Desarrollo de programas

Estructuras de programación

Un programa puede ser escrito utilizando tres tipos de estructuras de control:

a) secuenciales
b) selectivas o de decisión
c) repetitivas

Las Estructuras de Control determinan el orden en que deben ejecutarse las instrucciones de un algoritmo: si serán recorridas una luego de la otra, si habrá que tomar decisiones sobre si ejecutar o no alguna acción o si habrá repeticiones.

Estructura secuencial

Es la estructura en donde una acción (instrucción) sigue a otra de manera secuencial. Las tareas se dan de tal forma que la salida de una es la entrada de la que sigue y así en lo sucesivo hasta cumplir con todo el proceso. Esta estructura de control es la más simple, permite que las instrucciones que la constituyen se ejecuten una tras otra en el orden en que se listan. Por ejemplo, considérese el siguiente fragmento de un algoritmo:

En este fragmento se indica que se ejecute la operación 1 y a continuación la operación 2.

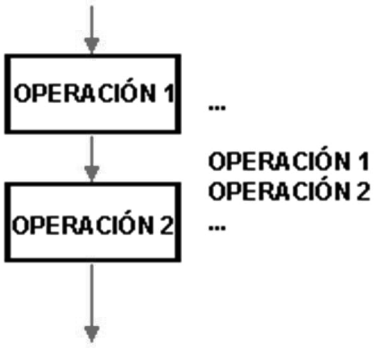


Figura 15: Diagrama de Flujo Secuencial

Estructura alternativa

Estas estructuras de control son de gran utilidad para cuando el algoritmo a desarrollar requiera una descripción más complicada que una lista sencilla de instrucciones. Este es el caso cuando existe un número de posibles alternativas que resultan de la evaluación de una determinada condición.

Este tipo de estructuras son utilizadas para tomar decisiones lógicas, es por esto que también se denominan estructuras de decisión o selectivas.

En estas estructuras, se realiza una evaluación de una condición y de acuerdo al resultado, el algoritmo realiza una determinada acción. Las condiciones son especificadas utilizando expresiones lógicas.

Las estructuras selectivas/alternativas pueden ser:

- Simples
- Dobles
- Múltiples

Alternativa simple (si-entonces/if-then)

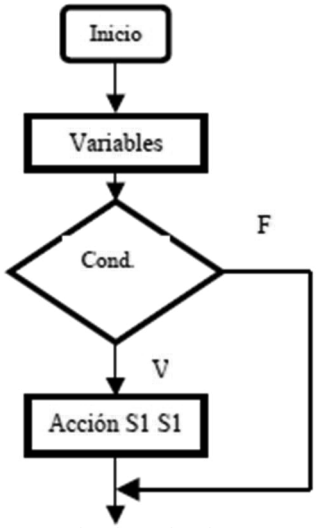


Figura 16: Diagrama de Flujo de alternativa simple

La estructura alternativa simple si-entonces (en inglés if-then) lleva a cabo una acción al cumplirse una determinada condición. La selección si-entonces evalúa la condición y:

- Si la condición es verdadera, ejecuta la acción S1
- Si la condición es falsa, no ejecuta nada.

En español:	En Inglés:
Si <condición> Entonces <acción S1> Fin_si	If <condición> Then <acción S1> End_if

Ejemplo:

```
INICIO
  ENTERO edad = 18
  SI (edad > 18)
  ENTONCES:
    puede manejar un auto
  FIN_SI
FIN
```

Alternativa Doble (si-entonces-sino/if-then-else)

Existen limitaciones en la estructura anterior, y se necesitará normalmente una estructura que permita elegir dos opciones o alternativas posibles, de acuerdo al cumplimiento o no de una determinada condición:

- Si la condición es verdadera, se ejecuta la acción S1
- Si la condición es falsa, se ejecuta la acción S2

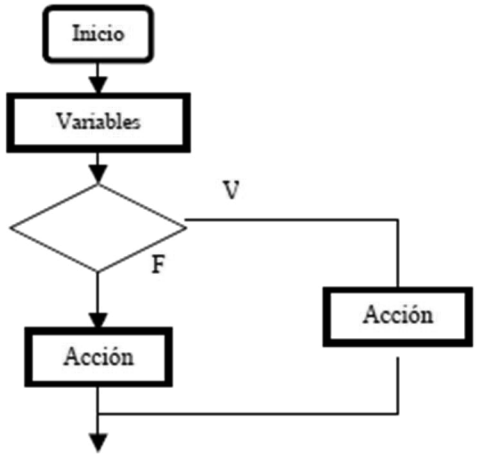


Figura 17: Diagrama de Flujo de alternativa doble

En español:

```
Si <condición>
entonces <acción S1>
sino <acción S2>
Fin_Si
```

En inglés:

```
If <condición>
then<acción>
else<acción S2>
End_if
```

Ejemplo:

```
INICIO
  BOOLEANO afueraLlueve = verdadero
  SI (afueraLlueve es verdadero)
  ENTONCES:
    me quedo viendo películas
  SINO:
    salgo al parque a tomar mates
  FIN_SI
FIN
```

Alternativa de Decisión múltiple (según_sea, caso de/case)

Se utiliza cuando existen más de dos alternativas para elegir. Esto podría solucionarse por medio de estructuras alternativas simples o dobles, anidadas o en cascada. Sin embargo, se pueden plantear serios problemas de escritura del algoritmo, de comprensión y de legibilidad, si el número de alternativas es grande.

En esta estructura, se evalúa una condición o expresión que puede tomar n valores. Según el valor que la expresión tenga en cada momento se ejecutan las acciones correspondientes al valor.

PSEUDOCÓDIGO:

Según sea <expresión>

<Valor1>: <acción1>

<valor2>: <acción2>

.....

[<otro>: <acciones>]

fin según

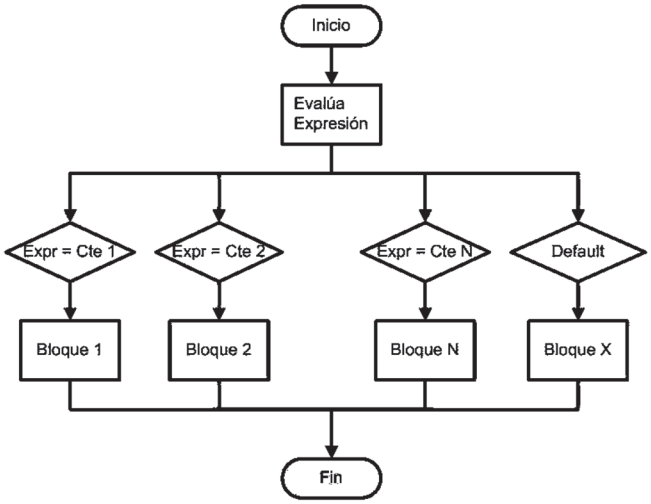


Figura 18: Diagrama de Flujo de Decisión Múltiple

Ejemplo en pseudocódigo:

```
INICIO
ENTERO posicionDeLlegada = 3
SEGUN SEA posicionDeLlegada
  1: entregar medalla de oro
  2: entregar medalla de plata
  3: entregar medalla de bronce
  otro: entregar mención especial
FIN
```

Es importante mencionar que la estructura anterior puede ser escrita usando los condicionales vistos anteriormente de la siguiente forma:

```
INICIO
ENTERO posicionDeLlegada = 3
SI (posicionDeLlegada = 1)
  ENTONCES:
    entregar medalla de oro
  SINO:
    SI (posicionDeLlegada = 2)
    ENTONCES:
      entregar medalla de plata
    SINO:
      SI (posicionDeLlegada = 3)
      ENTONCES:
        entregar medalla de bronce
      SINO:
        entregar mención especial
      FIN_SI
    FIN_SI
  FIN_SI
FIN
```

Podemos ver que usar condiciones anidadas podemos resolver el mismo problema, pero la estructura resultante es mucho más compleja y difícil de modificar.

Estructura repetitiva o iterativa

Durante el proceso de creación de programas, es muy común, encontrarse con que una operación o conjunto de operaciones deben repetirse muchas veces. Para ello es importante conocer las estructuras de algoritmos que permiten repetir una o varias acciones, un número determinado de veces.

Las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominan BUCLES. Y cada repetición del bucle se llama iteración.

Todo bucle tiene que llevar asociada una condición, que es la que va a determinar cuándo se repite el bucle y cuando deja de repetirse.

Un bucle se denomina también lazo o loop. Hay que prestar especial atención a los bucles infinitos, hecho que ocurre cuando la condición de finalización del bucle no se llega a cumplir nunca. Se trata de un fallo muy típico, habitual sobre todo entre programadores principiantes.

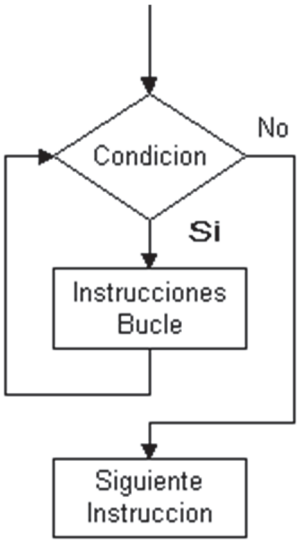
Hay distintos tipos de bucles:

- Mientras, en inglés: While
- Hacer Mientras, en inglés: Do While.
- Para, en inglés: For

Estructura mientras (while, en inglés)

Esta estructura repetitiva “mientras”, es en la que el cuerpo del bucle se repite siempre que se cumpla una determinada condición.

Diagrama de Flujo	Pseudocódigo:

 *Figura 19: Diagrama de Flujo de la estructura “mientras”*	`Mientras <cond> hacer`
	`  <acciones>`
	`fin mientras`

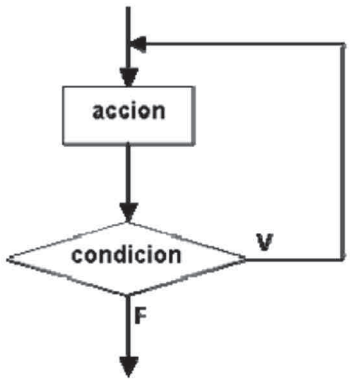
Ejemplo:

```
INICIO
  BOOLEANO tanqueLleno = falso
  MIENTRAS (tanqueLleno == falso)
  HACER:
    llenar tanque
  FIN_MIENTRAS
  // el tanque ya está lleno :)
FIN
```

Estructura hacer-mientras (do while, en inglés)

Esta estructura es muy similar a la anterior, sólo que a diferencia del while el contenido del bucle se ejecuta siempre al menos una vez, ya que la evaluación de la condición se encuentra al final. De esta forma garantizamos que las acciones dentro de este bucle sean llevadas a cabo, aunque sea una vez independientemente del valor de la condición.

Diagrama de Flujo	Pseudocódigo:

 *Figura 20: Diagrama de flujo de "hacer-mientras"*	`Hacer`
	`  <acciones>`
	`Mientras <cond>`
	`Fin mientras`

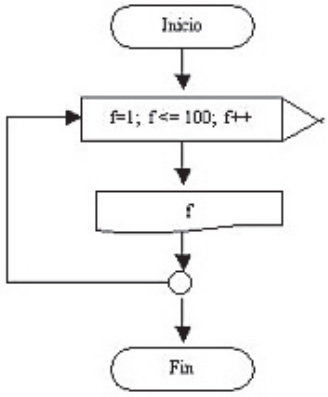
Ejemplo:

```
INICIO
BOOLEANO llegadaColectivo=false;
  HACER: esperar en la parada
  MIENTRAS (llegadaColectivo == false)
  FIN_MIENTRAS
FIN
```

Estructura para (for, en inglés)

La estructura for es un poco más compleja que las anteriores y nos permite ejecutar un conjunto de acciones para cada elemento de una lista, o para cada paso de un conjunto de elementos. Su implementación depende del lenguaje de programación, pero en términos generales podemos identificar tres componentes: la inicialización, la condición de corte y el incremento.

Diagrama de Flujo	Pseudocódigo:

 *Figura 21: Diagrama de Flujo de "para"*	`Para (inicialización; condición de corte; incremento)`
	`  <acciones>`
	`Fin para`