UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE INGENIERÍA CIVIL

“DESARROLLO DE MÓDULOS EN ENTORNO DE VISUAL BASIC PARA CALCULAR LA POBLACIÓN PROYECTO EN SISTEMAS DE ABASTECIMIENTO DE AGUA POTABLE”

TESIS

PARA OBTENER EL TÍTULO DE:

INGENIERO CIVIL

PRESENTA:

JOSÉ LUIS LÓPEZ FLORES

ASESOR:

DR. CONSTANTINO DOMÍNGUEZ SÁNCHEZ

MORELIA MICHOACÁN, NOVIEMBRE DE 2011

Contenido.

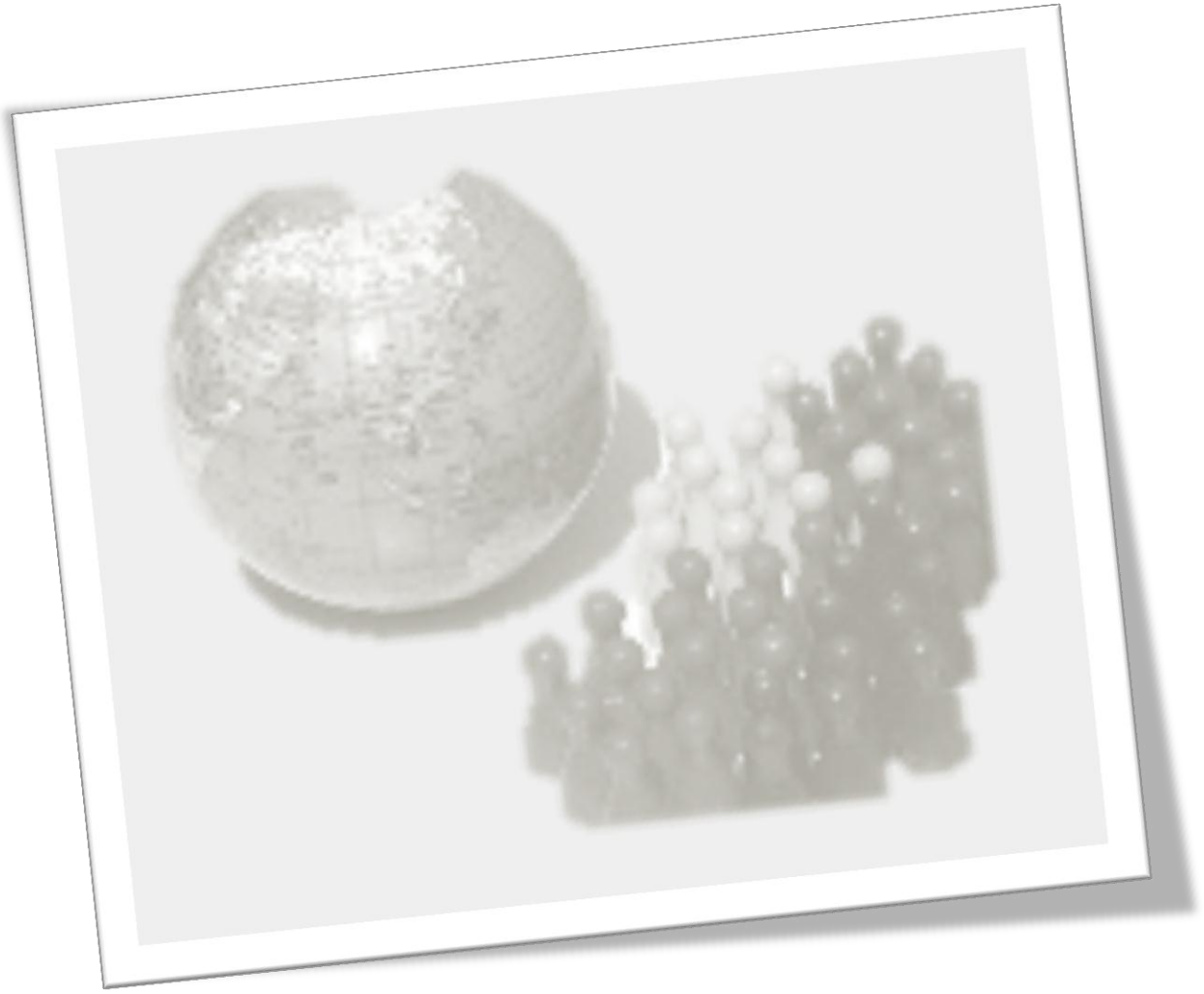
Agradecimientos.

Gracias, a la vida por darme la oportunidad de cumplir con uno de mis sueños mas anhelados; el ser ingeniero civil, gracias a mis padres y mi familia por todo su apoyo incondicional en momentos difíciles y de alegría, les estaré infinitamente agradecido.

A mi asesor de tesis el Dr. Constantino Domínguez por su apoyo para la realización de esta tesis y a todos mis maestros que contribuyeron con mi formación como ingeniero, a ellos que inculcaron en mi persona el valor de la responsabilidad y la vocación de esta profesión, a todos ellos gracias; llevare siempre en mi mente sus palabras y consejos y con ello ejercer la profesión con todo respeto y apego a los valores sociales.

De igual manera agradezco a mis compañeros que a todo lo largo de la carrera compartieron conmigo sus dudas y conocimientos y que luchamos juntos por lograr salir adelante en tiempos difíciles de exámenes, trabajos, etc.

1. Introducción.



En la actualidad el uso de herramientas informáticas para la solución de los problemas cotidianos en la vida del hombre ya no es una opción, sino una necesidad, es por ello que cada vez se pueden observar en la red cada vez mas programas para resolver este tipo de problemas, sobre todo en el ámbito ingenieril que es en donde involucra el programa producto de esta tesis.

Existen en internet muchos programas para resolver la mayoría de las necesidades de la ingeniería civil, existen de uso libre y otras que requieren de licencias, es decir tienen un costo para el usuario. El uso de todos estos programas es con la intención de ahorrar tiempo, dinero y esfuerzo, optimizando así el rendimiento y reduciendo errores al momento de realizar los diversos cálculos.

En esta tesis se busca diseñar un software que permita calcular la población de proyecto para obras de abastecimiento de agua potable y otras mas que requieran de dicho análisis como obras de alcantarillado, construcción de calles y avenidas, carreteras, etc. que involucren un determinado aumento de la población a servir.

Así mismo se busca contribuir con los demás programas existentes para el cálculo de las redes de abastecimiento, siendo el programa a diseñar un complemento a todos ellos.

El software que se busca desarrollar comprende la programación de las ecuaciones que permiten calcular la población de proyecto mediante una serie de métodos que se puedan adaptar a cada uno de los diferentes comportamientos de la población.

Se busca que el programa sea de fácil uso y comprensión para cualquier usuario, así mismo se busca que sea desarrollado para ser usado en la plataforma de Windows, que es el sistema operativo mas usado en nuestros alrededores para el caso de la ingeniería y otras profesiones.

Cave mencionar que ya existen programas para el cálculo o determinación de la población para el futuro, en Estados Unidos la secretaria de salud cuenta con un software de uso libre para determinar la población mediante una serie de datos como son índice de mortalidad, natalidad, migración, estado de salud, numero de mujeres vs numero de hombres, etc.

En este software se busca emplear los datos estadísticos de población proporcionados por el INEGI (Instituto Nacional de Estadística Geográfica e Informática) y con ello realizar proyecciones a años futuros que en realidad serán la vida útil de la obra de abastecimiento o de cualquier otro tipo.

Se realizara un ejecutable para su distribución y uso de los usuarios así mismo se elaborara un manual de usuario en el que los usuarios podrán consultar las dudas que se presenten durante el uso del software.

2. Antecedentes.

2.1. La situación en las actuales obras de abastecimiento.

"Antes de formular un proyecto de abastecimiento de agua, ha de decidirse el periodo de tiempo que las instalaciones servirán a la comunidad, antes de que deban abandonarse o ampliarse por resultar ya inadecuadas" (McGhee, 1981).

A dichos periodos de tiempo se les llama periodo de vida, y son de suma importancia al decidir los fondos que deben ser invertidos en las instalaciones, ya que entre mas grande sea el periodo de vida de un proyecto, mas fondos se deben invertir. Puesto que las ciudades constantemente están en crecimiento, el periodo de vida depende principalmente del grado de crecimiento de su población.

Cuando se realiza un proyecto, se debe prever que los elementos del sistema tengan capacidad para dar servicio durante un periodo a futuro a partir de su instalación, a este espacio de tiempo se le denomina, Periodo de Diseño. Al proyectar de esta manera se intenta satisfacer las necesidades de la sociedad que se comporta de manera dinámica.

El periodo de diseño es menor que la vida útil, por que se considera que durante éste los elementos funcionen sin tener gastos elevados que hagan su operación incosteable. Con respecto a la parte financiera de las obras se considera un Periodo Económico de las Obras, que se define como, el tiempo durante el cual la obra de ingeniería funciona económicamente. En sentido estricto al término de este periodo se debería construir una obra nueva pero la actual situación económica del país no lo permite. Se debe buscar siempre el máximo rendimiento de la inversión.

Normalmente la estimación de la vida útil del sistema se basa en la obra electromecánica y de control ya que esta dura mucho menos que la obra civil.

Actualmente se siguen patrones que indican los periodos de diseño para los diferentes elementos de los sistemas de agua potable así como para los de alcantarillado, los cuales se muestran en la siguiente tabla (recuérdese que no es lo mismo el periodo de diseño que la vida útil, siendo esta mayor).

Elemento	Periodo de diseño (años)
Pozo o embalse (presa)	5-20
Línea de conducción	5-20
Planta potabilizadora	5-10
Estación de bombeo	5-10
Tanque	5-20
Distribución primaria	5-20
Distribución secundaria	A saturación
Red de atarjeas	A saturación
Colector y emisor	5-20
Planta de tratamiento	5-20

Fuente: Valdez 1991

Así mismo se presenta la tabla en la cual se basan las estimaciones de vida útil en los elementos del Sistema de Agua Potable.

Elemento	Vida útil (años)
Pozo	
Obra civil	10-30
Obra electromecánica	8-20
Línea de conducción	20-40
Planta potabilizadora	
Obra civil	40
Obra electromecánica	15-20
Estación de bombeo	
Obra civil	40
Obra electromecánica	8-20
Distribución primaria	20-40
Distribución secundaria	15-30
Red de atarjeas	15-30
Colector y emisor	20-40
Planta de tratamiento	
Obra civil	40
Obra electromecánica	15-20

Fuente: Valdez 1991

2.2. El crecimiento de la población.

Es difícil estimar la población a años futuros. Se han utilizado varios métodos pero es mas preciso señalar que el ingeniero debe enjuiciar por si mismo cual de los métodos es más apropiado. El conocimiento de las industrias, estado de desarrollo de la región, situación con respecto a los transportes, próximo desarrollo debido a la instalación de industrias, todo entra en la estimación de la población futura o población de proyecto.

De igual manera, no se debe pasar por alto los sucesos extraordinarios como el de la instalación de una nueva industria se cual sea, ya que estos sucesos pueden transformar todos los cálculos en cuanto al futuro crecimiento, y exigirán un rápido aumento del consumo de agua y alcantarillado disponible. En caso contrario, la migración de la población debida al cierre de alguna industria o al final de alguna actividad económica en la región, la estimación futura de la población no tendrá afectaciones, ni exigencias que afectaran el proyecto debido a problemas de consumo de agua.

2.3. Recopilación, revisión y análisis de metodologías.

Al diseñar sistemas de abastecimiento de agua potable, nos basamos en una estimación de la población futura de la comunidad para la que se esta diseñando, la población proyecto, la cual corresponde al ultimo día del periodo de diseño que se fijó.

Ya sea a mayor o menor aproximación que se logre en la predicción de la población dependerá que la obra cumpla su cometido futuro, y que efectivamente al reducirse el grado de incertidumbre en el diseño pueda ser como se ha mencionado, mas económica.

Existen dos factores básicos del cambio en la población:

a. El aumento natural, es decir el exceso de los nacimientos sobre las muertes y
b. La migración neta, o sea el exceso o pérdida de la población que resulten del movimiento de las familias hacia adentro y hacia afuera de un área determinada.

Desagraciadamente las tasas de natalidad y muerte no se mantienen constantes a través del tiempo, es decir, que aun el hacer estimaciones de población de un año a otro encierra cierta incertidumbre e inexactitudes.

La interrelación de los dos factores del cambio en la población mencionados anteriormente, puede señalarse diciendo que, generalmente, mientras mayor sea la base de la población con que se trabaje, el crecimiento natural tendrá mas peso en el aumento de la población que la migración neta.

Es importante señalar además que las condiciones económicas tienen una influencia decisiva sobre los factores de crecimiento de la población, tanto en el aumento natural como en la migración neta. De esto se desprende que el análisis de las condiciones económicas es importante en la mecánica de la predicción del crecimiento de la población.

No importa el área para que se haga la estimación, deberán tenerse en cuenta, tanto las fuerzas económicas internas como las externas. Así como las condiciones mundiales afectan a la nación, las condiciones de las áreas metropolitanas afectan o influyen sobre las comunidades suburbanas.

La forma mas conveniente para la estimación de la población del proyecto se basa en su desarrollo pasado, tomando de los datos estadísticas y adaptar su comportamiento a modelos matemáticos, como son los que se mencionarán a lo largo de esta tesis.

Los atractivos de una comunidad como son el agua, alcantarillado, calles pavimentadas, comercios, zonas de recreación, etc., tanto como lugar para vivir como para trabajar, son también factores importantes en el crecimiento de su población.

Es importante destacar que deben tomarse determinadas precauciones y tener en cuenta algunos factores limitantes para hacer una buena predicción, por ejemplo, debe hacerse una buena estimación de la capacidad que puede admitir el terreno disponible para saber si una predicción determinada resulta o no razonable. Así, hay lugares que ya están congestionados de construcciones y tienen poco espacio para mas personas, en ello no importa cuales hayan sido las tendencias del pasado, las personas no pueden habitar por no existir mas espacio para ellas, es decir, que estas poblaciones están saturadas y por consiguiente no se puede suponer que tangan crecimiento futuro a la hora de estudiar el espacio disponible.

3. Generalidades de la programación.

El desarrollo de aplicaciones de cómputo surge a partir de la necesidad de resolver problemas numéricos que involucran la realización de una cantidad considerable de cálculos numéricos.

En la ingeniería, como en otras disciplinas, los programas de computadora representan una de las herramientas de apoyo mas importantes, por eso es que son cada vez mas los estudiantes, ingenieros y profesionistas en general quienes desarrollan aplicaciones de computo para ayudarse a resolver problemas.

Es por ello que los programas desarrollados para resolver este tipo de problemas representa una opción atractiva para todo aquel interesado en el tema, especialmente para el estudiante de ingeniería civil.

3.1. Las plataformas existentes y los lenguajes de programación.

El lenguaje de programación BASIC (Beginner´s All purpose Symbolic Introduction Code) nació en el año de 1964 como una herramienta orientada a principiantes, buscando una forma sencilla de realizar programas, empleando un lenguaje casi igual al usado cotidianamente y con instrucciones muy sencillas y escasas. Este lenguaje cubría casi todas las necesidades para la ejecución de programas.

En los años 70´s la evolución de BASIC fue escasa, dado el auge que tomaron en aquella época lenguajes de alto nivel como el FORTRAN y el COBOL. Mas adelante en la aparición de los primeros ordenadores personales y fue hasta mediados de los 80´s que el lenguaje BASIC resurgió como lenguaje de programación pensado para principiantes; sin embargo, fue gracias al entorno gráfico de Windows como Visual Basic tomó las ventajas sobre los otros lenguajes de computo y pudo satisfacer las necesidades que la programación requería.

Actualmente se han comercializado varias versiones de Visual Basic desde su salida al mercado. Cada versión supera y mejora la anterior. Dados los buenos resultados a nivel profesional de este producto, Visual Basic se ha convertido en la primera herramienta de desarrollo de aplicaciones en entorno de Windows. En la mayor parte de las aplicaciones, las herramientas aportadas por Visual Basic son mas que suficientes para lograr un programa fácil de realizar y de uso sencillo.

Por estas razones y debido a que en esta Facultad de Ingeniería Civil se imparte la materia de programación y en mi caso se me impartió utilizando Visual Basic es que se opto por realizar el programa en esta plataforma.

3.2. Características generales de Visual Basic.

Visual Basic es una herramienta de diseño de aplicaciones para Windows, en la que estas se desarrollan a partir del diseño de una interface gráfica. En una aplicación visual Basic, el programa esta formado por una parte de código puro, y otras partes asociadas a los objetos que forman la interface grafica. Es por tanto, un término medio entre la programación tradicional, formada por una sucesión lineal de código estructurado, y la programación orientada a objetos.

La creación de un programa en Visual Basic lleva los siguientes pasos:

1. Creación de interface de usuario. Esta interface será la principal vía de comunicación hombre-maquina, tanto para entrada de datos como para salida. Será necesario partir de una ventana (Formulario) a la que se le irán añadiéndolos controles necesarios.
2. Definición de las propiedades de los controles (Objetos) que se hayan colocado en dicho formulario. Estas propiedades determinarán la forma estática de los controles, es decir, como son los controles y para que sirven.
3. Generación del código asociado a los eventos que ocurran a estos objetos. A la respuesta a estos eventos (click, una tecla pulsada, etc.) se le llama procedimiento. Deberá generarse de acuerdo a las necesidades del programa.
4. Generación del código del programa. Un programa puede hacerse solamente con la programación de los distintos procedimientos que acompañan a cada objeto. Visual Basic ofrece la posibilidad de establecer un código de programa separado de estos eventos. Este código puede introducirse en unos bloques llamados Módulos, en otros bloques llamados funciones, y otros llamados procedimientos. Estos Procedimientos responden a un evento producido durante la ejecución del programa.

3.3. Uso de Visual Basic.

La aplicación de Visual Basic puede trabajar en dos modos distintos: en modo de diseño y en modo de ejecución. En el modo de diseño el usuario construye interactivamente la aplicación, colocando controles en el formulario, definiendo sus propiedades y desarrollando funciones para gestionar los eventos.

La aplicación se prueba en modo de ejecución. En este caso el usuario actúa sobre el programa (introduce eventos) y prueba como responde el programa. Todas las propiedades de los objetos se establecen en modo de diseño.

Para iniciar Visual Basic 6.0, que es la versión utilizada en esta tesis se pulsa sobre el botón Inicio/Programas/Microsoft Visual Basic 6.0 y se hace clic sobre el icono del programa.

En la ventana de Nuevo proyecto que aparece, se elige el icono EXE Estándar, para crear un programa típico como se observa en la siguiente figura.

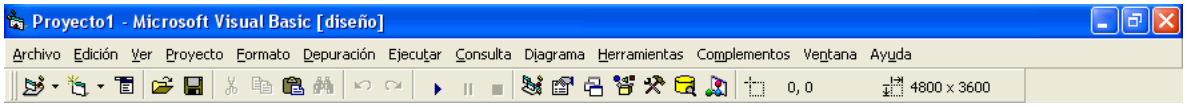
El siguiente paso es la creación de la interface grafica y su respectiva vinculación con el programa de cálculo. El diseño de la interface grafica tiene como objeto básico al formulario. Los formularios conforman la base de las aplicaciones, dado que en ellos se posicionan los controles y objetos con los que va a actuar el usuario.

3.4. Componentes de Visual Basic.

Cada uno de los controles con los que se trabaja en Visual Basic tienen sus correspondientes propiedades; estas se modifivan con fines visuales y de función según la índole del mismo. En las propiedades se establecen aspectos como son el tipo de letra, el color del fondo, dimensiones, posición en el formulario, tipo de borde, etc.

En la parte superior de la pantalla se sitúa la ventana principal de Visual Basic. En esta ventana aparece la barra de menús constituida por un conjunto de menús desplegables que facilitan los comandos necesarios para el trabajo en el entorno y la barra de herramientas, situada inmediatamente debajo, que da acceso rápido a algunas de esas operaciones.

Otra serie de botones que aparecen en esta barra de herramientas sirven para establecer criterios en Tiempo de ejecución Así, se podrá indicar la ejecución, interrumpirla o finalizarla, además de establecer otras opciones útiles en el momento de depurar la aplicación que se esté creando.



Ventana principal de Visual Basic.

La Caja de Herramientas que está anclada en la parte izquierda es un componente fundamental del entorno de desarrollo de Visual Basic.

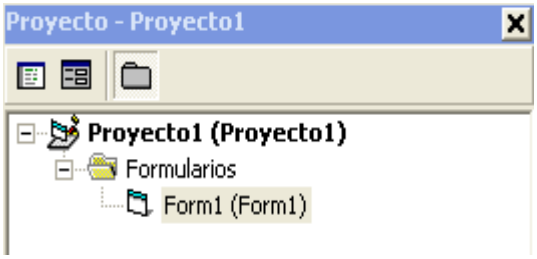
En esta caja se sitúan botones que representan los distintos Controles que se podrá utilizar durante el diseño de la aplicación. Estos controles serán situados en los distintos formularios que se vayan creando. La forma de trabajar es pulsar en el botón del control que se desea agregar al formulario y dibujarlo en éste o hacer clic sobre el botón, con lo que se crea un control de tamaño y posición predeterminada.

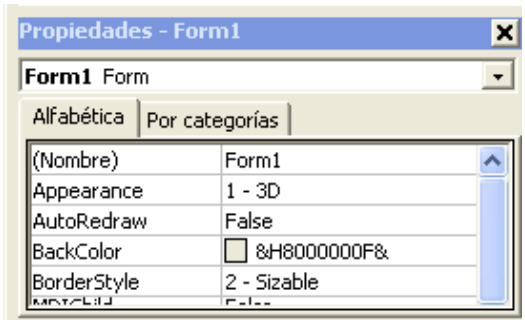
Es importante tener localizada y “a la mano” esta caja de herramientas ya que es uno de los componentes del entorno que más se utiliza en el diseño de la aplicación.

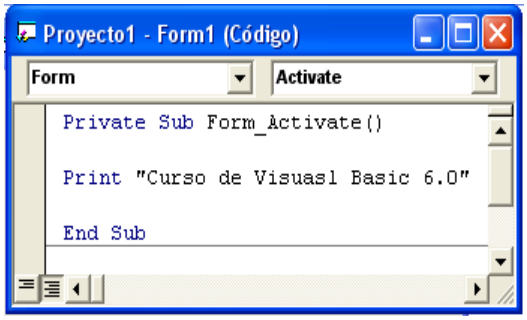
En la zona central se sitúa la ventana del formulario en el que se está trabajando. Si se han creado más formularios, estas ventanas se van situando en esa zona del entorno de programación. En la ventana del formulario es donde se agregan los distintos controles, gráficos e imágenes que constituyen la interfaz de la aplicación.

En la parte de la derecha se sitúa la ventana Proyecto. Esta ventana enumera los formularios y módulos del proyecto abierto. Si dicho proyecto está compuesto por numerosos archivos, esta ventana servirá para situarse en el formulario o archivo que nos interese.

Así, mediante el botón Ver objetos se podrá situar en un determinado formulario y mediante el botón Ver código se mostrará el código asociado con el elemento que esté seleccionado en la ventana.







Otro de los componentes fundamentales es la interfaz de la ventana de Propiedades, donde se enumeran las propiedades junto a los valore actuales del formulario o control que se tenga seleccionado. El conjunto de estas propiedades es dependiente del tipo de objetos al que se refiera.

Al hacer doble clic en un determinado formulario o control, aparece la ventana de código que tiene asociada. En esta ventana se escribe la parte del código que se corresponde con el objeto.

En la parte superior de la ventana de código aparecen dos listados de selección: en la de la izquierda se elige el Objeto al que se quiere hacer referencia y en la de la derecha el procedimiento o parte del código que se quiere programar.

3.5. Proyectos.

Visual Basic proporciona muchas herramientas para la creación de aplicaciones Windows. Este tipo de aplicaciones suelen ser sobre todo gráficas y se ejecutan en ventanas independientes.

Los proyectos se identifican como un archivo con la extensión .VBP que realiza el seguimiento de todos los componentes de la aplicación. Es decir, un proyecto está compuesto de varios archivos que son los que se van creando a medida que se desarrolla la aplicación. Así, un proyecto podrá contener uno o más archivos de formularios, donde se guarda todo lo referente a los formularios o ventanas que se vayan creando Estos archivos tienen la extensión .FRM y se guardan de forma independiente, uno por cada formulario.

Además se genera de forma automática un archivo .FRX para cada formulario que contenga datos binarios en alguna propiedad como puede ser una imagen o icono de los controles.

También se guardan los programas o códigos relacionados con el formulario en sí y con los controles que contiene.

Es posible que se desee agregar funcionalidad a la aplicación que no esté relacionada con ningún objeto en particular, como puedan ser procesos sobre archivos o utilidades que se desee utilizar en más de un lugar. En estos casos, ese código puede ser escrito en otro tipo de archivos como los archivos de módulos.

La extensión de un módulo estándar de Visual Basic es .BAS (Basic) y no está limitado a un único módulo sino que se puede crear los que estime convenientes. Así se podrían utilizar varios módulos para tratar distintos aspectos la aplicación.

Otro tipo de archivos que se pueden encontrar en un proyecto son los llamados módulos de clase, cuya extensión es .CLS, en esta tesis se usara solo para el procedimiento de imprimir, ya que aquí es donde ese establecen las propiedades de impresión como son pixeles por pagina, márgenes, encabezados y esta será la misma para todos los formularios que requieran imprimir alguna información.

Además de todos los archivos que se han comentado, existen los archivos con controles personalizados, que también pueden estar presentes en un proyecto.

Y finalmente un único archivo de proyecto, con la extensión .VBP, donde se guarda la información necesaria de cada uno de los archivos que forma el proyecto de programación y que han sido mencionados.

En definitiva, se entiende un proyecto en Visual Basic como la aplicación que se esté desarrollado. Toda la información que esté relacionada con esta aplicación se guardará en un archivo especial: el de proyecto.

Se puede trabajar a nivel de proyecto, a nivel de los archivos que lo componen. Así existen comandos que afectan a todo el proyecto, como pueden ser las opciones del menú Archivo: Nuevo proyecto, Abrir proyecto, Guardar proyecto o Guardar proyecto como.

3.6. Administrador de proyectos.

Cada una de estas opciones trabaja con el conjunto de proyecto. Por ejemplo, al guardar un proyecto, actualizará el archivo de proyecto y cada uno de los archivos que lo componen, sin embargo, otras opciones como Guardar archivo o Guardar archivo como sólo afectan al archivo de formulario o módulo que se tenga seleccionado.

Visual Basic permite que un mismo archivo esté presente en más de un proyecto. Se podría crear un formulario de bienvenida de introducción de datos o tener utilidades en un módulo independiente y usarlo en más de una aplicación, sin necesidad de crearlos cada vez.

Si se desea tener una copia individual de ese archivo, se podrá utilizar el comando Guardar archivo como una vez se haya agregado.

Por otra parte, también se tiene la posibilidad de eliminar un determinado archivo del proyecto. En este caso la referencia a ese archivo en el proyecto es suprimida. Se utiliza el comando Eliminar archivo y no se elimine desde el exterior de Visual Basic ya que en este último caso se produciría un error en el momento de abrir proyecto, al no poder encontrarlo.

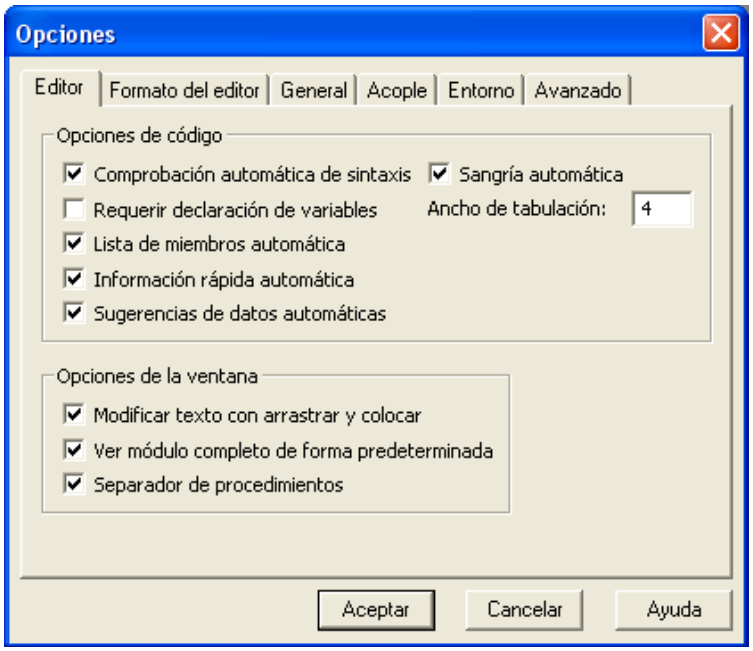
Con Agregar proyecto… se puede trabajar con varios proyectos dentro de una misma sesión del entorno de programación. Así, puede usar un proyecto como borrador donde poder probar el código, y a medida que se va depurando, incluirlo en el proyecto ejecutable de la aplicación.

De forma predeterminada Visual Basic ejecuta el primer proyecto ejecutable .EXE agregado a un grupo de proyectos.

El resultado final del proyecto debe ser un archivo ejecutable, que podrá tener o no extensiones en forma de librerías de enlace dinámico (DLL). Visual Basic incorpora una opción en el menú Archivo que crea un archivo ejecutable a partir de los archivos que componen el proyecto: Generar archivos EXE. El cual se empleara al final de la programación para tener al fin un programa que se pueda emplear en cualquier computadora que utilice Windows.

3.7. Opciones principales para comenzar a trabajar en Visual Basic.

En Visual Basic se pueden definir opciones del entorno de desarrollo del proyecto que se está creando, opciones de formato de código y de manejo de errores. Todas estas opciones se establecen través del comando Opciones del menú Herramientas, y de Propiedades en el menú Proyecto.



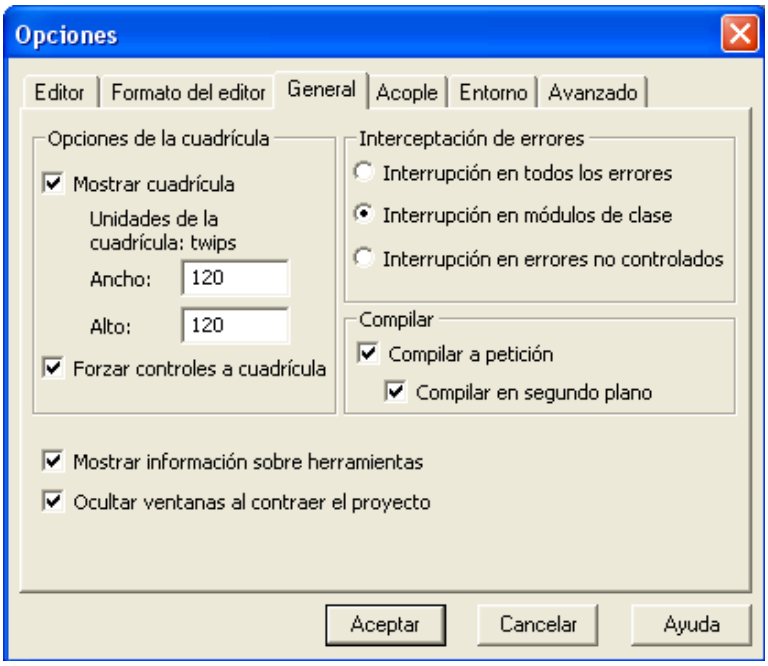
Al elegir este comando del menú Herramientas, aparece un cuadro de diálogo compuesto por seis fichas. La ficha Editor especifica la configuración de la ventana Proyecto y la de Código. En la sección Opciones de código hay que destacar las opciones de Comprobación automática de sintaxis, con la que Visual Basic comprueba de forma automática la sintaxis de cada línea de código que se escribe.

Con Requerir declaración de variable, le indicamos a Visual Basic que es necesario definir las variables que utilicemos en la aplicación.

Con Información rápida automática activa, Visual Basic nos proporciona información de los parámetros de la función que estamos utilizando.

La ficha Formato del editor sirve para establecer la apariencia del código que se escriba en Visual Basic. Aspectos como la fuente utilizada o el color de los distintos elementos del código, para poder diferenciarlos, son establecidos en esta ficha.

La ficha General establece aquellas opciones que se aplicarán sobre el entorno de desarrollo. Entre las opciones más interesantes de esta ficha, se pueden destacar la de Forzar controles a cuadrícula. Los formularios aparecen rellenos por una especie de cuadrícula que sirve para posicionar los controles de una forma alineada.

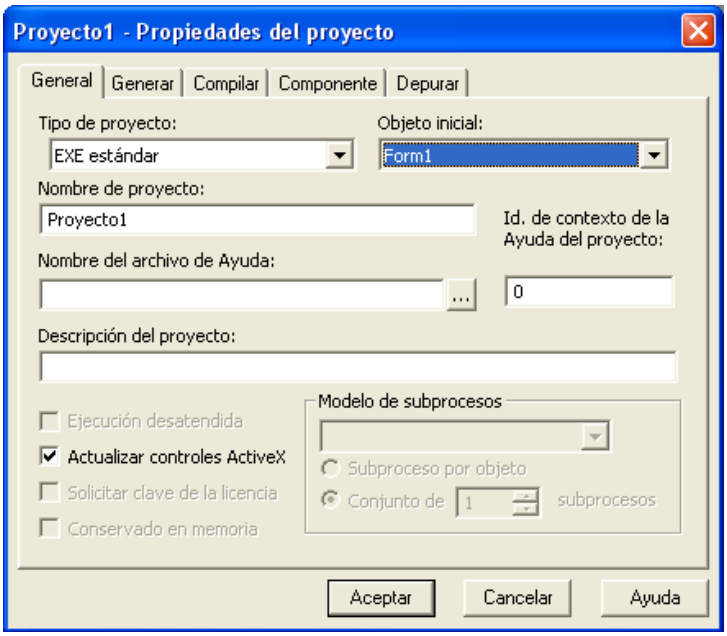


Si desactiva esta casilla, se tendrá una mayor libertad en la colocación de los controles, pero será más difícil alinearlos.

En el apartado de interceptación de errores se establece cuándo debe ponerse en modo de interrupción Visual Basic si reconoce un error de ejecución.

Una de las opciones más importantes es la de Preguntar si se guardan los cambios ya que permite guardar los cambios que haya efectuado en el proyecto, antes de ejecutarlo. Es importante porque es posible que, al ejecutar un proyecto, éste se bloquee y no responda, con lo que se perderían los cambios efectuados desde la última vez que se guardaron.

En la ficha propiedades del menú Proyecto se podrá modificar la configuración que afecta al proyecto actual. Así, se podrá indicar cuál debe ser el Objeto inicial de la aplicación (formulario inicial), el nombre del proyecto, el archivo de ayuda que tenga asociada si se ha creado alguno, etc.



Para este proyecto en lo general se trabajara con las opciones predeterminadas ya que para este caso no será necesaria ninguna otra forma de trabajo, solo se cambiaron las de guardar cada vez que se ejecute y que muestre los errores de código desde el momento que se esta escribiendo.

3.8. Tiempos de desarrollo.

Cuando se desarrolla el proyecto utilizando formularios, agregando controles, estableciendo propiedades o escribiendo códigos, se estará en tiempo de diseño.

En tiempo de diseño el entorno de programación de Visual Basic está a la disposición del programador, las distintas ventanas y menús pueden ser utilizados para diseñar la aplicación.

Cuando se pulse el botón Iniciar o se elige la opción equivalente del menú Ejecutar, se traslada al tiempo de ejecución, en el que interactúa con la aplicación del mismo modo que lo hará el usuario final. Este modo permite comprobar el resultado del esfuerzo de desarrollo.

Se debe distinguir, en este tipo de ejecución, las ventanas que corresponden a la aplicación que se está creando y la ventana propia de Visual Basic, que sigue situada en la parte superior de la pantalla. En ocasiones esta ventana no estará visible ya que puede que alguno de los formularios de la aplicación esté maximizado, ocupando toda la pantalla.

Cuando se está en tiempo de ejecución y se pulsa el botón Interrumpir se entra en el llamado tiempo de interrumpir. En este tiempo se realizarán operaciones de cara a la depuración de la aplicación.

Esto se estará realizando cada vez que se termine de escribir el código de cada uno de los formularios que se emplearán con el fin de detectar errores, así mismo se

realizara una ejecución de prueba para cada uno de ellos con ejemplos resueltos para su comprobación.

3.9. Sistema de ayuda de Visual Basic.

El lenguaje de programación de Visual Basic se compone de un gran número de instrucciones y funciones que se utilizarán a medida que se vayan programando las aplicaciones, además del número de propiedades, eventos y métodos aplicables sobre formularios y controles.

Para ello, Visual Basic incorpora un potente sistema de ayuda al que se podrá acudir en caso de necesitar información sobre cualquier situación que surja en el trabajo. Este sistema de ayuda forma parte del MSDN (Microsoft Developer Network). MSDN es la referencia esencial para programadores que utilizan las herramientas de desarrollo de Microsoft. Incluye más de 1 GB de información de programación técnica, código de ejemplos, etc.

Esta versión de MSDN Library se genera mediante el sistema de Ayuda HTML de Microsoft. Los archivos de Ayuda HTML se muestran en una ventana semejante a un explorador, en una ventana de Ayuda de tres paneles.

El panel de exploración se basa en cuatro fichas: Contenido, Índice, Buscar y Favoritos. Las cuatro fichas permiten encontrar la misma información, aunque la búsqueda se realiza de distinta forma en cada una de ellas.

Cuando se abra un determinado tema representado gráficamente por un libro, se puede encontrar de nuevo más libros u hoja de información inmediata. En el primer caso, se podrá volver a abrir el libro que representa un subtema, mientras que en el segundo, al abrir la hoja, la información requerida se mostrará directamente en pantalla.

La ficha Índice permite un acceso más rápido a la información buscada, eso sí, con la condición de que sea el usuario quien aporte una información más concreta sobre el tema de búsqueda.

En este caso los temas de ayuda se sitúan en una lista ordenada alfabéticamente y que aparece en la parte inferior de la ventana. Se podrá elegir un tema de la lista o escribirlo en la parte superior. Si existe más de un tema de ayuda relacionado con el texto que se ha introducido o elegido en la lista, aparecerá una nueva ventana en la que se deberá elegir el tema concreto.

En muchas hojas de información se podrá encontrar los llamados Hipervínculos. La forma de identificar un acceso de ese tipo es mediante el color azul y subrayado de su texto. Si se hace clic sobre un determinado acceso aparecerá información sobre el mismo de forma que la búsqueda de un tema original desencadena la obtención de información sobre otros temas distintos, navegando en el sistema de ayuda.

La ficha Buscar es similar a la ficha Índice ya que se tiene que teclear el tema sobre el que se desea información. Sin embargo, permite una búsqueda más amplia al poder introducir varias palabras o incluso frases completas. El resultado de la búsqueda

se amplía a todos los temas que puedan estar relacionados con alguna de las palabras introducidas.

Una búsqueda básica de temas se compone de la palabra o frase que se desea encontrar. Se pueden utilizar expresiones comodín, expresiones anidadas, coincidencias de palabras similares, la lista de resultados anterior o títulos de temas para afinar la búsqueda.

Sin embargo, el uso más frecuente que hará del sistema de ayuda será a través de la tecla F1. Si se desea información sobre alguna propiedad, método, instrucción del lenguaje, etc., la forma más rápida y cómoda de conseguirla es situar el cursor en dicha palabra y pulsar la tecla F1.

Para el caso del uso de la ayuda, cabe mencionar que se debe de instalar de manera separada esta librería, en este caso este disco fue proporcionado por el asesor de la misma manera que el disco de instalación de Visual Basic.

3.10. Estructura del código.

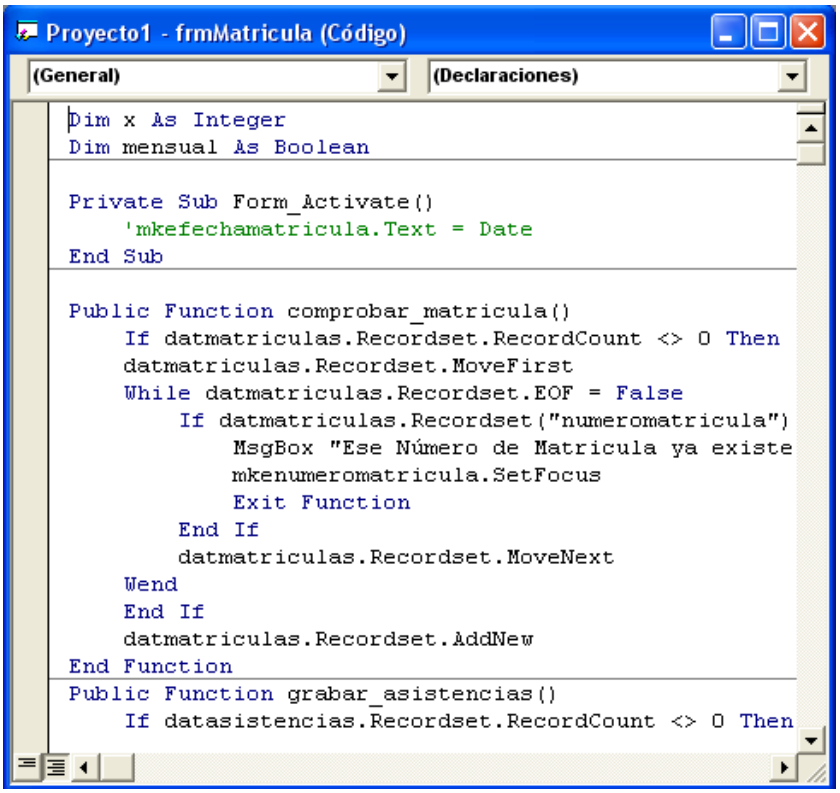
El código que se escribe en un proyecto de Visual Basic aparecerá siempre en un módulo.

Un módulo es un archivo del proyecto pudiendo ser un módulo de formulario, de clase o módulo de carácter general.

En cada módulo, el código se divide en dos secciones: declaraciones y procedimientos. Los procedimientos son unidades de código, como pequeños programas, escritos para realizar funciones determinadas, con un propósito bien definido.

En cualquier módulo, el programador dispone de una sección especial llamada General en la que sitúan las declaraciones y en la que se puede incluir otros procedimientos creados por el mismo programador.

En el apartado de declaraciones se pueden introducir las constantes, variables y tipo de datos que se necesiten en la aplicación.



Los procedimientos pueden tener parámetros, especificados entre paréntesis, que nos permiten comunicar al procedimiento alguna información que se necesite o que sea el propio procedimiento quien devuelva algún valor.

En un módulo de formulario el código que se sitúa se refiere tanto a dicho formulario como al resto de objetos que estén dibujados en él. En este tipo de módulos cabe destacar los procedimientos de evento que se encarga de dar respuesta programada a los eventos que ocurren en la aplicación.

Si se ha escrito el código para algún procedimiento de evento, éste aparece en negrita en la lista Procedimiento: de la ventana de código.

Por otra parte, el código de carácter general o que se desea compartir en más de un proyecto, se sitúa en un módulo general de Visual Basic. El código que aparece en este tipo de módulo no se relaciona con un objeto determinado sino que tiene carácter general.

En los módulos generales no se podrán incluir procedimientos de eventos como en el caso de los módulos de formulario Por otra parte, es conveniente añadir comentarios a las líneas de código que se escriba. De esta forma se podrán entender los programas aunque haya pasado tiempo que se escribieron.

Para añadir un comentario en una línea se utiliza el carácter (apostrofe). Al introducir este carácter en una línea, Visual Basic entiende todo lo que le sigue en dicha línea es un comentario y no lo toma en cuenta.

3.11. Objetos en Visual Basic.

Visual Basic no presenta todas las características de un lenguaje orientado a un objeto, se trabaja con objetos al crear la interfaz de la aplicación.

En este sentido se debe entender un objeto como todo elemento o entidad que puede identificar la ampliación. Desde un formulario hasta una línea gráfica que se dibuje, pasando por la propia aplicación que también es considerada como un objeto. Un objeto en Visual Basic se caracteriza por tres componentes: ***propiedades, métodos y eventos***.

Las propiedades son aquellas características o atributos que permiten establecer el aspecto de un objeto como el color, el tamaño, la posición en pantalla, etc., o el estado del mismo: activo, deshabilitado, maximizado, etc. Existen propiedades que sólo están disponibles en tiempo de diseco, otras sólo están disponibles o son sólo lectura en tiempo de ejecución.

Los métodos son pequeños programas que actúan sobre un determinado objeto y que establecen su comportamiento. Así un objeto puede moverse, ocultarse, etc. Se puede utilizar cualquier método que forme parte del objeto.

Finalmente, un evento es una situación que nos interesa identificar para establecer algún tipo de respuesta por parte del objeto. Así, se podría hacer clic sobre un determinado objeto, creando, de esta forma, un evento reconocible por el objeto. Que suceda algo o no como respuesta a este evento dependerá de que se haya programado alguna acción en el correspondiente procedimiento del evento.

3.11.1. Propiedades.

Cuando se dibujen objetos en los formularios se deseará establecer alguna de las propiedades que presenten. Dichas propiedades son aquellas características, propias del objeto, que hacen que se distinga un objeto de otro.

En una aplicación Windows podemos distinguir diversos tipos de ventanas, aunque lo normal es identificar una ventana inicial que se puede maximizar o minimizar y uno o más cuadros de diálogo cuyo tamaño suele ser fijo. Sin embargo, las barras de herramientas también son ventanas de una aplicación.

Normalmente, la ventana inicial o de arranque será la que presenta el menú principal, formado por menús desplegables en uno o más de un nivel. El resto de ventanas no deberían contener menús desplegable, aunque puede darse el caso de que así sea.

A continuación se presentan algunas propiedades que se utilizaran en los formularios. Las propiedades que se pueden establecer en tiempo de diseño se sitúan en la barra de Propiedades.

`BorderStyle`**:** establece el estilo del borde del formulario. Puede indicar que sea un formulario dimensionable o al estilo de los cuadros de diálogo, que no cambie de tamaño.

`Caption`: establecer el texto que aparece en la barra de título del formulario.

`ControlBox`: permite mostrar o no el menú de control de las ventanas de Windows. Desde este menú se podrá cerrar, mover, etc., la ventana.

`Enabled`: establece si el formulario puede o no responder a los eventos que genere el usuario.

`Font`: establece las características de los objetos de texto que se sitúen en el formulario.

`Icon`: permite cambiar el icono que representa el formulario y que está situado en la parte izquierda de la barra de título.

`Left, Top, Height y Width`: establece la posición en pantalla del formulario, así como sus dimensiones. Left es la coordenada x, Top la coordenada y, Height la altura y Width la anchura.

`MaxButton y MinButton`: Permiten indicar si se deben mostrar o no los botones maximizar y minimizar, respectivamente. Existen estilos de bordes de formulario (establecidos con la propiedad BorderStyle) que impide que se muestren los botones, independientemente del valor de las propiedades MaxButton y MinButton.

`MousePointer`: permite modificar la forma del puntero del ratón.

`Nombre`: nombre identificativo del formulario. Deberá utilizar este nombre en el código para referirse al formulario.

`Visible`: establece si el formulario se mostrará visible o no en tiempo de ejecución.

`WindwState`: establece el modo en el que se cargará inicialmente el formulario. Puede ser normal, es decir, como haya sido creado en tiempo de diseco, maximizado o minimizado.

Todas estas propiedades y muchas más están disponibles, en tiempo de diseño, a través de la ventana Propiedades.

La forma de establecer el valor de una propiedad depende de su naturaleza. Existen propiedades de valor numérico en las que simplemente se introduce el valor deseado; propiedades de carácter textual, en las que tendrá que introducir el texto; propiedades en las que tendrá que especificar un determinado archivo; propiedades compuestas en las que el valor está compuesto por más de una característica y propiedades booleanas, en las que sólo puede existir dos valores: True (verdadero) o False (falso).

Además, en tiempo de ejecución, también se puede cambiar o consultar el valor de algunas propiedades que sólo tienen sentido en ese tiempo de desarrollo.

Al escribir código, las formas de establecer una propiedad o de poder consultarla es a través de la sintaxis: `NombreObjeto.NombrePropiedad`. Recuérdese que se debe introducir el punto (.) entre los dos nombres.

Convención para nombrar objetos de Visual Basic.

A continuación se mostrará una convención, indicada por Microsoft, para nombrar los distintos objetos que se deseen crear en la aplicación, es decir, los formularios y controles.

Esta convención se basa en la inserción de un prefijo identificado de la naturaleza del objeto, junto al nombre descriptivo del mismo. El prefijo será de tres letras en todos los casos. Se indica el nombre del objeto tanto en inglés como en español.

Objetos (Ingles)	Objetos (Español)	Prefijo
Form	Formulario	frm
Check box	Casilla de verificación	chk
Combo box	Cuadro combinado	cbo
Dat-bound combo box	Cuadro combinado enlazado a datos	dbc
Command button	Botón de comando	cmd
Data	Control de datos	dat
Directory list box	Cuadro lista de directorios	dir
Drive list box	Cuadro lista de unidades	div
File list box	Cuadro lista de archivos	fil
Frame	Marco	frm
Grid	Cuadrícula	grd
Data-bound grid	Cuadricula enlazada a datos	dbg
Horizontal scroll bar	Barra de desplazamiento Horizontal	hsb
Image	Imagen	img
Label	Etiqueta	lbl
Line	Línea	lin
List box	Cuadro de lista	lst
Data-bound list box	Cuadro de lista enlazada datos	dbl
Menu	Menú	mnu
OLE container	Contenedor OLE	ole
Option button	Botón de opción	opt
Piture box	Cuadro de imagen	pic
Shape	Forma	shp
Text box	Cuadro de texto	txt
Timer	Temporizador	tmr
Vertical scroll bar	Barra de desplazamiento Vertical	vsb

3.11.2. Métodos.

Una vez que se ha establecido la parte estructural del formulario, es decir, sus propiedades, se puede pasar a programar el comportamiento que debe seguir, de forma que cumpla con el objetivo para el que fue creado, utilizando métodos.

Un ***método*** es un componente más del objeto, como lo puede ser una propiedad, que puede ser utilizado directamente. Para ello se deberá llamar al método e indicarle sobre qué objeto se desea aplicarlo, que deberá admitirlo.

La sintaxis en este caso es: `NombreObjeto. NombreMétodo [par1, , parn]`, donde la lista de parámetros [par1, ,parn] es opcional y depende de cada método.

Visual Basic determina lo que se quiere hacer a través del nombre que se introduzca detrás del punto; no existen paréntesis para la lista de parámetros.

Por ejemplo: el método más utilizado sobre un formulario es Show. Este método permite hacer visible un formulario en pantalla. Si el formulario no está cargado en memoria lo carga automáticamente y lo muestra.

Si desea mostrar el formulario frmPrincipal la sintaxis correcta seria: `FrmPrincipal.Show i` donde el parámetro de la i indica la forma en que se muestra el formulario: con valor 0, el formulario es no modal, es decir, el enfoque puede pasar a otro objeto sin necesidad de realizar alguna acción sobre el formulario.

En caso de tener valor 1, será necesario realizar alguna acción sobre el formulario (seguramente se tendrá que cerrar) para que el enfoque pueda pasar a otra ventana de la aplicación.

El número de métodos disponibles es realmente elevado y sensible al objeto sobre el que puede ser aplicado. A lo largo de la tesis se irán conociendo los más importantes y mas utilizados.

3.11.3. Eventos.

El tercer componente de un objeto es el conjunto de eventos que reconoce. Este conjunto de eventos está predefinido en el lenguaje de Visual Basic y no se podrán crear nuevos eventos para objetos.

El conjunto de eventos reconocible por un objeto está fuertemente ligado con los procedimientos de evento. Estos procedimientos son unidades de código que permiten especificar la respuesta que debe dar un objeto ante la ocurrencia de un evento que puede reconocer.

La definición de un procedimiento de evento tiene la forma:

```
PrivateSubNombreObjeto_NombreEvento([Listade parámetros])
EndSub
```

La palabra `Private` indica que el procedimiento sólo está visible en el módulo actual, en contraposición a lo que indica `Public`. La lista de parámetros puede ser opcional, pero no así los paréntesis.

3.12. Guardar un proyecto en disco.

Al guardar un proyecto nuevo se tendrá que dar nombre tanto al archivo de los formularios que lo compongan, como el de proyecto. Recuérdese que los distintos formularios y módulos que se creen se guardan en ficheros independientes.

Es conveniente crear una carpeta para guardar los distintos ficheros que se vayan creando en el proyecto de Visual Basic, para tenerlos siempre localizados.

Se utiliza la opción Guardar proyecto del menú Archivo o el botón equivalente de la barra de herramientas. Al guardar el proyecto también se guardarán los formularios del mismo.

3.13. Menús.

3.13.1. El editor de menús.

Una de las formas más elegantes de poner un gran número de comandos en la ventana del usuario es a través del uso de menús.

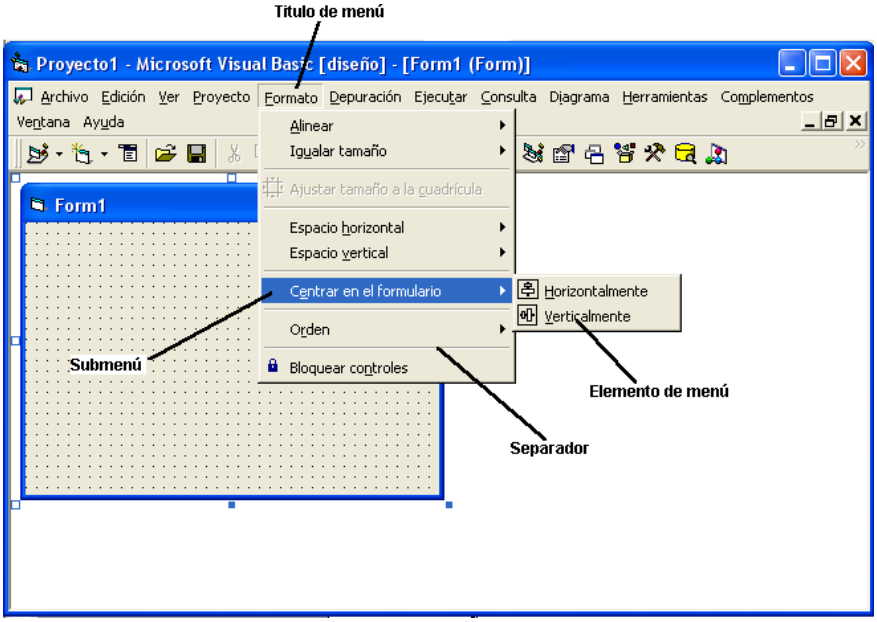
Como se ha visto en la mayoría de aplicaciones de Windows cuentan con un menú principal en la ventana inicial de la aplicación y otros menús, como puedan ser los contextuales, en una o más ventanas.

Al usar menús se está evitando ocupar mucho espacio en las ventanas ya que estos sólo aparecen cuando se despliegan explícitamente. Además, se pueden crear hasta cinco niveles de menús, por lo que el número de comandos y posibilidades es suficientemente amplio.

En Visual Basic se trabaja con el Editor de menús para crear, establecer propiedades y modificar los menús. Para acceder a este editor se deberá pulsar el botón en la barra de herramientas o elegir el comando equivalente del menú Herramientas.

Este botón sólo estará activo cuando se tenga abierto un formulario, ya que en menú siempre estará asociado con un determinado formulario.

El editor de menús presenta las principales propiedades que se pueden aplicar sobre los menús.



`Caption`: representa el texto que aparecerá en el menú. En este sentido se debe distinguir entre lo que es un título de menú, o el menú de nivel superior, y lo que es un elemento de menú o submenús.

En la figura anterior. Existe en menú principal en el que se sitúan los títulos de menú. Uno de estos títulos de menú es Formato.

Al desplegar el menú Formato aparecen elementos de menú, incluidos los separadores. Uno de estos elementos de menú es Fila.

Además, el elemento Fila actúa como un submenú ya que al posicionarse en esta opción se vuelve a desplegar otro menú, con más elementos de menú, entre los que se encuentra Mostrar.

Obsérvese como un elemento de menú puede actuar al mismo tiempo como submenú (o título de otro menú).

`Name`: es el nombre del control. La estrategia a seguir en este caso puede ser la de anteponer el prefijo mnu más el texto del título, si es un título de menú. En el caso de ser un elemento del menú, se debería incluir también el texto de su título de menú.

`Cheeckd`: establece si debe aparecer una marca de verificación a la izquierda del texto del objeto menú. Esta marca suele utilizarse cuando desea informar al usuario de la opción que está activa en un momento determinado.

`Enabled`: indica si el objeto podrá recibir eventos del usuario. Si tiene el valor False, el elemento aparecerá atenuado en el menú.

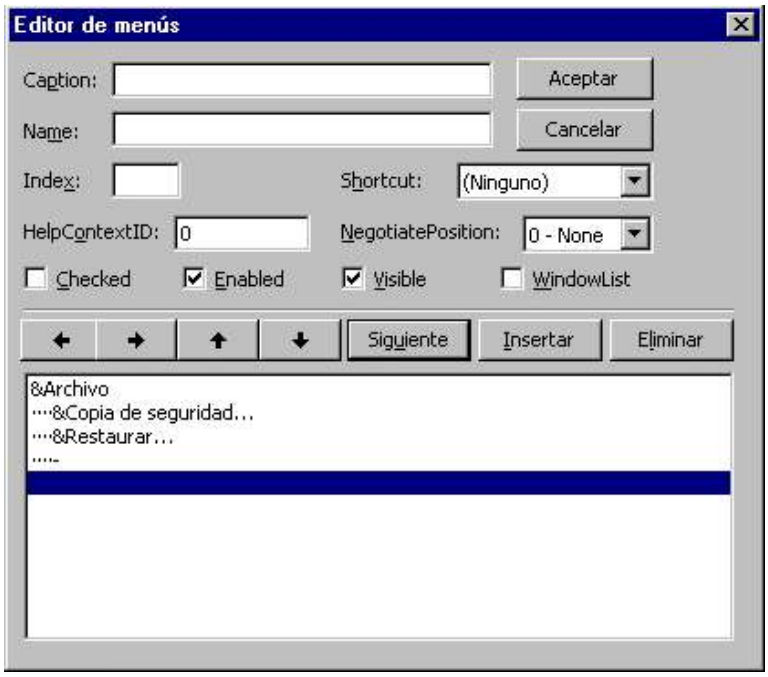
`Visible`: indica si el objeto debe mostrarse o no. Si se establece el valor False para esta propiedad, los elementos situados a la derecha ocupan el lugar del elemento no visible.

Las propiedades `Enabled` y `Visible` pueden tomar valor en tiempo de diseño o modificarlos en tiempo de ejecución. Por ejemplo, si se desea que no aparezcan títulos de un menú principal (y sus elementos) debido a las circunstancias del momento, se puede utilizar la propiedad Visible para ello.

3.13.2. Elementos del menú.

Utilizando los botones de sangría se determinará el nivel del objeto menú que se tenga seleccionado. Así, los títulos de menú se sitúan en el nivel superior, mostrándose en la lista inferior del editor de menús, lo más a la izquierda posible.

Al pulsar el botón ⇨ más sangría el nuevo objeto menú se sitúa en un nivel inferior. Lo contrario ocurre al pulsar ⇦.



Además con los botones ⇧ ⇩ se puede mover de posición elementos de menú que ya se haya creado. Finalmente si desea insertar un nuevo elemento, se utiliza el botón Insertar. El nuevo elemento se situará encima del elemento que se tenga seleccionado.

3.14. Cuadros de dialogo.

En la mayoría de aplicaciones Windows el usuario hace uso de los cuadros de diálogo para establecer opciones o introducir información que la aplicación necesita para realizar la acción indicada por el usuario. De ahí el nombre de esta ventana, ya que establece un “diálogo” entre el usuario y la aplicación para conseguir el objeto perseguido.

Otro uso posible de los cuadros de diálogo es presentar información al usuario. El usuario puede demandar detalles sobre algún aspecto de la aplicación (por ejemplo el cuadro de diálogo Acerca de que aparece en la mayoría de aplicaciones) o haber pedido la realización de un determinado proceso que implica mostrar el resultado en pantalla.

Los cuadros de diálogo se caracterizan, a diferencia de las ventanas principales de la aplicación, por ser modales y no permitir cambiar el tamaño de la ventana, además, no tienen una barra de menús desplegable.

En definitiva, un cuadro de diálogo es un formulario más del proyecto de programación de Visual Basic en el que se ha establecido ciertos valores en algunas de sus propiedades, para que tenga el aspecto y se comporte como tal.

En Visual Basic se pueden crear tres tipos de cuadros de diálogo: **personalizados, predefinidos** y a través del control **Diálogo Común**.

Para esta tesis solo se emplearan cuadros de dialogo predefinidos de los cuales se mencionan sus características y sintaxis a continuación.

Visual Basic incorpora dos funciones en su lenguaje que permiten agregar cuadros de diálogo a las aplicaciones de una forma rápida y sencilla, ya que no se tiene que preocupar del diseño, de la carga o cómo es mostrado el cuadro de diálogo.

En contra partida, se tendrá poco control sobre la apariencia del cuadro de diálogo que, por otra parte, será siempre modal.

Un formulario es modal cuando se tiene que cerrar (ocultar o descargar) dicho formulario para que el foco de la aplicación pase a otro formulario. Sin embargo, en un formulario no modal, el foco puede pasar a otro formulario sin necesidad de realizar ninguna acción en él.

La sintaxis completa de la función MsgBox es:

```
MsgBox(mensaje[,botones] [,título] [,archivoAyuda, contexto])
```

En esta tabla se muestran los valores que puede tomar el parámetro botones:

Constante	Valor	Descripción
VbOKCancel	0	Muestra solamente el botón aceptar.
VbOKCancel	1	Botones Aceptar y Cancelar.
VbAbortRetryIgnore	2	Botones Anular, Reintentar e Ignorar
VbYesNoCancel	3	Botones Si, No y Cancelar.
VbYesNo	4	Botones Sí y No
VbRetryCancel	5	Botones Reintentar y Cancelar
VbCitrical	16	Muestra el icono de mensaje crítico.
VbQuestion	32	Icono de interrogación.
VbExclamation	48	Icono de exclamación.
VbInformation	64	Icono de mensaje de información.
VbApplicationModal	0	Cuadro de diálogo modal de aplicación.

Existen dos tipos de cuadro modales: modal de aplicación y modal de sistema. Cuando un cuadro de diálogo es modal de aplicación, se deberá cerrar para poder interactuar con otra ventana de la misma aplicación. Sin embargo, se podrá cambiar de aplicación utilizando algunos de los métodos de Windows. Este tipo de cuadros de diálogo son los más usuales.

Cuando un cuadro de diálogo es modal de sistema, todas las aplicaciones que estén ejecutándose se suspenden hasta que el usuario responda al cuadro de diálogo y, en definitiva, lo cierre. Este tipo de cuadros de diálogo sólo tiene sentido en situaciones donde el mensaje que muestran es crítico para el sistema.

Otro parámetro útil de la función `MsgBox` es el título de dicho cuadro de diálogo. Si no se establece, Visual Basic utilizará el nombre del proyecto como título de la ventana.

El otro cuadro de diálogo predefinido que presenta Visual Basic, se consigue a través de la función `InputBox`. Esta función se utiliza cuando se necesita que el usuario introduzca alguna información. Como toda función devuelve el valor, que en este caso de

un será la cadena introducida por el usuario. En el caso de un `MsgBox` la única información que podrá obtener será qué botón ha pulsado el usuario para cerrarlo.

La sintaxis correcta es:

```
InputBox(mensaje[, título] [, estándar] [, posx] [, posy] [,
archivoAyuda,contexto])
```

Donde, aparte del mensaje que aparece en el cuadro de diálogo, puede indicarse el título de éste y la cadena que devolverá la función si no se escribe ninguna. También puede indicarse la posición en la pantalla del cuadro de diálogo.

La forma lógica de llamar a esta función será en una expresión de asignación, es decir, recogiendo el valor que introduzca el usuario.

Este tipo de cuadro de dialogo será el mas usado a lo largo de la programación para introducir los datos con los que se vaya a trabajar es por ello que se hace tanta énfasis en este tipo.

3.15. Controles básicos.

La primera fase en la creación de una aplicación con Visual Baste es la de diseñar la interfaz de usuario de dicha aplicación. Esta Interfaz está constituida por el conjunto de formularios que aparecen a lo largo de su ejecución.

Entre los numerosos controles que presenta Visual Basic se encuentran: etiquetas, cuadros de texto, botones de comando, etc.

Existen otros muchos controles e incluso controles personalizados que aparecen independientemente de Visual Basic. Estos controles se encuentran en archivos separados, por lo que deben ser incorporados a la caja de herramientas para poder utilizarlos.

Cada uno de estos controles son tratados como objetos en Visual Basic, por lo que tendrán su propio conjunto de propiedades, métodos y eventos.

A continuación se describen los controles más importantes que se utilizaran en el desarrollo de esta tesis.

3.15.1. Controles de entrada de datos.

Dos controles muy relacionados y que se utilizan en la función de entrada de datos son las etiquetas y los cuadros de texto.

Las etiquetas son controles que nos permiten mostrar texto en los formularios y que tienen la particularidad de que el usuario no puede modificar dicho texto (aunque el programador sí que lo puede hacer en tiempo de ejecución). Se podrán dibujar etiquetas en los formularios a través del control situado en la caja de herramientas.

Sin embargo, los cuadros de texto son el control estándar de entrada de datos en Visual Basic. Su propósito es el de permitir al usuario introducir aquella información que necesita la aplicación. El texto que se introduzca está a disposición del usuario, pudiendo llegar a modificarlo. El control de la caja de herramientas permite dibujar cuadros de texto.

Entre las numerosas propiedades de un objeto etiqueta cabe destacar:

- `Alignment`: establece la alineación del texto de la etiqueta.
- `Autosize`: si tiene valor True, el tamaño de la etiqueta se adapta automáticamente a su contenido.
- `Caption`: es el texto que se visualiza en la etiqueta.
- `Enabled`: permite o no interactuar con la etiqueta. Esta propiedad no se utiliza mucho ya que una etiqueta no puede recibir el foco. Si tiene valor False se mostrará en gris.
- `Font`: establece el estilo de la escritura, esto es, la fuente, tamaño y realces del texto que aparece en la etiqueta.

Los cuadros de texto son semejantes a las etiquetas, pero con la diferencia que el usuario puede modificar su contenido. El texto que se introduzca puede ser tanto numérico como alfanumérico (números y letras).

A diferencia de una etiqueta, el tamaño del cuadro de texto es fijo al no permitir la propiedad `Autosize`. Esto es lógico ya que un cuadro de texto suele estar pensado para introducir datos de distinta longitud.

Las propiedades más interesantes de un cuadro de texto son las siguientes:

- `Alignmet`: alineación de la información introducida. Utilice alineación a la izquierda para el texto y a la derecha para los números. También puede centrarla.
- `CausesValidation`: determina si el control desde el que ha cambiado el enfoque activa su evento valídate.
- `Enabled`: permite o no que el usuario realice acciones sobre el cuadro de texto.
- `Font`: exactamente igual que en el caso de las etiquetas.
- `Multiline`: muy importante ya que permite introducir más de una línea de texto en el objeto.
- `PasswordChar`: establece qué carácter se muestra al realizar una entrada en el objeto. Independientemente de lo que introduzca el usuario, en pantalla sólo se mostrará dicho carácter.
- `ScrollBars`: permite mostrar barras de desplazamiento para el caso de que el texto introducido no se pueda visionar completamente.
- `TabIndex`: establece el orden de tabulación fijado para el objeto.
- `TabStop`: indica si, al utilizar el tabulador para desplazarse entre los controles del formulario, debe o no tenerse en cuenta este control.
- `Text`: texto que se visualiza en el control y que suele haber sido introducido por el usuario.

- `ToolTipText`: texto mostrado cuando el ratón se sitúa sobre el control.

3.15.2. Control marco.

Si se desea estructurar un formulario en distintas secciones, agrupando controles en éstas para que la lectura sea más clara, el control que se debe utilizar es el control Marco (`Frame`).

Dicho control puede tener una intención puramente estética o de legibilidad como sería el caso mencionado o de mayor necesidad, como se podrá ver cuando se usa el control casilla de opción que será empleada al inicio del programa.

Para dibujar un control Marco, se utiliza el botón de la caja de herramientas.

La única propiedad interesante de un control marco, aparte de su nombre, es la propiedad `Caption` que se refiere al texto que se muestra en la parte superior izquierda del marco.

El control Marco nos permite introducir el concepto de contenedor. Cuando se dibujan controles en un formulario, estos son controles independientes que poseen la característica de estar situados en un determinado formulario. En este caso se dice que el objeto contenedor es el formulario.

Sin embargo, la incorporación de un control marco al formularlo puede cambiar este aspecto y permitir que sea dicho marco el contenedor de otros controles que se dibujan en su interior.

Esta característica tiene su importancia ya que los valores de las propiedades `Left` (Izquierda) y `Top` (Arriba) se establecen en función del objeto contenedor del control. Por ello, cuando se mueve un formulario también se mueven los controles que están situados en él.

Para que un objeto marco sea el objeto contenedor de otros objetos dibujados, deberá haberse dibujado primero el objeto marco y después dibujar el resto de los objetos en el interior del objeto marco. Si ya se tienen controles dibujados y se quieren introducir en un marco, entonces se deberán copiar o cortar y pegarlos en el marco. Así también será el control marco el objeto contenedor de los objetos pegados.

3.15.3. Botones de comando.

El botón de comando es el control más utilizado en cualquier aplicación Windows. La manera más sencilla de permitir que un usuario interactúe con la aplicación es proporcionarle un botón para que haga clic en él.

Los botones de comando son una manera natural e intuitiva de ejecutar acciones en la aplicación complementan en este sentido el uso de menús, ya sean despegables o contextuales.

Podrá dibujar un botón de comando a través del control situado en la caja de herramientas.

El control botón de comando (`ComandButton`) es muy sencillo, por lo que no se necesitarán establecer muchas propiedades. En `Caption` se indica el texto que deba aparecer en el botón y se introduce su nombre de objeto. El resto de propiedades se utilizarán pocas veces. Como en el resto de controles que presentan esta propiedad, también se podrá crear una tecla de acceso al botón.

El uso principal de los botones de comando es justamente realizar acciones en la aplicación. Sin embargo, también se debe situar algún botón de comando para poder cerrar expresamente la ventana en la que se encuentra.

Para es caso de la aplicación a crear se utilizaran los botones generales de la ventana para salir de la aplicación.

En los cuadros de diálogo normalmente aparecerán los botones los botones Aceptar y Cancelar mientras que en otras ventanas de la aplicación puede encontrarse con botones como el de Cerrar, cuya funcionalidad es la de cerrar (ocultar o descargar) el formulario abierto.

Se deben diseñar los formularios de forma que los controles que sirvan para lo mismo se sitúen siempre en la misma posición. Por ejemplo, se deben colocar siempre en el mismo lugar los botones que permitan cerrar las ventanas, de forma que el usuario llegue a acostumbrarse a verlos en esa posición y los use de una forma casi impulsiva.

Un buen lugar para estos botones es la parte inferior derecha, como puede observarse en muchas aplicaciones Windows.

Existen dos propiedades que se aplican a los botones comentados que permiten cerrar una ventana. Si se desea cerrar la ventana de forma que las acciones que se hayan podido hacer no lleguen a tener efecto (típico en el sentido de un botón Cancelar), será importante establecer la propiedad `Cancel` a `True`.

Al establecer esta propiedad se permite que el usuario pueda utilizar tanto el botón Cancelar como la tecla [Esc].

Sin embargo, si se desea que se lleven a cabo las acciones realizadas en la ventana, se debe utilizar un botón Aceptar y establecer su propiedad Default a True. De esta forma se permitirá utilizarla tecla [Intro] como si pulsara en dicho botón.

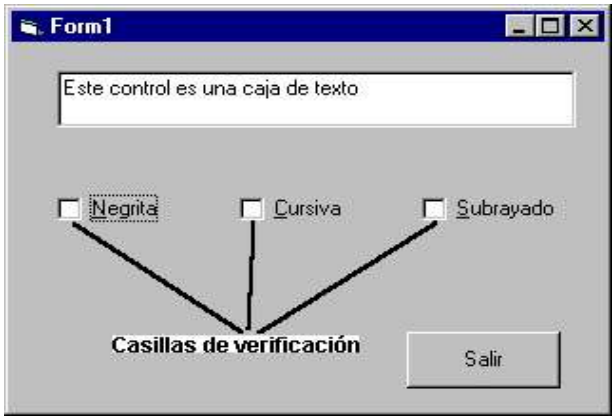
El botón Cerrar suele encontrarse en ventanas donde se realizan acciones sin necesidad de cerrar la ventana. Su propósito es el de permitir cerrar la ventana cuando el usuario lo decida.

3.15.4. Casillas de verificación.

En muchas ocasiones se desea proporcionar la posibilidad de elegir entre distintas opciones a los usuarios.

En estas situaciones se podrá incorporar a los formularios distintos controles que presentan opciones como pueden ser las casillas de verificación, botones de opción o a través de una lista.

Las casillas de verificación (`CheckBox`), permiten establecer opciones que no son excluyentes entre sí, es decir, se puede seleccionar una más de una, todas o incluso ninguna de las opciones.



Una casilla de verificación puede estar activada (`checked`), indicando que dicha opción ha sido seleccionada o desactivada (`unchecked`), indicado lo contrario. Además la casilla puede estar atenuada (`grayed`), indicando que el objeto no está disponible al tener un valor indeterminado. La propiedad que tiene estos valores es `Value`.

El evento más utilizado en este tipo de objetos es el momento en el que se pulsa en la casilla. En esta situación el usuario quiere indicar que desea activar o desactivar la casilla, dependiendo de cuál sea su valor.

3.15.5. Botones de opción.

Los botones de opción (`OptionButton`) también permiten presentar opciones al usuario, pero con la particularidad que sólo podrá seleccionar una de dichas opciones al mismo tiempo.

Por ello es importante establecer conjuntos de botones de opción. Si se desea que en un mismo formulario se pueda seleccionar más de un botón de opción, se deberán crear distintos conjuntos de botones de opción. Al igual que en los demás controles, para acceder a este control, es través del botón de la caja de herramientas.

Un botón de opción sólo puede presentar dos posibles valores en la propiedad `Value`: False, que indica que el botón no está activado, y `True`, que representa la situación en el que el botón está activado. El resto de propiedades de este control son similares a otros controles ya mencionados.

Es importante utilizar los botones de opción sólo en aquellos casos en el que el número de opciones posibles sea fijo a lo largo de la vida de la aplicación. En caso contrario, la inclusión de una opción o modificación de alguna de las existentes podría implicar la modificación de partes de la aplicación que trabajen con dichas opciones, algo no deseable.

3.15.6. Cuadros de lista.

Otra forma de presentar opciones al usuario es a través de una lista donde se sitúen dichas opciones.

De forma predeterminada, las opciones que contiene el cuadro de lista (`ListBox`) se presentan verticalmente en una única columna, aunque también puede establecerse más de una columna. Además, si es necesario, aparecerán barras de desplazamiento. Para dibujar un cuadro de lista en un formulario, utilice el botón.

Entre las propiedades que debe conocer de un cuadro de lista cabe destacar:

- `ListIndex`***:*** indica el índice del elemento seleccionado en la lista. El primer elemento de la lista tiene el valor 0 como índice.
- `ListCount`***:*** indica el número de elementos existentes en la lista en todo momento.
- `Text`***:*** texto del elemento seleccionado en la lista.

Todas estas propiedades son utilizadas en tiempo de ejecución, cuando se interactúa con la lista. En tiempo de diseño, sin embargo, se podrá indicar si desea que la lista se mantenga ordenada.

Para ello se utiliza la propiedad `Sorted` que predeterminadamente está a False, lo que indica que no se mantiene ordenada.

En esta tesis se usarán los cuadros de lista en la ventana principal del programa para seleccionar el método a emplear para realizar los cálculos de la población de proyecto y que al seleccionar uno de ellos nos permita pasar a la ventana de dicho método.

3.16. Fundamentos de programación.

La forma adecuada de programar una aplicación en este entorno de desarrollo es la siguiente: crear la interfaz, establecer propiedades y finalmente escribir código.

En esta parte se introduce formalmente las variables, constantes, procedimientos, funciones, estructuras de control, tipo de datos, etc.

3.16.1. Variables.

Una variable es una ubicación temporal de memoria donde el programador almacena datos que le interesa retener durante la ejecución de la aplicación.

Una variable puede contener texto, un valor numérico, una fecha o una propiedad de cierto objeto. En definitiva, es una forma de dar nombre a una porción de datos con la que deseamos trabajar en nuestro programa.

Las variables se caracterizan por un nombre que las identifica y por un tipo de datos, que establece el conjunto de valores posibles que pueden contener y operaciones en las que puede participar. El valor de una variable puede cambiar a lo largo de la vida de ésta (de ahí el nombre de variable).

En la mayoría de lenguajes de programación es necesario declarar una variable para poder utilizar en el programa. Sin embargo en Visual Basic esto no es obligatorio aunque sí recomendable.

Al declarar una variable se reserva a memoria para ella y se le indica a Visual Basic qué valores puede contener a través de su tipo de datos. Así Visual Basic podrá realizar comprobación de tipos y establecer errores en tiempo de compilación.

Si se desea establecer la declaración de una variable antes de ser utilizada en el código, se puede indicar en la ficha Editor del cuadro de diálogo Opciones en el menú Herramientas.

Al activar la casilla Declaración de variables requerida, Visual Basic introduce la instrucción `Option Explicit` en la sección de declaraciones de cada módulo nuevo que se cree, no de los ya existentes donde se tendrá que introducirla manualmente. Esta instrucción obligará a declarar las variables antes de utilizarlas, lo que es muy recomendable.

Se utilizarán variables por diversos motivos: almacenar una entrada del usuario, realizar cálculos intermedios, que será la razón de mayor importancia para la presente tesis, y establecer la salida en un formulario, etc.

Además, el uso de variables puede hacer más rápida la aplicación. Así, si se utiliza en muchas ocasiones un determinado resultado, es mejor guardarlo en una variable y utilizarla que tener que volver a establecer dicho resultado, lo que puede implicar distintos cálculos.

Por otra parte, es más rápido el acceso a una variable que a una propiedad de un objeto, por lo que en muchas ocasiones se guardará el valor de una propiedad utilizando una variable.

Por ejemplo:

```
Dim Variable
Variable= txtEntrada.Text
txtSalida.Text=Variable
```

La forma de declarar una variable es a través de la instrucción `Dim`. En la primera línea se está declarando la variable de nombre Variable y al mismo Visual Basic guarda espacio en memoria para poder utilizarla.

En la segunda línea ya utiliza la variable. En este caso sirve para guardar el valor que existe en un cuadro de texto llamado `txtEntrada` (representado por su propiedad Text).

En la tercera línea se está haciendo justo lo contrario. Utiliza la variable para establecer el valor de la propiedad `Text` del cuadro de texto `txtSalida`.

3.16.2. Tipos de datos.

El tipo de datos de una variable establece el conjunto válido de valores que ésta puede tomar, así como el conjunto de operaciones en las que puede tomar parte como operando.

En Visual Basic dicho tipo de datos se especifica al declarar la variable de forma que se guarda espacio en memoria para poder almacenar los valores de dicho tipo de datos.

Es importante indicar el tipo de datos ya que no todos tienen la misma representación en memoria, ocupando distinto espacio físico.

Sin embargo, de forma predeterminada y si no se indica el tipo de datos, Visual Basic establece el tipo `Variant` para todas las variables. Así, al utilizar una instrucción como `DimNombreVariable`, se está especificando implícitamente el tipo `Variant` para dicha variable.

Esto también se aplica cuando se utilizan las variables sin haberlas declarado previamente.

El tipo `Variant` es un tipo especial de datos que puede contener cualquier clase de datos excepto cadenas de longitud fija y tipos definidos por el usuario. Al utilizar variables de este tipo, no se tiene que preocupar de efectuar conversiones entre tipos para utilizarlas en distintos contextos, Visual Basic lo hace por usted.

La flexibilidad del tipo `Variant` se tiene que pagar con el aumento en el consumo de memoria y disminución de velocidad que implica dicho tipo de datos. Visual Basic escogerá automáticamente la representación en memoria más eficiente para dicha variable, guardando la memoria necesaria, cuando se le asigne un valor.

Esta representación puede variar a lo largo de la vida de la variable al ir asignando valores distintos.

Por ello, es responsabilidad del programador establecer siempre que sea posible un tipo para cada variable que utilice. La forma de hacerlo es indicándolo al final de la instrucción `Dim` de la forma:

```
Dim Nombre Variable As Nombre Tipo.
```

A continuación se indican los principales tipos de datos disponibles en Visual Basic de los cuales más adelante se mencionara cuales se emplearan y la razón:

Nombre del tipo de datos	*Tamaño de almacenamiento*	*Valores posibles*
Entero (`integer`)	2 bytes	-32.768 a 32,767
Entero largo (`Long`)	4 bytes	-2. 147.483.648 a 2. 147.483.647
Simple (`Single`)	8 bytes	-1.79769313486231 E308a -4.94065645841247F-324 para valores negativos.4.940656458412-47E-324a 1,79769313486232E308 para valores positivos.
Fecha (`Date`)	8 bytes	Del 1 de Enero de 100 al Fecha (Date) 8 bytes 31 de Diciembre de 9999.
Objeto (`Object`)	4 bytes	Cualquier referencia a un Objeto.
`Variant`	16 bytes+1 byte por cada carácter	Nulo, Error, cualquier valor Numérico de rango Doble, o cualquier texto de

3.16.3. Constantes.

S i en una aplicación se encuentra un valor que se repite frecuentemente, es posible que se guarde en una constante.

Las constantes son semejantes a las variables, pero con la particularidad de que su valor no puede cambiar a lo largo de la aplicación. Una constante es utilizada para aumentar la legibilidad del código. Por ejemplo en lugar de utilizar el valor 3.1416 se podría utilizar la constante Pi.

En definitiva se deberá darle un nombre a la constante y establecer su valor, que no podrá ser cambiado en el resto de código.

Para utilizar una constante, se deberá declararla previamente, en este caso sí que es obligatorio. La forma de declararla es a través de la instrucción `ConstNombreConstante=Expresión`, donde Expresión será un valor literal (número, texto, fecha, etc.) o un conjunto de palabras que se evalúen a un valor válido.

Por ejemplo:

```
Const FestivosSemana = 2
```

A partir de ahora podrá utilizar el identificador `FestivoSemana` en lugar del valor 2. Al hacer esto se podrá añadir legibilidad a los programas, además de permitir la modificación de opciones globales de una forma más sencilla.

Así, por ejemplo, si se quiere cambiar el número de días festivos en una semana para una aplicación, sólo se necesitará modificar el valor en la declaración de la constante y esta modificación se extenderá a todos los lugares donde se use.

Una constante puede ser utilizada en cualquier lugar donde su valor sea adecuado.

3.16.4. Operadores de Visual Basic.

En Visual Basic existe un gran número de operadores que se pueden utilizar para crear formulas de cierta complejidad.

Estos operadores trabajan con los operados sobre los que se aplican siempre que el valor que posean sea compatible con el operador. Hay que tener cuidado en este aspecto ya que Visual Basic realiza conversiones de tipo implícitas cuando lo necesita, produciendo, en ocasiones, resultados inesperados.

Los operadores más utilizados en una aplicación Visual Basic son los siguientes y que serán utilizados en la presente tesis son los siguientes:

Operador	*Operación que realiza*
+	Suma / Concatenación de cadenas de caracteres.
-	Resta.
*****	Multiplicación.
/	División.
****	División entera.
Mod	Resto de la división entera.
^	Exponenciación.
&	Concatenación de cadenas de caracteres.

El operador + es un caso especial. Dicho operador puede actuar como operador de texto y como operador numérico, efectuando operaciones distintas (se dice que el operador está sobrecargado).

En el primer caso realiza lo que se llama concatenación de cadenas, es decir, crea una cadena nueva al situar la segunda cadena dada inmediatamente después de la primera.

Por ejemplo si realiza la operación "hasta " + “luego", el resultado sería "hasta luego" (obsérvese que la primera cadena contenía espacio en blanco al final).

En el caso de contener valores numéricos, + actúa como el operador suma normal. Es decir una operación del tipo 1 +2 daría como resultado 3.

Por ello es conveniente utilizar el operador & para concatenar cadenas de caracteres en lugar del operador +.

Visual Basic realiza conversiones implícitas de tipo para poder utilizar un determinado operador siempre que los operadores tengan una representación válida para este operando.

3.16.5. Estructuras de decisión.

El lenguaje de Visual Basic incorpora estructuras de control que permiten controlar el flujo de ejecución de un programa.

Si no existen tales estructuras en su código, éste se ejecutará de izquierda a derecha y de arriba a abajo según se haya escrito.

Sin embargo, en muchas ocasiones se desea ejecutar una línea u otra dependiendo de cierta condición o bien, repetir un número de veces el mismo conjunto de líneas. En estos casos la introducción de las estructuras de control modifica el flujo normal de la ejecución.

La instrucción `If-Then-Else` es la estructura clásica de decisión y presenta la siguiente sintaxis:

```
If condición1 Then
[instrucciones1]
[ElseIf condición2 Then
[instrucciones2]]
…
[Else
[instruccionesN]]
End If
```

Donde los corchetes ([ ]) representan partes opcionales de la instrucción. Además es posible que exista más de una cláusula `ElseIf` en la misma instrucción `If-Then-Else`, de ahí la aparición de los puntos suspensivos (…).

Se debe entender esta instrucción de la siguiente forma: Si condición1 se cumple entonces se ejecuta el bloque instrucciones1, en caso contrario se ejecutaría el bloque de instrucciones (2…N-1) de la primera cláusula `ElseI`f cuya condición se cumpla. Si finalmente no se cumple ninguna de estas condiciones, se ejecutaría el bloque instrucciones N correspondiente a la cláusula `Else`.

Y donde condición es una expresión, es decir, cualquier conjunto de palabras, que se evalúan a Verdadero o Falso, sin posibilidad de poder tener otro valor. Se dice que una condición se cumple cuando se evalúa a Verdadero y que fracasa en caso contrario. Las condiciones también se conocen como expresiones lógicas.

Cuando existe un gran número de condiciones evaluar, es posible que la instrucción `If-Then-Else` se complique de gran manera. En estos casos Visual Basic permite el uso de otra estructura de decisión: la instrucción `Select Case`.

`Select Case` no incorpora mayor potencia al lenguaje, pero hace que el código sea más legible y eficiente. La sintaxis de la instrucción `Select Case` es la siguiente:

```
Select Case expresiónDeComparación
[Case listaExpresiones1]
[insrucciones1]
[Case Else
[instruccionesN]]
End Select
```

Este tipo de estructuras de decisión serán usadas frecuentemente para establecer rangos en los valores que el usuario ingrese al programa.

3.16.6. Expresiones lógicas.

El lenguaje de Visual Basic incorpora una serie de operadores de comparación que son útiles a la hora de establecer condiciones, los cuales son los siguientes:

=	**Igual a**
<>	Distinto a
>	Menor que
<	Mayor que
<=	Menor o igual que
>=	Mayor o igual que

Además de esto, se pueden utilizar los llamados operadores lógicos: `And, Or, Not, Xor`. Estos permiten establecer condiciones que dependan de más de un criterio de selección.

Todos los operadores mencionados hacen que la expresión en la que se encuentren se evalúe Verdadero o a Falso, sin posibilidad de cualquier otro valor.

3.16.7. Estructuras de repetición.

Otro tipo de estructuras que pueden modificar el flujo de ejecución son las estructuras de repetición, también llamadas bucles. Estas estructuras sirven para repetir una y otra vez un conjunto de instrucciones.

En este sentido puede querer repetir un número determinado de veces el conjunto de instrucciones o no conocer dicho número y desear repetirlas hasta que se cumpla cierta condición (o mientras se cumpla otra).

En definitiva existirán dos tipos de estructuras de repetición: aquellas en las que se conoce el número de repeticiones y aquellas que dicho número se establece en la propia ejecución.

La estructura de repetición `For ... Next` es adecuada cuando se conoce el número de veces que debe repetirse un conjunto de instrucciones y deseamos disminuirla cantidad de código escrito.

La sintaxis completa de esta estructura de repetición es la siguiente:

```
For contador = principio To fin [Step incremento]
[Instrucciones]
[ExitForl
[Instrucciones]
Next [contador]
```

Donde contador es el nombre de una variable que sirve como contador de las veces que se ha de ejecutar el bucle. A dicha variable se le asigna un valor inicial y se indica el valor final o aquel en el que, una vez superado, el bucle no volverá a repetirse.

En el cuerpo del bucle se sitúan el conjunto de instrucciones que deben ejecutarse cada vez, existiendo la posibilidad de introducir una instrucción `Exit For` para salir del bucle antes de que se repita el número de veces establecido.

Finalmente, una vez ejecutada cada una de las instrucciones del cuerpo del bucle se ejecuta la línea `Next` contador, en la que se aproxima el valor del contador hacia el valor final en el número de unidades indicado en el incremento.

Aproximarse puede ser incrementar el valor, si el valor inicial es menor que el valor final, o decrementarlo, si el valor inicial es mayor que el final. Si no se determina ningún incremento, Visual Basic toma como valor predeterminado una unidad.

En muchas ocasiones se desconoce cuántas veces debe repetirse un bucle. En estos casos se deberá utilizar una estructura de repetición que dependa de una condición.

En este sentido se puede repetir un bucle mientras se cumpla o hasta que se cumpla una condición. En ambas situaciones Visual Basic presenta estructuras adecuadas.

La estructura de repetición genérica es `Do...Loop`. Dicha estructura es utilizada cuando no conocemos cuantas veces debe ejecutarse el bucle.

La sintaxis de esta instrucción es:

```
Do [{While / Until} condición
[Instrucciones]
[ExitDo]
[Instrucciones]
Loop
Puede usar esta sintaxis válida equivalente:
Do
[Instrucciones]
[Exit Do]
Instrucciones]
Loop [{While / Until} condición]
```

Se puede usar la palabra `While` o bien la palabra `Until`. En el primer caso el bucle se repite mientras la condición se cumpla; en el segundo caso el bucle se repetirá hasta que la condición dé el valor Verdadero.

Por otra parte en la primera sintaxis, la condición es comprobada al principio del bucle, por lo que si no se cumple al iniciarse, entonces el cuerpo del bucle no se ejecutará. Sin embargo, en la segunda sintaxis, la condición es comprobada a la salida del bucle por lo que por lo menos una vez es seguro que el cuerpo del bucle se va a ejecutar.

Este tipo de estructuras de repetición serán muy empleadas en los métodos que involucren el uso de un indeterminado numero de datos para realizar los cálculos, como se vera en algunos de los códigos.

3.16.8. Procedimientos.

Los procedimientos son unidades de código en las que se puede dividir un programa para facilitar la programación. Un procedimiento es un conjunto de líneas de código que deben realizar operaciones o tareas bien definidas.

En Visual Basic existen cuatro tipos de procedimientos: `Sub, Function, Property y Event.`

Los procedimientos Sub ya se trataron con anterioridad, al haber trabajado con los procedimientos de evento que son de este tipo.

Los procedimientos `Function` se distinguen de los primeros en que siempre devuelven un valor asociado al nombre del procedimiento. Este valor puede ser usado en el código que lo ha llamado.

Finalmente los procedimientos `Property` sirven para crear y manipular propiedades personalizadas de los objetos y los `Event` para declarar eventos definidos por el usuario en un módulo de clase.

3.17. Efectos gráficos.

3.17.1. Control Line.

La incorporación de elementos gráficos a las aplicaciones, harán que éstas sean más agradables a los usuarios.

La impresión externa de la aplicación es muy importante, sobre todo al estar inmersa en un sistema operativo cuya interfaz de usuario es gráfica.

Visual Basic incorpora una serie de controles gráficos que permiten la incorporación de elementos de este tipo a los formularios.

Si se desea agregar líneas a los formularios, Visual Basic facilita el control Line en la caja de herramientas.

Este control tiene pocas propiedades, pudiendo establecer el grosor mediante la propiedad `Border With` o el color de la misma mediante `BorderColo`r.

También se podrá cambiar el estilo de línea, de forma que aparezca discontinua o a puntos mediante la propiedad `BorderStyle`.

El resto de propiedades son poco interesantes. En `(X1, Y1`) tiene el punto inicial y en (`X2, Y2`) el punto final de la línea.

3.17.2. Control `Shape`.

Visual Basic permite dibujar distintas formas a través del control `Shape` (Forma). Utilizando dicho control se puede dibujar un círculo, un óvalo, un rectángulo, un cuadrado, etc.

Se pueden dibujar formas en los formularios a través del botón de la caja de herramientas. La apariencia inicial del objeto `Shape` es la de un rectángulo. Dicha característica se establece a través de la propiedad `Shape`.

Utilizando la propiedad `Shape` se podrán dibujar cinco formas distintas: rectángulo (`Rectangle`), cuadrado (`Square`), óvalo (`Oval`), círculo (`Circle`), rectángulo con esquinas redondeadas (`Rounded Rectangle`) y cuadrado con esquinas redondeadas (`Rounded Square`).

3.18. Depurar la aplicación.

Al hablar de la depuración de una aplicación, nos referimos a la búsqueda, localización y corrección de los errores que ésta presente.

En toda aplicación se pueden encontrar tres tipos de errores: errores en tiempo de compilación, errores en tiempo de ejecución y errores lógicos.

Los errores en tiempo de compilación suelen ser los más sencillos de solucionar. Se deben, en la mayoría de casos, a escribir incorrectamente el nombre de alguna variable o a utilizar propiedades y métodos sobre objetos que no los poseen.

En la ficha Editor del cuadro de diálogo Opciones (menú Herramientas) se puede observar la presencia de la casilla Comprobación automática de sintaxis Si dicha casilla está activada, Visual Basic estará “pendiente” del código que escriba, avisándonos de errores sintácticos.

Otro tipo de errores que pueden suceder al programar una aplicación son los errores en tiempo de ejecución. Dichos errores se producen cuando Visual Basic encuentra una instrucción en el código que realiza una operación imposible de ejecutar.

La instrucción puede estar correctamente escrita según la sintaxis del lenguaje, por lo que no se ha detectado en tiempo de compilación y, sin embargo, provocar un error en tiempo de ejecución.

Los errores más difíciles de depurar son los errores lógicos. Estos errores ocurren cuando una aplicación no funciona de la forma que esperamos. No se produce ningún error detectable por Visual Basic, sino que simplemente el resultado no es el que pretendemos que se produzca.

Se deberán utilizar las herramientas de depuración de Visual Basic para seguir el código al ejecutarse y, de esta forma, localizar el error que produce que el resultado no sea el esperado.

3.19. Finalizar la aplicación.

Una vez que ha concluido el diseño de la aplicación y depurados los errores que se han podido detectar, es el momento de realizar unos cuantos pasos necesarios para finalizar la aplicación.

Entre estos pasos cabe destacar: compilar completamente la aplicación, crear un archivo ejecutable y preparar los medios de distribución para la instalación de la aplicación en un equipo que ejecute Windows 95/98 o posterior.

3.19.1. Creación del ejecutable.

Toda aplicación Windows será ejecutada al hacer doble clic en un determinado archivo ejecutable o a través de un acceso directo a dicho archivo.

Cuando se distribuya la aplicación al usuario final, éste debe ejecutarla como una aplicación más de Windows, por lo que es necesario crear un archivo ejecutable.

Visual Basic permite la creación de archivos ejecutables a través de la opción Generar proyecto EXE... del menú Archivo, donde el proyecto aparece como el nombre del proyecto ejecutable.

Al crear el archivo ejecutable, Visual Basic compila completamente el proyecto, es posible que se encuentren nuevos errores de compilación.

Será necesario especificar el nombre que se desea dar al archivo ejecutable de la aplicación y la ubicación de éste. Además, a través del botón Opciones, se podrá establecer ciertos valores descriptivos sobre la versión del archivo ejecutable.

En la sección Aplicación, se puede indicar el nombre de la aplicación y especificar un icono identificativo de la misma. Este icono será aquel que se utilice cuando la ventana principal de la aplicación sea minimizada.

Finalmente, se puede agregar información descriptiva de la aplicación en la sección Información de versión.

Una vez creado el archivo ejecutable, se puede utilizar como cualquier otro archivo de estas características. Haciendo doble clic sobre él, por ejemplo desde el Explorador de Windows y la aplicación debe iniciarse, independientemente de Visual Basic.

3.19.2. Asistente de instalación.

Una vez a que se ha creado el archivo ejecutable de la aplicación Windows y habiendo comprobado que funciona como cualquier otro archivo ejecutable, es el momento de preparar la distribución de la aplicación.

Para distribuir la aplicación al usuario final, se necesita crear un programa de instalación que se encargue de copiar los archivos necesarios en el equipo del usuario.

En este caso es muy posible que el usuario final no posea Visual Basic por lo que se tendrá que distribuirle todos los ficheros necesarios para ejecutar la aplicación.

Visual Basic incorpora un asistente que facilita la creación del programa de instalación, preguntando en cada uno de los pasos la información que se necesite.

El Asistente de empaquetado y distribución creará los discos u otros medios de distribución con los archivos necesarios para que instale la aplicación correctamente, además de comprimirlos con el objeto de que ocupen menos espacio en su medida de distribución.

En este caso los archivos serán incluidos en un CD junto con todos los ficheros necesarios para su instalación y correcto funcionamiento.

Con la creación de este asistente se obtienen los siguientes beneficios:

- Se crea un programa de instalación de nombre SETUP .EXE que utilizan los usuarios finales para instalar la aplicación en su equipo. Este programa realizará las preguntas necesarias al usuario para instalar en el directorio donde desee hacerlo.
- Genera un archivo ejecutable actualizado.
- Comprime los archivos de forma que ocupen menos espacio en el medio de distribución.
- Informa del número de discos si éste es el medio de distribución y del espacio libre que necesitarán tener el usuario en su disco duro para la instalación.
- Crea un elemento en el menú Inicio que le permite al usuario acceder a la aplicación.
- Registra la aplicación en el Registro de Windows.
- Permite la desinstalación de la aplicación a través del elemento Agregar o quitar programas del Panel de control de Windows.

Para facilitar el uso del programa, solo se creara el ejecutable con la extensión .exe para que este pueda ser usado por los usuarios con solo copiarlo y pegarlo en el escritorio.

4. Desarrollo de los módulos en entorno de Visual Basic.

Después de haber descrito Visual Basic y hablar acerca de la mayoría de las herramientas a emplear para la creación del programa que permita calcular la población proyecto y haber comentado ya algunos de los métodos, y opciones así como haber escrito ya algunas de las sintaxis a emplear daremos comienzo con el diseño, programación, compilación y ejecución del programa.

La mejor base para estimar las tendencias de la población futura de una comunidad es su pasado desarrollo, y la fuente de información mas importante sobre el mismo en México son los censos levantados por el Instituto Nacional de Estadística, geografía e Informática cada 10 años. Los datos de los censos de población pueden adaptarse a un modelo matemático como son el aritmético, geométrico, parabólico, etc. Estos y otros modelos son la base para el desarrollo de la aplicación propósito de esta tesis y que se describen a continuación.

4.1. Estructura general de los módulos.

Como se menciono anteriormente existen diversos métodos para calcular de manera aproximada la población futura de una comunidad por cual razón, se diseñara una aplicación que envuelva todos estos métodos es decir, la aplicación tendrá una ventana principal que permita a los usuarios decidir de acuerdo a las características particulares del proyecto el método que empleara.

Esta ventana contendrá en una ventana principal la lista de los métodos que se describirán extensamente a continuación.

Como en el capitulo anterior ya se describieron los procedimientos generales a seguir y los métodos posibles para realizar la aplicación, se mencionaran únicamente los pasos que se siguen para el diseño y posterior escritura del código.

Después de haber instalado en la computadora Visual Basic se procede a ejecutarlo desde el menú inicio. Una vez abierto se muestra la ventana de Nuevo Proyecto en la cual seleccionamos la opción EXE estándar y damos Abrir como se menciono anteriormente.

4.2. La ventana principal del programa.

Posteriormente se realiza el diseño de la ventana principal la cual por ser un formulario la llamaremos frmprincipal esto se hace en la ventana de propiedades al igual que las demás propiedades que se modificaran que se describirán a continuación:

`Caption` = JLProyect V1, que es el nombre asignado a la aplicación.

`Icon` = Se selecciono la imagen que se desea aparezca en la paleta de la ventana la cual fue seleccionada y cambiada al formato .ico

`Picture` = Se selecciono la imagen de fondo de la ventana, debe estar en formato mapa de bits.

El tamaño de la ventana no podrá ser modificado por el usuario y se determino de acuerdo al espacio necesario para encerrar todas las herramientas y botones. Se selecciono del menú herramientas la opción editor de menús para agregar los menús necesarios para guardar, abrir, imprimir, etc. necesarios para el buen funcionamiento de la aplicación.

El diseño general del formulario principal se muestra en la imagen siguiente:

Además se agrego un `ComboBox` que permite seleccionar el método a emplear y pasar al formulario de ese método. Se agregaron las etiquetas o `Label` con el nombre de la aplicación como imagen de fondo y dar una mejor apariencia a la ventana al momento de su ejecución.

Para escribir el código para este formulario solo basta dar un doble click al botón Siguiente para pasar a la ventana de código.

El código para este formulario es el siguiente:

```
'Código para el ComboBox, que dependiendo del método  a
emplear, nos pase a la ventana de dicho método.
Private Sub comboseleccionar_click()
m = comboseleccionar.ListIndex
Select Case m
Case 0
frmaritmetico.Visible = True
Case 1
```

```
frmgeometrico.Visible = True
Case 2
frmgeometricodecreciente.Visible = True
Case 3
frminterescompuesto.Visible = True
Case 4
frmlogistico.Visible = True
Case 5
frmincrementos.Visible = True
Case 6
frmparabola.Visible = True
Case 7
frmminimos.Visible = True
End Select
End Sub

'Codigo para mostrar la ventana de ayuda, que en este caso se
mostrara desde un PDF.
Private Sub mnuayuda2_Click()
frmayuda.Visible = True
On Error Resume Next
Shell "C:\Program Files\Adobe\Reader 6.0\Reader\AcroRd32 " &
"C:\JLProject\Ayuda de JL Project.pdf"
Shell "C:\Program Files\Adobe\Reader 7.0\Reader\AcroRd32 " &
"C:\JLProject\Ayuda de JL Project.pdf"
Shell "C:\Program Files\Adobe\Reader 8.0\Reader\AcroRd32 " &
"C:\JLProject\Ayuda de JL Project.pdf"
Shell "C:\Program Files\Adobe\Reader 9.0\Reader\AcroRd32 " &
"C:\JLProject\Ayuda de JL Project.pdf"
Shell "C:\Program Files\Adobe\Reader 10.0\Reader\AcroRd32 " &
"C:\JLProject\Ayuda de JL Project.pdf"
Shell "C:\Archivos de programa\Adobe\Acrobat
6.0\Acrobat\Acrobat.exe " & "C:\JLProject\Ayuda de JL
Project.pdf"
Shell "C:\Archivos de programa\Adobe\Acrobat
7.0\Acrobat\Acrobat.exe " & "C:\JLProject\Ayuda de JL
Project.pdf"
Shell "C:\Archivos de programa\Adobe\Acrobat
8.0\Acrobat\Acrobat.exe " & "C:\JLProject\Ayuda de JL
Project.pdf"
Shell "C:\Archivos de programa\Adobe\Acrobat
9.0\Acrobat\Acrobat.exe " & "C:\JLProject\Ayuda de JL
Project.pdf"
Shell "C:\Archivos de programa\Adobe\Acrobat
10.0\Acrobat\Acrobat.exe " & "C:\JLProject\Ayuda de JL
Project.pdf"
End Sub
```

```
'Código para mostrar la ventana de presentación del programa.
Private Sub mnuacerca_Click()
frmacerca.Visible = True
End Sub

'Código para el uso de los menús, desde los cuales se puede
pasar a las ventanas de cada método.
Private Sub mnuaritmetico_Click()
frmaritmetico.Visible = True
End Sub

Private Sub mnucuadrados_Click()
frmminimos.Visible = True
End Sub

Private Sub mnudecreciente_Click()
frmgeometricodecreciente.Visible = True
End Sub

Private Sub mnugeometrico_Click()
frmgeometrico.Visible = True
End Sub

Private Sub mnuincrementos_Click()
frmincrementos.Visible = True
End Sub

Private Sub mnuinteres_Click()
frminterescompuesto.Visible = True
End Sub

Private Sub mnulogistico_Click()
frmlogistico.Visible = True
End Sub

Private Sub mnuparabola_Click()
frmparabola.Visible = True
End Sub

Private Sub mnusalir_Click()
End
End Sub
Private Sub cmdsalir_Click()
Unload Me
End
End Sub
```

En este código ya se menciona el nombre que posteriormente le daremos a los formularios de cada uno de los métodos. En este código se hace uso del método `Select Case` que hace que en cada caso que se seleccione cada uno de los `Case "n"` se mostrara el formulario de dicho modulo.

Posterior al hacer click sobre una de las opciones, se pasara a otro formulario según la selección del método, al formulario del método que se halla seleccionado cada uno de los cuales se podrá manejar de manera independiente a los demás formularios.

Nótese que no se incluye el comando ni el botón para salir del formulario, ya que para salir de este se usara el método tradicional de las ventanas de Windows, dando click en el botón X.

4.3. Método aritmético.

El modelo aritmético tiene como característica un incremento de población constante para incrementos de tiempo iguales y, en consecuencia la velocidad de crecimiento, o sea la relación del incremento de habitantes y el periodo de tiempo es una constante; expresado como ecuación se tiene:

$$\frac{dP}{dt} = K_a \quad \text{O bien:} \quad dP = K_a dt \text{ ……..(4.1)}$$

Donde *P* es la población; *t* el tiempo y k_a una constante que significa un incremento de población en la unidad de tiempo (año, decenio, etc.).

Integrando:

$$\int_1^2 dP = \int_1^2 dt$$

$$P_2 - P_1 = K_a(t_2 - t_1) \text{……(4.2)}$$

De la ecuación 4.2 se obtiene K_a:

$$K_a = \frac{P_2 - P_1}{t_2 - t_1} \text{…….(4.3)}$$

Para un momento T cualquiera se tiene la ecuación lineal:

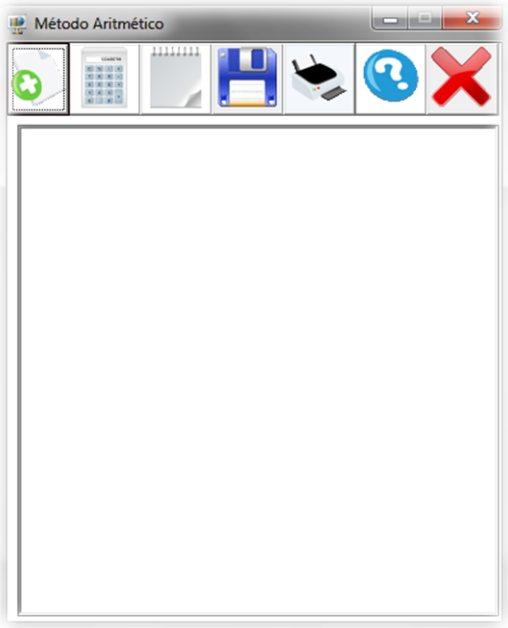
$$P = P_2 + K_a(T - t_2) \text{……(4.4)}$$

Donde el índice “2” se considera para los datos iniciales (P_2, población inicial en el tiempo t_2).

El diseño del formulario y de todos los demás consta de los siguientes objetos:

1) Un `RichTextBox` en donde se ira colocando la información (espacio en blanco).

2) Botón para cargar los datos.



3) Botón para realizar nuevos cálculos con los datos ya cargados.

4) Botón para borrar toda la información.

5) Botón para guardar la información y los cálculos.

6) Botón para imprimir.

7) Botón de ayuda.

8) Botón para cerrar el programa.

Como este método requiere solo de dos datos estadísticos se recomienda utilizar los dos mas recientes, ya que representan la tendencia mas reciente del crecimiento de la población.

El código para cada uno de los objetos queda de la siguiente manera:

```
'Declaración de variables
Dim PrinterRich As Class1
Dim a1 As Double, p1 As Double, a2 As Double, _
p2 As Double, ka As Double, pp As Double, _
añof As Double, nombre As String

'Boton de ayuda
Private Sub cmdayuda_Click()
On Error Resume Next
Shell "C:\Program Files\Adobe\Reader 6.0\Reader\AcroRd32 " &
"C:\JLProject\Ayuda de JL Project.pdf"
Shell "C:\Program Files\Adobe\Reader 7.0\Reader\AcroRd32 " &
"C:\JLProject\Ayuda de JL Project.pdf"
Shell "C:\Program Files\Adobe\Reader 8.0\Reader\AcroRd32 " &
"C:\JLProject\Ayuda de JL Project.pdf"
Shell "C:\Program Files\Adobe\Reader 9.0\Reader\AcroRd32 " &
"C:\JLProject\Ayuda de JL Project.pdf"
Shell "C:\Program Files\Adobe\Reader 10.0\Reader\AcroRd32 " &
"C:\JLProject\Ayuda de JL Project.pdf"
Shell "C:\Archivos de programa\Adobe\Acrobat
6.0\Acrobat\Acrobat.exe " & "C:\JLProject\Ayuda de JL
Project.pdf"
Shell "C:\Archivos de programa\Adobe\Acrobat
7.0\Acrobat\Acrobat.exe " & "C:\JLProject\Ayuda de JL
Project.pdf"
Shell "C:\Archivos de programa\Adobe\Acrobat
8.0\Acrobat\Acrobat.exe " & "C:\JLProject\Ayuda de JL
Project.pdf"
```

```
Shell "C:\Archivos de programa\Adobe\Acrobat
9.0\Acrobat\Acrobat.exe " & "C:\JLProject\Ayuda de JL
Project.pdf"
Shell "C:\Archivos de programa\Adobe\Acrobat
10.0\Acrobat\Acrobat.exe " & "C:\JLProject\Ayuda de JL
Project.pdf"
End Sub
```

Con la instrucción `Shell` indicamos que se debe abrir el archivo indicado en la ruta de la misma línea, la repetición de ello es con la intensión de que se pueda realizar en la mayoría de los equipos con las deferentes manera de tener instalado Acrobat Reader.

```
'Boton para borrar la información cargada y los calculos
realizados
Private Sub cmdborra_Click()
On Error Resume Next
Kill "C:\JLProject\temp\aritmetico.txt"
RichTextBox1.Text = ""
End Sub

'Botón para cargar los datos para comenzar a realizar los
cálculos.
Public Sub cmdcargar_Click()
On Error Resume Next
Kill "C:\JLProject\temp\aritmetico.txt"
nombre = InputBox("¿Cual es el nombre del Registro? ",
"Nombre")
a1 = Val(InputBox("¿Cual es el primer año de registro?",
"Primer año"))
p1 = Val(InputBox("¿Cual es la población para el primer año?",
"Población"))
a2 = Val(InputBox("¿Cual es el segundo año de registro?",
"Segundo año año"))
p2 = Val(InputBox("¿Cual es la población para el segundo
año?", "Población"))
ka = (p2 - p1) / (a2 - a1)
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
pp = p2 + (ka) * ((añof) - (a2))
Open "C:\JLProject\temp\aritmetico.txt" For Append As #1
Print #1, ""
Print #1, , ""
Print #1, "Fecha: " & Format(Date, "Long Date")
Print #1, "Hora:  " & Format(Time, "HH:MM:SS AMPM")
Print #1, ""
Print #1, "Nombre del proyecto: " & nombre
Print #1, ""
Print #1, "Datos Estadisticos:"
```

```
Print #1, "_____"
Print #1, "Año", "Población"
Print #1, "_____"
Print #1, a1, Format(p1, "#,##0")
Print #1, a2, Format(p2, "#,##0")
Print #1, "_____"
Print #1, ""
Print #1, "La población de proyecto para el año"; añof; "es de
"; Format(pp, "#,##0"); " habitantes."
Print #1, ""
Close #1
Open "C:\JLProject\temp\aritmetico.txt" For Input As #1
While Not EOF(1)
Line Input #1, Linealeida
RichTextBox1.Text = RichTextBox1.Text & Linealeida & Chr(13) &
Chr(10)
Wend
Close #1
Exit Sub
End Sub

'Botón para realizar nuevos cálculos con la información
cargada.
Public Sub cmdcalcula_Click()
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
pp = p2 + (ka) * ((añof) - (a2))
Open "C:\JLProject\temp\aritmetico.txt" For Append As #1
Print #1, Format(Time, "HH:MM:SS AMPM")
Print #1, "La población de proyecto para el año"; añof; "es de
"; Format(pp, "#,##0"); " habitantes."
Print #1, ""
Close #1
RichTextBox1.Text = ""
Open "C:\JLProject\temp\aritmetico.txt" For Input As #1
While Not EOF(1)
Line Input #1, Linealeida
RichTextBox1.Text = RichTextBox1.Text & Linealeida & Chr(13) &
Chr(10)
Wend
Close #1
End Sub

'Botón para guardar los cálculos realizados, junto con toda la
información agregada en dicho proyecto, esto se guardara en
la carpeta indicada en el código.
Public Sub cmdguardar_Click()
```

```
Open "C:\JLProject\Proyectos guardados\Metodo Aritmetico.txt"
For Append As #1
Print #1, "-----------------------------------------------"
Close #1
Open "C:\JLProject\temp\aritmetico.txt" For Input As #1
While Not EOF(1)
Line Input #1, Linealeida
Open "C:\JLProject\Proyectos guardados\Metodo Aritmetico.txt"
For Append As #2
Print #2, Linealeida
Close #2
Wend
Close #1
MsgBox ("El proyecto se ha guardado con exito en
C:\JLProject\Proyectos guardados\Metodo Aritmetico.txt")
End Sub

'Botón para mandar imprimir en la impresora predeterminada del
equipo en donde se use el programa.
Private Sub cmdimprimir_Click()
On Error Resume Next
Set PrinterRich = New Class1
With PrinterRich

'Encabezado y pie de página
        .Header = "JLProject - Método Aritmético"
        .Footer = ""
'Margenes
        .MarginTop = 1000
        .MarginLeft = 1000
        .MarginRight = 1000
        .MarginBottom = 1000
        .Imprimir RichTextBox1
      End With
End Sub
```

Básicamente lo que hace el programa en este método y en todos los otros métodos, como se indica en el código es programar las formulas obtenidas del libro Abastecimiento de Agua Potable de la UNAM y crear un archivo de forma temporal en formato .Txt en la carpeta “C:\JLProject\temp\aritmetico.txt" y en el ir escribiendo toda la información que se valla proporcionado a los métodos mediante los cuadros de dialogo y con ella realizar los cálculos necesarios y posteriormente leer ese archivo temporal, copiar la información y mostrarla en la zona te trabajo de la ventana que en este caso es un `RichTextBox` y que se actualiza cada vez que se realiza un determinado calculo.

Al iniciar en este y todos los demás métodos, el programa pedirá un nombre para el proyecto, el cual puede ser el nombre de la comunidad o población en donde se valla a

realizar la obra, se recomienda sea de fácil reconocimiento para que sea identificado posteriormente.

El programa asignara automáticamente la fecha y la hora en que se realizaron los cálculos y el nombre asignado a dicho calculo.

Al salir de la aplicación o al pulsar el botón borrar este archivo temporal se elimina y se vuelve a crear en el momento en que se vuelve a comenzar a cargar los datos para realizar un nuevo calculo. Este es el mismo procedimiento para todos los métodos y por ello nos limitaremos a ya no seguir describiéndolo para no ser repetitivos.

El modulo Class1 fue copiado de la ayuda de Visual Basic MSDN Library, ya que es de muy difícil programación y con el se puede mandar imprimir a cualquier tipo de impresora y papel deseado, siendo el tamaño carta el recomendado para este caso.

```
'Botón para salir de la ventana correspondiente a este método
y pasar a la ventana principal.
Private Sub cmdsalir_Click()
On Error Resume Next
Kill "C:\JLProject\temp\aritmetico.txt"
Unload Me
End Sub

'hacer que cuando se cierre la aplicación se borre todos los
archivos generados para usarse de forma temporal.
Private Sub Form_Unload(Cancel As Integer)
On Error Resume Next
Kill "C:\JLProject\temp\aritmetico.txt"
End Sub
```

Con este código lo que hace el programa es que al comenzar, el usuario de un click en el botón  (cargar datos) para comenzar a darle al programa la información necesaria para realizar los cálculos. Para este caso en necesario únicamente proporcionar dos datos estadísticos que son las poblaciones obtenidas de algún censo, es decir el programa peguntara el primer año de registro que es una fecha y su población y posteriormente el segundo año de registro o la otra fecha y la población en esa fecha. Al terminar de cargar estos datos el programa pedirá de manera automática una fecha a calcular, en la cual debemos de introducir el año del que se desea obtener la población de proyecto.

Cabe destacar que los comandos “Cargar datos” y “calcular” se declararon en este y todos los demás métodos como públicos, ya que de esta manera los otros botones de comando podrán hacer uso de la información que en ellos se proporciono al programa, es como una forma de entrelazar y permitir que uno use la información del otro.

Si se desea realizar uno o varios cálculos para otra fecha en especial solo se tendrá que dar un click en el botón  (Calcular). Esta acción se puede realizar tantas veces sea necesario, siempre y cuando se tengan cargados los datos.

Una vez cargados los datos y realizados los cálculos, se puede guardar esta información en formato .Txt dando click en el botón (guardar) lo cual generará un archivo en un bloc de notas y se guardara de manera automática en la carpeta "C:\JLProject\Proyectos guardados\Metodo Aritmetico.txt". De la misma forma si se desea imprimir la información bastará con dar click en el botón (imprimir) y se mandara automáticamente la información a la impresora predeterminada del equipo en uso.

Si la información cargada es errónea por cualquier razón o simplemente se desea eliminar para introducir un nuevo proyecto de debe de borrar la información cargada para ello basta con dar click en el botón (borrar) y se eliminará toda la información cargada así como los cálculos realizados quedando en blanco la zona de trabajo de la ventana.

Si se tiene alguna duda de como usar el programa en tiempo de ejecución o de uso, se puede recurrir a la ayuda dando click en el botón (ayuda), para lo cual el equipo en donde se este trabajando deberá tener instalado de forma correcta Adobe Reader 6 hasta 10 para poder hacer uso de la ayuda.

Los botones de calcular, ayuda, guardar, imprimir y borrar trabajan de la misma manera en todos los métodos de este programa, por lo que nos limitaremos a hablar solo del botón (cargar datos) que este varia un poco en algunos de los métodos como se ira viendo en cada uno de ellos.

4.4. Método geométrico.

El modelo geométrico de crecimiento de población se caracteriza por tener una velocidad de crecimiento directamente proporcional al valor de la población en cada instante de tiempo, es decir:

$$\frac{dP}{dt} = K_G P \quad \text{Ó} \quad \frac{dP}{P} = K_G dt \quad \text{.....(4.5)}$$

Donde K_G es la velocidad de crecimiento cuando la población P es la unidad.

Integrando la ecuación 4.5 se obtiene:

$$\int_1^2 \frac{dP}{P} = K_G \int_1^2 dt$$

$$LnP_2 - LnP_1 = K_G(t_2 - t_1) \text{ .....(4.6)}$$

$$K_G = \frac{LnP_2 - LnP_1}{t_2 - t_1} \text{ .....(4.7)}$$

Para un momento T cualquiera:

$$LnP = LnP_2 + K_G(T - t_2) \text{.....(4.8)}$$

Como se menciono en el método aritmético, solo se hablara del botón de comando “cargar datos” y en algunos casos el de “calcular”, ya que los demás funcionan de la misma manera que en el método aritmético y que se detallaron en su momento. Es por ello que también se hablara solo del código y funcionamiento de dicho botón de comando.

La ventana para este método es la siguiente:

Como se puede observar, es idéntica a la del método aritmético y su funcionamiento es también idéntico, solo cambia el código para los botones cargar datos y calcular ya que de ellos depende la forma de procesar los datos y realizar los cálculos, el código es el siguiente:

```
'Declaración de variables:
Dim PrinterRich As Class1
Dim a1 As Double, p1 As Double, _
a2 As Double, p2 As Double, _
k As Double, pp As Double, añof As Double, _
lnpf As Double, nombre As String
'Codigo para el botón cargar.
Public Sub cmdcargar_Click()
On Error Resume Next
Kill "C:\JLProject\temp\geometrico.txt"
nombre = InputBox("¿Cual es el nombre del Registro? ", "Nombre")
a1 = Val(InputBox("¿Cual es el primer año de registro?", "Primer
año"))
p1 = Val(InputBox("¿Cual es la población para el primer año?",
"Población"))
a2 = Val(InputBox("¿Cual es el segundo año de registro?", "Segundo
año año"))
```

```
p2 = Val(InputBox("¿Cual es la población para el segundo año?",
"Población"))
ka = (p2 - p1) / (a2 - a1)
Open "C:\JLProject\temp\geometrico.txt" For Append As #1
Print #1, ""
Print #1, , ""
Print #1, "Fecha: " & Format(Date, "Long Date")
Print #1, "Hora:  " & Format(Time, "HH:MM:SS AMPM")
Print #1, ""
Print #1, "Nombre del proyecto: " & nombre
Print #1, ""
Print #1, "Datos Estadisticos:"
Print #1, "_____________________________________________"
Print #1, "Año", "Población"
Print #1, "_____________________________________________"
Print #1, a1, Format(p1, "#,##0")
Print #1, a2, Format(p2, "#,##0")
Print #1, "_____________________________________________"
Print #1, ""
añof = Val(InputBox("¿Cual es el año del que se desea conocer la
población?", "Año futuro"))
k = (Log(p2) - Log(p1)) / (a2 - a1)
lnpf = Log(p2) + k * (añof - a2)
pp = Exp(lnpf)
Print #1, "Poblaciones calculadas:"
Print #1, ""
Print #1, "La población de proyecto para el año"; añof; "es de ";
Format(pp, "#,##0"); " habitantes"
Close #1
Open "C:\JLProject\temp\geometrico.txt" For Input As #1
While Not EOF(1)
Line Input #1, Linealeida
RichTextBox1.Text = RichTextBox1.Text & Linealeida & Chr(13) &
Chr(10)
Wend
Close #1
Exit Sub
End Sub

´Código para el botón calcular.
Private Sub cmdcalcula_Click()
añof = Val(InputBox("¿Cual es el año del que se desea conocer la
población?", "Año futuro"))
lnpf = Log(p2) + k * (añof - a2)
pp = Exp(lnpf)
Open "C:\JLProject\temp\geometrico.txt" For Append As #1
Print #1, ""
Print #1, Format(Time, "HH:MM:SS AMPM")
```

```
Print #1, "La población de proyecto para el año"; añof; "es de ";
Format(pp, "#,##0"); " habitantes."
Close #1
RichTextBox1.Text = ""
Open "C:\JLProject\temp\geometrico.txt" For Input As #1
While Not EOF(1)
Line Input #1, Linealeida
RichTextBox1.Text = RichTextBox1.Text & Linealeida & Chr(13) &
Chr(10)
Wend
Close #1
End Sub
```

El funcionamiento es igual al método aritmético, nos pedirá dos datos estadísticos (los dos años de registro y sus respectivas poblaciones) mediante cuadros de dialogo al igual que en todos los métodos y al finalizar pedirá el año del que se desea conocer la población de proyecto, el cual deberá ser mayor que el ultimo año de registro. Se pueden realizar mas cálculos para lo cual hay que dar click en el botón calcular y se podrán realizar tantos cálculos como se desee.

Guardar, borrar, imprimir y solicitar ayuda es igual que en el método aritmético y en todos los demás.

4.5. Método geométrico decreciente.

Cuando la población tiende a un valor máximo denominado “de saturación”, es conveniente estimar la población futura con los parámetros de la ley de crecimiento que puede considerarse geométrica decreciente.

La población puede llegar a este valor máximo de saturación, a causa de limitaciones de sus recursos económicos, naturales o del área urbanizable, entre otras. La velocidad de crecimiento seria directamente proporcional a la población faltante de saturación, es decir:

$$\frac{dP}{dt} = K_D(L - P) \quad \text{.....(4.11)}$$

Donde *L* es la población máxima o de saturación. La función de población se obtiene integrando la expresión 4.11:

$$\int_1^2 \frac{dP}{L - P} = K_D \int_1^2 dt$$

$$-Ln(L - P)\Big|_1^2 = K_D(t_2 - t_1)$$

$$-Ln\frac{(L-P_2)}{(L-P_1)} = K_D(t_2 - t_1)$$

De donde:

$$K_D = \frac{-Ln\frac{(L-P_2)}{(L-P_1)}}{t_2 - t_1}$$

…….(4.12)

Para una población *P* a un tiempo *T*, tomando como datos iniciales P_2 al tiempo t_2, se tiene:

$$-Ln\frac{L-P}{L-P_2} = K_D(T - t_2)$$

$$Ln\frac{L-P}{L-P_2} = -K_D(T - t_2)$$

$$\frac{L-P}{L-P_2} = e^{-K_D(T-t_2)}$$

$$L - P = (L-P_2)e^{-K_D(T-t_2)}$$

$$-P = -L + (L-P_2)e^{-K_D(T-t_2)}$$

$$P = L - (L-P_2)e^{-K_D(T-t_2)}$$

Restando P_2 a ambos lados de la igualdad:

$$P - P_2 = (L-P_2) - (L-P_2)e^{-K_D(T-t_2)}$$

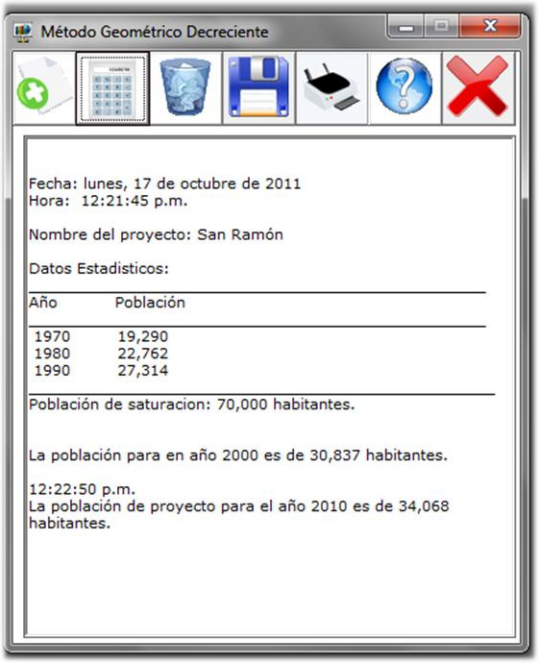
Asociando términos:

$$P - P_2 = (L-P_2)(1 - e^{-K_D(T-t_2)})$$

$$P = -P_2 + (L-P_2)(1 - e^{-K_D(T-t_2)})$$

…….(4.13)

La ventana de dicho método es la siguiente:



Su código y funcionamiento es el siguiente:

```
Public Sub cmdcargar_Click()
On Error Resume Next
Kill "C:\JLProject\temp\geometrico decreciente.txt"
nombre = InputBox("¿Cual es el nombre del Registro? ",
"Nombre")
L = Val(InputBox("¿Cual es la población de saturación?",
"Población de saturación"))
n = Val(InputBox("¿Cual es el número de datos equidistantes
con que se cuenta?", "Número de datos"))
eq = Val(InputBox("¿Cual es la equidistancia entre los años de
registro?", "Equidistancia"))
a1 = Val(InputBox("¿Cual es el primer año de registro?",
"Primer año"))
sk = 0
km = 0
pp = 0
Open "C:\JLProject\temp\geometrico decreciente.txt" For Append
As #1
Print #1, ""
Print #1, ""
Print #1, "Fecha: " & Format(Date, "Long Date")
Print #1, "Hora:  " & Format(Time, "HH:MM:SS AMPM")
Print #1, ""
Print #1, "Nombre del proyecto: " & nombre
Print #1, ""
```

```
Print #1, "Datos Estadisticos:"
Print #1, "_____________________________________________"
Print #1, "Año", "Población"
Print #1, "_____________________________________________"
'
P = Val(InputBox("¿Cual es la población para el año " & a1,
"Población"))
Print #1, a1, Format(P, "#,##0")
'
For x = 1 To n - 1 Step 1
pn = Val(InputBox("¿Cual es la población para el año " & (a1 +
eq * x), "Población"))
k = (-Log((L - pn) / (L - P))) / eq
sk = sk + k
Print #1, a1 + eq * x, Format(pn, "#,##0")
P = pn
Next x
'
km = sk / (n - 1)
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
a = (L - pn)
b = 1 - Exp(-km * (añof - (a1 + eq * (n - 1))))
pp = pn + a * b
Print #1, "_____________________________________________"
Print #1, "Población de saturacion: " & Format(L, "#,##0"); "
habitantes."
Print #1, ""
Print "1, "; Poblaciones; calculadas; ""
Print #1, ""
Print #1, "La población para en año " & añof; " es de " &
Format(pp, "#,##0"); " habitantes."
Print #1, ""
Close #1
Open "C:\JLProject\temp\geometrico decreciente.txt" For Input
As #1
While Not EOF(1)
Line Input #1, Linealeida
RichTextBox1.Text = RichTextBox1.Text & Linealeida & Chr(13) &
Chr(10)
Wend
Close #1
Exit Sub
End Sub

Private Sub cmdcalcula_Click()
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
```

```
b = 1 - Exp(-km * (añof - (a1 + eq * (n - 1))))
pp = pn + a * b
Open "C:\JLProject\temp\geometrico decreciente.txt" For Append
As #1
Print #1, Format(Time, "HH:MM:SS AMPM")
Print #1, "La población de proyecto para el año " & añof; " es
de " & Format(pp, "#,##0"); " habitantes."
Print #1, ""
Close #1
RichTextBox1.Text = ""
Open "C:\JLProject\temp\geometrico decreciente.txt" For Input
As #1
While Not EOF(1)
Line Input #1, Linealeida
RichTextBox1.Text = RichTextBox1.Text & Linealeida & Chr(13) &
Chr(10)
Wend
Close #1
Exit Sub
End Sub
```

En este método a inicio pide el nombre del proyecto, posteriormente nos pide el valor de la población de saturación de esa comunidad, después el primer año de registro, por ejemplo si se tiene los datos de población de una comunidad de los años 1970, 1980 y 1990 el primer año será 1970 la equidistancia entre los años de registro es 10 y el numero de datos es 3, posterior a esto el programa nos preguntara la población para cada uno de estos años y al final el año del que queremos conocer la población de proyecto.

La forma de cargar los datos en los demás métodos es la misma, que básicamente es saber con cuantos datos estadísticos contamos, que sean consecutivos y equidistantes y conocer los años de los cuales queremos conocer la población de proyecto.

4.6. Método del interés compuesto.

Cuando se supone un crecimiento en progresión geométrica, los valores que se obtienen para la población futura son mayores que los que se obtendrían suponiendo un crecimiento en progresión aritmética.

Este método es el normalmente usado en licenciatura para los cálculos de la población de proyecto, ya que toma directamente los datos de los tabuladores del INEGI y se toma como base solo la población actual.

La ecuación 4.8 puede escribirse como sigue:

$$LnP = LnP_0 + K_G t \text{.....(4.8)}$$

Donde P_0 es la población cuando t=0, tomando antilogaritmos:

$$P = P_0 e^{K_G t} \text{.....(4.9)}$$

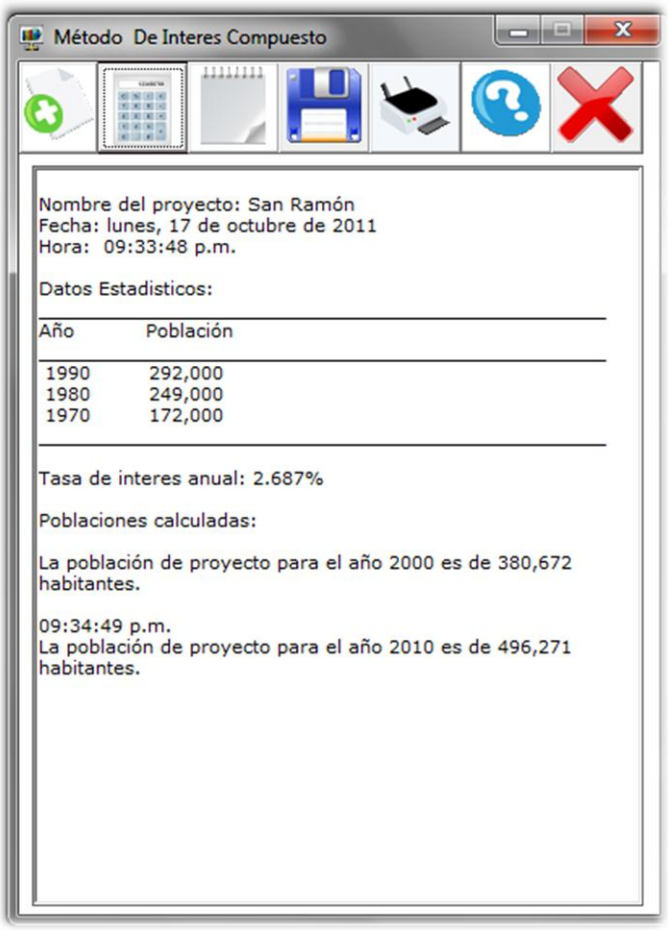
Esta ecuación es conocida como de capitalización con interés compuesto, es decir, el interés periódico se capitaliza aumentando el capital anterior. Usualmente e^{K_G} se representa como (1+i), donde i es la tasa de interés, quedando entonces:

$$P = P_0(1+i)^t \text{.....(4.10)}$$

Despejando i tenemos:

$$i = \sqrt[t]{\frac{P}{P_0}} - 1$$

La ventana correspondiente a este método es la siguiente:



El código correspondiente de este método es el siguiente:

```
Public Sub cmdcargar_Click()
On Error Resume Next
Kill "C:\JLProject\temp\interescompuesto.txt"
'lectura de datos
nombre = InputBox("¿Cual es el nombre del Registro? ",
"Nombre")
n = Val(InputBox("¿Cual es el número de datos equidistantes
con que se cuenta?", "Número de datos"))
```

```
While n < 2
MsgBox ("Debe de contar con almenos 2 datos.")
n = Val(InputBox("¿Cual es el número de datos equidistantes
con que se cuenta?", "Número de datos"))
Wend

eq = Val(InputBox("¿Cual es la equidistancia entre los años
de registro?", "Equidistancia"))
While eq < 1
MsgBox ("Debe de introducir un valor valido.")
eq = Val(InputBox("¿Cual es la equidistancia entre los años
de registro?", "Equidistancia"))
Wend

a1 = Val(InputBox("¿Cual es el primer año de registro?",
"Primer año"))
While a1 < 1
MsgBox ("Debe de proporcionar un valor valido para el primer
año.")
a1 = Val(InputBox("¿Cual es el primer año de registro?",
"Primer año"))
Wend

an = a1 + eq * (n - 1)
si = 0

Open "C:\JLProject\temp\interescompuesto.txt" For Append As
#1
Print #1, ""
Print #1, "Nombre del proyecto: " & nombre
Print #1, "Fecha: " & Format(Date, "Long Date")
Print #1, "Hora:  " & Format(Time, "HH:MM:SS AMPM")
Print #1, ""
Print #1, "Datos Estadisticos:"
Print #1, "_____________________________________________"
Print #1, "Año", "Población"
Print #1, "_____________________________________________"

MsgBox ("Los datos de población se deberan agragar a partir
del ultimo año de registro hacia atras tal como lo indicará
en cada petición.")

P = Val(InputBox("¿Cual es la población para el año " & (a1 +
eq * (n - 1)), "Población"))
While P < 1
MsgBox ("Debe de proporcionar un valor valido para la
población.")
```

```
P = Val(InputBox("¿Cual es la población para el año " & (a1 +
eq * (n - 1)), "Población"))
Wend

Print #1, (a1 + eq * (n - 1)), Format(P, "#,##0")

pn = Val(InputBox("¿Cual es la población para el año " & (a1
+ eq * (n - 2)), "Población"))
While pn < 1
MsgBox ("Debe de proporcionar un valor valido para la
población.")
pn = Val(InputBox("¿Cual es la población para el año " & (a1
+ eq * (n - 2)), "Población"))
Wend

Print #1, (a1 + eq * (n - 2)), Format(pn, "#,##0")
i1 = ((P / pn) ^ (1 / eq)) - 1

For x = n - 3 To 0 Step -1
pi = Val(InputBox("¿Cual es la población para el año " & (a1
+ eq * x), "Población"))
While pi < 1
MsgBox ("Debe de proporcionar un valor valido para la
población.")
pi = Val(InputBox("¿Cual es la población para el año " & (a1
+ eq * x), "Población"))
Wend

i = ((pn / pi) ^ (1 / eq)) - 1
si = si + i
pn = pi
Print #1, (a1 + eq * x), Format(pi, "#,##0")
Next x

añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
While añof < (a1 + eq * (n - 1))
MsgBox ("El año a calcular debe ser mayor que el ultimo año
de registro.")
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
Wend

im = (si + i1) / (n - 1)
pp = P * (1 + im) ^ (añof - (a1 + eq * (n - 1)))

Print #1, "_____"
Print #1, ""
```

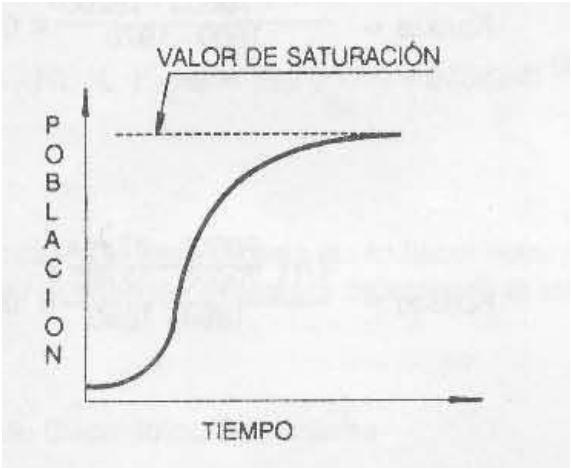
```
Print #1, "Tasa de interes anual: " & Format(im, "#.##0%")
Print #1, ""
Print #1, "Poblaciones calculadas:"
Print #1, ""
Print #1, "La población de proyecto para el año " & añof; "
es de " & Format(pp, "#,##0"); " habitantes."
Print #1, ""
Close #1

Open "C:\JLProject\temp\interescompuesto.txt" For Input As #1
While Not EOF(1)
Line Input #1, Linealeida
RichTextBox1.Text = RichTextBox1.Text & Linealeida & Chr(13)
& Chr(10)
Wend
Close #1
Exit Sub
End Sub
Private Sub cmdcalcula_Click()
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
While añof < (a1 + eq * (n - 1))
MsgBox ("El año a calcular debe ser mayor que el ultimo año
de registro.")
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
Wend

pp = P * (1 + im) ^ (añof - (a1 + eq * (n - 1)))
RichTextBox1.Text = ""
Open "C:\JLProject\temp\interescompuesto.txt" For Append As
#1
Print #1, Format(Time, "HH:MM:SS AMPM")
Print #1, "La población de proyecto para el año " & añof; "
es de " & Format(pp, "#,##0"); " habitantes."
Print #1, ""
Close #1
Open "C:\JLProject\temp\interescompuesto.txt" For Input As #1
While Not EOF(1)
Line Input #1, Linealeida
RichTextBox1.Text = RichTextBox1.Text & Linealeida & Chr(13)
& Chr(10)
Wend
Close #1
End Sub
```

Al igual que en los otros métodos, en este al iniciar el programa nos pedirá el nombre del proyecto, el numero de datos equidistantes, el primer año de registro y cuando

comience a pedir la población de cada uno de los años este comenzara a pedirlos desde el ultimo año de registro.

4.7. Método logístico-biológico.

Este modelo se usa para planeaciones a largo plazo con recursos fijos en vías de desarrollo, en consecuencia tiende a una población máxima limitada, generalmente para grandes ciudades o países. La concepción del modelo corresponde al crecimiento que tienen las moscas o cualquier otro insecto en un espacio fijo y con alimentación limitada, en donde al inicio la velocidad aumenta hasta un cierto valor a partir del cual decrece tendiendo al valor nulo por disminución de alimento y contaminación del medio. La teoría para la población la formulo P.F. Verhulst en 1844 y la aplico R. Pearl en 1924 a los estudios demográficos. La curva de crecimiento de población tiene forma de S como se muestra en la siguiente figura. Se le denomina comúnmente como el “Método de la S logística”.



A lo largo del tiempo las condiciones de desarrollo de una ciudad cambian y cualquier punto de la curva puede ser el arranque de otra nueva para otros factores de crecimiento, tales como desarrollos turísticos, recursos naturales por explotar; afectaciones que sufren por desarrollos cercanos o regionales, políticas demográficas o ejecución de obras de infraestructura.

El modelo matemático se plantea con la ecuación diferencial:

$$\frac{dP}{dt} = K_B(L - P) \quad \text{.....(4.14)}$$

Donde *L* es la población límite. La ecuación anterior expresa que la velocidad de crecimiento es proporcional a la población y al faltante de población para llegar al límite. Separando variables para integrar se tiene:

$$\frac{dP}{P(L-P)} = K_B dt$$

…….. (4.15)

Haciendo cabio de variable $P = \frac{1}{z}$ y entonces:

$$dP = -\frac{dz}{z^2}$$

Sustituyendo:

$$\frac{-\frac{dz}{z^2}}{\frac{1}{z}(L-\frac{1}{z})} = K_B dt$$

$$-\frac{dz}{z^2\frac{1}{2}(L-\frac{1}{z})} = K_B dt$$

$$-\frac{dz}{(Lz-1)} = K_B dt$$

……. (4.16)

Integrando:

$$-\frac{1}{L}\ln(Lz-1) = K_B t + A$$

Donde *A* es una constante de integración. Volviendo a sustituir $P = \frac{1}{z}$

$$-\frac{1}{L}\ln(\frac{L}{P}-1) = K_B t + A$$

La constante de integración *A* se determina para las siguientes condiciones iniciales: en *t=0, P=P₀*, así

$$A = -\frac{1}{L}\ln(\frac{L}{Po}-1)$$

Sustituyendo este valor de *A* queda:

$$\frac{1}{L}\ln\frac{\frac{L}{Po}-1}{\frac{L}{P}-1} = K_B t$$

$$\frac{(L-Po)}{L-P} = e^{K_B L}t$$

…….. (4.17)

Despejando *P* de la ecuación anterior tenemos:

$$P = \frac{L}{1 + \frac{L}{Po - 1} e^{-K_B L t}}$$

…… (4.18)

Si se hace $m = \frac{L}{Po-1}$ y $a = -K_B L$, la expresión 4.18 queda como sigue:

$$P = \frac{L}{(1 + me^{at})}$$

……(4.19)

La ecuación anterior se denomina “ecuación logística de Verhulst-Pearl”.

La determinación de los parámetros *L, a* y *m* es fácil sise conocen tres puntos de ordenadas equidistantes, por ejemplo (P_0, 0); (P_1, Δt) y (P_2, $2\Delta t$), o sea que la distancia es “Δt”. sustituyendo estas coordenadas en la expresión logística se obtienen tres ecuaciones para la determinación de los tres parámetros:

$$L = \frac{2P_0P_1P_2 - P_1^2(P_0 + P_2)}{P_0P_2 - P_1^2}$$

…..... (4.20)

$$m = \frac{L - P_0}{P_0}$$

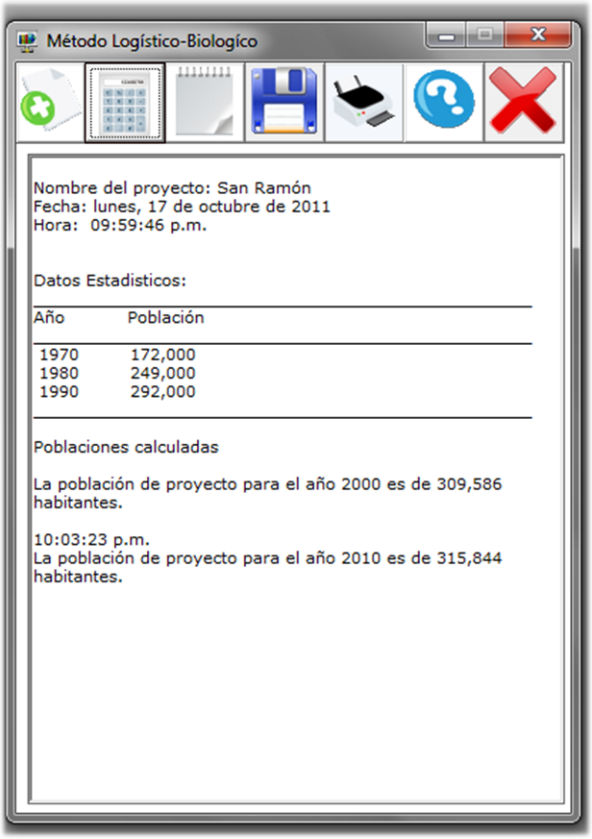
……. (4.21)

$$a = \frac{1}{\Delta t} \ln \left[\frac{P_0(L - P_1)}{P_1(L - P_0)}\right]$$

…….. (4.22)

Al aplicar estas ecuaciones a una serie de datos, se requiere seleccionar o deducir tres puntos que sean de la curva logística lo cual no siempre se logra y aún se pierde tiempo en la determinación cuando la serie no se ajusta al modelo.

La ventana correspondiente a este método con un ejemplo resuelto se muestra a continuación:



Este método requiere única y exclusivamente de 3 datos estadísticos equidistantes y su funcionamiento es el mismo que en los métodos anteriores.

El código correspondiente es el siguiente:

```
Public Sub cmdcargar_Click()
On Error Resume Next
Kill "C:\JLProject\temp\interescompuesto.txt"
'lectura de datos
nombre = InputBox("¿Cual es el nombre del Registro? ",
"Nombre")

n = Val(InputBox("¿Cual es el número de datos equidistantes
con que se cuenta?", "Número de datos"))
While n < 2
MsgBox ("Debe de contar con almenos 2 datos.")
n = Val(InputBox("¿Cual es el número de datos equidistantes
con que se cuenta?", "Número de datos"))
Wend

eq = Val(InputBox("¿Cual es la equidistancia entre los años de
registro?", "Equidistancia"))
While eq < 1
MsgBox ("Debe de introducir un valor valido.")
```

```
eq = Val(InputBox("¿Cual es la equidistancia entre los años de
registro?", "Equidistancia"))
Wend


a1 = Val(InputBox("¿Cual es el primer año de registro?",
"Primer año"))
While a1 < 1
MsgBox ("Debe de proporcionar un valor valido para el primer
año.")
a1 = Val(InputBox("¿Cual es el primer año de registro?",
"Primer año"))
Wend

an = a1 + eq * (n - 1)
si = 0

Open "C:\JLProject\temp\interescompuesto.txt" For Append As #1
Print #1, ""
Print #1, "Nombre del proyecto: " & nombre
Print #1, "Fecha: " & Format(Date, "Long Date")
Print #1, "Hora:  " & Format(Time, "HH:MM:SS AMPM")
Print #1, ""
Print #1, "Datos Estadisticos:"
Print #1, "_____________________________________________"
Print #1, "Año", "Población"
Print #1, "_____________________________________________"
MsgBox ("Los datos de población se deberan agragar a partir
del ultimo año de registro hacia atras tal como lo indicará en
cada petición.")
P = Val(InputBox("¿Cual es la población para el año " & (a1 +
eq * (n - 1)), "Población"))
While P < 1
MsgBox ("Debe de proporcionar un valor valido para la
población.")
P = Val(InputBox("¿Cual es la población para el año " & (a1 +
eq * (n - 1)), "Población"))
Wend
Print #1, (a1 + eq * (n - 1)), Format(P, "#,##0")
pn = Val(InputBox("¿Cual es la población para el año " & (a1 +
eq * (n - 2)), "Población"))
While pn < 1
MsgBox ("Debe de proporcionar un valor valido para la
población.")
pn = Val(InputBox("¿Cual es la población para el año " & (a1 +
eq * (n - 2)), "Población"))
Wend
```

```
Print #1, (a1 + eq * (n - 2)), Format(pn, "#,##0")
i1 = ((P / pn) ^ (1 / eq)) - 1

For x = n - 3 To 0 Step -1
pi = Val(InputBox("¿Cual es la población para el año " & (a1 +
eq * x), "Población"))
While pi < 1
MsgBox ("Debe de proporcionar un valor valido para la
población.")
pi = Val(InputBox("¿Cual es la población para el año " & (a1 +
eq * x), "Población"))
Wend

i = ((pn / pi) ^ (1 / eq)) - 1
si = si + i
pn = pi
Print #1, (a1 + eq * x), Format(pi, "#,##0")
Next x

añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
While añof < (a1 + eq * (n - 1))
MsgBox ("El año a calcular debe ser mayor que el ultimo año de
registro.")
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
Wend
im = (si + i1) / (n - 1)
pp = P * (1 + im) ^ (añof - (a1 + eq * (n - 1)))
Print #1, "_____________________________________________"
Print #1, ""
Print #1, "Tasa de interes anual: " & Format(im, "#.##0%")
Print #1, ""
Print #1, "Poblaciones calculadas:"
Print #1, ""
Print #1, "La población de proyecto para el año " & añof; " es
de " & Format(pp, "#,##0"); " habitantes."
Print #1, ""
Close #1
Open "C:\JLProject\temp\interescompuesto.txt" For Input As #1
While Not EOF(1)
Line Input #1, Linealeida
RichTextBox1.Text = RichTextBox1.Text & Linealeida & Chr(13) &
Chr(10)
Wend
Close #1
Exit Sub
End Sub
```

```
Private Sub cmdcalcula_Click()
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
While añof < (a1 + eq * (n - 1))
MsgBox ("El año a calcular debe ser mayor que el ultimo año de
registro.")
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
Wend

pp = P * (1 + im) ^ (añof - (a1 + eq * (n - 1)))
RichTextBox1.Text = ""
Open "C:\JLProject\temp\interescompuesto.txt" For Append As #1
Print #1, Format(Time, "HH:MM:SS AMPM")
Print #1, "La población de proyecto para el año " & añof; " es
de " & Format(pp, "#,##0"); " habitantes."
Print #1, ""
Close #1
Open "C:\JLProject\temp\interescompuesto.txt" For Input As #1
While Not EOF(1)
Line Input #1, Linealeida
RichTextBox1.Text = RichTextBox1.Text & Linealeida & Chr(13) &
Chr(10)
Wend
Close #1
End Sub
Private Sub cmdborra_Click()
On Error Resume Next
Kill "C:\JLProject\temp\interescompuesto.txt"
RichTextBox1.Text = ""
End Sub

Private Sub cmdguardar_Click()
Open "C:\JLProject\Proyectos guardados\Metodo del Interés
Compuesto.txt" For Append As #1
Print #1, "-----------------------------------------"
Print #1, ""
Close #1
Open "C:\JLProject\temp\interescompuesto.txt" For Input As #1
While Not EOF(1)
Line Input #1, Linealeida
Open "C:\JLProject\Proyectos guardados\Metodo del Interés
Compuesto.txt" For Append As #2
Print #2, Linealeida
Close #2
Wend
Close #1
```

```
MsgBox ("El proyecto se ha guardado con exito en
C:\JLProject\Proyectos guardados\Metodo del Interés
Compuesto.txt")
End Sub
```

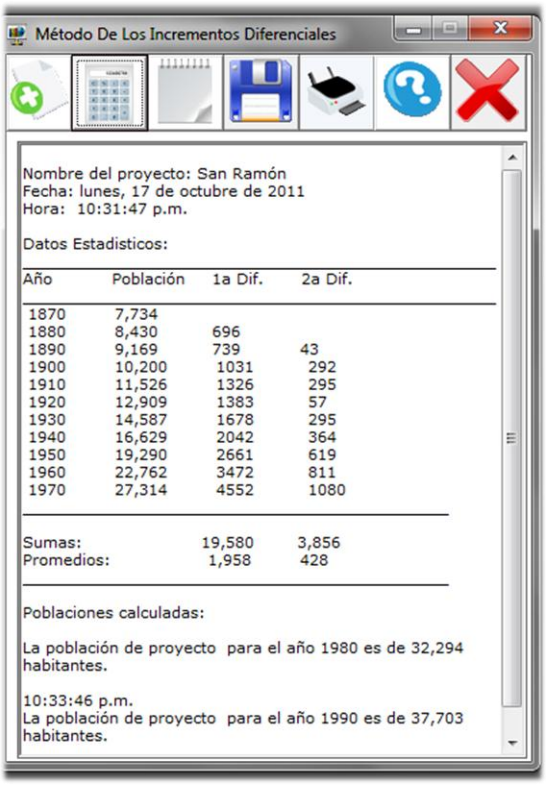
Los demás botones de comando funcionas de la misma manera que en los demás métodos.

4.8. Método de los incrementos diferenciales.

Este método consiste en considerar que la segunda diferencia entre los datos de población es constante lo cual equivale a ajustar los datos a los de una parábola de segundo grado. De igual manera que el método anterior, se requiere que los datos sean equidistantes para su aplicación.

Para la aplicación de este método se debe de contar con al menos 3 datos equidistantes y no hay límite de datos a utilizar.

La ventana correspondiente es la siguiente:



Y el código de los botones e comando cargar y calcular es el siguiente:

```
Public Sub cmdcargar_Click()
On Error Resume Next
Kill "C:\JLProject\temp\logistico.txt"
```

```
nombre = InputBox("¿Cual es el nombre del Registro? ",
"Nombre")
MsgBox ("Este metodo requiere unicamente de 3 datos
estadisticos.")

a1 = Val(InputBox("¿Cual es el primer año de registro?",
"Primer año"))
While a1 < 1
MsgBox ("Debe de proporcionar un valor valido para el primer
año.")
a1 = Val(InputBox("¿Cual es el primer año de registro?",
"Primer año"))
Wend

eq = Val(InputBox("¿Cual es la equidistancia entre los años de
registro?", "Equidistancia"))
While eq < 1
MsgBox ("Debe de introducir un valor valido.")
eq = Val(InputBox("¿Cual es la equidistancia entre los años de
registro?", "Equidistancia"))
Wend

Open "C:\JLProject\temp\logistico.txt" For Append As #1
Print #1, ""
Print #1, "Nombre del proyecto: " & nombre
Print #1, "Fecha: " & Format(Date, "Long Date")
Print #1, "Hora:  " & Format(Time, "HH:MM:SS AMPM")
Print #1, ""
Print #1, ""
Print #1, "Datos Estadisticos:"
Print #1, "_____________________________________________"
Print #1, "Año", "Población"
Print #1, "_____________________________________________"

p1 = Val(InputBox("¿Cual es la población para el año " & a1,
"Población"))
While p1 < 1
MsgBox ("Debe de proporcionar un valor valido para la
población.")
p1 = Val(InputBox("¿Cual es la población para el año " & a1,
"Población"))
Wend
Print #1, a1, Format(p1, "#,##0")

p2 = Val(InputBox("¿Cual es la población para el año " & a1 +
eq, "Población"))
While p2 < 1
```

```
MsgBox ("Debe de proporcionar un valor valido para la
población.")
p2 = Val(InputBox("¿Cual es la población para el año " & a1 +
eq, "Población"))
Wend
Print #1, a1 + eq, Format(p2, "#,##0")

p3 = Val(InputBox("¿Cual es la población para el año " & a1 +
2 * eq, "Población"))
While p3 < 1
MsgBox ("Debe de proporcionar un valor valido para la
población.")
p3 = Val(InputBox("¿Cual es la población para el año " & a1 +
2 * eq, "Población"))
Wend
Print #1, a1 + 2 * eq, Format(p3, "#,##0")

Print #1, "_____________________________________________________"
Print #1, ""

añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
While añof < (a1 + 2 * eq)
MsgBox ("El año a calcular debe ser mayor que el ultimo año de
registro.")
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
Wend

t = añof - a1
L = ((2 * p1 * p2 * p3) - ((p2 ^ 2) * (p1 + p3))) / ((p1 * p3)
- p2 ^ 2)
m = (L - p1) / p1
a = (1 / eq) * (Log((p1 * (L - p2)) / (p2 * (L - p1))))
pp = L / (1 + (m * Exp(a * t)))
Print #1, "Poblaciones calculadas"
Print #1, ""
Print #1, "La población de proyecto para el año " & añof; " es
de " & Format(pp, "#,##0"); " habitantes."
Print #1, ""
Close #1
Open "C:\JLProject\temp\logistico.txt" For Input As #1
While Not EOF(1)
Line Input #1, Linealeida
RichTextBox1.Text = RichTextBox1.Text & Linealeida & Chr(13) &
Chr(10)
Wend
Close #1
```

```
Exit Sub
End Sub

Private Sub cmdcalcula_Click()
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
While añof < (a1 + 2 * eq)
MsgBox ("El año a calcular debe ser mayor que el ultimo año de
registro.")
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
Wend

t = añof - a1
pp = L / (1 + (m * Exp(a * t)))
RichTextBox1.Text = ""
Open "C:\JLProject\temp\logistico.txt" For Append As #1
Print #1, Format(Time, "HH:MM:SS AMPM")
Print #1, "La población de proyecto para el año " & añof; " es
de " & Format(pp, "#,##0"); " habitantes."
Print #1, ""
Close #1
Open "C:\JLProject\temp\logistico.txt" For Input As #1
While Not EOF(1)
Line Input #1, Linealeida
RichTextBox1.Text = RichTextBox1.Text & Linealeida & Chr(13) &
Chr(10)
Wend
Close #1
End Sub
```

Para este método en particular no es posible darle al programa un año en especial para que realice el cálculo, si no que ira calculando en intervalos iguales a la equidistancia entre los años de registro, para lo cual será necesario dar click al botón calcular hasta llegar al año deseado.

4.9. Método de la parábola cúbica.

Este método considera que la curva de crecimiento se aproxima a la de una parábola cúbica del tipo:

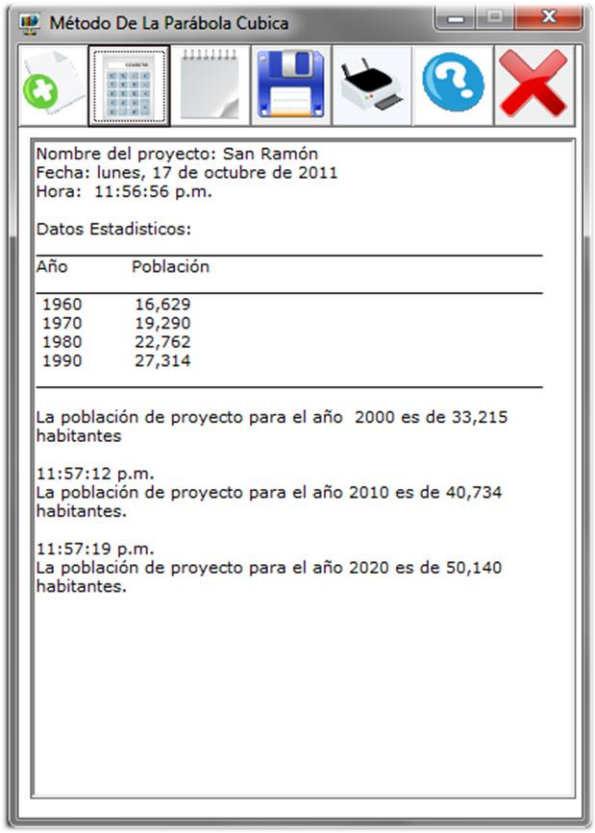
$$P = a + bx + cx^2 + dx^3$$

…… (4.23)

En donde *x*=año.

Para aplicar este método se requiere contar con al menos cuatro datos.

La ventana de dicho método con un ejemplo es la siguiente:



El código correspondiente es el siguiente:

```
Public Sub cmdcargar_Click()
On Error Resume Next
Kill "C:\JLProject\temp\parabola.txt"
nombre = InputBox("¿Cual es el nombre del Registro? ",
"Nombre")
MsgBox ("Este metodo requiere unicamente de 4 registros.")
'
eq = Val(InputBox("¿Cual es la equidistancia entre los años de
registro?", "Equidistancia entre años"))
While eq < 1
MsgBox ("Debe de introducir un valor valido.")
eq = Val(InputBox("¿Cual es la equidistancia entre los años de
registro?", "Equidistancia entre años"))
Wend
'
a1 = Val(InputBox("¿Cual es el primer año de registro?",
"Primer año"))
While a1 < 1500
MsgBox ("Debe de introducir un valor valido.")
```

```
a1 = Val(InputBox("¿Cual es el primer año de registro?",
"Primer año"))
Wend
'
p1 = Val(InputBox("¿Cual es la población para el año " & a1,
"Población"))
While p1 < 1
MsgBox ("Debe de introducir un valor valido.")
p1 = Val(InputBox("¿Cual es la población para el año " & a1,
"Población"))
Wend
'
a2 = a1 + eq
p2 = Val(InputBox("¿Cual es la población para el año " & a2,
"Población"))
While p2 < (p1 + 1)
MsgBox ("Debe de introducir un valor valido.")
p2 = Val(InputBox("¿Cual es la población para el año " & a2,
"Población"))
Wend
'
a3 = a2 + eq
p3 = Val(InputBox("¿Cual es la población para el año " & a3,
"Población"))
While p3 < (p2 + 1)
MsgBox ("Debe de introducir un valor valido.")
p3 = Val(InputBox("¿Cual es la población para el año " & a3,
"Población"))
Wend
'
a4 = a3 + eq
p4 = Val(InputBox("¿Cual es la población para el año " & a4,
"Población"))
While p4 < (p3 + 1)
MsgBox ("Debe de introducir un valor valido.")
p4 = Val(InputBox("¿Cual es la población para el año " & a4,
"Población"))
Wend

Open "C:\JLProject\temp\parabola.txt" For Append As #1
Print #1, "Nombre del proyecto: " & nombre
Print #1, "Fecha: " & Format(Date, "Long Date")
Print #1, "Hora:  " & Format(Time, "HH:MM:SS AMPM")
Print #1, ""
Print #1, "Datos Estadisticos:"
Print #1, "_____________________________________________"
Print #1, "Año", "Población"
Print #1, "_____________________________________________"
```

```
Print #1, a1, Format(p1, "#,##0")
Print #1, a2, Format(p2, "#,##0")
Print #1, a3, Format(p3, "#,##0")
Print #1, a4, Format(p4, "#,##0")
Print #1, "_____________________________________________________"
Print #1, ""
'
añof = Val(InputBox("Cual es el año del que desea conocer la
población?", "Año a Calcular"))
While añof < a4
MsgBox ("El año a calcular debe ser mayor que el ultimo año de
registro.")
añof = Val(InputBox("Cual es el año del que desea conocer la
población?", "Año a Calcular"))
Wend
'
a = p1
px1 = p2 - a
px2 = p3 - a
px3 = p4 - a
b = (3 * px1) - (1.5 * px2) + ((1 / 3) * px3)
C = (2 * px2) - (0.5 * px3) - (2.5 * px1)
d = (0.5 * px1) - (0.5 * px2) + ((1 / 6) * px3)
x = 3 + (añof - a4) / eq
P = a + (b * x) + (C * (x ^ 2)) + (d * (x ^ 3))
Print #1, "La población de proyecto para el año "; añof; "es
de "; Format(P, "#,##0"); " habitantes"
Print #1, ""
Close #1
Open "C:\JLProject\temp\parabola.txt" For Input As #1
While Not EOF(1)
Line Input #1, Linealeida
RichTextBox1.Text = RichTextBox1.Text & Linealeida & Chr(13) &
Chr(10)
Wend
Close #1
Exit Sub
End Sub
Private Sub cmdcalcula_Click()
añof = Val(InputBox("Cual es el año del que desea conocer la
población?", "Año a Calcular"))
While añof < a4
MsgBox ("Debe de introducir una fecha mayor que el ultimo año
de registro.")
añof = Val(InputBox("Cual es el año del que desea conocer la
población?", "Año a Calcular"))
Wend
'
```

```
x = 3 + (añof - a4) / eq
P = a + (b * x) + (C * (x ^ 2)) + (d * (x ^ 3))
'
Open "C:\JLProject\temp\parabola.txt" For Append As #1
Print #1, Format(Time, "HH:MM:SS AMPM")
Print #1, "La población de proyecto para el año"; añof; "es de
"; Format(P, "#,##0"); " habitantes."
Print #1, ""
Close #1
RichTextBox1.Text = ""
Open "C:\JLProject\temp\parabola.txt" For Input As #1
While Not EOF(1)
Line Input #1, Linealeida
RichTextBox1.Text = RichTextBox1.Text & Linealeida & Chr(13) &
Chr(10)
Wend
Close #1
End Sub
```

Para este caso en particular lo que se hizo para resolver e sistema de ecuaciones, fue encontrar una solución general a un sistema de 3 ecuaciones con tres incógnitas y es el que se muestra en el código y sustituir los valores en la ecuación particular de este método.

4.10. Método de los mínimos cuadrados.

Una relación lineal entre dos variables queda representada por una línea recta cuya ecuación general es $y = a + bx$. El método de los mínimos cuadrados es el procedimiento matemático utilizado para determinar los valores numéricos de las constantes "a" y "b" en la ecuación. El método utiliza el conjunto de observaciones que en este caso son años y número de habitantes.

Por ejemplo, sean "x" los años en una lista de datos y "y" el numero de habitantes de cada uno de esos años. Se calcula x^2, y^2 y el producto xy para cada serie de datos. Así mismo se realizan las sumatorias de todas las columnas, es decir, se calculan las sumatorias $\sum x, \sum y, \sum x^2, \sum y^2, \sum xy$.

Los valores de estas sumatorias se sustituyen en las ecuaciones normales:

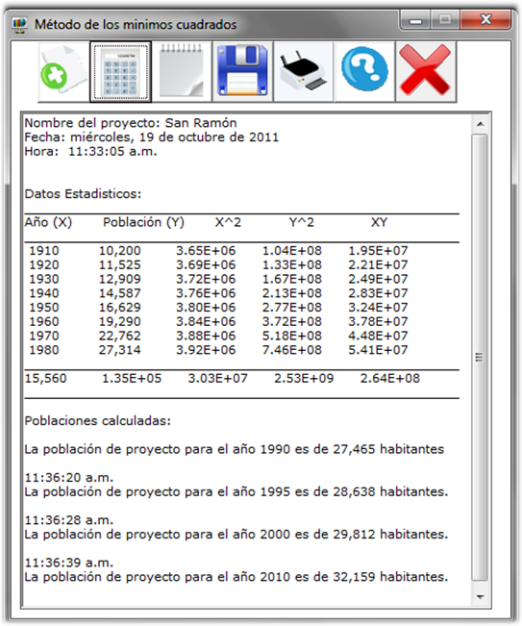
$$\sum y = na + b\sum x \quad \text{..... (4.24)}$$

$$\sum xy = a\sum x + b\sum x^2 \quad \text{..... (4.25)}$$

Donde *n* representa el número de pares de observaciones utilizadas en la regresión. Se forma de esta manera un sistema de dos ecuaciones con dos incógnitas; el

cual se resuelve para encontrar los valores de a y b y con ello se forma la ecuación de regresión $y = a + bx$ que permite calcular la población para cualquier año.

El diseño del formulario para este método con un ejemplo resuelto es el siguiente:



El código empleado para los botones de comando de este método es el siguiente:

```
Public Sub cmdcargar_Click()
On Error Resume Next
Kill "C:\JLProject\temp\minimos.txt"
nombre = InputBox("¿Cual es el nombre del Registro? ",
"Nombre")
n = Val(InputBox("¿Cual es el número de datos equidistantes
con que se cuenta?", "Número de datos"))
While n < 2
MsgBox ("Debe introducir como minimo 2 datos.")
n = Val(InputBox("¿Cual es el número de datos equidistantes
con que se cuenta?", "Número de datos"))
Wend
eq = Val(InputBox("¿Cual es la equidistancia entre los años de
registro?", "Equidistancia"))
While eq < 1
MsgBox ("Debe de introducir un valor valido.")
eq = Val(InputBox("¿Cual es la equidistancia entre los años de
registro?", "Equidistancia"))
Wend
a1 = Val(InputBox("¿Cual es el primer año de registro?",
"Primer año"))
While a1 < 1
MsgBox ("Debe de introducir un valor valido.")
```

```
a1 = Val(InputBox("¿Cual es el primer año de registro?",
"Primer año"))
Wend
Open "C:\JLProject\temp\minimos.txt" For Append As #1
Print #1, "Nombre del proyecto: " & nombre
Print #1, "Fecha: " & Format(Date, "Long Date")
Print #1, "Hora:  " & Format(Time, "HH:MM:SS AMPM")
Print #1, ""
Print #1, "Datos Estadisticos:"
Print #1, "_____________________________________________"
Print #1, "Año (X)", "Población (Y)"; "       X^2", "
Y^2", "        XY"
Print #1, "_____________________________________________"
sx = 0
sy = 0
sx2 = 0
sy2 = 0
sxy = 0
For x = 0 To n - 1 Step 1
a = a1 + eq * x
P = Val(InputBox("¿Cual es la población para el año " & a,
"Población"))
While P < 1
MsgBox ("Debe de introducir un valor valido.")
P = Val(InputBox("¿Cual es la población para el año " & a,
"Población"))
Wend
X2 = a ^ 2
Y2 = P ^ 2
xy = a * P
sx = sx + a
sy = sy + P
sx2 = sx2 + X2
sy2 = sy2 + Y2
sxy = sxy + xy
Print #1, a, Format(P, "#,##0"), Format(X2, "Scientific"),
Format(Y2, "Scientific"), Format(xy, "Scientific")
Next x
Print #1, "_____________________________________________"
Print #1, Format(sx, "#,##0"), Format(sy, "Scientific"),
Format(sx2, "Scientific"), Format(sy2, "Scientific"),
Format(sxy, "Scientific")
Print #1, "_____________________________________________"
w = (sy * sx2 - sx * sxy) / (n * sx2 - sx ^ 2)
z = (n * sxy - sy * sx) / (n * sx2 - sx ^ 2)
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
While añof < a
```

```
MsgBox ("El año a calcular debe ser mayor que el ultimo año de
registro.")
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
Wend
pp = w + z * añof
Print #1, ""
Print #1, "Poblaciones calculadas:"
Print #1, ""
Print #1, "La población de proyecto para el año"; añof; "es de
"; Format(pp, "#,##0"); " habitantes"
Print #1, ""
Close #1
Open "C:\JLProject\temp\minimos.txt" For Input As #1
While Not EOF(1)
Line Input #1, Linealeida
RichTextBox1.Text = RichTextBox1.Text & Linealeida & Chr(13) &
Chr(10)
Wend
Close #1
Exit Sub
End Sub
Private Sub cmdcalcula_Click()
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
While añof < a
MsgBox ("El año final debe ser mayor que el ultimo año de
registro.")
añof = Val(InputBox("¿Cual es el año del que se desea conocer
la población?", "Año futuro"))
Wend
pp = w + z * añof
RichTextBox1.Text = ""
Open "C:\JLProject\temp\minimos.txt" For Append As #1
Print #1, Format(Time, "HH:MM:SS AMPM")
Print #1, "La población de proyecto para el año " & añof; " es
de " & Format(pp, "#,##0"); " habitantes."
Print #1, ""
Close #1
Open "C:\JLProject\temp\minimos.txt" For Input As #1
While Not EOF(1)
Line Input #1, Linealeida
RichTextBox1.Text = RichTextBox1.Text & Linealeida & Chr(13) &
Chr(10)
Wend
Close #1
End Sub
```

5. Instalación y manejo del programa.

Para usar el programa, mismo que se puede proporcionar en un CD o en una memoria USB, es necesario copiar la carpeta denominada JLProject en el disco local C:, abrir dicha carpeta y crear un acceso directo en el escritorio del archivo JLProject.exe dando un click con el botón derecho del ratón y escoger la opción “Enviar al escritorio (crear acceso directo)” esto con la finalidad de no tener que estar abriendo la carpeta cada vez que se desee usar el programa.

Finalmente para hacer uso del programa bastara con dar doble click en el icono creado y automáticamente se abrirá el programa con la ventana principal para lo cual ya se ha descrito su funcionamiento y manejo para partir de ahí a una ventana especifica de alguno de los métodos de calculo.

Esta carpeta contiene los archivos necesarios para que el programa se pueda ejecutar sin ningún problema. Para evitar tener problemas con el funcionamiento del programa, no se deberán modificar ni mucho menos eliminarlos.

Si se desea recuperar algún archivo guardado, solo bastará con abrir la carpeta C:\JLProject y abrir el documento de texto correspondiente al método con el que se realizo el cálculo y buscarlo por su nombre o fecha de creación. Si la lista de proyectos ya es demasiado larga se pueden eliminar todos o algunos de los proyectos guardados, estos se van guardando en orden cronológico.

La información contenida se puede, si se desea, copiar para usarla para nuevos proyectos, presentar informes, modificarla, etc. pero se recomienda que no se borre hasta que se tenga la certeza de que ya no será útil en un futuro.

Conclusiones.

Se ha logrado el desarrollo de un programa que comprende la solución de los problemas de cálculo de la población de proyecto en los sistemas de abastecimiento de agua potable y demás obras civiles que así lo requieran. Con esta tesis se busca que en las obras de abastecimiento se realicen de manera adecuada los cálculos preliminares y con ellos determinar de manera correcta la dimensión de dichas obras, buscando con ello que estas resulten ser insuficientes antes de que terminen con su vida útil o en su defecto resulten sobradas y con ello un aumento en los costos de la obra.

En la actualidad solo se emplea la tasa de crecimiento y esta se multiplica por el periodo de servicio o vida útil de la obra, sin considerar otros factores importantes como puede ser la saturación del espacio en donde crece la población, la instalación de una zona industrial y con ello el crecimiento de la comunidad a sus alrededores, en fin, aspectos importantes que determinan el éxito o el fracaso de un determinado sistema de abastecimiento o cualquier otra obra.

Con este programa se puede determinar de manera rápida y confiable, según los métodos de cálculo, la población de proyecto; ya que con solo introducir algunos datos estadísticos y la selección del método correcto se obtienen las poblaciones para los periodos que el usuario desee dependiendo del tipo de obra.

El programa es de fácil manejo y permite guardar la información así como imprimirla para su posterior anexo a estudios preliminares de proyecto, contribuyendo así con la plataforma de programas para facilitar el trabajo del ingeniero civil.

Bibliografía.

1) Abastecimiento de Agua Potable, Enrique Cesar Valdez. UNAM.

2) Manual de Visual Studio 6.0. Lenguajes de Programación, Microsoft.

3) Visual Basic 6.0 Orientado a Bases de Datos, Carlos M. Rodríguez Bucarelly y etal.

4) Apuntes de la materia de programación.