Efrain Robles Pulido

Programación Estructurada

La programación estructurada utiliza las técnicas tradicionales del campo de programación. Hace un enfoque especifico que, normalmente, produce programas bien escritos y muy legibles, aunque no necesariamente un programa bien escrito y fácil de leer ha de ser estructurado.

La programación estructurada trata de escribir un programa de acuerdo con unas reglas y un conjunto de técnicas. Las reglas se basan en la modularización; es decir, en la división de un problema un subproblema más pequeño (módulos), que a su vez se pueden dividir en otros subproblemas. Cada subproblema (modulo) se analiza y se obtiene una solución para ese subproblema.

Una vez que se han resuelto los diferentes subproblemas o módulos se combinan todos ellos para resolver el problema global.

Técnicas de programación estructurada

Las técnicas de programación estructurada incluyen construcciones o estructuras (instrucciones) básicas de control.

- **Secuencia.**

Es la más sencilla de todas las estructuras, simplemente le indica al procesador que debe ejecutar de forma consecutiva una lista de acciones (que pueden ser, a su vez, otras estructuras de control). Para construir una secuencia de acciones basta con escribir una acción a continuación de la otra o, en algunos casos, se utiliza un operador al final de cada sentencia

- **Decisión** (también denominada *selección).*

Permite bifurcar el “flujo” del algoritmo en función de una expresión lógica. Se utilizan cuando en el desarrollo de la solución de un problema se debe tomar una decisión para establecer un proceso o un camino alternativo a seguir.

- **Bucles o lazos** (también denominada *repetición o iteración)*

Permite repetir una acción (o grupo de acciones) un n ´umero prefijado de veces o depender de la evaluación de una expresión lógica.

Efrain Robles Pulido

Las estructuras básicas de control especifican el orden en que se ejecutan las distintas instrucciones de un algoritmo o programa. Una construcción (estructura, instrucción o sentencia en la jerga de lenguajes de programación) es un bloque de instrucciones de un lenguaje y una de las operaciones fundamentales del lenguaje.

También existen, diferentes sentencias que especifican como saltar el orden secuencial, es decir, que la secuencia a ejecutar sea distinta de la siguiente en la secuencia. Llamada transferencia de control o control de flujo del programa.

La estructura de selección se implementa en uno de los tres formatos siguientes:

- Sentencia if (si): selección única
- Sentencia if-else (si-entonces-sino): selección doble
- Sentencia switch (según_sea): selección múltiple

La *estructura de repetición* se implementa en tres formatos diferentes

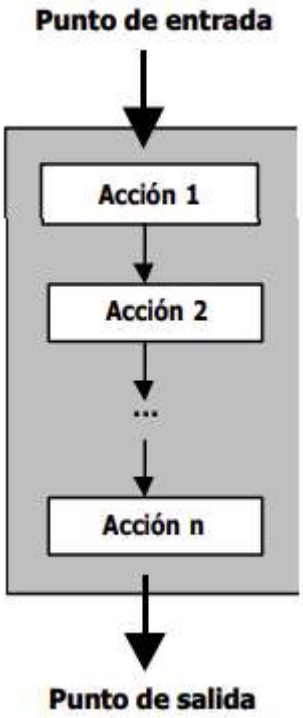
- Sentencia while (mientras)
- Sentencia do-while (hacer-mientras)
- Sentencia for (desde/para)

La *programación estructurada* promueve el uso de las tres sentencias de control:

- Secuencia
- Selección (sentencias, if, if-else, switch)
- Repetición (sentencias while, dowhile, for)

- **Secuencial**

Simplemente indica una secuencia de acciones a ser ejecutadas de forma consecutiva. La composición secuencial no es conmutativa.




```
leer a                    | read *, a
leer b                    | read *, b
c ← a + b                 | c = a + b
escribir c                | print *, c
```

Existe una notación alternativa, separando las sentencias mediante el carácter ;

```
leer a; leer b            | read *, a; read *, b
c ← a + b; escribir c     | c = a + b; print *, c
```

- **Decisión**

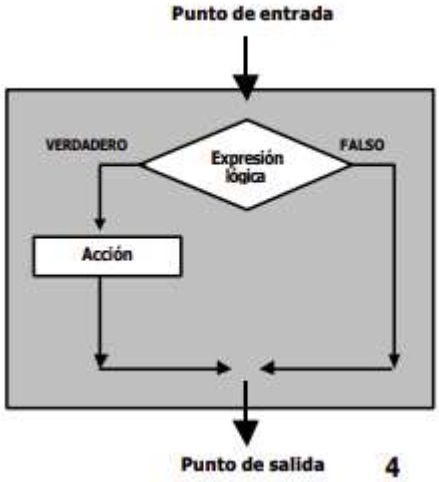
  - Selección simple

Evalúa una expresión lógica y ejecuta una acción (o grupo de acciones) si es cierta y no hace nada si es falsa.

```
si expresión lógica entonces | if (expresión lógica) then
  acciones                   |   acciones
fin_si                       | end if
```

- **Ejemplo:**

```
si radio>0 entonces          | if (radio>0) then
  longitud←2*pi*radio        |   longitud=2*pi*radio
fin_si                       | end if
```

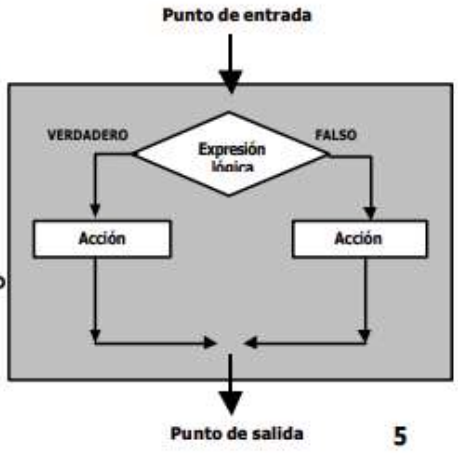


- Selección doble

Similar a la anterior, ejecuta una acción (o grupo de acciones) si la expresión es cierta y otra acción (o grupo) si es falsa.

```
si expresión lógica entonces
  acciones
si no
  acciones
fin_si
```

```
if (expresión lógica) then
  acciones
else
  acciones
end if
```



- **Ejemplo:**

```
si radio>0 entonces
  longitud←2·pi·radio
si no
  escribir 'Error'
fin_si
```

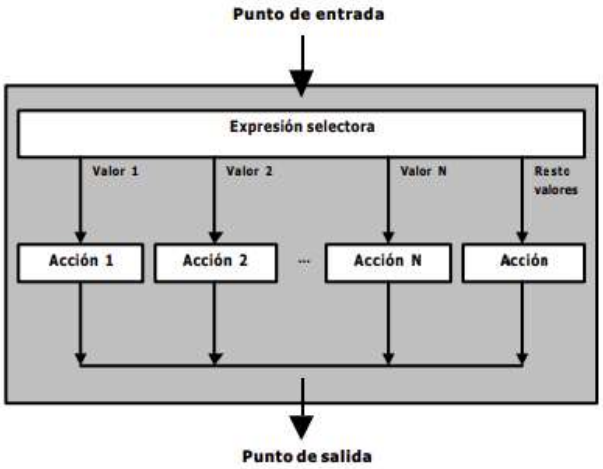
```
if (radio>0) then
  longitud=2*pi*radio
else
  print *, 'Error'
end if
```


- Selección múltiple

Evalúa una expresión que pueda tomar n valores (enteros, caracteres y lógicos pero nunca reales) y ejecuta una acción o grupo de acciones diferente en función del valor tomado por la expresión selectora.

```
segun expresión
  caso valor1:
    acción 1
  caso valor2:
    acción 2
  ...
  caso valorN:
    acción n
  otro caso:
    acción
fin según
```

```
select case (expresión)
  case (valor1)
    acción 1
  case (valor2)
    acción 2
  ...
  case (valorn)
    acción n
  case default
    acción
end select
```



- **Ejemplo:**

```
segun mes
  caso 1,3,5,7,8,10,12:
    escribir '31'
  caso 4,6,9,11:
    escribir '30'
  caso 2:
    escribir '28'
  otro caso:
    escribir 'Mes incorrecto'
fin según
```

```
select case (mes)
  case (1,3,5,7,8,10,12)
    print *, '31'
  case (4,6,9,11)
    print *, '30'
  case (2)
    print *, '28'
  case default
    print *, 'Mes incorrecto'
end select
```

❖ **Bucles o Lazos**

- Sentencia do

Permite repetir la ejecución de una acción o de un grupo de acciones un número determinado de veces.

```
desde indice←inicio hasta fin [con paso valor] hacer
  acción
fin desde
```

```
do indice=inicio, fin, paso
  acción
end do
```

El funcionamiento de la estructura es el siguiente:

- En primer lugar, se asigna a la variable indice el valor de inicio.
- El bucle se ejecuta mientras indice no alcance el valor de fin.
- En cada iteración el valor de indice es incrementado según el paso indicado y se ejecuta la acción o grupo de acciones encerrados en el bucle.
- En caso de que no se indique ningún paso el que se empleará será +1.

- **Ejemplos:**

```
desde n←1 hasta 10 hacer
  escribir n
fin desde
```

```
do n=1, 10
  print *, n
end do
```

```
desde n←10 hasta 1 hacer
  escribir n
fin desde
```

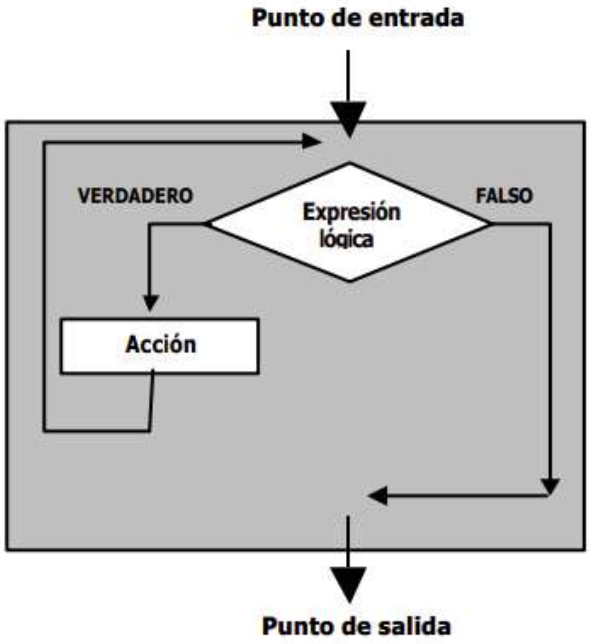
```
do n=10, 1
  print *, n
end do
```

- Sentencia while (mientras)

Repite una acción o grupo de acciones mientras una expresión lógica sea cierta.

```
mientras expresión lógica hacer
  acción
fin desde
```

```
do while (expresión lógica)
  acción
end do
```



Un aspecto muy importante de la presente estructura de control es que si la expresión lógica es inicialmente falsa el bucle no se ejecuta ni una sola vez; es decir, la estructura mientras iterará 0 ó más veces.

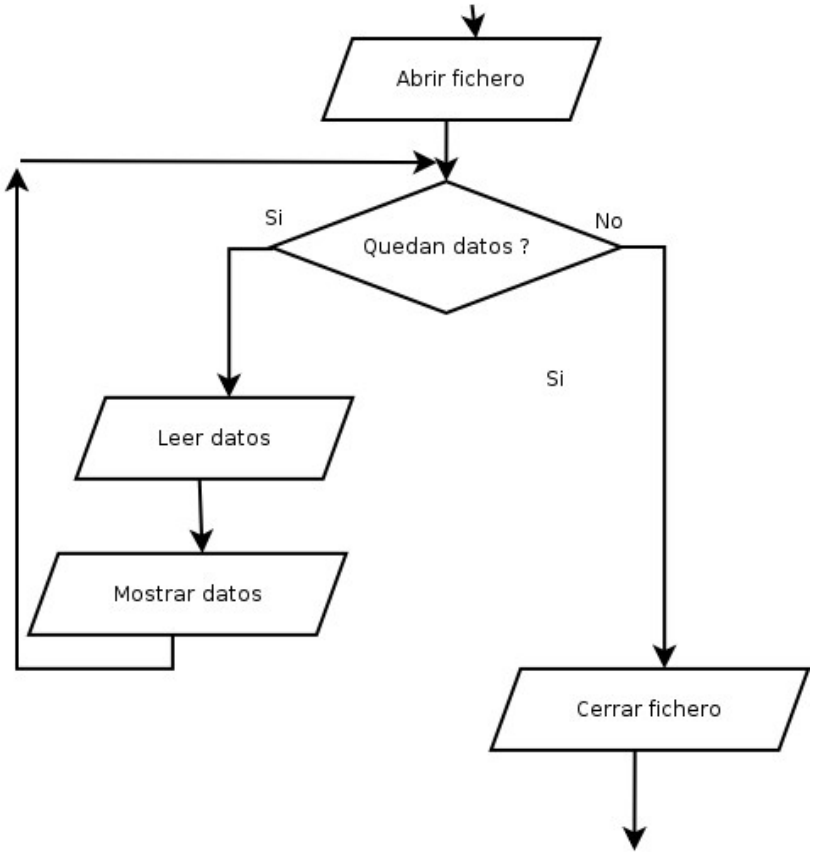
- Sentencia for (desde/para)

Se utiliza generalmente cuando la repetición está definida. Esta estructura maneja todos los detalles de la repetición controlada por contador

```
for ( expresion 1; expresion 2; expresion 3)
        instruccion 1
```

donde:

expresion 1	es el nombre de la variable de control y su valor de inicio.
expresion 2	es la que evalúa la condición de control.
instruccion 1	son las instrucciones que han de repetirse.
expresion 3	instrucción de incremento o decremento de la variable de control.



Bibliografía

Joyanes Aguilar, L., (2014). Programación en C, C++, Java y UML. McGraw Hill. 2a. Edición

Algorítmica y Lenguajes de Programación. (2021). Estructuras de Control. Consultado el 27 de agosto de 2021 en http://di002.edv.uniovi.es/~dani/asignaturas/transparencias-leccion4.PDF