```
Sub CATMAIN()
    On Error Resume Next

    CATIA = GetObject(, "CATIA.Application")

    If Err.Number <> 0 Then
        MsgBox("CATIA has been Launched. Please Wait.")
        CATIA = CreateObject("CATIA.Application")
        CATIA.Visible = True
    Else
        MsgBox("Connection with CATIA has been established successfully.")
    End If
End Sub
```

Programación Visual Basic

=> CATIA V5.GENERAL

Capítulo 1. Programación Orientada a Objetos. Conceptos Fundamentales

CATIA MACROS

info@catiamacros.com

Enero 2021

Índice

Índice Figuras

Control de Cambios

Versión	Motivos del cambio de versión	Páginas afectadas
V.01.21.1	Lanzamiento Programación Visual Basic en Catia v5. General	1-321

Derechos de autor

Prólogo

Sobre el autor

El autor del presente manual es un ingeniero aeronáutico con más de 15 años de experiencia con Catia V5 y otros programas como NX, Autodesk, etc.

Su primer contacto con el programa Catia V5 comenzó en los últimos años de la universidad donde desde el primer minuto se encontró en una perfecta sinergia con el software, es por ello que empezó a tomar cursos de duración de hasta 100 horas antes de terminar su titulación.

Tras estos cursos y lo aprendido a lo largo de los años de carrera, se dio cuenta que su pasión sería el Diseño Aeronáutico por lo que se enfocaría en esta disciplina.

Su carrera se ha centrado en el diseño de Aeronaves tanto Civiles como Militares en la empresa Airbus cuyos estándares de calidad, precisión, metodología etc. en relación al diseño es de los más altos de las ingenierías mecánicas.

Entre los proyectos para los que ha trabajado se pueden encontrar A320, A330, A330-MRTT, A350, A380, A400M y Airbus Beluga XL.

En su trabajo en Airbus, trabaja con ***macros*** a diario de todo tipo para agilizar y acelerar todo el proceso de diseño desde la fase conceptual hasta la elaboración de la documentación necesaria para la fabricación.

De forma paralela a su profesión de Diseño Aeronáutico, desarrolla su otra pasión que es la formación de Catia V5 con cursos en empresas líder en ingeniería de los principales ***módulos*** de Catia V5 (Part Design, Shape Design, Assembly Design, Drafting, etc.) así como más específicos como Composite Design, Electrical Harness Design, Functional Tolerancing And Anotation. A lo

largo de 8 años desarrolla más de 2000 horas de formación presencial desarrollando cursos de forma autodidacta para páginas web líderes en la formación online.

Al autor se le une personal, que sin su ayuda no habría sido posible este curso, de ámbitos tan variados como Ingenieros Informáticos en tareas de corrección y validación de códigos, Ingenieros Mecánicos realizando revisión de conceptos fundamentales de Ingeniería, Licenciados en Marketing responsables de Maquetación de documentos, Logos y Diseño Web.

Garantía de aprendizaje

¿Por qué el aprendizaje de programación en Catia V5 está garantizado sin tener ningún conocimiento previo de programación?

El autor del presente manual, se dio cuenta de que, tras conseguir un entendimiento y un uso avanzado del software, el siguiente paso para dominar por completo Catia V5, era desarrollar sus propios comandos mediante la creación de ***macros*** para agilizar su trabajo diario como diseñador en empresas como Airbus Defence And Space.

Por este motivo, decide aprender de forma autodidáctica, sin conocimientos previos en programación informática alguno, cómo aprender el lenguaje de programación ***Visual Basic*** y cómo aplicarlo en Catia V5. Es por esto que dedica innumerables horas, debido a la literatura existente incompleta, corta y confusa, a la investigación y aprendizaje del lenguaje de una forma generalizada y una vez aprendidos los conceptos fundamentales y la naturaleza del lenguaje, los traslada para aprender desde cero la programación en Catia hasta dominarla por completo, desde un punto de vista Global.

Para conseguir este objetivo desarrolla un manual completamente desde cero con el fin de afianzar sus conceptos y tenerlo a modo de consulta explicando con sus propias palabras los conceptos clave, el proceso de programación,

técnicas, metodologías etc. El presente curso es una ampliación de ese manual enfocado al alumno que viene sin conocimientos previos, poniéndose en la piel de él y guiarle hasta conseguir los objetivos del curso.

Es por este proceso de ponerse en la piel del alumno que el aprendizaje está garantizado ya que al haber aprendido el autor desde cero, el alumno sólo tendrá que seguir los pasos que tuvo que realizar el autor para aprender de forma autodidacta pero con la gran ventaja de que todo está ya desarrollado y pensado para que el alumno no tenga que gastar tiempo buscando información de varias fuentes, a veces contradictoria, y ponerlo todo junto para conseguir un conocimiento de base suficientemente robusto para dominar el arte de la programación.

Proceso de aprendizaje

Como ya se ha podido imaginar el alumno, el primer paso es aprender el lenguaje de programación, es decir, aprender la sintaxis del código, las estructuras de los ***procedimientos***, declaración de variables, crear ***bucles***, gestionar ***Formularios***, etc.

Desde la experiencia previa del autor, intentar aprender todo esto junto a la programación intrínseca de Catia V5 sin experiencia previa en programación es algo realmente complejo. Por este motivo, se va a explicar la programación en lenguaje ***Visual Basic*** a través de una ***Aplicación*** más sencilla y con la que se interactuará en el futuro con Catia V5 para obtener un input de datos.

Esta ***Aplicación*** no es otra que Excel. Mediante Excel se van a poder explicar los conceptos fundamentales como qué es un ***objeto***, un ***método***, una ***propiedad*** etc. a través de ejemplos muy sencillos y de fácil entendimiento que no sería posible en Catia V5. Hay que decir que la programación en Excel es un campo inmenso y daría hasta para un Máster, sin embargo, todo lo que se va a aprender en este manual, es lo imprescindible para hacer la transición con

confianza a la programación de Catia V5 y no sobra ningún conocimiento para cuando se haga la transición entre Excel y Catia V5.

Se recomienda tener un conocimiento medio-avanzado de Catia V5 que permita conocer la naturaleza de Catia V5 en los ***módulos*** MD2, cómo funcionan las operaciones, las relaciones Parents And Children, cómo organizar de forma eficiente un árbol a través de Geometrical Sets. Aun así, cuando se haga la transición a Catia, se incidirá en los posibles puntos que pueden traer algún problema al alumno que no tenga un amplio bagaje con el software.

Una vez adquiridos los conocimientos en Excel, se realizará la transición al lenguaje CATScript que se basa en un 99% en el recién aprendido, comentando las pequeñas diferencias al comienzo de su respectivo manual con un capítulo muy similar al primero de éste.

Cómo utilizar este manual y cómo se estructura. Vídeos.

Aprender un lenguaje de programación es algo complejo por su naturaleza. Es casi imposible, al principio sobre todo, focalizar en la explicación de un concepto sin tener que recurrir a conceptos más avanzados, es decir, se deberían aprender de forma global los conocimientos teóricos antes de empezar con ejemplos prácticos. Desafortunadamente aprender primero todos los conocimientos teóricos puede desanimar al alumno y hacer que abandone nada más empezar. Es por esto que se va a implantar una forma de aprendizaje diferente para evitar este hecho.

El presente manual va a estar formado por capítulos distribuidos según temas bien diferenciados entre sí. Estos capítulos van a estar formados por una amplia explicación teórica de los conceptos incluidos en la unidad situándolo en el contexto global del manual y haciendo unas breves referencias de los conceptos avanzados necesarios para su comprensión. Normalmente se hará uso de una definición técnica y a veces complicada y a continuación una

explicación más para usar en el día a día. Para poder explicar estos conceptos teóricos, se hará uso, en caso de ser necesario, también de pequeños ejemplos.

A medida que se vaya avanzando en los capítulos, el alumno tendrá suficientes conocimientos teóricos como para poder hacer ejercicios prácticos que engloben lo aprendido hasta entonces. Los ejercicios prácticos serán explicados paso a paso, explicando cada línea de código necesaria para entender el desarrollo del ejercicio.

Como pilar fundamental de este curso y de gran valor añadido, tanto los capítulos teóricos como los ejercicios prácticos van a ser apoyados por vídeos narrados en español de cada concepto, explicando paso a paso y línea a línea de código todo lo explicado de manera escrita. Dicho de otro modo, el presente manual se podría definir como una transcripción de los vídeos explicativos de cada concepto explicado en él, así como un vídeo de cómo se realizan los ejercicios tal y como un amigo o compañero de trabajo nos lo explicaría de forma clara y concisa.

Introducción a Visual Basic Script

Bienvenido a este curso de programación en ***Visual Basic*** con el que podrás revolucionar por completo tu conocimiento y uso de Catia V5 mediante la programación de ***macros*** con el cual podrás automatizar procesos e incluso tus propios comandos.

Como se ha dicho en el Prólogo, el primer paso para conseguir este fin tan deseado, es aprender el lenguaje de ***Visual Basic*** y esto se va a conseguir través del programa Excel.

Contenido del curso de Programación en Visual Basic General

El contenido del curso de ***Programación*** en ***Visual Basic*** va a estar formado por un número de capítulos cuya temática de cada uno va a estar diferenciada entre sí, pero fuertemente relacionados por lo que se hará uso del ***Aprendizaje Global***, es decir, cuando se explique un capítulo, se pondrá en contexto de todo el lenguaje.

En primer lugar, se van a desarrollar una serie de capítulos teóricos, aunque con ejemplos clarificadores, sobre la ***Programación Orientada a objetos*** y fundamentos teóricos del lenguaje.

En segundo lugar, se van a enseñar distintas estructuras de código predefinidas del lenguaje como puede ser una ***matriz***, mensajes de introducción y salidas de datos, ***Condicionales***, ***bucles***.

Ya, por último, se tratarán conceptos más avanzados como los tipos de ***procedimientos***, llamadas entre ellos, ***Formularios***, etc. para poder crear códigos realmente de utilidad y que son necesarios para el Curso Específico de CATIA.

A continuación, se muestran los capítulos de los que va a estar formado este primer Curso General.

Capítulo 1. Programación orientada a Objetos. Conceptos fundamentales

Introducción

Este quizás sea el capítulo más teórico y más difícil de asimilar por parte de alguien inexperto en programación. Sin embargo, es de vital importancia su estudio para poder continuar. Para realizar el aprendizaje lo más ameno posible, se van a introducir ejemplos de la vida real para que no quede ninguna duda en el alumno.

Como se ha dicho con anterioridad en varias ocasiones, la mayoría de los lenguajes de programación de hoy día van a estar orientados a ***objetos*** o por sus siglas en ingles ***OOP*** (***ORIENTED OBJECT PROGRAMMING***). El concepto de ***objeto*** puede resultar algo confuso al principio al alumno que nunca haya visto nada de programación con anterioridad y que se va a desgranar en el siguiente apartado en mucho más detalle. Es fundamental y no se recomienda seguir avanzando en el curso hasta que, al menos, se tenga una idea más o menos consolidada de qué es un ***objeto*** y los conceptos asociados a él.

Este paradigma de programación es usado por lenguajes simples como ***Visual Basic*** hasta más avanzados como ***C++***, que como se verá en el manual específico de programación en CATIA, también es usado por programadores para configurar el software de diseño.

Conceptos Fundamentales

A continuación, se van a explicar los conceptos fundamentales sin los que la ***Programación Orientada a objetos*** carecería de sentido. Esta explicación se va a realizar con ejemplos de la vida cotidiana para asimilar su concepto, para más tarde, en el Capítulo 4, aplicarlo en ***Visual Basic For Applications*** para aprender su uso.

Objeto

Se podría decir que es el concepto fundamental del lenguaje de programación actual ya que está orientado a ***objetos***, es decir, el código va a estar centrado en ellos.

Un ***objeto*** es una entidad existente, ya sea física o virtual, cuya naturaleza es poder llevar a cabo unas determinadas funciones o actividades y que va a estar definido por unas características. Esta definición así tan teórica puede descolocar y desanimar un poco al alumno. Sin embargo, la forma más fácil de entender este concepto es entender que el concepto de ***objeto*** se creó para hacer una semejanza entre el mundo real y el mundo informático para que la programación sea más intuitiva con el que mundo que nos rodea. Entonces ¿Por qué el concepto es tan difícil de entender de primeras? Esto es porque necesitamos crear un vínculo entre ambos mundos y la mejor manera es a través de un ejemplo.

Un ejemplo de ***objeto*** en la vida real sería un coche determinado, importante lo de determinado, que tiene características o ***propiedades*** que pueden ser color, longitud, altura etc. y que va a realizar funciones o ***métodos*** como acelerar, frenar, girar.

Con esto ya se puede deducir que un ***OOP*** va a ejecutar el código de una ***macro*** usando estos ***objetos*** que, teniendo unas determinadas ***propiedades***, va a llevar a cabo unas acciones a través de unos ***métodos***. Otro ejemplo de ***objeto*** podría ser una persona de 1,80 m, sexo femenino, de pelo rubio y esta persona puede tocar el violín, correr, conducir, etc.

Clase

Cada ***objeto*** de ***Visual Basic*** está integrado en una ***clase***. Una ***clase*** es una plantilla que enumera las ***propiedades*** y los ***métodos*** que puede tener un ***objeto***, pero no especifica cuáles de ellas tiene, ni su valor. Pasamos de ***clase*** a ***objeto*** especificando de manera única, qué ***propiedades*** y ***métodos***, de entre todas las que puede tener debido a la ***clase*** que sea, va a tener. Los

objetos son instancias de una ***clase*** específica; puede crear tantos ***objetos*** como sean necesarios una vez que haya definido una ***clase***.

Esto trasladado al mundo real, es que tenemos una ***clase*** que es Coche y un coche concreto es un ***objeto*** de la instancia de la ***clase*** coche único al poder realizar unas funciones únicas y tener unas ***propiedades*** de un determinado valor. Así cada ***instancia*** del ***objeto*** coche, tendrá sus propios ***métodos*** y ***propiedades*** para crear entidades únicas.

En ocasiones, en el presente manual y para no alargar el texto, se omite el concepto de ***clase*** y se pasa directamente a decir que un cierto elemento es un ***objeto*** de un determinado tipo.

Colección

Definido ya un tipo de ***clase***, se puede crear un grupo de instancias de esta ***clase*** a la que llamaremos ***colección*** y que deberá ser creada por el usuario mientras que la ***clase*** está ya definida en la ***Aplicación***. Se puede tener tres o cuatro instancias distintas entre sí que forman parte todos de la ***clase*** Coche.

Jerarquía

Las ***clases*** en un lenguaje de programación están estructuradas por ***jerarquía*** u organigrama por lo que se tendrán ***clases*** de primer orden muy generales como, por ejemplo, la ***Aplicación*** que se esté utilizando, (Excel y Catia) y conforme vayamos aumentando de orden conseguiremos ***clases*** más específicas como puede ser una celda en Excel o un Pad en Catia.

En el mundo real, se tendrá que el ***objeto*** Volante, deriva del ***objeto***, Salpicadero y así sucesivamente yendo hacia arriba hasta llegar al primer orden que sería el ***objeto*** Coche.

Ruta

Un concepto muy ligado a la ***jerarquía*** es la ruta. Para tener acceso o hacer uso de un ***objeto***, se debe realizar el camino hacia arriba y escribir todas las ***clases*** que tenga por encima en la ***jerarquía*** hasta llegar al primer orden. En el

ejemplo anterior, para poder declarar el ***objeto*** Volante, se debe hacer necesario declarar, el ***objeto*** Salpicadero, antes el ***objeto*** Chasis y así sucesivamente hasta el ***objeto*** Coche.

Sin embargo, muchas ***clases*** de la jerarquía pueden ser omitidas del código si éstas son las activas, esto es, que ***Visual Basic*** ya sobreentiende o sabe que, o bien en la altura del código que se está en ese momento programando, o porque se ha activado o usado un ***objeto*** de una ***clase*** superior en la ***Aplicación***, ya hemos bajado en la ***jerarquía*** o dicho de otro modo aumentado el orden. Esto se explicará más en profundidad en el Capítulo 4.

Por esto, se puede hablar de ***clase*** raíz que sería la ***clase*** de primer nivel y luego ***clases*** base que derivan de ***clase*** raíz. En el caso del volante, la manera de acceder a él sería Coche.Chasis.Salpicadero.Volante.

Orden

Una ***orden*** está formada por varias palabras en una misma línea de código, por ejemplo, Coche gira a la derecha a 90º a 50 km/h. Adelantando al Capítulo 4, un poco, estas palabras pueden estar separadas entre sí por puntos (por ejemplo, para indicar ***jerarquía***, explicado un poco más adelante), por espacios (por ejemplo, para indicar una ***variable***, un texto, etc.) por paréntesis o corchetes (normalmente para indicar un input necesario para un ***método***).

Instrucción

Es una ***función*** predefinida en ***VBA*** ya sea de la ***Aplicación*** que se esté utilizando o a través de las librerías externas y que tiene una sintaxis ya definida. Un ejemplo de instrucción puede ser ***MsgBox*** que tras su ejecución mostrará una ventana con un texto. En este ejemplo se ha puesto para que aparezca el texto habitual como primer ejemplo, “Hola Mundo”. En los capítulos siguientes, esta y otras instrucciones típicas serán explicadas.

Figura 1: Editor VBA

Variable

Se dice de una entidad capaz de albergar información siendo ésta modificable a medida que se desarrolla el código ya sea de forma automática o manual. Las variables pueden ser de muchos tipos: longitud, ángulo, número entero, número ***Decimal***, texto, ***objeto*** etc. Por ejemplo, podemos definir la ***variable*** LongitudCoche que puede tomar cualquier valor dentro de su propio tipo de ***variable*** que habrá que definir antes de darle un valor en el código.

Valor

Es cuánto va a valer la ***variable*** que como su propio nombre indica, puede variar a lo largo de la ejecución del programa. En el ejemplo del coche se puede decir LongitudCoche= 5000. No se especifica la unidad de medida en la inicialización de una ***variable***, es decir, darle valor.

Procedimientos

Un ***procedimiento*** es el conjunto de líneas de código o instrucciones que dan lugar a la ejecución de la ***macro***. En él se declaran los ***objetos***, las ***variables*** se les da valor y mediante las ***propiedades***, ***métodos*** de los ***objetos*** se va ejecutando el programa hasta alcanzar las ordenes deseadas. Pueden ser ***Sub*** cuya finalidad es ejecutar tareas o ***Function*** cuyo fin es devolver un valor de una ***variable*** o parámetro y ser usado por ***procedimientos Sub***.

Parámetro

Si la ***variable*** es definida a la hora de crear el ***procedimiento***, ya sea ***Sub*** o ***Function***, toma el nombre de ***parámetro*** y sin cuya declaración del valor, el ***procedimiento*** no puede funcionar.

Evento

Es una acción realizada por el usuario que desencadena la ejecución de la ***macro*** asociada a dicho ***evento***. Un ejemplo de ***evento*** puede ser desde hacer click en un botón, abrir o cerrar la ***Aplicación*** etc. por lo que tras estas acciones se ejecutará la ***macro*** que hayamos escrito.

Argumento

Un ***argumento*** es un input o dato necesario para que un ***procedimiento***, función o ***método*** haga su cometido. Puede haber ***argumentos*** tanto obligatorios, que serán los primeros en introducirse y los opcionales que irán después de los obligatorios.

Módulo

Un ***módulo*** es simplemente una entidad que se va a utilizar para organizar el código dentro del ***proyecto VBA*** en el que se esté trabajando. En él, irán los ***procedimientos*** que contendrán los códigos de la ***macro***. Se pueden crear tantos ***módulos*** como se necesite y tantos ***procedimientos*** dentro de ellos como se quiera. Sería como una carpeta con archivos dentro, pero en lugar de archivos, son códigos que se suelen agrupar por funcionalidad, aunque esto ya es decisión del programador.

Características de la OOP

Aunque no es imprescindible saberlas, sólo como referencia, los lenguajes de programación orientada a ***objetos*** tienen en común unas características algo abstractas:

Abstracción

Define características específicas de un ***objeto***, donde se capturan sus comportamientos y que lo distingue de los demás tipos de ***objetos*** y que logran definir los límites conceptuales respecto a quién está haciendo dicha abstracción. Cada ***objeto*** en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros ***objetos*** en el sistema sin revelar "cómo" se implementan estas características. Los procesos, las funciones o los ***métodos***

pueden también ser abstraídos, y, cuando lo están, existe una variedad de técnicas que son requeridas para ampliar una abstracción. El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real. La abstracción es clave en el proceso de análisis y diseño orientado a ***objetos***, ya que mediante ella podemos llegar a armar un conjunto de ***clases*** que permitan modelar la realidad o el problema que se quiere atacar.

Encapsulamiento

Con el encapsulamiento podemos definir el acceso de las entidades dentro del código de programación. De una manera más clara, cuando se crea un ***procedimiento***, este puede ser aislado del resto de ***procedimientos***, de manera que su código no puede ser llamado por otros. En ***VBA***, los ***procedimientos*** pueden ser ***Public*** o ***Private***.

Herencia

Las ***clases*** no se encuentran aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los ***objetos*** heredan las ***propiedades*** y el comportamiento de todas las ***clases*** a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento, permitiendo a los ***objetos*** ser definidos y creados como tipos especializados de ***objetos*** preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Esto suele hacerse habitualmente agrupando los ***objetos*** en ***clases***, y estas en árboles o enrejados que reflejan un comportamiento común.

Polimorfismo.

Comportamientos diferentes, asociados a ***objetos*** distintos, pueden compartir el mismo nombre; al llamarlos por ese nombre se utilizará el comportamiento correspondiente al ***objeto*** que se esté usando. O, dicho de otro modo, las referencias y las colecciones de ***objetos*** pueden contener ***objetos*** de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del ***objeto*** referenciado.

Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama asignación tardía o asignación dinámica. Algunos lenguajes proporcionan medios más estáticos (en "***tiempo de compilación***") de ***polimorfismo***, tales como las plantillas y la sobrecarga de operadores de ***C++.***

Modularidad

Se denomina ***"modularidad"*** a la ***propiedad*** que permite subdividir una ***Aplicación*** en partes más pequeñas (llamadas ***módulos***), cada una de las cuales debe ser tan independiente como sea posible de la ***Aplicación*** en sí y de las restantes partes. Estos ***módulos*** se pueden compilar por separado, pero tienen conexiones con otros ***módulos***. Al igual que la encapsulación, los lenguajes soportan la ***modularidad*** de diversas formas.

Principio de ocultación.

Cada ***objeto*** está aislado del exterior, es un ***módulo*** natural, y cada tipo de ***objeto*** expone una "interfaz" a otros ***objetos*** que especifica cómo pueden interactuar con los ***objetos*** de la ***clase***. El aislamiento protege a las ***propiedades*** de un ***objeto*** contra su modificación por quien no tenga derecho a acceder a ellas; solamente los propios ***métodos*** internos del ***objeto*** pueden acceder a su estado. Esto asegura que otros ***objetos*** no puedan cambiar el estado interno de un ***objeto*** de manera inesperada, eliminando efectos secundarios e interacciones inesperadas. Algunos lenguajes relajan esto, permitiendo un acceso directo a los datos internos del ***objeto*** de una manera controlada y limitando el grado de abstracción. La ***Aplicación*** entera se reduce a un agregado o rompecabezas de ***objetos***.

Recolección de basura.

La recolección de basura (*garbage collection*) es la técnica por la cual el entorno de ***objetos*** se encarga de destruir automáticamente, y por tanto desvincular la memoria asociada, los ***objetos*** que hayan quedado sin ninguna referencia a ellos. Esto significa que el programador no debe preocuparse por la asignación o liberación de memoria, ya que el entorno la asignará al crear un nuevo ***objeto*** y la liberará cuando nadie lo esté usando. En la mayoría de los lenguajes híbridos que se extendieron para soportar el Paradigma de

Programación Orientada a objetos como ***C++*** u ***Object Pascal***, esta característica no existe y la memoria debe desasignarse expresamente.

COM

Por sus siglas en inglés (***Component Object Model***), se denomina la interfaz creada por Microsoft para realizar la comunicación entre distintas Aplicaciones a través del código de una ***macro***. Por ejemplo, de ***VBA***, donde se crea el código a Excel o CATIA.

Controles ActiveX

Son los controles con los que el usuario final se puede comunicar con la ***macro*** para que se ejecute de una determinada manera o realice unas ejecuciones determinadas. Entre estos controles, se encuentran, botones, desplegables, listas, checks, etc.

OLE

Por sus siglas en inglés (***Object Linking And Embedding***) o lo que es lo mismo, la capacidad para editar, crear o eliminar ***objetos*** propios de una determinada ***Aplicación***, desde los controles creados por otra. Por ejemplo, modificar un tornillo introduciendo datos en un Excel e iniciar una ***macro*** que lo cree a través del click en un botón ***ActiveX***.

Estos conceptos, se entenderán mejor a medida que el alumno vaya avanzando en el curso. Esto es lo que se ha denominado aprendizaje Global, por lo que se recomienda repasar los conceptos si no han quedado del todo claros.