Informatique S1 – Programmation C
Exercices - Corrigés

TD 5 : Les boucles *for*

Dans ce TD, nous allons réaliser des exercices couvrant l'usage des boucles *for* et l'usage des boucles imbriquées dans le langage C.

Objectifs : présentation des opérateurs unaires ++, --, +=, -=, des boucles *for* et des boucles imbriquées.

- Opérateurs unaires ++, --, +=, -=
- Présentation boucle for
- Présentation boucles imbriquées
- Concepts complémentaires
  - Fonction de bibliothèque : getchar, putchar

Exercice 1 – Déclaration for

L'objectif de l'exercice est d'attirer l'attention vers la définition de la boucle for : *for (attribution ; test ; incrément) instructions*. On peut faire plusieurs attributions (séparées par des « , ») et différents incréments (aussi séparés par les « , ») dans les parties leur correspondant. L'exercice renforce la possibilité d'utilisés plusieurs attributions, et la possibilité d'utiliser les opérateurs unaires (« ++ ») pour l'incrément.

```
#include <stdio.h>

int main () {
   float x, y;  /* valeur fournis par l'utilisateur */
   float z;     /* z = x puissance y */
   int i;

   /* lecture des variables */
   printf ("Entrez x : ");
   scanf ("%f", &x);
   printf ("Entrez y : ");
   scanf ("%f", &y);

   /* on multiplie x y-fois */
   for (i=0, z=1; i < y; i++) {
        z = z * x;
   }

   /* presentation des resultats */
   printf ("x ^ y = %f \n", z);
}
```

Exercice 2 – while-for

Exercice essentiellement de traduction de while vers for, étant donné que la semaine dernière, ils ont fait en cours le même programme avec while et do-while.

```
#include <stdio.h>

/*  TD 5 : Factoriel de n
 *  Factoriel de n = 1 * 2 * ... * n-1 * n
 */

int main () {
```

```
  int n;
  int i;
  int fact;  /* factoriel de n */

  printf ("Entrez n : ");
  scanf ("%d", &n);

  for (i = 1, fact = 1; i <= n; i++ ) {
       fact = fact * i;
  }

  printf ("Factoriel de %d est %d \n", n, fact);
}
```

Exercice 3 – while-for

Idem l'exercice antérieur.

```
#include <stdio.h>

/*  TD 5 : calculer la moyenne d'yun exnemble de 10 numeros réels  */

int main () {
  float xi=0;
  int n=0;
  float somme = 0.0;
  float moy = 0.0;

  for (n=0; n<10; n++) {
      printf ("Entre un numero reel (x%d): ", n);
      scanf ("%f", &xi);
      somme = somme + xi;
  }

  moy = somme / 10;  //ou moy = somme / n;

  printf ("La moyenne est : %f \n", moy);
}
```

Exercice 4

Petit exercice rigole et facile qui fait un simple compte à rebours. Le détail ici est que le compte à rebours est inversé : on compte de i=n à i=0. Il nous faut utiliser donc for (i=n ; i >0 ; i--) au lieu de la traditionnel boucle for(i=0 ; i<n ; i++).

```
#include <stdio.h>
/*  TD 5 : Ecrivez un programme qui réalise un compte à rebours
    avant de se terminer. */
int main ()
{
   int n = 0;
   int i=0;
   printf ("Entrer un n. entier : ");
   scanf ("%d", &n);

   for (i=n; i>0 ; i--)
      printf ("... %d ...\n",i);

   printf ("!!!! BOOOOOMMMMMM !!!\n");

}
```

Exercice 6 – while-for

a) L'exercice demande d'écrire un programme capable d'afficher un tableau avec la conversion °C - °F de -15°C à +45°C. Cet exercice est l'opportunité de souligner deux faits : (i) la variable de contrôle du for n'est pas forcément un int (dans ce cas, il peut s'agir d'un float) ; (ii) l'attribution initiale du for n'est pas forcément « i=0 » ou « i=1 », c'est n'importe quelle valeur (dans ce cas, celc=-15).

```
#include <stdio.h>

/* Conversion C-F */
main () {
  float celc, fahr;

  printf ("Celsius\tFahrenheit \n");

  for (celc=-15; celc <= 45; celc++) {
      fahr = (celc*9)/5 + 32;
      printf (" %.1f \t %.2f\n", celc, fahr);
  }

}
```

b) Dans cet exercice, on demande aux étudiants d'afficher le même tableau, de 5 en 5 degrés. C'est l'opportunité de leur signaler que l'incrément n'est pas forcément « i++ », mais une affectation comme n'importe quelle autre (ici, on utilise celc = celc + 5).

```
#include <stdio.h>

/* Conversion C-F */
main () {
  float celc, fahr;

  printf ("Celsius\tFahrenheit \n");

  for (celc=-15; celc <= 45; celc=celc+5) {
      fahr = (celc*9)/5 + 32;
      printf (" %.1f \t %.2f\n", celc, fahr);
  }

}
```

Exercice 5 – Bloc d'instructions

L'objectif de cet exercice est de montre l'importance de l'indentation pour la visibilité du code source. Deux codes sont présentés. Le premier affiche correctement le carré et le cube des petits entiers, grâce à une boucle for. Le second affiche seulement le carré et le cube du nombre 5 à cause de la boucle for, qui ne compte pas un bloc définie par « { } » et qui donc contient uniquement la ligne « carre = a*a; ».

a) L'exécution du code A affiche les carrés et les cubes des nombres 1 à 5. Le code B affiche uniquement la ligne « 6 25 150 ». Leurs traces sont les suivants :

Code A	a	carre	cube
Pont d'observation 1	?	?	?
Pont d'observation 2	1	1	1
Pont d'observation 2	2	4	8
Pont d'observation 2	3	9	27

Pont d'observation 2	4	16	64
Pont d'observation 2	5	25	125
Pont d'observation 3	6	25	125

Code B	a	carre	cube
Pont d'observation 1	?	?	?
Pont d'observation 2	1	1	?
Pont d'observation 2	2	4	?
Pont d'observation 2	3	9	?
Pont d'observation 2	4	16	?
Pont d'observation 2	5	25	?
Pont d'observation 3	6	25	150

b) Différence entre les deux codes : code A contient une boucle for qui définit un bloc de 5 instructions, tandis que dans le code B la même boucle ne définit pas un bloc, exécutant uniquement une instruction

c) Le code A correspond mieux aux spécifications énoncées dans les commentaires.

Exercice 7 – Boucles imbriquées

a) Le programme affiche la table de multiplication de 1 à 10. Les trous dans le code correspondent uniquement aux boucles for. L'objectif est de montrer aux étudiants l'usage des boucles imbriqués.

```
#include <stdio.h>

int main () {
  int i, j ;
  int max = 10;

  /* affiche l'entete */
  printf ("      I");
  for (i=1; i<=max; i++)
     printf ("%4d", i);

  printf ("\n");
  printf ("--------");

  for (i=0; i<max; i++)
     printf ("----");

  printf ("\n");

  /* affiche les differents lignes */
  for (i=1; i<=max; i++) {
     printf ("%4d  I", i);
     for (j=1; j<=max; j++) {
         printf ("%4d", i*j);
     }
     printf ("\n");
  }
}
```

b) Pour afficher une table de multiplication jusqu'au numéro 15, il faut tout simplement modifier la valeur de la variable « max ». Rappel : les étudiants n'ont pas encore vu les « #define », cet exercice constitue une préparation pour la définition des constantes.

Exercice 8 – boucles imbriquées

```
#include<stdio.h>

/** TD 5: afficher des retangles et triangles
 * Source : poly de M. Troignon
 */

main()
{
    int x, y, i, j;
    printf("Entrez x svp : ");
    scanf("%i", &x);
    printf("Entrez y svp : ");
    scanf("%i", &y);

    /* exercice a: un retangle */
    for (i=1; i<=x; i++)
    {
        for (j=1; j<=y; j++)
        {
            putchar('x');
        }
        putchar('\n');
    }
    putchar('\n');


    /* exercuce c: un triangle */
    for (i=1; i<=x; i++) {
        for (j=1; j<=i; j++) {
            putchar ('x');
        }
        putchar ('\n');
    }
    putchar ('\n');

    /* exercice c: un triangle centralise */
    for (i=1; i<=x; i++)
    {
        for (j=1; j<=x-i; j++)
        {
            putchar(' ');
        }
        for (j=1; j<=2*i-1; j++)
        {
            putchar('x');
        }
        putchar('\n');
    }
}
```