Departamento de Ingeniería de Sistemas y Automática

PLAN DE PRÁCTICAS

Fundamentos de Informática

Curso 2009 / 2010

volumen 1

Escola Técnica Superior de Enxeñeiros Industriais de Vigo

Tabla de contenidos

Práctica 1. Introducción a Matlab

1.1 Introducción.

Matlab es un entorno de trabajo matemático y de simulación que es usado para modelar y analizar sistemas dinámicos. La forma de trabajar con Matlab es a través de un conjunto de comandos a los que generalmente se les pasan variables (es decir, operan con variables) y eventualmente devuelven otras variables como resultado.

El elemento básico de Matlab es una matriz compleja de doble precisión. Esta representación es bastante general e incluye vectores reales y complejos y escalares. Además, incluye también indirectamente polinomios.

1.2 Ejercicios.

1.2.1 Ejercicio 1.

Repetir los comandos que se listan a continuación y observar los resultados. Hacer uso del comando help cuando sea necesario. (help nombre_comando)

** Trabajo con matrices:*

```
>> A=[1  2   3;   4   5   6;   7   8   9]  ó
>> A=[1   2  3
      4   5   6
      7...8...9]
>> D=A(:,2)     Da un vector que es el segundo elemento de todas las
                filas y la 2ª columna
>> D=A(2:3,2:3) Da una submatriz: fila 2 a 3 y columna 2 a 3
>> B=A'         Traspuesta
>> C=A*B        Producto matricial
>> det(A)       Determinante
>> rank(A)      Rango
>> inv(A)       Inversa
>> p=poly(A)         Polinomio característico
```

Si a una matriz le añadimos un número en una posición no existente, la matriz crecerá hasta encajar el nuevo elemento y llenará con ceros los espacios no especificados.

```
>> a=[1   2;3  3]
>> a(3,3) = 1
>> a
```

**Matrices especiales*

```
>> eye(3)                Identidad 3x3
>> eye(3,2)              Identidad 3x2
>> ones(3)               Llena de unos tamaño 3x3
>> ones(3,2)             Llena de unos tamaño 3x2
```

```
>> zeros(3)                Llena de ceros tamaño 3x3
>> diag([1   2   3   4])   Diagonal con 1, 2, 3 y 4 en la diagonal
```

**Trabajo con vectores*

```
>> roots(p)
>> c=[1   2   3]
>> d=[4   5   6]
>> d=d'
>> c*d
>> d*c
>> x=1 : 5           Vector con elementos de 1 a 5 con incrementos de 1
>> x=0 : 0.5: 5    Vector con elementos de 0 a 5 con incrementos de 0.5
>> x=5 : -1: 0       Vector con elementos de 5 a 0 con decrementos de 1
```

**Trabajo elemento a elemento*

```
>> x=1 : 3
>> y= x
>> z=x.*y
```

** Trabajo con gráficas:*

```
>> x=[0.   .48   .84   1   .91   .6   .14]
>> plot(x)      %Gráfica en 2D
>> x=[1   2   3   4   5   6   7 ]
>> y=[5.3   6.2   3   7.8   12.3   10.4   7.8]
>> plot(x,y)
>> title('Mi grafico')
>> xlabel('x')
>> ylabel('y')
>> grid
```

1.2.2 Ejercicio 2

Cando se quiere ejecutar una secuencia de instrucciones hay dos opciones: Ejecutar instrucción a instrucción como hemos hecho en los ejercicios anteriores o hacer un fichero .m, como vamos a hacer en este ejercicio

1- Crear un nuevo fichero ".M": FILE->NEW

2- Teclear en la nueva ventana lo siguiente:

```
x=[1   2   3   4   5   6   7   8];
plot(x);
hold on;
y=[0   1.5   2   6   4   3   4   5];
plot(y);
title('Mi grafico');
xlabel('x');
ylabel('y');
grid;
hold off
```

3- Salvar el nuevo fichero con el nombre "nombre1.m"

4- Ejecutar el programa desde la línea de comandos de Matlab tecleando su nombre.

1.2.3 Ejercicio 3

1- Crear un nuevo fichero ".M": FILE->NEW
2- Teclear en la nueva ventana lo siguiente:

```
Z=[1   1     1      1      1   1
   1   1     1.5    1      1   1
   1   1.5   2      1.5    1   1
   1   1     1.5    1      1   1
   1   1     1      1      1   1
   0   0     0      0      0   0]

mesh(Z)
```

3- Salvar el nuevo fichero con el nombre "nombre2.m"
4- Ejecutar el programa desde la línea de comandos de Matlab tecleando su nombre.

1.2.4 Ejercicio 4

Escribir en un fichero M un programa que haga lo siguiente: Crear una matriz de 5 x 5 en la que se almacenarán los valores de medios de las alturas de la población de un país en cinco edades diferentes y con medidas hechas en cinco años diferentes, como por ejemplo:

	5 años	10 años	15 años	20 años	25 años
1990	1.30	1.45	1.69	1.76	1.7601
1991	1.301	1.4501	1.6901	1.766	1.766
1992	1.303	1.4499	1.70	1.765	1.766
1993	1.30	1.45	1.701	1.76	1.77
1994	1.31	1.4501	1.7001	1.77	1.7701

Calcular las medias de las alturas en cinco años. Representar en 2D los valores de las alturas medias para las cinco edades, (curva de crecimiento media).

1.3 APÉNDICE. Nociones básicas sobre Matlab

MATLAB (MATrix LABoratory) es un sistema basado en el cálculo matricial para desarrollar aplicaciones matemáticas y de ingeniería. Podemos pensar en MATLAB como un clase de lenguaje diseñado únicamente para realizar manipulaciones matriciales. Todas las variables que se manejen en MATLAB son matrices. Esto es, MATLAB tiene solo un tipo de datos, una matriz o un array rectangular de números. MATLAB posee un amplio conjunto de rutinas para obtener salidas gráficas.
Esta breve guía presenta una introducción a MATLAB. MATLAB posee una ayuda en línea a la que puede llamarse siempre que se desee. La orden he1p visualizará una lista de funciones y operadores predefinidos para los que hay disponible una ayuda en línea. La orden.
he1p 'nombre de función'
dará información sobre la función específica llamada de su finalidad y forma de uso. La orden
he1p he1p
dará información de como utilizar la ayuda en línea.
No se tratan en esta guía muchas características importantes y útiles. Para conocer estas particularidades el lector debería consultar la Edición de Estudiante de MATLAB y la Guía de Usuario.

1- Estructura general de los comandos Matlab

Matlab es un lenguaje de expresiones. Las expresiones son introducidas por el usuario e interpretadas y evaluadas por el sistema Matlab. La sintaxis general de los comandos de MATLAB es de la forma:

[output 1, output2, ...] = nombre_commando (input1, input2, ...)

donde las variables de salida constan entre corchetes y las de entrada entre paréntesis. Si hay una sola salida, los corchetes son opcionales.
La evaluación de las expresiones produce siempre una matriz. Por lo tanto, MATLAB trabaja con matrices.

NOTA:
Se puede hacer aquí una primera introducción a los convenios de las notaciones en informática:
[ ] Optativo
( ) Paso de parámetros (como en una función matemática y=f(x)=x+5

1.3.1 Estructura de datos en Matlab

El elemento básico de Matlab es una matriz compleja de doble precisión. Esta representación es bastante general e incluye vectores reales y complejos y escalares. Incluye también indirectamente polinomios. Los vectores fila se introducen usando corchetes y los elementos se separan usando espacios en blanco o comas. Para crear un vector columna se pueda usar la transformación:

```
>>x   =     [1,2, 3] , y=[1+j,2+pi*i, -sqrt(-1) ]

x=    1     2     3
y=    1.0000+1.0000i        2.0000+3.0000y   0-1.0000i
```

Las matrices se introducen línea a línea, separadas por punto y coma (o también con el retorno de carro).

```
A = [1   2   3
     4   4   6
     7   8   9 ]
```

1.3.2 Operaciones y funciones

Operaciones algebraicas básicas:

OPERACIÓN	***DESCRIPCIÓN***
+	Suma
-	Resta
*	Multiplicación
^	Potencia
‘	Transpuesta conjugada
sqrt	Raíz cuadrada
/	División

Operaciones relacionales básicas. El resultado es un valor lógico (Bolean):

OPERACIÓN	***DESCRIPCIÓN***
<	Menor que
<=	Menor que o igual a
>	Mayor que
>=	Mayor que o igual a
==	Igual
~=	No igual

Operaciones lógicas básicas:

OPERACIÓN	***DESCRIPCIÓN***
&	AND
|	OR
~	NOT

Operaciones trigonométricas básicas:

sin, cos, tang, .. ver apartado “Órdenes y funciones matriciales”.

Operaciones Logarítmicas básicas

log, log 10, 10log, ... ver apartado “Órdenes y funciones matriciales”.

Operaciones sobre Matrices

A/B inv(A)*B...devuelve x como resultado de Ax=B
B\A B*inv(A)...devuelve x como resultado de xA=B

Operaciones sobre arrays (matrices elemento a elemento)

+, -, .x, ./, .\, . ^ ej: C=A.*B multiplica elemento a elemento 2 matrices
c11=a11*b11
c12=a12*b12
.....

Caracteres especiales:

CARÁCTER	***DESCRIPCIÓN***
[ ]	Utilizado para formar vectores y matrices
( )	Precedencia de expresión aritmética
,	Separa elementos y argumentos de función
;	Final de filas, suprime la impresión (véanse los detalles que se dan a continuación)
:	Generación de vectores (véanse los detalles que se dan a continuación)
!	Ejecución de orden del sistema operativo
%	Comentarios (véanse los detalles que se dan a continuación)

Comandos generales.

```
>> cd           Muestra cual  es el directorio de trabajo
>> cd \subdir   Cambia el directorio de trabajo
>> dir          Listado del contenido del directorio de trabajo
>> pwd          Para saber en que directorio nos encontramos
>> who          Lista todas las variables del espacio de trabajo
>> whos         Igual que el anterior pero da más información
>> clear        Borra el espacio de trabajo
>> save n       Guarda el espacio de trabajo en fichero de nombre n.mat
>> load n       Carga el espacio de trabajo del fichero n.mat
>> help         Llama a la ayuda de Matlab. Lista todos los comandos
>> help n       Proporciona ayuda de un comando n.
>> clg          Borra la pantalla gráfica
>> quit         Salir de Matlab
```

1.3.3 Órdenes y funciones matriciales

ORDENES	***DESCRIPCIÓN***
Abs	Valor absoluto, magnitud compleja
angle	Ángulo de fase
ans	Respuesta cuando no se asigna expresión
atan	Arco tangente
axis	Escalado manual de ejes
bode	Representación en el diagrama de Bode
clear	Borra el espacio de trabajo
clg	Borra la pantalla gráfica
computer	Tipo de computador
conj	Complejo conjugado
conv	Convolución, multiplicación
corrcoef	Coeficientes de correlación,
cos	Coseno
cosh	Coseno hiperbólico
cov	Covarianza
deconv	Deconvolución, división,
det	Determinante
diag	Matriz diagonal
eig	Valores propios y vectores propios
exit	Finalizar programa
exp	Exponencial base e
expin	Matriz exponencial
eye	Matriz identidad
filter	Implementación de filtro directo
format long	Punto fijo escalado a 15 dígitos (Ejemplo. 1.33333333333333)
format long e	Punto flotante a 15 dígitos (Ejemplo: 1.33333333333333e+000)
format short	Punto fijo escalado a 5 dígitos (Ejemplo: 1.3333)
format short e	Punto flotante a 5 dígitos (Ejemplo: 1 ¡3333e+000)
freqs	Respuesta en frecuencia de la transformada de Laplace
freqz	Respuesta en frecuencia de la transformada-z
grid	Dibujar rejilla
Hold	Mantener la actual pantalla
i	√-1
inmag	Parte imaginaria
inf	Infinito
inv	Inversa
j	√-1
log	Logaritmo natural
loglog	Gráfica x-y loglog
logm	Logaritmo matricial
logspace	Vectores espaciados logarítmicamente
log10	Logaritmo en base 10
lqe	Diseño del estimador lineal cuadrático

lqr	Diseño del regulador lineal cuadrático
max mean median min	Valor máximo Valor medio Mediana Valor mínimo
NaN nyquist	No es un número Respuesta en frecuencia en el diagrama de Nyquist
ones	Constante
pi plot polar poly polyfit polyval polyvalin prod	Π Gráfica x-y lineal Gráfica polar Polinomio característico Ajuste de curva polinomial Evaluación polinomial Evaluación polinomial matricial Producto de elementos
quit	Finalizar el programa
rand rank real rem residue rlocus roots	Generación de números aleat6rios y matrices Calcula el rango de una matriz Parte real Resto o módulo Expansión en fracciones parciales Diagrama de lugar de las raíces Raíces de un polinomio
semilogx semilogy sign sin sinh size sqrt sqrtin std step sum	Diagrama semilogarítmico x-y (eje-x logarítmico) Diagrama semilogarítmico x-y (eje-y logarítmico) Función signo Seno Seno hiperbólico Dimensión de una matriz Raíz cuadrada Raíz cuadrada matricial Desviación estándar Respuesta a un salto unitario Suma de elementos
tan tanli text tide trace	Tangente Tangente hiperbólica Posicionado arbitrario de texto Título de una gráfica Traza de una matriz
who	Lista de todas las variables actualmente en memoria
xlabel	Etiqueta en el eje x
ylabel	Etiqueta en el eje y
zeros	Cero

Práctica 2. Introducción a Visual Basic

2.1 Objetivo de la práctica

El objetivo de esta práctica es familiarizarse con el entorno de Visual Basic y realizar un programa sencillo que ilustre el funcionamiento de dicho entorno.

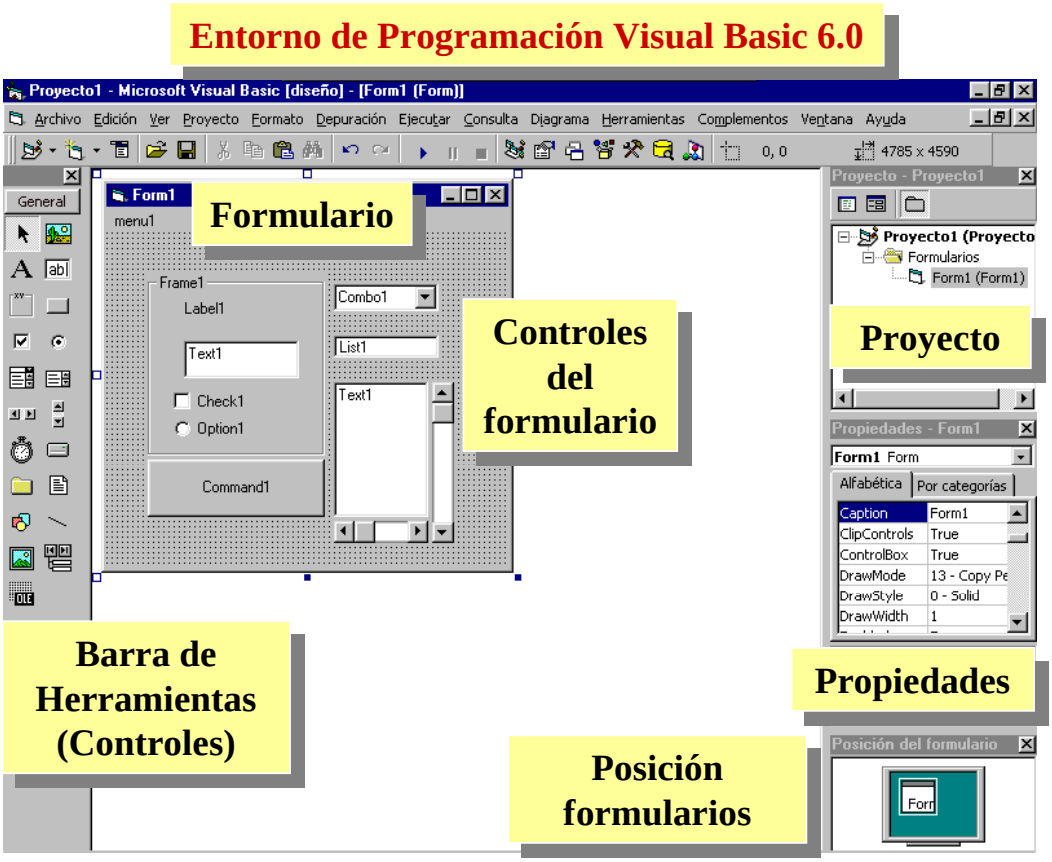
2.2 Ejercicios

2.2.1 Partes fundamentales del entorno

Las partes fundamentales del entorno son:

- Cuadro o barra de herramientas.
- Explorador de proyectos.
- Ventana de propiedades.
- Ventana de posición del formulario.
- Ventana de proyecto.

En la figura se muestra cada una de estas partes.



2.2.2 Manejo de módulos

Los módulos principales son los Formularios (.frm) y los de código y definiciones (.bas)

2.2.3 Cómo empezar a escribir nuestro programa

Menú de inicio de Windows → Microsoft Visual Studio → Visual Basic

Seleccionar el tipo de aplicación “EXE estándar” y Aceptar

Menú “Proyecto” → “Quitar Form1”

Menú “Proyecto” → “Agregar Módulo”

Escribir el código de nuestro programa el en módulo recién abierto

2.2.4 Como guardar nuestro trabajo

Visual Basic guarda un fichero por cada módulo con las extensiones indicadas anteriormente. Además guarda un fichero de proyecto con la extensión "vbp" que contiene la información de todos los módulos que componen el proyecto y de las propiedades de este.

Es importante guardar todos los módulos en el mismo directorio y prestar atención de donde propone Visual Basic para guardar.

2.3 Ejercicio 1

A continuación se muestra un programa en Visual Basic que solicita dos números y muestra posteriormente la resta e indica cual es mayor.

```
Option Explicit
Sub main()
        Dim numero1 As Integer, numero2 As Integer
        Dim resta As Integer
        numero1 = InputBox("Introduce número")
        numero2 = InputBox("Introduce número")
        resta = numero1 - numero2
        MsgBox "La resta es " & resta
        If numero1 > numero2 Then
           MsgBox "El número mayor es: " & numero1
        Else
           If numero2 > numero1 Then
              MsgBox "El número mayor es: " & numero2
           Else
              MsgBox "Los números son iguales"
           End If
        End If
End Sub
```

2.4 Ejercicio 2

Modificar el programa anterior para que se soliciten tres números y se realice y se muestre los siguiente:

- La resta entre el primero y el tercero.
- La suma de los tres.
- Mire e indique el menor del primero y el tercero.

Práctica 3. Constantes, variables y operadores

3.1 Objetivo de la práctica

Los objetivos que se persiguen con esta práctica están relacionados con los conceptos tratados en el tema 3 de teoría utilizando, para poder darle contenido, algunos operadores básicos que se tratan en el tema 4.

En concreto los objetivos son los siguientes:

- Definición de constantes y variables.
- Conocimiento y uso de los tipos de datos.

3.2 Exposición

Como resumen a lo tratado en las clases de teoría y con el fin de enfocar los temas que se abordan en esta práctica se exponen aquí los siguientes puntos:

3.2.1 Definición de constantes

[Public|Private] Const nombre_de_la_constante [As tipo] = expresión

3.2.2 Definición de variables

Dim nombre_variable [As tipo]

3.2.3 Tipos de datos

TIPO	DESCRIPCION	RANGO
Integer	Entero (2 bytes)	-32768 a 32767
Long	Entero largo (4 bytes)	-2147483648 a 2147483647
Single	Coma flotante (4 bytes)	-3.40E+38 a 3.40E+38
Double	Coma flotante (8 bytes)	-1.79D+308 a 1.79D+308
String	Cadena de caracteres	Hasta 64Kbytes aprox.
Byte	Carácter (1 byte)	0 a 255
Boolean	Boolean (2 bytes)	**True** o **False**
Date	Fecha y Hora (8 bytes)	1/1/100 a 31/12/9999

3.3 Ejercicio

3.3.1 Enteros y reales

Crear un programa en Visual Basic que declare tres variables enteras (**e1**, **e2** y **e3**) y otras tres reales (**r1**, **r2** y **r3**). El programa ha de hacer lo siguiente:

- Asignar a **r1** un valor solicitado al usuario.
- Asignar a **e1** el valor de **r1.**
- Asignar a **r2** un valor solicitado al usuario.
- Asignar a **e2** el valor de **r2**.
- Asignar a **e3** el resultado de dividir **e1** por **e2**.
- Asignar a **r3** el resultado de dividir **r1** por **r2**.
- Mostrar por pantalla ambos resultados (**e3** y **r3**).
- Sabiendo que **e3** y **r3** son el resultado de la división de los mismos números en ambos casos, razonar por qué son distintos.

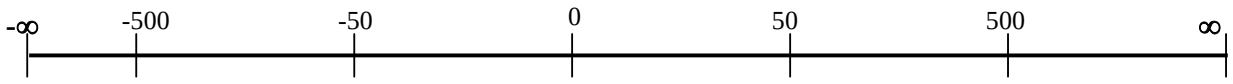
3.3.2 Variables y operadores I

Crear un programa en Visual Basic que contenga dos variables: ***numero*** como entero y ***mensaje*** como una cadena de caracteres. El programa ha de hacer lo siguiente:

- Sacará un cuadro de entrada de datos solicitando un número (usar la función ***InputBox***) y almacenar dicho número en la variable ***numero***.
- A continuación comprobará si ***numero*** es par o impar. En la variable ***mensaje*** almacenará un texto indicando si el número introducido es par o impar, junto con el valor del número. Finalmente mostrará el mensaje por pantalla. Usar los operadores *Mod* y *&*.

3.3.3 Variables y operadores II

Solicitar al usuario un valor **real** y sacar un mensaje indicando en cuál de los intervalos abajo mostrados se encuentra. Cuando se trate de un valor común a dos intervalos se considerará que pertenece al intervalo de la derecha: Si consideramos el valor 50, por ejemplo, se indicará que pertenece al intervalo [50, 500).



Práctica 4. Operadores y expresiones

4.1 Objetivo de la práctica

Los objetivos que se persiguen con esta práctica están relacionados con los conceptos tratados en el tema 3 y 4 de teoría.

En concreto los objetivos son los siguientes:

- Empleo de operadores y expresiones.
- Comienzo con bucles.

4.2 Ejercicio

4.2.1 Datos a tener en cuenta antes de empezar.

Antes de comenzar esta práctica el usuario debe consultar en la ayuda información sobre las funciones *Rnd*, *Int* y *Randomize*. Realizar toda la práctica en un único módulo (*.bas).

Cuando el programa se quede bloqueado en un bucle while, se puede interrumpir la ejecución con la combinación de teclas Control + Pausa.

4.2.2 Realización de la práctica.

En esta práctica se programará un juego que consiste en que el usuario debe adivinar un número que genera aleatoriamente el ordenador con un número de intentos limitado.

Primera versión.

Realizar un programa en Visual Basic que haga lo siguiente:

1.- Solicitar al usuario que introduzca un número N.
2.- Generar un número aleatorio entre 0 y N.
3.- Invitar al usuario a adivinar el número leyendo cada uno de los intentos y visualizar un mensaje según el número sea mayor, menor o igual.
4.- El programa finalizará cuando se adivine dicho número o se llegue al máximo de intentos (constante).

Segunda versión. Uso contadores y acumuladores

Mostrar por pantalla de intentos que han sido necesarios para adivinar el número y la suma de los datos que el usuario ha introducido.

Tercera versión. Uso de la sentencia *select*

Sacar distintos mensajes según el número de intentos necesarios para acertar el número. El mensaje se dará en función del porcentaje de intentos empleados por el usuario con respecto al número máximo de intentos disponibles (usar la sentencia "***select"***)

Práctica 5. Sentencias de control

5.1 Objetivo de la práctica

Los objetivos que se persiguen con esta práctica están relacionados con los conceptos asociados a las instrucciones de control de flujo.

Por otro lado, se pretende que el alumno aprenda a usar y se familiarice con las herramientas de depuración de VB (paso a paso, puntos de ruptura, etc)

En concreto los objetivos son los siguientes:

- Empleo de los distintos bucles.
- Uso de contadores.
- Acumulación de sumas y productos.
- Empleo de herramientas de depuración.

5.2 Ejercicio

5.2.1 Primera parte

Solicitar al usuario un valor entero **N y** mostrar la suma todos sus divisores excepto el uno y él mismo. Por ejemplo, si N=36, la suma de todos sus divisores es 54 (2+3+4+6+9+12+18).

5.2.2 Segunda parte

Escribir un programa que dados dos números enteros (z, v) y realice los dos siguientes cálculos:

$$\sum_{i=z}^{v}(i+2) \qquad\qquad \prod_{i=z}^{v}(i+2)$$

El programa debe realizar el cálculo independientemente de que z sea mayor o menor que v.

Emplear las herramientas de depuración.

a) Ejecutar el programa paso a paso y observar como varían los índices del bucle. ¿Qué valor tiene el índice (i) al salir del bucle?
NOTA: Para ello se seleccionará “agregar inspección” dentro del menú “Depuración”.

b) Usar un punto de inspección (*breakpoint*) para parar en la iteración que cumple que *i=x* (siendo x un valor, y no una variable) y ver el valor de la variable que almacena la suma parcial.

Práctica 6. Sentencias de control con vectores

6.1 Objetivo de la práctica

El objetivo de esta práctica es servir de repaso de las estructuras de control e introducir algunas de las estructuras de datos. Esto se puede resumir en los siguientes puntos:

- Repaso de bucles.
- Introducción a las matrices unidimensionales (vectores): definición y acceso a sus elementos.
- Recorrer los elementos de los vectores usando bucles.
- Repaso de las herramientas de depuración.

6.2 Ejercicio

6.2.1 Primera parte. Lectura

El programa comenzará solicitando al usuario dos vectores del mismo tamaño:

1.- Solicitar al usuario las dimensiones de dos vectores (ambos de la misma dimensión)
2.- Bucle para leer los elementos del primer vector *V1* que serán solicitados al usuario.
3.- Bucle para leer los elementos del segundo vector *V2* que serán solicitados al usuario.

6.2.2 Segunda parte. Producto escalar

Calcula el producto escalar de los dos vectores anteriores. Visualizar resultado.

6.2.3 Tercera parte. Media y desviación típica

Calcula la media y la desviación del vector *V1* de acuerdo con las siguientes fórmulas:

$$media = \frac{\sum_{i=1}^{N} x_i}{N} \qquad desviación = \sqrt{\frac{\sum_{i=1}^{N} (x_i - media)^2}{N}}$$