TP 1 : initiation au langage C

⏱️ 4h
📚 Algorithmie en langage C
📅 *Mis à jour le 03/10/2025*

🎯 Objectifs

- Découvrir l'environnement de développement
- Écrire ses premiers programmes

🔧 Matériel nécessaire

- Ordinateur avec compilateur gcc
- CodeBlocks/VS Code

Quelques indications

Dans ce TP, vous devrez utiliser les fonctions `printf()` et `scanf()` à de multiples reprises.

Syntaxe de `printf()`

Cette fonction permet d'afficher une chaîne de caractères formatée dans le flux `stdout` (dans la console/terminal). Elle prend en paramètre(s) :

- une chaîne de caractères (obligatoire) ;
- une suite de paramètres à injecter dans la chaîne de caractères (optionnel).

Ces paramètres seront alors injectés à la place d'identifiants de paramètres présents dans la chaîne de caractères.

Exemple : `printf("[%d, %d]", 4, 2)` → `[4, 2]`

> **Note**
>
> L'identifiant `%d` permet d'injecter la valeur d'une variable de type `int`.

> **Warning**
>
> Les paramètres à injecter ne sont optionnels que si la chaîne de caractères ne contient pas d'identifiants.

Voici quelques identifiants à retenir :

Identifiant	Type
`%d`	Entier : `int`
`%f`	Flottant : `float`
`%c`	Caractère : `char`
`%s`	Chaîne de caractères : `char*`

Syntaxe de `scanf()`

Cette fonction permet de lire une chaîne de caractères formatée depuis le flux stdin (saisie clavier dans la console/terminal). Elle prend en paramètre(s) :

- une chaîne de caractères contenant le format de lecture (obligatoire) ;
- une suite d'adresses de variables où stocker les valeurs lues (optionnel).

Les valeurs saisies seront alors stockées dans les variables dont les adresses correspondent aux identifiants de paramètres présents dans la chaîne de caractères.

Exemple :

```
int a, b;
scanf("[%d, %d]", &a, &b);
printf("%d\n", a);
printf("%d", b);
```

affiche

```
[10, 5] //saisie utilisateur
10
5
```

> ⚠ **Warning**
>
> Il faut impérativement utiliser l'opérateur `&` devant le nom des variables (sauf pour les chaînes de caractères) pour passer leur adresse.

> ⚠ **Warning**
>
> Les paramètres d'adresses ne sont optionnels que si la chaîne de caractères ne contient pas d'identifiants.

Partie 1 : Quelques programmes simples

> **Important**
>
> Créez un projet séparé pour chaque partie du TP.

Programme n°1

Q1. Écrivez un programme qui affiche le produit de deux nombres entiers saisis par l'utilisateur.

Q2. Quelle modification permettrait le calcul pour deux nombres réels ?

Programme n°2

Écrivez un programme qui affiche la distance euclidienne entre deux points A et B saisis par l'utilisateur. Il devra utiliser la syntaxe `(x,y)` pour chacun des deux points.

Programme n°3

Q1. Écrivez une fonction `afficher_triangle()` qui prend en paramètre un entier `n` et qui affiche un triangle de `n` lignes composés d'étoiles (`*`).

```
afficher_triangle(4);
```

affiche

```
*
**
***
****
```

> **Note**
>
> La fonction `afficher_triangle()` ne renvoie rien.

Q2. Écrivez un programme qui affiche un triangle dont la taille est saisie par l'utilisateur.

Programme n°4

Q1. Écrivez une fonction de prototype `int position(int tab[], int taille, int x);` qui renvoie la position de la première occurence de l'élément `x` dans un tableau `tab` de longueur `taille`. Si l'élément n'est pas présent dans le tableau, la valeur retournée est égale à `taille`.

```
int tab[4] = {1, 4, 6, 8};
int pos6 = position(tab, 4, 6);
int pos3 = position(tab, 4, 3);
printf("%d, %d", pos6, pos3);
```

affiche

```
2, 4
```

Q2. Écrivez un programme qui :

1. initialise un tableau de nombres entiers ;
2. demande à l'utilisateur un nombre entier ;
3. affiche la position de ce nombre dans le tableau s'il est présent ;
4. ou indique à l'utilisateur que le nombre n'est pas présent dans le tableau.

Partie 2 : le + ou -

Le jeu du plus ou moins consiste à deviner un nombre choisi par l'ordinateur en un minimum d'essais. À chaque coup, l'ordinateur indique si c'est *plus* ou *moins*, ou si le nombre a été trouvé.

Génération d'un nombre mystère

Nous allons générer aléatoirement un nombre mystère à deviner. Pour cela, il faut faire appel aux fonctions `srand()` et `rand()` de la bibliothèque `<stdlib.h>`.

À partir de la documentation https://nicolasj.developpez.com/articles/libc/hasard/, écrivez une fonction de prototype `int randint(int nmax);` qui renvoie un nombre entier compris entre `0` et `nmax` compris.

> **Note**
>
> La fonction `randint()` suppose qu'une initialisation du générateur de nombres pseudoaléatoires a déjà été faite.

Essai de l'utilisateur

Selon l'estimation de l'utilisateur, il faut répondre si c'est plus, moins, ou si le nombre est le bon. Écrivez une fonction de prototype `int essai(int nmyst);` qui :

1. demande à l'utilisateur de saisir un nombre ;
2. affiche si c'est plus, moins ou correct ;
3. renvoie 0 si c'est correct ;
4. renvoie 1 si l'utilisateur doit recommencer.

```
int a = essai(5);
printf("%d",a);
a = essai(5);
printf("%d",a);
```

```
2 // saisie utilisateur
C'est plus
1
5 // saisie utilisateur
Correct !
0
```

Boucle de jeu

Écrivez un programme permettant à l'utilisateur d'essayer tant qu'il n'a pas trouvé le nombre mystère.

Ajout d'un compteur d'essais

Modifiez le programme précédent afin d'afficher le nombre d'essais qui ont été nécessaires pour trouver le nombre mystère.

Ajout de niveaux

Modifiez le programme précédent pour donner la possibilité à l'utilisateur de choisir le niveau.

Niveau	**Intervalle**
1 - Facile	[0, 16]
2 - Moyen	[0, 256]
3 - Difficile	[0, 65536]

Partie 3 : tri de nombres

Dans cette partie, nous allons réaliser un programme permettant de trier un tableau de nombres entiers générés aléatoirement. On utilisera un tri à bulle.

Le tri à bulle

Le tri à bulles consiste à parcourir le tableau en permutant deux éléments consécutifs s'il ne sont pas dans le bon ordre.

Voici le pseudo-code de cet algorithme (Wikipedia).

```
tri_a_bulles(Tableau T)
    pour i allant de (taille de T)-1 à 1
        pour j allant de 0 à i-1
            si T[j+1] < T[j]
                (T[j+1], T[j]) ← (T[j], T[j+1])
```

Générer un tableau de valeurs aléatoires

Écrivez une fonction de prototype `void init_tab(int tab[], int taille);` qui initialise le tableau `tab` de taille `taille` avec des nombres entiers tirés aléatoirement entre 0 et `taille`.

Affichage d'un tableau

Écrivez une fonction de prototype `void afficher_tab(int tab[], int taille);` qui affiche le tableau `tab` de taille `taille`.

```c
int mon_tableau[5];
init_tab(mon_tableau, 5);
afficher_tab(mon_tableau, 5);
```

affiche par exemple

```
[2, 4, 1, 1, 3]
```

Permuter deux valeurs

Écrivez une fonction de prototype `void permuter(int tab[], int id1, int id2);` qui permute les éléments du tableau `tab` aux indices `id1` et `id2`.

Trier le tableau

Écrivez une fonction de prototype `void tri_a_bulles(int tab[], int taille);` qui trie le tableau `tab` de taille `taille` dans l'ordre croissant avec un tri à bulles.

Programme final

Écrivez un programme qui réalise les étapes suivantes :

1. demande à l'utilisateur la taille du tableau de valeurs ;
2. génère une liste aléatoire ;
3. affiche la liste ;
4. trie la liste ;
5. affiche la liste triée.

Utilisation d'une structure

Au lieu d'un tableau d'entiers, nous allons maintenant utiliser une structure `liste` qui contient une variable `valeurs` de type `int[MAX_TAILLE]` ainsi qu'une variable `taille` de type `int`.

```
#define MAX_TAILLE 100
struct liste {
    int valeurs[MAX_TAILLE];
    int taille;
};
```

Modifiez vos fonctions ainsi que le programme principal pour utiliser la structure `liste`. Les prototypes des fonctions deviennent :

- `liste init_tab(liste l);`
- `void afficher_tab(liste l);`
- `liste permuter(liste l, int id1, int id2);`
- `liste tri_a_bulles(liste l);`

Thomas Mongaillard
thomas.mongaillard@univ-lorraine.fr

Supports de TP/TD à l'ENSEM