Lorsque l'on veut modifier une variable en fonction de sa propre valeur, la notation est un peu lourde. Prenons par exemple cette instruction qui permet d'augmenter la variable *nombre* de 1, ce que l'on a fait tant de fois :

```
nombre = nombre + 1;
```

Le langage C++ propose en fait un opérateur combinant l'addition et l'affectation : `+=`. Il existe également de tels opérateurs pour tous les calculs arithmétiques du langage (et même d'autres opérations). Nous vous en présentons ci-dessous ceux qui correspondent aux opérations que nous avons utilisées jusqu'à présent.

```
nombre += ajout;
nombre -= retrait;
nombre *= facteur;
nombre /= diviseur;
nombre %= modulateur;
```

Observez alors que si l'on écrit :

```
lapins *= lapins + agitation;
```

pour appliquer la croissance d'une population de lapins, on multiplie *lapins* par *lapins* + *agitation*. C'est donc équivalent à une affectation avec des parenthèses :

```
lapins = lapins * (lapins + agitation);
```

Incrémentation et décrémentation de 1

On a souvent besoin d'ajouter 1 ou d'enlever 1 à une variable. On pourrait donc maintenant le faire de la manière suivante :

```
nbJoursDepuisRentree += 1;
nbJoursAvantVacances += 1;
```

Il existe en fait une notation encore un peu plus courte, que voici :

```
nbJoursDepuisRentree++;
nbJoursAvantVacances--;
```

On utilise là les opérateurs `++` et `--`. Il n'est pas si utile de s'en servir, mais ils sont néanmoins très répandus dans les codes de programmes.

Remarque : Ces opérateurs peuvent également se placer avant la variable à incrémenter ou décrémenter, comme suit :

```
++nbJoursDepuisRentree;
--nbJoursAvantVacances;
```

L'autre écriture est généralement utilisée car elle se rapproche plus des affectations usuelles. Il y a en fait une légère différence entre la *pré-incrémentation* (opérateur avant) et la *post-incrémentation* (opérateur après), qui se note lorsque l'on insère l'opération dans une expression : `++nb` augmente *nb* et renvoie sa nouvelle valeur, tandis que `nb++` augmente aussi *nb* mais renvoie son ancienne valeur.

Nous vous déconseillons cependant fortement d'insérer une opération de modification dans une expression pour éviter tout souci : faites en sorte de toujours les mettre seules sur une instruction.

Conclusion

Nous emploierons désormais ces opérateurs dans les cours et corrections. Nous vous encourageons à vous en servir également pour rendre votre code plus clair et plus concis.