



Introduction à la programmation en C



1. Présentation du langage C et de son historique

- **Création du C (1972)** : Dennis Ritchie a développé le C comme une évolution du langage B. Il était destiné à améliorer la flexibilité, la portabilité et la capacité de gestion des systèmes d'exploitation, tout en offrant un accès direct aux ressources matérielles de l'ordinateur. Le C a été conçu pour être plus puissant et plus flexible que ses prédécesseurs tout en restant relativement simple à utiliser pour des tâches de bas niveau.
- **Unix et le C** : Le C a acquis une notoriété importante avec son utilisation dans le développement du système d'exploitation **Unix**. Ce qui a permis une grande portabilité sur différentes machines.
- **Évolution et influence** : Le langage C a eu une influence majeure sur de nombreux langages modernes, notamment **C++**, **C#**, **Java**, **Objective-C**, et même **Python** dans une certaine mesure.



2. Installation des outils

- **Code::Blocks (Windows, Mac et Linux)**

Si vous êtes sous Windows, téléchargez le logiciel en prenant le programme qui contient **mingw-setup** dans le nom.

- **Visual Studio (Windows seulement)**

C'est l'IDE de Microsoft. Il est à la base payant, mais Microsoft a sorti une version gratuite intitulée Visual Studio Community. Il permet de programmer en C, en C++ et en bien d'autres langages.

- **Installez Xcode (Mac OS seulement)**

Xcode est disponible sur [App Store](#).

3. Structure de base d'un programme C

- Code le minimal en langage C

```
#include <stdio.h> } Directives de préprocesseur
#include <stdlib.h>

int main()
{
    printf("Hello World !"\n); } Instructions } Fonction
return 0;
}
```

Le mot **include** en anglais signifie "inclure" en français. Ces lignes demandent d'inclure des fichiers au projet, c'est-à-dire d'ajouter des fichiers pour la compilation

4. Types de données et opérateurs

➤ Déclarez des variables

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

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 (ouf !) : 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.

➤ Différents types de variables

- Ceux qui permettent de stocker des nombres entiers : **signed char, int, long;**
- ceux qui permettent de stocker des nombres décimaux : **float, double.**

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

Nom du type	Minimum	Maximum
<code>signed char</code>	-128	127
<code>int</code>	-32 768	32 767
<code>long</code>	-2 147 483 648	2 147 483 647
<code>float</code>	1.17549×10^{-38}	$3.40282 \times 10^{+38}$
<code>double</code>	2.22507×10^{-308}	$1.79769 \times 10^{+308}$

Pour les types entiers (**signed char**, **int**, **long**) il existe d'autres types dits **unsigned** (non, signés) qui, eux, ne peuvent stocker que des nombres positifs. Pour les utiliser, il suffit d'écrire le mot **unsigned** devant le type:

<code>unsigned char</code>	0 à 255
<code>unsigned int</code>	0 à 65 535
<code>unsigned long</code>	0 à 4 294 967 295

► Déclarez une variable

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

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(int argc, char *argv[]) // Équivalent de int main()
5 {
6     int nombreDeVies;
7
8     return 0;
9 }
```

► Affectez une valeur à une variable

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

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(int argc, char *argv[])
5 {
6     int nombreDeVies;
7     nombreDeVies = 5;
8
9     return 0;
10 }
```



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.

Exemple:

```
1 const int NOMBRE_DE_VIES_INITIALES = 5;
```

➤ Affichez le contenu d'une variable

On utilise en fait **printf** avec un symbole spécial à l'endroit où l'on veut afficher la valeur de la variable :

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


➤ Symboles Spéciaux

Format	Type attendu
"%d"	int
"%u"	unsigned int
"%ld"	long
"%f"	float
"%lf"	double

Exemple:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int nombreDeVies = 5; // Au départ, le joueur a 5 vies
    printf("Vous avez %d vies\n", nombreDeVies);
    return 0;
}
```

➤ Affichez plusieurs variables dans un même printf

```
1 int main(int argc, char *argv[])
2 {
3     int nombreDeVies = 5, niveau = 1;
4
5     printf("Vous avez %d vies et vous etes au niveau n° %d\n", nombreDeVies, niveau);
6
7     return 0;
8 }
```

➤ Récupérez une saisie

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

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.

Remarque

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"`

Exemple:

```
1 int main(int argc, char *argv[])
2 {
3     int age = 0; // On initialise la variable à 0
4
5     printf("Quel age avez-vous ? ");
6     scanf("%d", &age); // On demande d'entrer l'âge avec scanf
7     printf("Ah ! Vous avez donc %d ans !\n\n", age);
8
9     return 0;
10 }
```

5. Faites des calculs avec des variables

➤ Faites des calculs avec des variables

```
int resultat = 0;

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

➤ La division

```
int resultat = 0;

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

```
1 double resultat = 0;
2
3 resultat = 5.0 / 2.0;
4 printf ("5 / 2 = %lf", resultat);
```

➡ Faites des calculs entre variables

```
1 resultat = nombre1 + nombre2;
```

```
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;
}
```

➤ L'incrémentation

```
1 nombre = nombre + 1;
```

```
1 nombre++;
```

➤ La décrémentation

```
1 nombre = nombre - 1;
```

```
1 nombre--;
```

➤ Les autres raccourcis

```
1 int nombre = 2;  
2  
3 nombre += 4; // nombre vaut 6...  
4 nombre -= 3; // ... nombre vaut maintenant 3  
5 nombre *= 5; // ... nombre vaut 15  
6 nombre /= 3; // ... nombre vaut 5  
7 nombre %= 3; // ... nombre vaut 2 (car 5 = 1 * 3 + 2)
```