LENGUAJE DE PSEUDOCÓDIGO

METODOLOGÍA DE LA PROGRAMACIÓN

Ingeniería Informática

Primer curso, segundo cuatrimestre

Escuela Politécnica Superior de Córdoba

Universidad de Córdoba

Contenido

1. Sentencias

Las sentencias del lenguaje de pseudocódigo que se van a describir son:

- Asignación.
- Condicional simple: "*si*".
- Condiciona múltiple: "*según*".
- Iterativa "*mientras*".
- Iterativa "*hacer…mientras que*".
- Iterativa ***"repetir…hasta que".***
- Iterativa "*para*".

1.1. Sentencia de asignación

La sentencia de asignación permite almacenar un valor en un objeto.

- **Sintaxis**

  <objeto> ← <expresión> ;

  donde:

  - <objeto>: puede ser
    - un nombre de una variable,
    - un campo o atributo de una estructura o registro.
    - o una componente de un vector o matriz.
  - ←: operador de asignación.
  - < expresión>: expresión numérica o alfanumérica cuyo valor se va a asignar al objeto.
  - El objeto y la expresión han de tener el mismo tipo de datos.

- **Ejemplos**

  edad ← 21;

  *jugadores ← equipos * 11;*

  ciudades[3] ← "Córdoba";

  persona.nombre ← "Álvaro" ;

1.2. Sentencia condicional simple: "si"

La sentencia condicional simple "***si***" permite controlar la ejecución de dos grupos de sentencias dependiendo del resultado de una condición lógica (figura 1).

- **Sintaxis**[1]

  si *<condición>*

  entonces

[1] Los corchetes [ ] indican que es opcional.

<sentencias del consecuente>

[si_no

<sentencias de la alternativa>]

fin_si;

donde

- *<condición>: expresión lógica*
- *<sentencias del consecuente>: secuencia de una o más sentencias.*
- *<sentencias de la alternativa>: secuencia de una o más sentencias.*

- **Significado**
  1. Se evalúa la *<condición>*, que debe ser una expresión lógica.
  2. Si el resultado es ***verdadero o cierto*** entonces se ejecutan las *<sentencias del consecuente>.*
  3. Si el resultado es ***falso*** entonces se ejecutan las *<sentencias de la alternativa>*, si existen.

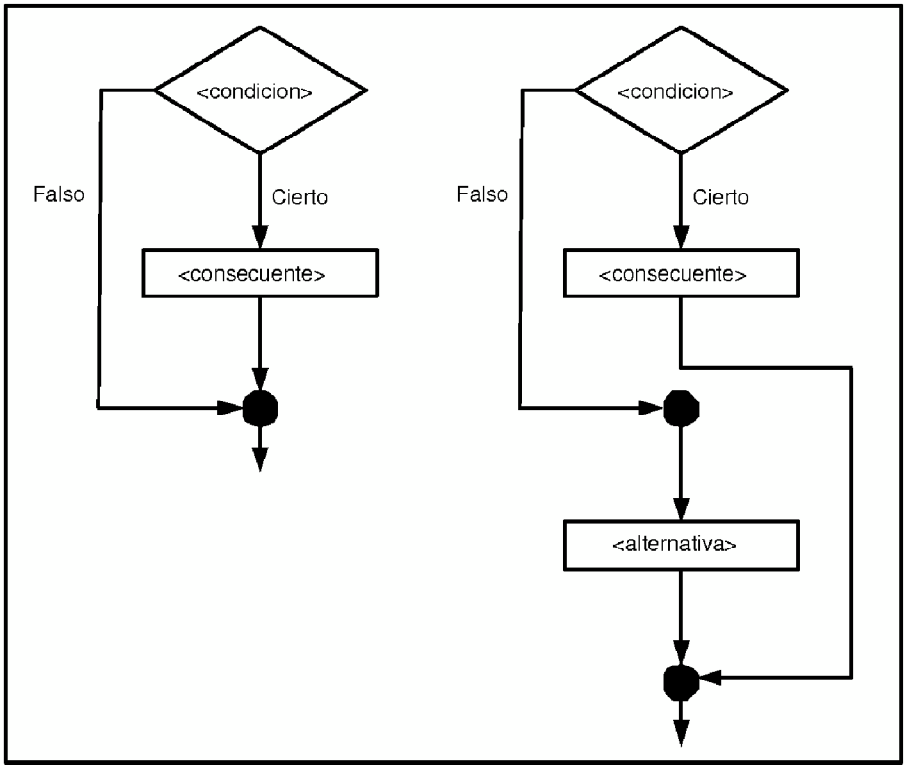

Figura 1. Funcionamiento de la sentencia condicional "*si*".

- **Ejemplos**

si *(x >= 0)*

entonces

resultado ← raíz_cuadrada(x);

fin_si;

si *(número_jugadores == 11)*

entonces

equipo_completo ← verdadero;

si_no

equipo_completo ← falso;

fin_si*;*

1.3. Sentencia condicional múltiple: "según"

La sentencia condicional múltiple **"según"** permite controlar la ejecución de varios grupos de sentencias dependiendo del resultado de una expresión.

- **Sintaxis**

según *<expresión clave>* ***hacer***

<valor *1> :*

<sentencias del bloque 1>

<valor 2>*:*

<sentencias del bloque 2>

[…]

*[****otros*** *:*

<sentencias del bloque otros>]

fin_según;

donde

- *<expresión clave>: expresión aritmética o alfanumérica*
- *<valor i>: dato numérico o alfanumérico.*
- *<sentencias del bloque i>: secuencia de una o más sentencias.*
- *<sentencias del bloque otros>: secuencia de una o más sentencias.*

- **Significado**
  1. Se evalúa la *<expresión clave>* y se obtiene un resultado.
  2. Si el resultado coincide con el *<valor i>* entonces se ejecutan las *<sentencias del bloque i>.*
  3. Si el resultado no coincide con ningún valor entonces se ejecutan las *<sentencias del bloque otros>*, si existen.

- **Ejemplo**

según *numero_lados* ***hacer***

3: objeto <- "triangulo";

4: objeto <- "cuadrado";

otros: objeto <- "no catalogado"

fin_según;

1.4. Sentencia iterativa: "mientras"

La sentencia iterativa "***mientras***" permite ejecutar una secuencia de sentencias ***mientras*** que una condición lógica, situada al **principio** del bucle, sea ***verdadera*** (figura 2).

- **Sintaxis**

mientras *<condición>* ***hacer***

<sentencias del bucle>

fin_mientras;

donde

- *<condición>: expresión lógica*
- *<sentencias del bucle>: secuencia de una o más sentencias.*

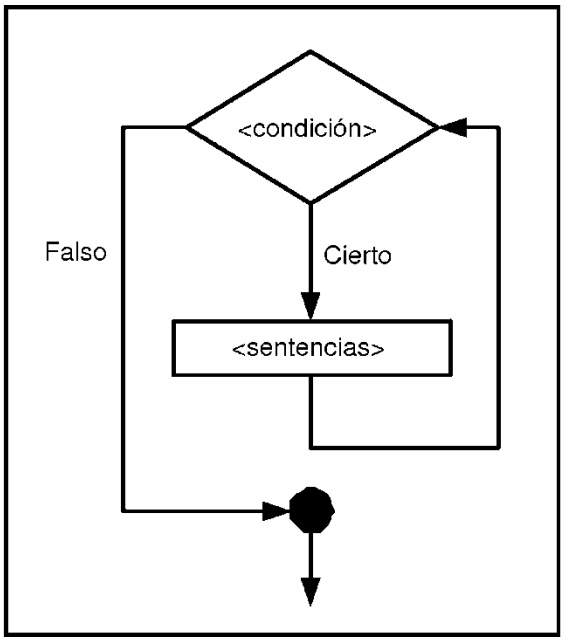

Figura 2. Funcionamiento de la sentencia "*mientras*".

- **Significado**
  1. Se evalúa la *<condición>*
  2. Si el resultado de la *<condición>* es ***verdadero*** entonces
     2.1. se ejecutan las sentencias del bucle
     2.2. y se vuelve al paso 1

3. Si el resultado de la <*condición*> es ***falso*** entonces termina la ejecución del bucle.

- ***Ejemplo***

i ← N;

factorial ← 1;

mientras *(i > 1)* ***hacer***

*factorial ← factorial * i;*

i ← i – 1;

fin_mientras;

1.5. Sentencia iterativa "hacer…mientras que"

La sentencia iterativa ***"hacer…mientras que"*** permite ejecutar una secuencia de sentencias ***mientras que*** la condición, situada al **final** del bucle, sea ***verdadera*** (figura 3).

- **Sintaxis**

hacer

<*sentencias del bucle*>

mientras_que <*condición*>;

donde

- <*sentencias del bucle*>: *secuencia de una o más sentencias.*
- <*condición*>: *expresión lógica*

- **Significado**

1. Se ejecutan las <*sentencias del bucle*>.
2. Se evalúa la <*condición*>
3. Si el resultado de la <*condición*> es ***verdadero*** entonces se vuelve al paso 1.
4. Si el resultado de la <*condición*> es ***falso*** entonces finaliza la ejecución del bucle.

- **Ejemplo**

i ← N;

factorial ← 1;

hacer

*factorial ← factorial * i;*

i ← i – 1;

mientras_que *(i > 1);*

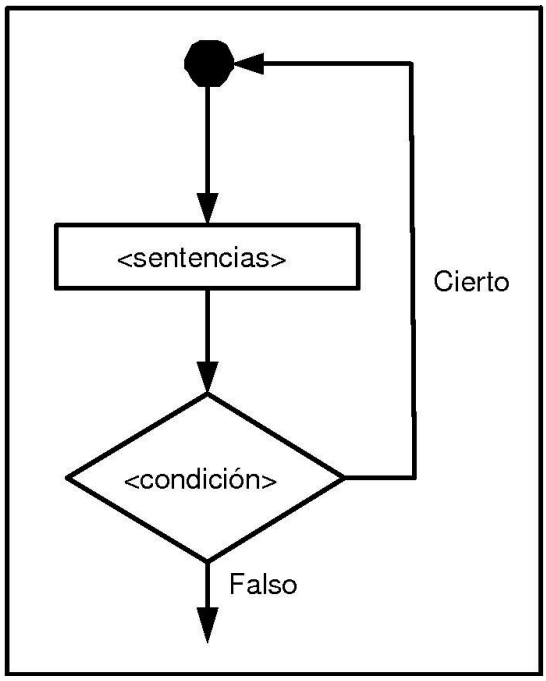

Figura 3. Funcionamiento de la sentencia *"hacer...mientras que".*

1.6. Sentencia iterativa: "repetir...hasta que"

La sentencia iterativa ***"repetir...hasta que"*** permite ejecutar una secuencia de sentencias ***hasta que*** una condición lógica, situada al **final** del bucle, sea ***verdadera*** (figura 4).

- **Sintaxis**

  repetir

  <sentencias del bucle>

  hasta_que *<condición>;*

  donde

  - *<sentencias del bucle>: secuencia de una o más sentencias.*
  - *<condición>: expresión lógica.*

- **Significado**
  1. Se ejecutan las *<sentencias del bucle>.*
  2. Se evalúa la *<condición>*
  3. Si el resultado de la *<condición>* es ***falso*** entonces se vuelve al paso 1.
  4. Si el resultado de la *<condición>* es ***verdadero*** entonces finaliza la ejecución del bucle.

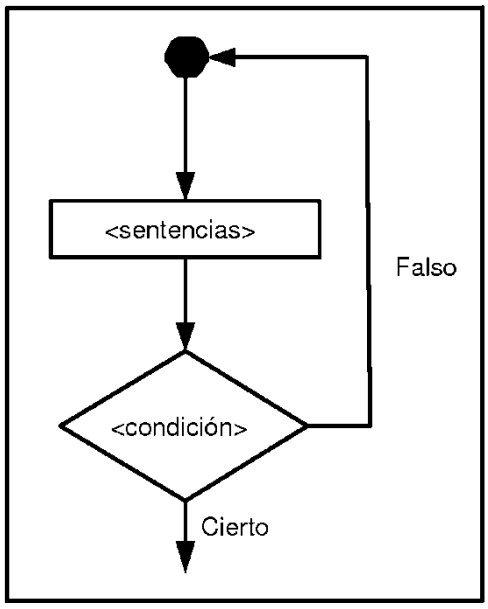

Figura 4. Funcionamiento de la sentencia *"repetir...hasta que".*

- ***Ejemplo***

```
i ← N;
factorial ← 1;
repetir
        factorial ← factorial * i;
        i ← i – 1;
hasta_que (i == 1);
```

1.7. Sentencia iterativa "para"

La sentencia iterativa "para" permite ejecutar una secuencia de sentencias utilizando una variable de control y expresiones aritméticas (figura 5).

- **Sintaxis**

```
para <contador>
        desde <expresión inicial>
        hasta <expresión final>
        [ paso <expresión de paso> ]
        hacer
                <sentencias del bucle>
fin_para;
```

donde:

- <*contador*>: variable que almacena el número de la iteración actual.

- <*expresión inicial*>: expresión aritmética
- <*expresión final*>: expresión aritmética
- <*expresión de paso*>: expresión aritmética
  - Si se omite, el valor por defecto es 1.
- <sentencias del bucle>: secuencia de una o más sentencias.

- **Significado**

1. Se evalúa la <*expresión inicial*>, la <*expresión final*> y la <*expresión de paso*>.
2. Se asigna a la variable <*contador*> el valor de la <*expresión inicial*>.
3. Si el valor de la <*expresión de paso*> es positivo y el valor de <*contador*> es menor o igual que el valor de la <*expresión final*> entonces
   3.1. Se ejecutan las <*sentencias del bucle*>
   3.2. Se incrementa valor de <contador> usando el valor de la <*expresión de paso*>.
   3.3. Se vuelve al paso 3
4. Si el valor de la <*expresión de paso*> es negativo y el valor de <*contador*> es mayor o igual que el valor de la <*expresión final*> entonces
   4.1. Se ejecutan las <*sentencias del bucle*>
   4.2. Se decrementa valor de <contador> usando el valor de la <*expresión de paso*>.
   4.3. Se vuelve al paso 4.

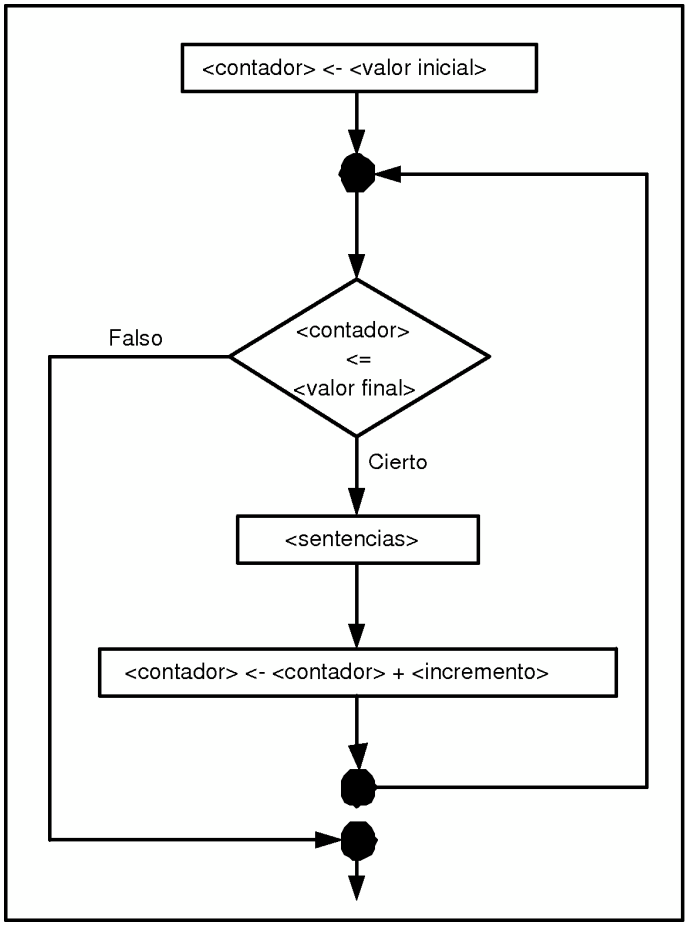

Figura 5. Funcionamiento de la sentencia "*para*" cuando el "paso" es positivo.

- **Ejemplo**

```
factorial ← 1;
para i desde N hasta 1 paso -1 hacer
        factorial ← factorial * i;
fin_para;
```