TP6 – Fonctions et portée des variables

L'objectif de la séance est de maîtriser :

- *La déclaration des fonctions (prototype).*
- *La définition des fonctions (implémentation).*
- *L'écriture des commentaires de fonction.*
- *L'appel des fonctions et la gestion de leur valeur de retour.*
- *La portée des variables.*

Bonnes pratiques (rappel)

- **Arborescence**

Créer un répertoire par TP et un projet par exercice avec ***Qt creator***.

- **Commentaires de fichier**

Vous devez écrire les commentaires en entête du fichier : où ? quoi ? qui ? quand ? et ajouter les commentaires si besoin au fil de votre code.

```
// chemin et nom du fichier source
// un énoncé de l'exercice
// NOM Prénom groupe de TP
// date format anglosaxon
```

- **Commentaires de fonction**

Vous devez écrire les commentaires en entête des lignes : une ligne pour dire ce que fait la fonction, une ligne par paramètre et une ligne pour la valeur de retour le cas échéant.

```
// compute square root
// x : value whose square root is computed /!\ positive or zero
// returns square root of x
double sqrt (double x)
```

- **Cartouche d'affichage**

Vous afficherez systématiquement à l'exécution du programme vos NOM Prénom et groupe de TP.

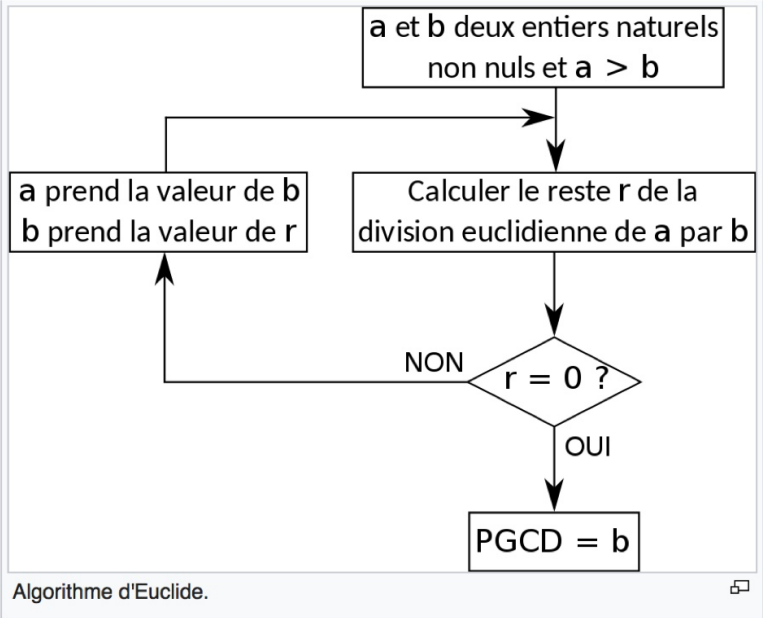
- **IDE**

Pour la création des projets et l'utilisation du debugger avec ***Qt creator*** voir le TP1. N'oubliez pas de sélectionner `qmake` comme choix du `Build System`.

1. P G C D

L'algorithme d'Euclide est un algorithme permettant de déterminer le plus grand commun diviseur (P.G.C.D.) de deux entiers dont on ne connaît pas la factorisation. L'algorithme est donné par le diagramme ci-dessous.

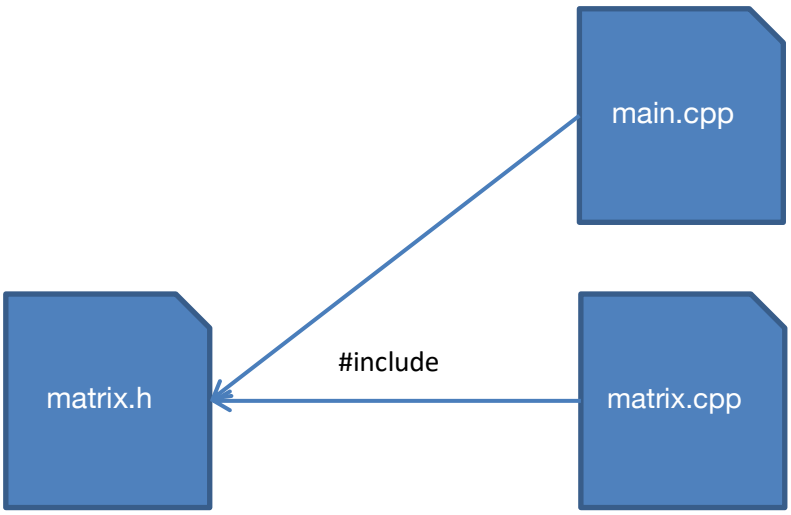
- Donnez la définition de la fonction `PGCD()` qui renvoie le PGCD de deux nombres.
- Proposez un programme de test complet.
- D'après vos tests les conditions suivantes « a et b deux entiers naturels non nuls et a > b » sont-elles nécessaires ?



Algorithme d'Euclide.

2. Matrice carrée

Pour l'exercice suivant vous créerez un projet mettant en œuvre la compilation séparée selon le schéma ci-dessous.



À partir d'une matrice carrée 3x3, proposez le code des fonctions permettant de :
- initialiser la matrice,
- afficher la matrice,
- retourner la somme des éléments de la partie triangulaire supérieure,
- donner la transposée de la matrice,
- dire si la matrice est symétrique,
- afficher les points-cols de la matrice.

RAPPEL

On appelle points-cols les éléments d'une matrice qui sont à la fois un maximum sur leur ligne et un minimum sur leur colonne.

Soit la matrice $A=(a_{i,j})_{1\le i\le n,1\le j\le p}\in M_{n,p}(E)$, on appelle transposée de A la matrice ${}^tA=(a_{i,j})_{1\le j\le p,1\le i\le n}$. Remarquons que ${}^tA\in M_{p,n}(E)$.

A est symétrique si ${}^tA=A$ ce qui exige que A soit une matrice carrée.

Soit la matrice $A=\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$, ${}^tA=\begin{bmatrix}1&4&7\\2&5&8\\3&6&9\end{bmatrix}$

A n'est pas symétrique, A[0][2] est un point-col, la somme de la partie triangulaire supérieure = 26.

3. Accès séquentiel et direct par usage d'indexation

On souhaite écrire un programme de recherche de noms d'étudiants selon une lettre donnée par usage d'un tableau créé dans le programme principal avec une capacité maximale de 200 étudiants.

- Donner la définition C++ d'une fonction qui demande à l'utilisateur du programme le nombre d'étudiants (renvoyé en référence) et les noms de ces étudiants par ***ordre alphabétique croissant*** et **en majuscules** (le tableau rempli est renvoyé par référence).

- Donner la définition d'une fonction qui renvoie le nombre de noms d'étudiant (on parle d'***occurrence***) dans le tableau des étudiants commençant par une lettre donnée.

- Donner la définition de la fonction

```
bool searchStudent(std::string students[],
   unsigned int numberOfStudent, std::string & name,
                                  unsigned int & position )
```

qui renvoie par référence la position d'un nom d'étudiant dans le tableau des étudiants s'il existe (0 sinon) [***recherche non optimisée***].

- Donner la définition de la fonction

```
unsigned int convertEntryArray(char letter)
```

qui associe à une lettre de l'alphabet (majuscule) un index sur le tableau indexé de recherche `indexedArray` (par exemple : pour la lettre `'A'` la fonction donnera l'index `0`).

- Donner la définition de la fonction

```
void initIndexedArray(std::string students[],
    unsigned int numberOfStudent, int indexedArray[])
```

qui initialise le tableau `indexedArray` sur la base de la première occurrence de la première lettre d'un nom d'étudiant. Ce tableau correspondant aux 26 lettres de l'alphabet et est initialisé arbitrairement à la valeur `-1` au départ.

- Donner la définition de la fonction

```
bool searchStudentByIndex(std::string students[],
          unsigned int numberOfStudent, int indexedArray[],
          std::string & name, unsigned int & position)
```

qui renvoie par référence la position d'un nom d'étudiant dans le tableau des étudiants s'il existe (0 sinon) par usage du tableau indexé **[*recherche optimisée*]**.

BONUS : Modifier votre programme en prenant en compte un tableau 2D d'index basés respectivement sur la première et la dernière occurrence de la première lettre du nom d'un étudiant pour vos recherches optimisées (pour chaque lettre de l'alphabet majuscule on a un index de début et un index de fin pour le tableau des noms d'étudiants).

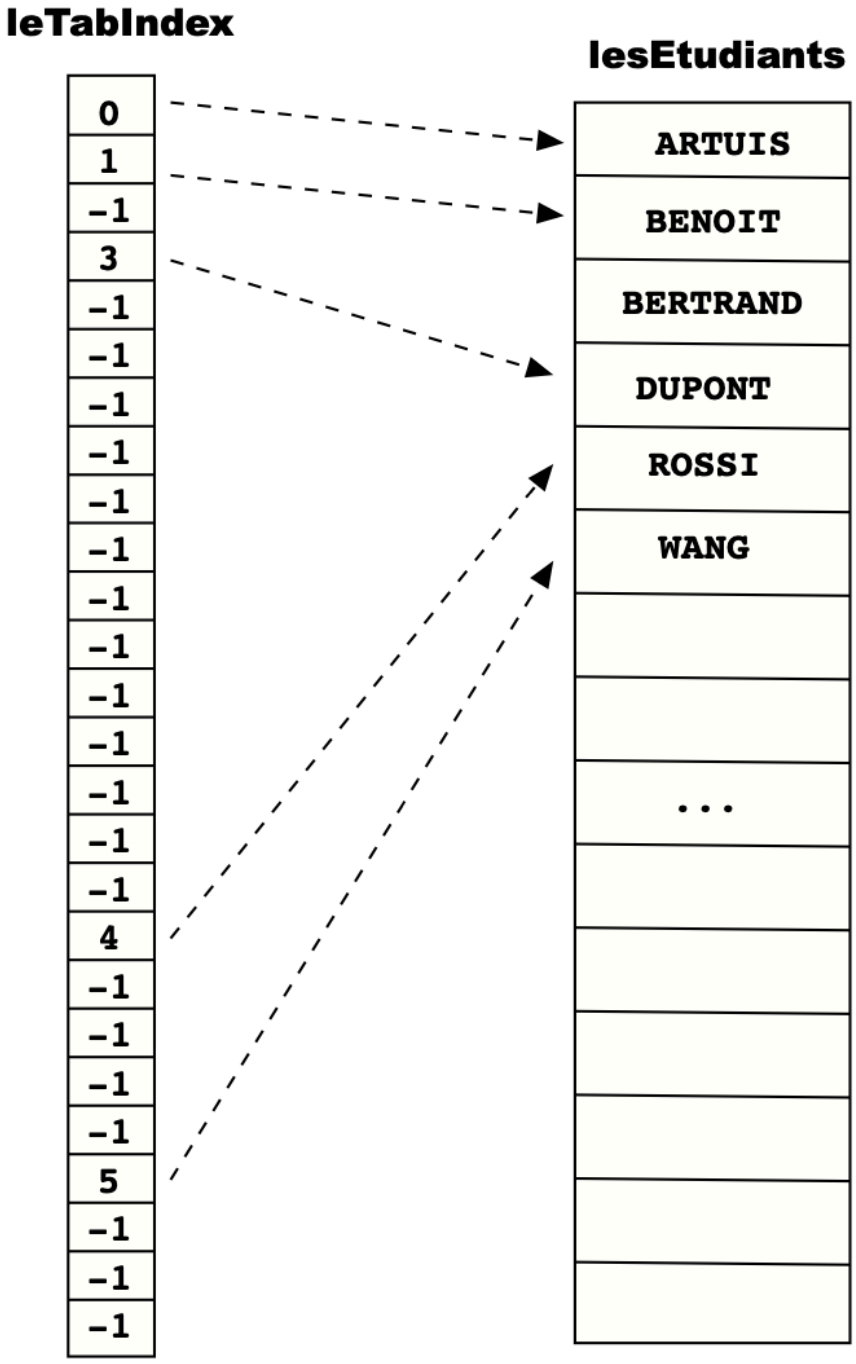


Figure 1 : un groupe d'étudiants de première année.