Licence Informatique 2014-2015 **J1INAW02 - Programmation 1**
Université Bordeaux **Devoir Surveillé 1h20**

Tous documents interdits

Barème indicatif : Exercice 1 (2), Exercice 2 (4), Exercice 3 (14)
La clareté et la lisibilité du texte seront prises en compte lors de la notation.

Exercice 1. Questions de cours (répondre en 5 lignes maximum)

1. Expliquer ce que représentent l'interface et l'implémentation d'un module.
2. Quel est l'intérêt de la compilation séparée ?

Exercice 2. Fonctions *"echanger"*

On considère les fonctions suivantes :

```
void echange1(int x, int y)
{
  int tmp;

  tmp = x;
  x   = y;
  y   = tmp;
}
```

```
void echange2(int *x, int *y)
{
  int tmp;

  tmp = *x;
  *x   = *y;
  *y   = tmp;
}
```

```
void echange3(int *x, int *y)
{
  int *tmp;

  tmp = x;
  x   = y;
  y   = tmp;
}
```

```
void echange4(int *x, int *y)
{
  int *tmp;

  *tmp = *x;
  *x   = *y;
  *y   = *tmp;
}
```

Expliquer ce qui sera affiché après exécution de chacun des blocs suivants :

1. `{int a = 1, b = 2; echange1(a,b); printf("a = %d, b = %d\n",a,b);}`
2. `{int a = 1, b = 2; echange2(&a,&b); printf("a = %d, b = %d\n",a,b);}`
3. `{int a = 1, b = 2; echange3(&a,&b); printf("a = %d, b = %d\n",a,b);}`
4. `{int a = 1, b = 2; echange4(&a,&b); printf("a = %d, b = %d\n",a,b);}`

Exercice 3. Module *temps*

Il s'agit de proposer un module bibliothèque permettant la gestion du temps. Un temps est une variable entière de type `long int` contenant des secondes calculées à partir d'un temps 0.

Le module aura les fonctions suivantes :
- `convertirEnMinutes` qui convertit un temps passé en paramètre en un nombre entier de minutes.
- `convertirEnHeures` qui convertit un temps passé en paramètre en un nombre entier d'heures.
- `afficherTemps` qui affiche le temps en secondes sous la forme "Le temps est de `temps` secondes".

1. **Implémentation du module**
   (a) Le temps sera défini comme un nouveau type `temps` qui redéfini le type `long int`. Donner l'instruction permettant cette opération.
   (b) Ecrire l'interface du module.
   (c) Ecrire l'implémentation du module.

2. **Programme de test**

   Il s'agit d'écrire un programme de test `test-temps.c` permettant de manipuler un tableau (alloué dynamiquement) de `temps`.
   Les temps seront passés en ligne de commande de la façon suivante : `./test-temps temps1 temps2 temps3...`.

   (a) Écrire une première version de la fonction `main` affichant un tableau de `temps` **alloué dynamiquement**. La fonction `main` appellera, si nécessaire, une fonction d'usage qu'on écrira plus tard.
   Observation : *seul le tableau de `temps` sera alloué dynamiquement.*

   (b) Quels sont les fichiers d'en-tête nécessaires à la compilation de cette première version du programme de test ?

   (c) Écrire une fonction
   `void echangerTemps(temps* tabTemps, int tailleTab, int ix1, int ix2)`
   permettant d'échanger deux `temps` qui se trouvent aux indices `ix1` et `ix2` dans le tableau `tabTemps` (remarquer la présence de la taille du tableau parmi les paramètres : elle doit donc être utilisée par la fonction).

   (d) Écrire une fonction `estCroissant` qui teste si le tableau est ordonné dans l'ordre croissant.

   (e) Afin de faciliter la lecture d'un temps, il est préférable de l'afficher sous forme d'horaire. Un horaire est défini par l'heure, les minutes et les secondes. Un horaire sera conservé dans une structure définie dans le programme de test. Expliciter cette structure.

   (f) Écrire une fonction `convertirHoraire` permettant de convertir un `temps` en un `horaire`.

   (g) Écrire la fonction `afficherHoraire` qui affiche un `horaire` sous la forme :
   heures h *minutes* min *secondes* sec.

   (h) Écrire les instructions qui complètent la fonction `main` afin de permettre de tester les fonctionnalités précédentes. Donner tous les fichiers d'en-tête nécessaires à la compilation.

   (i) Écrire la fonction d'usage de `./test-temps`.

   (j) Quelles sont les commandes qui permettent d'obtenir un exécutable du programme de test ?