C++ - TP 5
Les fonctions (suite) (4h)

1 Panel de fonctions

Dans cet exercice, vous allez implémenter plusieurs fonctions qui traitent des nombres entiers. Une fois ces fonctions codées, vous créerez un menu qui permet de les appeler.

1.1 Nombre premiers entre eux

A faire : Ecrire une fonction qui prend deux entiers en paramètres et qui retourne un booléen disant si les deux nombres sont premiers entre eux.

Pour rappel, deux nombres sont premiers entre eux s'ils n'ont aucun diviseur en commun.

1.2 Valeur approchée de racine carrée

La méthode de Newton permet de calculer la valeur approchée la racine carrée d'un nombre entier b. Elle utilise la propriété ci-dessous. On considère la suite :

$$x_{n+1} = \frac{1}{2}(x_n + \frac{b}{x_n})$$

initialisée par $x_0 = 1$. Alors cette suite converge vers $\sqrt{b}$.

A faire : Ecrivez une fonction qui prend en paramètre un entier b et qui retourne sa racine carrée (format double) à 10^{-6} près. Testez votre fonction.

1.3 Liste vers 9

On considère un nombre à deux chiffres $N = a_1a_0$ tel que $a_1 \neq a_0$. A partir de ce nombre, on procède à l'opération suivante : on considère le nombre symétrique de N, qu'on appelera $\tilde{N}$, et défini par $\tilde{N} = a_0a_1$. On calcule maintenant l'écart $|N - \tilde{N}|$. Cet écart est ajouté à la liste vers 9 de N, et on recommence l'opération à partir de ce nombre, jusqu'à tomber sur l'écart 9 qui sera le dernier nombre de la liste.

Exemple. Voici la liste vers 9 du nombre 31 :

$$31 \to 18 \to 63 \to 27 \to 45 \to 9$$

A faire : Ecrivez une fonction qui prend un entier N en paramètre et qui affiche la liste vers 9 de cet entier. Si N n'est pas un nombre à 2 chiffres, ou si ses deux chiffres sont identiques, vous indiquerez un message d'erreur. Testez votre fonction.

1.4 Menu

A faire : Codez maintenant un menu qui permette de lancer chacune des trois fonctions précédentes, ou bien de quitter le programme. Le menu se réaffiche après chaque exécution d'une fonction. Par exemple :

```
Faites votre choix:
1. Racine carrée
2. Liste vers 9
3. Nombres premiers entre eux
4. Quitter
```

2 Inclusion de fichier

Dans le fichier `TP5_data.h`, disponible sous Claco, vous trouverez la déclaration d'une constante et d'un tableau. Vous allez importer le fichier `TP5_data.h` et utiliser les données de ce fichier :

```
#include "TP5_data.h"
```

Attention : Vérifiez bien la syntaxe : ce sont des guillemets et non des chevrons autour de `TP5_data.h`.

Avec une telle inclusion, les données du fichier `data.h` sont considérées comme des données globales au programme (comme si vous aviez fait un copier-coller au-dessus du `main`). Vous pouvez donc les utiliser sans les recopier dans votre programme, et surtout, **il ne faut pas les redéclarer**.

A faire : Vous devez créer un programme qui comportera les fonctions suivantes :

- une fonction `compte` qui reçoit un entier en paramètre et retourne le nombre de valeurs supérieures et inférieures à ce paramètre (les valeurs égales ne sont pas comptabilisées) dans le tableau,
- une fonction `moyenne` qui calcule la valeur moyenne du tableau.
- un programme principal qui teste les différentes fonctions.

3 Triangle de Pascal

Le triangle de Pascal est une représentation des coefficients binomiaux (C_n^k) dans un triangle. Le triangle de Pascal se présente sous la forme :

		k								
		1	2	3	4	5	6	7	8	9
	1	1								
	2	1	1							
	3	1	2	1						
	4	1	3	3	1					
n	5	1	4	6	4	1				
	6	1	5	10	10	5	1			
	7	1	6	15	20	15	6	1		
	8	1	7	21	35	35	21	7	1	
	9	1	8	28	56	70	56	28	8	1

On remarque que chaque case de coordonnées (k, n) est calculée en faisant la somme $(k-1, n-1)+(k, n-1)$ (l'élément au-dessus de (k, n) et son prédécesseur). Ainsi à partir d'une ligne n, on peut calculer la ligne $n+1$. Exemple : La case $(3, 6)$ ($k = 3$ et $n = 6$) contient la valeur 10. Elle est calculée en faisant la somme des cases $(2, 5)$ (valeur 4) et $(3, 5)$ (valeur 6).

A faire : Ecrire un programme qui demande à l'utilisateur une valeur n et appelle une fonction `trianglePascal` qui affiche le triangle de Pascal ayant n lignes. Comme nous n'utilisons que des tableaux statiques pour le moment, vous déclarerez en début de programme une constante entière globale (notée `N`) qui représentera la taille maximum des tableaux utilisés. Le programme principal vérifiera que l'entier saisi par l'utilisateur soit strictement inférieur à `N` avant d'appeler la fonction.

Algorithme : 2 approches sont possibles :

— utiliser 2 tableaux : T qui contient la ligne $i-1$ et T' qui contient la ligne i à calculer. L'algorithme devient alors

```
initialiser T à {1,0,0,...,0};
pour i allant de 2 à n faire
        calculer T';
        recopier T' dans T;
finpour
afficher T;
```

— utiliser 1 seul tableau (à vous de réfléchir à l'algo)

Pour les plus rapides... (*"As-tu remarqué ? Quiconque est plus lent que toi est un idiot et quiconque est plus rapide est un crétin."*, George Carlin)

4 La clé du cadenas

Bruno est fier de son nouveau cadenas et de la combinaison à six chiffres qu'il a trouvée. Il se permet même de donner les indications suivantes à José : "Supposons que l'on écrive une liste d'entiers dans l'ordre naturel 1,2,3,4,5... Le premier chiffre de la combinaison est le chiffre qui vient, dans l'ordre d'écriture, juste après le millième chiffre 9 écrit ; le second chiffre de la combinaison est celui qui vient après le 2000e chiffre 9, et ainsi de suite jusqu'au 6e chiffre qui est, bien sûr, celui qui suit immédiatement le 6000e chiffre 9 !". Bruno ne se doute pas que José trouvera la combinaison !

A faire : Et vous, serez-vous capable d'écrire un programme qui puisse retrouver cette combinaison ?

Astuce : vous pouvez découper le programme en plusieurs fonctions :

— `int taille(int n)` qui retourne le nombre de chiffres qui compose le nombre n.

— `void codage(int n,int tab[],int &longueur)` qui stocke les chiffres du nombre n dans les premières cases du tableau *tab*, et met à jour le nombre de cases utilisées dans le paramètre *longueur*.

— Le programme principal, qui pour chaque nombre n, calcule son codage, et lit le tableau *tab* ainsi modifié pour compter les 9 et répondre au problème.

5 Permutation de multiples

On peut voir que le nombre 125874 et son double 251748 contiennent exactement les mêmes chiffres (en même nombre) mais dans un ordre différent.

Question : Quel est le plus petit entier positif X tel que $2X$, $3X$, $4X$, $5X$ et $6X$ contiennent les mêmes chiffres (en même nombre) ?

Pour cela nous vous proposons d'implémenter les fonctions suivantes :

— `void compteChiffres(int n, int t[])`. Cette fonction reçoit un nombre `n` à étudier et un tableau de taille 10 (qui sera un tableau de compteurs à remplir). La fonction compte le nombre d'occurrences de chaque chiffre de `n` et complète le tableau (`t[0]` contient le nombre de 0, `t[1]` contient le nombre de 1, etc). Exemple : si `n` vaut 2653, le tableau `t` est remplit pour contenir

indice	0	1	2	3	4	5	6	7	8	9
valeur	0	0	1	1	0	1	1	0	0	0

— `bool compareTab(int t1[], int t2[])`. Cette fonction reçoit 2 tableaux de taille 10 et compare les tableaux. S'ils sont identiques (même valeur dans chaque case), la fonction retourne `true`, sinon elle retourne `false`.

Algorithme : L'algorithme principal devient donc

```
Soit un nombre entier N = 1
Tant que le résultat n'est pas trouvé
  - compter les chiffres de N et mettre dans T
  - pour tout multiple M de N (de 2 à 6)
    - compter les chiffres de M et mettre dans T'
    - comparer T et T'
  - si pour tous les multiples la comparaison est OK, on a la solution
    sinon N++
FinTantque
```

6 Décomposition en produit de facteurs premiers

6.1 Décomposition

N'importe quel nombre entier naturel (non nul) peut-être décomposé en un produit de nombres premiers (décomposition en facteurs premiers). Par exemple $45 = 3 \times 3 \times 5$. En factorisant, on peut dire ce nombre est décomposable en un produit de puissances de nombres premiers (dans note exemple, $45 = 3^2 \times 5^1$).
Quelques exemples :

- $45 = 2^0 \times 3^2 \times 5^1$
- $360 = 2^3 \times 3^2 \times 5^1$
- $63 = 2^0 \times 3^2 \times 5^0 \times 7^1$
- $113400 = 2^3 \times 3^4 \times 5^2 \times 7^1$
- $3667356 = 2^2 \times 3^5 \times 5^0 \times 7^3 \times 11^1$

Les termes en puissance 0 peuvent être omis puisqu'ils valent 1.

A faire : Ecrire un programme qui demande un nombre n à l'utilisateur et affichez la décomposition du nombre en un produit de puissances de nombres premiers

6.2 PGCD : Plus Grand Commun Diviseur

Le PGCD de deux nombres entiers A et B supérieurs ou égaux à 2 a pour décomposition en facteurs premiers le produit des facteurs premiers apparaissant à la fois dans la décomposition de A et de B munis du plus petit des exposants trouvés dans la décomposition de A et de B.
Exemple : si $A = 2^3 \times 3^4 \times 5^2 \times 7^1 = 113400$ et $B = 2^2 \times 3^5 \times 7^3 \times 11^1 = 3667356$, alors $pgcd(A, B) = 2^2 \times 3^4 \times 7^1 = 2268$.

A faire : Ecrire un programme qui demande 2 nombres à l'utilisateur, affiche leur décomposition en facteurs premiers ainsi que le PGCD de ces 2 nombres (et sa décomposition).

6.3 PPCM : Plus Petit Commun Multiple

Le PPCM de deux nombres entiers A et B supérieurs ou égaux à 2 a pour décomposition en facteurs premiers le produit des facteurs premiers apparaissant dans A ou dans B munis du plus grand des exposants trouvés dans la décomposition de A et de B.
Exemple : si $A = 2^3 \times 3^4 \times 5^2 \times 7^1 = 113400$ et $B = 2^2 \times 3^5 \times 7^3 \times 11^1 = 3667356$, alors $ppcm(A, B) = 2^3 \times 3^5 \times 5^2 \times 7^3 \times 11^1 = 183367800$.

A faire : Modifier votre programme pour qu'il affiche également le PPCM des 2 nombres saisis (et sa décomposition).