VOLUMEN 21
XXVIII Verano De la Ciencia
ISSN 2395-9797
www.jóvenesenlaciencia.ugto.m

Manual

Mario Alberto Ibarra Manzano
Jesus Eduardo Gonzales Trigueros
Emilio Muñoz Rivera
Mariana Jaqueline Peñaran Prieto
Juan Ulises Calderon Huerta
Rodrigo Fabian Cervantes Martínez
División de Ingenierías Campus Irapuato Salamanca DICIS
{ibarram, je.gonzaleztrigueros, e.munozrivera, mj.penaranprieto, ju.calderonhuerta, rf.cervantesmartinez}@ugto.mx

Introducción.

Pac-Man ha sido un símbolo de la cultura de los videojuegos desde su lanzamiento en 1980, capturando la imaginación de jugadores de todas las edades. A lo largo de los años, ha enfrentado a fantasmas ingeniosos, devorado innumerables puntos y buscado salvar los laberintos. Sin embargo, esta vez, la experiencia será diferente.

La Inteligencia Artificial ha sido la fuerza impulsora detrás de una revolución en el mundo de los videojuegos, y Pac-Man no es una excepción.

En este manual, aprenderás cómo funciona esta IA de Pac-Man, los fundamentos de los algoritmos de búsqueda y planificación. Descubrirás cómo nuestra inteligencia artificial ha sido programada para enfrentarse a los famosos fantasmas y utilizar las diferentes instrucciones para superar los laberintos con éxito.

Nuestro objetivo es proporcionarte una guía completa y accesible, que te permita comprender los conceptos clave de la IA utilizada en Pac-Man y, si así lo deseas, experimentar y desarrollar tus propias mejoras en el sistema. Ya seas un estudiante de IA, un apasionado de los videojuegos o simplemente alguien curioso por explorar los límites de la tecnología, este manual es para ti.

ISSN 2395-9797

Contenido

Configuración del sistema.

overallEnemySpeed = 1/8; %Velocidad fantasma estándar, (predeterminado: 1/8, máximo posible: 1/2);
grumpyTime = 700; %Tiempo-incremento por el que los fantasmas permanecen gruñones "grumpy" (predeterminado: 700);
grumpyTimeSwitch = 200; %Tiempo-incremento que los fantasmas gruñones muestran que van a volverse normales de nuevo (predeterminado: 200);
newEnemyTime = 500; % Tiempo-incremento antes de que el siguiente fantasma salga de su jaula (predeterminado: 500)
fruitAppear = [300,1500]; % Tiempo en el que aparecen las frutas en el juego (predeterminado: entre 300 y 1500 tiempo después del inicio del nivel)
game.speed = 0.025; % Velocidad del juego (incremento de tiempo entre dos fotogramas)máxima posible sin retraso en máquina: 0.00
game.faster = -0.001; % Hacer el juego más rápido por nivel. (predeterminado: -0.001
game.maxSpeed = 0.015; % Velocidad máxima del juego (predeterminado: 0.01)
AI.init = 0.0; % IA inicial-> 0: (casi) sin aleatoriedad, 1: aleatoriedad completa
AI.improve = -0.1; % IA mejora por nivel (predeterminado: -0.1)
pacman.speed = 1/6; %Velocidad pacman (predeterminado: 1/6 =>pacman come exactamente dos monedas con cada ciclo de boca abierta/cerrada máximo posible: 1/2)
enemyPersonalities = 1; %Marcar si usar personalidades individuales para cada fantasma o no
showGhostTarget = 0; %Marcar si mostrar hacia dónde se dirige cada fantasma o no
autoPlay = 0; %Marcar si la reproducción automática está activada o no
invincible = 0; %Hacer a pacman invencible
soundsFlag = 1; %Marcar si los sonidos están activados o desactivados.

Funciones principales del sistema.

Monedas y Pastillas.

```
coins = gameData.gameData.coins;                %Datos de las monedas
coins.originalData = coins.data;                %Recuerda los datos para un nuevo juego
coins.plot(coins.data(:,1),coins.data(:,2),'.','Color',[255  185  151]/255,'MarkerSize',7);
%Trazar todas las monedas
pills.data = [2 8; 2 28; 27 8; 27 28];
pills.data = gameData.gameData.pillsData; % Posición de las pastillas energizantes
pills.originalData = pills.data;                % Recordar pastillas
pills.radius = 0.45;                            % Tamaño máximo de las pastillas.
```

Inicializar fantasmas.

```
enemies(1).pos = [14.5, 20];                %Posición del fantasma
enemies(1).dir = 0;                         %Actual dirección del fantasma (derecha-1,
abajo-2, izquierda-3, arriba-4)
enemies(1).oldDir = 1;                      %Ultima dirección del fantasma
enemies(1).speed = overallEnemySpeed;       %Velocidad del fantasma
enemies(1).status = 1;                      %Estado del fantasma (0-enjaulado, 1-
normal, 2-gruñon, 3-ojos)
enemies(1).statusTimer = -1;                % Recordar tiempo para cambiar de estado
enemies(1).curPosMov = [1, 3];              % Posibles movimientos de las casillas
enemies(1).textTimer = 0;                   % Recuerda cuando el enemigo fue comido

enemies(2).pos = [14.5, 16.5];              %Posición del fantasma
enemies(2).dir = 0;                         %Actual dirección del fantasma (derecha-1,
abajo-2, izquierda-3, arriba-4)
enemies(2).oldDir = 1;                      %Ultima dirección del fantasma
enemies(2).speed = overallEnemySpeed;       %Velocidad del fantasma
enemies(2).status = 0;                      %Estado del fantasma (0-enjaulado, 1-
normal, 2-gruñon, 3-ojos)
enemies(2).statusTimer = -1;                %Recordar tiempo para cambiar de estado
enemies(2).curPosMov = 0;                   %Posibles movimientos de las casillas
enemies(2).textTimer = 0;                   % Recuerda cuando el enemigo fue comido

enemies(3).pos = [12.5, 17.5];              % Posición del fantasma
enemies(3).dir = 0;                         % Actual dirección del fantasma (derecha-1,
abajo-2, izquierda-3, arriba-4)
enemies(3).oldDir = 1;                      % Ultima dirección del fantasma
enemies(3).speed = overallEnemySpeed;       % Velocidad del fantasma
enemies(3).status = 0;                      % Estado del fantasma (0-enjaulado, 1-
normal, 2-gruñon, 3-ojos)
enemies(3).statusTimer = -1;                % Recordar tiempo para cambiar de estado
enemies(3).curPosMov = 0;                   % Posibles movimientos de las casillas
enemies(3).textTimer = 0;                   % Recuerda cuando el enemigo fue comido
enemies(4).pos = [16.5, 17.5];              % Posición del fantasma
```

enemies(4).dir = 0; %Actual dirección del fantasma (derecha-1, abajo-2, izquierda-3, arriba-4)
enemies(4).oldDir = 1; %Ultima dirección del fantasma
enemies(4).speed = overallEnemySpeed; %Velocidad del fantasma
enemies(4).status = 0; %Estado del fantasma (0-enjaulado, 1-normal, 2-gruñon, 3-ojos)
enemies(4).statusTimer = -1; %Recordar tiempo para cambiar de estado
enemies(4).curPosMov = 0; %Posibles movimientos de las casillas
enemies(4).textTimer = 0; %Recuerda cuando el enemigo fue comido

Modos de dispersión o persecución (0: modo de persecución, 1: modo de dispersión)

ghostMode.timerValues = [250 1000; 1500 2000; 2650 3650; 3800 -1]; % cambia de un lado a otro del modo de persecución al modo de dispersión después de un tiempo. Al final solo persiguen
ghostMode.timerStatus = 1; %En qué intervalo de "ghostMode.timerValues" se encuentra
ghostMode.status = 0; % 0: persecución, 1: dispersión

Frutas

fruits.pos = [0, 0]; % Posición de la fruta
fruits.item = 1; % Fruta del nivel actual
fruits.score = [100 100 200 200 300 300 400 500]; %Puntuación para cada fruta
fruits.picked = zeros(1,8); %Veces en que se recogió cada fruta
fruits.timer = randi([fruitAppear(1),fruitAppear(2)],1); %Ventana de tiempo en que aparecerá la fruta
ghostFrame = 1; %Hacer que los fantasmas se tambaleen
grumpyColorChange = 0; %Determina el color del fantasma gruñón (azul o blanco)
grumpyTimeSwitchSave = 0; %La variable recuerda el estado del temporizador, por lo que todos los fantasmas gruñones cambian al mismo tiempo (azul-blanco-azul)
ghostPoints = 100; %Determina cuantos puntos agrega un fantasma a la puntuación (se duplican con cada muerte)

Inicializar pac-Man

pacman.size = 0.8; % Tamaño de pacman
pacman.pos = [14.5 8]; % Posición de pacman
pacman.dir = 0; % Dirección de pacman
pacman.oldDir = 1; % Antigua dirección de pacman
pacman.status = -2; % -2 es normal, -3 es golpeado por un fantasma.

Vidas de pacman

```
lives.orig = 3;                    %Vidas de pacman
lives.data = lives.orig;           %Recuerda las vidas predeterminadas de pacman.
```

Inteligencias Artificiales

IA 1.

```
% pacmanAI: Función que determina el siguiente cuadrado objetivo para el Pacman en función de la posición de los enemigos, las monedas y las píldoras disponibles.
% Parámetros de entrada:
%   - pacman: Estructura que representa la posición y estado del Pacman.
%   - enemies: Vector de estructuras que contiene información sobre los enemigos, incluyendo su posición y estado.
%   - allDirections: Vector que contiene todas las direcciones posibles de movimiento del Pacman.
%   - coins: Estructura que contiene información sobre las monedas disponibles, incluyendo sus posiciones.
%   - pills: Estructura que contiene información sobre las píldoras disponibles, incluyendo sus posiciones.
% Valor de retorno:
%   - targetSquare: Vector que representa la posición del cuadrado objetivo al que el Pacman se dirigirá.

function targetSquare = pacmanAI(pacman, enemies, allDirections, coins, pills)
% pacmanAI: Función que determina el siguiente cuadrado objetivo para el Pacman en función de la posición de los enemigos, las monedas y las píldoras disponibles.

% Inicialización de variables
c = 0;
distF = 100 * ones(4, 2);

% Cálculo de la distancia de Pacman hacia los enemigos
for nn = 1:4
   if enemies(nn).status == 1

% Distancia a los enemigos en su modo normal
      distF(nn, 1) = sqrt((pacman.pos(1) - enemies(nn).pos(:, 1)).^2 + (pacman.pos(2) - enemies(nn).pos(:, 2)).^2);
   elseif enemies(nn).status == 2

% Distancia a los enemigos en modo “grumpy”
```

```
    distF(nn, 2) = sqrt((pacman.pos(1) - enemies(nn).pos(:, 1)).^2 + (pacman.pos(2) - enemies(nn).pos(:, 2)).^2);
  end
end

% Obtener el índice del enemigo más cercano en su modo normal
[minDisF, minDisF_Index] = min(distF(:, 1))

% Obtener el índice del enemigo en modo "grumpy" más cercano
[minDisF2, minDisF2_Index] = min(distF(:, 2));


% Si hay enemigos cercanos en su estado normal, evitarlos
if (distF(1, 1) < 7 && enemies(1).status == 1) || (distF(2, 1) < 6 && enemies(2).status == 1) || (distF(3, 1) < 6 && enemies(3).status == 1) || (distF(4, 1) < 5 && enemies(4).status == 1)

% Si hay enemigos cercanos en su estado normal y muy cercanos, huir de ellos
  if (distF(1, 1) < 5 && enemies(1).status == 1) || (distF(2, 1) < 4 && enemies(2).status == 1) || (distF(3, 1) < 4 && enemies(3).status == 1) || (distF(4, 1) < 3 && enemies(4).status == 1)

%Calcular la dirección para alejarse del enemigo más cercano
    Vd = [(pacman.pos(1) - enemies(minDisF_Index).pos(:, 1)), (pacman.pos(2) - enemies(minDisF_Index).pos(:, 2))];
    V_n = [Vd(1) / minDisF, Vd(2) / minDisF];
    si = [pacman.pos(1) + V_n(1) * 30, pacman.pos(2) + V_n(2) * 30];
    c = 1;

  else

% Si no hay enemigos muy cercanos, elegir el siguiente objetivo entre las monedas y píldoras disponibles
    if isempty(pills.data) %Si no hay píldoras disponibles, dirigirse hacia la moneda más cercana
      dstC = sqrt((coins.data(:, 1) - pacman.pos(1)).^2 + (coins.data(:, 2) - pacman.pos(2)).^2);
      [mn_dstC, id_dstC] = min(dstC);
      si = coins.data(id_dstC, :);
    else

% Si hay píldoras disponibles, dirigirse hacia la píldora más cercana
      dstP = sqrt((pills.data(:, 1) - pacman.pos(1)).^2 + (pills.data(:, 2) - pacman.pos(2)).^2);
      [mn_dstP, id_dstP] = min(dstP);
      si = pills.data(id_dstP, :);
    end
  end
```

```
else

% Si no hay enemigos cercanos en su estado normal, dirigirse hacia la moneda más cercana
    dstC = sqrt((coins.data(:, 1) - pacman.pos(1)).^2 + (coins.data(:, 2) - pacman.pos(2)).^2);
    [mn_dstC, id_dstC] = min(dstC);
    si = coins.data(id_dstC, :);
end

% Si hay enemigos perseguidores muy cercanos, dirigirse hacia ellos
if ((enemies(1).status == 2 && (minDisF2 > 0 && minDisF2 < 3)) || (enemies(2).status ==
2 && (minDisF2 > 0 && minDisF2 < 3)) || (enemies(3).status == 2 && (minDisF2 > 0 &&
minDisF2 < 3)) || (enemies(4).status == 2 && (minDisF2 > 0 && minDisF2 < 3))) && c ==
0
    si = enemies(minDisF2_Index).pos;
end

% Cuadrado objetivo para el Pacman
targetSquare = si;
```

IA 2.

```
function targetSquare = pacmanAI(pacman, enemies, allDirections, coins, pills)
%pacmanAI: Función que determina el siguiente cuadrado objetivo para el Pacman en
función de la posición de los enemigos, las monedas y las píldoras disponibles.

% Descripción: Calcula el siguiente cuadrado objetivo (posición) para el Pacman, en función
de la posición de los enemigos y las monedas disponibles.

% Parámetros de entrada:

%    - allDist: Matriz 4x2 que almacena la distancia de los enemigos hacia el Pacman. La
columna 1 contiene las distancias de los enemigos asustados, la columna 2 contiene las
distancias de los enemigos perseguidores.

%    - enemies: Vector de estructuras que contiene información sobre los enemigos, incluyendo
su posición y estado.

%    - pacman: Estructura que representa la posición y estado del Pacman.

%    - coins: Estructura que contiene información sobre las monedas disponibles, incluyendo
sus posiciones.

% Valor de retorno:

%    - targetSquare: Vector que representa la posición del cuadrado objetivo al que el Pacman
se dirigirá.
```

```
% Inicialización de la matriz de distancias de los enemigos hacia el Pacman
allDist = 100 * ones(4, 2);


% Cálculo de la distancia de los enemigos hacia el Pacman
for nn = 1:4
   if enemies(nn).status == 1
% Si el enemigo está asustado, calcular la distancia utilizando la fórmula de la distancia euclidiana
      allDist(nn, 1) = sqrt((enemies(nn).pos(:, 1) - pacman.pos(1)).^2 + (enemies(nn).pos(:, 2) - pacman.pos(2)).^2);
   elseif enemies(nn).status == 2
% Si el enemigo está perseguidor, calcular la distancia utilizando la función 'norm'
      allDist(nn, 2) = norm(pacman.pos - enemies(nn).pos);
   end
end


% Comprobar si hay enemigos perseguidores
if enemies(1).status == 2 || enemies(2).status == 2 || enemies(3).status == 2 || enemies(4).status == 2
% Obtener el índice del enemigo perseguidor más cercano
   [minDist2, minDist2_Index] = min(allDist(:, 2));
% Establecer la posición del cuadrado objetivo como la posición del enemigo perseguidor más cercano
   si = enemies(minDist2_Index).pos;
else
% Si no hay enemigos perseguidores, dirigirse hacia la moneda más cercana
   dst = sqrt((coins.data(:, 1) - pacman.pos(1)).^2 + (coins.data(:, 2) - pacman.pos(2)).^2);
   [mn_dst, id_dst] = min(dst);
% Establecer la posición del cuadrado objetivo como la posición de la moneda más cercana
   si = coins.data(id_dst, :);
```

```
end


% Asignar la posición del cuadrado objetivo al valor de retorno

targetSquare = si;
```

IA 3

```
% Va por las monedas
%Calcula la distancia entre las monedas y pacman
 distancia_Coins = sqrt((coins.data(:, 1)-pacman.pos(1)).^2 + (coins.data(:,2)-
pacman.pos(2)).^2);
 %Encuentra la distancia mas corta y se dirige hacia ella
 [~, id_minDistancia_Coins] = min(distancia_Coins);
 pacman.pos
 targetSquare = coins.data(id_minDistancia_Coins, :);


% Se aleja de los enemigos
 %Se generan los vectores para almacenar la informacion de cada fantasma
 distancia_Amenaza = zeros(4,1);
 deltaX = zeros(4,1);
 deltaY = zeros(4,1);
 for nn = 1:4
 if (enemies(nn).status == 1)

 %Calcula la diferencia de coordenadas en el eje x, y, entre pacman y cada uno
 deltaX(nn,1) = enemies(nn).pos(:, 1)-pacman.pos(1);
 deltaY(nn,1) = enemies(nn).pos(:,2)-pacman.pos(2);


 %Realiza la suma de cada uno de los ejes para calcular el vector resultante
 SumX = sum(deltaX); SumY = sum(deltaY);
 fuerza_Resultante = sqrt((SumX).^2 + (SumY).^2);

 %Encuentra el angulo del vector resultante
 angulo = (atan(SumY/SumX));


 %Calcula las coordenadas a las que se debe dirigir pacman y se
 %le añade la constante 2, debido al espacio en los que puede no tener

 coordenada_Y = fuerza_Resultante*sin(angulo) + 2;
```

```
coordenada_X = fuerza_Resultante*cosd(angulo) + 2;


%Obtiene la distancia entre cada uno de los fantasmas y pacman

distancia_Amenaza(nn,1) = sqrt((deltaX(nn,1)).^2 + (deltaY(nn,1)).^2);
if (distancia_Amenaza(nn,1) <= 5)


%Condiciones para los enemigos a la izquierda de pacman

if (enemies(nn).pos(1) < pacman.pos(1)) && (enemies(nn).dir == 1)
targetSquare = pacman.pos + [coordenada_X,-coordenada_Y];
elseif (enemies(nn).pos(1) < pacman.pos(1)) && (enemies(nn).dir == 4)
targetSquare = pacman.pos + [coordenada_X,-coordenada_Y];
elseif (enemies(nn).pos(1) < pacman.pos(1)) && (enemies(nn).dir == 2)
targetSquare = pacman.pos + [coordenada_X,coordenada_Y];
elseif (enemies(nn).pos(1) < pacman.pos(1)) && (enemies(nn).dir == 3)
targetSquare = pacman.pos + [coordenada_X,coordenada_Y];


%Condiciones para los enemigos a la derecha de pacman

elseif (enemies(nn).pos(1) > pacman.pos(1)) && (enemies(nn).dir == 1)
targetSquare = pacman.pos + [-coordenada_X,-coordenada_Y];
elseif (enemies(nn).pos(1) > pacman.pos(1)) && (enemies(nn).dir == 4)
targetSquare = pacman.pos + [-coordenada_X,-coordenada_Y];
elseif (enemies(nn).pos(1) > pacman.pos(1)) && (enemies(nn).dir == 2)
targetSquare = pacman.pos + [-coordenada_X,coordenada_Y];
elseif (enemies(nn).pos(1) > pacman.pos(1)) && (enemies(nn).dir == 3)
targetSquare = pacman.pos + [-coordenada_X,coordenada_Y];


%Condiciones para los enemigos arriba de pacman

elseif (enemies(nn).pos(2) > pacman.pos(2)) && (enemies(nn).dir == 1)
targetSquare = pacman.pos + [-coordenada_X,-coordenada_Y];
elseif (enemies(nn).pos(2) > pacman.pos(2)) && (enemies(nn).dir == 4)
targetSquare = pacman.pos + [-coordenada_X,-coordenada_Y];
elseif (enemies(nn).pos(2) > pacman.pos(2)) && (enemies(nn).dir == 2)
targetSquare = pacman.pos + [coordenada_X,-coordenada_Y];
elseif (enemies(nn).pos(2) > pacman.pos(2)) && (enemies(nn).dir == 3)
targetSquare = pacman.pos + [coordenada_X,-coordenada_Y];


%Condiciones para los enemigos debajo de pacman
```

```
 elseif (enemies(nn).pos(2) < pacman.pos(2)) && (enemies(nn).dir == 1)
 targetSquare = pacman.pos + [-coordenada_X,coordenada_Y];
 elseif (enemies(nn).pos(2) < pacman.pos(2)) && (enemies(nn).dir == 4)
 targetSquare = pacman.pos + [-coordenada_X,coordenada_Y];
 elseif (enemies(nn).pos(2) < pacman.pos(2)) && (enemies(nn).dir == 2)
 targetSquare = pacman.pos + [coordenada_X,coordenada_Y];
 elseif (enemies(nn).pos(2) < pacman.pos(2)) && (enemies(nn).dir == 3)
 targetSquare = pacman.pos + [coordenada_X,coordenada_Y];


 %Condiciones para las aristas y esquinas

 elseif (pacman.pos(1) <= 3) && (enemies(nn).pos(2) >= pacman.pos(2))
 targetSquare = pacman.pos + [5,-5];
 elseif (pacman.pos(1) <= 3) && (enemies(nn).pos(2) <= pacman.pos(2))
 targetSquare = pacman.pos + [5,5];
 elseif (pacman.pos(1) >= 27) && (enemies(nn).pos(2) >= pacman.pos(2))
 targetSquare = pacman.pos + [-5,-5];
 elseif (pacman.pos(1) >= 27) && (enemies(nn).pos(2) <= pacman.pos(2))
 targetSquare = pacman.pos + [-5,5];
 elseif (pacman.pos(1) <= 3) && (pacman.pos(2) <= 3) && (enemies(nn).pos(2) >=
pacman.pos(2))
 targetSquare = pacman.pos + [5,-5];
 elseif (pacman.pos(1) <= 3) && (pacman.pos(2) >= 29) && (enemies(nn).pos(2) <=
pacman.pos(2))
 targetSquare = pacman.pos + [5,5];
 elseif (pacman.pos(1) >= 26) && (pacman.pos(2) <= 3) && (enemies(nn).pos(2) >=
pacman.pos(2))
 targetSquare = pacman.pos + [-5,-5];
 elseif (pacman.pos(1) >= 26) && (pacman.pos(2) >= 29) && (enemies(nn).pos(2) <=
pacman.pos(2))
 targetSquare = pacman.pos + [-5,5];
 end
 end
 end
 end
 pacman.pos


% Se come a los enemigos

 distancia_Comer = zeros(4,1);
 for nn = 1:4
 if (enemies(nn).status == 2)
 distancia_Comer(nn,1) = sqrt((enemies(nn).pos(:, 1)-pacman.pos(1)).^2
+(enemies(nn).pos(:,2)-pacman.pos(2)).^2);
```

```
 %Se come a los enemigos siempre y cuando estén dentro de un rango de distancia
considerable

 if(distancia_Comer(nn,1) <= 5)
 targetSquare = enemies(nn).pos;
 end
 end
 end
```

Glosario

Condicional if-else.

El condicional "if-else" es una estructura de control utilizada en la mayoría de los lenguajes de programación, incluyendo MATLAB. Su función principal es permitir que un programa tome decisiones y ejecute diferentes bloques de código según se cumpla una condición específica.

El funcionamiento del condicional "if-else" puede describirse en los siguientes pasos:

Se evalúa una condición: El programa verifica si una expresión lógica (la condición) es verdadera o falsa. Esta evaluación es crucial para determinar qué bloque de código se ejecutará.

Ejecución del bloque "if": Si la condición se evalúa como verdadera, el bloque de código dentro del "if" se ejecuta. Este bloque de código puede contener una o varias instrucciones que el programa seguirá cuando se cumpla la condición.

Ejecución del bloque "else": Si la condición se evalúa como falsa, el programa ejecuta el bloque de código dentro del "else" (si está presente). Este bloque de código representa lo que se debe hacer cuando la condición no se cumple.

Fin del condicional: Una vez que se ha ejecutado el bloque de código correspondiente (ya sea el "if" o el "else"), el programa continúa con la siguiente instrucción fuera del condicional.

El "elseif" es una extensión del condicional "if-else" que se utiliza cuando deseas evaluar múltiples condiciones diferentes en un programa. Permite manejar más de dos posibles casos, cada uno con su propio bloque de código para ejecutar.

Ciclo for.

El bucle "for" es una estructura de control que se utiliza para repetir un bloque de código un número específico de veces. Es especialmente útil cuando se necesita realizar una tarea o cálculo repetitivo en una secuencia de valores o elementos.

El bucle "for" se puede utilizar para recorrer matrices, generar secuencias numéricas, procesar datos y realizar cálculos en múltiples iteraciones.

Función.

Una función es un bloque de código que realiza una tarea específica y se puede reutilizar en diferentes partes de un programa. Las funciones ayudan a dividir un programa en partes más pequeñas y manejables, lo que facilita el mantenimiento, la comprensión y la organización del código.

Una función tiene un nombre y puede recibir cero o más parámetros (también llamados argumentos). Los parámetros son variables que permiten que la función acepte datos de entrada. La función realiza sus operaciones utilizando los parámetros y puede devolver un resultado o realizar una acción sin devolver un valor.

%

Es la forma en la que se comenta en el lenguaje de MATLAB.