Travaux dirigés 2 : premiers pas en C

Correction.

Note aux chargés de TD.
- Dans l'exercice 2, la fonction `printf` n'a pas encore été présentée. Ne pas rentrer dans les détails et limiter les explications au minimum (ne pas expliquer `%d` pour l'instant).
- Dernier exercice (carre/cube) : parler également de l'indentation. Si les `printf` passent mal, ne pas insister.

1 Identificateurs

Parmi les termes suivants quels sont ceux qui sont des identificateurs valides ? Dans le cas contraire expliquer pourquoi ce n'est pas le cas.

`rendez_vous`	`1par1`	`else`
`université`	`deux_par_deux`	`Else`
`conversion_binaire`	`caractère`	`Add-2-numbers`
`Licence`	`additionner_2_nombres`	`_var`

Correction. `université`, `caractère` ne sont pas valides : pas de caractère accentué.
`1par1` n'est pas valide : ne peut commencer que par une lettre (majuscule ou minuscule).

`Add-2-numbers` n'est pas valide : le tiret n'est pas un caractère valide pour construire des identificateurs.
`else` n'est pas valide : c'est un mot-clef du langage C.

Remarque : `Else` est autorisé (car le C fait la différence entre minuscule et majuscule). Néanmoins, un "warning" est généré.

2 Déclaration et affectation de variables impératives

Soit le programme suivant :

```
1  /* Declaration de fonctionnalites supplementaires */
2  #include <stdlib.h> /* EXIT_SUCCESS */
3  #include <stdio.h> /* printf */
4
5  /* Declaration des constantes et types utilisateurs */
6
7  /* Declaration des fonctions utilisateurs */
```

```
8
9   /* Fonction principale */
10  int main()
11  {
12      /* Declaration et initialisation des variables */
13      int x;
14
15      x = 3;
16      x = x + 1;
17      printf("x = %d\n", x);
18
19      /* valeur fonction */
20      return EXIT_SUCCESS;
21  }
22
23  /* Definitions des fonctions utilisateurs */
24
```

1. Que fait ce programme ?

   Correction.
   — Il déclare la variable impérative `x` (occupe un espace memoire)
   — Il affecte 3 à la variable `x`
   — Affecte à x la valeur de l'expression (x + 1) qui vaut 4 : la sous-expression x s'évalue comme la valeur de la variable (3), plus un.
   — Affiche la valeur de l'expression x qui s'évalue comme la valeur de la variable, c'est-à-dire 4.

2. Donner la trace du programme C. Pour cela vous utiliserez un tableau comportant 1 colonne pour le numéro de ligne + autant de colonnes que de variables utilisées dans le programme + 1 colonne pour l'affichage éventuel du programme.

   Correction. Il est très important qu'ils sachent faire une trace d'un programme C. Ils seront évalués dessus pendant les colles et aux examens. Le choix de la numérotation des lignes n'est pas imposé. On va d'habitude au plus court. Par contre, la numérotation doit être donnée explicitement dans le code.
   trace :

```
ligne          | x | sortie (affichage a l'ecran)
-----------------------------------------------
initialisation | ? |
15             | 3 |
16             | 4 |
17             |   |  x = 4
renvoie EXIT_SUCCESS
```

3 Ecriture d'un programme

1. Proposer un algorithme qui calcule le carré et le cube d'un valeur contenue dans une variable. Vous pourrez afficher le résultat.

Correction. On considère l'affichage comme la sortie du programme

```
Nom: Carre_Cube_de_X
Entree:
Sortie:
Algorithme
 Soit 3 variables entières X, carreX, cubeX
 X = 4
 carreX = X * X
 cubeX = carreX * X
 afficher carreX
 afficher cubeX
 Retour: OK
```

2. Ecrire le programme correspondant à l'algorithme

```c
/* Declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include<stdio.h>

/* Declaration des constantes et types utilisateurs */

/* Declaration des fonctions utilisateurs */

/* Fonction principale */
int main()
{
    /* Declaration et initialisation des variables */
    int x = 5;
    int carreX, cubeX;

    carreX = x * x;
    cubeX = carreX * x;

    printf("carre de %d: %d\n", x, carreX);
    printf("cube de %d: %d\n", x, cubeX);
    /* valeur fonction */
    return EXIT_SUCCESS;
}

/* Definitions des fonctions utilisateurs */
```

3. Effectuer la trace du programme

Correction. Trace : voir figure 1

Ligne	x	carreX	cubeX	sortie
initialisation	5	?	?	
17		25		
17			125	
19				carre de 5 : 25
20				cube de 25 : 125
22	renvoie `EXIT_SUCCESS`			

FIGURE 1 – Trace du programme C