Tema 8: Estructures

Tema 8: Estructures

1. Concepte d'estructura

En C una estructura és una col·lecció de diferents tipus de variables bàsiques referenciades sota el mateix nom. Per exemple, un treballador d'una empresa pot estar caracteritzat pel seu nom, els cognoms, l' edat...

El concepte d'estructura és molt important ja que és l'antecedent directe del concepte de classe en la Programació Orientada a Objectes.

Les estructures són, moltes vegades, anomenades registres o, en anglès, *records*. De fet, les estructures són, en molts aspectes, similars als registres de les bases de dades. Seguint aquesta analogia, a les variables components de les estructures se solen anomenar camps o *fields*.

2. Definició d'una estructura

Per definir una estructura es fa servir la següent sintaxi:

```
struct [<nom de l'estructura>]
{
   [<tipus> <nom de variable>[,<nom de variable>,...]];
   ..........
} [<variable de estructura>[,<variable de estructura>,...]];
```

Sempre es comença amb la paraula reservada `struct` seguida d'un identificador per referir-se al nom de l'estructura. Aquest identificador és optatiu i, en alguns casos, no és necessari.

A continuació, i entre claus, es defineix l'estructura pròpiament dita. Aquesta consta d'una col·lecció de variables amb els seus tipus corresponents.

Després de la col·lecció de variables membres o camps, es pot declarar les variables d'estructura. Aquesta declaració es pot fer també separada de la definició de l'estructura. Veurem tot això amb un exemple:

```
struct treballador
{
   char nom[50];
   char cognom1[30];
   char cognom2[30];
   int edat;
} enginyer, administratiu, auxiliar;
```

D'aquesta forma s'ha definit l'estructura `treballador` i s'han declarat tres variables d'aquest *tipus*. Per cada una de les variables es reserva 50+30+30+4=114 bytes.

Aquest procés es pot seguir també per separat, en dues fases:

```
//fase de definició de l'estructura
struct treballador
{
   char nom[50];
   char cognom1[30];
   char cognom2[30];
   int edat;
```

```
};
...
...
...
//fase de declaració de variables d'estructura
struct treballador enginyer, administratiu, auxiliar;
```

En la fase de definició de l'estructura no es reserva memòria per cap variable, només es fa un patró o motlle per poder declarar més tard variables amb aquesta estructura.

Posteriorment, per declarar una variable amb aquesta estructura s'escriu el nom de l'estructura seguit dels noms de les variables.

En el estàndard ANSI C és necessari posar davant de l'identificador de l'estructura la paraula `struct`. Aquesta paraula es pot, optativament, ometre en C++.

Si s'omet el nom de l'estructura en la seva definició es diu que s'ha creat una estructura anònima i no es pot, posteriorment a la definició de l'estructura, declarar cap variable amb aquesta estructura de forma directa.

Si una estructura té identificador, aquesta es pot passar com argument a una funció de la mateixa forma que qualsevol altre tipus de variable.

És una pràctica habitual definir les estructures globals, és a dir, fora de qualsevol funció, inclòs la funció `main()`. D'aquesta forma, totes les funcions podran declarar variables amb aquesta estructura.

Els camps d'una estructura poden ser altres estructures. Per exemple, és correcta aquesta definició:

```
struct data
{
   int dia;
   int mes;
   int any;
};

struct persona
{
   char nom [20] ;
   char cognoms[40] ;
   int edat;
   struct data naixement;
};
```

3. Inicialització de variables estructures

Per posar un valor inicial a les variables de tipus estructura es fa de la mateixa forma que amb els vectors. Els diferents valors dels camps són declarats un darrere de l'altre, separats per comes i tancats per claus. Per exemple:

```
struct treballador enginyer = {"Joan", "López", "Puig", 35};
```

4. Referència a una estructura

Les variables d'estructura es poden referenciar completes o bé es pot accedir individualment als elements de l'interior. En aquest últim cas es fa servir l'operador punt (`.`).

Per exemple, per referir-nos al camp `nom` de la variable `enginyer` es fa servir l'expressió: `enginyer.nom`.

5. Assignació d'estructures

L'assignació d'estructures està permesa sempre que es faci entre variables del mateix tipus d'estructura. Per exemple, si definim dues variables del tipus treballador amb la sentència:

```
struct treballador enginyer1={"Joan","López","Puig", 35}, enginyer2;
```

una assignació com:

```
enginyer2=enginyer1;
```

equival a:

```
enginyer2.nom=enginyer1.nom;
enginyer2.cognom1=enginyer1.cognom1;
enginyer2.cognom2=enginyer1.cognom2;
enginyer2.edat=enginyer1.edat;
```

6. Vectors d'estructures

L'ús simultani de vectors i estructures proporciona una poderosa eina per a l'emmagatzematge i manipulació de dades.

Per exemple, si l'empresa disposa de 10 enginyers, podem declarar una variable indexada com:

```
struct treballador enginyer[10];
```

En aquest cas, per referir-nos al nom de l'enginyer 0 escriurem: `enginyer[0].nom`.

7. Pas d'estructures a funcions

Quan es passa una variable d'estructura a una funció, aquesta es passa per valor, és a dir, es passa una còpia de la informació de la variable que aquesta funció pot modificar sense modificar la variable original.

8. Punters a estructures

De la mateixa forma que es pot declarar una variable punter que apunti a una variable de tipus bàsic, també és possible declarar un punter que apunti a una variable d'estructura. I mitjançant els punters és com es passaran les estructures per referència a les funcions.

Per declarar una variable punter a una estructura se segueix la sintaxi normal de posar un `*` davant el nom de l'estructura. Per exemple:

```
struct treballador enginyer;
struct treballador *persona;
```

Per trobar l'adreça d'una variable d'estructura s'escriu l'operador `&` davant el nom de la variable d'estructura.

Podem assignar el punter persona a una variable d'estructura de la següent forma:

```
persona = &enginyer;
```

Per referir-nos al camp `nom` de la variable apuntada per `persona` es podria escriure: `(*persona).nom` (el parèntesi és necessari degut a la major prioritat de l'operador punt). Aquest és un sistema correcte però gairebé en desús. El segon mètode i el més utilitzat per referir-nos a un membre d'una variable d'estructura apuntada per un punter d'estructura és amb ajuda de l'operador (->). En el codi escrit a continuació, les dues últimes línies fan exactament el mateix, escriuen el nom de la variable `enginyer` on apunta el punter `p`:

```
struct treballador *p;
p=&enginyer;
printf("%s\n", (*p).nom);
printf("%s\n", p->nom);
```

9. Exemple

```
#include<stdio.h>
#include<conio.h>
#define DIM 20

int introduirdades();
void mostradades(int dim);

struct clase
{
  char nom[10];
  char cognoms[20];
  float notaC;
}asi1b[DIM];


void main()
{
  int dim;
  clrscr();
  dim=introduirdades();
  mostradades(dim);
}

int introduirdades()
{
      int dim,i;
      printf("\nQuantes notes ha d'introduir: ");
      scanf("%d",&dim);
   fflush(stdin);
      for(i=0;i<dim;i++)
      {
       printf("\nAlumne %d:\n",i+1);
       printf("Nom: ");
       gets(asi1b[i].nom);
       printf("Cognoms: ");
       gets(asi1b[i].cognoms);
       printf("Nota Programaci¢: ");
       scanf("%f",&asi1b[i].notaC);
       fflush(stdin);
      }
      return dim;
}
```

```
void mostradades(int dim)
{
  int i;
  clrscr();
  gotoxy(5,3);
  printf("***** Notes de Programaci¢ ASI 1B *****");
  gotoxy(1,5);
  printf("Alumne");
  gotoxy(30,5);
  printf("Nota");
  for(i=0;i<dim;i++)
  {
       gotoxy(1,6+i);
       printf("%s %s",asi1b[i].nom,asi1b[i].cognoms);
       gotoxy(30,6+i);
       printf("%g",asi1b[i].notaC);
  }
  getch();
}
```

Apèndix: Biblioteca de problemes

1. Construïu un programa que gestioni una agenda telefònica. Haurà de guardar el nom, cognom primer, cognom segon i telèfon dels contactes. Guardeu un vector amb totes les dades. Definiu el vector i la seva dimensió com a variables globals. El programa haurà de permetre:

   - Entrar un nou contacte
   - Esborrar un contacte
   - Mostrar tota l'agenda
   - Cerca per nom
   - Cerca per cognom 1
   - Cerca per telèfon

2. Realitzeu una base de dades de pel·lícules. Les dades a guardar són: títol, director, protagonista i durada. El programa ha de permetre entrar noves pel·lícules, esborrar-les, mostrar-les i fer cerques per títol, director i protagonista. També ha de permetre realitzar l'ordenació de les pel·lícules per títol.

3. Els nombres complexos són una aplicació comú de les estructures. En aquest conjunt, les operacions aritmètiques: suma, resta, producte i divisió es poden definir trivialment a partir de les operacions aritmètiques de nombres reals.

   Un nombre complex és un parell ordenat de nombres reals (a, b) (habitualment s'escriu $a+bi$) . a rep el nom de part real, i b el nom de part imaginària.

   Si tenim dos nombres complexos: $a+bi$ i $c+di$, la suma es defineix com un altre nombre complex, la part real del qual és la suma de les parts reals dels nombres complexos inicials i la seva part imaginària és la suma de les parts imaginàries dels nombres complexos inicials, és a dir:

   $$a+bi + c+di = (a+c) + (b+d)i$$

   El producte de nombres complexos és una mica més sofisticat. Es defineix de la següent forma:

   $$(a+bi) *(c+di) = (a*c-b*d) + (a*d+b*c)i$$

   La divisió és possible sempre que la part real i la part imaginària del divisor no siguin simultàniament 0. Si tenim dos nombres complexos $z1=a+bi$ i $z2=c+di$, es defineix la divisió com:

   $$\frac{z1}{z2} = \frac{ac+bd}{c^2+d^2} + \frac{bc-ad}{c^2+d^2}\,i$$

   Construïu un programa que permeti realitzar les operacions bàsiques (suma, resta, multiplicació i divisió) amb nombres complexos.

4. En un club d'atletisme es vol seleccionar, entre un grup de n atletes, aquells amb marca personal superior a la mitjana del grup. Dissenyeu un programa que permeti realitzar aquesta selecció. El programa haurà de donar un llistat amb el nom d'aquells atletes que la superin.

5. En la secretaria del centre es disposa de dos arrays. En un d'ells es guarda la següent informació per alumne:

   - número d'expedient
   - nom
   - domicili

   i en l'altre, les notes dels exàmens, de la forma:

   - assignatura
   - número d'expedient
   - nota

   Es desitja dissenyar un programa que permeti:

   a) obtenir una relació d'assignatures i notes per alumne
   b) obtenir una relació d'alumnes i notes per assignatura

6. Dissenyeu un programa per a ordenar un grup de *n* alumnes, per a cadascun dels quals es disposa de les següents dades:

   - número d'expedient
   - cognoms i nom
   - grup

   de forma que, opcionalment, es pugui obtenir una relació ordenada de la següent forma:

   a) ordenació per ordre alfabètic
   b) ordenació per número d'expedient
   c) ordenació per grup, i dins de cada grup per ordre alfabètic