TP1 : Premiers programmes C

★ **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

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

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 dimanche. Si cette grandeur vaut 1, Δ est un lundi. Pour 2, c'est un mardi, 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 $N = 6$ et $N = 7$.

```
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
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.