yo ++ yo ++ yo ?

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

versión 1.0 de septiembre de 2013

A menudo encontramos en la literatura y en ejemplos de código el uso del operador de incremento de posfijo, como por ejemplo en la expresión i++, mientras que en el curso optamos por una notación de prefijo, por ejemplo + +i.

¿Hay alguna diferencia y, de ser así, cuál es? ¿Y por qué tal elección para el curso?

1 Efecto y valor

Sobre todo, no podrás entender completamente de qué se trata si no distingues entre lo que hace una expresión y lo que vale.

En C++, cualquier expresión hace algo y vale algo. Vale algo en el sentido de que uno puede, por ejemplo, ponerlo a la derecha de una tarea.

Por ejemplo, "i = 3" es una asignación, pero en C++ también es una expresión. Es una expresión que asigna el valor 3 a la variable i. Pero esta expresión también vale algo, es decir que perfectamente podemos escribir: j = i = 3; lo que significa: j = (i = 3); o también: “poner en j el valor de la expresión 'i = 3'”. Esto es perfectamente legal en C++.

Inmediatamente me apresuro a decir que es absolutamente necesario evitar el uso de este tipo de expresiones que hacen que el código sea mucho menos inteligible. ¡Recuerda que siempre debes escribir el código lo más claro posible!

Entonces, ¿cuál es el valor de la expresión "i = 3"? Sucede que vale la pena el resultado de la tarea, así que 3 aquí. Así j = (i = 3); es lo mismo que

```
yo = 3; j
= yo;
```

2 i++ te ++i

Ahora puedo explicar la diferencia entre las expresiones “i++” y “++i”: – estas dos expresiones hacen exactamente lo mismo : incrementan i; – por otro lado, sus valores son diferentes.

Si no utiliza el valor de estas expresiones diferencia.[1], así que no habrá

Usar su valor significaría, por ejemplo, escribir cosas como: j = ++i;, lo cual nuevamente desaconsejo.

Por lo tanto, su única diferencia está en el nivel de su valor: i++ es el valor de i antes del incremento, mientras que ++i es el valor de i después del incremento.

Si tomamos el siguiente ejemplo:

```
int i(3);
intj(yo); // i y j tienen el mismo valor int k(0); intl(0); k = +
+i; // operador
prefijado l =
j++; // operador de sufijo
```

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 misma cosa).

3 ¿Cuál prefieres?

Nuevamente, si no está utilizando el valor de retorno y si no está en un lenguaje que permita sobrecargar (= redefinir) estos operadores, entonces no habrá[2] ninguna diferencia práctica para usted entre estas dos notaciones.

Dicho esto, hay dos buenas razones para preferir la notación de prefijo (++i) a la notación de sufijo (i++): una conceptual y otra más pragmática en lenguajes donde estos operadores pueden estar sobrecargados.

La razón conceptual es la siguiente: ¿qué operación queremos expresar por i++ (o ++i)? Si es "i = i + 1", entonces el único que es completamente equivalente a él es ++i.

De hecho, ¿cuál es el valor de la expresión i = i + 1? De hecho, es el valor de i después del incremento, el mismo que el de ++i.

En otras palabras, qué reemplazar i = i + 1 en j = i = i + 1;

El único reemplazo válido entre i++ y ++i es el segundo: j = +
+i;

Tanto por la razón conceptual.

1. Y si no sobrecarga estos operadores, consulte la sección "¿Cuál preferir?".
2. Java no lo permite, pero C++ sí.

Ahora desde un punto de vista práctico: si el lenguaje permite sobrecargar estos operadores (y C++ lo permite), entonces siempre debería preferir la notación prefijada (++i) porque es significativamente menos costosa.

De hecho, afirmo, sin demostrarlo aquí, que el operador de prefijo puede implementarse sin ninguna copia, mientras que es imposible hacer el operador de sufijo sin una copia. Por lo tanto, el operador sufijo siempre cuesta una copia más que el operador prefijo. Si estos operadores están sobrecargados para objetos que son costosos de copiar, ¡entonces se sentirá la diferencia de rendimiento!

Por estas dos razones, es preferible utilizar el operador de prefijo (++i) al de sufijo (i++). Y por eso es así en este curso.