

Chapitre VII

Les pointeurs

Objectifs

Général

⇒ *Se familiariser avec la notion des pointeurs.*

Spécifiques

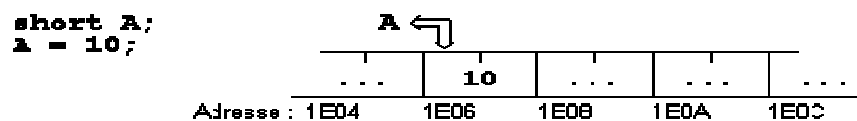
- ❑ *Rappeler la notion d'adressage et introduit l'adressage indirect par les pointeurs ;*
 - ❑ *Assimiler la notion de pointeur en langage C ;*
 - ❑ *Connaitre les opérations élémentaires sur les pointeurs.*
-

1. Adressage de variables

1.1. Adressage direct

Accès au contenu d'une variable son nom.

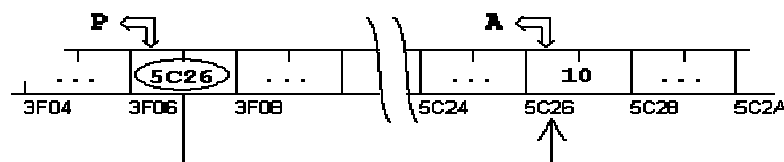
Exemple



1.2. Adressage indirect

Accès au contenu d'une variable, en passant par un pointeur qui contient l'adresse de la variable.

Exemple : Soit A une variable contenant la valeur 10 et P un pointeur qui contient l'adresse de A.



2. Les pointeurs

2.1. Définition

Un **pointeur** est une variable spéciale qui peut contenir l'**adresse** d'une autre variable.

Si un pointeur P contient l'adresse d'une variable A, on dit que '**P pointe sur A**'.

Remarque

- Un **pointeur** est une variable qui peut 'pointer' sur différentes adresses.
- Le **nom d'une variable** reste toujours lié à la même adresse.

2.2. Exemple :

```
int A = 10;
int *P;
P = &A;
printf(" La valeur de A est %d, L'adresse de A est %p \n", *P,P);
```

3. Les opérateurs de base

- L'opérateur 'adresse de' (&) permet d'obtenir l'adresse d'une variable.

&<NomVariable> fournit l'adresse de la variable <NomVariable>

- L'opérateur 'contenu de' (*) permet d'accéder au contenu d'une adresse.

***<NomPointeur> désigne le contenu de l'adresse référencée par le pointeur <NomPointeur>**

Déclaration d'un pointeur :

<Type> *<NomPointeur> déclare un pointeur <NomPointeur> qui peut recevoir des adresses de variables du type <Type>

4. Les opérations élémentaires sur les pointeurs

Les opérateurs * et & ont la même priorité que les autres opérateurs unaires (la négation !, l'incrément ++, la décrémentation --). Dans une même expression, les opérateurs unaires *, &, !, ++, -- sont évalués de droite à gauche.

Exemple (1) : Opérations élémentaires

```
P = &X;
```

Y = *P+1	⇔ Y = X+1
*P = *P+10	⇔ X = X+10
*P += 2	⇔ X += 2
++*P	⇔ ++X
(*P)++	⇔ X++

Exemple (2) : Pointeur NULL

```
int *P;

P = 0;
```

Exemple (3) : Soit P1 et P2 deux pointeurs sur **int**, alors l'affectation copie le contenu de P2 vers P1. P1 pointe alors sur le même objet que P2.

```
P1 = P2;
```

Application 1 :

```
int A = 10, B = 50;
int *P1, *P2;
P1 = &A;
B = *P1;
*P1 = 20;
P2=P1;
printf(" A = %d, &A est %p \n",.....,.....);
printf(" B = %d, &B est %p \n",.....,.....);
printf(" *P2= %d, P2 est %p \n",.....,.....);
```

Application 2 :

```
int A = 1;
int B = 2;
int 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;
```
