

# LE LANGAGE C++ MASTER I LA SURCHARGE D'OPÉRATEUR

Jean-Baptiste.Yunes@univ-paris-diderot.fr  
U.F.R. d'Informatique  
Université Paris Diderot - Paris 7

10 nov. 2012





# GÉNÉRALITÉS

- Les opérateurs surchargeables sont :

		<i>type</i>		



- Ne peuvent être surchargés :

- l'opérateur de portée

- la sélection de membre

- la sélection *via* pointeur

- l'opérateur conditionnel

- le calcul de taille

- la réflexion

# TECHNIQUE OPÉRATEUR COMME FONCTION MEMBRE (MÉTHODE D'INSTANCE)

- But : donner un sens à des expressions utilisant un ou des opérateurs du C++ qui combinent des valeurs dont au moins l'un des types est défini par l'utilisateur

Autoriser l'écriture d'expression en s'exprimant dans le langage du domaine

- 

-



- En première approximation

peut se lire

- Ainsi l'opérateur peut être vu comme une méthode de la classe de l'objet

- C'est la première technique de surcharge des opérateurs

Les opérateurs vus comme méthode d'instance

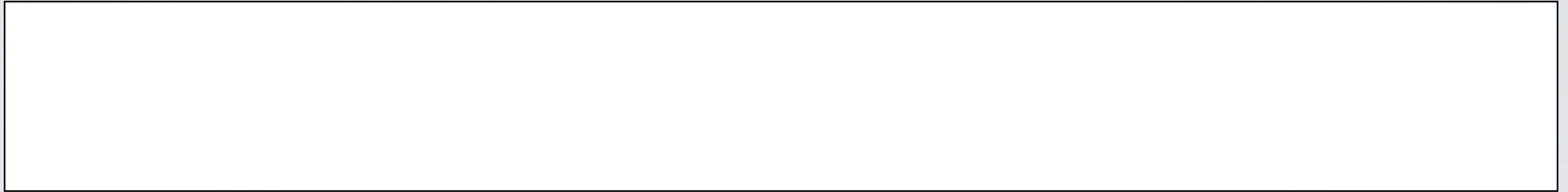
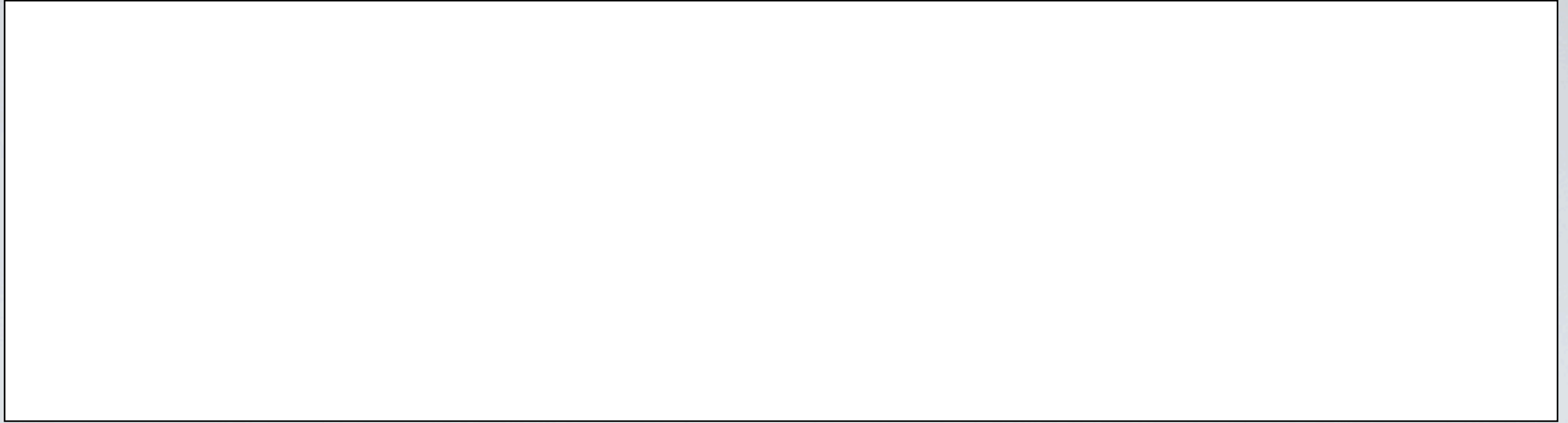
- La convention est que la méthode surchargeant l'opérateur *op* s'appelle *operatorop*, ainsi :

est traduit en

- Remarque : les **de** écritures sont valides



- Ainsi :



- C'est bien entendu un idéal, il existe des exceptions à cette *traduction* littérale.

- D'abord, certains opérateurs ne peuvent être surchargés (  $\text{op}$  et les  $\text{variable}$  ), donc tous les autres peuvent l'être...

- Les opérateurs unaires préfixes :  $\text{op}$  ,  $\text{op}$  ,  $\text{op}$  ,  $\text{op}$  , etc.  
 Pour lesquels la *traduction* de  
 $\text{op variable}$   
 est  
 $\text{variable op}$

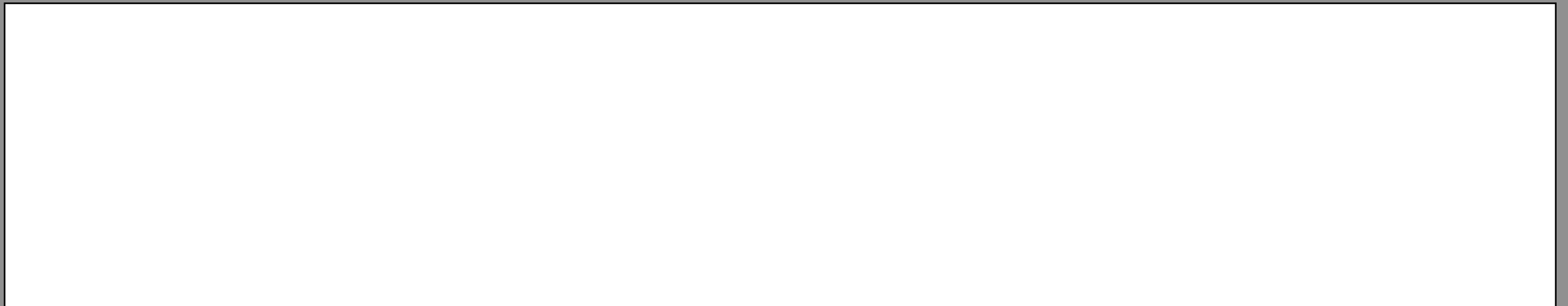
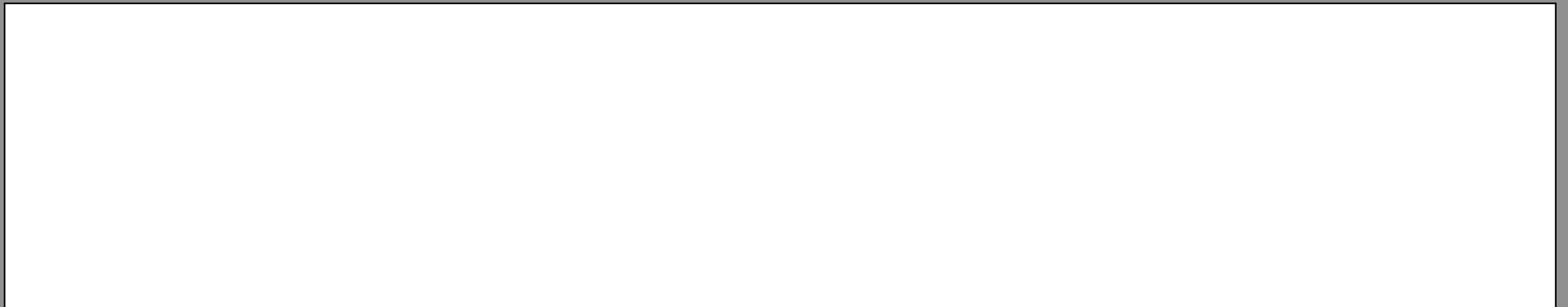
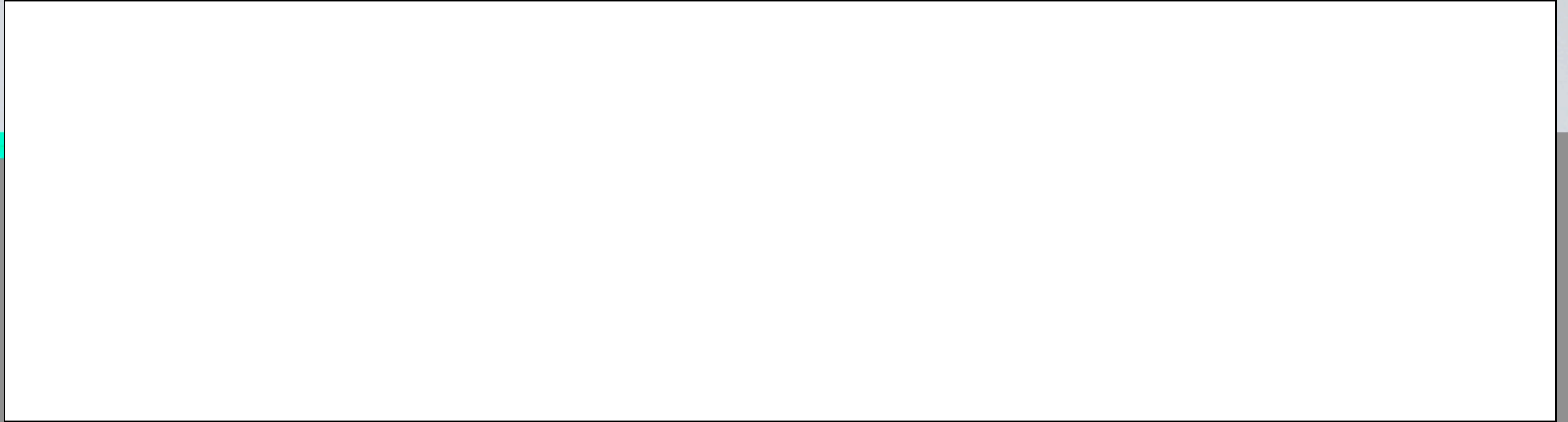


- Les opérateurs parenthésés : `arr[ ]` ,
- dans le cas des crochets, l'argument est habituellement de type `int` et correspond à l'*indice*

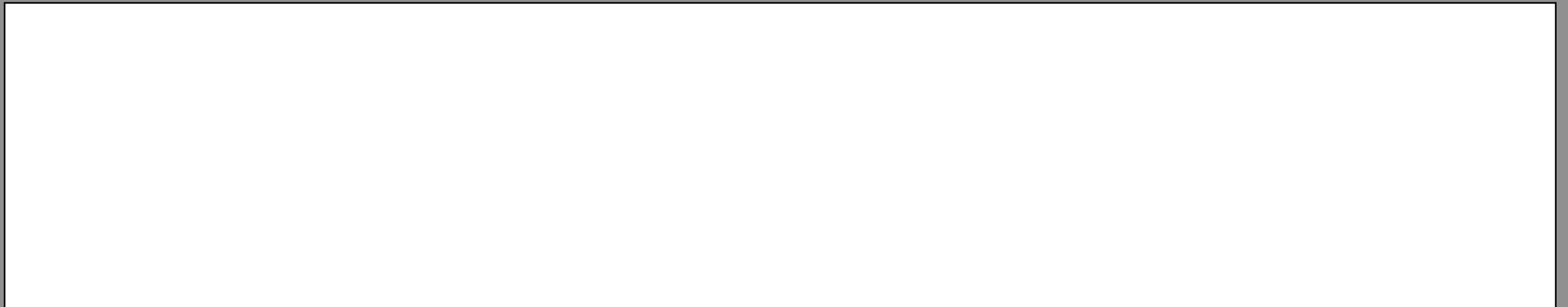
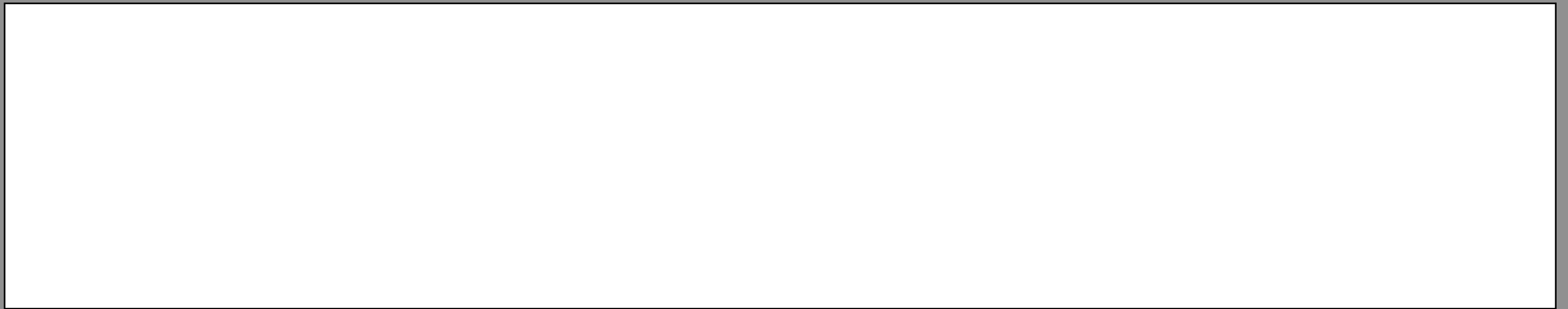
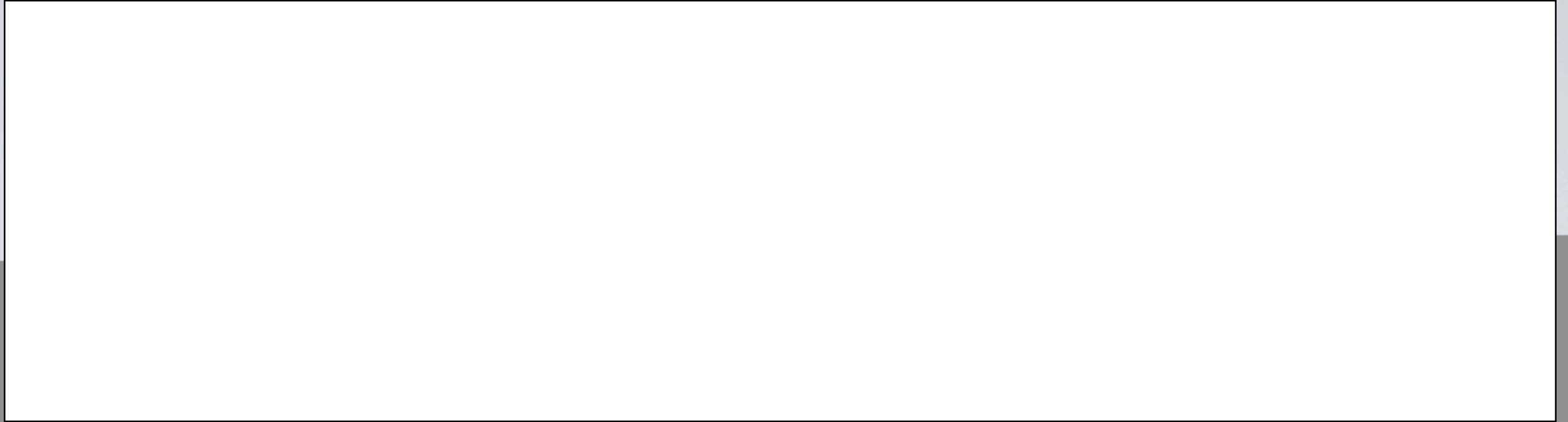
se traduit par

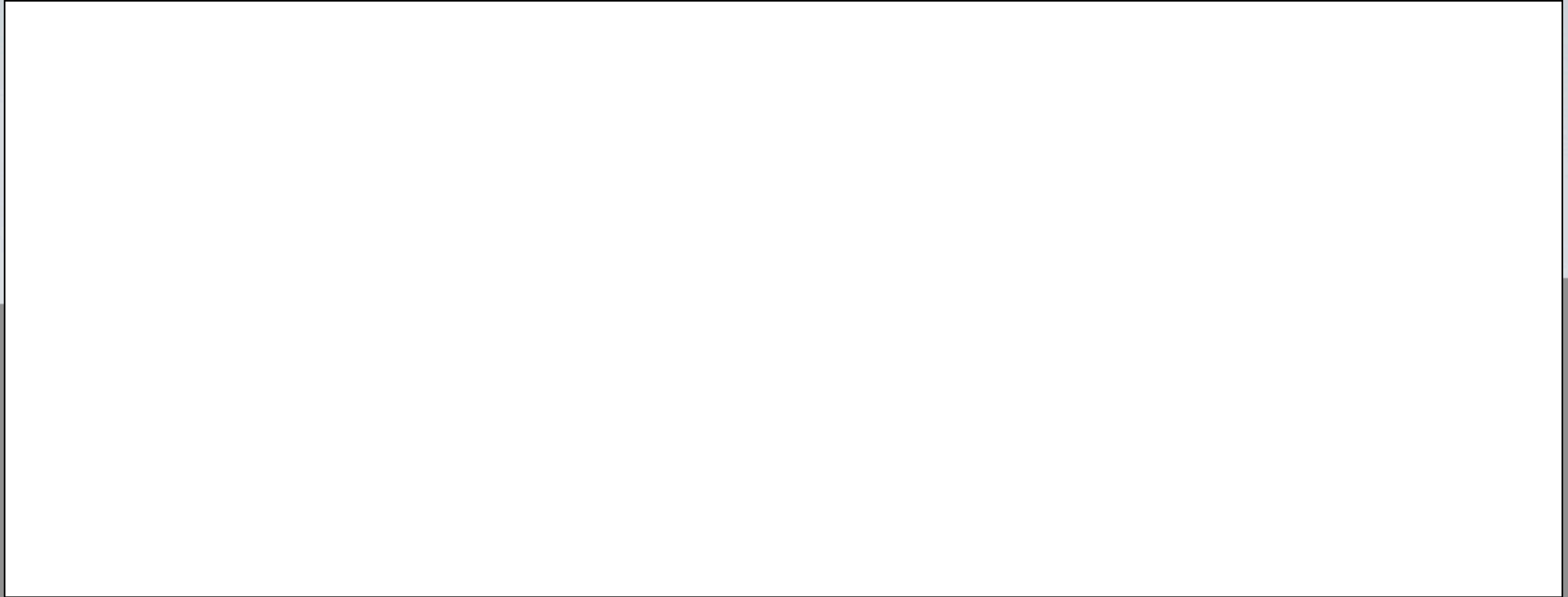
- Note : on peut surcharger cet opérateur avec différents types pour l'*indice*

- Ainsi :



- Ainsi :





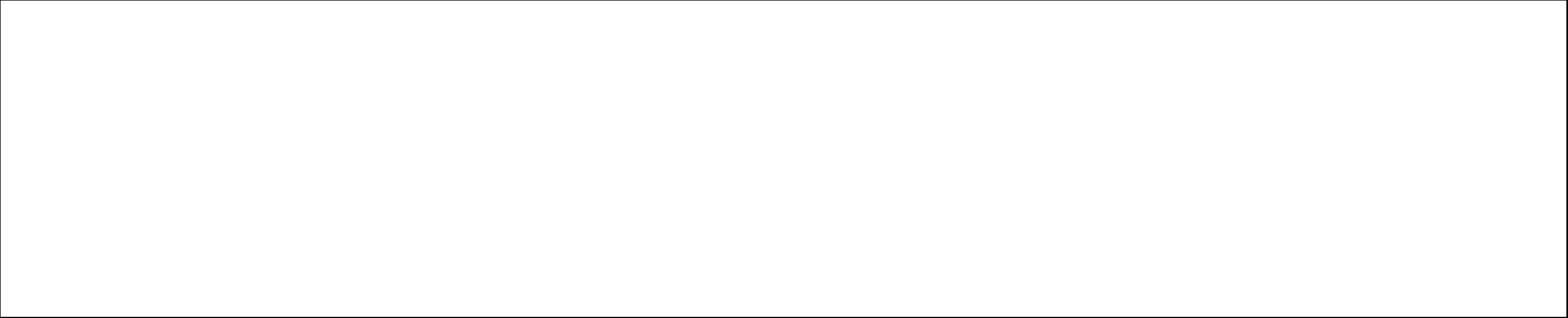
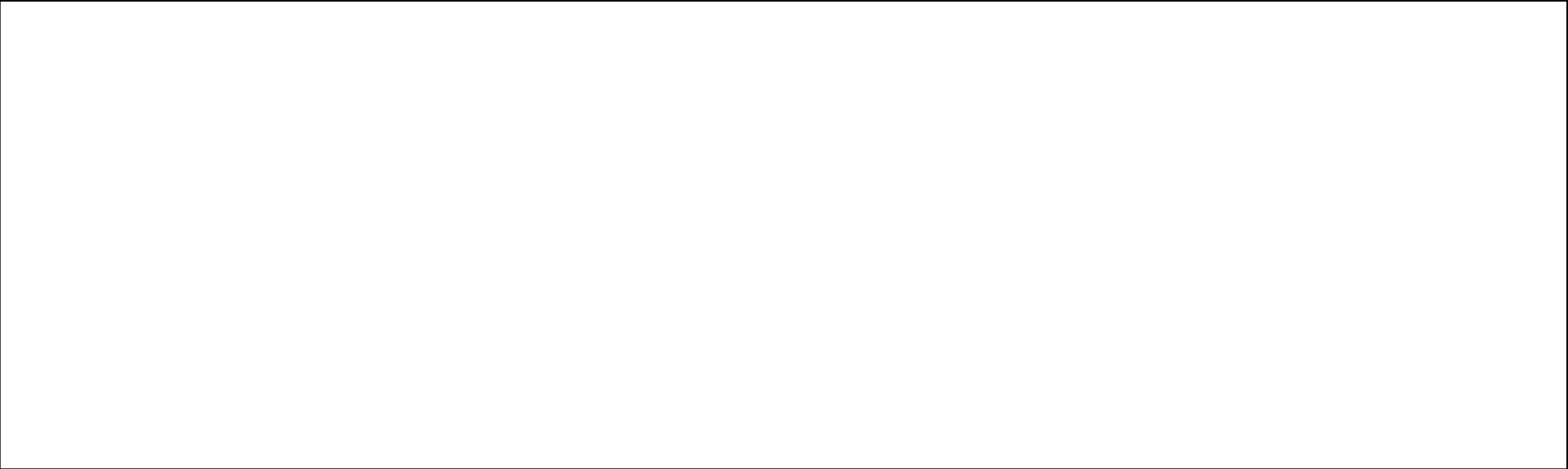




- dans le cas des parenthèses, on peut obtenir la surcharge avec un nombre d'arguments quelconque :

,

,



- L'opérateur de conversion :
  - Une conversion de la forme (à la C)

*type*

- peut s'écrire en notation fonctionnelle (à la C++)

*type*

- et s'implémente comme une fonction membre de la classe de l'expression

*type*



- On peut alors remarquer qu'un constructeur joue aussi (dans certains cas) le rôle d'un opérateur de conversion :
- un constructeur à un argument permet de créer à partir d'une expression d'un type donné un objet d'un autre type
- c'est si proche de la définition d'une conversion que le C++ considère que c'est effectivement une conversion

Conversion de    en  
par appel à







# TECHNIQUE OPÉRATEUR COMME FONCTION (ORDINAIRE)

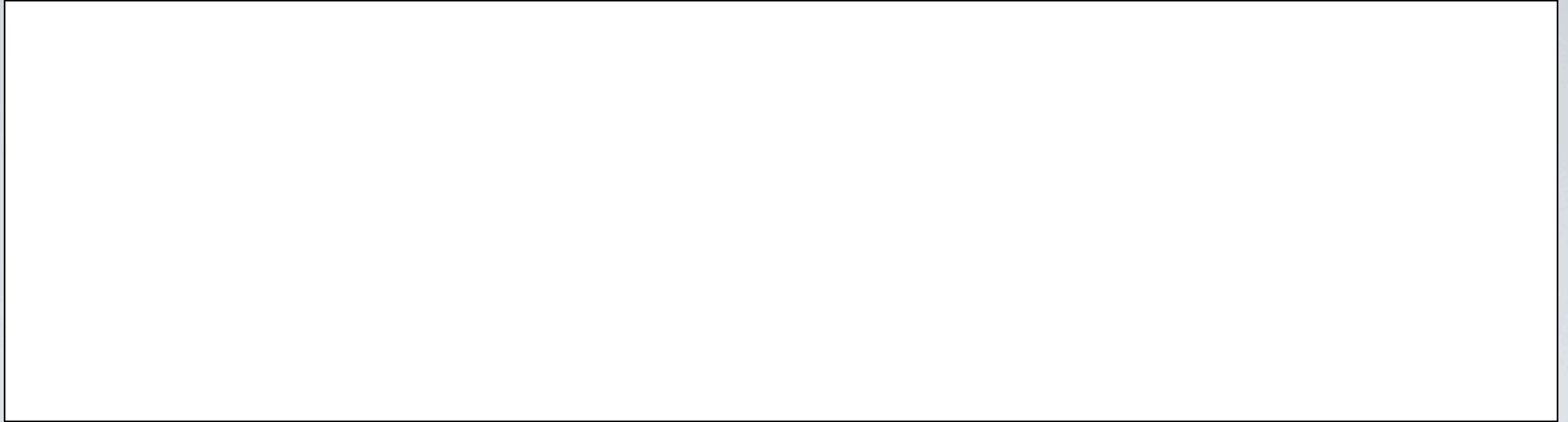

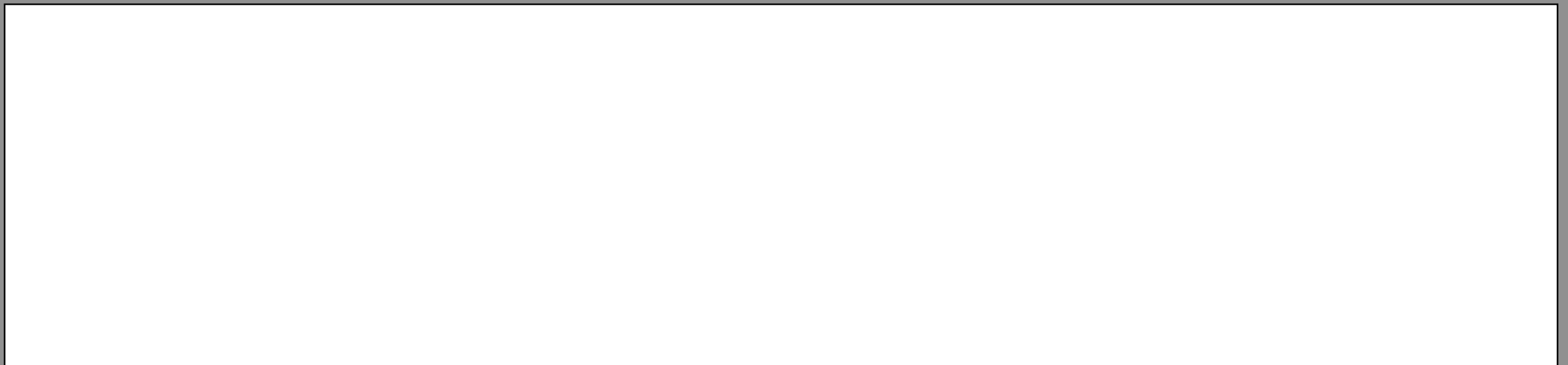



- Il existe des cas dans lesquels la technique *opérateur comme méthode de classe* se révèle impossible par exemple lorsque l'opérande gauche (ou principale) est d'un type dont on ne contrôle pas la définition, *i.e.* type primitif (même pas une classe) ou type classe mais donné sous forme de bibliothèque ou dont on ne contrôle pas le code source

## Les opérateurs vus comme fonctions statiques

- En ce cas, il suffit de considérer, par exemple, que  
se lit
- Remarque : l'addition est bien une fonction à deux opérandes (arité 2)

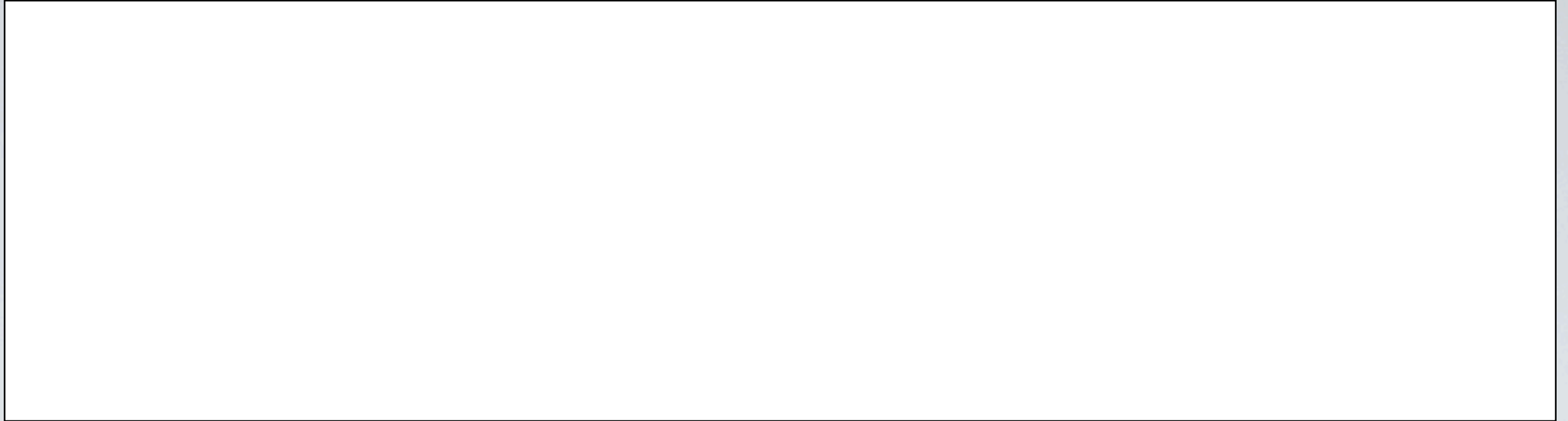
- Ainsi :

A large, empty rectangular box with a black border, intended for a diagram or drawing.A large, empty rectangular box with a black border, intended for a diagram or drawing.A large, empty rectangular box with a black border, intended for a diagram or drawing.

- 
- L'usage premier et le plus courant de la surcharge concerne les entrées/sorties. Il s'agit d'uniformiser les affichages des objets avec ceux des types primitifs...

- Tout d'abord, l'affichage de *base*

- La classe de est :

A large, empty rectangular box with a black border, intended for handwritten notes or a diagram.

- Attention : le flux est modifié donc pas ...

A large, empty rectangular box with a black border, intended for handwritten notes or a diagram.A large, empty rectangular box with a black border, intended for handwritten notes or a diagram.

- Le problème est que

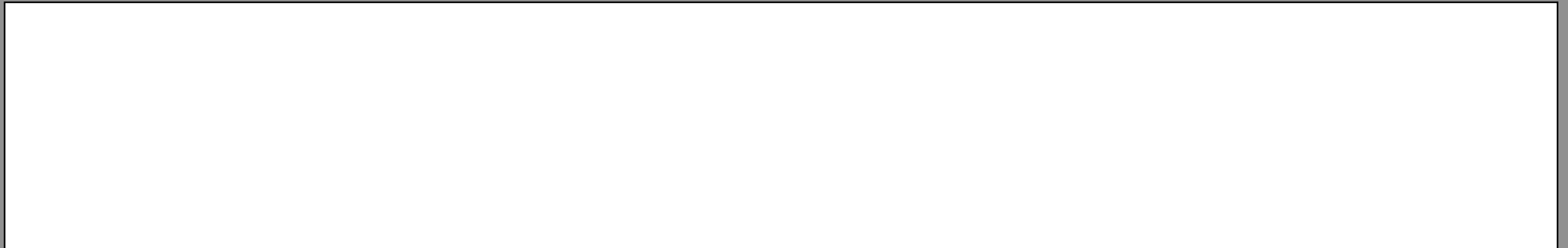
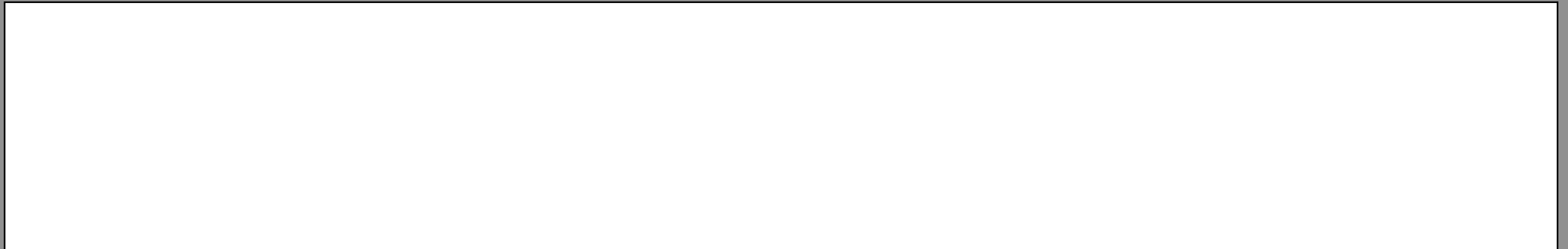
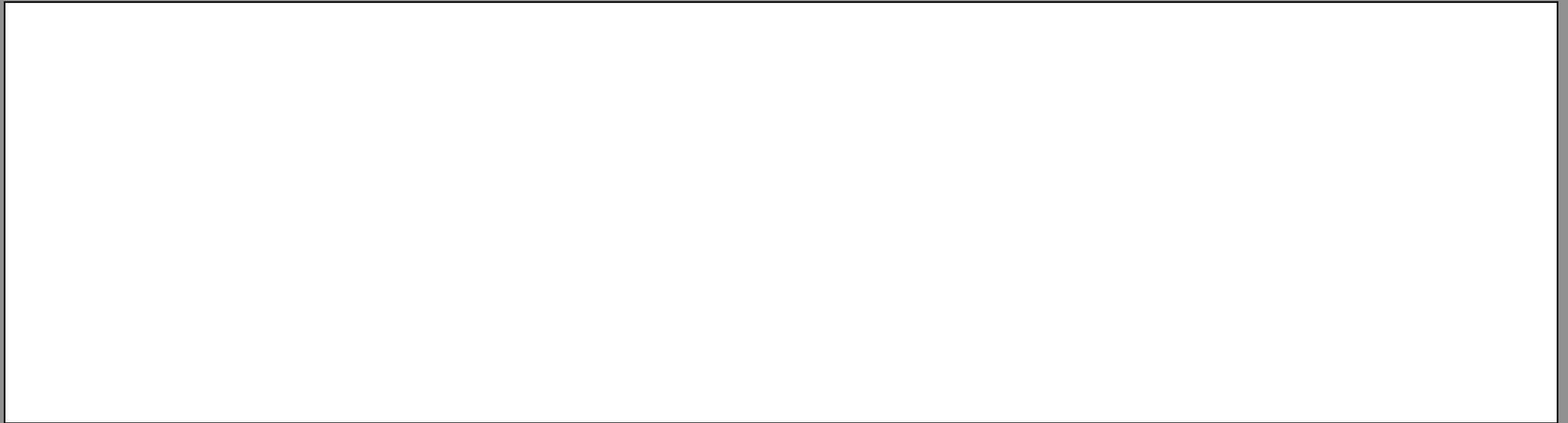
fonctionne mais pas

- Rappel : l'opérateur est associatif à gauche, donc l'expression

se traduit par

- il faut donc renvoyer une valeur!

- Il faut donc renvoyer le flux lui-même et par référence! :



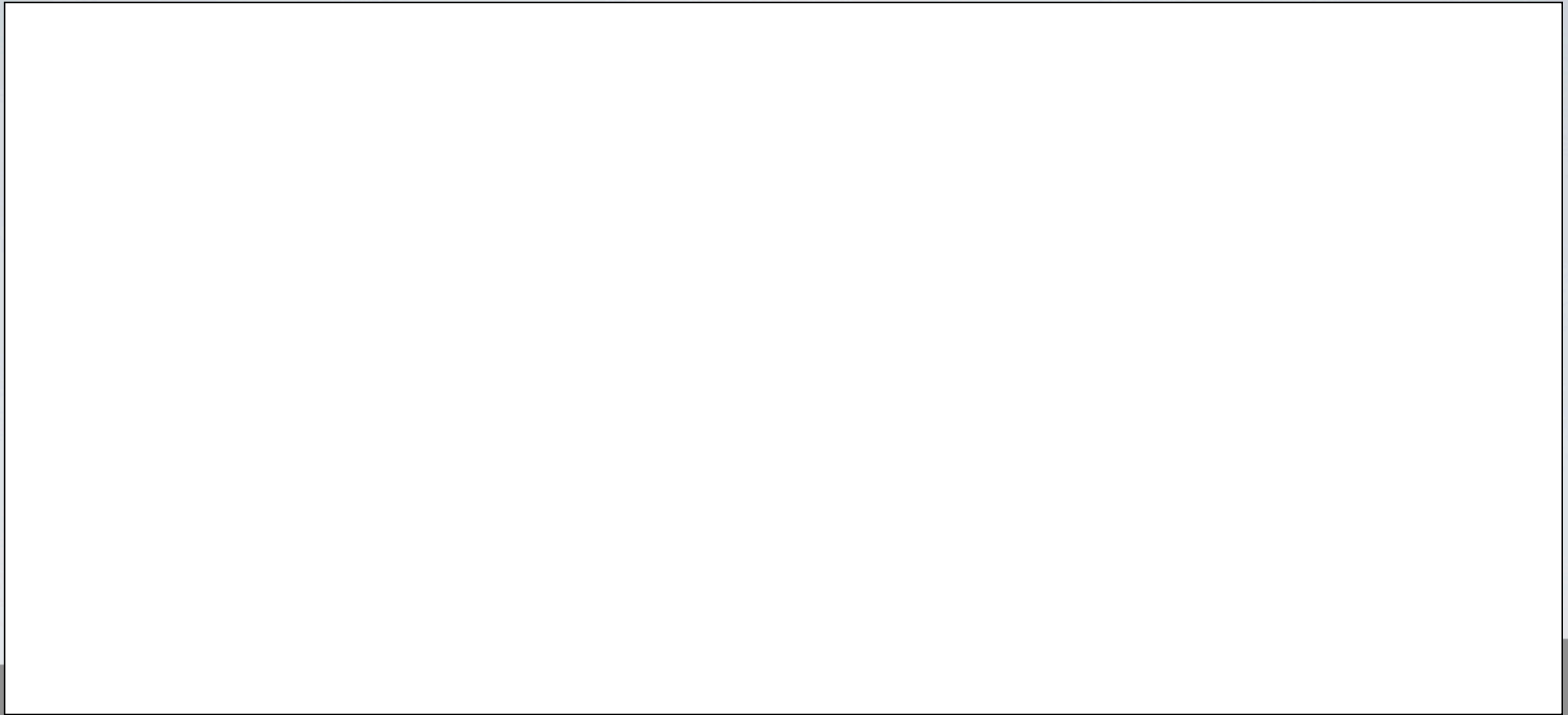
- Attention, la technique fonction (ordinaire) ne peut être employée
- ni pour les opérateurs parenthésés,
- ni pour les opérateurs d'affectation...





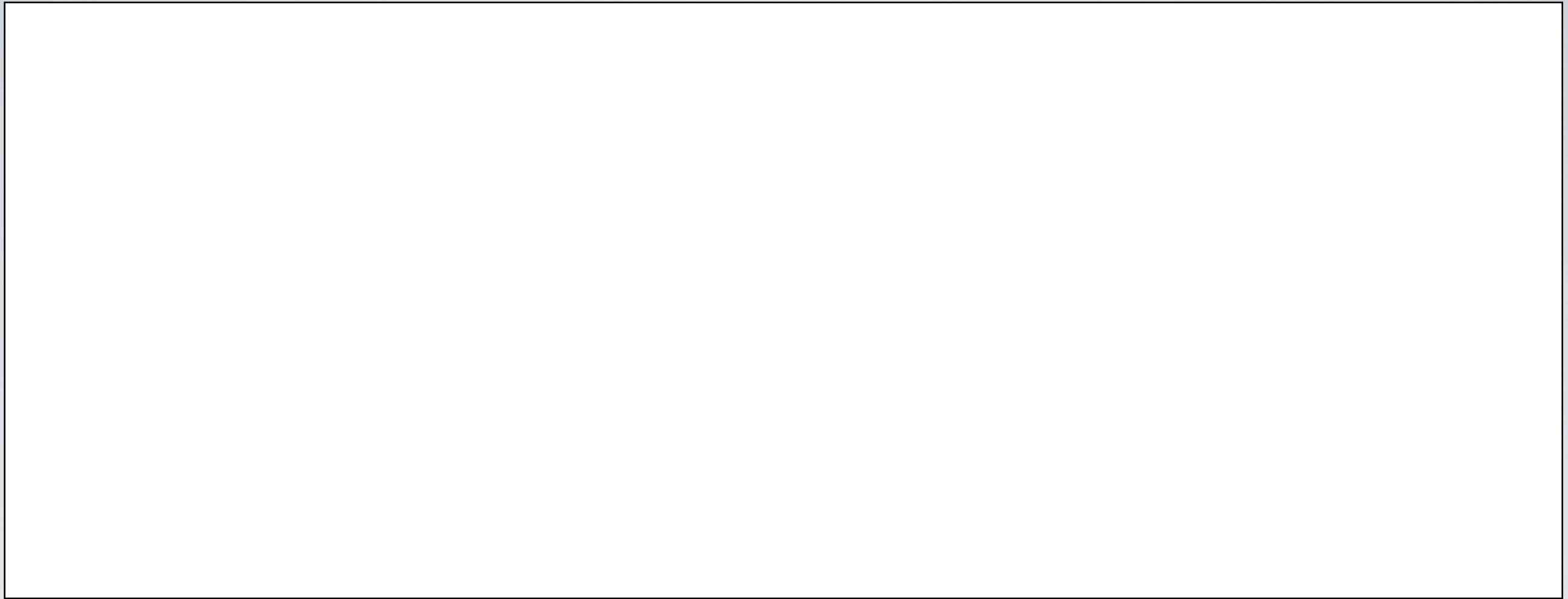
# DES SURCHARGES D'OPÉRATEURS ÉTRANGES...

- l'opérateur ,



- les opérateurs  $\rightarrow$  et  $*$

```
class A;
class PointerToA {
private:
    A *a;
    PointerToA(A *a) { this->a = a; }
public:
    PointerToA() { this->a = 0; }
    A &operator*() { return *a; }
    A *operator->() { return a; }
    A *operator=(A *a) { this->a = a; return a; }
    friend class A;
};
```



DES SURCHARGES VRAIMENT  
SPÉCIALES...



- et :

- l'idée est de contrôler finement les allocations et désallocations d'objets

- usages courants :

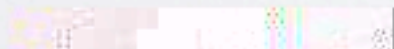
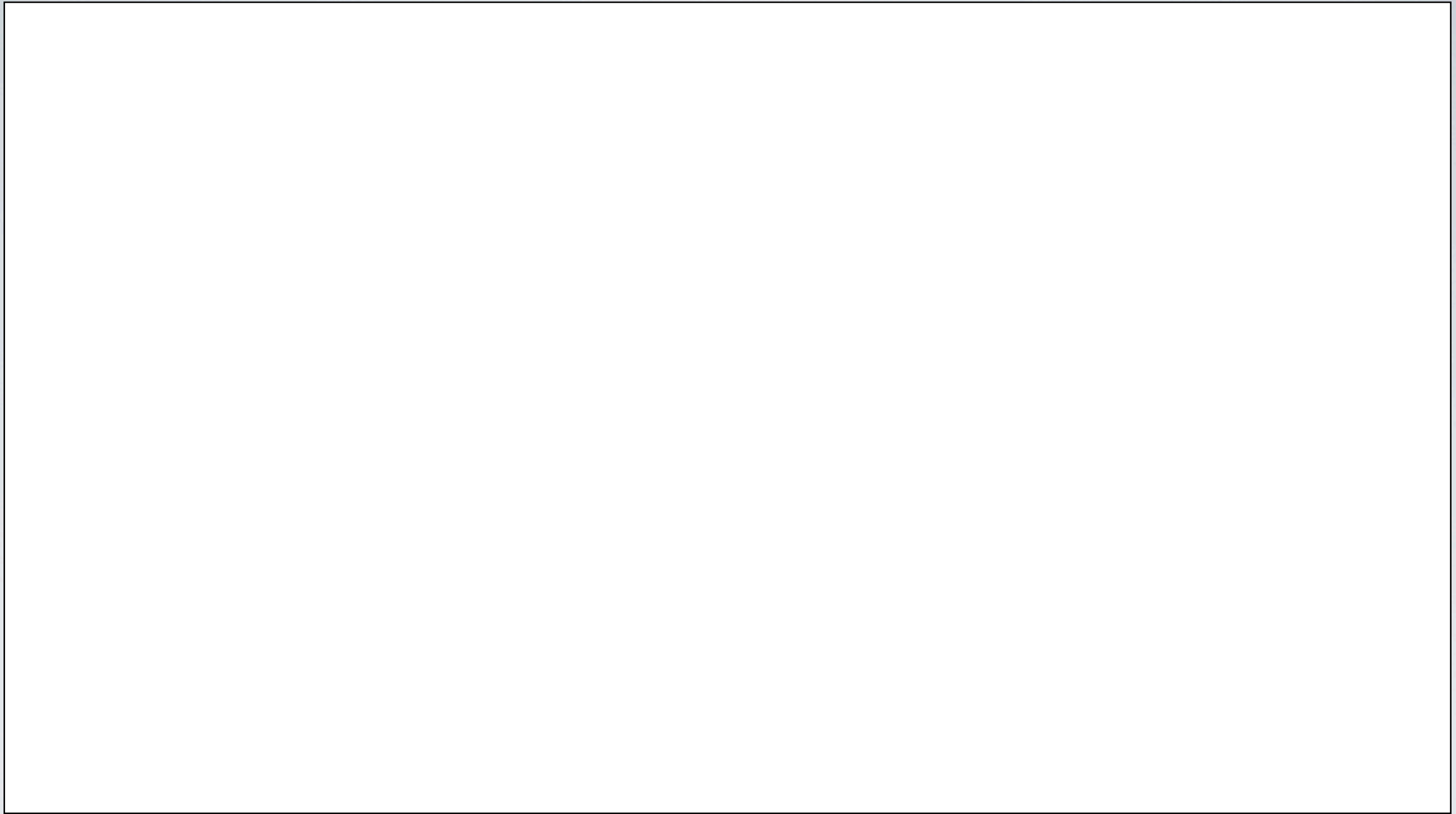
- *pool* d'objets pré-alloués

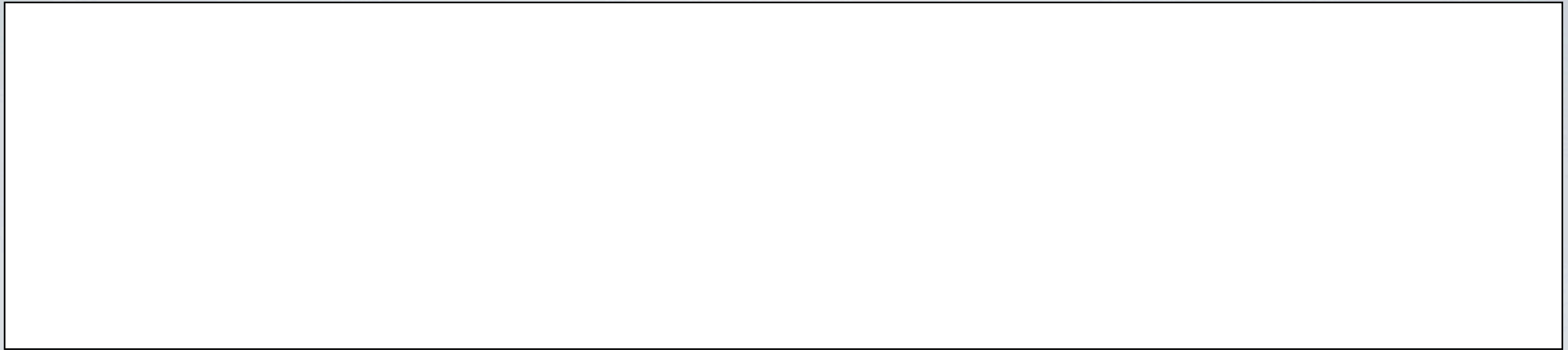
- singleton





- il est possible d'ajouter des arguments au **new**
- pour en obtenir des variantes...





- si besoin est on peut utiliser l'opérateur `global` qui est disponible par
- l'opérateur `new` peut être redéfini :
- en ce cas, l'allocation doit être réalisée *in fine* par le système... sinon récursion sans fin
- attention aussi aux appels de constructeurs
  - une instruction new : un appel à **new** + ctor!!!



# LES PIÈGES DE LA SURCHARGE

- Les pièges de la surcharge proviennent de l'usage habituellement fait des opérateurs et qui sous-entendent un comportement attendu :
  - Propriétés attendues : symétrie, commutativité, etc.
  - Usage dans des expressions multi-opérateurs
  - Le sens donné à l'opérateur doit être *naturel*
- Donc, il ne faut surcharger que si la sémantique existe dans le domaine du problème



- Symétrie/Commutativité : Soient  $A$  et  $B$
- Si  $A$  est de type  $n \times m$ , il serait bien surprenant que  $B$  n'existe pas et ne soit pas aussi de type  $m \times n$  ...

- ex :

- contre-ex : produit de matrices...

appel de l'autre opérateur...

La définition des autres opérateurs se fait à l'aide des précédents

- Multi-opérateurs :
  - L'exemple a déjà été donné pour les opérateurs d'entrées/sorties et
  - Pour les opérateurs pour lesquels la sémantique est la construction d'une nouvelle valeur, il faut renvoyer une valeur du type, sinon renvoyer une référence éventuellement constante
  - Attention aux priorités et aux associativités...

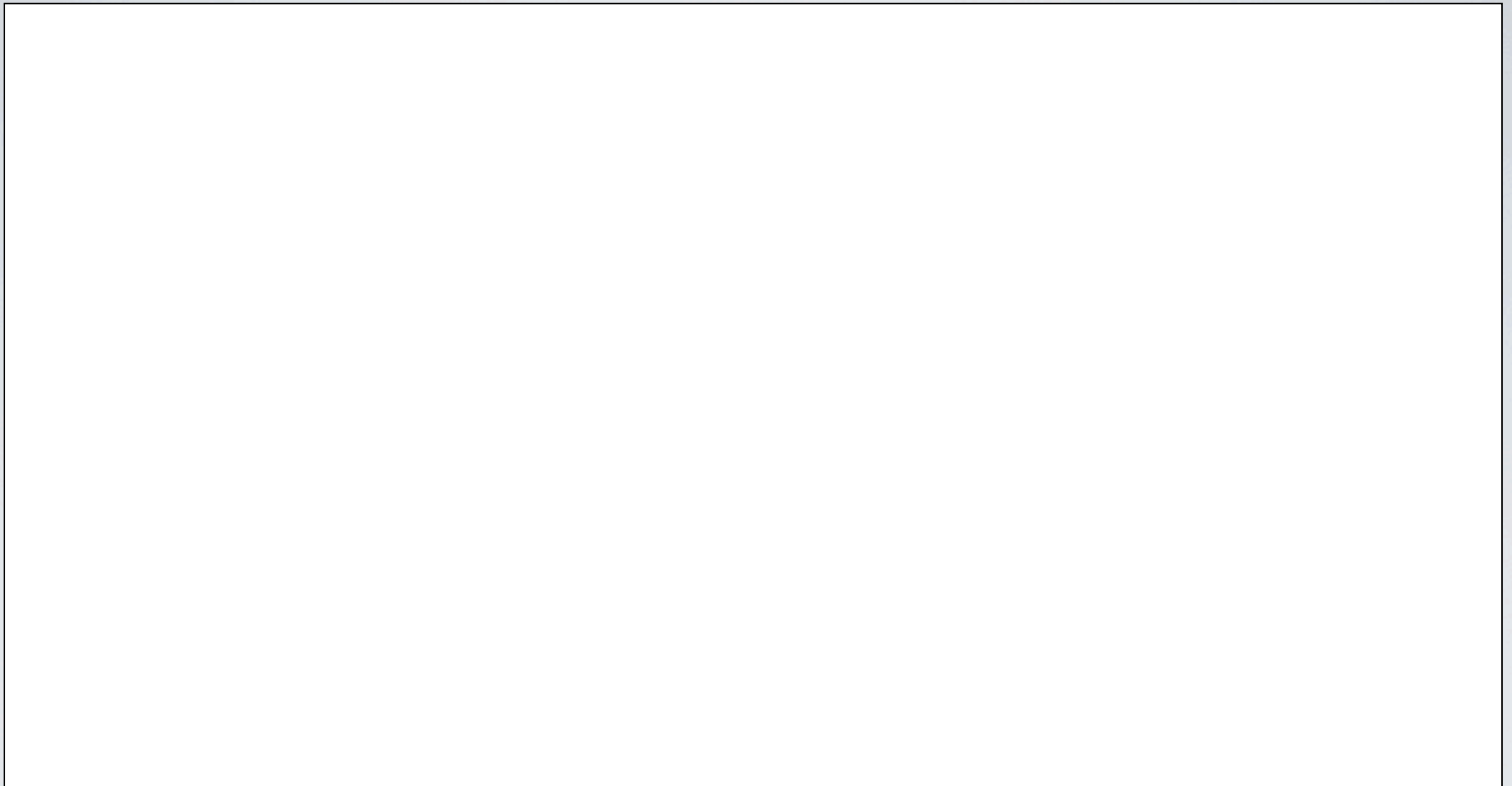
- L'opérateur d'affectation...

- Sa définition peut être **absolument** nécessaire

C'est la triplète infernale :

- Constructeur par copie
- Destructeur
- Opérateur d'affectation

- L'opérateur d'affectation... définition naïve...



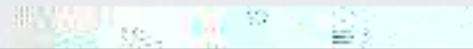
- Problèmes : optimisation (pas trop grave !?), auto-affectation (problématique), associativité (problématique)

- Auto-affectation... Il n'est pas interdit d'écrire .  
Moi ? Jamais! Ah ?

- Associativité... Il n'est pas interdit d'écrire qui se lit , ainsi
  - le type de retour doit être celui de l'objet appelant
  - le retour peut se faire par référence constante pour éviter une copie...



- Optimisation... Éviter une allocation dynamique lorsque le tableau est déjà de la bonne dimension...

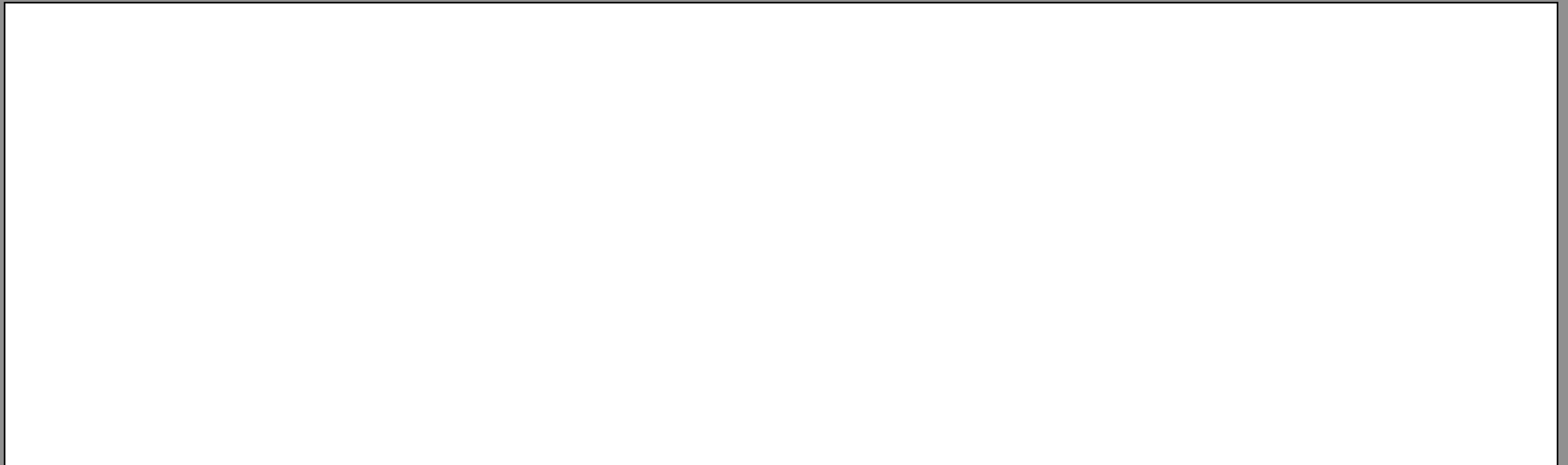




- Sémantique *naturelle* :

- ex : addition de nombres, incrémentation d'un itérateur ou d'un compteur, etc.
- ex : si  $\text{fct}$  est défini et  $\text{fct}$  aussi alors l'utilisateur s'attendra à ce que  $\text{fct}$  le soit et fonctionne bien (sur les valeurs produites) comme  $\text{fct}$  ...
- contre-ex : additionner une montre et un cheval pour fabriquer une liste ??? Pour un informaticien peut-être mais sinon c'est louche...

- Puisque les constructeurs peuvent être employés pour créer automatiquement des objets (dans le cadre de conversions), il peut être utile d'empêcher cela :
- en obligeant l'utilisateur à appeler lui-même explicitement le constructeur qui doit être qualifié par :





# DÉFINITION CANONIQUE D'UNE CLASSE

- Nous avons maintenant tout en main pour définir la forme canonique d'une classe :

pour les tableaux

pour les  
destructions  
polymorphes

pour les copies

pour les entrées/sorties

pour les affectations