LES OPERATEURS

Affectation

Réalisée par l'opérateur =

```
int a;
int b;

a = 20;
b = a;
a = 4;
```

L'affectation est réalisée par copie de la valeur.

LES OPERATEURS

Opérateurs Arithmétiques

Opérateur	Nom	Exemple
+	Addition	a = b + 4
-	Soustraction	a = b - c
*	Multiplication	a = b * 8
/	Division	a = b / c
%	Modulo	a = b % 2

LES OPERATEURS

Opérateurs Arithmétiques

l'Opérateur modulo (%) calcule et retourne le reste de la division entière du premier opérande par le second.

Exemple:

```
int a =5;
int b = 2;
int c;
c = a % b;  // 5 divisé par 2 : résultat entier = 2  reste= 1
```

c aura la valeur 1

LES OPERATEURS

Affectation représentation simplifiée

Opérateur	Exemple
a = a + 2	a += 2
b = b – 4	b -= 4
a = a / 2	a /= 2
a = a * 2	a *= 2
a = a % 3	a %=3

LES OPERATEURS

Incrémentation - Décrémentation

L'incrémentation est l'ajout d'une unité.

	Opérateur		Exemple
Incrémentation	++	a++	post-incrémentation
		++a	pré-incrémentation
Décrémentation	--	a--	post-incrémentation
		--a	pré-incrémentation

Si l'opérateur est placé avant la variable , la modification de la valeur est immédiate sinon la modification n'a lieu qu'à l'issue de l'exécution de la ligne d'instruction.

LES OPERATEURS
Priorité des opérateurs

Une expression complexe est évaluée en fonction de la priorité des opérateurs.

Exemples:

```
y = a * x * x + b * x + c;

y = a++ + c / z – d / 2 % f;
```

Il est souhaitable de rajouter des parenthèses pour une meilleure lisibilité et pour éviter des erreurs.

On écrira:

```
y = (a * x * x) + ( b * x ) + c;

y = ((a++ + c) / z) – ((d / 2) % f);
```

LES OPERATEURS
Priorité des opérateurs

	Opérateur
Parenthèses	( )
Incrémentation Décrémentation	++ --
Multiplication Division Modulo	* / %
Addition Soustraction	+ -
Décalage	<< >>
Comparaison	< <= > >=
Egalite	== !=
OU Exclusif	^
ET	&
OU	|
ET logique	&&
OU logique	||
Affectation	= += -= *= /= %=