

# Tema 6

## Metodología De Desarrollo De Programas (II)

### Desarrollo por refinamientos usando selección y bucles

- **Metodología de desarrollo de un esquema de selección:** Un esquema de selección consiste en plantear una acción compuesta como la realización de una acción entre varias posibles, dependiendo de ciertas condiciones. Es decir, se trata de elegir para realizar una sola entre varias posibles alternativas.  
Para desarrollar un esquema de selección debemos identificar sus elementos componentes. Por tanto habrá que:
  1. Identificar cada una de las alternativas del esquema, y las acciones correspondientes.
  2. Identificar las condiciones para seleccionar una alternativa u otra.
- **Metodología de desarrollo de un esquema de iteración:** Una iteración o bucle consiste en la repetición de una acción o grupo de acciones hasta conseguir el resultado deseado. Para desarrollar un esquema de iteración dentro de un programa deberemos identificar cada uno de sus elementos componentes. Al hacerlo hay que identificar simultáneamente las variables adecuadas para almacenar la información necesaria.  
En líneas generales se podría proceder de la siguiente manera:
  1. Identificar las acciones útiles a repetir, y las variables necesarias. Precisar el significado de estas variables al comienzo y final de cada repetición.
  2. Identificar cómo actualizar la información al pasar de cada iteración a la siguiente. Puede ser necesario introducir nuevas variables.
  3. Identificar la condición de terminación. Puede ser necesario introducir nuevas variables e incluso acciones adicionales para mantenerlas actualizadas.
  4. Identificar los valores iniciales de las variables, y si es necesario alguna acción para asignárselos antes de entrar en el bucle.

### Verificación de programas

Uno de los objetivos de la programación es la *corrección*. Un programa es correcto si produce siempre resultados de acuerdo con la especificación del programa.

En la práctica, la verificación de un programa se hace muchas veces mediante ensayos. Un *ensayo* consiste en ejecutar el programa con unos datos preparados de antemano y para los cuales se sabe cuál ha de ser el resultado a obtener. Si al ejecutar el programa no se obtienen los resultados esperados, se sabrá que hay algún error, y el programa se examina para determinar la causa del error y eliminarla. Este proceso se llama *depuración*.

Si la ejecución produce los resultados esperados, entonces el ensayo no suministra ninguna información, ya que puede que el programa sea correcto o incorrecto para otros valores que no se han ensayado.

- **Corrección parcial y total:** Usando las expresiones y reglas de la lógica, y conociendo la semántica (significado) de las acciones, es posible demostrar si un programa es o no correcto. Para programas que siguen el modelo imperativo el proceso de demostración se realiza en dos partes:
  1. **Corrección parcial:** Si el programa termina el resultado es correcto.
  2. **Corrección total:** Para todo dato de entrada válido el programa termina.
- **Razonamiento sobre sentencias de asignación:** Para analizar el comportamiento de un fragmento de programa correspondiente a una sentencia de asignación, comenzaremos por anotar delante de dicha sentencia todas las condiciones que sabemos que se cumplen inmediatamente antes de ejecutarla. A continuación anotaremos detrás de la sentencia las condiciones que podamos demostrar que se cumplen después de su ejecución, y que serán las siguientes:
  - Las condiciones anteriores en las que n intervenga la variable asignada.
  - La condición de que la variable tiene el valor asignado.

Las condiciones se suelen anotar entre llaves ({}).

- **Razonamiento sobre el esquema de selección:** Para analizar el comportamiento de un fragmento de programa correspondiente a un esquema de selección, comenzaremos por anotar delante de dicho esquema las condiciones que sabemos que se cumplen inmediatamente antes de examinar la condición. Puesto que la de selección decide la continuación por una u otra vía de las dos posibles, deduciremos que al comienzo de la alternativa "SI" se cumplirán las condiciones iniciales y además la condición de selección, y que al comienzo de la alternativa "NO" se cumplirán las condiciones iniciales y no se cumplirá la condición de selección.
- **Razonamiento sobre bucles: invariante, terminación:** Para analizar el comportamiento de un fragmento de programa correspondiente a un esquema de iteración habremos de identificar las condiciones que deben cumplirse siempre inmediatamente antes de examinar la condición de repetición. Estas condiciones constituyen el llamado *invariante* del bucle. Razonando como con el esquema de selección, deduciremos que al comienzo de cada repetición de la acción del bucle habrá de cumplirse el invariante y además la condición de repetición; y que al terminar las repeticiones y salir del bucle se cumplirá el invariante y además no se cumplirá la condición de repetición.

## **Eficiencia de programas. Complejidad algorítmica**

- **Medidas de eficiencia:** La eficiencia de un programa se analiza en función de:
  - El tiempo que tarda en ejecutarse un programa.
  - La cantidad de memoria usada para almacenar datos.

La *eficiencia en tiempo* nos indica que un programa será tanto más eficiente cuanto menos tiempo tarde en ejecutarse. También que establecer el tamaño de los datos o *tamaño del problema*, para, en función de ella, establecer la medida de la eficiencia del programa que los procesa. La función que da el tiempo de ejecución según el tamaño del problema se dice que mide la *complejidad algorítmica* del programa.

- **Análisis de programas:** La determinación de la eficiencia (o complejidad) de un programa se hace analizando los siguientes elementos:

1. Cuánto tarda en ejecutarse cada instrucción básica del lenguaje utilizado.
2. Cuántas instrucciones de cada clase se realizan durante una ejecución del programa.

Realizando el análisis del comportamiento del programa en el peor caso posible, el análisis de complejidad (número de instrucciones ejecutadas) de los esquemas básicos de los programas es el siguiente:

1. La complejidad de un esquema secuencial es la suma de las complejidades de sus acciones componentes.
  2. La complejidad de un esquema de selección equivale a la de la alternativa más compleja, es decir, de ejecución más larga, más la complejidad de la evaluación de la condición de selección.
  3. La complejidad de un esquema de iteración se obtiene sumando la serie correspondiente al número de instrucciones en las repeticiones sucesivas.
- **Crecimiento asintótico:** La forma en que crece el tiempo empleado por un proceso para tamaños grandes se dice que es su *comportamiento asintótico*, y se representa mediante la notación:  $O(f(n))$  siendo  $n$  el tamaño del problema,  $f$  la forma o función del crecimiento asintótico, y  $O$  (que se lee O-mayúscula) indica orden de crecimiento. Algunas formas de crecimiento típicas son:

Función	Resultado
$O(1)$	Complejidad constante. Ideal.
$O(\log n)$	Crecimiento logarítmico. Muy eficiente.
$O(n)$	Complejidad lineal. Muy eficiente.
$O(n \log n)$	Eficiente.
$O(n^2)$	Complejidad cuadrática. Menos eficiente.
$O(n^k)$	Complejidad polinómica. Poco eficiente
$O(2^n)$	Complejidad exponencial. Muy poco eficiente.