

Previously On...

```
int main() {  
    Tab t1(5);  
    Tab t2(3);  
    t2 = t1;    // ???  
    return 0;  
}
```

**Il a dit "initialisation",  
mais si c'est une...  
"affectation"**





# WHAT IF ... ?

```
Tab t1(5), t2(10), t3;
```

```
t3 = t1 + t2;    // Concaténation de tableau ?
```

```
if (t1 == t2) ... // Comparaison de tableau
```

```
t1[3] = 42       // Affectation de valeur directe
```

```
cout << t1;      // Affichage d'un tableau
```





# Surdéfinition d'opérateurs

# Comment ça marche ?

Définition :

```
bool Tab::operator == (const Tab & t) {  
    // Comparaison des éléments un à un  
}
```

```
Tab & Tab::operator = (const Tab & t) {  
    // Même travail que dans le constructeur de recopie  
}
```

# Comment ça marche ?

Définition :

```
int & Tab::operator [] (int i) {  
    // Renvoyer l'élément  
}
```

```
Tab Tab::operator + (Tab & t) {  
    // Concaténer les tableaux  
}
```

# Quelques règles

Il est possible de surdéfinir quasiment tous les opérateurs du langage :

+	-	*	/	%	^	&		~
!	=	<	>	+=	-=	*=	/=	%=
^=	&=	=	<<	>>	<<=	>>=	==	!=
<=	>=	&&		++	--	,	->*	->
()	[]	new	delete	new[]	delete[]			

<https://isocpp.org/wiki/faq/operator-overloading#overload-dot>



# Quelle est le prototype de la surdéfinition ?

CPP Référence : operators overloading !

<https://en.cppreference.com/w/cpp/language/operators>

cppreference.com

Create account

Search

Page

Discussion

View


Edit

History

C++

C++ language

Expressions



Your new development career awaits. Check out the latest listings.

## operator overloading

Customizes the C++ operators for operands of user-defined types.

### Syntax

Overloaded operators are [functions](#) with special function names:

<b>operator</b> <i>op</i>	(1)
<b>operator</b> <i>type</i>	(2)