Résumé Capsule 7.1: La surcharge des opérateurs

- Vrai ou faux? Par défaut, l'opérateur ++ est déjà défini et incrémente de 1 l'élément sur lequel il s'applique. Donc si l'objet A_ de la classe A existe, je peux faire A_++.

- Le compilateur cherche 2 façons de voir un opérateur. Par exemple, si j'écris Base += 5; il cherche soit :

  1. Base.operator+=(5); ou
  2.

- this est un pointeur qui pointe vers

- Certains opérateurs retournent une référence de l'objet courant pour permettre de faire des opérations

Aide-mémoire

Type opérateur		
+= Définie dans la classe ou globale	Pour faire : Base += 5 Base& Base::operator+=( );	Pour faire : Base += Base2 Base& Base::operator+=( );
*= Définie dans la classe ou globale	Pour faire : Base *= 5 Base& Base::operator*=( );	Pour faire : Base *= Base2 Base& Base::operator*=( );
+, *, /, -, % Définie globale à l'extérieur de la classe	Pour faire : Base1 + 5 CBase operator+(, int valeur); La surcharge des opérateurs +, *, /, - et % doit être écrite de façon globale et non dans une méthode de façon à permettre la commutativité	Pour faire : 5 + Base1 CBase operator+(, const CBase& Base);
++, – Définie dans la classe	Pour faire : Base1++ ou ++Base1, Base1-- ou --Base1 Forme préfixée (++Base1) : CBase & Cbase::operator++(); //Plus rapide à exécuter que l'autre forme Forme suffixée (Base1++) : CBase Cbase::operator++(int); Même chose pour l'opérateur --	

Aide-mémoire

<table>
<tr>
<td>>, <, ≥, ≤, ==,!=

Définie dans la classe ou globale</td>
<td>Pour faire : Base1 < Base2, Base1 > Base2, etc.

  Cbase::operator>(const CBase &Base) const;

Même style pour les autres opérateurs relationnels

Pour opérateur !=

return !(*this == Base);</td>
</tr>
<tr>
<td>=

Définie dans la classe</td>
<td>Pour faire : Base2 = Base1

CBase& CBase::operator=(const CBase &Base);

Pour empêcher : Base1 = Base1 pour des objets contenant des pointeurs

if (this != &Base)

{

SetEntier(Base.GetEntier());

}</td>
</tr>
<tr>
<td><< et >>

Déclarée seulement globalement</td>
<td>Pour utiliser <<:

ostream & operator<<( , );

Pour utiliser >>:</td>
</tr>
</table>

- On ne peut pas surcharger les opérateurs suivants :