AP1 TD7 – Sous programmes I : correction

Exercice 1 – Passage par adresse et par valeur

Soit le programme suivant :

```
ALGORITHME TD7

PROCEDURE ma_procedure(a, b, c, d)
PARAMETRES par VALEUR : a, b, d : entiers
PARAMETRES par ADRESSE: c : entiers
DEBUT
      c ← a + b
      d ← a × b
FIN

VARIABLES x, y, z, t: entiers
DEBUT
      saisir x
      saisir y
      ma_procedure(x, y, z, t)
      afficher z
      afficher t
FIN
```

1. Identifier les paramètres réels et les paramètres formels.
   Les paramètres réels sont ceux du programme appelant : *x, y, z, t*
   Les paramètres formels sont ceux du sous programme et permettent de récupérer la valeur des paramètres réels : *a, b, c, d*

2. Qu'affiche ce programme en supposant que l'utilisateur saisisse 2 dans *x* et 3 dans *y* ? Modifier le code pour obtenir un résultat plus logique.
   Après l'appel à *ma_procedure*, *z* vaut 5 et *t* ne vaut rien de précis.
   En effet, *c* est un paramètre par adresse, donc les modifications opérées sur *c* dans le sous programme se répercutent sur le paramètre réel *z* correspondant dans le programme appelant. En revanche, *d* étant un paramètre par valeur, le paramètre réel correspondant *t* n'est pas affecté par les modifications. *t* conserve donc sa valeur d'avant l'appel au sous programme.

   En « transformant » *d* en paramètre par adresse, *ma_procedure* est un sous programme qui prend en entrée deux entiers et qui renvoie via *c* la somme de ces deux entiers et via *d* leur produit.

Exercice 2 – Le classique des classiques

Ecrire une procédure *échanger* qui échange la valeur de deux variables.
On donne ci-dessous un exemple d'appel de la procédure *échanger*.

```
ALGORITHME exercice_2

PROCEDURE échanger(x, y)
  PARAMETRES par ADRESSE : x, y entiers
  VARIABLE : sauv : entier
  DEBUT
      sauv ← x
      x ← y
      y ← sauv
  FIN

VARIABLES A, B : entiers
DEBUT
      ...
      A ← 1
      B ← 2
      échanger(A, B)
      { ici, A vaut 2 et B vaut 1}
      ...
FIN
```

Exercice 3 – Procédure et fonction

```
PROCEDURE mystere(x, y, z)
PARAMETRES par VALEUR : x, y : réels
PARAMETRES par ADRESSE: z : réel
DEBUT
      SI x > y ALORS
            z ← x
      SINON
            z ← y
      FINSI
FIN
```

1. Que fait cette procédure *mystere*.
   Elle retourne via le paramètre formel *z* le plus grand des deux entiers *x* et *y*.
2. Au lieu d'une procédure, peut-on utiliser une fonction ? Si oui, en donner l'algorithme.
   Une fonction est à privilégier dans ce cas de figure car il s'agit de retourner ici **une et une seule** valeur au programme appelant.

```
FONCTION mystere(x, y) : réel
PARAMETRES : x, y : réels
DEBUT
      SI x > y ALORS
            retourner x
      SINON
            retourner y
      FINSI
FIN
```

Exercice 4 – Cirant pour tomettes

Ecrire une fonction *surface* qui calcule la surface d'un rectangle de largeur *l* et longueur *L* (ces deux grandeurs étant passées en paramètre).
Utiliser cette fonction *surface* dans un programme principal qui demande à l'utilisateur les dimensions d'une pièce rectangulaire et qui indique le nombre de pots de shampoing cirant pour tomettes nécessaires pour cirer le sol de cette pièce. Un pot permet de traiter 10 m^2.

```
ALGORITHME cirant_tomettes

CONSTANTE :  (UnPot : entier) ← 10

FONCTION surface(L, l) : réel
 PARAMETRES : L, l : réels
 DEBUT
      retourner L × l
 FIN

VARIABLES :    long, larg, surf : réel
               NbPots : entier
DEBUT
      saisir long, larg
      surf ← surface(long, larg)
      SI surf mod UnPot = 0 ALORS          { si surf est un multiple de 10 }
            NbPots ← surf / UnPot          { alors on prend pile poil le nb de pots}
      SINON
            NbPots ← surf div UnPot + 1    { sinon on prend un pot de plus }
      FINSI
      afficher NbPots
FIN
```

Exercice 5 – « SNCF, c'est possible » - 1988

Ecrire une fonction qui calcule le nombre d'années entières écoulées entre deux dates exprimée sous la forme *jour-mois-année*. Utiliser une structure.

Utiliser cette fonction dans un algorithme qui demande à l'utilisateur sa date de naissance et qui indique si celui-ci peut bénéficier de la carte de réduction *Sénior+* de la SNCF.

Indication. La carte *Sénior+* est accessible dès soixante ans.

```
ALGORITHME SNCF

CONSTANTE : (AGE : entier) ← 60

TYPE : date = STRUCTURE
      jour : entier
      mois : entier
      année : entier
FINSTRUCTURE

{ DiffAnnée : retourne le nombre d'années entières entre deux dates quelconques
 d2 désigne la date la plus récente }
FONCTION DiffAnnée(d1, d2) : entier
 PARAMETRES : d1, d2 : date
 VARIABLE :     diff : entier
 DEBUT
      diff ← d2.année - d1.année;

      SI d1.mois > d2.mois OU [ d1.mois = d2.mois ET d1.jour > d2.jour ] ALORS
            diff ← diff-1
      FINSI

      retourner diff
 FIN

{ Algorithme principal }
VARIABLES :    d, aujourdhui : date

DEBUT
      saisir d.jour, d.mois, d.année
      aujourdhui ← DateDuJour( )

      SI DiffAnnée(d, aujourdhui) ≥ AGE ALORS
            afficher « Vous êtes éligible à la carte Sénior+ »
      SINON
            afficher « Vous n'êtes pas éligible à la carte Sénior+ »
      FINSI
FIN
```

NB : on suppose l'existence d'une fonction *DateDuJour* qui renvoie la date du jour.