Feuille de TD/TP n° 4

Sous-programmes et récursivité

Exercice 1 : [Analyse de code] Indiquer, sans l'exécuter, ce qu'affiche le programme suivant :

```
procedure Ex1 is
   function F (N : Integer) return Integer;
   procedure B (BB : out Integer; AA : in Integer);

   procedure A (X : in Integer; Y : out Integer) is
   begin
      Put ("A : X = "); Put (X, 0); New_Line;
      B (Y, F (X));
   end;

   function F (N : Integer) return Integer is
      Z : Integer;
   begin
      Put ("F : N = "); Put (N, 0); New_Line;
      B (Z, N - 1);
      return Z / 2;
   end;

   procedure B (BB : out Integer; AA : in Integer) is
   begin
      if AA <= 0 then
         Put_Line ("B : Fin !");
      else
         Put ("B : AA = "); Put (AA, 0); New_Line;
         A (AA - 1, BB);
      end if;
   end;

begin
   Put (F (5), 0); New_Line;
end Ex1;
```

Exercice 2 :

- Écrire une procédure itérative qui prend en paramètre un entier et affiche chaque chiffre séparé par un point : par exemple, si l'entier `1234` est passé en paramètre, cela affichera `"1.2.3.4"`.
- Même question avec une procédure récursive.

- Écrire une fonction itérative qui prend en paramètre un entier et renvoie la somme de ses chiffres.
- Même question avec une fonction récursive.

Exercice 3 : Reprendre l'exercice 2 de la feuille n°3.

1. Écrire une procédure qui affiche les informations d'une carte passées en paramètres.
2. Écrire une fonction `PlusGrand` qui prend en paramètre les informations de deux cartes et renvoie `True` si et seulement si la première carte est plus grande que la seconde au sens donné dans la feuille n°3.
3. Écrire une procédure récursive qui affiche, dans l'ordre croissant, toutes les cartes comprises entre deux cartes saisies par l'utilisateur.

Exercice 4 : Reprendre l'exercice 5 de la feuille n°3.

1. Écrire une procédure qui affiche les infromations d'une date passées en paramètres.
2. Écrire une fonction `EstApres` qui prend en paramètre les informations de deux dates et renvoie `True` si et seulement si la deuxième date est après la première.
3. Écrire une fonction `EstValide` qui prend en paramètre une date et renvoie `True` si la date est valide (on considèrera que les mois de février ont toujours 29 jours).
4. Écrire une fonction `NbJours` qui prend en paramètre une date et renvoie le nombre de jours écoulés depuis l'origine des dates jusqu'à cette date. Si la date est négative, ce nombre sera négatif.
5. Écrire une procédure `DateJours` qui prend en paramètre d'entrée un nombre de jours, et écrit dans des paramètres de sortie la date (jour, mois, annee).