Introduction à l'informatique
TD3 - Premières fonctions

Exercice 1 (Premières fonctions)**.**
Voici une fonction qui calcule la surface d'un rectangle :

```
float surfaceRectangle(float longueur, float largeur) {
        return longueur * largeur;
}
```

(1) Implanter une fonction `surfaceDisque` qui calcule la surface d'un disque de rayon donné. On prendra $\pi = 3.1415926$.

(2) Implanter une fonction `surfaceTriangle` qui calcule la surface d'un triangle de base b de hauteur h.

Exercice 2 (Variables locales/globales)**.**
On considère les deux programmes suivants :

```
int i = 0;

int f(int j) {
    i = i + j;
    return i;
}

int main() {
    cout << i << endl;
    cout << f(1) << endl;
    cout << f(2) << endl;
    cout << f(3) << endl;
}
```

```
int f(int j) {
    int i = 0;
    i = i + j;
    return i;
}

int main() {
    cout << i << endl;
    cout << f(1) << endl;
    cout << f(2) << endl;
    cout << f(3) << endl;
}
```

(1) Quelle est la différence entre les deux programmes ?

(2) Une ligne du deuxième programme est incorrecte : le compilateur déclencherait une erreur. Laquelle ? La supprimer.

(3) Exécuter pas à pas les deux programmes en décrivant au fur et à mesure l'état de la mémoire et ce qui est affiché à l'écran.

(4) Décrire la différence de comportement et retrouver dans les notes de cours le commentaire à ce propos.

Exercice 3 (fonction puissance)**.**
On souhaite écrire une fonction `puissance` qui permet de calculer la valeur d'un nombre entier `a` à la puissance `b` :

(1) Écrire la signature de la fonction.

(2) Écrire la documentation de la fonction.

(3) Écrire les tests de la fonction sans oublier les cas particuliers.

(4) Écrire la fonction.

Exercice 4 (La trilogie code, documentation, tests)**.**
Analyser la fonction `volumePiscine` suivante :

```
/** Calcule le volume d'une piscine parallélépipédique
*  @param profondeur la profondeur de la piscine (en mètres)
*  @param largeur la largeur de la piscine (en mètres)
*  @param longueur la longueur de la piscine (en mètres)
*  @return le volume de la piscine (en litres)
**/
double volumePiscine(double profondeur, double largeur, double longueur) {
        return 100 * profondeur * largeur * longueur;
}
```

Munie des tests :

```
ASSERT( volumePiscine(5, 12, 5) == 30000 );
ASSERT( volumePiscine(1,  1, 5) == 500   );
```

(1) Est-ce que les tests passent ?

(2) Est-ce que la documentation, le code et les tests sont cohérents ?

(3) Corriger les anomalies éventuelles.

Exercice 5 (Fonction sans valeur de retour : affichage de booléens)**.**
Vous avez peut-être déjà remarqué en TP que l'affichage des booléens n'est pas très bon. Si on demande à afficher une variable de type bool dont la valeur est :

- `false` (par exemple : `cout << false << endl ;`), alors le programme affiche **0**.
- `true` (par exemple : `cout << true << endl ;`), alors le programme affiche **1**.

Écrivez une fonction qui prend en entrée une valeur booléenne et qui affiche :

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

Exercice ♠ 6 (Utilisation de fonctions)**.**
Le but de cet exercice est de coder une fonction `point_de_chute` qui calcule l'abscisse x_c à laquelle tombe un projectile lancé en $x = 0$ avec une vitesse v suivant un angle α (exprimé en degrés par rapport à l'horizontale). Implantez la fonction `point_de_chute`. On commencera par écrire sa documentation ainsi que des tests.
Rappels :

- *l'abscisse est donnée par la formule : $x_c = \frac{2v_x v_y}{g}$ où $v_x = v\cos(\alpha)$, $v_y = v\sin(\alpha)$ et g est l'accélération gravitationnelle (environ 9.8 $m.s^{-2}$ sur la planète Terre).*
- *en `C++`, les fonctions mathématiques sinus et cosinus sont implantées par les fonctions prédéfinies `sin(arg)` et `cos(arg)` dans `<cmath>`, où l'angle `arg` est exprimé en radians. La constante π se nomme `M_PI` dans cette bibliothèque.*

Exercice ♠ 7 (Fonction mystère)**.**
Analyser la fonction `mystere` suivante :

```
string mystere(int blop) {
        string schtroumpf = "";
        for ( int hip = 1; hip <= blop; hip++ ) {
                for ( int hop = 1; hop <= hip; hop++ ) {
                        schtroumpf += "*";
                }
                schtroumpf += "\n";
        }
        return schtroumpf;
}
```

Munie des tests suivants :

```
        ASSERT( mystere(0) == "");
        ASSERT( mystere(1) == "*\n");
        ASSERT( mystere(2) == "*\n**\n");
        ASSERT( mystere(3) == "*\n**\n***\n");
```

(1) Comment appelle-t-on cette fonction (quelle est sa *syntaxe*) ? Quelle est sa signature ?

(2) Que fait cette fonction (quelle est sa *sémantique*) ?
Indications : pour les chaînes de caractères, l'opérateur `+` représente la concaténation (e.g. `"Cou" + "cou"` s'évalue en `"Coucou"`) ; `x += expression` est un raccourci pour `x = x + expression`; dans une chaîne de caractères, `'\n'` représente un saut de ligne.

(3) Choisir un bon nom pour cette fonction et ses variables et en écrire la documentation.

Exercice ♠ 8 (Nombres premiers)**.**

(1) Écrire une fonction qui prend en argument (entrée) 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.)

(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.

Exercice ♠ 9.
Le but de cet exercice est de calculer la hauteur en fonction du temps $z(t)$ à laquelle se trouve un pot de fleur ($m = 3$kg) lâché à $t = 0$ depuis le 10ème étage ($h_0 = 27$m), en chute libre avec résistance de l'air ; puis de calculer le temps de chute.

(1) Implantez une fonction `chute_libre(t)` calculant $z(t)$ pour un V_0 donné, de valeur $V_0 = 80ms^{-1}$.
Indications :

- La hauteur s'exprime en fonction du temps par
$$z(t) = h_0 - \left(V_0 t + \frac{V_0^2}{g} \ln \left(\frac{1}{2}(1 + e^{-2tg/V_0}) \right) \right),$$
où V_0 est la vitesse limite de chute de l'objet et $g = 9.81ms^{-2}$.
- La fonction logarithme népérien est prédéfinie sous la forme `log(arg)` dans `<cmath>`.

(2) Que se passe-t-il si on varie h_0 et V_0 ? Généralisez votre fonction pour prendre en paramètres additionnels la hauteur initiale h_0 et la vitesse limite de chute V_0. Pour la gravité, définir une variable globale g. **Bonus** : définir cette variable globale comme une constante (nous irons sur Mars une autre fois).
Écrivez les appels à la fonction précédente pour calculer $z(t)$ pour différentes valeurs de V_0 (10, 40, 60, 120, 180).

(3) Écrivez une fonction `temps_de_chute` qui prend les mêmes paramètres que précédemment et utilise `chute_libre` de façon répétée et renvoie une approximation de la durée t_c de la chute du pot de fleur jusqu'au sol.

(4) La vitesse limite peut être obtenue en fonction de la masse volumique de l'air ρ, du coefficient de résistance aérodynamique C_x et de la section de l'objet S à l'aide de la formule $V_0 = \sqrt{\frac{2mg}{C_x \rho S}}$. Implantez une fonction `vitesse_limite` pour calculer cette formule. Puis implantez de nouvelles fonctions utilisant les précédentes pour calculer $z(t)$ et le temps de chute t_c en fonction des paramètres h_0, S, et m. (*On donne que la racine est codée par la fonction* `sqrt` *dans* `<cmath>`)