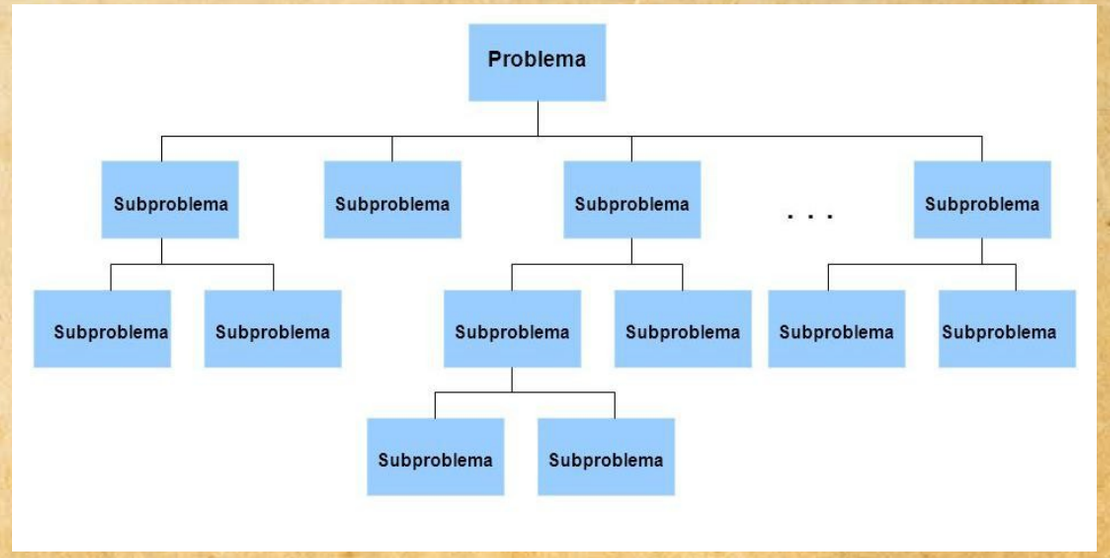
Programació

UT4.1 Introducció a la programació modular

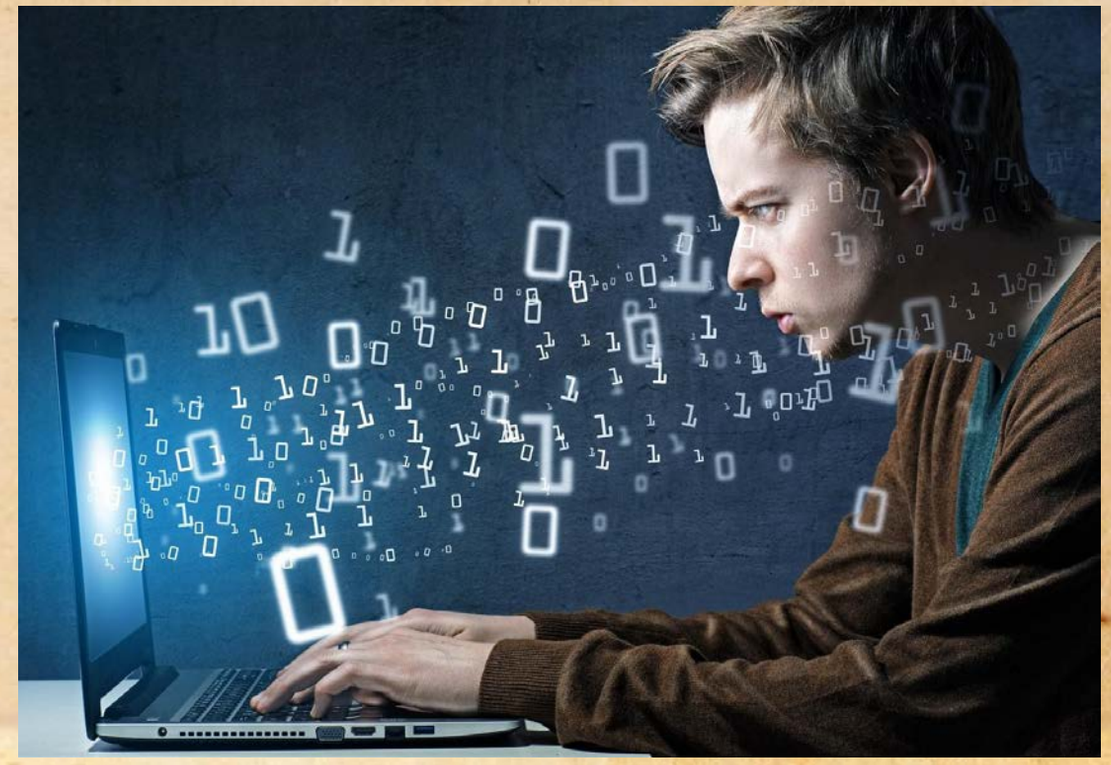
Introducció

- Fins ara, som capaços de **dissenyar algorismes complets** ad-hoc degut a la seua curta complexitat.
- A major complexitat, major mida del problema i per tant més complicat el seu disseny.
- Es recorre al disseny descendent



Descomposició de problemes

- En este punt, cal recordar que es un error molt comú afrontar un problema directament programant-lo.

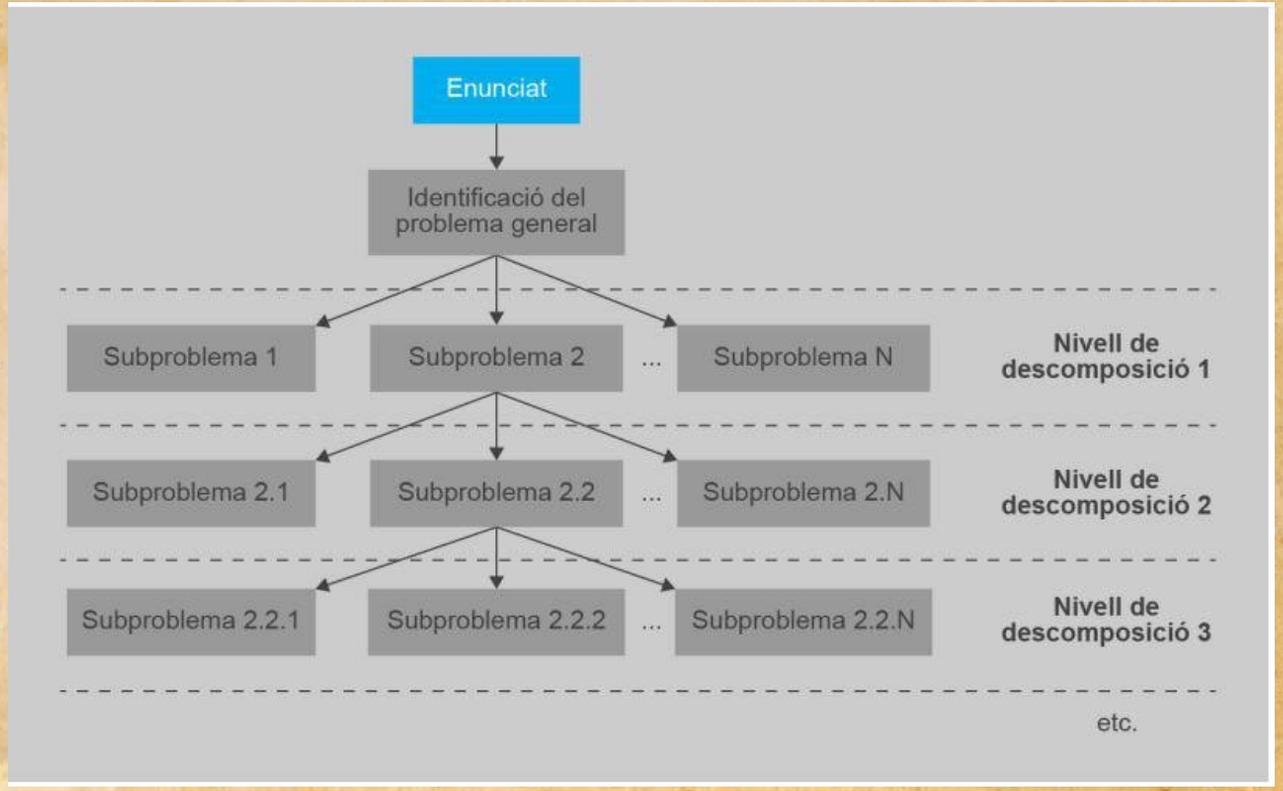


Descomposició de problemes

- L'estratègia a seguir és **dividir el problema en subproblemes**
- És **més fàcil de manipular i d'organitzar**
- La resolució dels subproblemes donarà lloc a **la resolució d'un problema de nivell superior**

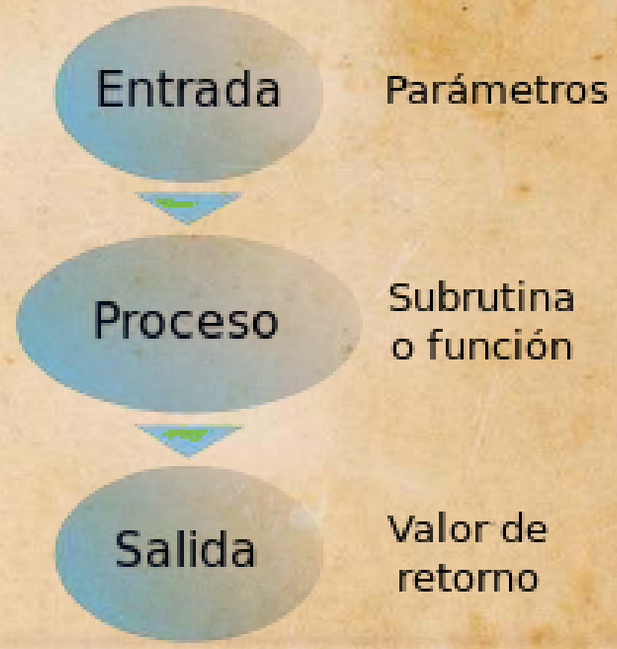
Disseny descendent

- També anomenat **"top-down"**, és la tècnica que es basa en partir d'un problema general i dividir-lo en problemes més simples, denominats subproblemes.



Disseny descendent

- La **descomposició en subproblemes** no és arbitrària, sinó que es planteja com un objectiu parcial, amb entitat pròpia, per a resoldre part del problema de nivell superior.



Objectius del disseny descendent

- Establir una **relació senzilla entre problemes** plantejats i el conjunt de tasques a fer per resoldre'ls
- Establir més fàcilment els **passos per a resoldre un problema**
- Fer **més fàcil de comprendre** estos subproblemes
- **Limitar els efectes de la interdependència** entre subproblemes

Exemples d'aplicació d'un disseny descendent. Recepta.

Fideus yakisoba vegetals

1) Recopilar els ingredients.
2) Cuinar els ingredients.
3) Preparació final

Exemples d'aplicació d'un disseny descendent. Recepta.

1) **Recopilar els ingredients**
    1) Comprar al super
    2) Col·locar-los a la taulell

Exemples de aplicació de un disseny descendent. Recepta

2) Cuinar els ingredients

1) Cuinar tallarins
    1) Preparar l'aigua
        1) Escalfar l'aigua
        2) Posar sal
    2) Bullir els tallarins
    3) Escórrer els tallarins
    4) Deixeu-los preparats
2) Cuinar carlota
    1) Partir la carlota
    2) Fregir la carlota
        1) Preparar oli per a fregir
        2) Pelar les carlotes
        3) Netejar oli de la paella
    3) Deixar les carlotes preparades

3) Cuinar cebes
    1) Partir cebes
    2) Fregir cebes
        1) Preparar oli per a fregir
        2) Pelar cebes
        3) Netejar oli de la paella
    3) Deixar les cebes peparades

Exemples de aplicació de un disseny descendent. Recepta

3) Preparació final

1) Mesclar ingredients preparats amb salsa yakitori
2) Saltar ingredients
    1) Preparar paella per saltar
    2) Cuinar removent els ingredients
3) Deixar el plat llest per a emplatar

Elaboració de algorismes

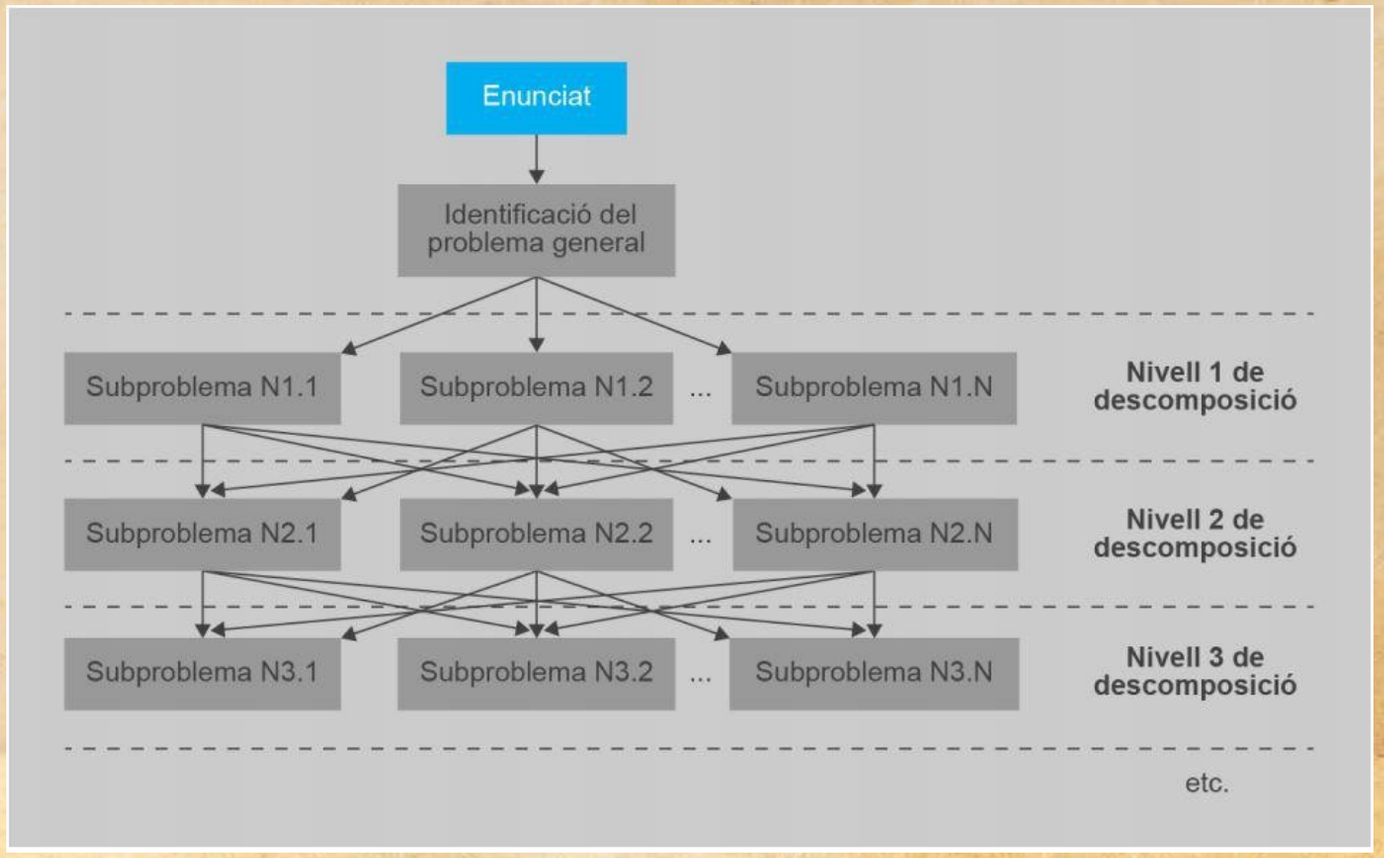
- Ara ja serà labor del programador, afrontar l'algorisme necessari que resol cada subproblema.
- Per exemple el problema "Preparar oli per a fregir":

1) Agafar botella d'oli
2) Abocar oli fins a un terç de la paella
3) Posar foc al màxim
4) Mentre l'oli no tire fum, esperar

- Noteu que és **un problema totalment independent del resta**
- Cada procés **requerirà executar-se en ordre** ja que cadascún és dependent de que estiga resolt per a que puga començar el següent.

Reutilització de problemes resolts

- La descomposició a problemes permet agrupar problemes que siguen idèntics, o al menys que la seua estructura ho siga.



Disseny descendent aplicat a la informàtica

- Exemple de un disseny descendent per un programa de complexitat baixa
- El problema serà un de conegut:
  - *"A partir d'una llista de 10 nombres enters, mostra'ls per pantalla ja ordenats"*

Abans de la descomposició

- Quin tipus de dades necessitaré i com els emmagatzemaré?
  - ***Alguna cosa que llija enters***
  - ***Array per a emmagatzemar-los***

Descomposició. Definició d'etapes

- **Cada etapa deu correspondre a una tasca concreta** amb un objectiu a resoldre clarament diferenciat.
- És una bona senyal d'una correcta descomposició, si **a cada tasca li podem posar un nom fàcilment**

Etapes definides (1r nivell)

- Llegir una llista d'enters
- Ordenar una llista d'enters (mètode de la bombolla)
- Mostrar una llista d'enters per pantalla

Etapes definides (2n nivell)

- Després de definir un primer nivell, **serà necessari reanalitzar cada subproblema** per vore si és massa complex o no.
- Al problema concret que hem dissenyat, podem vore que **no farà falta descompondre en més nivells**

Funcions

- Per a encapsular cada una de les etapes definides es faran ús del que a qualsevol llenguatge de programació es denomina **funció**
- **Funció:** conjunt d'instruccions amb un objectiu comú que es declaren de manera explícitament diferenciada dins del codi font

Funcions vs Procediments

Quan parlem de subprogrames, cal diferenciar entre:

FUNCIÓ

És una secció d'un programa que calcula un valor de manera independent a la resta del programa.

En escència, una funció és un miniprograma: té una entrada, un procés i una eixida.

Una funció té tres components importants:

- Els **paràmetres** , que són els valors que rep la funció com a entrada.
- El **codi de la funció** , que són les operacions que realitza la funció.
- El **resultat o valor de retorn** , que és el valor final que retorna la funció.

PROCEDIMENT

Un procediment és una secció d'un programa (igual que una funció) que fa diverses sentències de manera independent a la resta del programa. La diferència amb una funció és que un procediment no retorna cap valor com a resultat.

Els procediments són útils per agrupar seqüències de sentències que s'han de fer juntes. Usar procediments sol fer que els programes siguen més fàcils de llegir

Exemple pràctic: Suma de números

- Suposa que necessites trobar la suma dels enters del 1 al 10, del 20 al 37 i del 35 al 49, respectivament. Escriuries un codi com el següent:

```
int sum = 0;
for (int i = 1; i <= 10; i++)
  sum += i;
System.out.println("Sum from 1 to 10 is " + sum);

sum = 0;
for (int i = 20; i <= 37; i++)
  sum += i;
System.out.println("Sum from 20 to 37 is " + sum);

sum = 0;
for (int i = 35; i <= 49; i++)
  sum += i;
System.out.println("Sum from 35 to 49 is " + sum);
```

Exemple pràctic: Suma de números

- Observa que hi ha codi que es repeteix. La solució crear una **funció** que continga el codi repetit.

```
public static int sum(int i1, int i2) {
  int result = 0;
  for (int i = i1; i <= i2; i++)
    result += i;

  return result;
}
```

```
public static void main(String[] args) {
  System.out.println("Sum from 1 to 10 is " + sum(1, 10));
  System.out.println("Sum from 20 to 37 is " + sum(20, 37));
  System.out.println("Sum from 35 to 49 is " + sum(35, 49));
}
```