ALGORISMES I PROGRAMACIÓ (GESTIÓ AERONÀUTICA)
SESSIÓ 11/04/2005

1. Registres.

En C (i en la majoria de llenguatges), un registre és un nou tipus de dada format per una col·lecció de variables que s'agrupen sota un únic nom, proporcionant així una manera eficaç de mantenir informació relacionada. Les variables que composen aquests registres s'anomenen camps o membres de la estructura.

1.1 Definició de registres

Definir una estructura consisteix en dir el nombre i tipus de variables que la formaran i el nom del nou tipus de dades:

```
struct nom_estructura{
      tipus_1 variable1 ;
      tipus_2 variable2 ;
            .
            .
            .
      tipus_N vaiableN ;
};
```

La definició d'un registre forma una plantilla que s'utilitzarà per a crear noves variables de tipus `struct nom_estructura:`

Exemple> Per guardar la informació d'un CD de música ho podríem fer amb la següent estructura:

```
struct CD{
      char nom_grup[20];
      char nom_disc[50];
      char discografica[20];
};
```

Hem creat un nou tipus de dades, struct CD, que consta de 3 vectors de chars: nom_grup[20], nom_disc[50], discografica[20].

1.2 Declaració de registres

Fins ara, **NO** s'ha creat cap variable, tan sols hem definit la 'forma' que tindran les dades. Si ara volem declarar una variable que sigui de tipus struct CD, farem:

```
struct CD Disc_1;
```

Això crea una variable de tipus `struct CD` anomenada `Disc_1`.

També es poden declarar una o més variables d'estructura en el moment en què es declara el registre:

```
struct CD{
      char nom_grup[20];
      char nom_disc[50];
      char discografica[20];
} Disc_1, Disc_2, ··· , Disc_M;
```

1.3 Accés als membres d'una estructura

Als membres individuals del registre s'hi accedeix usant l'anomenat *operador punt* '**.**' I la sintaxi és:

```
nom_de_la_variable_registre.nom_del_camp
```

D'aquesta manera podem tractar cada camp com una variable normal. Així, per exemple, si volem assignar-li un valor al camp nom_grup de la variable `Disc_2`, farem:

```
strcpy(Disc_2.nom_grup,"Primus"); // incloure string.h
```

… i si volem mostrar el contingut de la variable per pantalla:

```
printf("%s", Disc_2.nom_grup);
```

o be,

```
puts(Disc_2.nom_grup);
```

Exercici #1> Feu un petit programa en el què definiu un registre tipus CD i una variable tipus struct CD i empleneu-ne els seus camps per pantalla. Un cop fet, mostreu, també per pantalla, el resultat.

Exercici #**2>** Afegiu al registre `CD`, un camp en què hi figuri l'any en què es va publicar el disc. Definiu dues variables tipus struct CD, ompliu tots els camps i mostreu el resultat per pantalla.

2. Arrays d'estructures

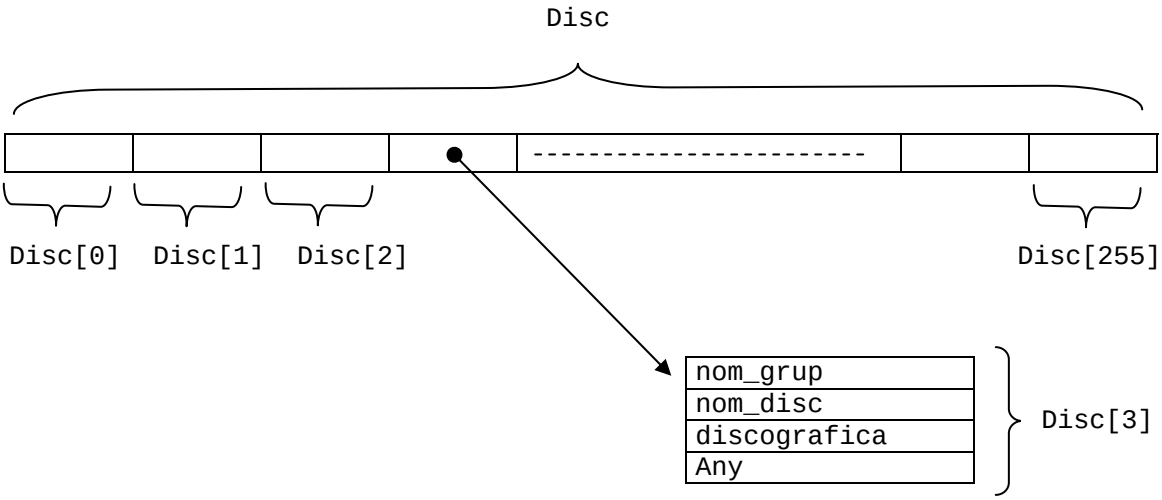
Ara que estem posats en l'exemple dels cd's, pensem en el següent. Fins ara, per a cada cd que volíem introduir, havíem de declarar una variable tipus `struct CD`. Així, si per exemple volíem definir 5 cd's, havíem de fer :

```
struct CD Disc_1, Disc_2, Disc_3, Disc_4, Disc_5 ;
```

Fins a 5 cd's encara es podria tolerar però... Què passa si algú té 256 cd's? Hem de declarar explícitament 256 variables? Hi ha una manera molt més còmode de fer-ho: els ARRAYS. Aleshores faríem:

```
struct CD Disc[256];
```

És a dir, hem definit un vector de 256 posicions de tipus `struct CD`. Fixeu-vos en la figura següent:



L'accés als camps de cadascun dels registres no varia:

Exemple>

```
struct CD Disc[256];

Disc[57].nom_grup="La Cabra Mecanica";
```

3. Definició de nous tipus de Dades: typedef

El llenguatge C permet de definir explícitament nous noms par als tipus de dades, usant la paraula clau `typedef`. Realment, **NO** creem un nou tipus de dades sinó que li donem un nom nou a un que ja existeix. Ho fem de la següent manera :

```
typedef tipus_existent nou_nom ;
```

Exemple> Anem a donar nous noms als tipus de dades coneguts fins ara:

```
typedef int TipusEnter;
typedef float TipusReal;
typedef char TipusCaracter;
```

En aquestes tres línies, hem dit que declarar una variable `int` o `TipusEnter` és el mateix. Igualment que `float` o `TipusReal` i `char` o `TipusCaracter`.

Pot semblar que això del typedef no és gaire útil però si que ho és: es pot utilitzar per tal de simplificar la declaració de variables tipus registre. Veiem un exemple que ho clarifiqui:

Exemple> Definició del tipus de dades NumComplex

```
struct complex{
      float part_real;
      float part_imaginaria;
};

struct complex NumComplex;
```

ens defineix una variable de tipus struct complex. Tot i que també podríem usar typedef i escriure:

```
typedef struct{
      float part_real;
      float part_imaginaria;
} complex;

complex NumComplex;
```

Així ens estalviem la paraula clau struct cada vegada que volem definir un variable de tipus complex.

L'ús de typedef fa que el codi esdevingui més fàcil i còmode.
RECORDEU→ typedef NO crea nous tipus de dades sinó que en renombra de ja existents.

Exercici #3> Useu el `typedef` per a renombrar les dades `struct CD`. Declareu un array de 5 CD's i ompliu les dades per pantalla. Mostreu-ne el resulta per comprovar que tot és correcte.

Exercici #4> Consulta discogràfica. Amplieu l'exercici anterior per fer un programa que permeti a l'usuari saber si teniu algun disc del grup que demana. Una vegada teniu les dades de tots els CD's de l'array, demaneu a l'usuari el nom de grup que vol buscar i mostreu per pantalla les dades de tots els discos del grup. Després pregunteu si vol tornar a fer una consulta i en cas afirmatiu repetiu la consulta (recordeu que els bucles serveixen per repetir accions dins d'un programa).