R101 — Introduction au développement

Initiation à la programmation en langage C
Fonctions

Clément Pagès
clement.pages@u-pec.fr

Université Paris-Est Créteil
BUT Informatique 1re année

2025-2026

Principe général

- Certaines parties du code source peuvent être utilisées dans des circonstances différentes. Exemples :
  - demander à l'utilisateur·rice de saisir une adresse mail.
  - vérifier un mot de passe.
- Les programmes deviennent de plus en plus longs et complexes.
- Les fonctions permettent de découper le code sources en brique élémentaires.

Principe général

- Certaines parties du code source peuvent être utilisées dans des circonstances différentes. Exemples :
  - demander à l'utilisateur·rice de saisir une adresse mail.
  - vérifier un mot de passe.
- Les programmes deviennent de plus en plus longs et complexes.
- Les fonctions permettent de découper le code sources en brique élémentaires.

Exemple

```c
float f(float a) { // Fonction f
    float tmp = 2.5*a-7.2;
    return tmp; // On renvoie le résultat
} // On aurait pu écrire return 2.5-a-7.2 directement

int main(void) {
    float x = 14.5;
    printf("%.2f\n", f(x)); // Affiche l'image de f en x
    return 0;
}
```

Créer une fonction

- On peut créer nos propres fonctions.
- Un programme en C commence toujours par la fonction `main()`.
- Les fonctions créées doivent être écrites avant la fonction `main()` dans le programme. (En fait, ce n'est pas obligatoire, on verra après pourquoi.)

Créer une fonction

- On peut créer nos propres fonctions.
- Un programme en C commence toujours par la fonction `main()`.
- Les fonctions créées doivent être écrites avant la fonction `main()` dans le programme. (En fait, ce n'est pas obligatoire, on verra après pourquoi.)

Définir une fonction

Une fonction est définie par :

- son nom ;
- le type de la valeur de retour, s'il y en a une ;
- ses paramètres (type et nom), s'il y en a ;
- son corps.

Type d'une fonction

Comme les variables, en C les fonctions sont typées :

- Une fonction a le type de sa valeur de retour.
- Une fonction qui ne renvoie pas de résultat est de type `void`.

Type d'une fonction

Comme les variables, en C les fonctions sont typées :

- Une fonction a le type de sa valeur de retour.
- Une fonction qui ne renvoie pas de résultat est de type `void`.

Fonction de type `float`

```c
float quiSuisJe(float a, float b, float c) {
    if(a<=b && a<c)        return a;
    else if(b<=a && b<=c) return b;
    else                   return c;
}

int main(void) {
    float x = 14.5;

    printf("%.2f\n", quiSuiJe(x, -4.5, 7)); // Affichage ?

    return 0;
}
```

Type d'une fonction

Comme les variables, en C les fonctions sont typées :

- Une fonction a le type de sa valeur de retour.
- Une fonction qui ne renvoie pas de résultat est de type `void`.

Fonction de type `void`

```c
void whoAmI(float a) {
    for(int i=1 ; i<=10 ; i++)
        printf("%d×%2.f = %.2f\n", i, a, i*a);
}

int main(void) {
    float x = 14.5;

    // Que fait cette ligne de code ?
    whoAmI(x);

    return 0;
}
```

Portée des variables

Une variable n'existe que jusqu'à la fin du bloc d'instructions dans lequel elle a été déclarée.

Codes qui compilent

```c
int main(void) {
    int n = 10;
    if(n%2==0) {
        int k = n/2;
        printf("%d", k);
    }
    return 0;
}
```

```c
int main(void) {
    int n = 10;
    int k;
    if(n%2==0)
        k = n/2;
    printf("%d", k);
    return 0;
}
```

Quelle est la valeur affichée ?

Portée des variables

Une variable n'existe que jusqu'à la fin du bloc d'instructions dans lequel elle a été déclarée.

Code qui ne compile pas

C

```c
int main(void) {
    int n = 10;
    if(n%2==0) {
        int k = n/2;
    }
    printf("%d", k);
    return 0;
}
```

Console

```
test.c: In function ‘main’:
test.c:8:22: error: ‘k’ undeclared (first use
      |                      in this function)
    6 |         printf("%d", k);
```

Portée des variables et fonctions

- Une variable déclarée à l'intérieur d'une fonction est détruite à la fin de l'exécution de la fonction.
- Les paramètres passés aux fonctions sont des copies des variables d'origine.
- Pour qu'un résultat puisse être récupéré à l'extérieur d'une fonction, utiliser un `return`.

On peut aussi utiliser les pointeurs, cf. cours ultérieur.

Portée des variables et fonctions

- Une variable déclarée à l'intérieur d'une fonction est détruite à la fin de l'exécution de la fonction.
- Les paramètres passés aux fonctions sont des copies des variables d'origine.
- Pour qu'un résultat puisse être récupéré à l'extérieur d'une fonction, utiliser un `return`.

On peut aussi utiliser les pointeurs, cf. cours ultérieur.

```c
int main(void) {
  int a = 5, b;

  increment(a);

  printf("Dans main, a = %d\n\n", a);
  b = increment(a);
  printf("Dans main, b = %d\n.", a);

  return 0;
}
```

```c
int increment(int n) {
  int tmp = n++;
  printf("Dans increment:
   tmp = %d.\n", tmp);
  return tmp;
}
```

```
Dans increment: tmp = 6.
Dans main, a = 5.

Dans increment: tmp = 6.
Dans main, b = 5.
```

Prototype d'une fonction

Principe

- Une fonction ne peut être appelée que si le compilateur la connait à l'avance : il faut que son code source soit placé avant l'endroit d'où la fonction est appelée.

Prototype d'une fonction

Principe

- Une fonction ne peut être appelée que si le compilateur la connait à l'avance : il faut que son code source soit placé avant l'endroit d'où la fonction est appelée.
- Le prototype d'une fonction résume le nom de la fonction, le type de retour, le type et le nombre d'arguments :
  `type_retour f(type_arg1, type_arg2, ...);`

Prototype d'une fonction

Principe

- Une fonction ne peut être appelée que si le compilateur la connait à l'avance : il faut que son code source soit placé avant l'endroit d'où la fonction est appelée.
- Le prototype d'une fonction résume le nom de la fonction, le type de retour, le type et le nombre d'arguments :
  `type_retour f(type_arg1, type_arg2, ...);`
- On peut les placer en haut des fichiers `*.c`.

Prototype d'une fonction

Principe

- Une fonction ne peut être appelée que si le compilateur la connait à l'avance : il faut que son code source soit placé avant l'endroit d'où la fonction est appelée.
- Le prototype d'une fonction résume le nom de la fonction, le type de retour, le type et le nombre d'arguments :
  `type_retour f(type_arg1, type_arg2, ...);`
- On peut les placer en haut des fichiers `*.c`.
- Permet d'écrire les fonctions dans n'importe quel ordre.

Prototype d'une fonction

Structure d'un code source

```c
#include <stdio.h>
#include <stlib.h>
// ... et toutes les autres librairies utilisées

/* Prototypes des fonctions */
float quiSuisJe(float, float, float);
void whoAmI(float);
int increment(int);

int main(void) {
    /* Code source de la fonction main. */
}
float quiSuisJe(float a, float b, float c) {
 /* Code source de la fonction quiSuisJe. */
}
void whoAmI(float a) {
    /* Code source de la fonction whoAmI. */
}
// etc.
```