Chapitre 2: Les structures

Les Structures

➔Une **structure** est une collection de plusieurs variables (**champs**) groupées dans un ensemble pour un traitement commode,

➔Les variables d'une structure sont appelées ***membres*** et peuvent être de n'importe quel type (tableaux, pointeurs, entiers ….)

```
struct Membre
{
    char      nom[80];
    char      adresse[200];
    int       *numero;
    float     amende[10];
};
```

Les Structures

Déclaration de Structures

➔ Déclaration de la structure

```
struct produit
{
        int code;
        int qte ;
        float prix ;
} ;
```

➔ Déclaration des variables de la structure

```
struct produit prd1 ;
struct produit prd1, prd2 ;
```

METHODE 1

➔ Déclaration de structure et variables de la même structure :

```
struct produit
{
        int code;
        int qte ;
        float prix ;
} prd1,prd2 ;
```

METHODE 2

Les Structures

Déclaration de Structures

➔ **Déclaration de la structure**

```
typedef struct
        {
                int code ;
                int qte ;
                float prix ;
        } Produit ;
Produit prd1, prd2 ;
```

METHODE 3

Les Structures

Utilisation des champs d'une structure

```
struct produit
{
        int code;
        int qte ;
        float prix ;
} ;
struct produit prd1, prd2 ;
```

prd1.code = 2015 ;

affecte la valeur 2015 au champ code de la structure prd1.

printf ("%f",prd1.prix) ;

affiche la valeur du champ prix de la structure prd1.

scanf ("%f",&prd2.prix) ;

Lit une valeur qui sera affectée au champ prix de la structure prd2.

➔*Notez bien la présence de l'opérateur &.*

Les Structures

Utilisation des champs d'une structure

```
struct produit
{
        int code;
        int qte ;
        float prix ;
} ;
struct produit prd1, prd2 ;
```

prd1=prd2;

➔**Possible pour le cas de structures de même type.**

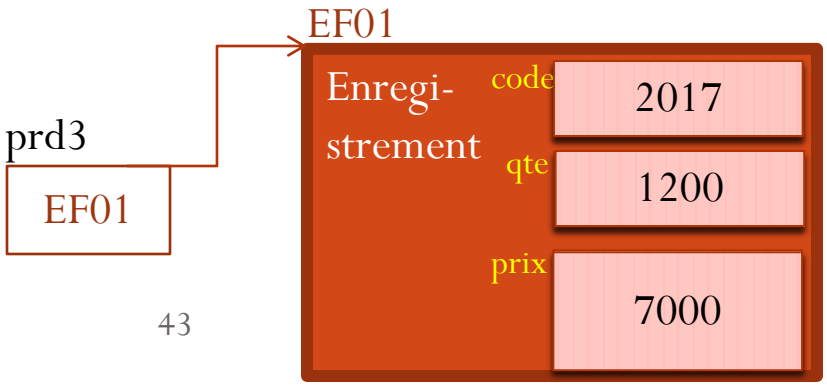
➔**Ça vient a remplacer :**

```
prd1.code  =  prd2.code;
prd1.qte   =   prd2.qte ;
prd1.prix  =   prd2.prix ;
```

Les Structures

Structure et Pointeur

```
typedef struct
        {
                int code ;
                int qte ;
                float prix ;
        } Enregistrement ;
Enregistrement *prd3 ;
```



➔**L'accès aux membres de la structure pointée par prd3 se fait de la manière suivante:**

```
prd3→code=2017;
prd3→qte=1200;
```

➔**L'affichage :**

```
printf("%d \n",Prd3→qte);
```

```
1200
7000
```

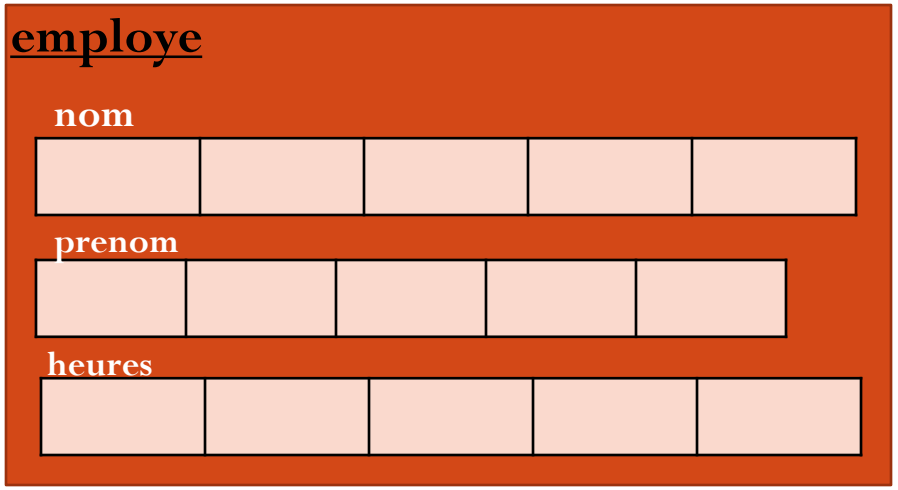
➔**La Lecture :**

```
scanf("%f",&Prd3→prix);
```

Les Structures

- **Structures contenant des tableaux:**

```
struct personne
        {
         char nom[30] ;
         char prenom [20] ;
         double heures [31] ;
        } employe;
```



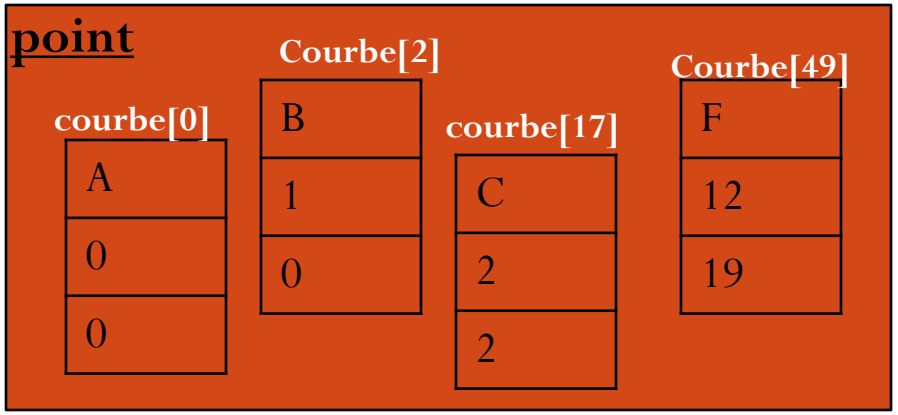
Réserve les emplacements pour une structure nommée **employe** .Ces derniers comportent trois champs:

- **nom** qui est un **tableau** de **30 caractères**,
- **prenom** qui est un **tableau** de **20 caractères**,
- **heures** qui est un **tableau** de **31 flottants**.

Les Structures

- **Tableaux de Structures:**

```
struct point {
                    char nom ;
                    int x ;
                    int y ;
                };
struct point courbe[50];
```



➔La structure point pourrait, par exemple, servir à représenter un point d'un plan, point qui serait défini par son nom (caractère) et ses deux coordonnées.

➔Le tableau courbe, pourrait servir à représenter un ensemble de 50 points du type ainsi défini.

Les Structures

- **Structures imbriquées:**

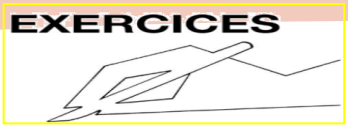
```
struct Date
{
    int    jour;
    int    mois;
    int    an;
};
```

```
struct Membre
{
    char            nom[80];
    char            adresse[200];
    int             numero;
    float           amende[10];
    struct  Date    emprunt;
    struct  Date    creation;
};
```

```
struct Livre
{
    char    titre[80];
    char    auteur[80];
    float   prix;
};
```

```
struct Pret
{
    struct  Livre   b;
    struct  Date    due;
    struct  Membre  *who;
};
```

Les Structures

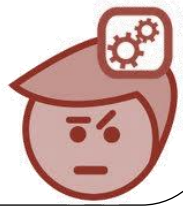


Exercice 1 :

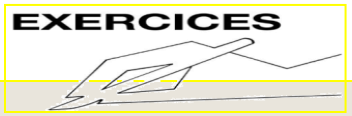
Ecrire un programme qui lit au clavier des informations dans un tableau de structures du type point défini comme suit:

```
typedef struct {
        char nom;
        double x ;
        double y;
            } point ;
```

➔Le nombre d'éléments du tableau est une constante.



Les Structures

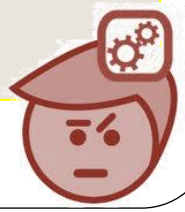


Exercice 2 :Soit la structure suivante :

```
typedef struct {
        char Matricule [10];
        char nom [50];
        char prenom [50];
        int Nb_heures_sup;
        float salaire_fixe ;
        float salaire ;
        float prime ;
    }  employe;
```

Ecrire un programme permettant de déclarer, de remplir et d'afficher un tableau de N éléments de type employé.

*NB : salaire=salaire_fixe+prime avec prime =Nb_heures_sup*10 ;*



Les Structures

Transmission d'une structure en argument d'une fonction

➔Transmission par Valeur

```
#include <stdio.h>
struct produit {
        int code ;
        float prix ;
} ;
main()
{
struct produit prd1 ;
void fct (struct produit p) ;
prd1.code = 1055; prd1.prix = 12.5;
printf ("\navant appel fct : %d %f",prd1.code,prd1.prix);
fct (prd1) ;
printf ("\nau retour dans main : %d %f", prd1.code,prd1.prix);
}
```

```
void fct (struct prdoduit prd)
{
prd.code = 0; prd.prix=1;
printf ("\ndans fct : %d %f", prd.code, prd.prix);
}
```

Résultat:

```
avant appel fct : 1055 12,5
dans fct : 0 1
au retour dans main : 1055 12,5
```

Les Structures

Transmission d'une structure en argument d'une fonction
➔Transmission par Adresse

```
#include <stdio.h>
struct produit {
        int code ;
        float prix ;
} ;
main()
{
struct produit prd2 ;
void fct (struct produit *) ;
prd2.code = 1055; prd2.prix = 12.5;
printf ("\navant appel fct : %d %f",prd2.code,prd2.prix);
fct (&prd2) ;
printf ("\nau retour dans main : %d %f", prd2.code,prd2.prix);
}
```

```
void fct (struct prdoduit *prd)
{
prd➔code = 0; prd➔prix=1;
printf ("\ndans fct : %d %f", prd➔code, prd➔prix);
}
```

Résultat:

```
avant appel fct : 1055 12,5
dans fct : 0 1
au retour dans main : 0 1
```

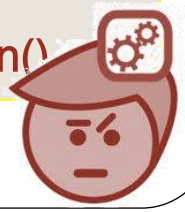
Les Structures

Exercice 3 :Soit la structure suivante :

```
typedef struct {
        char *nom;
        char *prenom ;
        long CNE ;
        char  filiere [4];    /* GI ou  GSTR*/
        float CC1;
        float CC2;
        float  Note_projet;
         float Note_module;
        char decision[3];      /* V: Valdé , NV: Non Validé ,R: Rattrapage
                } CYCLE_INGENIEUR ;
```

Ecrire un programme permettant de déclarer, de remplir et d'afficher un tableau de N étudiant .

N.B: Le tableau des étudiants doit être déclaré à l'intérieur de la fonction main()



Chapitre3: Les Fichiers

Les Fichiers

- **Définition**

Un **fichier** est un ensemble **d'informations stocké** sur une mémoire de masse (disque dur, disquette, bande magnétique, CD-ROM).

Types de Fichiers

Fichier Binaire: contient des données non textuelles. Ils ne prennent sens que s'ils sont traités par un programme adapté.

Exemples: code exécutable d'1 prog., fichiers son, vidéo, etc.

Fichier Texte: est formé de caractères ASCII, organisés en lignes, chacune terminée par un caractère de contrôle de fin de lignes.

Les fichiers textes peuvent être édités avec des éditeurs de texte et affichés de manière lisible à l'écran.

Les Fichiers

Types d'accès :

Séquentiel : le fichier est parcouru systématiquement depuis le début jusqu'à l'élément recherché

Direct : la position de l'élément recherché est fournie

Les Fichiers

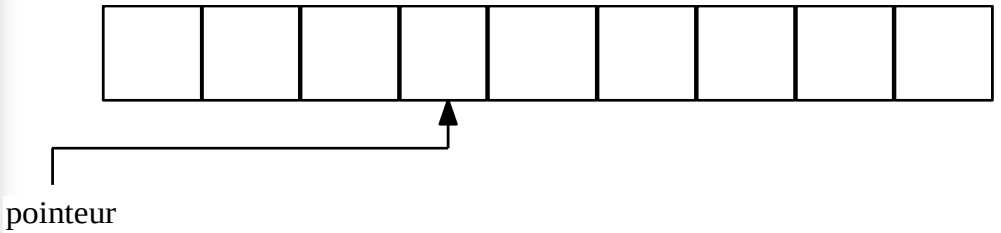
- **Opérations et Déclaration**

Opérations de Base (Librairie STDLIB.H)

- Créer
- Ouvrir
- Fermer
- Lire
- Ecrire
- Détruire
- Renommer

➔ *Déclaration:* **FILE *fichier;**

➔On définit un **pointeur** qui fournit l'adresse d'une cellule donnée.



NB: La déclaration des fichiers doit figurer AVANT la déclaration des autres variables.

Les Fichiers

- **Ouverture et fermeture de fichiers:**

➔**L'ouverture** se fait à l'aide de la fonction **fopen:**

FILE* f;

f=fopen(const char* name, const char* mode);

➔**La Fermeture** se fait à l'aide de la fonction **fclose :**

int fclose (FILE *f);

➔ **la fonction int feof(**FILE *fichier) retourne 0 tant que la fin du fichier n'est pas atteinte.

Exemple:

```
FILE *fichier ;
fichier = fopen("c :\listes.txt ", "w" ) ;
/* instructions et traitements*/
fclose(fichier) ;
```

Les Fichiers

- **Modes d'ouverture d'un fichier Texte:**

FILE *fopen(char *nom, char *mode);

mode :

« **r** » ouverture d'un fichier en **lecture** : le fichier doit exister, autrement la fonction fopen return NULL ;

« **w** » création et ouverture d'un fichier en **écriture** : si le fichier existe, son contenu est détruit ;

« **a** » ouverture d'un fichier en **écriture à la fin du fichier** : si le fichier n'existe pas, il est créé ;

« **r+** » ouverture d'un fichier **en lecture et écriture** : le fichier doit exister, autrement la fonction fopen return NULL ;

«**w+** » **création et ouverture** d'un fichier **en lecture et écriture** : si le fichier existe, son contenu est détruit ;

« **a+** » **ouverture d'un fichier en lecture et en écriture à la fin** du fichier : si le fichier n'existe pas, il est créé.

Les Fichiers

Ces modes d'accès ont pour particularités :

- Si le mode contient **la lettre r**, le fichier **doit exister**.
- Si le mode contient la **lettre w**, le fichier **peut ne pas exister**. Dans ce cas, il sera créé. Si le fichier existe déjà, **son ancien contenu sera perdu.**
- Si le mode contient **la lettre a**, le fichier **peut ne pas exister**. Dans ce cas, il sera créé. Si le fichier existe déjà, **les nouvelles données seront ajoutées à la fin du fichier précédent**.

Les Fichiers

- **Lecture et Ecriture dans les fichiers:**

Fonctions de lecture

```
int     fscanf(FILE* stream, const char* format, …);
int     fgetc(FILE* stream);
char*   fgets(char* buffer, int size, FILE* stream);
```

Fonctions d'écriture

```
int     fprintf(FILE* stream, const char* format, …);
int     fputc(int ch, FILE* stream);
int     fputs(const char* buffer, FILE* stream);
```

Les Fichiers

- **Exemple 1: Lecture / Ecriture :**

```
#include <stdio.h>
void main(void)
{
char titre[81];
float x[10];
int ind[10],  i=0,n=10;
FILE *f;
      f = fopen("monfichier.txt","w");
      if (f !=NULL){
         fprintf(f,"%s\n",titre);
         for (i=0; i < n; i++ ) {
           fprintf(f,"%f %d\n", x[i],ind[i]);
         }
       }
     fclose(f);
}
```

Ecriture

```
#include <stdio.h>
void main(void)
{
char titre[81];
float x[10];
int ind[10],  i=0;
FILE *f;
   f = fopen("monfichier.txt","r");
  if (f!= NULL) {
     fgets(titre,80,f);
     while(!feof(f)) {
        fscanf(f,"%f %d",&x[i],&ind[i]);
         i++;
     }
  }
 fclose(f);
}
```

Lecture

Les Fichiers

Exemple 2: lecture caractère par caractère

```
FILE *f;
      f = fopen(“monfichier.txt”,”r”);
do
    {
          caractereActuel = fgetc(f); // On lit le caractère
          printf("%c", caractereActuel); // On l'affiche
    } while (caractereActuel != EOF);
fclose(f);
```

Les Fichiers

Exemple 3: lecture d'une chaîne de caractères

```
FILE *f;
      f = fopen("monfichier.txt","r");
do
    {
  fgets(chaine, TAILLE_MAX, f);
    // On lit maximum TAILLE_MAX caractères du fichier, on stocke le tout dans "chaine"
    printf("%s", chaine); // On affiche la chaîne
    } while (!feof(f));
fclose(f);
```

Les Fichiers

Supprimer un fichier :

Pour supprimer un fichier, on utilise la fonction suivante :

```
int remove(const char* fichierASupprimer);
```

Exemple:

```
int main()
 {
        remove("test.txt");
        return 0;
 }
```

Les Fichiers

Renommer un fichier :

Pour renommer un fichier, on utilise la fonction suivante :

```
int rename(char* ancienNom, char* nouveauNom);
```

Exemple:

```
int main()
 {
        rename("test.txt", "test_renomme.txt");
        return 0;
 }
```

Les Fichiers

Exercice 1

Ecrire un programme qui permet de:

- Créer un fichier texte dont le nom est choisi par l'utilisateur
- remplir le fichier par une liste des étudiants(CNE, Nom, Prénom) : le nombre des enregistrements est déterminé par l'utilisateur(*Enregistrement par ligne)*
- Afficher son contenu

Exercice 2

1) Soit le fichier texte suivant, écrire un programme en c affichant le nombre de mots commençant par une majuscule.

Semestre 2 Module Programmation Avancée en C

Contrôle continu Numéro: 1

fichiers et Listes simplement chaînées

2) Comment récupérer directement le dernier mot « chaînnées » de ce fichier .

Les Fichiers

Exercice 3

Ecrire un programme qui permet de supprimer la 5ème ligne et la 8ème ligne d'un fichier texte. Chaque ligne comporte un enregistrement de type étudiant.

Les Fichiers Binaires

- **Modes d'ouverture d'un fichier binaire:**

FILE *fopen(char *nom, char *mode);

mode :

« **rb** » ouverture d'un fichier en **lecture** : le fichier doit exister, autrement la fonction fopen return NULL ;

« **wb** » création et ouverture d'un fichier en **écriture** : si le fichier existe, son contenu est détruit ;

« **ab** » ouverture d'un fichier en **écriture à la fin du fichier** : si le fichier n'existe pas, il est créé ;

« **rb+** » ouverture d'un fichier **en lecture et écriture** : le fichier doit exister, autrement la fonction fopen return NULL ;

«**wb+** » **création et ouverture** d'un fichier **en lecture et écriture** : si le fichier existe, son contenu est détruit ;

« **ab+** » **ouverture d'un fichier en lecture et en écriture à la fin** du fichier : si le fichier n'existe pas, il est créé.

Les Fichiers Binaires

Fonction de lecture des fichiers binaires

fread(void *pointeur, size_t taille, size_t nombre, FILE *flot);

- *pointeur*: est l'adresse du début des données à transférer,
- *taille:* la taille des objets à transférer,
- *nombre:* leur nombre.

Fonction d'écriture dans un fichier binaire

fwrite (void *pointeur, size_t taille, size_t nombre, FILE *flot);

- *pointeur*: est l'adresse du début des données à transférer,
- *taille:* la taille des objets à transférer,
- *nombre:* leur nombre.

Les Fichiers Binaires

Exemples:

Ecrire dans un fichier binaire :

```
int main(void) {
FILE *f_in, *f_out;
char F_SORTIE[]=''c:\nomfich.bin";
int tab1[50], tab2[50]; int i;
for (i = 0 ; i < NB; i++) tab1[i] = i;
if ((f_out = fopen(F_SORTIE, "wb"))
== NULL) { printf("Impossible
d'écrire dans le fichier");
return(-1);}
fwrite(tab1, 50 * sizeof(int), 1,
f_out);
fclose(f_out);
```

Lire à partir d'un fichier binaire :

```
if ((f_in = fopen(F_SORTIE, "rb"))
== NULL) { printf("Impossible de
lire dans le fichier "); return(-1); }
fread(tab2, 50 * sizeof(int), 1,
f_in);
fclose(f_in);
for (i = 0 ; i < 50; i++)
printf("%d\t",tab2[i]);
return(0);
}
```

Les Fichiers Binaires

Positionnement dans un fichier:

Il est possible d'accéder à un fichier en *mode direct*, c'est-à-dire que l'on peut se positionner à n'importe quel endroit du fichier. La fonction **fseek** permet de se positionner à un endroit précis:

int fseek(FILE *flot, long deplacement, int origine);

La variable *origine* peut prendre trois valeurs :

0 : début du fichier ;

1 : position courante ;

2 : fin du fichier.

Les Fichiers Binaires

Positionnement dans un fichier:

La fonction

int rewind(FILE *flot);

permet de se positionner au début du fichier.
Elle est équivalente à : **fseek(*flot*, 0, 0);**

La fonction

long ftell(FILE *flot);

retourne la position courante dans le fichier (***en nombre d'octets depuis l'origine***).

Les Fichiers Binaires

Exemple:

```
/* on se positionne a la fin du fichier */
fseek(f_in, 0, 2); printf("\n position %ld", ftell(f_in));
/* deplacement de 10 int en arriere */
fseek(f_in, -10 * sizeof(int), 2);
printf("\n position %ld", ftell(f_in));
 fread(&i, sizeof(int), 1, f_in); printf("\t i = %d", i);
/* retour au début du fichier */
rewind(f_in); printf("\n position %ld", ftell(f_in));
fread(&i, sizeof(int), 1, f_in); printf("\t i = %d", i);
/* deplacement de 5 int en avant */
fseek(f_in, 5 * sizeof(int), 1);
printf("\n position %ld", ftell(f_in));
fread(&i, sizeof(i), 1, f_in); printf("\t i = %d\n", i);
```

Les Fichiers Binaires

Exercice 1:

Développer un programme en C permettant de faire la gestion des *courriers électroniques,* chaque courrier
est identifié par: l'adresse de son expéditeur, son sujet, sa date d'envoi, son contenu et son état de lecture (Par défaut l'état des messages non lus prend la valeur 0).

1) Donnez la déclaration de(s) (la) structure(s) nécessaire(s) pour gérer ces données.
2) Ecrivez une fonction permettant d'ajouter un courrier électronique au fichier binaire ***Mail.bin*** sans écraser son contenu **.**
3) Ecrivez une fonction permettant de numéroter les enregistrements du fichier **Mail.bin.**

Exercice 1(suite):

4) Ecrivez une fonction permettant de copier les enregistrements impaires du fichier **Mail.bin** dans le fichier **Mail_Impairs.bin** et ceux paires dans le fichier **Mail_Pairs.bin.**

5) Ecrivez une fonction ***Chercher_Mail_Exped(char *nom_fich, char *adr_exp)*** permettant de rechercher un mail dans le fichier **Mail.bin** à base de l'adresse de son expéditeur**.** La fonction retournera le numéro de de l'enregistrement de la première occurrence **.**

7) Ecrivez une fonction ***MenuPrincipal()*** permettant d'afficher à l'utilisateur la liste des actions à faire.

8) Ecrivez une fonction ***main()*** permettant de faire appel aux fonctions développées**.**

Difficulté

- **Pb et difficultés**

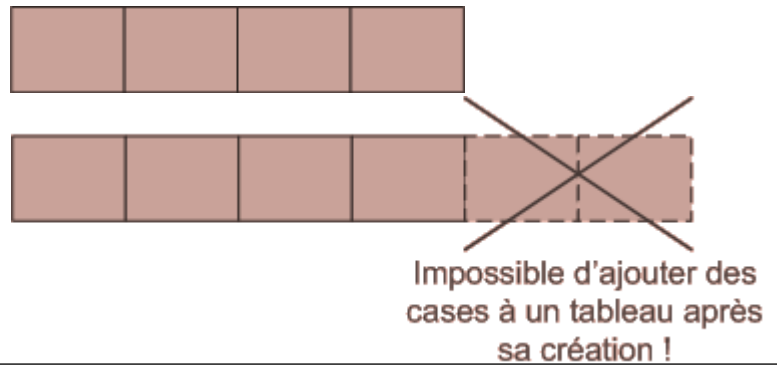
➔Stocker des données en mémoire, nous avons utilisé des variables simples :

- Type int, double. . .,
- Des tableaux,
- Des structures personnalisées. Si vous souhaitez

➔Stocker une série de données, le plus simple est en général d'utiliser des tableaux.

➔Limitation: (Exemple)

Int tab[4]



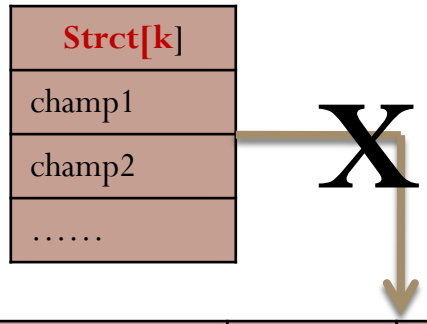
Difficulté

- **Pb et difficultés**

➔Lors de la manipulation de nombre variable d'instances d'une structure, et on souhaite insérer, supprimer dynamiquement;

➔*Les Tableaux de structures ne suffisent plus*

```
Struct strct{
    champ1;
    Champ2;
};
struct strct strct[n];
```



Strct[1]	Strct[2]	Strct[3]	Strct[4]	……	……	Strct[n]	Strct[n+1]
champ1	champ1	champ1	champ1	……	……	champ1	champ1
champ2	champ2	champ2	champ2	……	……	champ2	champ2
……	……	……	……	……	……	……	……