Chapitre 2: Les structures

AÏCHA EL GOLLI
aicha.elgolli@essai.ucar.tn

Introduction

- Nous avons déjà vu les tableaux qui permettent de regrouper un ensemble de valeurs de même type, chacune d'entre elles étant repérée par un indice. Une structure en revanche nous permet de regrouper des valeurs de types différents au sein d'une même variable. Chaque élément de la structure (nommé ``champ") est repéré par un nom.
- Contrairement aux tableaux qui sont des structures de données dont tous les éléments sont de même type, les enregistrements sont des structures de données dont **les éléments peuvent être de type différent** et qui se rapportent à la **même entité** (au sens de Merise)
- Les éléments qui composent un enregistrement sont appelés **champs**. Avant de déclarer une variable enregistrement, il faut avoir au préalable définit son type, c'est à dire le nom et le type des champs qui le compose. Le type d'un enregistrement est appelé **type structuré**. (Les enregistrements sont parfois appelé structures, en analogie avec le langage C)

Syntaxe

Algorithmique	C
Type Structure *nom_type* *nom_champ1: type_champ1 ;* … *nom_champN: type_champN ;* **FinStruct**	**Struct** nom_type **{** type_champ1 nom-champ1 **;** **…** type_champN nom_champN **;** **} ;**

Syntaxe

Algorithmique

```
Type Structure etudiant
nom : chaîne
prénom : chaîne
moyenne_pratique : réel
moyenne_theorique : réel
age : entier
FinStruct
```

C

```
#define LGNOM 30

struct etudiant
{
  char nom[LGNOM] ;
  char prenom[LGNOM] ;
  float moyenne_pratique ;
  float moyenne_theorique ;
  int age ;
} ;
```

Cette déclaration définit la structure etudiant mais ne déclare pas de variable correspondant à cette structure. Une fois que la structure est définie on peut déclarer des variables du type correspondant

Syntaxe

Algorithmique	C
Var etud: etudiant	/* déclare une variable etud de type struct etudiant */ **struct etudiant etud ;**

Cette déclaration définit la structure etudiant mais ne déclare pas de variable correspondant à cette structure. Une fois que la structure est définie on peut déclarer des variables du type correspondant

Syntaxe

Un champ d'une structure est traité exactement de la même manière qu'une variable normale. La désignation du champ se note en faisant suivre le nom de la variable de type structure par un point suivi du nom du champ.

Attention : le nom d'un champ est TOUJOURS précédé du nom de l'enregistrement auquel il appartient. On ne peut pas trouver un nom de champ tout seul, sans indication de l'enregistrement.

Les champs d'un enregistrement, tout comme les éléments d'un tableau, sont des variables à qui on peut faire subir les mêmes opérations (affectation, saisie, affichage,…).

Exemple

```
Algorithme Exemple
Type
Structure etudiant
nom : chaîne
prénom : chaîne
moyenne_pratique : réel
moyenne_theorique : réel
age : entier
FinStruct
Variables
etud : etudiant
Début
etud.nom <- "Rousseau" ;
etud.prenom  <- "Jean-Jaques" ;
etud.moyenne_pratique <-  7,5 ;
etud.moyenne_theorique <-  6,8 ;
etud.age<- 21 ;
afficher ("Nom:    ", etud.nom);
afficher ("Age:    ", etud.age);
FIN
```

```
#include <stdio.h>
#define LGNOM 30
struct etudiant  {
char nom[LGNOM] ;
char prenom[LGNOM] ;
float moyenne_pratique ;
float moyenne_theorique ;
int age ;
} ;
void main(){
/* déclare une variable etud de type struct etudiant */
struct etudiant etud ;
strcpy (etud.nom, "Rousseau") ;
strcpy (etud.prenom, "Jean-Jaques") ;
etud.age = 21 ;
 etud.moyenne_pratique = 7.5 ;
etud.moyenne_theorique = 6.8 ;
printf ("Nom:   %s\n", etud.nom) ;
printf ("Age:   %d\n", etud.age) ;
}
```

Assignation de structures

Contrairement aux tableaux, les structures de même type peuvent être assignées entre elles.

Par exemple:

```
  struct etudiant et1, et2 ;

...

et1 = et2 ;  /* copie toute la structure et2 dans et1 */
```

Utilisation du mot clé typedef

Le mot clé **typedef** dans le langage C, permet de définir des types synonymes dans le but de rendre les programmes plus clairs.

Exemple

```
typedef int entier ;
```

signifie que entier est un synonyme de int, de sorte que les déclarations suivantes sont équivalentes:

```
int a, b ;        ⇔        entier a, b ;
```

Cette déclaration est plus qu'une simple substitution, par exemple avec:

```
typedef int vecteur [3] ;
```

les déclaration suivantes sont équivalentes:

```
int x[3], y[3] ;        ⇔    vecteur x, y ;
```

Utilisation du mot clé typedef

Dans le cas des structures notre structure étudiant peut être définit comme suit:

```
struct etudiant
{
  char nom[LGNOM] ;
  char prenom[LGNOM] ;
  float moyenne_pratique ;
  float moyenne_theorique ;
  int age ;
} ;

typedef struct etudiant s_etudiant ;
```

ou plus simplement

```
typedef struct
{
  char nom[LGNOM] ;
  char prenom[LGNOM] ;
  float moyenne_pratique ;
  float moyenne_theorique ;
  int age ;
} s_etudiant ;
...

/* déclare la variable etud de type s_etudiant */
s_etudiant etud ;
```

Avec la deuxième variante le type struct etudiant n'existe plus, on doit utiliser le type s etudiant, tandis que dans la première variante struct etudiant et s_etudiant sont équivalentes.

EXERCICE

On souhaite créer un programme d'annuaire très simplifié qui associe à un nom de personne un numéro de téléphone.

1. Créer une structure Personne pouvant contenir ces informations (nom et téléphone). Le nom peut contenir 32 caractères et le numéro 16 caractères.

2. Créer une nouvelle structure qui va représenter le carnet d'adresses. Cette structure Carnet contiendra un tableau de 200 Personne et un compteur indiquant le nombre de personnes dans le tableau.

3. Ecrire une procédure saisir() qui permet de saisir les informations d'une personne passée en argument.

4. Rajouter une fonction qui affiche les informations contenues dans la structure Personne passée en argument.

5. Faire un programme qui demande de saisir n personnes, qui les ajoute dans un carnet puis qui affiche son contenu.

```
Type structure personne
          nom:chaine;
          numero:chaîne;
Fin structure;
Type structure carnet
       T[200]: personne;
          nb:entier;
Fin structure ;
procedure saisir(d/r P: personne)
debut
      Afficher("le nom SVP ! ");
       Entrer(p.nom) ;
       afficher("le num SVP ! ");
       Entrer(P.numero) ;
Fin procedure
{Nous verrons lors de la prochaine
séance la traduction exacte en C des
param d/r En classe nous avons réalisé
un passage en d}
```

```
Procedure affiche(p:personne)
Debut
 afficher("le numéro de telephone de “, P.nom,” est “,
p.numero);
Fin;
Algorithme annuaire
     C1:  carnet;
          i,n: entier;
Debut
          c1.nb<-0;
        repeter
           afficher("combien de personnes allez vous saisir?");
           Entrer(n);
         jusqu’à (n>0 et n<200);
         pour(i de 0 a n-1) Faire
               saisir(C1.T[i]) ;
                 C1.nb<-C1.nb+1;
         pour(i de 0 a n-1) Faire
                  affiche(c1.T[i]);

Fin;
```