Chapitre 6

Les types personnalisés

Sommaire

Objectif

Parfois, aucun des types disponibles en langage de programmation ne permet de représenter nos données complexes ou composées. Dans ce cas-là, il faut définir de nouveaux types de données appelés types personnalisés. L'objectif de ce chapitre est de voir comment définir et utiliser ces types.

I. Introduction

Jusqu'à présent, nous n'avons vu que quelques types de base (entier, réel, caractère et logique). Ces types ont été utilisés pour décrire des structures composées de plusieurs éléments de même type, telles que les tableaux et les chaînes de caractères. Néanmoins, ils nous ne permettent pas de regrouper des informations de différent type liées au même objet dans une seule structure, par exemple: comment décrire un nombre complexe ? Ou bien les informations sur un étudiant ?

Par conséquent, dans le présent chapitre, nous allons approfondir nos connaissances dans deux directions, à savoir:

- La possibilité de définir de nouveaux types par l'utilisateur de façon plus agréable pour décrire ses objets.
- Présenter les instructions complémentaires et nécessaires pour faciliter le traitement de ces nouveaux types.

Déclaration des types

Afin de pouvoir utiliser un nouveau type dans un algorithme, il faut d'abord le déclarer dans la partie de déclaration. Cette partie commence par le mot réservé "**Type**" et comprend la déclaration de tous les types définis par l'utilisateur qui seront utilisés dans l'algorithme.

II. Enregistrements (Structures)

Même si la structure de tableau nous permet de décrire une structure formée de plusieurs éléments de même type, il nous ne permet pas de regrouper des informations liées au même élément et qui n'ont pas forcément le même type. Or il existe d'autres données qui sont composées d'éléments de types différents comme par exemple:

- Les dates (année, mois, jour)
- Les nombres complexes (partie réelle, partie imaginaire)
- Les fiches estudiantines ou personnelles(nom, prénom, date_naissance, adresse)

La nature différente de ces éléments a poussé les programmeurs à utiliser des structures appelées "Les enregistrements" qui sont mieux adaptées pour représenter ce type de données.

Définition 6.1 : Enregistrement

Un enregistrement (ou structure) est une structure de donnée formée d'un ou de plusieurs éléments (appelés champs) pouvant être de types différents. Á la différence du tableau, ces éléments ne sont pas ne sont pas indexés.

A. Déclaration d'un enregistrement

La définition du type enregistrement précise pour chaque élément un identificateur de champ, dont la portée est limitée à l'enregistrement, et le type de ce champ.

La déclaration des enregistrements se fait dans la partie de déclaration des types comme suit :

Syntaxe :

Algorithme	langage C

nom_de_type= enregistrement:
champ1 : type_champ_1 ;
champ2 : type_champ_2 ;
. . . . . .
champN : type_champ_N;
fin enregistrement
	struct Nom_Structure
	{
	type_champ_1 champ1 ;
	type_champ_2 champ2 ;
	.....
	type_champ_N champN ;
	};

Une fois qu'on a défini un type structuré, on peut déclarer des variables enregistrements exactement de la même façon que l'on déclare des variables d'un type primitif.

Exemple :

Algorithme	langage C

TYPE
Etudiant=**enregistrement**
nom : chaîne de caractères [30] ;
prenom : chaîne de caractères [25] ;
age : entier ;
moyenne : réel ;
fin enregistrement.
 	
VAR
e1 , e2 : Etudiant ;
	struct Etudiant {
	char nom [30];
	char prenom [25];
	int age;
	float moyenne ;
	};
	void main (){
	struct Etudiant e1 , e2;
	}

Question : En langage C, Faut-il obligatoirement écrire le mot-clé **struct** lors de la déclaration des variables de type enregistrement ?

Réponse : L'instruction appelée ***typedef*** permet de créer de nouveaux noms de types (autrement dit, elle sert à créer un alias pour l'enregistrement).

Exemple :

```
typedef float reel ;
typedef struct Etudiant Etudiant ;
struct Etudiant {
    char nom [30];
    char prenom [25];
    int age;
    reel moyenne ;
};
void main (){
     reel r;
     Etudiant e1 , e2;
}
```

- ✓ ***typedef*** : permet de créer un alias pour l'enregistrement;
- ✓ ***struct Etudiant :*** c'est le nom de l'enregistrement pour lequel on veut créer un alias (c'est-à-dire un « équivalent ») ;
- ✓ ***Etudiant :*** c'est le nom de l'équivalent.

B. Manipulation d'un enregistrement

Un enregistrement doit être manipulé champ par champ. On accède à ses champs en indiquant le nom de l'enregistrement suivi d'un point et du nom du champ.

Prenons l'exemple précédent de l'étudiant, les champs d'un enregistrement peuvent être manipulés par l'affectation, Ou bien par lecture et écriture :

Exemple affectation :

```
e1.nom  ← 'khelifa ' ;
e1.prenom ← 'Rayen' ;
e1.age ← 17 ;
e1.moyenne ← 18.75 ;
e2  ←  e1 ;
```

Exemple lecture et écriture :

Algorithme	langage C
Lire (e1. nom) Lire (e1. prenom) Ecrire (e1.nom) Lire (e1. age) Lire (e1. moyenne)	void main () { Etudiant e1 , e2; puts (" Donnez votre nom : "); scanf ("%s", &e1.nom); puts (" Donnez votre prenom : "); scanf ("%s", &e1. prenom); puts (" Donnez votre age : "); scanf ("%d", &e1.age); puts (" Donnez votre moyenne : "); scanf ("%f", &e1. moyenne); e2=e1; }

Remarque : Comme pour les tableaux, il n'est pas possible de manipuler un enregistrement globalement, sauf pour affecter un enregistrement à un autre de même type (Par exemple : e2=e1 ;). Pour afficher un enregistrement par exemple, il faut afficher tous ses champs un par un.

Initialiser une structure

L'initialisation d'un enregistrement ressemble un peu à celle d'un tableau. En effet, on peut initialiser un enregistrement au moment de sa déclaration :

Exemple :

```
void main (){
Etudiant e = {" khelifa ", " Rayen ", 17, 18.75} ;
}
```

C. Les structures imbriquées

Un champ dans un enregistrement peut lui-même être un enregistrement.

Exemple :

```
typedef struct Date Date ;
struct Date {
  int jour ;
  int mois ;
  int annee ;
};
typedef struct Etudiant Etudiant ;
struct Etudiant {
  char nom [30];
  int age;
  float moyenne ;
  Date date_naissance ;
};
```

III. Conclusion

Nous avons vu à travers ce chapitre comment utiliser le type enregistrement, qui sont des types de variables personnalisés qu'on peut créer et utiliser dans des programmes. C'est au programmeur de les définir, contrairement aux types de base. Ces types personnalisés permettent de combler l'insuffisance des types de données de base offerts par les langages de programmation et par conséquent représenter des données complexes et composées pour un développement avancé.