TEMA 3. MANEJO DE LA CONSOLA.

Clase Console:

Precedida de la expresión Console. , existen una serie de métodos que permiten interrelacionarse con la pantalla, de todos los métodos existentes destacan cuatro:

* Console.Write (): Permite escribir datos en la consola, y no se produce salto de carro del cursor.
* Console.Write Line (): Escribe información en la consola y i produce salto de línea. No se utiliza para visualizar literales, para esto se utiliza en último.
* Console.Read (): Detiene la ejecución de un programa y espera a que el usuario introduzca datos que se asignan a una variable. No se produce el salto de carro.
* Console.ReadLine (): Igual que la anterior pero con salto de carro.

Para visualizar el contenido de una variable, una de las formas que se pueden utilizar son los marcadores de posición, que consiste en hacer una numeración entre llaves y empezando por cero, de las variables que se quieren visualizar, ejemplo: Console.Write (“El numero es: {0}”, a). Lo que hay detrás de la coma es la lista de variables, y lo que va entre llaves es la lista.

Ejer2:

Realizar un ejercicio que introduzca por teclado mi nombre y la edad y la visualiza por pantalla.

Module Module1

Sub Main()

Dim nombre As String

Dim edad As Integer

Console.Write("Introduce tu nombre: ")

nombre = Console.ReadLine()

Console.Write("Introduce tu edad: ")

edad = Console.ReadLine()

Console.WriteLine("El nombre y la edad son: {0}, {1} años", nombre, edad)

Console.Read()

End Sub

End Module

Ejer3:

Realizar un proyecto que me pida dos números por teclado, visualizando la suma, resta, multiplicación y división.

Module Module1

Sub Main()

Dim a As Integer

Dim b As Integer

Dim suma As Integer

Dim resta As Integer

Dim multiplicacion As Integer

Dim division As Double

Console.Write("Escriba el primer número: ")

a = Console.ReadLine()

Console.Write("Escriba el segundo número: ")

b = Console.ReadLine()

Console.WriteLine()

suma = a + b

Console.WriteLine("La suma es: {0}", suma)

resta = (a) - (b)

Console.WriteLine("La resta es: {0}", resta)

multiplicacion = a \* b

Console.WriteLine("La multiplicacion es: {0}", multiplicacion)

division = a / b

'para truncar un decimal a dos numeros se pone math.Round(variable,nº decimales) donde se quiera que aparezca la division

Console.WriteLine("La division es: {0}", Math.Round(division, 2))

Console.Read()

End Sub

End Module

ESTRUCTURAS DE CONTROL:

Una estructura de control permite la toma de decisiones y la repetición de acciones en un programa, para ello existen unas sentencias de control o instrucciones:

CONDICIONALES:

Sintaxis:

Estructura condicional simple:

IF condición THEN

Instructions

END IF

Estructura condicional doble:

IF condición THEN

Inst

ELSE

Inst

END IF

Estructura de control anidada:

IF condición THEN

Inst

ELSE

IF condición THEN

Inst

END IF

END IF

Ejer4

Realizar un proyecto que creando un menú de cuatro opciones, me visualice que opción se ha pulsado:

Module Module1

Sub Main()

Dim opcion As Integer

Console.WriteLine("Opción 1")

Console.WriteLine("Opción 2")

Console.WriteLine("Opción 3")

Console.WriteLine("Opción 4")

Console.Write("¿Qué opción desea pulsar? ")

opcion = Console.ReadLine()

If opcion = 1 Then

Console.WriteLine()

Console.WriteLine("Has pulsado la primera opción")

Console.Read()

End If

If opcion = 2 Then

Console.WriteLine()

Console.WriteLine("Has pulsado la segunda opción")

Console.Read()

End If

If opcion = 3 Then

Console.WriteLine()

Console.WriteLine("Has pulsado la tercera opción")

Console.Read()

End If

If opcion = 4 Then

Console.WriteLine()

Console.WriteLine("Has pulsado la cuarta opción")

Console.Read()

End If

End Sub

End Module

Condicionales múltiples:

Select case variable

Case opc1

Inst

Case opc2

Inst

Case opc3

Inst

Case else (cualquier otra)

Inst

End case

Para las opciones case se pueden poner varias opciones en uno mismo separados por comas, si se pone To, x ejemple 2 To 4, es un rango de opciones que va del dos a cuatro.

Ejer5:

Realizar un ejercicio que introduciendo por teclado una nota entera, me visualice en letra la calificación.

Module Module1

Sub Main()

Dim nota As Integer

Console.Write("Escriba la nota del alumno: ")

nota = Console.ReadLine()

Select Case nota

Case 1 To 3

Console.WriteLine()

Console.WriteLine("Nota: Muy insuficionete (1-3)")

Console.Read()

Case 1 To 5

Console.WriteLine()

Console.WriteLine("Nota: Insuficiente (3 - 5)")

Console.Read()

Case 5

Console.WriteLine()

Console.WriteLine("Nota: Suficiente (5)")

Console.Read()

Case 6

Console.WriteLine()

Console.WriteLine("Nota: Bien (6)")

Console.Read()

Case 7, 8

Console.WriteLine()

Console.WriteLine("Nota: Notable (7-8)")

Console.Read()

Case 9, 10

Console.WriteLine()

Console.WriteLine("Nota: Sobresaliente (9-10)")

Console.Read()

End Select

End Sub

End Module

ESTRUCTURAS REPETITIVAS:

For … Next: Estructura repetitiva que de antemano se sabe el número de veces que se va a realizar. Sintaxis:

FOR variablecontrol = valorinicial TO valorfinal [STEP salto(varaumento/decremento)]

INST

NEXT variablecontrol

Si no se pone step es de uno en uno. Ejemplo:

FOR a = 1 TO 10 [STEP 1]

Console.Write(“{0}”,a)

Next a

Para salir de un for se pone la condición de salida, se escribe Exit For evitando que se llegue al final.

Los for pueden estar anidados.

Ejer6:

Realizar un programa que aparezcan todas las tablas de multiplicar del o al 10:

Module Module1

Sub Main()

Dim res As Integer = 0

Console.WriteLine("\*\* TABLAS DE MULTIPLICAR \*\*")

Console.ReadLine()

For i As Integer = 0 To 10

Console.WriteLine()

Console.WriteLine("Tabla del {0}", i)

Console.WriteLine("-----------")

Console.WriteLine()

For j As Integer = 0 To 10

res = i \* j

Console.WriteLine("{0} \* {1} = {2}", i, j, res)

Next

Next

Console.Read()

End Sub

End Module

While… End While: Consiste en repetir una o un grupo de acciones mientras la condición sea verdadera. Para salir de un While se hace de dos formas, cumpliendo la condición del While o poniendo Exit While.

Do… Loop (Loop = Bucle): Es una estructura repetitiva que tiene cuatro posibles sintaxis:

1. Do While condición… Loop: Realiza una o un grupo de acciones mientras la condición sea cierta, es equivalente al While… End While.
2. Do… Loop While condición: Realiza una o un grupo de acciones mientras la condición sea cierta, con la diferencia de la anterior de que esta realiza la acción al menos una vez. Se usa cuando se necesite (al menos se ejecute una vez el código).
3. Do Until condición… Loop: Realiza una o un bucle de acciones hasta que la condición sea cierta.
4. Do… Loop Until condición: Realiza una o un grupo de acciones hasta que la condición sea cierta, se diferencia de la anterior en que al menos las acciones se ejecutan una vez.

Do… Loop tiene Exit Loop como el While para salir.

Ejer7:

Realizar una aplicación que me permita introducir números por teclado y visualice cuantos números se han introducido, la suma total de ellos, su media aritmética y cuál es el mayor. La condición de salida será pulsar lo que el programador estime necesario.

Module Module1

Sub Main()

Dim med As Integer

Dim may As Integer = 0

Dim sum As Integer = 0

Dim cont As Integer = 0

Dim cnd As Integer

Console.WriteLine("Introduzca números por teclado, para hayar la suma total, media aritmética y el mayor de ellos.")

Do

Console.Write("Introduzca un número (Para salir pulse cero [0]): ")

cnd = Console.ReadLine()

If cnd > may Then

may = cnd

End If

sum = sum + cnd

cont = cont + 1

Loop Until cnd = 0

med = sum / cont

Console.WriteLine("La suma total es {0}, la media es {1} y el número mayor es {2}", sum, med, may)

Console.Read()

End Sub

End Module

EXCEPCIONES:

Try… Catch… Finally… End Try: Se utiliza para captura Exceptions (Excepciones) o controlar posibles errores. Sintaxis:

Try

Acciones a controlar

Catch [Excepción]

Acción en caso de error 1

Catch [Excepción]

Acción en caso de error 2

…

[Finally]

Acción que se ejecuta SIEMPRE haya o no error.

End Try

Ejer8:

Con el ejercicio anterior haz un try…catch para controlar posibles excepciones:

Module Module1

Sub Main()

Dim med As Integer

Dim may As Integer = 0

Dim sum As Integer = 0

Dim cont As Integer = 0

Dim cnd As Integer

Console.WriteLine()

Console.WriteLine("Introduzca números por teclado, para hayar la suma total, media aritmética y el mayor de ellos.")

Do

Try

Console.Write("Introduzca un número (Para salir pulse cero [0]): ")

cnd = Console.ReadLine()

If cnd > may Then

may = cnd

End If

sum = sum + cnd

cont = cont + 1

med = sum / cont

Catch

Console.WriteLine("Se ha producido un error o excepción.")

Main()

End Try

Loop Until cnd = 0

Console.WriteLine("La suma total es {0}, la media es {1} y el mayor es {2}", sum, med, may)

Console.Read()

End Sub

End Module

ESTRUCTURAS DE DATOS:

En muchas ocasiones es necesario asignar mas de un valor a una variable, por lo que es necesario utilizar alguna Estructura interna que permita guardar esos datos y poder gestionarlos. Existen dos estructuras internas:

* Array: Es una estructura interna que permite guardar datos del mismo tipo en una única variable. Para hacer referencia a esos datos se utilizan uno o varios índices según la dimensión. .Net acepta hasta 32 dimensiones en un solo Array. Para definir un Array en .Net se utiliza la siguiente sintaxis:
  + Vectores o tablas unidimensionales:

Dim nomTab (longitud) As tipo

Empieza en 0 y tiene una posición mas de lo que se declara en la longitud.

Para introducir valores seria:

nomTab (posición) = valor

Ejemplo: Dim Dinero (7) As Integer

Existe la posibilidad de inicializar un Array en el momento de declararlo, siendo así:

Dim Tab () As Integer = {listaValores}

Ejemple: Dim Notas () As Integer = {5, 4, 8, 7, 6}

La dimensión es el número de elementos que tenga la listaValores, si se quieren dar más posiciones de los que se necesitan se ponen ceros.

Ejer9 (para el lunes):

Realizar un programa que permita introducir las calificaciones de cinco participantes en un concurso de cocina de tal manera que deberá aparecer cual es la puntuación más alta y a que numero de participante pertenece, la puntuación más baja y a que numero de participante pertenece y la nota media.

Deberá existir una opción de salir preguntando si se desea salir o no. Utilizando try…catch.

* + Matrices o Arrays bidireccionales: Tienen dos índices, que se haría de la siguiente forma:

Dim Tab (7, 3) As Integer

El primer número son las filas y el segundo número son las columnas.

No se pueden inicializar directamente.

* + Poliedros o Arrays Tridimensionales.
  + Arrays Paralelos: En muchas ocasiones, tenemos la necesidad de utilizar Arrays con diferentes tipos de datos (numéricos y de cadena), esto es imposible en una matriz, por lo que es aconsejable utilizar el concepto de Array Paralelo que consiste en utilizar varios vectores, gestionados a la par, es decir, gestionando el mismo índice de todos los Arrays.
* Estructuras (Structure): Es un Array especial en el sentido de que permite diferente tipos de datos a la hora de almacenarlos, a diferencia de los Arrays que solo permiten datos con los que se declara.

Una estructura se define con Structure… End Structure.

Una estructura está formada por dos partes:

* + Definir la propia estructura que se realizara justo antes del sub main(), fuera del sub Main(). Sintaxis:

Structure nombre

Dim…

Dim…

…

End Structure

Sub Main ()

End sub

* + Definir una variable especial del tipo nombre de la Estructura con la que voy a poder acceder a todas las variables, para poder leer y/o escribir en ellos. Si se define fuera del main, sirve para todo el proyecto y es: Public nomVar As nomStruct.

Cuando se declara en el main, seria local, siendo: Dim nomVar As nomStruct.

Para utilizarla seria: nomVar.nomVarStruc = “ “

Ejemplo Ejer10Estr:

Module Module1

Structure Libro

Dim titulo As String

Dim Autor As String

Dim Editorial As String

Dim Año As Integer

End Structure

Sub Main()

Dim libr As Libro

libr.titulo = "El Quijote"

libr.Autor = "Cervantes"

libr.Editorial = "Santillana"

libr.Año = 1580

Console.WriteLine("Los datos son: {0}, {1}, {2}, {3}", libr.titulo, libr.Autor, libr.Editorial, libr.Año)

Console.ReadLine()

End Sub

End Module

Para poder guardar más de un dato será necesario dimensionar la variable estructura de la siguiente forma:

Dim nomVar(dimensión) As nomEstr

Para referirse a un registro seria:

nomVar(0).nomVarEstr = “ “

Ejer11:

Hacer una agenda, con nombre, teléfono, dirección y nivel de amistad, con tres entradas y visualizar.

Module Module1

Structure Agenda

Dim nombre As String

Dim telef As Integer

Dim amis As Integer

End Structure

Sub Main()

Dim agn(2) As Agenda

Dim i As Integer

Try

Console.WriteLine("\*APLICACIÓN")

Console.WriteLine()

'introduccion de valores en la estructura

For i = 0 To 2

Console.WriteLine("Nombre del amigo {0}:", i + 1)

agn(i).nombre = Console.ReadLine()

Console.WriteLine("Telefono del amigo {0}:", i + 1)

agn(i).telef = Console.ReadLine()

Console.WriteLine("Nivel de amistad del amigo {0}:", i + 1)

agn(i).amis = Console.ReadLine()

Next

'visualizacion de los valores de la estructura

For i = 0 To 2

Console.WriteLine("Amigo {0}-> Nombre: {1}, Telefono: {2}, Amistad: {3}", i + 1, agn(i).nombre, agn(i).telef, agn(i).amis)

Next

Catch

Console.WriteLine("Se ha producido un error.")

Main()

End Try

Console.Read()

End Sub

End Module

Para utilizar varias estructuras seria (ejemplo):

Structure Fecha

Dim dia as Integer

End Structure

Structure Libros

Dim aut as String

Dim fecha\_publicacion as Fecha

End Structure

\* Fecha tiene que estar declarada obligatoriamente antes que libros.

Para utilizarla en el main seria:

var(i).fecha\_publicacion.dia = 23

Esto se puede hacer las veces que queramos.

SUBPROGRAMAS, PROCEDIMIENTOS Y FUNCIONES:

Un subprograma o procedimiento, representado por sub, es una forma de organizar código por bloques que tiene la característica de que no puede devolver ningún valor tras los cálculos.

Para poder llamar a un procedimiento basta con poner nombreProc (), cuando acaba el procedimiento, el procedimiento que llama sigue con la línea siguiente. El que llama al procedimiento es el nombrado llamador y el otro se denomina llamado.

Ejer12:

Realizar un programa, que calcule el área de un triangulo, utilizando un procedimiento denominado triangulo, y terminando con un saludo final diseñado en otro procedimiento.

Module Module1

Sub Triangulo()

Dim base, altura, result As Double

Console.WriteLine()

Console.WriteLine("Introducir la base del triángulo: ")

base = Console.ReadLine()

Console.WriteLine()

Console.WriteLine("Introducir la altura del triangulo: ")

altura = Console.ReadLine()

Console.WriteLine()

result = base \* altura / 2

Console.WriteLine("El área del triángulo es {0}", Math.Round(result, 2))

Console.WriteLine()

End Sub

Sub Saludo()

Console.WriteLine("Ejecución finalizada.")

Console.WriteLine("Gracias por utilizar la aplicación.")

End Sub

Sub Main()

Console.WriteLine("\*APLICAICÓN\*")

Console.WriteLine("Inicio de la aplicación.")

Triangulo()

Console.WriteLine()

Saludo()

Console.Read()

End Sub

End Module

Funciones (Function): Es un procedimiento o subprograma especial, que sirve para estructurar codigo. Tiene de especial, que la función (el propio nombre de la función), actua como una variable, es decir, una función devuelce un valor. Sintaxis:

Function nom\_funcion (parametros separados por comas si hay mas de uno) As tipoDatoDev

\_\_\_\_

\_\_\_\_

\_\_\_\_

End function

Para poder devolver un valor basta con poner el nombre de la funcion o opcionalmente return nom\_funcion, ya que nom\_funcion se puede utilizar como si fuese una variable.

Un parametro es un valor, que se puede transmitir entre diferentes procedimientos o funciones. Los datos que se transmiten se denominan parametros.

Los parametros actuales son los que se definen en el procedimiento llamado y los parametros formales son los que se declaran en el llamador.

Los parametros actuales, pueden ser de dos tipos, dependiendo de la posibilidad de que los valores originales, es decir, los parametros formales, puedan ser modificacdos en el procedimiento llamador. Existen por tanto dos tipos:

* Paso por valor: Se expresa con ByVal, que hace que los parametros originales no cambien en cuanto a su valor, es el valor por defecto.
* Paso por referencia: Permite modificar el valor formal del parametro, y se expresa con Byref.

Ejer13:

Module Module1

Sub cua(ByVal | Byref par As Integer)

par = 8

Console.WriteLine("Valor inicial " & par)

End Sub

Sub Main()

Dim a As Integer = 55

Console.WriteLine("Valor antes de la llamada {0}", a)

cua(a)

Console.WriteLine("Valor después de la llamada {0}", a)

Console.ReadLine()

End Sub

End Module

Ejer15:

Realizar un ejercicio, utilizando un menu, que me permita calcular el area del triangulo, cuadrado, circunferencia y salida, utilizando unicamente procedimientos o funciones con parametros.

DISEÑO DE CLASES:

Para definir una clase en .Net se necesita:

1. Creación de la clase: Para ello se utiliza la sintaxis class nomClase, en su interior se pone el conjunto de atributos y métodos, terminando con End Class.
2. Es necesario generar una variable especial, que cuando se define, se crea una instancia que permite, a partir de ese momento poder utilizar esta clase. Esta instancia utiliza la palabra reservada new.

Existen tres formatos para realizar o definir una instancia:

* 1. Dim nomVar As nomClase

nomVar = new nomClase ()

* 1. Dim nomVar As nomClase = new nomClase ()
  2. Dim nomVar As new nomClase ()

1. Una vez definidos los métodos, en tercer lugar se podrá acceder a los elementos de la clase, bastara con poner la variable de la instancia seguida con un punto (igual que structure).

Ejercicio Class:

Realizar un programa que introduciendo por teclado los datos de una agenda (nom, apel, tel, dir), me permita visualizarlos utilizando una clase:

Module Module1

Class Agenda

Public nom, apel, dir As String

Public num As Integer

Sub introducirDatos()

Try

Console.WriteLine("Nombre del contacto: ")

nom = Console.ReadLine()

Console.WriteLine()

Console.WriteLine("Apellido del contacto: ")

apel = Console.ReadLine()

Console.WriteLine()

Console.WriteLine("Telefono del contacto: ")

num = Console.ReadLine()

Console.WriteLine()

Console.WriteLine("Direccion del contacto: ")

dir = Console.ReadLine()

Console.WriteLine()

Console.Read()

Catch

Console.WriteLine("Se ha producido un error.")

introducirDatos()

End Try

End Sub

Sub mostrar()

Console.Clear()

Console.WriteLine("Nombre: {0}", nom)

Console.WriteLine("Apellido: {0}", apel)

Console.WriteLine("Teléfono: {0}", num)

Console.WriteLine("Dirección: {0}", dir)

Console.Read()

End Sub

End Class

Sub Main()

Dim var As Agenda = New Agenda

var.introducirDatos()

var.mostrar()

Console.ReadLine()

End Sub

End Module

Ejercicio ClassCalculos:

Realizar un programa que utilizando una clase que incorporara un menú con una serie de opciones, como son calcular el area del triangulo, cuadrado, rombo y circunferencia, cuyos cálculos deberán ser programados en otra clase.

Module Module1

Public opc As Integer

Class Menu

Sub menu()

Console.WriteLine()

Console.WriteLine("\*MENU\*")

Console.WriteLine()

Console.WriteLine("1. Triángulo")

Console.WriteLine("2. Cuadrado")

Console.WriteLine("3. Rombo")

Console.WriteLine("4. Cincuferencia")

Console.WriteLine("5. Salir")

Console.WriteLine()

End Sub

End Class

Class Calculos

Public area, calc As Double

Sub triangulo(ByVal \_base As Double, ByVal \_altura As Double)

area = \_base \* \_altura / 2

Console.WriteLine("El área del triángulo es: {0}", Math.Round(area, 2))

Console.ReadLine()

End Sub

Sub cuadrado(ByVal \_lado As Double)

calc = \_lado \* \_lado

Console.WriteLine("El área del cuadrado es: {0}", Math.Round(calc, 2))

Console.ReadLine()

End Sub

Sub rombo(ByVal \_diMay As Double, ByVal \_diMen As Double)

calc = \_diMay \* \_diMen / 2

Console.WriteLine("El área del rombo es: {0}", Math.Round(calc, 2))

Console.ReadLine()

End Sub

Sub circunferencia(ByVal \_radio As Double)

calc = Math.PI \* Math.Pow(\_radio, 2)

Console.WriteLine("El área de la circunferencia es: {0}", Math.Round(calc, 2))

Console.ReadLine()

End Sub

End Class

Sub Main()

Dim menu As New Menu()

Dim calc As New Calculos()

Dim base, altura, lado, radio, diMay, diMen As Integer

Dim opcsal As Char

Try

menu.menu()

Console.WriteLine("¿Qué opción desea?")

opc = Console.ReadLine()

Select Case opc

Case 1

Console.Clear()

Console.WriteLine("La opción es hayar el área del triángulo.")

Console.WriteLine("Introduzca la base:")

base = Console.ReadLine()

Console.WriteLine("Introduzca la altura: ")

altura = Console.ReadLine()

calc.triangulo(base, altura)

Main()

Case 2

Console.Clear()

Console.WriteLine("La opción es hayar el área del cuadrado.")

Console.WriteLine("Introduzca el lado: ")

lado = Console.ReadLine()

calc.cuadrado(lado)

Main()

Case 3

Console.WriteLine("La opción es hayar el área del rombo.")

Console.WriteLine("Introduzca la diagonal mayor:")

diMay = Console.ReadLine()

Console.WriteLine("Introduzca la diagonal menor: ")

diMen = Console.ReadLine()

calc.rombo(diMay, diMen)

Main()

Case 4

Console.Clear()

Console.WriteLine("La opción es hayar el área de la circunferencia.")

Console.WriteLine("Introduzca el radio: ")

radio = Console.ReadLine()

calc.circunferencia(radio)

Main()

Case 5

Console.WriteLine("¿Desea salir? Si - s / No - n")

opcsal = Console.ReadLine()

Select Case opcsal

Case "s", "S"

Console.WriteLine("Saliendo...")

Console.Clear()

Console.WriteLine("Gracias por utilizar el programa.")

Case "n", "N"

Console.WriteLine("Colviendo a ejecutar...")

Main()

Case Else

Console.WriteLine("La opción no es valida")

End Select

Case Else

Console.WriteLine("La opción no es valida.")

Main()

End Select

Catch

Console.WriteLine("Se ha producido un error.")

Main()

End Try

Console.ReadLine()

End Sub

End Module

AGREGAR UNA CLASE COMO FICHERO EN UN PROYECTO:

Consiste en seguir los siguientes pasos:

1. Ir a nombre del Proyecto, botón derecho agregar, y luego en clase.
2. Aparecerá la opción del nombre del fichero, con extensión .vb (fichero de código)
3. Se abrirá un IDE, donde se escribirá la o las clases
4. Una vez escrito se guardara.

Este fichero aparecerá en el explorador de soluciones y podrá ser usado (llamar a todas las clases y métodos), de forma convencional (como siempre), en el programa principal.