CORRIGES

Cahier de TD n° 2

Cahier de TD n° 2

THEME 1 : Tableaux : définition de tableaux, taille maximum, taille utile, affichage, saisie.

Vrai / faux

Définition et syntaxe

saisie / affichage de tableaux

THEME 2 : tableaux : utilisation et algorithmes classiques

Recherche de la deuxième plus grande valeur d'un tableau

Suppression de doublons

Fusion de tableaux triés

THEME 3 : tableaux de caractères : travailler avec du texte

Message personnalisé

Noms de fichiers

THEME 4 : tableaux à plusieurs dimensions

Calcul matriciel

THEME 1 : Tableaux : définition de tableaux, taille maximum, taille utile, affichage, saisie.

Vrai / faux :

répondez aux questions suivantes par vrai ou faux

- Un tableau a toujours une taille maximum **V**
- Un tableau a toujours une taille utile **V 0 << tailleUtile <= tailleMaximum**
- Un tableau se manipule comme une variable 'classique' **F**

Un tableau tab est un pointeur constant dont la valeur est l'adresse du début du tableau qui est &tab[0]. tab <==> &tab[0]

Quelque soit i dans [0, taille_maximale] tab + i <==> &tab[i]

```
int tab[7]; /* pointeur constant */
int * p; /* pointeur variable */
int x = 3;
tab[3] = 23;
tab[2] = 12;
```

tab est un pointeur constant,

tab = p; FAUX ça ne compile pas

tab++; FAUX

tab = tab + 3; FAUX

tab = tab - 7; FAUX

```
p = &x;
```

p = p + 1; // TERRITOIRE INTERDIT <==> entier qui suit x en mémoire!!!!!

compile mais pb à l'exécution

ARITHMETIQUE DES POINTEURS EST TRANSPARENTE

```
p = &tab[2]; /* ou p = tab + 2 */
/* repère relatif du pointeur: p[0] vaut tab[2] soit 12, p[1] vaut tab[3] soit 23*/
p = p + 3; /* p pointe sur tab[2 + 3] soit tab[5], p vaut &tab[5] */
```

p = p - 13; /* zone interdite tab[5-13] soit tab[-8] , compile mais pb à l'exécution */

- Lors de la définition d'un tableau, on doit mettre une valeur numérique entre les crochets

Vrai sauf vals3 donc F

```
double t[3];
long vals[4]={1,2,3,4}; /* taille à mettre à jour si une valeur de plus.*/
long vals2[18000];
long vals3[]={1,2,3,4}; /* le compilateur détecte 4 valeurs d'initialisation, donc il reserve un tableau de 4 long.*/
```

- On peut utiliser une **expression entière** comme indice pour manipuler les variables contenues dans un tableau **V**

long i; tab[**i**], tab[**2 * i + 3**]…. OK

double j = 3.88; tab[j] … PAS OK

- Les indices des variables d'un tableau de taille maximum *N* vont de 1 à *N*

  *non en langage C, C++, C#, Java: de **O à N-1***

  oui en Fortran: années 50 (voir CERN, Orsay CNRS LIPN)

- L'ordinateur vérifie systématiquement que les indices sont valides lors des manipulations des variables d'un tableau **HELAS FAUX ==> BUGS....**

```
long vals[]={1,2,3,4};
vals[10] = 12; /* COMPILE MAIS PLANTAGE A L'EXECUTION
vals[-7] = 12; /* COMPILE MAIS PLANTAGE A L'EXECUTION

il existe i entier qui vaut 13 (on ne le sait pas vraiment)
vals[i]

En C, le compilateur ne vérifie pas les débordements de tableau
à gauche et à droite
```

p = p - 13;

p = p + 33;

- Un indice peut être un entier **V**, un réel **F** ou un caractère **F**
- Un tableau ne peut contenir que des variables entières ou caractères **F**

il existe des tableaux de double, flottants, de tableaux (2D) ….

- On peut initialiser les variables d'un tableau lors de sa définition **V**

```
long vals1[4]={1,2,3,4};
long vals2[]={1,2,3,4};
```

Définition et syntaxe :

Parmi les définitions de tableaux suivantes, indiquez celles qui sont incorrectes et pourquoi. Si c'est possible, indiquez la définition qu'il aurait fallu écrire.

- `long tablo={1,2,3,4,5};` **FAUX**

`long tablo[]={1,2,3,4,5};`

- `double t[3];` **VRAI**
- `tableau x[124];` **FAUX**

`le type tableau n'existe pas en C et n'est pas à priori défini par le programmeur`

- `double tab_reel[];` **FAUX le compilateur ne peut connaître la taille mémoire à réserver.**

  `double * tab_reel; /* OK pour un pointeur ….*/`

- ```
  long taille;
  taille = 5;
  char tabc[taille]={'a', 'b', 'c', 'd', 'e'};
  ```
  FAUX dans le cadre du cours langage C
  VRAI en C++
- `long vals[4]={1,2,3,4};` **VRAI**
- `long vals2[18000];` **VRAI**

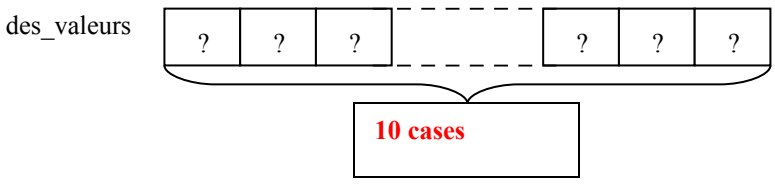
saisie / affichage de tableaux

rappel : lorsque l'on indique que l'on saisit ou que l'on affiche un tableau, il faut comprendre : on saisit des valeurs à ranger dans les variables d'un tableau, on affiche les valeurs des variables contenues dans un tableau. De plus, on utilise quasiment systématiquement des boucles lorsque l'on traite un tableau.

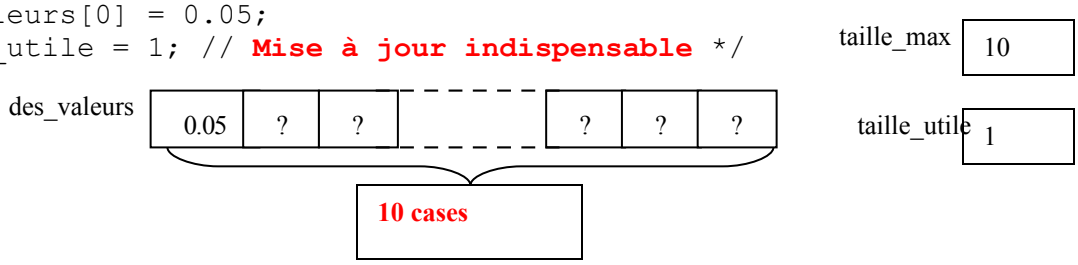
Que fait le programme suivant ? Remplissez les schémas au fur et à mesure, et complétez les instructions sur fond gris à la fin du programme.

```
int main()
{
double des_valeurs[10]; /* tableau non initialisé ==> valeurs aléatoires
long taille_max, taille_utile; /* non initialisées */
long compt; /* non initialisé */
```

légende : ? <==> valeur aléatoire
tableau statique ==> pointeur constant et taille maximale constante

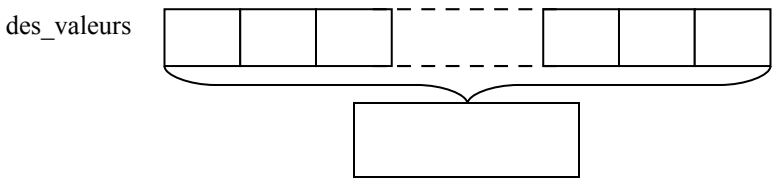

```

```
taille_max = 10;
des_valeurs[0] = 0.05;
taille_utile = 1; // Mise à jour indispensable */
```



IMPOSSIBLE DE MODIFIER LA TAILLE MAXIMALE D'UN TABLEAU STATIQUE
POSSIBLE DE MODIFIER LA TAILLE MAXIMALE D'UN TABLEAU DYNAMIQUE souplesse....
en C: malloc, realloc, free
en C++: new, delete

```
for (compt =1 ; compt <= taille_max-1; compt++)
{
      des_valeurs[compt] = des_valeurs[compt-1] + 0.1;
}
```




```
// affichage de toutes les valeurs du tableau

for(compt =  0; compt < taille_utile; compt++)
{
      printf("la variable d'indice %ld du tableau vaut :%ld\n ",
compt, des_valeurs[compt]);
}
```

```
for (compt =  0; compt < taille_utile; compt++)
{
      cout << "la variable d'indice " << cmpt  << " du tableau vaut : " << des_valeurs[compt] << endl;
}
```

- Ecrire un programme qui fait la saisie de valeurs de type caractère et qui les range dans un tableau contenant au maximum 50 caractères. L'utilisateur peut arrêter la saisie en saisissant le caractère '&'. **Avant de commencer le programme, écrivez bien quelles sont les conditions auxquelles on arrête (ou on continue) la boucle dans laquelle se fait la saisie.**

```
#include <iostream>
#include <math.h>
#define TAILLE_MAXIMALE 5
#define CARACTERE_FIN '&'

using namespace std;

int main()
{
   char c;
   char tab[TAILLE_MAXIMALE];
   long tailleUtile = 0;
   long i;

   for (i = 0; (i < TAILLE_MAXIMALE) && (c != CARACTERE_FIN); i++)
   {
      cout << "caractere : "; cin >> c;
      tab[i] = c;
      tailleUtile++;
   }

   if (c == CARACTERE_FIN)
      tailleUtile--; // le caractère de fin ne doit pas être pris en compte
   for (i = 0; i < tailleUtile; i++)
      cout << tab[i] << ' ';
   cout << endl;

   return 0;
}
```

- Ecrire un programme qui fait la saisie de valeurs de type réel, qui les range dans un tableau et qui effectue les calculs suivants :
    a) s'il y a plus de 3 valeurs : calcul de la somme des valeurs absolues
    b) s'il y a exactement 2 valeurs : calcul de leur moyenne harmonique $\overline{h}$:

$$\frac{2}{\overline{h}} = \frac{1}{x_1} + \frac{1}{x_2}$$

où x_1 et x_2 sont les deux valeurs du tableau.

    c) s'il y a exactement 4 valeurs : calcul de leur moyenne géométrique $\overline{g}$:

$$\overline{g} = \sqrt[4]{x_1.x_2.x_3.x_4}$$

où x1,…x4 sont les quatre valeurs du tableau.

Pour le calcul de la racine quatrième : calculer la racine quatrième revient à élever le nombre à la puissance ¼, soit 0,25. On utilisera une instruction de calcul : *puissance(x,y)* qui calcule le nombre *x* (réel) élevé à la puissance *y* (nombre réel), et qui donne un résultat réel.

```
#include <iostream>
#include <iomanip> // les manipulateurs du C++ cf fixed et setprecision
#include <math.h>

#define TAILLE_MAXIMALE 10

using namespace std;

int main()
{
   long tab[TAILLE_MAXIMALE];
   long tailleUtile = 0;
   long i;
   do
   {
      cout << "Nombre de valeurs dans [2, " << TAILLE_MAXIMALE << "] : ";
      cin >> tailleUtile;
   } while ((tailleUtile < 2) || tailleUtile > TAILLE_MAXIMALE);
```

```
    for (i = 0; i < tailleUtile; i++)
    {
       cout << "entier : "; cin >> tab[i];
    }
    cout << "Tableau : ";
    for (i = 0; i < tailleUtile; i++)
       cout << tab[i] << ' ';
    cout << endl;

    cout << fixed << setprecision(2);
    if (tailleUtile == 4)
    {
        double moyenneGeometrique = 1;
        for (i = 0; i < tailleUtile; i++)
            moyenneGeometrique *= tab[i];
        moyenneGeometrique = pow(moyenneGeometrique, 0.25);
       cout << "La moyenne geometrique est : " << moyenneGeometrique << endl;
    }
    else if (tailleUtile == 2)
    {
       double moyenneHarmonique;
       moyenneHarmonique = (2.0 * tab[0] * tab[1]) / (tab[0] + tab[1]);
       cout << "La moyenne harmonique est : " << moyenneHarmonique << endl;
    }
    else if (tailleUtile >= 3)
    {
         double sommeValeursAbsolues = 0;
         for (i = 0; i < tailleUtile; i++)
                    sommeValeursAbsolues += fabs(tab[i]);
          cout << "La somme des valeurs absolues est : " << sommeValeursAbsolues << endl;
    }
     return 0;

}
```

THEME 2 : tableaux : utilisation et algorithmes classiques

Recherche de la deuxième plus grande valeur d'un tableau

On considère un tableau (peu importe le type des variables qu'il contient, c'est à vous de le choisir) comportant au maximum *N* variables et dont *M* variables sont utilisées. En vous inspirant de l'algorithme de recherche de la valeur maximum, écrivez un programme qui recherche la deuxième plus grande valeur du tableau (celle qui se rapproche le plus du maximum sans l'atteindre).

```
#include <iostream>
#include <limits.h> // il existe la constante LONG_MIN le plus petit entier long possible
#define TAILLE_MAXIMALE 30
#define TAILLE_UTILE 12

using namespace std;

int main()
{
   // valeurs à modifier pour vos tests...
   long tab[TAILLE_MAXIMALE] = {248, 85, 3, 15, 82, 7, 12, 5, 3, 17, 1, 85, 248};

   long i;
   for (i = 0; i < TAILLE_UTILE; i++)
      cout << tab[i] << ' ';
   cout << endl;

   long premierMax, secondMax;
   for (i = 0, premierMax = secondMax =  LONG_MIN; i < TAILLE_UTILE; i++)
   {
      if (premierMax < tab[i])
      {
         secondMax = premierMax;
         premierMax = tab[i];
      }
     else if (premierMax == tab[i])
           ; // instruction vide RIEN A FAIRE DANS CE CAS
      else if (secondMax < tab[i])
         secondMax = tab[i];
   }

   cout << endl << endl;
   cout << "premier maximum : " << premierMax << endl;
   cout << "second maximum : " << secondMax << endl << endl;

   for (i = 0; i < TAILLE_UTILE; i++)
      cout << tab[i] << ' ';
   cout << endl;
   return 0;
}
```

Suppression de doublons

Soit un tableau contenant des variables entières. Ecrivez un programme à qui l'on fournit une valeur entière et qui supprime tous les exemplaires sauf un de cette valeur si elle se trouve en plusieurs exemplaires dans le tableau.

```
#include <iostream>
#include <limits.h>
#define TAILLE_MAXIMALE 30
#define TAILLE_UTILE 12

using namespace std;

int main()
{
   int tab[] = {5, 3, 15, 82, 7, 12, 5, 3, 17, 1, 82, 5};

   long i;
   for (i = 0; i < TAILLE_UTILE; i++)
      cout << tab[i] << ' ';
   cout << endl;

   long x;
   cout << "valeur a supprimer : ";
   cin >> x;

   // SUPPRESSION D'UNE SEULE OCCURENCE
   long drapeau;
   for (i = 0, drapeau = 1; (i < TAILLE_UTILE) && (drapeau != 0); i++)
   {
      if (tab[i] == x)
      {
         tab[i] = LONG_MIN; // on barre la case
         drapeau = 0;
      }
```

```
/*
   for (i = 0; i < TAILLE_UTILE; i++)
   {
      if (tab[i] == x)
      {
         tab[i] = LONG_MIN; // on barre la case
        break;
      }
   }
*/
   for (i = 0; i < TAILLE_UTILE; i++)
      cout << tab[i] << ' ';
   cout << endl << endl;

   for (i = 0; i < TAILLE_UTILE; i++)
   {
      if (tab[i] != LONG_MIN)
         cout << tab[i] << ' ';
   }
   cout << endl;
   return 0;
}
```

// SUPPRESSION DE TOUTES LES occurrences SAUF LA PREMIERE

```
#include <iostream>
#include <limits.h>
#define TAILLE_MAXIMALE 30
#define TAILLE_UTILE 12

using namespace std;
int main()
{
   int tab[] = {5, 3, 15, 82, 7, 12, 5, 3, 17, 1, 82, 5};
   long i;
   for (i = 0; i < TAILLE_UTILE; i++)
      cout << tab[i] << ' ';
   cout << endl;
   long x;
   cout << "valeur a supprimer : ";
```

```
    cin >> x;

    long drapeau;
    for (i = 0, drapeau = 1; i < TAILLE_UTILE; i++)
    {
      if (tab[i] == x)
      {
          if (drapeau != 1) // pas la première occurrence
                  tab[i] = LONG_MIN; // on barre la case
          drapeau = 0;
      }
    }

    for (i = 0; i < TAILLE_UTILE; i++)
      cout << tab[i] << ' ';
    cout << endl << endl;

    for (i = 0; i < TAILLE_UTILE; i++)
    {
      if (tab[i] != LONG_MIN)
          cout << tab[i] << ' ';
    }
    cout << endl;

    return 0;
}
```

A partir du programme précédent, écrire un programme qui supprime tous les doublons (exemplaires multiples d'une valeur) d'un tableau.

```
#include <iostream>
#include <limits.h>
#define TAILLE_MAXIMALE 30
#define TAILLE_UTILE 12

using namespace std;
int main()
{
  int tab[] = {5, 3, 15, 82, 7, 12, 5, 3, 17, 1, 82, 5};
  long i;
```

```
    for (i = 0; i < TAILLE_UTILE; i++)
        cout << tab[i] << ' ';
    cout << endl;
    long x;
    cout << "valeur a supprimer : ";
    cin >> x;

    // SUPPRESSION DE TOUTES LES occurrences
    for (i = 0; i < TAILLE_UTILE; i++)
    {
        if (tab[i] == x)
        {
            tab[i] = LONG_MIN; // on barre la case
        }
    }

    for (i = 0; i < TAILLE_UTILE; i++)
        cout << tab[i] << ' ';
    cout << endl << endl;

    for (i = 0; i < TAILLE_UTILE; i++)
    {
        if (tab[i] != LONG_MIN)
            cout << tab[i] << ' ';
    }
    cout << endl;

    return 0;
}
```

SOLUTION INTERACTIVE

On demande à l'utilisateur s'il veut supprimer une seule ou toutes les occurrences.

On utilise **une seule boucle** qui discrimine les cas: le recours à une variable drapeau est alors nécessaire. Un break simplifie parfois bien les choses....

```
#include <iostream>
#include <limits.h>
#define TAILLE_MAXIMALE 30
#define TAILLE_UTILE 12

using namespace std;

int main()
{
   int tab[] = {5, 3, 15, 82, 7, 12, 5, 3, 17, 1, 82, 5};

   long i;
   for (i = 0; i < TAILLE_UTILE; i++)
      cout << tab[i] << ' ';
   cout << endl;

   long x;
   cout << "valeur a supprimer : ";
   cin >> x;

   // 0 toutes les occurences
   // 1 une seule occurence
   // 2 si 1 seule occurence déjà supprimée car on veut sortir de la boucle
   long drapeau;
   cout << "toutes les occurences de " << x << " 1 oui/0 non : ";
   cin >> drapeau;

   for (i = 0; (i < TAILLE_UTILE) && (drapeau !=2); i++)
   {
      if (tab[i] == x)
      {
         if (drapeau == 1) // toutes les occurences
            tab[i] = LONG_MIN; // on barre la case
         else if (drapeau == 0) // une seule occurence
```

```
            {
                tab[i] = LONG_MIN; // on barre la case
                drapeau = 2; // IL FAUT Y PENSER pour pouvoir sortir de la boucle!!
            }
        }
    }
    /*
        for (i = 0; i < TAILLE_UTILE; i++)
        {
            if (tab[i] == x)
            {
                if (drapeau == 1)  // toutes les occurences
                    tab[i] = LONG_MIN; // on barre la case
                else // le drapeau vaut nécessairement 0 et on veut supprimer 1 seule occurence
                {
                    tab[i] = LONG_MIN; // on barre la case
                    break; // pour sortir de la boucle car il n'est pas nécessaire de poursuivre
                }
            }
        }
    */
    for (i = 0; i < TAILLE_UTILE; i++)
        cout << tab[i] << ' ';
    cout << endl << endl;

    for (i = 0; i < TAILLE_UTILE; i++)
    {
        if (tab[i] != LONG_MIN)
            cout << tab[i] << ' ';
    }
    cout << endl;

    return 0;
}
```

Fusion de tableaux triés

On dispose de deux tableaux triés par ordre croissant, on désire réunir les valeurs de ces deux tableaux dans un troisième tableau qui devra lui aussi être trié. On veut éviter de recopier les valeurs du premier tableau, puis à la suite les valeurs du deuxième tableau puis de trier le tout. Proposez un algorithme qui insère les valeurs dans le troisième tableau directement à leur bonne place, et écrivez le programme correspondant.

```
#include <iostream>
#define TAILLE_MAXIMALE1 5
#define TAILLE_MAXIMALE2 7
using namespace std;


int main()
{
   long tab1[TAILLE_MAXIMALE1];
   long tab2[TAILLE_MAXIMALE2];
   long tab3[TAILLE_MAXIMALE1 + TAILLE_MAXIMALE2];
   long tailleUtile1 = 0;
   long tailleUtile2 = 0;
   long tailleUtile3 = 0;
   long i, j, k;

   cout << "Saisie des " << TAILLE_MAXIMALE1 << " valeurs du premier tableau par ordre croissant : " << endl;
   for (i = 0; i < TAILLE_MAXIMALE1; i++)
   {
      cout << "entier : ";
      cin >> tab1[i];
      tailleUtile1++;
   }
   cout << "Saisie des " << TAILLE_MAXIMALE2 << " valeurs  du second tableau par ordre croissant : " << endl;
   for (i = 0; i < TAILLE_MAXIMALE2; i++)
   {
      cout << "entier : ";
      cin >> tab2[i];
      tailleUtile2++;
   }


    cout << "Tableau1 : ";
   for (i = 0; i < tailleUtile1; i++)
      cout << tab1[i] << ' ';
   cout << endl;


   cout << "Tableau2 : ";
   for (i = 0; i < tailleUtile2; i++)
      cout << tab2[i] << ' ';
```

```
cout << endl;

// fusion des 2 tableaux triés par ordre croissant

// tant qu'il y a des éléments dans les 2 tableaux
for (   i = j = k = 0; (i < tailleUtile1) && (j < tailleUtile2); k++)
{
   if  (tab1[i] < tab2[j])
   {
      tab3[k] = tab1[i];
      i++;
   }
   else  //  tab2[j] <= tab1[i]
   {
      tab3[k] = tab2[j]; // *(tab3 + k) ...
      j++;

   }
}
// tant qu'il y a encore des éléments dans le premier tableau
while (i < tailleUtile1)
{
   tab3[k] = tab1[i];
   i++;
   k++;
}
// tant qu'il y a encore des éléments dans le second tableau
while (j < tailleUtile2)
{
   tab3[k] = tab2[j];
   j++;
   k++;
}
tailleUtile3 = k; // mise à jour de la taille utile du tableau trié
cout << "Tableau3 : ";
for (i = 0; i < tailleUtile3; i++)
   cout << tab3[i] << ' ';
cout << endl;
return 0;
}
```

THEME 3 : tableaux de caractères : travailler avec du texte

Message personnalisé

Ecrire un programme qui saisit un nom d'utilisateur (sous la forme d'un texte) et qui répond par un message de bienvenue comportant le nom de l'utilisateur saisi précédemment. Le nom de l'utilisateur doit se trouver au milieu du message de bienvenue.(message et nom sont 2 chaines de caractères stockant les informations).

```
#include <iostream>
#include <string.h>

#define SIZE 64
#define BUFSIZE 256

using namespace std;

int main()
{
  char nom[SIZE];
  char message[BUFSIZE];

  cout << "nom : "; cin >> nom;

  strcpy(message, "Bienvenue ");
  strcat(message, nom);
  strcat(message, " dans le monde merveilleux des chaines de caracteres");

  cout << "nom : " << nom << endl;
  cout << "message : " << message << endl;

   return 0;
}
```

Noms de fichiers

Lorsque l'on écrit des programmes utilisant des fichiers, on accède à ces fichiers en précisant leur nom sous la forme de texte, donc de tableaux de caractères. En général, l'utilisateur entre un nom de fichier court (ou relatif par rapport au répertoire courant), mais l'ordinateur a besoin de connaître le nom long ou absolu du fichier.

Ecrire un programme qui saisit un nom de disque dur (sous la forme d'une lettre pour désigner le disque), le nom d'un répertoire où sont stockés des fichiers de l'utilisateur (ce peut être un nom comportant plusieurs répertoires si on a besoin de parcourir plusieurs niveaux dans l'arborescence), et le nom d'un fichier (celui que veut manipuler l'utilisateur). Le programme doit afficher le nom absolu (nom long) du fichier.

```
#include <iostream>
#include <string.h>

#define BUFSIZE 256

using namespace std;

        int main()
        {
           char disk[4];
           char dirname[BUFSIZE/2];
           char filename[BUFSIZE/2];
           char pathname[BUFSIZE + 4];

           cout << "disque : ";
           cin >> disk;
           cout << "repertoire terminé par /: ";
           cin >> dirname;
           cout << "nom de fichier : ";
           cin >> filename;

           strcpy(pathname, disk);
           strcat(pathname, dirname);
           strcat(pathname, filename);

           cout << "chemin absolu: " << pathname << endl;

           return 0;
        }
```

THEME 4 : tableaux à plusieurs dimensions

délicat en passage de paramètres des fonctions

Calcul matriciel

Une matrice en deux dimensions est définie en mathématiques comme un ensemble de coefficients repérés par leur numéro de ligne et numéro de colonne. Une matrice M à n lignes et p colonnes est un ensemble de coefficients m_{ij}, avec i compris entre 1 et n et j compris entre 1 et p.

$$M= \underbrace{\begin{pmatrix} m_{11} & m_{12} & & m_{1p} \\ m_{21} & \dots & & \\ & & & \\ & & & \\ m_{n1} & & & m_{np} \end{pmatrix}}_{p \text{ colonnes}} \Bigg\} n \text{ lignes}$$

On peut naturellement les représenter par des tableaux à 2 dimensions.

Si M est une matrice ayant le même nombre de lignes que de colonnes (c'est à dire que $n=p$), on dit que la matrice est carrée et on peut calculer sa trace $Tr(M) = \sum_{i=1}^{i=n} m_{ii}$

a) Ecrire un programme qui initialise une matrice carrée avec des **valeurs aléatoires** et qui calcule sa trace.

On peut multiplier entre elles deux matrices A et B pour obtenir le produit $A.B$ (attention avec les matrices la multiplication n'est plus une opération commutative) à la seule condition que le nombre de colonnes de la matrice A soit égal au nombre de lignes de la matrice B.

Si A est une matrice à n lignes et p colonnes et B une matrice à p lignes et q colonnes, alors le produit $A.B$ est une troisième matrice (nommons la C) à n lignes et q colonnes telle

$$\forall\, i \in [1..n], \forall\, j \in [1..q], c_{ij} = \sum_{k=1}^{k=p} a_{ik}.b_{kj}$$

que :

Où a_{ik}, b_{kj} et c_{ij} sont des coefficients des matrices A, B et C.

b) Ecrivez un programme qui initialise deux matrices avec des entiers au hasard et qui réalise le produit de ces deux matrices. Vous pouvez vous aider d'un schéma pour matérialiser comment se fait le produit avant de tenter d'écrire le programme.

```
#include <iostream>
#include <stdlib.h>     // srand, rand
#include <time.h>       // time

using namespace std;

int main()
{
   double matrice[10][10]; // matrice carrée
double identite[10][10]; // matrice carrée


   srand(time(NULL)); // initialisation de l'aléatoire
   // initialisation de la matrice carrée
   for (int i = 0; i < 10; i++)
   {
      for (int j = 0; j < 10; j++)
      {
         matrice[i][j] = rand() % 100; // coefficients < 100
      }
   }

  // initialisation de la matrice carrée identité
   for (int i = 0; i < 10; i++)
   {
      for (int j = 0; j < 10; j++)
      {
          if (i == j)
         {
             matrice[i][j] = 1;
        }
        else
          matrice[i][j] = 0;
      }
   }


   // affichage de la matrice carrée
   cout << "MATRICE CARRE" << endl;
   for (int i = 0; i < 10; i++)
   {
      for (int j = 0; j < 10; j++)
      {
         cout << matrice[i][j] << '\t'; // affichage de la ligne i
      }
      cout << endl; // passage à la ligne i+1
   }

   // calcul de la trace d'une matrice carrée
   long trace = 0;
```

```
for (int i = 0; i < 10; i++)
{
   for (int j = 0; j < 10; j++)
   {
      if (i == j)
      {
          trace += matrice[i][j];
      }
   }
}

cout << "trace = " << trace << endl << endl;

double A[10][5];
double B[5][8];
double P[10][8];

// initialisation de la matrice A
for (int i = 0; i < 10; i++)
{
   for (int j = 0; j < 5; j++)
   {
      A[i][j] = rand() % 100; // coefficients < 100
   }
}

// initialisation de la matrice B
for (int i = 0; i < 5; i++)
{
   for (int j = 0; j < 8; j++)
   {
      B[i][j] = rand() % 100; // coefficients < 100
   }
}

// initialisation de la matrice produit
for (int i = 0; i < 10; i++)
{
   for (int j = 0; j < 8; j++)
   {
      P[i][j] = 0;
   }
}

// calcul du produit matriciel
for (int i = 0; i < 10; i++)
{
   for (int k = 0; k < 8; k++)
   {
      for (int j = 0; j < 5; j++)
      {
         P[i][k] += A[i][j] * B[j][k];
      }
   }
}

// affichage de la matrice produit
cout << "MATRICE PRODUIT" << endl;
```

```
    for (int i = 0; i < 10; i++)
    {
       for (int j = 0; j < 8; j++)
       {
          cout << P[i][j] << '\t'; // affichage de la ligne i
       }
       cout << endl << endl; // passage à la ligne i+1
    }

    return 0;
}
```

MEMO

tab[i] <==> *(tab+i)

&tab[i] <==> &(*(tab+i)) <==> tab + i

rq: **& et * opérateurs « réciproques »**