Algorismes i pseudocodi

Juan García Cortés 2020
garcia_juacor@gva.es
@juan_garciaTIC

Índex

1. Algorismes i pseudocodi

Els principals objectius d'aquest bloc són:

- Entendre què és un algorisme
- Conéixer les diferents maneres de representar un algorisme
- Comprendre la importància de pensar abans de posar-se a programar

1.1. Algorisme

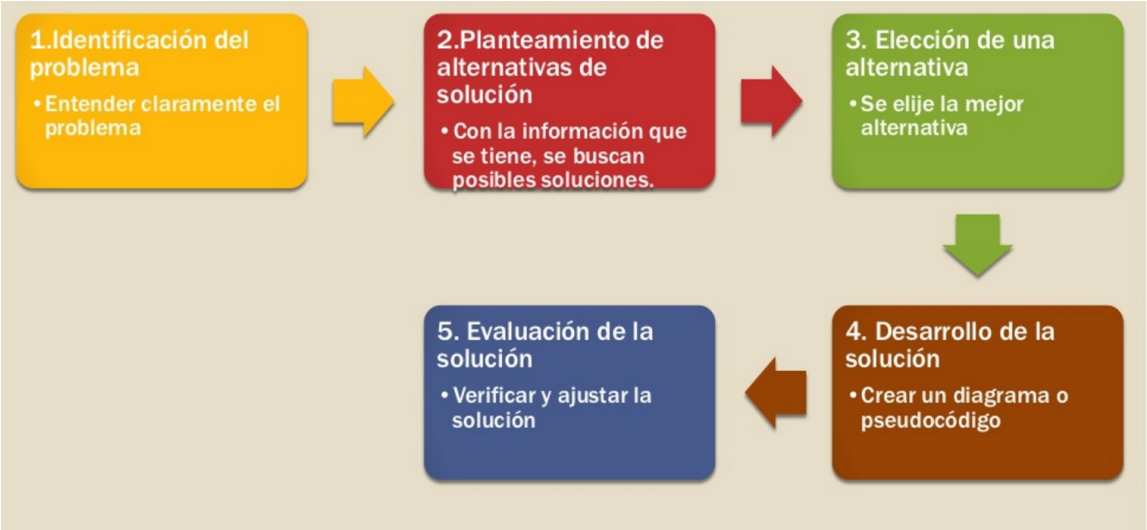
Un algorisme és un conjunt ordenat de passos a seguir (instruccions concretes) que porten a resoldre un determinat problema. Ha de complir amb les següents condicions:

- Ser **correcte** (Resol el problema)
- Ser **finit** (Condueix a la solució en un temps donat)
- Ser **flexible** (No és exclusiu per a una mena de problema sinó que serveix com a mètode general per a diferents dades)
- Ser **clar** (comprensible per altres persones)
- Ser **eficient** (estalvi de temps i recursos)
- Ser **portable** (independent de la màquina o del llenguatge utilitzat)

L'algorisme sorgeix de la necessitat de resoldre un problema donat. Per a això, sempre serà possible crear múltiples solucions. L'algorisme que seleccionem serà aquell que obtinga els resultats esperats en el menor temps possible i amb el menor cost. Per a això hem de reflexionar sobre:

- Quina informació o resultats es volen obtindre?
- A través de quins processos es podran obtindre els resultats?
- Es requereix algun procés intermedi?
- Quin tipus de dades seran necessaris?
- Quines variables?
- Quines condicions exigeix el problema per a la seua solució?

Un dels mètodes més apropiats per a desenvolupar un algorisme és el que es detalla a continuació:



Font: Slideshare de Paco González Caballero

Encara que no siguem conscients, en la vida diària estem utilitzant diàriament algorismes, molts són molt senzills, però no deixen de ser algorismes i que podem descriure utilitzant el llenguatge natural, és a dir, amb pseudocodi. Vegem un exemple molt bàsic:

ALGORISME posar-se les sabates.

INICI

Agafar les sabates.

SI els cordons de les sabates estan nuats **LLAVORS** deslligar les sabates

Asseure's en una cadira.

Posar-se la sabata dreta al peu dret.

Lligar-se els cordons de la sabata dreta.

Posar-se la sabata esquerra al peu esquerre.

Lligar-se els cordons de la sabata esquerra.

FI

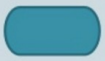
1.2. Representació d'un algorisme

Una vegada que s'ha triat la millor alternativa per a solucionar el problema o repte per al qual es crea l'algorisme és el moment de representar-lo seguint algun d'aquests mètodes:

- Llenguatge natural (espanyol, anglés, etc.)
- Diagrames de flux
- pseudocodi

1.2.1. Diagrames de Flux

A continuació es mostren una sèrie de símbols útils per a dur a terme aquest tipus de representacions.

símbolo	Función	Símbolo	Función
Terminal 	Indicar el inicio y fin del diagrama	Teclado 	Introducir datos manualmente por el teclado
Entrada/salida 	Entrada o salida simple de información	Decisión 	Indica operaciones lógicas o de comparación y tienen dos salidas dependiendo del resultado.
Proceso 	Realizar cualquier operación o calculo con la información		
Salida a Impresora 	Salida de informacion a la impresora	Conectores 	Une dos partes del diagrama a la misma o diferente página
Salida a Pantalla 	Mostrar información de salida a la pantalla	Flechas de Flujo 	Indica la direccion del flujo de la información

Font: Slideshare de Paco González Caballero

En Internet existeixen nombroses eines que poden ajudar-te a crear diagrames de flux. Ací et proposem dues gratuïtes:

- draw.io: Permet guardar els diagrames en el disc dur del teu ordinador, Gdrive, OneDrive o Dropbox.
- Diagramly: Possibilita crear diagrames de flux o mapes mentals.

En aquest document (veure) trobaràs una sèrie de regles i exemples per a crear algorismes i diagrames de flux.

1.2.2. pseudocodi

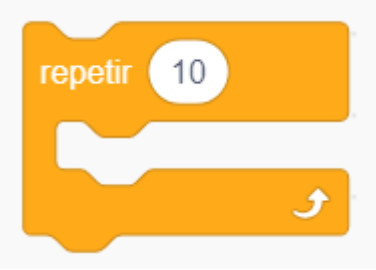
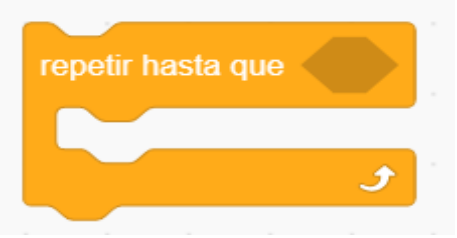
El pseudocodi és sens dubte de les representacions més utilitzades. És una manera

d'expressar l'algorisme utilitzant el llenguatge natural, comprensible per a qualsevol persona, però afegint certes instruccions típiques dels llenguatges de programació.

No existeix una sintaxi estàndard per al pseudocodi, però com hem comentat, en el pseudocodi es reflecteixen les instruccions típiques dels llenguatges de programació, com les instruccions **condicionals**:

```
SI condició LLAVORS
    instruccions/passos a realitzar si es compleix la condició
SI NO
    instruccions/passos a realitzar si NO es compleix la condició
FI SI
```

I les instruccions repetitives:

REPETIR n vegades instruccions/passos a realitzar **FI REPETIR**	
REPETIR FINS A condició d'eixida instruccions/passos a realitzar fins que es complisca la condició d'eixida del bucle	

FI REPETIR	

En definitiva, el pseudocodi es tracta d'un fals llenguatge, ja que apel·la a les normes d'estructura d'un llenguatge de programació encara que està pensat perquè puga ser llegit per un ésser humà i no interpretat per una màquina.

El pseudocodi, en aquest sentit, és considerat com una descripció d'un algorisme que resulta independent d'altres llenguatges de programació. Perquè una persona puga llegir i interpretar el codi en qüestió, s'exclouen diverses dades que no són clau per al seu enteniment. Vegem un parell d'exemples orientats a crear algorismes matemàtics:

Exemple: Realitzar el pseudocodi d'un programa que permeta calcular l'àrea d'un rectangle. S'ha d'introduir la base i l'altura per a poder realitzar el càlcul.

```
Programa: àrea
    Entorn: BASE, ALTURA, AREA són nombres enters
    Algorisme:
    escriure "Introduïsca la base i l'altura"
    llegir BASE, ALTURA
    calcular AREA = BASE * ALTURA
    escriure "L'àrea del rectangle és " AREA
Fiprograma
```

Exemple: Realitzar el pseudocodi que permeta a l'usuari introduir per teclat dues notes, calculant la suma i el producte de les notes.

```
Programa: Suma Producte
    Entorn: NOTA1, NOTA2, SUMA, PRODUCTE són nombres enters
    Algorisme:
    escriure "Introduïsca les notes"
    llegir NOTA1, NOTA2
    calcular SUMA = NOTA1 + NOTA2
    calcular PRODUCTE = NOTA1 * NOTA2
    escriure "La suma de les dues notes és: " SUMA
    escriure "El producte de les dues notes és: " PRODUCTE
```

```
Fiprograma
```

Cap ordinador podria interpretar aquestes instruccions. Per a crear un programa a partir de l'algorisme, una vegada refinat el pseudocodi, hauríem de reescriure'l en un llenguatge de programació: C, C++, Java, Scratch.

1.3. Pensar abans de programar

Si comences a programar directament, llevat que el que faràs siga una cosa realment molt senzilla, el més probable és que l'algorisme que crees no siga ni clar ni eficient. Aprendre a programar és molt més que escriure codi: implica aprendre a pensar. D'ací la importància d'introduir la programació i el pensament computacional a les nostres aules: és un camí que permet fomentar el pensament crític i l'autonomia en els nostres alumnes. En aquest article (en anglés) s'explica com la programació és, en realitat, un mitjà per a aconseguir la vertadera fi última de l'educació: aprendre a pensar.

Que un programa funcione no significa que estiga ben fet. Segurament tots podríem construir una casa posant 10 columnes enmig del saló perquè se sostinga, però no seria eficient ni estètic.

En programació, els usuaris no veuran les columnes que hi ha dins de la casa, però si tu com a programador afiges 10 columnes enmig del “saló” del teu programa, aqueix programa serà molt difícil de mantindre o d'actualitzar en el futur.

Vegem un exemple molt senzill. Suposem que volem fer un programa que demane un número a l'usuari i mostre la taula de multiplicar d'aqueix número. Si no ens parem a pensar, una solució a la qual podríem haver arribat amb Scratch i que funcionaria seria la següent:

Però aquest no és un bon algorisme, ja que estem repetint la mateixa instrucció 10 vegades (hem ficat 10 columnes en el nostre saló). Si abans de posar-nos a programar, ens asseguérem a pensar una mica en un algorisme que calcule la taula de multiplicar, segurament arribaríem a una solució com aquesta que segurament tots som capaces d'entendre:

```
ALGORISME mostrar taula de multiplicar

DADES

     número (el número que demanem a l'usuari)
```

```
        multiplica (el número pel qual anem multiplicant)
    INICI
        Demanar un número a l'usuari i guardar-lo en número
        multiplica = 1
        REPETIR 10 vegades
            ESCRIURE EL TEXT "número x multiplica ="
            ESCRIURE el resultat de multiplicar número x
            multipica
            INCREMENTAR en 1 multiplica
        FI REPETIR
    FI
```

Una vegada que hem pensat l'algorisme, el problema ja està resolt, i simplement hauríem d'utilitzar un llenguatge de programació per a escriure aquest algorisme (Scratch, Java, Javascript, C++, etc.).

Vegem com quedaria aquest algorisme una vegada que l'hem programat en Scratch: