`i++` ou `++i` ?

J.-C. Chappelier, J. Sam, V. Lepetit

version 1.0 de septembre 2013

On rencontre souvent dans la littérature et dans des exemples de code l'utilisation de l'opérateur d'incrémentation postfixé, comme par exemple dans l'expression `i++`, alors que dans le cours nous avons opté pour une notation préfixée, par exemple `++i`.

Y a-t-il une différence et si oui quelle est-elle ? Et pourquoi un tel choix pour le cours ?

1 Effet et valeur

Avant tout vous ne pourrez pas pleinement comprendre ce dont il s'agit si vous ne distinguez pas ce que *fait* une expression de ce qu'elle *vaut*.

En C++, **toute** expression *fait* quelque chose et *vaut* quelque chose. Elle *vaut* quelque chose en ce sens qu'on peut par exemple la mettre à droite d'une affectation.

Par exemple, « `i = 3` » est une affectation, mais en C++, c'est aussi une expression. C'est une expression qui *fait* une affectation de la valeur 3 à la variable `i`. Mais cette expression *vaut* aussi quelque chose, c'est-à-dire que l'on peut parfaitement écrire : `j = i = 3;` ce qui signifie : `j = (i = 3);` ou encore : « met dans `j` la *valeur* de l'expression '`i = 3`' ». Ceci est tout à fait licite en C++.

Je m'empresse immédiatement de dire qu'il faut absolument éviter d'utiliser ce genre d'expressions qui rendent le code beaucoup moins intelligible. Rappelez vous que vous devez toujours écrire le code le plus clair possible !

Que vaut donc l'expression « `i = 3` » ? Il se trouve qu'elle vaut le résultat de l'affectation, donc 3 ici. Ainsi `j = (i = 3);` est la même chose que

```
i = 3;
j = i;
```

2 i++ et ++i

Je peux maintenant expliquer la différence entre les expressions « `i++` » et « `++i` »:
- ces deux expressions *font* exactement la **même** chose : elles incrémentent `i` ;
- en revanche, leurs *valeurs* sont **différentes**.

Si vous n'utilisez pas la valeur de ces expressions [1], il n'y aura donc pour vous aucune différence.

Utiliser leur valeur signifierait par exemple écrire des choses comme : `j = ++i;`, ce que je vous déconseille une fois de plus de faire.

Leur seule différence est donc au niveau de leur valeur : `i++` *vaut* la valeur de `i` *avant* incrémentation alors que `++i` vaut la valeur de `i` *après* incrémentation.

Si l'on prend l'exemple suivant :

```
int i(3);
int j(i); // i et j ont la même valeur
int k(0);
int l(0);
k = ++i; // opérateur préfixé
l = j++; // opérateur postfixé
```

A l'issue de ce bout de code, `i` et `j` auront tous les deux la valeur 4 (les deux opérateurs *font* la même chose), mais `k` aura la valeur 4 alors que `l` aura la valeur 3 (les deux opérateurs ne *valent* pas la même chose).

3 Lequel préférer ?

Encore une fois, si vous n'utilisez pas la valeur de retour et si vous n'êtes pas dans un langage qui permette de surcharger (= redéfinir) ces opérateurs [2], alors il n'y aura pour vous aucune différence pratique entre ces deux notations.

Ceci dit, il y **deux** bonnes raisons de préférer la notation préfixée (`++i`) à celle postfixée (`i++`) : l'une conceptuelle et l'autre plus pragmatique dans les langages où ces opérateurs peuvent être surchargés.

La raison conceptuelle est la suivante : quelle opération voulons nous exprimer par `i++` (ou `++i`) ? Si c'est « `i = i + 1` » alors la seule qui lui soit *totalement* équivalent est `++i`.

En effet, quelle est la valeur de l'expression `i = i + 1` ? C'est bien la valeur de `i` *après* incrément, la même que celle de `++i`.

En d'autres termes, par quoi remplacer `i = i + 1` dans
`j = i = i + 1;`
Le seul remplacement valide entre `i++` et `++i` est le second :
`j = ++i;`

Voilà pour la raison conceptuelle.

1. Et si vous ne surchargez pas ces opérateurs, voir la section «Lequel préférer ?».
2. Java ne le permet pas, mais C++ oui.

D'un point de vue pratique maintenant : si le langage permet de surcharger ces opérateurs (et C++ le permet), alors il faut *toujours* préférer la notation préfixée (`++i`) car elle est sensiblement moins coûteuse.

Je prétends en effet, sans le démontrer ici, que l'opérateur préfixé peut s'implémenter *sans* aucune copie, alors qu'il est impossible de faire l'opérateur postfixé sans copie. L'opérateur postfixé coûte donc toujours une copie de plus que l'opérateur préfixé. Si ces opérateurs sont surchargés pour des objets coûteux à copier, alors la différence de performance se fera sentir !

Pour ces deux raisons, il est donc préférable d'utiliser l'opérateur préfixé (`++i`) à celui postfixé (`i++`). Et c'est pour cela qu'il en est ainsi dans ce cours.