TP 2 - Fonctions, Pointeurs, Tableaux

Votre TP comportera deux projets *Code :: Blocks* : une pour chaque partie. De plus, chacun des deux projets contiendra trois fichiers :
- le fichier `main.c` (généré automatiquement par Code : :Blocks si vous choisissez Console application), et ne contenant que la fonction main (pas d'autres fonctions). Dans cette fonction main vous répondrez à chacun des exercices les uns à la suite des autres, en testant chacune des fonctions comme demandé. Essayez de séparer correctement chaque exercice tant dans le code que dans l'affichage lors de l'exécution.
- un fichier `fonctions.h` contenant les signatures (déclarations) des fonctions écrites.
- un fichier `fonctions.c` contenant les définitions (corps) des fonctions de `fonctions.h`.

Pour cela, une fois votre projet créé, sélectionnez le et allez dans *File > New > File....* Sélectionnez C/C++ header, donnez lui le nom (extension *.h*) ainsi que l'emplacement voulus et confirmez. Répetez l'opération en sélectionnant C/C++ Source et en lui donnant l'extension *.c*.
Vous remarquerez qu'en principe, Code : :Blocks a généré automatiquement les instruction de pré-processing sur le header `#ifndef FONCTIONS_H_INCLUDED` etc...

PARTIE 1 : FONCTIONS

Exercice 1 Deux nombres entiers positifs sont dits amis si chacun est égal à la somme des diviseurs propres de l'autre plus un. Par exemple, 220 et 284 sont amis puisque :
- diviseurs propres de 220 : 2, 4, 5, 10, 11, 20, 22, 44, 55, 110
- diviseurs propres de 284 : 2, 4, 71 ,142
- 1+2+4+5+10+11+20+22+44+55+110=284
- 1+2+4+71+142=220

Écrivez une fonction C (`int sommeDiviseursPropres(int n)`) qui calcule et retourne la somme des diviseurs propres d'un entier n. Écrivez une fonction C (`int ami(int a,int b)`) qui renvoie 1 si les deux nombres a et b sont amis et 0 sinon.
Dans le main, testez votre fonction `ami` avec plusieurs valeurs.

Exercice 2 (Équation du second degré)

1. Écrire une fonction `discriminant` qui prend comme paramètres les coefficients a, b et c de ax^2+bx+c et qui renvoie le discriminant $b^2 - 4ac$.
2. Écrire une fonction `nombreRacines` qui prend comme paramètres les coefficients a, b et c de ax^2+bx+c et qui renvoie le nombre de racines réelles distinctes de l'équation (utilisez la fonction `sqrt` de la librairie math.h pour calculer la racine carrée).
3. Écrire une fonction `afficheSolution` qui prend comme paramètres les coefficients a, b et c de $ax^2 + bx + c$ et qui affiche le nombre et les racines de l'équation. Cette fonction ne retourne donc rien.
4. Dans le main, testez votre fonction nombreRacines avec plusieurs polynômes.

Exercice 3 Le nombre π peut être approché en utilisant la suite de Leibnitz :

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + ...$$

On est assuré que l'erreur sur le calcul est inférieur à $\frac{1}{p}$ si le dernier terme ajouté à la somme est inférieur en valeur absolue à $\frac{1}{p}$. Écrire une fonction qui prend en paramètre un réel $\epsilon < 1$ et qui calcule la valeur de $\frac{\pi}{4}$ avec une erreur inférieure à ϵ.
Testez ensuite cette fonction dans le main, avec plusieurs valeurs de ϵ.

Exercice 4 On désigne par C_n^p le nombre de combinaisons sans répétitions de p objets parmi n. On peut les calculer de deux manières différentes (entre autres).

$$C_n^p = C_{n-1}^p + C_{n-1}^{p-1}$$

$$C_n^p = \frac{p}{n} C_{n-1}^{p-1}$$

avec $C_k^k = 1$ et $C_n^0 = 1$

1. Utiliser la première relation de récurrence pour écrire une fonction récursive `combinaisons1` qui calcule et retourne C_n^p
2. Ecrire une seconde fonction `combinaisons2` à partir de la seconde relation.
3. Testez les deux fonctions dans le main.
4. A votre avis, quelle est la meilleure façon de faire (plus rapide) ? Vous pouvez par exemple modifier les deux fonctions afin de les faire afficher des résultats intermédiaires.

PARTIE 2 : POINTEURS ET TABLEAUX

Exercice 5

1. Ecrire une fonction `Identique` qui prend deux paramètres de type `int*` et qui teste si ils référencent la même zone mémoire.
2. Ecrire une fonction `ValeurIdentique` qui prend deux paramètres de type `int*` et qui teste si les valeurs pointées sont identiques.
3. Dans le main, testez les fonctions `ValeurIdentique` et `Identique` et montrez qu'elles ne sont pas équivalentes.

Exercice 6 Ecrire une fonction `echange` qui permette d'échanger les valeurs de deux entiers reçus en paramètre (elle ne retourne rien). La tester dans le main.

Exercice 7 Ecrire une fonction `saisieTemps` prenant trois paramètres (heure, minute, seconde) permettant de saisir une heure à l'aide d'un seul appel à `scanf` qui initialisera les trois variables reçues en paramètre. La tester dans le main en vérifiant que les trois variables ont bien été modifiées (les afficher par exemple).
Bonus : la fonction doit demander à l'utilisateur de saisir une heure tant qu'elle n'est pas valide (exemple : les minutes supérieures à 60 etc...).

Exercice 8 Ecrire une fonction `afficheTab` qui prend en paramètre un tableau dynamique (et sa taille) et qui affichage les éléments du tableau (la fonction ne retourne donc rien).
Vous êtes encouragés à utiliser cette fonction dans les exercices suivants afin d'afficher vos tableaux.

Exercice 9 Écrivez une fonction `numOccurences` qui prend en paramètre un tableau de 10 entiers, un entier x et qui calcule le nombre d'occurrences de x dans le tableau. Testez votre fonction dans le main.

Exercice 10 Ecrivez une fonction `produitScalaire` prenant en argument deux tableaux dynamique u et v de même taille (la taille sera également en paramètre) représentant deux vecteurs de taille d et qui calcule et retourne leur produit scalaire. On rappelle que le produit scalaire de $\vec{u}$ et $\vec{v}$ est égal à :

$$\vec{u}.\vec{v} = \sum_{i=1}^{d} u_i \times v_i.$$

Tester cette fonction dans le main.

Exercice 11 Écrire une fonction `ecritureBinaire` qui prend en paramètre un entier x positif et un tableau T de 8 entiers, et qui calcule et stocke le codage binaire de x dans T. Par exemple, si $x = 39$, le tableau doit valoir $T = [0, 0, 0, 1, 0, 1, 1, 1]$ car $39 = 0\times2^7+0\times2^6+0\times2^5+1\times2^4+0\times2^3+1\times2^2+1\times2^1+1\times2^0$.

Exercice bonus :

Exercice 12 Une chaîne de caractère est représentée en C par un tableau de caractères (`char`). En utilisant cela, définir les fonctions :

1. `prefixe` qui prend 2 chaînes de caractères `ch1` et `ch2` et qui teste si `ch1` est un préfixe de `ch2` (exemple : ['c', 'h', 'a', 't'] est un préfixe de ['c', 'h', 'a', 't', 'e', 'a', 'u']).
2. `palindrome` qui prend 1 chaîne de caractères `ch1` et qui teste si `ch1` est un palindrome.

Ces deux fonctions retournent un entier représentant le résultat du test (1 pour vrai et 0 pour faux).
Testez ces deux fonctions dans le main (indication : on peut définir un tableau de caractères ainsi : `char mot[]="plateau";`).