

Capítulo 1

Conceptos básicos



Al término de este capítulo,
el alumno será capaz de

- Identificar, describir y diferenciar conceptos básicos de la programación; así como comprender la importancia de las etapas para la elaboración de programas.

Contenido

- 1.1 Definición de lenguaje de programación
- 1.2 Definición de algoritmo
- 1.3 Definición de programa de computadora
- 1.4 Etapas o pasos en la creación de un programa

1.1 Definición de lenguaje de programación

El lenguaje de programación es la combinación de símbolos y reglas que permiten la elaboración de programas con los cuales la computadora puede realizar tareas o resolver problemas de manera eficiente.

Los lenguajes de programación se clasifican en:

1. *Lenguaje máquina*. Las instrucciones son directamente entendibles por la computadora y no necesitan traductor para que la CPU (unidad de procesamiento central) pueda entender y ejecutar el programa. Utiliza un código binario (0 y 1), se basa en bits (abreviatura inglesa de dígitos binarios).
2. *Lenguaje de bajo nivel (ensamblador)*. Las instrucciones se escriben en códigos alfabéticos conocidos como mnemotécnicos.
3. *Lenguaje de alto nivel*. Es semejante al lenguaje humano (en general en inglés), lo que facilita la elaboración y comprensión del programa. Por ejemplo Basic, Pascal, Cobol, Fortran, C, etcétera.

1.2 Definición de algoritmo

Se denomina *algoritmo* al conjunto de pasos ordenados y finitos que permiten resolver un problema o tarea específica. Los algoritmos son independientes del lenguaje de programación y de la computadora que se vaya a emplear para ejecutarlo.

Todo algoritmo debe ser:

1. *Finito* en tamaño o número de instrucciones (tiene un primer paso y un último paso) y tiempo de ejecución (debe terminar en algún momento). Por lo tanto, debe tener un punto particular de inicio y fin.
2. *Preciso*. Debe tener un orden entre los pasos.
3. *Definido*. No debe ser ambiguo (dobles interpretaciones); si se ejecuta el mismo algoritmo el resultado siempre será el mismo, sin importar las entradas proporcionadas.
4. *General*. Debe tolerar cambios que se puedan presentar en la definición del problema.

Toda actividad que realizamos la podemos expresar en forma de algoritmo. Existen dos tipos de algoritmos, los que se desarrollan para ser ejecutados por una computadora, llamados *algoritmos computacionales*, y los que realiza el ser humano, es decir, *algoritmos no computacionales*; como ejemplos de éstos tenemos:

1. Cambiar un neumático (llanta) de un automóvil.
2. Preparar unos “huevos a la mexicana”.
3. Calcular el área de un triángulo.

Ejemplo



Un algoritmo para cambiar el neumático desinflado de un automóvil

1. Inicio¹.
2. Bajar la herramienta y el neumático (llanta) de repuesto del automóvil.
3. Aflojar los birlos del neumático pinchado.
4. Acomodar el gato.
5. Levantar el automóvil.
6. Quitar los birlos del neumático desinflado.
7. Quitar el neumático desinflado.
8. Colocar el neumático de repuesto.
9. Fijar los birlos del neumático de repuesto.
10. Bajar el automóvil.
11. Apretar en forma definitiva los birlos del neumático de repuesto.

(continúa)

¹ Existen autores que en este tipo de algoritmos (no computacionales) *no* utilizan el *Inicio* y el *Fin*, ambos son opcionales.

(continuación)

12. Quitar el gato.
13. Guardar el neumático desinflado y la herramienta.
14. Fin.

Si revisamos, este algoritmo es finito (tiene 12 pasos²) y tiene un orden.

Un algoritmo para preparar unos “huevos a la mexicana”

Ejemplo



1. Poner la sartén en la estufa.
2. Poner aceite en la sartén.
3. Encender la estufa.
4. Cortar cebolla, tomate y chile en pedazos pequeños.
5. Poner la cebolla, el tomate y el chile en la sartén.
6. Abrir los huevos y verterlos en un recipiente.
7. Batir los huevos.
8. Poner los huevos batidos en la sartén.
9. Revolver la cebolla, tomate y el chile con los huevos hasta que queden estos últimos cocidos.
10. Vaciarlos en un plato.

Este algoritmo también es finito (tiene 10 pasos) y algunos pasos pueden estar en otro orden, por ejemplo los cuatro primeros puntos pudieran estar en un orden diferente y seguiríamos teniendo el mismo resultado.

Un algoritmo para calcular el área de un triángulo

Ejemplo



1. Inicio.
2. Solicitar (leer) los datos (la base y la altura).
3. Multiplicar la base por la altura y el resultado dividirlo entre dos, y guardarlo en una variable.
4. Mostrar (imprimir) el resultado almacenado en la variable.
5. Fin.

Al igual que en los dos ejemplos anteriores, se cumplen todas las características, solamente que este último algoritmo no es una situación cotidiana de la vida sino un cálculo específico el cual tiene un resultado exacto, o sea un valor.

1.3 Definición de programa de computadora

Existen diferentes conceptos; sólo mencionaremos tres:

1. Es un algoritmo desarrollado en un determinado lenguaje de programación, para ser utilizado por la computadora; es decir, es una serie de pasos o instrucciones ordenadas y finitas que pueden ser procesadas por una computadora, a fin de permitirnos resolver un problema o tarea específica.
2. Secuencia de instrucciones mediante las cuales se ejecutan diferentes acciones de acuerdo con los datos que se desee procesar en la computadora.
3. Expresión de un algoritmo en un lenguaje preciso que puede llegar a entender una computadora.

² Sin considerar inicio y fin.

Como se comentó en la sección anterior, *no* todo algoritmo puede llegar a ser programa de computadora, debido a que existen algunos algoritmos que requieren ser realizados físicamente. Los programas que puede ejecutar una computadora son más de carácter de proceso lógico, por ejemplo el tercer algoritmo en el cual se realizan cálculos respectivos.

1.4 Etapas o pasos en la creación de un programa

Las fases para la creación de un programa son siete, aunque para algunos autores pueden describirse en sólo seis, pues omiten la primera porque es una etapa algo obvia. Las etapas se describen a continuación.

1.4.1 Definición del problema

Esta fase la proporciona el enunciado del problema, el cual requiere una definición clara y precisa (no debe ser ambiguo). Es importante que se entienda perfectamente lo que pretendemos que haga la computadora para poder continuar con la siguiente etapa.

1.4.2 Análisis del problema

Una vez que se ha comprendido lo que se desea que la computadora haga, la etapa de análisis es muy importante ya que en ésta se identifican tres factores indispensables:

1. Qué información se necesita para obtener el resultado deseado (datos de entrada).
2. Qué información se desea producir (datos de salida).
3. Los métodos y fórmulas que se necesitan para procesar los datos y producir esa salida.

1.4.3 Diseño y técnicas para la formulación de un algoritmo

La etapa de diseño se centra en desarrollar el algoritmo basándonos en las especificaciones de la etapa del análisis; podemos representar un algoritmo mediante el *diagrama de flujo* o el *pseudocódigo*.


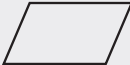

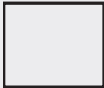

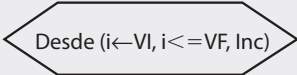


Diagrama de flujo

Un diagrama de flujo es la representación gráfica de un algoritmo; dicha representación gráfica se lleva acabo cuando varios símbolos (que indican diferentes procesos en la computadora) se relacionan entre sí mediante líneas que indican el orden en que se deben ejecutar las instrucciones para obtener los resultados deseados. Los símbolos utilizados han sido reglamentados por el Instituto Nacional de Normalización Estadounidense (ANSI, *American National Standards Institute*) y los apreciamos en la tabla 1.1.

Características de los diagramas de flujo:

- Todo diagrama debe tener un inicio y un fin.
- No se especifica la declaración de variables.
- Se deben usar solamente líneas de flujo horizontales y/o verticales.
- Se debe evitar el cruce de líneas utilizando los conectores.
- Se deben usar conectores sólo cuando sea necesario.
- No deben quedar líneas de flujo sin conectar.
- Se deben trazar los símbolos de manera que se puedan leer de arriba hacia abajo y de izquierda a derecha.
- Se debe evitar la terminología de un lenguaje de programación o máquina.
- Los comentarios se deben utilizar ya sea al margen o mediante el símbolo gráfico *comentarios* para que éstos sean entendibles por cualquier persona que lo consulte.
- Si el diagrama abarca más de una hoja es conveniente enumerarlo e identificar de dónde viene y a dónde se dirige.
- Sólo los símbolos de decisión pueden y deben tener más de una línea de flujo de salida.

Tabla 1.1 Símbolos gráficos más utilizados para dibujar diagramas de flujo de algoritmos

Símbolo	Descripción
	Inicio y final del diagrama de flujo.
	Entrada (leer) y salida de datos (imprimir).
	Símbolo de decisión. Indica la realización de una comparación de valores.
	Símbolo de proceso. Indica la asignación de un valor en la memoria y/o la ejecución de una operación aritmética.
	Líneas de flujo o dirección. Indican la secuencia en que se realizan las operaciones.
	Repetitiva desde número de iteraciones o repeticiones.
	Impresión ³
	Conectores

Pseudocódigo

El pseudocódigo es la combinación del lenguaje natural (español, inglés o cualquier otro idioma), símbolos y términos utilizados dentro de la programación. Se puede definir como un lenguaje de especificaciones de algoritmos.

El pseudocódigo se creó para superar las dos principales desventajas del diagrama de flujo: es lento de crear y difícil de modificar sin un nuevo redibujo. Por otra parte, el pseudocódigo es más fácil de utilizar ya que es similar al lenguaje natural. Al contrario de los lenguajes de programación de alto nivel no tiene normas que definan con precisión lo que es y lo que no es pseudocódigo, por lo tanto varía de un programador a otro.

En este documento todo pseudocódigo lo iniciaremos con la palabra reservada *principal* para especificar la función denominada (*main*) en lenguaje C. Todo programa, al igual que toda función, debe contener las palabras reservadas de inicio (`{`) y fin (`}`) delimitando las instrucciones.

1.4.4 Codificación

En la etapa de codificación se transcribe el algoritmo definido en la etapa de diseño en un código reconocido por la computadora; es decir, en un lenguaje de programación; a éste se le conoce como código fuente. Por ejemplo el lenguaje “C” es un lenguaje de programación y es el que utilizaremos en el presente curso.

³ Algunos autores utilizan el símbolo para salida de impresora; nosotros lo utilizaremos para salida de datos, a fin de diferenciarla de la entrada, ya que es válido en ambos casos.

1.4.5 Prueba y depuración

La prueba consiste en capturar datos hasta que el programa funcione correctamente. A la actividad de localizar errores se le llama *depuración*. Existen dos tipos de pruebas: de sintaxis y de lógica.

Las *pruebas de sintaxis* se ejecutan primero, son las más sencillas y las realiza el compilador del programa cada vez que se ejecuta el programa hasta que el código no presente errores, es decir que la sintaxis que requiere el lenguaje sea la correcta, de lo contrario el propio compilador va mostrando los errores encontrados para que se modifiquen y se pueda ejecutar el código; estos errores pueden ser falta de paréntesis, o puntos y comas o palabras reservadas mal escritas.

Las *pruebas de lógica* son las más complicadas ya que éstas las realiza el programador; consisten en la captura de diferentes valores y revisar que el resultado sea el deseado, es decir el programador tendría que modificar el código hasta que el programa funcione correctamente.

1.4.6 Documentación

Es la guía o comunicación escrita que permite al programador o al usuario conocer la funcionalidad del programa.

La documentación sirve para que el código fuente sea más comprensible para el programador o para otros programadores que tengan que utilizarlo, así como para facilitar futuras modificaciones (mantenimiento).

Hay dos tipos de documentación:

- *Interna*. Se generan en el mismo código y generalmente es mediante comentarios.
- *Externa*. Son los manuales y es independiente al programa. También puede ser la ayuda en el mismo software.

1.4.7 Mantenimiento

Se dice que un programa no se termina al 100%, ya que es necesario hacer algún cambio, ajuste o complementación para que siga funcionando correctamente; para llevarlo a cabo se requiere que el programa esté bien documentado.

Todos los programas tienen actualizaciones, por lo que surgen versiones diferentes. Por ejemplo: Windows 3.11, 95, 98, 2000, Millennium, Xp, Vista y 7.