C - Opérations mathématiques

En programmation, les opérations mathématiques permettent de faire des calculs avec des variables : additionner, soustraire, multiplier, diviser, ou encore obtenir le reste d'une division. Le langage C fonctionne comme les mathématiques classiques, mais il faut faire attention aux types de données (entiers, réels) et à la façon dont la division est calculée.

5TTR 6TTR Découverte

Parfait 👍 Voici un **article de cours complet** sur les **opérations mathématiques en langage C**, rédigé selon ta méthodologie (enseignement explicite, format pour élèves + notes enseignant).

Les opérations mathématiques en C

Objectifs

À la fin de ce chapitre, tu sauras :

- Utiliser les opérateurs arithmétiques de base (`+`, `-`, `*`, `/`, `%`).
- Comprendre la différence entre division entière et division réelle.
- Combiner plusieurs opérations dans une même expression.
- Utiliser les parenthèses pour contrôler l'ordre des calculs.
- Afficher les résultats avec `printf`.

Les opérateurs de base

En C, les opérations mathématiques se font à l'aide d'**opérateurs** :

Opérateur	Signification	Exemple	Résultat
`+`	addition	`5 + 2`	`7`
`-`	soustraction	`5 - 2`	`3`
`*`	multiplication	`5 * 2`	`10`
`/`	division	`5 / 2`	`2` (division entière)
`%`	reste de la division	`5 % 2`	`1`

Exemple simple

```c
#include <stdio.h>

int main() {
    int a = 10, b = 3;

    printf("Addition : %d\n", a + b);
    printf("Soustraction : %d\n", a - b);
    printf("Multiplication : %d\n", a * b);
    printf("Division entière : %d\n", a / b);
    printf("Reste (modulo) : %d\n", a % b);

    return 0;
}
```

Résultat

```
Addition : 13
Soustraction : 7
Multiplication : 30
Division entière : 3
Reste (modulo) : 1
```

Division entière vs réelle

⚠️ En C, si tu divises **deux entiers**, le résultat est **un entier** :

```c
int a = 7, b = 2;
printf("%d", a / b);  // Affiche 3 (et pas 3.5)
```

Pour obtenir une **division réelle**, il faut utiliser des `float` ou `double` :

```c
float x = 7, y = 2;
printf("%.2f", x / y);  // Affiche 3.50
```

💡 `%.2f` affiche le résultat **arrondi** à **2 chiffres après la virgule**.

L'ordre des opérations

Comme en mathématiques, le C suit les priorités habituelles :

1. Parenthèses `( )`

2. Multiplication et division `*`, `/`, `%`
3. Addition et soustraction `+`, `-`

Exemple :

```c
int resultat = 5 + 2 * 3;      // = 11
int resultat2 = (5 + 2) * 3;   // = 21
```

Opérations combinées

Tu peux évidemment faire plusieurs calculs dans la même expression :

```c
int a = 4, b = 2, c = 3;
int r = (a + b) * c - 1;
printf("Résultat : %d\n", r);
```

Affectation et raccourcis utiles

Il existe des **formes abrégées** pour modifier la valeur d'une variable :

Forme classique	Raccourci	Signification
`x = x + 3;`	`x += 3;`	ajoute 3 à x
`x = x - 4;`	`x -= 4;`	retire 4 à x
`x = x * 2;`	`x *= 2;`	multiplie x par 2
`x = x / 3;`	`x /= 3;`	divise x par 3

Incrémentation et décrémentation (`i++` , `++i` , `i--` , `--i`)

En programmation, il est très courant de **modifier la valeur d'une variable numérique** à chaque étape d'un calcul ou d'une boucle. Au lieu d'écrire des formules longues comme `i = i + 1`, le langage C propose des **raccourcis mathématiques** très utilisés : les opérateurs d'**incrémentation** et de **décrémentation**.

Incrémenter et décrémenter une variable

- `i++` → ajoute **1** à la variable `i`
- `++i` → ajoute aussi **1** à la variable `i`
- `i--` → retire **1** à la variable `i`
- `--i` → retire aussi **1** à la variable `i`

Exemples :

```
int i = 5;
i++;  // i devient 6
i--;  // i redevient 5
```

Ces opérateurs sont donc une forme abrégée d'opérations arithmétiques très fréquentes :

- `i++` équivaut à `i = i + 1`
- `i--` équivaut à `i = i - 1`

Différence entre `i++` et `++i`

Ces deux écritures **ont le même effet final** (augmenter `i` de 1), mais **pas au même moment** dans le calcul.

Opérateur	Étape du calcul	Exemple	Valeur utilisée
`i++`	**post-incrémentation** : on utilise `i`, puis on ajoute 1	`x = i++;`	`x` reçoit l'ancienne valeur de `i`
`++i`	**pré-incrémentation** : on ajoute 1, puis on utilise `i`	`x = ++i;`	`x` reçoit la nouvelle valeur de `i`

Exemple concret :

```
int i = 5;
int a = i++;  // a = 5, puis i = 6
int b = ++i;  // i = 7, puis b = 7
```

Même chose pour la décrémentation

De la même façon :

```
int i = 3;
int a = i--;  // a = 3, puis i = 2
int b = --i;  // i = 1, puis b = 1
```

À retenir

- `i++` ou `++i` → ajoutent 1 à `i`

- `i--` ou `--i` → retirent 1 à `i`
- La différence entre préfixe (`++i`) et postfixe (`i++`) n'a d'importance que **dans une expression mathématique**.
- Ces opérateurs sont très utiles pour **faire évoluer une variable** dans une boucle ou un calcul.

Exercices

1. Calcule la somme, la différence et le produit de 8 et 4.
2. Calcule le quotient et le reste de la division de 17 par 5.
3. Affiche le résultat de `(3 + 2) * 5`.
4. Déclare deux `float` et affiche leur moyenne avec 2 décimales.
5. Si `x = 10`, affiche le résultat après `x += 5`, puis après `x--`.

À retenir

- Les opérateurs arithmétiques sont : `+`, `-`, `*`, `/`, `%`.
- Une division entre deux entiers donne un **résultat entier**.
- Utilise des `float` ou `double` pour obtenir des nombres à virgule.
- Les **parenthèses** permettent de choisir l'ordre du calcul.
- Les opérateurs `+=`, `-=`, `++`, `--` sont des raccourcis pratiques.