Funciones en programación

$$f(x) = 3x + 1$$

Es una secuencia de código que permite realizar una tarea en específico y que es invocada por otro programa (programa principal). Para poder funcionar las funciones requieren de variables de entrada y después del proceso realizado propiamente por la función devuelven variables de salida. Reglas que se deben de seguir para la construcción de una función en Matlab son las siguientes:

- El nombre de la función debe de ser igual al nombre del archivo que la contiene
- Para mandar a llamar una función desde un programa principal, ésta debe estar en la misma carpeta de la computadora.
- Los nombres de las funciones deben obedecer las mismas reglas que se siguen para nombrar un script en Matlab (no puedan empezar con números, no pueden tener el mismo nombre que algunas otras funciones nativas de Matlab y no pueden tener caracteres especiales).
- Deben de contener al menos una variable de entrada y/o una de salida
- Las variables dentro de una función son independientes a las de otros programas, no se pueden tener acceso a ellas a menos que estén definidas como salida.
- No es válido que en una función se utilicen instrucciones para lectura o despliegue de datos (input, disp, fprintf).
- Las funciones en Matlab deben contener en **LA PRIMERA** línea de código una sentencia con la siguiente sintaxis:

```
function [ VariablesDeSalida ] = NombreDeLaFuncion( VariablesDeEntrada )
%Se comenta la explicacion de que hace la funcion
%   y cuales son las variables con las que trabaja

end
```

function.- Esta es una palabra reservada y debe estar en la PRIMERA línea del código de la función obligatoriamente, al escribirla Matlab la reconocerá y se pondrá en color azul.

VariablesDeSalida.- El argumento de salida se refiere a la o las variables que se tendrán después de la ejecución del código y las cuáles son las únicas que se podrán leer por el programa que invoque la función.

NombreDeLaFunción.- El nombre de la función tiene varias restricciones, por ejemplo: no debe iniciar con números, no debe contener caracteres especiales excepto guión bajo, longitud máxima de 64 caracteres, no debe ser el mismo nombre de una variable que se use en el mismo código, no debe ser una de las palabras reservadas de Matlab.

VariablesDeEntrada.- El argumento de entrada se refiere a la o las variables que se requieren para realizar la tarea que se programa en la función.

Ejemplo 1

Programar una función llamada "Primos" que determine si un número dado (n) es primo o no. A la salida la función debe arrojar un 0 si el número no es primo y un 1 si el número sí es primo.

Un número primo es aquel que sólo es divisible entre 1 y sí mismo.

¿El residuo de 7/1 es igual a 0? Sí
¿ El residuo de 7/2 es igual a 0? No
¿ El residuo de 7/3 es igual a 0? No
¿ El residuo de 7/4 es igual a 0? No
¿ El residuo de 7/5 es igual a 0? No
¿ El residuo de 7/6 es igual a 0? No
¿ El residuo de 7/7 es igual a 0? Sí

4
¿ El residuo de 4/1 es igual a 0? Sí
¿ El residuo de 4/2 es igual a 0? Sí
¿ El residuo de 4/3 es igual a 0? No
¿ El residuo de 4/4 es igual a 0? Sí

```
function Resultados = Primos(n)
%PRIMOS determina si un n?mero es primo o no
%   PRIMOS(n) determina si n es primo la salida es 1
%   o si n no es primo la salida es 0

contador=0;  %Me ayuda a contar cu?ntos n?meros fueron divisibles
for i=1:n
   if rem(n,i)==0
      contador=contador+1;
   end
end
if contador==2
   %Es primo
   Resultados = 1;
else
   %No es primo
   Resultados = 0;
end

end
```

Ejemplo 2

Utilizando la función del ejemplo anterior. Realizar un programa que permita determinar cuántos elementos son primos dentro de una matriz ingresada por el usuario.

Primer método para ingresar matrices

```
f=input('Ingresa el n?mero de filas: ');
c=input('Ingresa el n?mero de columnas: ');

for i=1:f
   for j=1:c
      fprintf('Ingresa el elemento en la fila %g y columna %g:',i,j)
      A(i,j) = input(' ');
   end
end
disp(A)
```

Segundo método para ingresar matrices

```
A=input('Ingresa la matriz: ');
```

El usuario tendrá que escribir [1 2 5; 0 3 4 ; 2 7 0]

```
A=input('Ingresa la matriz: ');
%Con estas instrucciones s? cu?ntas filas y cu?ntas columnas
%tiene la matriz que ingres? el usuario
T=size(A);
f=T(1);
c=T(2);
contador=0;
for i=1:f
   for j=1:c
      n=A(i,j);
      Resultados = Primos(n);
      if Resultados == 1
         contador = contador + 1;
      end
   end
end
fprintf('\nHay %g n?meros primos en la matriz\n',contador)
```

Probar con la siguiente matriz

$$\begin{bmatrix} 3 & 1 & 0 \\ 11 & 5 & 19 \end{bmatrix}$$

Ejemplo 3

Utilizando el programa anterior, a parte de contar cuántos números son primos, también tiene que desplegarlos y guardar y desplegar su posición dentro de la matriz ingresada por el usuario.

```
A=input('Ingresa la matriz: ');
%Con estas instrucciones s? cu?ntas filas y cu?ntas columnas
%tiene la matriz que ingres? el usuario
```

```
T=size(A);
f=T(1);
c=T(2);

fprintf('\nLos n?meros primos son: ')
contador=0;
for i=1:f
    for j=1:c
        n=A(i,j);
        Resultados = Primos(n);
        if Resultados == 1
            contador = contador + 1;
            %n es primo
            fprintf('%g, ',n)
            P(contador,1:2)=[i, j];
        end
    end
end
fprintf('\n\nHay %g n?meros primos en la matriz\n',contador)
fprintf('\n Las posiciones de los n?meros primos en la matriz son:\n')
disp(P)
```

Ejemplo 3

Un número perfecto es aquel que es igual a la suma de sus divisores, excepto el mismo. Por ejemplo: el 6 es perfecto (6 = 3 + 2 + 1) y el 12 no lo es (12≠ 6 + 4 + 3 + 2 + 1). Realizar una función en Matlab que determine si un número n es perfecto o no. Si lo es la variable de salida deberá ser un 1 sino un 0.