UD1.- Fonaments de la Programació

“El principi de la saviesa per a un enginyer de software és reconèixer la diferència entre fer que un programa funcione i aconseguir que ho faça correctament.”

Roger Pressman

1.- Introducció.

Els coneixements de programació de sistemes informàtics són uns coneixements fonamentals per a qualsevol professional del món de la informàtica. Tant si la programació és la seva especialitat com si no, els coneixements de programació són imprescindibles per a gestionar qualsevol sistema informàtic, i per a configurar-lo per tal que aquest sistema tingui el funcionament que vol. En aquest mòdul s'imparteixen coneixements que habiliten els estudiants per a interpretar i crear la programació d'un sistema informàtic.

Aquest mòdul de programació bàsica és eminentment pràctic, i si bé és útil fer una primera lectura per a entrar en contacte amb els conceptes presentats, és imprescindible acompanyar aquesta lectura de la realització dels exemples i exercicis presentats en la documentació i, sobretot, anar fent els exercicis i les activitats que es proposen a la part web. Si no es fa així, serà molt difícil assolir els coneixements que es presenten i avançar en la comprensió de nous conceptes.

Com en qualsevol procés d'aprenentatge, cal començar pel principi.

És important tenir clars un conjunt de conceptes bàsics que ajudin a comprendre els conceptes més avançats que vindran posteriorment. En aquest cas, es tracta d'establir què és un programa, com funciona i quin és el model general per crear-ne un. Només un cop ho tingueu clar us podeu plantejar seure davant de l'ordinador i començar a programar.

La raó principal per la qual una persona utilitza un **ordinador** és per a **resoldre problemes** (en el sentit més general de la paraula), o en altres paraules, processar una informació per a obtindre un resultat a partir d'unes dades d'entrada.

Els ordinadors resolen els problemes **mitjançant la utilització de programes escrits** pels programadors. Per tant, la tasca dels programadors és trobar una **solució** a eixos problemes .

Com ha de ser la nostra solució?

- **Correcta i eficaç:** si resol el problema adequadament.
- **Eficient**: si ho fa en un temps mínim i amb un ús òptim dels recursos de sistema.

Però quan els problemes són **complexos**, és necessari **descompondre'ls** en **subproblemes més simples** i, al seu torn, en uns altres **més xicotets**. Aquestes estratègies reben el nom de:

- **Disseny descendent.** Metodologia de disseny de programes, consistent en la **descomposició del problema en problemes més senzills** de resoldre.

- **Disseny modular.** Metodologia de disseny de programes, que consisteix a **dividir la solució a un problema en mòduls més xicotets o subprogrames**. Les solucions dels mòduls s'uniran per a obtindre la solució general del problema). Aquest sistema es basa en el lema divideix i venceràs.

Els programes d'ordinador no són més que mètodes per a resoldre problemes. Per això, **per a escriure un programa, el primer és que el programador sàpiga resoldre el problema que estem tractant.**

El programador ha **d'identificar** quines són les **dades d'entrada** i a partir d'ells obtindre les dades d'eixida, és a dir, la solució, a la qual s'arribarà per mitjà del processament de la informació que es realitzarà mitjançant la utilització d'un mètode per a resoldre el problema que denominarem **algorisme**.

2.- Algorisme.

Un **ALGORISME** un conjunt **ordenat** i **finit** d'operacions que permeten resoldre un problema.

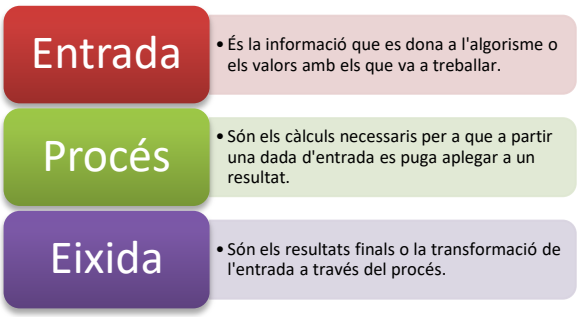
Els algorismes són independents dels llenguatges de programació i dels dispositius on s'executen, un algorisme pot ser expressat en distints llenguatges de programació.

Un algorisme ha de ser **PRECÍS**, **FINIT** i **DEFINIT**.

- **PRECÍS**: ha de tindre **instruccions clares** i indicar **l'ordre** de realització de cada pas
- **FINIT**: té un nombre finit de passos, **ha d'acabar** en un temps finit. Si no acabara mai, no es resoldria el problema.
- **DEFINIT**: Totes les operacions han d'estar definides de manera precisa i sense ambigüitat, de manera que si se segueix l'algorisme dues vegades, s'ha d'obtindre el mateix resultat.

Per a definir un algorisme hem de tindre en compte que consta de 3 parts: **l'ENTRADA, el PROCÉS i l'EIXIDA.**

Un algorisme pot tindre varies dades d'entrada, i com a mínim una dada d'eixida.



Un clar exemple d'algorisme és una recepta de cuina, on tenim uns passos que cal seguir en un ordre i han d'estar ben definits, té un temps finit i té unes dades d'entrada (ingredients) i una eixida (el plat).

Per exemple, l'algorisme per a fregir un ou podria ser el següent:

Dades d'entrada: Ou, oli, paella, foc.

Dades d'eixida: ou caigut.

Procediment:

1. Posar l'oli en la paella.
2. Posar la paella al foc.
3. Quan l'oli estiga calent, trencar l'ou i introduir-lo.
4. Cobrir l'ou d'oli.
5. Quan l'ou estiga fet, retirar-lo.

La codificació d'un **ALGORISME** mitjançant un llenguatge de programació s'anomena **PROGRAMA**

Per tant, un programa informàtic no es més que un conjunt d'instruccions per a un ordinador. Un programa no fa res, a menys que les seues instruccions siguen executades per el processador.
Els llenguatges de programació són un mitjà per expressar l'algorisme, i l'ordinador, un processador per a executar-los. El disseny d'algorismes ha de ser una tasca que necessitarà de creativitat i coneixements de tècniques de programació.

Quan parlen de programes, ens referim tant als programes **executables** com al seu **codi font**.

El **codi font** és escrit en un llenguatge de programació i es transforma en un **executable** quan es **compilat**.

Un programa es pot referir tant a un programa executable com al seu codi font, el qual és transformat en un executable quan és compilat.
Generalment el codi font dels programes és escrit per professionals coneguts com a programadors. El codi font és escrit en un llenguatge de programació que segueix un dels següents dos paradigmes: imperatiu o declaratiu. El codi font pot ser convertit en una imatge executable per un compilador. Quan es demana que el programa sigui executat, el processador executa el programa instrucció per instrucció, fins que el programa s'acaba.

3.- Objectes d'un Programa.

Anomenem **OBJECTE** d'un programa a tot allò que es pot manipular per diferents instruccions formen part del programa. En ells s'emmagatzemaran tant les dades d'entrada com les d'eixida (resultats).

Tots els objectes tenen els següents **atributs**:

- **Nom**: l'identificador de l'objecte.
- **Tipus**: conjunt de valors que pot prendre.
- **Valor**: element del *Tipus* que se li assigna.

Per exemple :

int laMeuaVariable = 2

A l'exemple trobem una variable de de nom ***laMeuaVariable*** de tipus **enter** on se li assigna el valor 2 (de tipus enter).

Els principals objectes que trobem al llarg d'un programa són:

- Constants.
- Variables.
- Expressions.
- Operadors. (Relacionals, Aritmètics, Lògics o Booleans. Paréntesis. Alfanumèrics)

3.1.- Constants.

El seu valor no varia al llarg de l'execució del programa. Un costant és posar-li un nom a un valor concret, de manera que s'utilitza el seu nom cada volta que volem referenciar eixe valor.

Per exemple:

pi = 3,141592 **e** = 2,718281 **IVA** = 0,21

Si ara volem calcular la longitud d'una circumferència de **radi =4** calcularíem :

2*pi*radi

Les constants també són utilitzades per **simplificar les modificacions i manteniments dels programes**. Imagina que estem escrivint un programa comptable o financer, on utilitzem moltes voltes operacions amb el IVA, si en un futur l'import canvia, sols tindríem que modificar el valor de la constant IVA una volta, en canvi, si utilitzem cada volta 0,21 hem de modificar eixe valor en cada operació.

3.2.- Variables

Són objectes el valor dels quals pot ser modificat al llarg de l'execució d'un programa.
Per exemple: una variable per calcular l'àrea d'una circumferència determinada, una variable per a calcular una factura, etc.

3.3.- Expressions

Les expressions segons el resultat que produïsquen es classifiquen en:

- **Numèriques:** Són les que produeixen **resultats** de tipus **numèric**. Es construeixen mitjançant els operadors aritmètics.
  Per exemple:

```
pi *sqr(x)
(2*x)/3
```

- **Alfanumèriques:** Són les que produeixen **resultats** de tipus **alfanumèric**. Es construeixen mitjançant operadors alfanumèrics
  Per exemple:

```
Jaume " + "Aragó"
```

- **Booleanes o lògiques**: Són les que produeixen **resultats** de tipus **Vertader o Fals**. Es construeixen mitjançant els operadors relacionals i lògics.
  Per exemple:

```
A < 0
(a > 1) and (b < 5)
```

3.4.- Operadors.

Són els **símbols** que fan d'enllaç entre els arguments d'una **expressió**:

1.- Relacionals.
2.- Aritmètics.
3.- Lògics o Booleans.
4.- Parèntesi ().
5.- Operador Alfanumèric (+).

1.- Operadors Relacionals

S'utilitzen per formar expressions que al ser avaluades **retornen un valor booleà** (***Vertader o Fals***)

Operador	Definición
<	Menor que
>	Mayor que
=	Igual que
>=	Mayor o igual que
<=	Menor o igual que
<>	Distinto que

Exemples:

Expresión	Resultado
A' < 'B'	Verdadero, ya que en código ASCII la A está antes que la B
1 < 6	Verdadero
10 < 2	Falso

2.- Operadors Aritmètics

S'utilitzen per realitzar operacions aritmètiques:

Operador	Definición
+	Suma
-	Resta
*	Multiplicación
^	Potencia
/	División
%	Resto de la división

Exemples:

Expresión	Resultado
3 + 5 - 2	6
24 % 3	0
8 * 3 - 7 / 2	26

3.- Lògics o Booleans.

Operador	Definición
No	Negación
Y	Conjunción
O	Disyunción

El comportament d'un operador lògic es defineix mitjançant la seva corresponent **taula de veritat**, en ella es mostra el resultat que produeix l'aplicació d'un determinat operador a un o dos valors lògics. Les operacions lògiques més usuals són:

- **NO lògic (NOT) o negació:**

A	NOT A
V	F
F	V

L'operador **NOT** inverteix el valor: Si és veritat (V) retorna fals (F), i viceversa.

- **O lògica (OR) o disjunció:**

A	B	A OR B
V	V	V
V	F	V
F	V	V
F	F	F

L'operador OR retorna veritable (V) si algun dels dos valors és veritat
En qualsevol altre cas, retorna Fals (F).

- **Y lògica (AND) o conjunció:**

A	B	A AND B
V	V	V
V	F	F
F	V	F
F	F	F

L'operador AND retorna Veritat (V) només si tots dos valors són Veritat.
En qualsevol altre cas retorna Fals (F).

Exemples (Suposant que a<b):

Expresión	Resultado
9 = (3*3)	Verdadero
3 <> 2	Verdadero
9 = (3*3) Y 3 <> 2	Verdadero
3 > 2 Y b < a	Verdadero Y Falso = Falso
3 > 2 O b < a	Verdadero O Falso = Verdadero
no(a < b)	No Verdadero = Falso
5 > 1 Y NO(b < a)	Verdadero Y no Falso = Verdadero

4.- Parèntesi ().

El parèntesi serveix per anidar expressions.

Exemple:

Operació (3*2) + (6/2)	Resultat 9

5.- Operador Alfanumèric (+).

Uneix dades de tipus **alfanumèric**. També s'anomena **concatenació**.
Exemples:

Expresión	Resultado
"Ana " + "López"	Ana López
"saca" + "puntas"	sacapuntas

3.4.- Ordre d'avaluació dels operadors:

A l'hora de resoldre una expressió, l'ordre a seguir és el següent:

1. Parèntesi. ()
2. Potència. ^
3. Multiplicació i divisió. * /
4. Sumes i restes. + -
5. Concatenació. +
6. Relacionals. < <= > >= !=
7. Negació. NOT
8. Conjunció. AND
9. Disjunció. OR

L'avaluació d'operadors d'igual ordre es realitza de esquerra a dreta. Cal tindre en compte que este ordre d'avaluació pot variar segons determinats llenguatges de programació.