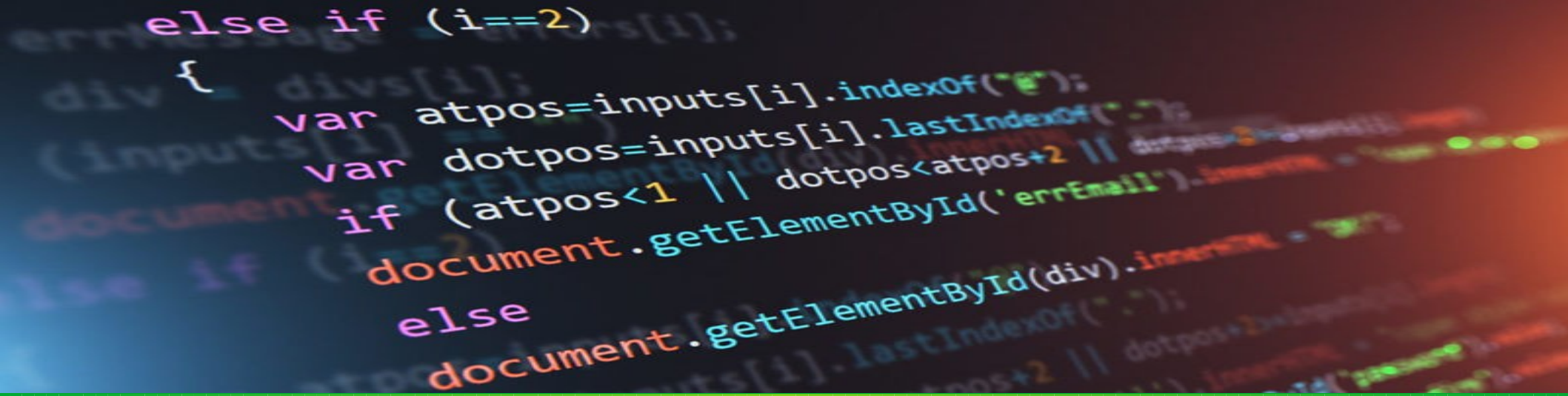


PROGRAMACIÓ

UNITAT 1: CONCEPTES BÀSICS

CONTINGUTS

1. INTRODUCCIÓ
2. ALGORITME
3. CICLE DE VIDA D'UN PROGRAMA
4. DOCUMENTACIÓ
5. OBJECTES D'UN PROGRAMA
6. REPRESENTACIÓ (Diagrames de flux, Pseudocodi)

INTRODUCCIÓ

- L’ordinador s’utilitza principalment per **resoldre problemes** mitjançant l’utilització de **programaes** escrits per programadors.
- **Pogrames:** Mètodes per resoldre problemes.
- Per a escriure un programa, el primer és que el programador sàpia **resoldre el problema** que estem tractant, **identificar** quines són les **dades d'entrada** i a partir d'ells obtindre les dades d'eixida, és a dir, la **solució**.

ALGORITME

Conjunt **ordenat** i **finit** d'operacions que permeten resoldre un problema.

Característiques:

- Nombre finit de passos
- Acaba en un temps finit.
- Operacions definides de manera precisa i sense ambigüitat.
- Pot tindre diverses dades d'entrada i com a mínim una dada d'eixida.

Ejemplo, el algoritmo para ***freír un huevo*** podría ser el siguiente:

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

Dades d'eixida: ou caigut.

Procedimiento:

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 en un ordinador es denomina programa.**

CICLE DE VIDA D'UN PROGRAMA

1. <u>Fase de definició:</u> Es declara quina és la **situació de partida** i l'entorn de dades d'entrada, els resultats desitjats, on han de registrar-se i quina serà la **situació final** a la qual ha de conduir el problema **després de ser implementat**.

2. <u>Fase de desenvolupament:</u> Es dissenyen estructures de dades i dels programes, **s'escriuen i documenten** aquests, i **es prova** el programari. En aquesta fase, es converteix l'algorisme en programa.

3. <u>Fase de manteniment:</u> Posada a punt del programa. Consta de les següents etapes:

- Detecció d'errors.
- Depuració d'errors.
- Prova del programa.

→ **En cadascuna d'aquestes fases es poden detectar problemes que ens fan replantejar-nos conceptes de la fase anterior i refer el programari creat amb les oportunes correccions.**

DOCUMENTACIÓ

La major part dels projectes exigeixen la realització d'una **planificació prèvia**. Aquesta planificació ha de determinar el model de **cicle de vida** a seguir, els **terminis** per a completar cada fase i els **recursos** necessaris a cada moment. Tot això s'ha de plasmar en una **documentació** completa i detallada de tota l'aplicació.

La **documentació** associada al programari pot classificar-se en **interna** i **externa**.

- **Documentació interna:** Correspon a la que s'inclou dins del codi font dels programes. Ens aclareixen aspectes de les pròpies instruccions del programa.
- **La documentació externa** és la que correspon a tots els documents relatius al disseny de l'aplicació, a la descripció de la mateixa i els seus mòduls corresponents, als manuals d'usuari i els manuals de manteniment.

OBJECTES D'UN PROGRAMA

Entenem per **objecte** d'un programa **tot allò que puga ser manipulat per les instruccions**. En ells s'emmagatzemaran tant les dades d'entrada com els d'eixida (resultats). Els seus atributs són:

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

OBJECTES D'UN PROGRAMA

CONSTANTS

Són objectes el **valor dels quals roman invariable** al llarg de l'execució d'un programa. Exemples de constant són el número **pi**, el número **e** o l'**IVA** aplicable.

Per exemple: pi = 3.141592 e = 2.718281 IVA = 0.16

VARIABLES

Són objectes el **valor dels quals pot ser modificat** al llarg de l'execució d'un programa.

Per exemple: a = 2

Ací el nom o **identificador** de la variable és **a**, i el **valor** assignat és **2**. Això no significa que posteriorment no puga canviar el seu valor per un altre. Per exemple:

b = 10

a = b + a

Com **b** val **10** i **a** inicialment val **2**, **a** prendrà el valor de 10 més 2, és a dir **12**.

OBJECTES D'UN PROGRAMA

EXPRESIONS

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: ”Don ” + “José” subcadena(nombre, 1, 5)

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

  Per exemple: a < 0 (a > 1) and (b < 5)

OBJECTES D'UN PROGRAMA

OPERADORS

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

Relacionals:

- S'usen per a formar expressions que en 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

OBJECTES D'UN PROGRAMA

Exemples

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

- **ASCII** (acrònim anglés de American Standard Code for Information Interchange - Codi Estàndard Estaunidenc per a l'Intercanvi d'Informació), és un codi de caràcters basat en l'alfabet llatí, tal com s'usa en anglés modern. Pots veure la taula en el següent enllaç: https://ascii.cl/es/

OBJECTES D'UN PROGRAMA

Aritmètics:

S'utilitzen per a 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	21

OBJECTES D'UN PROGRAMA

Lògics o booleans:

La combinació d'expressions amb aquests operadors produeixen el resultat vertader o fals.

Operador	Definición
No	Negación
Y	Conjunción
O	Disyunción

- El comportament d'un operador lògic es defineix mitjançant la seua 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:

OBJECTES D'UN PROGRAMA

- **NO** lògic (**NOT**) o negació:

A	NOT A
V	F
F	V

- **O** lògica (**OR**) o disjunció:

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

→ Operador unari (aplicat a un únic operant). Canvia el valor de vertader (V) a fals (F) i viceversa.

→ Operador n-ari (aplicat a 2 o més operands). Si tots els operands són F retorna F; si hi ha algun que siga V retorna V.

OBJECTES D'UN PROGRAMA

- **I** lògica (**AND**) o conjunció:

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

- Operador n-ari . Si tots els operands són V retorna V; si hi ha algun que siga F retorna 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

OBJECTES D'UN PROGRAMA

Parèntesi ():

Nien expressions.

Exemples:

Operació (3 * 2) + (6 / 2) Resultat 9

Operador Alfanumèric (+)

Uneixen dades de tipus alfanumèric.

Exemples:

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

Ordre d'avaluació dels operadors:

1. Parèntesi.
2. Potència.
3. Multiplicació i divisió.
4. Sumes i restes.
5. Concatenació.
6. Relacionals.
7. Negació.
8. Conjunció.
9. Disjunció.

➔ L'avaluació d'operadors d'igual ordre es realitza d'esquerra a dreta. Aquest ordre d'avaluació té algunes modificacions en determinats llenguatges de programació.

REPRESENTACIÓ

Existeixen diverses formes de representació d'algorismes. Les més importants són els **diagrames de flux** i el **pseudocodi**.

Diagrames de flux

Durant el disseny d'un programa i en les seues fases d'anàlisis i programació, sorgeix la necessitat de **representar d'una manera gràfica els fluxos** que van seguir les dades manipulades per aquest, així com la seqüència lògica de les operacions per a la resolució del problema.

Aquesta **representació gràfica** ha de tindre les següents qualitats:

1. Senzillesa en la seua construcció.
2. Claredat en la seua compressió.
3. Normalització en el seu disseny.
4. Flexibilitat en les seues modificacions.

En la representació de **diagrames de flux** és convenient seguir les següents regles:

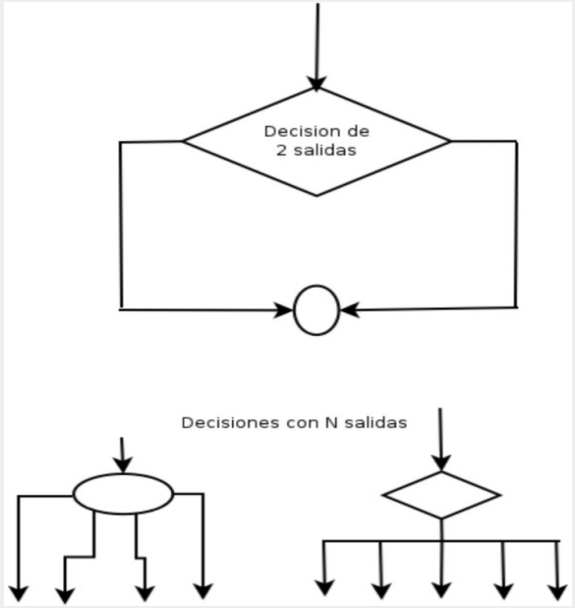
- El començament del programa figurarà en la part superior del **diagrama de flux**.
- El símbol de començament haurà d'aparéixer una sola vegada en el **diagrama de flux**.
- El flux de les operacions serà, sempre que siga possible de dalt a baix i d'esquerra a dreta.
- S'evitaran sempre els encreuaments de línies utilitzant connectors.

REPRESENTACIÓ

Aquesta serà la **representació** que utilitzarem durant el curs:

Símbols d'operació

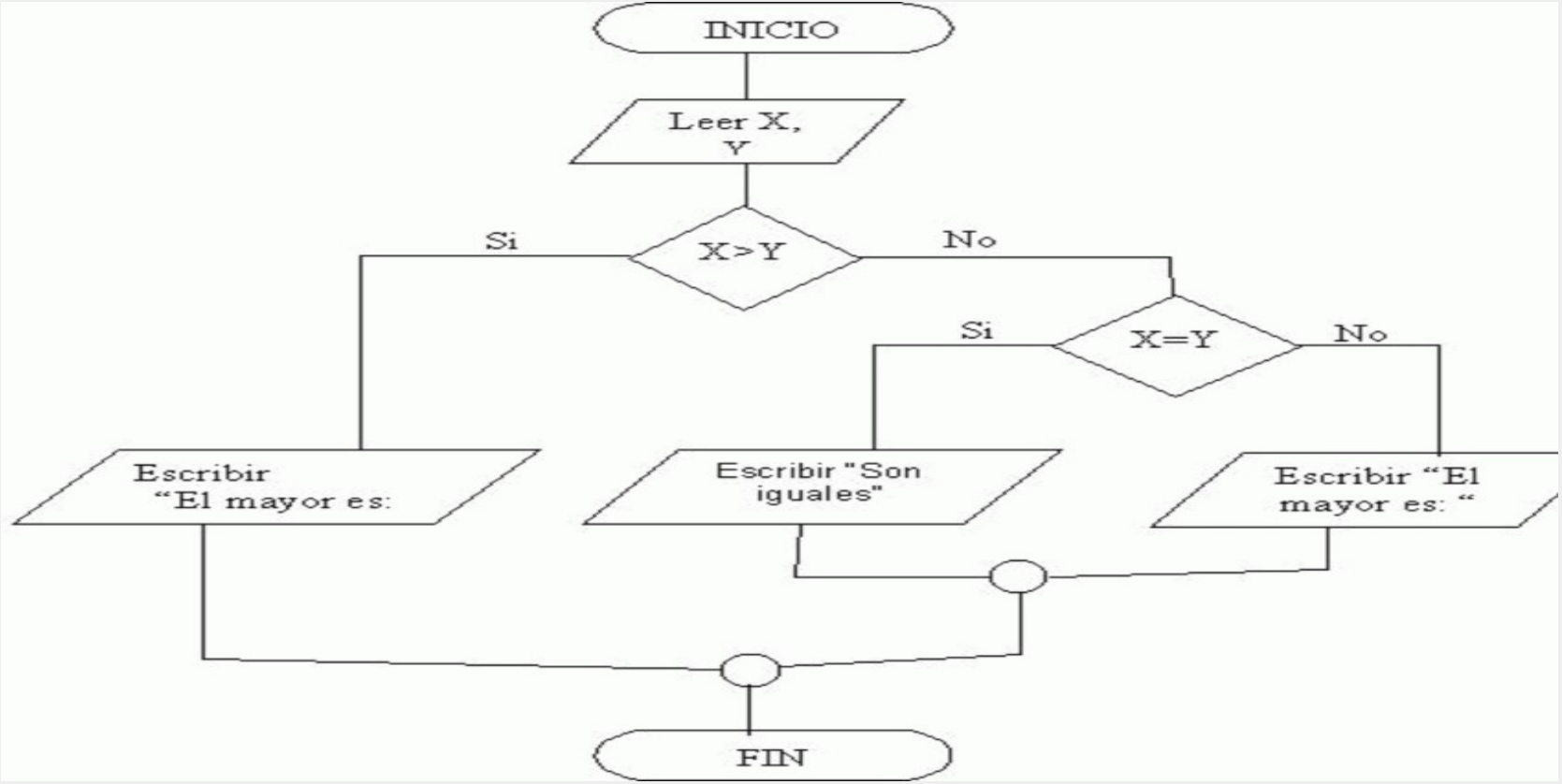
Símbols de decisió



Símbols de connexió

REPRESENTACIÓ

Exemple: Representació gràfica mitjançant **diagrama de flux** de l'algorisme que llig dos números "X" e "I", determina si són iguals, i en cas de no ser-ho, indica quin d'ells és el major:



REPRESENTACIÓ

Pseudocodi

A més de les representacions gràfiques, un programa pot descriure's mitjançant un **llenguatge intermedi** entre el llenguatge natural i el llenguatge de programació.

La **notació en pseudocodi** es caracteritza per:

a) Facilitar l'obtenció de la solució mitjançant la utilització del disseny descendent o Top-down.

b) Ser una manera de codificar els programes o algorismes fàcil d'aprendre i utilitzar.

c) Possibilitar el disseny i desenvolupar els algorismes d'una manera independent del llenguatge de programació que es vaja a utilitzar quan s'implemente el programa.

d) Facilitar la traducció de l'algorisme a un llenguatge de programació específic.

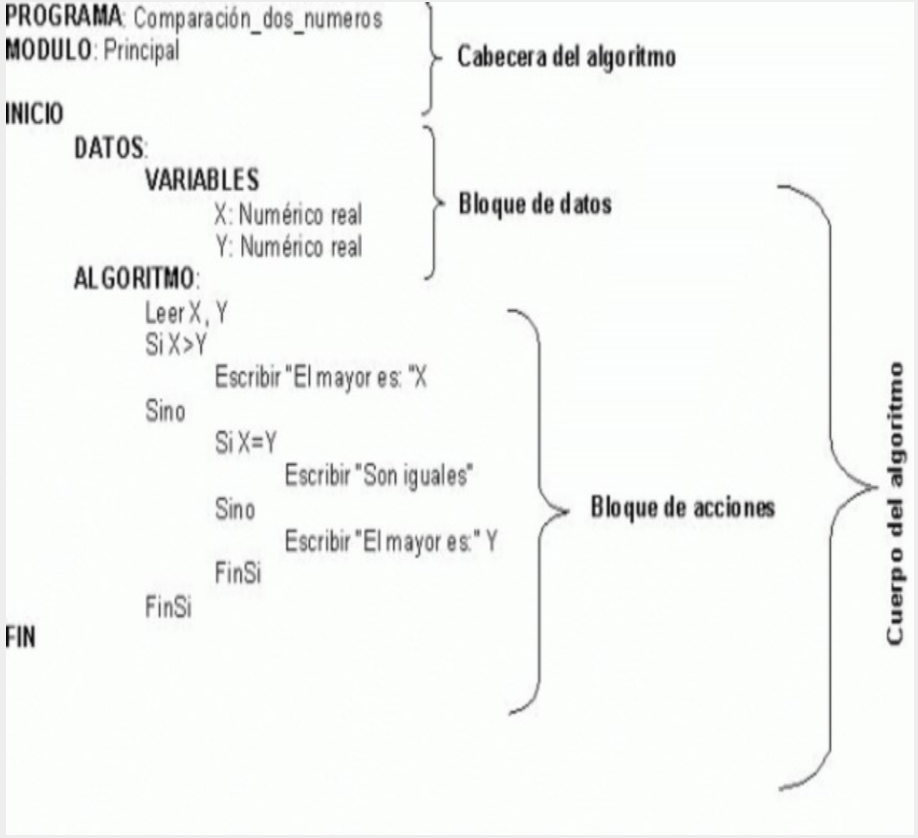
e) Permetre un gran flexibilitat en el disseny de l'algorisme que soluciona el problema, ja que es poden representar les accions d'una manera mes abstracta, no estant sotmeses a les regles tan rígides que imposa un llenguatge de programació.

f) Possibilitar futures correccions i actualitzacions en el disseny de l'algorisme per la utilització una sèrie de normes, que delimiten el treball del desenvolupador.

Quan s'escriu un algorisme mitjançant la utilització de pseudocodi, s'ha de "**sagnar**" el text respecte al marge esquerre, amb la finalitat que es comprenga més fàcilment el disseny que s'està realitzant.

REPRESENTACIÓ

Exemples:




```
algoritmo Sumar

variables
  entero a, b, c

inicio
  escribir( "Introduzca el primer número (entero): " )
  leer( a )
  escribir( "Introduzca el segundo número (entero): " )
  leer( b )
  c ← a + b
  escribir( "La suma es: ", c )
fin
```