TP1 : Premiers programmes C

CSH : Initiation au C et au shell

★ **Exercice 1: Premier programme, et entrées/sorties.**

▷ **Question 1:** Écrivez un programme C qui calcule le maximum de 3 entiers saisis au clavier par l'utilisateur, et qui affiche le résultat. Compilez-le sous le nom ***max3***.

▷ **Question 2:** Vérifiez que votre programme est correct. Quels tests effectuez–vous pour cela ?

★ **Exercice 2: Encore des entrées/sorties : problèmes de date.** Les signes astrologiques sont, en fonction de la date, les suivants :

- Bélier : 21/3 au 19/4
- Taureau : 20/4 au 20/5
- Gémeaux : 21/5 au 20/6
- Cancer : 21/6 au 21/7
- Lion : 22/7 au 22/8
- Vierge : 23/8 au 22/9
- Balance : 23/9 au 22/10
- Scorpion : 23/10 au 21/11
- Sagitaire : 22/11 au 21/12
- Capricorne : 22/12 au 19/1
- Verseau : 20/1 au 18/2
- Poisson : 19/2 au 20/3

▷ **Question 1:** Écrivez un programme demandant la date de naissance de l'utilisateur et indique son signe astrologique. Si les valeurs entrées pour le jour ou le mois ne sont pas valides, afficher un message d'erreur et quiter le programme.

▷ **Question 2:** Indiquez le jour de la semaine à laquelle l'utilisateur est né, en utilisant la formule de Zeller : Le jour de la semaine de la date $\Delta = (j, m, a)$ est

$$w = j + \lfloor 2.6 \times t - 0.2 \rfloor + d + \left\lfloor \frac{d}{4} \right\rfloor + \left\lfloor \frac{c}{4} \right\rfloor + 5 \times c$$

$$\text{Avec } t = \begin{cases} m+10 & (\text{si } m \leq 2) \\ m-2 & (\text{sinon}) \end{cases} ; b = \begin{cases} a-1 & (\text{si } m \leq 2) \\ a & (\text{sinon}) \end{cases} ; c = \left\lfloor \frac{b}{100} \right\rfloor ; d = b - 100 \times c.$$

Ensuite, si $w \equiv 0 \pmod 7$, Δ est un lundi. Si cette grandeur vaut 1, Δ est un mardi. Pour 2, c'est un mercredi, etc.

▷ **Question 3:** Demandez également son nom à l'utilisateur pour améliorer l'esthétique des affichages.

★ **Exercice 3: Manipuler des entiers : le triangle de Pascal**

On souhaite écrire un programme qui construit le triangle de Pascal de degré N et l'affiche jusqu'à la diagonale incluse.

Les éléments du triangle de Pascal se calculent comme suit (si P est la matrice mémorisant le triangle) : $P_{i,j} = P_{i-1,j} + P_{i-1,j-1}$

Voici quelques exemples d'affichage pour différentes valeurs de N.

Affichage quand N vaut 6

```
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
```

Affichage quand N vaut 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
```

▷ **Question 1:** Écrivez ce programme. Dans un premier temps, vous écrirez simplement la fonction `int pascal(int i, int j)` et vous calculerez chacune des valeurs de cette façon.

L'approche choisie est relativement inefficace : Afficher le triangle de taille 100 est par exemple très très lent. Le problème vient du fait que de nombreux calculs sont effectués plusieurs fois. Par exemple, calculer $P_{5,3}$ demande de calculer $P_{3,2}$ à deux reprises (une fois pour avoir $P_{4,2}$ et une fois pour avoir $P_{4,3}$). Pour résoudre ce problème, nous allons utiliser un tableau pour stocker les calculs déjà faits afin d'éviter de recalculer deux fois la même chose.

▷ **Question 2:** Ajoutez un tableau d'entiers en global à votre programme en écrivant les lignes suivantes en dehors de toute fonction. Elles créent un tableau à deux entrées de taille 100×100, et on peut augmenter la taille en changeant la valeur de la macro.

```
#define MAX 100
int tab[MAX][MAX];
```

Ajoutez une double boucle au début de votre fonction `main()` pour remplir le tableau de 0.

▷ **Question 3:** Modifiez votre fonction `pascal()`. Avant de faire le moindre calcul, vous consulterez la case correspondante du tableau. Si cette case contient une valeur non-nulle, la fonction retournera directement cette valeur sans la recalculer. Si la case contient 0, la fonction calculera le résultat comme avant, et stockera le résultat dans la case avant de le retourner en résultat. Comparez la vitesse d'affichage du triangle de hauteur 100.

▷ **Question 4:** (optionnelle) Il est possible d'optimiser la quantité de mémoire nécessaire en ne stockant qu'une seule ligne du tableau. Il faut alors faire les calculs dans le bon ordre pour ne pas effacer des cases dont des calculs futurs auront besoin. Calculer de droite à gauche permet de faire ceci.

★ **Exercice 4: Tableaux de caractères : Cadavres exquis** L'objectif de cet exercice est de constituer un écrivain automatique. Pour cela, il faut initialiser plusieurs tableaux de chaines de caractères : un tableau de sujets accompagnés d'un article, un tableau de verbes conjugués à la troisième personne, un tableau de compléments d'objets directs au masculin et un tableau d'adjectifs au masculin.

Sujet	*Adjectif*	*Verbe*	*Complément*	*Adjectif*
Le cadavre	exquis	boira	le vin	nouveau
Le chat	noir	mange	un rat	gris
Le chien	pouilleux	aime	un os	moelleux
La grenouille	gluante	capture	un moustique	agressif

Ce jeu fut inventé par les surréalistes vers 1954. La première pharse construite sur ce modèle fut «Le cadavre exquis boira le vin nouveau», ce qui donna le nom du jeu.

▷ **Question 1:** Le premier problème a résoudre est de savoir comment stocker les morceaux de phrases qui seront combinés ensuite. Le plus simple est d'utiliser des tableaux statiques de chaines statiques. Vous avez deux possibilités, présentées dans les deux blocs suivants. Choisissez l'une d'entre elles, et écrivez le début de votre programme.

Première approche pour le stockage des données

```
char *nom[]={"le cadavre","le chat","le chien","la grenouille"};
char *adj1[]={"exquis","noir","pouilleux","gluant"};
...
```

Deuxieme approche pour le stockage des données

```
char *elements[][]={
  {"le cadavre","le chat","le chien","la grenouille"},
  {"exquis","noir","pouilleux","gluant"},
  ...
  };
```

▷ **Question 2:** Complétez votre programme.
Obtenir un nombre aléatoire en C n'est pas très compliqué. Il faut utiliser la fonction `rand()` qui retourne un entier entre 0 et `RAND_MAX`. Le problème est que la suite pseudo-aléatoire est toujours la même par défaut. Initialiser cette suite à l'heure actuelle est une bonne solution à ce manque de variété.

```
#include <stdlib.h>
#include <time.h>

int main() {
  // Initialise le pseudo aleatoire
  srand(time(NULL));
  // tire un nombre entre 0 et 3
  int nb = rand() % 4;
}
```

★ **Exercice 5: Structures en C : le programme annuaire**

On souhaite créer un programme en C gérant un annuaire très simplifié qui associe à un nom de personne un numéro de téléphone.

▷ **Question 1:** Créer une structure Personne pouvant contenir ces informations (nom et téléphone). Le nom peut contenir 32 caractères et le numéro 16 caractères.

▷ **Question 2:** Créer une nouvelle structure qui va représenter le carnet d'adresses. Cette structure Carnet contiendra un tableau de 20 Personne et un compteur indiquant le nombre de personnes dans le tableau.

▷ **Question 3:** Créer ensuite une fonction qui renvoie une structure Personne en prenant en argument un nom et un téléphone.

▷ **Question 4:** Rajouter une fonction qui affiche les informations contenues dans la structure Personne passée en argument.

▷ **Question 5:** Créer une fonction qui ajoute une personne dans un carnet.

▷ **Question 6:** Créer une fonction qui affiche un carnet.

▷ **Question 7:** À partir des étapes précédentes, faire programme gérant un carnet d'adresse. Créer un menu qui propose d'ajouter une nouvelle personne, d'afficher le carnet ou de quitter.

Une extension possible serait d'ajouter une fonction de sauvegarde de l'annuaire dans un fichier, et de faire en sorte que les données sauvegardées puissent être lues automatiquement au démarrage.

★ **Exercice 6: Pour aller plus loin : les pyramides en C.**

Reprenez problème des pyramides vu en TD et en TP dans le module de TOP [1]. Implémentez la forme complète de ce problème, où les numéros des billes ne sont pas forcément des nombres successifs. Utilisez l'algorithme le plus efficace vu pour ce problème, c'est-à-dire le backtracking avec remplissage par diagonale (et utilisation de la fonction `propage()`).

▷ **Question 1:** Quel est le nombre entier n le plus petit tel que l'on puisse remplir une pyramide de hauteur 6 avec toutes les numéros de billes utilisées inférieurs à n ?

▷ **Question 2:** Quelle est la taille de pyramide maximale pour laquelle vous parvenez à trouver un remplissage ?

Si vous parvenez à résoudre cet exercice, bravo. Faites un mail à `Martin.Quinson@loria.fr` *pour montrer votre solution.*

1. Ces sujets sont disponibles sur la page `http://www.loria.fr/~quinson/teach-prog.html`