Travaux dirigés n° 2

Les fonctions

Exercice 1 (Date du lendemain)
On désire écrire un algorithme permettant, après saisie de la date du jour, d'afficher la date du lendemain :

- un message d'erreur doit être affiché si la valeur saisie pour le mois n'est pas dans [1 ; 12] ;
- un message d'erreur doit être affiché si la valeur saisie pour le jour n'est pas conforme (en tenant compte du mois et de l'année).

Rappel : une année bissextile est une année comportant 366 jours (au lieu de 365). Une année est bissextile lorsqu'elle est soit divisible par 4, mais non divisible par 100, soit divisible par 400.

1°) Écrivez une fonction permettant de déterminer si une année est bissextile.

2°) Écrivez une fonction donnant le nombre de jours d'un mois d'une année déterminée.

3°) Écrivez une fonction permettant de déterminer si une date est valide.

4°) Écrivez, finalement, un algorithme qui, après saisie d'une date et vérification de sa conformité, permet d'obtenir la date du lendemain.

5°) Écrivez une fonction permettant de comparer deux dates. Elle retournera vrai si la première date est inférieure ou égale à la seconde.

Exercice 2 (Traces d'exécution)
Effectuez la trace d'exécution des algorithmes "*Algo1*" et "*Algo2*" suivants :

```
Algorithme Algo1
Déclarations
      Variables
            x, y : entier
Début
{m1}  x ← 1
{m2}  y ← 2
{m3}  x ← f(x + y, y)
{m4}  écrire(x + " " + y)
{m5}  y ← f(x + y, y)
{m6}  écrire(x + " " + y)
Fin


Fonction f(u : entier, v : entier) : entier
Déclarations
      Variables locales
            x, resultat : entier
Début
{f1}  x ← 2 * u + 1
{f2}  Si x > 4 Alors
{f2a.1}     resultat ← v - u
      Sinon
{f2b.1}     resultat ← v + u
      FinSi
{f3}  retourner(resultat)
Fin
```

```
Algorithme Algo2
Déclarations
      Variables
            i : entier
Début
{m1}  i ← 2
{m2}  écrire("Principal: " + i + " " + g(i))
Fin


Fonction f(i : entier) : entier
Début
{f1}  i ← i + 1
{f2}  écrire("f: " + i)
{f3}  retourner(i)
Fin


Fonction g(j : entier) : entier
Début
{g1}  j ← j + 2
{g2}  écrire("g: " + j)
{g3}  retourner(f(j) + j)
Fin
```

Exercice 3 (Nombres premiers)
On dispose d'une fonction `estPremier(n : entier) : booléen` qui teste si l'entier n est un nombre premier.

1°) Écrivez une fonction qui compte le nombre de nombres premiers compris dans un intervalle d'entiers positifs fixé à $[a, b]$. Par exemple : `comptePremiers(5,15) == 4`.

2°) Écrivez une fonction qui, pour un entier n donné, calcule le nombre premier de rang n. Par exemple, `premier(1)` retourne 2 car 2 est le premier nombre premier et `premier(5)` retourne 11 car 11 est le cinquième.

3°) Écrivez une fonction qui détermine le rang d'un nombre premier.

4°) Écrivez une fonction qui détermine le plus petit diviseur ($\neq 1$) d'un entier au moins égal à 2.

5°) Déduisez-en une fonction qui teste si un entier est premier.

Exercice 4 (Suites numériques)
1°) Soit une suite numérique $(u_i)_{i\in\mathbb{N}}$.

Pour $n \in \mathbb{N}$ on pose :

$$S_n = \sum_{i=0}^{n} u_i \qquad P_n = \prod_{i=0}^{n} u_i \qquad M_n = \max\{u_i \mid 0 \leq i \leq n\} \qquad m_n = \min\{u_i \mid 0 \leq i \leq n\}$$

Écrivez les fonctions permettant de calculer les termes des suites $(S_n)_{n\in\mathbb{N}}$, $(P_n)_{n\in\mathbb{N}}$, $(M_n)_{n\in\mathbb{N}}$ et $(m_n)_{n\in\mathbb{N}}$. On suppose disposer de la fonction `u(i : entier) : entier` qui calcule le terme u_i.

2°) Soit $(x, n) \in \mathbb{R} \times \mathbb{N}$, écrivez une fonction qui calcule $S = \sum_{k=0}^{n} \frac{x^k}{k!}$

Exercice 5 (Coefficient binomial et triangle de Pascal)
On désire calculer et afficher le triangle de Pascal. Il est possible de le faire en utilisant les coefficients binomiaux : les coefficients $\begin{pmatrix} n \\ p \end{pmatrix}$ pour $0 \leq p \leq n$ se trouvent sur la n-ième ligne du triangle, la première ligne étant la ligne 0.

1°) Soit $(n, p) \in \mathbb{N} \times \mathbb{N}$ avec $0 \leq p \leq n$, écrivez une fonction qui calcule le coefficient binomial.
Rappel : $C_n^p = \begin{pmatrix} n \\ p \end{pmatrix} = \frac{n!}{p!(n-p)!} = \frac{n(n-1)\ldots(n-p+1)}{p!}$

2°) Écrivez la procédure `afficheLigne(n : entier)` qui affiche la ligne n du Triangle de Pascal. Si l'on passe 3 comme paramètre, l'appel à cette fonction produira l'affichage 1, 3, 3 et 1 (la première ligne est la ligne 0).

3°) Écrivez la procédure `affichePascal(n : entier)` qui affiche les n premières lignes du Triangle de Pascal.

Exercice 6 (La suite de Syracuse)
On considère la suite de Syracuse, définie par la donnée de $S_0 \in \mathbb{N}^*$ et par la relation de récurrence : $S_{n+1} = S_n/2$ si S_n est pair ; $S_{n+1} = 3S_n + 1$ si S_n est impair.

1°) Écrivez une fonction/procédure qui prend en paramètre S_0 et un nombre de termes, puis qui affiche le nombre de termes demandés.

2°) Écrivez un algorithme qui saisit S_0 et un nombre de termes, puis qui affiche le nombre de termes demandés en utilisant la fonction/procédure de la question précédente.

Après un nombre de termes qui varie en fonction de S_0, la suite de Syracuse finit toujours par atteindre la valeur 1.

3°) Modifiez la fonction/procédure écrite précédemment de sorte à ce qu'elle affiche tous les termes jusqu'à rencontrer la valeur 1 et retourne la longueur de la suite, c'est-à-dire le nombre de termes qui ont été nécessaires à l'atteinte de cette valeur.

4°) En plus du nombre de termes nécessaires, on veut aussi retourner la moyenne des termes calculés. Est-ce possible de le faire ?