Chapitre 3

Les types personnalisés

Jusque ici, nous avions toujours utilisé les types prédéfinis (connus et fournis par le langage de programmation), ces types nous permet de manipuler des variables avec des valeurs atomiques (des entiers, des réels, des caractères etc...). Mais dans certaines situations, le programmeurs à besoin de plus que ces types primitifs. Dans ce chapitre, nous allons voir comme définir des types avec des valeurs spécifiques.

3.1 Enumérations

Une énumération (ou type énuméré) est un type dont le domaine de valeurs est défini par le programmeur. Elle permet de décrire un type simple homogène. Puisqu'une énumération est un type, elle figure avant la déclaration de variables.

Algorithme algo_enum

---

enumeration <id_enum> $\{val_1, val_2, ..., val_N\}$

Var: <Partie_déclarations> ;

Début

<Partie_actions> ;

Fin.

```
1 #include <stdio.h>
2 enum Nom {val1, val2, ..., valN}
```

Exemple

```
1 #include <stdio.h>
```

Algorithme algorithme_couleurs

enumeration couleur $\{bleu, vert, jaune, rouge\}$;

Var: x, y : couleur ;

Début
si(x=bleu) **alors** $y \longleftarrow rouge$;
finsi;
Fin.

```
2 enum couleur {bleu, vert, rouge};
3 main()
4 {
5 couleur x,y;
6 if(x==bleu)
7 y=rouge;
8 }
```

3.2 Les enregistrements (types personnalisés)

Avant ce chapitre, nous n'utilisions que des types définis par défaut sur les langages de programmation. Ici, nous présentons les enregistrements, ce sont des types définis par le programmeur, par opposition aux types par défaut. Les enregistrements sont utilisés pour décrire des informations complexes et hétérogènes (de types différents) . Ils seront particulièrement utiles par la suite pour travailler sur les fichiers séquentiels.

3.2.1 Déclaration d'un type enregistrement

Imaginons que l'on veuille décrire, de façon informatique, l'état-civil d'un individu. Il nous faudra collecter des informations aussi diverses que son nom, son prénom, son sexe, sa date de naissance, etc. Ces informations de nature et donc de type distinct peuvent être associées à des variables distinctes :

- Nom
- Prénom
- Date de naissance
- Adresse

Il est clair que la manipulation d'un individu au travers de toutes ces variables risque d'être fastidieuse. Pour manipuler plusieurs individus, il faudra un nombre de variables égal à cinq fois le nombre d'individus. De plus, toutes ces informations sont cohérentes entre elles puisqu'elles représentent un état-

civil d'une seule personne, il semble donc naturel de pouvoir les regrouper dans un objet unique manipulable en tant que tel.

La notion de composition d'objets (type personnalisé), que l'on retrouve sous une forme ou une autre dans la plupart de langages de programmation, va nous le permettre. un enregistrement (ou structure en C) permet de regrouper sous un même identificateur plusieurs données qui peuvent être complètement différentes. Ces données sont appelées les champs de l'enregistrement (ou attributs). La syntaxe d'un enregistrement est donnée comme suit :

Type
Nom_type = enregistrement
attribut_1
attribut_2
⋮
attribut_N
Fin.

En C :

```
struct nom_type {
attribut_1
attribut_2
.
.
.
attribut_N
} ;
```

En C, on peut rajouter avant *struct* le mot clé *typedef*, ça nous permet de définir un type avec un nom qu'on donne à la fin de la définition de l'enregistrement.

```
typedef struct nom_type {
attribut_1
attribut_2
.
.
.
attribut_N
} Nom_Type;
```

Exemple 6 *Le type enregistrement qui représentera l'état-civil précédent aura la forme suivante :*

Type EtatCivil = enregistrement
nom : chaîne de caractères
prénom : chaîne de caractères
adresse : chaîne de caractères
naissance : Entier
Fin.

En C :

```c
typedef struct  EtatCivil {
char * nom;
char * prenom;
char * adresse;
int naissance;
} EtatCivil;
```

3.2.2 Déclaration d'une variable de type personnalisé

Une fois qu'on déclare notre type composé (personnalisé), ça devient un type comme les types standards. Donc, pour déclarer une variable d'un type personnalisé il suffit de l'utiliser dans la déclarations des variables.

variable : Type_personnalisé ;

En C :

```c
type_personnalise identifiant;
```

Exemple 6 (suite) Pour définir un individu, il suffira de déclarer une variable de type EtatCivil :

individu : ÉtatCivil ;

En C :

```c
struct EtatCivil individu;
//Si on rajoute typedef a la declaration on enleve le mot cle
   struct avant chaque declaration
EtatCivil individu;
```

3.2.3 Accée aux attributs (champs)

Puisque les objets peuvent être de types différents, qu'il n'est pas possible d'utiliser une notation indicée comme avec les tableaux. On accède aux champs d'une enregistrement grâce à une notation pointée. On fait suivre le nom de la variable de type enregistrement d'un point suivi du nom du champ que l'on veut désigner.

---

variable.attribut_1;
variable.attribut_2;
⋮
variable.attribut_N;

En C :

```
1 variable.attribut_1;
2 variable.attribut_2;
3 .
4 .
5 .
6  variable.attribut_N;
```

Exemple 6 (suite)

Pour afficher les informations de l'individu, il suffira d'accéder à chaque l'attribut de variable individu :

Algorithme manipulation$_e$*nregistrement*

Procedure remplire_personne(S/personne : EtatCivil)
Début
Ecrire ("Donner le nom de la personne") ;
Ecrire(personne.nom) ;
Ecrire ("Donner le prenom de la personne") ;
Ecrire(personne.prenom) ;
Ecrire ("Donner l'adresse de la personne") ;
Ecrire(personne.adresse) ;
Ecrire ("Donner la date de naissance de la personne") ;
Ecrire(personne.naissance) ;
Fin.

Procedure afficher_personne(E/personne : EtatCivil)
Début
Ecrire ("le nom est :", personne.nom) ;
Ecrire ("le prénom est :", personne.prenom) ;
Ecrire ("l'adresse est :", personne.adresse) ;
Ecrire ("la date de naissance est :", personne.naissance) ;
Fin.

Procedure main()
Var : tab_personnes : tableau [1..3] EtatCivil ;
Début
Pour i **de** 1 **à** 3
remplire_personne(tab_personnes[i]) ; **fpour;**
Pour i **de** 1 **à** 3
afficher_personne(tab_personnes[i]) ; **fpour;**
Fin.

En C :

```c
//fonction qui va remplir les information d'une personne
void remplir_personne(EtatCivil & personne)
{
 printf("Donner le nom de la personne \n");
 scanf("%s",&personne.nom);
 printf("Donner le prenom de la personne \n");
 scanf("%s",&personne.prenom);
 printf("Donner l'adresse de la personne \n");
 scanf("%s",&personne.adresse);
```

```
10  printf("Donner la date de naissance de la personne \n");
11  printf("%s",&personne.naissance);
12 }
13 //fonction qui va afficher les information d'une personne
14 void afficher_personne(EtatCivil personne)
15 {
16  printf("le nom est: %s",personne.nom);
17  printf("le prenom est: %s",personne.prenom);
18  printf("l'adresse est: %s",personne.adresse);
19  printf("la date de naissance est: %s",personne.naissance);
20 }
21 //la fonction main va cr er un tableau de 3 personnes, par la
     suite ses informations seront remplis et affichees
22 void main()
23 {
24 EtatCivil tab_personnes[3];
25 for(int i=0; i<3;i++)
26 remplir_personne(tab_personnes[i]);
27 for(int i=0; i<3;i++)
28 afficher_personne(tab_personnes[i]);
29 }
```