Fundamentos de Programación – Introducción a la Programación:
2.- PROCESO DE CREACIÓN DE UN PROGRAMA

Maider Huarte Arrayago

Fundamentos de Programación – Introducción a la Programación:

2.- PROCESO DE CREACIÓN DE UN PROGRAMA

2.- PROCESO DE CREACIÓN DE UN PROGRAMA

El proceso de creación de un programa, desde el momento en que se decide que hay que hacerlo hasta que se consigue, pasa por diferentes fases, todas relacionadas entre sí. La siguiente figura muestra esas fases y sus relaciones:

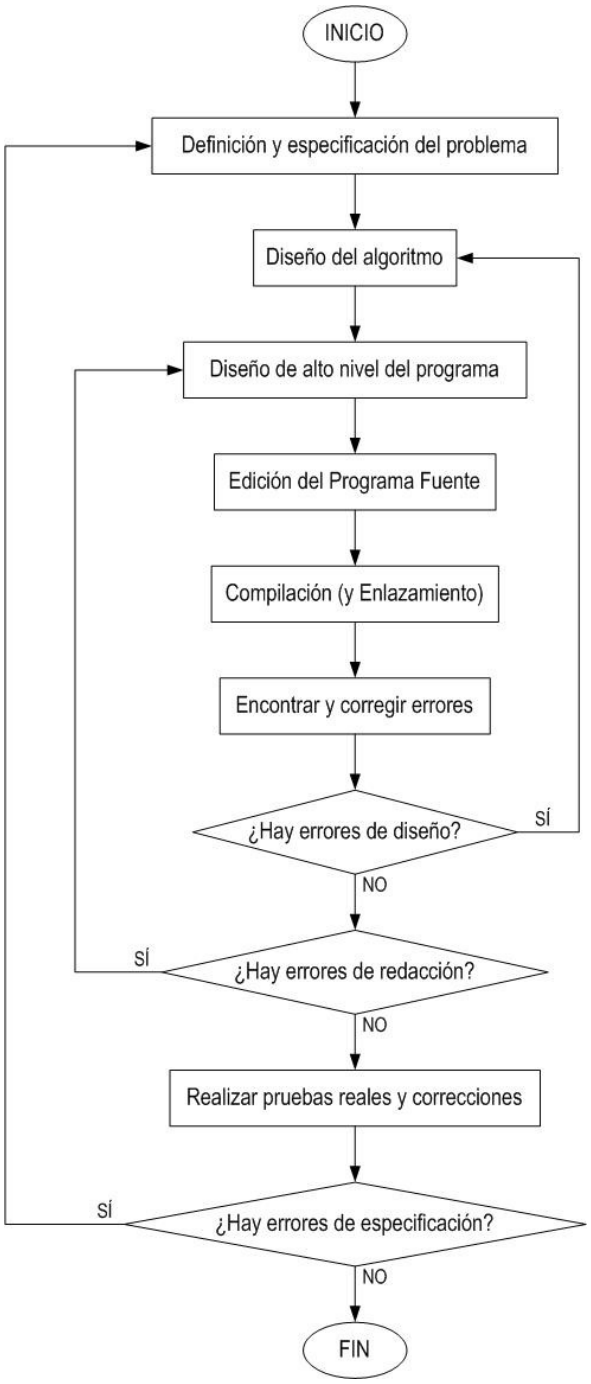


Figura 1: Proceso de creación de un programa

2.1.- DEFINICIÓN Y ESPECIFICACIÓN DEL PROBLEMA

Esta es la primera fase en el proceso de creación de un programa. En esta fase, se estudia el problema que hay que resolver mediante el programa que se va a crear.

Hay que especificar bien cuáles son los requerimientos del problema, y los resultados buscados.

Es conveniente documentar los resultados de esta fase, para posibles reclamaciones posteriores del cliente.

2.2.- DISEÑO DEL ALGORITMO

Los pasos a seguir por el computador en la resolución de un problema, conforman un algoritmo.

Habrá que consultar primero, teniendo en cuenta las especificaciones obtenidas en la fase anterior, si ya existe algún algoritmo resolutorio, o tendremos que inventarlo nosotros.

A la hora de diseñar un algoritmo, se suele usar la técnica *divide y vencerás*. Con esta técnica, lo que se hace es desgranar el problema en otros problemas más sencillos de resolver, y cada uno de esos problemas, a su vez, en otros problemas aún más fáciles de resolver.

La representación gráfica de un algoritmo en esta asignatura, ha de seguir una serie de normas, que se indican a continuación:

- El INICIO y el FIN de un algoritmo, se indican escribiendo esas palabras dentro de elipses.

- Los diferentes estados por los que se pasa en un algoritmo, se indican mediante rectángulos o rombos:
  - Rectángulos: Son estados en los que se ejecutan pasos sencillos.
    - A un rectángulo se puede **llegar desde más de un estado** (desde varios rectángulos o rombos).
    - Sin embargo, desde un rectángulo sólo se puede **ir a otro estado** único (rectángulo o rombo).
  - Rombos: Son estados en los que se toma una decisión binaria, es decir, se plantea una pregunta y se resuelve con un SÍ o un NO.
    - A un rombo se puede **llegar desde más de un estado** (desde varios rectángulos o rombos)

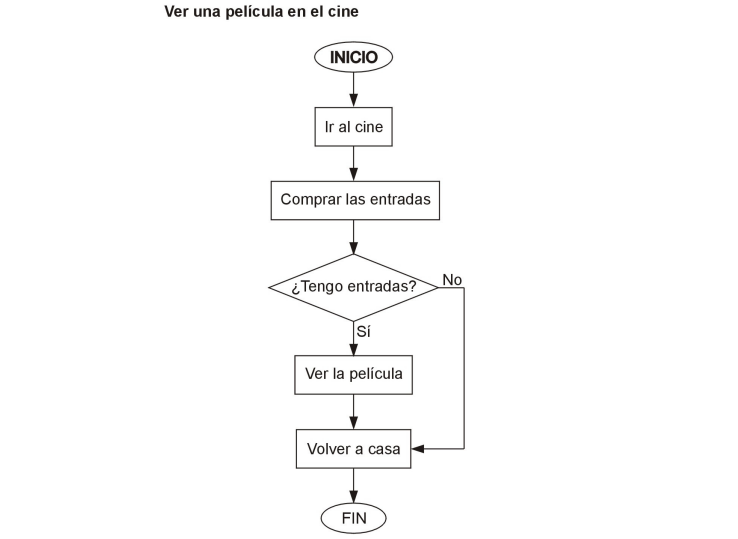
- Sin embargo, desde un **rombo sólo se puede ir a otros dos estados** (rectángulo o rombo); uno de ellos será el estado correspondiente al SÍ y el otro el correspondiente al NO, de forma que habrá que indicar por escrito cuál es cuál.

- La indicación de ir de un estado a otro, se hace mediante flechas, de forma que la punta de la misma indica el estado destino. En el caso de los estados de toma de decisión (los rombos), siempre hay dos flechas de salida, una hacia el estado del SÍ y otra hacia el estado del NO. En cada una de esas flechas, habrá que indicar a qué estado van.

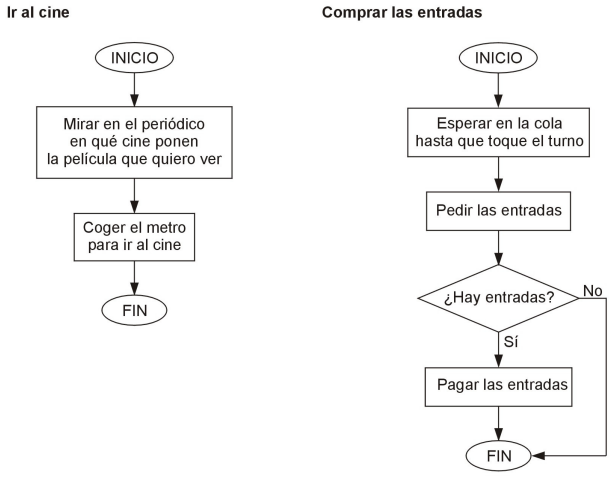
Ejemplo:

Problema: Ver una película en el cine.

ALGORITMO NIVEL 1



ALGORITMOS NIVEL 2



La solución al problema de ir a ver una película al cine, se ha diseñado con un algoritmo de 5 pasos. Los 2 primeros pasos, se han resuelto mediante otros dos algoritmos, mientras que los demás no ha hecho falta, ya que son suficientemente sencillos como para ser llevados a cabo sin ninguna otra explicación.

En programación, al diseñar algoritmos, se va bajando de nivel de complejidad hasta encontrar problemas particulares que sean sencillos de resolver, mediante pasos directamente programables, en el Lenguaje de Programación seleccionado. NO han de aparecer, sin embargo, expresiones propias de ese lenguaje de programación. Hay que tener en cuenta que el algoritmo depende más de la técnica de programación que del lenguaje en sí, y que el mismo algoritmo puede ser programado con lenguajes de programación diferentes que sigan la misma técnica de programación.

2.3.- DISEÑO DE ALTO NIVEL DEL PROGRAMA

En esta fase, se plasma lo indicado por el algoritmo diseñado en la fase anterior, en instrucciones escritas en el lenguaje de programación seleccionado. Para ello, independientemente del lenguaje de programación, se sigue un formato general que tiene tres secciones:

- 1ª Sección: Descripciones
- 2ª Sección: Programa Principal
- 3ª Sección: Funciones (subprogramas)

2.3.1.- 1ª Sección: Descripciones

En esta primera sección, se hace la descripción de los tipos de datos y funciones que se van a usar en el resto del programa.

2.3.2.- 2ª Sección: Programa Principal

En esta sección, se refleja a través de instrucciones en el lenguaje de programación seleccionado, el algoritmo de 1er nivel obtenido en la fase anterior.

Hay que tener en cuenta, que las instrucciones se ejecutan, a no ser que se indique lo contrario, de forma secuencial en el computador. Esto, como ya se vio en el tema anterior (1.2.1.2.- Unidad de Control), se controla mediante el Contador de Programa de la Unidad de Control de la CPU.

2.3.3.- 3ª Sección: Funciones

Si en la fase anterior se han generado más de un nivel de algoritmos, es en esta sección donde se codifican en instrucciones del lenguaje de programación seleccionado, todos los algoritmos de niveles diferentes al 1, normalmente, de forma independiente al nivel al que pertenecen.

Siguiendo con el ejemplo del punto anterior (**2.2.- DISEÑO DEL ALGORITMO**), en esta sección se codificarían los algoritmos llamados *Ir al cine* y *Comprar las entradas.*

2.4.- EDICIÓN DEL PROGRAMA FUENTE

Mediante el programa de edición adecuado, se escribe el resultado de la fase anterior, en un fichero apropiado para la herramienta de traducción a usar. Así, se consigue un Fichero Fuente.

2.5.- COMPILACIÓN (Y ENLAZAMIENTO)

El Fichero Fuente conseguido en la fase anterior, se traduce, generalmente mediante un Compilador, obteniendo un Fichero Objeto. Si ese Fichero Objeto no fuera ejecutable, habría que usar, además, un Enlazador para conseguir un Fichero Ejecutable a partir del Fichero Objeto.

Podría ocurrir que el Fichero Fuente tuviera errores sintácticos. En ese caso, el Compilador no crea ningún Fichero Objeto, sino que indica la presencia de esos errores. A veces incluso el Compilador da información más o menos detallada de en qué punto del Fichero Fuente está cada error y de cómo arreglarlo.

Hay que tener en cuenta pues, que para que se cree un Fichero Objeto como consecuencia de la compilación de un Fichero Fuente, ese Fichero Fuente no ha de tener ningún error sintáctico.

Si en vez de usarse un Compilador, se usara un Intérprete, éste dejaría de ejecutar el programa en cuanto encontrara un error sintáctico.

2.6.-ENCONTRAR Y CORREGIR ERRORES

En esta fase, se dispone ya de un Fichero Ejecutable, y se ejecuta, comprobando que cumple todas las especificaciones y que consigue los resultados definidos en la 1ª fase del proceso (**2.1.- DEFINICIÓN Y ESPECIFICACIÓN DEL PROBLEMA**).

Normalmente, los programas, sobre todo si son muy largos o complicados, no salen bien a la primera. Entonces, será necesario volver a alguna fase anterior, dependiendo del tipo de error del que se trate.

En el proceso de creación de un programa, se pueden dar 4 tipos de errores:

- **Errores sintácticos**: Son errores que se producen al escribir mal alguna instrucción. Aparecen y se corrigen en la anterior Fase de Compilación.
- **Errores de redacción**: Son errores que se producen al redactar una instrucción sintácticamente bien, pero conceptualmente mal. Para corregirlos, será necesario volver a la anterior fase de edición del Programa Fuente (**2.4.- EDICIÓN DEL PROGRAMA FUENTE**).

  Ejemplo:

  En un programa, podríamos escribir que queremos comparar si a>b, pero lo que realmente queremos comparar es si b>a. Es un error conceptual que se ha producido cuando hemos redactado los nombres de las variables al revés. El Compilador no va a considerarlo como error, porque no puede saber qué es lo que realmente queremos.

- **Errores de diseño**: Si el algoritmo usado para resolver el problema no es el adecuado, se producen errores de diseño. Será necesario volver a la fase de diseño del algoritmo para corregirlos (**2.2.- DISEÑO DEL ALGORITMO**).
- **Errores de especificación**: Son errores que aparecen en la siguiente fase de realizar pruebas reales y corregir errores.

2.7.- REALIZAR PRUEBAS REALES Y CORREGIR ERRORES

En esta fase, cuando ya no hay errores sintácticos, de redacción ni de diseño, se pone el programa en manos del cliente.

Aquí, también se pueden dar errores. Serán errores de especificación, que se producen porque las especificaciones y definiciones de la primera fase del proceso no se hicieron bien en su momento. La razón puede ser que el cliente no explicó bien el problema que quería resolver con el programa, o

que el diseñador no lo entendió bien. En cualquier caso, será necesario volver a la primera fase de definición y especificación del problema (**2.1.- DEFINICIÓN Y ESPECIFICACIÓN DEL PROBLEMA**), volviendo a realizar todo el proceso.

2.8.- DOCUMENTACIÓN

Hacer la documentación de un programa, es casi tan importante como el proceso de creación del programa en sí. La documentación ha de reflejar toda la información existente sobre el programa, y ha de constar, por lo menos, de los siguientes documentos:

- **Listados del programa**: Los diferentes Ficheros Fuente que conforman el programa.
- **Organigramas**: Documentos que ilustran los algoritmos usados en la resolución del problema.
- **Manual del Usuario**: Documento explicativo de cómo se usa el programa, orientado a usuarios que no tienen porqué tener conocimientos de informática.
- **Manual de Mantenimiento**: Es un documento orientado a una persona con conocimientos técnicos suficientes como para poder entender los organigramas y listados del programa, pero que no ha participado en el proceso de creación del mismo.
  Los programas suelen adaptarse con el tiempo a nuevas necesidades, y esas adaptaciones no tienen porqué hacerlas las mismas personas que hicieron el proceso de creación del programa original. Este Manual es necesario para ayudar a que otras personas puedan hacer esas adaptaciones.