Algoritmes

Objectius

- S'han identificat les diferents fases de creació d'un algoritme
- S'ha comprovat la necessitat de crear un algoritme
- S'han identificat les tècniques de descomposició d'un problema

Resolució de problemes

Igual que les tasques de certa complexitat del món real, la creació d'un programa requereix un pas previ on cal reflexionar sobre què és exactament el que vols fer i com fer-ho.

➜ No és recomanable afrontar aquesta tasca seient directament davant de l'ordinador i començant a escriure línies de codi

Quan t'enfrontes a un problema nou és imprescindible una etapa en la qual estudiar el problema, les dades que vols tractar i les tasques que ha de dur a terme, és a dir, l'algoritme del programa.

La resolució de problemes mitjançant un ordinador, suposa la creació d'aplicacions que passen per diferents fases:

- Fase d'identificació
- Fase de resolució del problema
- Fase d'implementació en un llenguatge de programació

➜ Crea la SOLUCIÓ de manera LÒGICA, sense pensar en el llenguatge que vas utilitzar per a codificar-la.

Fase d'Identificació

Has d'entendre el problema que has de resoldre, això suposa entendre tots els conceptes que apareixen en l'enunciat.

Per exemple, si demanen: "*Realitza una aplicació que genera primitives aleatòries*", has de conéixer el significat dels conceptes de les paraules: genera, primitiva i aleatori, si no, no sabràs que demanen.

Fase de resolució del problema

Aquesta fase inclou

- l'anàlisi del problema.
- el disseny de l'algoritme.
- la verificació de l'algoritme.

Anàlisi del problema

El primer pas per a trobar la solució a un problema és analitzar-lo.

Has d'examinar acuradament el problema a fi de formar-te una idea clara sobre les accions a realitzar i les dades necessàries per a aconseguir crear la solució.

En l'exemple de la primitiva, has d'adonar-te que una primitiva són 6 nombres de l'1 al 49 sense repeticions, que generar és sinònim de crear i que aleatori és sinònim de "a l'atzar".

Disseny de l'algoritme

➜ Un ALGORITME és una seqüència ordenada de passos, sense ambigüitats, que condueixen a la resolució d'un problema donat.

L'algoritme es pot expressar en un llenguatge natural (valencià, castellà, etc.), en un llenguatge més estricte com el pseudocodi, o mitjançant una notació gràfica com són un ordinograma, un diagrama de Nassi-Shneiderman, un diagrama d'activitat, etc.

Un algoritme ha de ser:

- Precís, cal definir l'ordre de realització de cadascun dels passos.
- Definit, l'execució de l'algoritme, proporcionant-li les mateixes dades, sempre, proporciona els mateixos resultats.
- Finit, l'algoritme ha d'acabar, és a dir, ha de tindre un nombre finit de passos.

En un algoritme has de definir:

- La informació d'**entrada**, és la informació proporcionada a l'algoritme.
- Les operacions o càlculs necessaris per a la solució del problema, el **procés**.
- La informació d'**eixida**, és la informació retornada per l'algoritme.

Normalment l'ordre és:

1. Eixida. Què vols obtindre?
2. Entrada. Què necessites?
3. Procés. Què cal fer amb l'entrada per a obtindre l'eixida?

Per exemple, es demana: "*Realitza una aplicació que calcula la superfície d'un rectangle*". El primer que has de fer és entendre el que et demanen, per tant, has de saber el que és un rectangle, una superfície i què implica el seu càlcul. És a dir, conéixer tots els conceptes que apareixen en l'enunciat.

Per a la realització de l'algoritme, pots plantejar-te les preguntes següents:

1. Eixida
   - Quines són les dades d'eixida? (l'àrea del rectangle)
   - Quantes dades d'eixida es produiran? (1)
   - Quin format i precisió tindran els resultats? (un nombre amb decimals)
   - On es realitza l'eixida? (en pantalla)
   - Quines unitats s'utilitzaran? (s'indiquen en els missatges)
2. Entrada
   - Quines dades són necessàries? (la base i l'altura del rectangle)
   - Quines unitats s'utilitzaran? (s'indiquen en els missatges)
   - D'on s'obté l'entrada? (de teclat)
3. Procés
   - Com es calcula la superfície d'un rectangle? (superfície = base * altura)
   - Es controla l'entrada? (no, es considera que l'usuari no s'equivoca)

L'ordre de les preguntes no és fix, pots preguntar, primer, com es calcula la superfície per a després esbrinar quines dades d'entrada es necessiten.

L'algoritme per a resoldre el problema és el següent:

1. Entrada: visualitzar el text "tots els valors són en metres"
2. Entrada: visualitzar el text "base = " i llegir de teclat la base (un nombre real)
3. Entrada: visualitzar el text "altura = " i llegir de teclat l'altura (un nombre real)
4. Procés: Calcular superfície (superfície = base * altura)
5. Eixida: Visualitzar el text "àrea = " seguit de la superfície seguit de "m^2"

➜ L'ALGORITME ha de ser INDEPENDENT del llenguatge de programació, però fàcilment traduïble a qualsevol d'ells.

Verificació de l'algoritme

Una vegada que has acabat d'escriure un algoritme és necessari comprovar si fa la tasca per a la qual s'ha dissenyat i produeix un resultat correcte i esperat, és a dir, has de crear un pla de proves.

➜ El PLA DE PROVES es defineix durant el desenvolupament de l'algoritme.

Comprovar un algoritme suposa la seua execució manual, usant dades significatives que abasten tot el rang de valors.

Pots realitzar una traça de l'algoritme, anotant en un full de paper els canvis que es produeixen en les dades manejades en l'algoritme fins a l'obtenció del resultat final.

Per a l'algoritme anterior, la traça seria la següent. La taula té una columna per a l'acció que s'executa i una columna per al contingut de cada dada després de l'execució de l'acció (base, altura, superfície).

acció	base	altura	superfície
Llegir de teclat 4.6 per a base	4.6	0	0
Llegir de teclat 5.1 per a altura	4.6	5.1	0
Calcular la superfície	4.6	5.1	23.46
Visualitzar en pantalla superfície	4.6	5.1	23.46

En l'ordinador utilitza les facilitats de depuració que et dona l'entorn.

Fase d'implementació

Quan l'algoritme està dissenyat, representat mitjançant pseudocodi i verificat s'ha de passar a la fase de codificació o traducció de l'algoritme a un determinat llenguatge de programació, que haurà de ser completada amb l'execució i comprovació de l'aplicació en l'ordinador (el pla de proves).

La solució en Java és

```
double base;                                  // defineix la dada base, un nombre amb decimals
double altura;                                // defineix la dada altura, un nombre amb decimals
Scanner teclat = new Scanner(System.in);      // crea un objecte Scanner que permet llegir de teclat
System.out.print("els valors estan en metres"); // visualitza el text "els valors estan en metres"
System.out.print("base = ");                  // visualitza el text "base = "
base = teclat.nextDouble();                   // obté un double del teclat i el guarda en base
System.out.print("altura = ");                // visualitza el text "altura = "
altura = teclat.nextDouble();                 // obté un double del teclat i el guarda en altura
double area = base * altura;                  // multiplica base per  altura i guarda el resultat en area
System.out.println("àrea = " + area +"m²");   // visualitza el text "àrea = " amb el contingut de area i "m²"
```

Aquest codi s'escriu en el mètode main de la classe principal

Descomposició del problema

La capacitat dels humans per entendre problemes complexos és limitada, ja que, només som capaços d'abastar uns pocs elements simultàniament. Per tant, si la resolució d'un problema suposa la manipulació de molts elements diferents, és molt fàcil, no ja simplement equivocar-se, sinó tan sols saber per on començar.

Es necessita una estratègia per a afrontar la resolució de problemes complexos, una de les solucions més obvies, és descompondre el problema complex en un conjunt de problemes més simples.

Existeixen dues tècniques a l'hora de descompondre un problema

- Descendent (top-down), es basa en el principi de "*divideix i venceràs*". Aquesta tècnica identifica les tasques més importants del problema i les separa. Després, a cada tasca separada se li aplica la mateixa tècnica d'identificació de tasques i la corresponent separació, fins a arribar a una tasca individual fàcil de resoldre i de codificar.

- Ascendent (bottom-up), es resolen els problemes individuals amb detall i aquestes solucions s'integren per a resoldre un problema més complex, es reitera la integració fins a resoldre el problema inicial.

Nosaltres usarem el disseny descendent.

➜ És convenient cercar subproblemes idèntics o pareguts, per així poder reutilitzar les solucions.

Aquests subproblemes es transformen en mètodes a l'hora d'implementar el programa.

A l'hora de descompondre un problema en subproblemes, has de tindre en compte:

- Si un problema que sembla complex es descompon en molt pocs nivells, val la pena donar una segona ullada.
- Si un problema no massa complex té massa nivells, potser s'ha anat massa lluny en la descomposició.
- El nombre de passos inclosos en un subproblema no ha de ser molt gran, i ser fàcil de seguir i entendre.
- Si el nombre de passos inclosos en un subproblema és molt gran, potser cal aplicar nous nivells de descomposició.
- Els noms assignats als subproblemes han de ser autoexplicatius i expressar clarament la tasca que estan resolent.

Descomposició de cuinar fideus japonesos yakisoba, la taula mostra els diferents nivells de descomposició dels passos a seguir.

Nivell1	Nivell 2	Nivell 3
Cuinar fideus japonesos yakisoba 1. Recopilar els ingredients 2. Cuinar les tallarines 3. Cuinar les carlotes 4. Cuinar les cebes 5. Preparació final	Recopilar els ingredients 1. Comprar els ingredients 2. Disposar-los sobre la taula	
	Cuinar les tallarines 1. Preparar l'aigua 2. Bullir les tallarines 3. Escórrer les tallarines 4. Apartar les tallarines	Preparar l'aigua 1. Escalfar l'aigua 2. Posar sal a l'aigua
	Cuinar les carlotes 1. Tallar les carlotes 2. Fregir les carlotes 3. Apartar les carlotes	Fregir les carlotes 1. Preparar la paella de fregir 2. Posar oli a la paella 3. Rossejar les carlotes 4. Tirar l'oli de la paella
	Cuinar les cebes 1. Tallar les cebes 2. Fregir les cebes 3. Apartar les cebes	Fregir les cebes 1. Preparar la paella de fregir 2. Posar oli a la paella 3. Rossejar les cebes 4. Tirar l'oli de la paella
	Preparació final 1. Barrejar els ingredients amb la salsa yakitori 2. Saltejar els ingredients 3. Posar els ingredients als plats de servir	

Orientació a objectes

Un programa orientat a objectes es compon, igual que una màquina, de la unió de cadascuna de les seues peces, que interactuant, faran que el tot funcione.

En el nostre cas les peces són un conjunt de línies de codi que representen l'abstracció d'un element de la vida real. Aquest conjunt de línies de codi recull les propietats i els comportaments de l'objecte.

La descomposició del problema complex suposa

- Obtindre els objectes.
- Associar les dades a un objecte particular.
- Definir els comportaments d'un objecte particular (que fa amb les seues dades).

Quan es té els objectes, cal relacionar-los per a resoldre el problema general. Relacionar dos objectes consisteix en el fet que un objecte demana a l'altre l'execució d'algun dels comportaments que té definit.