Bases de programmation - TD 1 : Algorithmique - CORRECTION

IUT Aix-Marseille / DUT R&T 1ière année
J. Seinturier (http://www.seinturier.fr)

1. Déroulement d'un algorithme

Exercice 1.1 : Donner pour chacun des 3 algorithmes suivants leur nombre de variables et d'instructions.

```
algorithme alg1
données A, B : entier
début
  A ← 1
  B ← A + 3
  A ← 3
Fin
```

```
algorithme alg2
données A, B, C : entier
début
  A ← 3
  B ← 10
  C ← A + B
  B ← A + B
  A ← C
fin
```

```
algorithme alg3
données A, B : entier
début
  A ← 5
  B ← 2
  A ← B
  B ← A
fin
```

Variables :	2	3	2
Instructions :	3	5	4

Exercice 1.2 : Donner les valeurs des variables à la fin de l'exécution des 3 algorithmes précédents.

Alg1 : a = 3, B = 4. alg2 : A = 13, B = 13, C = 13. alg3 : A = 2, B = 2.

2. Ecriture d'algorithmes simples

Exercice 2.1 : Ecrire un algorithme permettant d'échanger les valeurs de deux variables A et B, et ce quel que soit leur contenu préalable. (vous pouvez utiliser autant de variables que vous le souhaitez.)

```
algorithme echange2
données
  a : entier
  b : entier
  c : entier
début
  c ← a
  a ← b
  b ← c
fin
```

Exercice 2.2 : Idem que l'exercice 2.1 mais sans utiliser d'autre variable que A et B.

```
algorithme echange2b
données
  a : entier
  b : entier
début
  a ← a + b
  b ← a - b
  a ← a - b
fin
```

Exercice 2.3 : Soit trois variables A, B et C. Ecrire un algorithme transférant à B la valeur de A, à C la valeur de B et à A la valeur de C (toujours quels que soient les contenus préalables de ces variables).

```
algorithme echange3
données
  a : entier
  b : entier
  c : entier
  tmp : entier
début
  tmp ← c
  c ← b
  b ← a
  a ← tmp
fin
```

Exercice 2.4 : Ecrire un algorithme permettant de calculer la valeur de la fonction $f(x) = x^2 - 8x + 7$ pour un x donné. Tester l'algorithme avec les valeurs de x suivantes : $-2, 10, 5, 3$

```
algorithme fonction3
entrées
  x : réel
données
  fx : réel
début
  fx ← x × x − 8 × x + 7
fin
```

Exercice 2.5 : Ecrire un algorithme calculant le Δ une équation du second degré de la forme $ax^2 + bx + c$. Pour rappel, $\Delta = b^2 - 4ac$. L'algorithme prendra comme entrée les variables a, b et c comme valeur des coefficients et aura comme sortie la variable delta contenant la valeur Δ.

```
algorithme delta
entrées
  a : réel
  b : réel
  c : réel
sortie
  delta : réel
début
  delta ← b × b − 4 × a × c
fin
```

3. Entrées et sorties

Exercice 3.1 : Ecrire un algorithme qui demande un nombre à l'utilisateur, puis qui calcule et affiche son carré.

```
algorithme carré
données
  n : réel
début
  n ← lire()
  n ← n × n
  ecrire(n)
fin
```

Exercice 3.2 : Ecrire un algorithme qui demande à l'utilisateur le prix HT (hors taxe) d'un article, le nombre d'articles vendus et le taux de TVA (Taxe sur la Valeur Ajoutée), et qui affiche le prix total TTC (Toutes Taxes Comprises) de tous les articles. Pour rappel, prix TTC = prix HT + prix HT × TVA, TVA étant un pourcentage. Tester l'algorithme pour l'achat de 6 articles à 3 euros avec une TVA de 5% puis pour l'achat de 5 articles à 8 € avec une TVA de 19.6%.

```
algorithme tva
données
  prix : réel
  nombre : réel
  taux : réel
  total : réel
début
  prix ← lire()
  nombre ← lire()
  taux ← lire()
  total ← (prix + prix × taux) × nombre
  ecrire(total)
fin
```

4. Blocs conditionnels

Exercice 4.1 : Ecrire un algorithme qui demande un nombre à l'utilisateur, et l'informe ensuite si ce nombre est strictement positif ou non.

```
algorithme prod-positif
données
  n : réel
début
  n ← lire()
  si n > 0 alors
    ecrire(''Positif strict'')
  sinon
    ecrire(''Non positif strict'')
  finsi
fin
```

Exercice 4.2 : Ecrire un algorithme qui prend deux nombres réels en entrée et qui l'informe l'utilisateur si leur produit est négatif ou positif (on considère *0* comme positif).

```
algorithme signe
données
  n : réel
  m : réel
début
  n ← lire()
  m ← lire()
  si (n ≥ 0 et m ≥ 0) ou (n < 0 et m < 0) alors
    ecrire(''Positif'')
  sinon
    ecrire(''Négatif'')
  finsi
fin
```

Exercice 4.3 : Ecrire un algorithme qui demande l'âge d'un enfant à l'utilisateur. Ensuite, il l'informe de sa catégorie : "Poussin" de 6 à 7 ans, "Pupille" de 8 à 9 ans, "Minime" de 10 à 11 ans, "Cadet" après 12 ans

```
algorithme classe_age
données
  age en Entier
début
  ecrire("Entrez l'âge de l'enfant : ")
  age ← lire()
  si age ≥ 12 alors
    écrire("Catégorie Cadet")
  sinon
    si age ≥ 10 alors
      écrire("Catégorie Minime")
    sinon
      si age ≥ 8 alors
        écrire("Catégorie Pupille")
      sinon
        si age >= 6 alors
          écrire("Catégorie Poussin")
        finsi
      finsi
    finsi
  finsi
fin
```

5. Algorithmes plus complexes

Exercice 5.1 : Ecrire un algorithme qui demande à l'utilisateur d'entrer une heure précise en demandant l'heure puis les minutes et affiche l'heure qu'il sera une minute plus tard. Par exemple, si l'utilisateur tape 21 puis 32, l'algorithme doit répondre *21 heure(s) 33*. Si l'utilisateur tape *23* puis *59*, la réponse doit alors être *0* heure *00*. On considère que les valeurs entrées par l'utilisateur sont correctes.

```
algorithme heure
données
  h : entier
  m : entier
début
  h ← lire()
  m ← lire()
  m ← m + 1
  si m = 60 alors
    m ← 0
    h ← h + 1

    si h = 24 alors
      h ← 0
    finsi

  finsi

  ecrire(h)
  ecrire('' heures '')
  ecrire(m)
fin
```

Exercice 5.2 : Un magasin de reprographie facture 0,10€ les dix premières photocopies, 0,09€ les vingt suivantes et 0,08€ au-delà. Ecrire un algorithme qui demande à l'utilisateur le nombre de photocopies effectuées et qui affiche le coût total des photocopies.

```
algorithme photocopies
donnees
  n : entier
  p : réel
début
  ecrire("Nombre de photocopies : ")
  n ← lire()

  si n ≤ 10 alors
    p ← n × 0.1
  sinon
    si n ≤ 30 alors
      p ← 10 × 0.1 + (n – 10) × 0.09
    sinon
      p ← 10 × 0.1 + 20 × 0.09 + (n – 30) × 0.08
    finsi
  finsi

  ecrire("Le prix total est: ")
  ecrire(p)
fin
```

Exercice 5.3 : Ecrire un algorithme qui affiche la ou les solutions d'une équation du second degré de la forme $ax^2 + bx + c$. Utiliser pour cela l'algorithme de calcul du Δ écrit durant l'exercice **2.5**. Pour rappel, si Δ est négatif, il n'existe pas de solution. Si Δ est nul, il existe une unique solution qui est $-b / 2a$. Si Δ est positif, il existe deux solutions qui sont $x1 = \frac{-b-\sqrt{\Delta}}{2a}$ et $x2 = \frac{-b+\sqrt{\Delta}}{2a}$.

```
algorithme second_degre
entrées
  a, b, c : réel
données
  d : réel
  sol1, sol2 : réel
début
  d ← delta(a, b, c)
  si d < 0 alors
    ecrire(''Pas de solution'')
  sinon
    si d = 0 alors
      sol1 ← -b / (2 × a)
      ecrire(''1 solution : '')
      ecrire(sol1)
    sinon
      sol1 ← (-b – sqrt(d)) / (2 × a)
      sol2 ← (-b + sqrt(d)) / (2 × a)
      ecrire(sol1)
      ecrire(sol2)
    finsi
  finsi
fin
```

Exercice 5.4 : Les habitants d'une ville paient l'impôt selon les règles suivantes :

- les hommes de plus de 20 ans paient l'impôt
- les femmes paient l'impôt si elles ont entre 18 et 35 ans
- les autres ne paient pas d'impôt

Ecrire un algorithme qui demande l'âge et le sexe d'un habitant et affiche si celui-ci est imposable. Tester l'algorithme avec une femme de 20 ans, un homme de 18 ans et un homme de 35 ans.

Nous utilisons dans cette correction le type **caractère** afin de stocker le sexe de l'habitant : 'M' pour masculin et 'F' pour féminin. Ne pas oublier les côtes ' lors de l'utilisation de caractères.

```
algorithme population
données
  sexe : caractère
  age : entier
  c1 : booléen
  c2 : booléen
début
  ecrire("Entrez le sexe (M/F) : ")
  sexe ← lire()
  ecrire("Entrez l'âge: ")
  age ← lire()
  c1 ← sexe = 'M' et age > 20
  c2 ← sexe = 'F' et (age > 18 et age < 35)
  si c1 ou c2 alors
    ecrire("Imposable")
  sinon
    ecrire("Non Imposable")
  finsi
fin
```

Exercice 5.5 : Ecrire un algorithme qui demande un numéro de jour, de mois et d'année à l'utilisateur et renvoie s'il s'agit ou non d'une date valide. Pour rappel, le mois de février compte 28 jours, sauf si l'année est bissextile, auquel cas il en compte 29. L'année est bissextile si elle est divisible par quatre, toutefois, les années divisibles par 100 ne sont pas bissextiles, mais les années divisibles par 400 le sont.

Cet algorithme accepte une solution très simple. Celle-ci consiste à dire qu'il y a quatre cas pour qu'une date soit valide : celui d'un jour compris entre 1 et 31 dans un mois à 31 jours, celui d'un jour compris entre 1 et 30 dans un mois à 30 jours, celui d'un jour compris entre 1 et 29 en février d'une année bissextile, et celui d'un jour de février compris entre 1 et 28.

Une date valide respecte alors les conditions suivantes :

- K1 : Le mois est il différent de février et le jour compris entre 1 et 31
- K2 : Le mois est il différent de février et le jour compris entre 1 et 30
- K3 : le mois est il février, l'année bissextile et le jour compris entre 1 et 29
- K4 : le mois est il février, l'année quelconque et le jour compris entre 1 et 28

La résolution de la condition K3 demande à vérifier si l'année est bissextile, ce qui peut être évalué grâce à un booléen B dont la valeur et le résultat de l'évaluation de la divisibilité de l'année

```
algorithme date
données
  j, m, a : entier
  B : booléen
  K1, K2, K3, K4 : booléen
début
  ecrire("Entrez le numéro du jour")
  j ← lire()
  ecrire("Entrez le numéro du mois")
  m ← lire()
  ecrire("Entrez l'année")
  a ← lire()
  B ← ((a % 4 = 0) et non ((a % 100) = 0) ou (a % 400 = 0)
  K1 ← (m=1 ou m=3 ou m=5 ou m=7 ou m=8 ou m=10 ou m=12) et (j ≥ 1 et j ≤ 31)
  K2 ← (m=4 ou m=6 ou m=9 ou m=11) et (j ≥ 1 et j ≤ 30)
  K3 ← m = 2 et B et j ≥ 1 et j ≤ 29
  K4 ← m = 2 et J ≥ 1 et J ≤ 28
  si K1 ou K2 ou K3 ou K4 alors
    ecrire("Date valide")
  sinon
    ecrire("Date non valide")
  finsi
fin
```

Exercice 5.6 : Une compagnie d'assurance automobile propose à ses clients quatre familles de tarifs identifiables par une couleur: bleu, vert, orange et rouge. Le tarif dépend de la situation du conducteur :

- moins de 25 ans et permis depuis moins de deux ans : tarif rouge, si toutefois il n'a jamais été responsable d'accident. Sinon, la compagnie refuse de l'assurer.
- moins de 25 ans et permis depuis plus de deux ans, ou de plus de 25 ans mais permis depuis moins de deux ans a : tarif orange s'il n'a jamais provoqué d'accident, tarif rouge pour un accident, sinon il est refusé.
- plus de 25 ans avec permis depuis plus de deux ans : tarif vert s'il n'est à l'origine d'aucun accident et du tarif orange pour un accident, du tarif rouge pour deux accidents, et refusé au-delà
- De plus, pour encourager la fidélité des clients acceptés, la compagnie propose un contrat de la couleur immédiatement la plus avantageuse s'il est entré dans la maison depuis plus de cinq ans. Ainsi, s'il satisfait à cette exigence, un client normalement "vert" devient "bleu", un client normalement "orange" devient "vert", et le "rouge" devient orange.

Ecrire l'algorithme permettant de saisir les données nécessaires (sans contrôle de saisie) et de traiter ce problème. Penser à bien formaliser le problème avant d'écrire un algorithme trop compliqué.

Afin d'écrire l'algorithme sans avoir de trop nombreuses conditions à gérer, il faut bien étudier l'énoncé et en dégager un barème de calcul permettant de quantifier la situation du conducteur. La couleur du tarif dépend alors uniquement de ce barème. La méthode de calcul est assez simple :

- au départ, p vaut 0 ;
- si l'assuré a + de 25 ans, p est incrémenté de 1 ;
- si l'assuré à un permis depuis + de 2 ans, p est incrémenté de 1
- p est ensuite additionné du nombre d'accident déclarés
- p est ensuite diminué de 1 si le conducteur est assuré depuis plus de 5 ans et si p n'est pas supérieur à 3 (la réduction ne peut s'appliquer en cas d'accidents trop nombreux)

```
algorithme assurance
données
  age : entier
  perm : entier
  acc : entier
  assur : entier
  c1, c2, c3 : booléen
  situ : caractère
  p : entier
début
  ecrire("Entrer l'âge: ")
  age ← lire()
  ecrire("Entrer le nombre d'années de permis: ")
  perm ← lire()
  ecrire("Entrer le nombre d'accidents: ")
  acc ← lire()
  ecrire("Entrer le nombre d'années d'assurance: ")
  assur ← lire()
  c1 ← age >= 25
  c2 ← perm >= 2
  c3 ← assur > 5
  p ← 0
  si non c1 alors
    p ← p + 1
  finsi
  si non c2 alors
    p ← p + 1
  finsi

  p ← p + acc

  si p < 3 et c3 alors
    p ← p - 1
  finsi
  si p = -1 alors
   situ ← 'B'
  sinon
    si p = 0 alors
      situ ← 'V'
    sinon
      si p = 1 alors
        situ ← 'O'
      sinon
        si p = 2 alors
          situ ← 'R'
        sinon
          situ ← "X"
        finsi
      finsi
    finsi
  finsi

  ecrire("Votre situation : ")
```

```
  ecrire(situ)
fin
```