POLYTECH LILLE

Université Lille1 Sciences et Technologies

**Département IMA / 3A (S5)** **Programmation Structurée 2011/2012**

http://laure.gonnord.org/pro/teaching/

**TP 11 Transitions**

# Objectifs

L'objectif de ce TP est de faire le lien avec les cours de S6 "Programmation Avancée" (section 1), et "Analyse Numérique" (section 2). Passer 1H sur chaque section !

# 1 Deux mini-exercices de pointeurs

(On pourrait faire les mêmes exercices avec des entiers à la place des caractères)

**Exo 1 Chaînes de caractères statiques** Dans `tp11.c` :

- Déclarer une chaîne de caractère dans un tableau de taille statique N, N étant une constante fixée à 100.
- Initialiser cette chaîne avec "turlututu", puis l'imprimer.
- Après avoir inclus la librairie `string.h`, utiliser la fonction `strlen` pour calculer, puis imprimer la taille de la chaîne.
- Dans un terminal, observer la documentation de la librairie `string.h` (`man 3 string`) et y retrouver la signature de la fonction `strlen`. Qu'en déduisez-vous ?
- Déclarer une autre chaîne, l'initialiser, puis imprimer directement le résultat de la concaténation des deux chaînes (`strcat`).
- Déclarer une troisième chaîne statique, et essayer de récupérer le résultat de la concaténation, observer le message d'erreur de compilation. Supprimer cette déclaration.
- Récupérer le résultat de la concaténation à l'aide d'une variable de type `char*`, puis imprimer.

►Conclusion, différence entre `char*` et `char[N]` ? une question de taille !

**Exo 2 : char*, fgets et segmentation fault** Toujours dans le même .c :

- Déclarer une chaîne statique `ch6` de taille N, et l'initialiser avec `fgets` (`man 3 fgets`) :
  ```
  char* varquisertarien = fgets(ch6,5,stdin);
  ```
  (demande sur le terminal/l'entrée standard, une chaîne, et stocke dans ch6 les 4 premiers caractères). Imprimer `varquisertarien` et `ch6` pour vérifier.
- Que se passe-t-il si vous déclarez `ch6` comme étant un `char*`, à la compilation ? à l'exécution ? Pourquoi ?

►Il manque une façon de "donner une taille" à un `char*`. (cf PA, deuxième semestre)

## 2 Matrices statiques, pivot de Gauss

On vous fournit un fichier `tp11_mat.c` contenant des fonctions de base pour les matrices de taille statique.

**Pivot de Gauss - Résolution de Ax=b** Adapté d'une partie du chapitre 1 d'Analyse Numérique (les formules sont exactement les mêmes, les indices aussi !). On cherche un résoudre un système $Ax = b$ avec A matrice $n \times n$, $b$ vecteur de taille $n$ :

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + .... + a_{1,n}x_n = b_1, \\ a_{2,1}x_1 + a_{2,2}x_2 + .... + a_{2,n}x_n = b_2, \\ ... \\ a_{n,1}x_1 + a_{n,2}x_2 + .... + a_{n,n}x_n = b_n, \end{cases} \tag{1}$$

où les coefficients $a_{i,j}$, pour $i, j \in \{1, \cdots, n\}$ sont les coefficients de la matrice $A$. *On suppose que $A$ est inversible, donc que le système n'a qu'une unique solution.* **Attention !** en maths les indices vont de 1 à $n$ compris, en algo de 0 à $n-1$ !

**Étape 1 : résolution dans le cas triangulaire** *En prenant un exemple simple*, se convaincre que si A est triangulaire supérieure, alors on peut calculer aisément l'unique solution du système à l'aide des formules suivantes :

$$\begin{cases} x_n = \dfrac{b_n}{a_{n,n}}, \\ x_{n-1} = \dfrac{b_{n-1} - a_{n-1,n}x_n}{a_{n-1,n-1}}, \\ \cdots \\ x_1 = \dfrac{b_1 - a_{1,2}x_2 - ... - a_{1,n}x_n}{a_{1,1}}. \end{cases} \tag{2}$$

1. Vérifier que $x_i = \dfrac{b_i - \sum_{k=i+1}^{n} a_{i,k}x_k}{a_{i,i}}$.
2. Écrire donc une fonction C (attention aux indices), de signature :

   ```
   void solve_triangular_system(float A[N][N], float b[N],float result[N]);
   ```

   qui étant donnés $A$ triangulaire supérieure, $b$ un vecteur, calcule et stocke dans *result* l'unique solution (obtenue par les formules que l'on vient d'obtenir).
3. Tester votre fonction avec $A = \begin{pmatrix} 2 & 3 \\ 0 & -5.5 \end{pmatrix}$ et $b = \begin{pmatrix} 3 \\ 2.5 \end{pmatrix}$. On doit trouver $\begin{pmatrix} 2.181818 \\ -0.454545 \end{pmatrix}$.

**Étape 2 : pivotage pour arriver triangulaire supérieure** On cherche donc, étant donné $A$ inversible quelconque, et $b$ vecteur, à rendre $A$ triangulaire supérieure, et à modifier $b$ en conséquence pour que le système obtenu soit équivalent au système initial. On va modifier $A$ et $b$, par étapes, de façon à ce qu'à l'étape $i$ la matrice $A$ soit de la forme :

$$\begin{pmatrix} a_{1,1}^{(1)} & \dots & \dots & \dots & \dots & \dots & \dots & \dots & a_{1,n}^{(1)} \\ 0 & \ddots & & & & & & & \vdots \\ \vdots & \ddots & \ddots & & & & & & \vdots \\ \vdots & & 0 & a_{i,i}^{(i)} & & & & & \vdots \\ \vdots & & \vdots & a_{i+1,i}^{(i)} & \ddots & & & & \vdots \\ \vdots & & \vdots & \vdots & & \ddots & & & \vdots \\ \vdots & & \vdots & \vdots & & & \ddots & & \vdots \\ 0 & \dots & 0 & a_{n,i}^{(i)} & \dots & \dots & \dots & \dots & a_{n,n}^{(i)} \end{pmatrix}.$$

Après l'étape $n-1$ la matrice $A$ sera donc triangulaire supérieure.

1. *En prenant un exemple de taille* $3 \times 3$, vérifier que les formules suivantes réalisent l'étape $i$ de l'algorithme de Gauss :
   - Pour $k \leq i$ et pour $j = 1, \cdots, n,$, la matrice $A$ et le vecteur $b$ sont inchangés.
   - Pour $k > i$, **en supposant que** $a_{i,i} \neq 0$, [1] on modifie $A$ et $b$ de la façon suivante :

$$a_{k,j} \leftarrow a_{k,j} - \frac{a_{k,i}}{a_{i,i}} a_{i,j}, \text{ pour } j = 1, \cdots, n, \tag{3}$$

$$b_k \leftarrow b_k - \frac{a_{k,i}}{a_{i,i}} b_i. \tag{4}$$

2. $i$ varie de quoi à quoi ?
3. Écrire donc une fonction C (attention aux indices), de signature :

   ```
   void gauss_pivot_simple(float A[N][N], float b[N])
   ```

   qui transforme $A$ et $b$ de façon à rendre $A$ triangulaire supérieure en gardant le système $Ax = b$ équivalent à l'initial.
4. Tester votre fonction avec $A = \begin{pmatrix} 2 & 3 \\ 5 & 2 \end{pmatrix}$ et $b = \begin{pmatrix} 3 \\ 10 \end{pmatrix}$. . On doit trouver après modifications $A = \begin{pmatrix} 2 & 3 \\ 0 & -5.5 \end{pmatrix}$ et $b = \begin{pmatrix} 3 \\ 2.5 \end{pmatrix}$.

Les notions acquises en Programmation Structurée, notamment concernant la conception d'algorithmes et leur réalisation en C, sont un pré-requis du cours d'Analyse Numérique du S6. Le langage algorithmique a été normalisé entre les deux enseignements. Les choix d'implémentation seront quelquefois différents en Matlab, qui possède des fonctions de traitement mathématiques spécialisées. En bonus, on vous fournit un tableau récap C/Matlab !

1. Le cours d'Analyse numérique vous dira quoi faire dans le cas contraire !