ALGORITMOS Y PROGRAMAS

CODE UD2 – Tema 2

IES Plurilingüe Antón Losada Diéguez

Adrián Fernández González

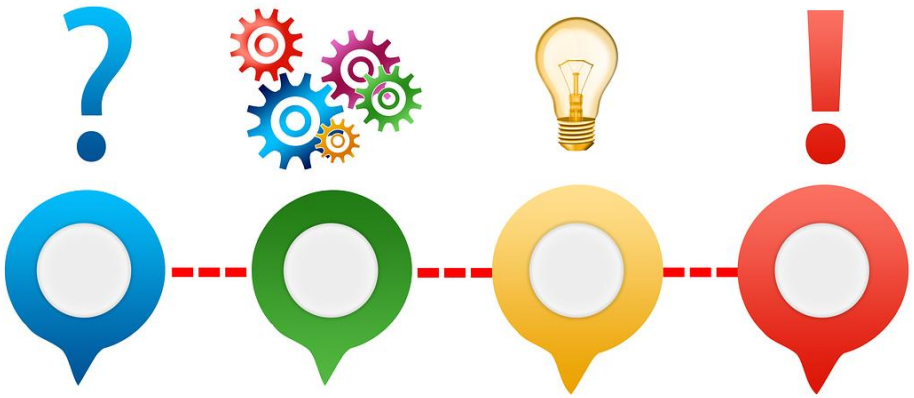
Tabla de contenido

Algoritmos y programas

1. Introducción

El principal objetivo de la informática y, sobre todo, los lenguajes de programación, es la resolución de problemas.

Para ello, se realizan cuatro fases fundamentales, primero se analiza el problema en cuestión, luego se diseñan una serie de algoritmos, se crea el programa en base a ellos y se ejecuta dicho programa.



1.1. Definición y análisis del problema

En esta primera etapa, se estudia el problema a resolver, entender qué es lo que necesitamos para luego poder resolverlo.

Una buena definición del mismo, facilita una resolución eficaz, por lo que es necesario detallar hasta el más mínimo detalle con la mayor precisión posible. Todo esto ha de redactarse en un documento para su posterior consulta.

Esta primera etapa es una de las más tediosas, pero a la vez vital, ya que facilita las siguientes fases y evita posibles errores o que pasemos por alto algo.

2. Algoritmo y programa

Un algoritmo es un conjunto ordenado y finito de operaciones que permite solucionar un problema.

Este ha de dar un resultado preciso, seguir una secuencia finita y clara de pasos, ser independiente del dispositivo y la arquitectura del mismo y siempre ha de devolver el mismo resultado para unas mismas entradas.

Por su lado, un programa es la implementación en lenguaje máquina y/o lenguaje de alto nivel de uno o varios algoritmos. De esta forma, estos algoritmos pueden ser entendidos por una máquina en concreto para ser ejecutados y resolver el problema para el que fueron diseñados.

3. Estructura de un algoritmo

Todo algoritmo está compuesto fundamentalmente por datos, variables y constantes, operadores y expresiones, instrucciones y delimitadores.

3.1 Datos

Existen dos tipos de datos, los simples y los compuestos o estructurados. Los simples son los básicos, los más utilizados por prácticamente todos los lenguajes y representan un carácter, un número o un valor lógico. Los compuestos o estructurados están formados por múltiples datos simples como puede ser una cadena de caracteres.

Los estructurados pueden ser de dos tipos según los datos que lo formen. Si estos datos son del mismo tipo, se denominan homogéneos, mientras que, si son distintos, se denominan heterogéneos.

3.2 Variables y constantes

Una variable es un dato almacenado en memoria al que se le asocia un nombre identificador para su acceso y modificación durante la ejecución del programa.

Si esta variable no modifica su valor durante toda la ejecución, se le denomina constante.

3.3 Operadores y expresiones

Los operadores son una serie de símbolos que indican la operación a realizar entre dos datos u operandos.

Un conjunto de operadores y operandos se denomina expresión.

Los operadores pueden dividirse en varios tipos, pero los más comunes son, de asignación (=), aritméticos (+, -, *, etc), lógicos (AND, OR, NOT) y relacionales (==, <, >=, etc).

Algunos operadores, como los de asignación, pueden combinarse con otros, formando operandos compuestos, por ejemplo += hará una suma de los valores de ambos operandos y asignará dicho valor al operando de la izquierda, que sería una variable o una constante.

Hay ciertos operandos especiales que sustituyen a una instrucción, como el operando ternario (? :), que hace la función de una instrucción alterativa o el operando de concatenación (.) que une dos cadenas de caracteres.

3.4 Instrucciones

Son las distintas órdenes que forman el algoritmo.

3.4.1 Declarativas

Sirven para establecer las variables y constantes a utilizar durante el programa, así como su tamaño y tipo si fuese necesario.

```
int numero;
```

3.4.2 Primitivas

Permiten la lectura y escritura de datos y la asignación de valores a las variables y constantes.

```
numero = 5;
```

3.4.3 Alterativas

Modifican la ejecución del flujo del programa mediante condiciones que establecen si un grupo de instrucciones son ejecutadas o no en función del valor de unas variables.

Existen tres tipos:

-Simples: Si se cumple la condición, se ejecuta un grupo de instrucciones, si no, se ignoran. Por ejemplo el if.

-Dobles: Si se cumple la condición se ejecuta un grupo de instrucciones, si no, se ejecutan otras. Por ejemplo el if else.

-Múltiples: En función del valor de una variable, se ejecuta un grupo de instrucciones, varios o ninguno. Por ejemplo el switch.

```
switch(numero)
{
    case 1:
        System.out.println("Es uno");
    break;
    case 2:
        System.out.println("Es dos");
    break;
    default:
        System.out.println("Es cualquier otra cosa");
}
```

3.4.4 Repetitivas o iterativas

Modifican la ejecución del flujo mediante la repetición de un mismo conjunto de instrucciones mientras se cumpla una condición. Por ejemplo el for, foreach, while o do while.

Dicha condición puede ser un número de veces establecido a priori o indeterminado, dependiendo de que se cumpla una condición dentro del conjunto de operaciones del bucle. Por ejemplo, si buscamos una línea determinada de un fichero, el bucle terminará cuando la encontremos, aunque no sabemos a priori cuántas veces se repetirá.

```
for($i = 0; $i < count($arr); $i++)
{
    echo "El elemento " . $i . " tiene el valor " . $arr[$i];
}
```

```
for(String line : texto)
{
    System.out.println(line);
}
```

3.4.5 Procedimentales

Son agrupaciones de varias instrucciones bajo un mismo nombre, de tal modo que permite su reutilización en distintas partes del programa.

Estos pueden recibir una serie de datos sobre los que trabajar y devolver, o no, un dato.

En los distintos lenguajes se denominan procedimientos y funciones, siendo el primero un conjunto de operaciones que realizan una acción en concreto y en el segundo caso, además de realizar las operaciones, devuelve un valor.

```
public int getValue()
{
    return 5;
}

function mult(p1, p2)
{
    return p1 * p2;
}
```

3.5 Delimitadores

Son el conjunto de caracteres y elementos que conforman la estructura del lenguaje. Estos son los corchetes, llaves, espacios y tabuladores, entre otros.

Cada lenguaje tiene los suyos propios y facilitan la estructuración y lectura del código.

Aunque no hay un estándar y, en un principio, cada lenguaje establece los suyos propios, si hay una cierta tendencia y una serie de buenas prácticas para mejorar la legibilidad y el seguimiento del código.

Aunque no es necesario tenerlos en cuenta a la hora de definir los algoritmos, puesto que son independientes del lenguaje en el que se vaya a implementar, si es una buena práctica utilizar los más comunes para estructurar las operaciones de una forma legible.

4. Técnicas descriptivas

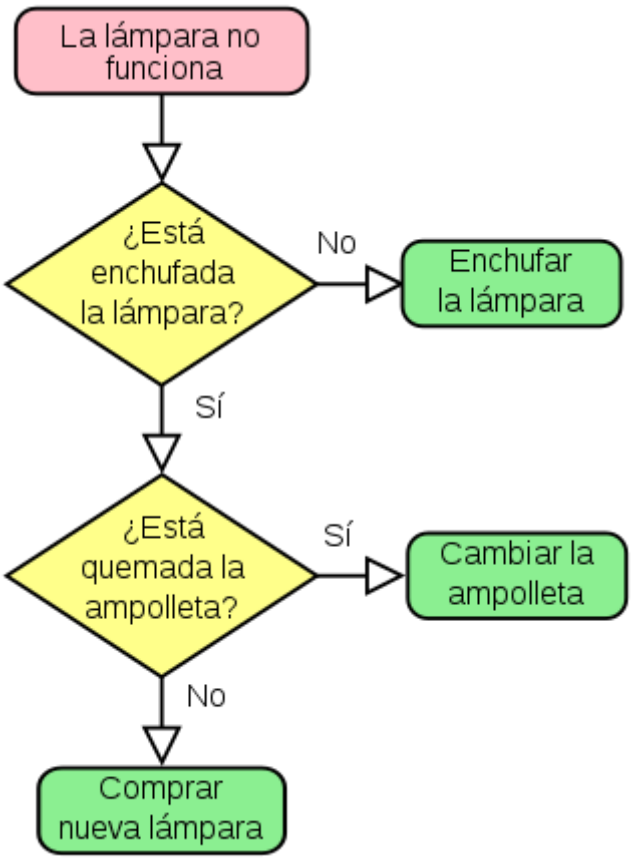
A la hora de describir los algoritmos se utilizan una serie de técnicas o notaciones, fáciles de seguir, entendibles por otros y lo suficientemente detalladas y precisas como para implementar el algoritmo, ejecutarlo y usarlo para resolver el problema para el que fue diseñado.

Para ello, hay dos formas fundamentales de hacerlo, mediante texto o mediante representación gráfica.

4.1 Diagramas de flujo

Son una representación gráfica del algoritmo que muestra claramente la secuencia de acciones a realizar, así como los cambios de flujo, las ramas que estos toman y la entrada y salida de los datos. Se basan en una serie de símbolos normalizados que indican cada tipo de acción.

Todo diagrama de flujo tiene un inicio y un final y toda acción está conectada con otra.



4.1.1 Inconvenientes

Los principales problemas son:

- A mayor algoritmo, mayor complejidad y mayor dificultad para leer y seguir el diagrama.
- Es necesario conocer el significado de los símbolos utilizados y, aún así, cada elemento contiene mucho texto describiendo cada parte.
- No representan las instrucciones a realizar, solo el camino a seguir por el algoritmo.

4.2 Pseudocódigo

El pseudocódigo es un tipo de notación textual que describe los pasos, instrucciones, entrada y salida de datos y hasta uso de variables. Es un paso intermedio entre el lenguaje natural y un lenguaje de programación.

Mediante texto se estipulan todas las operaciones a realizar de una forma que sea fácil de leer y fácil de traducir a un lenguaje de programación sin entrar en detalles ni características de un lenguaje en concreto.

No hay unas normas concretas de cómo describir ni una sintaxis estándar, por lo que cada persona establece el suyo teniendo en cuenta que ha de ser fácilmente entendible por otros. Es habitual, sin embargo, que tenga ciertas similitudes con las estructuras de lenguajes más utilizados en la actualidad, utilizando sus mismas simbologías o similares.

Pseudocódigo estilo Fortran:

```
programa bizzbuzz
hacer i = 1 hasta 100
    establecer print_number a verdadero
    si i es divisible por 3
        escribir "Bizz"
        establecer print_number a falso
    si i es divisible por 5
        escribir "Buzz"
        establecer print_number a falso
    si print_number, escribir i
    escribir una nueva línea
fin del hacer
```

Pseudocódigo estilo Pascal:

```
procedimiento bizzbuzz
para i := 1 hasta 100 hacer
    establecer print_number a verdadero;
    Si i es divisible por 3 entonces
        escribir "Bizz";
        establecer print_number a falso;
    Si i es divisible por 5 entonces
        escribir "Buzz";
        establecer print_number a falso;
    Si print_number, escribir i;
    escribir una nueva línea;
fin
```

Pseudocódigo estilo C:

```
subproceso funcion bizzbuzz
para (i <- 1; i<=100; i++) {
    establecer print_number a verdadero;
    Si i es divisible por 3
        escribir "Bizz";
        establecer print_number a falso;
    Si i es divisible por 5
        escribir "Buzz";
        establecer print_number a falso;
    Si print_number, escribir i;
    escribir una nueva línea;
}
```

4.2.1 Inconvenientes

- No hay un estándar, cada uno lo crea basándose en sus propias estructuras y nomenclaturas que puede que otros no estén acostumbrados.
- Es habitual escribirlo en tu idioma, por lo que alguien de otro lugar tendría que traducirlo.
- Se centra principalmente en la descripción de las instrucciones, dejando un poco de lado el flujo, haciéndose difícil de seguir en algunos casos.

4.2.2 Ventajas

- Es similar al lenguaje de programación, por lo que el paso de este a programar el menor, teniendo cosas parcialmente hechas.
- Es sencillo de leer y aprender, al ser similar al lenguaje natural.
- Las estructuras iterativas se ven claramente, a diferencia de los métodos gráficos.
- Tiene un tamaño reducido y permite ver con más claridad la solución.