Les tableaux et les pointeurs

Rappel

Un pointeur est une variable ayant pour valeur l'adresse physique dans la RAM d'une autre variable.

Par exemple, soient les instructions suivantes :

1. float X;
2. float *PX;
3. PX = &X;
4. scanf("%f",PX);
5. printf("%f",*PX);

Dans l'instruction 1 : float X

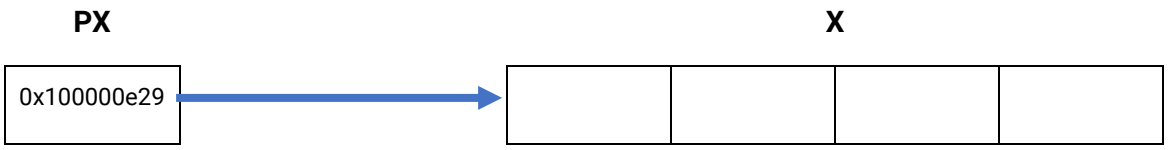
Nous déclarons une variable de type float.

Instruction 2 : float *PX

Nous déclarons un pointeur PX. Comme vous le constatez PX est précédée par une étoile.

Instruction 3 : PX = &X

Nous affectons l'adresse de X, notée &X, au pointeur PX. A Ce niveau, voilà, ce qu'on a au niveau de la RAM, un pointeur PX qui contient l'adresse physique dans la RAM du premier octet de la variable X. C'est pour cela, qu'on dit que PX pointe sur X. La variable X occupe 4 octets.



Instruction 4 : scanf("%f",PX)

Nous appelons la fonction scanf pour entrer un réel depuis le clavier. Cette valeur sera rangée dans l'adresse contenue dans PX, et cette adresse fait référence à la variable X. Donc nous pouvons écrire :

1. scanf("%f",PX);
2. scanf("%f",&X);

PX ⇔ &X les deux désignent l'adresse de X dans la RAM

Instruction 5 : printf("%f",*PX)

Dans cette instruction, nous appelons la fonctions printf pour afficher un réel (flaot). La valeur à afficher est contenue dans la case dans la RAM ayant pour adresse situant dans PX. (PX contient l'adresse de X dans la RAM). Cette instruction peut être écrite de deux façons :

1. printf("%f",X);
2. printf("%f",*PX);

X⇔ *PX les deux désignent la valeur contenue dans X.

Un pointeur qui pointe sur un tableau

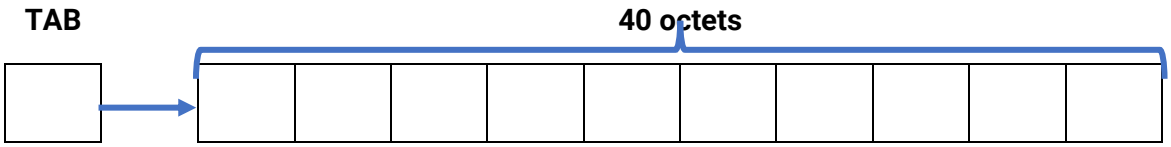
Soit les instructions suivantes :

1. float TAB[10];
2. float *PT;
3. PT = &TAB[0];

L'instruction 1 : float TAB[10];

Dans cette instruction, nous déclarons un tableau de 10 réels. Chaque case va occuper 4 octets. Les dix cases du tableau sont contiguës. C'est à dires les cases se suivent l'une après l'autre.

Au niveau de la RAM, nous avons, un pointeur TAB qui pointe sur l'adresse du premier octet de la première case du tableau. TAB contient l'adresse physique du premier octet de la première case du tableau. Dans notre exemple, chaque case du tableau occupe 4 octets (tableau de type float).



Ici TAB joue deux rôles. Il est en même temps un pointeur et un tableau (Vous allez comprendre pourquoi je dis que TAB est un pointeur).

L'instruction 2 : float *PT

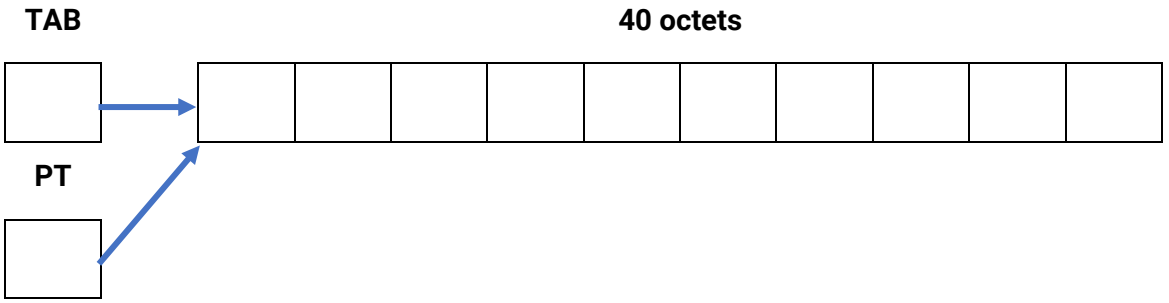
Dans cette instruction, nous déclarons un pointeur nommée PT. Qui ne pointe sur aucune variable.

L'instruction 3 : PT = &TAB[0]

Dans cette instruction, nous affectons l'adresse physique dans la RAM de la première case du tableau TAB, notée &TAB[0] au pointeur PT. Le pointeur PT contient l'adresse physique dans la RAM de la première case du tableau TAB. On dit ainsi que le pointeur PT pointe sur le tableau TAB.

De façon générale, pour qu'un pointeur pointe sur tableau, il suffit qu'il contient l'adresse physique de la première case du tableau.

Au niveau de la RAM, nous avons deux pointeurs TAB et PT qui pointent sur notre tableau.



Pour affecter l'adresse physique du tableau TAB au pointeur PT, nous avons deux façons :

1. PT = &TAB[0]
2. PT = TAB

Vous remarquez que nous avons écrit :

PT = TAB,

C'est-à-dire que TAB contient l'adresse physique du premier élément du tableau. Nous concluons que TAB est un pointeur constant. C'est-à-dire que son contenu ne change pas. Vous ne pouvez jamais écrire :

TAB = PT

TAB contient l'adresse physique du tableau lors de la déclaration et son adresse ne change jamais jusqu'à la fin du programme.

Ce qu'il faut retenir :

&TAB[0] ⇔ TAB sont correspondantes

Remplir un tableau en utilisant un pointeur

Soient les instructions suivantes :

```
1. float TAB[10];
2. float *PT;
3. int i;
4. PT = &TAB[0];
5. for(i=0; i<10; i++)
    scanf("%f",PT+i);
```

Instruction 1:

Dans cette instruction nous déclarons un tableau TAB de 10 réels.

Instruction 2

Nous déclarons un pointeur PT.

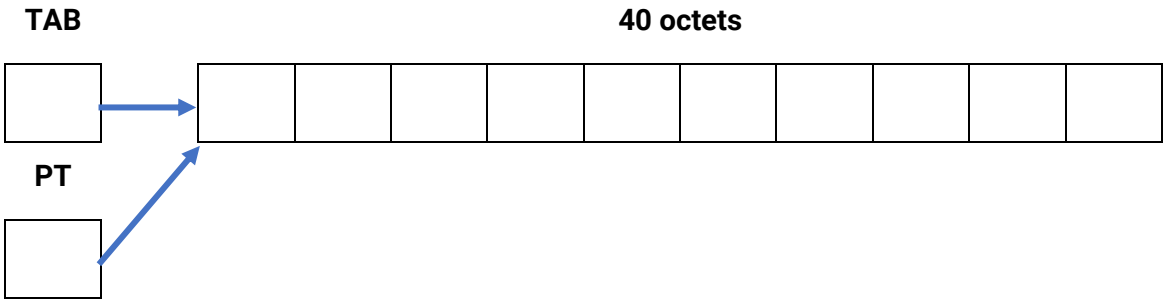
Instruction 3

Nous déclarons un compteur nommé i pour parcourir le tableau TAB.

Instruction 4 :

Nous affectons au pointeur PT l'adresse physique du tableau TAB. Ainsi, PT pointe sur le tableau TAB.

A ce niveau, nous avons dans la RAM les variables suivantes :



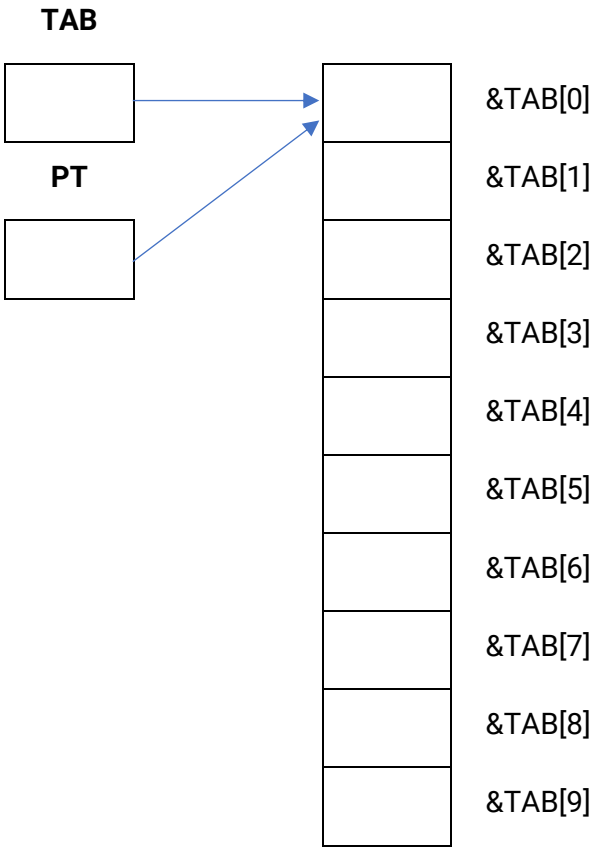
Instruction 5

```
for(i=0; i<10; i++)
 scanf("%f",PT+i);
```

Nous utilisons la boucle for pour entrer les éléments du tableau TAB. Dans la fonction scanf, nous n'avons pas utilisé &TAB[i]. C'est-à-dire nous pouvons écrire cette boucle en utilisant la notation tableau :

```
for(i=0; i<10; i++)
 scanf("%f",&TAB[i]);
```

Pour entrer les éléments du tableau en utilisant le pointeur PT, au lieu d'écrire &TAB[i] nous allons écrire PT + i. Je vous explique :



Vous le constatez, à côté de chaque élément du tableau TAB, j'ai mis son adresse physique. Par exemple,

1. Adresse du premier élément est &TAB[0]
2. Adresse du deuxième élément est &TAB[1]
3. Adresse du troisième élément du TAB est &TAB[2]
4. Et l'adresse du dixième élément du TAB est &TAB[9]

Nous savons que PT contient l'adresse du premier élément du tableau TAB. C'est-à-dire que : PT correspond à &TAB[0]. Donc nous pouvons écrire : scanf("%f",PT); ou scanf("%f",&TAB[i]);

Alors quel est le correspondant de &TAB[1] en terme de pointeur PT? C'est-à-dire quelle est l'adresse du deuxième élément du tableau TAB ?

PT + 1 correspond à l'adresse du deuxième élément du tableau TAB. C'est à dire à partir de l'adresse contenue dans PT, il s'agit du premier élément du tableau, nous demandons

au système d'exploitation de passer, non à l'octet suivant mais de sauter 4 octets et d'aller à l'octet 5. L'octet 5 est le début du deuxième élément du tableau. Ne pas oublier que nous travaillons avec des variables de type float qui occupent 4 octets dans la RAM (le nombre d'octets changent en fonction du type). Ainsi nous disons que :

PT + 1 ⇔ &TAB[1]

PT + 2, désigne l'adresse du troisième élément du tableau. PT + 2 correspond à &TAB[2]. C'est dire que nous demandons au système d'exploitation de chercher l'adresse physique qui se trouve à l'octet 4*2+1 c'est-à-dire à l'octet 9. L'octet 9 est le début du troisième élément du tableau.

Ainsi pour récapituler, nous pouvons écrire :

PT + 0 ⇔ &TAB[0]
PT + 1 ⇔ &TAB[1]
PT + 2 ⇔ &TAB[2]
PT + 3 ⇔ &TAB[3]
PT + 4 ⇔ &TAB[4]
PT + 5 ⇔ &TAB[5]
PT + 6 ⇔ &TAB[6]
PT + 7 ⇔ &TAB[7]
PT + 8 ⇔ &TAB[8]
PT + 9 ⇔ &TAB[9]

Donc, nous pouvons remplacer dans scanf &TAB[i] par PT + i et nous aurons :

```
for(i=0; i<10; i++)
 scanf("%f",PT+i);
```

Attention de faire précéder PT + 1 par l'opérateur & dans scanf. C'est à dire vous écrivez scanf("%f",&PT+i). Ce n'est pas correct.

Afficher un tableau en utilisant un pointeur

Soit l'instruction suivante :

```
for(i=0; i<10; i++)
        printf("%f\n",*(PT+i));
```

Cette instruction peut être remplacée la notation tableau :

```
for(i=0; i<10; i++)
        printf("%f\n",TAB[i]);
```

Dans la section précédente, nous avons dit que :

PT + i correspond à &TAB[i], c'est à dire que PT + i correspond à l'adresse du i ème élément +1 du tableau TAB. Par exemple ; PT + 4 correspond à l'adresse du cinquième élément du tableau TAB.

Dans notre cas actuel, nous voulons afficher le contenu du i ème élément du tableau et non son adresse.

En effet, pour accéder au contenu d'un élément du tableau en utilisant la notation tableau, nous utilisons TAB[i] où i désigne le numéro de la case dans le tableau.

Comment accéder aux éléments du tableau en utilisant le pointeur PT ?

PT désigne l'adresse du premier élément du tableau. Alors que *PT désigne le contenu du premier élément du tableau TAB. C'est-à-dire que :

*PT correspond à TAB[0]

Ainsi nous aurons :

*(PT + 0) ⇔ TAB[0]
*(PT + 1) ⇔ TAB[1]
*(PT + 2) ⇔ TAB[2]
*(PT + 3) ⇔ TAB[3]
*(PT + 4) ⇔ TAB[4]
*(PT + 5) ⇔ TAB[5]
*(PT + 6) ⇔ TAB[6]
*(PT + 7) ⇔ TAB[7]
*(PT + 8) ⇔ TAB[8]
*(PT + 9) ⇔ TAB[9]

Pour écrire le premier élément du tableau :

```
printf("%f\n",*(PT+0));
```

Pour écrire le deuxième élément du tableau :

```
printf("%f\n",*(PT+1));
```

Pour écrire le troisième élément du tableau :

```
printf("%f\n",*(PT+2));
```

Pour écrire le dixième élément du tableau :

```
printf("%f\n",*(PT+9));
```

Voici le programme permettant de lire et d'afficher les 10 éléments du tableau en utilisant un pointeur.

```
#include <stdio.h>
int main()
{
   float TAB[10];
   float *PT;
   int i;
   PT = &TAB[0];
   for(i=0; i<10; i++)
      scanf("%f",PT+i);

   for(i=0; i<10; i++)
      printf("%f\n",*(PT+i));
}
```

Pré ou Post incrémentation sur un pointeur

1. float TAB[10];
2. float *PT;
3. PT = &TAB[0];
4. PT++ ;

Instruction 1 : Nous déclarons un tableau de 10 réels.

Instruction 2 : Nous déclarons un pointeur PT.

Instruction 3 : le pointeur PT pointe sur le premier élément du tableau.

Instruction 4 :

PT++ est l'équivalent de PT = PT +1

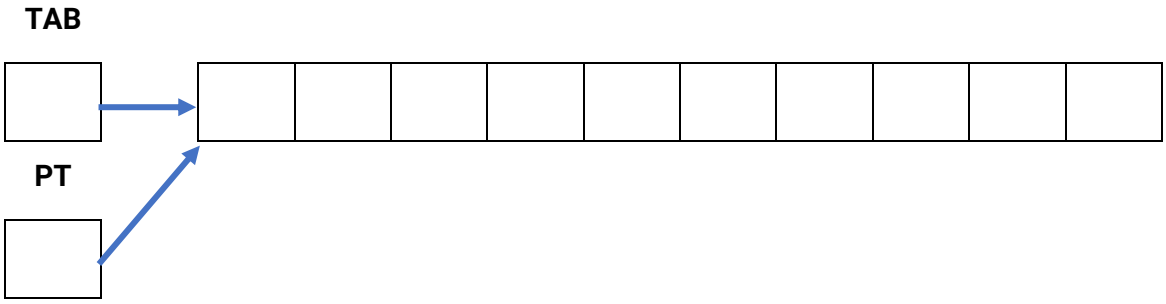
PT + 1 correspond à l'adresse du deuxième élément du tableau TAB. C'est-à-dire que :

PT + 1 correspond à &TAB[1].

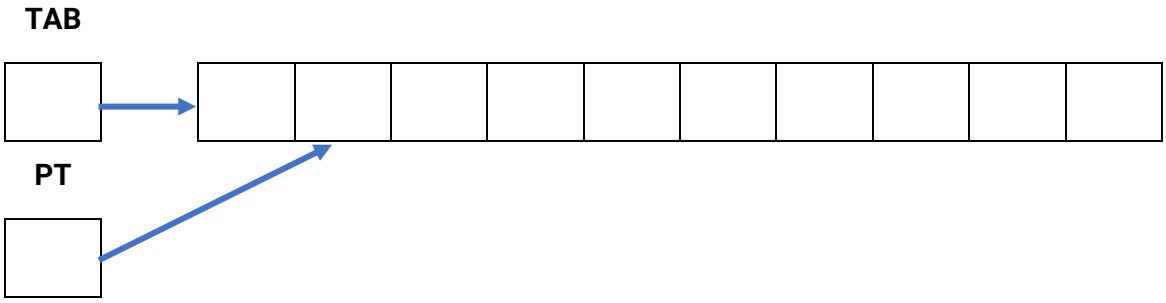
Ainsi, PT++ est l'équivalent de PT = &TAB[1]

Après l'exécution de cette instruction PT++ , PT pointe sur le deuxième élément du tableau.

Avant l'exécution de PT++, nous avons,



Après l'exécution de l'instruction PT++, nous aurons :



Quelle différence entre PT + i et PT++ ?

Dans PT + i, le pointeur PT reste toujours pointé sur le premier élément du tableau. C'est-à-dire que le pointeur PT garde toujours l'adresse physique du premier élément du tableau même si i s'incrémente.

L'instruction PT++ permet au pointeur de changer son contenu. A chaque incrémentation, le pointeur PT pointe sur l'élément suivant.

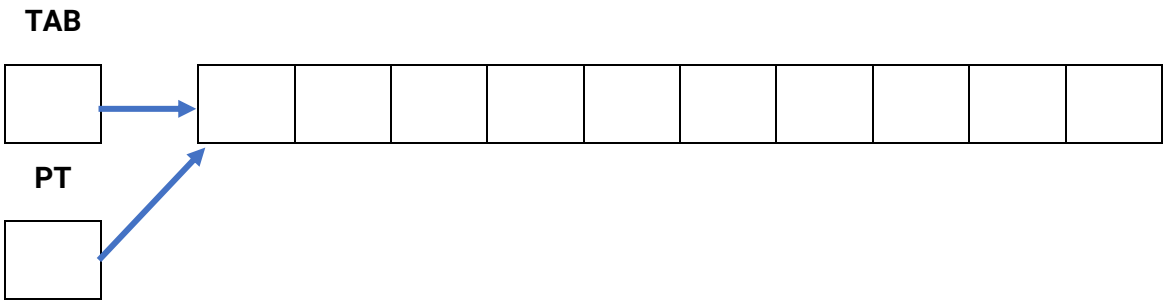
Soit l'exemple suivant :

```
PT = &TAB[0];
for(i=0; i<10; i++)
        scanf("%f",PT++);
```

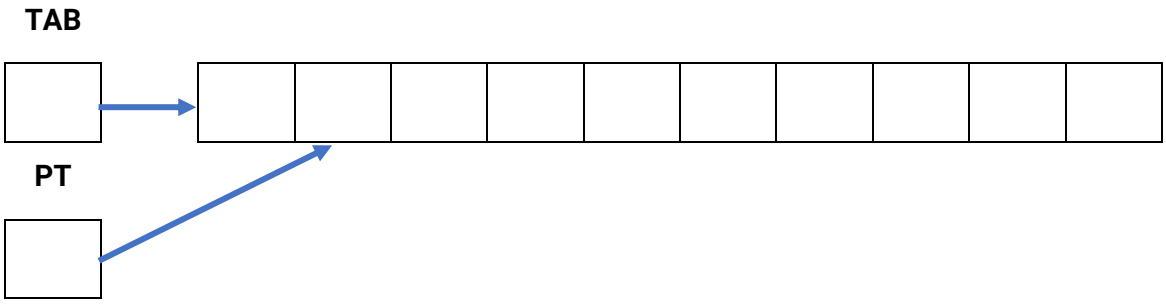
Dans cet exemple, nous remplissons les éléments du tableau en utilisant le pointeur PT. Vous remarquez que à chaque incrémentation, PT pointe sur l'élément suivant jusqu'à la fin du tableau.

A la fin de la boucle for, PT pointe sur l'octet qui vient après le dixième élément du tableau TAB. C'est-à-dire que PT pointe en dehors du tableau.

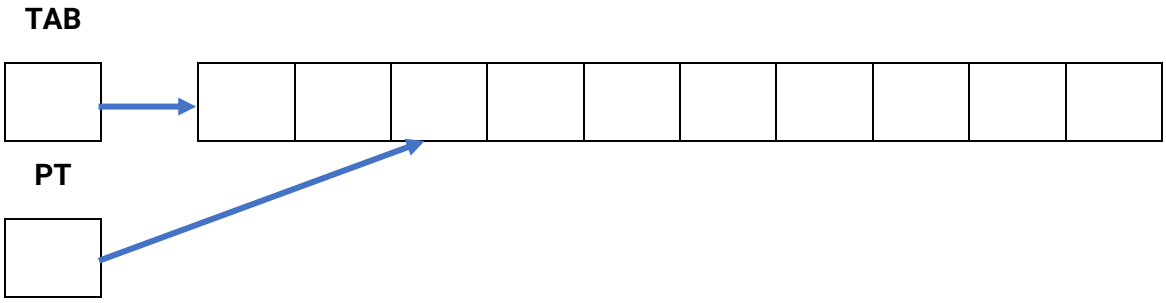
Pour i=0, PT pointe sur le premier élément du TAB



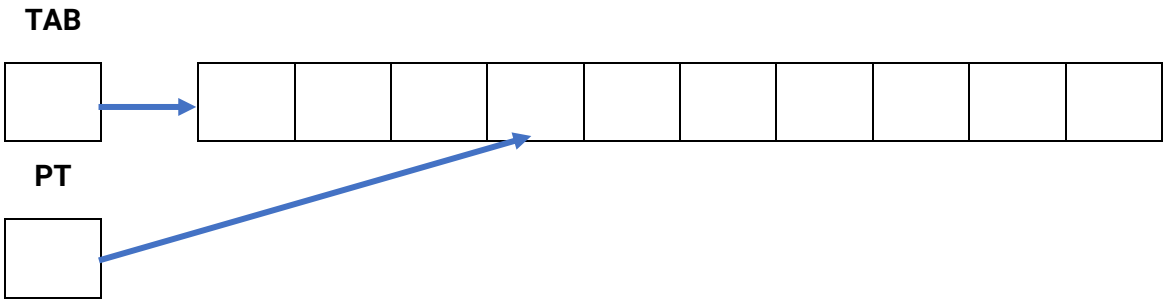
Pour i=1, PT pointe sur le deuxième élément du TAB



Pour i=2, PT pointe sur le troisième élément du TAB

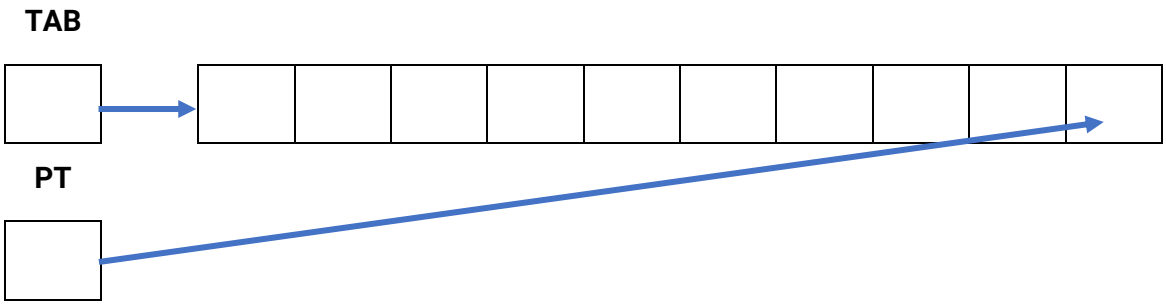


Pour i=3, PT pointe sur le quatrième élément du TAB

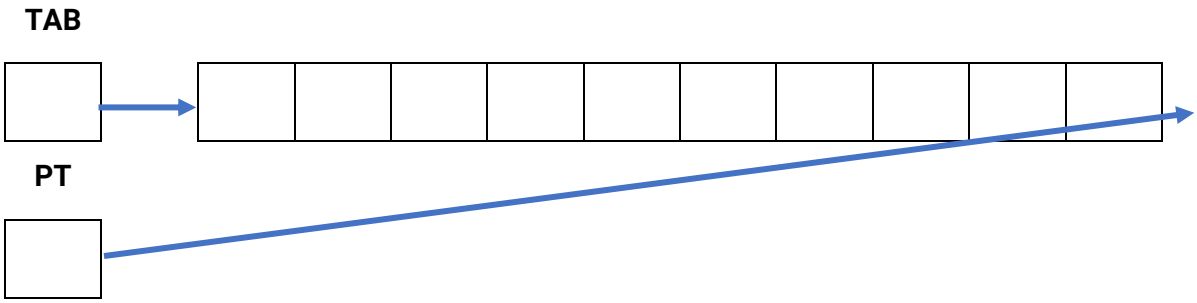


…

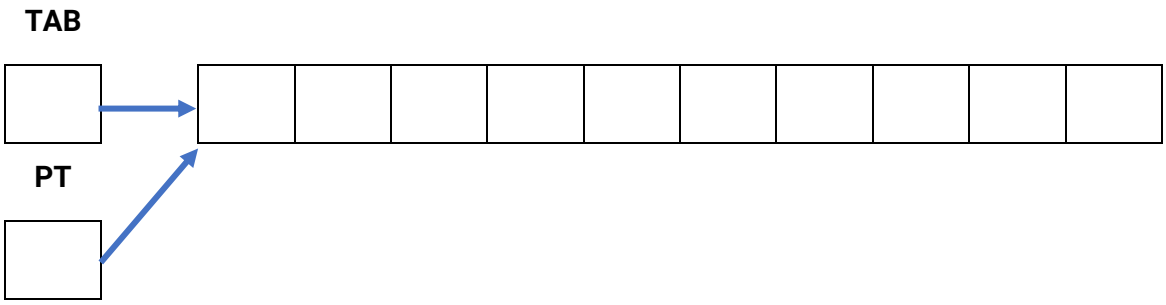
Pour i=9, PT pointe sur le dixième élément du TAB



Pour i=10, PT pointe en dehors du TAB



Pour revenir à l'état initial, c'est-à-dire, nous voulons que PT pointe sur le premier élément du TAB, il faut, écrire : PT = &TAB[0]



Soit le programme suivant permettant de remplir et d'afficher le tableau de 10 réels en utilisant le post incrémentation sur le pointeur PT

```
#include <stdio.h>
int main()
{
   float TAB[10];
   float *PT;
   int i;
   PT = &TAB[0];
   for(i=0; i<10; i++)
       scanf("%f",PT++);
   PT = &TAB[0];
   for(i=0; i<10; i++)
       printf("%f\n",*(PT++));
}
```

Vous remarquez qu'avant l'instruction affichant les éléments du tableau :

```
for(i=0; i<10; i++)
       printf("%f\n",*(PT++));
```

il y' a instruction `PT = &TAB[0];` permettant de réinitialiser le pointeur PT et qui doit pointer sur le premier élément du tableau afin de commencer l'affichage des éléments du tableau depuis le début.

A noter : L'étoile avant PT++ dans printf n'est pas obligatoire.