

Correction TD algorithmique

Affectation

Exercice 1

Donner le résultat de l'exécution de l'algorithme suivant :

```
algo affect1b
  var a : entier
  var b : entier
Début
  b ← 5
  a ← b+1
  b ← 2
Fin
```

Il vaut faire passer la notion de variable et la notion de stockage mémoire. Une variable n'a donc pas d'historique et à un instant donné ne peut contenir qu'une valeur.

a	6
b	2

Exercice 2

Donner le résultat de l'exécution de l'algorithme suivant :

```
algo affect2
  var a : entier
  var b : entier
Début
  a ← 2
  a ← a+1
Fin
```

Une variable peut recevoir le résultat d'une fonction de sa propre valeur.
La variable b est déclarée mais non utilisée (instanciée), ce n'est pas bien.

a	3
b	?

Exercice 3

Donner le résultat de l'exécution de l'algorithme suivant :

```
algo affect3
  var a : entier
  var b : entier
Début
  b ← a+1
  a ← 2
Fin
```

La variable a est utilisée en partie droite d'une affectation sans avoir été instanciée, ce n'est pas bien.

a	2
---	---

b	?
---	---

Exercice 4

Donner le résultat de l'exécution de l'algorithme suivant :

algo affect4

var a : entier

var b : entier

Début

a+5 ← 3

Fin

L'expression a+5 ← 3 n'est pas syntaxiquement correcte. ce n'est pas bien.	
a	?
b	?

Exercice 5

Donner le résultat de l'exécution de l'algorithme suivant :

algo affect5

var a : entier

var b : entier

Début

a ← 5

b ← a+4

a ← a+1

b ← a-4

Fin

a	6
b	2

Exercice 6

Donner le résultat de l'exécution de l'algorithme suivant :

algo affect6

var a : entier

var b : entier

var c : entier

Début

a ← 3

b ← 10

c ← a+b

b ← a+b

a ← c

Fin

a	13
b	13
c	13

Exercice 7

Donner le résultat de l'exécution de l'algorithme suivant :

algo affect7

var a : entier

var b : entier

Début

a ← 5

b ← 7

a ← b

b ← a

Fin

a	7
b	7

Exercice 8

Donner le résultat de l'exécution de l'algorithme suivant :

algo affect8

var a : entier

var b : entier

Début

a ← 5

b ← 7

b ← a

a ← b

Fin

a	5
b	5

Exercice 9

Écrire un algorithme permettant d'inverser deux variables.

//Le coup des deux aquariums

algo Echangeab

var a : entier

var b : entier

var c : entier

Début

a ← 5

b ← 7

c ← a

a ← b

b ← c

Fin

Conditionnelles

Exercice 10

Écrire un algorithme qui lit deux valeurs entières et affiche le maximum des deux.

```
algo Max
  var a : entier
  var a : entier
Début
  a ← Lire()
  b ← Lire()
  Si (b < a) alors
    Ecrire("a est plus grand que b")
  Sinon
    Ecrire("b est plus grand que a")
  Fin Si
Fin
```

Exercice 11 (calcul de remise (1))

a la caisse d'un supermarché (bien connu à l'Isle d'abeau), nous bénéficions d'une remise de 1% sur le montant de nos achat lorsque celui-ci dépasse 300 euros. Écrire un algorithme qui après lecture du montant initialement du, affiche le montant à payer.

Remarques :

- maladroit d'avoir plus d'une instruction d'écriture du résultat.
- maladroit d'avoir une clause sinon
- inutile d'avoir recours à une autre variable.

```
algo Remise1
  var montant : réel
  const tauxRemise ← 0,01: réel
Début
  montant ← Lire ()
  Si (300 < montant) alors
    montant ← montant * (1-tauxRemise)
  Fin Si
  Ecrire("Vous devez " + montant)
Fin
```

Exercice 12 (calcul de remise (2))

Même exercice avec :

- 1% de remise pour un achat compris entre 300 et 750 euros
- 2% au delà de 750 euros

```
algo Remise2
  var montant : réel
  var tauxRemise : réel
Début
```

```
montant ← Lire ()
Si (750 <montant) alors
    tauxRemise ← 0,02
Sinon
    Si (300 < montant) alors
        tauxRemise ← 0,01
    Sinon
        tauxRemise ← 0
    Fin Si
Fin Si
montant ← montant * (1-tauxRemise)
Ecrire("Vous devez " + montant)
Fin
```

Exercice 13

Lire trois valeurs entières a, b et c. afficher le maximum des trois.

```
algo Max1
    var a : entier
    var b : entier
    var c : entier
    var max : entier
Début
    a ← Lire ()
    b ← Lire ()
    c ← Lire ()
    Si (b ≤ a et c ≤ a) alors
        Max ← a
    Sinon
        Si (a ≤ b et c ≤ b) alors
            Max ← b
        Sinon
            Si (a ≤ c et b ≤ c) alors
                Max ← c
            FinSi
        Fin Si
    Fin Si
    Ecrire("Le max est " + max)
Fin

algo Max2
    var a : entier
    var b : entier
    var c : entier
    var max : entier
Début
    a ← Lire ()
    b ← Lire ()
```

```
c ← Lire ()
Si (b ≤ a et c ≤ a) alors
    Max ← a
Sinon
    Si (c ≤ b) alors
        Max ← b
    Sinon
        Max ← c
    Fin Si
Fin Si
Ecrire("Le max est " + max)
Fin

algo Max3
    var a : entier
    var b : entier
    var c : entier
    var max : entier
Début
    a ← Lire ()
    b ← Lire ()
    c ← Lire ()
    max ← a
    Si (max ≤ b) alors
        max ← b
    Fin Si
    Si (max ≤ c) alors
        max ← c
    Fin Si
    Ecrire("Le max est " + max)
Fin
```

Exercice 14 (calcul d'une facture d'électricité)

Trouver le prix à payer sachant qu'une facture inclut une somme de 4 euros de frais fixes et que s'ajoute un prix en fonction de la consommation :

- 0,1 euro/kWH pour les 100 premiers kilowatts heures
- 0,07 euro/kWH pour les 150 suivants
- 0,04 euro/kWH au-delà

```
algo FactureElectricite
    var consommation : réel
    var prix : réel
    const fraisFixe ← 4 : réel
    const prixTranche1 ← 0,1 :réel
    const prixTranche2 ← 0,07: réel
    const prixTranche3 ← 0,04 : réel
    const limTranche12 ← 100 : entier
    const limTranche23 ← 150 : entier
```

Début

consommation ← Lire()

Si (consommation ≤ limTranche12) **alors**

 prix ← fraisFixe + consommation * prixTranche1

Sinon

Si (consommation ≤ limTranche12 + limTranche23) **alors**

 prix ← fraisFixe + limTranche12 * prixTranche1 + (consommation – limTranche12) * prixTranche2

Sinon

 prix ← fraisFixe + limTranche12 * prixTranche1 + limTranche23 * prixTranche2 + (consommation – (limTranche12 + limTranche23)) * prixTranche3

Fin Si

Fin Si

 Ecrire("Le prix à payer est de " + prix)

Fin

Exercice 15 (Estimation du prix de revient d'un véhicule)

Il existe un barème pour l'évaluation du prix de revient kilométrique des véhicules.

Écrire un algorithme effectuant le calcul de ce prix en fonction de nb, nombre de kilomètres parcourus.

Règles :

Nb de km \ puissance fiscale	5cV	6cV
jusqu'à 5000	$n1 * 0,43 (= p1)$	$n * 0,47$
de 5001 à 20000	$(n2 * 0,23) + p1 (=p2)$	$(n * 0,27) + 1000$
au delà de 20000	$(n3 * 0,28) + p2$	$n * 0,32$

où n est le nombre total de kilomètres parcourus, n1 le nombre de kilomètres parcourus entre 0 et 5000, n2 le nombre de kilomètres parcourus entre 5001 et 20000 et n3 le nombre de kilomètre parcourus au delà de 20000. Exemple : si j'ai parcouru 8500 km, n=8500, n1=5000, n2=3500 et n3=0.

algo PrixRevient

var puissance : entier

var nbKm : entier

var prix : réel

Début

 puissance ← Lire()

 nbKm ← Lire()

Si (puissance =5) **alors**

 //cas 5 cV

Si (nbKm ≤ 5000) **alors**

 prix ← nbKm * 0,43

Sinon

Si (nbKm ≤ 20 000) **alors**

 prix ← 5000*0,43 + (nbKm – 5000) * 0,23

Sinon

 prix ← 5000*0,43 + (15000)*0,23+(nbKm-20000)*0,28

Fin Si

```

    Fin Si
Sinon
//cas 6 cV
    Si (nbKm ≤ 5000) alors
        prix ← nbKm * 0,47
    Sinon
        Si (nbKm ≤ 20 000) alors
            prix ← nbKm * 0,27 + 1000
        Sinon
            prix ← nbKm * 0,32
        Fin Si
    Fin Si
Fin Si
Ecrire(" Le prix est de : " + prix )
Fin
```

Exercice 16 (Le monnayeur)

Un distributeur qui rend de la monnaie doit rendre en priorité les pièces les plus grosses. En supposant que la machine rend des jetons de 5, 2 et 1 unités et qu'elle doit vous rendre nb unités, écrire un algorithme qui simule le rendu. On suppose que la caisse de départ de la machine est illimitée. I.e. il y a toujours assez de jetons en caisse pour le rendu.

```

algo Monnayeur
    var nb : entier
    var nb5, nb2, nb1 : entier
Début
    nb ← Lire()
    nb5 ← nb/5
    nb ← nb % 5 //ou nb ← nb - (nb/5)*5
    nb2 ← nb /2
    nb ← nb % 2 //ou nb ← nb - (nb/2)*2
    nb1 ← nb
    Ecrire(" Il faut rendre")
    Ecrire(nb5+"jetons de 5")
    Ecrire(nb2+"jetons de 2")
    Ecrire(nb1+"jetons de 1")
Fin
```

Exercice 17

Même exercice avec une caisse de départ limitée

```

algo MonnayeurcaisseLimite
    var nb : entier
    var nb5, nb2, nb1 : entier
    var nb5Dispo, nb2Dispo, nb1Dispo : entier
Début
    nb ← Lire()
    nb5Dispo ← Lire()
    nb2Dispo ← Lire()
```



```
nb1Dispo ← Lire()

nb5 ← nb5/5
Si (nb5 > nb5Dispo) alors
    nb5 ← nb5Dispo
Fin Si
nb ← nb – nb5*5
nb2 ← nb /2
Si (nb2 > nb2Dispo) alors
    nb2 ← nb2Dispo
Fin Si
nb ← nb- nb2*2
Si (nb > nb1Dispo) alors
    Ecrire(" Impossible")
Sinon
    nb1 ← nb
    //nb5Dispo ← nb5Dispo – nb5
    //nb2Dispo ← nb2Dispo – nb2
    //nb1Dispo ← nb1Dispo – nb1
    Ecrire(" Il faut rendre")
    Ecrire(nb5+"jetons de 5")
    Ecrire(nb2+"jetons de 2")
    Ecrire(nb1+"jetons de 1")
Fin Si
Fin
```

boucle – structure itérative

Exercice 18

Exécuter l'algorithme suivant.

algo boucle2

var a : *entier*

Début

a ← 5

Tant que (a > 0) **Faire**

Ecrire(a)

a ← a - 2

Fin Tant que

Fin

Sorties :

5

3

1

Variables :

a -1

Exercice 19

Lire un caractère et l'afficher jusqu'à ce que l'on saisisse 'y'. Réaliser deux versions cet algorithme :

1. une version avec affichage du 'y' final
2. une version sans affichage du 'y' final

algo LireaffavecY

var car : *caractère*

Début

faire

car ← Lire()

Ecrire(car)

Tant que (car ≠ 'y')

Fin

algo LireaffavecY

var car : *caractère*

Début

car ← Lire()

Tant que (car ≠ 'y') **Faire**

Ecrire(car)

Lire(car)

Fin Tant que

Fin

Exercice 20

Écrire un algorithme qui affiche les n premiers entiers (de 1 à n).

```
algo NPremier  
  var n,i : entier  
Début  
  n ← Lire()  
  i ← 1  
  Tant que (i ≤ n) Faire  
    Ecrire(i)  
    i ← i + 1  
  Fin Tant que  
Fin  
  
algo NPremier2  
  var n: entier  
Début  
  n ← Lire()  
  Tant que (n < 0) Faire  
    Ecrire(n)  
    n ← n - 1  
  Fin Tant que  
Fin
```

Exercice 21

Écrire un algorithme qui affiche la somme des n premiers entiers

```
algo SommePremiers  
  var n, i, somme : entier  
Début  
  n ← Lire()  
  somme ← 0  
  i ← 1  
  Tant que (i ≤ n) Faire  
    Somme ← somme + i  
    i ← i + 1  
  Fin Tant que  
  Ecrire(i)  
Fin
```

```
algo SommePremiersv2  
  var n, i, somme : entier  
Début  
  n ← Lire()  
  somme ← 0  
  Pour i de 1 à n pas de 1 faire  
    Somme ← somme + i  
  Fin Pour  
  Ecrire(i)
```

Fin

Exercice 22 (Remboursement d'emprunt)

calculer le nombre d'années nécessaires au remboursement d'un emprunt à taux d'intérêt fixe et dont le remboursement annuel est fixe également. (attention, le remboursement de la première année doit être strictement supérieur à l'intérêt payé la première année)

```
algo Remboursement
  var nbannee : entier
  var emprunt : réel
  var taux : réel
  var remboursementannuel : réel

Début
  emprunt ← Lire()
  taux ← Lire()
  remboursementannuel ← Lire()
  Si (remboursementannuel ≤ emprunt * taux) alors
    Ecrire("Impossible")
  Sinon
    nbannee ← 0
    Tant que (0 < emprunt) faire
      nbannee ← nbannee + 1
      emprunt ← emprunt * (1+taux) - remboursementannuel
    Fin Tant que
    Ecrire(nbannee)
  Fin Si

Fin
```

Exercice 23 (codage)

coder un nombre nb écrit en base 10 en base base (base < 10). Rappel : il faut faire les divisions successives de nb par base jusqu'à obtenir un quotient nul.

Nb : on formate le résultat par une concaténation de chaînes.

```
algo codage
  var nb, base : entier
  var rep : chaîne

Début
  nb ← Lire()
  base ← Lire()
  rep ← ""
  Faire
    rep ← nb % base + rep // concaténation de chaînes
    nb ← nb /base
  Tant que (nb≠0)
  Ecrire(rep)

Fin
```

Exercice 24

Écrire un algorithme qui fasse deviner un nombre entier aTrouver en donnant des indications (trop grand, trop petit) avec nbEssai autorisé. Il faut obtenir un affichage final Gagné ! ou Perdu !.

```
algo Devine
  var aTrouver : entier
  var gagne : booléen
Début
  aTrouver ← Lire()
  nbEssai ← Lire()
  gagne ← faux
  Tant que ((1 ≤ nbEssai) et (Non gagne)) Faire
    essai ← Lire()
    Si (essai=aTrouver) alors
      gagne ← vrai
    Sinon
      Si (aTrouver < essai) alors
        Ecrire("Trop grand")
      Sinon
        Ecrire("Trop petit")
      Fin Si
    Fin Si
    nbEssai ← nbEssai -1
  Fin Tant que
  Si (gagne) alors
    Ecrire("Gagné !")
  Sinon
    Ecrire("Perdu !")
  Fin Si
Fin
```

Exercice 25 (Décomposition en facteurs premiers)

Décomposer un nombre en nombres premiers. Essayer les divisions du nombre par les tous les entiers (à partir de 2) et faire afficher simplement les différents diviseurs. N.b.

On effectue les divisions du nombre par les différents entiers, qu'ils soient premiers ou no, de toute façon, un nombre qui n'est pas premier ne pourrait diviser car tous ses diviseurs (plus petit que lui) auraient précédemment divisé le nombre.

```
algo Décomposition1
  var nombre : entier
  var d: entier
Début
  nombre ← Lire()
  d ← 2
  Tant que (nb>1) Faire
    Si (nb % d = 0) alors
      Ecrire(d)
      nb ← nb/d
    Sinon
```

```
                d ← d + 1
            Fin Si
        Fin Tant que
Fin
```

```
algo Décomposition2
    var nombre : entier
    var d: entier
Début
    Lire(nombre)
    d ← 2
    Tant que (nb>1) Faire
        Tant que (nb % d = 0) Faire
            Ecrire(d)
            nb ← nb/d
        Fin Tant que
        d ← d + 1
    Fin Tant que
Fin
```

Exercice 26 (Décomposition en facteurs premiers (subsidaire))

Même exercice mais avec affichage des puissances.

```
algo Décomposition3
    var nombre : entier
    var d: entier
    var cpt : entier
Début
    nombre ← Lire()
    d ← 2
    cpt ← 0
    Tant que (nb>1) Faire
        Tant que (nb % d = 0) Faire
            cpt ← cpt + 1
            nb ← nb/d
        Fin Tant que
        Si (0 < cpt) alors
            Ecrire(d+"ppuissance " + cpt)
            cpt ← 0
        Fin Si
        d ← d + 1
    Fin Tant que
Fin
```

Exercice 27

Écrire un algorithme qui affiche la somme des n premiers entiers

```
algo SommeNPremierPour
    var n : entier
```

```
    var somme : entier
    var i : entier
Début
    n ← Lire()
    Somme ← 0
    Pour i de 1 à n Faire
        Somme ← somme + i
    Fin pour
    Ecrire("La somme est: " + somme)
Fin
```

Exercice 28 (Placement d'argent)

Si l'on place somme euros au 1 janvier de l'année anDepot à taux% (en accumulant les intérêts), quelle va être la somme présente sur le compte le 1 janvier de de l'année anRetrait ?

```
algo Placement
    var anDepot : entier
    var anRetrait : entier
    var taux : réel
    var somme : réel
    var i : entier
Début
    anDepot ← Lire()
    anRetrait ← Lire()
    somme ← Lire()
    Pour i de anDepot + 1 à anRetrait pas de 1 Faire
        Somme ← somme * (1+taux)
    Fin pour
    Ecrire("Vous avez sur votre compte : " + somme)
Fin
```