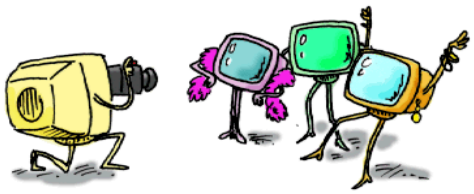
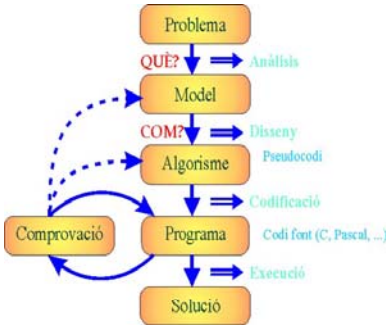
Tema 2: Disseny de l'algorisme

- **2.1 Definició i descripció d'algorismes**
  - **Algorisme**
  - **Disseny de l'algorisme**
- **2.2 Eines de programació**
  - **Diagrama de flux**
  - **Pseudocodi**
  - **Codificació del programa**
- **2.3 Exemples**



2.1 Definició i descripció d'algorismes (I)

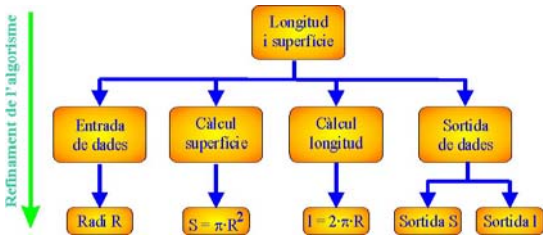
- Algorisme →
  - Mètode de resolució d'un problema que indica la seqüència d'operacions a realitzar per tal d'aconseguir el resultat esperat
  - Paraula d'origen àrab: Mohammed Ben Musa Al Khowârizmi
  - Al s. IX va escriure les regles de com fer les quatre operacions aritmètiques en base 10.
  - Conjuntament amb Euclides (s.IX a.c.), que va escriure un mètode per torbar el m.c.d. de dos nombres enters, es considera el pare dels algorismes.
  - En la resolució de problemes per ordinador emprarem els algorismes per trobar la seqüència de passos que porten al resultat esperat. De forma global, l'esquema a seguir és el de la figura adjunta.



  - Els algorismes han de ser
    - Precisos → No han de ser ambigus.
    - Finits → La seva execució ha d'acabar en algun moment.
    - Deterministes → Tota execució de l'algorisme amb unes determinades dades ha de donar, sempre, el mateix resultat.
    - Coherents → Ha de resoldre el problema plantejat

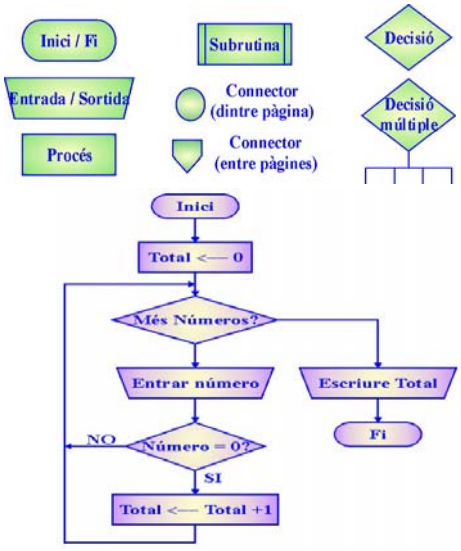
2.1 Definició i descripció d'algorismes (II)

- Disseny de l'algorisme
  - Per a solucionar un problema amb la computadora cal indicar-li, fidelment, els passos que ha de seguir.
  - Per tant, la computadora cal conèixer
    - Entrades → Dades amb les que ha d'operar
    - El procés → Seqüència d'accions a realitzar
    - Sortides → Els resultats que esperem
  - Per a resoldre problemes normalment haurem d'aplicar metodologies concretes d'aproximació al problema, com ara dividir les accions a realitzar en mòduls o aplicar tècniques del *divideix i conqueriràs*.
  - Aquesta metodologia de disseny agafa el nom de metodologia *top-down*.
- Exemple: troba l'algorisme que, donat el radi, calcula la longitud d'un cercle i la seva superfície.



2.2 Eines de programació (I)

- Diagrames de flux (*flowchart*)
  - És un mètode de representació gràfica de l'algorisme.
  - Consta de símbols (capses) per indicar funcions i fletxes (línies de flux) que indiquen el seqüenciament de l'algorisme.
  - La figura adjunta en mostra els símbols principals
- Exemple
  - Trobar el diagrama de flux que, davant una entrada de nombres (com ara 5,3,0,0,2,7,0,4, ...) imprimeixi el número de zeros que s'han entrat.



2.2 Eines de programació (II)

- Pseudocodi
  - És un llenguatge d'especificació d'algorismes que simplifica el pas al llenguatge de programació final.
  - Permet pensar en l'algorisme des del punt de vista de llenguatge natural, més que no pas centrar-se en les instruccions pròpies del llenguatge de programació

  - Exemple
    - Escriure el pseudocodi de l'algorisme que compta i imprimeix el nombre de zeros d'una seqüència de nombres (és immediat a partir del diagrama de flux anterior)

```
1. Inici
        2. Total ← 0
        3. Mentre números fer
                3.1 Llegir número
                3.2 Si número = 0 aleshores
                        3.2.1 Fer Total ← Total + 1
                FiSi
          Fi_mentre
        4. Escriure (Total)
5. Fi
```


2.2 Eines de programació (III)

- Codificació del programa
  - És l'escriptura de l'algorisme desenvolupat en l'apartat anterior en un llenguatge de programació.
  - És important recordar que l'escriptura de l'algorisme en pseudocodi o diagrama de flux és independent del llenguatge de programació a emprar.
  - La conversió de l'algorisme al llenguatge de programació amb què es treballi es fa substituint les paraules reservades a les homònimes del llenguatge de programació

  - Exemple
    - Trobar el diagrama de flux que, davant una entrada de nombres enters positius (com ara 5,3,0,0,2,7,0,4, ...) imprimeixi el número de zeros que s'han entrat. Una entrada d'un número negatiu farà sortir del programa.

```
#include <stdio.h>
void main()
{
    int numero, total=0;
    do
    {
        scanf("%i",&numero);
        if (numero == 0) total++;
    }while (numero>=0);
    printf("Numero de 0s = %i", total);
}
```

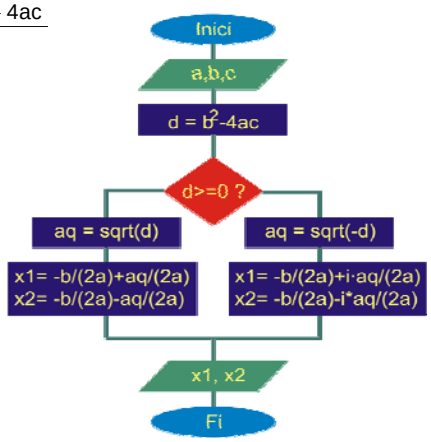
2.3 Exemples (I)

- Donada l'equació de segon grau $ax^2+bx+c=0$ trobar les seves solucions
  - Recordar que la solució d'una equació de segon grau ve donada per la fòrmula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

  - Per tant: Entrades: a,b,c. Sortides: x.
  - Algorisme:

```
Real a, b, c, d, aq
Inici
    Llegir (a,b,c)
    d ← a²-4ac
    Si d>=0 Aleshores Fer
            aq←sqrt(d)
            x1 ← -b/(2a)+aq/(2a)
            x2 ← -b/(2a)-aq/(2a)
    SiNo Fer
            aq←sqrt(-d)
            x1 ← -b/(2a)+i·aq/(2a)
            x2 ← -b/(2a)-i·aq/(2a)
    FiSi
    Escriure (x1,x2)
Fi
```

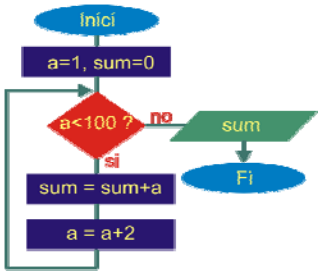


Nota1: Veurem com tractem el fet de tenir arrels complexes. De moment ho deixem així.
Nota2: Variables internes emprades: d, aq.

2.3 Exemples (II)

- Realitzar la suma dels nombres senars des d'1 fins a 100.
  - A diferència del cas anterior, ara hem de detectar la condició final
  - Donat que coneixem les condicions inicial i final podem emprar la instrucció *Per*
  - S'utilitzen dues variables:
    - a (entera) porta el compteig del número amb el que estem treballant
    - sum (entera) porta la suma fins al moment
  - Per tant: Entrada: a. Sortida: sum.
  - Algorisme:

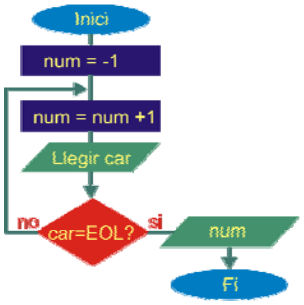
```
Enter a, sum
Inici
    sum ← 0
    Per (a ← 1, a<100) Fer
            sum ← sum + a
            a ← a+2
    FiPer
    Escriure (sum)
Fi
```



2.3 Exemples (III)

- Comptar totes les lletres d'una frase que entrem per teclat i que s'acaba amb un salt de ratlla.
  - En aquest cas només tenim noció que l'algorisme s'ha d'acabar en el moment que entrem un *return* (salt de ratlla, EOL, \n en C).
    - Per tant: Entrada: car (el caràcter que entrem per teclat – l'entrada estàndar).
      Sortida: num (el número de caràcters entrat)
  - Algorisme:

```
Caracter car
Enter num
Inici
        num ← -1
        Fer
                num ← num+1
                Llegir (car)
        Mentre (car <> EOL)
        Escriure (num)
Fi
```



Com a indicació, en C aquest algorisme es podria escriure de la següent forma:

```
void main()
{
        int i=0;
        while (getchar() != '\n') i++;
        printf("numero caracters = %i\n", i);
}
```