CFGS DAM. Entorns de Desenvolupament

UD1. Desenvolupament de Programari

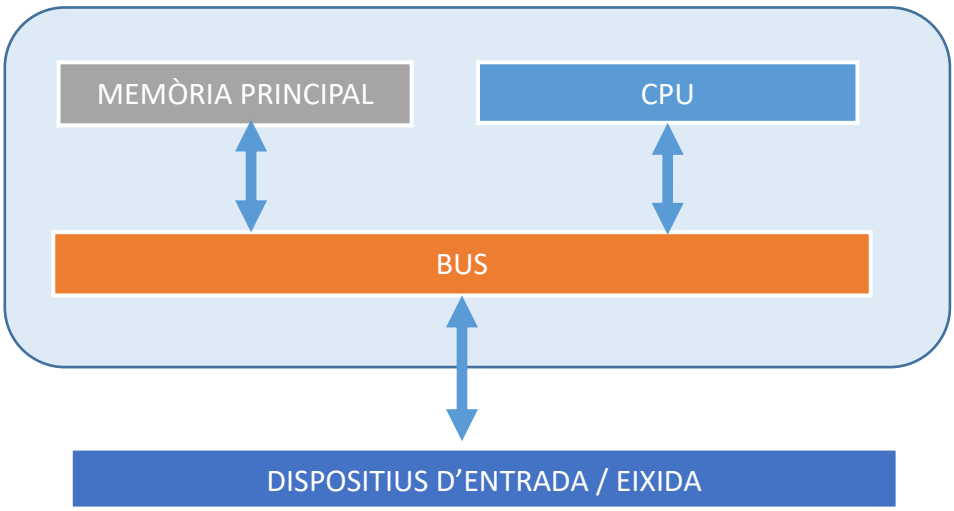
Continguts

1. Concepte de programa infomàtic

L'ordinador

Màquina capaç de processar instruccions a gran velocitat.

ARQUITECTURA DE VON NEUMANN.



Explicació ampliada del model

Aquestos son el components abreviats d'un ordinador, però no hem d'oblidar que dins de la CPU ens trobem altres com el **rellotge**, els **registres**, la **ALU**, la **memòria caché…** Estos components treballen **de forma conjunta** per tal d'oferir una **resposta adequada**.

Algoritme

Si volem que l'ordinador faça alguna cosa, li haurem d'indicar que ha de fer.

> L'algoritme és el **conjunt d'accions ordenades** que ha de seguir l'ordinador per a resoldre un **problema** o fer una **tasca**.

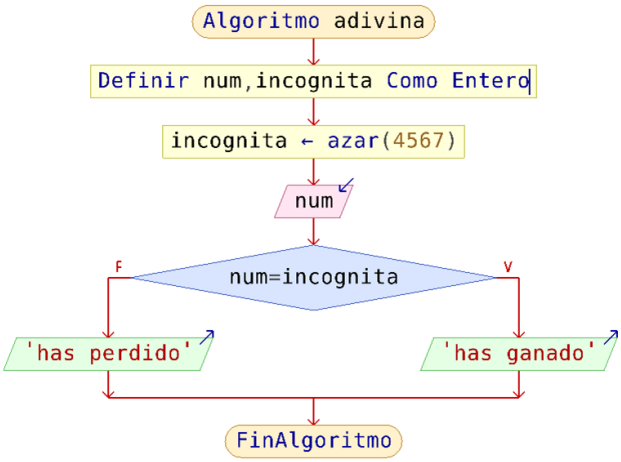
Si volem jugar amb l'ordinador **a endevinar un número**, haurem d'indicar-li **com es juga**. Els passos s'escriuen en un llenguatge natural:

1. Escriu un número, i amaga'l.
2. Pregunta'm un número.
3. Si el meu número és igual al teu, llavors, contesta "has guanyat", si no ho és, contesta "has perdut".

En un llenguatge més estricte (pseudocodi de PSeInt):

```
1  Algoritmo holamundo
2      Definir num, incognita Como Entero;
3      incognita<-azar(10);
4      Leer num;
5      Si num=incognita Entonces
6          Escribir "Has guanyat";
7      SiNo
8          Escribir "Has perdut";
9      Fin Si
10 FinAlgoritmo
```

De manera gràfica (ordinograma):



En tots els casos les accions s'expressen en el nostre **llenguatge natural** (espanyol).

Programa

L'algoritme es transforma en un **programa**, on cada **acció** es transforma a **instruccions del programa**.

> Es un arxiu escrit en **llenguatge màquina** o **interpretat** per la màquina que realitza alguna acció, una funció específica.

Les **instruccions** són molt més **estrictes** que les **accions del llenguatge natural**, la transformació des de pseudocodi o des d'un ordinograma és menor.

Les **instruccions** s'escriuen en un llenguatge triat **Java**, **C**, **C++**, **PHP**, **Javascript**, Python, Pascal, etc.

Programa font

> El programa font és el **text amb les instruccions** del programa.

A més, del text de les instruccions cal afegir el text necessari per a crear un programa correcte en el llenguatge triat.

L'exemple anterior escrit a **Java**:

```
public class adivina {
    public static void main(String args[]) throws IOException {
            BufferedReader bufEntrada = new BufferedReader(new
InputStreamReader(System.in));
            int incognita;
            int num;
            incognita = Math.floor(Math.random()*4567);
            num = Integer.parseInt(bufEntrada.readLine());
            if (num==incognita) {
                    System.out.println("has guanyat");
            } else {
                    System.out.println("has perdut");
            }
    }
}
```

L'exemple escrit en **PHP**:

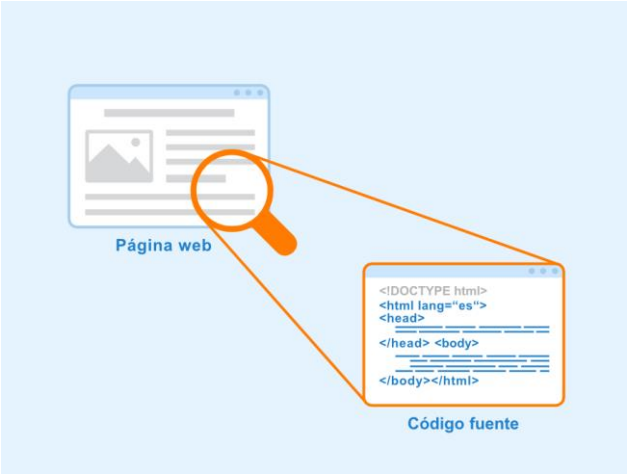
```
<?php
	$stdin = fopen('php://stdin','r');
	settype($num,'integer');
	settype($incognita,'integer');
	$incognita = rand(0,4566);
	fscanf($stdin,"%d",$num);
	if ($num==$incognita) {
		echo 'has guanyat',PHP_EOL;
	} else {
		echo 'has perdut',PHP_EOL;
	}
?>
```

L'exemple escrit en **Python 3**:

```
from random import randint

if __name__ == '__main__':
	num = int()
	incognita = int()
	incognita = randint(0,4566)
	num = int(input())
	if num==incognita:
		print("has guanyat")
	else:
		print("has perdut")
```

Cada vegada que vulguem modificar el comportament del programa **haurem d'editar el text del programa font**.



El programa font està en el **disc dur** o en un **element extern**. És el codi que entén una persona, però l'ordinador només entén **0** i **1**, per tant cal **transformar el text a codi binari**.

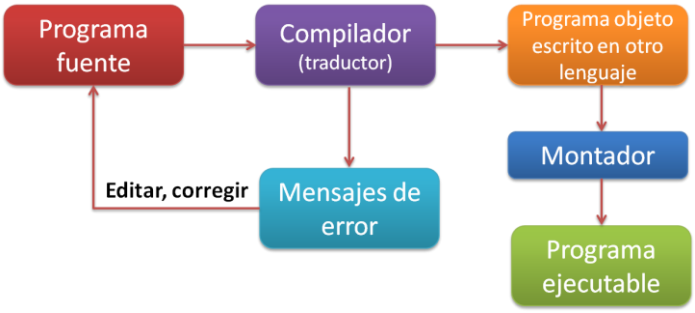
Programa objecte

> El programa objecte és la traducció del programa font a format binari (Codi màquina).

El programa objecte **no és directament executable** i està en el disc dur.

Per tal d'obtindre aquest programa ens podem ajudar de dos tipus de traductors: compiladors o interpretes, **depenent del llenguatge** triat.

Per exemple per a compilar **codi C** s'utilitza el compilador **GCC** y per a interpretar **Python** podem gastar l'interpret Python3.



Programa executable

El **programa objecte** necessita un altre pas per a ser un programa **executable**, afegir-li els recursos necessaris: llibreries, espai de memòria, fluxos de comunicación, etc. El programa executable està en la **memòria principal**.

En **funció del llenguatge** triat s'usa un programa **diferent**:

- Un enllaçador que transforma tot el codi objecte a codi que executa la màquina (**C**, **C++** i **Go**).
- Un intèrprete que transforma una instrucció del codi objecte a codi que executa la màquina. El programa s'executa instrucció a instrucció (**Javascript**, **Python** i **Ruby**). Es un procés directe, no hi ha compilació.
- Una màquina virtual que transforma el codi intermedi al codi que executa la màquina real (**Java**, **.Net**). Per exemple, Java emplea el Java Runtime Enviroment (JRE).

Com ha de ser el programa desenvolupat

Cal elaborar el programa de tal forma que siga:

- **Correcte**, un programa serà correcte si fa el que ha de fer, de manera que s'han d'especificar de manera **molt clara** quines són les dades sobre els quals es treballarà i el que s'ha de fer amb elles.
- **Eficient**, ha de consumir la **menor quantitat de recursos** (temps i/o memòria) possible.
- **Clar**, és molt important la **claredat** i **llegibilitat** de tot programa, ja que facilitarà al màxim la tasca de **manteniment posterior** del programari, que **no serà feta per el programador necessàriament**.
- **Modular**, els programes solen **dividir-se en subprogrames** (mòduls), per a **reduir la complexitat** del que s'està implementant i facilitar la **reutilització de codi**.
- **Documentat**, la documentació facilita la **claredat** i la **reutilització del codi**.

2. Tipus de llenguatges de programació

2.1. Llenguatge de programació

Un programa i les sentències que el formen s'escriuen **seguint les regles d'un llenguatge de programació**.

Un llenguatge de programació consta de:

- Un **lèxic**, el conjunt de caràcters permesos.
- Una **sintaxi**, que indica com escriure les expressions del llenguatge.
- Una **semàntica**, que indica el significat de les expressions.

2.2. Segons el nivell

Dins dels llenguatges de programació es poden establir **moltes classificacions**. Nomenarem les més importants:

- **Baix nivell** (no s'assembla tant al llenguatge humà)
  - **Depenen del maquinari de la màquina**.
  - **Llenguatge assemblador**, cada instrucció s'escriu amb un mnemotècnic de 3 o 4 lletres (mov, add…) que es transformarà a llenguatge màquina 0 i 1.
- **Alt nivell** (s'assembla al llenguatge humà)
  - Són **independents del maquinari**.

- Utilitzen un llenguatge pròxim al llenguatge natural.
- Cada instrucció es traduirà a multitud d'instruccions en assemblador.

2.3. Segons el procés de traducció

El codi font pot ser traduït de diferents formes per a ser **executat en la màquina**:

- **Interpretat**, el codi font es tradueix, **línia a línia**, i s'executa, sense guardar la traducció realitzada. És més **dinàmic**, es poden canviar elements durant l'execució, però **cada línia s'ha de traduir cada vegada que s'executa**, per tant, és més **lent** (Javascript, Python i Ruby).

- **Compilat**, el codi font es **tradueix sencer**, i posteriorment s'enllaça amb els recursos necessaris, per a fer un **programa ejecutable**. És més **estàtic**, no es pot canviar el traduït, però és més **ràpid**. (C, C++ i Go).

- S'anomena **codi intermedi** a la **compilació del codi font**. Es crea un codi intermedi (bytecodes a Java) de tot el text, que utilitzarà una **màquina virtual**. Es pot utilitzar en tots els sistemes que tinguen aquella màquina virtual, però és **més lent** ja que està la **màquina virtual pel mig** (Java, .Net)

2.4. Segons la forma de programar

Llenguatges **imperatius** o **procedurals**, usen bàsicament l'assignació, a més de les instruccions de selecció i iteració. Els programes es basen en la idea d'una **seqüència d'instruccions que construeixen l'algoritme** (C, ADA, Pascal, FORTRAN).

Llenguatges **funcionals**, usen **funcions matemàtiques** que es combinen utilitzant condicionals, recursividad, composició funcional, no usen l'assignació (LISP, Scheme, Haskell).

Llenguatges **lògics** o **declaratius**, el programa és un conjunt de declaracions lògiques sobre el resultat que hauria d'obtindre la funció (se li indica què hauria d'obtindre). L'execució del programa aplica aqueixes declaracions per a obtindre una sèrie de solucions a un problema (PROLOG, SQL).

https://holamundo.io/2022/12/13/programacion-imperativa-vs-programacion-declarativa/

Llenguatges **orientats a objectes**, usen **objectes** que implementen part del programa, encapsulant les dades, que defineixen el seu estat, al costat dels mètodes que manegen les dades. Així cada part del programa **és més independent**, facilitant el manteniment (Java, C++, C#).

Els llenguatges **multiparadigma**, que serien aquells que admeten **diferents formes d'arribar a una solució**: PHP, Python o JavaScript, entre d'altres.

3. On s'usen?

Cada llenguatge té un o més **camps d'aplicació**:

- Computació científica, per a la realització de càlcul complexos de manera ràpida i precisa (FORTRAN).
- Gestió d'informació, vendes, inventaris, censos, etc… (COBOL, SQL).
- Intel·ligència artificial, per a la modelització de sistemes humans, de deducció lògica etc… (PROLOG, LISP, Python).
- Sistemes informàtics, sistemes operatius, compiladors, protocols, electrònica… (C).
- Entorn web, pàgines, aplicacions, etc… (Java, Perl, PHP, JavaScript)

4. Fases de desenvolupament d'una aplicació

Per a realitzar un projecte, començarem per veure quins són els **objectius que volem aconseguir** i després pensarem que coses hem de fer per a aconseguir aquestes finalitats.

El cicle de vida d'un sistema d'informació comprèn les següents etapes:

- **Planificació**
- **Anàlisi**
- **Disseny**
- **Implementació**
- **Proves**
- **Instal·lació**
- **Ús i manteniment**

4.1. Planificació

Àmbit del projecte, quines coses **abastarà** el nostre projecte.

Estudi de viabilitat, si el projecte és possible o no, quins **problemes** pot suscitar en l'empresa, quins **beneficis** pot aportar.

Estimació de **costos**, què costarà en diners.

Planificació temporal, estimació del **termini de lliurament** dels elements del projecte.

Assignació de recursos, determinar les necessitats (personal, maquinari, programari) necessàries per a cada etapa del projecte.

4.2. Anàlisi

Determinar què hem de fer.

- **Captura de requeriments**:
  - Anàlisi del sistema actual (si existeix).
  - Necessitats del client.

- **Modelat**:
  - Modelatge de dades.
  - Modelatge de processos.

Modelar es abstraure els procediments y dades del sistema per tal de veure com se relacionen per a funcionar.

4.3. Disseny

Determinar **com han de ser les dades i els processos**.

- Disseny de la base de dades.
- Disseny de les aplicacions.

4.4. Implementació

Adquisició de components (programari ja creat).

Creació i integració (hardware i software) dels **recursos necessaris** perquè el sistema funcione.

4.5. Proves unitàries

Proves d'unitat, es prova cada mòdul per separat, per a comprovar que el mòdul fa el que es va dissenyar.

Proves d'integració, es proven tots els mòduls junts, per a comprovar que la interacció entre els mòduls és correcta.

- **Proves alfa**, primeres proves del sistema complet.
- **Proves beta**, últimes proves del sistema complet.
- **Test d'acceptació**, prova amb el client per a l'acceptació del sistema.

4.6. Instal.lació / ús / manteniment

Instal·lació del sistema en el **lloc definitiu**.

Ús del sistema per part dels **usuaris finals**.

- Manteniment **adaptatiu**, adaptar el sistema a noves circumstàncies (nou maquinari).
- Manteniment **correctiu**, eliminar errors.
- Manteniment **perfectiu**, millorar el sistema.

4.7. Documentació

Durant totes les fases del desenvolupament s'ha **creat documentació**. La documentació d'una fase **facilita el treball en les fases posteriors**.

Ens centrarem en la **documentació tècnica**. Tenim **dos tipus** de documentació:

- **Externa**, és la documentació sobre un suport extern al programa: estudi de requisits del sistema, algoritmes, codis fonts, manuals d'usuari, etc.

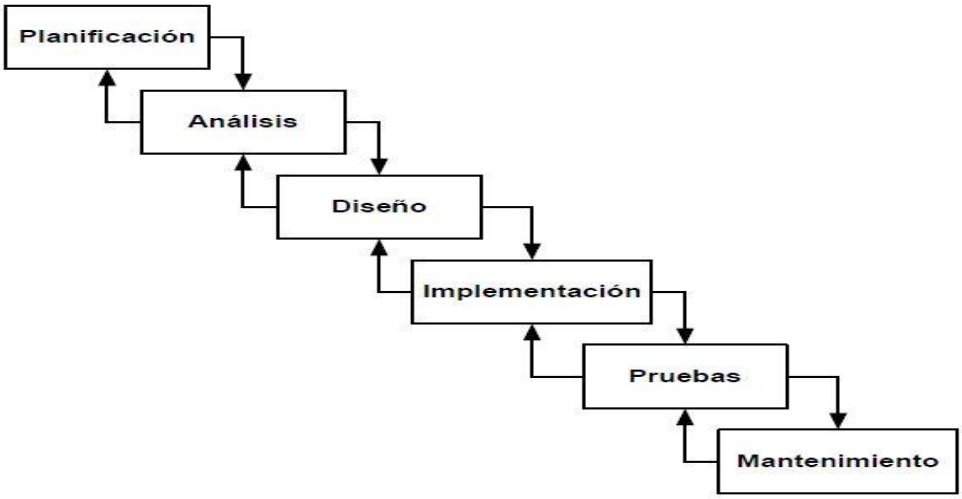
- **Interna**, és la documentació interna al programa, **comentaris** que el programador escriu en el codi font d'un programa són aclariments que ajudaran quan es vulga revisar o modificar el codi font.

En les següents unitats es treballarà la creació de la documentació externa del software creat. Emprarem **JAVADOC** (en JDK) per tal d'obtindre de manera automàtica la **documentació** del software creat amb JAVA. Es detallaran el **paquets**, les **classes** i una **descripció de les mètodes** i les mateixes **classes**.

5. Models de cicle de vida

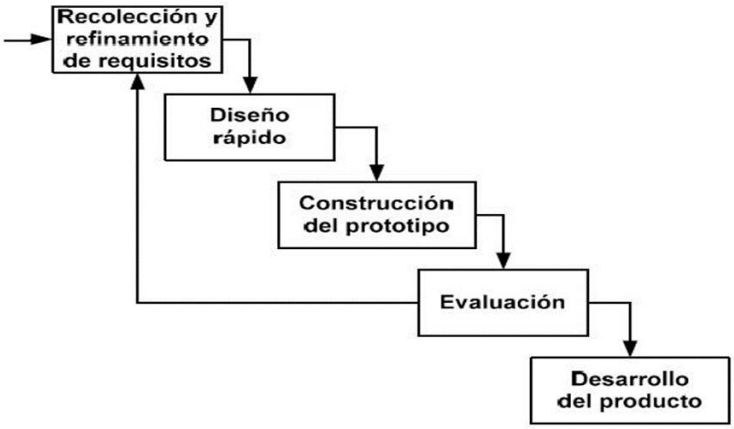
5.1. Model en cascada

Es tracta del **model clàssic**, en el que cada acció a soles s'executa **una vegada**. A més, en cadascuna **es verifiquen el resultats de l'anterior**.



5.2. Model prototipat

Un **prototip** és un sistema desenvolupat de manera **parcial**, així que es pot **adaptar i modificar molt ràpidament**.



5.3. Model en espiral

Cada volta suposa un **recorregut de totes les fases de desenvolupament**, i pot donar totes les voltes que siga necessari.