1.1. Blocs d’un programa informàtic.

Un programa informàtic és un conjunt de successos ordenats de manera seqüencial en el temps, un darrere l’altre, és a dir, és una seqüència d’accions (instruccions) que manipulen un conjunt d’objectes (dades).

Els blocs d’un programa informàtic són figures que representen funcions individuals que es connecten entre ells per a crear un programa.

Existeixen dos tipus de blocs que componen un programa informàtic:

-**Bloc de declaracions**: en aquest es poden identificar tots els objectes o dades que utilitza el programa, com ara les constants, les variables, els arxius…

-**Bloc d’instruccions**: són el conjunt d’accions i passos que segueix l’operador per aconseguir els objectius i resultats que s’havia proposat. Les diferents instruccions se solen agrupar en aquest bloc i, l’inici i la fi de cada bloc queden identificats en Java perquè les instruccions estan envoltades en claus, {…}.

El bloc d’instruccions està compost per tres parts segons la seva funció:

- Entrada de dades: són les instruccions que guarden a la memòria interna dades procedents de dispositius externs.

- Procés: són les instruccions que modifiquen els objectes d’entrada i en creen uns altres de nous.

- Sortida de resultats: són el conjunt d’instruccions que agafen les dades finals de la memòria interna i els envien als dispositius externs.

2.4.2 Tipus de dades compostes

Els tipus de dades compostes permeten aglutinar en una única variable una quantitat arbitrària de dades individuals pertanyents a algun tipus concret, és a dir, són aquelles que permeten emmagatzemar més d'un valor dins d'una única variable. Aquestes queden empaquetades d'una manera fàcil de gestionar a l'hora de consultar i modificar cada valor individual.

El tipus compost més estès en tots els llenguatges és l'**array**, disponible en la majoria de llenguatges d'alt nivell. Aquest permet emmagatzemar en una única variable un conjunt de valors en forma de seqüència, amb l'única restricció que tots pertanyin al mateix tipus de dades. Per tal de diferenciar els diferents valors emmagatzemats, l'array gestiona el seu contingut d'acord amb posicions que segueixen un ordre numèric: el valor emmagatzemat a la primera posició, a la segona, a la tercera, etc.

A part dels arrays, hi ha altres tipus compost de dades de diferents graus de complexitat i amb l'objectiu de solucionar problemàtiques semblants, però amb certs matisos.

2.5. Estructures de selecció, repetició i salt.

Les **estructures de repetició** o iteratives permeten repetir una mateixa seqüència d'instruccions diverses vegades, mentre es compleixi una certa condició. Anomenem bucle al conjunt d'instruccions que s'ha de repetir un cert nombre de vegades, i anomenem iteració cada execució individual del bucle.

La utilitat més directa d'una estructura de repetició es que, en cas que es vulguin executar exactament les mateixes instruccions moltes vegades, no les hagi d'escriure moltes vegades. En una estructura de repetició, l'evolució del mateix resultat que s'està calculant pot ser la senyal de sortida per deixar de fer iteracions. En podem trobar de quatre tipus:

- Repetir si es compleix una condició: la sentència "while"

La sentència while permet repetir l'execució del bucle mentre es verifiqui la condició lògica. Aquesta condició es verifica al principi de cada iteració. Si la primera vegada, tot just quan s'executa la sentència per primer cop, ja no es compleix, no s'executa cap iteració.

- Repetir almenys un cop: la sentencia "do/while"

La sentencia do/while permet repetir l'execució del bucle mentre es verifiqui la condició lògica. A diferència de la sentencia while, la condició es verifica al final de cada iteració. Per tant, independentment de com avaluï la condició, com a mínim sempre es durà a terme la primera iteració.

- Repetir un cert nombre de vegades: la sentencia "for"

La sentència for permet repetir un nombre determinat de vegades un conjunt d'instruccions.

- Combinació d'estructures de repetició

Com passava amb les estructures de selecció, res no impedeix combinar diferents estructures de repetició, imbricades unes dins de les altres, per dur a terme tasques més complexes. En darrera instància, la combinació i imbricació d'estructures de selecció i repetició conformarà el vostre programa. Quan es combinen estructures de repetició cal ser molt acurats i tenir present quines variables de control estan associades a la condició lògica de cada bucle. Tampoc no us oblideu de sagnar correctament cada bloc d'instruccions, per poder així identificar ràpidament on acaba i comença cada estructura.

En les **estructures de selecció**, hi ha diferents tipus de sentències, cadascuna amb les seves particularitats. Normalment, la diferència principal amb les de repetició està vinculada al moment en què s'avalua la condició per veure si cal tornar a repetir el bloc d'instruccions o no. La utilitat de les estructures de selecció va molt més enllà de ser una manera senzilla d'estalviar-se escriure el mateix codi moltes vegades. En podem trobar de quatre tipus:

- Una desviació temporal del camí: selecció simple

L'estructura de selecció simple permet controlar el fet que s'executi un conjunt d'instruccions si i només si es compleix la condició lògica (és a dir, el resultat d'avaluar la condició lògica és igual a true). En cas contrari, no s'executen.

- Dos camins alternatius: la sentència "if/else"

L'estructura de selecció doble permet controlar el fet que s'executi un conjunt d'instruccions, només si es compleix la condició lògica, i que se n'executi un altre, només si no es compleix la condició lògica.

- Diversos camins: la sentència "if/else if/else"

L'estructura de selecció múltiple permet controlar el fet que en complir-se un cas entre un conjunt finit de casos s'executi el conjunt d'instruccions corresponent.

- Combinació d'estructures de selecció

Les sentències que defineixen estructures de selecció són instruccions com qualssevol altres dins d'un programa, si bé amb una sintaxi una mica més complicada. Per tant, res no impedeix que tornin a aparèixer dins de blocs d'instruccions d'altres estructures de selecció. Això permet crear una disposició de bifurcacions dins el flux de control per tal de dotar al programa d'un comportament complex, per comprovar si es compleixen diferents condicions d'acord amb cada situació.

Les **estructures de salt** són instruccions que ens permeten trencar amb l'ordre natural d'execució dels nostres programes, poguent saltar a un determinat punt o instrucció. En Java existeixen dues sentències que permeten modificar el flux seqüencial d'un programa i provoquen un salt en l'execució. Aquestes sentències són break i continue. Les dos s'utilitzen amb les estructures de repetició per interrompre l'execució amb break o tornar al principi amb continue. A més, el break s'utilitza per interrompre l'execució d'un switch. En trobem de tres tipus:

- Break

La instrucció break permet acabar l'execució d'una estructura condicional o iterativa, de manera que l'execució del nostre programa segueix per la instrucció següent a una d'aquestes estructures.

- Continue

La instrucció continue serveix per aturar l'execució d'un bucle, però si el break feia que s'executés la següent instrucció darrere del bucle, el continue ens porta de nou al bucle, que es tornarà a executar, sempre que es compleixi la condició d'entrada.

- Return

La sentència return s'utilitza sobretot per acabar l'execució d'un mètode o funció. D'aquesta manera tornem a la instrucció que va fer la crida a aquest mètode.

2.6. Tractament de cadenes.

En programació, una cadena de caràcters és una seqüència ordenada d'elements que pertanyen a un cert llenguatge formal o alfabet anàlogues a una fórmula o a una oració. En general, una cadena de caràcters és una successió de caràcters (lletres, nombres o altres signes o símbols). Si no es posen restriccions a l'alfabet, una cadena podrà estar formada per qualsevol combinació finita dels caràcters disponibles.

Una cadena sol ser representada entre cometes dobles superiors ("..."), mentre que un caràcter d'aquesta cadena sol ser representat entre cometes simples ('...'). Un cas especial de cadena és la que conté zero caràcters, a aquesta cadena se li diu cadena buida.

La variable cadena és una cadena que pot emmagatzemar fins a sis caràcters, tenint en compte que es requereix un espai per emmagatzemar el caràcter nul al final de la cadena.

Les cadenes són una part fonamental de la majoria dels programes, així doncs Java té diverses característiques incorporades que faciliten la manipulació de cadenes. Java té una classe incorporada en el paquet java.lang que encapsula les estructures de dades d'una cadena. Aquesta classe, anomenada String és la representació com a objecte d'una matriu de caràcters que no es pot canviar.

Dins de les cadenes de caràcters podem trobar diverses funcions associades:

- strcpy: La funció strcpy es troba a la biblioteca <string.h> i s'utilitza per copiar una cadena de caràcters (font) en el lloc que ocupava una altra (destí). Aquesta còpia és destructiva, és a dir, que tots els caràcters que eren a la cadena destí desapareixen, encara que la cadena destí fos més llarga que la cadena font .La cadena destí es posa com a primer argument de la funció i la cadena font com a segon.
- strcat: Al programa anterior vam veure que la funció strcpy és desctructiva, però hi ha una altra funció a la llibreria <string.h> que copia una cadena (font) a una altra (destí) sense destruir aquesta, és a dir, que copia una cadena darrere de l'altra aquesta funció és coneguda com strcat.

- strlen: aquesta funció retorna el total (sencer) de caràcters que conformen una cadena (excloent el caràcter nul \ 0).
- strcmp: strcmp (abreviatura de ((string comparison))). La funció strcmp rep dues cadenes, a i b, retorna un enter.