Computación I
Curso 2024

Facultad de Ingeniería
Universidad de la República

Octave

Scripts

- Sucesión de comandos o instrucciones guardados en un archivo .m
- Ejecución
  - Para invocarlo se lo llama por el nombre del archivo.
  - Se ejecuta como si se estuvieran digitando los comandos en la consola, uno tras otro.
  - Pueden ser invocados desde el entorno o desde otro script.

Octave

Scripts

- Variables
  - Alcance global.
  - Puede utilizar y modificar las variables definidas en el entorno interactivo.
  - Puede definir nuevas variables en el entorno interactivo.
- Utilidad
  - Programas utilizados repetidamente.
  - Inicialización de variables de entorno.

Octave

Scripts

- Archivo mi_script.m
- Contenido

```
x = 10;
b = x + 4;
z = x + b + 3;
```

- Ejecución
  - >> mi_script
- Resultado
  - Se crean las variables x, b, y z en el entorno interactivo.
  - Sus valores serán
    - x = 10
    - b = 14
    - z = 27

Octave

Funciones

- Sucesión de comandos o instrucciones guardados en un archivo .m
- Posee características propias de las funciones matemáticas
  - Recibe valores (parámetros) de entrada.
  - Realiza algún cálculo o tarea.
  - Devuelven un resultado (parámetros de salida)

Octave

Funciones

- Sintaxis
  - Cabezal del archivo .m

```
function salida = nombre_funcion (ent1, ent2,
  ...)
```

  - ***Salida*** es el resultado que devuelve la función
    - Puede ser un vector o una matriz
  - Los parámetros ***enti*** son los datos necesarios para la ejecución.

Octave

Funciones

- Sintaxis
  - Al final del archivo

    ```
    endfunction
    ```
  - Indica el final de la función.

Octave
Funciones

- Ejecución
  - El nombre de la función NO debe coincidir con el nombre del archivo .m aunque es recomendable que así sea.
  - Se invoca por su nombre indicando los parámetros necesarios para su ejecución.
  - Pueden ser invocadas desde la consola o desde otra función

Octave
Funciones

- Variables
  - Alcance Local, crea y encapsula sus propias variables.
  - Estás variables existen el tiempo que dure la ejecución de la función.
  - **NO** afecta variables definidas en el entorno interactivo.
  - **NO** puede definir nuevas variables en el entorno interactivo.

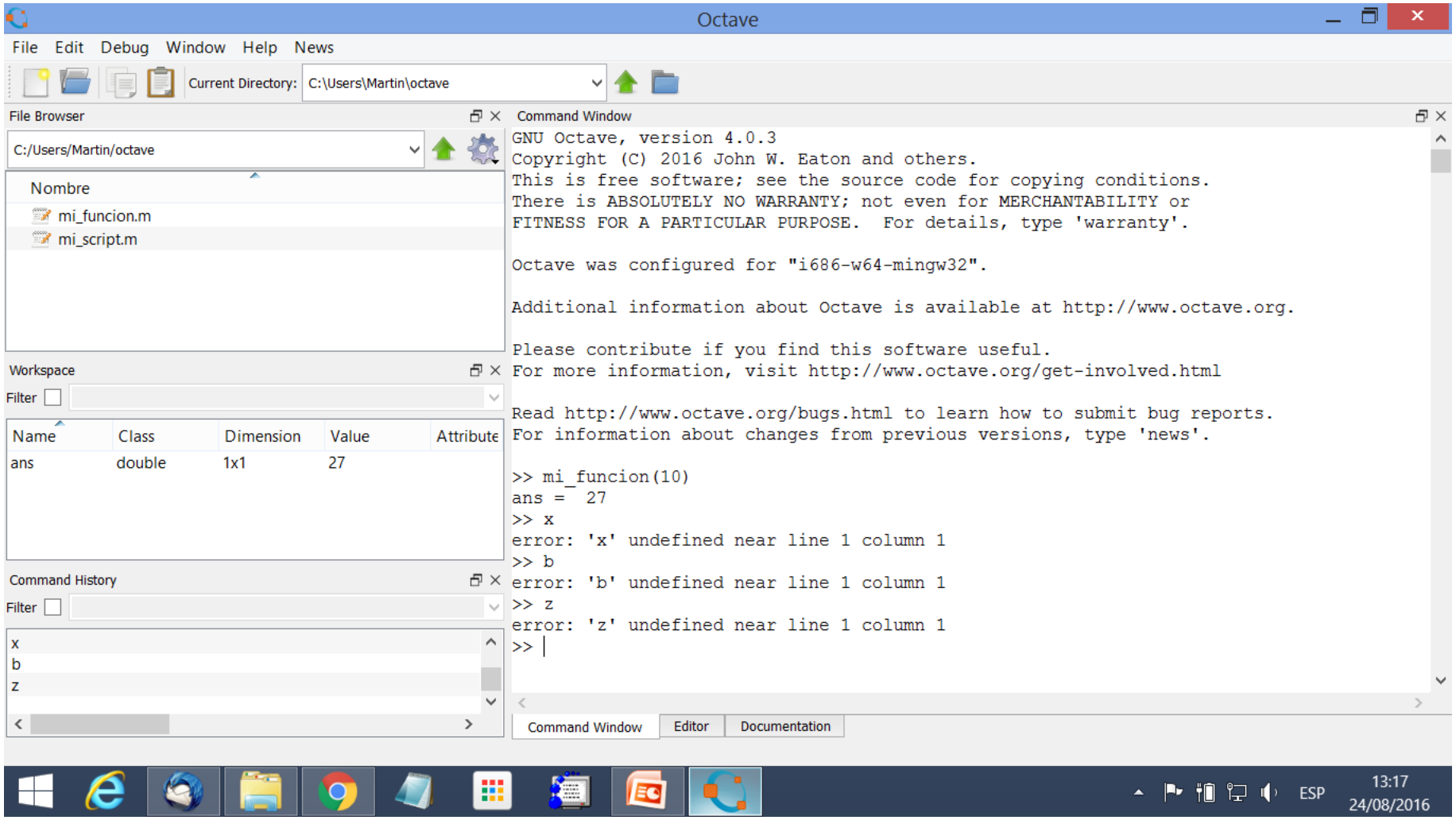
Octave
Funciones

- Archivo mi_funcion.m
- Contenido

```
function z = mi_funcion(x)
   b = x + 4;
   z = x + b + 3;
endfunction
```

- Ejecución
  - >> mi_funcion(10)
- Resultado
  - No se crea ninguna variable en el entorno interactivo.

Octave Funciones



Octave

Funciones

- No utilizar instrucciones para desplegar resultados dentro de una función.
  - Los resultados de una función se deben devolver en los parámetros de salida.
- No utilizar instrucciones para leer los datos de entrada dentro de una función.
  - Las entradas de una función se deben cargar en los parámetros de entrada.

Programación Estructurada

- Tradicionalmente se considera a Edgser W. Dijsktra de la Universidad de Hainover como el padre de la Programación Estructurada.
- En 1965 propuso esta filosofía en un volúmen titulado Notas de Programación Estructurada. Pero no ha sido sino hasta mediados de la década de los setentas cuando comenzó a popularizarse esta filosofía.

Programación Estructurada

- Técnica en la cual la escritura de un programa se realiza tan claramente como es posible mediante el uso de tres estructuras lógicas de control:
  - Secuencia
  - Selección
  - Iteración

Programación Estructurada

- Características
  - Puede ser leído en secuencia, desde el comienzo hasta el final sin perder la continuidad de la tarea que cumple el programa.
  - Tiene un único punto de entrada (arriba).
  - Tiene un único punto de salida (abajo).

Programación Estructurada

Teorema Estructural o Fundamental

- Cualquier programa, no importa el tipo de trabajo que ejecute, puede ser elaborado utilizando únicamente las tres estructuras básicas (secuencia, selección, iteración).

Programación Estructurada

- Ventajas
  - Programas más fáciles de entender
  - Reducción de esfuerzo en las pruebas
  - Reducción de costos de mantenimiento
  - Programas más sencillos
  - Aumento de la productividad del programador
  - Los programas quedan mejor documentados internamente

Programación Estructurada
Alteraciones

- ***goto*** **NO USAR**
  - Saltos no condicionales
- ***exit*** **NO USAR**
  - Corta la ejecución de un programa
- ***loop*** **NO USAR**
  - Repetición sin condiciones
- ***break*** **NO USAR**
  - Rompe un ciclo

Programación Estructurada
Alteraciones

- La utilización de goto, exit, loop y break es una mala práctica de programación porque viola los principios de la programación estructurada.

Estructuras de control
Ejercicios

Escribir una función que:

1) Sume todos los elementos de un vector, devolver -1 si el vector es vacío.

Estructuras de control

Solución Ejercicio1 – sin devolver -1

```
function resultado = sumar(v)
  n = length(v);
  resultado = 0;
  for i = 1:n
     resultado = resultado + v(i);
  endfor
endfunction
```

Estructuras de control

Solución Ejercicio1

```
function resultado = sumar(v)
  n = length(v);
  if n == 0
      resultado = -1;
  else
      resultado = 0;
      for i = 1:n
         resultado = resultado + v(i);
      endfor
  endif
endfunction
```

Estructuras de control
Ejercicios

Escribir una función que:

2) Busque un elemento en un vector y devolver su posición. Si el elemento no pertenece al vector, devolver -1.

Estructuras de control

Solución Ejercicio2

```
function posicion = buscar(v,elem)
  i=1;
  encontre = 0;
  n = length(v);
  while i<=n & ~encontre
     if (v(i) == elem)
         encontre = 1;
     else
         i = i + 1;
     endif
  endwhile
  if encontre
      posicion = i;
  else
      posicion = -1;
  endif
endfunction
```

Estructuras de control

Solución Ejercicio2

```
function posicion = buscar(v,elem)
  i=1;
  posicion = -1;
  n = length(v);
  while i<=n & posicion == -1
     if (v(i) == elem)
         posicion = i;
     else
         i = i + 1;
     endif
  endwhile
endfunction
```

Estructuras de control

Solución Ejercicio2

```
function posicion = buscar(v,elem)
  i=1;
  n = length(v);
  while i<=n && v(i) ~= elem
       i = i + 1;
  endwhile
  if i == n + 1
      posicion = -1;
  else
      posicion = i;
  endif
endfunction
```

Estructuras de control

Solución Ejercicio2 – Recorrida en el otro sentido

```
function posicion = buscar(v,elem)
  n = length(v);
  i = n;
  while 1<=i && v(i) ~= elem
       i = i - 1;
  endwhile
  if i == 0
      posicion = -1;
  else
      posicion = i;
  endif
endfunction
```

Estructuras de control
Ejercicios

Escribir una función que:

3) Intercale los elementos de dos vectores del mismo largo.

Estructuras de control

Solución Ejercicio1

```
function res = intercalar(v1,v2)
  n = length(v1);
  res = zeros(1,2*n);
  for i = 1:n
     res(2*i-1) = v1(i);
     res(2*i) = v2(i);
  endfor
endfunction
```

Estructuras de control

Solución Ejercicio1 - Alternativa

```
function res = intercalar(v1,v2)
  n = length(v1);
  res = [];
  for i = 1:n
     res = [ res, v1(i), v2(i)];
  endfor
endfunction
```