

Travaux Pratiques N°9

Les Pointeurs

I. Définition :

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.

II. Déclaration :

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

III. 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=a, *pi ;
Pi=&a ;
```

Remarque:

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

```
int *pi ;
Pi=0xffffe ;
```

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

Exemple :

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

IV. Affectation du contenu d'une variable à travers un pointeur

L'affectation du contenu d'un objet à un pointeur se fait en utilisant l'opérateur d'indirection (*) selon la syntaxe : *pointeur= valeur ou expression.

Exemple :

```
int x,*p ;
...
P=&x ;
*p=34 ;
...
```

V. Arithmétique des pointeurs

On peut essentiellement déplacer un pointeur dans un plan mémoire à l'aide des opérateurs d'addition, de soustraction, d'incrémentation et de décrémentation. Le déplacement ne peut se faire que d'un nombre de cases mémoire multiple du nombre de cases réservées en mémoire pour la variable sur laquelle il point ($n * \text{sizeof}(\text{type})$).

Exemple :

```
int x,*p ;
...
P=&x ;
*p=0x421 ;
...
*(p+3)=0x53 ;
Pi++ ;
```

VI. Utilisation de la fonction scanf avec les pointeurs

Pour saisir un caractère ou un nombre, on donne en fait à la fonction `scanf` l'adresse de ce caractère ou nombre. Pour le cas des pointeurs, on ne donne à la fonction `scanf` que le nom du pointeur.

Exemple :

```
float *pf ;
scanf("%f ", pf) ;
```

VII. Travail demandé

Exercice 1

Supposant que `adr1` et `adr2` sont deux pointeurs pointant sur deux réels. Le contenu de la variable pointée par `adr1` vaut -45,78 et le contenu de la variable pointée par `adr2` vaut 678,89.

Écrire un programme qui affiche les valeurs de : `adr1`, `adr2` et des deux valeurs des variables 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.

1. Définir une variable **var** d'un type primitif
2. Déclarer deux pointeurs **pVar1** et **pVar2** de ce type primitif
3. Indiquer que **pVar1** pointe sur **var**
4. Ajouter 2 à **var** (en utilisant **pVar1** et non **var**)
5. Affecter **pVar1** à **pVar2**
6. Ajouter 5 à la variable pointée par **pVar2**
7. Afficher les adresses des trois variables, puis leurs contenus et enfin les valeurs pointées par **pVar1** et **pVar2**

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;
}
```

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=++*P2**P1;					
P1=&A;					
*P2=*P1/=*P2;					