VB) orientado a objetos, pero es considerado "para aplicaciones" porque está con más frecuencia integrado dentro de otra aplicación para personalizar la funcionalidad de esa aplicación.

Visual Basic for Applications

Vea "VBA".

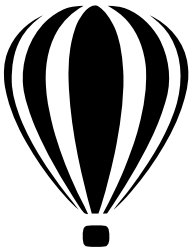
Visual Studio Tools for Applications

Vea "VSTA".

VSTA

El sucesor de VBA.

VSTA está basado en Microsoft Visual Studio 2008. El ambiente de desarrollo integrado (IDE, por sus siglas en inglés) para VSTA se puede utilizar con CorelDRAW y Corel PHOTO-PAINT para soportar dos lenguajes de programación adicionales (Visual Basic.NET y C#) y sacar ventaja de la plataforma .NET nativamente.



Indice

D

E

N

O

P

R

S

T

U

V

W

Z

Guía de Programación de Macros CorelDRAW® Graphics Suite.

Las especificaciones del producto, precios, empaquetado, soporte e información técnica ("especificaciones") se refieren sólo a la versión en Inglés. Las especificaciones para todas las otras versiones (incluyendo versiones en otros lenguajes) pueden variar.

011129

Guía traducida al español por el Corel-Experto: Felipe de Jesús Guzmán Llamas, Guadalajara, México.
Publicada con autorización en el WWW.CORELCLUB.ORG - Club Internacional de Usuarios de Corel.