

Tema 2

Programació estructurada. Disseny d'algorismes

1. Concepte de programació estructurada
- 2 Algorismes
- 3 Elements d'un algorisme
- 4 Comptadors, acumuladors i interruptors
- 5 Alguns algorismes bàsics
- 6 Exercicis

1. Concepte de programació estructurada

És un conjunt de tècniques de programació que incorporen:

- **Disseny descendent:** tècnica consistent en descompondre successivament accions complexes en accions més simples. Divideix i venceràs. Funcions.
- **Estructures de control:** descriuen el flux d'execució d'una successió d'accions:
 - ✓ **Seqüencial:** s'executen les ordres de dalt cap a baix.
 - ✓ **Bifurcació:** executar un o altre conjunt d'instruccions, depenent d'alguna condició.
 - ✓ **Repetició (o bucle):** repetir un conjunt d'instruccions mentre es compleix una condició.

2 Algorismes

Què és un algorisme?

És una descripció clara i no ambigua de les accions necessàries per a solucionar un problema en un ordre determinat. És com un programa però no està escrit en cap llenguatge de programació en concret. Serveix per a indicar els distints passos que ha de tindre el programa sense entrar en detall.

Quins elements té un algorisme?

- Instruccions:
 - D'entrada
 - D'eixida
 - D'assignació
- Estructures de control:
 - Bifurcacions
 - Repeticions

Com es fa un algorisme?

S'ha d'usar un mètode independent de qualsevol llenguatge de programació. Hi ha diverses maneres, però en vorem només dos:

- ✓ **Diagrama de flux (o ordinograma):** eina gràfica on hi ha uns símbols (que representen accions) units per fletxes (que indiquen l'ordre d'execució).
- ✓ **Pseudocodi:** combinació entre llenguatge natural i llenguatge de programació.

Qualitat d'un algorisme

Per a resoldre un problema determinat hi pot haver infinitat d'algorismes. La qualitat d'un algorisme depèn de:

- Senzillesa: l'algorisme ha de ser el més senzill possible
- Eficiència: cal minimitzar els recursos que calen per a executar-lo. Els recursos que cal minimitzar són: temps i memòria (RAM i disc dur).

3 Elements d'un algorisme

3.1 Instruccions d'entrada i d'eixida

Servixen perquè el programa intercanvie informació amb un medi extern (generalment teclat i pantalla).

Entrada

En una operació d'entrada (o lectura) el programa assigna a una variable un valor introduït per teclat (o ratolí o des d'un fitxer o base de dades...)

Eixida

En una operació d'eixida (o escriptura) el programa mostrarà una dada (una constant o el valor d'una variable o el resultat d'una expressió) per pantalla (o per impressora o la deixarà en un fitxer o base de dades...).

INSTRUCCIONS D'ENTRADA I D'EIXIDA	
PSEUDOCODI	DIAGRAMA DE FLUX
<pre>Algoritmo area_del_cercle Escribir "Dis-me el radi:" Leer radi Escribir "Area: ", (3.14*radi*radi) FinAlgoritmo</pre>	<pre>graph TD Start([Algoritmo area_del_cercle]) --> Input[/Dis-me el radi:/] Input --> Process[radi] Process --> Output[/Area: ', (3.14*radi*radi)/] Output --> End([FinAlgoritmo])</pre>

Exercicis: instruccions d'entrada i d'eixida

1. Demana 2 números per teclat i mostra la suma, resta, multiplicació i divisió.

Nota: descarrega't el programa Pseint per a provar els teus algorismes, tant en pseudocodi com en diagrames de flux.

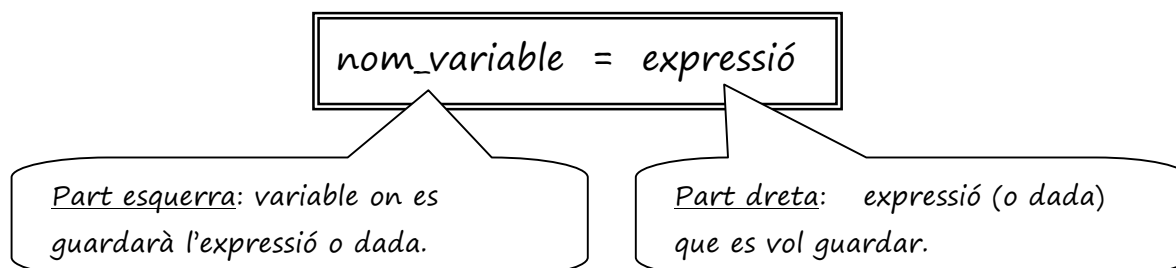
3.2 Instruccions d'assignació

Una assignació consisteix en guardar un valor en una variable.

INSTRUCCIONS D'ASSIGNACIÓ	
PSEUDOCODI	DIAGRAMA DE FLUX
<pre>Algoritmo notes nota1 = 5 nota2 = 7 notaMitja = (nota1 + nota2) / 2 Escribir notaMitja FinAlgoritmo</pre>	<pre>graph TD Start([Algoritmo notes]) --> A[nota1 ← 5] A --> B[nota2 ← 7] B --> C[notaMitja ← (nota1+nota2) / 2] C --> D[/notaMitja/] D --> End([FinAlgoritmo])</pre>

En la primera instrucció de l'exemple anterior, està posant el valor 5 dins de la variable nota1. En la segona instrucció posa un 7 en la variable nota2. I en la tercera, suma les dos variables, les divideix entre 2 i el resultat el posa dins la variable notaMitja.

És a dir, una assignació consta de 2 parts separades per un operador d'assignació. Sol usar-se l'operador "=" (o bé una fletxeta cap a l'esquerra: "←"):



Consideracions:

- ✓ El resultat de l'expressió que es vol guardar deu ser del mateix tipus que la variable on es guarda.
- ✓ Cal tindre en compte que les assignacions **NO** són equacions matemàtiques.

Per exemple, $x = x + 1$ no té sentit com a equació però sí com a instrucció en un algorisme. És el que anomenem un increment de la variable. En eixa instrucció se li assigna a la variable numèrica x un valor que correspon al valor que tenia abans eixa variable més una unitat. És a dir:

4

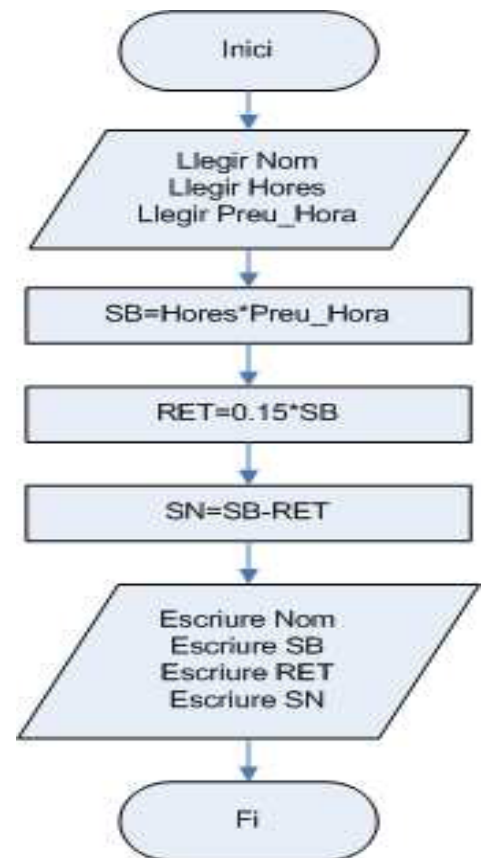
$x = 0;$	// ara x val 0
$x = x + 1;$	// ara x val 1
$x = x + 7;$	// ara x val 8

Exercicis: instruccions d'assignació

2. Fes el diagrama de flux per al càlcul del salari d'un treballador tenint en compte estes consideracions:

- El programa demanarà per teclat el nom del treballador, la quantitat de hores que ha treballat i el preu per hora.
- La retenció aplicada és del 15%.
- Es mostrarà per pantalla:
 - nom del treballador
 - sou brut (hores * preu)
 - import retingut (15% sobre el sou brut)
 - sou net (sou brut menys l'import retingut)

Nota: En este gràfic està el diagrama de flux que soluciona l'exercici, però està en altra notació. Fes-ho en la notació del programa Pseint.

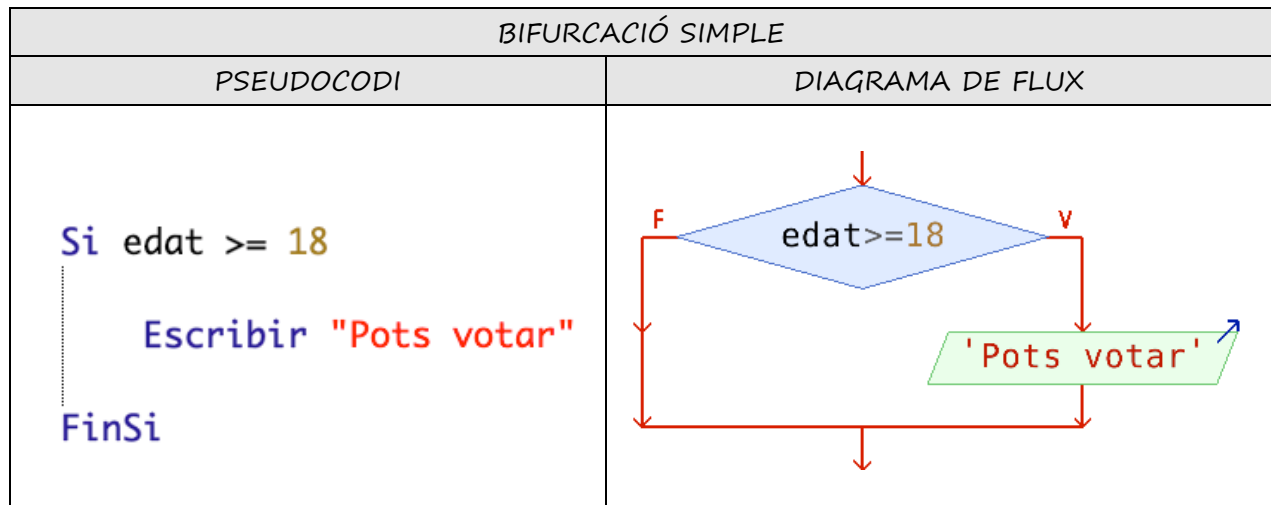


3.3 Instruccions de bifurcació

Les instruccions de bifurcació (o selecció) serveixen per a quan volem executar un conjunt d'ordres només si es compleix alguna condició determinada.

BIFURCACIÓ DOBLE	
PSEUDOCODI	DIAGRAMA DE FLUX
<pre> Si edat >= 18 Escribir "Pots votar" Sino Escribir "No pots votar" FinSi </pre>	<pre> graph TD Entry(()) --> Decision{edat >= 18} Decision -- F --> NoPotsVotar[/No pots votar/] Decision -- V --> PotsVotar[/Pots votar/] NoPotsVotar --> Exit(()) PotsVotar --> Exit </pre>

En el pseudocodi és opcional posar la part del “si_no”. En l'ordinograma podem no posar res en alguna de les 2 branques:



També podem posar instruccions de bifurcació dins d'altres.

Exercicis resolts: instruccions de bifurcació

3. Fes un algorisme (mitjançant un diagrama de flux) que calcule l'àrea d'un cercle a partir del radi, controlant que el valor del radi que es dona siga positiu.

Algoritmo areaCercleOK

Escribir "Dis-me el radi:"

Leer radi

si radi > 0 Entonces

area = 3.14 * radi * radi

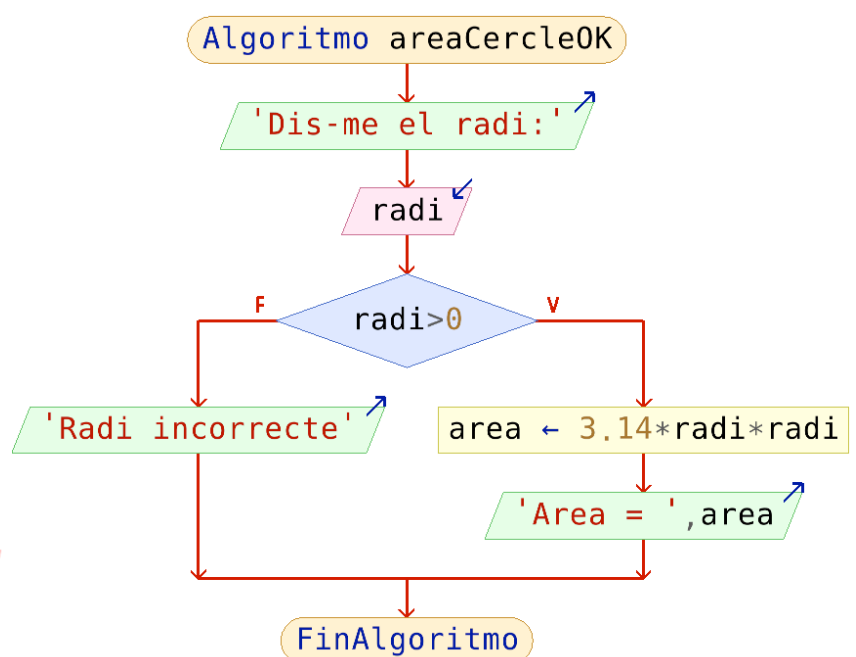
Escribir "Area = ", area

Sino

Escribir "Radi incorrecte"

FinSi

FinAlgoritmo

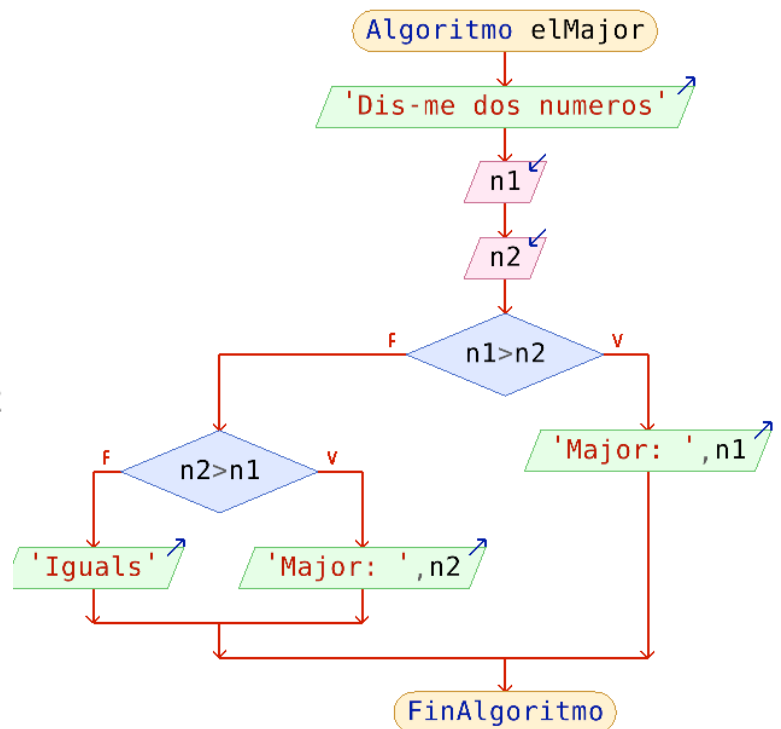


4. Fes un algorisme que llisti 2 números i que mostre quin és el major; o bé, si és el cas, que diga que són iguals.

```

Algoritmo elMajor
Escribir "Dis-me dos numeros"
Leer n1
Leer n2
si n1 > n2 Entonces
    Escribir "Major: ", n1
Sino
    si n2 > n1 Entonces
        Escribir "Major: ", n2
    Sino
        Escribir "Iguals"
    FinSi
FinSi
FinAlgoritmo

```

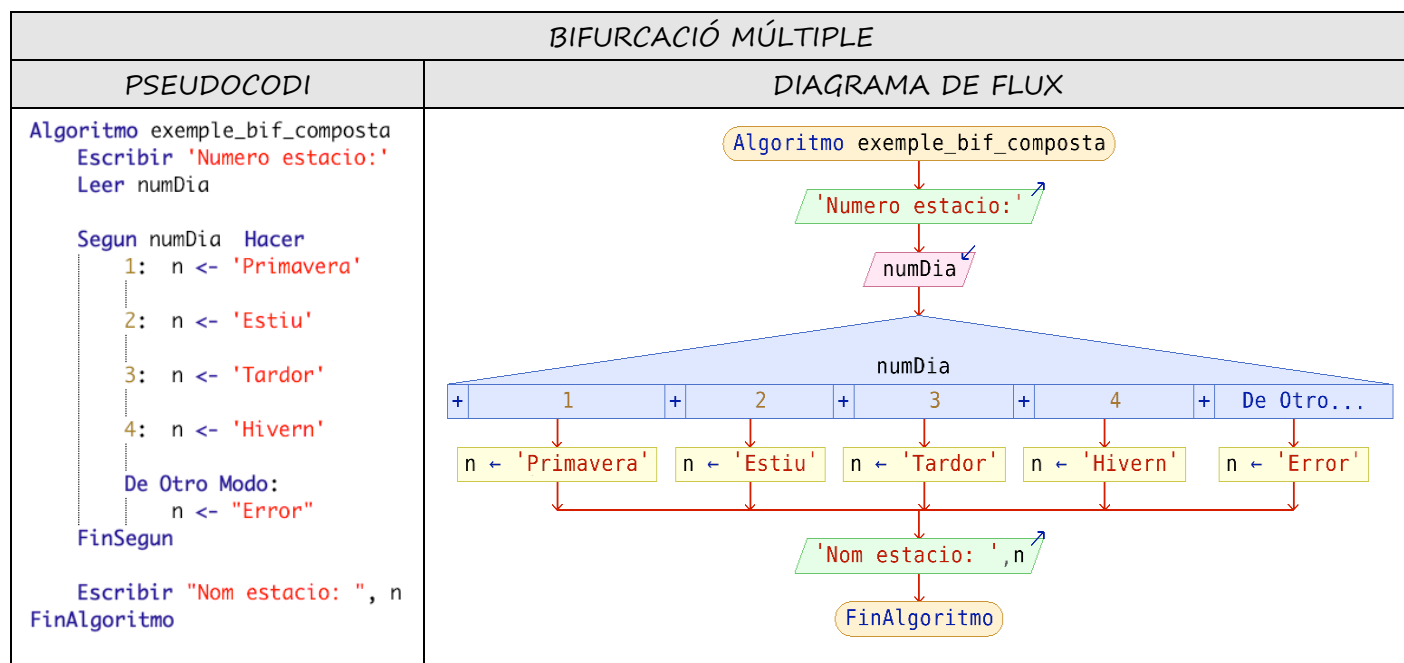


Exercicis: instruccions de bifurcació

5. Fes un diagrama de flux per a llegir un número de teclat i dir si és parell. No s'ha de dir res en cas contrari.
- 6. Demanar un número i dir si és parell o imparell.
- 7. Donats 2 números, calcular quin és el més gran
- 8. Donats 3 números, calcular quin és el més gran
- 9. Donats 3 números, calcular quin és el més gran i el més menut
- 10. Donats 3 números, calcular quins són els 2 més menuts
- 11. Donats 3 números, comprova si poden correspondre a les mesures dels costats d'un triangle (pista: la suma dels 2 més xicotets ha de ser major que el gran).
12. Transformar una nota numèrica a la forma: Molt deficient, Insuficient, Suficient, Bé, Notable, Excel·lent. També cal mostrar error si la nota és negativa o >10.
13. Llegir dos números de teclat i una lletra, que serà el codi d'operació (Suma/Resta/Multiplicació/Divisió). Caldrà mostrar el resultat de l'operació demanada. Si no s'ha introduït un codi d'operació correcte, cal mostrar un error.

Instruccions de bifurcació múltiple

En la bifurcació simple, si es complia una condició executàvem un bloc d'accions. I, en cas contrari, altre bloc. Amb una instrucció de bifurcació múltiple podrem executar més de 2 blocs d'accions.



En l'exemple, veiem que si el valor de *numDia* és 1, s'executarà la instrucció (o conjunt d'instruccions) corresponent. Si fóra 2, les instruccions del 2, etc. I si el valor de *numDia* no fóra ni 1, ni 2, ni 3, ni 4, s'executaria el bloc d'instruccions de l'apartat "De Otro Modo".

Nota: les bifurcacions simples i dobles depenen del resultat d'una expressió lògica (vertader o fals) però les bifurcacions múltiples depenen del valor d'una variable o expressió (de tipus enter o caràcter).

Exercicis: instruccions de bifurcació composta

14. Fes un diagrama de flux per a llegir un número de teclat corresponent al dia de la setmana (1 al 7) i mostrar el nom corresponent (dilluns, dimarts...).

3.4 Instruccions de repetició

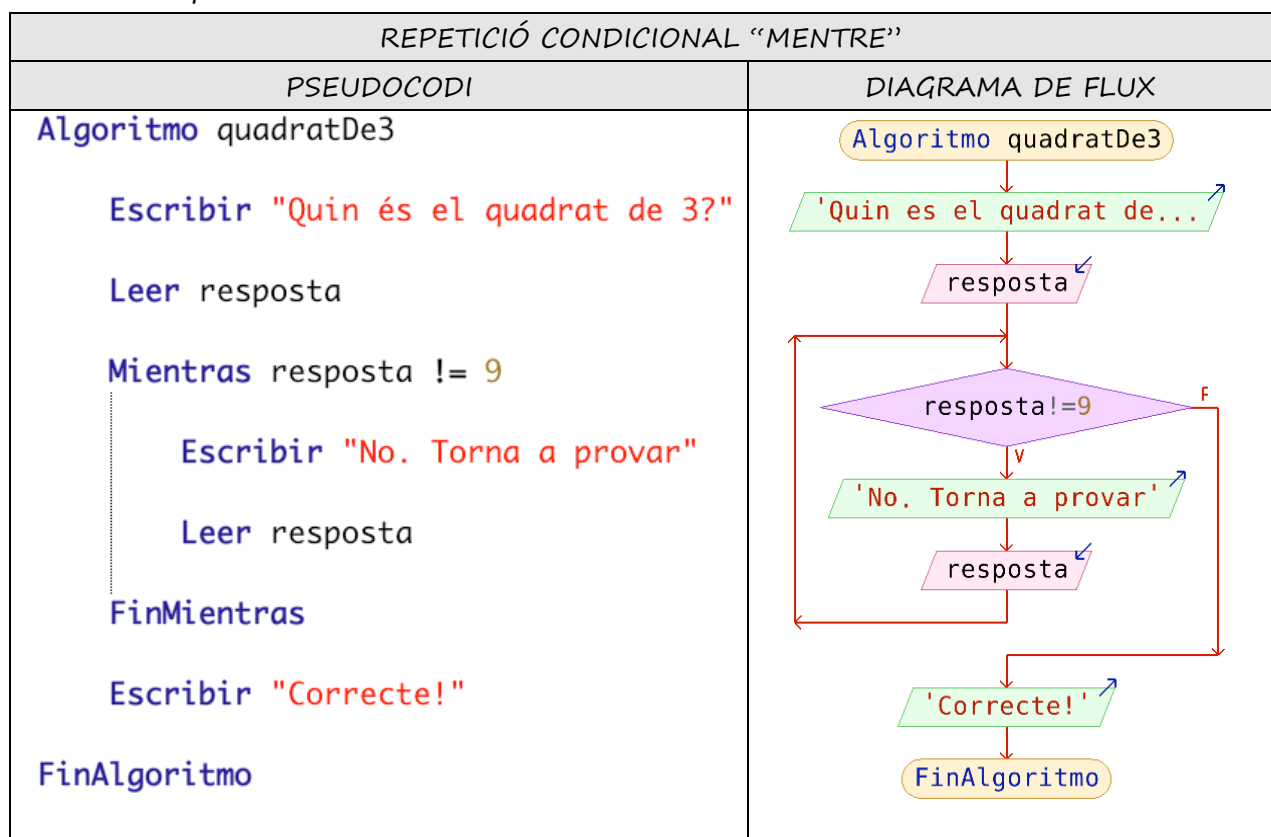
Amb les estructures de repetició podrem fer que un grup de sentències s'execute diverses vegades. Quantes?

- Mentre es complisca una condició: repeticions condicionals
- Un nombre determinat de vegades: repeticions incondicionals

3.4.1 Repetició condicional “Mentre”

Amb l'estructura “MENTRE”, posarem en un bloc aquelles instruccions que volem que s'executen repetidament mentre es complisca una determinada condició.

Exemple:



Exercicis: instruccions de repeticions condicionals “mentre”

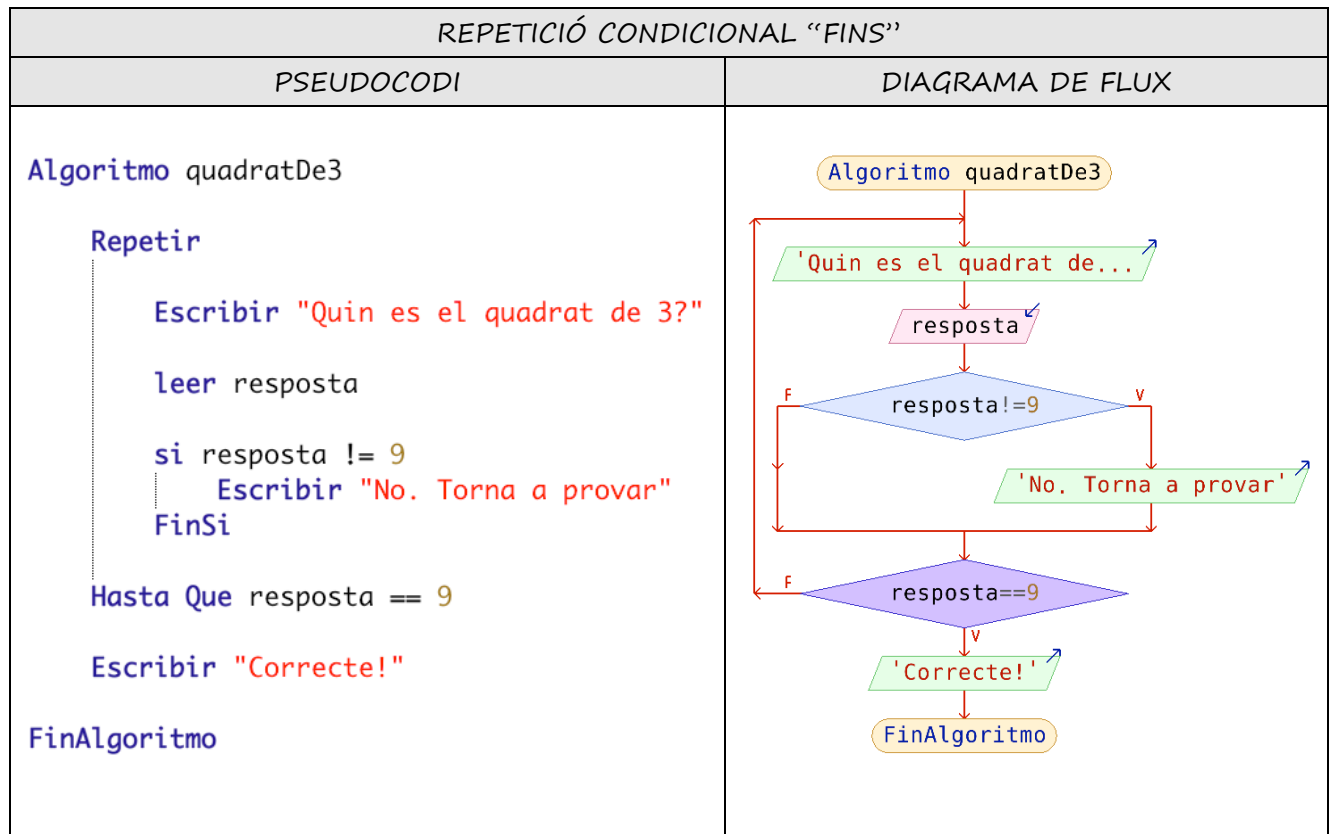
15. Fes un algorisme tal que, donat el radi, calcule l'àrea del cercle, però demanant repetidament el radi fins que l'usuari ens done un radi positiu.
16. Demana any de naixement i de mort d'una persona. Caldrà demanar-ho repetidament fins que siguin dades coherents. És a dir: la data de defunció no pot ser anterior a la de naixement.

3.4.1 Repetició condicional “Fins”

Amb l'estructura “FINS”, posarem en un bloc aquelles instruccions que volem que s'executen repetidament fins que es complisca una determinada condició.

És paregut al bucle “MENTRE”. Vegem-ho amb un exemple.

Exemple:



A l'exemple veiem que, a diferència del bucle *MENTRE*, en el bucle *FINS*:

- Sempre s'entra al bucle almenys 1 vegada, ja que la condició està al final del bucle.
- La condició no és de continuar el bucle, sinó la condició per a eixir del bucle. Per tant la condició serà la contrària de la que posaríem en un *Mentre*.

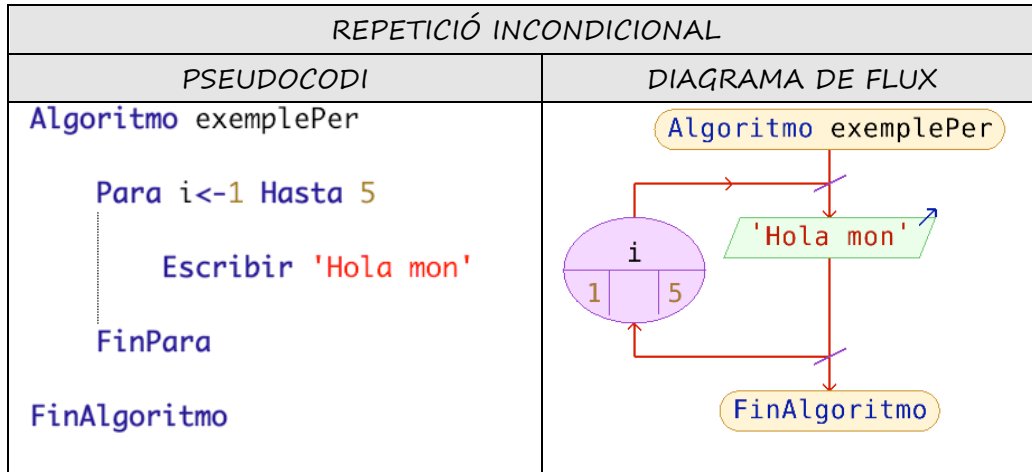
Exercicis: instruccions de repeticions condicionals “fins”

Fes els exercicis de l'apartat anterior però usant bucle de tipus “fins”

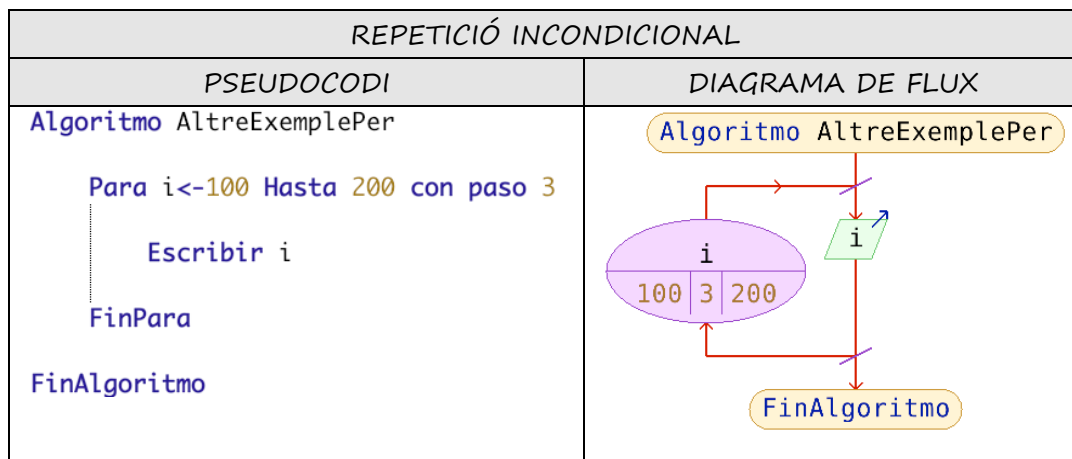
3.4.3 Repetició incondicional ("per")

Amb l'estructura "PER" (for als llenguatges de programació), posarem en un bloc les instruccions que volem que s'executen una quantitat de voltes determinada.

Exemple 1: Volem mostrar 5 voltes "Hola, món!"



Exemple 2: mostrar els números que hi ha entre el 100 i el 200, de 3 en 3.



Exercicis: instruccions de repeticions incondicionals

17. Demanar per teclat quants números es volen mostrar. A continuació, es mostraran els números des d'eixe número fins a l'1 (en eixe ordre).
18. Programa que demane un valor inicial i un valor final. Caldrà mostrar els valors que hi ha entre l'inicial i el final però de 3 en 3. Cal tindre en compte que el valor inicial pot ser major que el final. És a dir:

Exemples: $vi=10 \quad vf=20 \quad \rightarrow$ Mostrarà: 10, 13, 16, 19
 $vi=20 \quad vf=10 \quad \rightarrow$ Mostrarà: 20, 17, 14, 11

19. Imprimix la taula de multiplicar del 9.

4 Comptadors, acumuladors i interruptors

A vegades, les variables s'utilitzen per a unes finalitats concretes. Anem a veure quines són eixes finalitats i com cal tractar eixes variables.

4.1 Comptadors

Un comptador és una variable destinada a comptar quantes vegades ocorre alguna cosa. Sol usar-se dins de bucles (de qualsevol tipus: "mentre", "fins" o "per").

L'ús del comptador és així:

- Abans del bucle: el comptador s'inicialitza (generalment a 0).
- Dins del bucle: el comptador s'autoincrementa

```
comptador = 0
BUCLE
...
    comptador = comptador + 1
...
FI_BUCLE
```

Exemple: volem saber quants números negatius introduïm

ÚS DE COMPTADORS	
PSEUDOCODI	DIAGRAMA DE FLUX
<pre>Algoritmo ExempleComptador cNeg = 0 Para i<-1 Hasta 10 leer num si num<0 cNeg = cNeg + 1 FinSi FinPara Escribir cNeg, " negatius" FinAlgoritmo</pre>	<pre>graph TD Start([Algoritmo ExempleComptador]) --> Init[cNeg ← 0] Init --> Read[/num/] Read --> Cond{num < 0} Cond -- V --> Inc[cNeg ← cNeg + 1] Cond -- F --> Loop((i: 1 to 10)) Inc --> Loop Loop --> Print[/cNeg, 'negatius'/] Print --> End([FinAlgoritmo])</pre>

Exercicis: ús de comptadors

20. Abans has fet els exercicis de bucles incondicionals. Torna'ls a fer però ara amb bucles condicionals.
21. Indica quants divisors (no quins) té un número donat.
22. Pregunta quina és l'arrel quadrada de 225 fins que siga encertat. Finalment, mostra quants intents s'han fet.
23. Imprimix quants números hi ha entre 1 i 100 que són múltiples de 2, quants múltiples de 3 i quants múltiples de 2 i de 3 al mateix temps.
24. Llig uns quants números (fins que posem el 0). Mostra quants positius, quants negatius i quants acaben en 0.
25. Fes un diagrama de flux que calcule quants dígets té una xifra donada.

Pista: cal anar dividint la xifra entre 10 mentre es pugui i comptar quantes vegades s'ha dividit. Per a fer una divisió sense decimals pots usar la funció trunc:

Solució:

Algoritmo quantitatDígets

```
leer num

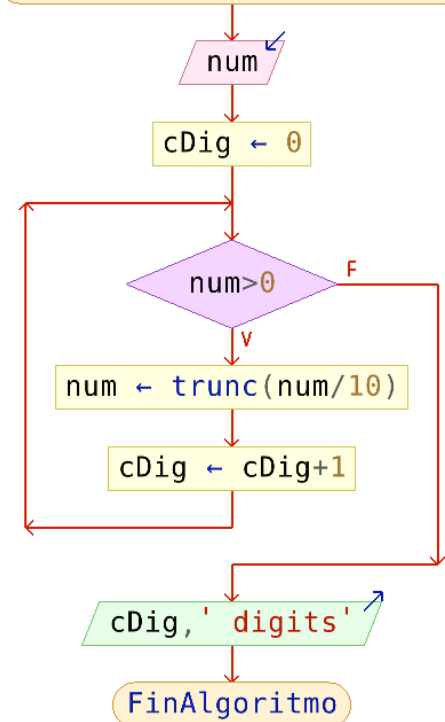
cDig = 0

mientras num > 0
    num = trunc(num / 10)
    cDig = cDig + 1
FinMientras

escribir cDig, " dígets"

FinAlgoritmo
```

Algoritmo quantitatDígets



4.2 Acumuladors

Un acumulador és una variable destinada a acumular diferents quantitats.

Un acumulador és com un comptador però en compte de sumar 1, sumarem diferents quantitats:

```

acumulador = 0
BUCLE
...
    acumulador = acumulador + quantitat
...
FI_BUCLE
    
```

Exemple: volem acumular l'import d'una factura

ÚS D'ACUMULADORS	
PSEUDOCODI	DIAGRAMA DE FLUX
<p>Algoritmo exempleAcumulador</p> <p>Escribir "Quantitat:" leer q</p> <p>total = 0</p> <p>Mientras q != 0</p> <p> Escribir "Preu:" Leer p</p> <p> total = total + (q*p)</p> <p> Escribir "Quantitat:" leer q</p> <p>FinMientras</p> <p>Escribir "Total: ", total</p> <p>FinAlgoritmo</p>	<pre> graph TD Start([Algoritmo exempleAcumulador]) --> Input1[/Quantitat:/] Input1 --> Var1[/q/] Var1 --> Init1[total ← 0] Init1 --> Decision{q != 0} Decision -- F --> Output1[/Total: ', total/] Decision -- V --> Input2[/Preu:/] Input2 --> Var2[/p/] Var2 --> Process1[total ← total + (q*p)] Process1 --> Input3[/Quantitat:/] Input3 --> Var3[/q/] Var3 --> Decision Output1 --> End([FinAlgoritmo]) </pre>

Exercicis: ús d'acumuladors

26. Demana quantitats fins que s'introdueixi la quantitat 0. Caldrà mostrar la suma de totes les quantitats.
27. Demana les notes dels 23 alumnes de la classe. Mostra la nota mitja.
28. Mostra els números naturals que hi ha entre 2 números introduïts per teclat i calcula la suma dels parells i la dels imparells. Per últim, mostra els 2 totals.
29. Introdueix 2 valors A i B ($A < B$). Incrementa A de 2 en 2 i decrementa B de 3 en 3 fins que $A > B$.
30. Demana 2 números per teclat i mostra la multiplicació dels dos... però sense usar l'operador de la multiplicació (*). És a dir: hauràs de sumar un dels dos números tantes vegades com diu l'altre número.

4.2.1 Acumuladors de productes

Generalment, la quantitat va sumant-se a l'acumulador, però també podria multiplicar-se. Cal tindre en compte això per a iniciar l'acumulador. Així tenim que, generalment:

- a) Si volem sumar quantitats, el valor_inicial deu ser 0.
- b) Si volem multiplicar quantitats, el valor_inicial deu ser 1.

Exercici resolt: ús d'acumuladors de productes

31. Donats 2 números (base i exp) calcula la potència ($base^{exp}$). Se suposa que la potència no és un operador ni cap funció predefinida.

Algoritmo exempleAcumProductos

Escribir 'Base:'

Leer base

Escribir 'Exponent:'

Leer expo

pot ← 1

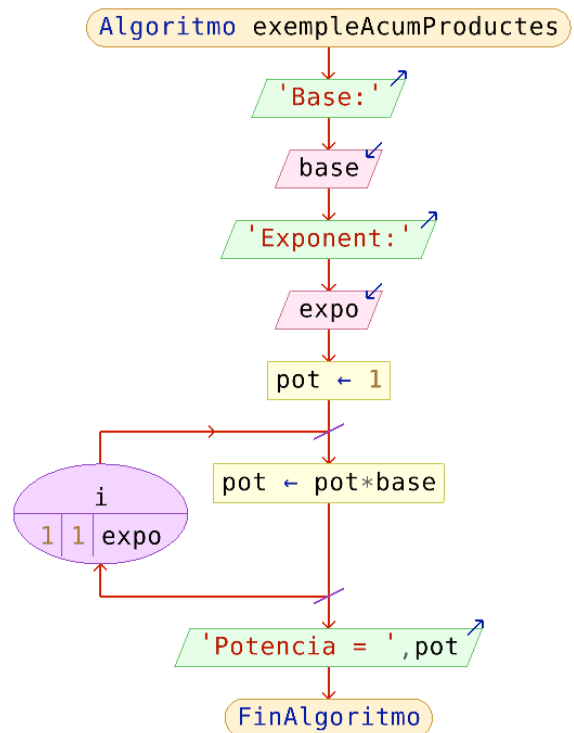
Para i ← 1 Hasta expo con paso 1

 pot ← pot * base

FinPara

Escribir 'Potencia = ', pot

FinAlgoritmo



Exercicis: ús d'acumuladors de productes

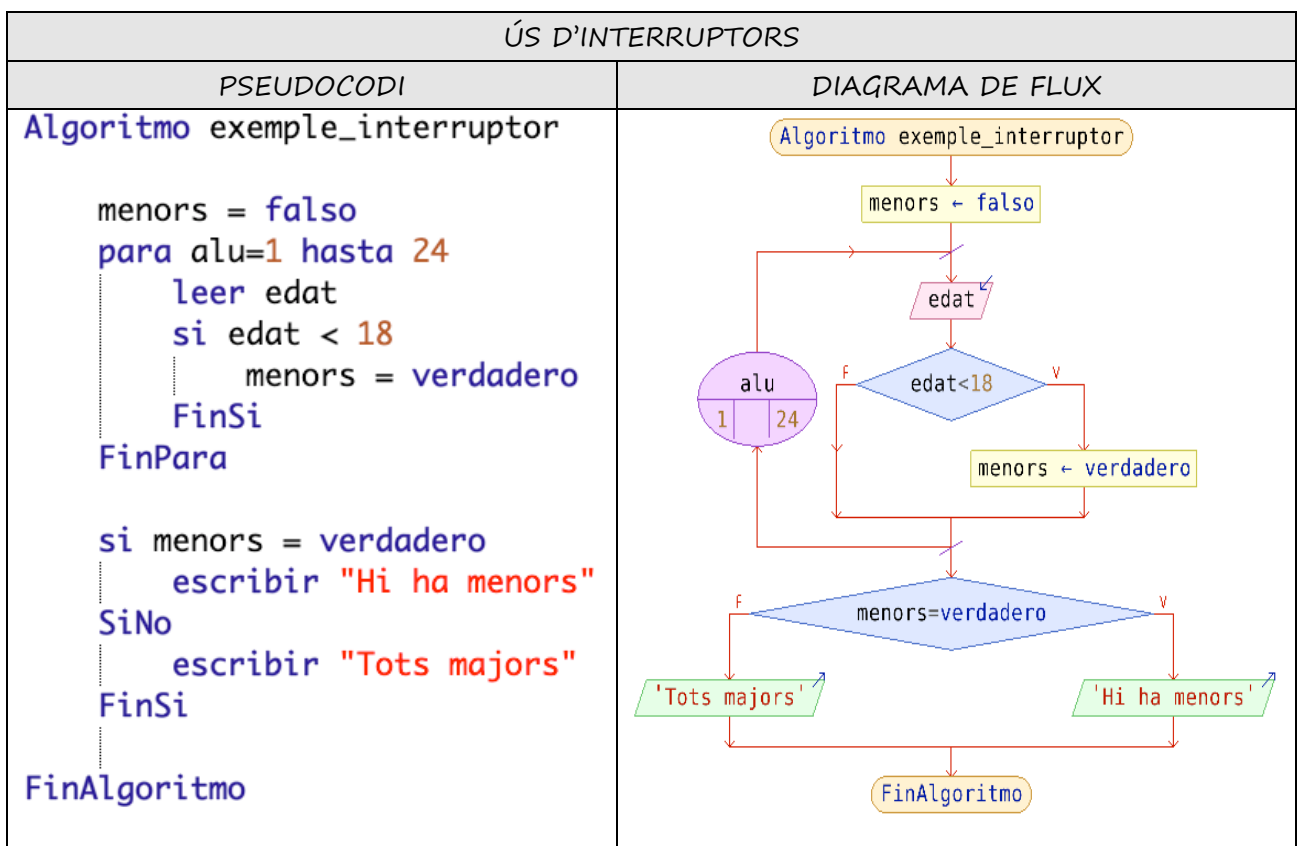
- 32. Mostra el producte de tots els números imparells entre l'1 i el 40.
- 33. Mostra la suma, el producte i la mitjana dels 100 primers números naturals.
- 34. Calcula el factorial d'un número introduït per teclat.

4.3 Interruptors

Els interruptors (també coneguts com commutadors, switches, indicadors, banderes o flags) són variables destinades a indicar si en alguna de les iteracions d'un bucle (mentre, fins que, per a) **ha passat o no** una cosa determinada.

Les variables que fan d'interruptor només estan destinades a guardar 2 possibles valors: ha passat una cosa / no ha passat una cosa.

Per això solen ser variables booleanes (lògiques), ja que només poden guardar 2 valors: vertader o fals. En Pseint, els possibles valors són *verdadero* i *falso*. Ja vorem que en Python és *True* i *False*, i en Java (i C++) són *true* i *false* (en minúscula).



Però si un llenguatge no admetera el tipus booleà (com és el llenguatge C), podríem usar una variable de tipus enter (per exemple usant els valors 0 i 1) o de tipus caràcter (per exemple usant els valors 'V' i 'F', o bé, 'S' i 'N'...).

Exercicis: ús d'interruptors

35. Demana un número per teclat i digues si és un número primer o no.

Nota: és primer si és major que 1 i només és divisible per 1 i per ell mateix.

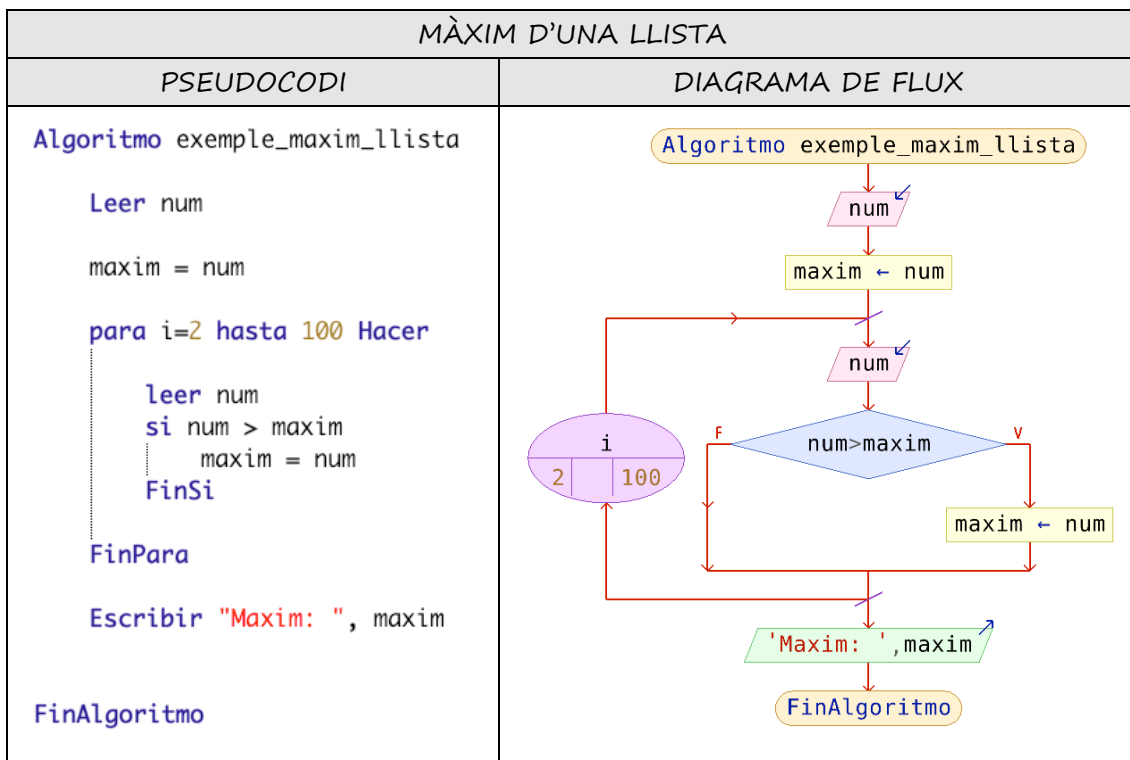
5 Alguns algorismes bàsics

5.1 Obtindre el major d'una llista de números

La idea consisteix en recórrer la llista i, en cada moment, tindre guardat en una variable el valor major recorregut fins eixe moment. Eixa variable li podem dir “màxim”.

Per a això, si volem obtindre el màxim de 100 valors, al principi de l'algorisme direm que el màxim és el primer d'eixos valors. Després recorrerem els 99 nombres restants: si algun d'ells és major que el que tenim en la variable “màxim”, haurem d'actualitzar el valor d'esta variable.

En eixir del bucle, ja tindrem en la variable “màxim” el valor que volíem.



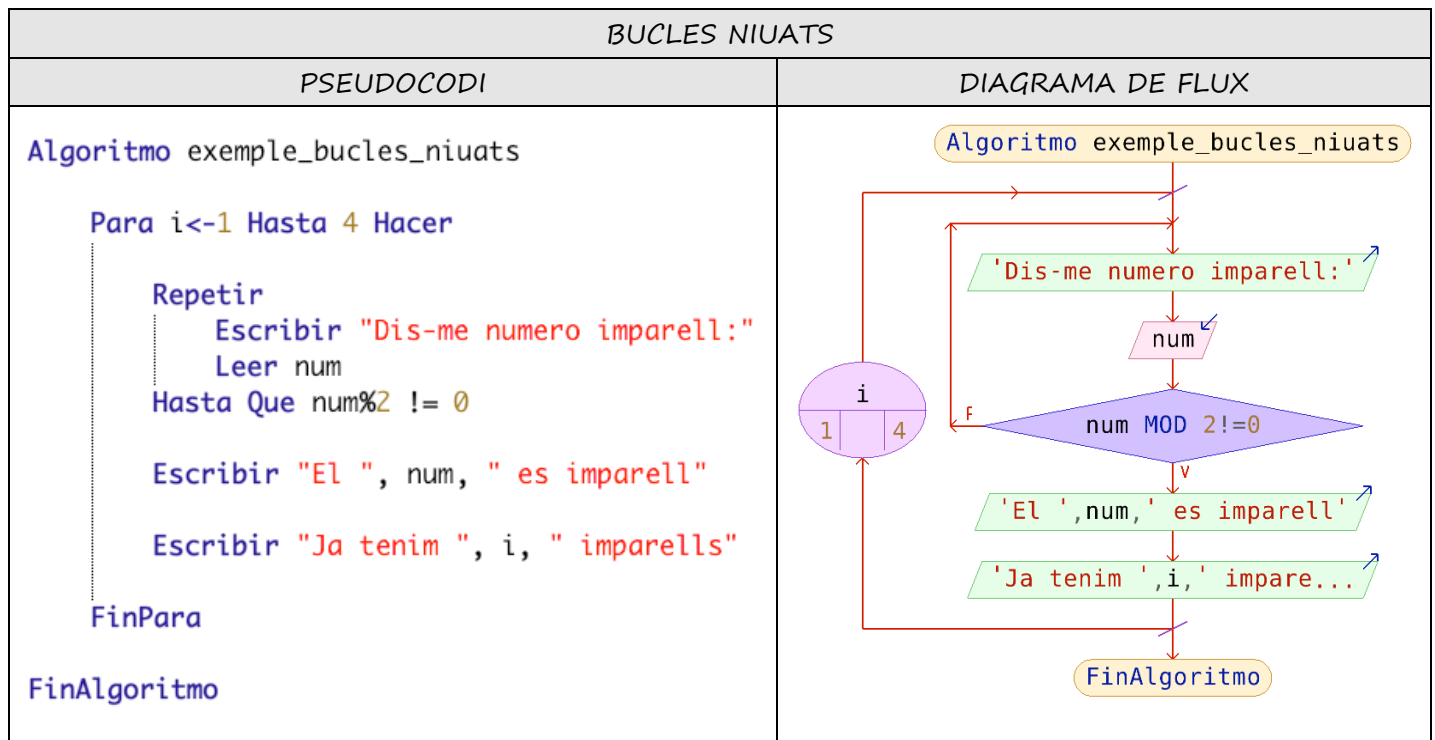
Exercicis: càlcul del major, etc

36. Llig uns quants números (fins que posem el 0). Mostra quin és el major i quin és el menor.

5.2 Bucles niuats

Podem posar un bucle (o més) dins d'un altre.

Per exemple, si volem demanar un número imparell i repetir el procés fins que el número introduït siga imparell, necessitarem un bucle. Però si volem demanar 4 números imparells, haurem de posar el bucle anterior dins d'un altre bucle.



Nota: L'operador "residu de divisió entera" es pot expressar en el programa Pseint de dos formes: amb % i amb mod.

Quan veiem el llenguatge Python, vorem els nucs niuats més detalladament. Mentrestant, intenta fer els següents exercicis amb algorismes.

Exercicis: bucles niuats

36.1 Dibuixa un quadrat de "*" de grandària n (demanada per teclat). Per exemple, si n és 4, cal dibuixar:

```

* * * *
* * * *
* * * *
* * * *

```

Nota: en Pseint, la instrucció Escribir ... fa un salt de línia després d'escriure. Si no volem eixe salt, cal usar Escribir sin saltar ...

36.2 Dibuixa un triangle de "*" d'altura n (demanada per teclat). Per exemple, si n és 4:

```
*
* *
* * *
* * * *
```

36.3 Dibuixa un triangle de "*" d'altura n (demanada per teclat). Per exemple, si n és 4:

```
* * * *
* * *
* *
*
```

36.4 Dibuixa un rectangle de "*" de grandària alt x ample (dades demanades per teclat). Per exemple, si és 4 d'alt i 7 d'ample, dibuixarem:

```
* * * * * * *
* * * * * * *
* * * * * * *
* * * * * * *
```

36.5 Igual que abans però un rectangle "buit":

```
* * * * * * *
*               *
*               *
* * * * * * *
```

37. Mostra les taules de multiplicar del 2 al 9.

Pista: ja havies fet amb un bucle, una taula de multiplicar. Ara es tracta de posar eixe tros de codi dins d'un altre bucle, ja que volem moltes taules de multiplicar.

37.1 Demana un número per teclat (n) i mostra per pantalla les següents línies:

```
1 = 1
1+2 = 3
1+2+3 = 6
...
(fins a n línies)
```

37.2 Mostra els 15 primers números primers: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47 (El 1 no és primer, per definició).

5 Exercicis

38 En un pàrquing es paga 2 € l'hora completa. I per als minuts restants, es paga a 4 cèntims el minut, però no pot excedir el preu d'una hora. Calcula què li toca pagar a un conductor per un determinat temps donat en minuts.

Nota: recorda que pots eliminar els decimals amb la funció trunc.

39 Donat un temps en segons, calcula els segons que falten per a convertir-se en minuts sencers. Per exemple, per a un temps de 70 segons, en faltarien 50.

40 Donat un temps en minuts, calcula els dies, hores i minuts que li corresponen.

41 Pregunta quina és l'arrel quadrada de 225 fins que siga encertat.

42 Càlcul del sou d'un treballador. Pregunta quantes hores ha treballat (a partir de les 40 hores es consideraran extres). Pregunta quin és el preu de l'hora normal (el preu de l'hora extra és un 50% més del preu de l'hora normal). Què cobra el treballador?

43 Modifica l'exercici de les hores extres per a obtindre la suma dels salaris de tots els treballadors. Tots la mateix tarifa. Acabarà quan posem 0 hores treballades.

44 Demana N notes d'un estudiant i calcula:

- a) Quantes notes té suspeses
- b) Quantes aprovades
- c) La mitjana de les notes
- d) La mitjana de les notes aprovades i la mitjana de les suspeses

45 Escriu un algorisme que calcule el total d'una factura d'un article determinat del qual s'adquireixen N unitats a un preu P. L'IVA és el 16%. Si l'import a pagar (amb IVA) és superior a 300 €, s'aplicarà un descompte del 5%.

46 Modifica l'exercici de la factura per a demanar moltes voltes el preu i la quantitat de diferents articles. Si les unitats introduïdes són 0, vol dir que no es demanaran més articles. El possible descompte s'aplicarà al final de la factura.

47 Algorisme que vaja demanant lletres des de teclat i que pare quan es trobe amb una vocal. Cal escriure esta vocal.

48 Mostra els números entre 100 i 200 que són múltiples de 3 però que no són múltiples de 2.

49 Mostra tots els divisors d'un número donat. Utilitza l'operador mod.

- 50 Demana per teclat la quantitat de números que vol introduir l'usuari. A continuació, llig de teclat eixa quantitat de números i digues de cadascun si és parell i positiu al mateix temps.
- 51 Imprimix la taula de multiplicar d'un número introduït per teclat.
- 52 Dir si un número és o no d'Armstrong (si és igual a la suma dels seus dígit al cub, p.e: $153=1^3+5^3+3^3$). Si no ho és, tornar-ho a intentar per a més números.
- 53 Algorisme que mostre tots els números d'Armstrong de l'1 al 1000.
- 54 Comprova si un número donat és perfecte. Nota: es diu que un número és perfecte si és igual a la suma dels seus divisors excepte ell mateix.
- 55 Mostra els números primers menors de 100.