TP 9 – les tableaux

Les tableaux sont très utilisés en C++. Dans de très nombreux cas, vous aurez besoin de les utiliser et donc de les maitriser. Pour commencer, nous étudierons comment les initialiser, puis comment les utiliser, pour enfin travailler avec des tableaux de dimension supérieur à 1.

L'initialisation

Avant de pouvoir utiliser un tableau, il est nécessaire de **donner une valeur** à chacune des cases. En effet, par défaut les cases du tableau ont des valeurs aléatoires, et donc avant de pouvoir les utiliser, il faut leur donner une valeur. C'est le même principe que pour les variables : il faut donner une valeur dès le début. Alors que pour une variable « simple » on a juste à faire `variable=0`, on ne peut plus faire la même chose avec un tableau. En effet, on a besoin d'initialiser de nombreuses variables (**toutes les cases du tableau**), et il n'est pas possible de le faire d'un coup.

Comme on va le voir, travailler avec un tableau est très souvent **synonyme de boucle**... C'est le cas parce qu'il faut faire un même traitement plusieurs fois : pour chaque case, on fait une opération. Dans le principe, initialiser un tableau peut se faire comme ça :

```
Pour toutes les cases du tableau, faire :
     Initialiser la i ème case avec la valeur par défaut
```

Deux remarques très importantes :

- Il faut que la boucle s'occupe de tout le tableau : l'initialisation du compteur doit partir de la première case (**0**) et s'arrêter à la dernière case (**taille du tableau**).
- A l'intérieur de la boucle, il faut travailler sur une case du tableau : celle du **compteur**. En effet, la case du compteur (`tab[cpt]`) changeant à chaque tour de boucle, à la finale, l'opération est effectuée sur tout le tableau.

Voila donc la méthode pour initialiser un tableau de 20 cases et de type entier :

```
int cpt=0 ;
while (cpt<20){
        tab[cpt]=0 ;
        cpt++ ;
}
```

```
for(int cpt=0;cpt<20;cpt++){
        tab[cpt]=0 ;
}
```

Les deux méthodes sont identiques et à connaitre car c'est la base des traitements sur les tableaux !

Important : l'intérieur de la boucle doit être fait en pensant au traitement qu'il faut faire **sur une case** et le compteur se chargera de reproduire ce traitement sur tout le tableau.

Initialisation Bis.

Il existe une autre manière d'initialiser un tableau : demander à l'utilisateur de **saisir les valeurs**, plutôt que de mettre un zéro (ou une autre valeur) par défaut. C'es notamment le cas lorsqu'il faut saisir des numéros, des lettres, bref, quand il faut donner d'autres valeurs qu'un zéro. La ligne `tab[cpt]=0;` est donc à remplacer par un `cin>>tab[cpt];`

Utiliser les valeurs d'un tableau

Maintenant que l'on a des valeurs dans le tableau, il faut les utiliser. Mais il n'y a pas qu'une façon de traiter un tableau. A chaque application son traitement. On peut néanmoins classer en deux grandes catégories les traitements.

Un même traitement pour chaque case.

Utilisation : Mettre en majuscule chaque case d'un tableau de caractère, mettre tous les nombres d'un tableau en valeur positive (valeur absolue), mettre au carré chaque nombre d'un tableau…

Ce traitement n'est pas plus compliqué que l'initialisation. L'astuce est de se souvenir que le traitement se fait sur une case, et qu'il suffisse de faire le traitement sur la case du compteur (`tab[cpt]`) pour que le traitement soit fasse sur tout le tableau.

```
Pour toutes les cases du tableau, faire :
      Faire le calcul sur la case cpt
```

Prenons l'exemple de la mise en majuscule du tableau de caractères. On sait qu'il faut soustraire au caractère 32 pour que la lettre minuscule devienne majuscule (grâce au code ASCII). Le traitement sera donc de mettre à la place du caractère un autre caractère (code ASCII du caractère moins 32). Il faut donc faire : `tab[cpt]=tab[cpt]-32;` Puisque cette instruction se trouve à **l'intérieur d'une boucle**, `cpt` va de 0 jusqu'à la dernière valeur, et donc l'opération se fera sur toutes les cases. Donc, dans une version simplifiée, le calcul donne :

```
int cpt=0 ;
while (cpt<20){
        tab[cpt]=tab[cpt]-32 ;
        cpt++ ;
}
```

```
for(int cpt=0;cpt<20;cpt++){
        tab[cpt]=tab[cpt]-32 ;
}
```

Le principe est donc bien simple : il suffit de savoir faire le calcul sur une case pour que l'on sache le faire sur tout le tableau. Il faut juste entourer ce calcul par une boucle qui démarre au bon endroit et qui s'arrête au bon moment…

Un résultat pour tout un tableau.

Utilisation : Calculer la moyenne des valeurs d'un tableau, calculer le produit des valeurs du tableau… Ce traitement doit être un peu plus malin que le précédent. En effet, puisqu'il faut un résultat, il faut introduire une **nouvelle variable** ; nous l'appellerons `resultat`. C'est dans cette variable que nous calculerons les valeurs intermédiaires. Dans le cas d'une somme, on ajoute au résultat la valeur de la case, on a à faire le calcul suivant sur toutes les cases : `resultat=resultat+tab[cpt].`

Exercices

Pour savoir comment faire une boucle, décomposez le travail :

- ***Quel est l'intervalle du tableau ?*** Trouvez la case de **départ** (très souvent 0…) et initialisez le compteur. Ensuite trouvez la **dernière case** (souvent la taille du tableau, mais peut dépendre aussi d'une autre condition : si on saisi un 0…) et faites la (ou les) condition(s) d'arrêt qui correspond à l'énoncé.
- ***Que doit-on faire sur chaque case ?*** Imaginez que vous travaillez avec **une seule case** et faites l'opération qui est demandée. Pensez à utiliser la case du compteur pour que cette opération soit répétée sur chaque case.

Exo 1 :

Faire un programme qui demande à l'utilisateur 10 nombres (pensez « initialisation du tableau par l'utilisateur »…) et qui les affiches sur une même ligne, tous séparés par -->. Commencez par afficher les nombres à la suite, puis trouvez le moyen d'afficher la flèche entre chaque nombre.

Exemple : l'utilisateur saisi 10 15 13 26 58 87 1 23 65 80, l'ordinateur affiche :

10 --> 15 --> 13 --> 26 --> 58 --> 87 --> 1 --> 23 --> 65 --> 80

Exo 2 :

Faire un programme qui demande à l'utilisateur de saisir des nombres positifs. La saisie s'arrête s'il saisi un nombre négatif, et dans ce cas on affiche tout les nombres positifs qu'il a saisi. Pour faire ce programme, il faut penser à deux choses :

- Dans tous les cas, le nombre de valeurs saisies ne peuvent pas être supérieurs au nombre de valeurs du tableau… La condition de la boucle doit donc en tenir compte (en plus du fait que la saisie s'arrête quand le nombre est négatif). Vous prendrez un nombre suffisamment grand pour éviter que ça arrive (100).
- L'affichage ne doit pas afficher les nombres négatifs !

Exo 3 :

Faire un programme qui demande à l'utilisateur de saisir des notes (des réels). La saisie s'arrête s'il saisi une note négative, et dans ce cas, on calcul la moyenne des notes qu'il a saisi. On se rend bien compte que l'exercice est très proche de l'exercice précédent, mais on ne souhaite plus afficher le contenu du tableau mais bien effectuer un calcul. Ce calcul se déroulera en deux étapes : tout d'abord, il faut calculer la somme de toutes les notes saisies, et ensuite diviser ce résultat par le nombre de chiffres positifs saisis. Réfléchissez bien à l'endroit où mettre ses calculs (avant, dans ou après la boucle). Pour répondre à cette question, une piste est de se demander si on veut faire le calcul pour chaque case ou uniquement une fois…

Les tableaux multidimensionnels

Contrairement à ce que l'on peut penser, il existe plus de 3 dimensions ! Le cerveau humain ne peut en représenter que 4, mais il peut très bien manipuler d'autres dimensions… Même si on saurait pas dessiner tableau de dimension 5, sachez qu'ils existent : `tab[13][2][5][5][2];` est un tableau à 5 dimension ! Une utilisation de ce tableau pourrait être de stocker vos notes du BTS : la première dimension correspond aux élèves (13 cases), la seconde l'année (2 cases), la troisième vos matières (5), la quatrième vos notes (5) et la cinquième la répartition de chaque note entre les TP et les devoirs. Cet exemple est purement fictif, mais il prouve bien qu'il existe des utilisations de tableau de dimension supérieur à 1.

Il ne faut pas être effrayé par **l'apparente complexité**, travailler avec ces tableaux n'est pas plus compliqué que de travailler avec les tableaux à une dimension, ils requièrent juste un peu plus de rigueur. Il y a plusieurs cas à traiter. Commençons par le cas où le traitement doit se faire de la même manière sur toutes les cases.

Un même traitement pour chaque case.

Utilisation : Mettre en majuscule chaque case d'un tableau de caractère, mettre tous les nombres d'un tableau en valeur positive (valeur absolue), faire du flou sur une image…

Le **principe est exactement le même** que pour un tableau à une dimension, ce qui change c'est la boucle. En effet, il faut maintenant utiliser autant de **boucles imbriquées** qu'il y a de dimensions puisqu'une boucle ne travaille que sur une dimension (une ligne). Dans le cas d'un tableau à la Excel (dimension 2), il faut deux boucles : l'une travaillant sur une ligne, l'autre répétant ce travail pour toutes les lignes. Ainsi, si on travaille sur un tableau de 20 lignes et 10 colonnes :

```
int cpt=0 ;
while (cpt<20){
   int cpt1=0;
   while (cpt1<10){
     //traitement pour la case
    // tab[cpt][cpt1]
     cpt1++ ;
   }
   cpt++ ;
}
```



```
for (int cpt=0;cpt<20;cpt++){
  for (int cpt1=0;cpt1<10;cpt1++){
     //traitement pour la case
    // tab[cpt][cpt1]
  }
}
```

Donc on se rend compte qu'il faut deux compteurs : l'un pour les cases d'une ligne, l'autre pour les lignes du tableau. Ensuite, l'intérieur de la boucle se fait de la même manière qu'avant : on réfléchit au traitement qu'il faut faire sur une case (`tab[cpt][cpt1]`) et les boucles se chargent de refaire le traitement sur tout le tableau.

Important : il ne faut pas oublier de **remettre à 0** les compteurs avant de rentrer dans une boucle. Si on oublie, le programme ne fera le traitement que sur la première ligne !

Exo4 :

Faire un programme qui demande à l'utilisateur ses notes de toutes ses matières. On suppose que l'utilisateur a 3 matières avec 2 notes à chaque fois. Le tableau devra donc avoir 3 lignes et 2 colonnes… Ensuite, affichez le contenu du tableau. La qualité de l'affichage n'est pas importante, mais faites attention que toutes les valeurs soient affichées !

Un traitement pour chaque ligne.

Bien souvent, il faut réaliser un **traitement particulier** pour une ligne puis faire un calcul intermédiaire avant de passer à la ligne suivante. C'est le cas par exemple quand on veut afficher les moyennes pour chaque ligne. Si on reprend le programme précédent, on peut vouloir calculer les moyennes de chaque matière. Pour cela, on devra calculer, pour chaque ligne, la somme des valeurs puis la diviser par deux (il n'y a que deux notes pour chaque matière). Puis, une fois ce calcul fait, afficher la moyenne avant de refaire ce traitement sur la ligne suivante. Voila une manière de procéder :

```
int cpt=0 ;
while (cpt<3){
   int cpt1=0;
   int moyenne=0;
   while (cpt1<2){
      moyenne=moyenne+tab[cpt][cpt1] ;
      cpt1++ ;
   }
   cout<<"moyenne :"<<moyenne/2<<endl ;
   cpt++ ;
}
```

```
for (int cpt=0;cpt<3;cpt++){
   int moyenne=0;
   for (int cpt1=0;cpt1<2;cpt1++){
      moyenne=moyenne+tab[cpt][cpt1] ;
   }
   cout<<"moyenne :"<<moyenne/2<<endl ;
}
```

Il faut bien comprendre qu'ici, pour chaque ligne, il faut faire **toujours la même chose** : calculer la somme des éléments puis afficher la moyenne à l'écran. Donc on fait ce traitement, et on met autour une autre boucle qui répétera ce traitement pour toutes les lignes. Notre traitement de tableau multidimensionnel est fini !

Exo 5 :

Ecrire un programme en C++ calculant la somme des éléments d'un tableau à deux dimensions (on demandera donc à l'utilisateur deux séries de nombres et on affichera les résultats des calculs). Il faut additionner la première ligne avec la seconde ligne, et non la somme de tous les nombres du tableau. Il faut donc afficher 10 résultats si le tableau fait 10 cases de long…

Exo 6 :

Ecrire un programme en C++ qui demande 5 mots à l'utilisateur de 10 lettres au maximum (ce sera la taille de chaque ligne du tableau). On aura donc un tableau : `char tab[5][10]`. Le but est d'afficher ces 5 mots, les un en dessous des autres.