Introduction à l'informatique

TP3 - Les fonctions

Vos fonctions doivent être documentées, et testées si possibles avec des `ASSERT`. À chaque fois, on écrira un programme principal qui utilise les fonctions créées.

Ce TP sera fait sur deux séances, et est long. Si toutefois vous sentez que vous allez rapidement arriver à son terme, contactez rapidement Joël Gay.

Exercice 1 (Première fonction)**.**
Le fichier `ex1.cpp` génère des erreurs lors de la compilation. Trouver et corriger ces erreurs afin d'obtenir le comportement attendu.

Exercice 2 (Fonction simple : la calculatrice)**.**
Compléter le programme `ex2.cpp` en vous appuyant sur la documentation et les tests fournis. Compléter les tests jusqu'à ce que vous ayez entièrement confiance en votre code.

Exercice 3 (Fonction sans valeur de retour)**.**
Implémenter la fonction permettant d'afficher une valeur booléenne dans la console que vous avez peut-être vue en TD. Cette fonction pourra vous être utile pour d'autres exercices. Pour rappel, cette fonction prend en entrée une valeur booléenne et affiche :

- **"VRAI"** si cette valeur est égale à `true`.
- **"FAUX"** si cette valeur est égale à `false`.

Remarque *: une telle fonction ne peut être testée avec* ***`ASSERT`*** *car elle ne renvoie (mot clé* ***`return`****) rien !*

Exercice 4 (Nombres premiers)**.**
(Cet exercice implémente un exercice du TD que vous n'avez peut-être pas eu le temps de faire.)

(1) Écrire une fonction qui prend en argument un entier n et teste si n est un nombre premier (c'est-à-dire renvoie `true` si n est premier et `false` sinon. On rappelle qu'un nombre est premier s'il a exactement deux diviseurs distincts : 1 et lui-même.) Écrivez des tests à cette fonction.

(2) Écrire une fonction qui prend en argument un entier n et affiche tous les nombres premiers entre 1 et n.

(3) Écrire une fonction qui affiche les n premiers nombres premiers.

(4) Peut-on tester les deux dernières fonctions ? Exécutez les simplement dans le `main`.

Exercice 5 (Comparaison de nombres réels)**.**
L'opérateur == n'est pas très utile à cause des erreurs d'arrondis dans le cas des nombres réels. Dans cet exercice, vous allez implémenter une fonction qui teste l'égalité entre deux nombres réels. Par définition, on dit que x et y sont égaux à ε près si :

$$|x - y| \leq \varepsilon|x| \qquad \text{et} \qquad |x - y| \leq \varepsilon|y|$$

(où ε est généralement un nombre très petit).

- Implémenter votre propre fonction `absolue` qui calcule la valeur absolue d'un réel de type `double` (on aurait pu utiliser la fonction `abs` de la librairie `cmath`, mais dans cet exercice, il vous est demandé de créer votre propre fonction `absolue`). Vous n'écrirez pas les tests de la fonction `absolue` pour l'instant.
- Implémenter la fonction `egal` qui teste l'égalité entre deux réels passés en paramètres à ε près, en utilisant la formule ci-dessus et la fonction `absolue` que vous avez implémentée.
- Écrire les tests qui vérifient que pour $\varepsilon = 10^{-7}$ (ça se note `1e-7`), les deux nombres 15.999999 et 16.0 sont considérés comme égaux par la fonction, et que pour $\varepsilon = 10^{-8}$, les deux nombres 15.999999 et 16.0 sont considérés comme différents par la fonction.
- Écrire des tests qui vérifient le bon fonctionnement de la fonction `absolue`. Vous voudriez écrire par exemple `ASSERT(absolue(-2.5) == 2.5)`, mais comme on vient de le voir, le symbole `==` marche mal pour les réels. Ainsi, au lieu d'utiliser `==`, vous utiliserez la fonction `egal` pour effectuer les comparaisons entre ces deux nombres réels (vous pourrez utiliser un ε de 10^{-5}).
- Écrire un programme principal qui exécute les tests définis précédemment et qui affiche le résultat de l'égalité de 1.000001 et 0.999999 avec $\varepsilon = 10^{-5}$ et $\varepsilon = 10^{-7}$. Vous pourrez utiliser la fonction qui affiche la valeur d'un booléen implémentée dans l'exercice précédent.

Exercice 6 (Fonctions imbriquées)**.**

(1) Définir une variable globale `PI` égale à 3.1415926535. *(Rq : on peut ajouter le mot-clé* `const` *avant sa déclaration pour définir que c'est une constante.)*

(2) Écrire 6 fonctions différentes qui :
   - calcule le périmètre d'un cercle en fonction de son rayon,
   - calcule la surface d'un cercle en fonction de son rayon,
   - calcule le volume d'une sphère en fonction de son rayon,
   - calcule le périmètre d'un carré en fonction de son côté,
   - calcule la surface d'un carré en fonction de son côté,
   - calcule le volume d'un cube en fonction de son côté.

(3) Écrire une fonction de test qui teste les 6 fonctions précédentes. En particulier, vous vérifiez que, pour un :
   - cercle/sphère de rayon 2 : perimetre=12.56, surface=12.56 et volume=33.5.
   - carré/cube de côté 2.25 : perimetre=9, surface=5.07 et volume=11.4.

(4) Écrire une fonction qui aura 2 paramètres : un réel `a` et un entier `type` qui prendra la valeur 1 ou 2.

- Si `type` vaut 1, la fonction affichera le périmètre et la surface du cercle de rayon `a`, ainsi que le volume de la sphère de rayon `a`.
- Si `type` vaut 2, la fonction affichera le périmètre et la surface du carré de coté `a`, ainsi que le volume du cube de coté `a`.

Si l'entier `type` vaut 1 ou 2 et si `a` ≥ 0, la fonction effectuera les affichages demandés, puis retournera `true`. Si l'entier `type` est différent de 1 ou 2 ou si a < 0, la fonction affichera un message d'erreur, puis retournera `false`.

(5) Écrire un programme principal qui exécute la fonction de test de la question 3, puis qui demande à l'utilisateur de saisir le type et la distance du coté/rayon et qui affiche les informations attendues (comme dans l'image ci-dessous). Ce programme redemandera à l'utilisateur de re-saisir les informations si le type ou la distance ne sont pas corrects. Il demandera enfin à l'utilisateur s'il souhaite recommencer ou non, et recommencera dans le cas échéant.

```
Saisir le type (1 pour cercle ou 2 pour carre) et la valeur du rayon ou du cote :1 4
Le perimetre d'un cercle de rayon 4 est : 25.1327
La surface d'un cercle de rayon 4 est : 50.2655
Le volume d'une sphere de rayon 4 est : 268.083
Voulez-vous recommencer ? (o/n) :o
Saisir le type (1 pour cercle ou 2 pour carre) et la valeur du rayon ou du cote :3 7
Erreur : type non defini !Le type doit etre 1 pour cercle ou 2 pour carre
Saisir le type (1 pour cercle ou 2 pour carre) et la valeur du rayon ou du cote :2 -8
Erreur : la distance a doit etre un nombre positif
Saisir le type (1 pour cercle ou 2 pour carre) et la valeur du rayon ou du cote :2 5
Le perimetre d'un carre de cote 5 est : 20
La surface d'un carre de cote 5 est : 25
Le volume d'un cube de cote 5 est : 125
Voulez-vous recommencer ? (o/n) :n
```

Exercice 7 (Passage par références)**.**
On souhaite écrire un programme qui calcule les racines d'un polynôme du 2^{nd} degré :

- Écrire une fonction qui calcule les racines d'un polynôme du second degré. Cette fonction prendra en paramètre les 3 coefficients du polynôme, ainsi que deux variables qui seront utilisées comme paramètre de sortie permettant à la fonction de retourner les racines (si elles existent). Par ailleurs, la fonction retournera comme valeur de retour le nombre de racines du polynôme.
- Écrire le programme principal qui demande à l'utilisateur de saisir les 3 coefficients et affiche le ou les racines comme dans l'image ci-dessous. Il demandera ensuite à l'utilisateur s'il souhaite recommencer ou non, et recommencera dans le cas échéant.

```
Entrez les 3 coefficients du polynome :
1 2 3
Ce polynome n'a pas de racine !

Voulez-vous recommencer ? (o/n)o
Entrez les 3 coefficients du polynome :
36 -30 6
Ce polynome a deux racines egales a : 0.5 et : 0.333333

Voulez-vous recommencer ? (o/n)o
Entrez les 3 coefficients du polynome :
9 -162 729
Ce polynome a une racine unique egale a : 9

Voulez-vous recommencer ? (o/n)n
```

Exercice 8 (Fonction récursive)**.**
On veut calculer la somme des n premiers carrés pour tout n supérieur ou égal à 0. Par exemple, si n vaut 3, ce sous-programme calculera $1^2 + 2^2 + 3^2$.

- Écrire une fonction récursive qui calcule cette somme (ainsi que la documentation et les tests qui vont avec).
- Écrire le programme principal qui calcule et affiche la sommes des n premiers carrés pour ($n = 4$ et $n = 16$).

Exercice 9 (Comparaison entre itératif et récursif)**.**
On veut calculer le terme de rang n de la suite de Fibonacci de deux façons différentes : itérative et récursive. Le fichier `ex9.cpp` contient un programme principal ainsi que la documentation et la signature des fonctions qui permettront de faire ce calcul des deux façons. Pour rappel, la suite de Fibonacci est telle que :

$$U_0 = 1 \quad et \quad U_1 = 1 \quad et \quad U_n = U_{n-1} + U_{n-2}$$

- Écrire la fonction itérative `fibonacciIter` qui calcule le terme de rang n de la suite de Fibonacci en utilisant une boucle `for`, n étant un entier positif passé en paramètre de la fonction. Écrire aussi les tests pour cette fonction dans `testFibonacciIter`.
- Écrire une fonction récursive `fibonacciRec` qui calcule le terme de rang n de la suite de Fibonacci, n étant un entier positif passé en paramètre de la fonction. Écrire aussi les tests pour cette fonction dans `testFibonacciRec`.
- Exécuter le programme principal. Que vaut U_{50} ? Pourquoi ? Modifier vos fonctions pour corriger ce problème.
- Exécuter à nouveau le programme principal. Qu'observez-vous au niveau des performances ?
- Exécuter pas à pas (sur papier) les deux fonctions `fibonacciIter` et `fibonacciRec`. Essayer de trouver une explication à ces différences de performance.

****

EXERCICES SUPPLEMENTAIRES

****

Exercice ♠ 10 (Opérations sur les chaînes de caractères)**.**
On souhaite réaliser un programme qui détermine le nom et le prénom d'un étudiant à partir de son adresse email. Par exemple, à partir de l'adresse email `thomas.dupont@u-psud.fr`, le programme déterminera que le nom est `Dupont` et le prénom est `Thomas`. Pour cela, il vous est demandé de réaliser un ensemble de fonctions qui effectuent des opérations sur les chaînes de caractères. Vous êtes libre de choisir si les paramètres doivent être passés par valeur ou par référence pour ces fonctions.

Pour vous aider, vous pourrez utiliser les deux fonctions suivantes de la librairie standard sur les chaînes de caractères :

```
string s = "bonjour" ; // soit une chaîne de caractères s

int l = s.length()     // retournera le nombre de caractères de s
                       // ici, l prendra la valeur 7

char c1 = s.at(0);     // retournera le caractère à la position donnée
char c2 = s.at(3);     // le premier caractère est à l'indice 0
                       // ici, c1 prendra la valeur 'b' et c2 la valeur 'j'
```

Nous verrons ça avec les tableaux, mais `s.at(0)` *est une variable comme une autre. On peut notamment l'affecter (*`s.at(0) = 'c'`*).*

- Écrire une fonction qui prend en paramètre une chaîne de caractères et un caractère particulier, et qui donne l'indice de la première occurrence du caractère dans la chaîne de caractères. Par exemple, si on utilise la fonction avec `"bonjour"` et `'b'`, la fonction donnera 0 (premier indice).
- Écrire une fonction qui coupe une chaîne de caractères en deux sous-chaînes à un indice donné. Le caractère à l'indice donné sera ignoré. Par exemple, si on utilise la fonction avec `"bonjour"` et l'indice 3, la fonction donnera `"bon"` et `"our"`. Cette fonction rendra vrai si le découpage en deux sous-chaînes est possible, faux sinon.
- Écrire une fonction qui coupe une chaîne de caractères en deux sous-chaînes au niveau de la première occurrence d'un caractère. Le caractère utilisé pour la découpe sera ignoré. Par exemple, si on utilise la fonction avec `"bonjour"` et le caractère `'o'`, la fonction donnera `"b"` et `"njour"`. Cette fonction rendra vrai si le découpage en deux sous-chaînes est possible, faux sinon. La fonction devra ré-utiliser les deux fonctions précédentes.
- Écrire une fonction qui ajoute une majuscule dans une chaîne de caractères passée en paramètre à un indice donné. Par exemple, si on utilise la fonction avec `"bonjour"` et l'indice 3, la fonction donnera `"bonJour"`. Cette fonction rendra vrai si l'ajout de la majuscule est possible, faux sinon. La fonction pourra ré-utiliser certaines des fonctions précédentes.
  Les `char` *sont en fait codés par des entiers (c'est le code ASCII). Les lettres majuscules sont comprises entre 65 et 90. Les minuscules entre 97 et 122. On passe des majuscules aux minuscules en ajoutant 32 et en stockant le résultat dans un nouveau* `char`*.*
- Écrire le programme principal qui demande l'utilisateur une adresse email et qui affiche le nom et le prénom de l'étudiant auquel correspond cette adresse email.

Exercice ♠ 11.
Ceci est l'exercice 4 du TD4.

(1) Écrire une fonction qui prend en entrée une date sous la forme de trois entiers jour / mois / année, et teste si c'est une date valide. Pour l'instant, on ignore les années bissextiles. Par exemple :
   — `date_valide(28, 5, 1973)` renvoie `true`
   — `date_valide(31, 2, 2015)` renvoie `false`

(2) Écrire une fonction qui prend en argument une date valide sous la forme de trois entiers et affiche la date du lendemain. Par exemple :
`jour_suivant(18, 12, 2021)` affiche `Le jour suivant est le 19 12 2021.`

(3) Reprendre la question 1 en considérant les années bissextiles (une année est bissextile si elle est divisible par 4, mais pas 100 sauf si elle est divisible par 400).

(4) Reprendre la question 2 pour que le programme n'affiche pas mais modifie les arguments passés par référence.

(5) Écrire une fonction qui prend en argument une date valide et renvoie le jour de la semaine de cette date.

(6) Combien de 1er du mois ont été des dimanches au XX e siècle ? (on donne que le 1er janvier 1901 était un mardi).
Pour les curieux cela correspond au problème Projet Euler 19 qui donne plein de problèmes pouvant être résolus informatiquement.

Exercice ♠ 12 (Modélisation d'un problème et notion d'algorithme)**.**
Soient 2 cruches de capacité respectives 5 et 7 litres. Ces cruches ne sont pas graduées. Le but de ce problème est que l'une des deux cruches contienne 4 litres.

(1) Formalisez le problème en termes de variables et d'actions à effectuer. Trouvez une suite d'actions permettant d'atteindre une solution.

(2) Implémentez l'algorithme établi sous la forme d'un programme en langue C++. Ce programme devra afficher la suite d'actions à effectuer pour obtenir le résultat, comme ci-dessous :

```
Entrez le nombre de litre(s) final souhaite:4
cruche 5l = 0 & cruche 7l = 0
cruche 5l = 0 & cruche 7l = 7 <---  Remplir la cruche de 7l
cruche 5l = 5 & cruche 7l = 2 <---  Transvaser celle de 7l dans celle de 5l
cruche 5l = 0 & cruche 7l = 2 <---  Vider la cruche de 5l
cruche 5l = 2 & cruche 7l = 0 <---  Transvaser celle de 7l dans celle de 5l
cruche 5l = 2 & cruche 7l = 7 <---  Remplir la cruche de 7l
cruche 5l = 5 & cruche 7l = 4 <---  Transvaser celle de 7l dans celle de 5l
Solution trouvee : la cruche de 7l contient 4 litre(s)
```

(3) Peut-on faire une suite d'actions qui permette de trouver une façon d'avoir l'une des deux cruches qui contienne 1 litre ? 2 litres ? 3 litres ? 4 litres ? 5 litres ? 6 litres ? 7 litres ?

(4) Adaptez votre programme pour qu'il demande à l'utilisateur un nombre entre 1 et 7 (inclus), et qui affiche la suite d'actions à effectuer pour obtenir le résultat.