TD 3 — Structures itératives en C

Dans ce TD, nous allons voir comment utiliser les structures algorithmiques de répétition : boucles `POUR`, `TANT QUE` et `RÉPÉTER … TANT QUE`.

Travail à rendre

Vous devrez déposer sur EPREL une archive `zip` nommée `R101_TD3_<NOMDEFAMILLE>.zip`, contenant trois projets `Code::Blocks` : un pour chaque partie.

Chaque projet contiendra les réponses à tous les exercices de la partie correspondante. Vous veillerez à ce que les affichages dans la console permettent de facilement se repérer à travers les exercices.

1 La boucle `POUR`

On rappelle qu'en langage C, l'exécution démarre avec le code de la fonction `main`. Dans cette partie, nous allons étudier la boucle `for`, qui est l'implémentation en C de la structure itérative `POUR`.

Du bon usage de la boucle `for`

On rappelle que les boucles `POUR` sont à utiliser lorsque le nombre d'itération est connu à l'avance, dès le début de la boucle. La boucle `for` en C obéit évidemment à ce principe.

Exemple. Voici une boucle qui compte par ordre décroissant de 8 à 0 :

```c
for(int i=0 ; i<=8 ; i++) {
    printf("%d", 8-i);
}
```

Exercice 1.

1) Écrire un programme qui affiche à l'écran `Bonjour !`

2) Modifier ce programme de deux manières différentes, pour qu'il affiche `Bonjour !` sur cinq lignes :

a. avec cinq `printf`.

b. avec un seul `printf`

3) Modifier votre programme pour qu'il affche 30 lignes de `Bonjour !`

Exercice 2. Écrire un programme qui, étant données deux variables `longueur` et `largeur`, initialisées à une valeur strictement positive quelconque, affiche à l'écran un rectangle d'étoiles ayant `longueur` étoiles en longueur et `largeur` étoiles en largeur. Par exemple, si `longueur = 3` et `largeur = 4`, votre programme doit afficher :

```
Console
***
***
***
***
```

Exercice 3. Écrire un programme qui, étant données deux variables `cote` et `largeur`, initialisées à une valeur strictement positive quelconque, affiche à l'écran un triangle rectangle isocèle de longueur `cote`. Voici l'affichage attendu lorsque d'une part `cote = 6` et d'autre part `cote = 4`.

```
Console
*       *
**      **
***     ***
****    ****
*****
******
```

2 La boucle `while` et la boucle `do… while`

La boucle **`while`** est la syntaxe en C pour la boucle `TANT QUE`. Par exemple, que fait le programme suivant ?

```c
int x = 0;

while(x<100) {
    printf("Saisir un nombre :");
    scanf("%d", &x);
}
```

Exercice 4. Écrire un programme qui demande la saisie d'un multiple de 3 au clavier, et répète tant que la saisie n'est pas correcte.

Exercice 5 (Conversion degrés Celcius — degrés Farentheit)**.** Pour convertir les degrés Farentheit en degrés Celcius, on utilise la formule suivante : $C = (F - 32) \times 5/9$.

1) Écrire un programme qui demande la saisie d'une température `tempF` en degrés Farentheit, comprise entre 0 °F et 30 °F. Répéter la demande tant que la saisie n'est pas valide.

2) Afficher ensuite toutes les conversions de températures en degrés Celcius pour les valeurs comprises entre `tempF` et 100 °F. On affichera le résultat comme un **`float`** avec deux chiffres après la virgule.

La boucle **`do`**`{/* instructions */}` **`while`**`(/* condition */);` est la structure algorithmique `RÉPÉTER … TANT QUE`. C'est l'analogue de la boucle **`while`**, sauf que le code dans la boucle s'exécute *au moins une fois*.

Exercice 6. **1)** Que fait l'extrait de code suivant ?

```c
do {
    print("Entrez une lettre minuscule : ");
    scanf("%c", &);
} while (c<'a' || c>'z'); // N'oubliez pas le point-virgule à la fin !
```

2) Intégrez ce code à un programme C, et adaptez-le pour demander la saisie d'un flottant compris entre `100.5` et `210.75`.

Exercice 7. **1)** Écrire un programme dans lequel vous définissez une variable de type entier `int codeSecret = 42`, puis un variable `int codeSaisi` initialisée à `0`. Utilisez une boucle pour continuer à demander de saisir le code tant que le code saisi et le code secret ne correspondent pas.

2) Dans cette situation, quelle est la structure de boucle la plus appropriée ? Justifier.

3 Exercices d'application

Exercice 8. Écrire un programme qui calcule et affiche la somme des entiers de 1 à n, l'entier n étant saisi au clavier : $\sum_{k=1}^{n} k$.

Exercice 9 (Suite de Fibonacci)**.** On définit la suite de Fibonacci $(F_n)_{n\in\mathbb{N}}$ par :

$$F_0 = 1, F_1 = 1$$
$$\forall n \in \mathbb{N},\ F_{n+2} = F_{n+1} + F_n$$

1) Calculer (à la main) les termes F_2, F_3, F_4, F_5.

2) Écrire un programme qui demande la saisie d'un entier N au clavier, puis calcule et affiche F_N.

3) Compléter le programme qui demande la saisie d'un entier m au clavier, puis qui calcule le plus petit entier N tel que $F_N \geq m$. Afficher alors F_N et N à l'écran.

À retenir

- Le langage C fournit trois structures de boucles :
  - La boucle `for` lorsque le nombre d'itération est connu à l'avance.
  - La boucle `while` pour répéter des instructions tant qu'une condition est vérifiée.
  - La boucle `do{...}while` pour exécuter au mois une fois des instructions et les répéter tant qu'une condition est vérifiée.
- Les blocs sont délimités par des accolades.