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 ?

   Correction. — *Vu en cours. Tout est expression en C (avec effet de bord) mais on ment ici et on considère que ce sont des instructions. Ne sert a rien de les embrouiller et en plus c une mauvaise pratique de les utiliser comme expressions (source d'erreurs, de confusion).*

   ```
   for(instruction1; expression_booléenne; instruction2)
   {
       corps de la boucle : le bloc défini par la séquence d'instructions entre {}
   }
   ```

   - *instruction1 : sert à initialiser la variable de boucle*
   - *expression_ booléenne : (s'évalue à vrai ou faux) si vrai exécute le corps de boucle, sinon passe a l'instruction suivant la boucle. On parle parfois de la garde de la boucle.*
   - *instruction2 : prépare l'itération suivante ; il est important de comprendre que cette instruction doit forcément modifier la valeur de (au moins une variable, on ne leur parle que de la var de boucle) la variable intervenant dans l'expression booléenne SINON la val de expression_ booléenne est constante et on ne sort jamais de la boucle*

---

2. Faire la trace du programme. Qu'affiche le programme ?

Correction. _ *L'exécution de $i = i + 1$ est mise à la ligne 18 pour montrer que c à la fin du corps de la boucle, mm si c pas très clair.*

```
ligne          | i | affichage (sortie/écriture à l'écran)
-----------------------------------------------
initialisation | ? |
15             | 0 |
17             |   | i = 0
18             | 1 |
17             |   | i = 1
18             | 2 |
17             |   | i = 2
18             | 3 |
17             |   | i = 3
18             | 4 |
17             |   | i = 4
18             | 5 |
21             |   | i vaut 5 après l'exécution de la boucle.
```

Il affiche :

```
i = 0
i = 1
i = 2
i = 3
i = 4
i vaut 5 après l'exécution de la boucle.
```

---

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)`.

Correction. _____ *Ici, les corrections peuvent se faire plus ou moins rapidement, en fonction de la compréhension des étudiants. Chaque séquence demande la modif d'un ou de plusieurs arguments afin d'insister sur leur rôle.*
Pour `1 2 3 4 5`*, c'est soit* `i < 6` *soit* `i <=5`*. Ca dépend du problème.*

```
/* correction (0,0) (1,1) (2,2)*/
/* déclaration de fonctionalités supplémentaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */

/* déclaration constantes et types utilisateurs */

/* déclaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* déclaration et initialisation variables */
    int i; /* var. de boucle */

    for(i = 0; i < 3; i = i + 1)
    {
```

```
        printf("(%d,%d) ", i, i);
    }
    /* i >= 3 */

    printf("\n");

    return EXIT_SUCCESS;
}

/* definitions des fonctions utilisateurs */
```

---

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 ?

   Correction. _____ *2 boucles à la suite, 1 seule variable (on la réutilise)* _____

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 ?

   Correction. _ *C'est un produit cartésien : {0,1,2} x {0,1,2}. 2 boucles imbriquées, 2 variables. La différence, c'est 1) 2 à la suite ; 2) 2 imbriquées*

```
/* déclaration de fonctionalités supplémentaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */

/* déclaration constantes et types utilisateurs */

/* déclaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* déclaration et initialisation variables */
    int i; /* var. de boucle */
    int j; /* var. de boucle */

    for(i = 0; i < 3; i = i + 1)
    {
        for(j = 0; j < 3; j = j + 1)
        {
            printf("(%d,%d) ", i, j);
        }
        /* j >= 3 */
    }
    /* i >= 3 */

    /* passe a la ligne pour faire joli */
    printf("\n");

    return EXIT_SUCCESS;
}
```

```
/* definitions des fonctions utilisateurs */
```

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).

Correction.

```
/* déclaration de fonctionalités supplémentaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */

/* déclaration constantes et types utilisateurs */

/* déclaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* déclaration et initialisation variables */
    int n = 4;
    int somme = 0; /* élément neutre pour l'addition */
    int i; /* var. de boucle */

    for(i = 1; i <= n; i = i + 1) /* i allant de 1 à n */
    {
        /* ajoute i à la somme partielle */
        somme = somme + i;
    }
    /* i > n */

    /* somme vaut 0 + 1 + ... + n */
    printf("somme = %d\n",somme);

    return EXIT_SUCCESS;
}

/* definitions des fonctions utilisateurs */
```

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

Correction. *Le plus facile est d'appliquer le schéma se traduction du for (vu en cours) au for précédent (en laissant la variable i dans un registre ou en la stockant dans la mémoire). Voici quelques (anciennes) corrections plus détaillées (attention, elles partent de i = 0) mais il ne faut pas passer plus que cinq-dix minutes sur cet exercice. Il suffit de laisser les étudiants y réfléchir un peu, puis donner la structure du programme.*

La correction suivante reprend la traduction en assembleur (`gcc -S`*) d'un* `while (...) ...`*, ou d'une boucle* `for( i = 0; i <= n; i = i + 1)` *en C. L'algorithme consiste en copier la formule de la somme :*

- *la somme vaut* 0*, i vaut* 0*.*
- *Tant que $i \leq n$, ajouter x_i à la somme puis ajouter* 1 *à i.*

Le programme est alors :

Instructions	Cycles	CP	r0	r1	r2	r3	15	16
Initialisation	0	1	?	?	?	?	3	?
`valeur 0 r1`	1	2		0				
`valeur 0 r2`	2	3			0			
`saut 7`	3	**7**						
`lecture 15 r0`	4	8	3					
`soustr r2 r0`	5	9	3					
`sautpos r0 4`	6	**4**						
`add r2 r1`	7	5		0				
`valeur 1 r3`	8	6				1		
`add r3 r2`	9	7			1			
`lecture 15 r0`	10	8	3					
`soustr r2 r0`	11	9	2					
`sautpos r0 4`	12	**4**						
`add r2 r1`	13	5		1				
`valeur 1 r3`	14	6				1		
`add r3 r2`	15	7			2			
`lecture 15 r0`	16	8	3					
`soustr r2 r0`	17	9	1					
`sautpos r0 4`	18	**4**						
`add r2 r1`	19	5		3				
`valeur 1 r3`	20	6				1		
`add r3 r2`	21	7			3			
`lecture 15 r0`	22	8	3					
`soustr r2 r0`	23	9	0					
`sautpos r0 4`	24	**4**						
`add r2 r1`	25	5		6				
`valeur 1 r3`	26	6				1		
`add r3 r2`	27	7			4			
`lecture 15 r0`	28	8	3					
`soustr r2 r0`	29	9	-1					
`sautpos r0 4`	30	10						
`ecriture r1 16`	31	11						6
`stop`	32	12						

FIGURE 1 – Somme des entiers de 0 à 3

```
1   valeur 0 r1     :: initialisation a zero de l'accumulateur pour la somme
2   valeur 0 r2     :: initialisation a zero de l'indice de boucle, i
3   saut 7          :: saut sur la condition d'execution de boucle
4   add r2 r1       <corps_de_boucle> :: ajoute i a l'accumulateur
5   valeur 1 r3
6   add r3 r2         :: increment de 1 l'indice de boucle </corps_de_boucle>
7   lecture 15 r0   <condition_de_boucle>
8   soustr r2 r0      :: r0 vaut n - i
9   sautpos r0 4      :: si n >= i saut sur corps_de_boucle </condition_de_boucle>
10  ecriture r1 16  :: ecriture du resultat
11  stop
12  ?
13  ?
14  ?
15  3               <-- n
16  ?               <-- valeur de retour
```

La boucle incrémente i. Dans amil, à cause de la difficulté à écrire des tests tels que $i \leq n$, il serait plus rapide sur cet exercice de décrémenter n jusqu'à 0 comme à la question précédente :

Instructions	Cycles	CP	r0	r1	15	16
Initialisation	0	1	?	?	3	?
`lecture 15 r0`	1	2	3			
`lecture 15 r1`	2	3		3		
`add 1 r1`	3	4		4		
`mult r0 r1`	4	5		12		
`div 2 r1`	5	6		6		
`ecriture r1 16`	6	7				6
`stop`	7	8				

FIGURE 2 – Somme des entiers de 0 à 3 (3)

– *x vaut n, la somme vaut* 0

– *Tant que $x \geq 0$, ajouter x à la somme et décrementer x.*

Cette réponse est correcte et on peut encourager dans un premier temps les étudiants qui cherchent dans cette direction, mais, pour les préparer à la suite du cours, il faut leur donner la correction qui incrémente l'indice.

Parmi les autres programmes corrects possibles, des étudiants peuvent même penser à la formule de Gauss $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$.

Solution de Gauss (trace figure 2).

```
1   lecture 15 r0
2   lecture 15 r1
3   add 1 r1
4   mult r0 r1
5   div 2 r1
6   ecriture r1 16
7   stop
8
9
10
11
12
13
14
15  3
16  ?
17  ?
```

---

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.
******
******
******
```

Correction. *Durée 3/4 d'heure ?*

Les algos sont à faire (les extraire du code).

*Vous pouvez dans un premier temps supprimer la boucle la plus imbriquée en leur demandant d'afficher un rectangle de longueur exactement "**********".*

```
/* declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */

/* declaration constantes et types utilisateurs */

/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* declaration et initialisation variables */
    int largeur = 3; /* largeur du rectangle en nb d'etoiles */
    int longueur = 6; /* longueur du rectangle en nb d'etoiles */
    int i; /* var. de boucle */
    int j; /* var. de boucle */

    printf("Affichage d'un rectangle d'etoiles de longueur %d et largeur %d.\n",longueur,lar

    for(i = 0;i < largeur;i = i + 1) /* chaque ligne d'étoiles */
    {
        /* affiche longueur etoiles */
        for(j = 0;j < longueur;j = j + 1) /* chaque colonne d'etoiles */
        {
            /* affiche une etoile */
            printf("*");
        }
        /* j >= longueur */

        /* passe a la ligne suivante */
        printf("\n");
    }
    /* i >= largeur */

    return EXIT_SUCCESS;
}

/* definitions des fonctions utilisateurs */
```

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.
*
**
***
****
*****
```

Correction. *Durée 3/4 d'heure ?*

Les algos sont à faire (les extraire du code).

```
/* declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */

/* declaration constantes et types utilisateurs */

/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* declaration et initialisation variables */
    int cote = 2; /* cote du demi-carré en nb d'etoiles */
    int i; /* var. de boucle */
    int j; /* var. de boucle */

    printf("Affichage d'un demi-carre d'etoiles de cote %d.\n",cote);

    for(i = 1;i <= cote;i = i + 1) /* chaque numero de ligne d'étoiles */
    {
        /* affiche autant d'etoiles que le numero de ligne */
        for(j = 0;j < i;j = j + 1) /* chaque colonne d'etoiles */
        {
            /* affiche une etoile */
            printf("*");
        }
        /* j >= i */

        /* passe a la ligne suivante */
        printf("\n");
    }
    /* i > cote */

    return EXIT_SUCCESS;
}

/* definitions des fonctions utilisateurs */
```

---

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.
    *
   **
  ***
 ****
*****
```

Correction.

```
/* declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf, scanf */

/* declaration constantes et types utilisateurs */

/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    int cote; /* cote du demi-carré droit en nb d'etoiles a saisir par l'utilisateur*/
    int i; /* var. de boucle */
    int j; /* var. de boucle */

    /* saisie cote (optionnel) */
    printf("Entrer la taille du demi-carré :\n");
    scanf("%d", &cote);

    /* affichage du demi-carre droit */
    printf("Affichage d'un demi-carre droit d'etoiles de cote %d.\n",cote);

    for(i = 1;i <= cote;i = i + 1) /* chaque numero de ligne d'étoiles */
    {
        /* affiche les blancs */
        for(j = 0;j < cote - i;j = j + 1) /* chaque colonne de blancs */
        {
            /* affiche un blanc */
            printf(" ");
        }
        /* j >= cote - i */

        /* affiche autant d'etoiles que le numero de ligne */
        for(j = 0;j < i;j = j + 1) /* chaque colonne d'etoiles */
        {
            /* affiche une etoile */
            printf("*");
        }
        /* j >= i */

        /* passe a la ligne suivante */
        printf("\n");
    }
    /* i > cote */

    return EXIT_SUCCESS;
}

/* definition de fonctions utilisateurs */
```

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).

Correction.

```
/* declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf, scanf */

/* declaration constantes et types utilisateurs */

/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    int n; /* taille de la serie a saisir par l'utilisateur*/
    int elt; /* un element de la serie a saisir par l'utilsateur */
    int somme = 0; /* somme de la serie a calculer */

    int i; /* var. de boucle */

    /* demande la taille de la serie a l'utilisateur */
    printf("Combien d'elements dans la série ? ");
    scanf("%d", &n);

    /* saisie serie (n entiers) et calcul incremental de la somme */
    for(i = 0; i < n; i = i + 1) /* chaque entier de la serie */
    {
        /* saisir sa valeur */
        scanf("%d", &elt);

        /* l'ajoute a la somme partielle */
        somme = somme + elt;
    }
    /* i >= n */

    /* somme contient la somme des elements de la serie. */
    printf("La somme des valeurs de cette serie est : %d\n",somme);

    return EXIT_SUCCESS;
}

/* definitions des fonctions utilisateurs */
```