Delphi vs. Visual Basic.

Una comparación entre estas dos herramientas RAD.

Una comparación entre Delphi y Visual Basic

Introducción

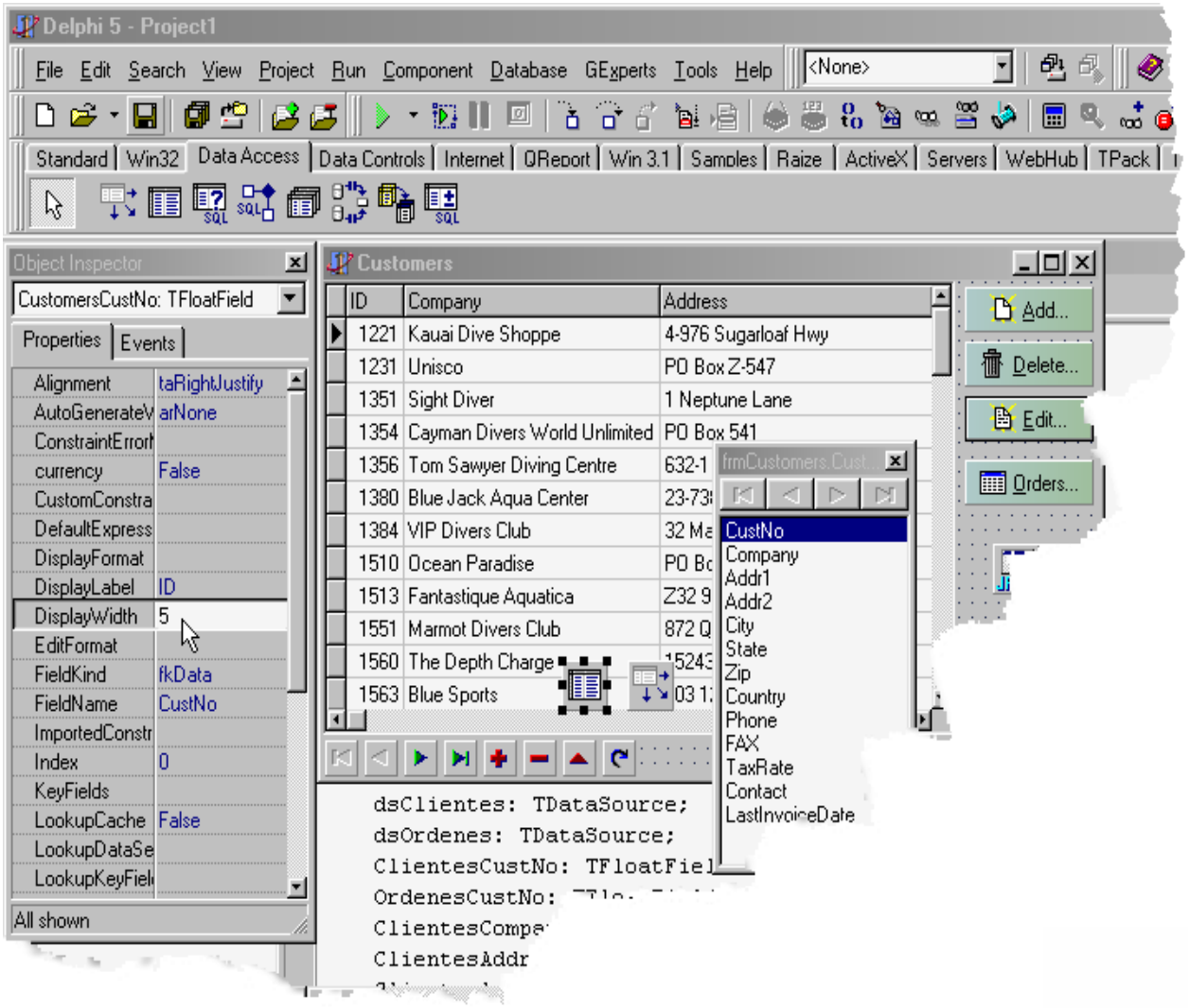
¿Cuál es mejor? ¿Delphi o Visual Basic? Es una pregunta simple y directa, pero desafortunadamente no tiene una respuesta simple y directa. La razón es que —aunque ambas son herramientas RAD visuales— tienen particularidades y diferencias muy significativas que hacen imposible llegar a una conclusión directa y general. Lo que podemos hacer sin embargo es analizar y comparar sus diversas características esperando que le sirva como guía para la evaluación. Mi elección ha sido Delphi porque en la comparación le di más peso a ciertos factores, priorizándolos sobre los otros, y entonces el resultado fué Delphi. Con requisitos diferentes, sería perfectamente válido elegir el Visual Basic. Su opción de lenguaje queda en usted y dependerá de la importancia que le dé a cada característica.

Curva de aprendizaje

Visual Basic es muy fácil de aprender y de utilizar, no solamente porque el lenguaje de programación no es OOP y es fácil de aprender y codificar (al final viene de BASIC), sino también porque el IDE es simple y cómodo de usar, y los objetos de base de datos que vienen con Visual Basic son más fáciles de utilizar que sus contrapartes de Delphi, aunque claro que no son tan potentes.

Visual Basic hace muchas cosas por el programador. Por ejemplo, los objetos cuentan referencias y esto significa que por ejemplo si creamos un objeto asignándolo a una variable local, el objeto será liberado automáticamente cuando la función o el procedimiento finalice (a menos que lo asignamos a una variable no local). Visual Basic tiene un sistema de administración sofisticado de memoria y utiliza un "colector de basura" (garbage colector) así que es rápido liberando memoria.

En cuanto al acceso a bases de datos, en Visual Basic es más simple. Se usa un solo componente que abre el Recordset, ofrece interfaz visual de navegador y se enlaza con los controles de datos, mientras que en Delphi hay que usar tres componentes para ello. La gran ventaja de los Recordsets de Visual Basic sobre los Datasets de Delphi es que los primeros manejan consultas actualizables automáticamente: se puede tener una consulta de dos tablas y que sea "viva", mientras que en Delphi tenemos que utilizar actualizaciones cacheadas y un componente UpdateSQL con las respectivas consultas SQL para insertar, actualizar o eliminar un registro. Además, cuando uno actúa contra un servidor de base de datos en Delphi tiene que utilizar un componente Transaction, mientras que esto no es necesario en Visual Basic. Visto desde la perspectiva de un programador Visual Basic, el acceso a datos de Delphi resulta ser demasiada molestia cuando se lo compara con los Recordsets y el Control de Datos de Visual Basic que simplifican notoriamente esta cuestión.



En Delphi, la funcionalidad del Data Control de Visual Basic se divide en tres componentes. En la imagen se puede observar que los datos están disponibles en tiempo de diseño y que por ejemplo uno puede cambiar la etiqueta predeterminada de los campos, así como su ancho de visualización, formato de visualización, máscara de edición, etc.

Delphi es más difícil de aprender que Visual Basic, pero no para quienes están familiarizados por ejemplo con Turbo Pascal, FreePascal, o incluso C/C++. Delphi es más difícil de usar, pero tiene sus ventajas. Por ejemplo, los objetos generalmente no cuentan referencias y esto significa que el programador tiene que encargarse de liberar los objetos no usados creados por un procedimiento o una función cuando ese procedimiento o función termina. La ventaja es que tenemos más libertad en la manipulación del objeto y podemos liberarlo cuando no lo necesitemos más, sin importar cuántas variables apunten a él. En cuanto a la administración de memoria, Delphi no tiene un colector de basura como Visual Basic, pero tiene su propio sistema de administración de memoria, el que está optimizado para bloques pequeños de datos. El acceso a base de datos puede resultar algo incómodo al principio para quienes están acostumbrados a Visual Basic, pero es muy potente, flexible y extensible.

El IDE merece una mención especial. Particularmente, el editor de código de Visual Basic tiene algunas cosas interesantes. Borland inventó el "Code Insight", pero personalmente pienso que Microsoft lo implementó mejor. Los valores posibles de una variable o un parámetro, o las propiedades o los métodos posibles de un objeto aparecen automática e inmediatamente. El formato de mayúsculas/minúsculas también es una característica agradable (por ejemplo si uno declara una variable con el nombre "Doc", y después escribe "DOc", "doc", "dOc", etc., la variable se ajusta al formato que aparece en la sentencia declarativa —es decir "Doc"—). Una cosa apreciable del IDE de Visual Basic es que cuando uno está depurando puede modificar una sentencia y continuar la ejecución con los cambios en efecto, sin tener que recomenzar la aplicación. Otra característica agradable del Visual Basic es la ventana Inmediato, donde uno puede ejecutar sentencias interpretadas.

Pero la facilidad de programación tiene que venir a expensas de algo. Visual Basic es fácil de aprender al principio y no requiere una base fuerte de educación formal en programación para poder aprenderlo, pero en tanto que uno avanza y demanda más y más de él, comienza uno a encontrarse conque VB tiene limitaciones serias que comienzan a recordarle que no por nada la "B" de Basic viene de "Beginner" —principante— (BASIC = Beginner's All-purpose Symbolic Instruction Code = código simbólico de instrucciones de propósito general para principiantes). Esto no significa necesariamente que los programadores de Visual Basic se cambiarán eventualmente a un lenguaje más poderoso. Al contrario: la mayoría de los usuarios de Visual Basic serán realmente felices con él por años mientras nunca necesitan ir más allá de sus límites, pero la verdad es que Visual Basic ha sido el "lenguaje de entrada" para muchos de los que hoy programan en Delphi. Aquí hay algunos testimonios:

- Why Borland's Delphi?

Un hecho notable sobre Delphi es que uno puede realmente hacerlo todo y puede acceder "fácilmente" a todo lo que la máquina y el sistema operativo tienen para ofrecer. En Visual Basic, para llamar una función API hay que buscar la declaración de la función API, constantes y tipos en una base de datos para copiar y pegar en los programas. Visto desde la perspectiva de un programador Delphi, esto es demasiada molestia cuando se lo compara con Delphi, donde uno puede simplemente llamar una función API como si fuera una función incorporada. Por supuesto, un lenguaje OOP es muy difícil de aprender si uno no tiene una base fuerte de programación, pero una vez que uno lo domina, le da la capacidad de escribir código reutilizable, extensible y fácil de mantener. Con los componentes de datos de Delphi uno puede tener abierta una tabla o consulta de la base de datos y ver los datos en tiempo de diseño, algo agradable en el momento de fijar el ancho de las columnas en una rejilla de datos. El IDE de Delphi no ofrece las características que describí arriba para el de Visual Basic, pero ofrece otras cosas. Por ejemplo el Editor de Código de Delphi ofrece la posibilidad elegir el esquema de mapeo del teclado y plantillas de código para ahorrarle escritura (algo que curiosamente esperaba encontrar en Visual Basic en vez de Delphi). En lo que hace a depuración de errores, en Delphi uno dispone de algunas características avanzadas como las ventanas CPU y FPU, y la posibilidad de depurar programas multi-hilos.

> "*Visual Basic hace más fáciles las cosas fáciles, Delphi hace más fáciles las cosas difíciles*"
> (Computerworld, 1998).

Propósito

¿Para qué se propone utilizar el lenguaje de programación? Esta una cuestión clave al elegir un lenguaje de programación. Aquí hay un testimonio interesante de un ex-programador de Visual Basic:

- Visual Basic Hardcore - por Bruce McKinney, 1999

Y abajo está la contestación de un programador de Visual Basic que admite las limitaciones Visual Basic, pero también dice una cosa importante: cada lenguaje "*tiene sus fortalezas y debilidades únicas, a pesar del "bombo" de comercialización producido por aquellos con intereses creados en esos lenguajes*" en una clara alusión a... Bueno, mejor léalo usted mismo:

- La herramienta incorrecta para el trabajo correcto - por Don Kiely, 1999

Visual Basic es ideal para simples aplicaciones de interfaz de usuario (front-end), e inadecuado para cosas más complejas, mientras que Delphi es ideal tanto para la interfaz de usuario (front-end) como para el procesamiento de fondo (back-end). ¿Qué significa esto? Por ejemplo, con Visual Basic uno puede desarrollar una herramienta de compresión (como Winzip) diseñando la interfaz de la aplicación y escribiendo algo de código para llamar a las rutinas de compresión de una determinada DLL u objeto ActiveX (escrito en otro lenguaje) que proporcione la capacidad de compresión. Con Delphi uno puede hacer lo mismo que con Visual Basic, pero además puede escribir la DLL o el objeto ActiveX que realiza la tarea de compression/decompression. No es que sea imposible escribir una DLL o un objeto ActiveX en Visual Basic: es técnicamente factible, pero muy inconveniente (véase Rendimiento y tamaño del código).

> Delphi es más un lenguaje de "todo propósito" que Visual Basic

Cuota de mercado

Uno de los méritos que se deben reconocer a Visual Basic es que —gracias a su sencillez— puso la

programación en Windows al alcance de literalmente millones de personas.

En particular, el éxito de Visual Basic en la comunidad de habla hispana es abrumador porque a la facilidad de empleo le tenemos que agregar que el IDE, la ayuda en línea y la documentación impresa están traducidos al español. Oí que en España Delphi se entrega con la documentación impresa en español. Hey Borland, ¿por qué no tenemos esa opción en los países de habla hispana de América Latina? Están perdiendo a lo grande aquí...

El propósito y la facilidad de uso han acomodado a Visual Basic con una cuota de mercado muchísimo más grande que la de Delphi en el mercado global. Sin embargo, en ciertos nichos de mercado, Delphi supera notoriamente a Visual Basic.

PROGRAMACIÓN "INTERNA"

Las medianas y grandes empresas que emplean programadores para desarrollar y mantener sus sistemas de gestión interna eligen claramente a Visual Basic por sobre Delphi. Este es el mercado de "aplicaciones simples de interfaz de usuario", y a sabiendas que no hay tantos programadores en Delphi como en Visual Basic, los ejecutivos optan por Visual Basic porque sencillamente prefieren tener un ejército de programadores a sus pies cuando necesitan a alguien.

Estamos hablando de empresas que tienen bastante dinero como para pagar las licencias de la última versión de Visual Basic (véase Obsolescencia), para estar al día con el hardware (véase Rendimiento y tamaño del código), y para pagar por una solución cuando su personal IT no puede desarrollar algo en Visual Basic debido a sus limitaciones intrínsecas.

Solamente aquellas empresas que demandan más que "aplicaciones simples de interfaz de usuario" de su personal IT usan Delphi.

En una especie de proceso de realimentación, o de "círculo vicioso" si lo prefiere, la posición dominante de Visual Basic presiona a los programadores a seguir el camino de Visual Basic para ingresar al mercado laboral, derivando en otra razón para elegir Visual Basic en vez de Delphi.

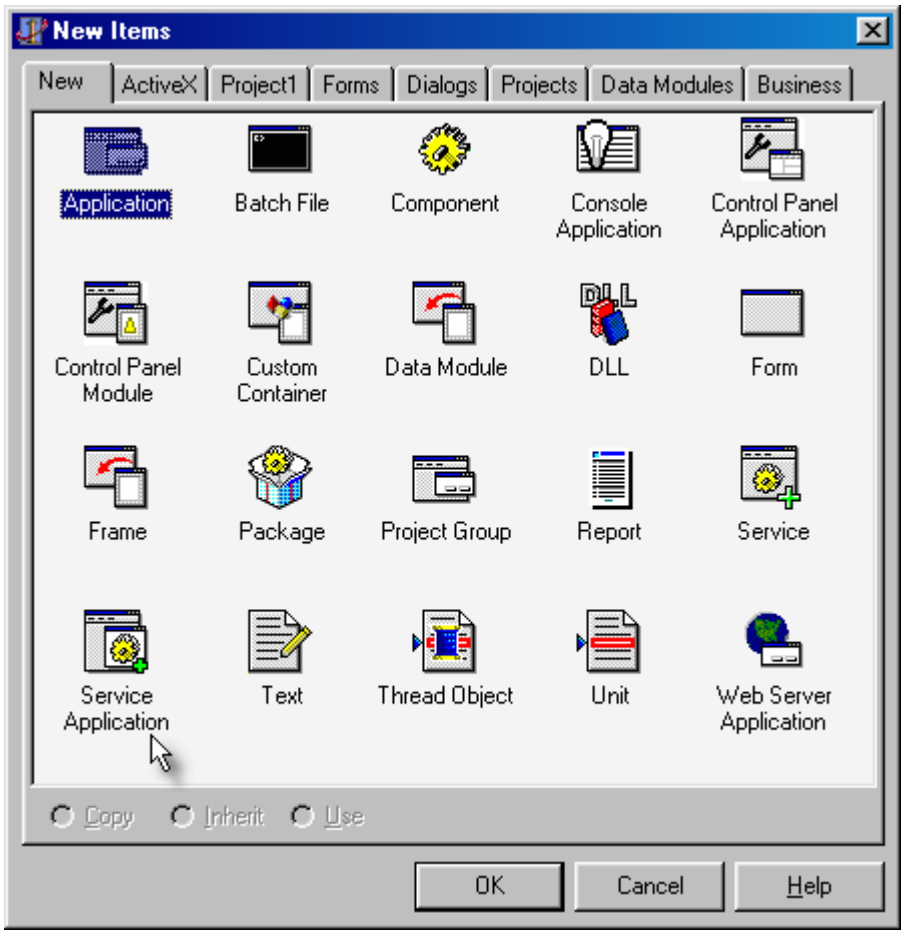
DESARROLLADORES DE SISTEMAS DE GESTIÓN

Caen en esta categoría las corporaciones de software, firmas de consultoría de software y los desarrolladores (individuales) independientes que se dedican al desarrollo de sistemas de gestión (facturación, inventarios, nómina, etc.), generalmente hechos a la medida.

La posición de Delphi es quizás un poco mejor en este mercado, pero Visual Basic todavía gana por mucho porque es casi el mismo caso que la programación interna que discutí arriba, sólo que simplemente es llevada a adelante por un tercero.

Sin embargo, en el caso de los desarrolladores individuales podemos ver una inversión de papeles. En esta subcategoría particular, Visual Basic gana solamente porque hay muchos programadores en Visual Basic desempleados que toman de vez en cuando trabajos de desarrolladores independientes mientras que siguen buscando un trabajo fijo, pero aquellos que realmente han decidido ganarse la vida como desarrolladores independientes de software claramente han hecho de Delphi su opción para el desarrollo bajo Windows (por lo menos éste es el caso de todos mis amigos), siendo quizás cinco razones dominantes (entre otras) que:

1. Los que deciden ganarse la vida como desarrolladores independientes no son principiantes
2. Estando relevados de la presión de utilizar Visual Basic para incorporarse al mercado del trabajo, quedan en libertad de elegir la herramienta de desarrollo que desean
3. Las aplicaciones Delphi tienen menos requisitos de sistema (la pequeña y mediana empresa —las que acuden a desarrolladores individuales— no pueden actualizar su hardware regularmente como las empresas grandes)
4. Hay un montón de componentes VCL freeware disponibles (los desarrolladores individuales generalmente no pueden invertir demasiado en la compra de soluciones de otros desarrolladores de software)
5. No pueden permitirse actualizar su hardware y herramienta de desarrollo cada año, así que prefieren comprar una herramienta que pueda permanecer con ellos por largo tiempo (véase Obsolescencia)



El rendimiento de Delphi no lo limita a simples aplicaciones de intarfaz de usuario. Con Delphi uno puede crear DLLs, objetos ActiveX, servidores y aplicaciones de servicios, etc.

OTROS DESARROLLADORES DE SOFTWARE

Finalmente, tenemos el mercado de aquellos que desarrollan aplicaciones shareware y comerciales (diferentes de los sistemas de gestión), controles ActiveX, bibliotecas, etc. En este mercado, Visual Basic tiene una posición muy débil comparado con Delphi, aunque para hacer honor a la verdad, C++, particularmente Visual C++ todavía prevalece en este mercado, pero puedo ver cómo Delphi está creciendo lenta pero constantemente en este mercado a expensas de Visual C++ porque ofrece un poder y rendimiento más cercano a C++ y una facilidad de uso más cercana a Visual Basic, trayéndonos lo mejor de ambos mundos, más la adición de una jerarquía de componentes visuales y no visuales listos para usar, fáciles de utilizar, altamente estandarizados y fácilmente extensibles (la VCL).

> Visual Basic es la elección de los ejecutivos, los principiantes y de quienes se ven forzados a utilizarlo para conseguir un o simplemente para seguir las masas. Delphi es la opción de los desarrolladores independientes.

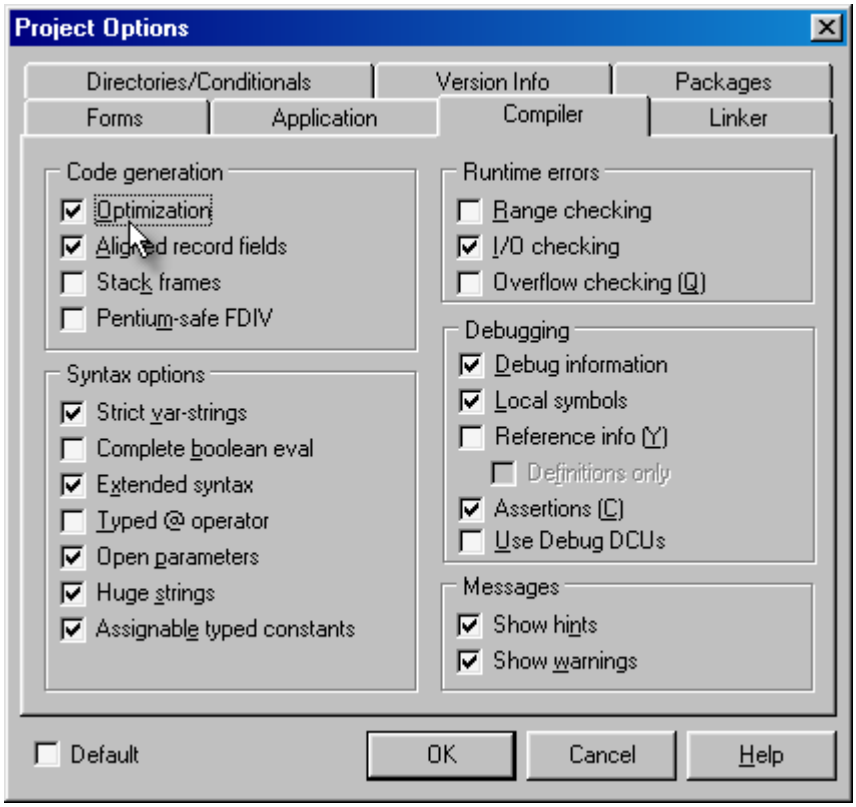
Rendimiento y tamaño del código

El único ítem donde he visto que Visual Basic tiene un buen rendimiento es en el acceso a datos, y quizás en los informes, porque la capa de acceso a datos y la generación de reportes no están hechas con Visual Basic, pero en lo que respecta estrictamente a generación de código, a pesar del hecho que Visual Basic ahora produce código nativo en vez de pseudocódigo (P-Code), todavía está muy lejos de alcanzar una velocidad comparable a Delphi, por más optimizaciones que uno le active. Esto sucede porque Delphi tiene un compilador optimizante y hasta trae opciones de compilación para optimizar la velocidad de su código. Para decir más, si uno no está contento con el funcionamiento de una rutina en Delphi, puede afinar el código para exprimir hasta el último nanosegundo, hasta el punto que —si uno quiere— puede incluir instrucciones en lenguaje ensamblador. En lo que hace a rendimiento de código, Delphi es ideal para tareas que requieren programación pesada.

Aparte del rendimiento del código generado, la VCL realiza cacheo de los objetos de Windows (por ejemplo

fuentes, plumas, brochas, etc.) para lograr un uso más eficiente de los recursos de Windows.

Mientras más recursos del sistema, mejor —por supuesto—, pero la velocidad de ejecución y el uso eficiente de recursos permiten que las aplicaciones Delphi funcionen razonablemente bien en las viejas máquinas Pentium con poca memoria, como las que todavía se encuentran en muchas empresas pequeñas y medianas.



Delphi tiene un compilador optimizante. Algunas opciones (optimizaciones, alineación de campos de registros, sin marcos de pila —*stack frames*—, sin evaluación boolena completa y sin controles de rango ni de sobreflujo —*overflow*—) ayudan a los profesionales a obtener lo mejor de su código. De ser necesario, se pueden usar directivas de compilador para activar/dsactivar las distintas opciones de compilación en el mismo código fuente.

La velocidad y el tamaño del código generalmente conflictúan, pero cuando comparamos Delphi y Visual Basic, Delphi gana claramente en ambos extremos. Una aplicación pequeña de base de datos usando ADO y algunos de terceros cabe en un solo diskette de instalación en el caso de Delphi, mientras que su contraparte del Visual Basic cabría en no menos de dos o tres diskettes (en ambos casos no estoy incluyendo ADO que ocupa aproximadamente 8 Mb). ¿Por qué hay tanta diferencia? Delphi optimiza el tamaño del código todo lo que puede, genera ejecutables independientes (puede también generar ejecutables que trabajen con las bibliotecas runtime si uno desea) y utiliza SmartLinking (enlace inteligente) para evitar incluir procedimientos y funciones en el ejecutable que no están referenciados en el código. Visual Basic no tiene otra opción más que trabajar con bibliotecas runtime y objetos ActiveX (que son más grandes y de no tan alto rendimiento como sus contrapartes VCL).

> Delphi produce aplicaciones independientes que son pequeñas y entregan gran rendimiento, haciéndolas más amistosas con los recursos del sistema y los anchos de banda de los módems, haciendo de Delphi una opción ideal sobre Visual Basic para los desarrolladores independientes, tanto aquellos que producen soluciones a medida para PyMEs como los que producen aplicaciones shareware o comerciales a distribuir por Internet.

Extensibilidad

Visual Basic y Delphi son extensibles. Ambos IDEs soportan "add-ons" (agregados) para añadir asistentes y otras extensiones al entorno, pero lo que es más interesante para los programadores —o mejor dicho, más necesario— es la posibilidad de utilizar objetos visuales y no visuales aparte de los que vienen originalmente con la herramienta de desarrollo.

OBJETOS ACTIVEX

En Visual Basic uno puede utilizar objetos ActiveX. Vienen en archivos .OCX que son como DLLs que tienen funciones especiales para exponer objetos con sus propiedades, métodos y eventos. El acceso a propiedades y métodos de objetos ActiveX es lento porque se acceden con IDs. Las propiedades de cadena de los objetos ActiveX son UNICODE, mientras que las aplicaciones trabajan normalmente con cadenas ANSI, agregando otra sobrecarga (la conversión UNICODE-ANSI o ANSI-UNICODE) al acceder a propiedades de cadena.

Los objetos ActiveX proporcionan encapsulación, pero no herencia. La herencia se puede simular declarando los mismos métodos y propiedades de la "clase base" e implementándolos como llamadas a los métodos y las propiedades respectivos de la "clase base". La "clase derivada" no es por consiguiente otra cosa sino una envoltura de la "clase base". No sólo que esto hace el código fuente largo y difícil de mantener, sino que también hace la invocación de propiedades y de métodos del nuevo objeto más de dos veces de lenta que el objeto original.

COMPONENTES VCL

Aquí es donde la OOP hace la diferencia. En Delphi uno puede usar objetos ActiveX si quiere, pero Delphi ofrece algo muchísimo mejor: la VCL (Visual Component Library), una jerarquía de clases que van desde las que ofrecen una funcionalidad básica común hasta aquellas altamente especializadas como una tabla de base de datos. De estas clases uno puede derivar sus propias clases, y uno tiene que codificar solamente lo que quiera agregar o redefinir, siendo ésta la razón por la cual los componentes de VCL son mucho más pequeños, más rápidos y más fácilmente mantenibles que los objetos ActiveX. Son más rápidos debido a las siguientes razones:

1. Las referencias a las propiedades y métodos se resuelven en tiempo de compilación ("early binding": enlace temprano) o —en el caso de métodos virtuales o dinámicos— en tiempo de ejecución ("late binding": enlace tardío), pero en este último caso, el código usado para obtener la dirección de un método en la VMT (Virtual Methods Table: tabla de métodos virtuales) es muy pequeño y eficiente comparado con la manera usada por ActiveX para encontrar sus métodos y propiedades. Si una propiedad hace referencia directa a un miembro de datos, no se llama a ninguna función. Con ActiveX siempre hay una llamada a una función.

2. Normalmente la mayoría de las propiedades y de los métodos de la clase base no se redefinen, así que cuando uno invoca una de estas propiedades y métodos, la llamada va directamente a las de la clase base sin tener que pasar por ningún intermediario. La herencia de ActiveX es simulada, haciendo más caras todas las invocaciones a métodos y propiedades.

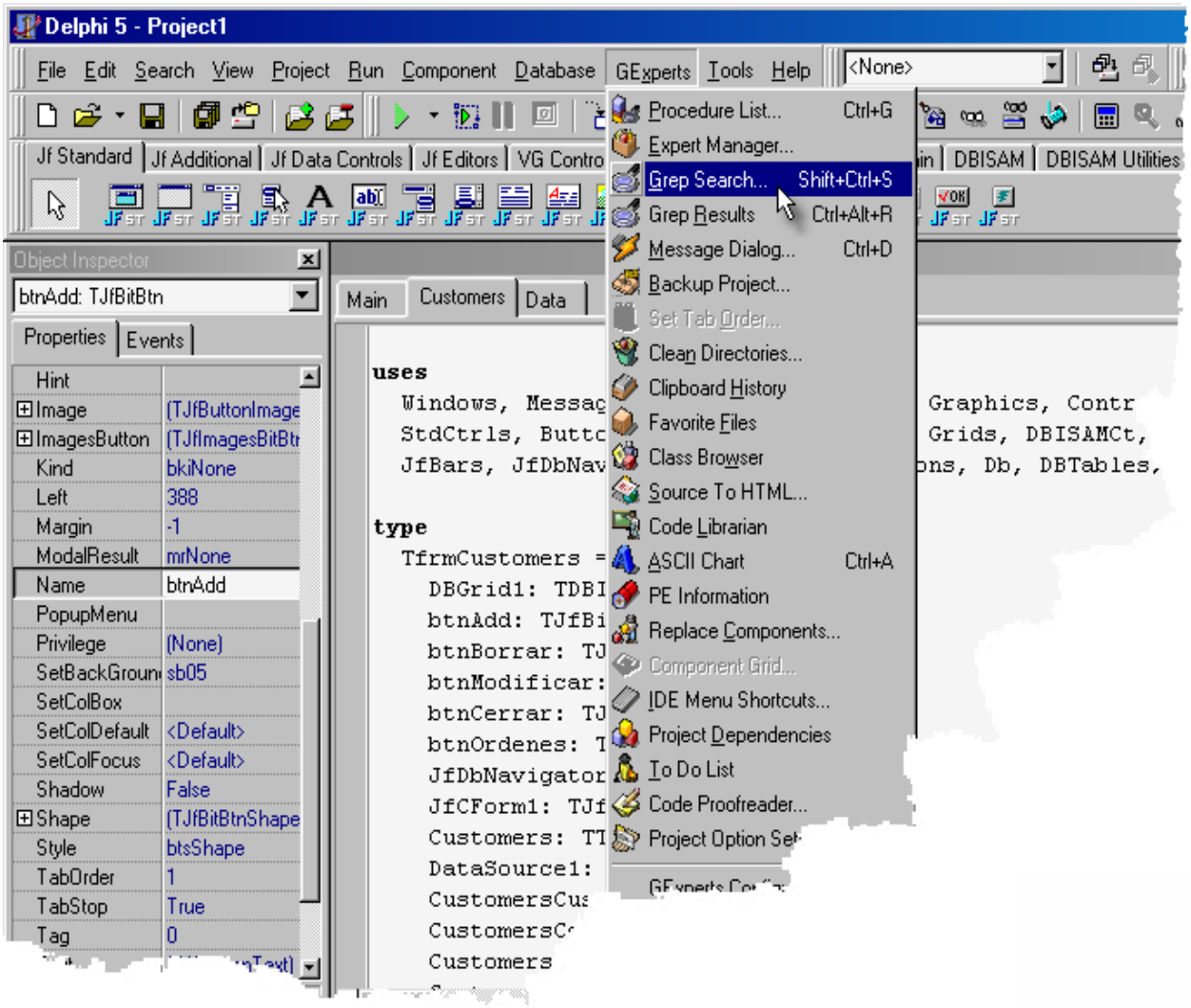
3. Uno utiliza normalmente cadenas ANSI (no cadenas UNICODE como los objetos ActiveX), así que no hay ninguna sobrecarga por conversión.

4. Son parte del ejecutable, y eso significa que no hay que abrir un archivo para usarlos, cargar una lista de las funciones que exponen, etc.

Además de ser más rápidos y significativamente más pequeños, los componentes VCL consumen menos recursos del sistema que sus contrapartes ActiveX y su mecanismo de herencia permite la extensibilidad.

DISPONIBILIDAD DE BIBLIOTECAS, OBJETOS, ETC. DE TERCEROS

Hay muchos objetos ActiveX y DLLs en la red que se pueden utilizar con Visual Basic, pero la mayoría es de paga y no vienen con código fuente, ni siquiera después que uno paga sus licencias, y cuando traen código fuente, por lo general éste está en otro lenguaje (generalmente Visual C++ o Delphi), y esto significa que si uno desea modificar estos fuentes necesita 1) saber algo de ese lenguaje, y 2) tener la herramienta de desarrollo respectiva, y no creo que valga la pena comprar una licencia tan sólo para ocasionalmente hacer modificaciones menores a objetos y bibliotecas.



La disponibilidad de muchos componentes VCL y expertos para el IDE de altísima calidad, tanto freeware como de paga, es uno de los puntos más fuertes de Delphi ya que aceleran el desarrollo de aplicaciones y ayudan a los desarrolladores a agregar funcionalidades particulares un gran atractivo visual a sus aplicaciones.

Las perspectivas en Delphi son mucho mejores, porque además de ActiveX y de DLLs, uno puede encontrar muchos componentes VCL en la red, siendo freeware la mayoría de ellos, muchos de ellos vienen con código fuente, y la mayoría de los que son de paga vienen generalmente con el código fuente después de que uno lo registra o por una tarifa adicional. ¿Y en qué lenguaje está escrito el código fuente? Una muy pequeña minoría están en C++ Builder, pero la enorme mayoría están en Delphi, por supuesto, así que uno no tiene que aprender un nuevo lenguaje o utilizar una herramienta de desarrollo diferente si desea modificarlos.

> Delphi no se limita a ActiveX. Los componentes VCL son más rápidos, más pequeños, por lo general los fuentes están disponibles (y en Delphi) y el mecanismo de la herencia permite derivar nuevos componentes VCL rápida fácilmente.

Obsolescencia

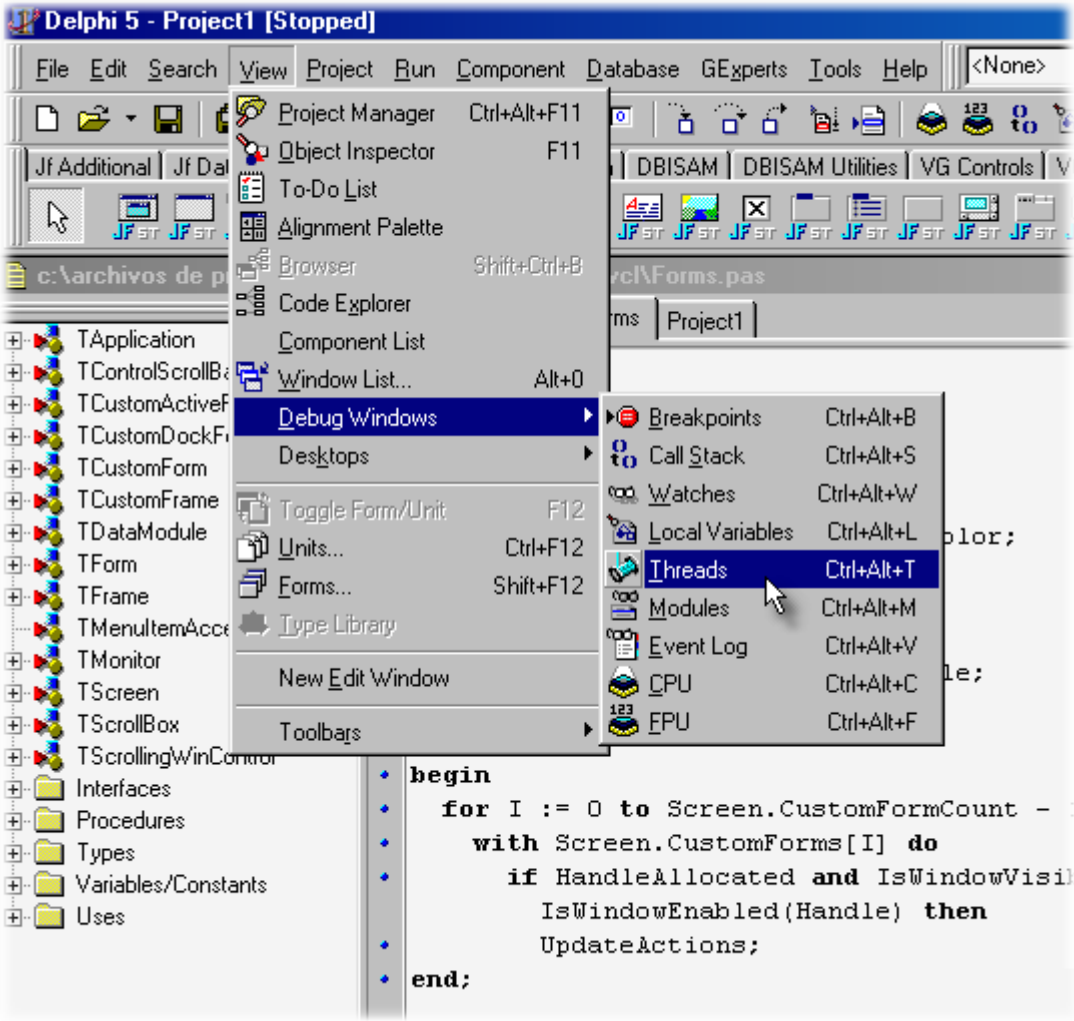
Entre programadores en Visual Basic, ¿cuántos de ellos todavía están programando en Visual Basic 4.0? ¿Y cuántos están programando en Visual Basic 5.0? Casi todos los programadores en Visual Basic que conozco están programando en Visual Basic 6. Con Delphi sucede algo totalmente diferente. Mucha gente todavía está programando en Delphi 3 o 4, y quizás la minoría tiene Delphi 5, cuando Delphi 6 acaba de salir a la luz. ¿Por qué es eso? ¿Es que las actualizaciones son demasiado caras? No... Pensándolo bien, sí lo son! Permítanme explicarme. Lo que sucede con Visual Basic es que —ávidos de más características— los programadores Visual Basic se actualizan rápidamente tan pronto como hay una nueva versión disponible. Delphi 3 C/S es tan poderoso que todavía hoy no es obsoleto, y esto explica por qué muchos programadores en Delphi 3 no ven la necesidad de pagar por una actualización cuando ya tienen lo que necesitan. La extensibilidad tiene también mucho que ver con esto.

Los programadores Visual Basic necesitan actualizarse demasiado rápido. Delphi es tan poderoso, funcional y extensible que sus versiones no se hacen obsoletas fácilmente: a la larga, Delphi es más barato.

Poder de programación y funcionalidad

Entre muchas otras cosas, con Delphi uno puede:

1. Escribir aplicaciones, bibliotecas, objetos ActiveX y componentes VCL de calidad comercial, rápidos y pequeños.
2. Crear reportes por código (o modificar existentes, por ejemplo para cambiar o agregar una columna)
3. En general, todo lo que se puede hacer visualmente (en tiempo de diseño) se puede hacer programáticamente (en tiempo de ejecución)
4. Escribir fácilmente aplicaciones multi-hilos
5. Ver los datos de las tablas y consultas en tiempo de diseño
6. Llamar a funciones de la API como si fueran funciones incorporadas
7. Introducir código en lenguaje ensamblador para mejor desempeño donde sea necesario
8. Aprovechar los montones de componentes VCL disponibles para darle a sus aplicaciones un "look and feel" distintivo y para ofrecer una funcionalidad mejorada.
9. Derivar fácilmente componentes VCL existentes para añadirles funcionalidad
10. Acceder nuevas tecnologías no presentes cuando salió el IDE (por ejemplo acceso a ADO con Delphi 4) sin tener que actualizarse



El IDE de Delphi ofrece a los programadores características avanzadas como la capacidad de depurar aplicaciones multi-hilos o de ejecutar código paso a paso en ensamblador!

Delphi ofrece a programadores un poder y una funcionalidad que deja a Visual Basic y sus

> limitaciones atrás.

Portabilidad

Si usted está interesado en Linux, hay un Delphi para Linux (Kylix), y no veo que habrá un Visual Basic para Linux...

> Kylix lleva Delphi a la plataforma Linux, adonde VB no irá.

Comparaciones de NTSL

La NTSL (National Software Testing Laboratories, Inc.) preparó dos informes que comparan Delphi 4 con Visual Basic 5 y Visual Basic 6. Estos informes se publican en el sitio web de Microsoft:

- Client/Server Development Tools - By NTSL, 1999
- Benchmark: Rapid Application Development Tools - By NTSL, 1999

Hay algunas cosas de estos informes sobre las que quisiera puntualizar:

1. Se compara Delphi 4, y no Delphi 5
2. El hardware usado para las pruebas oculta las ineficiencias de Visual Basic. Me pregunto por qué la prueba no incluyó varias configuraciones de hardware (¿qué tal la mitad o dos tercios de la velocidad de CPU y la mitad de la memoria?)
3. Hay muchas características que (deliberadamente) no se han evaluado. ¿Será porque Delphi posee muchas características que Visual Basic no y no quieren que se note que Delphi es una herramienta marcadamente superior?
4. La selección de las pruebas es demasiado arbitraria. Parece bastante como que estudiaron las debilidades de Delphi y seleccionaron las pocas pruebas donde Visual Basic tenía un funcionamiento comparable o mejor (tómele el tiempo a un bucle For de 1 a 1000000000 para ver a lo que me refiero). Es claro para mí que las pruebas son (deliberadamente) no abarcativas. ¿Será porque no quieren mostrar que Delphi es el claro ganador en muchas otras pruebas? http://www.mers.com/INPRISE/FAQ/2780.HTML
5. No compro que el algoritmo Sieve usado para encontrar números primos se ejecute más rápido en Visual Basic que en Delphi. Pienso que este podría ser el resultado de una pobre codificación en Delphi. "Curiosamente" no se revela el código utilizado en las pruebas. También curiosamente, recuerdo que el viejo Clarion Desktop Developer decía ser más rápido que C en esta prueba... ¡Sí claro!
6. Los evaluadores admitieron no tener un buen conocimiento de Delphi. ¿Significa esto que los benchmarks están comparando el código producido por un programador avezado en Visual Basic contra el código producido por un novato en Delphi?
7. Los evaluadores fueron contratados por Microsoft. ¡Por supuesto! ;)
8. Los informes se publican bajo copyright de Microsoft.

> Me parece que estos reportes no son independientes y que están demasiados influenciados como para ser considerados seriamente.

Soporte

El soporte de producto es usualmente un factor crítico a la hora de decidir qué herramienta usar. Microsoft tiene presencia física en casi todas las ciudades más importantes del mundo. He escuchado que si uno espera lo suficiente en el teléfono va siendo transferido de persona en persona hasta que llega a hablar con el el programador que escribió la función que le da problema. ¡Qué soporte! Pero quisiera hacer una pregunta: ¿qué clase de soporte es que lo tengan a uno esperando dos o cuatro horas al teléfono para que al final le digan que el error ha sido corregido en la última versión (o sea "pague el upgrade"), o que será corregido en la nueva versión (o sea "espere y pague el upgrade cuando esté disponible")? Cuando tengo un problema quiero una solución, y la quiero ya, y no quiero pagar el costo de una actualización. Una de las cosas que me gustan de Borland es que son muy atentos con estas cuestiones (por ejemplo si uno encuentra un error en un componente VCL, capaz que alguien ya halla desarrollado un parche o encontrado una forma de circunvenir el problema, o si uno necesita una funcionalidad añadida para un componente, si es una tarea común probablemente alguien haya desarrollado un componente sustituto, o sinó tal vez alguien en el grupo de noticias le pueda dar una pista o algo de código fuente para hacer lo que quiere). El soporte técnico es muy

importante para mí, y la arquitectura, la extensibilidad y la disponibilidad de los fuentes de la VCL me hacen sentir muy seguro, y me siento muy respaldado al ver que los técnicos de Borland y los miembros de la Comunidad que participan de los grupos de noticias respondiendo preguntas poseen un alto nivel de conocimiento y predisposición para contestar desde las preguntas más simples a las más escabrosas.

Borland es una empresa con experiencia y dedicación a las herramientas de desarrollo, no una división de una empresa de sistemas operativos. Sé qué esperar de Borland hacia el futuro, y sé que seguirá desarrollando y dando soporte a Delphi, pero la verdad es que no sé con qué cosas va a salir Microsoft en el futuro, si seguirán desarrollando Visual Basic o si un día decidirán dejar de soportar Visual Basic y dirán que quieren que los programadores Visual Basic migren a un Microsoft Universal Language o algo así...

> El soporte técnico es muy importante. No se deje guiar por la cercanía física de una oficina y pregunte a sus colegas sobre su experiencia con el soporte de uno u otro proveedor... ¡y cuánto le costó!

Enlaces a más comparaciones, testimonios, etc.

- Thirteen ways to loathe VB - por Verity Stob, 2000
- Why I really like Delphi - por Null Writer, 1999
- Saying Goodbye to Hardcore Visual Basic - por Bruce McKinney, 1999
- The End of Hardcore Visual Basic - por Bruce McKinney, 1999
- Hardcore Visual Basic - por Bruce McKinney, 1999
- The Wrong Tool for the Right Job - por Don Kiely, 1999
- The Case for Delphi - por Alan C. Moore, Ph.D., 1999
- The cult of Delphi - por Marco Cantù, 1999
- Delphi. Reasons why. - por Greg Lorriman, 1999
- Client/Server Development Tools - por NTSL, 1999
- Benchmark: Rapid Application Development Tools - por NTSL, 1999
- Free Value - por Peter Jackson - Published in PC Magazine UK, 1999
- Delphi 4 and VB6 take aim - por Peter Coffee, 1998
- Delphi vs Visual Basic - por MER Systems Inc., 1998
- RAD Programming Tools Shootout - por Les Kendall, 1997
- Delphi 3 closes in on Visual Basic 5.0 - por Maggie Biggs, 1997
- Visual Basic and Delphi Head to Head - por Stephen W. Plain, 1997
- Comparing OOP Languages: Java, C++, Object Pascal - por M. Cantù, 1997
- Should Delphi people be worried about C++ Builder? - por David Lipschitz, 1997
- Delphi Leaves Visual Basic In the Dust...Again - por Paul Bonner, 1997
- Battle of the Visual Masters - por David S. Linthicum, 1996
- A Comparison of Client/Server Development Tools; Powerbuilder 5.0 vs. Delphi 2.0 - por Michael Lant, 1996
- Delphi vs Visual Basic - por Borland Staff, 1995
- Why Shun Delphi? - por Graham Perkins
- Why not VB.Net?
    - VB.Net for the VB6 Developer - por Bob Butler, 2001
    - Moving to VB.Net - por Bob Butler, 2001
    - Developers cry foul over new Microsoft language - por Mary Jo Foley, 2001
    - .NET Changes Can't Be Ignored - 2001
    - The New Visual Basic - por Mike James, 2000
    - Get Your Designs in Gear for VB.NET - por Paul R. Reed Jr., 2000
    - Visual Basic.NET Revealed - por Bruce McKinney, 2000
- Why Delphi?
    - Why Borland's Delphi? - por varios autores
    - Why Delphi ? - por W3 Data ApS
    - Why Delphi ? Why not ?!!?? - por Matlus
    - Ten Things I Love About Delphi - por Charles Calvert, 2000
    - Stand up and be counted - por Marco Cantù, 1999
- Case studies

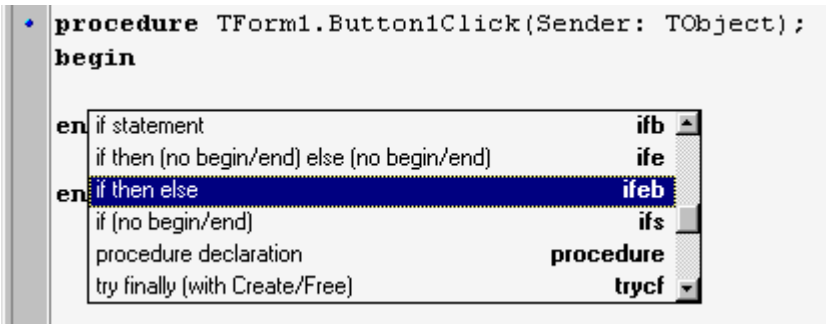
- ASSESS: Delphi Helps American Skandia Compete - por Denis Perrotti, 2000
- California's CalJOBS Project - por The EDD Development Team, 1997
- Inquire Within - Engaging Kiosk System Reels in Apartment Shoppers - por David Rippy, 1996
- Microsoft Visual Basic and Delphi - The Decision Making of Systems of Emergency Management - por Binh Lam, 1996

Conclusiones

Visual Basic es adecuado para simples aplicaciones de interfaz de usuario, típicamente aplicaciones de gestión (facturación, inventarios, nóminas, etc.). Su facilidad de uso lo hace la opción correcta para los programadores principiantes. Delphi es aún mejor para aplicaciones de interfaz de usuario pues la disponibilidad de componentes VCL permite que uno desarrolle interfaces de calidad superior tanto en términos de funcionalidad como de presentación, pero no es tan fácil de utilizar para los principiantes, que son el grupo mayoritario y para quienes prevalece la facilidad sobre la calidad del trabajo, velocidad de ejecución, tamaño del código, uso de los recursos del sistema, tiempo de desarrollo, costos o cualquier otro factor.

Las corporaciones prefieren tener muchos candidatos disponibles cuando necesitan cubrir un puesto de trabajo, que tener pocos candidatos para cubrir una posición que requiera más calificaciones, y por esta razón prefieren Basic Visual sobre Delphi. Si usted desea conseguir un trabajo, aprenda Visual Basic.

Pienso que solamente es posible convencer a un ejecutivo para que elija Delphi en vez de Visual Basic cuando hay requisitos específicos que lo justifican, por ejemplo si la compañía no puede tener el último hardware, no pueden comprar las actualizaciones a la última versión de Visual Basic, no puede permitirse pagar soluciones cada vez que se topan con una limitación en Visual Basic, necesitan una GUI (Interfaz de Usuario Gráfica) de mejor calidad (más funcional y visualmente más atractiva) o cuando el personal de Visual Basic no puede entregar las aplicaciones en término (esto sucede normalmente cuando se están tropezando con alguna limitación, puesto que usualmente se complica el código para circunvenirla y entonces se hace difícil de mantener) o cuando se requieren menores tiempos de desarrollo para aplicaciones que no son tan simples.



El editor de código de Delphi tiene algunas características de productividad, como las plantillas de código —*Code Templates*— definibles por el usuario que le ahorran escribir bloques comunes de código, y el explorador de código —*Code Explorer*— que le ayudan a examinar todos los elementos (clases, métodos, variables, procedimientos, funciones, etc.) definidos en una unidad.

Delphi es más difícil de usar que Visual Basic al principio, no hay duda de ello, pero permite hacerlo todo. Como alguien dijo, "Delphi es más compatible con Windows que Visual Basic". Aquellos programadores que cargan en su equipaje algún conocimiento de un lenguaje orientado a objetos (como C++, Turbo Pascal 5.5+ o FreePascal) no tienen problemas para aprender Delphi. Son habitualmente aquellos que han tenido una base de educación formal en programación (universidades, politécnicos, etc.) o aprendieron por sí mismos. El mercado de Delphi no está limitado a las aplicaciones de interfaz de usuario: Delphi se usa también para el back-end (servidores, bibliotecas, objetos ActiveX, etc.), un mercado fuera del alcance de Visual Basic.

Los paquetes de componentes VCL de terceros ayudan a los desarrolladores a darle a sus aplicaciones un toque de distinción.

Visual Basic hace muchas cosas por los programadores, así que muchos que lo usan están muy cómodos con él (probablemente Delphi les parezca demasiado arcano), y si no se han encontrado con las limitaciones de Visual Basic, ¿por qué cambiar? Visual Basic es la herramienta correcta para ellos. Por otro lado, veo muchos que están descontentos, y pienso que estarían mucho mejor programando en Delphi, así como también veo algunos programadores en Delphi (principiantes con mala, poca o ninguna preparación formal o informal en programación) que tal vez deberían estar programando en Visual Basic. Delphi es una herramienta profesional para programadores profesionales, mientras Visual Basic es para aquellos que se conforman con lo poco que la herramienta les brinda ya sea porque no necesitan más, no tienen suficiente preparación o experiencia en la programación en Windows o en programación orientada a objetos, o porque no demandan demasiado de la herramienta de programación, o no quieren o no les gusta programar, o no tienen tiempo para aprender, o no saben inglés, o porque con Visual Basic es más fácil conseguir un trabajo porque las empresas prefieren contratar gente que llene ese perfil. A pesar de ello, debe decirse que Visual Basic es una gran herramienta y que ha puesto la programación al alcance de un montón de profesionales que vienen de carreras fuera de la programación (Analistas de Sistemas de Información, Ingenieros en Electrónica, etc.).

> La mejor herramienta para el trabajo es la que mejor se adapta a sus requerimientos, limitaciones y expectativas.

Este artículo fué presentado en el Boletín para Desarrolladores: