Travaux dirigés 2 : la structure de contrôle `for`

L'objectif de ce TD TP est de vous familiariser avec la notion d'itération en programmation. On parle communément de *boucle*. Cette notion sera illustrée sur des problèmes de comptage et de répétition d'actions.

1 Itération : l'instruction `for`

Soit le programme suivant :

```
1  /* déclaration de fonctionalités supplémentaires */
2  #include <stdlib.h> /* EXIT_SUCCESS */
3  #include <stdio.h> /* printf */
4
5  /* déclaration constantes et types utilisateurs */
6
7  /* déclaration de fonctions utilisateurs */
8
9  /* fonction principale */
10 int main()
11 {
12     /* déclaration et initialisation variables */
13     int i; /* variable de boucle */
14
15     for(i = 0; i < 5; i = i + 1)
16     {
17         printf("i = %d\n",i);
18     }
19     /* i >= 5 */
20
21     printf("i vaut %d après l'exécution de la boucle.\n",i);
22
23     return EXIT_SUCCESS;
24 }
25
26 /* definitions des fonctions utilisateurs */
```

1. Quelle est la signification de chaque argument du `for` ? Quelles instructions composent le corps de la boucle ?
2. Faire la trace du programme. Qu'affiche le programme ?
3. Modifiez le programme afin que la séquence affichée soit exactement (faire cinq programmes) :
   – `0 1 2 3 4`,
   – `1 2 3 4`,
   – `1 2 3 4 5`,
   – `1 3 5`
   – puis, enfin `(0,0) (1,1) (2,2)`.
4. Modifiez le programme afin que la séquence affichée soit :
   – `0 1 2 0 1 2`,
   – puis `0 1 2 0 1 2 3`.
   De combien de boucles avez-vous besoin ? De combien de variables de boucles ?
5. Modifiez le programme afin que la séquence affichée soit :

– `(0,0) (0,1) (0,2) (1,0) (1,1) (1,2) (2,0) (2,1) (2,2).`

De combien de boucles avez-vous besoin ? De combien de variables de boucles ? Quelle est la différence de structuration des boucles entre le point 4 et le point 5 ?

1.1 Exercice type : calcul de $\Sigma_1^n i$

Écrire un programme qui calcule et affiche la somme des entiers de 1 à n : $\Sigma_1^n i$, où n est un entier quelconque (tester avec différentes valeurs).

Comment feriez vous pour écrire le même programme en assembleur (amil) ?

2 Affichage de figures géométriques

2.1 Exercice type : affichage d'un rectangle d'étoiles

Écrire un programme qui, étant données deux variables, `longueur` et `largeur`, initialisées à des valeurs strictement positives quelconques,affiche un rectangle d'étoiles ayant pour longueur `longueur` étoiles et largeur `largeur` étoiles. Exemple :

```
Affichage d'un rectangle d'etoiles de longueur 6 et largeur 3.
******
******
******
```

2.2 Exercice type : affichage d'un demi-carré d'étoiles

Écrire un programme qui affiche, étant donnée la variable, `cote`, initialisée à une valeur quelconque, un demi-carré d'étoiles (triangle rectangle isocèle) ayant pour longueur de côté `cote` étoiles. Exemple :

```
Affichage d'un demi-carre d'etoiles de cote 5.
*
**
***
****
*****
```

3 Exercices facultatifs

3.1 Affichage d'un demi-carré droit d'étoiles

Écrire un programme qui affiche un demi-carré droit d'étoiles de côté spécifié par l'utilisateur. Exemple d'exécution :

```
Entrer la taille du demi-carré :
5
Affichage d'un demi-carre droit d'etoiles de cote 5.
    *
   **
  ***
 ****
*****
```

3.2 Calcul de la somme d'une série d'entiers saisie par l'utilisateur

Écrire un programme qui demande à l'utilisateur combien d'entiers composent sa série, lit la série d'entiers et affiche la somme des valeurs de la série.

Indication : l'instruction `scanf("%d", &a)` permet de réaliser une saisie utilisateur d'un entier dont la valeur sera affectée à la variable `a` (comme toute variable, `a` doit être préalablement déclarée).