Exercices du chapitre 5

Sommaire

01-*-Mécanisme de l'instruction répétitive tantque

Simuler l'exécution de chacun des algorithmes suivants :

```
Algorithme Tanque1
Variables
  nb : entier
Début
  nb ← 10
  tantque nb < 40 faire
    écrire(nb)
    nb ← nb + 10
  fintantque
  écrire ('le nombre
      vaut ', nb)
Fin
```

```
Algorithme Tanque2
Variables
  nb : entier
Début
  nb ← 10
  tantque nb > 40 faire
    écrire(nb)
    nb ← nb + 10
  fintantque
  écrire ('le nombre
      vaut ', nb)
Fin
```

```
Algorithme Tanque3
Variables
  x : entier
Début
  lire(x)
  tantque x <= 3 faire
    écrire('x')
    x ← x + 1
  fintantque
  écrire (x)
Fin
```

ligne	nb	Ecran
1		

ligne	nb	écran
1		

ligne	x	écran
1		

02-*-Comptage d'un référendum (oui/non)

Ecrire un algorithme qui pilote le dépouillement d'un référendum dans lequel seuls les *OUI* et les *NON* sont possibles. L'assesseur saisit un 'O' pour chaque bulletin *OUI*, un 'N' pour chaque bulletin *NON*. Il interrompt la saisie par tout autre caractère.
L'algorithme affiche le nombre de votants, de *OUI*, de *NON*, ainsi que les pourcentages correspondants.

03-*-Comptage d'un référendum (oui/non/blancs/nul)

Reprendre l'algorithme de l'exercice précédent. Cette fois, il peut y avoir des bulletins blancs et des bulletins nuls. La saisie s'arrête par un caractère particulier, par exemple 'X' (qui ne doit pas être compté comme nul).

04-*-Chargement d'un camion

Ecrire un algorithme qui pilote le chargement d'un camion. Le camion est caractérisé par une capacité, masse qu'il peut transporter sans être en surcharge, qui sera donnée par l'utilisateur avant le début du chargement.
Puis des paquets arrivent pour être chargés dans le camion. Chaque paquet est caractérisé par sa masse entrée par l'utilisateur.
Si un paquet peut être chargé sans mettre le camion en surcharge, l'algorithme donne l'ordre de chargement. Le chargement doit s'arrêter avant le premier paquet qui ferait dépasser la capacité du camion.
Lorsque le chargement est terminé, l'algorithme doit afficher :

- le nombre de paquets chargés dans le camion
- la masse totale des paquets chargés dans le camion.

On suppose que l'utilisateur donne des valeurs positives pour la capacité du camion et pour la masse de chaque paquet.

05-*-Mécanisme de l'instruction répétitive pour

Simuler l'exécution de l'algorithme ci-dessous, puis écrire un algorithme équivalent utilisant une répétitive tant que

```
Algorithme Pour1
Variables
  nb : entier           /* nombre courant */
  i : entier            /* variable de boucle */
Début
  nb ← 10
  pour i de 1 à 4 faire
    écrire(nb)
    nb ← nb + 5
  finPour
  écrire("nombre = ", nb)
Fin
```

06-*-Moyenne

Ecrire un algorithme qui demande à l'utilisateur une suite de valeurs positives, et qui, sans mémoriser ces valeurs, calcule après la fin de saisie leur moyenne.
On écrira deux versions de cet algorithme :

Version1 : On demande à l'utilisateur, avant la saisie de la suite de nombres, combien de nombres va comporter la suite.

Version 2 : On ne pose pas de question préalable, mais l'utilisateur indique qu'il a terminé la saisie en entrant un nombre spécial ne pouvant pas faire partie de la suite, par exemple le nombre -1.

07-*-Affichage n°1

Affichage de n 'a', n étant saisi au préalable. Ecrire l'algorithme correspondant.

Exemple d'exécution
n : 16
aaaaaaaaaaaaaaaa

Jusqu'ici, dans les exercices abordés, la forme exacte de l'affichage n'a pas eu d'importance. On supposera que l'instruction "écrire" affiche un texte à l'écran sans passer à la ligne. Pour Passer à la ligne, il suffira d'afficher un caractère spécial, une constante nommée CRLF (pour Carriage Return Line Feed).
Exemple :
écrire("ceci est", " du texte", CRLF) affiche ceci est du texte avec passage à la ligne
écrire(CRLF) passe à la ligne suivante.

08-**-Minimum, maximum et leur rang

Ecrire un algorithme qui étant donné une suite de nombres entiers demandés à l'utilisateur, annonce, à l'issue de la saisie, le minimum (ou le maximum, ou les deux) de ces nombres et son (leur) rang(s) dans la suite. On supposera que les nombres saisis par l'utilisateur sont compris entre deux bornes fixées dans l'algorithme.
Dans le premier cas : minimum et maximum sont fonction des bornes
Dans le second cas : minimum et maximum sont fonction du premier nombre saisi

09-**-Saisie de notes

Ecrire un algorithme qui permet la saisie de plusieurs notes comprises entre 0 et 20, en contrôlant leur validité. Après chaque saisie valide, l'algorithme demande : *Encore une note (o/n)?*
Si une note n'est pas valide, le message suivant sera affiché : *Erreur, la note doit être comprise entre 0 et 20.*

10-**-Affichage n°2

Affichage de p lignes de n 'a', n et p étant saisis au préalable. Ecrire l'algorithme correspondant.

aaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa

11-**-Affichage n°3

Affichage de 4 'a' par ligne, sauf éventuellement sur la dernière ligne. Le nombre total de 'a' est saisi au préalable. Ecrire l'algorithme correspondant.

Si n=10	Si n=8	Si n=3
aaaa aaaa aa	aaaa aaaa	aaa

12-**-Affichage n°4

Affichage n°4 : vous ne devez utilisez que la structure pour.

Si on a saisi 4

```
aaaa
aaa
aa
a
```

Si on a saisi 5

```
aaaaa
aaaa
aaa
aa
a
```

13-**-Affichage n°5

Si on a saisi 10

```
1 2 3 4 5 6 7 8 9 10
```

Si on a saisi 4

```
1 2 3 4
```

14-**-Affichage n°6

Si on a saisi 5

```
2 4 6 8 10
```

Si on a saisi 8

```
2 4 6 8 10 12 14 16
```

15-**-Affichage n°7

On ne gérera pas l'alignement des nombres. Il suffira de les espacer de 4 caractères.

Si on a saisi 10

```
1    2    3    4
5    6    7    8
9    10
```

Si on a saisi 8

```
1    2    3    4
5    6    7    8
```

Si on a saisi 3

```
1    2    3
```

16-***-Affichage n°8

Si on a saisi 4

```
1    2    3    4
5    6    7
8    9
10
```

Si on a saisi 3

```
1    2    3
4    5
6
```

17-***-Affichage n°9

Si on a saisi 10

```
aa
aa
aa
aa
aa
```

Si on a saisi 13

```
aaa
aaa
aaa
aa
aa
```

Si on a saisi 5

```
a
a
a
a
a
```

Si on a saisi 3

```
a
a
a
```

18-***-Recherche de répétitions 1

On entre au clavier une série de chiffres ; la fin des entrées est marquée par la saisie du chiffre 0. On veut afficher les répétitions :
Un chiffre répété n fois (saisi n + 1 fois consécutivement) sera affiché n fois. Vous pourrez utiliser l'opérateur de concaténation.
Pour la suite 3 4 4 8 5 5 5 6 9 1 1 1 1 7 3 3 5 5 0, on obtient : 4 5 5 1 1 1 3 5

19-***-Recherche de répétitions 2

On entre au clavier une série de chiffres ; la fin des entrées est marquée par la saisie du chiffre 0. On veut afficher les répétitions :
Un chiffre n fois (saisi n + 1 fois consécutivement) sera affiché1 fois. Vous pourrez utiliser l'opérateur de concaténation.
Pour la suite 3 4 4 8 5 5 5 6 9 1 1 1 1 7 3 3 5 5 0, on obtient : 4 5 1 3 5

CORRIGES

01-*-Mécanisme de l'instruction répétitive tantque

```
1  Algorithme Tanque1
2  Variables
3    nb : entier
4  Début
5    nb ← 10
6    tantque nb < 40 faire
       écrire(nb)
       nb ← nb + 10
     fintantque
     écrire ('le nombre
         vaut ', nb)
   Fin
```

```
Algorithme Tanque2
Variables
  nb : entier
Début
  nb ← 10
  tantque nb > 40 faire
    écrire(nb)
    nb ← nb + 10
  fintantque
  écrire ('le nombre
      vaut ', nb)
Fin
```

```
Algorithme Tanque3
Variables
  x : entier
Début
  lire(x)
  tantque x <= 3 faire
    écrire(x)
    x ← x + 1
  fintantque
  écrire (x)
Fin
```

Solution

ligne	nb	écran
1	10	
2		
3		10
4	20	
5		
2		
3		20
4	30	
5		
2		
3		30
4	40	
5		
2		
6		Le nombre vaut 40

ligne	nb	écran
1	10	
2		
6		Le nombre vaut 10

ligne	x	écran
1	1	
2		
3		1
4	2	
5		
2		
3		2
4	3	
5		
2		
3		3
4	4	
5		
2		
6		4

02-*-Comptage d'un référendum (oui/non)

```
Algorithme Referendum1
Variables
  vote      : caractere         /* le vote (O/N) */
  nbVotants : entier            /* nombre de votants */
  nbOui     : entier            /* nombre de oui */
  nbNon     : entier            /* nombre de non */
  tauxOui   : réel              /* pourcentage de oui */
  tauxNon   : réel              /* pourcentage de non */
Début
  nbVotants ← 0
  nbOui ← 0
  nbNon ← 0
  répéter
    écrire(quel vote (O/N) ?)
    lire(vote)
    selon vote dans
      'O' : nbOui ← nbOui + 1
      'N' : nbNon ← nbNon + 1
    finselon
    nbVotants ← nbVotants + 1
  jusqu'à vote <> 'O' et vote <> 'N'
  nbVotants ← nbVotants - 1
  si nbVotants <> 0 alors
    si nbOui <> 0 alors
      tauxOui ← nbOui / nbVotants
    sinon
      tauxOui ← 0
    finsi
    tauxNon ← 100 - tauxOui
    écrire(nombre de votants ", nbVotants)
    écrire("oui : ", nbOui, " pourcentage ", tauxOui)
    écrire("non : ", nbNon, " pourcentage ", tauxNon)
  sinon
    écrire("pas de votant")
  finsi
Fin
```

03-*-Comptage d'un référendum (oui/non/blancs/nul)

```
Algorithme Referendum2
Variables
  vote     : caractere          /* le vote (O/N) */
  nbVotants: entier             /* nombre de votants */
  nbOui    : entier               /* nombre de oui */
  nbNon    : entier               /* nombre de non */
  nbBlanc  : entier             /* nombre de blancs */
  nbNul    : entier               /* nombre de nuls */
  tauxOui  : réel               /* pourcentage de oui */
  tauxNon  : réel               /* pourcentage de non */
  tauxBlanc: réel               /* pourcentage de blancs */
  tauxNul  : réel               /* pourcentage de nuls */
Début
  nbVotants ← 0
  nbOui ← 0
  nbNon ← 0
  répéter
    écrire(quel vote (O/N) ?)
    lire(vote)
    selon vote dans
      'O' : nbOui ← nbOui + 1
      'N' : nbNon ← nbNon + 1
      'B' : nbBlanc ← nbBlanc + 1
      'U' : nbNul ← nbNul + 1
    finselon
    nbVotants ← nbVotants + 1
  jusqu'à vote <> 'O' et vote <> 'N'
  nbVotants ← nbVotants - 1
  si nbVotants <> 0 alors
    si nbOui <> 0 alors
      tauxOui ← nbOui / nbVotants
    sinon
      tauxOui ← 0
    finsi
    si nbNon <> 0 alors
      tauxNon ← nbNon / nbVotants
    sinon
      tauxNon ← 0
    finsi
    si nbBlanc <> 0 alors
      tauxBlanc ← nbBlanc / nbVotants
    sinon
      tauxBlanc ← 0
    finsi
    tauxNul ← 100 - tauxOui - tauxNon - tauxBlanc
    écrire(nombre de votants ", nbVotants)
    écrire("oui : ", nbOui, " pourcentage ", tauxOui)
    écrire("non : ", nbNon, " pourcentage ", tauxNon)
    écrire("blanc : ", nbBlanc, " pourcentage ", tauxBlanc)
    écrire("non : ", nbNul, " pourcentage ", tauxNul)
  sinon
    écrire("pas de votant")
  finsi
Fin
```

04-*-Chargement d'un camion

```
Algorithme ChargementCamion
Variables
  capacite  : réel              /* capacité du camion en kg */
  masse     : réel              /* masse d'un paquet en kg*/
  charge    : réel              /* charge du camion */
  nbPaquets : entier            /* nombre de paquets */
  finChargement : booleen       /* vrai si fin du chargement */
Début
  charge ← 0
  nbPaquets ← 0
  finChargement ← faux
  écrire(capacité du camion ?)
  lire(capacite)
  répéter
    écrire(masse du paquet ?)
    lire(masse)
    si (charge + masse) <= capacite alors
      charge ← charge + masse
      nbPaquets ← nbPaquets + 1
    sinon
      finChargement ← vrai
    finsi
  jusqu'à finChargement
  écrire(nombre de paquets : ", nbPaquets)
  écrire("charge du camion : ", charge)
Fin
```

05-*-Mécanisme de l'instruction répétitive pour

```
Algorithme Pour1

Variables
   nb : entier            /* nombre courant */
   i : entier           /* variable de boucle */

    Début
1     nb ← 10
2     pour i de 1 à 4 faire
3       écrire(nb)
4       nb ← nb + 5
5     finPour
6     écrire("nombre = ", nb)
    Fin
```

ligne	Nb	i	i<= 4	affichage
1	10			
2		1	Vrai	
3				10
4	15			
5		2		
2			Vrai	
3				15
4	20			
5		3		
2			Vrai	
3				20
4	25			
5		4		
2			Vrai	
3				25
4	30			
5		5		
2			Faux	
6				Nombre = 30

06-*-Moyenne

```
Algorithme Moyenne_1
Variables
  nbSaisi     : réel              /* nombre saisi */
  moy         : réel              /* moyenne */
  nbNombres   : entier            /* nombre de nombres saisis */
  i           : entier            /* compteur de boucles */
Début
  nbNombres ← 0
  moy ← 0
  écrire("combien de nombres ?")
  lire(nbNombres)
  pour i de 1 à nbNombres
    écrire("nombre positif :")
    lire(nbSaisi)
    moy ← moy + nbSaisi
  finpour
  si nbNombres > 0 alors
    moy ← moy / nbNombres
    écrire(le moyenne est ", moy)
  finsi
Fin
```

```
Algorithme Moyenne_2
Constante
  SENTINNELLE     = -1            /* arrêt si -1 */
Variables
  nbSaisi     : réel              /* nombre saisi */
  moy         : réel              /* moyenne */
  nbNombres   : réel              /* nombre de nombres saisis */
Début
  nbNombres ← 0
  moy ← 0
  écrire("nombre (", SENTINELLE, " pour arrêter) : ")
  lire(nbSaisi)
  tantque nbSaisi <> SENTINELLE
    moy ← moy + nbSaisi
    nbNombres ← nbNombres + 1
    écrire("nombre (", SENTINELLE, " pour arrêter) : ")
    lire(nbSaisi)
  fintantque
  si nbNombres > 0 alors
    moy ← moy / nbNombres
    écrire(le moyenne est ", moy)
  finsi
Fin
```

07-*-Affichage n°1

```
Algorithme AffichageA_1
Variables
  n  : entier        /* nombre de 'a' à afficher  */
  i  : entier        /* compteur de boucles  */

Début
  lire(n)
  pour i variant de 1 à n
    écrire('a')
  finpour
  écrire(CRLF)
Fin
```

08-**-Minimum, maximum et leur rang

```
Algorithme MinMax_1
Constante
  BORNE_INF       = 0              /* borne inférieure */
  BORNE_SUP       = 100            /* borne supérieure */
Variables
  nb          : réel               /* nombre saisi */
  nbNombres   : entier             /* nombre de nombres */
  min         : réel               /* minimum */
  max         : réel               /* maximum */
  rgMin       : entier             /* rang du minimum */
  rgMax       : entier             /* rang du maximum */
  i           : entier             /* indice de boucle */
Début
  min ← BORNE_SUP
  max ← BORNE_INF
  rang ← 0
  écrire("Combien de nombres ? ")
  lire(nbNombres)
  pour i de 1 à nbNombres
    écrire("nombre entre ", BORNE_INF, " et ", BORNE_SUP)
    lire(nb)
    si nb > max alors
      max ← nb
      rgMax ← i
    sinon
      si nb < min alors
        min ← nb
        rgMin ← i
      finsi
    finsi
  finpour
  si nbNombres > 0 alors
    écrire("minimum = ", min, "de rang ', rgMin)
    écrire("maximum = ", max, "de rang ', rgMax)
  finsi
Fin
```

```
Algorithme MinMax_2
Constante
  BORNE_INF       = 0              /* borne inférieure */
  BORNE_SUP       = 100            /* borne supérieure */
Variables
  nb          : réel               /* nombre saisi */
  nbNombres   : entier             /* nombre de nombres */
  min         : réel               /* minimum */
  max         : réel               /* maximum */
  rgMin       : entier             /* rang du minimum */
  rgMax       : entier             /* rang du maximum */
  i           : entier             /* indice de boucle */

Début
  écrire("Combien de nombres ? ")
  lire(nbNombres)
  si nbNombres > 0 alors
    écrire("nombre : ")
    lire(nb)
    min ← nb
    max ← nb
    pour i de 2 à nbNombres
      écrire("nombre entre ", BORNE_INF, " et ", BORNE_SUP)
      lire(nb)
      si nb > max alors
        max ← nb
        rgMax ← i
      sinon
        si nb < min alors
          min ← nb
          rgMin ← i
        finsi
      finsi
    finpour
    écrire("minimum = ", min, "de rang ', rgMin)
    écrire("maximum = ", max, "de rang ', rgMax)
  finsi
Fin
```

09-**-Saisie de notes

```
Algorithme SaisieControlee
Constantes
  BORNE_INF   = 0
  BORNE_SUP   = 20
Variables
  note          : réel             /* note */
  reponse       : caractere        /* réponse (o/n) */

Début
  repeter
    repeter
      écrire("note : ")
      lire(nb)
      si note < BORNE_INF ou note > BORNE_SUP alors
        écrire("Erreur, la note doit être comprise entre ",
            BORNE_INF, " et ", BORNE_SUP)
    jusqu'à note >= BORNE_INF et note <= BORNE_SUP
    écrire("Encore une note (o/n)?")
    lire(reponse)
  jusqu'à reponse <> 'o'
Fin
```

10-**-Affichage n°2

```
Algorithme AffichageA_2_1
/* on affiche p lignes de n ‘a’ */
Variables
  n  : entier    /* nombre de 'a’ à afficher sur une ligne */
  p  : entier    /* nombre de lignes */
  i  : entier    /* compteur de a sur une ligne */
  j  : entier    /* compteur de lignes */

Début
  ecrire(‘Nombre de a à afficher sur une ligne’)
  lire(n)
  ecrire(‘Nombre de lignes’)
  lire(p)
  pour j variant de 1 à p
    pour i variant de 1 à n
      écrire(‘a’)
    finpour
    écrire(CRLF)
  infpour
Fin

Algorithme AffichageA_2_2
/* on affiche n * p ‘a’. On passe à la ligne suivante quand il y a
eu n ‘a’ d'affichés sur la ligne en cours */
Variables
  n     : entier     /* nombre de 'a’ à afficher sur une ligne */
  p     : entier     /* nombre de lignes */
  i     : entier     /* compteur de a sur une ligne */
  nbTot : entier      /* nombre total de 'a' */
  nbALigne  : entier     /* nombre de 'a' sur la ligne */
Début
  ecrire(‘Nombre de a à afficher sur une ligne’)
  lire(n)
  ecrire(‘Nombre de lignes’)
  lire(p)
  nbTot ← n * p
  nbALigne ← 0
  pour j variant de 1 à nbTot
    écrire(‘a’)
    nbALigne ← nbALigne + 1
    si nbALigne = n alors   /* n 'a' affichés sur la ligne */
      écrire(CRLF)          /* passage à la ligne */
      nbALigne ← 0          /* aucun ‘a’ sur la nouvelle ligne */
    finsi
  finpour
Fin
```

11-**-Affichage n°3

```
Algorithme : AffichageA_3
Constantes
  LARGEUR = 4
Variables
  n   : entier        /* nombre de 'a' à afficher sur une ligne */
  cpt_lig  : entier   /* compteur de 'a' affichés sur une ligne */
  cpt_tot  : entier   /* compteur de 'a' affichés au total */

Début
  ecrire('Nombre de a à afficher sur une ligne')
  lire(n)
  cpt_tot ← 0
  cpt_lig ← 0
  tantque (cpt_tot < n)  /* on n'a pas affiché tous les 'a') */
    si (cpt_lig = LARGEUR) alors    /* on a fini la ligne  */
      écrire(CRLF)
      cpt_lig ← 0  /* on remet à 0 le nb de 'a' écrits sur la
                      ligne */
    fsi
    écrire('a')
    cpt_lig ← cpt_lig + 1
    cpt_tot ← cpt_tot + 1
  fintantque
Fin
```

12-**-Affichage n°4

```
Algorithme AffichageA_4
Variables
  n  : entier    /* nombre de lignes à afficher */
  j  : entier    /* compteur de 'a' affichés sur une ligne */
  i  : entier    /* compteur de lignes affichées */

Début
  ecrire('Nombre de a à afficher sur une ligne')
  lire(n)
  pour i variant de 1 à n
    pour j variant de 1 à n-i+1
      écrire('a')
    finpour
    écrire(CRLF)
  finpour
Fin
```

13-**-Affichage n°5

```
Algorithme AffichageA_5
Variables
  n  : entier    /* nombre saisi */
  i  : entier    /* compteur de nombres affichés */
Début
  ecrire('Nombre de a à afficher sur une ligne')
  ecrire('Nombre de a à afficher sur une ligne')
  lire(n)
  pour i variant de 1 à n
      écrire(i, " ")
  finpour
  écrire(CRLF)
Fin
```

14-**-Affichage n°6

```
Algorithme AffichageA_6
Variables
  n  : entier    /* nombre saisi */
  i  : entier    /* compteur de nombres affichés */
Début
  ecrire('Nombre de a à afficher sur une ligne')
  lire(n)
  pour i variant de 1 à n
      écrire(2*i, " ")
  finpour
  écrire(CRLF)
Fin
```

15-**-Affichage n°7

```
Algorithme AffichageA_7
Constantes
  NB_COL = 4
Variables
  n       : entier    /* nombre de nombres à afficher */
  cpt_lig : entier    /* cpt de nombres affichés sur une ligne */
  i       : entier    /* cpt de nombres affichés au total */
Début
  ecrire('Nombre de a à afficher sur une ligne')
  lire(n)
  cpt_lig ← 1
  pour i de 1 à n
    si (cpt_lig = NB_COL) alors  /* on a fini la ligne */
      écrire(CRLF)
      cpt_lig ← 0
    finsi
    écrire(i, '\t')
    cpt_lig ← cpt_lig  + 1
  finpour
Fin
```

16-***-Affichage n°8

```
Algorithme AffichageA_8
Variables
  n   : entier    /* nombre saisi au clavier */
  j   : entier    /* compteur de chiffres sur une ligne */
  i   : entier    /* compteur de lignes */
  k   : entier    /* variable servant à l'affichage */

Début
  ecrire('Nombre de a à afficher sur une ligne')
  lire(n)
  k ← 1
  pour i variant de 1 à n
    pour j variant de 1 à n-i+1
      écrire(k, '\t')
      k ← k + 1
    finpour
    écrire(CRLF)
  finpour
Fin
```

17-***-Affichage n°9

Analyse :
La difficulté réside dans le fait que le raisonnement par colonne semble facile alors que l'on affiche ligne par ligne.
Raisonnement par colonne : on peut connaître le nombre de colonnes pleines nb_pleine et le nombre de a dans la colonne incomplète nb_incomplète:
nb_pleine = n division entière 5 ….. n / 5
nb_incomplète = reste de la division de n par 5 …… n / 5
Pour raisonner en ligne, il faut connaître :
le nombre de lignes à afficher et sur chacune d'elle, le nombre de a.
Le nombre de lignes à afficher :
si n < 5 c'est le nombre de lignes = nb_incomplète
sinon le nombre de ligne est 5
le nombre de a sur une ligne est égal aux nombre de colonnes complète + le nombre de a de la colonne incomplète qu'il reste à afficher.

```
Algorithme AffichageA_9_1
Constantes
  MAX_LIGNE = 5     /* nombre maximum  de lignes */
Variables
  nb_ligne        : entier  /* nombre de lignes à afficher */
  nb_pleine       : entier  /* nombre de colonnes pleines */
  nb_incomplete   : entier  /* nombre de ‘a’ dans la colonne
                               incomplète  */
  j : entier  /* compteur de lignes */
  i : entier  /* compteur pour afficher une ligne */
  k : entier  /* compteur de ‘a’ de la colonne incomplète déjà
                        affichés  */
  nb_a : entier   /* indicateur de a de colonne incomplète
                     0 s'il n'y a plus de a incomplet, 1 sinon */

Début
  ecrire(‘Nombre de a à afficher sur une ligne’)
  lire(n)
  /* calcul du nombre de lignes */
  si (n < MAX_LIGNE) alors
    nb_ligne ← n
  sinon
    nb_ligne ← MAX_LIGNE
  finsi

  nb_pleine ← n div MAX_LIGNE
  nb_incomplete ← n mod MAX_LIGNE
  k ← 0

  pour j variant de 1 à nb_ligne
  /* y-a-t'il un a de colonne incomplète sur la ligne ? */
    si ((nb_incomplete != 0) ET (k <= nb_incomplete)) alors
      nb_a ← 1
      k ← k + 1
    sinon
      nb_a ← 0
    finsi

    // affichage d'une ligne
    pour i variant de 1 à nb_pleine + nb_a
      écrire(‘a’)
    finpour
    écrire(CRLF)
  finpour
Fin
```

Deuxième solution : on affiche d'abord les lignes ayant un nombre de 'a' le plus grand, puis on affiche les autres.

```
Algorithme AffichageA_9_2
Constantes
  MAX_LIGNE = 5     /* nombre maximum  de lignes */
Variables :
 nb_entier : entier
 nb_mod : entier
 i : entier
 j : entier
Début
  ecrire('Nombre de a à afficher sur une ligne')
  lire(n)
  nb_entier ← n div MAX_LIGNE
  nb_mod  ← n mod MAX_LIGNE
  si (nb_entier = 0) alors
    /* le nombre de colonnes est 1 */
    pour i variant de 1 à nb_mod
      écrire('a')
      écrire(CRLF)
    finpour
  sinon   /*le nombre de colonnes à afficher est >=  2 */
     /* les colonnes avec un 'a' en plus */
    pour i variant de 1 à nb_mod
      pour j variant de 1 à nb_entier + 1
         écrire('a')
      finpour
      écrire(CRLF)
    finpour
     /* les colonnes avec un 'a' en moins */
    pour i variant de 1 à MAX_LIGNE-nb_mod
      pour j variant de 1 à nb_entier
         écrire('a')
      finpour
      écrire(CRLF)
    finpour
  finsi
Fin
```

18-***-Recherche de répétitions 1

```
Algorithme Repetition_1
Constantes
  ARRET = 0                          /* saisie d'arrêt */
Variables
  chiffreCourant    : entier         /* entier saisi */
  chiffrePrecedent  : entier         /* entier précédent saisi */
  resultat          : chaine         /* chaîne à afficher */

Début
  resultat ← ""
  lire(chiffreCourant)
  tantque (chiffreCourant != ARRET)
    chiffrePrecedent ← chiffreCourant
    lire(chiffreCourant)
    si (chiffreCourant = chiffrePrecedent) alors
        resultat ← resultat + "  " + chiffreCourant
    finsi
  fintantque
  écrire(resultat, CRLF)
Fin
```

19-***-Recherche de répétitions 2

```
Algorithme Repetition_2
// affichage en début de série
Constantes
  ARRET = 0 /* saisie d'arrêt */
Variables
  chiffreCourant    : entier  /* entier saisi */
  chiffrePrecedent  : entier  /* entier précédent saisi */
  resultat          : chaine  /* chaîne à afficher */
  ecrit             : booléen /* vrai si la répétition a déjà été
                                 écrite */
Début
  resultat ← ""
  ecrit ← faux
  lire(chiffreCourant)
  tantque (chiffreCourant != ARRET)
    chiffrePrecedent ← chiffreCourant
    lire(chiffreCourant)
    si (chiffreCourant = chiffrePrecedent) alors
      si (NON(ecrit)) alors
        resultat ← resultat + "  " + chiffreCourant
        ecrit ← vrai
      finsi
    sinon
      ecrit = faux
    finsi
  fintantque
  écrire(resultat, CRLF)
Fin
```