L'une des erreurs les plus courantes avec les tableaux est d'essayer de lire un élément qui n'existe pas, c'est-à-dire d'utiliser un indice trop grand :

```
int hauteurs[3] = {1, 2, 3};
cout << hauteurs[0] << endl;
cout << hauteurs[1] << endl;
cout << hauteurs[2] << endl;
cout << hauteurs[3] << endl;
```

```
warning: array subscript is above array bounds
```

```
1
2
3
-1076055816
```

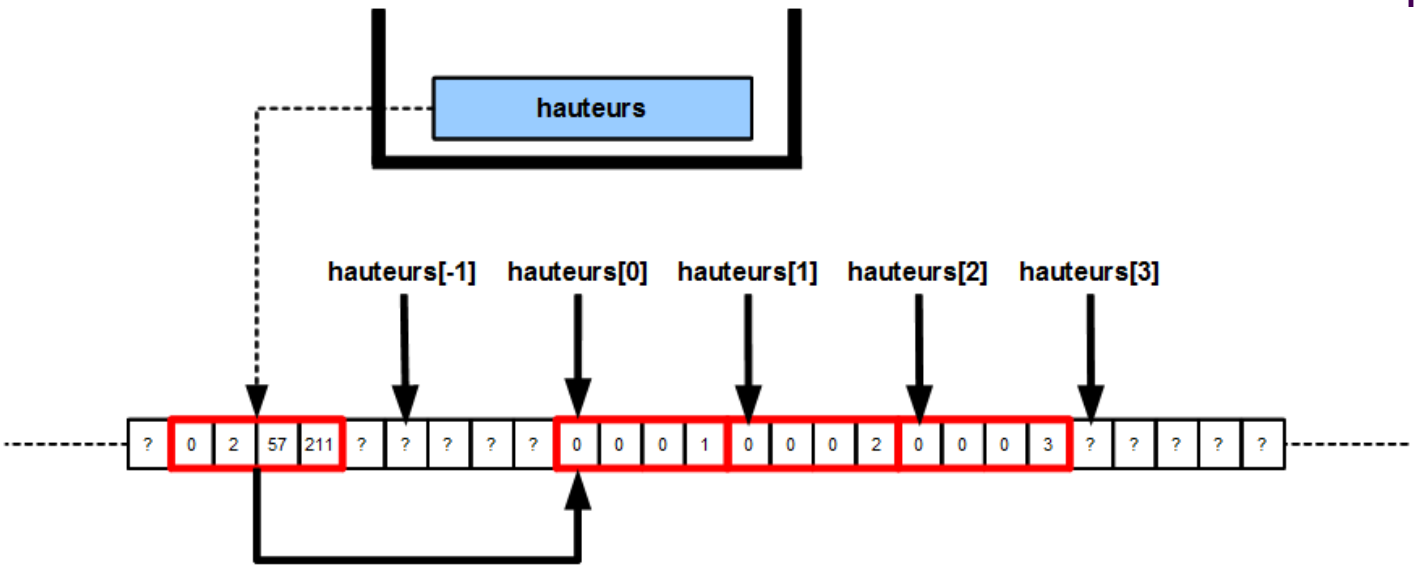
Un avertissement s'affiche donc, qui nous préviens d'un des indices utilisé était trop grand. On a en effet utilisé l'indice 3 alors que les indices vont de 0 à 2.

Que se passe-t-il si on essaie d'utiliser un indice négatif ? On obtient le warning suivant :

```
warning: array subscript is below array bounds
```

Attention : cet avertissement est possible car il est facile de voir que le tableau n'a que 3 éléments et donc que `hauteurs[3]` est invalide. Dans de cas plus complexes, le compilateur est incapable de voir à l'avance qu'il peut y avoir une erreur, et aucun avertissement ne sera affiché.

Vous avez cependant remarqué que le programme s'est tout de même exécuté, et qu'il a affiché une valeur. Où est-il allé chercher cette valeur ? Souvenez-vous du schéma de la mémoire que nous avons déjà vu :



Si on utilise l'indice 3 ou -1, par exemple, on est simplement allé lire les 4 octets situés après (ou avant) le tableau et on les a interprété comme un entier qu'on a affiché.

Ici on a simplement affiché une mauvaise valeur mais si cette zone mémoire avait, par exemple, été donnée à un autre processus (un autre logiciel pour simplifier) sur l'ordinateur alors le programme se serait arrêté avec une grosse erreur. En effet, un programme n'a pas le droit d'aller lire des données appartenant à un autre programme et si on essaie de le faire, le système d'exploitation arrête tout de suite le programme. Cela vous sera signalé sur le site, souvent par le message "accès mémoire en dehors des zones allouées". Chez vous, ce sera souvent une "Segmentation fault".

Marquer comme lu et continuer