**Faites des calculs avec des variables**

Dans ce chapitre, on va apprendre à réaliser la plupart des calculs qu'un ordinateur sait faire. Nous réutiliserons ce que nous venons tout juste d'apprendre pour faire des calculs :

* ajouter des variables entre elles ;
* les multiplier ;
* enregistrer le résultat dans une autre variable, etc.

### Revoyez les calculs de base

Votre ordinateur est une calculatrice très basique, puisqu'on ne peut faire que des opérations très simples :

* addition ;
* soustraction ;
* multiplication ;
* division ;
* modulo.

Si vous voulez faire des opérations plus compliquées :

* carrés ;
* puissances ;

il vous faudra les programmer, c'est-à-dire **expliquer à l'ordinateur comment les faire**.

Fort heureusement, nous verrons plus loin dans ce chapitre qu'il existe une bibliothèque mathématique livrée avec le langage C, qui contient des fonctions mathématiques toutes prêtes.

#### L'addition, la soustraction et la multiplication

Pour faire une addition, on utilise le signe  + .

Vous devez mettre le résultat de votre calcul dans une variable. On va donc par exemple créer une variable resultat de type int et faire un calcul :

int resultat = 0;

resultat = 5 + 3;

Bien sûr, rien ne s'affiche à l'écran avec ce code. Si vous voulez voir la valeur de la variable, rajoutez un printf comme vous savez maintenant si bien le faire :

printf("5 + 3 = %d", resultat);

À l'écran, cela donnera :

5 + 3 = 8

Pour les autres opérations, c'est la même chose, seul le signe utilisé change :

| **Opération** | **Signe** |
| --- | --- |
| Addition | + |
| Soustraction | - |
| Multiplication | \* |
| Division | / |
| Modulo | % |

Il n'y a pas de difficulté particulière pour ces opérations, à part pour les deux dernières (la division et le modulo). Nous allons donc parler un peu plus en détail de chacune d'elles.

#### La division

Les divisions fonctionnent normalement sur un ordinateur quand il n'y a pas de "reste" :6 / 3font 2, votre ordinateur vous donnera la réponse juste.

Mais prenons maintenant une division avec reste, comme5 / 2… Le résultat devrait être 2.5. Et pourtant ! Regardez ce que fait ce code :

int resultat = 0;

resultat = 5 / 2;

printf ("5 / 2 = %d", resultat);

5 / 2 = 2

Il y a un gros problème : on a demandé5 / 2, on s'attend à avoir "2.5", et l'ordinateur nous dit que ça fait "2" !

En fait, quand il voit les chiffres 5 et 2, votre ordinateur fait une division de nombres entiers (aussi appelée « division euclidienne »). Cela veut dire qu'il tronque le résultat, il ne garde que la partie entière (le 2).

Hé mais je sais pourquoi ! C'est parce que resultat est un int! Si ça avait été un double, il aurait pu stocker un nombre décimal à l'intérieur, c'est ça ?

Eh non, ce n'est pas la raison ! Essayez le même code en transformant juste resultat endouble, et vous verrez qu'on vous affiche quand même 2. Parce que les nombres de l'opération sont des nombres entiers, l'ordinateur répond par un nombre entier.

Si on veut que l'ordinateur affiche le bon résultat, il va falloir transformer les nombres 5 et 2 de l'opération en nombres décimaux, c'est-à-dire écrire 5.0 et 2.0 (ce sont les mêmes nombres, mais l'ordinateur considère que ce sont des nombres décimaux, donc il fait une division de nombres décimaux) :

double resultat = 0;

resultat = 5.0 / 2.0;

printf ("5 / 2 = %lf", resultat);

5 / 2 = 2.500000

Là, le nombre est correct. Bon : il affiche des tonnes de zéros derrière si ça lui chante, mais le résultat reste quand même correct.

Cette propriété de la division de nombres entiers est très importante. Il faut que vous reteniez que pour un ordinateur : 5 / 2 = 2. C'est un peu surprenant, mais c'est sa façon de calculer avec des entiers.

Si vous voulez avoir un résultat décimal, il faut que les nombres de l'opération soient décimaux : 5.0 / 2.0 = 2.5.

En fait, en faisant une division d'entiers comme5 / 2, votre ordinateur répond à la question « Combien y a-t-il de fois 2 dans le nombre 5 ? ». La réponse est deux fois.

Mais alors comment on fait pour récupérer le reste de la division ?

C'est là que super-modulo intervient.

#### Le modulo

Le modulo est une opération mathématique qui permet d'obtenir **le reste d'une division**.

C'est peut-être une opération moins connue que les quatre autres, mais pour votre ordinateur ça reste une opération de base… probablement pour combler le problème de la « division d'entiers » qu'on vient de voir.

Le modulo se représente par le signe %.

Voici quelques exemples de modulos :

* 5 % 2 = 1 ;
* 14 % 3 = 2 ;
* 4 % 2 = 0.

Le modulo 5 % 2  est le reste de la division 5 / 2, c'est-à-dire 1. L'ordinateur calcule que5 = 2 \* 2 + 1(c'est ce 1, le reste, que le modulo renvoie).

De même, pour 14 % 3 , le calcul est 14 = 3 \* 4 + 2(modulo renvoie le 2).

Enfin, pour 4 % 2, la division tombe juste, il n'y a pas de reste, donc modulo renvoie 0.

Bon, j'ai une bonne nouvelle : on a vu toutes les opérations de base. Finis les cours de maths !

### Faites des calculs entre variables

Maintenant que vous savez faire les cinq opérations de base, on va s'entraîner à faire des calculs entre plusieurs variables.

En effet, rien ne vous empêche de faire :

resultat = nombre1 + nombre2;

Cette ligne fait la somme des variables nombre1 et nombre2, et stocke le résultat dans la variable resultat.

Imaginez un programme qui demande deux nombres à l'utilisateur :

1. Stockez ces deux nombres dans des variables.
2. Faites la somme de ces variables.
3. Stockez le résultat dans une variable appelée resultat.
4. Vous n'avez plus qu'à afficher le résultat du calcul à l'écran.

Essayez de coder vous-même ce petit programme, c'est facile et ça vous entraînera !

La réponse est ci-dessous :

**int main(int argc, char \*argv[])**

**{**

**int resultat = 0, nombre1 = 0, nombre2 = 0;**

**// On demande les nombres 1 et 2 à l'utilisateur :**

**printf("Entrez le nombre 1 : ");**

**scanf("%d", &nombre1);**

**printf("Entrez le nombre 2 : ");**

**scanf("%d", &nombre2);**

**// On fait le calcul :**

**resultat = nombre1 + nombre2;**

**// Et on affiche l'addition à l'écran :**

**printf ("%d + %d = %d\n", nombre1, nombre2, resultat);**

**return 0;**

**}**

Entrez le nombre 1 : 30

Entrez le nombre 2 : 25

30 + 25 = 55

Notre programme est capable d'additionner deux nombres et d'afficher le résultat de l'opération !

Vous pouvez essayer avec n'importe quel nombre (du moment que vous ne dépassez pas les limites d'un type int), votre ordinateur effectuera le calcul en un éclair.

Je vous conseille de faire la même chose avec les autres opérations pour vous entraîner (soustraction, multiplication…). Vous ne devriez pas avoir trop de mal, vu qu'il y a juste un ou deux signes à changer. Vous pouvez aussi ajouter une troisième variable et faire l'addition de trois variables à la fois, ça fonctionne sans problème :

resultat = nombre1 + nombre2 + nombre3;

Dans la vidéo suivante, on récapitule comment faire des calculs en utilisant les variables :

### Utilisez les raccourcis

Il existe en C des techniques permettant de raccourcir l'écriture des opérations.

C'est pratique quand on a des opérations répétitives. Vous allez voir ce que je veux dire par là tout de suite, avec ce qu'on appelle l'incrémentation.

#### L'incrémentation

Vous serez souvent amenés à ajouter 1 à une variable : au fur et à mesure du programme, vous aurez des variables qui augmentent de 1 en 1.

Imaginons que votre variable s'appelle nombre. Sauriez-vous comment faire pour ajouter 1 à cette variable, sans savoir quel est le nombre qu'elle contient ? Voici comment on doit faire :

nombre = nombre + 1;

On fait le calcul nombre + 1, et on range ce résultat dans la variable… nombre! Du coup, si notre variable nombre valait 4, elle vaut maintenant 5. Si elle valait 8, elle vaut maintenant 9, etc.

Cette opération est répétitive. Les informaticiens étant des gens particulièrement fainéants, ils n'avaient guère envie de taper deux fois le même nom de variable. Ils ont donc inventé un raccourci pour cette opération : **l'incrémentation**.

Cette instruction produit exactement le même résultat que le code qu'on vient de voir :

nombre++;

Cette ligne, bien plus courte que celle de tout à l'heure, signifie :

« Ajoute 1 à la variable nombre».

Il suffit d'écrire le nom de la variable à incrémenter, de mettre deux signes  +  , et bien entendu, de ne pas oublier le point-virgule  ;  .

Vous avez peut-être remarqué que le signe  ++  se trouve dans le nom du langage C++. C'est un clin d'œil des programmeurs, et vous êtes maintenant capable de le comprendre !

C++ signifie qu'il s'agit du langage C "incrémenté", c'est-à-dire, du langage C à un niveau "supérieur". En pratique, le C++ permet surtout de programmer différemment, mais il n'est pas "meilleur" que le C : juste différent.

#### La décrémentation

C'est tout bêtement l'inverse de l'incrémentation : on enlève 1 à une variable.

Même si on fait plus souvent des incrémentations que des décrémentations, cela reste une opération pratique que vous utiliserez de temps en temps.

La décrémentation, si on l'écrit en forme « longue », ça donne :

nombre = nombre - 1;

Et maintenant en forme « raccourcie » :

nombre--;

Au lieu de mettre un++, vous mettez un-- : si votre variable vaut 6, elle vaudra 5 après l'instruction de décrémentation.

#### Les autres raccourcis

Il existe d'autres raccourcis qui fonctionnent sur le même principe, pour toutes les opérations de base :+ - \* / %.

Cela permet d'éviter une répétition du nom d'une variable sur une même ligne. Ainsi, si vous voulez multiplier par deux une variable :

nombre = nombre \* 2;

Vous pouvez l'écrire d'une façon raccourcie comme ceci :

nombre \*= 2;

Si le nombre vaut 5 au départ, il vaudra 10 après cette instruction.

Pour les autres opérations de base, cela fonctionne de la même manière :

int nombre = 2;

nombre += 4; // nombre vaut 6...

nombre -= 3; // ... nombre vaut maintenant 3

nombre \*= 5; // ... nombre vaut 15

nombre /= 3; // ... nombre vaut 5

nombre %= 3; // ... nombre vaut 2 (car 5 = 1 \* 3 + 2)

On peut utiliser toutes les opérations de base, et ajouter, soustraire, multiplier par n'importe quel nombre. Ce sont des raccourcis à connaître si vous avez un jour des lignes répétitives à taper dans un programme.

### Puisez dans la bibliothèque mathématique

En langage C, il existe ce qu'on appelle des bibliothèques "standard", c'est-à-dire des bibliothèques toujours utilisables. Ce sont en quelque sorte des bibliothèques de base qu'on utilise très souvent.

Les bibliothèques sont des ensembles de fonctions toutes prêtes. Ces fonctions ont été écrites par des programmeurs avant vous, elles vous évitent en quelque sorte d'avoir à réinventer la roue à chaque nouveau programme.

Vous avez déjà utilisé les fonctions printf et scanf de la bibliothèque stdio.h. Mais il existe une autre bibliothèque qui contient de nombreuses fonctions mathématiques toutes prêtes : math.h .

En effet, les cinq opérations de base que l'on a vues sont loin d'être suffisantes !

Tenez par exemple, on ne peut pas faire de puissances en C ! Comment calculer un simple carré ? Vous pouvez toujours essayer de taper 5$^2$  dans votre programme, mais votre ordinateur ne le comprendra jamais car il ne sait pas ce que c'est… À moins que vous le lui expliquiez en lui indiquant la bibliothèque mathématique !

Pour pouvoir utiliser les fonctions de la bibliothèque mathématique, il est indispensable de mettre la directive de préprocesseur suivante en haut de votre programme :

#include <math.h>

Une fois que c'est fait, vous pouvez utiliser toutes les fonctions de cette bibliothèque.

Je vous présente les fonctions les plus importantes dans ce [document](https://course.oc-static.com/courses/19980/La+biblioth%C3%A8que+math%C3%A9matique+-+ANNEXE+%5BApprenez+%C3%A0+programmer+en+C%5D+.pdf).

Vous n'avez peut-être pas tous le niveau en maths pour comprendre ce que font ces fonctions. Si c'est votre cas, pas d'inquiétude. Lisez juste, cela ne vous pénalisera pas pour la suite.

### En résumé

* Les opérations connues par votre ordinateur sont très **basiques**: l'addition, la soustraction, la multiplication, la division et le modulo (le reste de la division).
* Il est possible d'**effectuer des calculs entre des variables**. C'est d'ailleurs ce qu'un ordinateur sait faire de mieux : il le fait bien et vite.
* L'**incrémentation** consiste à ajouter 1 à une variable. On écrit variable++.
* La **décrémentation** est l'opération inverse : on retire 1 à une variable avec variable--.
* Pour augmenter le nombre d'opérations connues par votre ordinateur, il faut charger la **bibliothèque mathématique** (c'est-à-dire #include <math.h>).
* Cette bibliothèque contient des **fonctions mathématiques plus avancées**, telles que la puissance, la racine carrée, l'arrondi, l'exponentielle, le logarithme, etc.