**Les variables**

Une variable, c'est une petite information temporaire qu'on stocke dans la mémoire vive (RAM).

On dit qu'elle est "variable" car c'est une valeur qui peut changer pendant le déroulement du programme.

Nos programmes sont remplis de variables.

En langage C, une variable est constituée de deux choses :

1. Une **valeur** : c'est le nombre qu'elle stocke, par exemple 5.
2. Un **nom** : c'est ce qui permet de la reconnaître. En programmant en C, on n'aura pas à retenir l'adresse mémoire à la place, on va juste indiquer des noms de variables. C'est le compilateur qui fera la conversion entre le nom et l'adresse.

**Donnez un nom à vos variables**

En langage C, chaque variable doit donc avoir un nom. Pour une variable qui retient le nombre de vies dans un programme de jeu, on aimerait bien l'appeler "Nombre de vies" ou quelque chose du genre. il y a quelques contraintes.

Vous ne pouvez pas appeler une variable n'importe comment :

* il ne peut y avoir que des minuscules, des majuscules et des chiffres ;
* votre nom de variable doit commencer par une lettre ;
* les espaces sont interdits mais on peut utiliser le caractère "underscore"  \_  pour séparer des mots. C'est le seul caractère différent des lettres et chiffres autorisé ;
* les accents ("é", "à", "ê", etc) et les symboles (+, -, \*, /, %, =) sont interdits.

De plus, le langage C fait la différence entre les majuscules et les minuscules. Pour votre culture, sachez qu'on dit que c'est un langage qui "respecte la casse".

Les variables largeur  , LARGEUR ou encore LArgEuR sont trois variables différentes en langage C, même si pour nous ça a l'air de signifier la même chose !

Voici quelques exemples de noms de variables corrects :nombreDeVies  , prenom  ,nom  ,numero\_de\_telephone  ,numeroDeTelephone.

Chaque programmeur a sa propre façon de nommer des variables. Pendant ce cours, je vais vous montrer ma manière de faire :

* je commence tous mes noms de variables par une lettre minuscule ;
* s'il y a plusieurs mots dans mon nom de variable, je mets une lettre majuscule au début de chaque nouveau mot.

Quoi que vous fassiez, faites en sorte de donner des noms clairs à vos variables. On aurait pu abréger nombreDeVies  en l'écrivant par exemple ndv. C'est peut-être plus court, mais c'est beaucoup moins clair pour vous quand vous relisez votre code. N'ayez donc pas peur de donner des noms un peu plus longs pour que ça reste compréhensible.

**Utilisez différents types de variables**

Notre ordinateur n'est en fait rien d'autre qu'une (très grosse) machine à calculer. Il ne sait traiter que des nombres :

* les **nombres entiers positifs et négatifs**, comme 45 ou -87 ;
* les **nombres décimaux** positifs ou négatifs, comme 75,909 ou -76,9.

Lorsque vous lui demandez de stocker un nombre, vous devez dire de quel type il est.

Voici les principaux types de variables existant en langage C, que l'on peut classer en deux catégories :

* ceux qui permettent de stocker des nombres entiers : signed char  ,int  ,long ;
* ceux qui permettent de stocker des nombres décimaux (aussi appelés "nombres flottants") : float  ,double .

Votre ordinateur ne connaît pas la virgule, il utilise le point. Vous ne devez donc pas écrire 54,9 mais plutôt 54.9 !

Le tableau ci-dessous montre les fourchettes de valeurs minimales et maximales garanties par le langage :

| **Nom du type** | **Minimum** | **Maximum** |
| --- | --- | --- |
| signed char | -127 | 127 |
| int | -32 767 | 32 767 |
| long | -2 147 483 647 | 2 147 483 647 |
| float | -1 x1037 | 1 x1037 |
| double | -1 x1037 | 1 x1037 |

En réalité, il est fort probable que vous puissiez stocker des valeurs plus élevées que celles-ci. Cependant, veillez à garder ces valeurs en tête lorsque vous choisissez un type, c'est important.

Pourquoi avoir créé trois types pour les nombres entiers ? Un seul aurait été suffisant, non ?

Oui, mais on a créé à l'origine plusieurs types pour économiser de la mémoire. Ainsi, quand on dit à l'ordinateur qu'on a besoin d'une variable de type char, on prend moins d'espace en mémoire que si on avait demandé une variable de type int.

Toutefois, c'était utile surtout à l'époque où la mémoire était limitée. Aujourd'hui, nos ordinateurs ont largement assez de mémoire vive pour que ça ne soit plus vraiment un problème. Il ne sera donc pas utile de se prendre la tête pendant des heures sur le choix d'un type. Si vous ne savez pas si votre variable risque de prendre une grosse valeur, mettez int ou double pour un flottant.

En résumé, on fera surtout la distinction entre nombres entiers et flottants :

* pour un nombre **entier**, on utilisera le plus souvent int;
* pour un nombre **flottant**, on utilisera généralement double.

**Déclarez une variable**

On y arrive. Maintenant, créez un nouveau projet console que vous appellerez « variables ».
On va voir comment déclarer une variable, c'est-à-dire **demander à l'ordinateur la permission d'utiliser un peu de mémoire**.

Déclarer une variable, c'est très simple maintenant que vous savez tout ce qu'il faut. Il suffit :

1. D'indiquer le type de la variable que l'on veut créer.
2. D'insérer un espace.
3. D'indiquer le nom que vous voulez donner à la variable.
4. Et enfin, de ne pas oublier le point-virgule.

Par exemple, si je veux créer ma variable nombreDeVies de type int, je dois taper la ligne suivante :

**int nombreDeVies;**

Et c'est tout ! Quelques autres exemples stupides pour la forme :

**int noteDeMaths;**

**double sommeArgentRecue;**

Vous devez faire les déclarations de variables au début des fonctions. Comme pour le moment on n'a qu'une seule fonction (la fonction main), vous allez déclarer la variable comme ceci :

**#include <stdio.h>**

**#include <stdlib.h>**

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

**{**

 **int nombreDeVies;**

 **return 0;**

**}**

Si vous lancez le programme ci-dessus, vous constaterez avec stupeur… qu'il ne fait rien.

Comment ça se fait ?

Alors, avant que vous ne m'étrangliez en croyant que je vous mène en bateau depuis tout à l'heure, laissez-moi juste dire une chose pour ma défense : en fait, il se passe des choses, mais vous ne les voyez pas.

Lorsque le programme arrive à la ligne de la déclaration de variable, il demande bien gentiment à l'ordinateur s'il peut utiliser un peu d'espace dans la mémoire vive.

Si tout va bien, l'ordinateur répond « Oui bien sûr, fais comme chez toi ». Généralement, cela se passe sans problème.

Le seul souci qu'il pourrait y avoir, c'est qu'il n'y ait plus de place en mémoire… Mais heureusement cela arrive rarement, car pour remplir toute la mémoire rien qu'avec des int, il faut vraiment le vouloir !

Soyez donc sans crainte, vos variables devraient normalement être créées sans souci.

Astuce : si vous avez plusieurs variables du même type à déclarer, inutile de faire une ligne pour chaque variable.

Il vous suffit de séparer les différents noms de variables par des virgules sur la même ligne.

En tapant int nombreDeVies, niveau, ageDuJoueur;  , vous créez trois variables de type int :

1. nombreDeVies.
2. niveau.
3. ageDuJoueur .

Et maintenant ?

Maintenant qu'on a créé notre variable, on va pouvoir lui donner une valeur.

**Affectez une valeur à une variable:**

Si vous voulez donner une valeur à la variable nombreDeVies, il suffit de procéder comme ceci :

nombreDeVies = 5;

Rien de plus à faire. Vous indiquez le nom de la variable, un signe égal, puis la valeur que vous voulez y mettre.

Ici, on vient de donner la valeur 5 à la variable nombreDeVies.

Notre programme complet ressemble donc à ceci :

**#include <stdio.h>**

**#include <stdlib.h>**

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

**{**

 **int nombreDeVies;**

 **nombreDeVies = 5;**

 **return 0;**

**}**

Là encore, rien ne s'affiche à l'écran, tout se passe dans la mémoire.

Quelque part dans les tréfonds de votre ordinateur, une petite case de mémoire vient de prendre la valeur 5. N'est-ce pas magnifique ?

On peut s'amuser si on veut à changer la valeur par la suite :

**int nombreDeVies;**

**nombreDeVies = 5;**

**nombreDeVies = 4;**

**nombreDeVies = 3;**

Dans cet exemple, la variable prend d'abord la valeur 5, puis 4, et enfin 3. Comme votre ordinateur est très rapide, tout cela se passe extrêmement vite. Vous n'avez pas le temps de cligner des yeux que votre variable vient de prendre les valeurs 5, 4 et 3… et ça y est, votre programme est fini.

Quand on déclare une variable, quelle valeur a-t-elle au départ ?

En effet, quand l'ordinateur lit cette ligne :

int nombreDeVies;

... il réserve un petit emplacement en mémoire, d'accord. Mais quelle est la valeur de la variable à ce moment-là ? Y a-t-il une valeur par défaut (par exemple 0) ?

Eh bien, accrochez-vous : la réponse est non. Non, non et non, il n'y a pas de valeur par défaut. En fait, l'emplacement est réservé mais la valeur ne change pas. On n'efface pas ce qui se trouve dans la « case mémoire ». Du coup, votre variable prend la valeur qui se trouvait là avant dans la mémoire, et **cette valeur peut être n'importe quoi**!

Si cette zone de la mémoire n'a jamais été modifiée, la valeur est peut-être 0. Mais vous n'en êtes pas sûr, il pourrait très bien y avoir le nombre 363 ou 18 à la place, c'est-à-dire un reste d'un vieux programme qui est passé par là avant !

Il faut donc faire très attention à ça si on veut éviter des problèmes par la suite. Le mieux est d'initialiser la variable dès qu'on la déclare. En C, c'est tout à fait possible. En gros, ça consiste à combiner la déclaration et l'affectation d'une variable dans la même instruction :

**int nombreDeVies = 5;**

Ici, la variable nombreDeVies est déclarée et prend tout de suite la valeur 5. L'avantage, c'est que vous êtes sûr après que cette variable contient une valeur correcte, et pas du n'importe quoi.

Comment on fait pour qu'une variable garde la même valeur pendant toute la durée du programme ? Et que personne n'ait le droit de changer ce qu'elle contient ?

Grâce aux **constantes**, justement parce que leur valeur reste constante.

Pour déclarer une constante, il faut utiliser le mot const juste devant le type quand vous déclarez votre variable.

Il faut obligatoirement lui donner une valeur au moment de sa déclaration. Après, il sera trop tard : vous ne pourrez plus changer la valeur de la constante.

**Exemple :**

const int NOMBRE\_DE\_VIES\_INITIALES = 5;

Par convention, on écrit les noms des constantes tout en **majuscules**. Cela permet de distinguer facilement les constantes des variables.

Notez qu'on utilise l'underscore  \_  à la place de l'espace.

Une constante s'utilise comme une variable normale.

La seule chose qui change, c'est que si vous essayez de modifier la valeur de la constante plus loin dans le programme, le compilateur vous indiquera qu'il y a une erreur avec cette constante (les erreurs de compilation sont affichées en bas de l'écran).

Et le compilateur vous affichera alors un mot doux du genre :

[Warning] assignment of read-only variable 'NOMBRE\_DE\_VIES\_INITIALES'

Traduction : « Triple idiot, pourquoi tu essaies de modifier la valeur d'une constante ? ».

**Affichez le contenu d'une variable**

On sait afficher du texte à l'écran avec la fonction printf.

Maintenant, on va voir comment afficher la valeur d'une variable avec cette même fonction.

On utilise en fait printf de la même manière, sauf que l'on rajoute un symbole spécial à l'endroit où l'on veut afficher la valeur de la variable :

printf("Il vous reste %d vies");

Ce « symbole spécial » est en fait un %  suivi d'une lettre (dans mon exemple, la lettre d).

Cette lettre permet d'indiquer ce que l'on doit afficher.

d  signifie que l'on souhaite afficher un int.

Il existe plusieurs autres possibilités, mais pour des raisons de simplicité on va se contenter de retenir celles-ci :

| **Format** | **Type attendu** |
| --- | --- |
| "%d" | int |
| "%u" | unsigned int |
| "%ld" | long |
| "%f" | float |
| "%f" | double |

Remarquez que le format utilisé pour afficher un float et un double est identique.

Bon, on a indiqué qu'à un endroit précis on voulait afficher un nombre entier, mais on n'a pas précisé lequel ! Il faut donc indiquer à la fonction printf quelle est la variable dont on veut afficher la valeur. Pour ce faire, vous devez taper le nom de la variable après les guillemets, et après avoir rajouté une virgule :

printf("Il vous reste %d vies", nombreDeVies);

Le %d sera remplacé par la variable indiquée après la virgule, à savoir nombreDeVies.

**On se teste ça dans un programme ?**

**#include <stdio.h>**

**#include <stdlib.h>**

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

**{**

 **int nombreDeVies = 5; // Au départ, le joueur a 5 vies**

 **printf("Vous avez %d vies\n", nombreDeVies);**

 **printf("\*\*\*\* B A M \*\*\*\*\n"); // Là il se prend un grand coup sur la tête**

 **nombreDeVies = 4; // Il vient de perdre une vie !**

 **printf("Ah desole, il ne vous reste plus que %d vies maintenant !\n\n", nombreDeVies);**

 **return 0;**

**}**

Ce programme affiche ceci à l'écran :

Vous avez 5 vies

\*\*\*\* B A M \*\*\*\*

Ah desole, il ne vous reste plus que 4 vies maintenant !

Vous devriez reconnaître ce qui se passe dans votre programme.

1. Au départ le joueur a 5 vies, on affiche ça dans un printf.
2. Ensuite, le joueur prend un coup sur la tête (d'où le BAM).
3. Finalement il n'a plus que 4 vies, on affiche ça aussi avec un printf.

Bref, c'est plutôt simple.

Je vous propose tout de même de regarder cette vidéo qui résume comment créer et utiliser une variable :

**Affichez plusieurs variables dans un même printf**

Il est possible d'afficher la valeur de plusieurs variables dans un seul printf.

Il suffit d'indiquer des %d ou des %f là où vous voulez, puis d'indiquer les variables correspondantes dans le même ordre, séparées par des virgules :

**printf("Vous avez %d vies et vous etes au niveau n° %d", nombreDeVies, niveau);**

Veillez à indiquer vos variables dans le bon ordre.

1. Le premier %d sera remplacé par la première variable : nombreDeVies.
2. Et le second %d par la seconde variable : niveau.

Si vous vous trompez d'ordre, votre phrase ne voudra plus rien dire.

Allez, un petit test maintenant. Notez que j'enlève les lignes tout en haut (les directives de préprocesseur commençant par un #) ; je vais supposer que vous les mettez à chaque fois, maintenant :

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

**{**

 **int nombreDeVies = 5, niveau = 1;**

 **printf("Vous avez %d vies et vous etes au niveau n° %d\n", nombreDeVies, niveau);**

 **return 0;**

**}**

Ce qui affichera :

Vous avez 5 vies et vous etes au niveau n° 1

**Récupérez une saisie**

Les variables commencent à devenir intéressantes maintenant. On va apprendre à demander à l'utilisateur de taper un nombre dans la console. Ce nombre, on va le récupérer et le stocker dans une variable. Une fois que ça sera fait, on pourra faire tout un tas de choses avec, vous verrez.

Pour demander à l'utilisateur d'entrer quelque chose dans la console, on va utiliser scanf.

Cette fonction ressemble beaucoup à printf :

1. Vous devez mettre un format pour indiquer ce que l'utilisateur doit entrer : int, float...
2. Ensuite, vous devez indiquer le nom de la variable qui va recevoir le nombre.

Voici comment faire, par exemple :

**int age = 0;**

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

On doit mettre :

* %d entre guillemets ;
* &  devant le nom de la variable qui va recevoir la valeur.

Attention, il y a une petite divergence de format entre  printf  et  scanf  !

Pour récupérer un float, c'est le format"%f"qu'il faut utiliser.

Mais pour le type double, c'est le format"%lf".

**double poids = 0;**

**scanf("%lf", &poids);**

Revenons à notre programme.

Lorsque celui-ci arrive à un scanf, il se met en pause et attend que l'utilisateur entre un nombre. Ce nombre sera stocké dans la variable age.

Voici un petit programme simple qui demande l'âge de l'utilisateur et qui le lui affiche ensuite :

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

**{**

 **int age = 0; // On initialise la variable à 0**

 **printf("Quel age avez-vous ? ");**

 **scanf("%d", &age); // On demande d'entrer l'âge avec scanf**

 **printf("Ah ! Vous avez donc %d ans !\n\n", age);**

 **return 0;**

**}**

Quel age avez-vous ? 20

Ah ! Vous avez donc 20 ans !

Le programme se met donc en pause après avoir affiché la question "Quel age avez-vous ?". Le curseur apparaît à l'écran, vous devez taper un nombre entier (votre âge). Tapez ensuite sur "Entrée" pour valider, et le programme continuera à s'exécuter.

Ici, tout ce qu'il fait après, c'est afficher la valeur de la variable age  à l'écran :

"Ah ! Vous avez donc 20 ans !".

Vous avez compris le principe : grâce à la fonction scanf  , on peut donc commencer à interagir avec l'utilisateur.

Notez que rien ne vous empêche de taper autre chose qu'un nombre entier :

* si vous rentrez un nombre décimal, comme 2.9, il sera automatiquement tronqué, c'est-à-dire que seule la partie entière sera conservée. Dans ce cas, c'est le nombre 2 qui aurait été stocké dans la variable ;
* si vous tapez des lettres au hasard ("éèydf"), la variable ne changera pas de valeur. Ce qui est bien ici, c'est qu'on avait initialisé notre variable à 0 au début. De ce fait, le programme affichera "0 ans" si ça n'a pas marché. Si on n'avait pas initialisé la variable, le programme aurait pu afficher n'importe quoi !

**En résumé**

* Nos ordinateurs possèdent plusieurs types de mémoire comme les registres, la mémoire cache, la mémoire vive et le disque dur. Pour « retenir » des informations, notre programme a besoin de stocker des données dans la mémoire. Il utilise pour cela la **mémoire vive**.
* Dans notre code source, les **variables** sont des données stockées temporairement en mémoire vive. La valeur de ces données peut changer au cours du programme.
* À l'opposé, on parle de **constantes** pour des données stockées en mémoire vive. La valeur de ces données ne peut pas changer.
* Il existe plusieurs types de variables, qui occupent plus ou moins d'espace en mémoire. Certains types comme int sont prévus pour stocker des nombres entiers, tandis que d'autres comme double stockent des nombres décimaux.
* La fonction scanf permet de demander à l'utilisateur de saisir un nombre.