Tema 6

Algoritmos

Una vez que se tiene una idea de cual es la estructura y funcionamiento de la computadora digital es posible preparar el camino para lograr su programación. En primer lugar hay que dejar claro que un programa es una realización concreta de un algoritmo que resuelve un problema, por lo que la tarea difícil es en la mayoría de los casos la de hallar el algoritmo.

En este tema se van a tratar varios aspectos relacionados con la programación. En primer lugar se mostrará la forma de exponer los algoritmos usando el lenguaje natural de las personas. Estas descripciones reciben el nombre de pseudocódigo, y no están ligadas a ninguna máquina concreta. Posteriormente se mostrará cómo crear representaciones gráficas de los algoritmos llamadas diagramas de flujo. Finalmente, se tratan los problemas que aparecen al tratar de modificar programas previamente escritos. La solución a tales problemas pasa por un conjunto de reglas que, restringiendo la libertad del programador, permiten producir programas legibles y fácilmente modificables.

6.1 Algoritmos y pseudocódigo

Un algoritmo[1] es un "conjunto ordenado y finito de operaciones que permite hallar la solución de un problema". No debe confundirse algoritmo con programa, este último es la codificación del algoritmo en algún lenguaje de programación o en instrucciones de la máquina. La resolución de problemas mediante computadora conlleva dos pasos: hallar un algoritmo y su posterior codificación, lo cual se ha ilustrado de forma gráfica en la figura 6.1.

[1] La palabra procede de al-Jwarizmi o Al Juarismí, sobrenombre del célebre matemático Mohamed ben Musa (aprox. 780-850) considerado padre del Álgebra.

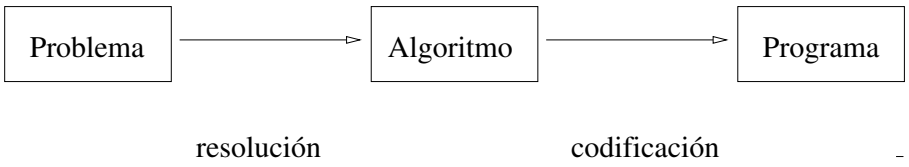


Figura 6.1: Proceso de resolución de problemas mediante computadora.

Un algoritmo es una explicación no ambigua de cómo resolver un problema. Como ejemplo considérese una receta de cocina, en ella se indican los pasos a seguir para resolver la tarea de producir cierto plato. Los algoritmos con que trataremos aquí tienen un carácter más matemático.

El pseudocódigo es un modo de especificar la solución de un problema, es decir, es una expresión de un algoritmo dada en lenguaje natural. A modo de ejemplo considérese el problema consistente en hallar la media aritmética de dos valores a y b.

1	*Obtener el primer valor,* a
2	*Obtener el segundo valor,* b
3	*Asignar a la media* m *el valor* $(a+b)/2$
4	*Fin*

Cada renglón de la lista anterior contiene una sentencia que especifica una operación a realizar con los datos para obtener los resultados. Las operaciones necesarias para resolver el problema han sido especificadas en lenguaje natural (el lenguaje de las personas). De este modo el pseudocódigo es una receta válida para todo programador, cualquiera que sea el lenguaje de programación que vaya a usar. En el ejemplo anterior las operaciones a realizar vienen dadas en secuencia; esto es, una detrás de otra. Pero éste no es siempre el caso, como ocurre en el siguiente ejemplo. Considérese la tarea de hallar el valor absoluto de un número x.

1	*Si* x *es positivo, el resultado es* x
2	*Si no, el resultado es* $-x$
3	*Fin*

Muchos algoritmos requieren la repetición de un conjunto de operaciones cierto número de veces. En tales casos es conveniente disponer de un modo para indicar dicha repetición en lugar de repetir las sentencias. Por ejemplo, el algoritmo para hallar la media aritmética de las componentes de un vector v de dimensión 100 podría ser el indicado en la tabla siguiente.

1 *Iniciar la suma parcial sp a cero*
2 *Iniciar el índice k a uno*
3 *Hacer:*
  3.1 *Dar a sp el valor* $sp + v_k$
  3.2 *Incrementar el índice k*
  3.3 *Si* $k > 100$ *ir al punto* 4, *si no ir a* 3
4 *El resultado es* $sp/100$
5 *Fin*

Los tres ejemplos anteriores presentan tres situaciones muy frecuentes en programación y que serán tratadas con detalle en este capítulo.

Los algoritmos no sirven sólo para operaciones matemáticas; como ha sido el caso de los ejemplos mostrados, también pueden emplearse en otros contextos como el ejemplo de la receta de cocina. Resulta fácil imaginar que si la receta no está bien explicada es posible que no pueda llegarse con éxito al objetivo. Tomemos como ejemplo una receta para freir un huevo:

1 *Poner aceite en la sartén*
2 *Colocar la sartén en el fuego*
3 *Romper el huevo haciendo caer el contenido en la sartén*
4 *Tirar las cáscaras a la basura*
5 *Poner sal en la yema*
6 *Si el huevo está sólido ir a* 7, *si no esperar*
7 *Servir el huevo, fregar la sartén*
8 *Fin*

Dejando aparte la habilidad manual necesaria, es obvio que si la persona que lee la receta desconoce el significado de alguna acción será incapaz de preparar el huevo frito. Esta observación también se puede aplicar a los pseudocódigos de algoritmos matemáticos, por lo que hay que poner especial cuidado en utilizar sólo aquellos elementos que se suponen conocidos por el operador (humano o máquina) que ha de realizar las operaciones.

En la literatura especializada se denomina PROCESADOR a la entidad que realiza las operaciones indicadas por el algoritmo. Las acciones que el procesador conoce se denominan PRIMITIVAS. En un algoritmo puede haber operaciones no primitivas siempre y cuando sean definidas usando acciones primitivas. Sobre este tema se volverá más adelante.

6.2 Objetos y operaciones

En la descripción de algoritmos es preciso con frecuencia hacer referencia a objetos que hay que manipular. Considérese el ejemplo de hallar el valor absoluto expuesto anteriormente. En el pseudocódigo hemos utilizado sentencias como "*Si x es positivo, el resultado es x*". En ella hay un objeto x que, por familiaridad con la notación matemática, hemos identificado sin problema como un número cualquiera.

Al realizar algoritmos se manejan diversos tipos de informaciones, no sólo numéricas. Considérese por ejemplo el problema de cifrar un mensaje para transmitirlo de forma secreta. Un modo sencillo de cifrar mensajes es cambiar cada letra por la siguiente en el alfabeto. De este modo, el mensaje: "estamos bien" se convierte en "ftubnpt cjfo". El pseudocódigo siguiente permite realizar la tarea de cifrado.

```
1  Iniciar índice k a 1
2  Hacer:
     2.1  Tomar letra la letra de la palabra que
          ocupa el lugar k
     2.2  Reemplazar dicha letra por la siguiente en
          el alfabeto
     2.3  Incrementar el índice k en una unidad
     2.4  Si k es mayor que el número de letras ir a
          3, si no ir a 2
3  Fin
```

En este caso se ha identificado cada letra de la palabra por un símbolo l_i. Además, se ha considerado que al sumar uno a una letra se consigue la siguiente letra del alfabeto. Ciertamente, éstos no son usos corrientes de la notación matemática.

Para poder escribir algoritmos de forma cómoda y precisa es conveniente nombrar a distintos objetos (como números o letras) usando para ello identificadores o nombres. El identificador simboliza una cantidad que puede ser constante o que puede variar a lo largo de la tarea. Cada identificador hace referencia a un único objeto que es de un tipo determinado (número entero, letra, palabra, número real, número complejo, etc.)

En resumen, los objetos tienen un único nombre que lo identifica. Poseen también un tipo que no varía durante el algoritmo. El valor o cantidad representado por el nombre puede variar durante el algoritmo, en tal caso se dice que el objeto es una VARIABLE. Por contra, existen objetos cuyo valor no cambia, son las llamadas CONSTANTES. Para aclarar los conceptos anteriores considérese el problema de obtener dos números enteros a y b y hallarles la media.

1	*preguntar el valor de* a *y* b
2	*la media* m *es* $\frac{a+b}{2}$
3	*Fin*

La tabla siguiente muestra los atributos de los objetos que aparecen en el algoritmo anterior.

objeto	nombre	valor	tipo
primer número	a	variable	número entero
segundo número	b	variable	número entero
media	m	variable	número real
dos	2	constante	número entero

Obsérvese que los valores de a, b y m pasan de ser indeterminados a tener un valor concreto tras los pasos 1 y 2. Por este motivo a y b son variables. La única constante es el número dos, que se ha representado con el identificador 2.

Los objetos son manipulados mediante operaciones como la suma, la resta, etc. o mediante otras acciones "romper", "vaciar". Los enunciados en los que se indican las manipulaciones son llamados EXPRESIONES. Las expresiones muestran la forma en que tal manipulación se realiza, para ello se combinan los nombres de objetos con signos que simbolizan operaciones. Por ejemplo, al escribir m *es* $\frac{a+b}{2}$ se indica que la variable m ha de cambiar su valor tomando el resultado de sumar a y b y dividir por dos. Las operaciones involucradas son la suma aritmética, la división aritmética y la asignación, operación que necesita un comentario adicional.

No se ha de confundir la asignación con la igualdad Matemática, aunque en muchos lenguajes de programación se utiliza para la asignación el signo $=$. Como es sabido, en Matemáticas se emplea para establecer una relación de equivalencia entre dos términos. En lenguajes de programación como C y MATLAB, el signo igual se usa para ASIGNAR valores a variables. De este modo $x = 2$ significa que la variable x toma el valor dos en ese momento, pudiendo más adelante alterarse dicho valor mediante una nueva asignación. Con objeto de evitar confusiones, se usará el signo $\leftarrow$, de forma que se escribirá $x \leftarrow 2$ para indicar que el valor 2 es asignado a la variable x.

La descripción de un algoritmo no será de ninguna utilidad si contiene operaciones no conocidas por la persona (o mecanismo) que lo recibe. Los lenguajes de alto nivel ponen a disposición del programador muchas operaciones diferentes, por lo que escribir el programa a partir del algoritmo resulta fácil. En cambio, cuando se usa un lenguaje de bajo nivel es preciso describir cada tarea en función de operaciones simples, por lo que la escritura es larga y tediosa.

Los lenguajes algorítmicos tienen reglas para la escritura destinadas a obtener algoritmos legibles. No hay espacio en esta obra introductoria para describir en detalle un lenguaje al-

gorítmico. Sin embargo podemos citar las siguientes restricciones, algunas de las cuales han sido ya comentadas.

- El lenguaje algorítmico debe tener palabras reservadas como "iniciar", "asignar", "incrementar". Los objetos no pueden tomar como nombre ninguna de estas palabras.
- Cada objeto debe tener un identificador. Los identificadores pueden contener letras y números, pero habitualmente comienzan por una letra. Además se evita que los identificadores contengan símbolos que indican operaciones, como $+$, $-$, etc.
- El conjunto de acciones primitivas ha de ser especificado con total claridad formando un conjunto pequeño pero suficiente.

A continuación se indican las convenciones que se utilizan en esta obra para presentar los algoritmos. Se ha prescindido de algunos de los aspectos formales de la algoritmia en aras de una presentación más concisa e intuitiva.

6.3 Lenguaje algorítmico

Para este curso se ha seleccionado un lenguaje algorítmico de complejidad moderada que permite realizar ejercicios de resolución de problemas. Los algoritmos expresados con este lenguaje resultan de fácil codificación en la mayoría de lenguajes de alto nivel como C, Java, MATLAB, Pascal, Basic, Fortran.

A continuación se presentan las características del lenguaje.

6.3.1 Objetos

Los tipos de objetos que se utilizarán son también los habituales de las Matemáticas: números enteros y números reales. Hay que tener en cuenta que la computadora trabaja con números consistentes en un número finito de potencias de dos. Esto quiere decir que sólo algunos enteros y reales pueden representarse. Para aclarar esto considere el caso se números con infinitos decimales como π o números sin decimales que exceden la capacidad de representación de los registros de la máquina como por ejemplo $10^{10^{10}}$.

Para simplificar las cosas se va a considerar que la computadora proporciona una buena aproximación a los enteros y reales que aparecen en la mayoría de casos prácticos.

Hay un tipo especial que es el carácter o elemento literal que viene dado por el código ASCII correspondiente del signo en cuestión. Puesto que dicho código es un número entero se tiene que el elemento literal no es más que un caso especial de número entero comprendido en el intervalo $[0, 255]$.

Además de los tipos simples: enteros, reales y caracteres, se van a considerar sus agrupaciones formando vectores y matrices.

6.3.2 Nombres de los objetos

Los objetos se identificarán mediante un nombre. A fin de facilitar la codificación posterior se eligirán nombres que puedan ser usados en los lenguajes de programación. Para ello el nombre o identificador consistirá en una combinación de caracteres que cumpla los requisitos siguientes:

- El nombre comenzará por una letra
- El nombre no contendrá espacios en blanco ni signos como +, -, (, etc.
- El nombre será único, no pudiendo coincidir con el nombre de otro objeto
- El nombre no podrá coincidir con nombres de funciones ni contener signos que expresen operaciones primitivas como incrementar, asignar, etc.
- El nombre podrá contener números lo cual no ha de llevar a confusión con el uso de subíndices

6.3.3 Operaciones primitivas

Los lenguajes de programación de alto nivel incorporan muchas operaciones para que resulte más fácil la codificación de algoritmos. Es habitual disponer de operaciones avanzadas como senos, cosenos, logaritmos, exponenciación, raíces, cálculos con números complejos, etc.

Sin embargo, para aprender a programar no hace falta manejar una lista tan amplia de operaciones. Con la idea de simplificar se van a permitir únicamente unas pocas operaciones que permitan resolver un amplio número de problemas. La lista de operaciones primitivas queda de este modo reducida a un conjunto mínimo fácil de recordar y que se indica a continuación.

Operaciones aritméticas . En este grupo se encuentran la suma de dos números enteros o reales, la resta de dos números enteros o reales, la multiplicación de dos números enteros

o reales, la división de un número entero o real por otro distinto de cero. Además se considera la operación cambio de signo que es equivalente a multiplicar por el entero -1. La forma de indicar estas operaciones será la habitual mediante los símbolos de la suma (+), resta y cambio de signo (-), división (/) y multiplicación ($\cdot$).

Asignación . Esta operación sirve para llevar el resultado de una expresión a una variable. La asignación se indica en muchos lenguajes mediante el signo =, lo cual puede dar lugar a confusiones. Para evitarlas se usará exclusivamente la flecha $\leftarrow$ para indicar asignaciones. Así una expresión como $x \leftarrow 3+2$ indica que se ha de asignar el resultado de la operación $3+2$ a la variable x.

Operaciones de comparación . Sirven para comparar los valores de objetos. En este grupo se incluye la operación mayor que $>$, menor que $<$, igualdad $=$, distinto a $\neq$, y combinaciones como mayor o igual $\geq$, menor o igual $\leq$. Estas operaciones dan como resultado un valor 0 o 1. El valor 0 es equivalente a un resultado falso en la comparación, mientras que el valor 1 indica un resultado verdadero, dicho de otro modo el 1 indica que se cumple la comparación. Por ejemplo $3 > 2$ da como resultado 1 mientras que $3 = 2$ da como resultado 0.

Operaciones lógicas . Sirven para relacionar varias comparaciones. En este grupo se encuentran las operaciones Y (o producto lógico), O (o suma lógica), NO (o negación). Como ejemplo de uso considere la expresión $x < 1$ Y $z > 2$ que sirve para comprobar si x es menor que 1 y al mismo tiempo z es mayor que 2.

Paréntesis . Los paréntesis se usarán con el mismo significado que en Matemáticas, para forzar a realizar los cálculos en un determinado orden. De este modo no será lo mismo escribir $(3+2)\cdot 4$ que $3+(2\cdot 4)$.

Lectura . Es una operación especial que conlleva dos acciones: en primer lugar la obtención de un dato del exterior y en segundo lugar la asignación de dicho dato a la variable indicada. Así pues la expresión *leer x* equivale a obtener un dato (normalmente del teclado) y asignar el dato a la variable x.

Escritura . Es una operación especial que consiste en enviar un valor a un dispositivo externo, normalmente la pantalla.

Subindexación . A fin de poder trabajar con vectores y matrices de forma sencilla se admite una operación más consistente en el acceso a componentes individuales de vectores y matrices mediante subíndices. De este modo si v es el nombre de un vector de enteros y x un entero, entonces la expresión $v_2 \leftarrow x+3$ indica que se ha de asignar el valor $x+3$ al segundo elemento del vector v.

Llamadas a funciones . Además de las operaciones primitivas comentadas se permitirá de forma ocasional el uso de funciones. Una expresión como $y \leftarrow Seno(3.5)$ indica que se ha de calcular la función *Seno* con argumento (variable independiente) igual a 3.5 y el resultado se ha de asignar a la variable y. Sólo se permitirá el uso de funciones en dos

supuestos: o bien funciones previamente realizadas por el alumno o bien funciones que explícitamente se hayan permitido para un problema particular.

6.3.4 Ejemplos

A fin de ilustrar el uso del lenguaje algorítmico se desarrollan ahora las soluciones a unos problemas simples. Tenga en cuenta que la forma final de indicar los algoritmos será mediante diagramas de flujo por lo que estos ejercicios representan un paso intermedio.

Suma de dos números

Como primer ejercicio considere el problema consistente en *"leer dos números y escribir la suma"*. En primer lugar se ha de decidir qué objetos utilizar y proporcionarles un identificador. Puesto que el enunciado no indica lo contrario conviene tomar dos objetos del tipo variables reales para los datos. A continuación se toma la determinación de denominar a al primer sumando y b al segundo sumando. Por otra parte la suma será otro objeto cuyo tipo es real. Se elige como identificador c.

Un posible algoritmo es el siguiente:

1	*Leer* A
2	*Leer* B
3	$C \leftarrow A + B$
4	*Escribir* C
5	*Fin*

Conviene observar que para asignar su valor a la variable c se ha usado la notación $c \leftarrow a+b$, que significa *introducir en c el valor a+b*; esto no debe inducir a pensar que $c = a + b$ en el sentido matemático dado usualmente a tales ecuaciones. La interpretación correcta es que la variable c toma en ese instante el valor correspondiente a la suma $a+b$, pudiendo posteriormente cambiar este valor por otro en el curso del algoritmo.

Media de dos números

El enunciado del problema es *"calcular y escribir la media de dos números enteros"*. Los datos a leer son dos enteros. Se elige como identificadores x e y. El resultado es un valor real que se denominará m. Además será preciso utilizar la constante 2.

Un posible algoritmo es el siguiente:

1	*Leer* x
2	*Leer* y
3	$m \leftarrow (x + y)/2$
4	*Escribir* m
5	*Fin*

6.4 Diagramas de flujo

Un diagrama de flujo u organigrama es una representación gráfica de las distintas operaciones que deben ser realizadas por un operador (humano o mecánico) para la resolución de un problema, indicando el orden lógico de las mismas, las posibles alternativas, etc.

Existen muchos tipos de representaciones gráficas. En este texto se va a usar un conjunto de símbolos para representar acciones y decisiones, conectados por flechas que indican el orden en que se realizan. Hay que hacer notar que la representación gráfica de un algoritmo no es exclusiva de la informática, sino que puede usarse en cualquier entorno que requiera planificación. Más aún, la representación del algoritmo es independiente de la programación en un lenguaje u otro o de la máquina en que vaya a ser ejecutado.

6.4.1 Bloques constructivos

La figura 6.2 muestra el conjunto de símbolos que se van a usar para realizar diagramas de flujo. De izquierda a derecha y de arriba a abajo se tiene:

- **Comienzo de módulo.** Marca el comienzo del algoritmo o de un módulo del mismo.
- **Proceso.** Este elemento indica la realización de operaciones diversas. Equivale a una o más sentencias del pseudocódigo.
- **Bifurcación condicional.** Permite elegir una de entre dos opciones en función de cierta condición que se ha de especificar. Permite representar operaciones de decisión como "*si ... , entonces ... ; si no ...*", como se ha visto en el ejemplo del pseudocódigo para calcular el valor absoluto de un número.
- **Operación de entrada o salida.** Gracias a este bloque se pueden tomar datos del exterior (entrada) o enviar resultados al exterior de la computadora (salida). Equivale a las operaciones de lectura y escritura del pseudocódigo.

- **Fin de módulo.** Marca el final de un módulo.

- **Función o subprograma.** Los subprogramas son bloques de código preparados para ser incluidos en programas, de forma que se ahorre tiempo de preparación de los mismos. Un subprograma se hace cargo de una parte de la tarea que puede ser repetida en varias partes del programa o en programas distintos. Por ejemplo, el cálculo de la norma de un vector o el determinante de una matriz.

  Las funciones son subprogramas que se comportan como programas pues aceptan datos y devuelven resultados. El nombre de función proviene de su semejanza con las funciones matemáticas. Esta semejanza estriba en la toma de datos y la devolución de resultados.

- **Módulo.** Este símbolo indica que la operación es realizada por un bloque que se detalla en otro lugar. Un módulo se diferencia de un subprograma en que éste tiene siempre entidad propia en el código, mientras que el módulo es el resultado de una división en la representación gráfica y su existencia no afecta a la codificación. El módulo permite dividir un diagrama en partes por cuestiones de claridad de presentación. Las funciones en cambio tienen mayor versatilidad, como se verá más adelante.

- **Inicio y actualización.** Realización de operaciones iniciales de un módulo. Asignación de valores iniciales y modificación de índices y contadores, usados sobre todo en bucles.

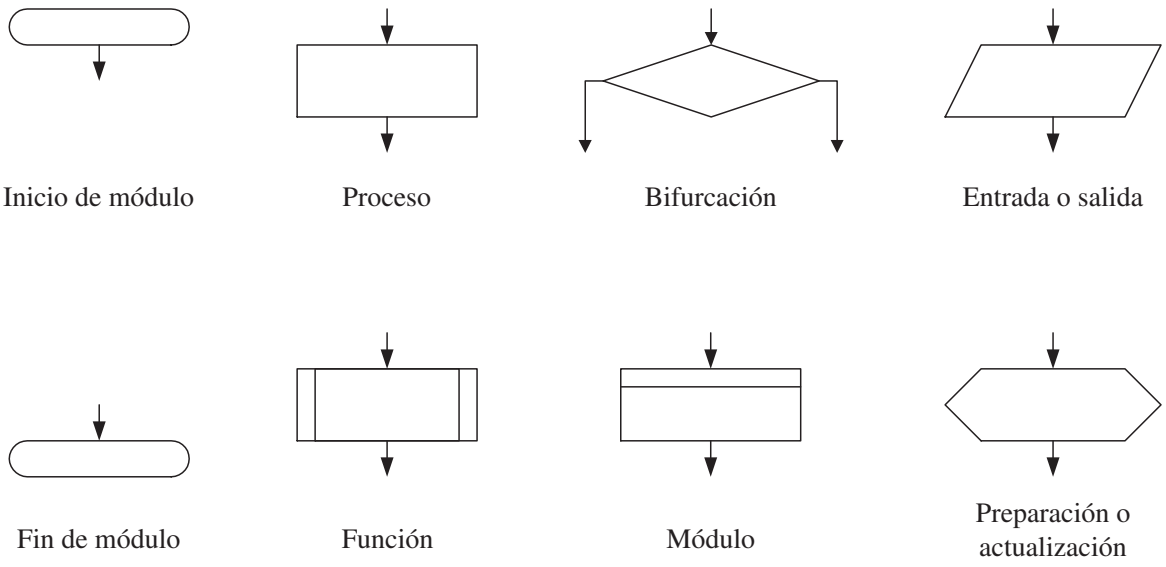


Figura 6.2: Bloques para la confección de diagramas de flujo.

A modo de ejemplo considérese el problema consistente en *"leer dos números y escribir la suma"*. El pseudocódigo correspondiente a este programa ya se ha comentado anteriormente y se muestra en la tabla 6.1.

1	*Leer primer sumando A*
2	*Leer segundo sumando B*
3	*Hallar la suma de los sumandos* $C \leftarrow A + B$
4	*Escribir C*
5	*Fin*

Tabla 6.1: Programa para sumar dos números y escribir el resultado.

La figura 6.3 muestra el diagrama de flujo correspondiente al algoritmo de la tabla 6.1. Puede observarse que el diagrama de flujo simplemente proporciona una forma gráfica al algoritmo.

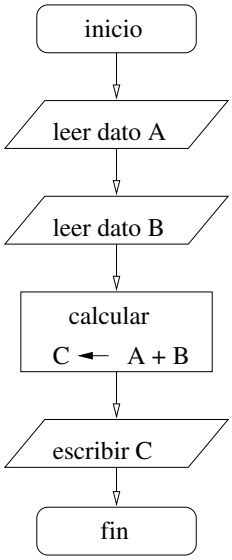


Figura 6.3: Diagrama de flujo de un programa para sumar dos números.

6.4.2 Tabla de objetos

Un diagrama de flujo define las operaciones que se van a realizar con ciertos objetos. No obstante, para que el algoritmo quede completamente especificado es preciso además indicar qué objetos van a ser utilizados.

Cuando se trabaja con computadoras es de gran importancia destacar las características de los objetos usados por un programa. La información acerca de si un objeto es constante o variable, si es de tipo entero, real o literal y otras características se puede proporcionar por medio de una tabla.

Por ejemplo, para el algoritmo cuyas acciones vienen explicadas en el diagrama de la figura

6.3 sería necesaria una tabla donde se declarasen los objetos que se utilizan. La tabla puede tener la forma siguiente:

Objeto	Identificador	Tipo	Valor
Primer sumando. Dato.	A	real	variable
Segundo sumando. Dato.	B	real	variable
Resultado (suma de A y B)	C	real	variable

Las declaraciones de objetos son realizadas en los lenguajes de alto nivel mediante sentencias especiales. Su importancia no es poca pues hay muchos errores que pueden derivarse de un uso incorrecto de objetos. Por ejemplo, si el cociente de una división se almacena en una variable que es entera, los decimales que se produzcan se pierden. Esto da lugar sin duda a fallos de cálculo con repercusiones imprevisibles.

La tabla de objetos complementa al diagrama y facilita el análisis del mismo. Por este motivo el identificador dado al objeto y su explicación han de ser lo más claros posibles. De este modo se facilita la tarea de un programador que necesite analizar un algoritmo hecho por otra persona.

6.4.3 Ejemplos

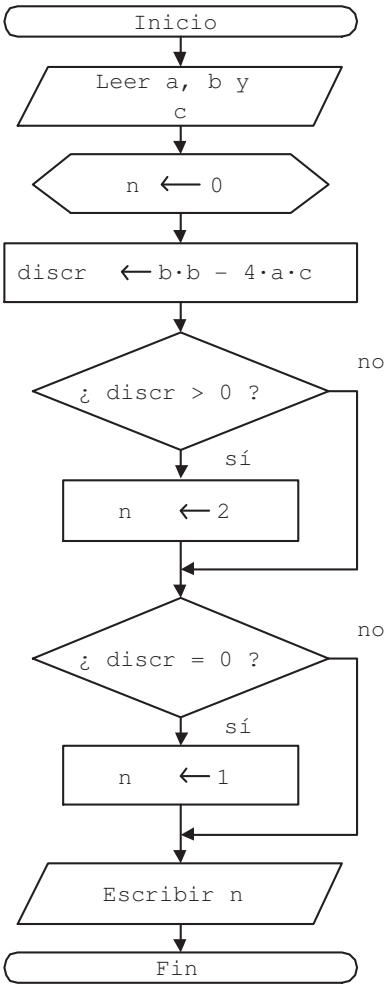
En este apartado se van a mostrar algunos ejemplos de diagramas de flujo que permitan concretar y aclarar las ideas recién presentadas. No se pretende que el lector sea capaz de producir todavía diagramas como los que siguen. Esa tarea se deja para capítulos posteriores. En lugar de eso el lector debe intentar simplemente utilizar los algoritmos para, de este modo, familiarizarse con el uso de los diagramas y tablas.

Se trata de algoritmos muy simples y que sin embargo contienen la esencia de muchos de los problemas que se propondrán en capítulos posteriores.

Ecuación de segundo grado

El problema que se desea resolver se enuncia del siguiente modo: *"Calcular y escribir n, el número de soluciones reales y distintas de la ecuación de segundo grado del tipo* $ax^2+bx+c=0$. *El algoritmo debe tomar como datos los coeficientes a, b y c, supuestos* $\neq 0$. *El resultado ha de ser un número* $n \in 0, 1, 2$ *según haya ninguna, una o dos soluciones reales".*

El diagrama de flujo y la tabla de objetos pertenecientes a un algoritmo que posiblemente resuelve el problema anterior se muestran en la figura 6.4



Primer coeficiente de la ecuación $ax^2+bx+c=0$	a	Variable real
Segundo coeficiente de la ecuación $ax^2+bx+c=0$	b	Variable real
Tercer coeficiente de la ecuación $ax^2+bx+c=0$	c	Variable real
Discriminante, calculado como b^2-4ac	discr	Variable real
Valores que puede tomar el resultado	0, 1 ,2	Constantes enteras
Resultado. Número de raíces reales y distintas de la ecuación $ax^2+bx+c=0$	n	Constante entera

Figura 6.4: Algoritmo para calcular el número de soluciones reales y distintas de una ecuación de segundo grado.

Para analizar el algoritmo es conveniente hacer algunas pruebas a mano con un conjunto de datos controlado. Por ejemplo, si $a = 1$ y $b = c = 0$ se obtiene la ecuación $x^2 = 0$ que tiene como solución trivial $x = 0$, por lo que el algoritmo debiera proporcionar el valor $n = 1$. Se deja como ejercicio al lector seguir los pasos que el diagrama de flujo indica. El resultado proporcionado por el diagrama ha de compararse con el resultado previamente conocido.

Hay que resaltar que este tipo de pruebas no permite validar el algoritmo completamente. Esto es así porque un algoritmo puede dar resultados correctos en muchas situaciones y sin embargo no ser válido universalmente pues puede que contenga errores que no se han manifestado con esos datos. A pesar de ello la prueba de algoritmos en casos previamente conocidos es un buen ejercicio en esta etapa inicial.

El análisis de algoritmos es una disciplina compleja que, por falta de espacio, no podemos abordar de modo sistemático. En lugar de eso dejamos al lector la tarea de usar su perspicacia y sentido común para intentar decretar la validez de los algoritmos que se propongan.

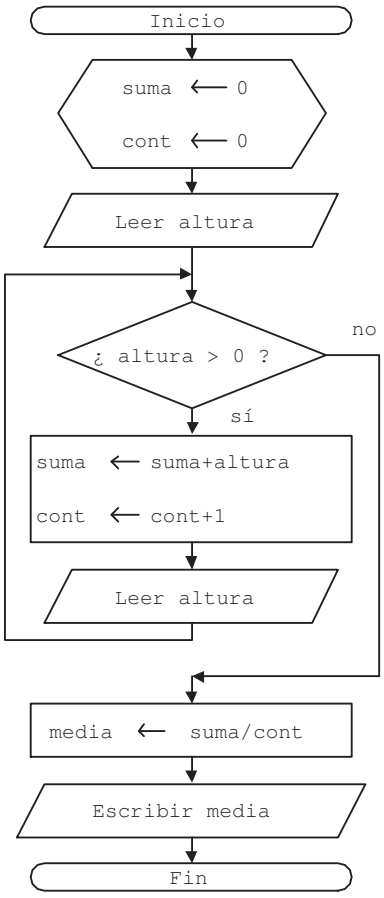
Media de alturas

El problema que se desea resolver se enuncia del siguiente modo: *Se desea crear un programa de computadora que permita obtener la media de alturas de un grupo de personas en una habitación. Se quiere que el operador del programa (el usuario del programa) vaya midiendo a las personas y escribiendo en el teclado sus alturas en centímetros. Cuando no queden más personas por medir el operador escribirá una altura negativa. Ésta será la señal para que el programa deje de leer alturas y calcule la media de los valores previamente introducidos.*

El diagrama de flujo y la tabla de objetos pertenecientes a un algoritmo que posiblemente resuelve el problema anterior se muestran en la figura 6.5

Es posible realizar pruebas del algoritmo con uno o varios conjuntos de datos previamente analizados. Por ejemplo, se supone que en la habitación hay dos personas con alturas 172cm y 178cm respectivamente. Teniendo en cuenta estos datos se han de seguir las indicaciones contenidas en el diagrama del mismo modo que se llevarían a cabo los pasos de una receta de cocina. En cada paso las variables han de modificarse de acuerdo con las operaciones marcadas en el bloque.

Es fácil ver que hay que proporcionar tres alturas: 172, 178 y finalmente -1 para indicar que no hay más personas. Durante el desarrollo del algoritmo la variable *suma* pasa de valer cero a 172 y luego $172 + 178$. El resultado que se obtiene viene dado por la variable *media* que se calcula como *suma*/*cont* que resulta valer 250/2=175. Este resultado coincide con el resultado correcto para estos datos concretos.



Variable usada para leer sucesivamente las alturas de cada persona	altura	Variable entera
Contador del número de personas medidas	cont	Variable entera
Variable usada para agregar las alturas leídas	suma	Variable entera
Resultado. Media de alturas calculada dividiendo la suma por el número de personas	media	Variable real
Valor inicial	0	Constante entera
Valor para incrementos	1	Constante entera

Figura 6.5: Algoritmo para calcular la media de alturas.

Del mismo modo puede ponerse a prueba el algoritmo con otros conjuntos de datos, por ejemplo con las alturas: 172cm, 174cm y 178cm. Un caso particularmente interesante ocurre cuando el algoritmo es usado para calcular la media de alturas del conjunto vacío, es decir, cuando se utiliza con cero personas.

6.5 Fases del proceso de resolución de problemas

Es difícil concretar en proceso de resolución de problemas pues cada problema plantea dificultades especiales. No obstante hay una serie de reglas que conviene observar a la hora de construir algoritmos.

- Definición del problema, de forma que quede absolutamente claro qué es lo que se pretende resolver. Además hay que especificar cuáles son los datos y cuáles los resultados buscados.
- Esbozo de la solución. El problema debe partirse en trozos para los cuales se sospecha que puede haber soluciones conocidas. Es preciso asegurarse que al resolver los trozos se consigue la solución del problema total.
- Resolución de los subproblemas y prueba de la validez de los mismos.
- Expresión en pseudocódigo de los algoritmos correspondientes a cada parte.
- Prueba de validez del algoritmo que surge de unir todas las partes.

Para seguir estas reglas se dispone de la ayuda de la programación estructurada, que será tratada más adelante. El tipo de problemas que se resuelve para dar los primeros pasos en programación es muy simple por lo que no es fácil ver la utilidad de las recomendaciones anteriores.