# TD 5 : Tableaux dynamiques

Ne pas oublier de créer un répertoire pour placer les codes de ce TP !

**Toutes les fonctions écrites doivent être testées dans le programme principal !**

**Exercice 1** (*lecture de code : double pointeur*)
Qu'affiche le code suivant ?

```
#include<stdio.h>


int main(){
int a,b,c;
int *pa,*pb;
int **ppa;
pa=&a;
pb=&b;
ppa=&pa;
*pa=10;
*pb=-2;
c=b*2;
**ppa=*pb+5;
*ppa=&c;
*pa=*pb+a;
printf("%d, %d, %d \n",a,b,c);
return 0;
}
```

**Exercice 2** (*Tableaux dynamiques 1D*)

- Écrire une fonction `randTab` qui prend trois arguments entiers $n$, $x$, $y$ et qui renvoie l'adresse d'un tableau ayant $n$ cases, et contenant des entiers aléatoires entre $x$ et $y$.
- Tester la fonction dans le programme principal. Les valeur $x$, $n$ et $y$ seront saisies par l'utilisateur. Afficher le tableau obtenu.
- Écrire une fonction `tabPair` qui prend un tableau en paramètre, et qui renvoie l'adresse d'un tableau contenant uniquement les valeurs paires du tableau passé en paramètre.
- Écrire une fonction `noDouble` qui prend un tableau en paramètre, et qui renvoie un tableau ayant supprimé toutes les valeurs identiques contiguës.
  Exemple : [1,1, 1, 8, 2, 7, 7, 8, 9, 2] devient [1, 8, 2, 7, 8, 9, 2].

**Exercice 3** (*Tableau de Prénom*) Dans cet exercice, nous allons stocker une liste de prénom dans un tableau. Puisqu'une chaîne de caractère est un tableau, il s'agira donc d'un tableau de tableau.

1. Déclarer un double pointeur de type char `tabPrenom` pointant sur NULL. Ce pointeur servira comme pointeur sur un tableau contenant les adresses des chaîne de caractères à stocker.
2. Demander à l'utilisateur combient de prénoms il souhaite stocker. Allouer l'espace necessaire sur lequel devra pointer `tabPrenom`.
3. Remplir ce tableau de chaîne de caractères. Pour chaque prénom à saisir :
   - Demander d'abord à l'utilisateur le nombre de lettres du prénom.
   - Allouer un tableau de caractère en fonction du nombre de lettres saisi précédement (attention, il faut penser au caractère de fin ajouté automatiquement en fin des chaînes de caractères, il faut donc ajouter une case supplémentaire par rapport au nombre de lettres).
   - Saisir le prénom à stocker.

4. Écrire une procédure `affichePrenom` prenant un tableau de caractère en paramètre et qui affiche la liste des prénoms stockés.
5. Modifier le tableau de prénom pour que les premières lettres de chaque prénom soit des majuscules si ce n'est pas le cas (regarder la table ASCII).

**Exercice 4** (*Calculatrice*)

On rappelle que pour la fonction main peut prendre des arguments et que sa version complète est `int main(int argc, char **argv)` avec

— argc le nombre de paramètres passés à l'execution +1
— argv un tableau contenant les chaînes de caractères correspondant aux différents paramètre (le premier paramètre est `*argv[1]`).

1. Écrire un programme qui affiche le résultat de l'addition de deux nombres passé en paramètre lors de l'execution.
2. Modifier ce programme pour qu'il affiche le résultat d'une opération (`+ - *` ou `/`) entre deux nombres entiers passés en paramètres. (la fonction`atoi` appartenant à la bibliothèque `stdlib` permet de transformer une chaîne de caractère en entier).
3. Modifier le programme précédent pour gerer des opération sur les nomnbres réel.(la fonction `atof` appartenant à la bibliothèque `stdlib` permet de transformer une chaîne de caractère en réel).

**Exercice 5** (*Triangle de Pascal*)

Les coefficients binomiaux $\binom{n}{k}$ sont des coefficients entiers très utiles en mathématiques, notament en statistique.

Le triangle de Pascal est une méthode graphique qui permet de trouver facilement le résultat de coefficients binomiaux sans avoir besoin de calculer des factorielles. Il s'agit d'un tableau dont chaque "case" représente le résultat d'un coefficient binomial avec $n$ representant les lignes, et $k$ les colonnes. Puisque $k \leq n$, ce tableau est en réalité un triangle (voir le schéma).
Le calcul des coefficients se base sur la relation récursive suivante :
$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$
et se construit comme indiqué sur l'exemple suivant :

| $n$ \ $k$ | 0 | 1 | 2 | 3 | 4 | 5 | ⋯ |
|---|---|---|---|---|---|---|---|
| 0 | 1 | | | | | | |
| 1 | 1 | 1 | | | | | |
| 2 | 1 | 2 | 1 | | | | |
| 3 | 1 | 3 | 3 | 1 | | | |
| 4 | 1 | 4 | 6 | 4 | 1 | | |
| 5 | 1 | 5 | 10 | 10 | 5 | 1 | |
| 6 | 1 | 6 | 15 | 20 | 15 | 6 | 1 |

$\binom{n-1}{k-1} + \binom{n-1}{k} = \binom{n}{k}$

Une "case" d'un triangle de Pascal se calcul donc en additionnant la case du dessus et la case au dessus à gauche (sachant que $\binom{0}{0} = 1$).
Dans cet exercice nous representerons le triangle de Pascal par **un ensemble de tableaux donc chacun represente une ligne du triangle**. Nous allons donc utiliser un tableau de tableaux.

1. Coder une fonction/procédure ***choixNMax(...)***(à vous de choisir les arguments) qui va demander à l'utilisateur de saisir un nombre $n\_max$ représentant l'argument $n$ maximum du coefficient binomial qu'il souhaite calculer (sachant que $n\_max$ doit être un entier strictement positif).
2. Combien de lignes doit faire le triangle de Pascal souhaité ?

3. Coder une fonction/procédure **créerTriangle(...)** (à vous de choisir les arguments) qui va retourner une variable contenant les adresses de chacun des tableaux répresentant les lignes. Cette fonction va construire le triangle de Pascal contenant le bon nombre de lignes souhaitées et les bonnes valeurs dans chaque case. Toutes les cases du triangle doivent pouvoir être accessibles via la variable retournée.
4. Écrire une procédure ***void affichePascal(int **tab, int n_max)*** permettant d'afficher le triangle de Pascal representé par tab avec le coefficient $n$ allant jusqu'à $n\_max$. L'affichage doit être conforme avec l'exemple qui suit ($n\_max = 7$) :

```
1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
1  5  10  10  5  1
1  6  15  20  15  6  1
1  7  21  35  35  21  7  1
```

5. Dans le ***programme principal***, en utilisant, entre autres, les fonctions codées précédemment, demander à l'utilisateur de saisir un $n_{max}$ puis des coefficients $n$ et $k$ (en respectant toutes les règles que doivent respecter ces coefficients), créer et afficher le triangle de pascal correspondant, et afficher la valeur du résultat de $\binom{n}{k}$ grace au triangle créé.