R101 — Introduction au développement

Initiation à la programmation en langage C — Introduction

Clément Pagès
clement.pages@u-pec.fr

Université Paris-Est Créteil
BUT Informatique 1re année

2025-2026

Objectifs

- Initiation au développement.
- Ressource nécessaire pour le développement d'applications et l'optimisation des applications informatiques.

Références bibliographiques

Livres de référence

- *The C Programming Language*, B. W. Kernighan, D. M. Ritchie, *Prentice Hall*, 1978.
- *Le langage C — C ANSI*, B. W. Kernighan, D. M. Ritchie, *Prentice Hall*, 1994 2e édition. Traduit par J.-F. Groff et E. Mottier.
- *Langage C — Manuel de référence*, Harbison S.P., Steele Jr. G.L., *Masson*, 1990. Traduit en français par J.C. Franchitti.

Sites Internet

Zeste de Savoirs, *Le langage C*,
zestedesavoir.com/tutoriels/755/le-langage-c-1/

Sommaire

Qu'est-ce que la programmation ?

- Donner des consignes à une machine.
- Les consignes sont données dans un langage de programmation qui n'est pas la langue naturelle.
- Un formule un algorithme compréhensible par un ordinateur.
- Les consignes sont écrites dans un fichier texte, appelé code source.

Qu'est-ce que la programmation ?

- Donner des consignes à une machine.
- Les consignes sont données dans un langage de programmation qui n'est pas la langue naturelle.
- Un formule un algorithme compréhensible par un ordinateur.
- Les consignes sont écrites dans un fichier texte, appelé code source.

C

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

int main(void) {
  int age=0;
  printf("Quel âge avez-vous ? ");
  scanf("%d", &age);

  if(age>=18) printf("Vous êtes majeur.");
  else        printf("Vous êtes mineur.");
}
```

Console

```
Quel âge avez-vous ? 35
Vous êtes majeur.
```

Les langages de programmation

- Il en existe des centaines : certains généralistes (C, C++, Java, Ruby, Ada, etc.) d'autres spécialisés (Scilab, Coq, PHP, etc.)
- Il existe plusieurs paradigmes pour programmer : impératif, fonctionnel, orienté objet, procédural, etc.
- Il existe des langages de différents niveaux :

  Les langages bas niveau sont proches de la machine.
  Les langages haut niveau sont proches du raisonnement humain.
- Dans cette ressource, nous allons étudier les langages C et Python en programmation impérative.

Le langage C

- C'est un langage de bas niveau, très proche de la machine.
- Il est né dans les années 1970, en même temps que les systèmes d'exploitation Unix (GNU/Linux, BSD, Mac OS, puis plus tard Android).
- Beaucoup de langages s'en sont inspirés. Il fait partie de la culture générale indispensable de tout·e développeur·se.

Le langage C

- C'est un langage de bas niveau, très proche de la machine.
- Il est né dans les années 1970, en même temps que les systèmes d'exploitation Unix (GNU/Linux, BSD, Mac OS, puis plus tard Android).
- Beaucoup de langages s'en sont inspirés. Il fait partie de la culture générale indispensable de tout·e développeur·se.

C

```c
#include <stdio.h>
int main(void) {
    printf("hello, world!");
    return 0;
}
```

Console

```
> hello, world!
```

Le langage C

Les paradigmes

Programmation impérative On écrit des séquences d'instruction qui se déroulent dans le sens de la lecture, et qui modifient l'état du programme.

Programmation procédurale On organise le code source en fonctions qui effectuent des tâches spécifiques et sont réutilisables. C'est une façon d'organiser son code source pour le rendre plus lisible et flexible.

Programmation structurée C prend en charge les structures conditionnelles (`if`, `if else`, `if else if else`) et les boucles (`while`, `for`).

Programmation système C est largement utilisé pour le développement de systèmes d'exploitation, de pilotes de périphériques.

Programmation sur les données Le C permet la manipulation directe de la mémoire. Il offre la possibilité de définir des structures de données personnalisées.

Sommaire

Le compilateur

- Le langage C est un langage compilé : le code source est converti en un fichier exécutable (`.exe` sur Windows, sans extension sur les systèmes Unix).

Le compilateur

- Le langage C est un langage compilé : le code source est converti en un fichier exécutable (`.exe` sur Windows, sans extension sur les systèmes Unix).
- Le programme qui effectue la traduction s'appelle le compilateur.

Le compilateur

- Le langage C est un langage compilé : le code source est converti en un fichier exécutable (`.exe` sur Windows, sans extension sur les systèmes Unix).
- Le programme qui effectue la traduction s'appelle le compilateur.
- Le compilateur le plus connu est le « GNU C compiler ».

Le compilateur

- Le langage C est un langage compilé : le code source est converti en un fichier exécutable (`.exe` sur Windows, sans extension sur les systèmes Unix).
- Le programme qui effectue la traduction s'appelle le compilateur.
- Le compilateur le plus connu est le « GNU C compiler ».

Le compilateur

- Le langage C est un langage compilé : le code source est converti en un fichier exécutable (`.exe` sur Windows, sans extension sur les systèmes Unix).
- Le programme qui effectue la traduction s'appelle le compilateur.
- Le compilateur le plus connu est le « GNU C compiler ».

Console
```
> gcc codeSource.c -o codeSource
> ls
codeSource.c courdeSource
```

Le fichier `codeSource` est produit, c'est l'exécutable.

Environnement de développement

En ligne de commande

Sur GNU/Linux

- Un éditeur de texte (à ne pas confondre avec un traitement de texte) : Vim, Emacs, Kate, gedit, etc.
- Un terminal en ligne de commande.
- Le compilateur `gcc` est déjà installé.

Environnement de développement

En ligne de commande

Sur GNU/Linux

- Un éditeur de texte (à ne pas confondre avec un traitement de texte) : Vim, Emacs, Kate, gedit, etc.
- Un terminal en ligne de commande.
- Le compilateur `gcc` est déjà installé.

Sur Windows

- Installer un terminal pour avoir un environnement POSIX. Je conseille MSys2 (`https://www.msys2.org/`).
- Ouvrir le terminal puis exécuter les commandes `pacman -Syu`, `pacman -Su` puis `pacman gcc`.
- Installer un éditeur de texte. Je conseille Notepad++ (`https://notepad-plus-plus.org/`).

Environnement de développement

Avec une interface graphique

- La compilation en ligne de commmandes est plus élégante, plus configurable, plus instructive...
- ... mais aussi plus difficile.
- Déconseillée pour cette ressource d'introduction.

Environnement de développement

Avec une interface graphique

- La compilation en ligne de commmandes est plus élégante, plus configurable, plus instructive...
- ... mais aussi plus difficile.
- Déconseillée pour cette ressource d'introduction.

Code::Blocks

- Nous allons utiliser un environnement de développement, Code::Blocks.
- Code::Block est un environnement de développement libre, open source, multi-plateformes.

Sommaire

Premier programme

main.c

```c
#include <stdio.h> // Inclusions de bibliothèques utiles.

int main(void) { // Fonction main : porte d'entrée du programme
    /* Affiche « Hello world! » dans la console.
       \n : saut de ligne */
    printf("Hello world\n");

    return 0; // Valeur de retour : 0, tout s'est bien passé.
}
```

Console

```
> gcc main.c -o helloWorld
> ./helloWorld
Hello world!
```

Instructions I

Instructions

Les instructions sont des commandes individuelles qui indiquent à l'ordinateur quoi faire. Les instructions C incluent les opérations mathématiques, les opérations de comparaison, les boucles (comme `for` et `while`), les structures conditionnelles (comme `if` et `switch`), etc.

Exemple

```c
printf("4.5+7=%f", 4.5+7);
```

C

```
4.5+7=11.500000
```

Console

Instructions II

En C, une instruction se termine toujours par un point-virgule.

```c
printf("4.5+7=%f", 4.5+7)
```

C

```
>gcc prgmc.c
prgm.c: In function ‘main’:
prgm.c:5:34: error: expected ‘;’ before ‘}’ token
```

Console

Structure générale d'un code source en C

```c
/* inclusion des fichiers d'en-tête de bibliothèques */
#include <stdlib.h>
/* déclaration des constantes et types utilisateur */
/* déclaration des fonctions (prototypes) */
/* déclaration des variables globales */
/* définition de la fonction principale main */
int main(void) {
    /* déclarations des variables locales */
    /* corps de définition de la fonction main */
}
/* définition des fonctions */
```

Les commentaires

Les commentaires permettent d'ajouter des notes explicatives dans le code source. Il existe deux sortes de commentaires.

Les commentaires sur une seule ligne Précédés de `//`.

Les commentaires sur plusieurs lignes Commencent par `/*` et terminent par `*/`.