Résumé: Encapsulation d'un tableau d'entier et trinité

Lors de l'encapsulation d'un tableau d'entier, je vais créer mon tableau (avec l'opérateur new X[Taille]) dans quelle(s) fonction(s) de ma classe?

Et où vais-je placer mon « delete [] »? :

Si j'ai une classe CTabEntiers construite comme ceci :

```
CTabEntiers::CTabEntiers(int Taille) {
    Table_ = new int[Taille];
    Taille_ = Taille;
}
CTabEntiers::~CTabEntiers() {
   delete [] Table_;
}
void CTabEntiers::SetElement(int Index, int Entier)
{
   Table_[Index] = Entier;
}
int CTabEntiers::GetElement(int Index) const
{
   return Table_[Index];
}
```

Que se passe-t-il lorsqu'on passe un objet de la classe CTabEntiers en paramètre par valeur et pourquoi?

Que se passe-t-il lorsqu'on construit un objet de la classe CTabEntiers à partir d'un autre et pourquoi?

Ceci nous amène donc au concept de la Trinité.

La Trinité consiste à redéfinir ces trois services lorsqu'on a un pointeur comme attribut de notre classe :

-
-
-