

Tema 5

Estructuras Básicas De La Programación Imperativa

Programación estructurada

La programación estructurada es una metodología de programación que fundamentalmente trata de construir programas que sean fácilmente comprensibles. Un programa no solamente debe funcionar correctamente, sino que además debe estar escrito de manera que se facilite su comprensión posterior.

Esta metodología está basada en la técnica de desarrollo de programas por *refinamientos sucesivos*. Inicialmente, se plantea la operación global a realizar por el programa, y se descompone en otras más sencillas. A su vez, estas últimas vuelven a ser descompuestas nuevamente en otras todavía más elementales. Este proceso de descomposición continúa hasta que todo se puede escribir utilizando las estructuras básicas disponibles en el lenguaje de programación que se está empleando.

- **Representación de la estructura de un programa:** La estructura de los programas imperativos se representa tradicionalmente mediante *diagramas de flujo*. Estos diagramas contiene dos elementos básicos, correspondientes a *condiciones* y *acciones*. Las condiciones equivalen a preguntas a las que se puede responder SI o NO.
- **Secuencia:** La estructura más sencilla para emplear en la descomposición es utilizar una *secuencia* de acciones o partes que se ejecutan de forma sucesiva.
- **Selección:** La estructura de *selección* consiste en ejecutar una acción u otra, dependiendo de una determinada condición que se analiza a la entrada de la estructura.
- **Iteración:** La *iteración* es la repetición de una acción mientras que se cumpla una determinada condición. La estructura de iteración más general es aquella en que la condición se analiza a la entrada de la estructura y antes de iniciar cada nueva repetición.
- **Estructuras anidadas:** Cualquier parte o acción del programa puede a su vez estar constituida por cualquiera de las estructuras descritas. Por tanto, el anidamiento entre ellas puede ser tan complejo como sea necesario.

Expresiones condicionales

Para poder utilizar las estructuras de selección e iteración es necesario expresar las condiciones que controlan ambas estructuras. Esto se realiza mediante la construcción de expresiones condicionales.

Operador		Comparación	
>		Mayor que	
<		Menor que	
=		Igual a	
>=		Mayor o igual que	
<=		Menor o igual que	
<>	#	Diferente a	
Operador		Operación lógica	
AND	&	Conjunción	
OR		Disyunción	
NOT	~	Negación	
Orden		Operadores	
1º	Negación	NOT	~
2º	Multiplicativos	* / DIV MOD	AND &
3º	Aditivos	+ -	OR
4º	Comparación	> < = >= <= <> #	

Estructuras básicas en Modula-2

- **Secuencia:** Para programar una secuencia de acciones se concatan las sentencias que forman la secuencia de acciones, separadas unas de otras por un punto y coma (;).

```
AcciónA;
AcciónB
```

- **Sentencia IF:** La selección consiste en evaluar la expresión de Condición1, y a continuación ejecutar o bien la AcciónA (si se cumple la condición), o bien la AcciónB (si la condición no se cumple). Se pueden encadenar varias condiciones si van en cascada, es decir, en su salida del NO llevan una nueva expresión Condición2 y sus acciones Acción2 correspondientes.

```
IF Condición1 THEN
  AcciónA
[ ELSIF Condición2 THEN
  Acción2 ]
[ ELSE
  AcciónB ]
END
```

- **Sentencia WHILE:** La estructura de iteración se consigue mediante la sentencia WHILE, que tiene el siguiente formato:

```
WHILE Condición DO
    Acción
END
```

Mientras la expresión Condición resulta cierta, se ejecuta la Acción de forma repetitiva. Cuando el resultado es falso finaliza la ejecución de la sentencia. Si la Condición resulta falsa en la primera evaluación, la acción no se ejecuta nunca.

- **Sentencia FOR:** La sentencia FOR es una variación de la estructura iteración cuando se conoce el número de iteraciones a realizar, el formato es el siguiente:

```
FOR Var := ValIni TO ValFin
    BY ValInc DO
    Acción
END
```

Para todo valor de la variable Var desde un valor inicial ValIni hasta un valor final ValFin ejecutar la Acción e incrementar la variable Var en un incremento ValInc (positivo o negativo) y comprobar si la variable Var alcanza el valor de ValFin.