Chapitre 7 : ***LES POINTEURS***

Introduction

La plupart des langages de programmation offrent la possibilité d'accéder aux données dans la mémoire de l'ordinateur à l'aide de pointeurs, c'est à dire à l'aide de variables auxquelles on peut attribuer les adresses d'autres variables.

I. Adressage des variables

1. Adressage direct

Dans la programmation, nous utilisons des variables pour stocker des informations. La valeur d'une variable se trouve à un endroit spécifique dans la mémoire interne de l'ordinateur. Le nom de la variable nous permet alors d'accéder directement à cette valeur.

***Adressage direct** : Accès au contenu d'une variable par le nom de la variable.*

Exemple

```
short A;
A = 10;
```

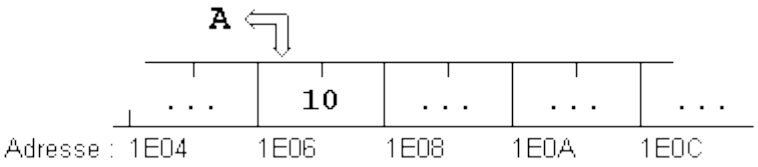


FIG. A

2. Adressage indirect

Si nous ne voulons ou ne pouvons pas utiliser le nom d'une variable A, nous pouvons copier l'adresse de cette variable dans une variable spéciale P, appelée pointeur. Ensuite, nous pouvons retrouver l'information de la variable A en passant par le pointeur P.

***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. En mémoire, A et P peuvent se présenter comme suit :

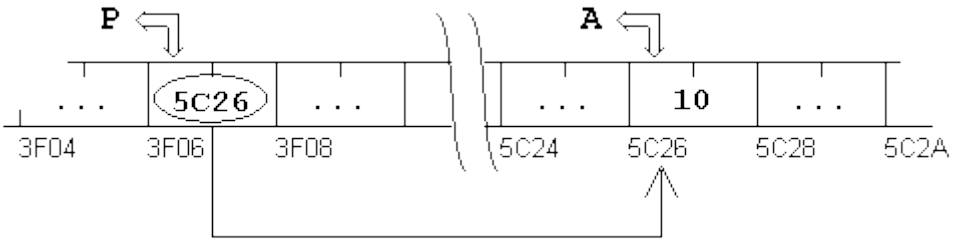


FIG. B

II. Définition

Un pointeur est une variable spéciale qui peut contenir l'adresse d'une autre variable. En C, chaque pointeur est limité à un type de données. Il peut contenir l'adresse d'une variable simple de ce type ou l'adresse d'une composante d'un tableau de ce type. Si un pointeur P contient l'adresse d'une variable A, on dit que : 'P pointe sur A'.

Remarque

Les pointeurs et les noms de variables ont le même rôle. Ils donnent accès à un emplacement dans la mémoire interne de l'ordinateur. Il faut quand même bien faire la différence :

** Un pointeur est une variable qui peut 'pointer' sur différentes adresses.*

** Le nom d'une variable reste toujours lié à la même adresse.*

1. Les opérateurs de base

Lors du travail avec des pointeurs, nous avons besoin

- d'un opérateur 'adresse de': & pour obtenir l'adresse d'une variable.

*- d'un opérateur 'contenu de': * pour accéder au contenu d'une adresse.*

- d'une syntaxe de déclaration pour pouvoir déclarer un pointeur.

a. L'opérateur 'adresse de' : &

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

L'opérateur & nous est déjà familier par la fonction scanf, qui a besoin de l'adresse de ses arguments pour pouvoir leur attribuer de nouvelles valeurs.

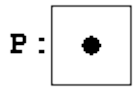
Exemple

```
int N;
printf("Entrez un nombre entier : ");
```

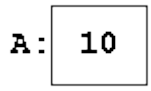
scanf("%d", &N);

Représentation schématique

Soit P un pointeur non initialisé



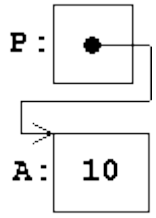
et A une variable (du même type) contenant la valeur 10 :



Alors l'instruction

P = &A;

affecte l'adresse de la variable A à la variable P. En mémoire, A et P se présentent comme dans la figure FIG. B. Dans notre représentation schématique, nous pouvons illustrer le fait que 'P pointe sur A' par une flèche:



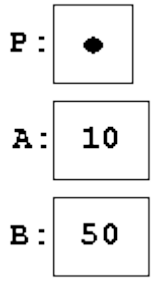
***b. L'opérateur 'contenu de' : **

**<NomPointeur>*

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

Exemple

Soit A une variable contenant la valeur 10, B une variable contenant la valeur 50 et P un pointeur non initialisé:



Après les instructions,

P = &A;

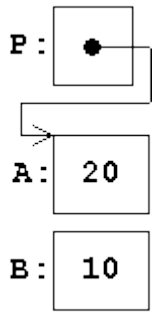
*B = *P;*

**P = 20;*

- P pointe sur A,

*- le contenu de A (référencé par *P) est affecté à B, et*

*- le contenu de A (référencé par *P) est mis à 20.*



c. Déclaration d'un pointeur

*<Type> *<NomPointeur> ;*

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

Une déclaration comme

*int *PNUM;*

peut être interprétée comme suit:

*"*PNUM est du type int"*

ou

"PNUM est un pointeur sur int"

ou

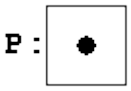
"PNUM peut contenir l'adresse d'une variable du type int"

Remarque

Lors de la déclaration d'un pointeur en C, ce pointeur est lié explicitement à un type de données. Ainsi, la variable PNUM déclarée comme pointeur sur int ne peut pas recevoir l'adresse d'une variable d'un autre type que int.

d. Le pointeur NULL

Seule exception, la valeur numérique 0 (zéro) est utilisée pour indiquer qu'un pointeur ne pointe 'nulle part'.



*int *P;*

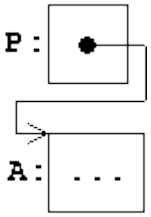
P = 0;

Finalement, les pointeurs sont aussi des variables et peuvent être utilisés comme telles. Soit P1 et P2 deux pointeurs sur int, alors l'affectation

P1 = P2;

copie le contenu de P2 vers P1. P1 pointe alors sur le même objet que P2.

Résumons:



Après les instructions:

int A;

*int *P;*

P = &A;

A désigne le contenu de A

&A désigne l'adresse de A

P désigne l'adresse de A

**P désigne le contenu de A*

En outre:

&P désigne l'adresse du pointeur P

**A est illégal (puisque A n'est pas un pointeur)*