PROGRAMACIÓN I

CARRERA: ANÁLISIS DE SISTEMAS
SEMESTRE: SEGUNDO

INTRODUCCIÓN

El presente libro es una guía para el estudio del lenguaje de programación Visual Basic Net que es un lenguaje orientado a objetos desarrollado por Microsoft.

"**Visual Basic:** Dirigido estudiantes del segundo semestre de Análisis de Sistemas del Instituto Universitario de Tecnología para la informática IUTEPI, una puerta de entrada a la programación de aplicaciones para Windows.

Por otra parte, en este manual, les brindaremos herramientas necesarias del lenguaje de programación mencionado, que fue desarrollado por Microsoft, este ha sido la elección preferida de millones de desarrolladores durante años, gracias a su sintaxis sencilla y su entorno de desarrollo visual intuitivo.

También, te guiaremos paso a paso en el aprendizaje de Visual Basic, conceptos básicos y técnicas de aplicación para descubrir cómo crear interfaces gráficas atractivas, interactuar con bases de datos, gestionar eventos y mucho más. Tanto si eres un principiante absoluto como si ya tienes experiencia en programación, encontrarás en estas páginas los conocimientos y recursos que necesitas para convertirte en un desarrollador de aplicaciones en lenguaje Visual Basic."

¿Qué es Visual Basic?

Es un lenguaje de programación que permite crear aplicaciones (programas) para Windows. Permite construir en forma fácil, programas con una interfaz gráfica que puede incorporar elementos como:

- ✓ Ventana
- ✓ cuadros de texto
- ✓ botones
- ✓ cuadros de diálogo
- ✓ botones de opción y de selección
  barras de desplazamiento
- ✓ menús…

Versiones de Visual Basic.

Primera versión Visual Basic 1.0 (1991)

Última versión Visual Basic 6.0 (1998 con soporte hasta 2008)

Visual Basic.NET fue el sucesor de Visual Basic 6.0.

Parecido a Visual Basic 6.0, pero tienen diferencias significativas que hacen que ambos no sean compatibles.

Para trabajar en Visual Basic.net se utiliza generalmente el entorno de desarrollo Visual Studio de Microsoft, el cual incluye varios lenguajes de programación como: tales como Visual C++, Visual C#, Visual F#, y Visual Basic .NET

Las siguientes versiones de Visual Basic (todas basadas en Visual Basic.NET) son:

Visual Basic 7.0, para Visual Studio .NET 2002 y 2003.

Visual Basic 8.0 (Visual Basic 2005), para Visual Studio 2005.

Visual Basic 9.0 (Visual Basic 2008), para Visual Studio 2008.

Visual Basic 10.0 (Visual Basic 2010) para Visual Studio 2010.

Visual Basic 11.0, para Visual Studio 2013, 2018, 2019

Visual Basic Code

Visual BASIC, significado de las siglas: Beginners' **A**ll-purpose **S**ymbolic **I**nstruction **C**ode (Código simbólico de instrucciones de propósito general para principiantes; y es una familia de lenguajes de programación de alto nivel. **Características:** Los compiladores de Visual Basic generan código que requiere una o más librerías de enlace dinámico para que funcione, conocidas comúnmente como **DLL** (sigla en inglés de *Dynamic-Link Library*); en algunos casos reside en el archivo llamado MSVBVMxy.DLL (siglas de "Microsoft Visual Basic Virtual Machine x.y", donde x.y es la versión) y en otros en VBRUNXXX.DLL ("Visual Basic Runtime X.XX").

Estas bibliotecas **DLL** proveen las funciones básicas implementadas en el lenguaje, conteniendo rutinas en código ejecutable que son cargadas *bajo demanda* en tiempo de ejecución. Además de las esenciales, existe un gran número de bibliotecas del tipo DLL con variedad de funciones, tales como las que facilitan el acceso a la mayoría de las funciones del sistema operativo o las que proveen medios para la integración con otras aplicaciones.

ÍNDICE

UNIDAD I

INTRODUCCIÓN A LA PROGRAMACIÓN CON VISUAL BASIC NET

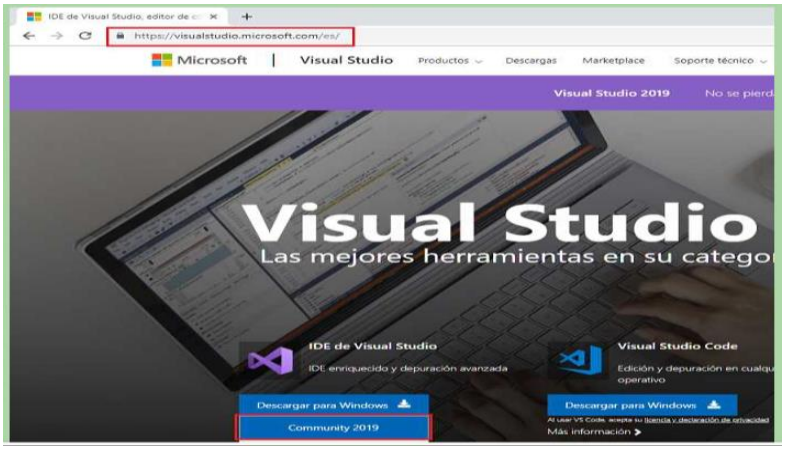
Instrucciones básicas del lenguaje de programación

Entorno de trabajo de Visual Basic

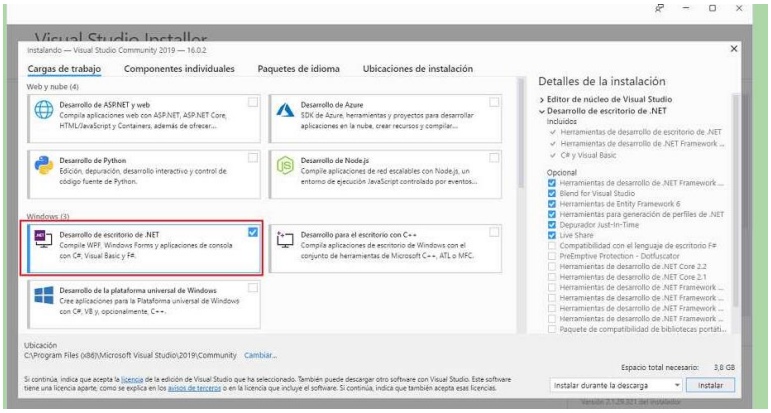
El código visual Basic**,** contiene un entorno de desarrollo integrado o IDE que compone un editor de textos para edición del código fuente, un depurador, un compilador (y enlazador) y un editor de interfaces gráficas o GUI llamado **Visual Studio Basic. Net**. El Entorno de Desarrollo Integrado (Integrated Development Environment) o IDE es el que proporciona todas las facilidades para programar en Visual Basic.

Descarga del entorno para programar con Visual Basic.Net

Podemos utilizar el Visual Studio Community 2019 que tiene entre otros lenguajes el Visual Basic .Net y lo podemos descargar desde aquí. También si ya tiene instalado el Visual Studio 2017 o 2015 puede hacer la mayor parte de este curso.



Cuando procedamos a instalar debemos seleccionar como mínimo "Desarrollo de escritorio de .Net":



En el caso de estudiar luego los cursos de aplicaciones para la web, aplicaciones universales para Windows etc. podemos instalarlas en otro momento.

Codificación del problema con el lenguaje Visual Basic .Net

No debemos perder de vista que el fin último es realizar un programa de computación que permita automatizar una actividad para que muchos procesos sean desarrollados por la computadora. El diagrama de flujo es un paso intermedio para poder ser interpretado por la computadora. El paso siguiente es la codificación del diagrama de flujo en un lenguaje de computación, en nuestro caso emplearemos el lenguaje Visual Basic.

Conceptos básicos para codificar un programa.

Lenguaje de computación: Conjunto de instrucciones que son interpretadas por una computadora para realizar operaciones, mostrar datos por pantalla, sacar listados por impresora, entrar datos por teclado, etc.

Variable: Es un depósito donde hay un valor. Consta de un nombre y pertenece a un tipo.

Para el ejemplo planteado la variable Horas Trabajadas almacena la cantidad de horas trabajadas por el operario. La variable Valor Hora almacena el precio de una hora de trabajo. La variable Sueldo almacena el sueldo a abonar al operario.

En el ejemplo tenemos tres variables.

Tipos de variable: Una variable puede almacenar:

- Valores Enteros (100, 260, etc.)
- Valores Reales (1.24, 2.90, 5.00, etc.)
- Cadenas de caracteres ("Juan", "Compras", "Listado", etc.)

Elección del nombre de una variable:

Debemos elegir nombres de variables representativas. En el ejemplo el nombre **Horas Trabajadas** es lo suficientemente claro para darnos una idea acabada sobre su contenido. Podemos darle otros buenos nombres. Otros no son tan representativos, por ejemplo **HTr.** Posiblemente cuando estemos resolviendo un problema dicho nombre nos recuerde que almacenamos las horas trabajadas por el operario pero cuando pase el tiempo y leamos el diagrama probablemente no recordemos ni entendamos qué significa **HTr.**

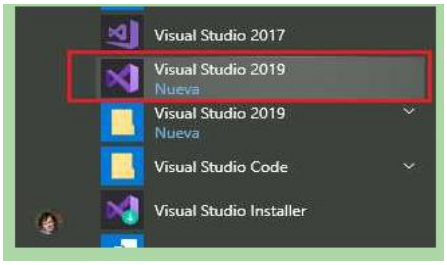
Consideraciones a tener en cuenta en cada proyecto.

Hay que tener en cuenta que el entorno de programación "Microsoft Visual Studio" no ha sido desarrollado pensando en un principiante de la programación y cuenta con un conjunto de herramientas que en un principio no utilizaremos. Cuenta con un gran conjunto de opciones que puede atemorizarnos, pero a medida que avancemos con el curso comenzaremos a entenderlas.

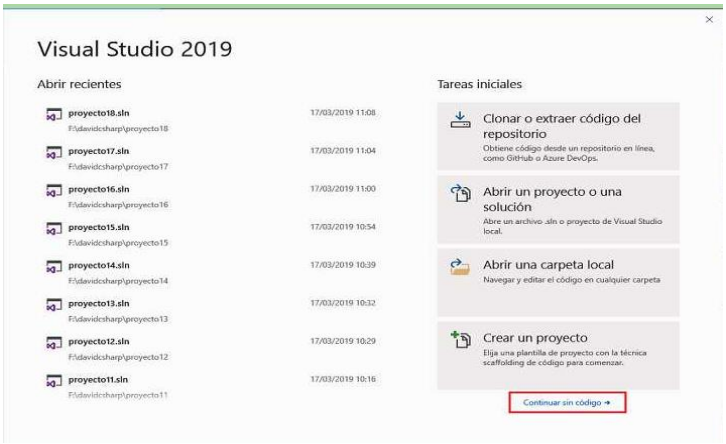
En ese sentido, veremos los pasos para la creación de un proyecto en **Visual Basic .Net**

Pasos.

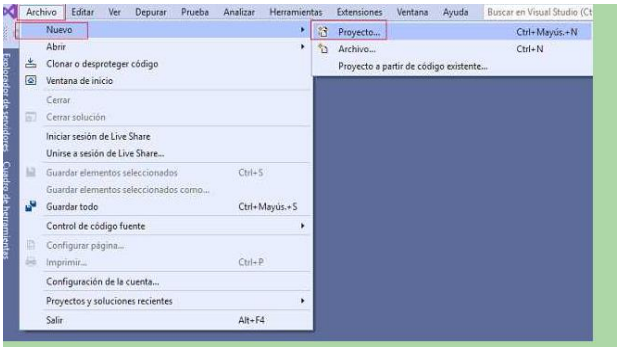
1 - Ingresemos al "Microsoft Visual Studio".



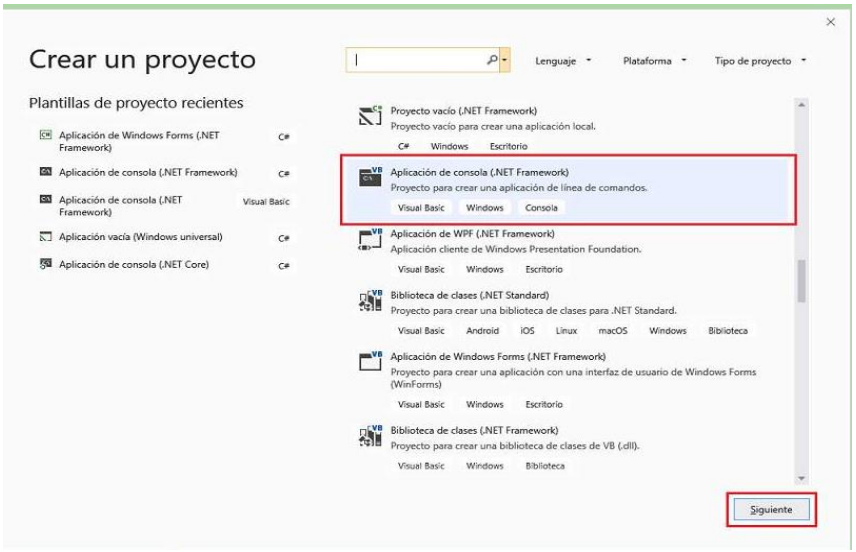
Aparece un diálogo donde tenemos distintas opciones para abrir proyectos anteriores, clonar proyectos almacenados en GitHub etc, nosotros elegiremos la opción de 'Continuar sin código' para conocer el entorno de Visual Studio 2019:



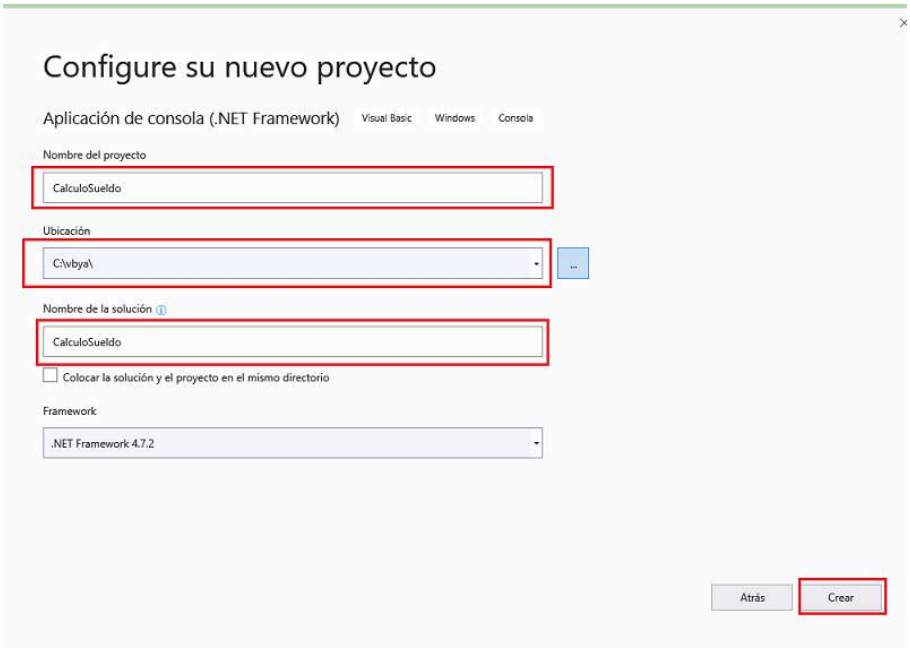
2 - Creación del proyecto. Para esto seleccionamos desde el menú la opción "Archivo" -> "Nuevo" -> "Proyecto..."



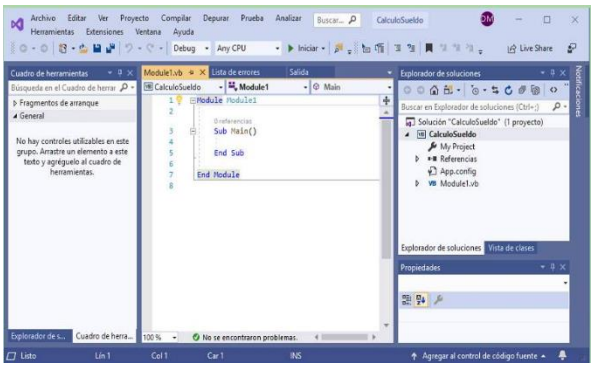
Aparece un diálogo donde debemos seleccionar "Aplicación de consola (.Net Framework)" para el lenguaje Visual Basic:



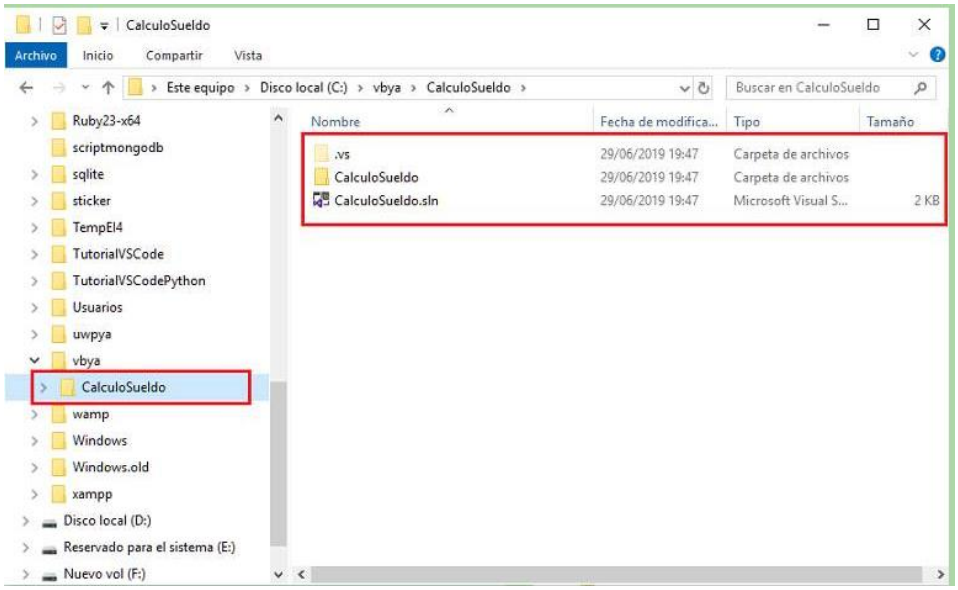
El diálogo siguiente nos solicita ingresar el "Nombre del proyecto", la "ubicación", y "nombre de la solución" (podemos usar el mismo texto para el "nombre de la solución" y "nombre del proyecto"):



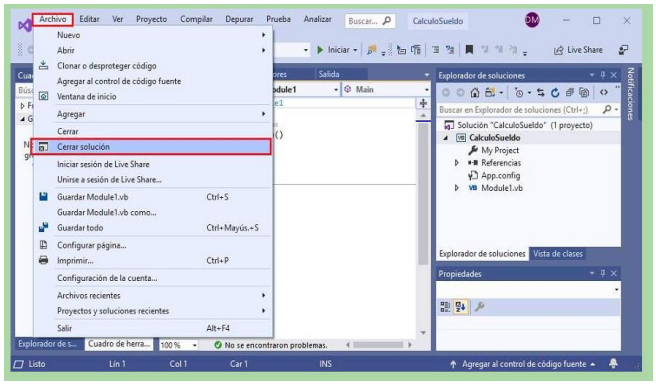
Podemos ver que el entorno nos generó automáticamente el esqueleto de nuestro programa:



3 - Si vamos al Explorador de archivos de Windows podemos ver que tenemos creada la carpeta con nuestro programa:

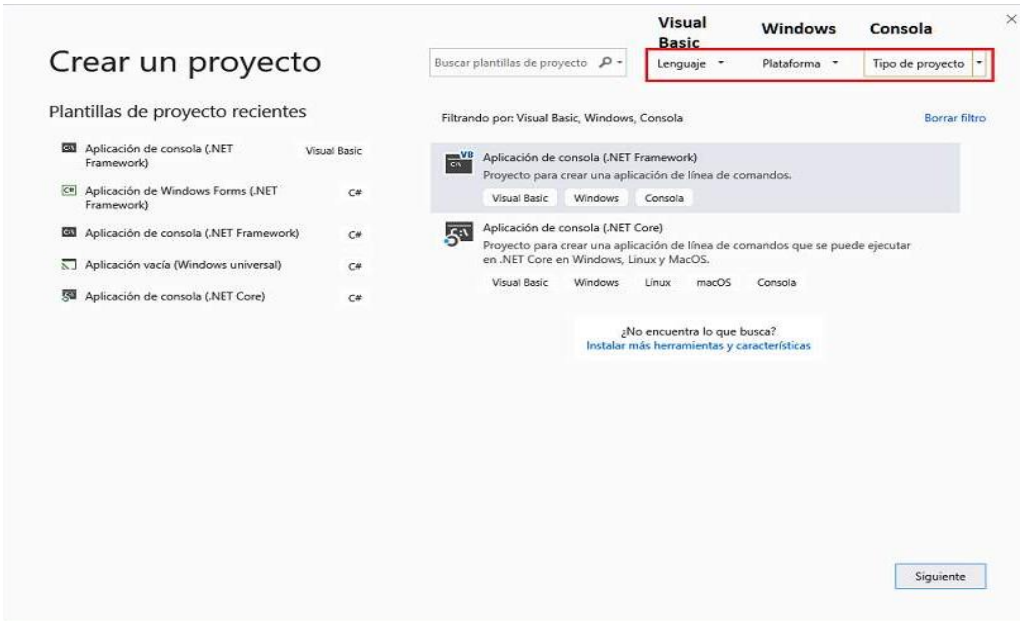


4 - Una vez que finalizamos de trabajar con el proyecto lo cerramos desde el menú de opciones Archivo -> Cerrar solución:



Acotaciones.

Cuando creamos un proyecto en Visual Studio .Net podemos filtrar el tipo de proyecto a implementar para que nos sea más fácil su selección:



Del lado izquierdo podemos ver las últimas plantillas de proyecto utilizadas, por lo que muy fácilmente podemos seleccionar "Aplicación de consola (.NET Framework) Visual Basic".

UNIDAD II

MANEJO DE VARIABLES, OPERADORES LÓGICOS Y OPERADORES MATEMÁTICOS

Variables

A menudo, hay que almacenar valores al realizar cálculos matemáticos con Visual Basic. Por ejemplo, tal vez le interese calcular varios valores, compararlos y realizar diversas operaciones con ellos, según el resultado de la comparación. Para poder compararlos, debe conservar los valores.

Las *variables* tienen un nombre (la palabra que se usa para hacer referencia al valor contenido por la variable, también un número o un numero y una letra del Abecedario ej: n1, sueldo1, nota2, entre otros.). Las variables también tienen un tipo de datos (que determina el tipo de datos que puede almacenar la variable). Las variables pueden representar una matriz si tienen que almacenar un conjunto indexado de elementos de datos estrechamente relacionados.

Tipos de datos

El tipo de datos de un elemento de programación hace referencia *al tipo de datos que puede contener y cómo almacena los datos*. Los tipos de datos se aplican a todos los valores que se pueden almacenar en la memoria del equipo o participar en la evaluación de una expresión

A continuación, se muestra una tabla con los tipos de datos para ser usados en Visual Basic:

TIPO DE DATOS	ABREVIATURA	MEMORIA REQUERIDA	RANGO DE VALORES	EQUIVALENCIA CON USADAS EN PSEUDOCÓDIGO	OBSERVACIONES
Integer (Entero)	%	2 bytes ó 4 bytes según versión	- 32768 a 32767 ó -2.147.483.648 a 2.147.483.647 según versión	Entero	Uso en contadores, control de bucles etc.
Long (Entero largo)	&	4 bytes ó 8 bytes según versión	- 2147483648 a 2147483647 ó -9,2E+18 a 9,2E+18 según versión	Entero	Igual que integer pero admite un rango más amplio
Single (Decimal simple)	!	4 bytes	$-3{,}4\cdot10^{38}$ a $3{,}4\cdot10^{38}$	Real	Hasta 6 decimales o más según versión. También admite enteros
Double (Decimal doble)	#	8 bytes	$-1{,}79\cdot10^{308}$ a $1{,}79\cdot10^{308}$	Real	Hasta 14 decimales o más según versión. También admite enteros
Boolean (Booleano)	No tiene	2 bytes	True o False	Booleano	False es el valor predeterminado y va asociado a cero
String	$	10 bytes + 1 byte por cada carácter	0 a $2\cdot10^{9}$ caracteres	Alfanumérica	Texto de longitud variable
Variant (Object en otras versiones)	No tiene	Variable	Los del tipo double para números o los del tipo string para texto	No tiene	Admite números enteros, decimales o text

Tabla N1 Tipos de datos en Visual BASIC Net

Declaración de Variables

Para declarar una variable en Visual Basic Net, se usa la instrucción **Dim** y se especifica el nombre de la variable la instrucción **As** y el **tipo de variable.**

Su sintaxis será:

```
Dim [Nombre de variable] As [Tipo de variable]
```

Operadores en Visual Basic Net

Visual Basic net clasifica los operadores con que trabaja de la siguiente manera:

· **Operadores aritméticos:** operadores utilizados para ejecutar cálculos matemáticos. suma +, Resta - Multiplicación*, División /.

· **Operadores de comparación**: operadores utilizados para efectuar comparaciones.

· **Operadores de concatenación**: operadores utilizados para combinar cadenas de caracteres.

· **Operadores lógicos**: operadores usados para realizar operaciones lógicas.

Operadores Aritméticos	Operadores de comparación	Operaciones de concatenación	Operadores lógicos
Exponente ^	< (Menor que)	&	Operador And
Multiplicación *	<= (Menor o igual que)	+	Operador Eqv
División /	> (Mayor que)		Operador Imp
Operador Mod	>= (Mayor o igual que)		Operador Not
Operador +	= (Igual a)		Operador Or
Operador -	<> (Distinto de)		Operador Xor

Tabla N 2 Operadores en Visual BASIC Net

La mayoría de estos operadores han sido explicados en el curso “Bases de la programación nivel I” de aprenderaprogramar.com. El operador & tiene interés para

concatenar cadenas. Por ejemplo "Hoy puede" & "ser un gran día" equivale a "Hoy puede ser un gran día". Te remitimos a la ayuda de Visual Basic para obtener mayor información sobre operadores. Ten en cuenta que puede haber pequeñas diferencias entre las distintas versiones de Visual Basic.

En cuanto a las funciones matemáticas, algunas de las funciones disponibles son las siguientes:

Funciones Matemáticas	Significado	Funciones Matemáticas	Significado
Abs	Valor absoluto	Log	Logaritmo neperiano
Atn ó Atan según versión	Arcotangente	Rnd	Generación de número aleatorio
Cos	Coseno	Sgn ó Sign según versión	Signo de un número
Exp	Exponenciación	Sin	Seno
Fix	Truncamiento tipo 1	Sqr ó Sqrt según versión	Raíz cuadrada
Int	Truncamiento tipo 2	Tan	Tangente

Tabla N3 Funciones matemáticas en Visual BASIC Net

Estructura de programación secuencial

Cuando en un problema sólo participan operaciones, entradas y salidas se la denomina una estructura secuencial.

Los problemas diagramados y codificados previamente emplean solo estructuras secuenciales.

La programación requiere una práctica ininterrumpida de diagramación y codificación de problemas.

Problema:

Realizar la carga de dos números enteros por teclado e imprimir su suma y su producto.

Tenemos dos entradas num1 y num2, dos operaciones: realizar la suma y el producto de los valores ingresados y dos salidas, que son los resultados de la suma y el producto de los valores ingresados. En el símbolo de impresión podemos indicar una o más salidas, eso queda a criterio del programador, lo mismo para indicar las entradas por teclado.

Programa:

```
Module Module1

    Sub Main()

        Dim num1, num2, suma, producto As Integer

        Console.Write("Ingrese primer valor:")

        num1 = Console.ReadLine ()

        Console.Write ("Ingrese segundo valor:")

        num2 = Console.ReadLine()

        suma = num1 + num2

        producto = num1 * num2

        Console.Write ("La suma de los dos valores es:")

        Console.WriteLine (suma)

        Console.Write ("El producto de los dos valores es:")

        Console.WriteLine (producto)

        Console.ReadKey()
```

```
    End Sub

End Module
```

Recordemos que tenemos que seguir todos los pasos vistos para la creación de un proyecto (fijar un nombre de proyecto, seleccionar la carpeta donde se almacenará etc.)

Algunas cosas nuevas que podemos notar:

Podemos definir varias variables en la misma línea:

```
Dim num1, num2, suma, producto As Integer
```

Si llamamos a la función WriteLine en lugar de Write, la impresión siguiente se efectuará en la próxima línea:

```
Console.WriteLine(suma)
```

Problemas propuestos:

Realizar la carga del lado de un cuadrado, mostrar por pantalla el perímetro del mismo (El perímetro de un cuadrado se calcula multiplicando el valor del lado por cuatro)

Escribir un programa en el cual se ingresen cuatro números, calcular e informar la suma de los dos primeros y el producto del tercero y el cuarto.

Realizar un programa que lea cuatro valores numéricos e informar su suma y promedio.

Se debe desarrollar un programa que pida el ingreso del precio de un artículo y la cantidad que lleva el cliente. Mostrar lo que debe abonar el comprador.

Estructuras condicionales simples y compuestas

No todos los problemas pueden resolverse empleando estructuras secuenciales. Cuando hay que tomar una decisión aparecen las estructuras condicionales. En nuestra vida diaria se nos presentan situaciones donde debemos decidir. ¿Elijo la carrera A o la carrera B?, ¿Me pongo este pantalón?

Para ir al trabajo, ¿Elijo el camino A o el camino B? Al cursar una carrera, ¿Elijo el turno mañana, tarde o noche?.

Por supuesto que en un problema se combinan estructuras secuenciales y condicionales.

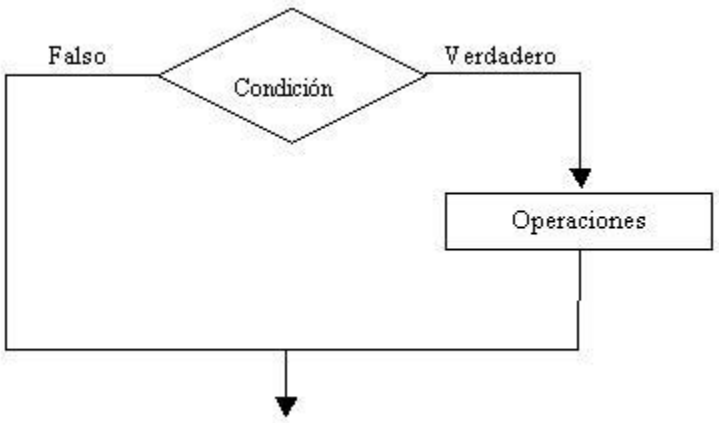
Unidad III

PROGRAMAS CONDICIONALES Y CICLOS FOR Y WHILE

Estructura condicional simple.

Cuando se presenta la elección tenemos la opción de realizar una actividad o no realizar ninguna.

Representación gráfica:



Podemos observar: El rombo representa la condición. Hay dos opciones que se pueden tomar. Si la condición da verdadera se sigue el camino del verdadero, o sea el de la derecha, si la condición da falsa se sigue el camino de la izquierda donde no hay ninguna actividad.

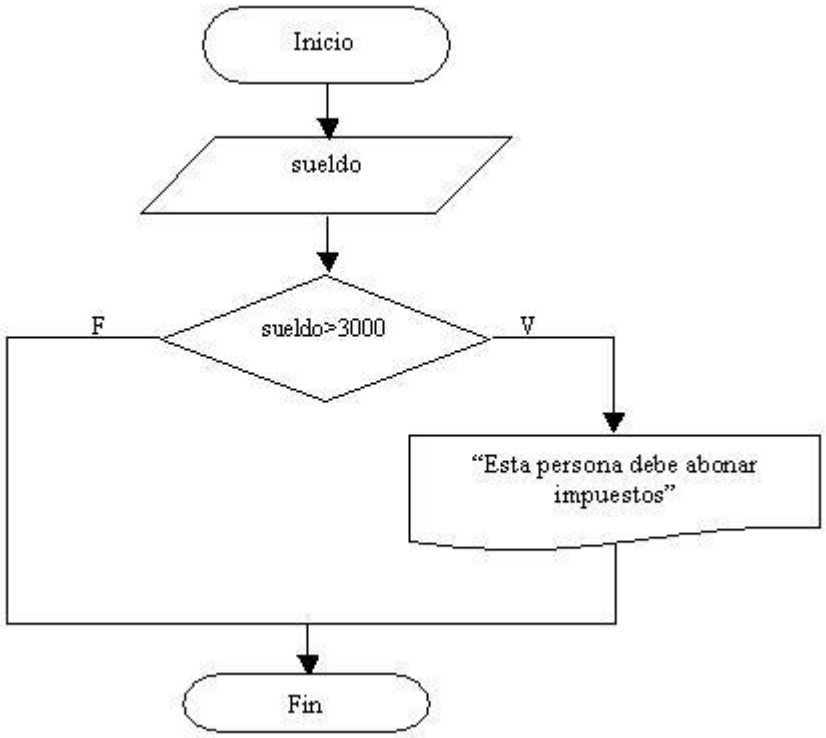
Se trata de una estructura CONDICIONAL SIMPLE porque por el camino del verdadero hay actividades y por el camino del falso no hay actividades.

Por el camino del verdadero pueden existir varias operaciones, entradas y salidas, inclusive ya veremos que puede haber otras estructuras condicionales.

Problema:

Ingresar el sueldo de una persona, si supera los 3000 pesos mostrar un mensaje en pantalla indicando que debe abonar impuestos.

Diagrama de flujo:



Podemos observar lo siguiente: Siempre se hace la carga del sueldo, pero si el sueldo que ingresamos supera 3000 pesos se mostrará por pantalla el mensaje "Esta persona debe abonar impuestos", en caso que la persona cobre 3000 o menos no aparece nada por pantalla.

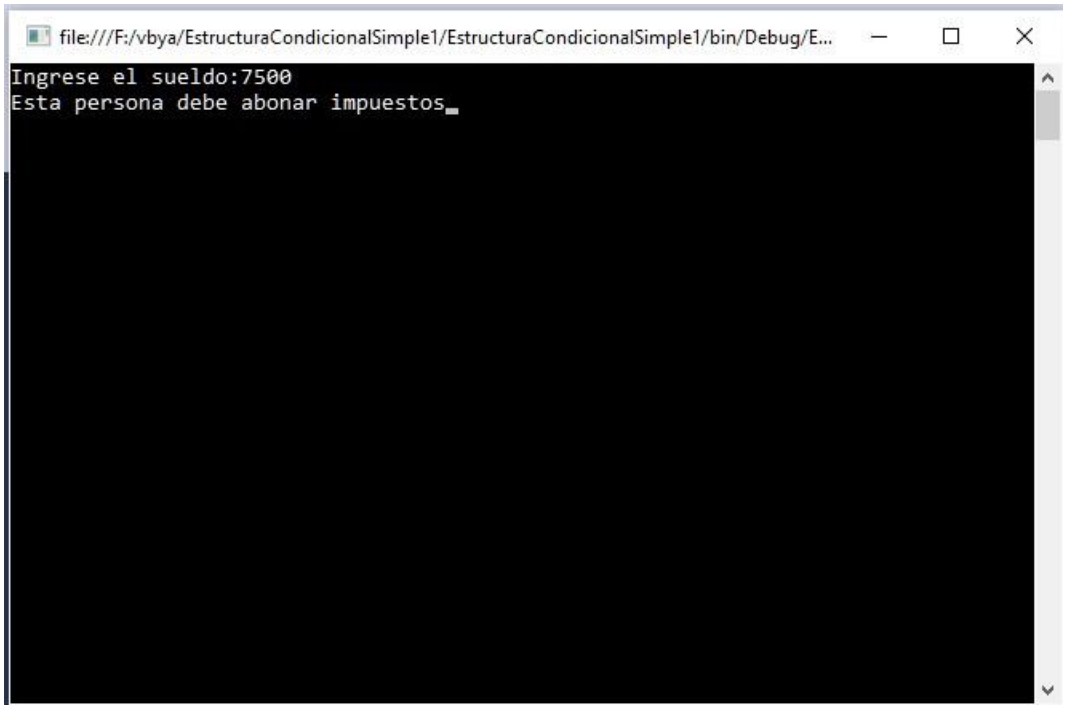
Programa:

Module Module1

```
    Sub Main()
        Dim sueldo As Single
        Console.Write("Ingrese el sueldo:")
        sueldo = Console.ReadLine()
        If sueldo > 3000 Then
            Console.Write ("Esta persona debe abonar impuestos")
        End If
        Console.ReadKey()
    End Sub
End Module
```

La palabra clave "If" indica que estamos en presencia de una estructura condicional; seguidamente disponemos la condición. La actividad dentro del If finaliza con la palabra clave End If.

Es común que el contenido del If se indente (corrido hacia la derecha) para una fácil lectura del programa. Ejecutando el programa e ingresamos un sueldo superior a 3000 pesos. Podemos observar cómo aparece en pantalla el mensaje "Esta persona debe abonar impuestos", ya que la condición del If es verdadera.

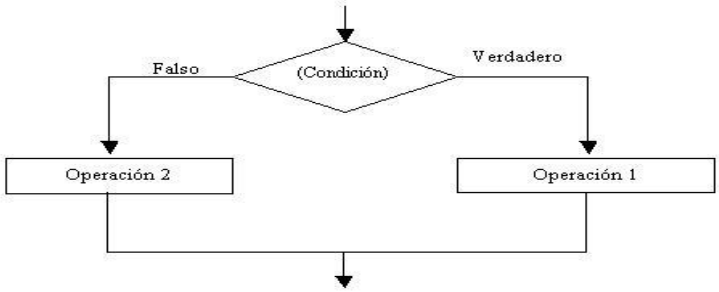


Volvamos a ejecutar el programa y carguemos un sueldo menor o igual a 3000 pesos. No debe aparecer mensaje en pantalla.

Estructura condicional compuesta.

Cuando se presenta la elección tenemos la opción de realizar una actividad u otra. Es decir tenemos actividades por el verdadero y por el falso de la condición. Lo más importante que hay que tener en cuenta que se realizan las actividades de la rama del verdadero o las del falso, NUNCA se realizan las actividades de las dos ramas.

Representación gráfica:

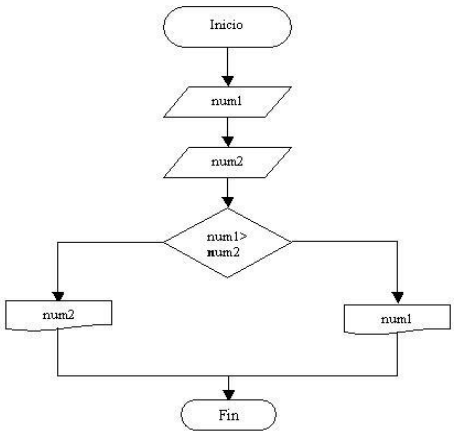


En una estructura condicional compuesta tenemos entradas, salidas, operaciones, tanto por la rama del verdadero como por la rama del falso.

Problema:

Realizar un programa que solicite ingresar dos números distintos y muestre por pantalla el mayor de ellos.

Diagrama de flujo:



Se hace la entrada de num1 y num2 por teclado. Para saber cual variable tiene un valor mayor preguntamos si el contenido de num1 es mayor (>) que el contenido de num2, si la respuesta es verdadera vamos por la rama de la derecha e imprimimos num1, en caso que la condición sea falsa vamos por la rama de la izquierda (Falsa) e imprimimos num2. Como podemos observar nunca se imprimen num1 y num2 simultáneamente. Estamos en presencia de una ESTRUCTURA CONDICIONAL COMPUESTA ya que tenemos actividades por la rama del verdadero y del falso.

Programa:

```
Module Module1

    Sub Main()

        Dim num1, num2 As Integer
```

```
        Console.Write("Ingrese primer valor:")
        num1 = Console.ReadLine()
        Console.Write("Ingrese segundo valor:")
        num2 = Console.ReadLine()
        If num1 > num2 Then
            Console.Write(num1)
        Else
            Console.Write(num2)
        End If
        Console.ReadKey()
    End Sub

End Module
```

Cotejemos el diagrama de flujo y la codificación y observemos que el primer bloque después del If representa la rama del verdadero y el segundo bloque después de la palabra clave Else representa la rama del falso.

Compilemos el programa, si hubo errores sintácticos corrijamos y carguemos dos valores, como por ejemplo:

Ingrese el primer valor: 10

Ingrese el segundo valor: 4

10

Si ingresamos los valores 10 y 4 la condición del If retorna verdadero y ejecuta el primer bloque. Un programa se controla y corrige probando todos sus posibles resultados. Ejecutemos nuevamente el programa e ingresemos:

Ingrese el primer valor: 10

Ingrese el segundo valor: 54

54

Cuando a un programa le corregimos todos los errores sintácticos y lógicos ha terminado nuestra tarea y podemos entregar el mismo al USUARIO que nos lo solicitó. Hay que tener en cuenta que al disponer una condición debemos seleccionar que operador relacional se adapta a la pregunta.

Ejemplos:

Se ingresa un número multiplicarlo por 10 si es distinto a 0. (<>)

Se ingresan dos números mostrar una advertencia si son iguales. (=)

Los problemas que se pueden presentar son infinitos y la correcta elección del operador sólo se alcanza con la práctica intensiva en la resolución de problemas.

Problemas propuestos

1. Realizar un programa que lea por teclado dos números, si el primero es mayor al segundo informar su suma y diferencia, en caso contrario informar el producto y la división del primero respecto al segundo.

2. Se ingresan tres notas de un alumno, si el promedio es mayor o igual a siete mostrar un mensaje "Promocionado".

3. Se ingresa por teclado un número positivo de uno o dos dígitos (1..99) mostrar un mensaje indicando si el número tiene uno o dos dígitos.

(Tener en cuenta que condición debe cumplirse para tener dos dígitos, un número entero)

Estructuras condicionales anidadas

Decimos que una estructura condicional es anidada cuando por la rama del verdadero o del falso de una estructura condicional hay otra estructura condicional.

Problema:

Confeccionar un programa que pida por teclado tres notas de un alumno, calcule el promedio e imprima alguno de estos mensajes:
Si el promedio es >=7 mostrar "Promocionado".
Si el promedio es >=4 y <7 mostrar "Regular".
Si el promedio es <4 mostrar "Reprobado".

Primeramente preguntamos si el promedio es superior o igual a 7, en caso afirmativo va por la rama del verdadero de la estructura condicional mostramos un mensaje que indica "Promocionado" (con comillas indicamos un texto que debe imprimirse en pantalla).

En caso que la condición nos de falso, por la rama del falso aparece otra estructura condicional, porque todavía debemos averiguar si el promedio del alumno es superior o igual a cuatro o inferior a cuatro. Estamos en presencia de dos estructuras condicionales compuestas.

Programa:

```
Module Module1

    Sub Main()
        Dim nota1, nota2, nota3, promedio As Integer
        Console.Write("Ingrese primer nota:")
        nota1 = Console.ReadLine()
        Console.Write("Ingrese segunda nota:")
```

```
        nota2 = Console.ReadLine()
        Console.Write("Ingrese tercer nota:")
        nota3 = Console.ReadLine()
        promedio = (nota1 + nota2 + nota3) / 3
        If promedio >= 7 Then
            Console.Write("Promocionado")
        Else
            If promedio >= 4 Then
                Console.Write("Regular")
            Else
                Console.Write("Reprobado")
            End If
        End If
        Console.ReadKey()
    End Sub
End Module
```

Codifiquemos y ejecutemos este programa. Al correr el programa deberá solicitar por teclado la carga de tres notas y mostrarnos un mensaje según el promedio de las mismas. Podemos definir un conjunto de variables del mismo tipo en una misma línea:

Dim nota1, nota2, nota3, promedio As Integer

Esto no es obligatorio pero a veces, por estar relacionadas, conviene. A la codificación del If anidado podemos observarla por el **else** del primer If.

Problemas propuestos

1. Se cargan por teclado tres números distintos. Mostrar por pantalla el mayor de ellos.
2. Se ingresa por teclado un valor entero, mostrar una leyenda que indique si el número es positivo, nulo o negativo.

3. Confeccionar un programa que permita cargar un número entero positivo de hasta tres cifras y muestre un mensaje indicando si tiene 1, 2, o 3 cifras. Mostrar un mensaje de error si el número de cifras es mayor.
4. Un postulante a un empleo, realiza un test de capacitación, se obtuvo la siguiente información: cantidad total de preguntas que se le realizaron y la cantidad de preguntas que contestó correctamente. Se pide confeccionar un programa que ingrese los dos datos por teclado e informe el nivel del mismo según el porcentaje de respuestas correctas que ha obtenido, y sabiendo que:

Nivel máximo: Porcentaje>=90%.

Nivel medio: Porcentaje>=75% y <90%.

Nivel regular:Porcentaje>=50% y <75%.

Fuera de nivel: Porcentaje<50%.

Estructuras condicionales con condiciones compuestas y operadores lógicos.

Hasta ahora hemos visto los operadores:

relacionales (>, <, >=, <= , =, <>)

matemáticos (+, -, *, /, Mod)

pero nos están faltando otros operadores imprescindibles:

lógicos (And, Or).

Estos dos operadores se emplean fundamentalmente en las estructuras condicionales para agrupar varias condiciones simples.

Traducido se lo lee como “Y”. Si la Condición 1 es verdadera Y la condición 2 es verdadera luego ejecutar la rama del verdadero. Cuando vinculamos dos o más condiciones con el operador "And", las dos condiciones deben ser verdaderas para que

el resultado de la condición compuesta de Verdadero y continúe por la rama del verdadero de la estructura condicional.

La utilización de operadores lógicos permite en muchos casos plantear algoritmos más cortos y comprensibles.

Problema:

Confeccionar un programa que lea por teclado tres números distintos y nos muestre el mayor

La primera estructura condicional es una ESTRUCTURA CONDICIONAL COMPUESTA con una CONDICION COMPUESTA. Podemos leerla de la siguiente forma: Si el contenido de la variable **num1** es mayor al contenido de la variable **num2** Y si el contenido de la variable **num1** es mayor al contenido de la variable **num3** entonces la CONDICION COMPUESTA resulta Verdadera. Si una de las condiciones simples da falso la CONDICION COMPUESTA da Falso y continua por la rama del falso. Es decir que se mostrará el contenido de num1 si y sólo si **num1>num2 y num1>num3**. En caso de ser Falsa la condición, analizamos el contenido de **num2 y num3** para ver cual tiene un valor mayor.
En esta segunda estructura condicional no se requieren operadores lógicos al haber una condición simple.

Programa:

```
Module Module1

    Sub Main()
        Dim num1, num2, num3 As Integer
        Console.Write("Ingrese primer valor:")
        num1 = Console.ReadLine()
        Console.Write("Ingrese segundo valor:")
        num2 = Console.ReadLine()
        Console.Write("Ingrese tercer valor:")
        num3 = Console.ReadLine()
```

```
        If num1 > num2 And num1 > num3 Then
            Console.Write(num1)
        Else
            If num2 > num3 Then
                Console.Write(num2)
            Else
                Console.Write(num3)
            End If
        End If
        Console.ReadKey()
    End Sub


End Module
```

Operador Or

Traducido se lo lee como "O". Si la condición primera es Verdadera o la condición segunda es Verdadera, luego ejecutar la rama del Verdadero. Cuando vinculamos dos o más condiciones con el operador "Or", con que una de las dos condiciones sea Verdadera alcanza para que el resultado de la condición compuesta sea Verdadero.

Problema:

Se carga una fecha (día, mes y año) por teclado. Mostrar un mensaje si corresponde al primer trimestre del año (enero, febrero o marzo) Cargar por teclado el valor numérico del día, mes y año. Ejemplo:

dia:10 mes:1 año:2016.

La carga de una fecha se hace por partes, ingresamos las variables día, mes y año. Mostramos el mensaje "Corresponde al primer trimestre" en caso que el mes ingresado por teclado sea igual a 1, 2 ó 3. En la condición no participan las variables día y año.

Programa:

```
Module Module1

    Sub Main()
        Dim dia, mes, año As Integer

        Console.Write("Ingrese nro de día:")
        dia = Console.ReadLine()
        Console.Write("Ingrese nro de mes:")
        mes = Console.ReadLine()
        Console.Write("Ingrese nro de año:")
        año = Console.ReadLine()
        If mes = 1 Or mes = 2 Or mes = 3 Then
            Console.Write("Corresponde al primer trimestre")
        End If
        Console.ReadLine()
    End Sub

End Module
```

Problemas propuestos

1. Realizar un programa que pida cargar una fecha cualquiera, luego verificar si dicha fecha corresponde a Navidad.
2. Se ingresan tres valores por teclado, si todos son iguales se imprime la suma del primero con el segundo y a este resultado se lo multiplica por el tercero.
3. Se ingresan por teclado tres números, si todos los valores ingresados son menores a 10, imprimir en pantalla la leyenda "Todos los números son menores a diez".
4. Se ingresan por teclado tres números, si al menos uno de los valores ingresados es menor a 10, imprimir en pantalla la leyenda "Alguno de los números es menor a diez".

5. Escribir un programa que pida ingresar la coordenada de un punto en el plano, es decir dos valores enteros x e y (distintos a cero). Posteriormente imprimir en pantalla en que cuadrante se ubica dicho punto. (1º Cuadrante si x > 0 Y y > 0 , 2º Cuadrante: x < 0 Y y > 0, etc.)
6. De un operario se conoce su sueldo y los años de antigüedad. Se pide confeccionar un programa que lea los datos de entrada e informe:
   a) Si el sueldo es inferior a 500 y su antigüedad es igual o superior a 10 años, otorgarle un aumento del 20 %, mostrar el sueldo a pagar.
   b)Si el sueldo es inferior a 500 pero su antigüedad es menor a 10 años, otorgarle un aumento de 5 %.
   c) Si el sueldo es mayor o igual a 500 mostrar el sueldo en pantalla sin cambios.
7. Escribir un programa en el cual: dada una lista de tres valores numéricos distintos se calcule e informe su rango de variación (debe mostrar el mayor y el menor de ellos)

Estructura repetitiva Do While (condición) ... Loop

Hasta ahora hemos empleado estructuras SECUENCIALES y CONDICIONALES. Existe otro tipo de estructuras tan importantes como las anteriores que son las estructuras REPETITIVAS.

Una estructura repetitiva permite ejecutar una instrucción o un conjunto de instrucciones varias veces.

Una ejecución repetitiva de sentencias se caracteriza por:
- Las sentencias que se repiten. El test o prueba de condición antes de cada repetición, que motivará que se repitan o no las sentencias.

Estructura repetitiva Do While (condición) ... Loop

Funcionamiento: En primer lugar se verifica la condición, si la misma resulta verdadera se ejecutan las operaciones que indicamos por la rama del Verdadero.

A la rama del verdadero la graficamos en la parte inferior de la condición. Una línea al final del bloque de repetición la conecta con la parte superior de la estructura repetitiva. En caso que la condición sea Falsa continúa por la rama del Falso y sale de la estructura repetitiva para continuar con la ejecución del algoritmo.

El bloque se repite **MIENTRAS** la condición sea Verdadera.

Importante: Si la condición siempre retorna verdadero estamos en presencia de un ciclo repetitivo infinito. Dicha situación es un error de programación, nunca finalizará el programa.

Problema 1:

Realizar un programa que imprima en pantalla los números del 1 al 100.

Si no conocemos las estructuras repetitivas podemos resolver el problema empleando una estructura secuencial. Inicializamos una variable con el valor 1, luego imprimimos la variable, incrementamos nuevamente la variable y así sucesivamente.

La primera operación inicializa la variable x en 1, seguidamente comienza la estructura repetitiva while y disponemos la siguiente condición (x <= 100), se lee MIENTRAS la variable x sea menor o igual a 100.

Al ejecutarse la condición retorna VERDADERO porque el contenido de x (1) es menor o igual a 100. Al ser la condición verdadera se ejecuta el bloque de instrucciones que contiene la estructura while que finaliza en la palabra clave Loop. El bloque de instrucciones contiene una salida y una operación.
Se imprime el contenido de x, y seguidamente se incrementa la variable x en uno.

La operación x = x + 1 se lee como "en la variable x se guarda el contenido de x más 1". Es decir, si x contiene 1 luego de ejecutarse esta operación se almacenará en x un 2.

Al finalizar el bloque de instrucciones que contiene la estructura repetitiva se verifica nuevamente la condición de la estructura repetitiva y se repite el proceso explicado anteriormente.

Mientras la condición retorne verdadero se ejecuta el bloque de instrucciones; al retornar falso la verificación de la condición se sale de la estructura repetitiva y continúa el algoritmo, en este caso finaliza el programa.

Lo más difícil es la definición de la condición de la estructura while y qué bloque de instrucciones se van a repetir. Observar que si, por ejemplo, disponemos la condición x >=100 (si x es mayor o igual a 100) no provoca ningún error sintáctico pero estamos en presencia de un error lógico porque al evaluarse por primera vez la condición retorna falso y no se ejecuta el bloque de instrucciones que queríamos repetir 100 veces.

No existe una RECETA para definir una condición de una estructura repetitiva, sino que se logra con una práctica continua solucionando problemas.

Una vez planteado el diagrama debemos verificar si el mismo es una solución válida al problema (en este caso se debe imprimir los números del 1 al 100 en pantalla), para ello podemos hacer un seguimiento del flujo del diagrama y los valores que toman las variables a lo largo de la ejecución:

x

1

2

3

4

.

.

100

101 Cuando x vale 101 la condición de la estructura repetitiva retorna falso, en este caso finaliza el diagrama.

Importante: Podemos observar que el bloque repetitivo puede no ejecutarse ninguna vez si la condición retorna falso la primera vez. La variable x debe estar inicializada con algún valor antes que se ejecute la operación x = x + 1 en caso de no estar inicializada aparece un error de compilación.

Programa:

```
Module Module1


    Sub Main()
        Dim x As Integer
        x = 1
        Do While x <= 100
            Console.Write(x)
            Console.Write(" - ")
            x = x + 1
        Loop
        Console.ReadKey ()
    End Sub


End Module
```

Recordemos que un problema no estará 100% solucionado si no hacemos el programa en Visual Basic .Net que muestre los resultados buscados.

Probemos algunas modificaciones de este programa y veamos qué cambios se deberían hacer para:

1 - Imprimir los números del 1 al 500.

2 - Imprimir los números del 50 al 100.

3 - Imprimir los números del -50 al 0.

4 - Imprimir los números del 2 al 100 pero de 2 en 2 (2,4,6,8 ....100).

Respuestas:

1 - Debemos cambiar la condición del Do While con x<=500.

2 - Debemos inicializar x con el valor 50.

3 - Inicializar x con el valor -50 y fijar la condición x<=0.

4 - Inicializar a x con el valor 2 y dentro del bloque repetitivo incrementar a x en 2

x = x + 2

Problema 2:

Escribir un programa que solicite la carga de un valor positivo y nos muestre desde 1 hasta el valor ingresado de uno en uno. Ejemplo: Si ingresamos 30 se debe mostrar en pantalla los números del 1 al 30.

A la prueba del diagrama la podemos realizar dándole valores a las variables; por ejemplo, si ingresamos 5 el seguimiento es el siguiente:

n x

5 1 (Se imprime el contenido de x)

2 " "

3 " "

4 " "

5 " "

6 (Sale del while porque 6 no es menor o igual a 5)

Programa:

```
Module Module1

    Sub Main()
        Dim n, x As Integer
        Console.Write("Ingrese el valor final:")
        n = Console.ReadLine()
        x = 1
        Do While x <= n
            Console.Write(x)
            Console.Write(" - ")
            x = x + 1
        Loop
        Console.ReadKey()
    End Sub

End Module
```

Los nombres de las variables n y x pueden ser palabras o letras (como en este caso). La variable x recibe el nombre de CONTADOR. Un contador es un tipo especial de variable que se incrementa o decremento con valores constantes durante la ejecución del programa. El contador x nos indica en cada momento la cantidad de valores impresos en pantalla.

Problema 3:

Desarrollar un programa que permita la carga de 10 valores por teclado y nos muestre posteriormente la suma de los valores ingresados y su promedio.

En este problema, a semejanza de los anteriores, llevamos un CONTADOR llamado x que nos sirve para contar las vueltas que debe repetir el while. También aparece el concepto de ACUMULADOR (un acumulador es un tipo especial de

variable que se incrementa o decrementa con valores variables durante la ejecución del programa)

Hemos dado el nombre de suma a nuestro acumulador. Cada ciclo que se repita la estructura repetitiva, la variable suma se incrementa con el contenido ingresado en la variable valor.

La prueba del diagrama se realiza dándole valores a las variables:

valor	suma	x	promedio
	0	0	

(Antes de entrar a la estructura repetitiva estos son los valores).

5	5	1
16	21	2
7	28	3
10	38	4
2	40	5
20	60	6
5	65	7
5	70	8
10	80	9
2	82	10
8	90	11
9		

Este es un seguimiento del diagrama planteado. Los números que toma la variable valor dependerá de qué cifras cargue el operador durante la ejecución del programa. El promedio se calcula al salir de la estructura repetitiva (es decir primero sumamos los 10 valores ingresados y luego los dividimos por 10)

Hay que tener en cuenta que cuando en la variable valor se carga el primer número (en este ejemplo 5) al cargarse el segundo valor (16) el valor anterior 5 se pierde, por ello la necesidad de ir almacenando en la variable "suma" los valores ingresados.

Programa:

```
Module Module1


    Sub Main()
        Dim x, suma, valor, promedio As Integer
        x = 1
        suma = 0
        Do While x <= 10
            Console.Write("Ingrese un valor:")
            valor = Console.ReadLine()
            suma = suma + valor
            x = x + 1
        Loop
        promedio = suma / 10
        Console.Write("La suma de los 10 valores es:")
        Console.WriteLine(suma)
        Console.Write("El promedio es:")
        Console.Write(promedio)
        Console.ReadKey()
    End Sub
End Module
```

Problema 4:

Una planta que fabrica perfiles de hierro posee un lote de n piezas. Confeccionar un programa que pida ingresar por teclado la cantidad de piezas a procesar y luego ingrese la longitud de cada perfil; sabiendo que la pieza cuya longitud esté comprendida en el rango de 1,20 y 1,30 son aptas. Imprimir por pantalla la cantidad de piezas aptas que hay en el lote.

Podemos observar que dentro de una estructura repetitiva puede haber estructuras condicionales (inclusive puede haber otras estructuras repetitivas que veremos más adelante)

En este problema hay que cargar inicialmente la cantidad de piezas a ingresar (n), seguidamente se cargan n valores de largos de piezas. Cada vez que ingresamos un largo de pieza (largo) verificamos si es una medida correcta (debe estar entre 1,20 y 1,30 el largo para que sea correcta), en caso de ser correcta la CONTAMOS (incrementamos la variable cantidad en 1)

Al contador cantidad lo inicializamos en cero porque inicialmente no se ha cargado ningún largo de pieza. Cuando salimos de la estructura repetitiva porque se han cargado n largos de piezas mostramos por pantalla el contador cantidad (que representa la cantidad de piezas aptas)

En este problema tenemos dos CONTADORES:

x (Cuenta la cantidad de piezas cargadas hasta el momento)

cantidad (Cuenta los perfiles de hierro aptos)

Programa:

```
Module Module1
    Sub Main()
        Dim x, cantidad, n As Integer
        Dim largo As Single
        x = 1
```

```
        cantidad = 0
        Console.Write("Cuantas piezar procesará:")
        n = Console.ReadLine()
        Do While x <= n
            Console.Write("Ingrese la medida de la pieza:")
            largo = Console.ReadLine()
            If largo >= 1.2 And largo <= 1.3 Then
                cantidad = cantidad + 1
            End If
            x = x + 1
        Loop
        Console.Write("La cantidad de piezas aptas son:")
        Console.Write(cantidad)
        Console.ReadKey()
    End Sub
End Module
```

Problemas propuestos:

1. Ha llegado la parte fundamental, que es el momento donde uno desarrolla individualmente un algoritmo para la resolución de problemas.
2. El tiempo a dedicar a esta sección EJERCICIOS PROPUESTOS debe ser mucho mayor que el empleado a la sección de EJERCICIOS RESUELTOS. La experiencia dice que debemos dedicar el 80% del tiempo a la resolución individual de problemas y el otro 20% al análisis y codificación de problemas ya resueltos por otras personas.
   Es de vital importancia para llegar a ser un buen PROGRAMADOR poder resolver problemas en forma individual.
3. Escribir un programa que solicite ingresar 10 notas de alumnos y nos informe cuántos tienen notas mayores o iguales a 7 y cuántos menores.
4. Se ingresan un conjunto de n alturas de personas por teclado. Mostrar la altura promedio de las personas.

5. En una empresa trabajan n empleados cuyos sueldos oscilan entre $100 y $500, realizar un programa que lea los sueldos que cobra cada empleado e informe cuántos empleados cobran entre $100 y $300 y cuántos cobran más de $300. Además, el programa deberá informar el importe que gasta la empresa en sueldos al personal.
6. Realizar un programa que imprima 25 términos de la serie 11 - 22 - 33 - 44, etc. (No se ingresan valores por teclado)
7. Mostrar los múltiplos de 8 hasta el valor 500. Debe aparecer en pantalla 8 - 16 - 24, etc.
8. Realizar un programa que permita cargar dos listas de 15 valores cada una. Informar con un mensaje cuál de las dos listas tiene un valor acumulado mayor (mensajes "Lista 1 mayor", "Lista 2 mayor", "Listas iguales") Tener en cuenta que puede haber dos o más estructuras repetitivas en un algoritmo.
9. Desarrollar un programa que permita cargar n números enteros y luego nos informe cuántos valores fueron pares y cuántos impares. Emplear el operador "%" en la condición de la estructura condicional:

```
If valor Mod 2=0 Then        //Si el if da verdadero luego es par.
```

Estructura repetitiva Do ... Loop While (condición)

La estructura Do ... Loop While (condición) permite ejecutar al menos una vez su bloque repetitivo, a diferencia del Do While (condición) ... Loop que puede no ejecutar el bloque. Esta estructura repetitiva se utiliza cuando conocemos de antemano que por lo menos una vez se ejecutará el bloque repetitivo.
La condición de la estructura está abajo del bloque a repetir, a diferencia del Do While (condición) ... Loop que está en la parte superior.

El bloque de operaciones se repite MIENTRAS que la condición sea Verdadera. Si la condición retorna Falso el ciclo se detiene. Es importante analizar y ver que las operaciones se ejecutan como mínimo una vez.

Problema 1:

Escribir un programa que solicite la carga de un número entre 0 y 999, y nos muestre un mensaje de cuántos dígitos tiene el mismo. Finalizar el programa cuando se cargue el valor 0.

En este problema por lo menos se carga un valor. Si se carga un valor mayor o igual a 100 se trata de un número de tres cifras, si es mayor o igual a 10 se trata de un valor de dos dígitos, en caso contrario se trata de un valor de un dígito. Este bloque se repite mientras se ingresa en la variable valor un número distinto a cero. Cuando se ingresa el cero la condición del ciclo repetitivo se evalúa falsa y por lo tanto sale del bloque repetitivo.

Programa:

```
Module Module1


    Sub Main()
        Dim valor As Integer
        Do
            Console.Write("Ingrese un valor entre 0 y 999 (0 finaliza):")
            valor = Console.ReadLine()
            If valor >= 100 Then
                Console.WriteLine("Tiene 3 dígitos.")
            Else
                If valor >= 10 Then
                    Console.WriteLine("Tiene 2 dígitos.")
                Else
                    Console.WriteLine("Tiene 1 dígito.")
                End If
            End If
        Loop While (valor <> 0)
    End Sub
```

End Module

Problema 2:

Escribir un programa que solicite la carga de números por teclado, obtener su promedio. Finalizar la carga de valores cuando se cargue el valor 0.

Cuando la finalización depende de algún valor ingresado por el operador conviene el empleo de la estructura Do ... Loop While (condición), por lo menos se cargará un valor (en el caso más extremo se carga 0, que indica la finalización de la carga de valores)

Definimos un contador cant que cuenta la cantidad de valores ingresados por el operador (no lo incrementa si ingresamos 0) El valor 0 no es parte de la serie de valores que se deben sumar.

Definimos el acumulador suma que almacena todos los valores ingresados por teclado. La estructura repetitiva Do ... Loop While (condición) se repite mientras se ingrese un valor distinto a 0. Con el valor cero la condición del ciclo retorna falso y continúa con el flujo del diagrama. Disponemos por último una estructura condicional para el caso que el operador cargue únicamente un 0 y por lo tanto no podemos calcular el promedio ya que no existe la división por 0.

En caso que el contador cant tenga un valor distinto a 0 el promedio se obtiene dividiendo el acumulador suma por el contador cant que tiene la cantidad de valores ingresados antes de introducir el 0.

Programa:

```
Module Module1

    Sub Main()
        Dim suma, cant, valor, promedio As Integer
        suma = 0
        cant = 0
        Do
```

```
            Console.Write("Ingrese un valor (0 para finalizar):")
            valor = Console.ReadLine()
            If valor <> 0 Then
                suma = suma + valor
                cant = cant + 1
            End If
        Loop While valor <> 0
        If cant <> 0 Then
            promedio = suma / cant
            Console.Write("El promedio de los valores ingresados es:")
            Console.Write(promedio)
        Else
            Console.Write("No se ingresaron valores.")
        End If
        Console.ReadKey()
    End Sub
End Module
```

El contador cant DEBE inicializarse antes del ciclo, lo mismo que el acumulador suma. El promedio se calcula siempre y cuando el contador cant sea distinto a 0.

Problema 3:

Realizar un programa que permita ingresar el peso (en kilogramos) de piezas. El proceso termina cuando ingresamos el valor 0. Se debe informar:
a) Cuántas piezas tienen un peso entre 9,8 Kg. y 10,2 Kg.?, cuántas con más de 10,2 Kg.? y cuántas con menos de 9,8 Kg.?

b) La cantidad total de piezas procesadas.

Los tres contadores cont1, cont2, y cont3 se inicializan en 0 antes de entrar a la estructura repetitiva. A la variable suma no se la inicializa en 0 porque no es un acumulador, sino que guarda la suma del contenido de las variables cont1, cont2 y cont3.

La estructura se repite mientrasno se ingrese el valor 0 en la variable peso. Este valor

no se lo considera un peso menor a 9,8 Kg., sino que indica que ha finalizado la carga de valores por teclado.

Programa:

```
Module Module1

    Sub Main()
        Dim cant1, cant2, cant3, suma As Integer
        Dim peso As Single
        cant1 = 0
        cant2 = 0
        cant3 = 0
        Do
            Console.Write("Ingrese el peso de la pieza (0 pera finalizar):")
            peso = Console.ReadLine()
            If peso > 10.2 Then
                cant1 = cant1 + 1
            Else
                If peso >= 9.8 Then
                    cant2 = cant2 + 1
                Else
                    If peso > 0 Then
                        cant3 = cant3 + 1
                    End If
                End If
            End If
        Loop While peso <> 0
        suma = cant1 + cant2 + cant3
        Console.Write("Piezas aptas:")
        Console.WriteLine(cant2)
        Console.Write("Piezas con un peso superior a 10,2:")
        Console.WriteLine(cant1)
```

```
        Console.Write("Piezas con un peso inferior a 9,8:")
        Console.WriteLine(cant3)
        Console.ReadKey()
    End Sub
End Module
```

Problemas propuestos:

Realizar un programa que acumule (sume) valores ingresados por teclado hasta ingresar el 9999 (no sumar dicho valor, indica que ha finalizado la carga). Imprimir el valor acumulado e informar si dicho valor es cero, mayor a cero o menor a cero.

En un banco se procesan datos de las cuentas corrientes de sus clientes. De cada cuenta corriente se conoce: número de cuenta y saldo actual. El ingreso de datos debe finalizar al ingresar un valor negativo en el número de cuenta.
Se pide confeccionar un programa que lea los datos de las cuentas corrientes e informe:
a) De cada cuenta: número de cuenta y estado de la cuenta según su saldo, sabiendo que:

Estado de la cuenta 'Acreedor' si el saldo es >0.

'Deudor' si el saldo es <0.

'Nulo' si el saldo es =0.

b) La suma total de los saldos acreedores.

UNIDAD III

PROGRAMAS CONDICIONALES Y CICLOS FOR Y WHILE

Estructura repetitiva For... Next

Cualquier problema que requiera una estructura repetitiva se puede resolver empleando la estructura Do While (condición) ... Loop. Pero hay otra estructura repetitiva cuyo planteo es más sencillo en ciertas situaciones.
En general, la estructura For... Next se usa en aquellas situaciones en las cuales CONOCEMOS la cantidad de veces que queremos que se ejecute el bloque de instrucciones. Ejemplo: cargar 10 números, ingresar 5 notas de alumnos, etc. Conocemos de antemano la cantidad de veces que queremos que el bloque se repita.

En su forma más típica y básica, esta estructura requiere una variable que cumple la función de un CONTADOR de vueltas. El primer valor que indicamos es el valor inicial que toma el contador. Cada vez que se ejecuta el bloque a repetir el contador se incrementa en uno y si no supera el valor final especificado en el For vuelve a ejecutar el bloque.

Si conocemos la cantidad de veces que se repite el bloque es muy sencillo emplear un For, por ejemplo si queremos que se repita 50 veces el bloque de instrucciones puede hacerse así:

La variable del For puede tener cualquier nombre. En este ejemplo se la ha definido con el nombre f. Analicemos el ejemplo:

- La variable f toma inicialmente el valor 1.

- Se ejecutan la/s operación/es.

- Al finalizar de ejecutarlas automáticamente se incrementa en 1 el contador f.

- Si el contador f no supera a 50 vuelve a ejecutar el bloque.

- El proceso se repetirá hasta que la variable f llegue al valor 51.

En este momento finaliza el ciclo repetitivo.

La variable f no conviene modificarla dentro del For. La variable f puede ser inicializada en cualquier valor y finalizar en cualquier valor.

Problema 1:

Realizar un programa que imprima en pantalla los números del 1 al 100.

Inicialmente f vale 1 y como no es superior a 100 se ejecuta el bloque, imprimimos el contenido de f, al finalizar el bloque repetitivo se incrementa la variable f en 1, como 2 no es superior a 100 se repite el bloque de instrucciones. Cuando la variable del For llega a 101 sale de la estructura repetitiva y continúa la ejecución del algoritmo que se indica después del círculo. La variable f (o como sea que se decida llamarla) debe estar definida como una variable más.

Programa:

```
Module Module1

    Sub Main()

        Dim f As Integer

        For f = 1 To 100

            Console.Write(f)

            Console.Write("-")

        Next

        Console.ReadKey()

    End Sub

End Module
```

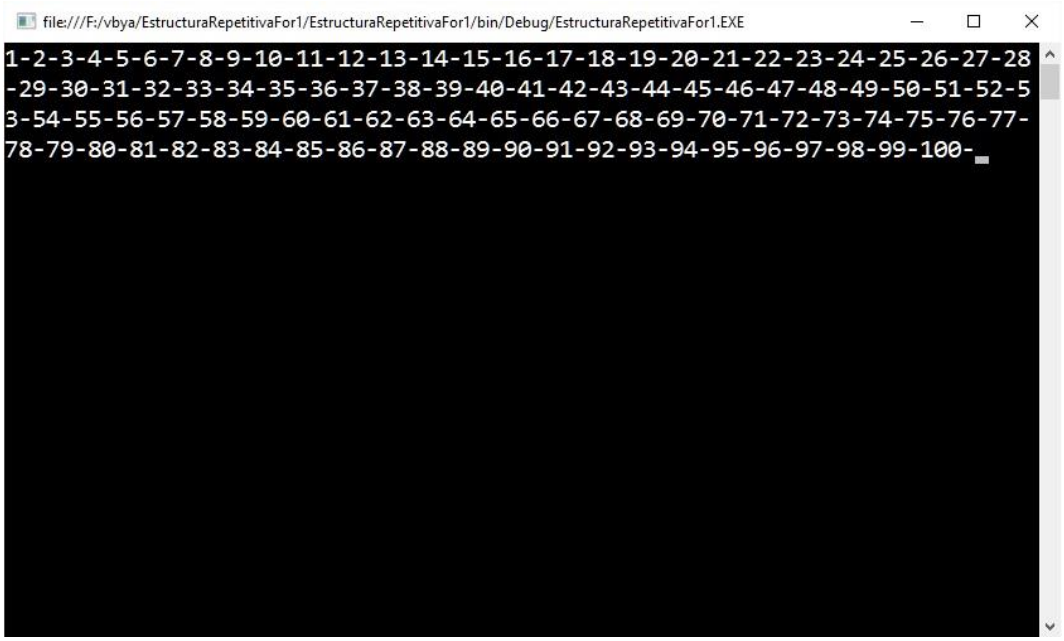
El comienzo del bloque For es:

For f = 1 To 100

f previamente está definida como variable entera.

El ciclo repetitivo For finaliza con la palabra clave: Next

La ejecución del programa da como resultado la siguiente pantalla:



Problema 2:

: Desarrollar un programa que permita la carga de 10 valores por teclado y nos muestre posteriormente la suma de los valores ingresados y su promedio. Este problema ya lo desarrollamos, lo resolveremos empleando la estructura for

En este caso, a la variable del for (f) sólo se la requiere para que se repita el bloque de instrucciones 10 veces.

Programa:

Module Module1

```
Sub Main()
    Dim suma, f, valor, promedio As Integer
    suma = 0
    For f = 1 To 10
        Console.Write("Ingrese valor:")
        valor = Console.ReadLine()
        suma = suma + valor
    Next
    Console.Write("La suma es:")
    Console.WriteLine(suma)
    promedio = suma / 10
    Console.Write("El promedio es:")
    Console.Write(promedio)
    Console.ReadKey()

End Sub

End Module
```

El problema requiere que se carguen 10 valores y se sumen los mismos. El promedio se calcula fuera del For luego de haber cargado los 10 valores.

Problema 3:

Escribir un programa que solicite por teclado 10 notas de alumnos y nos informe cuántos tienen notas mayores o iguales a 7 y cuántos menores.

Para resolver este problema se requieren tres contadores:

aprobados (Cuenta la cantidad de alumnos aprobados)

reprobados (Cuenta la cantidad de reprobados)

f (es el contador del For)

Dentro de la estructura repetitiva debemos hacer la carga de la variable nota y verificar con una estructura condicional si el contenido de la variable nota es mayor o igual a 7 para incrementar el contador aprobados, en caso de que la condición retorne falso debemos incrementar la variable reprobados. En este caso, a la variable del for (f) sólo se la requiere para que se repita el bloque de instrucciones 10 veces.

Programa:

```
Module Module1


    Sub Main()
        Dim suma, f, valor, promedio As Integer
        suma = 0
        For f = 1 To 10
            Console.Write("Ingrese valor:")
            valor = Console.ReadLine()
            suma = suma + valor
        Next
        Console.Write("La suma es:")
        Console.WriteLine(suma)
        promedio = suma / 10
        Console.Write("El promedio es:")
        Console.Write(promedio)
        Console.ReadKey()
    End Sub


End Module
```

El problema requiere que se carguen 10 valores y se sumen los mismos. El promedio se calcula fuera del For luego de haber cargado los 10 valores.

Problema 3:

Escribir un programa que solicite por teclado 10 notas de alumnos y nos informe cuántos tienen notas mayores o iguales a 7 y cuántos menores.

Para resolver este problema se requieren tres contadores:

aprobados (Cuenta la cantidad de alumnos aprobados)

reprobados (Cuenta la cantidad de reprobados)

f (es el contador del For)

Dentro de la estructura repetitiva debemos hacer la carga de la variable nota y verificar con una estructura condicional si el contenido de la variable nota es mayor o igual a 7 para incrementar el contador aprobados, en caso de que la condición retorne falso debemos incrementar la variable reprobados.

Los contadores aprobados y reprobados deben imprimirse FUERA de la estructura repetitiva.
Es fundamental inicializar los contadores aprobados y reprobados en cero antes de entrar a la estructura For. **Importante:** Un error común es inicializar los contadores dentro de la estructura repetitiva. En caso de hacer esto los contadores se fijan en cero en cada ciclo del For, por lo que al finalizar el For como máximo el contador puede tener el valor 1.

Programa:

```
Module Module1

    Sub Main()

        Dim aprobados, reprobados, f, nota As Integer
        aprobados = 0
        reprobados = 0
        For f = 1 To 10
            Console.Write("Ingrese la nota:")
            nota = Console.ReadLine()
```

```
            If nota >= 7 Then
                aprobados = aprobados + 1
            Else
                reprobados = reprobados + 1
            End If
        Next
        Console.Write("Cantidad de aprobados:")
        Console.WriteLine(aprobados)
        Console.Write("Cantidad de reprobados:")
        Console.Write(reprobados)
        Console.ReadKey()
    End Sub
End Module
```

Definimos todas las variables de tipo entera:

```
Dim  aprobados, reprobados, f, nota As Integer
```

Iniciamos los dos contadores en cero previo al for:

```
aprobados = 0

reprobados = 0
```

El contador del For se inicia con el valor 1:

```
For f = 1 To 10
```

Dentro del ciclo For cargamos la nota y según el valor ingresado incrementamos uno de los dos contadores:

```
            Console.Write("Ingrese la nota:")

            nota = Console.ReadLine()

            If nota >= 7 Then
```

```
            aprobados = aprobados + 1
        Else
            reprobados = reprobados + 1
        End If
```

Fuera del ciclo For procedemos a mostrar la cantidad de aprobados y reprobados:

```
        Console.Write("Cantidad de aprobados:")
        Console.WriteLine(aprobados)
        Console.Write("Cantidad de reprobados:")
        Console.Write(reprobados)
```

Problema 4:

Escribir un programa que lea 10 números enteros y luego muestre cuántos valores ingresados fueron múltiplos de 3 y cuántos de 5. Debemos tener en cuenta que hay números que son múltiplos de 3 y de 5 a la vez

Tengamos en cuenta que el operador matemático Mod retorna el resto de dividir un valor por otro, en este caso: valor Mod 3 retorna el resto de dividir el valor que ingresamos por teclado, por tres. Veamos: si ingresamos 6 el resto de dividirlo por 3 es 0, si ingresamos 12 el resto de dividirlo por 3 es 0. Generalizando: cuando el resto de dividir por 3 al valor que ingresamos por teclado es cero, se trata de un múltiplo de dicho valor.

Ahora bien ¿por qué no hemos dispuesto una estructura If anidada? Porque hay valores que son múltiplos de 3 y de 5 a la vez. Por lo tanto con If anidados no podríamos analizar los dos casos. Es importante darse cuenta cuando conviene emplear If anidados y cuando no debe emplearse.

Programa:

```
Module Module1
```

```
    Sub Main()
        Dim mul3, mul5, valor, f As Integer
        mul3 = 0
        mul5 = 0
        For f = 1 To 10
            Console.Write("Ingrese un valor:")
            valor = Console.ReadLine()
            If valor Mod 3 = 0 Then
                mul3 = mul3 + 1
            End If
            If valor Mod 5 = 0 Then
                mul5 = mul5 + 1
            End If
        Next
        Console.Write("Cantidad de valores ingresados múltiplos de 3:")
        Console.WriteLine(mul3)
        Console.Write("Cantidad de valores ingresados múltiplos de 5:")
        Console.Write(mul5)
        Console.ReadKey()
    End Sub

End Module
```

El If para verificar si el valor cargado es múltiplo de 3 (el contador mul3 lo iniciamos en cero antes del For):

```
            If valor Mod 3 = 0 Then

                mul3 = mul3 + 1

            End If
```

Problema 5:

Escribir un programa que lea n números enteros y calcule la cantidad de valores mayores o iguales a 1000. Este tipo de problemas también se puede resolver empleando la estructura repetitiva for. Lo primero que se hace es cargar una variable que indique la cantidad de valores a ingresar. Dicha variable se carga antes de entrar a la estructura repetitiva For.La estructura For permite que el valor inicial o final dependa de una variable cargada previamente por teclado.

Tenemos un contador llamado cantidad y f que es el contador del For. La variable entera n se carga previo al inicio del For, por lo que podemos fijar el valor final del for con la variable n, es decir previo a que inicie el For ya sabemos la cantidad de veces que debe repetirse. Por ejemplo, si el operador carga 5 en n la estructura repetitiva For se ejecutará 5 veces. La variable valor se ingresa dentro de la estructura repetitiva, y se verifica si el valor de la misma es mayor o igual a 1000, en dicho caso se incrementa en uno el contador cantidad. Fuera de la estructura repetitiva imprimimos el contador cantidad que tiene almacenado la cantidad de valores ingresados mayores o iguales a 1000.

Programa:

```
Module Module1

    Sub Main()

        Dim cantidad, n, f, valor As Integer

        cantidad = 0

        Console.Write("Cuantos valores ingresará:")

        n = Console.ReadLine()

        For f = 1 To n

            Console.Write("Ingrese el valor:")

            valor = Console.ReadLine()
```

```
            If valor >= 1000 Then
                cantidad = cantidad + 1
            End If
        Next
        Console.Write("La cantidad de valores ingresados mayores o iguales a 1000 son:")
        Console.Write(cantidad)
        Console.ReadKey()
    End Sub
End Module
```

Otra variante de la estructura repetitiva For...Next

Podemos en ciertos casos hacer que el contador del for se incremente en un valor distinto a uno. Veamos cómo sería el programa para imprimir los números de 1 al 3 pero incrementando en 0,25 cada paso:

```
Module Module1
    Sub Main()
        Dim f As Single
        For f = 1 To 3 Step 0.25
            Console.Write(f)
            Console.Write(" - ")
        Next
        Console.ReadKey()
```

```
End Sub

End Module
```

El contador del For es una variable de tipo Single (es decir podemos guardar valores con decimal)

Aparece la nueva palabra clave Step dentro del For que le indica cuanto debe modificarse el contador en cada vuelta (cuando no indicamos Step luego el incremento por defecto es 1):

```
For f = 1 To 3 Step 0.25
```

Problemas propuestos

1. Ha llegado nuevamente la parte fundamental, que es el momento donde uno desarrolla individualmente un algoritmo para la resolución de un problema.

2. Confeccionar un programa que lea n pares de datos, cada par de datos corresponde a la medida de la base y la altura de un triángulo. El programa deberá informar:
   a) De cada triángulo la medida de su base, su altura y su superficie.
   b) La cantidad de triángulos cuya superficie es mayor a 12.

3. Desarrollar un programa que solicite la carga de 10 números e imprima la suma de los últimos 5 valores ingresados.

4. Desarrollar un programa que muestre la tabla de multiplicar del 5 (del 5 al 50).

5. Confeccionar un programa que permita ingresar un valor del 1 al 10 y nos muestre la tabla de multiplicar del mismo (los primeros 12 términos) Ejemplo: Si ingreso 3 deberá aparecer en pantalla los valores 3, 6, 9, hasta el 36.

6. Realizar un programa que lea los lados de n triángulos, e informar:
   a) De cada uno de ellos, qué tipo de triángulo es: equilátero (tres lados iguales), isósceles (dos lados iguales), o escaleno (ningún lado igual)
   b) Cantidad de triángulos de cada tipo.

7. Escribir un programa que pida ingresar coordenadas (x,y) que representan puntos en el plano. Informar cuántos puntos se han ingresado en el primer, segundo, tercer y cuarto cuadrante. Al comenzar el programa se pide que se ingrese la cantidad de puntos a procesar.

8. Se realiza la carga de 10 valores enteros por teclado. Se desea conocer:
   a) La cantidad de valores ingresados negativos.
   b) La cantidad de valores ingresados positivos.
   c) La cantidad de múltiplos de 15.
   d) El valor acumulado de los números ingresados que son pares.

9. Se cuenta con la siguiente información:
   Las edades de 50 estudiantes del turno mañana.
   Las edades de 60 estudiantes del turno tarde.
   Las edades de 110 estudiantes del turno noche.
   Las edades de cada estudiante deben ingresarse por teclado.
   a) Obtener el promedio de las edades de cada turno (tres promedios)
   b) Imprimir dichos promedios (promedio de cada turno)
   c) Mostrar por pantalla un mensaje que indique cual de los tres turnos tiene un promedio de edades mayor.

Cadenas de caracteres

En Visual Basic .Net hemos visto que cuando queremos almacenar un valor entero definimos una variable de tipo Integer, si queremos almacenar un valor con decimales definimos una variable de tipo Single. Ahora si queremos almacenar una cadena de

caracteres (por ejemplo un nombre de una persona) debemos definir una variable de tipo String.

Más adelante veremos en profundidad y detenimiento los conceptos del manejo de String, por ahora solo nos interesa la mecánica para trabajar con cadenas de caracteres.

Problema 1:

Solicitar el ingreso del nombre y edad de dos personas. Mostrar el nombre de la persona con mayor edad.

Programa:

```
Module Module1

    Sub Main()
        Dim nombre1, nombre2 As String
        Dim edad1, edad2 As Integer
        Console.Write("Ingrese el nombre:")
        nombre1 = Console.ReadLine()
        Console.Write("Ingrese edad:")
        edad1 = Console.ReadLine()
        Console.Write("Ingrese el nombre:")
        nombre2 = Console.ReadLine()
        Console.Write("Ingrese edad:")
        edad2 = Console.ReadLine()
        Console.Write("La persona de mayor edad es:")
        If edad1 = edad2 Then
            Console.Write("Tienen la misma edad")
        Else
            If (edad1 > edad2) Then
                Console.Write(nombre1)
            Else
                Console.Write(nombre2)
```

```
            End If
        End If
        Console.ReadKey()
    End Sub

End Module
```

Para almacenar un nombre debemos definir una variable de tipo String y su ingreso por teclado se hace llamando al método ReadLine del objeto Console:

```
nombre1 = Console.ReadLine()
```

Problema 2:

Solicitar el ingreso de dos nombres de personas. Mostrar un mensaje si son iguales o distintos.

Programa:

```
Module Module1

    Sub Main()

        Dim nombre1, nombre2 As String

        Console.Write("Ingrese primer nombre:")

        nombre1 = Console.ReadLine()

        Console.Write("Ingrese segundo segundo:")

        nombre2 = Console.ReadLine()

        If nombre1 = nombre2 Then

            Console.Write("Los nombres son iguales")

        Else
```

```
            Console.Write("Los nombres son distintos")

        End If

        Console.ReadKey()

    End Sub

End Module
```

Para comparar si el contenido de dos String es igual se utiliza el operador = como si se estuvieran comparando dos enteros. La condición se verifica verdadero si los contenidos de los dos String son exactamente iguales, es decir si cargamos "Martinez" en nombre1 y "martinez" en nombre2 luego retorna falso ya que no es lo mismo la "M" mayúscula y la "m" minúscula.

Declaración de métodos con parámetros y/o retorno de datos

Cuando uno plantea una clase en lugar de especificar todo el algoritmo en un único método (lo que hicimos en los primeros pasos de este tutorial) es dividir todas las responsabilidades de las clase en un conjunto de métodos.

Un método hemos visto que tiene la siguiente sintaxis:

```
Public Sub [nombre del método]()

    [algoritmo]

End Sub
```

Veremos que hay varios tipos de métodos:

Métodos con parámetros.

Un método puede tener parámetros:

```
Public Sub [nombre del método]([parámetros])

    [algoritmo]
```

```
End Sub
```

Los parámetros los podemos imaginar como variables locales al método, pero su valor se inicializa con datos que llegan cuando lo llamamos.

Problema 1:

Confeccionar una clase que permita ingresar valores enteros por teclado y nos muestre la tabla de multiplicar de dicho valor. Finalizar el programa al ingresar el -1.

Programa:

```
Module Module1

    Public Class TablaMultiplicar

        Public Sub CargarValor()

            Dim valor As Integer

            Do

                Console.Write("Ingrese un valor (-1 para finalizar):")

                valor = Console.ReadLine()

                If valor <> -1 Then

                    Calcular(valor)

                End If

            Loop While valor <> -1

        End Sub


        Public Sub Calcular(ByVal v As Integer)

            Dim f, tabla As Integer
```

```
            For f = 1 To 10
                tabla = f * v
                Console.Write(tabla)
                Console.Write("-")
            Next
            Console.WriteLine()
        End Sub
    End Class
    Sub Main()
        Dim tablamultiplicar As New TablaMultiplicar()
        tablamultiplicar.CargarValor()
    End Sub
End Module
```

En esta clase no hemos definido ningún atributo, no se requieren.

El método Calcular recibe un parámetro llamado "v" de tipo entero, luego lo utilizamos dentro del método para mostrar la tabla de multiplicar de dicho valor:

```
        Public Sub Calcular(ByVal v As Integer)
            Dim f, tabla As Integer
            For f = 1 To 10
                tabla = f * v
                Console.Write(tabla)
                Console.Write("-")
```

```
    Next

    Console.WriteLine()

End Sub
```

Como vemos los parámetros se indican entre paréntesis luego del nombre del método. Se le antecede la palabra clave ByVal al nombre del parámetro)

Un método puede no tener parámetros como hemos visto en problemas anteriores o puede tener uno o más parámetros (en caso de tener más de un parámetro los mismos se separan por coma)

El método CargarValores no tiene parámetros y tiene por objetivo cargar un valor entero por teclado y llamar al método Calcular para que muestre la tabla de multiplicar del valor que le pasamos por teclado:

```
Public Sub CargarValor()

    Dim valor As Integer

    Do

        Console.Write("Ingrese un valor (-1 para finalizar):")

        valor = Console.ReadLine()

        If valor <> -1 Then

            Calcular(valor)

        End If

    Loop While valor <> -1

End Sub
```

Como vemos al método Calcular lo llamamos por su nombre y entre paréntesis le pasamos el dato a enviar (debe ser un valor o variable entera)

En este problema en la Main solo llamamos al método CargarValor, ya que el método Calcular luego es llamado por el método CargarValor:

```
Sub Main()

    Dim tablamultiplicar As New TablaMultiplicar()

    tablamultiplicar.CargarValor()

End Sub
```

Métodos que retornan un dato.

Un método puede retornar un dato:

```
Public Function [nombre del método]([parámetros]) As [Tipo de dato]

 [algoritmo]

 return [dato]

End Function
```

Cuando queremos que un método retorne un dato debemos definir una Function en lugar de un procedimiento Sub.

Cuando un método retorna un dato al final de la declaración de la función indicamos el tipo de dato que devuelve. Luego dentro del algoritmo en el momento que queremos que finalice el mismo y retorne el dato empleamos la palabra clave Return con el valor respectivo.

Problema 2:

Confeccionar una clase que permita ingresar tres valores por teclado. Luego mostrar el mayor y el menor.

Programa:

```
Module Module1
```

```
Public Class MayorMenor

    Public Sub cargarValores()

        Dim valor1, valor2, valor3, mayor, menor As Integer

        Console.Write("Ingrese primer valor:")

        valor1 = Console.ReadLine()

        Console.Write("Ingrese segundo valor:")

        valor2 = Console.ReadLine()

        Console.Write("Ingrese tercer valor:")

        valor3 = Console.ReadLine()

        mayor = CalcularMayor(valor1, valor2, valor3)

        menor = CalcularMenor(valor1, valor2, valor3)

        Console.WriteLine("El valor mayor de los tres es:" & mayor)

        Console.WriteLine("El valor menor de los tres es:" & menor)

    End Sub


    Public Function CalcularMayor(ByVal v1 As Integer, ByVal v2 As Integer, ByVal
v3 As Integer) As Integer

        Dim m As Integer
        If v1 > v2 And v1 > v3 Then
            m = v1
        Else
            If v2 > v3 Then
                m = v2
            Else
```

```
            m = v3
        End If
    End If
    Return m
End Function

Public Function CalcularMenor(ByVal v1 As Integer, ByVal v2 As Integer, ByVal
v3 As Integer) As Integer

    Dim m As Integer
    If v1 < v2 And v1 < v3 Then
        m = v1
    Else
        If v2 < v3 Then
            m = v2
        Else
            m = v3
        End If
    End If
    Return m
End Function
End Class

Sub Main()
    Dim mm As New MayorMenor()
    mm.cargarValores()
    Console.ReadKey()
End Sub
End Module
```

Si vemos la sintaxis que calcula el mayor de tres valores enteros es similar al algoritmo visto en conceptos anteriores:

```
Public Function CalcularMayor(ByVal v1 As Integer, ByVal v2 As Integer, ByVal
v3 As Integer) As Integer

    Dim m As Integer

    If v1 > v2 And v1 > v3 Then

       m = v1
    Else
       If v2 > v3 Then
          m = v2
       Else
          m = v3
       End If
    End If
    Return m
End Function
```

Lo primero que podemos observar que el método retorna un entero y recibe tres parámetros, como retorna un dato utilizamos la palabra clave Función:

```
Public Function CalcularMayor (ByVal v1 As Integer, ByVal v2 As Integer, ByVal
v3 As Integer) As Integer
```

Dentro del método verificamos cuál de los tres parámetros almacena un valor mayor, a este valor lo almacenamos en una variable local llamada "m", al valor almacenado en esta variable lo retornamos al final con un Return.

La llamada al método calcularMayor lo hacemos desde dentro del método CargarCalores:

```
mayor = CalcularMayor(valor1, valor2, valor3)
```

Debemos asignar a una variable el valor devuelto por el método CalcularMayor. Luego el contenido de la variable mayor lo mostramos:

```
Console.WriteLine("El valor mayor de los tres es:" & mayor)
```

Para mostrar un String y un Integer en el método WriteLine podemos utilizar el operador & para concatenar los dos datos. Esta línea la veníamos escribiendo:

```
Console.Write("El valor mayor de los tres es:")

Console.WriteLine (mayor)
```

La lógica es similar para el cálculo del menor

Estructura de datos tipo vector

Hemos empleado variables de distinto tipo para el almacenamiento de datos (variables Integer, Single, String) En esta sección veremos otros tipos de variables que permiten almacenar un conjunto de datos en una única variable.

Un vector es una estructura de datos que permite almacenar un CONJUNTO de datos del MISMO tipo.Con un único nombre se define un vector y por medio de un subíndice hacemos referencia a cada elemento del mismo (componente)

Problema 1:

Se desea guardar los sueldos de 5 operarios. Según lo conocido deberíamos definir 5 variables si queremos tener en un cierto momento los 5 sueldos almacenados en memoria. Empleando un vector solo se requiere definir un único nombre y accedemos a cada elemento por medio del subíndice.

Para definir una propiedad de una clase de tipo vector utilizamos la siguiente sintaxis:

```
Private sueldos(4) As Integer
```

Estamos definiendo un vector llamado sueldos que puede almacenar 5 componentes (esto debido a que las componentes comienzan a numerarse en cero).

Los elementos o componentes del vector sueldos solo pueden almacenar valores enteros. No podemos tener componentes de distinto tipo.

Programa:

```
Module Module1
    Public Class PruebaVector1
        Private sueldos(4) As Integer
        Public Sub Cargar()
            Dim f As Integer
            For f = 0 To 4
                Console.Write("Ingrese importe del sueldo:")
                sueldos(f) = Console.ReadLine()
            Next
        End Sub

        Public Sub Imprimir()
            Dim f As Integer
            For f = 0 To 4
                Console.WriteLine(sueldos(f))
            Next
            Console.ReadKey()
        End Sub
    End Class

    Sub Main()
        Dim pv As New PruebaVector1()
        pv.Cargar()
        pv.Imprimir()
```

```
    End Sub
End Module
```

Como atributo de la clase declaramos el vector:

```
Private sueldos(4) As Integer
```

Lo definimos como atributo de la clase ya que lo utilizaremos en los dos métodos.

Para cargar cada componente debemos indicar entre paréntesis que elemento del vector estamos accediendo, es común utilizar un ciclo For para indicar el índice de la componente:

```
Public Sub Cargar()
  Dim f As Integer
  For f = 0 To 4
    Console.Write("Ingrese importe del sueldo:")
    sueldos(f) = Console.ReadLine()
  Next
End Sub
```

Cuando f vale cero estamos accediendo a la primer componente del vector (en nuestro caso sería):

```
sueldos(f) = Console.ReadLine()
```

Lo mas común es utilizar una estructura repetitiva For para recorrer cada componente del vector.

Utilizar el For nos reduce la cantidad de código, si no utilizo un For debería en forma secuencial implementar el siguiente código:

```
Console.Write("Ingrese importe del sueldo:")
```

```
        sueldos(0) = Console.ReadLine()

        Console.Write("Ingrese importe del sueldo:")

        sueldos(1) = Console.ReadLine()

        Console.Write("Ingrese importe del sueldo:")

        sueldos(2) = Console.ReadLine()

        Console.Write("Ingrese importe del sueldo:")

        sueldos(3) = Console.ReadLine()

        Console.Write("Ingrese importe del sueldo:")

        sueldos(4) = Console.ReadLine()
```

La impresión de las componentes del vector lo hacemos en el otro método y también utilizamos un For para reducir el código a programar:

```
    Public Sub Imprimir()

      Dim f As Integer

      For f = 0 To 4

        Console.WriteLine(sueldos(f))

      Next

      Console.ReadKey()

    End Sub
```

Siempre que queremos acceder a una componente del vector debemos indicar entre paréntesis la componente, dicho valor comienza a numerarse en cero y continua hasta el valor que indicamos al definir el vector, en nuestro caso creamos el vector indicando un 4, es decir tiene 5 elementos:

```
    Private sueldos(4) As Integer
```

Por último en este programa creamos un objeto en la Main y llamamos a lo métodos de Cargar e Imprimir el vector:

```
Sub Main()
    Dim pv As New PruebaVector1()
    pv.Cargar()
    pv.Imprimir()
End Sub
```

Problema 2:

Definir un vector de 5 componentes de tipo Single que representen las alturas de 5 personas. Obtener el promedio de las mismas. Contar cuántas personas son más altas que el promedio y cuántas más bajas.

Programa:

```
Module Module1

    Public Class PruebaVector2
        Private alturas(4) As Single
        Private promedio As Single


        Public Sub Cargar()
            Dim f As Integer
            For f = 0 To 4
                Console.Write("Ingrese la altura de la persona:")
                alturas(f) = Console.ReadLine()

            Next
        End Sub


        Public Sub CalcularPromedio()

            Dim suma As Single
```

```
        suma = 0
        Dim f As Integer
        For f = 0 To 4
            suma = suma + alturas(f)
        Next
        promedio = suma / 5
        Console.WriteLine("Promedio de alturas:" & promedio)
    End Sub


    Public Sub MayoresMenores()

        Dim may, men As Integer
        may = 0
        men = 0
        Dim f As Integer
        For f = 0 To 4
            If alturas(f) > promedio Then
                may = may + 1
            Else
                If alturas(f) < promedio Then
                    men = men + 1
                End If
            End If
        Next
        Console.WriteLine("Cantidad de personas mayores al promedio:" & may)
        Console.WriteLine("Cantidad de personas menores al promedio:" & men)
        Console.ReadKey()
    End Sub
End Class

Sub Main()
```

```
        Dim pv2 As New PruebaVector2()
        pv2.Cargar()
        pv2.CalcularPromedio()
        pv2.MayoresMenores()
    End Sub

End Module
```

Definimos como atributo un vector donde almacenaremos las alturas. También definimos un atributo "promedio" ya que se lo calculará en un método y lo consultará en otro:

```
    Private alturas(4) As Single

    Private promedio As Single
```

Procedemos seguidamente a cargar todos sus elementos en el método Cargar:

```
    Public Sub Cargar()

        Dim f As Integer

        For f = 0 To 4

            Console.Write("Ingrese la altura de la persona:")

            alturas(f) = Console.ReadLine()

        Next

    End Sub
```

En otro método procedemos a sumar todas sus componentes y obtener el promedio. El promedio lo almacenamos en un atributo de la clase ya que lo necesitamos en otro método:

```
    Public Sub CalcularPromedio()
        Dim suma As Single
```

```
    suma = 0
    Dim f As Integer
    For f = 0 To 4
        suma = suma + alturas(f)
    Next
    promedio = suma / 5
    Console.WriteLine("Promedio de alturas:" & promedio)
End Sub
```

Por último en un tercer método comparamos cada componente del vector con el atributo promedio, si el valor almacenado supera al promedio incrementamos un contador en caso que sea menor al promedio incrementamos otro contador:

```
Public Sub MayoresMenores()
    Dim may, men As Integer
    may = 0
    men = 0
    Dim f As Integer
    For f = 0 To 4
        If alturas(f) > promedio Then
            may = may + 1
        Else
            If alturas(f) < promedio Then
                men = men + 1
            End If
        End If
    Next
    Console.WriteLine("Cantidad de personas mayores al promedio:" & may)
    Console.WriteLine("Cantidad de personas menores al promedio:" & men)
    Console.ReadKey()
End Sub
```

Importante:

En este problema podemos observar una ventaja de tener almacenadas todas las alturas de las personas. Si no conociéramos los vectores tenemos que cargar otra vez las alturas por teclado para compararlas con el promedio.Mientras el programa está en ejecución tenemos el vector alturas a nuestra disposición. Es importante tener en cuenta que cuando finaliza la ejecución del programa se pierde el contenido de todas las variables (simples y vectores)

Problema 3:

Una empresa tiene dos turnos (mañana y tarde) en los que trabajan 8 empleados (4 por la mañana y 4 por la tarde). Confeccionar un programa que permita almacenar los sueldos de los empleados agrupados por turno.Imprimir los gastos en sueldos de cada turno.

Programa:

```
Module Module1
    Public Class PruebaVector3

        Private turnoMan(3) As Single
        Private turnoTar(3) As Single

        Public Sub Cargar()
            Console.WriteLine("Sueldos de empleados del turno de la mañana.")
            Dim f As Integer
            For f = 0 To 3
                Console.Write("Ingrese sueldo:")
                turnoMan(f) = Console.ReadLine()
            Next
            Console.WriteLine("Sueldos de empleados del turno de la tarde.")
```

```
        For f = 0 To 3
            Console.Write("Ingrese sueldo:")
            turnoTar(f) = Console.ReadLine()
        Next
    End Sub

    Public Sub CalcularGastos()
        Dim man As Single = 0
        Dim tar As Single = 0
        Dim f As Integer
        For f = 0 To 3
            man = man + turnoMan(f)
            tar = tar + turnoTar(f)
        Next
        Console.WriteLine("Total de gastos del turno de la mañana:" & man)
        Console.WriteLine("Total de gastos del turno de la tarde:" & tar)
        Console.ReadKey()
    End Sub
End Class


Sub Main()

    Dim pv3 As New PruebaVector3()

    pv3.Cargar()

    pv3.CalcularGastos()

End Sub

End Module
```

Definimos dos atributos de tipo vector donde almacenaremos los sueldos de los empleados de cada turno:

```
Private turnoMan(3) As Single
Private turnoTar(3) As Single
```

Mediante dos estructuras repetitivas procedemos a cargar cada vector:

```
Public Sub Cargar()
   Console.WriteLine("Sueldos de empleados del turno de la mañana.")
   Dim f As Integer
   For f = 0 To 3
      Console.Write("Ingrese sueldo:")
      turnoMan(f) = Console.ReadLine()
   Next
   Console.WriteLine("Sueldos de empleados del turno de la tarde.")
   For f = 0 To 3
      Console.Write("Ingrese sueldo:")
      turnoTar(f) = Console.ReadLine()
   Next
End Sub
```

En otro método procedemos a sumar las componentes de cada vector y mostrar dichos acumuladores:

```
Public Sub CalcularGastos()

   Dim man As Single = 0
   Dim tar As Single = 0
   Dim f As Integer
   For f = 0 To 3
      man = man + turnoMan(f)
      tar = tar + turnoTar(f)
   Next
   Console.WriteLine("Total de gastos del turno de la mañana:" & man)
```

```
        Console.WriteLine("Total de gastos del turno de la tarde:" & tar)
        Console.ReadKey()
    End Sub
```

En la Main creamos un objeto de la clase PruebaVector3 y llamamos a sus dos métodos:

```
Sub Main()
    Dim pv3 As New PruebaVector3()
    pv3.Cargar()
    pv3.CalcularGastos()
End Sub
```

Problemas propuestos

1. Desarrollar un programa que permita ingresar un vector de 8 elementos, e informe:
   El valor acumulado de todos los elementos del vector.
   El valor acumulado de los elementos del vector que sean mayores a 36.
   Cantidad de valores mayores a 50.

2. Realizar un programa que pida la carga de dos vectores numéricos enteros de 4 elementos. Obtener la suma de los dos vectores, dicho resultado guardarlo en un tercer vector del mismo tamaño. Sumar componente a componente.

3. Se tienen las notas del primer parcial de los alumnos de dos cursos, el curso A y el curso B, cada curso cuenta con 5 alumnos. Realizar un programa que muestre el curso que obtuvo el mayor promedio general.

4. Cargar un vector de 10 elementos y verificar posteriormente si el mismo está ordenado de menor a mayor.

Interfaces visuales - Windows Forms

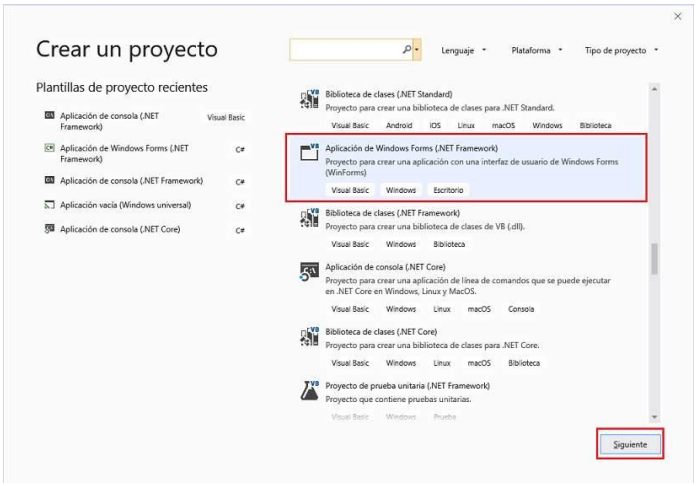
Hasta ahora hemos resuelto todos los algoritmos haciendo las salidas a través de una consola en modo texto. La realidad que es muy común la necesidad de hacer la entrada y salida de datos mediante una interfaz más amigable con el usuario.

En Visual Basic .Net existen varias librerías de clase para implementar interfaces visuales. Utilizaremos primero las Windows Forms.

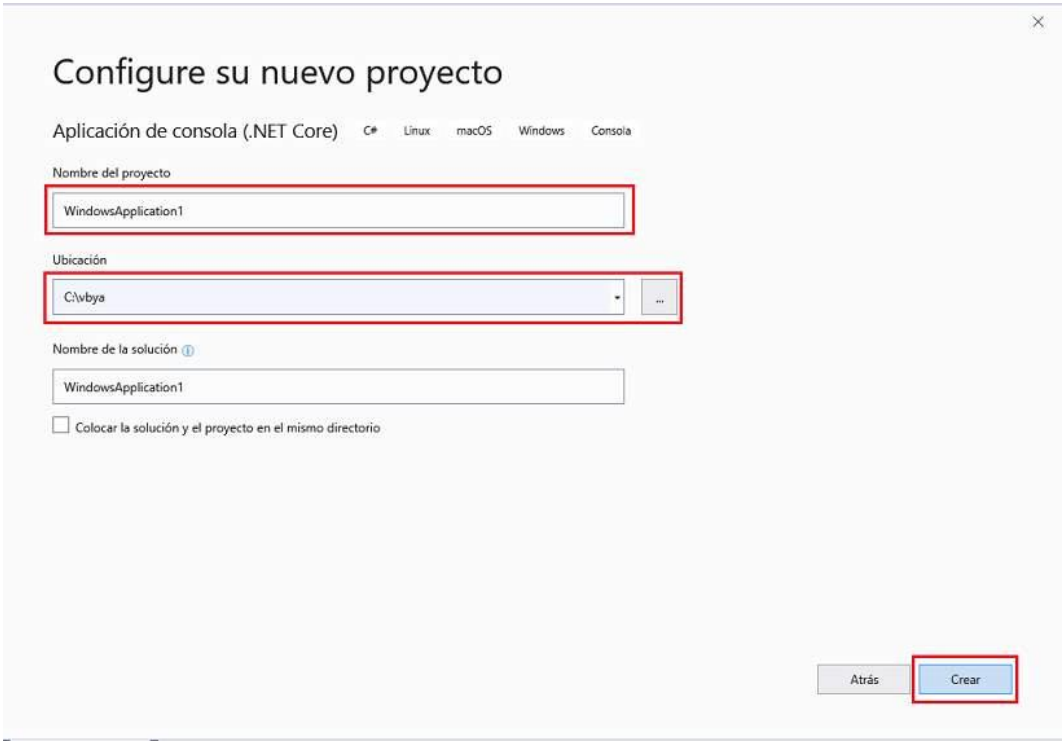
Para crear una aplicación que utilice esta librería debemos crear un proyecto. Los pasos son los siguientes:

Desde el menú de opciones del Visual Studio .Net seleccionamos la opción: Archivo -> Nuevo -> Proyecto...

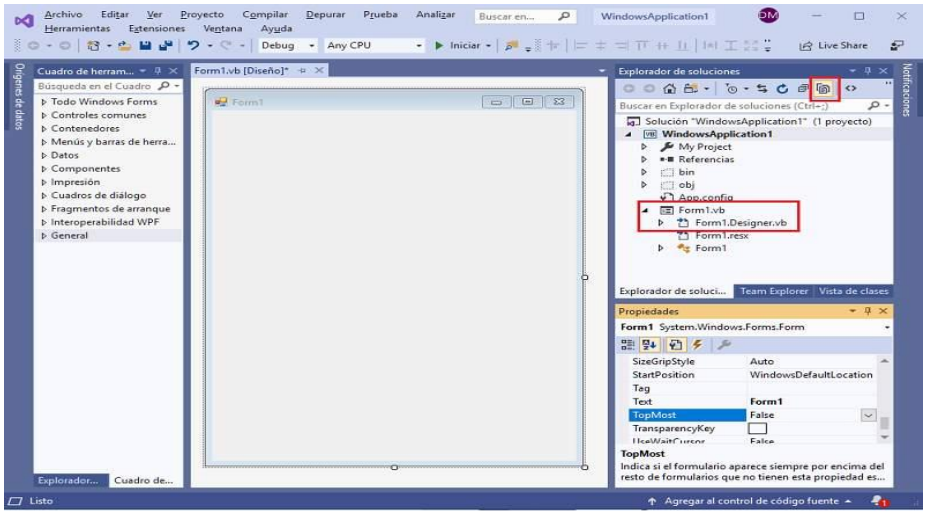
Seleccionamos la plantilla "Aplicación de Windows Forms (.NET Framework) Visual Basic Windows Escritorio".



En el diálogo siguiente definimos un nombre al proyecto y la carpeta donde se almacenará:

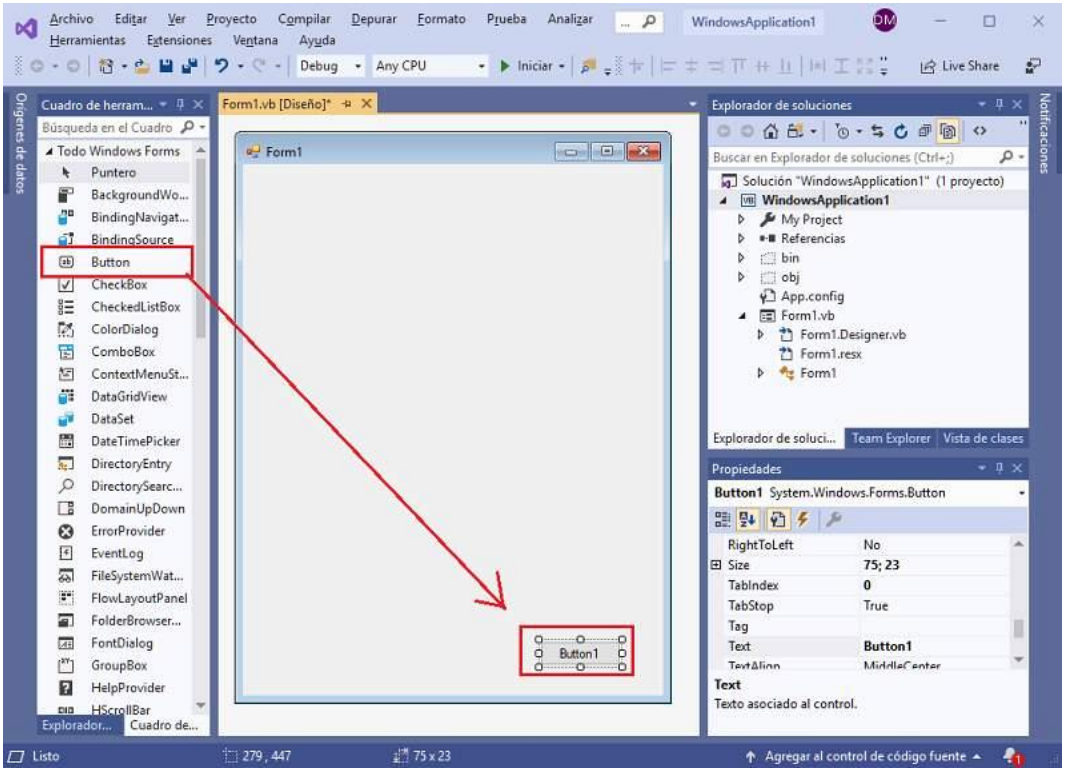


Ahora ya tenemos un esqueleto para desarrollar nuestra aplicación. Si vemos la ventana del "Explorador de soluciones y presionamos el ícono superior para "Mostrar todos los archivos" veremos: Form1.vb y Form1.Designer.vb:

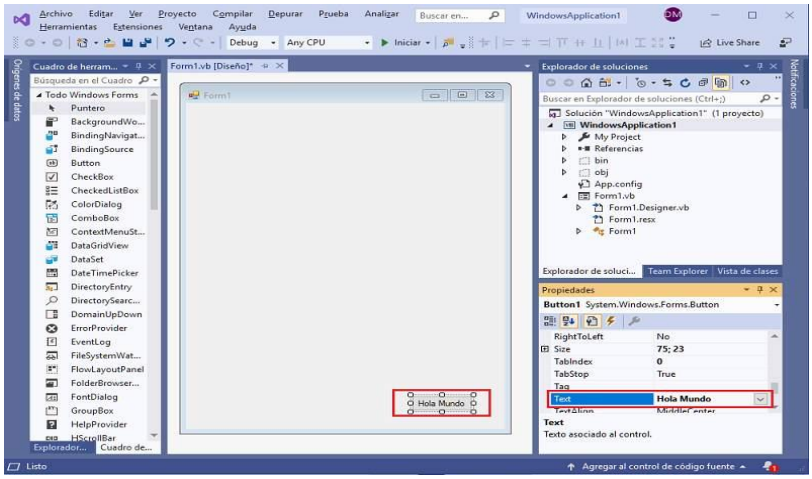


En la parte central tenemos el Form listo para disponer controles con el mouse.

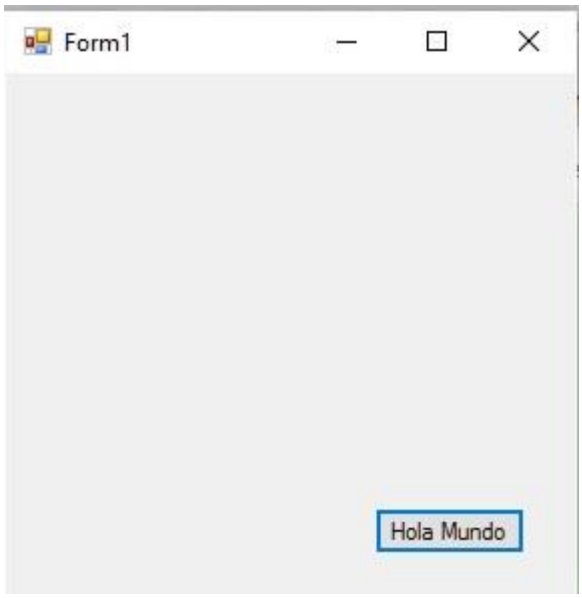
Ahora podemos seleccionar un control visual de la ventana "Cuadro de herramientas" que se encuentra a la izquierda (seleccionemos el control Button) y seguidamente presionemos el botón izquierdo del mouse dentro del formulario que se encuentra en la parte central del Visual Studio .net:



Pasemos a analizar la ventana "Propiedades" que nos muestra las propiedades del objeto seleccionado del formulario. Podemos por ejemplo si tenemos seleccionado el botón cambiar la propiedad Text (la misma cambia la etiqueta que muestra el botón):



Cuando ejecutamos la aplicación el resultado es muy distinto a la interfaz en modo texto vista hasta el momento:



Por último vamos a ver los contenidos de los archivos generados automáticamente por el Visual Studio .Net.

Form1.Designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _

Public Class Form1

    Inherits System.Windows.Forms.Form
```

```
'Form reemplaza a Dispose para limpiar la lista de componentes.
<System.Diagnostics.DebuggerNonUserCode()> _
Protected Overrides Sub Dispose(ByVal disposing As Boolean)
    Try
        If disposing AndAlso components IsNot Nothing Then
            components.Dispose()
        End If
    Finally
        MyBase.Dispose(disposing)
    End Try
End Sub

'Requerido por el Diseñador de Windows Forms
Private components As System.ComponentModel.IContainer
'NOTA: el Diseñador de Windows Forms necesita el siguiente procedimiento
'Se puede modificar usando el Diseñador de Windows Forms.
'No lo modifique con el editor de código.
<System.Diagnostics.DebuggerStepThrough()> _
Private Sub InitializeComponent()
    Me.Button1 = New System.Windows.Forms.Button()
```

```
    Me.SuspendLayout()
    '
    'Button1
    '
    Me.Button1.Location = New System.Drawing.Point(279, 447)
    Me.Button1.Name = "Button1"
    Me.Button1.Size = New System.Drawing.Size(75, 23)
    Me.Button1.TabIndex = 0
    Me.Button1.Text = "Hola Mundo"
    Me.Button1.UseVisualStyleBackColor = True
    '
    'Form1
    '
    Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
    Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
    Me.ClientSize = New System.Drawing.Size(394, 498)
    Me.Controls.Add(Me.Button1)
    Me.Name = "Form1"
    Me.Text = "Form1"
    Me.ResumeLayout(False)
End Sub
```

```
    Friend WithEvents Button1 As Button

End Class
```

Este archivo nunca lo modificaremos manualmente sino será el entorno del Visual Studio .Net que lo generará a medida que dispongamos controles visuales dentro de nuestro Form1.

Form1.vb

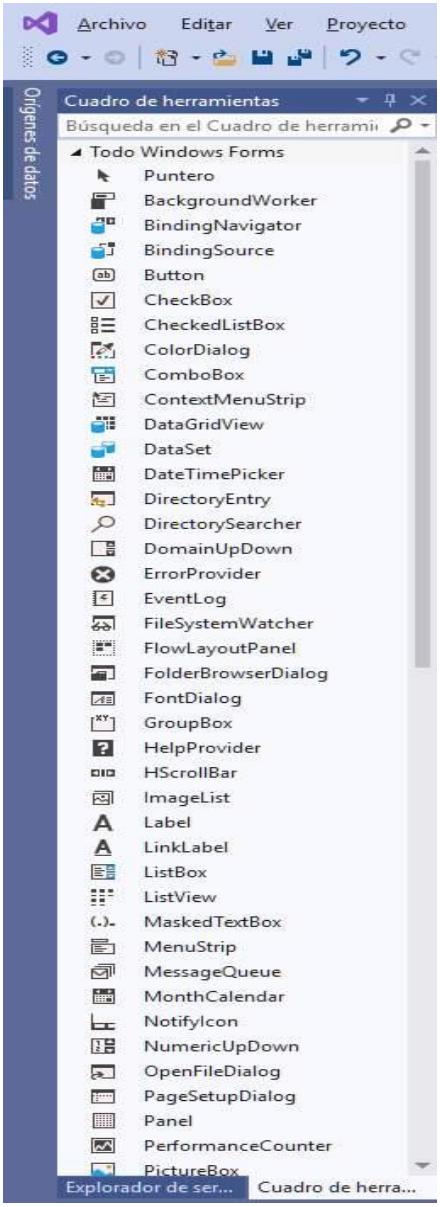
```
Public Class Form1

End Class
```

Este archivo es donde agregaremos la lógica para nuestro formulario. El archivo contiene parte de la clase Form1 (es parcial con el que se genera automáticamente).

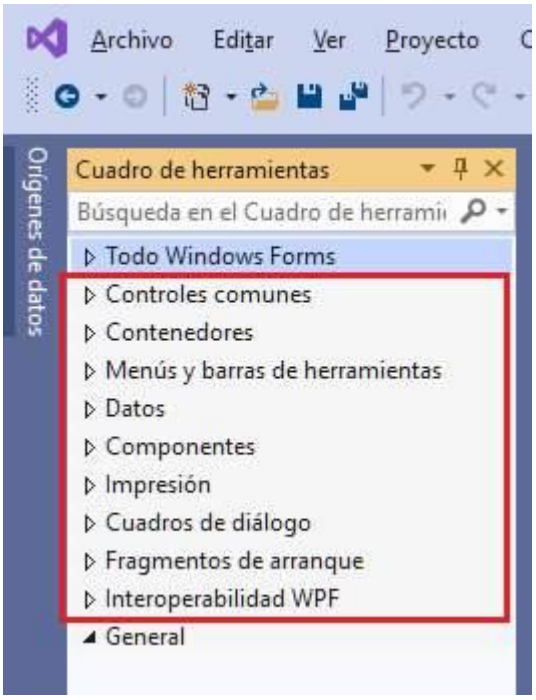
Cuadro de herramientas - Windows Forms

El cuadro de herramientas contiene todas las componentes visuales que nos permiten confeccionar nuestro formulario.

Podemos ver todos los controles visuales en forma completa:



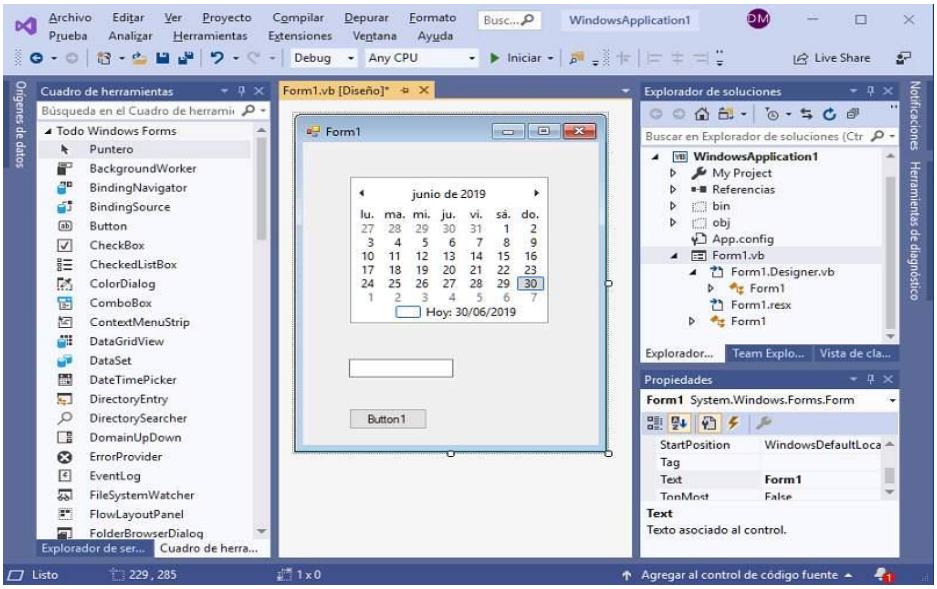
O agrupados por su uso (Menús, datos etc.):



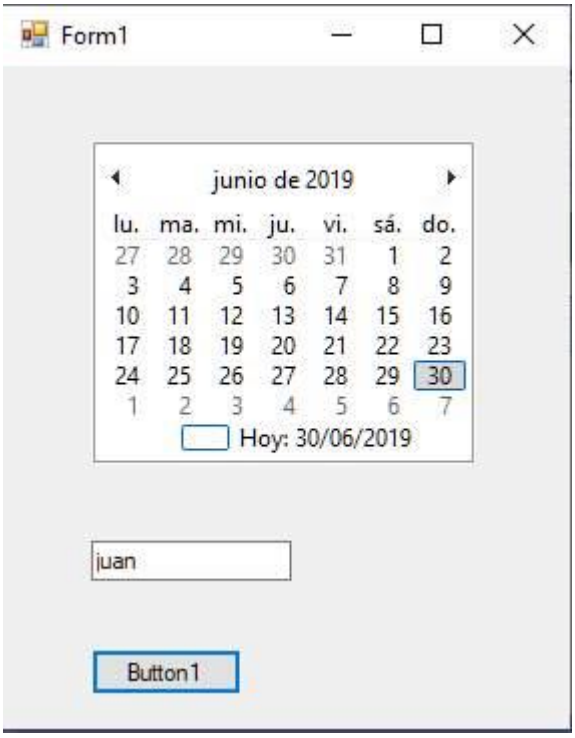
Problema 1:

: Desarrollar un programa que muestre un objeto de cada una de las siguientes clases: MonthCalendar, TextBox y Button

La interfaz visual debe ser parecida a esta:



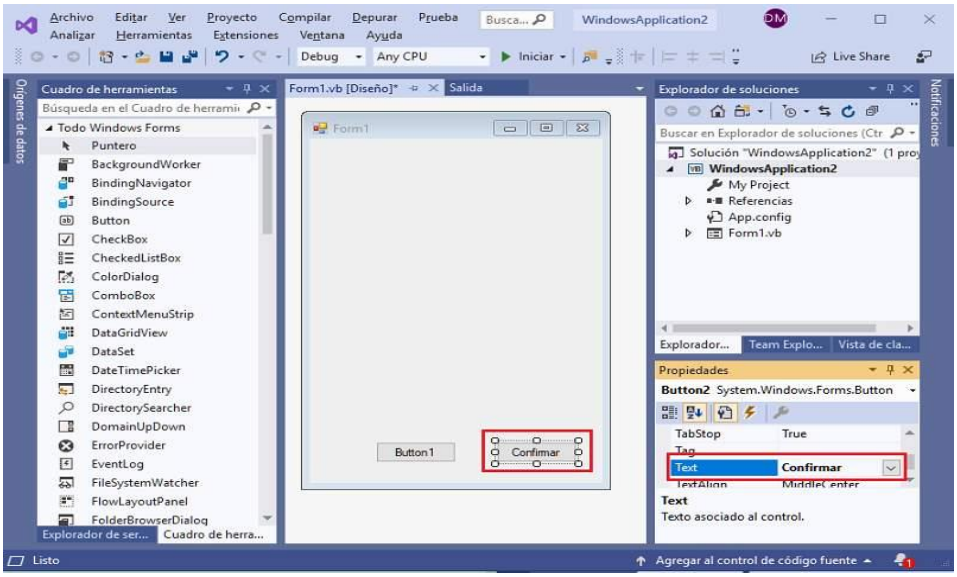
Hasta ahora solo hemos creado una interfaz visual, como podemos ver algunas componentes en tiempo de ejecución tienen funcionalidad (el objeto de la clase MonthCalendar si ejecutamos el programa nos permite seleccionar una fecha, cambiar de mes etc., el control de la clase TextBox nos permite ingresar una cadena de caracteres, pero el objeto de la clase Button cuando se presiona podemos ver que se visualiza que es hundido con el mouse pero no hace nada):



Ventana de propiedades - Windows Forms

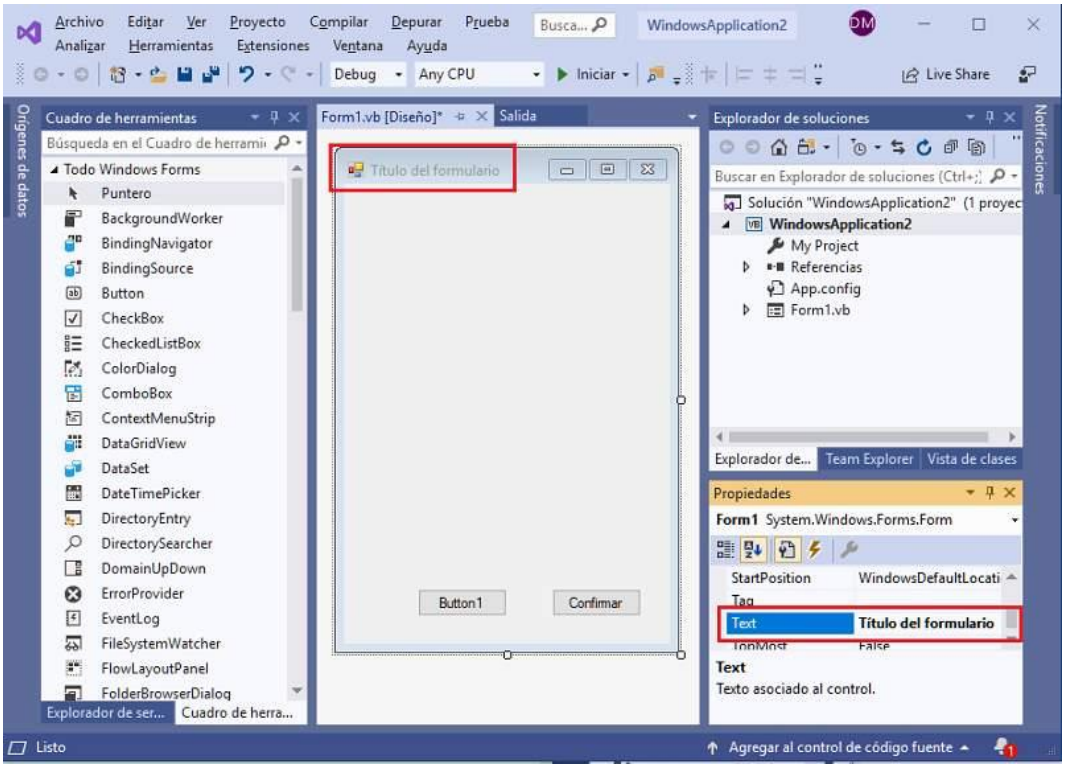
La "ventana de propiedades" nos permite inicializar los valores de las propiedades del objeto que se encuentra seleccionado en el formulario (Button, MonthCalendar, TextBox etc.)

Por ejemplo si disponemos dos objetos de la clase Button y seleccionamos uno de ellos podremos editar las propiedades del mismo en la "ventana de propiedades":



A medida que seleccionamos un objeto en la ventana de "Diseño" podemos ver como se actualiza la "ventana de propiedades", por ejemplo la propiedad Text de la clase Button permite fijar la etiqueta que muestra el botón.

El formulario también es un objeto, esto quiere decir que si lo seleccionamos luego la "ventana de propiedades" nos muestra las propiedades de la clase Form:

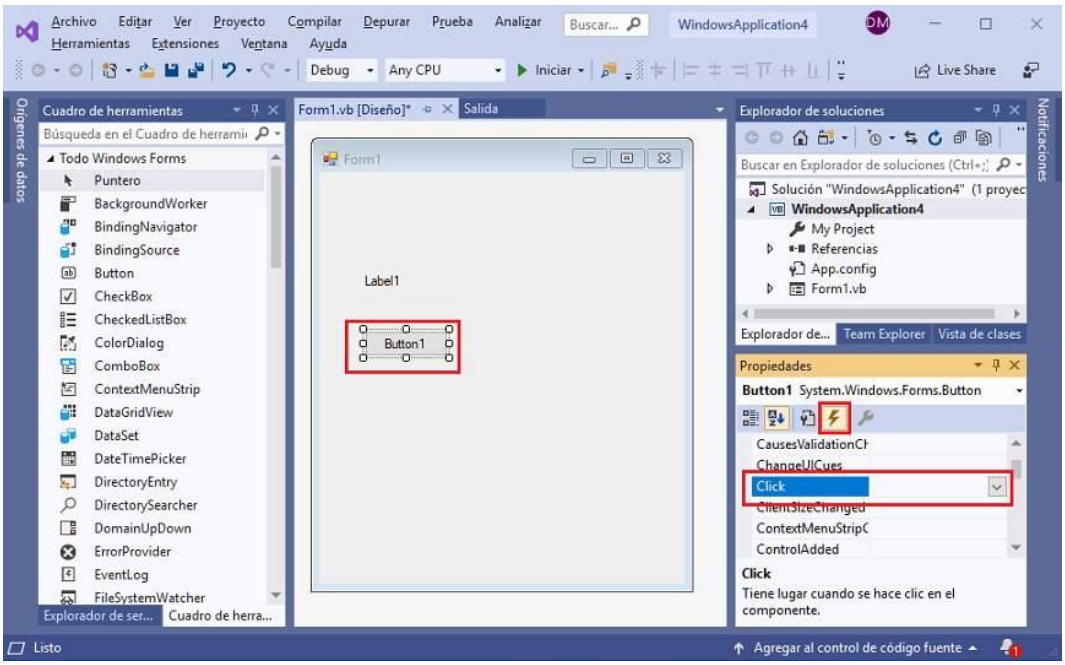


Problema propuesto

Elaborar una interfaz gráfica que muestre una calculadora (utilizar objetos de la clase Button y un objeto de la clase TextBox donde se mostrarían los resultados y se cargarían los datos), tener en cuenta que solo se debe implementar la interfaz y no la funcionalidad de una calculadora.

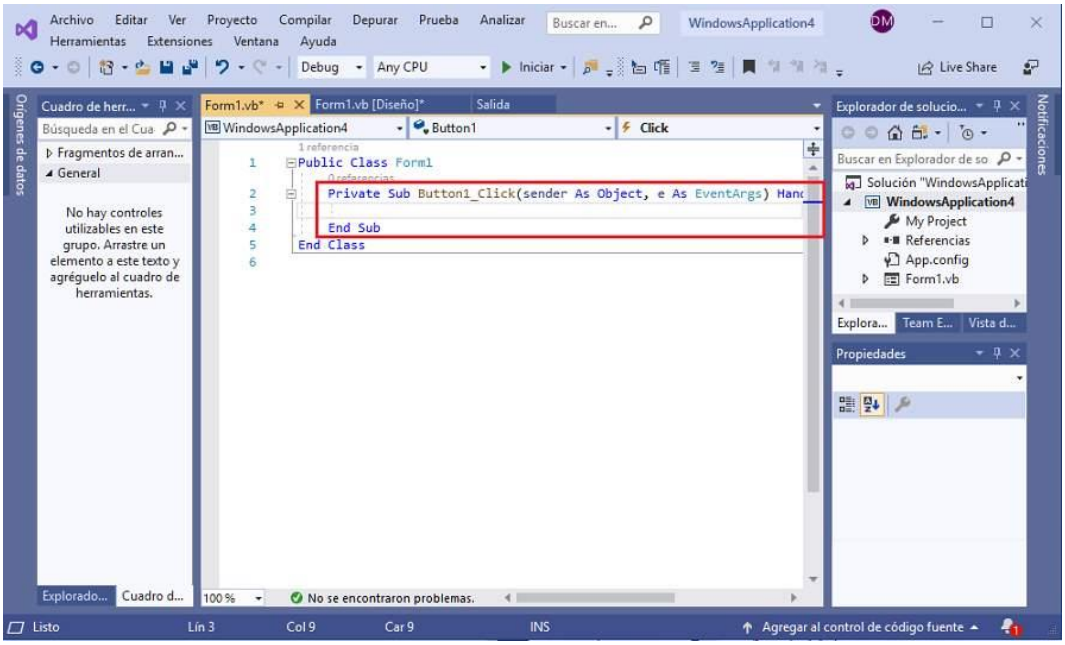
Ventana de eventos - Windows Forms

La ventana de eventos coincide con la ventana de propiedades. Para activar la lista de eventos disponibles para un objeto debemos presionar el ícono:



Podemos observar la lista de eventos que puede reaccionar el objeto seleccionado en ese momento. Por ejemplo si tenemos seleccionado un objeto de la clase Button el evento más común que deberemos implementar es el Click (este evento se dispara cuando en tiempo de ejecución del programa se presiona el botón)

Para disponer el código para dicho evento debemos hacer doble clic sobre dicho evento (esto hace que se active la ventana del editor y genere automáticamente el método asociado a dicho evento):



Problema:

Confeccionar un programa que al presionar un botón se muestre en un objeto de la clase Label el string "Hola Mundo".

Programa:

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Label1.Text = "Hola Mundo"
    End Sub
End Class
```

Hay que tener en cuenta que la clase anterior es parcial (el archivo Form1.Designer.vb contiene la definición de los dos objetos y la inicialización de sus propiedades y evento, recordemos que este archivo no hay que modificar):

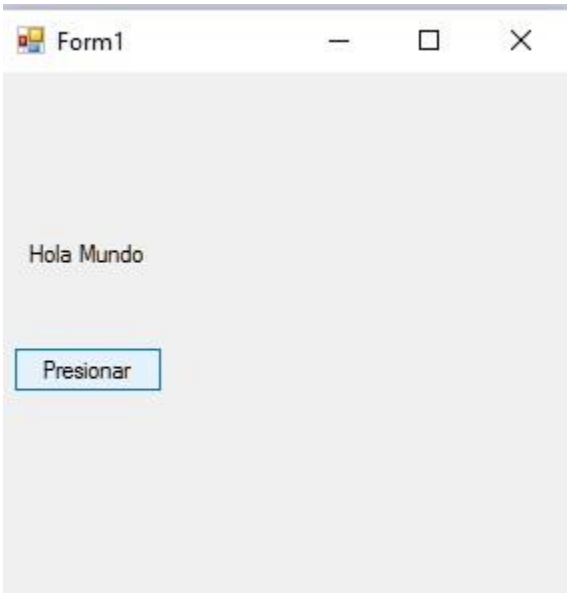
```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _

Partial Class Form1
```

```
Inherits System.Windows.Forms.Form


'Form reemplaza a Dispose para limpiar la lista de componentes.
<System.Diagnostics.DebuggerNonUserCode()> _
Protected Overrides Sub Dispose(ByVal disposing As Boolean)
    Try
        If disposing AndAlso components IsNot Nothing Then
            components.Dispose()
        End If
    Finally
        MyBase.Dispose(disposing)
    End Try
End Sub


'Requerido por el Diseñador de Windows Forms

Private components As System.ComponentModel.IContainer


'NOTA: el Diseñador de Windows Forms necesita el siguiente procedimiento

'Se puede modificar usando el Diseñador de Windows Forms.

'No lo modifique con el editor de código.

<System.Diagnostics.DebuggerStepThrough()> _

Private Sub InitializeComponent()

    Me.Label1 = New System.Windows.Forms.Label()

    Me.Button1 = New System.Windows.Forms.Button()

    Me.SuspendLayout()
```

```
'
'Label1
'
Me.Label1.AutoSize = True
Me.Label1.Location = New System.Drawing.Point(38, 92)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(39, 13)
Me.Label1.TabIndex = 0
Me.Label1.Text = "Label1"
'
'Button1
'
Me.Button1.Location = New System.Drawing.Point(41, 144)
Me.Button1.Name = "Button1"
Me.Button1.Size = New System.Drawing.Size(75, 23)
Me.Button1.TabIndex = 1
Me.Button1.Text = "Button1"
Me.Button1.UseVisualStyleBackColor = True
'
'Form1
'
```

```
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)

        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font

        Me.ClientSize = New System.Drawing.Size(331, 373)

        Me.Controls.Add(Me.Button1)

        Me.Controls.Add(Me.Label1)

        Me.Name = "Form1"

        Me.Text = "Form1"

        Me.ResumeLayout(False)

        Me.PerformLayout()


    End Sub


    Friend WithEvents Label1 As Label

    Friend WithEvents Button1 As Button

End Class
```

Al ejecutar el programa si presionamos el botón vemos como cambia el contenido de la Label (esto debido a que en el evento Click del botón cambiamos el valor de la propiedad Text del objeto de la clase Label):



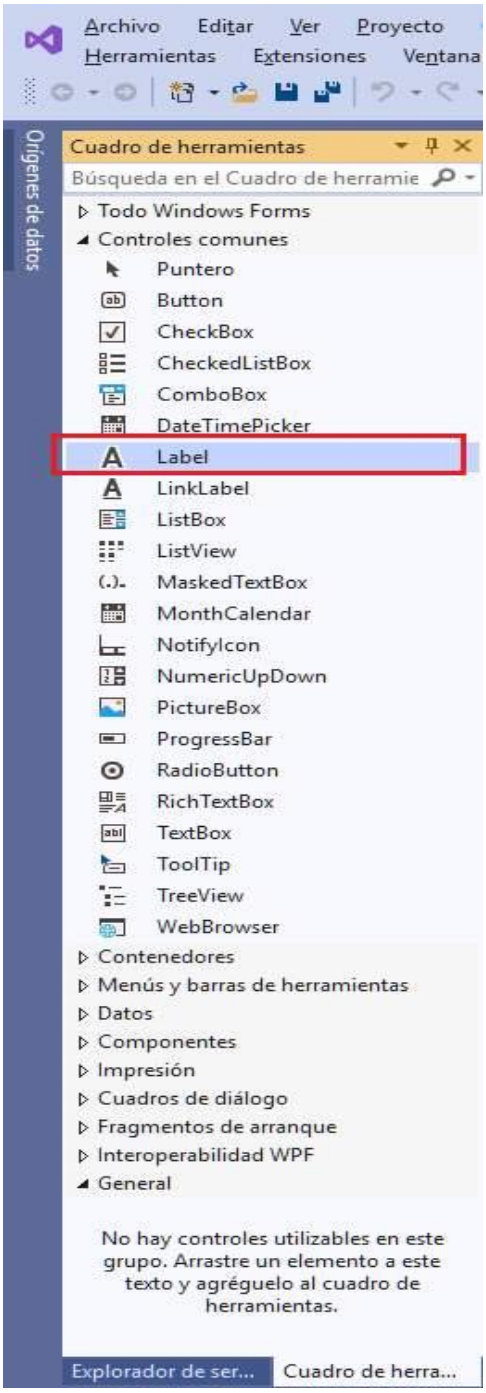
Problema propuesto

Disponer 7 objetos de la clase Button con los días de la semana. Fijar en los atributos Text de cada botón los días de la semana. Al presionar un botón mostrar en un objeto de la clase Label el día seleccionado.

Controles comunes - Label

En el "cuadro de herramientas" podemos ver las componentes visuales agrupadas. En la pestaña de Controles comunes podemos acceder a los controles visuales que normalmente toda aplicación requiere.

El primer control que vamos a analizar es la clase Label. Un control de tipo Label nos permite mostrar básicamente un texto inicializando la propiedad Text.



Las propiedades más comunes de este control son:

Text: Es el string que muestra el control.

BackColor: Define el color de fondo del control.

ForeColor: Define el color del texto.

Font: Define la fuente del texto.

BorderStyle: Define si la label tiene un borde visible.

AutoSize: Permite o no redimensionar la label en forma automática.

Cursor: Definimos el ícono del cursor a mostrar cuando disponemos el mouse dentro del control.

Visible: Determina si el control está visible u oculto cuando ejecutamos el programa.

Problema propuesto

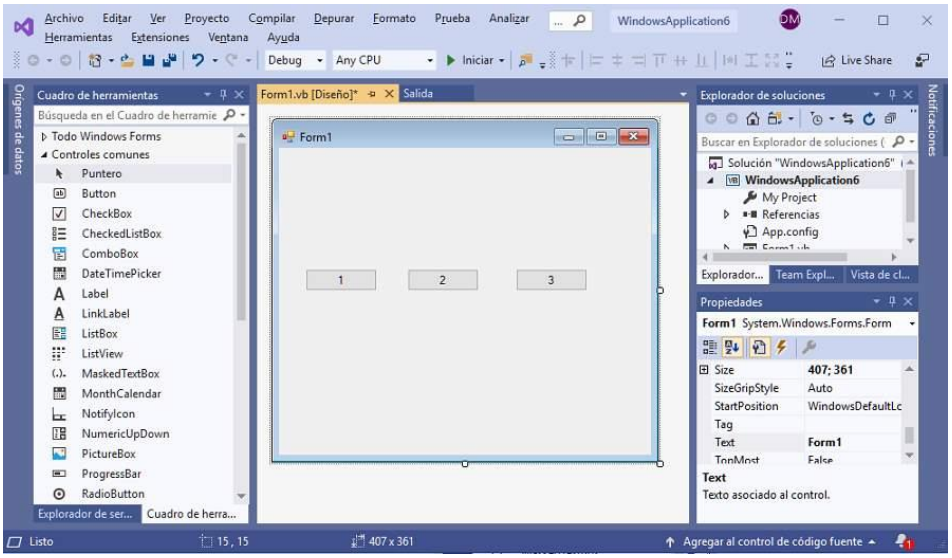
1. Crear una aplicación que muestre en 6 objetos de la clase Label con algunos nombres de controles visuales contenidos en la pestaña de "controles comunes" del cuadro de herramientas

Controles comunes - Button

Un control común a disponer dentro de un Form son los botones, esto se hace disponiendo objetos de la clase Button.

Problema 1:

Confeccionar un formulario que muestre tres objetos de la clase Button, disponer como etiqueta en cada botón los valores 1, 2 y 3. Cuando se presiona el botón mostrar en el título del formulario el valor de la etiqueta del botón presionado.



Programa:

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Text = Button1.Text
    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
        Text = Button2.Text
    End Sub

    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
        Text = Button3.Text
    End Sub
End Class
```

Para el evento Click de cada botón inicializamos la propiedad Text del Form1 con la propiedad Text del botón presionado (como la clase Form1 hereda de la clase Form luego accedemos a la propiedad Text sin anteceder nombre alguno: Text = Button1.Text):

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click

    Text = Button1.Text

End Sub
```

Problema 2:

Modificar el problema anterior para que se acumulen en el título del formulario los valores de los botones presionados.

Programa:

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Text = Text & Button1.Text
    End Sub


    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click

        Text = Text & Button2.Text

    End Sub


    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click

        Text = Text & Button3.Text

    End Sub

End Class
```

Concatenamos el valor actual de la propiedad Text del formulario con el valor de la propiedad Text del botón respectivo con el operador &:

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Text = Text & Button1.Text
End Sub
```

Problema 3:

Similar al problema anterior solo permitir mostrar hasta 10 caracteres en el título del formulario.

Programa:

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        If Text.Length < 10 Then
            Text = Text & Button1.Text
        End If
    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click

        If Text.Length < 10 Then

            Text = Text & Button2.Text

        End If

    End Sub


    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
```

```
        If Text.Length < 10 Then
            Text = Text & Button3.Text
        End If
    End Sub
End Class
```

Como la propiedad Text es de tipo String luego podemos acceder a la propiedad Length para conocer la cantidad de caracteres almacenados:

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        If Text.Length < 10 Then
            Text = Text & Button1.Text
        End If
    End Sub
```

Problema propuesto.

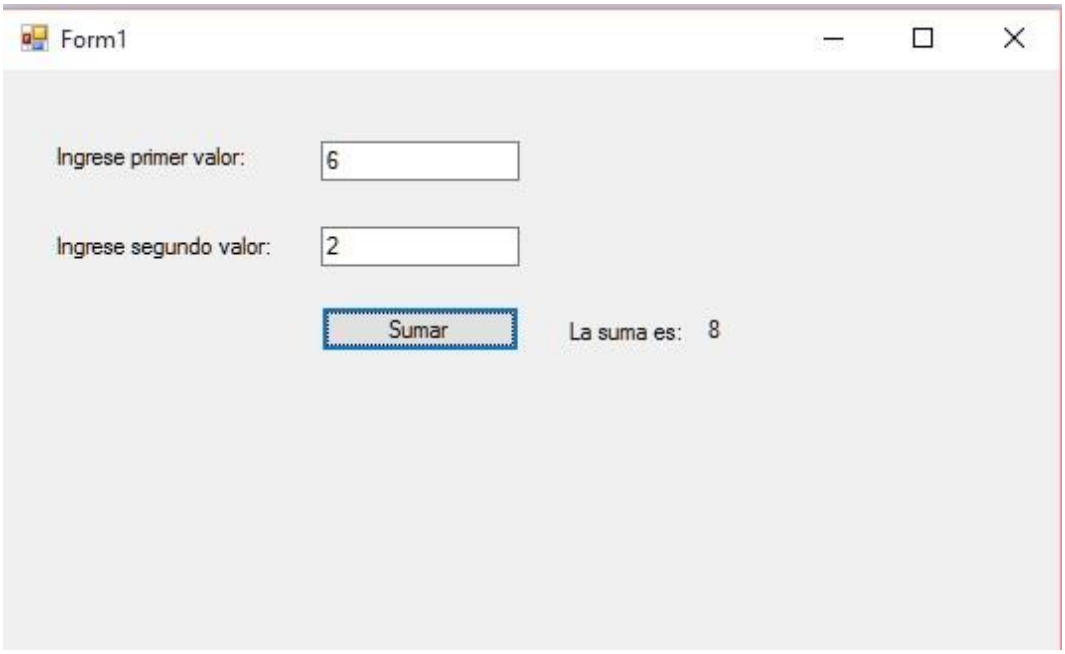
Elaborar una interfaz gráfica que muestre una calculadora (utilizar objetos de la clase Button y un objeto de la clase Label donde se muestra el valor ingresado), tener en cuenta que solo se debe implementar la interfaz y la carga de un valor de hasta 12 dígitos.

Controles comunes - TextBox

El control más común para la entrada de datos por teclado es el TextBox.

Problema 1:

Confeccionar un programa que permita ingresar dos valores enteros por teclado y al presionar un botón mostrar en una Label la suma de dichos valores.



Programa:

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click

        Dim valor1 As Integer
        Dim valor2 As Integer
        Dim suma As Integer
        valor1 = TextBox1.Text
        valor2 = TextBox2.Text
        suma = valor1 + valor2
        Label4.Text = suma
    End Sub
End Class
```

Para saber el valor almacenado en un TextBox disponemos de la propiedad Text. Como la propiedad Text es de tipo String al asignarla a una variable Integer automáticamente se transforma en Integer el String:

```
valor1 = TextBox1.Text
```

Sumamos los dos enteros:

```
suma = valor1 + valor2
```

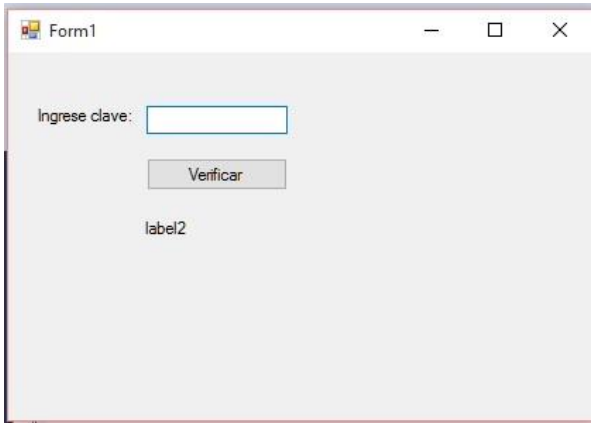
Y finalmente cargamos en un objeto de la clase Label el resultado de la suma. Como la variable suma es un entero al asignársela a la propiedad Text de la Label automáticamente se transforma en String:

```
Label4.Text = suma
```

Problema 2:

Solicitar que se ingrese una clave. Si se ingresa la cadena "abc123" mostrar un mensaje de clave correcta en caso contrario mostrar clave incorrecta. Utilizar un control de tipo TextBox para el ingreso de la clave y una Label para mostrar el resultado al presionar un botón.

Inicializar la propiedad UseSystemPasswordChar con el valor true (esto hace que cuando el operador tipee caracteres dentro del TextBox se visualicen como asteriscos)



Programa:

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
```

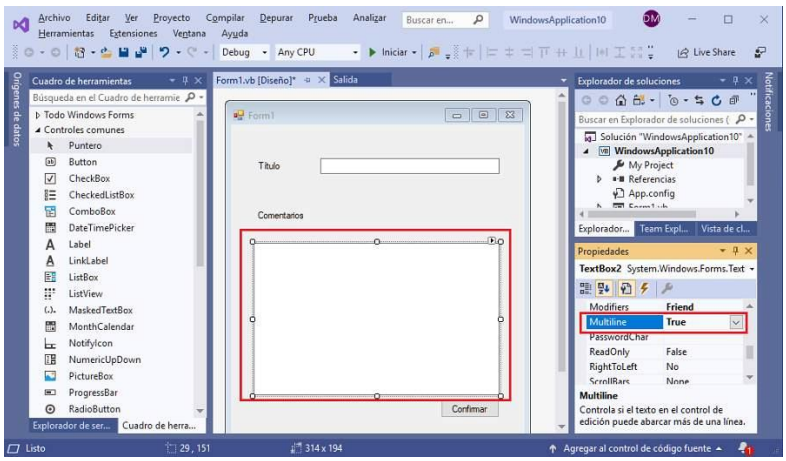
```
    If TextBox1.Text = "abc123" Then
        Label2.Text = "Clave correcta"
    Else
        Label2.Text = "Clave incorrecta"
    End If
  End Sub
End Class
```

Para verificar si la clave es correcta comparamos la cadena cargada en el TextBox1 con la cadena "abc123".

Hay otra propiedad en la clase TextBox llamada PasswordChar, si la propiedad UseSystemPasswordChar está configurada con false podemos inicializar la propiedad PasswordChar con el caracter que queremos que se muestre al ingresar datos en el TextBox. Probar de inicializarlo con el caracter '+' y veremos que en vez de aparecer asteriscos aparecen caracteres '+'

Problema 3:

Disponer un control de tipo TextBox e inicializar la propiedad Multiline con el valor True (esto permite ingresar múltiples líneas dentro de un TextBox)



Programa:

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click

        MessageBox.Show(TextBox2.Text)

    End Sub

End Class
```

Cuando se presiona un botón se muestra en un cuadro de mensajes (MessageBox) el texto ingresado en el TextBox2:

```
MessageBox.Show(TextBox2.Text)
```

Problema propuesto

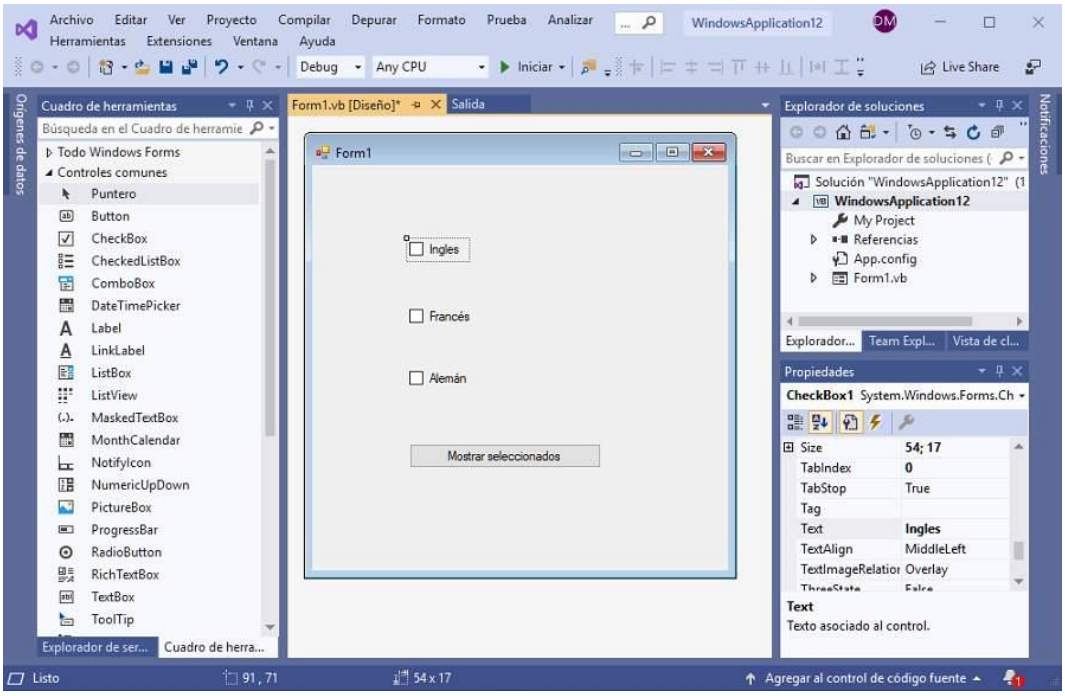
2. Solicitar el ingreso de una clave de hasta 10 caracteres en un control de tipo TextBox (inicializar la propiedad MaxLength con el valor 10) Mostrar en un cuadro de mensajes la clave ingresada al presionar un botón.

Controles comunes - CheckBox

El control CheckBox permite implementar un cuadro de selección (básicamente un botón de dos estados: seleccionado o no seleccionado)

Problema 1:

Confeccionar un programa que muestre 3 objetos de la clase CheckBox con etiquetas de tres idiomas. Cuando se presiona un botón mostrar en la barra de títulos del Form todos los CheckBox seleccionados hasta el momento.



Programa:

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click

        Text = ""

        If CheckBox1.Checked = True Then

            Text = Text + "(Inglés)"

        End If

        If CheckBox2.Checked = True Then

            Text = Text + "(Francés)"

        End If

        If CheckBox3.Checked = True Then
```

```
            Text = Text + "(Alemán)"

        End If

    End Sub

End Class
```

La clase CheckBox tiene una propiedad llamada Checked (si tiene el valor true significa que el CheckBox esta seleccionado, en caso contrario no esta seleccionado)

En el evento Click del botón primero borramos el contenido del título del Form1:

```
        Text = ""
```

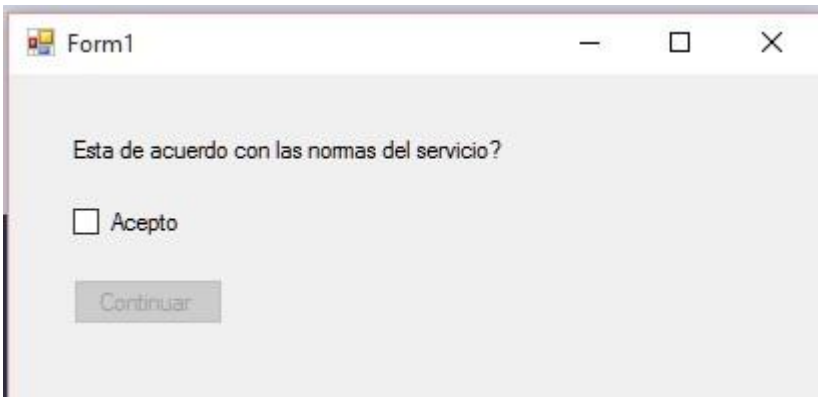
Y seguidamente mediante estructuras If verificamos el estado de cada CheckBox, en caso de estar seleccionado concatenamos al título del Form1 el valor que representa ese CheckBox:

```
        If CheckBox1.Checked = True Then

            Text = Text + "(Inglés)"

        End If

        If CheckBox2.Checked = True Then

            Text = Text + "(Francés)"

        End If

        If CheckBox3.Checked = True Then

            Text = Text + "(Alemán)"

        End If
```

Problema 2:

Disponer un control Label que muestre el siguiente mensaje: "Esta de acuerdo con las normas del servicio?", luego un CheckBox y finalmente un objeto de tipo Button

desactivo (propiedad Enabled con False). Cuando se tilde el CheckBox debemos activar el botón (para esto debemos responder al evento CheckedChanged)



Programa:

```
Public Class Form1

    Private Sub CheckBox1_CheckedChanged(sender As Object, e As EventArgs) Handles CheckBox1.CheckedChanged

        If CheckBox1.Checked = True Then
            Button1.Enabled = True
        Else
            Button1.Enabled = False
        End If
    End Sub
End Class
```

Debemos implementar el evento CheckedChanged del objeto CheckBox1 (preguntamos si el CheckBox se encuentra seleccionado o no, en caso de estar seleccionado activamos el botón asignando a la propiedad Enabled el valor True):

```
Private Sub CheckBox1_CheckedChanged(sender As Object, e As EventArgs) Handles CheckBox1.CheckedChanged

    If CheckBox1.Checked = True Then

        Button1.Enabled = True

    Else
        Button1.Enabled = False
    End If
End Sub
```

Problema propuesto

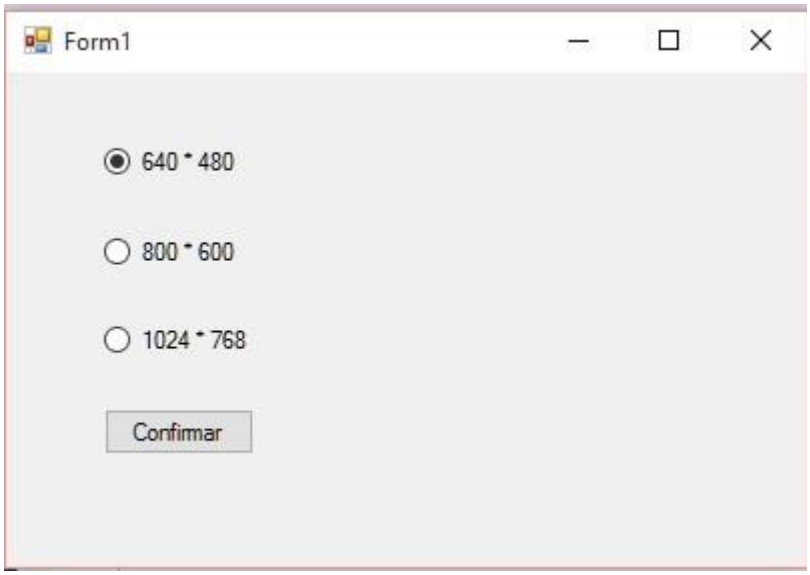
Disponer tres objetos de la clase CheckBox con nombres de navegadores web. Cuando se presione un botón mostrar en el título del Form los programas seleccionados.

Controles comunes - RadioButton y GroupBox

Otro control visual muy común es el RadioButton que normalmente se muestran un conjunto de RadioButton y permiten la selección de solo uno de ellos.

Problema 1:

Confeccionar un programa que muestre 3 objetos de la clase RadioButton que permitan configurar el ancho y alto del Form. Cuando se presione un botón actualizar el ancho y alto.



Programa:

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        If RadioButton1.Checked = True Then
            Width = 640
            Height = 480
        Else
            If RadioButton2.Checked = True Then
                Width = 800
                Height = 600
            Else
                If RadioButton3.Checked = True Then
                    Width = 1024
                    Height = 768
                End If
```

```
        End If
    End If
End Sub
End Class
```

Todos los controles que se disponen dentro de un Form están asociados, es decir que cuando seleccionamos uno se desmarca la actual.

El control RadioButton tiene una propiedad llamada Checked que almacena true o false, por eso que por medio de un conjunto de if verificamos cuál de los radio esta seleccionado:

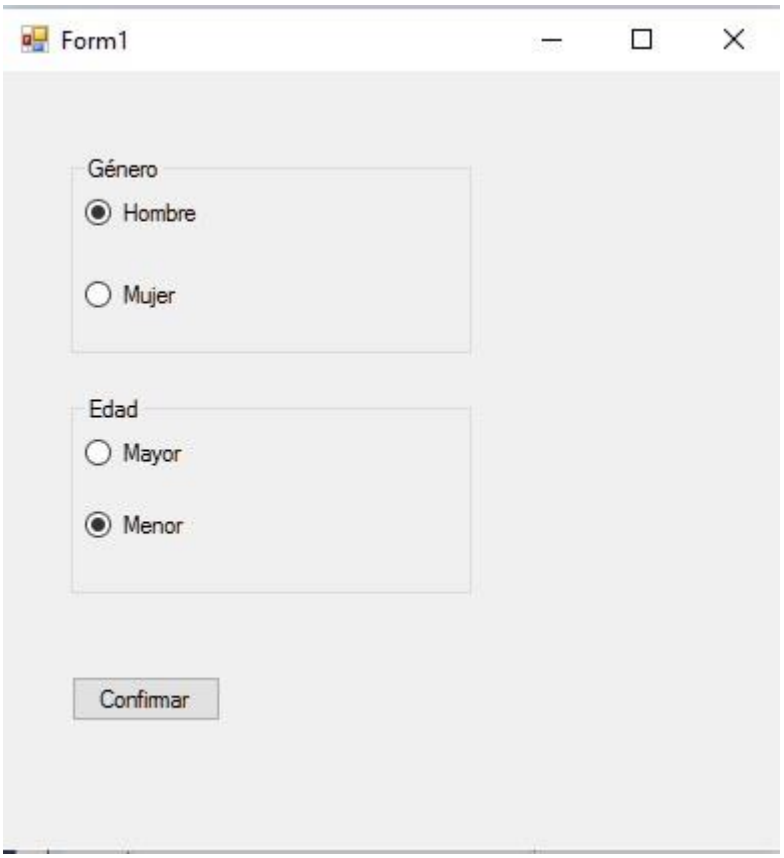
```
If RadioButton1.Checked = True Then
    Width = 640
    Height = 480
Else
    If RadioButton2.Checked = True Then
        Width = 800
        Height = 600
    Else
        If RadioButton3.Checked = True Then
            Width = 1024
            Height = 768
        End If
    End If
```

```
End If
```

Para cambiar el ancho y alto del Form accedemos a las propiedades Width y Height.

Problema 2:

Desarrollar una interfaz visual que muestre dos grupos independientes de RadioButton, en uno permitir seleccionar si es hombre o mujer y en el otro si es mayor o menor de edad. Mostrar un mensaje de la selección al presionar un botón



Debemos disponer dos objetos de la clase GroupBox y dentro de cada uno de ellos dos objetos de tipo RadioButton, con esto logramos que cada grupo sea independiente.

Programa:

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click

        Dim seleccionado As String = ""

        If RadioButton1.Checked = True Then

            seleccionado = seleccionado + "(hombre)"

        Else

            If RadioButton2.Checked = True Then

                seleccionado = seleccionado + "(mujer)"

            End If

        End If

        If RadioButton3.Checked = True Then

            seleccionado = seleccionado + "(mayor)"

        Else

            If RadioButton4.Checked = True Then

                seleccionado = seleccionado + "(menor)"

            End If

        End If

        MessageBox.Show(seleccionado)

    End Sub

End Class
```

Problema propuesto

1- Permitir el ingreso de dos números en controles de tipo TextBox y mediante dos controles de tipo RadioButton permitir seleccionar si queremos sumarlos o restarlos. Al presionar un botón mostrar en el título del Form el resultado de la operación.

Controles comunes - ComboBox

El control ComboBox permite seleccionar un String de una lista. Para inicializar los string que contendrá el ComboBox debemos acceder a la propiedad Items

Un evento muy útil con este control es cuando el operador selecciona un Item de la lista. Para capturar la selección de un item debemos codificar el evento SelectedIndexChanged.

Problema 1:

Cargar en un ComboBox los nombres de varios colores. Al seleccionar alguno mostrar en la barra de título del Form el String seleccionado.

Programa:

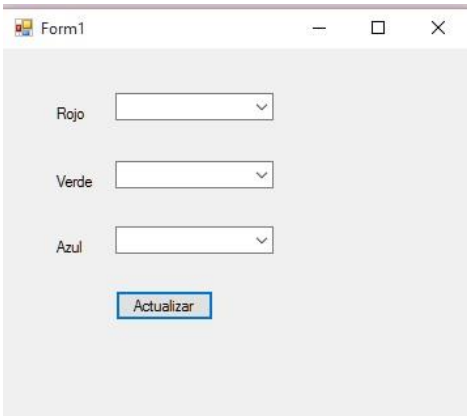
```
Public Class Form1

    Private Sub ComboBox1_SelectedIndexChanged(sender As Object, e As EventArgs) Handles ComboBox1.SelectedIndexChanged

        Text = ComboBox1.Text

    End Sub

End Class
```

Cuando se selecciona un string de la lista se dispara el evento SelectedIndexChanged y procedemos a extraer el texto seleccionado del ComboBox y lo mostramos en el título del Form:

```
Text = ComboBox1.Text
```

Problema 2:

Disponer tres controles de tipo ComboBox con valores entre 0 y 255 (cada uno representa la cantidad de rojo, verde y azul). Luego al presionar un botón pintar el fondo del Form con el color que se genera combinando los valores de los ComboBox.



Hay un evento muy importante en la clase Form llamado Load (seleccionemos el formulario y generemos dicho evento) Este evento se dispara cuando se carga el formulario con todos los controles visuales. En este evento cargaremos los tres ComboBox con los números.

Programa:

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim f As Integer
        For f = 0 To 255
            ComboBox1.Items.Add(f)
            ComboBox2.Items.Add(f)
            ComboBox3.Items.Add(f)
```

```
        Next

        ComboBox1.SelectedIndex = 0
        ComboBox2.SelectedIndex = 0
        ComboBox3.SelectedIndex = 0
    End Sub


    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim rojo As Integer = ComboBox1.Text
        Dim verde As Integer = ComboBox2.Text
        Dim azul As Integer = ComboBox3.Text
        BackColor = Color.FromArgb(rojo, verde, azul)
    End Sub
End Class
```

La carga manual de cada ComboBox nos haría perder mucho tiempo en tiempo de diseño por lo que lo hacemos mediante un algoritmo. Cuando se carga el Form se ejecuta el evento Load donde mediante un For procedemos a añadir los 256 valores:

```
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        Dim f As Integer
        For f = 0 To 255
            ComboBox1.Items.Add(f)
            ComboBox2.Items.Add(f)
            ComboBox3.Items.Add(f)
        Next
```

La propiedad Items del ComboBox tiene un método llamado Add que añade un elemento a la lista.

Luego para dejar seleccionado por defecto el primer item añadido iniciamos la propiedad SelectedIndex:

```
ComboBox1.SelectedIndex = 0
ComboBox2.SelectedIndex = 0
ComboBox3.SelectedIndex = 0
```

En el evento Click del botón procedemos a extraer el valor seleccionado de cada ComboBox:

```
Dim rojo As Integer = ComboBox1.Text

Dim verde As Integer = ComboBox2.Text

Dim azul As Integer = ComboBox3.Text
```

Para cambiar el color de fondo del Form actualizamos la propiedad BackColor. El color lo generamos llamando al método estático FromArgb de la clase Color:

```
BackColor = Color.FromArgb(rojo, verde, azul)
```

Problema propuesto

- Solicitar el ingreso del nombre de una persona y seleccionar de un control ComboBox un país. Al presionar un botón mostrar en la barra del título del Form el nombre ingresado y el país seleccionado.

UNIDAD VI

ESTRUCTURAS DE ARRANQUE

FORMULARIOS MDI

MDI: Definición: MDI significa en inglés **Múltiple Document Interfaz** y en español Interfaz de múltiples documentos y sirve para desplegar varios formularios a partir de un **Formulario principal.**

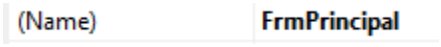
La contraparte del **MDI,** es el **SDI** que es un **Single Document Interfaz,** es decir, cuando el proyecto es un solo formulario o única interfaz.

En un **MDI** hay un **formulario principal** ó **formulario Padre** que contendrá a los **formularios secundarios o formularios hijos**.

En este libro se enseñan dos métodos usando dos ejemplos para crear **MDI** y ambos tienen igual resultado.

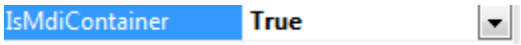
Ejercicio MDI

- Abra Visual Basic para crear un nuevo proyecto y asígnele el nombre **Formulario MDI**
- Al formulario que se despliega cámbiele la propiedad **Name** a **FrmPrincipal**

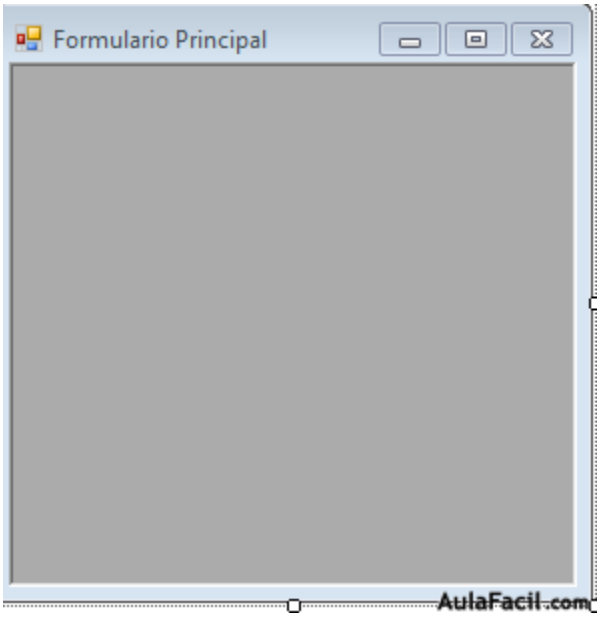


Y la propiedad **IsMdiContainer** cámbiela a True

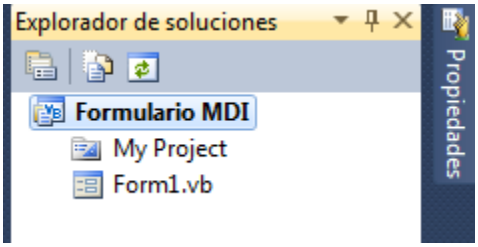
Con la acción anterior, le da la categoría de Padre al formulario principal.



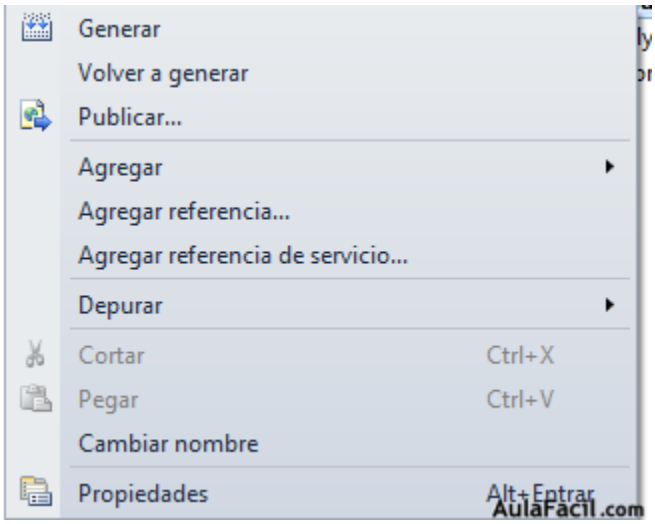
Al hacer esto, el formulario se vuelve de un color más oscuro que el tradicional.



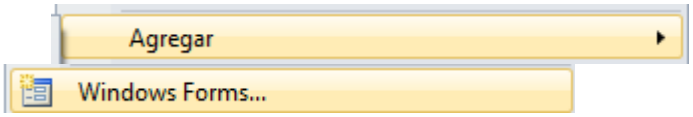
- Agregue los formularios hijos dando clic derecho en el proyecto Formulario **MDI** en el Explorador de soluciones



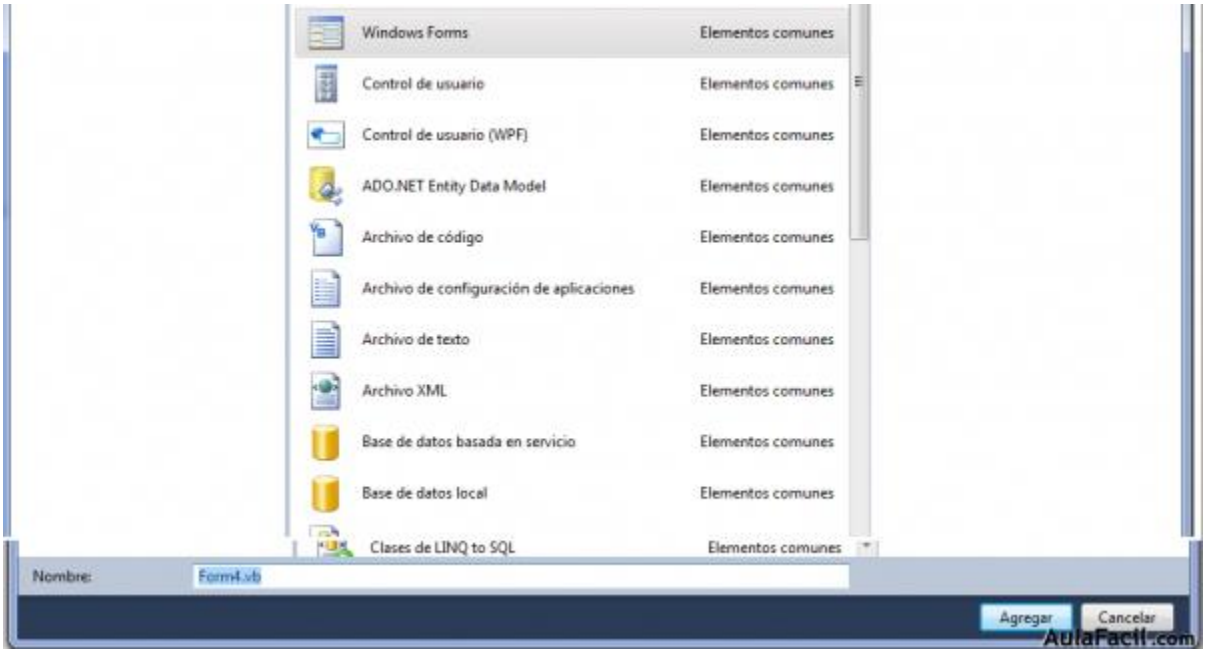
Y se desplegará el siguiente menú contextual.



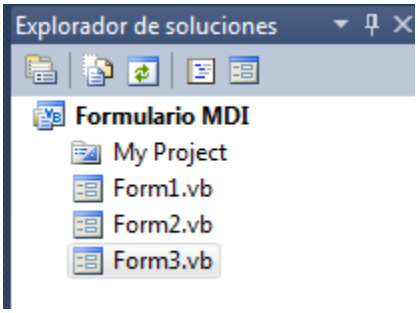
- Elija Agregar Windows Forms



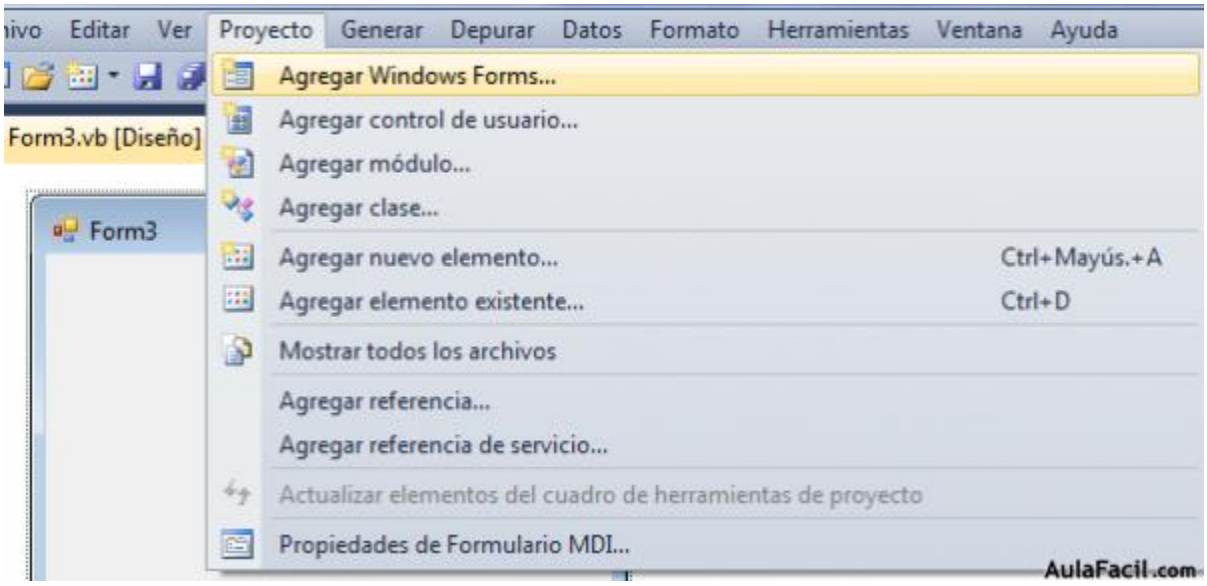
Luego se mostrará la siguiente ventana ya conocida por usted, en donde después de seleccionar Windows Forms, le da clic en **Agregar;** pero antes debe elegir el nombre del nuevo formulario; En este ejemplo déjelo como está, **Form1.Vb** que dicho sea de paso en la siguiente imagen aparece como Form4.Vb por que es el tercer formulario que se agrega al proyecto.



Observe que en el explorador de soluciones ya aparece el formulario agregado.

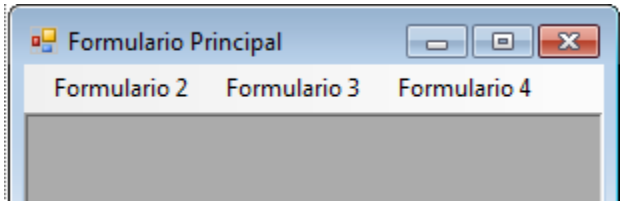


- Agregue otro formulario al proyecto pero para conocer otra forma de hacerlo, hágalo desde el menú **Proyecto ---> Agregar Windows Forms...**



Agregue un formulario más, usando uno de los dos métodos que se han explicado en este libro.

- Asegúrese que la propiedad **IsMdiContainer** de los formularios 2,3 y 4 esté en **False** y la del formulario uno en **True**. Esto permitirá que los primeros queden como contenido (**hijos**) y el segundo como contenedor (**padre)**
- Con el Formulario 1 desplegado, agregue un control Menú Strip y configúrelo (ya sabe como) para que quede como en la siguiente imagen.



Al dar clic en los menús correspondientes se mostrarán los formularios 2,3 y 4. El formulario uno no se programó en el menú por que es el formulario contenedor o principal.

- Dé doble clic en la pestaña del menú Formulario 2 para abrir el editor de código, y escriba las siguientes instrucciones.

Form2.Show()

- Clic en la pestaña Formulario 3 y escriba en el editor de código lo siguiente: **Form3.show()** y en la pestaña Formulario 4 al abrir el editor de código, escriba la siguiente instrucción: **Form4.show()**

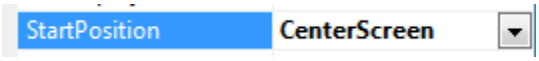
El código completo quedará así:

```
Public Class FrmPrincipal

    Private Sub Formulario1ToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Formulario1ToolStripMenuItem.Click
        Form2.Show()
    End Sub
    Private Sub Formulario3ToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Formulario3ToolStripMenuItem.Click
        Form4.Show()
    End Sub

    Private Sub Formulario2ToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Formulario2ToolStripMenuItem.Click
        Form3.Show()
    End Sub
End Class
```

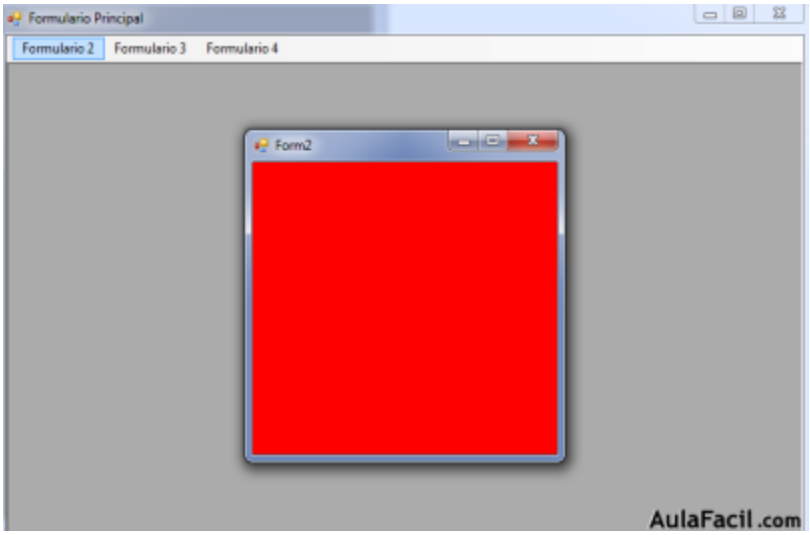
- Antes de probar el programa y para diferenciar los formularios configure la propiedad **BackColor** del formulario 2 en Rojo, del formulario 3 en Azul y la del Formulario 4 en Aqua.

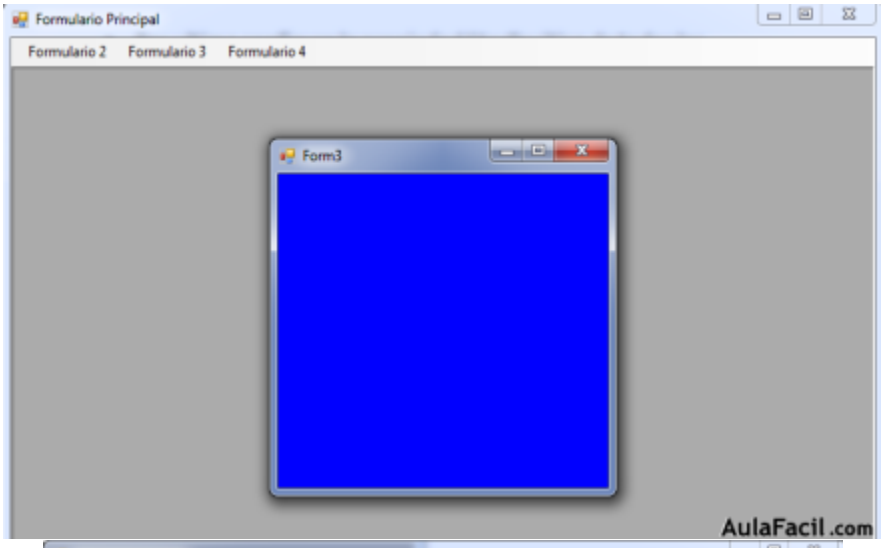
- Amplié el tamaño del Formulario 1 para que sea el más grande, por que recuerde que es el formulario contenedor.

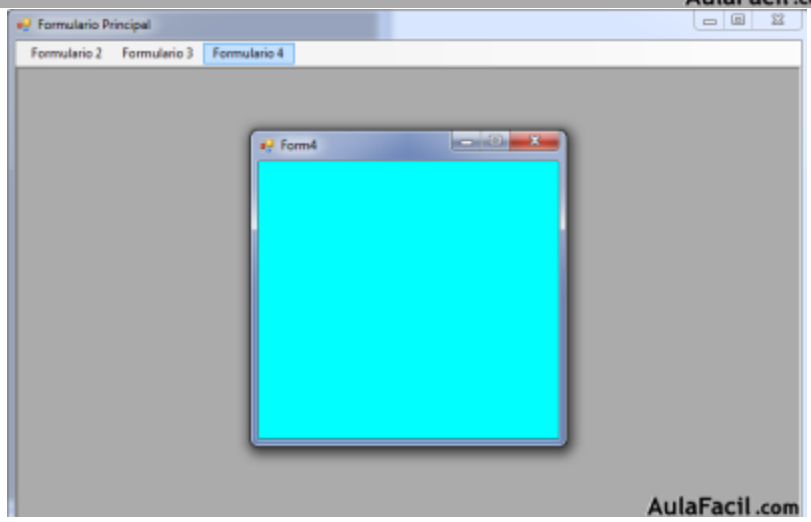
- Por último configure la propiedad **StartPosition** de todos los formularios en **CenterScreen** lo que hará que al ejecutar el programa los formularios aparezcan en el centro del monitor.



- Ejecute el programa y observe los resultados al dar clic en los menús correspondientes.







Si todo le salió bien, entonces guarde todo el proyecto, sino, pues revise para ver si hizo todos los pasos anteriores o identificar cualquier error.

A continuación, haremos otro ejercicio en donde se crea un formulario múltiple pero usando otro método.

Ejercicio MDI

- Cree un nuevo Proyecto y asígnele el nombre **MDI Pictures**
- Clic derecho en el proyecto, en el explorador de soluciones o en el menú proyecto y seleccione **Agregar nuevo elemento** y se abrirá la ventana como la de la imagen siguiente, en donde elegirá **Formulario primario MDI**

Bases De Datos en VISUAL BASIC NET

Definición de Base De Datos.

Una base de datos es una colección de valores o datos que pueden ser números telefónicos, nombres de personas, valores numéricos, etc.

Todos en la vida manejamos datos que necesitamos almacenar en algún lado.

Para eso sirven las bases de datos, para guardar datos y utilizarlos cuando los necesitemos.

Los mismos principios se aplican, pero un poco más complejos, al manipular con Visual Basic, bases de datos en **SQL**

Instalación del motor de base de datos SQL Server

Instalación de "SQL Server Developer" en nuestra computadora.

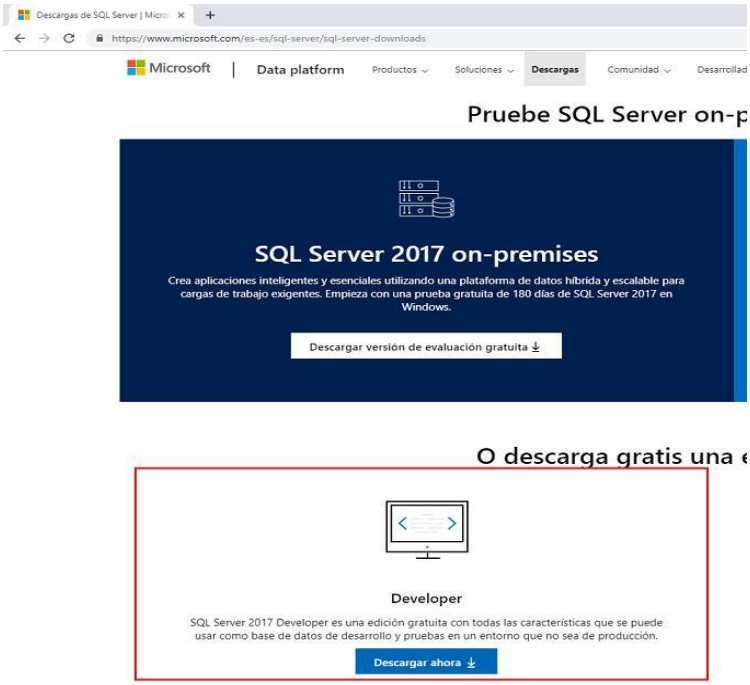
Procederemos a instalar el motor de base de datos SQL Server con el objetivo de confeccionar aplicaciones en Visual Basic .Net que puedan acceder a una base de datos (si ya tiene instalado el SQL Server puede obviar este concepto)

"SQL Server Developer" es una edición gratuita con todas las características que se puede usar como base de datos de desarrollo y pruebas en un entorno que no sea de producción.

El primer paso será descargar el "SQL Server Developer".

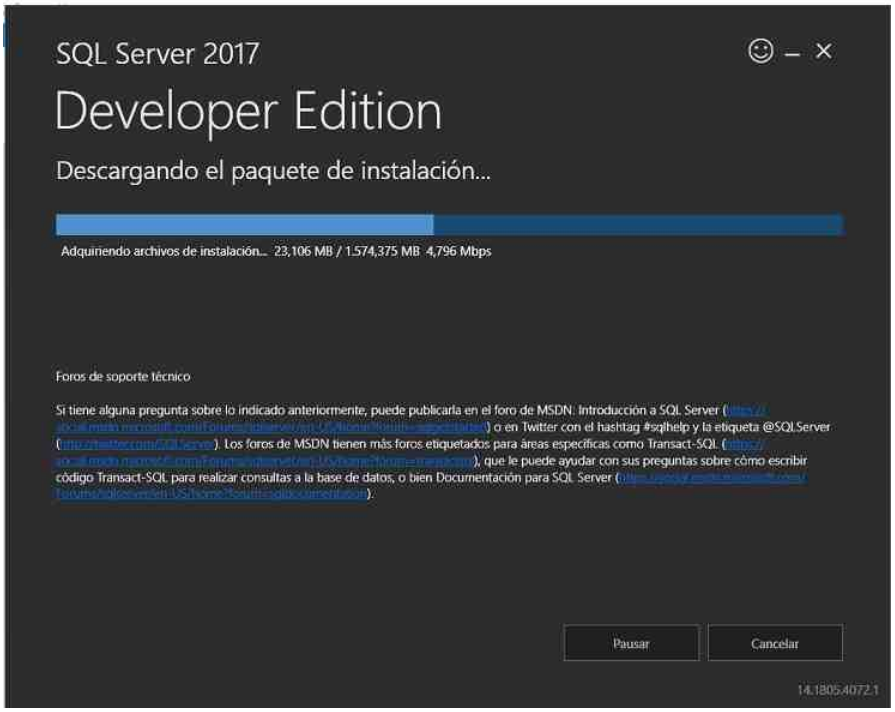
Debemos elegir la versión "SQL Server Developer:

Crédito: EL TÍO TECH

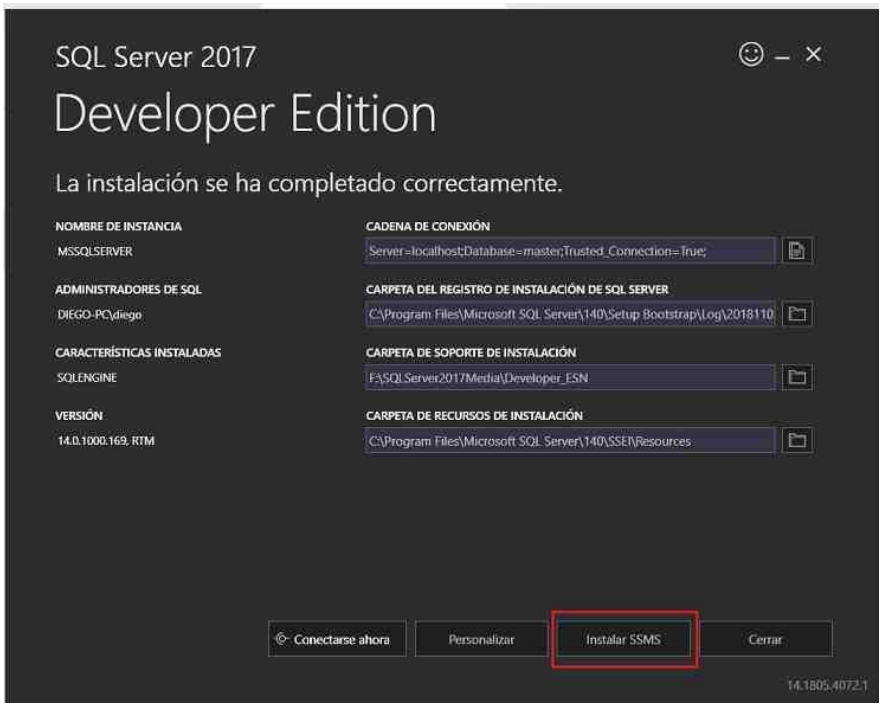


Luego de descargar el instalador procedemos a elegir la opción de "Instalación básica":

Luego deberemos esperar mientras se descarga todo el software:



Ya tenemos instalado el servidor de base de datos SQL Server, ahora elegimos la opción "Instalar SSMS" (SQL Server Management Studio, es un software visual que nos va ha permitir conectarnos a nuestro servidor de base de datos que acabamos de instalar, crear bases de datos, sus tablas, insertar datos etc.):

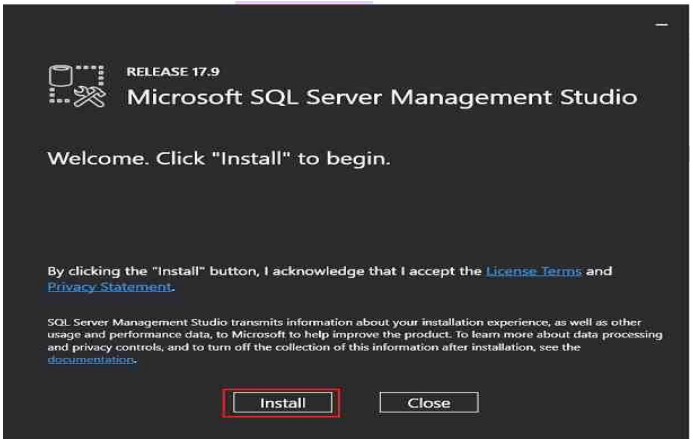


Instalación del SQL Server Management Studio

En el concepto anterior descargamos e instalamos el motor de base de datos SQL Server. Por otro lado también descargamos el SQL Server Management Studio que nos permitirá administrar las bases de datos.

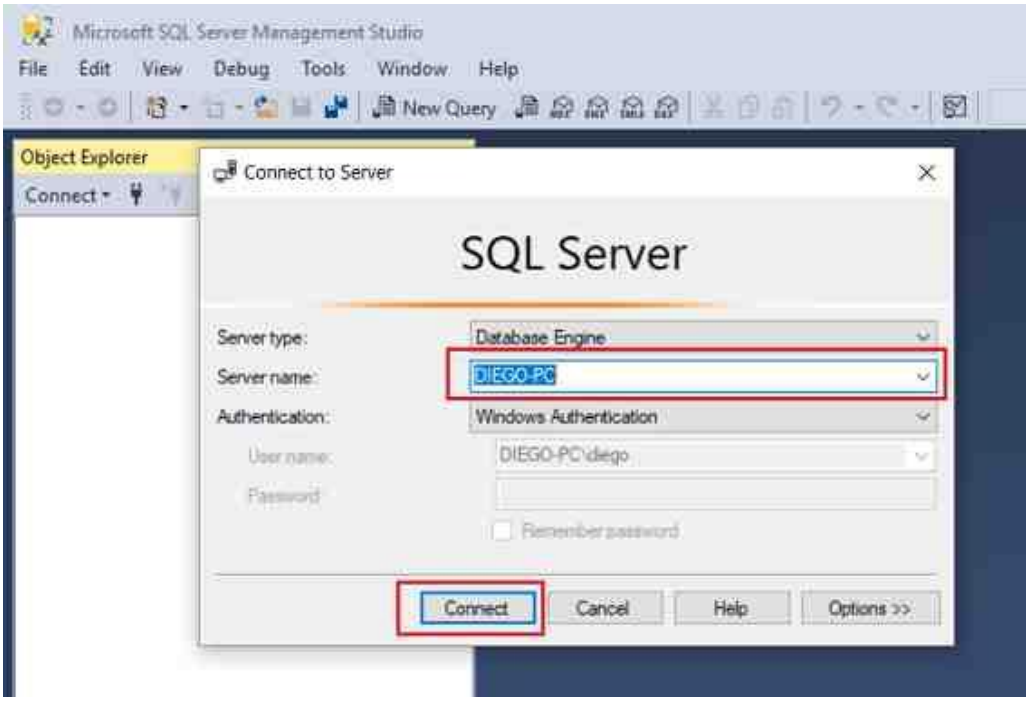
Instalación del "SQL Server Management Studio (SSMS)".

Descargamos el SSMS si no seleccionamos el botón "Instalar el SSMS" en el concepto anterior.

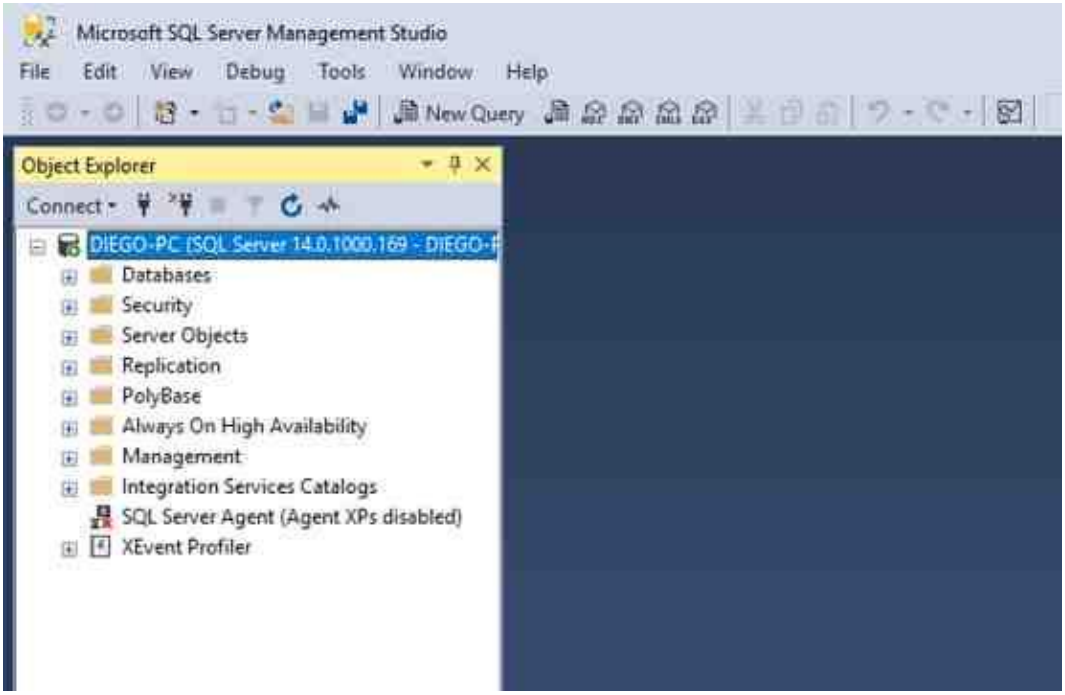


Una vez instalado el "SSMS" podemos acceder desde el menú de opciones de Windows:

Inmediatamente ingresamos nos aparece un diálogo donde veremos el nombre del servidor de Sql Server que acabamos de instalar y procederemos a conectarnos al mismo:



Si llegamos hasta acá significa que ya tenemos instalado en forma correcto el "SQL Server" y el "SQL Server Management Studio (SSMS)":

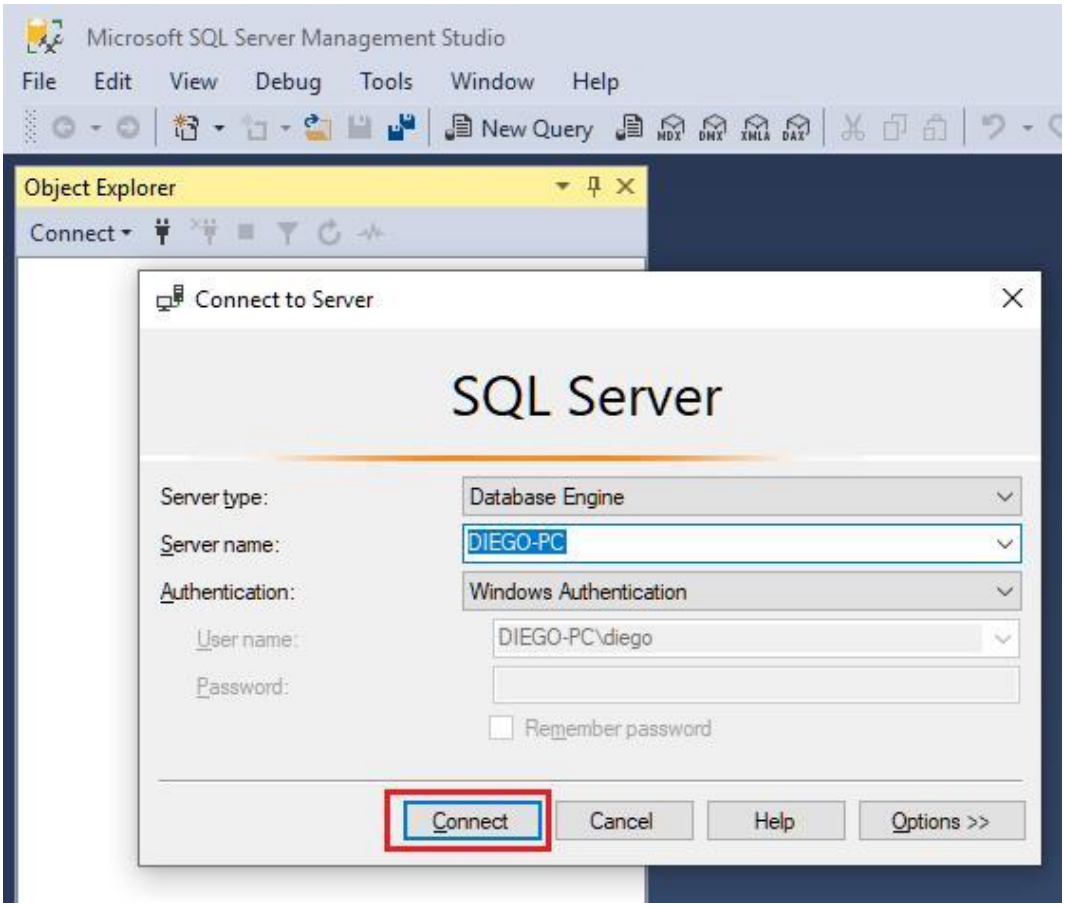


En los siguientes conceptos aprenderemos a acceder a las bases de datos que crearemos desde esta herramienta.

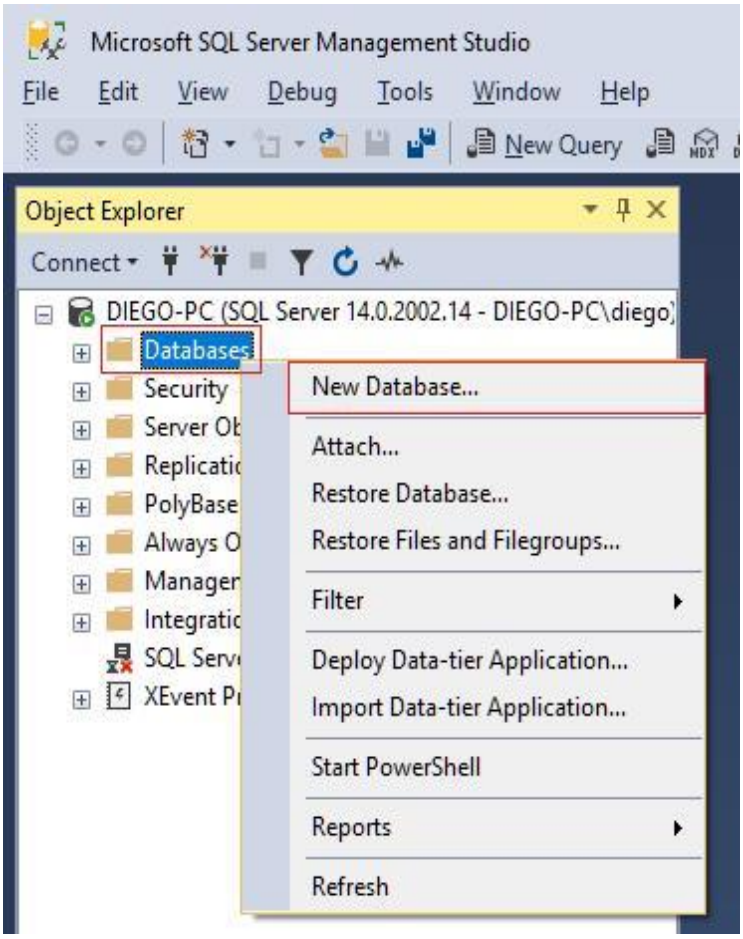
Ejecución del SQL Server Management Studio

Desde el menú de inicio de Windows podemos acceder al SQL Server Management Studio:

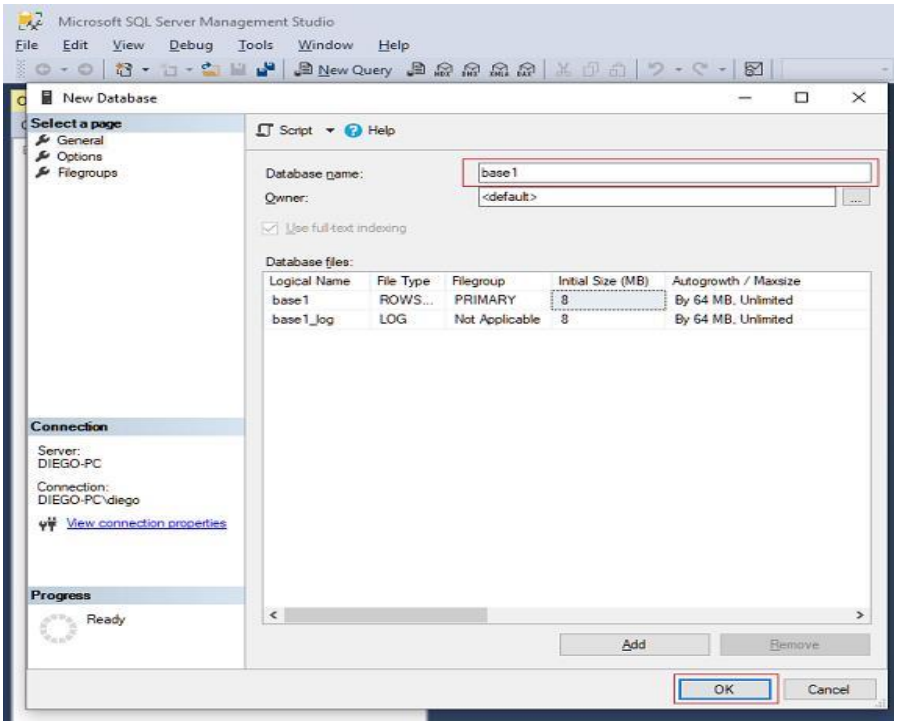
Cuando iniciamos el programa aparece un diálogo donde debemos ingresar "Nombre del Servidor" y la "Autenticación":



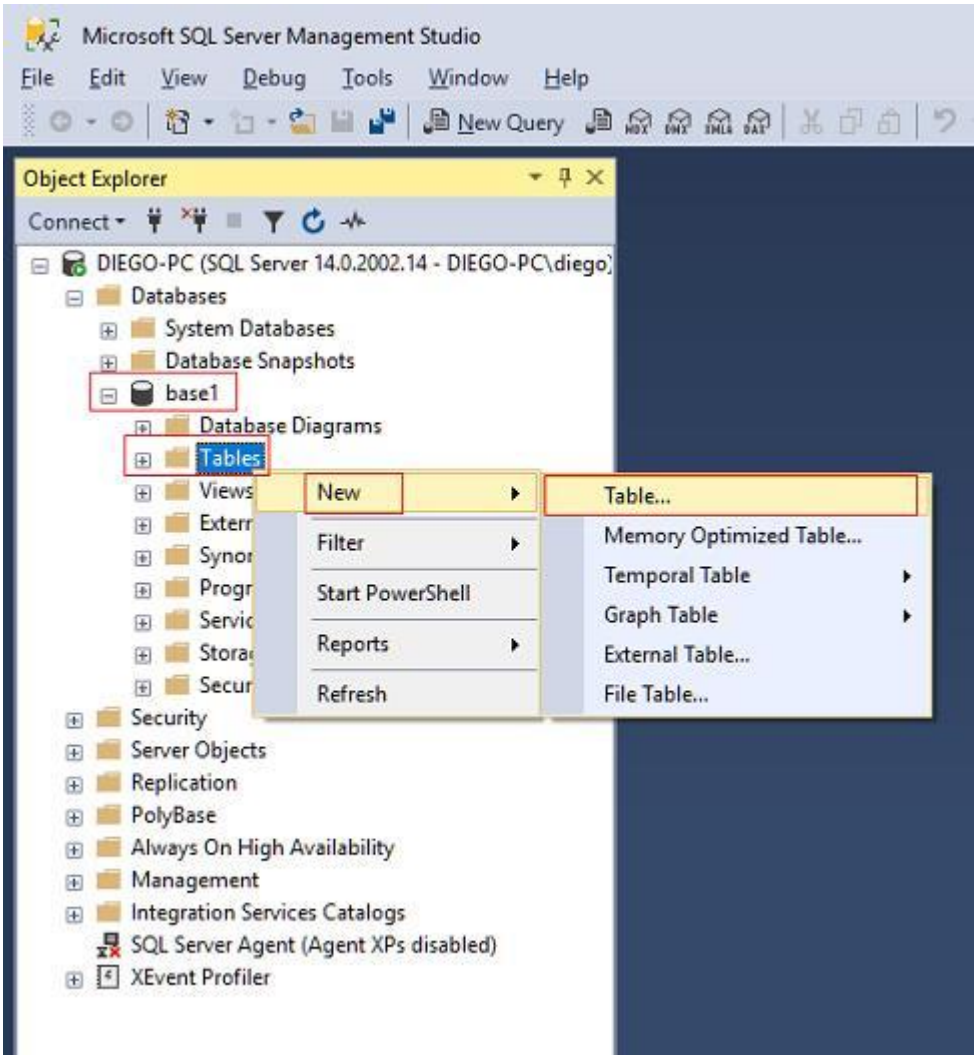
Ya luego de la conexión al servidor podemos por ejemplo proceder a crear nuestra primer base de datos, presionamos el botón derecho del mouse estando posicionado en "Base de datos" y elegimos "Nueva base de datos...":



Elegiremos como nombre para la base de datos a crear: "base1":



Ahora seleccionamos la "base1" y presionamos el botón derecho del mouse sobre "tablas" y elegimos la primer opción "tabla":



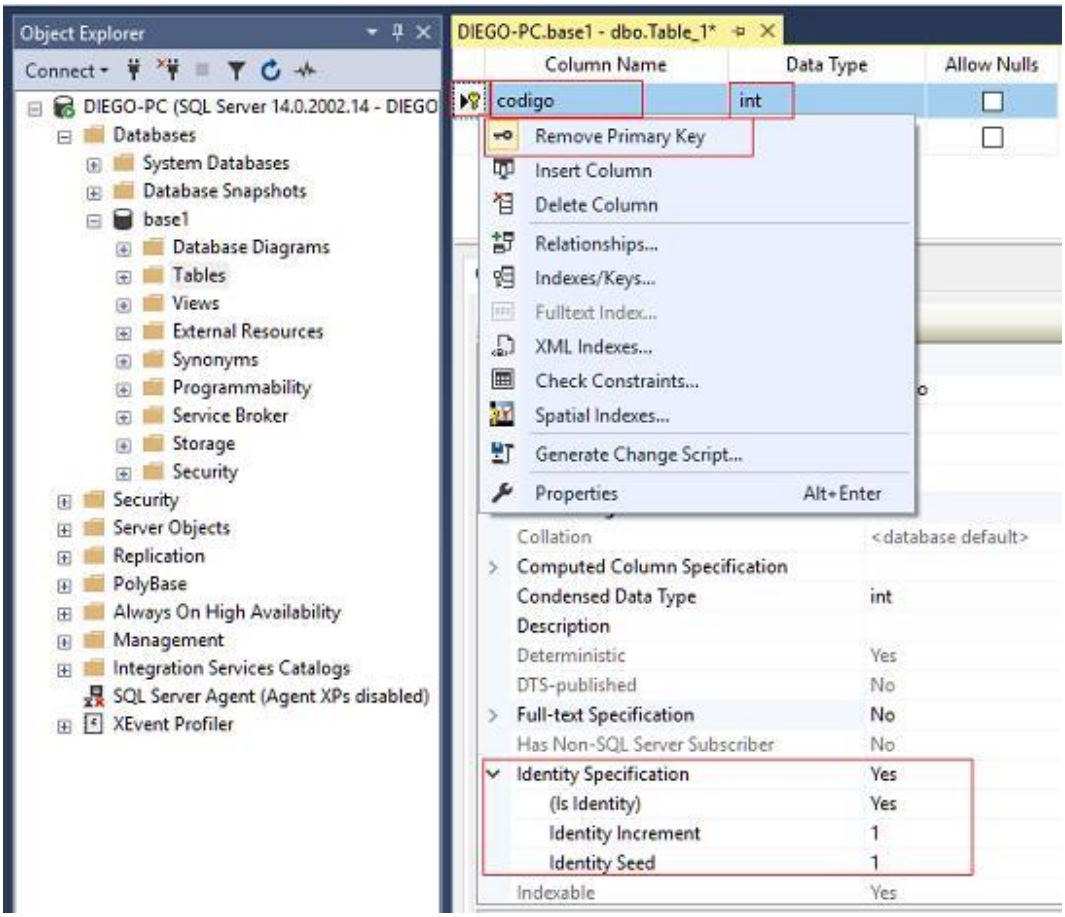
Vamos a crear una tabla llamada "articulos" que la definiremos con tres campos:

codigo int primary key identidad

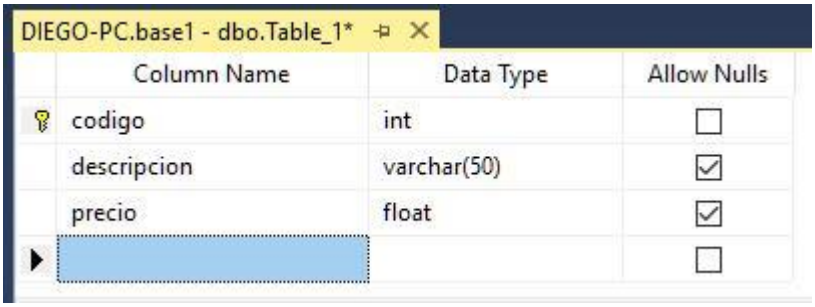
descripcion varchar 50

precio float

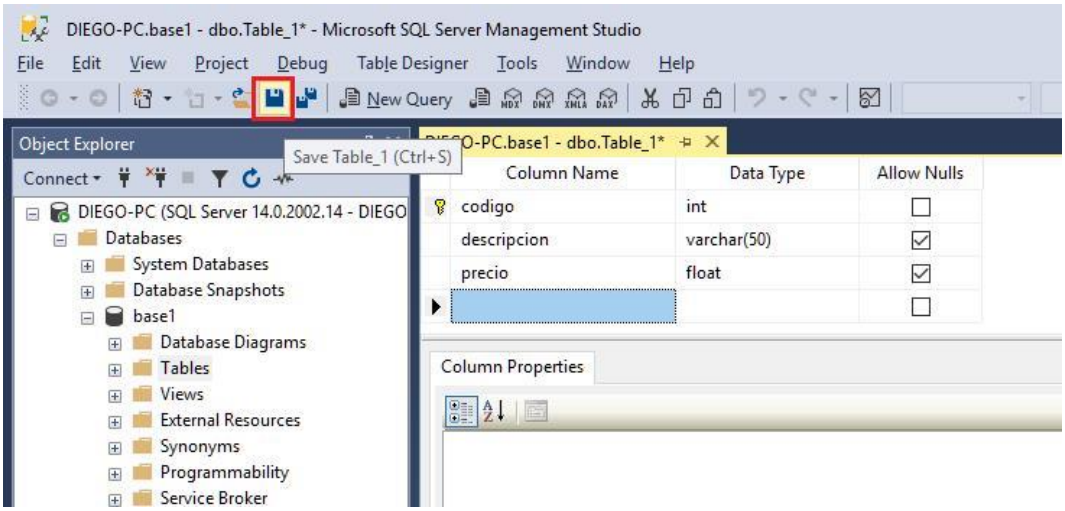
El primer campo lo creamos (debemos presionar el botón derecho del mouse sobre el campo para definirlo como clave primaria):



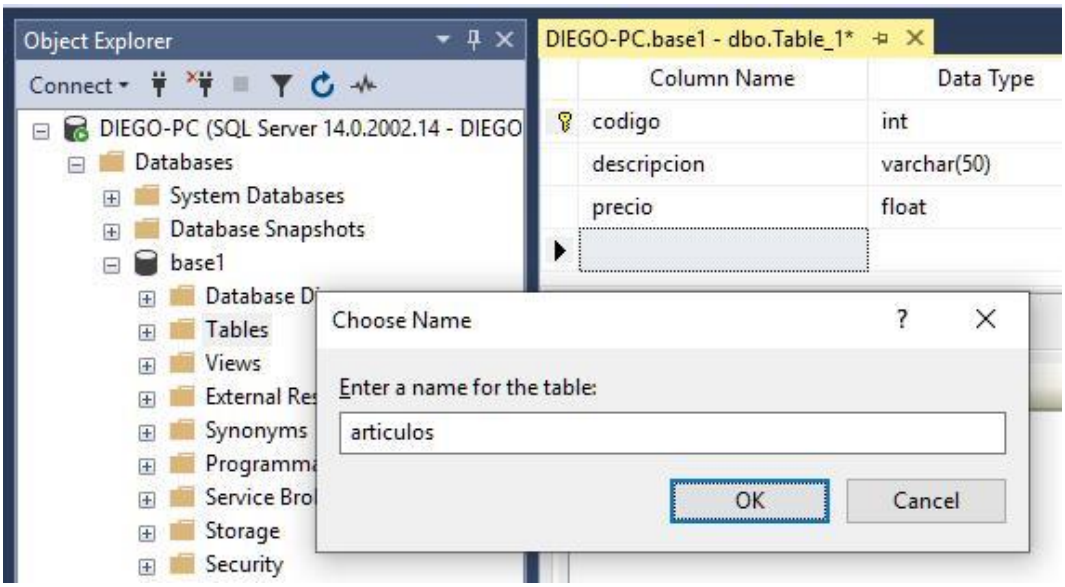
Los tres campos que definimos son:



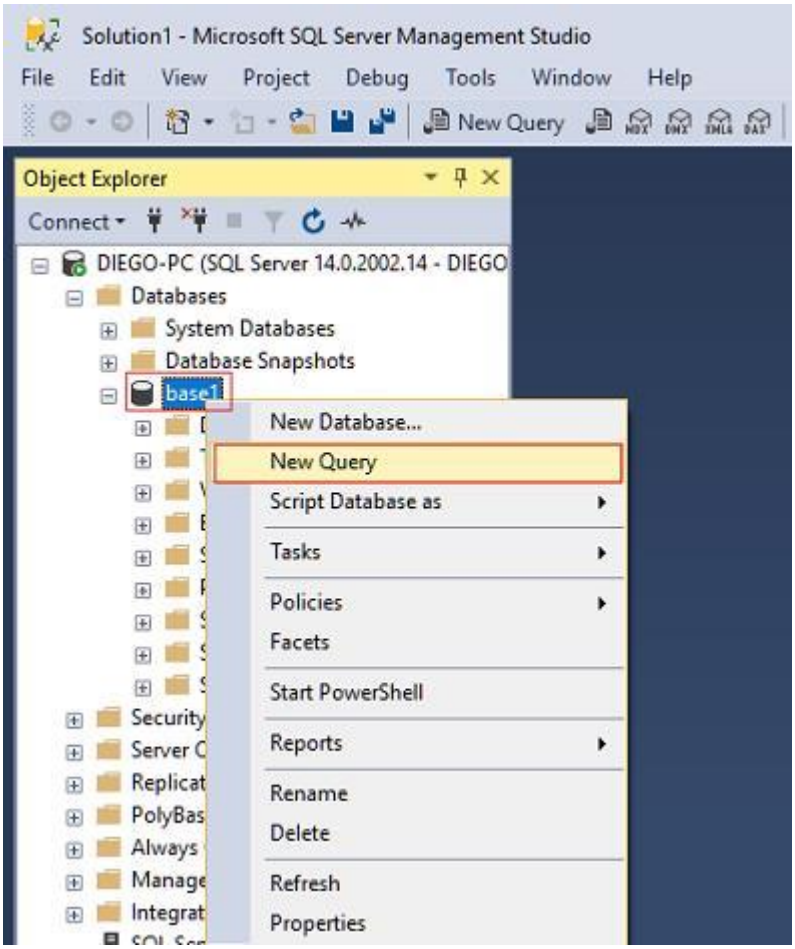
Y presionando el ícono siguiente definimos el nombre de la tabla:



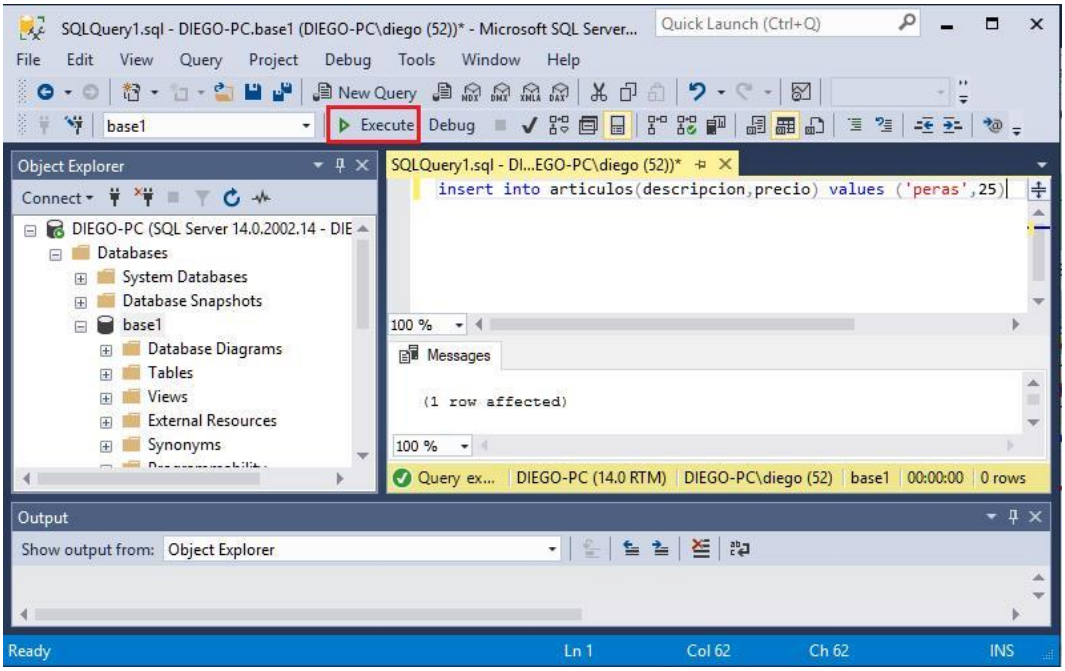
Asignamos como nombre a la tabla "artículos":



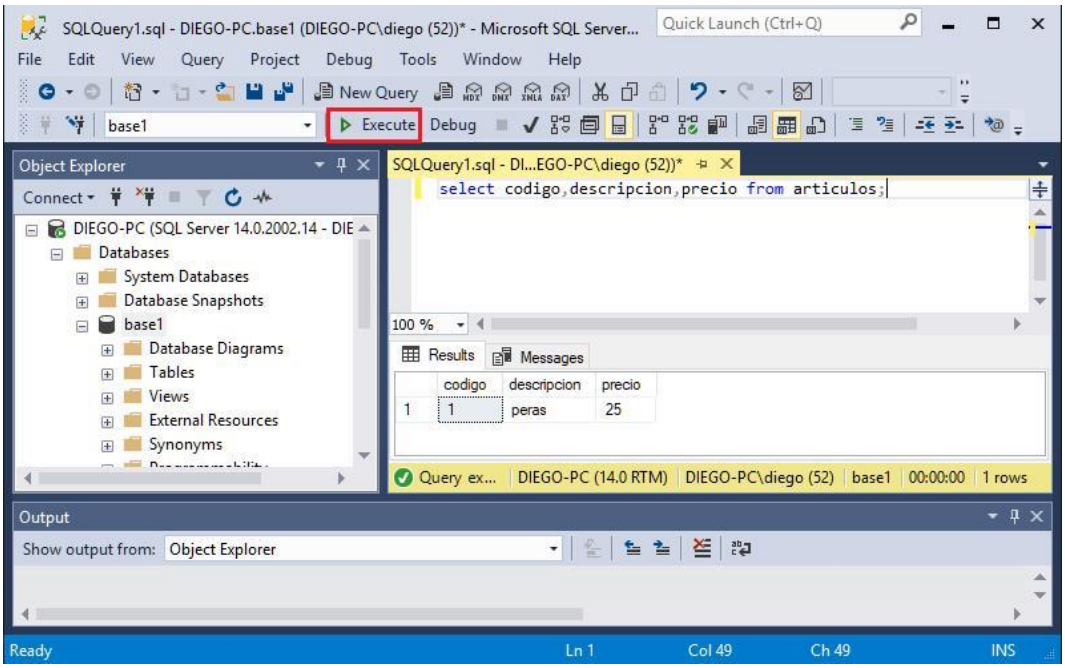
Para ejecutar comando SQL sobre una determinada base de datos procedemos a posicionar la flecha del mouse sobre una base de datos, por ejemplo "base1" y presionamos el botón derecho del mouse seleccionando "Nueva consulta":



En la ventana de consultas podemos escribir y ejecutar comandos SQL:



Si escribimos luego del comando insert un select podemos ver el registro añadido:



SqlConnection: Conexión con el SQL Server

Desde .Net disponemos de un conjunto de clases para conectarnos y pedir al SQL Server que ejecute comandos.

El primer paso es conectarnos con nuestro servidor de base de datos que instalamos en los conceptos anteriores y vimos como podemos comunicarnos utilizando el Microsoft SQL Server Management Studio.

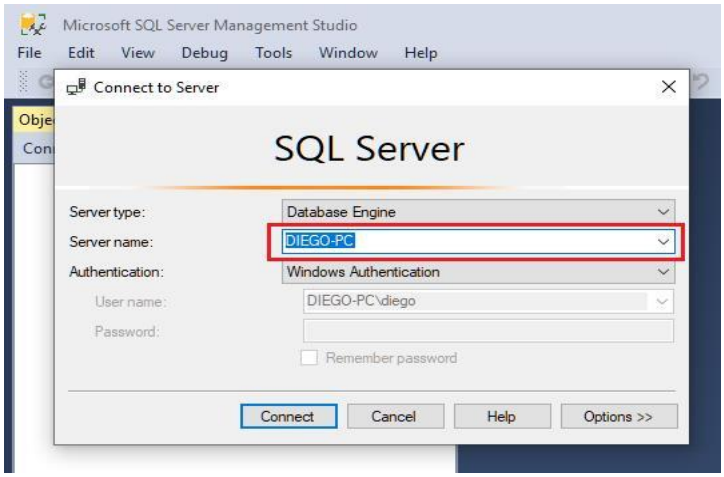
Problema

Confeccionar una aplicación que permita conectarnos con nuestro servidor de base de datos y seleccionar una determinada base de datos. Luego de la conexión cerrar la misma.

Para conectarnos con el servidor y seleccionar una base de datos debemos utilizar la clase SqlConnection:

```
Dim conexion As SqlConnection

conexion = New SqlConnection("server=DIEGO-PC;database=base1;integrated security = true")
```

El String que le pasamos al constructor del SqlConnection le indicamos como primer dato el nombre de nuestro servidor, en su máquina lo puede ubicar cuando arranca el SQL Server Management Studio:



El segundo dato que se le pasa es el nombre de la base de datos con la que nos comunicaremos, en nuestro caso creamos previamente la base de datos "base1" y finalmente el último dato obligatorio es indicar "integrated security = true", esto hace que se utilice el usuario propio de Windows y no necesitemos por el momento crear otros usuarios propios de SQL Server.

Creemos un nuevo proyecto en Visual Studio y demos como nombre: BaseDeDatos1. La interfaz visual es un simple botón:

El código fuente es:

```
Imports System.Data.SqlClient

Public Class Form1
```

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click

    Dim conexion As SqlConnection

    conexion = New SqlConnection("server=DIEGO-PC;database=base1;integrated security = true")

    conexion.Open()

    MessageBox.Show("Se abrió la conexión con el servidor SQL Server y se seleccionó la base de datos")

    conexion.Close()

    MessageBox.Show("Se cerró la conexión.")

  End Sub

End Class
```

Lo primero que hacemos es incluir el espacio de nombres:

```
Imports System.Data.SqlClient
```

Este espacio de nombres agrupa un conjunto de clases orientadas a conectarse con el SQL Server, si no las importamos no se pueden utilizar.

Este espacio de nombres define la clase SqlConnection.

Para el evento Click del button1 definimos y creamos un objeto de la clase SqlConnection, el nombre del servidor es el que podemos ver desde el SQL Server Management Studio:

```
    Dim conexion As SqlConnection

    conexion = New SqlConnection("server=DIEGO-PC;database=base1;integrated security = true")
```

Abrimos la conexión:

conexion.Open()

Mostramos un mensaje que la conexión se abrió:

MessageBox.Show("Se abrió la conexión con el servidor SQL Server y se seleccionó la base de datos")

Cerramos la conexión:

conexion.Close()

Mostramos finalmente un mensaje del cierre de la conexión:

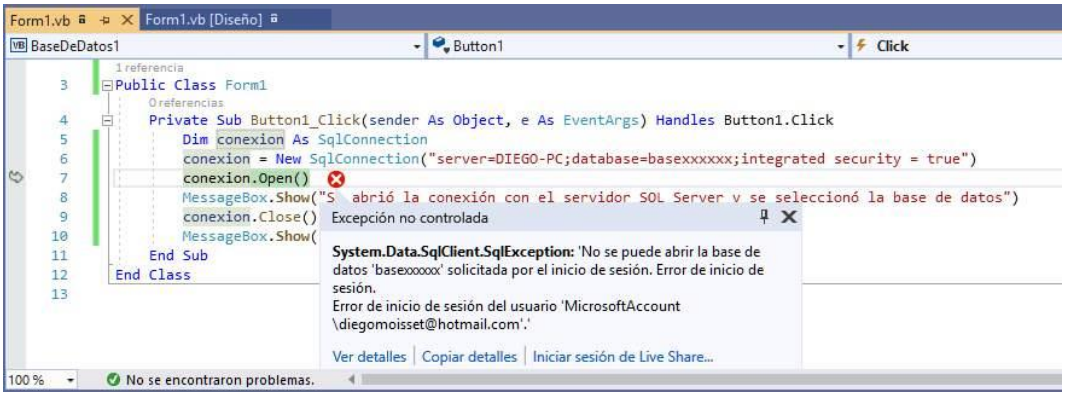
MessageBox.Show("Se cerró la conexión.")

Si aparece algún error es que especificó en forma incorrecta la cadena de conexión, por ejemplo si disponemos un nombre equivocado de servidor por ejemplo:

conexion = New SqlConnection("server=**********;database=base1;integrated security = true")

Luego debe detenerse nuestro programa indicando donde ocurrió el error:

También ocurrirá un error si disponemos un nombre de base de datos inexistente:



Si todo está correcto podemos seguir con los próximos conceptos para pedir que SQL Server ejecute comando y recuperar los resultados.

SqlCommand (insert): Insertar registro

Vimos en el concepto anterior como conectarnos al servidor de SQL Server desde Visual Basic .Net y seleccionar una base de datos. Veremos ahora como pedir que el SQL Server ejecute un comando "INSERT".

Trabajaremos con la base de datos "base1" que creamos en conceptos anteriores y con la tabla "articulos" que también creamos previamente con la estructura:

codigo int primary key identidad

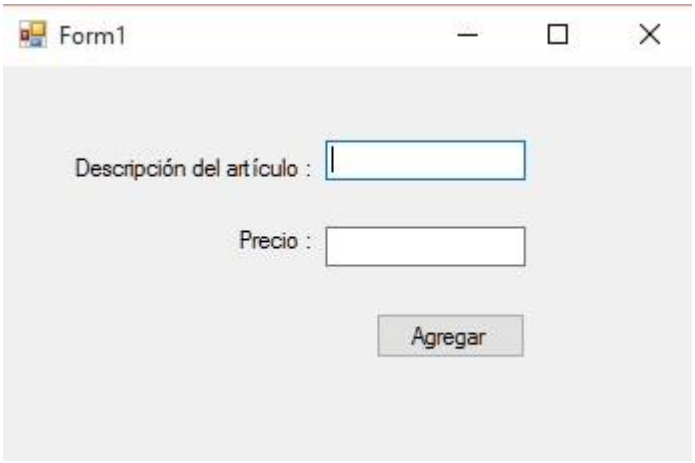
descripcion varchar 50

precio float

Problema

Implementar una interfaz visual para el alta o carga de registros en la tabla artículos.

Crearemos un proyecto llamado "basededatos2" con la siguiente interfaz visual:



El código fuente de la aplicación para efectuar altas es:

```
Imports System.Data.SqlClient


Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click

        Dim conexion As SqlConnection

        conexion = New SqlConnection("server=DIEGO-PC ; database=base1 ; integrated security = true")

        conexion.Open()

        Dim descri As String = TextBox1.Text

        Dim precio As String = TextBox2.Text

        Dim cadena As String = "insert into articulos(descripcion,precio) values ('" & descri & "'," & precio & ")"

        Dim comando As SqlCommand
```

```
        comando = New SqlCommand(cadena, conexion)

        comando.ExecuteNonQuery()

        MessageBox.Show("Los datos se guardaron correctamente")

        TextBox1.Text = ""

        TextBox2.Text = ""

        conexion.Close()

    End Sub

End Class
```

Importamos el espacio de nombres que contiene las clases SqlConnection y SqlCommand:

```
Imports System.Data.SqlClient
```

Lo primero que hacemos para el evento Click del botón es definir y crear un objeto de la clase SqlConnection y proceder a abrir la conexión:

```
        Dim conexion As SqlConnection

        conexion = New SqlConnection("server=DIEGO-PC ; database=base1 ; integrated security = true")

        conexion.Open()
```

Tenemos que guardar en la variable cadena un comando INSERT válido tomando los datos ingresados en los dos TextBox:

```
        Dim descri As String = TextBox1.Text

        Dim precio As String = TextBox2.Text

        Dim cadena As String = "insert into articulos(descripcion,precio) values ('" & descri & "'," & precio & ")"
```

Como el campo descripción es de tipo varchar debemos incluir las comillas simples:

''' & descri & '''

Y como el precio es float no incluimos las comillas simples:

'' & precio & ''

Si cargamos por teclado por ejemplo el producto: manzanas y con un precio : 20, luego la variable cadena tiene el valor:

insert into articulos(descripcion,precio) values ('manzanas',20)

Como podemos ver es una forma sencilla de elaborar un comando SQL siempre y cuando tengamos pocos campos que iniciar.

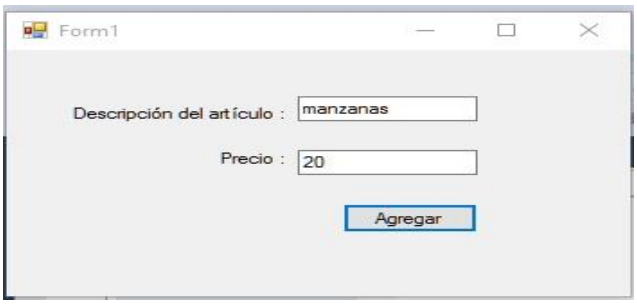
Creamos un objeto de la clase SqlCommand y le pasamos al constructor un String con el comando SQL y la referencia a la conexión.

```
Dim comando As SqlCommand

comando = New SqlCommand(cadena, conexion)
```

Seguidamente llamamos al método ExecuteNonQuery que procede a comunicarse con el servidor para que se ejecute el comando SQL configurado en la línea anterior:

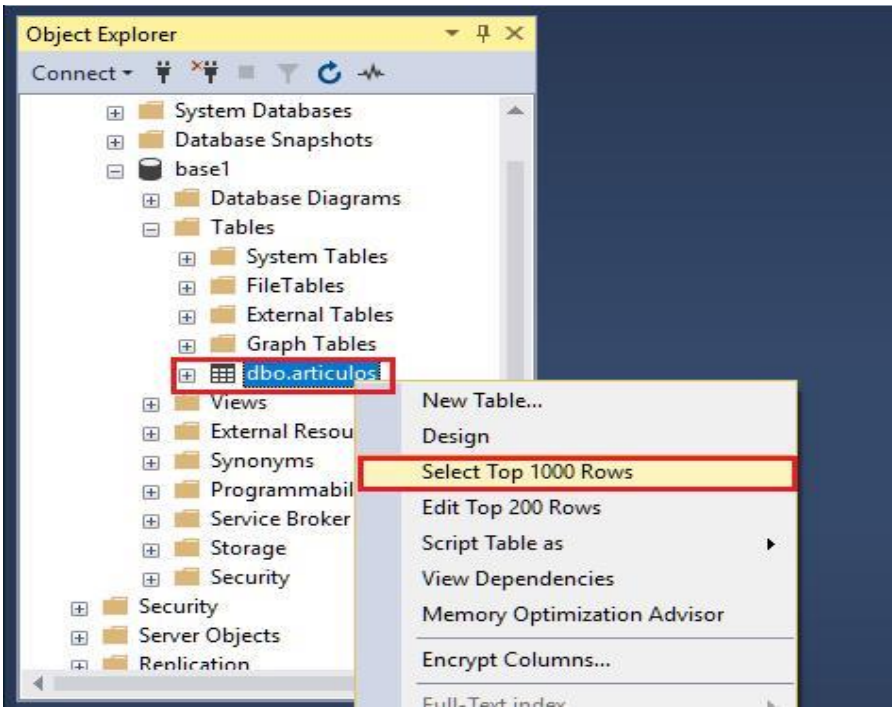
```
comando.ExecuteNonQuery();
```

Ejecutemos el programa y realicemos la carga de un artículo:

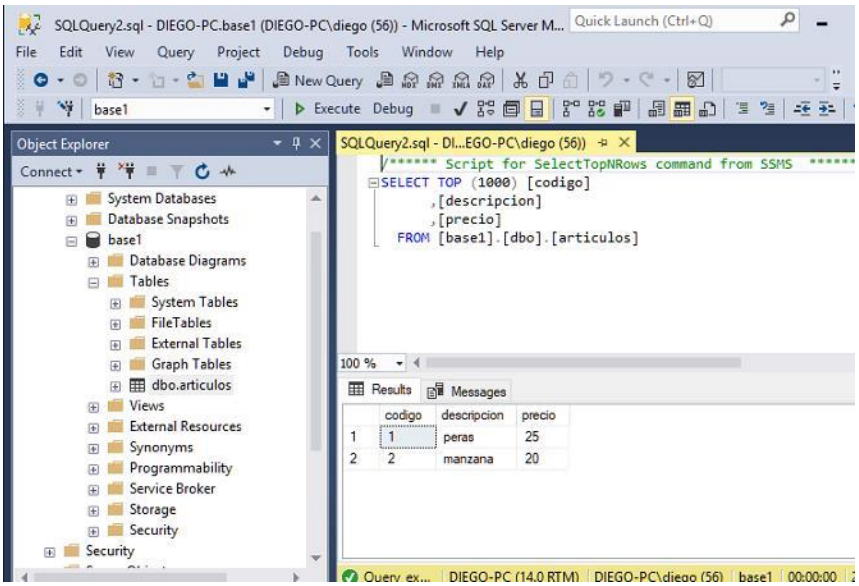


Por el momento la única forma que tenemos para consultar la tabla "articulos" es abrir el "Microsoft SQL Server Management Studio" seleccionar la base de datos "base1"

y luego presionar el botón derecho del mouse eligiendo la opción "Seleccionar las primeras 1000 filas":



Y de esta forma poder ver los registros que almacena la tabla "articulos":



SqlCommand (select): Consultar registros

En el concepto anterior recuperamos todos los registros de una tabla. Ahora veremos como podemos rescatar uno en particular.

Problema

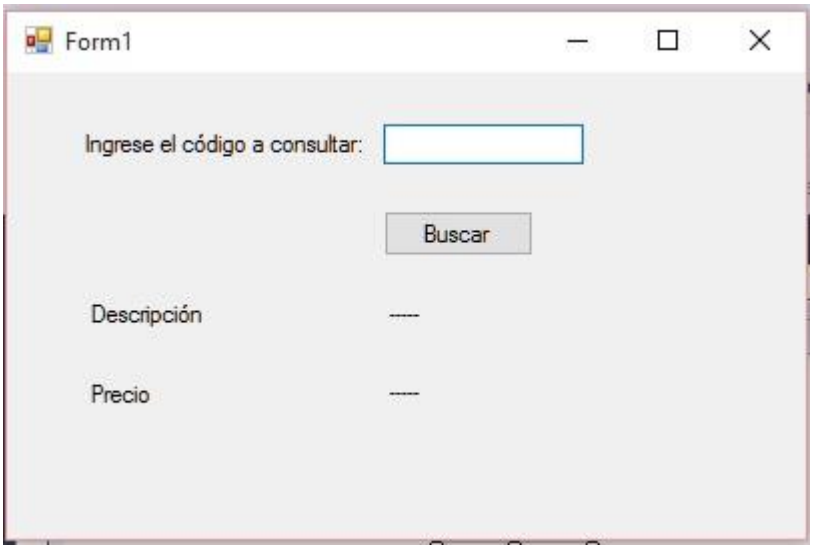
Implementar la consulta de un artículo ingresando por teclado el código y recuperando la descripción y el precio.

Crear un proyecto llamado: BaseDatos4 y definir la siguiente interfaz visual:

5 objeto de la clase Label.

1 objeto de la clase Button.

1 objeto de la clase TextBox.



En el evento Click procedemos a buscar el código del artículo ingresado en el TextBox. El código fuente es:

```
Imports System.Data.SqlClient


Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click

        Dim conexion As SqlConnection

        conexion = New SqlConnection("server=DIEGO-PC ; database=base1 ; integrated security = true")

        conexion.Open()

        Dim cod As String = TextBox1.Text

        Dim cadena As String = "select descripcion, precio from articulos where codigo=" & cod

        Dim comando As SqlCommand

        comando = New SqlCommand(cadena, conexion)

        Dim registro As SqlDataReader

        registro = comando.ExecuteReader()

        Label4.Text = ""

        Label5.Text = ""

        If registro.Read() = True Then

            Label4.Text = registro("descripcion")

            Label5.Text = registro("precio")

        Else
```

```
MessageBox.Show("No existe un artículo con el código ingresado")

        End If

        conexion.Close()

    End Sub

End Class
```

Importamos el espacio de nombres:

```
Imports System.Data.SqlClient
```

En el evento Click del botón "Buscar" procedemos a conectarnos con el motor de base de datos y abrir la conexión:

```
        Dim conexion As SqlConnection

        conexion = New SqlConnection("server=DIEGO-PC ; database=base1 ; integrated security = true")

        conexion.Open()
```

Seguidamente confeccionamos un String con el comando SQL select para recuperar la descripción y el precio del artículo cuyo código coincide con el valor ingresado por teclado:

```
        Dim cod As String = TextBox1.Text

        Dim cadena As String = "select descripcion, precio from articulos where codigo=" & cod
```

Creamos un objeto de la clase SqlCommand pasando el comando SQL y la referencia a la conexión:

```
        Dim comando As SqlCommand

        comando = New SqlCommand(cadena, conexion)
```

Recuperamos un objeto de la clase SqlDataReader que retorna el objeto de la clase SqlCommand mediante el llamando al método EcecuteReader:

```
Dim registro As SqlDataReader

registro = comando.ExecuteReader()
```

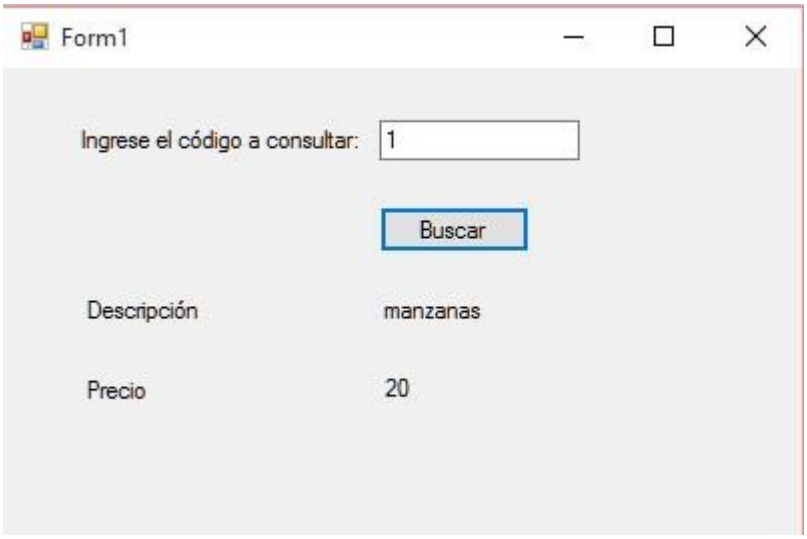
Si el resultado del comando select recuperó un registro de la tabla articulos luego la llamada al método Read se verifica verdadero y procedemos a mostrar el resultado por pantalla:

```
If registro.Read() = True Then

   Label4.Text = registro("descripcion")

   Label5.Text = registro("precio")
```

En el caso que hayamos ingresado un código inexistente procedemos a mostrar un mensaje por el Else:

```
Else

   MessageBox.Show("No existe un artículo con el código ingresado")

End If
```

La interfaz en funcionamiento es:


Form1
Ingrese el código a consultar:
1
Buscar
Descripción
manzanas
Precio
20

REFERENCIAS CONSULTADAS

1. Aprender a Programar. **Operadores lógicos y matemáticos en Visual Basic. Math. Comentarios en VB. Rem. Ejemplos (CU00314A)**Operadores lógicos y matemáticos en Visual Basic. Math. Comentarios en VB. Rem. Ejemplos (CU00314A) (aprenderaprogramar.com)

2. Aula Fácil **Base de Datos** 🥇▷【 Bases De Datos - Visual Basic paso a paso 】 (aulafacil.com)

3. Aula Fácil. **FORMULARIOS MDI.**🥇▷【 Formularios MDI - Visual Basic paso a paso 】 (aulafacil.com)

4. Declaración de Variables Visual basic declaracion de una variable - Buscar con Google

5. El tío Tech **Curso gratis de VB** GitHub Copilot gratis en Visual Studio https://visualstudio.microsoft.com/es

   https://eltiotech.com/curso-gratis-vba-para-macros-en-excel/

6. SSMS

RECURSOS INTERACTIVOS

1. Conectar una base de datos Access a Visual Studio Net https://youtu.be/ugmXldSSz2Q?si=gN-0WDKbeGxnRxmv

2. Declaración de Variables https://www.youtube.com/watch?v=PmGyjTg_vVw

3. Ejercicios prácticos con Condicionales https://youtu.be/nUSvEfzD4mc?si=EuXaQgRMrm8aodAg

4. Formularios https://www.youtube.com/shorts/_SVl9f5lp3k?feature=share

5. Proyectos y ejercicios con bucles https://youtu.be/hFz_sta3Ce0?si=I5vL1JhqHReWFm_3

6. Proyecto Práctico https://youtu.be/cmrwdaJ_waA?si=iBtRRnR1xdR3_ijP https://youtu.be/NzCzMqiJnoE?si=7wL4xDk5HHd5i16k

7. Proyecto Práctico **Listado de números y sumatoria. (Uso de ciclo FOR) VB 2010** https://youtu.be/vInek3jFsuA?si=JS9dvVXLPbLc_6ca

8. Proyecto Práctico **Suma de dos Números** https://youtu.be/NzCzMqiJnoE?si=GZwGXp1rNSMelXs_

9. Proyecto Práctico **Invertir una palabra** **https://youtu.be/Deg_923n5cQ?si=_L8wz0sP98BYxu4t**

10. Proyecto Práctico **Detectar símbolos y contarlos VB 2010** https://youtu.be/03yi__INGoM?si=_xKNIpsqe_Mvdq5M