Algoritmos. Definición

Un algoritmo se puede definir como una secuencia de instrucciones que representan un modelo de solución para determinado tipo de problemas. O bien como un conjunto de instrucciones que realizadas en orden conducen a obtener la solución de un problema.

Para realizar un programa es conveniente el diseño o definición previa del algoritmo. El diseño de algoritmos requiere creatividad y conocimientos profundos de la técnica de programación. Luis Joyanes, programador experto y autor de muchos libros acerca de lógica y programación nos dice “en la ciencia de la computación y en la programación, los algoritmos son más importantes que los lenguajes de programación o las computadoras. Un lenguaje de programación es sólo un medio para expresar un algoritmo y una computadora es sólo un procesador para ejecutarlo”.

Loa algoritmos son independientes de los lenguajes de programación. En cada problema el algoritmo puede escribirse y luego ejecutarse en un lenguaje diferente de programación. El algoritmo es la infraestructura de cualquier solución, escrita luego en cualquier lenguaje de programación.

Características de los algoritmos

- Preciso. Definirse de manera rigurosa, sin dar lugar a ambigüedades.
- Definido. Si se sigue un algoritmo dos veces, se obtendrá el mismo resultado.
- Finito. Debe terminar en algún momento.
- Puede tener cero o más elementos de entrada.
- Debe producir un resultado. Los datos de salida serán los resultados de efectuar las instrucciones.

Se concluye que un algoritmo debe ser suficiente para resolver el problema. Entre dos algoritmos que lleven a un mismo objetivo, siempre será preferible el más corto (se deberá analizar la optimización de tiempos y / o recursos).

Etapas para la solución de un problema por medio del computador

1. Análisis del problema, definición y delimitación (macroalgoritmo). Considerar los datos de entrada, el proceso que debe realizar el computador y los datos de salida.
2. Diseño y desarrollo del algoritmo (se utiliza pseudocódigo, escritura natural del algoritmo, diagramas de flujo, etc.)
3. Prueba de escritorio. Seguimiento manual de los pasos descritos en el algoritmo. Se hace con valores bajos y tiene como fin detectar errores.
4. Codificación. Selección de un lenguaje de programación y digitación del pseudocódigo haciendo uso de la sintaxis y estructura gramatical del lenguaje seleccionado.

5. Compilación o interpretación del programa. El software elegido convierte las instrucciones escritas en el lenguaje a las comprendidas por el computador.
6. Ejecución. El programa es ejecutado por la máquina para llegar a los resultados esperados.
7. Depuración (debug). Operación de detectar, localizar y eliminar errores de mal funcionamiento del programa.
8. Evaluación de resultados. Obtenidos los resultados se los evalúa para verificar si son correctos. (Un programa puede arrojar resultados incorrectos aún cuando su ejecución no muestra erorres).

Algoritmos cualitativos y algoritmos cuantitativos

Un algoritmo es cualitativo cuando en sus pasos o instrucciones no están involucrados cálculos numéricos. Las instrucciones para armar un aeromodelo, para desarrollar una actividad física o encontrar un tesoro, son ejemplos de algoritmos cualitativos.

Trate de diseñar el algoritmo para estos casos

- Tomar mate
- Utilizar una guía telefónica
- Cocinar siguiendo una receta
- Cambiar una llanta de automóvil
- Buscar una palabra en el diccionario

Los algoritmos cuantitativos involucran cálculos numéricos.

Ejemplos:

- Solución de un factorial
- Solución de una ecuación de segundo grado
- Encontrar el mínimo común multiplicador.

Técnicas de representación

Para la representación de un algoritmo, antes de ser convertido a lenguaje de programación, se utilizan algunos métodos de representación escrita, gráfica o matemática. Los métodos más conocidos son:

- Diagramación libre (Diagramas de flujo)
- Diagramas Nassi-Shneiderman
- Pseudocódigo
- Lenguaje natural (español, inglés, etc.)
- Fórmulas matemáticas

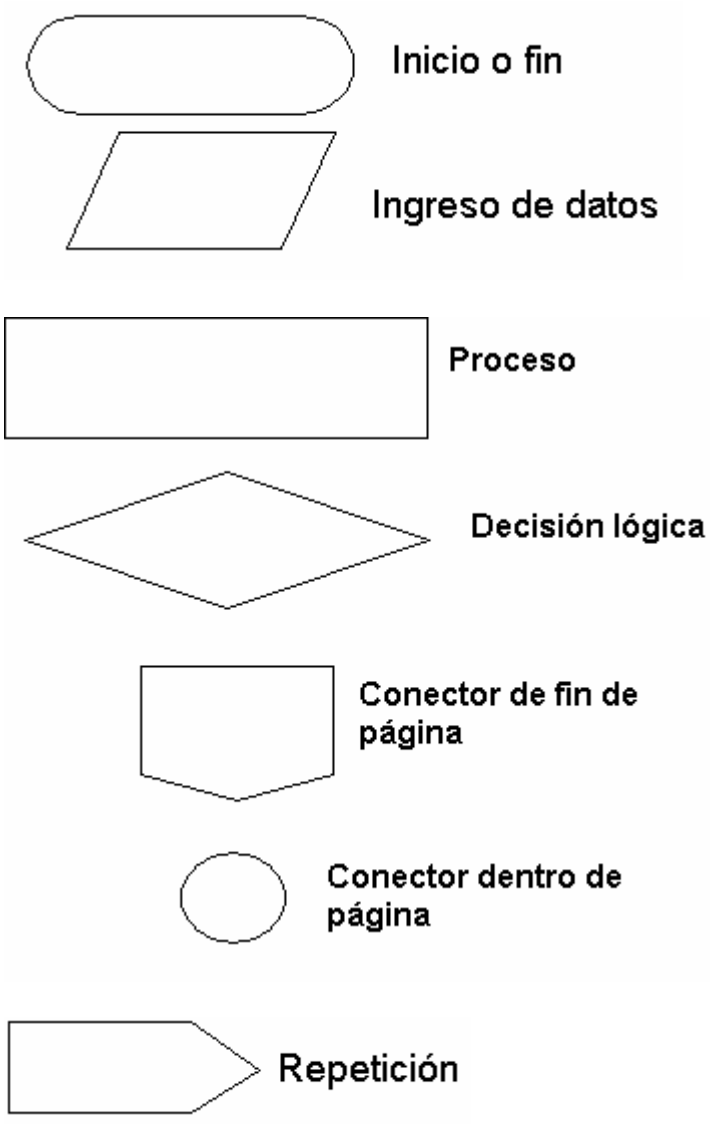
El lenguaje natural puede no ser suficientemente preciso, permitiendo ambigüedades, obteniendo una descripción no del todo satisfactoria. Las fórmulas, propias del lenguaje matemático, son un buen sistema de

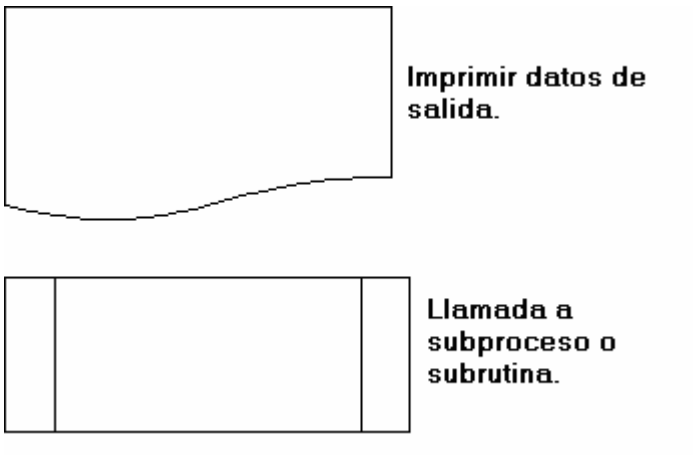
representación, pero no suelen ser fáciles de convertir en programas. Por lo tanto, trataremos en este curso los tres primeros modelos.

Diagramas de flujo.

Es quizás la forma de representación más antigua. Algunos autores suelen llamarlos también como diagramas de lógica o flujogramas.

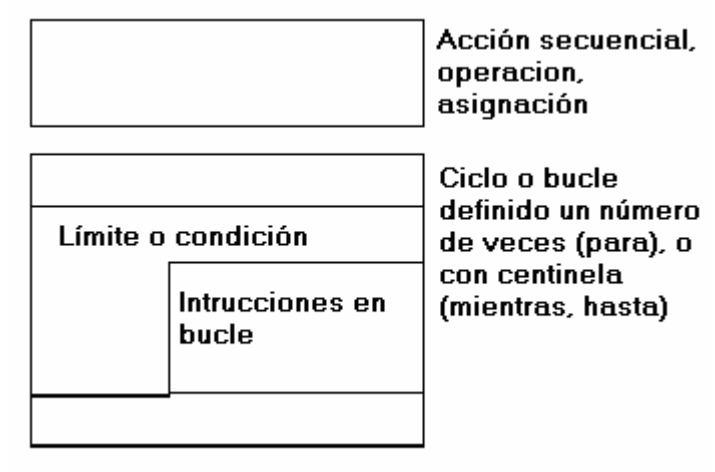
Un diagrama de flujo utiliza cajas estándar tales como las que se muestran en las figuras 1, 2 y 3:

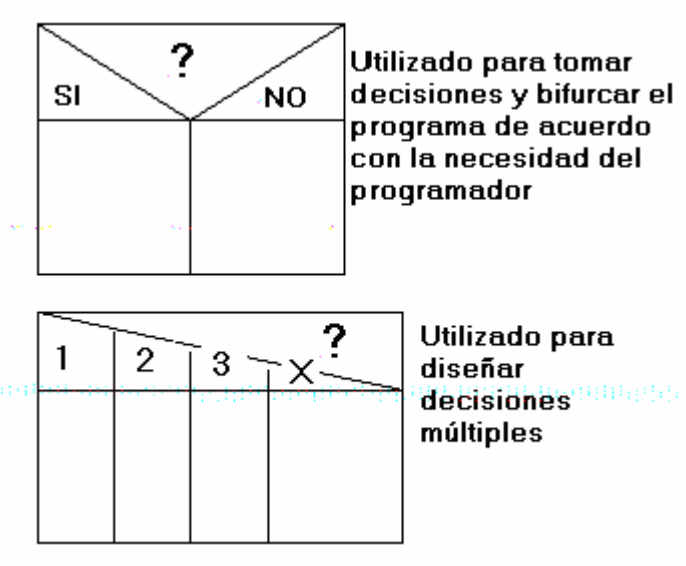




Diagramas Nassi-Schneiderman o Chapin

También conocidos como Diagramas de Chapin, corresponden a uno de los tipos de diagramación estructurada. Las acciones se escriben en rectángulos o cajas sucesivas. Se pueden escribir diferentes acciones en una caja. La simbología utilizada es como vemos en las figuras siguientes.





Pseudocódigo

Es la técnica que permite expresar la solución de un problema mediante un algoritmo escrito en palabras normales de un idioma (por ejemplo, el español), utilizando palabras imperativas. Es común encontrar en pseudocódigo palabras como: Inicie, lea, imprima, sume, divida, calcule, finalice. No hay un léxico obligado para el pseudocódigo, pero con el uso frecuente se han establecido algunos estándares. Este es un ejemplo de un programa escrito en pseudocódigo:

Inicie

{Calcule el salario neto y deducción de 6%} {Esto es un comentario}

Lea nombre, horas, valor_hora

Salario_bruto=horas*valor_hora

Deduccion=Salario_bruto*6%

Salario_neto=Salario_bruto – Deduccion

Imprima nombre, Salario_bruto, Deduccion, Salario_neto

Finalice

Técnicas de diagramación

En nuestra asignatura, por su facilidad y adecuada representación de los problemas a resolver, utilizaremos para represetar los algoritmos, a la técnica de diagramas de flujo.

A su vez, para un mejor ordenamiento en la realización de esos diagramas, se han elaborado técnicas de diseño de los mismos.

Nosotros utilizaremos las denominadas top-down y estructrurada.

La primera de ellas, la top-down, persigue la descomposición de un problema en partes, tomando en primer lugar la dimensión total, para luego ir identificando sus partes componentes e ir tratándolas en forma particular y con mayor grado de detalle cada vez, hasta llegar a una expresión final de resolución simple, trivial o ya conocida.

La diagramación estructurada no indica la forma en que se pueden utilizar y vincular los símbolos gráficos entre sí.

De esta manera se distinguen la siguientes estructuras elementales, que luego al combinarse entre sí, dan lugar al diagrama total.

Estructuras:

Secuencia

Decisión simple

Decisión múltiple

Repetición con condición inicial

Repetición con condición final