Cours: Algorithmes et Structures des Données

Chapitre 9: Les Enregistrements & les Pointeurs

Réalisé par:
Dr. Sakka Rouis Taoufik

1

Chapitre 8 : Les enregistrements & les pointeurs

I. Definition & Declaration

Un pointeur est une variable ou constante dont la valeur est une adresse mémoire d'un objet. L'adresse d'un objet est indispensable de son type. Pour les types de base, on pourra se définir des pointeurs de caractères, des pointeurs d'entiers et des pointeurs de réels.

Un pointeur se déclare à l'aide du type de l'objet pointé précédé de l'opérateur d'indirection (symbole*).

Syntaxe :

Type_Objet_Pointée *Nom_Pointeur ;

Exemples :

```
char *pc ; /*pc est un pointeur pointant sur un objet de type char */
int *pi ; /*pi est un pointeur pointant sur un objet de type int */
float *pf ; /*pf est un pointeur pointant sur un objet de type float */
```

2

II. Affectation de l'adresse d'un objet à un pointeur

L'affectation de l'adresse d'un objet à un pointeur se fait à travers l'opérateur d'adresse (&) selon la syntaxe : **Pointeur=&objet.**

Exemples :

```
int i=5, *pi ;
pi=&i ;
```

Remarque:

On ne peut pas affecter directement une valeur à un pointeur. Ainsi, l'écriture suivante est interdite :

```
int  *pi ;
pi=0xfffe ;
```

Par contre, on peut définir la valeur d'une adresse en utilisant l'opérateur de « cast ».

Exemple :

```
int *pi ;
pi= (int*) 0xfffe ; /* p est l'adresse 0xfffe et pointe sur un entier*/
```


III. Affectation du contenue d'une variable à travers un pointeur

A. Cas d'une seule variable simple

L'affectation du contenu d'un objet à un pointeur se fait en utilisant l'operateur d'indirection (*) selon la syntaxe :

```
*pointeur= valeur ou expression.
```

Exemple :

```
int x,*p ;
…
p=&x ;
*p=34 ;
…
```

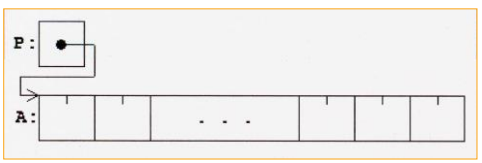
III. Affectation du contenue d'une variable à travers un pointeur

B. Cas d'un ensemble contiguë de valeurs (Tableau)

- Chaque opération avec des indices de tableaux peut aussi être exprimée à l'aide de pointeurs.
- Le nom d'un tableau est un pointeur **constant** sur le premier élément du tableau.

Exemple:

```
int A [10];
int * P;
P = A;      /*est équivalente à   P = &A[0]; */
```



5

III. Affectation du contenue d'une variable à travers un pointeur

B. Cas d'un ensemble contiguë de valeurs (Tableau)

- Si P pointe sur une composante quelconque d'un tableau, alors P + 1 pointe sur la composante suivante.
  - P + i pointe sur la i<u>ième</u> composante à droite de *P.
  - P - i pointe sur la i<u>ième</u> composante à gauche de *P.

Exemple:

```
int A[5] ;  int *P;
P=A;
```

- Incrémentation et décrémentation d'un pointeur

Si P pointe sur l'élément A[i] d'un tableau, alors après l'instruction	
`P++;`	P pointe sur A[i+1]
`P+=n;`	P pointe sur A[i+n]
`P--;`	P pointe sur A[i-1]
`P-=n;`	P pointe sur A[i-n]

`*(P+1)`	désigne le contenu de A[1]
`*(P+2)`	désigne le contenu de A[2]
...	...
`*(P+i)`	désigne le contenu de A[i]

6

III. Affectation du contenue d'une variable à travers un pointeur

B. Cas d'un ensemble contiguë de valeurs (Solution 1)

```
int A [10], n=10, i ;
int * P;
P = A;
/*remplissage du tableau par n reels*/
printf("\n Entrez %d données réelles \n",n);
  for (i=0; i<n; i++)
        scanf ("%d", P+i);
/*   P+ i ⇔ &P[i] ⇔ &A[i] */
/*affichage du tableau */
printf("\n Voici les éléments \n");
  for (i=0;i<n ; i++)
        printf ("%d \t ", P [i]);
  /*  *(P+ i) ⇔ P [i] ⇔ A[i] */
```


III. Affectation du contenue d'une variable à travers un pointeur

B. Cas d'un ensemble contiguë de valeurs (Solution 2)

```
int A [10], n=10 ;
int * P;

/*remplissage du tableau par n reels*/
printf("\n Entrez %d données réelles \n",n);
  for (P=A; P<A+n; P++)
        scanf ("%d", P);

/*affichage du tableau */
printf("\n Voici les éléments \n");
  for (P=A; P<A+n; P++)
        printf ("%d \t ", * P );
```

III. Affectation du contenue d'une variable à travers un pointeur

C. Cas d'un enregistrement (structure en C)

Exemple:

```
/* Soution 1 */
struct Complexe  {
        float reel;
        float img;        };
struct Complexe C1;

/* Soution 2 */
typedef struct {
        float reel;
        float img; } Complexe ;
 Complexe C1  ;
```


III. Affectation du contenue d'une variable à travers un pointeur

C. Cas d'un enregistrement (structure en C)

Exemple: /*accès aux champs d'une structure*/

```
struct Complexe  {
        float reel;
        float img;        } ;
struct Complexe c1;
struct Complexe * PC;
/* Cas d'une variable simple: l'accès aux champs se fait en
utilisant l'opérateur • (point)  */
 c1•reel=5.2; c1•img=3.2;
 PC = &c1;
/*Cas d'une variable de type pointeur: (2 méthodes) l'accès aux
champs se fait en utilisant l'opérateur • (point) ou en utilisant
l'opérateur spécial -> */
(*PC)• reel=4.4;  /* ⇔ PC -> reel=4.4 */
```

IV. Allocation dynamique de mémoire

- L'allocation dynamique consiste à réserver manuellement de l'espace en mémoire pour **une variable** ou **un tableau**.
- L'allocation dynamique de mémoire est basée principalement sur les deux méthode malloc et free de la bibliothèque<stdlib.h>.

- malloc (« Memory ALLOCation », c'est-à-dire « Allocation de mémoire ») : demande au système d'exploitation la permission d'utiliser de la mémoire ;

- free(« Libérer ») : permet d'indiquer à l'OS que l'on n'a plus besoin de la mémoire qu'on avait demandée. La place en mémoire est libérée, un autre programme peut maintenant s'en servir au besoin.

11

IV. Allocation dynamique de mémoire

A. Allocation de mémoire pour une simple variable

Exemple :

```
void main ( ) {
    int* memoireAllouee = NULL;

   memoireAllouee = malloc (sizeof (int));
   /* Allocation de la mémoire*/

     printf("Quel age avez-vous ? ");
    scanf("%d", memoireAllouee);
    printf ("Vous avez %d ans\n", *memoireAllouee);

   free(memoireAllouee);
   /* Libération de mémoire */
}
```

12

IV. Allocation dynamique de mémoire

B. Allocation de mémoire pour un tableau

Exemple :

```
int n=5;
float *tab;
tab =(float *) malloc (n* sizeof (float) );
/*reservation de n * 4 octes*/

/*Remplissage du tab   */
…
/*Affichage du tab */
…

free (tab);
/*liberation de la mémoire reservée*/
```


IV. Allocation dynamique de mémoire

C. Allocation de mémoire pour un enregistrement

Exemple:

```
struct Fiche_etudiant  {
                char nom[25];
                char prenom[25];
                int age;
                int matricule;     };
struct Fiche_etudiant * pointeur = NULL;

pointeur= malloc (sizeof (Fiche_etudiant));
/* reservation de
25*sizeof(char) + 25*sizeof(char) + sizeof(int) +sizeof(int) */

free (pointeur) ;
```

V. Exercices d'application

Exercice 1

Supposant que adr1 et adr2 sont des pointeurs pointant sur des réels. Le contenu de adr1 vaut -45,78; le contenu de adr2 vaut 678,89.
Écrire un programme qui affiche les valeurs de adr1, adr2 et valeurs pointées par adr1 et Adr2.

Exercice 2

Cet exercice n'a pas vraiment de sens mais il permet de se familiariser avec les pointeurs de types primitifs.
Définir une variable **var** d'un type primitif
Déclarer deux pointeurs **pVar1** et **pVar2** de ce type primitif
Indiquer que **pVar1** pointe sur **var**
Ajouter 2 à **var** (en utilisant **pVar1** et non **var**)
Affecter **pVar1** à **pVar2**
Ajouter 5 à la variable pointée par **pVar2**
Afficher les adresses des trois variables, puis leurs contenus et enfin les valeurs pointées par **pVar1** et **pVar2**

15

V, Exercices d'application

Exercice 3

On donne le programme C suivant :

```
void main ( ) {
    int A=1, B=2, C=3 ;
    int *P1, *P2 ;
    P1=&A ;
    P2=&C ;
    *P1= (*P2)++ ;
    P1=P2 ;
    P2=&B ;
    *P1-=*P2;
    ++*P2;
    *P1*=*P2;
    A=++*P2**P1;
    P1=&A;
    *P2=*P1/=*P2;
    }
```

16

V. Exercices d'application

Complétez le tableau suivant pour chaque instruction du programme ci-dessus :

	A	B	C	P1	P2
Initialisation	1	2	3	-	-
P1=&A ;	1	2	3	&A	-
P2=&C ;					
***P1= (*P2)++ ;**					
P1=P2 ;					
P2=&B ;					
***P1-=*P2;**					
++*P2;					
***P1*=*P2;**					
A=++*P2P1;**					
P1=&A;					
***P2=*P1/=*P2;**					

..

V. Exercices d'application

Exercice 4 : Fréquence

Soit T un tableau contenant n éléments de type entier et x un entier quelconque.

Écrire une fonction **int frequence (int * T , int n , int x)** qui retourne le nombre d'apparitions de x dans le tableau T.

Exercice 5 : Occurrence de 0

Soit T un tableau contenant n éléments de type entier,

Écrire une fonction **void SuppOccZero (int * T , int *n)** qui permet d'effacer toutes les occurrences de la valeur 0 dans le tableau T et tasser les éléments restants.

Rq: Le nombre d'éléments doit changer!!

18

V. Exercices d'application

Exercice 6 : Inverse

On souhaite écrire un programme C qui lit la dimension N d'un tableau T de type int remplit le tableau par des valeurs entrées au clavier et affiche le tableau. Ranger ensuite les éléments du tableau T dans l'ordre inverse sans utiliser de tableau d'aide. Afficher le tableau résultant.

Pour cefait on vous demande d'écrire les fonction c suivantes:

void remplir (int *t, int n)
void affiche (int *t, int n)
void inverser (int *t, int n)

Idée: Echanger les éléments du tableau à l'aide de deux indices qui parcourent le tableau en commençant respectivement au début et à la fin du tableau et qui se rencontrent en son milieu.

V. Exercices d'application

Exercice 7:

Écrire les sous-programmes C permettant de :

- saisir un nombre complexe z1.
- afficher les parties réelle et imaginaire du nombre z1.
- afficher les parties réelle et imaginaire du nombre z2 image de z1 par symétrie centrale.
- Calculer le produit de 2 nombres complexes z1, z2 passés en paramètres.
- Calculer la somme de 2 nombres complexes z1, z2 passés en paramètres.

On utilisera les formules de calcul suivantes :

- (a + b i) + (c + d i) = (a + c) + (b + d) i
- (a + b i) * (c + d i) = (a * c – b * d) + (a* d + b*c) i

V. Exercices d'application

Exercice 8

Créer un tableau TabEmp qui contiendra les informations sur un ensemble d'employés d'une entreprise (Matricule, Nom, Salaire, Etat_Civil), le remplir puis afficher le nombre d'employés dont le salaire est compris entre 500 et 700 D.

Exercice 9:

On suppose qu'un produit est caractérisé par son libellé, son prix d'achat et son prix de vente. Chaque produit est livré par un seul fournisseur. Un fournisseur est caractérisé par son code, sa raison sociale et son numéro de téléphone. On souhaite écrire un programme C qui permet de déminer et d'afficher la liste des produits hors stock (afficher juste le libellé et le code du fournisseur de chaque produit fini).

V. Exercices d'application

Exercice 10:

Soit un annuaire téléphonique comportant un ensemble de paires <nom, tel> avec nom est le nom de l'abonné et tel son numéro.

On vous demande de trouver le numéro de téléphone d'un abonné donné.

Solution 1: représentation de l'annuaire en deux tableaux indicées en parallèle

Solution 2: représentation de l'annuaire en utilisant un tableau (ou pointeur) sur un enregistrement composé de deux champs.

V. Exercices d'application

Exercice 11:

On suppose qu'une personne est caractérisée par son nom, son prénom, son numéro de téléphone et sa date de naissance (jour, mois et année sont séparés).

Déclarer les enregistrements nécessaires pour l'algorithme principal de la question 3

Ecrire une fonction C qui permet la saisie d'un ensemble de personnes.

Ecrire ensuite la fonction main qui permettra de:

1) Déclarer un pointeur sur une personne;
2) Saisir ces personnes
3) Donner une date et afficher un message de félicitation si cette date correspond à l'anniversaire d'une personne saisie.

V. Exercices d'application

Exercice 12:

Un cercle est caractérisé par son centre (de type Point) et son rayon (de type float). Chaque cercle peut :

Informer sur la position de son centre

Informer sur la taille de son rayon

Se déplacer

Changer de taille (rayon doit rester strictement positif)

Calculer sa surface

Calculer son périmètre

Fournir toutes ses caractéristiques. (y compris la surface et le périmètre)

Définir les structures de données et les sous-programmes permettant de:

- Créer un cercle C, afficher ses caractéristiques, le déplacer vers l'origine, mettre son rayon à 1 et enfin afficher ses caractéristiques de nouveau.
- Créer un point P dans un endroit quelconque et créer deux cercles concentriques C1 et C2 de rayons différents et ayant P comme centre. Le programme doit afficher la position la position du cercle C1, déplacer C2 avec des pas donnés puis afficher de nouveau la position du cercle C1.