Blocs d'un programa informàtic

1. Un programa informàtic (programa) és una seqüència d'accions (instruccions) que manipulen un conjunt d'objectes (dades).

2. Hi ha dues parts o blocs que componen un programa:

   Bloc de declaracions: en aquest es detallen tots els objectes que utilitza el programa (constants, variables, arxius, etc.).

   Bloc d'instruccions: conjunt d'accions o operacions que s'han de dur a terme per aconseguir els resultats esperats.

   El bloc d'instruccions està compost al seu torn per tres parts, encara que en ocasions no estan perfectament delimitades, i apareixeran barrejades en la seqüència de el programa, podem localitzar-les segons la seva funció. Aquestes són:

1. Entrada de dades: instruccions que s'emmagatzemen en la memòria interna dades procedents d'un dispositiu extern.
2. Procés o algoritme: instruccions que modifiquen els objectes d'entrada i, de vegades, creant-ne de nous.
3. Sortida de resultats: conjunt d'instruccions que prenen les dades finals de la memòria interna i els envien als dispositius externs.

Capçalera	És sol especificar: ● Nom del programa ● Dades d'entrada ● Dades de sortida
Funcions	definició de les funcions pròpies crades pel programador per a utilitzar-les en repetides ocasions.
Declaracions	Definicions i tipus de: ● variables ● constants ● nous tipus de dades
Assignacions	valors inicials dels identificadors declarats prèviament.
Entrades	instruccions per a guardar en la memòria els valors d'alguns identificadors.
Control	Instruccions de control del flux del programa. Poden ser: ● Alternatives ● Repetitives

Sortides	instruccions per a retornar els resultats obtinguts.

Tipus de dades compostes

- Que es una dada composta?

Una data composta o estructurada és aquella que permet emmagatzemar i manipular més d'un valor

- Tipus de dades compostes
  - ☐ Predefinits: Els tipus string o també anomenat "Cadena de càracters" es una secuencia de caracters que es tractada com una única data

exemple:
typedef char cadena[5]
cadena sr1 , sr2

  - ☐ Definits: hi ha dos tipus de dades definits i són el següents

- Struct: També anomenat estructura o registre es una agrupació d'elements de diferents tipus, cada camp es un representa mitjançant un caracter propi

exemple:

```
struct Nom
{
        Tipus:Nom_camp1
        Tipus:Nom_camp2
}
```

- Array: una array o vector serveix per a agrupar variables d'un mateix tipus

exemple:

        Tipus Nom [Numero d'elements]

(que és , STRING , STRUK , ARRAY)
https://plataforma.josedomingo.org/pledin/cursos/programacion/curso/u26/

Los datos compuestos son el tipo opuesto a los tipos de datos atómicos. Los datos compuestos se pueden romper en subcampos que tengan significado.

Un ejemplo sencillo es el número de su teléfono celular 56 2 99110101. Realmente, este número consta de varios campos, el código del
país (56, Chile), el código del área (2, Santiago) y el número propiamente dicho, que corresponde a un celular porque empieza con 9.

En algunas ocasiones los datos compuestos se conocen también como datos o tipos agregados. Los tipos agregados son tipos de datos cuyos valores constan de colecciones de elementos de datos. Un tipo agregado se compone de tipos de datos previamente definitivos.

Existen tres tipos agregados básicos:

Arreglos (arrays) y Matrices (tablas)
Un array (o arreglo) es una estructura de datos con elementos homogéneos, del mismo tipo, numérico o alfanumérico, reconocidos por un nombre en común. Para referirnos a cada elemento del array usaremos un índice (empezamos a contar por 0).

Registros
Un registro, en programación, es un tipo de dato estructurado formado por la unión de varios elementos bajo una misma estructura. Estos elementos pueden ser, o bien datos elementales (entero, real, carácter,...), o bien otras estructuras de datos. A cada uno de esos elementos se le llama campo.

Secuencias de texto o cadenas.

Estructures de selecció, repetició i salt

Les **estructures de selecció** permeten prendre decisions sobre quin conjunt d'instruccions cal executar en un punt del programa. O sigui, seleccionar quin codi s'executa en un moment determinat entre camins alternatius. Tota estructura de selecció es basa en l'avaluació d'una expressió que ha de donar un resultat booleà: true (cert) o false (fals). Aquesta expressió s'anomena la **condició lògica** de l'estructura.
En podem trobar de cuatre 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.

- **El commutador: la sentència "switch"**

És que no es basa a avaluar una condició lògica composta per una expressió booleana, sinó que estableix el flux de control a partir de l’avaluació d’una expressió de tipus enter o caràcter (però mai real).
La sentència switch enumera, un per un, un conjunt de valors discrets que es volen tractar, i assigna les instruccions que cal executar si l’expressió avalua en cada valor diferent. Finalment, especifica què cal fer si l’expressió no ha avaluat en cap dels valors enumerats.

Sobre **l’estructura de repetició** o bucle podem dir fa possible l'execució repetida d'una o més instruccions. Les estructures de repetició ens permeten executar diverses vegades unes mateixes línies de codi.

Aquestes estructures descriuen processos que es repeteixen diverses vegades en la solució de el problema. El conjunt d'accions que es repeteixen conformen el cos del bucle i cada execució s'anomena iteració.
Com a l’estructura de selecció, també en podem trobar de cuatre 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 sentència "do/while"**

La sentència do/while permet repetir l’execució del bucle mentre es verifiqui la condició lògica. A diferència de la sentència 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 sentència "for"**

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

Tractament de cadenes de text

Fins ara, el tipus de dada caràcter no ha estat gaire rellevant a l'hora d'exposar exemples significatius de codi font de programes. És força habitual que en el vostre dia a dia tracteu amb conjunts de dades numèriques individuals, cadascuna independent de les altres: dates, distàncies, intervals de temps, quantitats, etc.

Per tant, són dades que té sentit emmagatzemar i processar mitjançant un programa per automatitzar-ne el càlcul. En canvi, segurament no és gaire habitual que constantment useu conjunts de caràcters individuals i independents: lletres 'a', 'b', 'c', etc.

La veritable utilitat dels caràcters és poder usar-los agrupats en forma de paraules o frases: un nom, una pregunta, un missatge, etc. Per tant, si un llenguatge de programació ha de ser útil, s'ha de tenir en compte aquesta circumstància.

https://ioc.xtec.cat/materials/FP/Recursos/fp_asix_m03_/web/fp_asix_m03_htmlindex/WebContent/u3/a2/continguts.html