Pseudocodi

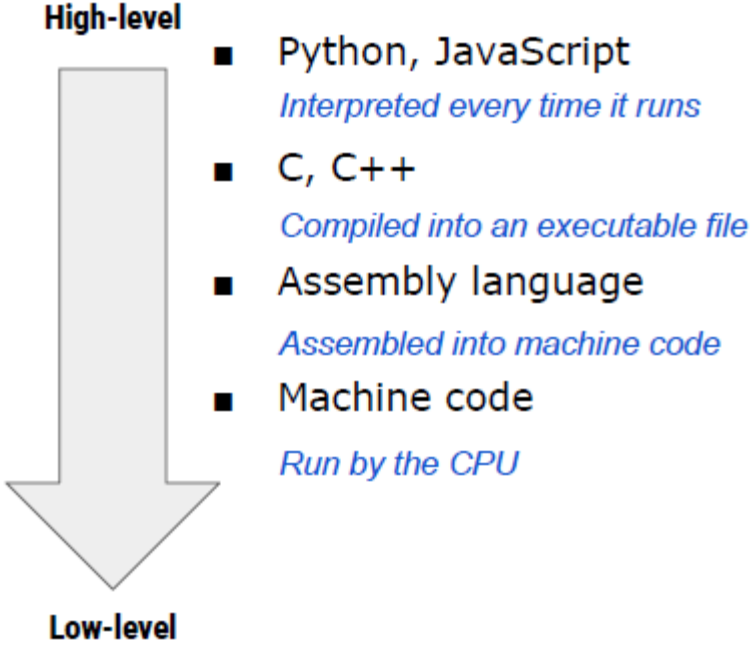
1. Introducció
2. Inici, final i comentaris
3. Assignacions
4. Entrada / Sortida
5. Condicionals
6. Estructures iteratives
7. Crida de funcions

Introducció

El pseudocodi és el llenguatge de ***programació de més alt nivell***.
Això vol dir que **és més proper al llenguatge natural**, és a dir, al llenguatge humà. NO hi ha un estàndard universal.

Per contra, els **llenguatges de baix nivell**, són els llenguatges més **propers al codi màquina**, és a dir, al codi binari.

El pseudocodi l'usem en la fase de disseny per a definir una **lògica algorítmica** que resolgui un problema del món real que després podrà ser implementat en qualsevol llenguatge de programació.



Inici, final i comentaris

2. Inici, final i comentaris

Per a especificar l'inici i el final dels nostres algoritmes en pseudocodi usarem les paraules clau **INICI** i **FINAL.**

A continuació veiem un exemple d'ús:

```
INICI
        // Codi de l'algoritme
FINAL
```

Com podem veure, usarem dues barres // per a especificar que el codi a continuació es tracta d'un **comentari** en forma de text i que no forma part de la lògica de l'algoritme a executar.

Assignacions

3. Assignacions

Un algoritme necessitarà fer ús de **variables** per a **guardar les dades** que va generant. Per exemple, pensem en l'algoritme d'una calculadora; com a mínim necessitarà dues variables inicials per a fer una suma.

Les **variables** poden ser de **diferents tipus de dades**: numèriques, de text, binàries...

Quan donem un valor a una variable direm que fem una **assignació**.

En pseudocodi escriurem les assignacions així:

```
a ← 4
b ← "Hola!"
c ← FALS
```

Entrada / Sortida

Per a que la màquina pugui interactuar amb l'usuari, farem ús d'**instruccions d'entrada** i de **sortida** *(input / output)*.

Això vol dir que quan l'usuari escrigui un valor amb el teclat, l'algoritme **llegirà la dada** i **farà una assignació a una variable** per a guardar-la.

Per a especificar la **lectura d'una dada** per teclat, en pseudocodi ho farem així:

llegir(<*variable*>)

A partir d'aquest moment, la variable assolirà el valor llegit per teclat.

D'altra banda, quan vulguem que la màquina **mostri un valor** a l'usuari, ho farà a través de la **pantalla del terminal** en forma de text.

Per a mostrar un valor per pantalla, en pseudocodi ho farem així:

escriure(*"Text"*) o també: **escriure**(*<variable>*)

Podem mostrar un **text literal** usant comes dobles o també podem mostrar **el valor d'una variable** en concret.
Si volem fer les dues coses alhora, podem fer ús de la **concatenació**.
Concatenar vol dir ajuntar diferents fragments de text en un de sol.

Exemple:

```
x ← 3
escriure("El valor de la variable és: ", x)
```

6. Entrada / Sortida: exemple

Exemple de I/O:

```
INICI
    escriure("Benvingut/da al programa!")
    escriure("Escriu un número entre 1 i 3")
    llegir(x) //S'assignarà el valor llegit per teclat a la variable x
    y ← x * 2
    escriure("El resultat de multiplicar ", x, " per 2 és ", y)
FINAL
```

Condicionals

Els algoritmes es programen usant combinacions de:

1. Estructures condicionals
2. Estructures iteratives

Les **estructures condicionals simples** defineixen accions a fer en funció d'una condició lògica que té resposta binària (*sí* o *no*).
Per exemple: **si plou**, **aleshores agafarem el paraigües**.

En pseudocodi, ho escriurem així:

```
si (<condició lògica es compleix>) aleshores
        // acció
si no
        // acció alternativa
fi
```

Desenvolupem l'exemple:

Si plou, **aleshores agafarem el paraigües**.

```
INICI
    si (plou = TRUE) aleshores
        paraigües ← 1
    si no
        paraigües ← 0
    fi
FINAL
```

També ens pot interessar dissenyar **condicionals** amb **resposta múltiple** en funció del valor d'una variable. En direm *switch-case.*

En pseudocodi, ho escriurem així:

```
llegir(<variable>)
segons (<variable>):
        cas 1:
                // acció 1
        cas 2:
                // acció 2
        cas n:
                // acció n
fi
```

Nota: Hem de saber el valor de la *<variable>* abans de començar el switch-case, en cas contrari, l'algoritme no sabrà per on ha de continuar.

4. Condicionals II: *switch-case* exemple

Desenvolupem l'exemple d'un **menú**:

```
INICI
   escriure("Menú d'opcions. Escull 1, 2 o 3.")
   llegir(opció)
   segons (opció):
         cas 1:
               escriure("Has triat l'opció 1.")
         cas 2:
               escriure("Has triat l'opció 2.")
         cas 3:
               escriure("Has triat sortir.")
   fi
FINAL
```

Estructures iteratives

Les estructures iteratives ens serveixen per programar **accions repetitives**. Veurem **tres casos d'estructures iteratives**.

Per exemple, quan caminem, farem avançar una cama i després l'altre repetidament fins a arribar a un lloc.

Quan **no sabem quantes iteracions** haurem de fer a priori, direm que es tracta d'un *while* i en pseudocodi ho escriurem així:

```
mentre (<condició lògica es compleix>) fes
        //accions
fi
```

Com reescriuries l'exemple de caminar fins a un lloc en pseudocodi?

5. Estructures iteratives: *while* exemple

Com reescriuries l'exemple de **caminar des d'un punt A fins a un punt B** en pseudocodi tenint en compte que A parteix de la posició 0 i que no sabem a quina posició es troba B?


```
INICI
    mentre (A != B) fes
        A ← A + 1
    fi
FINAL
```

Nota: *Fixem-nos que a dins de la condició lògica hem usat* != *per a especificar una desigualtat.*
També ho podríem representar usant <>, o usant la negació NOT= o negant tota la igualtat (NOT(A = B)).

També es pot donar el cas de que, tot i que **no sabem quantes iteracions** acabarem fent, **com a mínim en farem una**. Aquest tipus d'estructura iterativa l'anomenarem *do-while*. *("Dispara primer, pregunta després")*

En pseudocodi, ho escriurem així:

```
fes:
        //accions
mentre (<condició lògica es compleix>)
```

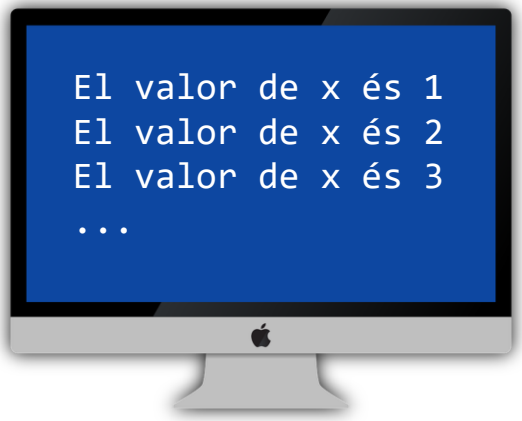
5. Estructures iteratives II: *do-while* exemple

A continuació veiem un exemple d'estructura iterativa *do-while:*

```
INICI
    x ← 1

    fes:
        escriure("El valor de x és ", x)
        x ← x + 1
    mentre (x < 10)

FINAL
```



Què imprimirà per pantalla?

Quan **sabem del cert quantes iteracions** hem de fer, usarem un *for.*

En pseudocodi, ho escriurem així:

```
per(i = <n> fins a <m>):
        //accions
fi
```

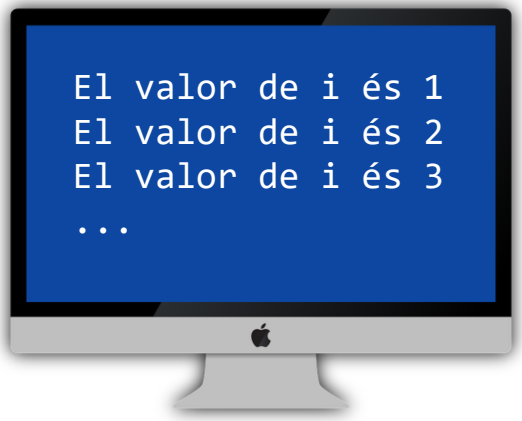
La variable i l'anomenarem índex. Li **assignarem** un **valor inicial** *<m>* i **s'incrementarà en 1 automàticament a cada pas d'iteració**.

El valor *<n>* representa l'últim valor que prendrà la variable i.

Nota: *La variable i* ***no sempre*** *ha de començar pels valors 0 o 1.*

A continuació veiem un exemple d'estructura iterativa *for:*

```
INICI
    per(i = 1 fins a 10):
        escriure("El valor de i és ", i)
    fi
FINAL
```



Què imprimirà per pantalla?

Crida de funcions

7. Crida de funcions

En programació, entenem el concepte de **funció** com a un subprograma que realitza un petit conjunt de tasques concretes relacionades entre elles i que pot ser executat múltiples vegades de forma independent.
A vegades, quan escrivim el pseudocodi d'un programa, no ens interessa desenvolupar tota la lògica d'una acció concreta perquè no ens cal entrar al detall o perquè és una part del codi que la dissenyarà una altra persona. Ens limitem a cridar l'acció.

En pseudocodi, farem crides a funcions d'aquesta manera:

```
INICI
    si (plou = TRUE) aleshores
        comprar_paraigües()
    fi
FINAL
```

“La felicitat té a veure amb saber escoltar-se a un mateix“

EMELIE FORSBERG (1986, Suècia)