Licence 1 - section B
TD 5 d'éléments d'informatique

Catherine RECANATI – Département d'Informatique – Institut Galilée

Semaine du 21 au 25 novembre 2016

1 Boucles for

Exercice 1.1 Affichages de la variable de boucle (compteur)
Soit le programme suivant :

```
/* déclaration de fonctionnalités 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;   /* variable de boucle */

    for (i = 0; i < 5 ; i = i + 1)
    {
        printf("i = %d\n", i);
    }
    /* i >= 5 */

    printf("i vaut %d après l'exécution de la boucle.\n", i);

    return EXIT_SUCCESS;
}

/* definitions des fonctions utilisateurs */
```

1. Quelle est la signification de chaque expression figurant dans le `for` ? Quelles sont la ou les instructions qui composent le corps de la boucle ?

   ```
   for  ( instruction1; expression_logique ; instruction2 )
   {
       corps de la boucle : le bloc est défini par la séquence d'instructions
       figurant ici entre les accolades
   }
   ```

   — `instruction1` : elle sert généralement à initialiser la variable de boucle
   — `expression_logique` : ou expression_booleenne (elle s'évalue à vrai ou faux). On parle aussi parfois de la garde de la boucle ou de condition d'entrée (dans la boucle). Si elle est vraie (i.e. non nulle) on exécute le corps de la boucle, sinon, on passe à l'instruction située après la boucle `for`.
   — `instruction2` : prépare l'itération suivante ; il est important de comprendre que cette instruction doit forcément modifier la valeur de (au moins une variable, mais on ne leur parle que de la var de boucle) la variable intervenant dans l'expression booléenne car sinon la val de expression_booléenne est constante, et on ne sort jamais de la boucle

— `corps de la boucle` . Si le bloc est réduit à une seule instruction, les accolades sont facultatives

2. Faire la trace du programme. Qu'affiche le programme ?
   Remarque : L'exécution de $i = i + 1$ est placée ligne 18 pour montrer qu'elle est éxecutée à la fin du corps de la boucle, même si ce n'est pas clair relativement aux lignes effectives du programme.

```
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 5 fois le programme pour que la séquence affichée soit exactement celle des 5 cas suivants :
   — `0 1 2 3 4`
   — `1 2 3 4`
   — `1 2 3 4 5`
   — `1 3 5`
   — `(0,0) (1,1) (2,2).`
   Ici, les corrections peuvent se faire plus ou moins rapidement, en fonction de la compréhension des étudiants. Chaque séquence demande la modification d'un ou de plusieurs arguments ce qui doit vous permettre d'insister sur leurs rôles.
   Pour `1 2 3 4 5`, c'est soit i < 6, soit i <=5.
   Pour le dernier cas : `(0,0) (1,1) (2,2)`

```c
/* déclaration de fonctionnalités 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 */
```

Exercice 1.2 Affichage n fois de "Coucou"

Écrire un programme qui affiche n fois la chaîne de caractères `"Coucou\n"`.

On n'introduit pas nécessairement de variable n car on peut utiliser une constante dans le programme. (Ils vont voir les constantes symboliques en C bientôt). Mais si le groupe suit bien, on pourra lire n d'abord à la console avec `scanf` ou définir une constante.

```
/* déclaration de fonctionnalités 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 < 5; i = i + 1)
    {
        printf("Coucou\n");
    }
    /* i >= 5 */

    return EXIT_SUCCESS;
}

/* definitions des fonctions utilisateurs */
```

Exercice 1.3 Calcul de la somme des n premiers entiers $\Sigma_1^n i$

Écrire un programme qui calcule et affiche la somme des entiers de 1 à n, n étant un entier quelconque.

```
/* déclaration de fonctionnalités 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 somme = 0; /* élément neutre pour l'addition */
    int i;  /* var. de boucle */

    for(i = 1; i <= 4; i = i + 1)  /* i allant de 1 à 4 */
    {
        /* ajoute i à la somme partielle */
        somme = somme + i;
    }
    /* i > 4 */
```

```
    /* somme vaut 0 + 1 + 2 + 3 + 4 */
    printf("somme = %d\n", somme);

    return EXIT_SUCCESS;
}
```

Exercice 1.4 Suite du premier exercice (affichages de la variable de boucle)

1. Modifiez le programme du premier exercice afin que la séquence affichée soit :
   — `0 1 2 0 1 2.`
   — `0 1 2 0 1 2 3.`
   De combien de boucles avez-vous besoin ? De combien de variables de boucles avez-vous besoin ?
   Correction : on a besoin de 2 boucles à la suite, 1 seule variable (on la réutilise)
2. 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 avez-vous besoin ? Quelle est la différence de structuration des boucles entre le cas 1 et le cas 2 ?
   Réponse : 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 fonctionnalités 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 */
```

2 Equivalences entre boucle while et boucle for

Exercice 2.1 Boucles while et boucles for

Ecrire avec une boucle while les morceaux de code écrits avec une boucle for, et inversement.

1. ```
   for ( i = 0 ;  i  <  5 ;  i = i-1)
       printf("le compteur i vaut %d\n", i) ;
   ```
   ```

2.
```
while ( i >=0) {
     printf("le compteur i vaut %d\n", i) ;
     i = i-1 ;
}
```

Correction :

1.
```
i = 0;
while ( i < 5) {
    printf("le compteur i vaut %d\n", i) ;
    i = i + 1;
}
```
2.
```
for ( ; i >=0 ; i = i + 1)
     printf("le compteur i vaut %d\n", i) ;
```

(On pourra aussi remplacer les instructions `i = i + 1 ;` par `i++ ;`).