

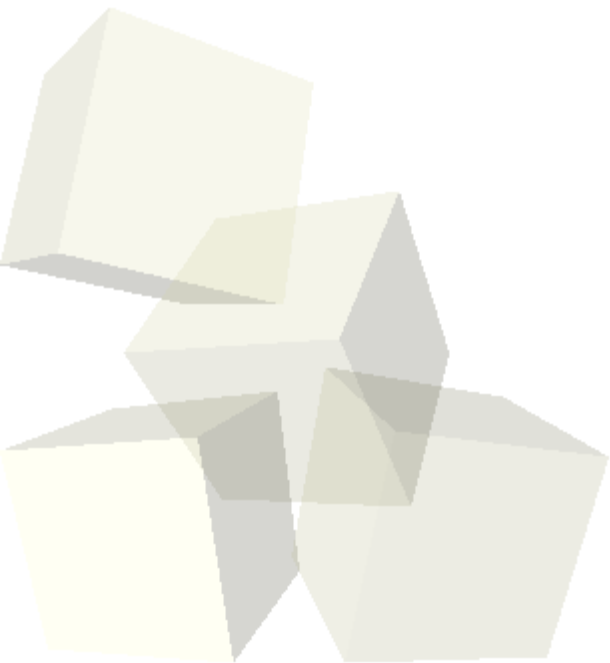


Tema 2

Algorismes i programes

Informàtica
Grau en Física
Universitat de València

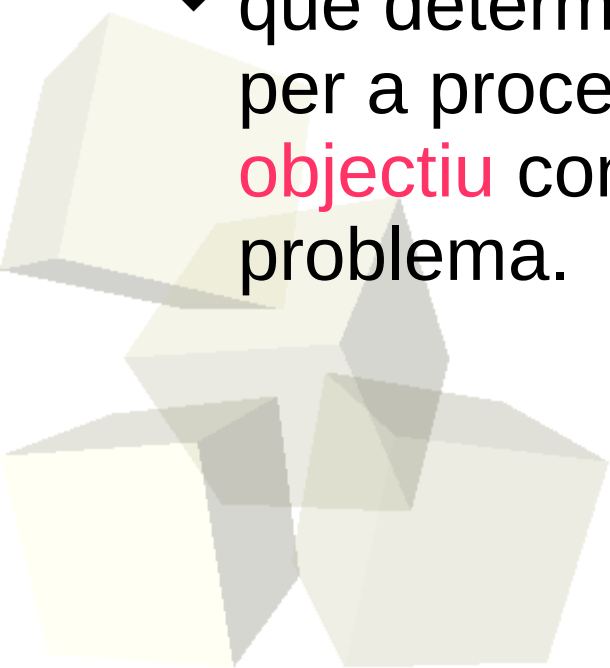
Francisco.Grimaldo@uv.es
Ariadna.Fuertes@uv.es





■ Un algorisme és:

- ♦ Una successió **finita** de **passos** o accions,
- ♦ especificats de manera **no ambigua**,
- ♦ que s'executen en un temps **finit** i
- ♦ que determinen quines operacions s'han de realitzar per a processar un conjunt de dades i assolir un **objectiu** concret, açò és, arribar a la resolució d'un problema.





Resolució d'un problema

■ Els passos a seguir són:

1) Anàlisi del problema:

- Definició del problema.
- Especificació de les entrades.
- Especificació de les eixides.

2) Disseny / cerca de l'algorisme:

- Selecció / millora de l'algorisme.
- Disseny modular o descendent.
- Refinament per passos.

3) Programació de l'algorisme:

- Codificació del programa en:
 - Pseudocodi.
 - Organigrames o diagrames de flux.

4) Traducció / execució / comprovació del programa.



Exemple: Cerca de nombres primers

- **Problema:** Cercar els nombres primers entre 2 i un cert valor MAX.

1) **Anàlisi del problema:**

- ♦ Què és un nombre primer? Quines entrades tenim? Què volem obtenir com a resultat?

2) **Cerca de l'algorisme:**

♦ Mètode 1:

1. $X = 2$
 2. $I = 2$
 3. Fer (X / I)
 4. Si I és menor que X i la divisió és entera, aleshores X **no és primer** i cal passar a (7).
 5. Si I és igual a X , aleshores X **es primer** i cal passar a (7).
 6. Incrementar I i passar a (3).
 7. Si X és menor que **MAX**, aleshores cal incrementar X i passar a (2).
- Millora de l'algorisme: Parar la divisió si I és major que $X/2$.

♦ Mètode 2: Garbell d'Eratòstenes

1. Posar tots els nombres entre 2 i **MAX** en una llista.
2. Si hi ha nombres sense cancel·lar, el primer d'ells és "**primer**".
3. Cancel·lar de la llista tots els múltiples del primer nombre.
4. Llevar el primer nombre de la llista.
5. Llevar els nombres cancel·lats de la llista i passar a (2).

3) **Programació** + 4) **Traducció / execució / comprovació.**



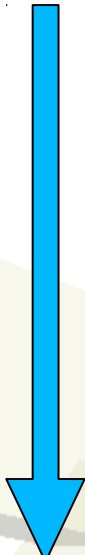
Exemple: Suma successió aritmètica

■ Anàlisi del problema:

- ♦ Descripció de la successió: $(a_1, a_2, a_3, \dots, a_n)$ on $a_i = a_1 + (i-1) \cdot d$
- ♦ Entrades: Primer terme (a_1), distància (d), nombre de termes (n).
- ♦ Eixida: Suma de tots els termes.
$$\sum_{i=1}^n a_i = S_n$$

■ Cerca de l'algorisme:

- ♦ Mètode 1: Calcular els termes i sumar-los progressivament.

- 
1. $S_n \leftarrow 0$
 2. $i \leftarrow 1$
 3. $X \leftarrow a_1$
 4. Si i és major que n , aleshores cal anar a (9).
 5. $S_n \leftarrow S_n + X$
 6. $X \leftarrow X + d$
 7. $i \leftarrow i + 1$
 8. Tornar a (4).
 9. Mostrar el resultat (S_n).
 10. FINAL.

- ♦ Mètode 2:

1. Aplicació de la fórmula:
2. Mostrar el resultat (S_n).
3. FINAL.

$$S_n = n \cdot a_1 + \frac{n \cdot (n-1)}{2} \cdot d$$



Criteris de selecció entre programes

- **Funcionament** del programa.
- **Claredat**:
 - ◆ Organització i sagnat.
 - ◆ Comentaris (per a l'usuari i per al corrector).
- **Estil** de programació:
 - ◆ Ús correcte de les variables.
 - ◆ Algorismes emprats.
 - ◆ Ús de funcions i procediments.



Exemple: Múltiples de cinc

■ Problema:

- ♦ Calcular la quantitat de múltiples de cinc que hi ha entre 0 i un nombre introduït pel teclat.

■ Mètode 1:

1. $DIV \leftarrow 5$
2. Llegir (MAX)
3. $I \leftarrow 0$
4. Si DIV és major de MAX, aleshores cal anar a (8).
5. $I \leftarrow I + 1$
6. $DIV \leftarrow DIV + 5$
7. Anar a (4).
8. Mostrar (I).
9. FINAL.

■ Mètode 2:

1. $DIV \leftarrow 5$
2. Llegir (MAX)
3. $I \leftarrow$ part entera de la divisió MAX / DIV
4. Mostrar (I).
5. FINAL.



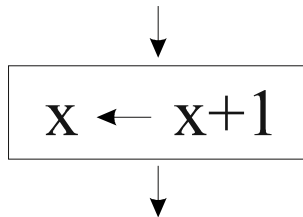
Programació de l'algorisme

- El **pseudocodi** és una manera d'escriure algorismes de manera poc estricta (una sintaxi relaxada) o estructures de dades poc detallades, però que tracta d'apropar les idees de l'algorisme a estructures i sintaxis paregudes a les dels llenguatges d'alt nivell amb els quals programarem l'algorisme.
- Els **diagrames de flux** són dibuixos que representen de manera gràfica la successió de tasques de l'algorisme. Les tasques es representen mitjançant rectangles, rombes i romboides. El flux es representa mitjançant fletxes que enllacen les diverses tasques.

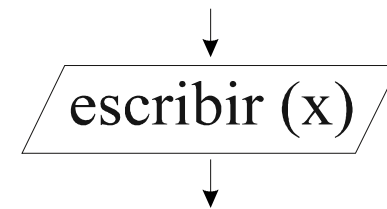


Disseny de diagrames de flux

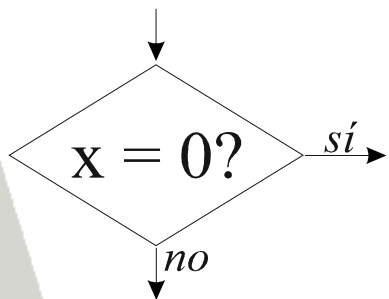
- Les **instruccions** es representen amb rectangles:



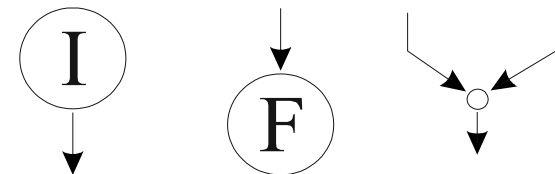
- Les **entrades** i **eixides**, amb romboïdes:



- Les **condicions**, amb rombes:



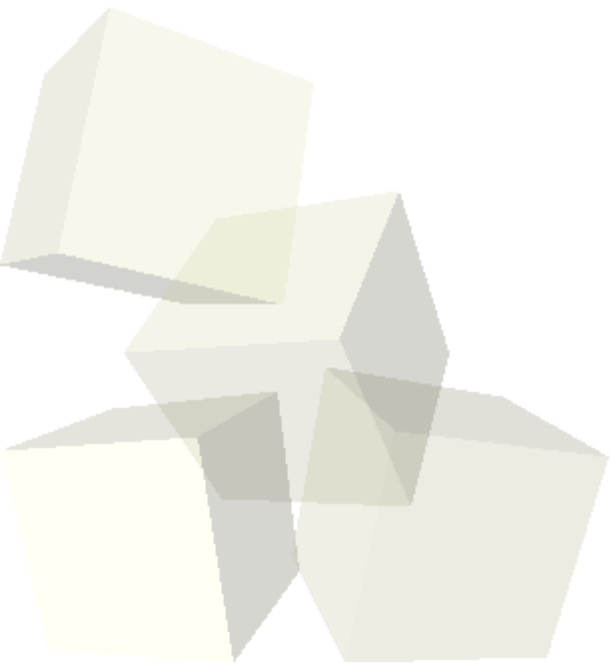
- L'**inici**, el **final** i els **punts de reunió** de flux, amb cercles:





■ Problema:

- ◆ Escriviu un algorisme que calcule els quadrats dels nombres enters positius fins que el quadrat siga major o igual que 100.
-
- Representeu l'algorisme anterior en forma de diagrama de flux.





Conceptes de variable i constant

- **Variable:** adreça de memòria el contingut de la qual pot canviar al llarg de l'execució d'un programa:
 - ♦ Variable local: Són pròpies de cada mòdul i només existeixen mentre s'executa el mòdul. Quan l'execució del mòdul acaba, aquestes variables desapareixen.
 - ♦ Variable global: Són variables que existeixen durant tota l'execució de l'algorisme. Així doncs, hom pot accedir-hi en qualsevol moment de l'execució del programa.
- **Constant:** Adreça de memòria el contingut de la qual roman constant al llarg de l'execució d'un programa.



- Anomenarem estructures de control a les accions que tenen com a objectiu el fet de marcar l'**ordre d'execució** de les instruccions i que es faran servir per escriure els algorismes de manera concisa i sense ambigüitats.
- Totes les estructures de control que estudiarem estaran formades per uns elements bàsics (lèxic) i una estructura (sintaxi).
- **Tipus:** Seqüencials, selectives i repetitives.



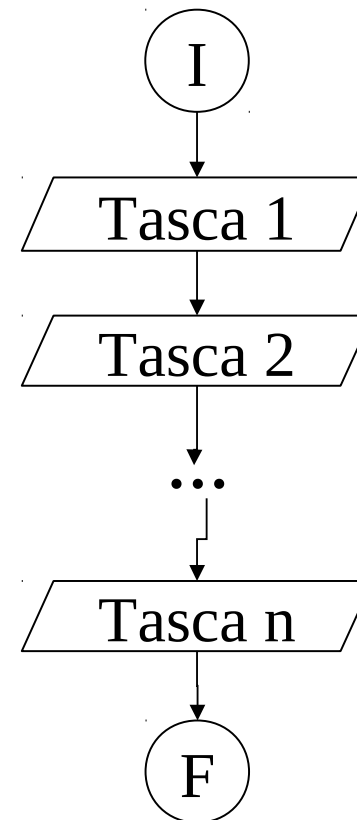
Estructures seqüencials

- En una estructura seqüencial, una instrucció segueix una altra en una seqüència lineal:

Pseudocodi:

Inici
Tasca 1
Tasca 2
...
Tasca n
Final

Diagrames de flux:





Exemple d'estructures seqüencials

- Producte escalar de dos vectors bidimensionals:
 - ♦ $(ax, ay) \cdot (bx, by) = ax * bx + ay * by$

Pseudocodi:

Inici

Llegir (**ax**)

Llegir (**ay**)

Llegir (**bx**)

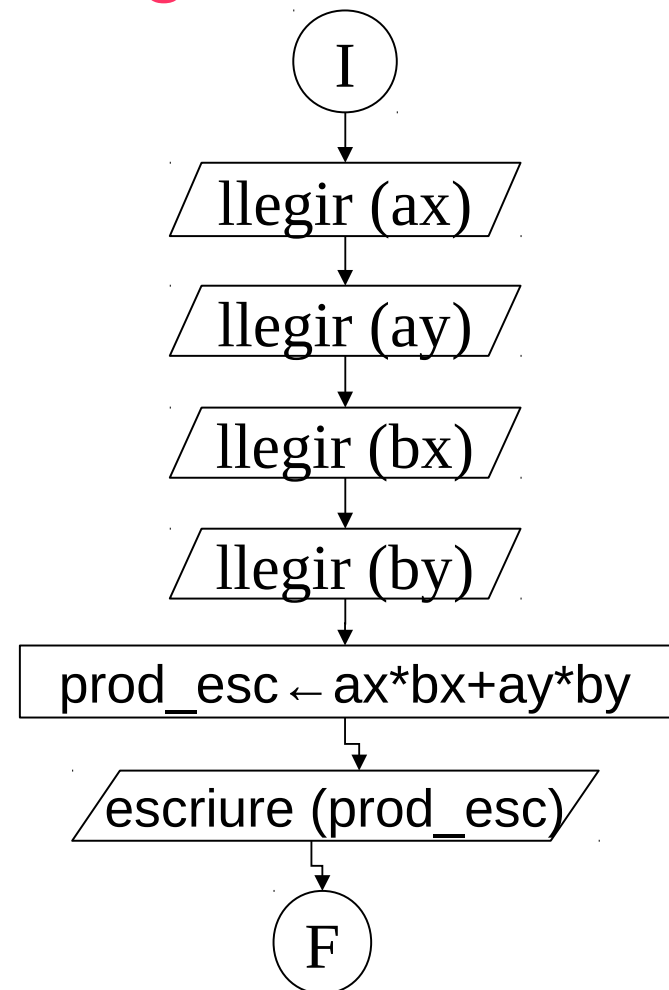
Llegir (**by**)

prod_esc $\leftarrow ax * bx + ay * by$

Escriure (**prod_esc**)

Final

Diagrames de flux :





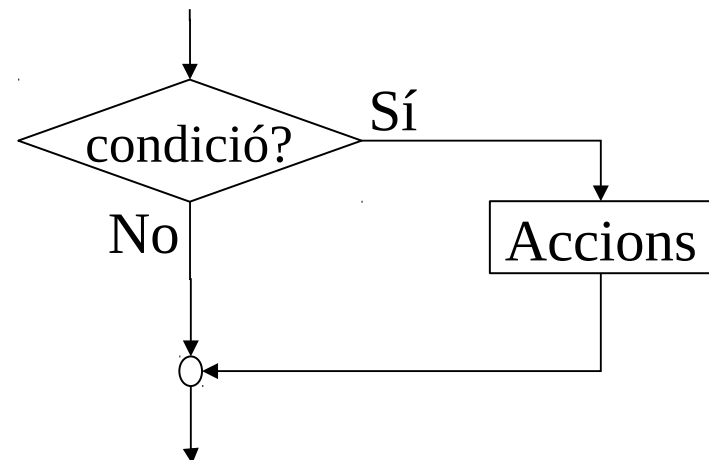
Estructures selectives

- Són les que prenen un determinat camí dins del flux del programa en funció d'una condició o del valor d'una variable.
- Alternatives senzilles:
 - ♦ Es realitza una única acció o un conjunt d'accions si es compleix una determinada condició.

Pseudocodi:

...
Si (condició) aleshores
 Accions
Final_si

Diagrames de flux:





Exemple de selectives senzilles

- Ordenar dos nombres llegits pel teclat.

Pseudocodi:

Inici

Llegir (a)

Llegir (b)

Si ($a > b$) aleshores

$\text{aux} \leftarrow a$

$a \leftarrow b$

$b \leftarrow \text{aux}$

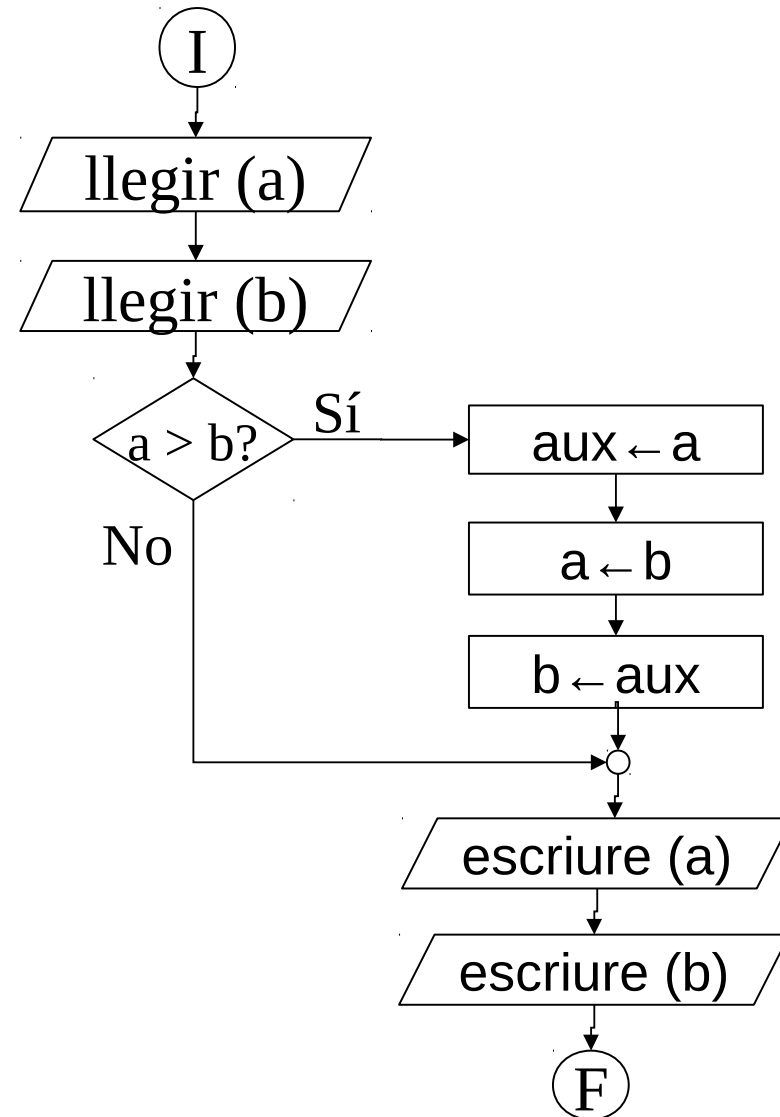
Final_si

Escriure (a)

Escriure (b)

Final

Diagrammes de flux:





Estructures selectives dobles

■ Alternatives dobles:

- ♦ Si una condició es compleix, es realitzen unes accions, si no, se'n realitzen unes altres.

Pseudocodi:

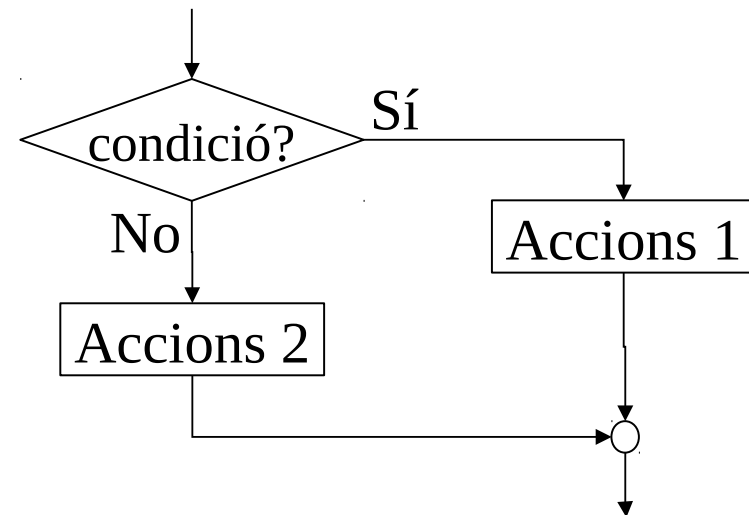
...

Si (condició) aleshores
Accions 1

Si no
Accions 2

Final_si

Diagrames de flux:





Exemple de selectives dobles

- Donat un nombre, cal decidir si és positiu o no.

Pseudocodi:

Variables

x: Enter

Inici

Llegir(x)

Si ($x < 0$) aleshores

Escriure("Negatiu")

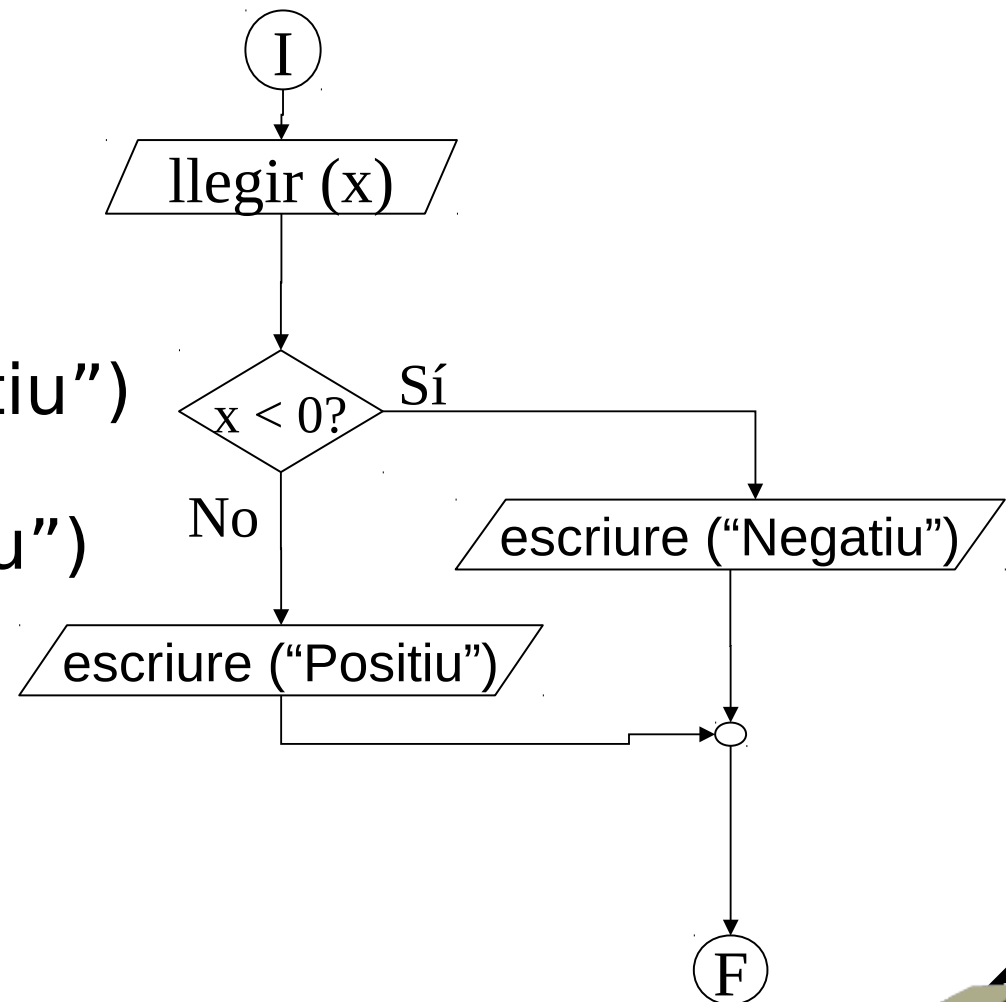
Si no

Escriure("Positiu")

Final_si

Final

Diagrames de flux:





Estructures selectives múltiples

■ Alternatives múltiples:

- ◆ Depenent del valor d'una variable, es realitzen unes accions o unes altres.

Pseudocodi:

...

Segons_siga (**variable**) fer

Cas **valor_1**: Accions 1

Cas **valor_2**: Accions 2

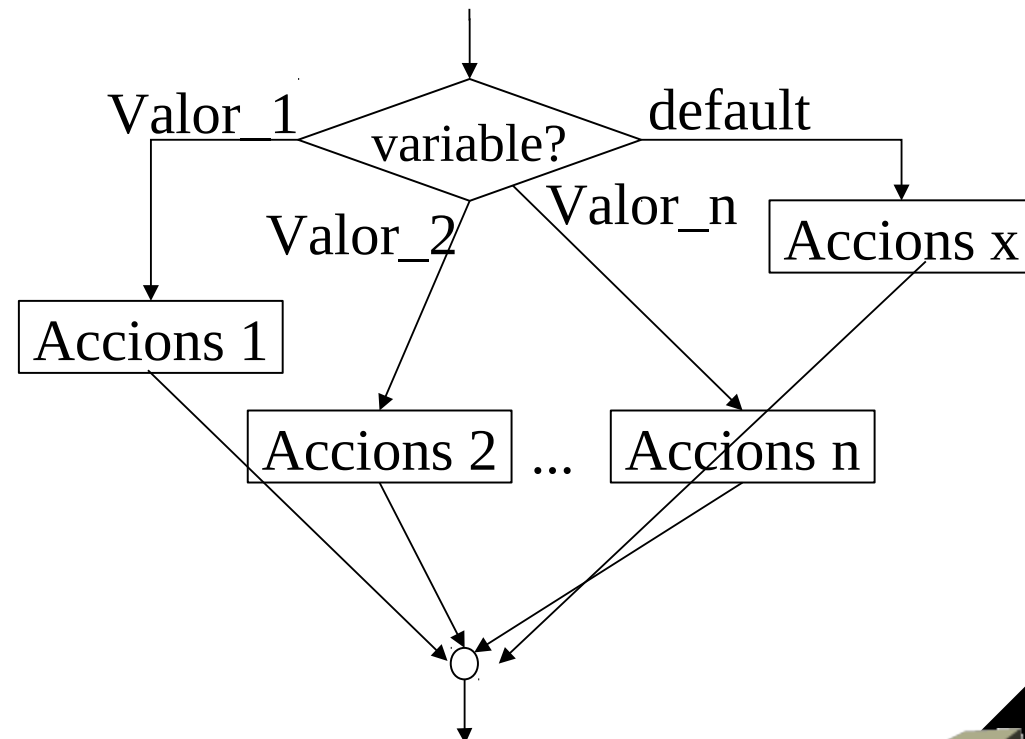
...

Cas **valor_n**: Accions n

Default: Accions x

Final_segons_siga

Diagrames de flux:





Exemple de selectives múltiples

- Feu un programa que demane dos nombres i una operació (suma, resta, multiplicació o divisió) i done el resultat d'operar els nombres amb l'operació seleccionada.

Pseudocodi:

Inici

Escriure ("Nombres?")

Llegir (x,y)

Escriure ("Operació?")

Escriure ("1-Suma, 2-Resta")

Escriure ("3-Multi, 4-Divisio")

Llegir (op)

Segons_siga (op) fer

Cas 1: $\text{res} \leftarrow a + b$

Escriure (res)

Cas 2: $\text{res} \leftarrow a - b$

Escriure (res)

Cas 3: $\text{res} \leftarrow a * b$

Escriure (res)

Cas 4: $\text{res} \leftarrow a / b$

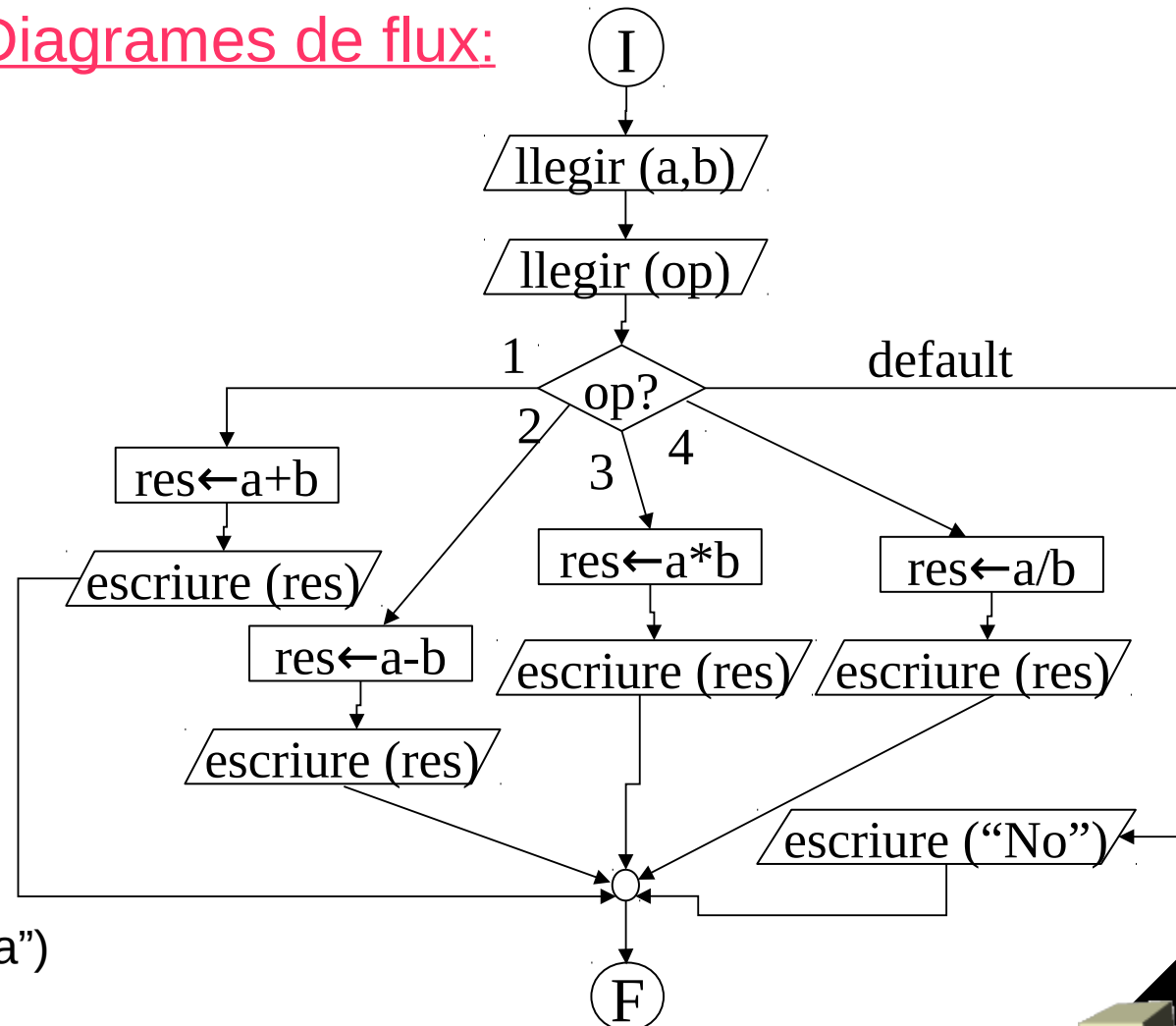
Escriure (res)

Default: Escriure ("Op. No vàlida")

Final_segons_siga

Final

Diagrames de flux:





Estructures repetitives (bucles)

- Un bucle és un conjunt d'instruccions del programa que s'executen repetidament, bé un nombre determinat de vegades, bé mentre siga certa una determinada condició (compte amb els bucles infinits).
- Tot bucle conté els elements següents (no necessàriament en aquest ordre):
 - ♦ Iniciació de les variables del bucle.
 - ♦ Decisió (continuem amb el bucle o acabem).
 - ♦ Cos del bucle.
- Hi ha tres tipus de bucle:
 - ♦ Des de...fins a.
 - ♦ Fer...mentre.
 - ♦ Mentre...fer.



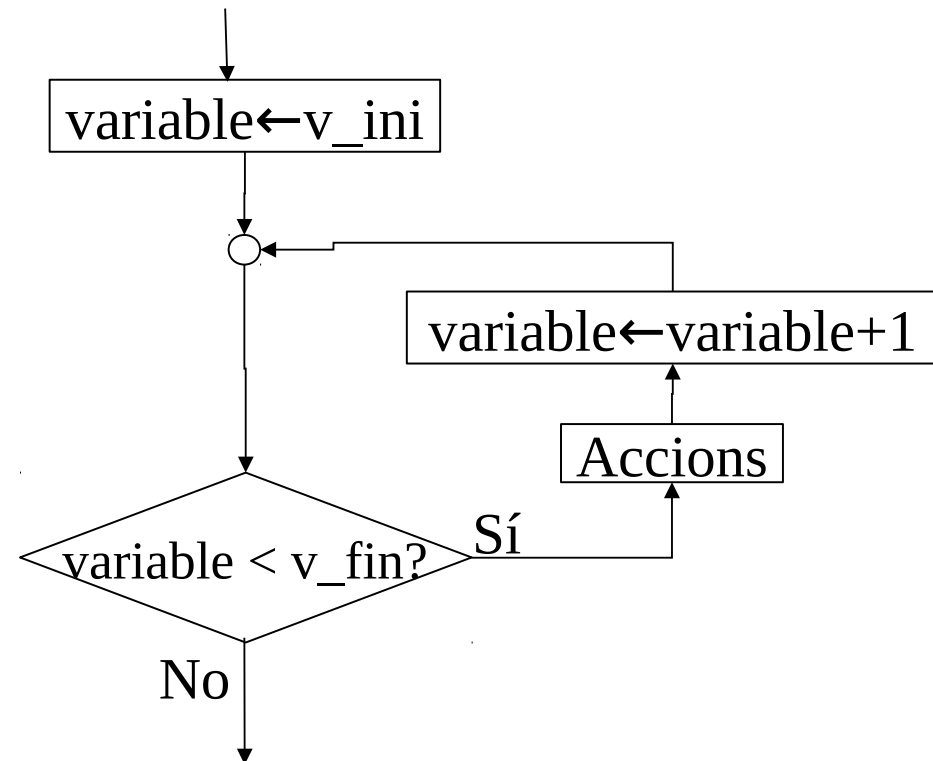
Bucle Des de...fins a

- L'emprem quan sabem el nombre de vegades que volem que s'execute una tasca.

Pseudocodi:

...
Des de **variable** ← v_ini fins a v_fin fer
Accions
Final_Des_de

Diagrames de flux:





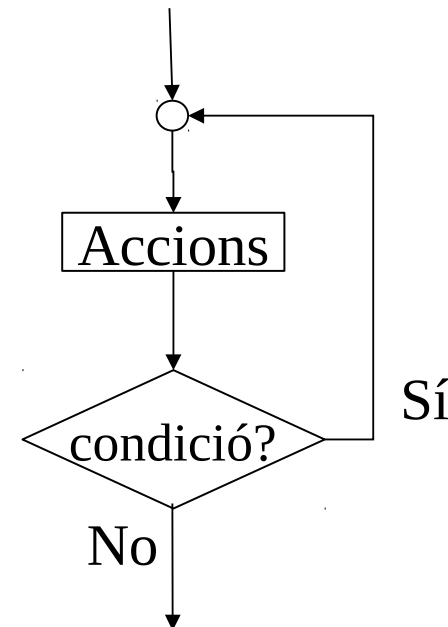
Bucle Fer...mentre

- L'emprem quan sabem la condició que fa que les tasques es repetisquen diverses vegades.
- Les accions es realitzaran almenys una vegada abans de comprovar la validesa de la condició.

Pseudocodi:

...
Fer
 Accions
 Mentre (condició)

Diagrames de flux:



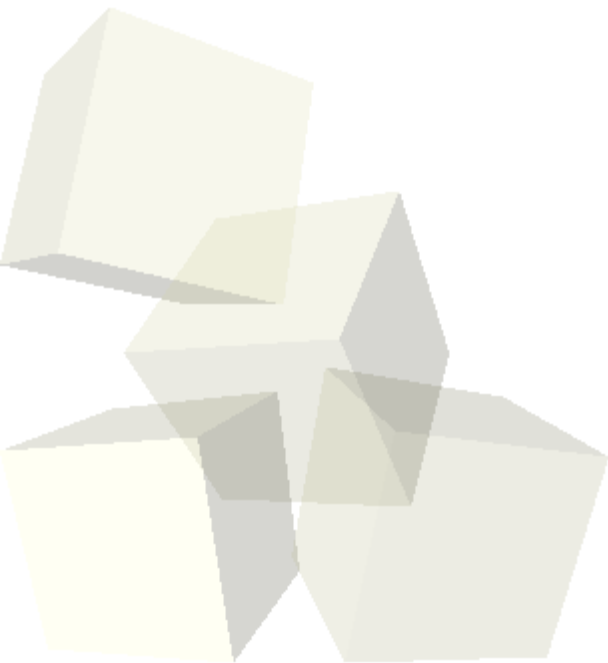
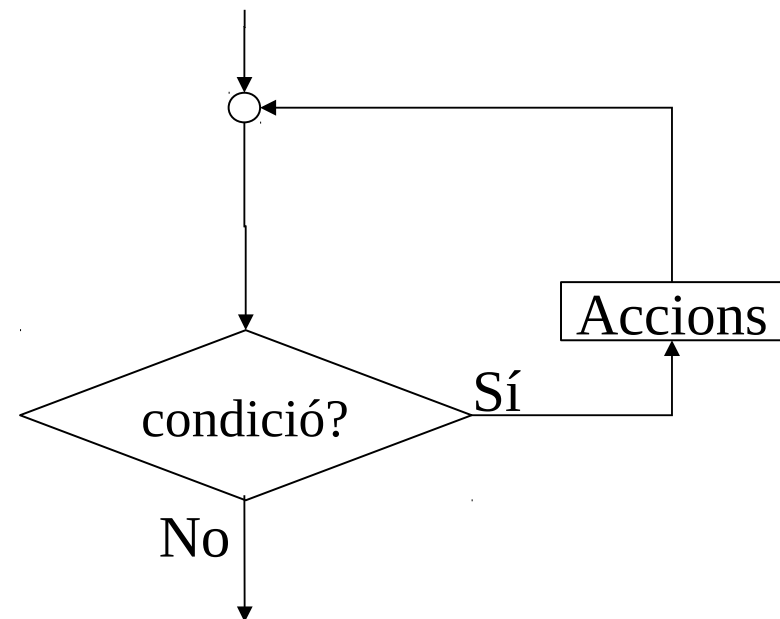


- Paregut al bucle Fer...mentre, però la comprovació es fa abans de realitzar la tasca.

Pseudocodi:

...
Mentre (condició)
 Accions
Final_Mentre

Diagrames de flux:





Bucles independents o imbricats

- Hi ha dues maneres de fer servir els bucles:
 - ♦ **Bucles independents:** Quan usem els bucles l'un darrere de l'altre. Açò és, quan en finalitzar un bucle comença el següent i les tasques d'ambdós són independents.
 - ♦ **Bucles imbricats:** Són bucles que estan dins d'uns altres bucles. Per tant, l'execució dels bucles interns depèn de l'execució dels bucles externs.





- Les característiques bàsiques que han de complir els **subprogrames o mòduls** són:
 - a) Realització d'una tasca específica.
 - b) Ús de paràmetres per a caracteritzar l'actuació dels mòduls.
 - c) Hi ha fonamentalment dos tipus diferents de subprogrames:
 - **Les funcions:** retornen el resultat de la seua actuació mitjançant un únic valor (o més, si rep paràmetres per referència).
 - **Els procediments:** no retornen cap resultat (poden tornar diversos valors mitjançant els paràmetres per referència).



Conceptes de variable i constant

- **Variable:** adreça de memòria el contingut de la qual pot canviar al llarg de l'execució d'un programa:
 - ♦ Variable local: Són pròpies de cada mòdul i només existeixen mentre s'executa el mòdul. Quan l'execució del mòdul acaba, aquestes variables desapareixen.
 - ♦ Variable global: Són variables que existeixen durant tota l'execució de l'algorisme. Així doncs, hom pot accedir-hi en qualsevol moment de l'execució del programa.
- **Constant:** Adreça de memòria el contingut de la qual roman constant al llarg de l'execució d'un programa.



Pas de paràmetres a mòduls

■ Per valor:

- ♦ Solament es té en compte el valor del paràmetre passat al mòdul.
- ♦ Açò és, durant l'execució del mòdul es reserva un nou espai per al paràmetre on es copia el valor passat.
- ♦ En acabar l'execució del mòdul, l'espai per al paràmetre desapareix.

■ Per referència:

- ♦ El que rep la funció es una referència del paràmetre que se li passa.
- ♦ Llavors, no es reserva cap espai nou durant l'execució del mòdul.
- ♦ Qualsevol modificació que es faça del paràmetre estarà disponible a partir del punt des del qual es féu la crida al mòdul.



- Diem que estem realitzant una **programació estructurada** quan:
 - ♦ Emprem el disseny descendent o modular a l'hora de dissenyar els algorismes.
 - ♦ Descomponem, d'acord amb el disseny descendent, el programa en mòduls independents (prohibit l'ús de variables globals).
 - ♦ Per a escriure els algorismes (i els programes) només fem servir els tres tipus d'estructures de control vistes en aquesta presentació.



Recurrències o recursivitat

- És la propietat de cridar a una funció des d'ella mateixa o des d'una altra que ha estat cridada per la primera.
- Una característica molt important de la recursivitat és que, com succeïa en les estructures repetitives, cal que hi haja un **punt o condició d'acabament** perquè en algun moment la recursivitat s'ature.

