

## Séance 3&4 - Prise en main des fonctions

### A FAIRE SUR PAPIER

**Exercice 1** Ecrire les signatures des fonctions suivantes :

- une fonction, nommée `admis` qui permet de déterminer si un étudiant, identifié par son numéro et dont on connaît la moyenne, est admis ou non.
- une fonction, nommée `moyennePonderee`, qui calcule la moyenne pondérée d'une note de contrôle continu et d'une note d'examen dont les coefficients sont donnés.
- une fonction, nommée `tableMult`, qui affiche à l'écran la table de multiplication d'un entier positif donné.
- une fonction nommée `tablesMult`, qui affiche à l'écran toutes les tables de multiplication de 2 à 10.
- une fonction nommée `premier`, qui permet de déterminer si un entier positif est premier ou non.
- une fonction nommée `nombreDiviseurs`, qui calcule le nombre de diviseurs d'un entier positif.

**Exercice 2** Ecrire dans un programme, les appels suivants de fonctions (définies dans l'exercice précédent). Ajouter dans le programme, lorsque c'est possible, l'affectation du résultat de cet appel à une variable. Ajouter dans le programme, si nécessaire, l'instruction qui permet d'afficher à l'écran ce qui résulte de l'exécution de la fonction.

- L'appel de la fonction `admis` pour un étudiant, identifié par son numéro 20156708 et dont la moyenne est égale à 12.6.
- L'appel de la fonction `moyennePonderee` avec une note de contrôle continu 12 (coefficient 3) et une note d'examen 13 (coefficient 5).
- L'appel de la fonction `admis` pour un étudiant, identifié par son numéro 20156708 dont la note de contrôle continu est égale à 15 (coefficient 2) et la note d'examen 8 (coefficient 5).
- L'appel de la fonction `tableMult`, qui affiche à l'écran la table de multiplication de 5..
- L'appel de la fonction `tablesMult`, qui affiche à l'écran toutes les tables de multiplication de 2 à 10.
- L'appel de la fonction `premier`, qui calcule si 567 est premier ou non.
- L'appel de la fonction `nombreDiviseurs`, qui calcule le nombre de diviseurs d'un entier positif avec  $x = 8$ .

**Pour écrire une fonction, on veillera par la suite à distinguer les fonctionnalités de saisie, de calcul et d'affichage.**

**Exercice 3** Écrire :

- une fonction, nommée `f1`, se contentant d'afficher "bonjour".
- une fonction, nommée `f2`, qui prend un paramètre entier tel que, pour une valeur donnée  $n$ , l'exécution de `f2(n)` affiche "bonjour"  $n$  fois.

**Exercice 4** Écrire 2 fonctions permettant de déterminer si la valeur de l'argument est multiple de 2 (pour la première fonction) ou multiple de 3 (pour la seconde fonction).

Utiliser ces deux fonctions dans un petit programme qui permet la saisie d'un nombre entier et qui affiche si ce dernier est pair, multiple de 3 et/ou multiple de 6. Lorsque le nombre est un multiple de 6, le programme n'affichera que cette seule information. Simuler l'exécution du programme pour les entiers suivants : 2,3,4,6,8,9.

**Exercice 5** Écrire une fonction nommée `saisieNote` qui :

- affiche à l'écran un message destiné à l'utilisateur pour préciser le domaine des valeurs attendues. Par exemple "Donnez une note entre 0 et 20".
- réalise le contrôle de la saisie.
- réitère la demande tant que la valeur saisie n'est pas comprise dans le domaine des valeurs attendues.

Ecrire une fonction main qui réalise la saisie de deux notes.

## A FAIRE SUR MACHINE

**Exercice 6** On veut calculer la moyenne de  $n$  notes saisies au clavier. L'utilisateur saisit au départ le nombre de notes (nb) dont il souhaite calculer la moyenne. Chaque note saisie doit faire l'objet d'un contrôle (note entre 0 et 20).

1. Ecrire une fonction **moyenne** prenant un entier  $n$  en paramètre, et qui demande les  $n$  notes et calcule leur moyenne.
2. Ecrire un programme permettant de tester vos fonctions. Vous testerez les cas suivants : nb=0, nb=1, nb=5.

**Exercice 7** On souhaite trouver les entiers  $x$  inférieurs à 1000 tel que  $x$  est égal à la somme des cubes de ses chiffres. Par exemple  $371 = 3^3 + 7^3 + 1^3$ .

1. Ecrire une fonction qui teste chaque entier compris entre 2 et 999
2. Ecrire une fonction qui teste chaque triplet  $(a, b, c)$  avec  $a, b$  et  $c$  compris entre 0 et 9.

**Exercice 8 (Prochain premier)**

1. Écrire une fonction **premier** qui prend un paramètre entier et renvoie true si n est premier et false sinon. Tester votre fonction pour les entiers 2,3,4,7,9.
2. Écrire une fonction **prochainpremier** qui prend un paramètre entier et qui renvoie le plus petit nombre premier supérieur ou égal à  $n$ .
3. Écrire un programme qui demande un entier  $n$  à l'utilisateur et affiche le nombre premier immédiatement supérieur ou égal à  $n$ . Tester votre programme pour les entiers 11, 21.

**Exercice 9** Écrire une fonction arbre qui, pour un entier naturel impair  $base$ , affiche un triangle isocèle d'étoiles, ayant pour base,  $base$  étoiles. Ecrire un programme où la valeur de la base sera saisie par l'utilisateur et qui réalisera un contrôle de saisie. Avec 5 comme base, nous obtenons par exemple :

```
*
***
*****
```

Tester la fonction arbre avec  $base = 1$ ,  $base = 3$ ,  $base = 5$ .

## A FAIRE SUR PAPIER

**Exercice 10** On désigne par  $C_n^p$  le nombre de combinaisons sans répétitions de  $p$  objets parmi  $n$ . On peut les calculer de deux manières différentes (entre autres).

$$C_n^p = C_{n-1}^p + C_{n-1}^{p-1}$$

$$C_n^p = \frac{n}{p} C_{n-1}^{p-1}$$

avec  $C_k^k = 1$  et  $C_n^0 = 1$

1. Utiliser la première relation de récurrence pour écrire une fonction récursive qui calcule  $C_n^p$
2. Ecrire une seconde fonction à partir de la seconde relation.
3. Ecrire une fonction **triangle** qui pour un entier naturel donné k, affiche le triangle de Pascal pour n et p variant de 0 à k et  $n \geq p$

```

C0^0
C1^0 C1^1
C2^0 C2^1 C2^2
  ⋮   ⋮   ⋮
Ci^0 Ci^1   ⋮   Ci^j
  ⋮   ⋮   ⋮   ⋮
C10^0                                C10^10
```

```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
⋮ ⋮   ⋮
⋮ ⋮   ⋮   ⋮
1 10 45 120 210 252 ... 45 10 1
```

## A FAIRE SUR MACHINE

**Exercice 11** Le nombre  $\pi$  peut être approché en utilisant la suite de Leibnitz :

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots$$

On est assuré que l'erreur sur le calcul est inférieure à  $\frac{1}{p}$  si le dernier terme ajouté à la somme est inférieur en valeur absolue à  $\frac{1}{p}$ . Écrire une fonction qui prend en paramètre un réel  $\epsilon < 1$  et qui calcule la valeur de  $\frac{\pi}{4}$  avec une erreur inférieure à  $\epsilon$ . Écrire un programme qui demande de saisir une précision inférieure à 1 et qui affiche l'approximation de  $\pi$ . Tester votre programme avec les précisions 0.1, 0.01, 0.001, 0.00001.

**Exercice 12 (Ackermann)** Écrire une fonction récursive calculant la valeur la fonction d'Ackermann, définie par pour  $m \geq 0$  et  $n \geq 0$  :

$$\begin{aligned} A(0, n) &= n + 1 \text{ pour } n \geq 0, \\ A(m, 0) &= A(m - 1, 1) \text{ pour } m > 0, \\ A(m, n) &= A(m - 1, A(m, n - 1)) \text{ sinon.} \end{aligned}$$

Écrire un programme qui calcule la valeur de la fonction d'Ackermann pour deux entiers saisis au clavier. Tester votre programme pour les couples de valeurs (3,4) et (2,6).

**Exercice 13** On veut écrire un programme qui affiche pour une date donnée, le jour de la semaine. Ecrire trois fonctions de saisie : **saisieAnnee**, **saisieMois** et **saisieJour** qui vérifient la validité des dates saisies. Un mois sera codé par un entier de 1 à 12. On prend en compte que le mois de janvier a 31 jours. Pour une année bissextile, le mois de février compte 29 jours et 28 jours sinon. On notera le jour de la semaine par un entier compris entre 0 et 6 avec 0 pour dimanche, 1 pour lundi, 2 pour mardi...

Ecrire une fonction **afficheJour** qui affiche en clair le jour de la semaine.

Ecrire une fonction **compteJours** qui calcule le nombre de jours écoulés depuis le début de l'année et le mois m compris.

Ecrire une fonction **compteNombresAnneesBissextiles** qui calcule le nombre d'années bissextiles jusqu'à l'année a comprise.

Ecrire une fonction **codeJourAnnee** qui calcule le code (entre 0 et 6) du jour de la semaine pour une date (jour, mois, année).

Ecrire le programme qui demande une date à l'utilisateur et affiche le jour de la semaine correspondant.

Tester votre programme grâce à l'utilitaire cal a qui affiche le calendrier de l'année a.

Faire des tests significatifs : 1 janvier 2004, 15 février 2004, 30 mars 2004, 1 janvier 2015, 18 février 2015, 5 juin 2015.