IMT Nord Europe
École Mines-Télécom
IMT-Université de Lille

Luc Fabresse
luc.fabresse@imt-nord-europe.fr
CERI Systèmes Numériques

Note : Dans ce TP et tous les suivants, organisez vos programmes en fonctions. Essayez d'écrire des fonctions réutilisables et réutilisez les chaque fois que l'occasion se présente.

Tableaux, structures et pointeurs

1. Ecrire et tester une fonction `echangeContenu` qui interverti le contenu de deux variables entières. Pour illustrer le traitement réalisé par cette fonction, supposons que l'on dispose de deux variables entières : a initialisée à 10 et b initialisée à 20. Après l'appel, a aura pour valeur 20 et b sera égale à 10. La fonction `echangeContenu` ne retourne rien. Elle a en revanche deux paramètres. A vous de déterminer leurs types.
2. **Matrices**
   Écrire un programme qui permet de multiplier deux matrices carrées de taille 5. Vous pouvez définir cette taille comme une constante pré-processeur (cf. ci-dessous `SIZE`).

```c
#include <stdint.h>
#include <stdlib.h>

#define SIZE 5

int main(void) {
   //matrices en ligne * colonne
   int64_t matrice1[][SIZE]={{1,2,3,4,5},{1,2,3,4,5},{1,2,3,4,5},{1,2,3,4,5},{1,2,3,4,5}};
   int64_t matrice2[][SIZE]={{6,7,8,9,10},{6,7,8,9,10},{6,7,8,9,10},{6,7,8,9,10},{6,7,8,9,10}};
   int64_t matriceResultat[SIZE][SIZE];

   matrix_mult(matriceResultat,matrice1,matrice2);
   matrix_print(matriceResultat);

   return EXIT_SUCCESS;
}
```

   (a) Écrire les fonctions `matrix_mult` et `matrix_print` pour faire fonctionner la fonction `main` ci-dessus. Utilisez des tableaux. N'utilisez **pas** de pointeurs pour l'instant. Notez que `matrix_mult` prend 3 paramètres dont le premier est la matrice résultat calculée.

   Rappel : Soit A, B et C des matrices telles que $C = A.B$. Le produit de deux matrices $A.B$ ne peut se définir que si le nombre de colonnes de A est égal au nombre de lignes de B.

$$\begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} . \begin{pmatrix} b_{11} & \dots & b_{1q} \\ \vdots & \ddots & \vdots \\ b_{p1} & \dots & b_{pq} \end{pmatrix} = \begin{pmatrix} c_{11} & \dots & c_{1q} \\ \vdots & \ddots & \vdots \\ c_{m1} & \dots & c_{mq} \end{pmatrix} \text{ où } \begin{cases} n = p \\ c_{ij} = \sum_{k=1}^{n} a_{ik} * b_{kj} \end{cases}$$

   (b) Que pensez-vous de la fonction `mult_matrice` en terme de réutilisation ?

   (c) Comment feriez-vous pour généraliser ces deux fonctions pour des tailles quelconques de matrices ? On continue à supposer que l'initialisation des matrices est faite directement dans le `main`.

3. **Manipuler des dates**

   (a) Définir les types :
   - `Mois` représente les mois avec une énumération (cf. cours)
   - `Date` représente une date avec une structure (cf. cours)

   (b) Dans la fonction `main`, ajouter le code suivant :

```c
Date d;
initialiseDate(&d); //Pourquoi a t-on ajouté un &?
afficheDate(&d);
```

Ecrivez le code des 2 fonctions :
- `initialiseDate` qui initialise une date dont l'adresse a été passée en paramètre en utilisant des `scanf`
- `afficheDate` qui affiche une date dont l'adresse a été passée en paramètre

(c) Dans la fonction `main`, ajouter le code suivant :

```
Date d;
d = creerDateParCopie();
afficheDate(&d);
```

Ecrivez le code de la fonction `creerDateParCopie` qui doit également créer et initialiser une date en utilisant des `scanf`

(d) Dans la fonction `main`, ajouter le code suivant :

```
Date *date;
date = newDate();
afficheDate(date);
//...
free(date);
```

Ecrivez le code de la fonction `newDate` qui créé une Date retourne son adresse (cf. Cours 3 `malloc`). N'oubliez pas de restituer (`free`) la mémoire allouée dans ce cas.

(e) Comparer ces 3 fonctions. Quelle version (`initialiseDate`, `creerDateParCopie` ou `newDate`) ne devrait pas être utilisée et pourquoi ?

(f) (*bonus*) Définir les fonctions suivantes :
- `unsigned int nbreJours(Mois mois, unsigned int annee)` : retourne le nombre de jours du mois de l'années correspondante.
- `bool dateValide(Date uneDate)` : indique si `uneDate` est valide ou non
- `unsigned int jourDansAnnee(Date uneDate)` : retourne le numéro du jour dans l'année correspondante à la date donnée en paramètre.

4. Faire "beer-song" sur `exercism.org` (string, array, `malloc/free`)

5. **Bonus :** Fractions rationnelles et sommes harmoniques
On représente une fraction rationnelle ($\frac{x}{y}$) par un type enregistrement (`struct`) ayant deux champs entiers : `numerateur` et `denominateur`.

(a) Écrire une fonction *simplifie* qui, étant donné une fraction rationnelle (créer un type !), a pour résultat la fraction irréductible correspondante.
Par exemple : $\frac{45}{60} = \frac{3}{4}$,

(b) Écrire une fonction permettant d'additionner deux fractions rationnelles ; la fraction obtenue devra être irréductible.

(c) Écrire une fonction qui étant donné N, un entier positif, calcule :

$$H(N) = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{N}$$

Le résultat doit être donné sous la forme d'une fraction irréductible.