Exercices du chapitre 2

Exercices

Corrigés

01-*-Chercher l'erreur

Relevez les erreurs dans l'algorithme suivant en les expliquant.

```
Algorithme AlgoACorriger

/* Déclarations */
Constantes
/* identificateur                    rôle */
   PI               = 3.14           /* le nombre PI */
Variables
/* identificateur    : type           rôle */
   m                 : entier
   n                 : réel
   p                 : réel
   q                 : réel
   c                 : caractère
   d                 : caractère
   b1                : booléen
   b2                : booléen
   1x                : caractère

/* Instructions */
Début
  m ← 7
  p ← n + p
  c ← 'u'
  x ← 2.5
  b1 ← c != 'r'
  b2 ← (m == 7) OU b1
  n ← m * PI
  m * 3 ← m + 5
  p = 7.0
  PI ← 3.14159
  q ← 3m
Fin
```

02-*-Simulation instructions élémentaires

Réaliser deux simulations de l'algorithme suivant avec des données différentes :

```
Algorithme Corde

/* Déclarations */
Variables
/* identificateur    : type           rôle */
   perimetre         : entier         /* périmètre de la poulie */
   lgCorde           : entier         /* longueur de la corde */
   nbTours           : entier         /* nombre de tours */
   lgReste           : entier         */ longueur restante */
Début
  /* entrée des données */
1 écrire("Quel est en cm le périmètre de la poulie (un entier) ")
```

```
2 lire(perimetre)
3 écrire("Quel est en cm la longueur de la corde (un entier) ")
4 lire(lgCorde)
  /* calculs */
5 nbTour ← lgCorde div perimetre
6 lgReste  ← lgCorde mod perimetre
  /* affichage des résultats */
7 écrire ("on peut faire ", nbTour, "tours")
8 écrire("il reste ", lgreste, " cm non enroulés")
Fin
```

Attention ! : les numéros de ligne ne font pas partie de l'algorithme. Ils ne sont présents que pour faciliter la simulation.

ligne	perimetre	lgCorde	nbTours	lg_reste	écran
1					
2					
3					
4					
5					
6					
7					
8					
1					
2					
3					
4					
5					
6					
7					
8					

03-*-Simulation instructions élémentaires

Ecrire les déclarations et simuler les algorithmes suivants (numéroter les lignes et construire les tableaux)

```
Algorithme : Algo1
Début
  prix ← 15
  taux ← 10.5
  remise ← (prix * taux)
  remise ← remise / 100
  prix ← prix - remise
  écrire(prix)
Fin
```

```
Algorithme : Algo2
Début
  a ← 10
  b ← 70
  q1 ← (a + b) / 5
  q2 ← (a + b) div 5
  r2 ← (a + b) mod 5
  écrire(q1, q2, r2)
  q1 ← (a + b) / 3
  q2 ← (a + b) div 3
  r2 ← (a + b) mod 3
  écrire(q1, q2, r2)
Fin
```

04-*-Euro

Ecrire un algorithme qui demande une somme exprimée en euros puis calcule et affiche son équivalent en francs français.

05-**-Echange et permutation circulaire

1)

```
Algorithme EchangeV1

/* Déclarations */
Variables
   var1              : entier
   var2              : entier

Début
  lire(var1, var2)
  écrire(var1, var2)  /*  les  valeurs  des  variables  apparaissent
                      dans l'ordre de leur saisie */
  ...
  écrire(var1, var2)  /*  ici  elles  apparaissent  dans  l'ordre
                      inverse de leur saisie */
Fin
```

Ecrire la séquence d'instructions qui **échange le contenu** des deux variables. Le but de cet exercice est d'échanger le contenu de deux variables et non pas de les intervertir à l'affichage. L'affichage des variables avant et après l'échange sert seulement à vérifier que l'échange a bien eu lieu.

2)

```
Algorithme EchangeV2

/* Déclarations */
Variables
   var1              : réel
   var2              : réel
   var3              : réel

Début
  lire(var1, var2, var3)
  écrire(var1, var2, var3)   /*   les   valeurs   des   variables
                        apparaissent dans l'ordre de leur saisie */
  ...
  écrire(var1, var2, var3)  /*ici  elles  apparaissent  dans  l'ordre
                            inverse de leur saisie */
Fin
```

Ecrire la séquence d'instructions qui effectue une **permutation circulaire à droite du contenu** des trois variables.
Si le premier affichage donne “ `3,5  6,2  10,4` ”, le deuxième devra donner “ `10,4  3,5  6,2` ”

06-***-Position d'un jeton sur un damier

Sur un damier rectangulaire de n lignes et p colonnes (n et p connus) on dispose, ligne après ligne, (n * p) jetons numérotés de 1 à n * p.
Ecrire un algorithme qui demande la valeur du jeton et renvoie le numéro de ligne et le numéro de colonne de la case où il se

	1	2	**3**	4	5	6
1	1	2	3	4	5	6
2	7	8	9	10	11	12
3	13	14	15	16	17	18
4	19	20	**21**	22	23	24
5	25	26	27	28	29	30

CORRIGES

01-*-Chercher l'erreur

```
Algorithme AlgoACorriger

/* Déclarations */
Constantes
/* identificateur    : type          rôle */
   PI                = 3.14          /* le nombre PI */
Variables
/* identificateur    : type          rôle */
   m                 : entier
   n                 : réel
   p                 : réel
   q                 : réel
   c                 : caractère
   d                 : caractère
   b1                : booléen
   b2                : booléen
   1x                : caractère     /* 1x n'est pas un
                                        identificateur correct : ne
                                        doit pas commencer par un
                                        chiffre */

/* Instructions */
Début
  m ← 7
  p ← n + p              /* n n'est pas initialisé */
  c ← 'u'
  x ← 2.5                /* x n'est pas déclaré */
  b1 ← c != 'r'
  b2 ← (m == 7) OU b1
  n ← m * PI
  m * 3 ← m + 5          /* une variable doit être en partie
                            gauche d'une affectation */
  p = 7.0
  PI ← 3.14159           /* pas d'affectation sur PI qui est une
                            constante */
  q ← 3m                 /* 3m est un identificateur ni valable,
                            ni connu. Si cela doit correspondre à un
                            produit, il faut utiliser l'opérateur * :
                            3 * m */
Fin
```

02-*-Simulation instructions élémentaires

```
Algorithme Corde

/* Déclarations */
Variables
/* identificateur    : type          rôle */
   perimetre         : entier        /* périmètre de la poulie */
   lgCorde           : entier        /* longueur de la corde */
   nbTours           : entier        /* nombre de tours */
   lgReste           : entier        */ longueur restante */
Début
  /* entrée des données */
1 écrire("Quel est en cm le périmètre de la poulie (un entier) ?")
2 lire(perimetre)
3 écrire("Quel est en cm la longueur de la corde (un entier) ?")
4 lire(lgCorde)
  /* calculs */
5 nbTour ← lgCorde div perimetre
6 lgReste  ← lgCorde mod perimetre
  /* affichage des résultats */
7 écrire ("on peut faire ", nbTour, "tours")
8 écrire("il reste ", lgreste, " cm non enroulés")
Fin
```

Solution

Attention ! : les numéros de ligne ne font pas partie de l'algorithme. Ils ne sont présents que pour faciliter la simulation.

ligne	perimetre	lgCorde	nbTours	lg_reste	écran
1					Quel est en cm le périmètre de la poulie (un entier) ?
2	***10***				
3					Quel est en cm la longueur de la corde(un entier) ? (*)
4		***94***			
5			***9***		
6				***4***	
7					on peut faire 9 tours
8					il reste 4 cm non enroulés
					Simulation avec d'autres valeurs
1					Quel est en cm le périmètre de la poulie (un entier) ?
2	***13***				
3					Quel est en cm la longueur de la corde(un entier) ? (*)
4		***143***			
5			***11***		
6				***0***	
7					on peut faire 11 tours
8					il reste 0 cm non enroulés

(*) : la faute d'orthographe laissée dans l'énoncé n'a pas été corrigée dans la simulation pour bien montrer qu'un ordinateur exécute ce que le programmeur lui demande d'exécuter !

03-*-Simulation instructions élémentaires

```
Algorithme : Algo1
Début
  1 prix ← 15
  2 taux ← 10.5
  3 remise ← (prix * taux)
  4 remise ← remise / 100
  5 prix ← prix - remise
  6 écrire(prix)
Fin
```

```
Algorithme : Algo2
Début
  1 a ← 10
  2 b ← 70
  3 q1 ← (a + b) / 5
  4 q2 ← (a + b) div 5
  5 r2 ← (a + b) mod 5
  6 écrire(q1, q2, r2)
  7 q1 ← (a + b) / 3
  8 q2 ← (a + b) div 3
  9 r2 ← (a + b) mod 3
  10 écrire(q1, q2, r2)
Fin
```

Solution

Il faut numéroter les lignes pour les repérer dans les tableaux de simulation

```
Algorithme Algo1
  /* Déclarations */
Variables
  prix      : réel
  taux      : réel
  remise    : réel
```

	prix	taux	remise	écran
1	15			
2		10.5		
3			157.5	
4			1.575	
5	13.425			
6				13.425

```
Algorithme Algo2
  /* Déclarations */
Variables
  a, b, q2, r2  : entier
  q1            : réel
```

	a	b	q1	q2	r2	écran
1	10					
2		70				
3			16			
4				16		
5					0	
6						16 16 0
7			26.666			
8				26		
9					2	
10						26.666 26 2

04-*-Euro

```
Algorithme Euro
  /* Déclarations */
Constantes
  UN_EURO          = 6.55957 /* valeur de l'euro en francs */
Variables
  sEuro            : réel              /* somme en euros */
  sFranc           : réel              /* somme en francs */
  /* Instructions */
Début
  écrire("somme en euros ")            /* entrées des données */
  lire(sEuro)
  sFranc ← sEuro * UN_EURO             /* traitement */
  écrire ("valeur en francs : ", sFranc) /* sortie des résultats*/
Fin
```

05-**-Echange et permutation circulaire

1)

```
Algorithme EchangeV1

/* Déclarations */
Variables
   var1              : entier
   var2              : entier

Début
  lire(var1, var2)
  écrire(var1, var2)  /*  les  valeurs  des  variables  apparaissent
                          dans l'ordre de leur saisie */
  ...
  écrire(var1, var2)  /*  ici  elles  apparaissent  dans  l'ordre
                          inverse de leur saisie */
Fin
```

2)

```
Algorithme EchangeV2

/* Déclarations */
Variables
   var1              : réel
   var2              : réel
   var3              : réel

Début
  lire(var1, var2, var3)
  écrire(var1, var2, var3)  /*    les    valeurs    des    variables
                        apparaissent dans l'ordre de leur saisie */
  ...
  écrire(var1, var2, var3)  /* ici elles apparaissent dans l'ordre
                               inverse de leur saisie */
Fin
```

1) il faut déclarer une variable intermédiaire , par exemple, aux de type entier.

```
aux ← var1
var1 ← var2
var2 ← aux
```

2) il faut déclarer une variable intermédiaire , par exemple, aux de type réel.

```
aux ← var1
var1 ← var3
var3 ← var2
var2 ← aux
```

D'autres séquences sont possibles.

06-***-Position d'un jeton sur un damier

```
Algorithme Jeton
  /* Déclarations */
Variables
  valeur    : entier          /* valeur du jeton */
  n         : entier          /* nombre de lignes du damier */
  p         : entier          /* nombre de colonnes du damier */
  lig       : entier          /* numéro de lignes du jeton */
  col       : entier          /* numéro de colonnes du jeton */
  /* Instructions */
Début
  écrire("nombre de lignes")   /* entrées des données */
  lire(n)
  écrire("nombre de colonnes")
  lire(p)
  écrire("valeur du jeton")
  lire(valeur)               /* supposée appartenir au damier */
  col ←  1 + (valeur – 1) mod p        /* traitement */
  lig ← 1 + (valeur – 1) div p
  écrire(lig, col)           /* sortie des résultats */
Fin
```

Autre solution : on peut remplacer les deux affectations du traitement par l'instruction conditionnelle :

```
si valeur mod P = 0 alors    /* on est sur la dernière colonne */
  col ← p
  lig ← valeur div p
sinon
  col ← valeur mod p
  lig ← 1 + valeur div p
finsi
```