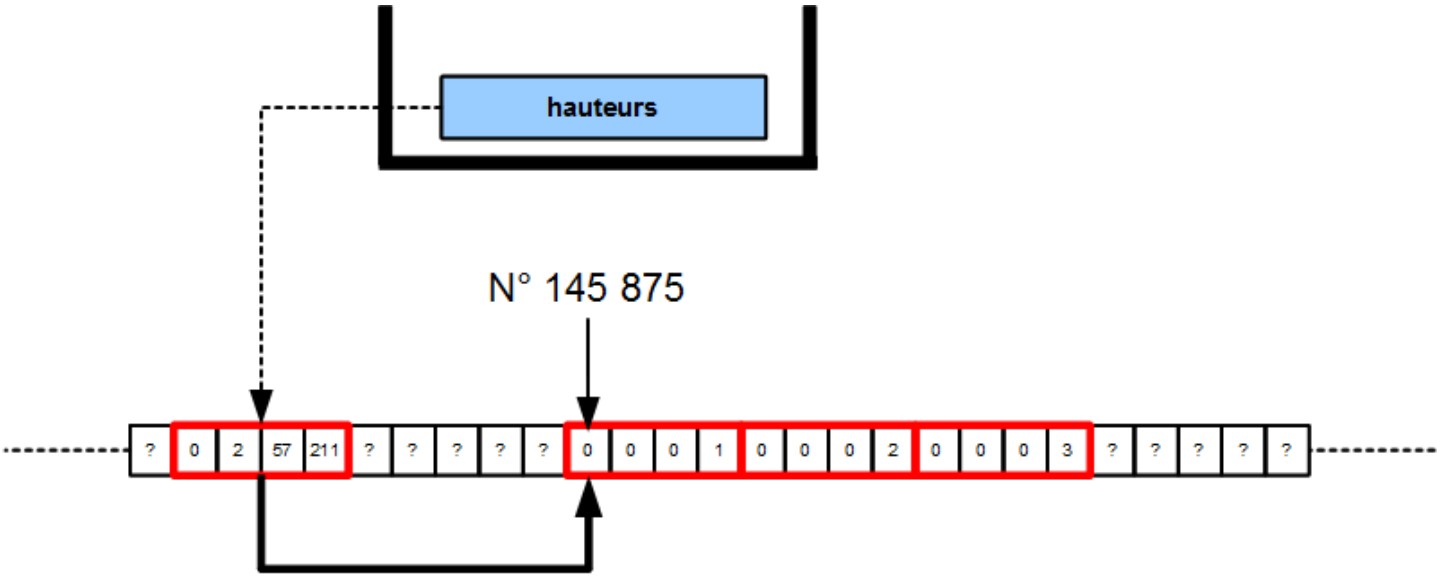
Dans ce cours, nous allons nous intéresser à la manière dont les tableaux sont représentés en mémoire. On reprendra les schémas sur les pointeurs qu'on a déjà utilisé.

Considérons la déclaration de tableau suivante :

```
int hauteurs[3] = {1, 2, 3};
```

Alors la représentation en mémoire est la suivante (comme précédemment, la valeur 145 875 est juste un exemple) :



Suprenant, non ? Est-ce que c'est ce que vous imaginiez ?

Regardons un peu plus en détails comme cela fonctionne :

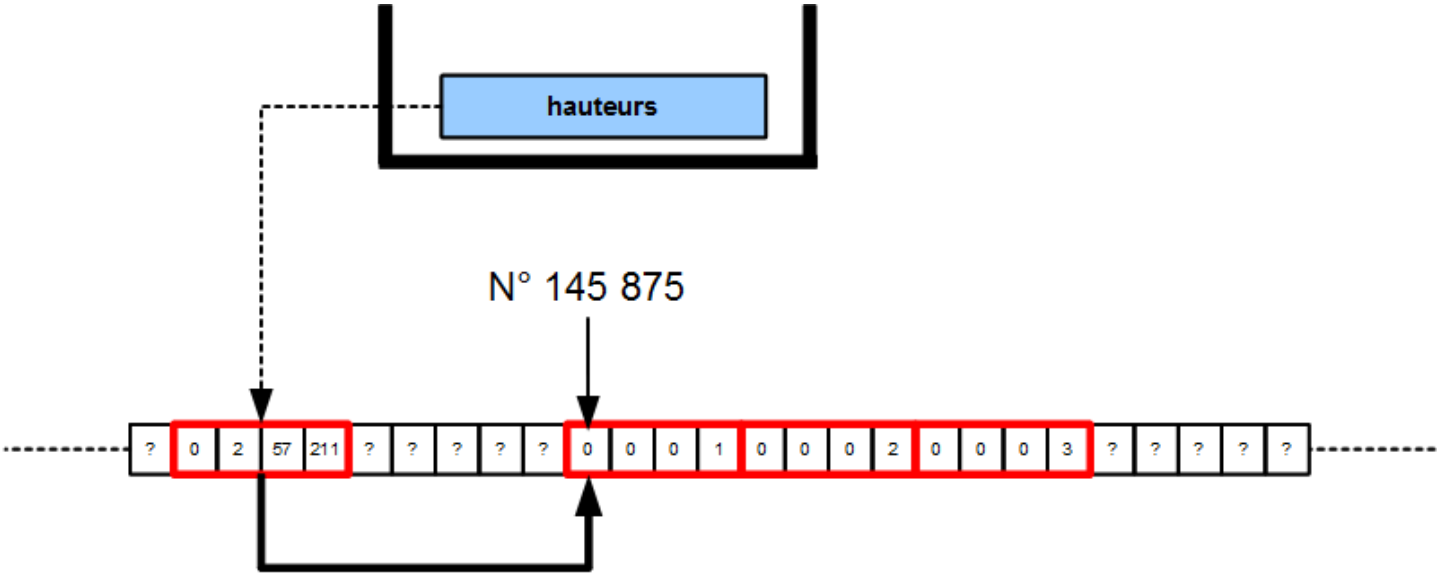
- On voit que la variable `hauteurs` envoie vers un espace en mémoire qui contient l'adresse d'une autre case de la mémoire, la numéro 145 875.
- Le groupe de 4 cases qui démarre à la numéro 145 875, contient la première valeur du tableau, les 4 cases suivantes contiennent la seconde valeur du tableau, et ainsi de suite.
- Un tableau se comporte en pratique **PRESQUE** comme un pointeur vers un entier.

Arithmétique de pointeurs et tableaux

Considérons le code suivant :

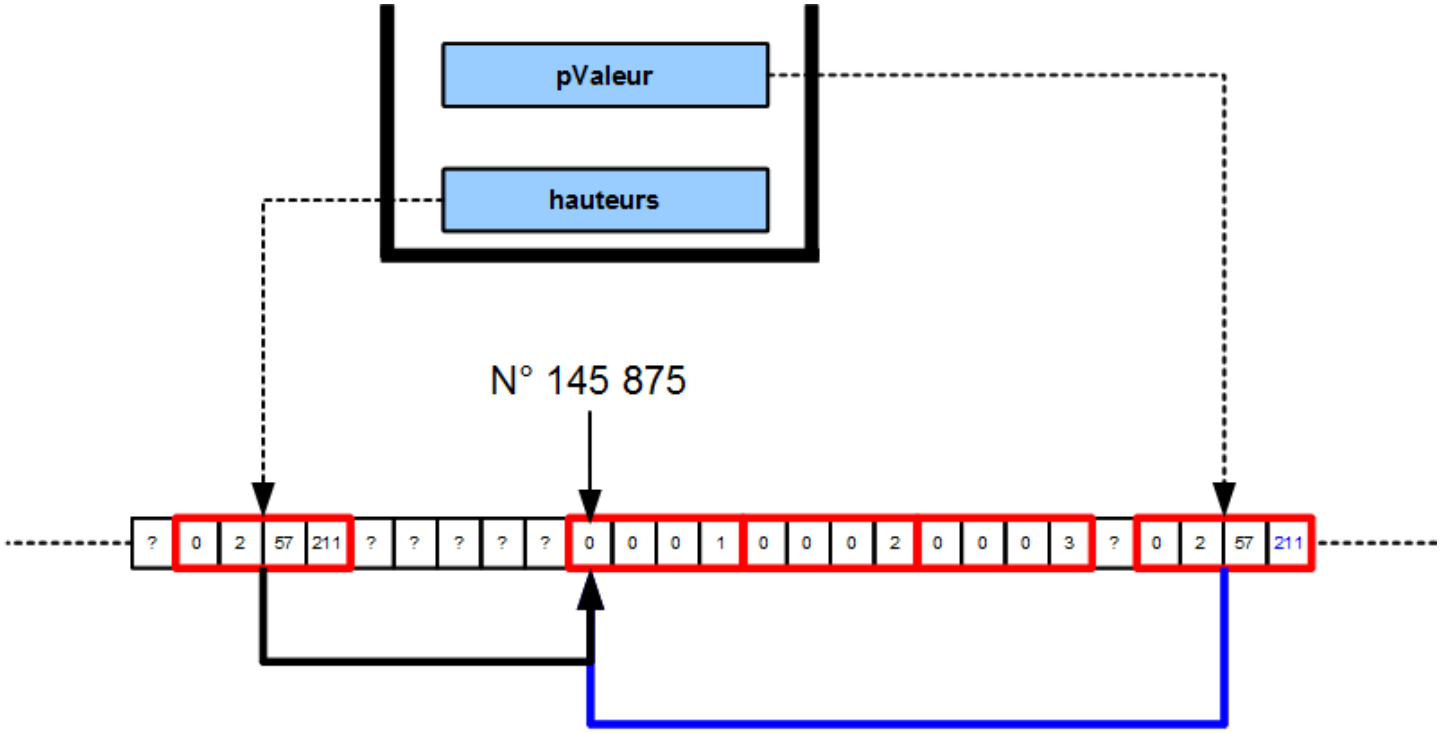
```
int hauteurs[3] = {1, 2, 3};

int* pValeur = &hauteurs[0];
pValeur = pValeur + 1;
*pValeur = 4;
pValeur = pValeur + 1;
*pValeur = 9;
```

On va regarder ce qui se passe, ligne à ligne. Au début on à ceci :



Ensuite, on crée un pointeur `pValeur` qu pointe vers le premier élément du tableau, à l'aide du code

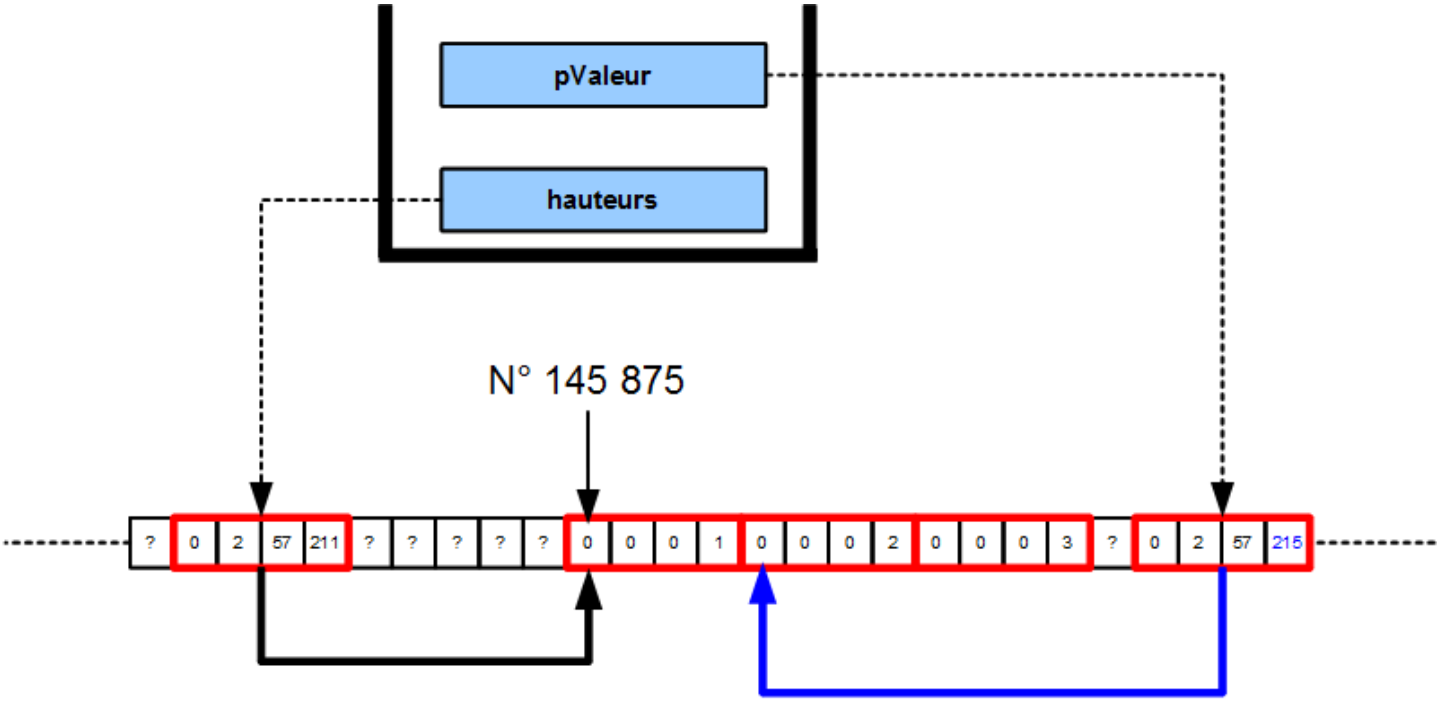
```
int* pValeur = &hauteurs[0];
```



Puis on ajoute 1 à ce pointeur, à l'aide du code

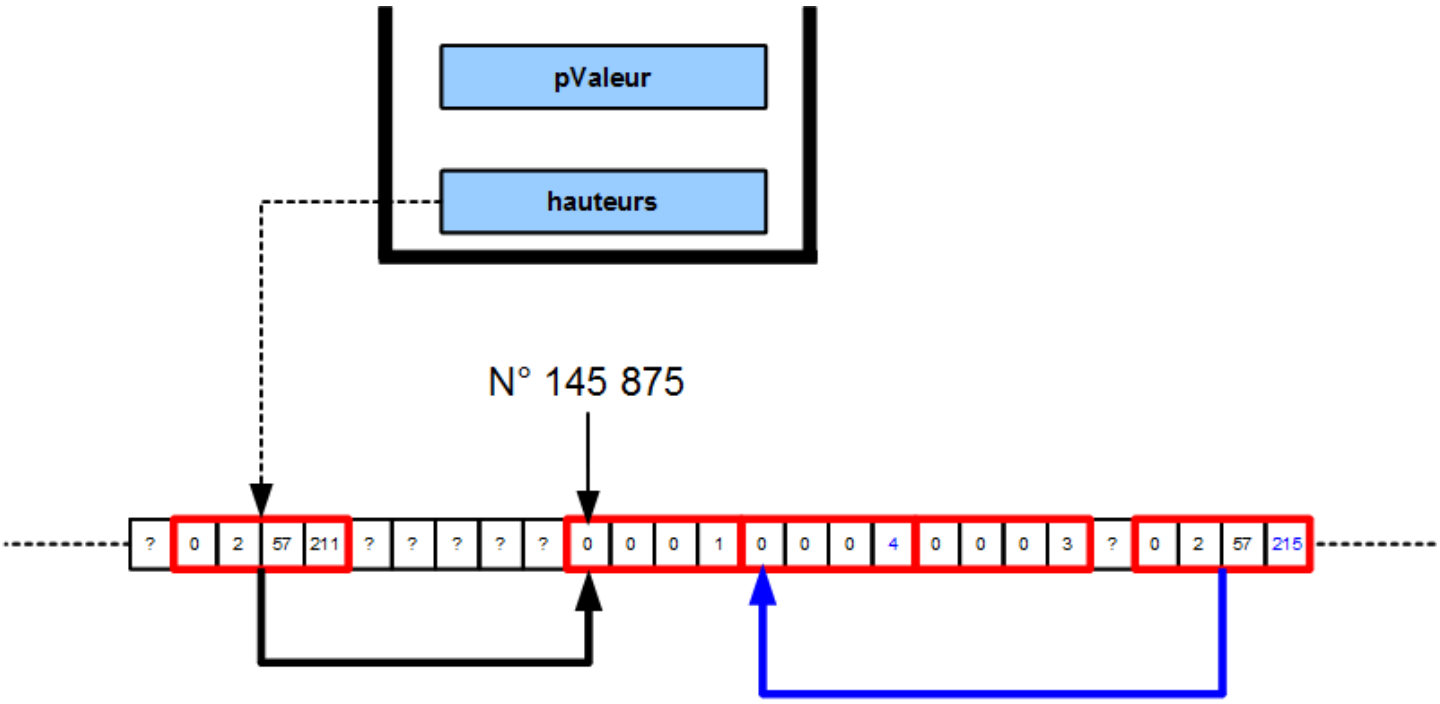
```
pValeur = pValeur + 1;
```

ce qui a pout effet de le décaler d'une case dans le tableau (et donc de 4 cases en mémoire). Notez le passage de 211 à 215 dans la dernière case du pointeur.



On modifie alors la case vers laquelle pointe le pointeur, en lui donnant la valeur 4

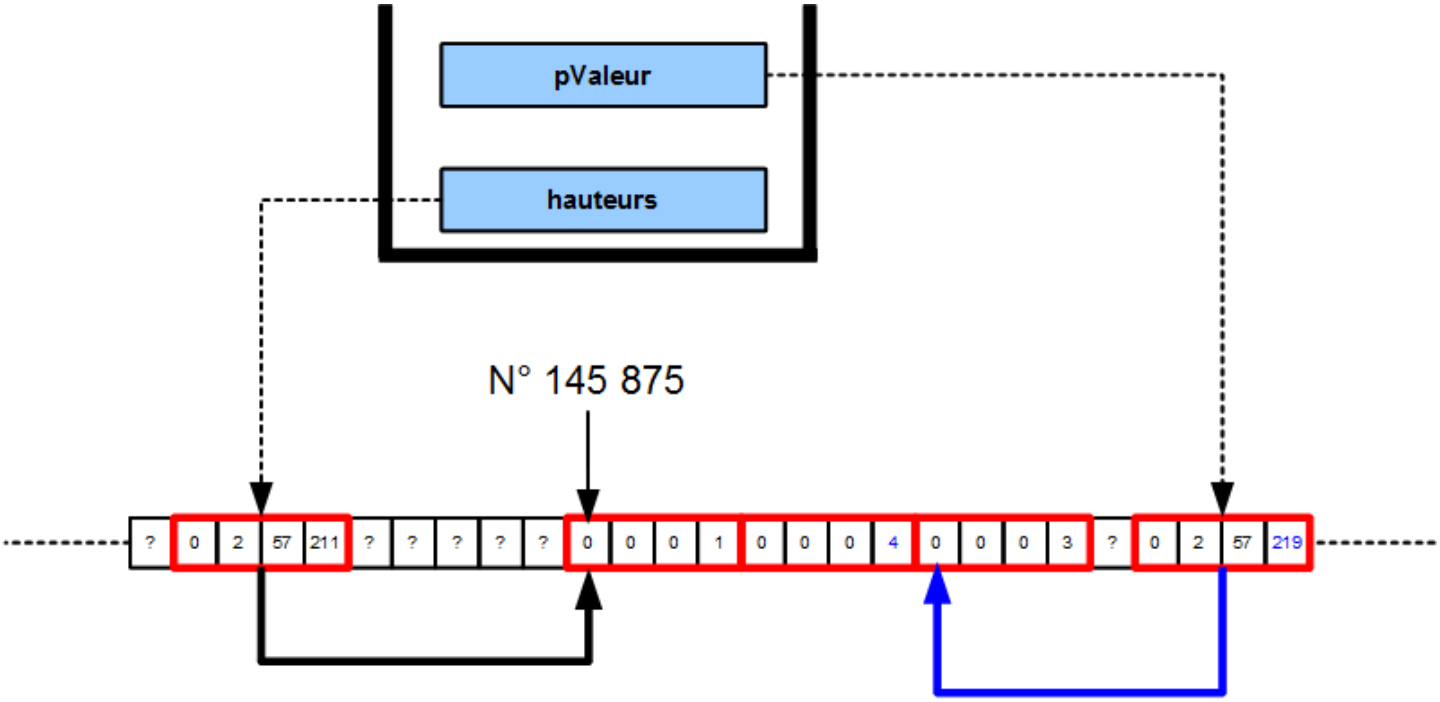
```
*pValeur = 4;
```



Puis on ajoute 1 à ce pointeur, à l'aide du code

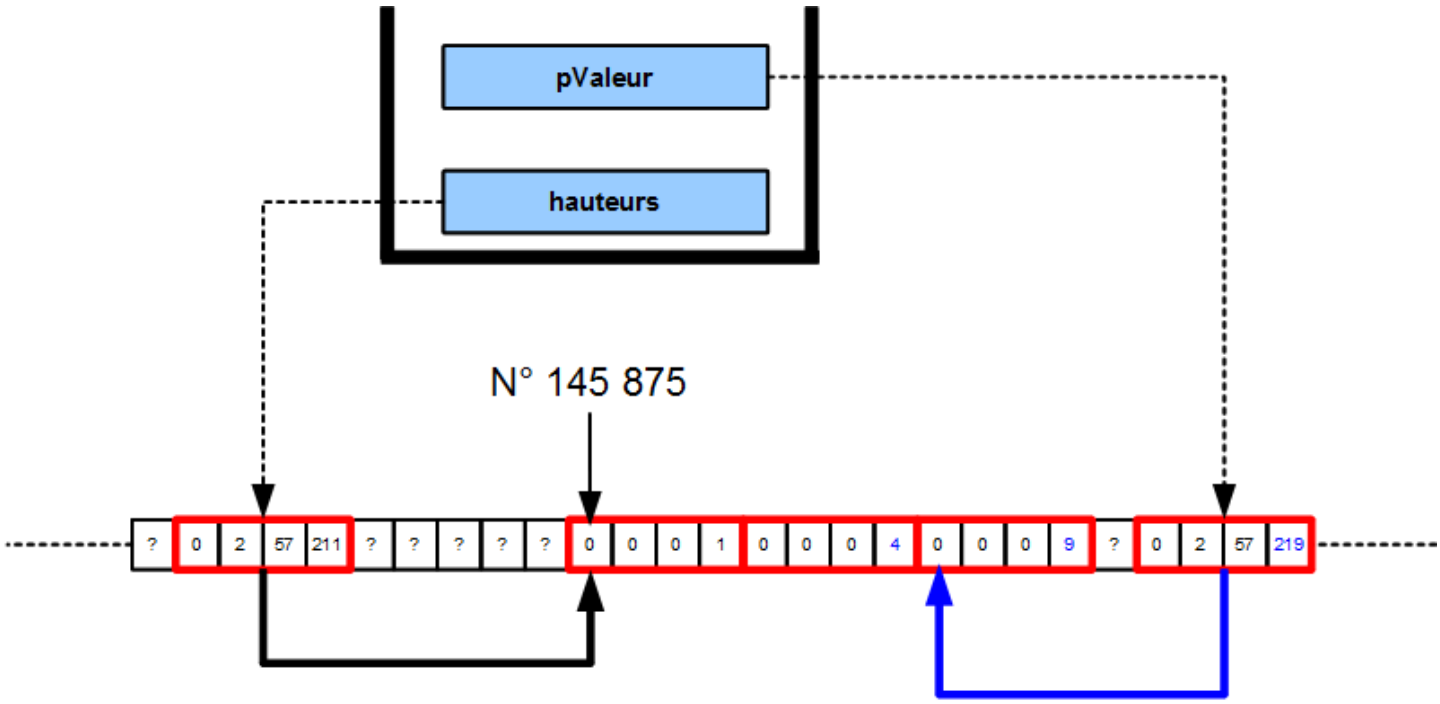
```
pValeur = pValeur + 1;
```

ce qui a pout effet de le décaler d'une case dans le tableau (et donc de 4 cases en mémoire). Notez le passage de 215 à 219 dans la dernière case du pointeur.



On modifie alors la case vers laquelle pointe le pointeur, en lui donnant la valeur 9

```
*pValeur = 9;
```



Code équivalent

On aurait pu remplacer le code précédent par celui-ci :

```
int hauteurs[3] = {1, 2, 3};

// Ancien code
// int* pValeur = &hauteurs[0];
// Nouveau code
int* pValeur = hauteurs;

pValeur = pValeur + 1;
*pValeur = 4;
pValeur = pValeur + 1;
*pValeur = 9;
```

On peut donc affecter au pointeur `pValeur` la contenu de la variable `hauteurs`, qui est l'adresse en mémoire du premier élément du tableau. La variable `hauteurs` se comporte donc presque comme un pointeur : elle ne l'est pas totalement, par on ne pourrait **PAS** effectuer l'opération inverse, c'est-à-dire affecter à `hauteurs` le contenu de la variable `pValeur`.

Quelques précisions

Comme l'ordinateur sait que le pointeur est un pointeur vers un entier, il sait qu'il pointe vers un élément prenant 4 cases en mémoire, donc quand on ajoute 1 au pointeur, l'ordinateur sait qu'il faut le décaler de 4 cases en mémoire.

Si on avait un pointeur vers un autre type de donnée (booléen, double...) qui est stocké sur plus ou moins de 4 cases en mémoire, alors ajouter 1 au pointeur le décalerait du nombre de cases correspondante.