1. Enregistrements (Structures)

Il est souvent pratique de regrouper plusieurs variables en une seule variable composée. Cette technique peut être réalisée avec l'utilisation de structures ou d'enregistrements.

Retenir

- Un type enregistrement ou type structuré est un type T de variable v obtenu en juxtaposant plusieurs variables v1; v2; … ayant chacune un type T1;T2; …
- Les différentes variables vi sont appelées champs de v elles sont repérées par un identificateur de champ
- Si v est une variable de type structuré T possédant le champ ch, alors la variable v:ch est une variable comme les autres :(type, adresse, valeur)
- Les structures permettent de rassembler des valeurs de type différent. Par exemple, pour une adresse, on a besoin du numéro (int) et du nom de la rue (char).

1.1. Déclaration

En algorithmique

nom_de_type= structure:
{nom_du_champ_1 : type_du_champ_1
nom_du_champ_2 : type_du_champ_2
...
nom_du_champ_n : type_du_champ_n
}

En langage C

typedef struct nom_de_struct {
nom_du_champ_1 : type_du_champ_1;
nom_du_champ_2 : type_du_champ_2;
...
};
Déclaration de variable structurée :
struct nom_de_struct nom_type;
Ou avec une définition de type :

typedef struct nom_de_struct nom_type;

Chaque élément déclaré à l'intérieur de la structure est appelé un **champ**. Le nom donné à la structure est appelé **étiquette de structure**. Ici, on a en fait déclaré un type de structure, pas une variable.

On déclare les variables associées à une structure de cette manière :

struct adresse chez_Khaled , chez_Zahia ;

Car si la structure d'une adresse est toujours la même (numéro et nom de la rue), ***chez_Khaled*** et ***chez_Zahia,*** qui sont des ***struct adresse*** différentes, n'habitent pas au même endroit.

On peut initialiser une structure lors de sa déclaration :

struct adresse chez_Khaled={ 15 , "rue_Liberté" } ;

Exemple de déclaration de type structuré :

```
typedef  struct date {
char nom_jour [9]; // lundi, mardi, ..., dimanche
int num_jour; // 1, 2, ..., 31
int mois; // 1, 2, ..., 12
int annee;
};
date  = {"vendredi", 21, 10, 2011};
```

1.2. Manipulation

On accède aux données contenues dans les champs d'une structure et faisant suivre le nom de la strucure par un point "." et le nom du champ voulu :

chez_julie.numero=19 ;

strcpy(chez_julie.rue,"avenue Pasteur") ;

Si 2 structures ont le même type, on peut effectuer :

chez_pierre=chez_julie ; /* Pierre a emménagé chez Julie ! */

Mais on ne peut pas comparer 2 structures (avec == ou !=).

Rem :

On suppose déclarées des variables v;w d'un type structuré.

On dispose donc de variables v.nom_du_champ_ide type type_du_champ_i Toute opération valide sur une variable de type type_du_champ_iest valide sur v.nom_du_champ_i.

De plus, l'affectation v = w est valide. Elle est équivalente auxaffectations :

v.nom_du_champ_1 = w.nom_du_champ_1

v.nom_du_champ_2 = w.nom_du_champ_2

…….

2. Tableau de structure

On déclare un tableau de structure de la même façon qu'un tableau de variables simples : le nombre d'éléments est précisé entre crochets.

struct adresse pers[100] ;

Cette déclaration nécessite que la structure **adresse** ait déjà été déclarée avant. **pers** est alors un tableau dont chaque élément est une structure de type **adresse**. Et **pers[i].rue** fait référence au champ "rue" de la i_{eme} personne de la structure **pers**.

Exemple:

```
#include <stdio.h>
typedef  struct  point
{
double abs;
double ord;
};
int main()
{
point p[10];
int i;
p[0].ord = 0;
p[0].abs = 1;
for(i = 1 ; i < 10 ; i++)
{ p[i].ord = p[i - 1].ord + 1.;
p[i].abs = p[i - 1].abs + 2.; }
for(i = 0 ; i < 10 ; i++)
{ printf("p[%d] = (%f, %f)\n", i, p[i].abs, p[i].ord); }
}
```

3. Structure de structure

On peut utiliser une structure comme champ d'une autre structure. Dans la lignée des exemples précédents, on peut définir une structure **adresse** qui pourra être utilisée dans la structure **repertoire**. Elle peut également être utilisée dans une autre structure.

```
typedef struct adresse
{
```

```
int numero ;
char rue[50] ;
} ;
struct repertoire
 {
char nom[20] ;
char prenom[20] ;
struct adresse maison ;
 } ; /* déclaration d'un champ structure de type 'adresse' appelé 'maison' */
struct repertoire monrepertoire[100] ;
strcpy (monrepertoire[0].nom,"Ahmed") ;
strcpy (monrepertoire[0].prenom,"Ali") ;
monrepertoire[0].maison.numero = 19 ;
strcpy (monrepertoire[0].maison.rue,"avenue_Didouche") ;
strcpy (monrepertoire[1].nom,"Fouad") ;
strcpy(monrepertoire[1].prenom,"Mustapha") ;
monrepertoire[1].maison.numero = 15 ;
strcpy (monrepertoire[1].maison.rue,"rue_soleil") ;
```

Lorsqu'un tableau fait partie des champs d'une structure, on peut accéder aux valeurs de ce tableau par :

```
char initiale ;
initiale = monrepertoire[1].prenom[0] ; /* initiale de Ahmed */
```