Introduction à l'informatique

TP4 - Les tableaux et structures

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 (Fonction simple et tableaux)**.**

(1) Écrire une fonction permettant de calculer le carré d'un nombre entier donné comme paramètre.

(2) Utiliser la fonction précédente dans un programme principal qui demandera à l'utilisateur d'entrer une valeur `val` au clavier comprise entre 1 et `VALMAX`=200. Le programme calculera les valeurs successives des carrés des valeurs entre 0 et `val` et les mémorisera dans un tableau `res`.

(3) Compléter votre programme de sorte que chaque valeur du tableau précédent devienne (pour `indice` > 0) : `res.at(indice) = indice`^{2} `- (indice-1)`^{2}. Vous réutiliserez les valeurs des carrés mémorisés dans le tableau **sans recalculer ces valeurs (sans recalculer les carrés de nombres)**.
On obtiendra par exemple la sortie suivante :

```
Entrez un entier entre 1 et 200 : 6
Lecture de l'entier : 6
0^2 = 0
1^2 = 1
2^2 = 4
3^2 = 9
4^2 = 16
5^2 = 25
6^2 = 36
res.at(6) = 36 - 25 = 11
res.at(5) = 25 - 16 = 9
res.at(4) = 16 - 9 = 7
res.at(3) = 9 - 4 = 5
res.at(2) = 4 - 1 = 3
res.at(1) = 1 - 0 = 1
```

Exercice 2 (Fonctions sur les tableaux)**.**

(1) Écrire une fonction qui crée un tableau d'entier de type `vector` et de taille `n` où `n` est un paramètre de la fonction. Elle demande ensuite à l'utilisateur d'initialiser les cases du tableau une par une en entrant une valeur au clavier. Enfin, elle retourne le tableau ainsi créé.

(2) Écrire une fonction qui permet d'afficher un tableau de type `vector` donné comme paramètre. On affichera 5 éléments par lignes séparés par des points virgules.

(3) Écrire une fonction qui calcule la moyenne des éléments d'un tableau de type `vector` passé en paramètre et qui retourne un `double` représentant cette moyenne.

(4) Écrire une fonction qui retourne le plus grand des éléments du tableau d'entier de type `vector` passé en paramètre.

(5) Écrire une fonction qui retourne vrai ou faux suivant que le tableau d'entier de type `vector` passé en paramètre est trié dans l'ordre croissant.

(6) Écrire une fonction qui retourne vrai ou faux suivant que le tableau d'entier de type `vector` passé en paramètre est trié dans l'ordre décroissant.

(7) Écrire une fonction qui prend en paramètre un tableau d'entiers de type `vector` et qui rend un nouveau tableau qui est une copie triée en ordre croissant du tableau passé en paramètre. On utilisera la méthode du tri bulle :

```
PROCEDURE Tri_bulle (Tableau A)
    n = taille de A
    passage = 0
    REPETER
        permutation = FAUX
        POUR i allant de 0 à n - 1 - passage (exclus) FAIRE
            SI A.at(i) > A.at(i+1) ALORS
                echanger A.at(i) et A.at(i+1)
                permutation = VRAI
            FINSI
        FIN POUR
        passage = passage + 1
    TANT QUE (permutation == VRAI)
```

Écrire un programme principal définissant un tableau de 7 entiers. Le programme principal initialisera le tableau, affichera le tableau, calculera et affichera la moyenne des éléments du tableau, affichera la valeur max du tableau et qui affichera si le tableau est trie par ordre croissant, ou par ordre décroissant, ou n'est pas trié.
Enfin, le programme principal effectuera le tri du tableau et affichera le tableau trié :

```
Initialisation du tableau t2......
Donner la valeur de tab.at(0) : 8
Donner la valeur de tab.at(1) : 5
Donner la valeur de tab.at(2) : 2
Donner la valeur de tab.at(3) : -1
Donner la valeur de tab.at(4) : -3
Donner la valeur de tab.at(5) : 6
Donner la valeur de tab.at(6) : 7


.........................affichage de t2......
8 ; 5 ; 2 ; -1 ; -3 ;
6 ; 7 ;
La moyenne des elements de t2 est de : 3
Le plus grand des elements de t2 a comme valeur :  8
Le tableau t2 n'est pas trie ....
le tableau trie vaut :
-3 ; -1 ; 2 ; 5 ; 6 ;
7 ; 8 ;
```

Exercice 3 (Recherche dans un tableau)**.**

(1) Écrire une fonction `bool contient(vector<int> tab, int x)` retournant vrai si le tableau `tab` contient l'élément `x` et faux sinon.

(2) Écrire une fonction
`vector<int> intersection(vector<int> tab1, vector<int> tab2)` retournant un tableau des éléments communs de `tab1` et `tab2` (sans répétition).

(3) Écrire une fonction qui prend en paramètre un tableau d'entiers et un nombre `x` et qui retourne deux tableaux d'entiers : le premier contiendra tous les éléments strictement inférieurs à `x` et le deuxième contiendra tous les éléments strictement supérieurs à `x`.

(4) Écrire un programme principal qui vérifie le bon fonctionnement de ces fonctions.

Exercice 4 (Structures Point2D et Segment)**.**

(1) Définir une structure `Point2D` qui représente un point dans un repère 2D à partir de son abscisse x et de son ordonnée y.

(2) Définir une structure `Segment` qui représente un segment composé de 2 points.

(3) Écrire une fonction qui calcule la distance entre deux points.

(4) Écrire une fonction qui calcule la longueur d'un segment
(on pourra ré-utiliser la fonction de la question 3).

(5) Écrire une fonction qui calcule le milieu d'un segment.

(6) Écrire un programme principal qui initialise un segment, calcule sa longueur, et détermine le point qui se trouve au milieu du segment.

Exercice 5 (Tableaux à deux dimensions)**.**
Quelques fonctions sur les tableaux à deux dimensions :

(1) Écrire une fonction qui affiche les éléments d'un tableau d'entiers à deux dimensions.

(2) Écrire une fonction qui compte le nombre d'éléments plus grands que 10 dans un tableau d'entiers à deux dimensions.

(3) Écrire une fonction qui teste si un élément `x` appartient à un tableau `tab` d'entiers à deux dimensions et si oui, retourne les coordonnées i et j du dernier élément égal à `x` dans le tableau (dernier dans l'ordre lexicographique (ligne, colonne)).

(4) Écrire le programme principal qui initialise un tableau d'entiers à deux dimensions de taille 3×5 et qui utilise les trois fonctions précédentes.

Exercice 6 (Carre magique)**.**
Un carré magique est un carré rempli de nombres qui, lorsque l'on en fait la somme sur chaque ligne, colonne ou diagonale, donne le même résultat. Par exemple :

Ordre 3 :

8	1	6
3	5	7
4	9	2

Ordre 4 :

4	14	15	1
9	7	6	12
5	11	10	8
16	2	3	13

(1) Écrire une fonction qui teste si un tableau 2D d'entiers passés en paramètre est un carré magique.

(2) Écrire une fonction qui affiche un tableau 2D d'entiers passés en paramètre, puis affiche si ce tableau est magique ou non.

(3) Écrire le programme principal qui initialise deux tableaux 2D d'entiers et affiche s'ils se sont des carrées magiques ou non.

```
Le carre :
8 1 6
3 5 7
4 9 2
est un carre magique !

Le carre :
8 1 6
3 5 7
6 9 2
n'est pas un carre magique !

Le carre :
8 1 6
3 5 7
4 9 2 3
n'est pas un carre magique !

Le carre :
4 14 15 1
9 7 6 12
5 11 10 8
16 2 3 13
est un carre magique !
```

Exercice ♠ 7 (Saisie de notes)**.**

Écrire un programme qui demande le nombre de notes à saisir de 1 à 100. Les notes sont des réels de 0.00 à 20.00. Le programme affichera la moyenne, la variance et l'écart type des notes entrées.
Remarque :

— Variance : moyenne des carrés des écarts à la moyenne

— Ecart type : racine carrée de la variance

(1) Tracer un diagramme vertical et afficher la ou les fréquences maximale(s). (On pourra utiliser le symbole tabulation ^pour aligner correctement.)

(2) Il tracera un histogramme horizontal : la hauteur d'une colonne pour une note donnée étant égale au nombre d'étudiants ayant eu cette note. Les notes apparaissent en abscisse, les nombres d'étudiants en ordonnée. (Attention à l'espacement qui dépend du nombre de chiffres d'un nombre)

Exemple d'affichage :

```
Saisie du nombre de notes :
Veuillez saisir un entier entre 1 et 100 : 20
Saisie des notes individuelles :
Veuillez saisir un reel entre 0 et 20 : 1
Veuillez saisir un reel entre 0 et 20 : 2
Veuillez saisir un reel entre 0 et 20 : 2
Veuillez saisir un reel entre 0 et 20 : 2
Veuillez saisir un reel entre 0 et 20 : 2
Veuillez saisir un reel entre 0 et 20 : 2
Veuillez saisir un reel entre 0 et 20 : 3
Veuillez saisir un reel entre 0 et 20 : 5
Veuillez saisir un reel entre 0 et 20 : 5
Veuillez saisir un reel entre 0 et 20 : 5
Veuillez saisir un reel entre 0 et 20 : 5
Veuillez saisir un reel entre 0 et 20 : 5
Veuillez saisir un reel entre 0 et 20 : 20
Veuillez saisir un reel entre 0 et 20 : 20
Veuillez saisir un reel entre 0 et 20 : 20
Veuillez saisir un reel entre 0 et 20 : 20
Veuillez saisir un reel entre 0 et 20 : 20
Veuillez saisir un reel entre 0 et 20 : 15
Veuillez saisir un reel entre 0 et 20 : 6
Veuillez saisir un reel entre 0 et 20 : 0
```

```
Resultats:
        moyenne : 8 / 20
        ecart type : 7.53658
 0       |*
 1       |*
 2       |*****
 3       |*
 4       |
 5       |*****
 6       |*
 7       |
 8       |
 9       |
 10      |
 11      |
 12      |
 13      |
 14      |
 15      |*
 16      |
 17      |
 18      |
 19      |
 20      |*****


La frequence max est : 5 intervalle [2, 3 [
La frequence max est : 5 intervalle [5, 6 [
La frequence max est : 5, atteinte pour la note maximale : 20



        *       *                                       *
        *       *                                       *
        *       *                                       *
        *       *                                       *
  *  *  *  *     *  *                        *           *
  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

Exercice ♠ 8 (Calcul des nombres premiers par le crible d'Eratosthène)**.**

Un nombre est dit premier, s'il admet exactement 2 diviseurs distincts (lui-même et l'unité) : 1 n'est donc pas premier. On désigne sous le nom de crible d'Eratosthène (vers 276 av. J.-C) une méthode de recherche des nombres premiers plus petits qu'un entier naturel n donné.

Comment faire ?

(1) On écrit la liste de tous les nombres de 1 jusqu'à N.

(2) On élimine 1.

(3) On souligne (on garde) 2 et on élimine tous les multiples de 2.

(4) Puis on fait de même avec 3.

(5) On remarque que le nombre suivant : 4 a été éliminé lors du traitement de 2. On choisit alors le plus petit nombre non éliminé (ici 5), on le souligne (garde) et on élimine tous ses multiples.

(6) On réitère le procédé jusqu'à avoir traité les N entiers (variation : trouvez une meilleure borne supérieure qui garantit qu'on a bien tous les nombres premiers voulus).

Les nombres soulignés (i.e. non éliminés) sont les nombres premiers jusqu'à N.
Pour simuler informatiquement ce crible, on utilisera un tableau de N entiers qui mémorise, selon une convention à définir, si le nombre correspondant est éliminé ou non. On utilisera N = 3000.

Vous devez obtenir ceci (attention la liste est incomplète) :

```
2   3   5   7   11   13   17   19   23   29
31   37   41   43   47   53   59   61   67   71
73   79   83   89   97   101   103   107   109   113
127   131   137   139   149   151   157   163   167   173
179   181   191   193   197   199   211   223   227   229
233   239   241   251   257   263   269   271   277   281
283   293   307   311   313   317   331   337   347   349
353   359   367   373   379   383   389   397   401   409
419   421   431   433   439   443   449   457   461   463
467   479   487   491   499   503   509   521   523   541
547   557   563   569   571   577   587   593   599   601
607   613   617   619   631   641   643   647   653   659
661   673   677   683   691   701   709   719   727   733
739   743   751   757   761   769   773   787   797   809
811   821   823   827   829   839   853   857   859   863
877   881   883   887   907   911   919   929   937   941
947   953   967   971   977   983   991   997   1009   1013
```

Exercice ♠ 9 (Entiers positifs parfaits)**.**

Écrivez un programme qui imprime la liste de tous les entiers positifs ”parfaits” inférieurs à 10 000.

Un entier est dit ”parfait” s'il est égal à la somme de tous ses diviseurs, sauf lui-même bien sûr. Les deux premiers nombres parfaits sont 6 et 28 : 6 = 1+3+2, 28 = 14+2+7+4+1.

Attention : il y a très peu de nombres parfaits.

Pour chaque nombre décrété parfait par votre programme, on peut afficher à l'écran la liste de ses diviseurs (faites le, en essayant de ne pas calculer deux fois la liste des diviseurs !) et vérifier à la main que la somme est bien celle attendue.

Exercice ♠ 10 (Problème du voyageur de commerce)**.**
Un voyageur de commerce veut visiter 20 villes définies par leurs noms dans un tableau :

```
vector<string> nom_villes = {"Auxerre", "Bordeaux", "Le Mans", "Lens",
 "Lille", "Lorient", "Lyon", "Marseille", "Monaco", "Nancy", "Nantes",
 "Nice", "Paris", "Rennes", "Saint Etienne", "Sedan", "Sochaux", "Toulouse",
 "Troyes", "Valenciennes"};
```

Il souhaite visiter les villes en supposant que la route qui relie ces villes est une ligne droite. Ceci permet de calculer facilement une distance entre 2 villes étant données leur longitudes et leurs latitudes **en degrés** :

```
struct Coordonnees {
  double lon;
  double lat;
};
vector<Coordonnees> coordonnes_villes = { {47.47, 3.34}, {44.53, 0.34},
{48.0, 0.11}, {50.25, 2.49}, {50.38, 3.3}, {47.44, -3.21}, {45.45, 4.49},
{43.17, 5.22}, {43.44, 7.25}, {48.41, 6.11}, {47.12, -1.33}, {43.42, 7.16},
{48.51, 2.20}, {48.6, -1.41}, {45.26, 4.23}, {49.42, 4.56}, {47.30, 6.49},
{43.36, 1.26}, {48.17, 4.4}, {50.21, 3.31} };
```

Le voyageur démarre par exemple de Paris et veux visiter les autres villes et revenir a Paris en minimisant son trajet. Une approche (non optimale mais néanmoins souvent acceptable) consiste à choisir comme prochaine ville de destination la ville la plus proche non encore visitée.

Le calcul entre 2 villes de latitude lat1 et lat2 et de longitude long1 et long2 exprimées **en radian** est donné par la formule :

```
Distance = RAYON_TERRE * acos(cos(lat1_rad) * cos(lat2_rad)
           * cos(long2_rad - long1_rad) + sin(lat1_rad) * sin(lat2_rad)))
```

On définira `RAYON_TERRE = 6366` et on pourra utiliser la constante `M_PI` de la libraire `<cmath>` pour les conversions de degrés en radians.

Par exemple, si le voyageur de commerce part de Paris, vous devriez obtenir le résultat suivant :

```
On part de Paris et on va a... Auxerre, soit un voyage de 171.359 kms, pour un total de 171.359 kms
On part de Auxerre et on va a... Troyes, soit un voyage de 141.039 kms, pour un total de 312.399 kms
On part de Troyes et on va a... Sedan, soit un voyage de 139.597 kms, pour un total de 451.995 kms
On part de Sedan et on va a... Valenciennes, soit un voyage de 164.185 kms, pour un total de 616.181 kms
On part de Valenciennes et on va a... Lille, soit un voyage de 18.8896 kms, pour un total de 635.07 kms
On part de Lille et on va a... Lens, soit un voyage de 91.146 kms, pour un total de 726.216 kms
On part de Lens et on va a... Le Mans, soit un voyage de 363.843 kms, pour un total de 1090.06 kms
On part de Le Mans et on va a... Rennes, soit un voyage de 181.563 kms, pour un total de 1271.62 kms
On part de Rennes et on va a... Nantes, soit un voyage de 164.632 kms, pour un total de 1436.25 kms
On part de Nantes et on va a... Lorient, soit un voyage de 211.882 kms, pour un total de 1648.14 kms
On part de Lorient et on va a... Bordeaux, soit un voyage de 509.917 kms, pour un total de 2158.05 kms
On part de Bordeaux et on va a... Toulouse, soit un voyage de 165.36 kms, pour un total de 2323.41 kms
On part de Toulouse et on va a... Saint Etienne, soit un voyage de 391.594 kms, pour un total de 2715.01 kms
On part de Saint Etienne et on va a... Lyon, soit un voyage de 35.7434 kms, pour un total de 2750.75 kms
On part de Lyon et on va a... Marseille, soit un voyage de 265.126 kms, pour un total de 3015.88 kms
On part de Marseille et on va a... Nice, soit un voyage de 217.31 kms, pour un total de 3233.19 kms
On part de Nice et on va a... Monaco, soit un voyage de 10.2398 kms, pour un total de 3243.43 kms
On part de Monaco et on va a... Sochaux, soit un voyage de 434.084 kms, pour un total de 3677.51 kms
On part de Sochaux et on va a... Nancy, soit un voyage de 129.652 kms, pour un total de 3807.16 kms
Et enfin on rentre a Paris, soit un voyage de 434.572 kms, pour un total de 4241.74 kms
```