TD 6 Programmation – DUT 1 – Révisions

P. Courtieu

Exercice 1 (Question de cours)

On veut écrire une fonction `estPositif` qui teste si son argument (un entier) est strictement supérieur à zéro. On doit pouvoir utiliser cette fonction dans un test comme suit :

```
int n = lireInt(); /* demande une chaîne de caractères à l'utilisateur*/
if (estPositif(a)) {
   printf("strictement positif\n");
} else {
   printf("négatif ou nul\n");
}
```

Question 1.1

Que doit retourner cette fonction ? Pourquoi ?

Question 1.2

Écrivez cette fonction, n'oubliez pas de déclarer le type de retour et les arguments. Vous aurez les points si le code ci-dessus peut marcher avec votre fonction.

Exercice 2 (Question de cours)

Pour chacun des morceaux de codes suivants répondez aux questions suivantes :

1. est-il accepté par le compilateur ? Si non justifiez en quelques mots.
2. si oui est-il correct (pas de risque de plantage à l'exécution)) ? Si non justifiez en quelques mots.
3. si les deux réponse ci-dessous son « oui », alors donnez la valeur de la variable `x` à la fin du morceau de programme.

Question 2.1

```
int x = "toto";
```

Question 2.2

```
int x = 6.9;
```

Question 2.3

```
int n = 12.2;
float x = (n + 7.8);
```

Question 2.4

```
int s = "toto";
char x = s[2];
```

Question 2.5

```
int s = "toto";
char x = s[4];
```

Exercice 3 Boucles simples

Question 3.1

Écrivez une procédure **void** `fromTo(int d, int f)` qui affiche tous les entiers entre `d` et `f` (on suppose `d<f`, sinon la procédure n'affiche rien).

Par exemple, `fromTo(4,12);` doit afficher :

```
4
5
6
7
8
9
10
11
12
```

Question 3.2

Écrivez une procédure **void** `fromToStep(int d, int f,int n)` qui se comporte comme `fromTo` ci-dessus mais affiche seulement les entiers de `n` en `n`. Autrement dit la procédure doit afficher `d`, puis `d`+3, `d` + 6 etc sans dépasser `f`.

Par exemple, `fromToStep(4,12,3);` doit afficher :

```
4
7
10
```

Question 3.3

Écrivez une fonction **int** `compteZeros(int t [], int taille)` qui retourne le nombre de cases contenant la valeur `0` dans le tableau t (de longueur `taille`).

Exercice 4 Boucles imbriquées

REMARQUE : Dans la suite les tableaux à double entrées sont considérés comme des tableaux de lignes. Donc la 2^e case de la 4^e ligne est désignée par `t[3][1]`.

Question 4.1

Écrivez une procédure **`void`** `carre(`**`int`** `n)` qui affiche `n` lignes de largeur `n`, remplies de caractères `'-'`.
Par exemple, `carre(4);` doit afficher :

```
----
----
----
----
```

Question 4.2

Écrivez une procédure **`void`** `compteA(`**`char`** `t [][10], `**`int`** `nblignes)` qui retourne le nombre d'occurrences du caractère 'a' dans le tableau à double entrées `t`. Le tableau est supposé avoir `nblignes` lignes, chacune de longueur 10.

Question 4.3

Écrivez une fonction **`int`** `toutesLettre(`**`char`** `t [][10], `**`int`** `nblignes)` qui retourne `1` si le tableau `t` (à double entrées) contient au moins une occurrence de chaque lettre de l'alphabet (minuscule), et `0` sinon. Le tableau est supposé avoir `nblignes` lignes, chacune de longueur 10.

Note : Vous aurez un point bonus si vous ne parcourez qu'une seule fois le tableau.