Chapitre six

Les types personnalisés

Le langage C permet de créer de nouveaux types de variables autres que les types de base qu'on a déjà vu. Ce qu'on appelle par « **types de variables personnalisés** », nous allons en voir deux sortes : les **énumérations** et **les structures.**

Les énumérations :

Les énumérations constituent une façon un peu différente de créer ses propres types de variables. Une énumération est une liste de « valeurs possibles » pour une variable.

Une variable personnalisée est de type énuméré peut seulement prendre une des valeurs que vous prédéfinissez par une énumération.

Type énuméré en algorithmique :

```
Type
    nom-type = (val1, val2, val3, …, val n) ;
```

Exemple :

```
Type    semaine = (samedi, dimanche, lundi, mardi, mercredi, jeudi, vendredi) ;
```

Type énuméré en C :

```
enum  semaine {samedi, dimanche, lundi, mardi, mercredi, jeudi, vendredi
              }
```

Note : enum semaine constitue un nouveau type qui sera utilisé pour déclarer des variables.

Exemple :

```
enum  semaine jour ;
```

jour peut avoir une des valeurs de la liste définis pour enum semaine :

```
jour = dimanche ;
```

Le compilateur associe automatiquement un nombre à chacune des valeurs possibles de l'énumération. Dans l'exemple samedi correspond à 0 dimanche à 1 ainsi de suite…
Quant on désire associer d'autres valeurs on peut le faire comme dans l'exemple qui suit :

```
enum naturel { DIX = 10, ONZE, DOUZE, TREIZE, QUATORZE, QUINZE };
```

Directive typedef :

La principale utilité des typedef est de faciliter l'écriture des programmes, et d'en augmenter la lisibilité. Par exemple si on désire donner un nom pour un type on utilise typedef :

```
Typedef  int entier ;
```

Et on peut déclarer :

```
entier i,j ;
```

```
typedef    enum  semaine
                 {samedi, dimanche, lundi, mardi, mercredi, jeudi, vendredi
                  } semaine;
```

Et dans ce cas on aura une déclaration plus aisée :

semaine jour ;

les structures en C :

Les structures permettent au programmeur de définir une composition de sous-ensembles de données pour former un nouveau type, cette composition est appelée : structure.
Les composantes de cette structure sont appelées des champs, chacun de ces champs doit avoir un identificateur. C'est-à-dire une structure est un assemblage de variables qui peuvent avoir différents types.

Syntaxe

```
struct Nom_Structure
{
   int variable1;
   int variable2;
   double variable3;
};
```

```
struct personne
  { char nom[20] ;
    char prenom [20] ;
  }
```

Les déclarations de variables se feront par :

struct personne p1,p2;

Pour éviter le mot struct à la déclaration de variables on peut utiliser typedef comme suit

```
typedef struct
  {
  ...
  } PERSONNE;
```

Les déclarations de variables se feront par :

PERSONNE p1,p2;

Exemple2 :

```
    typedef   struct  Date {
                              int Jour ;
                              int Mois ;
                              int Année ;
                             } Date ;

    typedef   struct   Etudiant {
```

```
                    int Mat ;
                    char Nom  [15] ;
                    char Prenom  [15] ;
                    Date Date_Naissance  ;
                  } Etudiant ;
// déclaration et affectation
Main()
  { Etudiant  E  ;
     E .Mat =34561 ;
     strcpy(  E .Nom , ''benali'') ;
```

Exercice :
Ecrire un programme qui :

- Permet de définir un étudiant par son matricule, note1, note2, note3, note4, et sa moyenne.
- La saisie de données de 30 étudiants.
- Calculer la moyenne de chaque étudiant.
- Afficher les étudiants admis.