Correction des exercices de A1

Table des matières

Exercice 1 :

Énoncé :
Corriger le pseudo-code suivant :
1 a et B sont des entiers
2 A ← 4 ; LIRE B
3 C'est une variable ← A+B;
4 ECRIRE c'est une variable

Erreurs :

- a et A sont différents : attention à la « casse ».
- deux instructions sur la même ligne : à séparer.
- pas de symbole ; en fin de ligne
- choisir un nom de variable court, significatif et sans espace

Correction :
1 A et B sont des entiers
2 A ← 4
3 LIRE B
4 c ← A+B
5 ECRIRE c

Exercice 2 :

Énoncé :
Commenter le code suivant :
1 S←0
2 for i←1 to 10
3 S←S+1
4 ECRIRE S

Correction :

```
1 S←0                   // Initialisation de S
2 for i←1 to 10         // Répétitions 10 fois
3       S←S+1           // augmentation de 1 (à chaque itération)
4 ECRIRE S              // Affichage de S
```

Exercice 3 :

Énoncé :

```
1 r←0
2 while r*r <= n
3       r←r+1
4 r←r-1
```

Exemple :

#ligne	n	r	Commentaires
1	5	0	Initialisation
2	5	0	0*0<=5, on rentre dans la ligne 3
3	5	1	r←1
2	5	1	1*1<=5, on rentre dans la ligne 3
3	5	2	r←2
2	5	2	2*2<=5, on rentre dans la ligne 3
3	5	3	r←3
2	5	3	3*3<=5, on sort de la boucle
4	5	2	r←2

En vous inspirant de l'exemple ci-dessus, réaliser une trace de l'algorithme précédent avec n=20.

Correction :

#ligne	n	r	Commentaires
1	20	0	Initialisation
2	20	0	0*0<=20, on rentre dans la ligne 3
3	20	1	r←1
2	20	1	1*1<=20, on rentre dans la ligne 3
3	20	2	r←2
2	20	2	2*2<=20, on rentre dans la ligne 3
3	20	3	r←3
2	20	3	3*3<=20, on rentre dans la ligne 3
3	20	4	r←4
2	20	4	4*4<=20, on rentre dans la ligne 3
3	20	5	r←5
2	20	5	5*5 > 20, on sort de la boucle while
4	20	4	r←4

Exercice 4 :

Énoncé :
Écrire un algorithme en pseudo code qui calcule la moyenne de trois nombres a, b et c.
Le résultat sera stocké dans une variable m.

Correction :
méthode sans boucle répétitive :

```
1 Lire a                    // Initialisation par l'utilisateur des 3 variables a, b et c
2 Lire b
3 Lire c
4 m←a+b+c                   // Initialisation de m
5 m←m/3                     // modification de m
6 ECRIRE m                  // Affichage de m
```

méthode avec boucle répétitive :

```
1 m←0                       // Initialisation de m
2 for i←1 to 3              // répétition 3 fois du bloc suivant
3       Lire a              // Saisie d'un des nombres par l'utilisateu
4       m←m+a               // Modification de m
5 m←m/3
6 ECRIRE m                  // Affichage de m
```

Exercice 5 :

Énoncé :
Ecrire un algorithme qui renvoie le max de deux nombres a et b. Le résultat sera stocké dans une variable max.

Correction :

```
1 Lire a                    // Initialisation par l'utilisateur des 2 variables a et b
2 Lire b
3 if a<b then               // Comparaison des deux nombres
4       max←b               // stockage du maximum
5 else
6       max←a               // stockage du maximum
7 ECRIRE max                // Affichage du maximum
```

Exercice 6 :

Énoncé :
Écrire un algorithme qui demande un nombre entier non nul de départ, et qui calcule la somme des entiers jusqu'à ce nombre. Par exemple, si l'on entre 5, le programme doit calculer : 1 + 2 + 3 + 4 + 5 = 15.

Correction :
Il suffit d'utiliser une boucle répétitive FOR ou WHILE :
Avec FOR :

```
1 Lire n                    // Initialisation par l'utilisateur de la variable  n
2 res ← 1
3 for i←2 to n              // répétition (n-1) fois du bloc suivant
4       res←res+i           // ajout du nouvel entier
5 ECRIRE res
```

Avec WHILE :

```
1 Lire n                    // Initialisation par l'utilisateur de la variable  n
2 res ← 1
3 i← 1                      // Compteur de tour
4 while i<=n                // répétition (n-1) fois du bloc suivant
5       res←res+i           // ajout du nouvel entier
6       i← 1+1              // incrémentation du compteur
7 ECRIRE res
```

Exercice 7 :

Énoncé :
Écrire un algorithme qui stocke dans une variable max le maximum de trois variables a, b et c données.

Correction :
Il suffit d'imbriquer deux tests IF THEN ELSE :

```
1 Lire a                    // Initialisation par l'utilisateur des 3 variables  a, b et c
2 Lire b
3 Lire c
4 if a<b then               // Comparaison des deux nombres
5       if b<c then
6               max←c       // stockage du maximum
7       else                // ici la condition « b<c » est fausse donc b>=c
8               max←b       // stockage du maximum
9 else                      // ici la condition « a<b » est fausse donc a>=b
10      if a<c then
11              max←c
12      else                // ici la condition « a<c » est fausse donc a>=c
13              max←a
// pas d'affichage demandé ici, seulement un stockage : pas « d'ECRIRE max » ici
```

Exercice 8 :

Énoncé :
Trouver l'erreur dans l'algorithme suivant :

```
1 for truc ← 1 to 10
2       truc ← truc * 2
```

Erreur : truc est une variable fixée à la boucle FOR : ce compteur comptabilise le nombre d'itération. Cette variable est modifiée automatiquement par retour à la ligne 1. Il ne faut absolument pas modifier cette variable dans le bloc d'instruction(s) répétée(s) dans cette boucle.

Exercice 9 :

Énoncé :
Trouver l'erreur dans l'algorithme suivant :

```
1 while i > 0
2       i ← i - 1
```

Erreur :
i est une variable qui sert de compteur mais qui n'a pas été initialisée.

Correction possible :

```
1 Lire i
2 while i > 0
3       i ← i - 1
```

Exercice 10 :

Énoncé :
1. Écrire un algorithme en pseudo code qui renvoie le max des éléments d'une liste nommée liste1 ayant n éléments.
2. Écrire une trace d'exécution en prenant la list1 de l'exemple.

Correction :
1. Il suffit de balayer toutes la liste en comparant chaque élément avec le maximum temporaire

```
1 Lire liste1                   // Initialisation par l'utilisateur de la liste listel
2 Lire n                        // Initialisation par l'utilisateur de la taille n de la liste liste1
3 max ← liste1[0]               // Stockage du premier élément de la liste
4 for i←1 to n-1                // Balayage de tous les éléments de la liste ;le dernier est liste1[n-1].
5       if max<liste1[i]
6               max← liste1[i]  // Nouveau maximum temporaire trouvé
7 ECRIRE max                    // Affichage u maximum trouvé
```

2. Écrire une trace d'exécution en prenant la list1 de l'exemple.
Correction :

#ligne	max	i	liste[i]	Commentaires
1				Initialisation : liste1=[3,5,2,14,8]
2				Saisie de n=5.
3	3		3	Le premier élément de la liste est liste1[0] soit 3
4		1		première itération avec i=1 :
5			5	3<5 VRAI : on rentre dans la ligne 6
6	5			max←5
4		2		deuxième itération avec i=2 :
5			2	5<2 FAUX : on ne rentre pas dans la ligne 6, retour à la ligne 4
4		3		troisième itération avec i=2 :
5			14	5<14 VRAI : on rentre dans la ligne 6
6	14			max←14
4		4		dernière itération avec i=4 :
5			8	14<8 FAUX : on ne rentre pas dans la ligne 6
7	14			Affichage du résultat obtenu

Exercice 11 :

Énoncé :
Écrire un algorithme qui permet d'échanger le contenu de deux variables a et b.

Correction :
Il suffit de penser au problème concret de la permutation du contenu de deux verres pleins : il suffit d'en prendre un troisième ! Une troisième variable notée _c sera utile ici.

```
1 Lire a            // Initialisation par l'utilisateur des variables  a et b
2 Lire b
3 _c←a              // La valeur stockée dans a est temporairement stockée dans _c
4 a←b               // La valeur stockée dans b remplace l'ancienne de a : elle disparaît donc de a
5 b←_c              // Heureusement qu'elle a été préalablement conservée dans _c : elle est mise dans b
```

Exercice 12 :

Énoncé :
Après avoir réalisé la trace de cet algorithme avec a=17 et b=3, précisez que représente a et i après exécution de algorithme :

```
1 i←0
2 u←b
3 while a>=b
4       a←a-u
5       i←i+1
```

Correction :

#ligne	contrôle	a	i	u	Commentaires
1			0		i←0
2				3	u←3
3	17>3 VRAI	17			entrée dans la boucle WHILE
4		14			a←17-3
5			1		i←0+1
3	14>3 VRAI				nouvelle itération dans la boucle WHILE
4		11			a←14-3
5			2		i←1+1
3	11>3 VRAI				nouvelle itération dans la boucle WHILE
4		8			a←11-3
5			3		i←2+1
3	8>3 VRAI				nouvelle itération dans la boucle WHILE
4		5			a←8-3
5			4		i←3+1
3	5>3 VRAI				nouvelle itération dans la boucle WHILE
4		2			a←5-3
5			5		i←4+1
3	2>3 FAUX				arrêt de la boucle WHILE

Ainsi, a stocke finalement 2 et i le nombre 5.
Intérêt de l'algorithme : $17=5\times3+2$
a stocke le reste de la division euclidienne du nombre initialement présent dans a par le nombre stocké dans b.
i stocke le quotient de la division euclidienne du nombre initialement présent dans a par le nombre stocké dans b.

Exercice 13 :

Énoncé :
Écrire un algorithme qui dit si un nombre appartient à une liste.

En sortie l'algorithme renvoie un booléen : `True` signifiant que la valeur est dans la liste `False` sinon.

Correction :
Avec FOR : on balaye toute la liste puis on affiche le booléen informant si le nombre a été trouvé dans la liste :

```
1 Lire liste1                    // Initialisation par l'utilisateur de la liste listel
2 Lire n                         // Initialisation par l'utilisateur de la taille n de la liste liste1
3 Lire p                         // Initialisation par l'utilisateur de l'élément à chercher dans la liste
4 res←False                      // Initialisation du booléen stockant l'information de la présence.
5 for i←0 to n-1                 // Balayage de tous les éléments de la liste ;le dernier est liste1[n-1].
6       if p=liste1[i]           // Cas où l'élément n a été trouvé dans la liste
7               res← True
8 ECRIRE res                     // Affichage du booléen
```

Avec WHILE : il suffit de s'arrêter dès que l'élément a été trouvé ; plus rapide ainsi si l'élément a été trouvé mais la condition de poursuite est plus compliquée à écrire (cf. ligne 6).

```
1 Lire liste1                    // Initialisation par l'utilisateur de la liste listel
2 Lire n                         // Initialisation par l'utilisateur de la taille n de la liste liste1
3 Lire p                         // Initialisation par l'utilisateur de l'élément à chercher dans la liste
4 res←False                      // Initialisation du booléen stockant l'information de la présence.
5 i←0                            // Initialisation du compteur de position
6 while res=False AND i<n        // On continue tant que l'on n'a pas trouvé l'élément et qu'il reste des
                                 éléments à tester das la liste
7       if n=liste1[i]           // Cas où l'élément n a été trouvé dans la liste
8               res← True
9       i←i+1                    // incrémentation du compteur de position pour passer à l'élément suivant
10 ECRIRE res                    // Affichage du booléen
```

Exercice 14 :

Énoncé :
Voici un algorithme :

```
1 nb ← 0
2 n ← longueur de la liste
3 for i ← 0 to n-1
4       if liste[i]='a' then
5               nb ← nb + 1
```

1.a. Exécuter l'algorithme avec la liste liste=['r', 'a', 'd', 'a', 'r']

Correction :

#ligne	contrôle	nb	n	i	liste[i]	Commentaires
1		0				nb←0
2			5			5 caractères dans la liste ['r', 'a', 'd', 'a', 'r']
3				0		Première itération de la boucle FOR
4	Liste[0]='a' FAUX				'r'	on ne rentre pas dans la ligne 5
3				1		Deuxième itération de la boucle FOR
4	Liste[1]='a' VRAI				'a'	on rentre dans la ligne 5
5		1				nb←0+1
3				2		i←1+1
4	Liste[2]='a' FAUX				'd'	on ne rentre pas dans la ligne 5
3				3		Quatrième itération de la boucle FOR
4	Liste[3]='a' VRAI				'a'	on rentre dans la ligne 5
5		2				nb←1+1
3				4		Dernière itération de la boucle FOR
4	Liste[4]='a' FAUX				'r'	on ne rentre pas dans la ligne 5

Énoncé :

1.b. Expliquer ce que fait cet algorithme.

Correction :
Ainsi, nb comptabilise le nombre d'élément 'a' de la liste saisie.

Énoncé :

```
1 nb ← 0
2 n ← longueur de la liste
3 for i ← 0 to n-1
4       if liste[i]='a' then
5               nb ← nb + 1
```

1.c. Exécuter l'algorithme avec la liste liste=['e', 's' , 't']

Correction :

#ligne	contrôle	nb	n	i	liste[i]	Commentaires
1		0				nb←0
2			3			3caractères dans la liste ['e', 's' , 't']
3				0		Première itération de la boucle FOR
4	Liste[0]='a' FAUX				'e'	on ne rentre pas dans la ligne 5
3				1		Deuxième itération de la boucle FOR
4	Liste[1]='a' FAUX				's'	on ne rentre pas dans la ligne 5
3				2		Troisième itération de la boucle FOR
4	Liste[2]='a' FAUX				't'	on ne rentre pas dans la ligne 5

Énoncé :

Transformation de l'algorithme.

2. Transformer l'algorithme pour qu'il teste la présence de la lettre 'a'.

Correction :
Il suffit d'utiliser un booléen et

- **une boucle FOR : on balaye toute la liste ;**
- **ou une boucle WHILE : on passe à TRUE lorsque l'on repère le premier 'a' dans la liste et on s'arrête ou on parcours toute la liste en restant à FALSE.**

Avec FOR : on balaye toute la liste puis on affiche le booléen informant si le nombre a été trouvé dans la liste :

```
1 Lire liste1                   // Initialisation par l'utilisateur de la liste listel
2 Lire n                        // Initialisation par l'utilisateur de la taille n de la liste liste1
3 res←False                     // Initialisation du booléen stockant l'information de la présence.
4 for i←0 to n-1                // Balayage de tous les éléments de la liste ;le dernier est liste1[n-1].
5       if liste1[i]='a'        // Cas où l'élément n a été trouvé dans la liste
6               res← True
7 ECRIRE res                    // Affichage du booléen
```

Avec WHILE : il suffit de s'arrêter dès que l'élément a été trouvé ; plus rapide ainsi si l'élément a été trouvé mais la condition de poursuite est plus compliquée à écrire (cf. ligne 6).

```
1 Lire liste1                   // Initialisation par l'utilisateur de la liste listel
2 Lire n                        // Initialisation par l'utilisateur de la taille n de la liste liste1
3 res←False                     // Initialisation du booléen stockant l'information de la présence.
4 i←0                           // Initialisation du compteur de position
5 while res=False AND i<n       // On continue tant que l'on n'a pas trouvé l'élément et qu'il reste des
                                   éléments à tester das la liste
6       if liste1[i]='a'        // Cas où l'élément n a été trouvé dans la liste
7               res← True
8       i←i+1                   // incrémentation du compteur de position pour passer à l'élément suivant
9 ECRIRE res                  // Affichage du booléen
```

Énoncé :

Transformation de l'algorithme.

3. On peut utiliser une instruction `in` . Cette instruction est utilisée en PYTHON. Par exemple :

`for elt in ['a', 'e', 'i'] va boucler pour tous les éléments de la liste. elt va prendre la valeur 'a', puis ensuite la valeur 'e', puis ensuite la valeur 'i'.`

En utilisant cette nouvelle instruction, transformer l'algorithme pour qu'il teste la présence d'une voyelle.

Correction :

Modification par rapport au programme avec FOR en rouge (cf. lignes 4 et 5)

```
1 Lire liste1                    // Initialisation par l'utilisateur de la liste listel
2 Lire n                         // Initialisation par l'utilisateur de la taille n de la liste liste1
3 res←False                      // Initialisation du booléen stockant l'information de la présence.
4 for elt in liste1              // Balayage de tous les éléments de la liste ;le dernier est liste1[n-1].
5       if elt='a'               // Cas où l'élément n a été trouvé dans la liste
6               res← True
7 ECRIRE res                     // Affichage du booléen
```

Énoncé :

Transformation de l'algorithme.

4. Transformer l'algorithme pour qu'il compte le nombre de voyelles.

Correction :

Il suffit de :

- **supprimer les lignes lié au booléen de présence**
- **rajouter un compteur nb**
- **changer le test sur la variable elt**

Modifications par rapport au programme précédent en rouge : rajout des lignes 3 et 8 et l'affichage

```
1 Lire liste1                    // Initialisation par l'utilisateur de la liste listel
2 Lire n                         // Initialisation par l'utilisateur de la taille n de la liste liste1
3 nb←0                           // Initialisation du compteur du nombre de 'a'
4 for elt in liste1              // Balayage de tous les éléments de la liste ;le dernier est liste1[n-1].
6       if elt in ['a','e','i','o','u','y']          // Cas où l'élément est une voyelle
8               nb←nb+1          / incrémentation du nombre de 'a'
9 ECRIRE nb                      // Affichage du nombre de 'a' désormais
```

Exercice 15 :

Énoncé :
En PYTHON, une chaîne de caractères est représentée par une liste.
Par exemple : "Il fait beau" est représenté par la liste ['I' , 'l' , ' ' , 'f', 'a', 'i', 't', " , 'b' , 'e' , 'a' , 'u']

L[0]='I' , L[4]='a', etc

1. Proposer un table d'exécution pour l'algorithme suivant :

```
1 phrase ="arret"
2 lettre ← 'r'
3 fin ← ''
4 for elt in phrase
5       if elt <> lettre then
6               fin ← fin + elt
```

<> signifie "différent de".

Correction :

#ligne	contrôle	lettre	fin	elt	Commentaires
2		‘r’			lettre ← 'r'
3			‘’		initialisation
4				‘a’	Premier caractère de phrase
5	'a'<>’r’ VRAI				Passage à la ligne 6
6			‘a’		on rajoute elt donc ‘a’ à la chaîne fin
4				‘r’	Deuxième itération de la boucle FOR : deuxième caractère
5	'r'<>’r’ FAUX				on ne passe pas à la ligne 6 : itération finie : retour à la ligne 4
4				‘r’	Troisième itération de la boucle FOR : troisième caractère
5	'r'<>’r’ FAUX				on ne passe pas à la ligne 6 : itération finie : retour à la ligne 4
4				‘e’	Quatrième itération de la boucle FOR : quatrième caractère
5	'e'<>’r’ VRAI				Passage à la ligne 6
6			‘ae’		on rajoute elt donc ‘e’ à la chaîne fin
4				‘t’	cinquième itération de la boucle FOR : cinquième caractère
5	't'<>’r’ VRAI				Passage à la ligne 6
6			‘aet’		on rajoute elt donc ‘t’ à la chaîne fin

Commentaires possibles pour comprendre chaque ligne de code :
1 : // Initialisation de la chaîne de caractères, c'est-à-dire de la liste de caractères.
2 : // Initialisation de la lettre cherchée
3 : // Initialisation de la chaîne de caractères stockant le résultat
4 : // Balayage de toute la chaîne phrase caractère par caractère
5 : // Cas où la lettre (le caractère) n'est pas un 'r'
6 : // Ajout du caractère différent de 'r'

2. L'algorithme précédent a été modifié. Que fait ce nouvel algorithme présenté ci-dessous ?

```
1 phrase ="C'est un trou de verdure où chante une rivière."
2 lettre ← 'r'
3 fin ← ''
4 for elt in phrase
```

```
5       if elt <> lettre then
6               fin ← fin + elt
```

<> signifie "différent de".

Rôle : l'algorithme permet de supprimer tous les 'r' d'une phrase saisie.

Énoncé :

3. Comment modifier l'algorithme pour que la variable `fin` soit égale à "rrrrr"

Correction :
Il suffit de changer le symbole « différent de » par « égal »
Modification par rapport au programme précédent en rouge : (cf. lignes 5)

```
1 phrase ="C'est un trou de verdure où chante une rivière."
2 lettre ← 'r'
3 fin ← ''
4 for elt in phrase
5       if elt = lettre then
6               fin ← fin + elt
```

Exercice 16 :

Énoncé :

1. Faire une trace de l'algorithme suivant avec liste = [3,2,5,4,1]:

```
1 PG ← 0
2 IPG ← 0
3 n ← longueur de la liste
4 for i← 0 to n-1
5       if liste[i] > PG then
6               PG ← liste[i]
7               IPG ← i
```

Correction :

Trace d'exécution :

#ligne	contrôle	PG	IPG	n	i	liste[i]	Commentaires
1		0					PG←0
2			0				IPG←0
3				5			5 caractères dans la liste [3,2,5,4,1]
4					0		Première itération de la boucle FOR
5	Liste[0]>0 VRAI					3	on rentre dans la ligne 6
6		3					PG←Liste[0]
7			0				IPG←0
4					1		Deuxième itération de la boucle FOR
5	Liste[1]>3 FAUX					2	on ne rentre pas dans la ligne 6
4					2		Troisième itération de la boucle FOR
5	Liste[2]>3 VRAI					5	on rentre dans la ligne 6
6		5					PG←Liste[0]
7			2				IPG←0
4					3		Quatrième itération de la boucle FOR
5	Liste[3]>5 FAUX					4	on rentre dans la ligne 6
4					4		Dernière itération de la boucle FOR
5	Liste[4]>5 FAUX					1	on ne rentre pas dans la ligne 6

Énoncé :

2. Que fait cet algorithme ? Que représentent les variables PG et IPG ?

Correction :

Cet algorithme permet de trouver le plus grand élément d'une liste ainsi que l'indice de la première occurrence de celui-ci.
PG stocke le plus grand élément trouvé dans la liste ;
IPG stocke l'indice de la première occurrence de ce plus grand élément trouvé.

Exercices de renforcements :

Exercice 17 :

Énoncé :
L'algorithme ci-dessous a pour but de calculer le prix à payer en fonction du nombre d'objets acheté d'un produit. Le produit coûte 2€50 pièce.
Corriger le pseudo-code suivant :
1 A est un entier et B est un réel.
2 Lire A
3 A prend la valeur 2.50B
4 ECRIRE B

Erreurs :
- choisir un nom de variable significatif ; par exemple :prix et n pour nombre
- expliciter toutes les opérations

Correction :
1 n est un entier et prix est un réel.
2 Lire n
3 prix prend la valeur 2.50*n
4 ECRIRE prix

Exercice 18 :

Énoncé :
Écrire un algorithme en pseudo code qui calcule la moyenne de cinq nombres a, b, c, d et e.
Le résultat sera stocké dans une variable moy.

Correction :
méthode sans boucle répétitive :

```
1 Lire a                    // Initialisation par l'utilisateur des 5 variables a, b, c, d et e
2 Lire b
3 Lire c
4 Lire d
5 Lire e
6 moy←a+b+c+d+e             // Initialisation de moy
7 moy←moy/5                 // modification de moy
8 ECRIRE moy                // Affichage de moy
```

méthode avec boucle répétitive :

```
1 moy←0                     // Initialisation de moy
2 for i←1 to 5              // répétition 5 fois du bloc suivant
3       Lire a              // Saisie d'un des nombres par l'utilisateut
4       moy←moy+a           // Modification de moy
5 moy←moy/5
6 ECRIRE moy                // Affichage de moy
```

Exercice 19 :

Énoncé :
Ecrire un algorithme qui renvoie le max de deux nombres a et b. Le résultat sera stocké dans une variable max.

Correction :

```
1 Lire a                    // Initialisation par l'utilisateur des 2 variables a et b
2 Lire b
3 if a<b then               // Comparaison des deux nombres
4       max←b               // stockage du maximum
5 else
6       max←a               // stockage du maximum
7 ECRIRE max                // Affichage du maximum
```

Exercice 20 :

Énoncé :
La factorielle d'un nombre entier non nul (factorielle se note avec un !). Par exemple $4!=4\times3\times2\times1=24$. Ainsi, $1!=1$ et pour $n>1$: $n!=n\times(n-1)\times...\times2\times1=n\times(n-1)!$

Écrire un algorithme qui demande un nombre entier n non nul de départ, et qui calcule n!, c'est-à-dire le produit des entiers jusqu'à ce nombre n.

Correction :
Il suffit d'utiliser une boucle répétitive FOR ou WHILE et d'utiliser le lien :
$n!=n\times(n-1)\times...\times2\times1=n\times(n-1)!$
Avec FOR :

```
1 Lire n                    // Initialisation par l'utilisateur de la variable  n
2 res ← 1
3 for i←2 to n              // répétition (n-1) fois du bloc suivant
4       res←res×i           // multiplication pour utiliser  n!=n×(n−1)×...×2×1
5 ECRIRE res
```

Avec WHILE :

```
1 Lire n                    // Initialisation par l'utilisateur de la variable  n
2 res ← 1
3 i← 1                      // Compteur de tour
4 while i<=n                // répétition (n-1) fois du bloc suivant
5       res←res×i           // multiplication pour utiliser  n!=n×(n−1)×...×2×1
6       i← 1+1              // incrémentation du compteur
7 ECRIRE res
```

Exercice 21 :

Énoncé :
Écrire un algorithme qui stocke dans une variable min le minimum de trois variables a, b et c données.

Correction :
Il suffit d'imbriquer deux tests IF THEN ELSE :

```
1 Lire a                    // Initialisation par l'utilisateur des 3 variables  a, b et c
2 Lire b
3 Lire c
4 if a>b then               // Comparaison des deux nombres
5       if b>c then
6               min←c       // stockage du minimum
7       else                // ici la condition « b>c » est fausse donc b<=c
8               min←b       // stockage du minimum
9 else                      // ici la condition « a>b » est fausse donc a<=b
10      if a>c then
11              min←c
12      else                // ici la condition « a>c » est fausse donc a<=c
13              min←a
```

// pas d'affichage demandé ici, seulement un stockage : pas « d'ECRIRE min » ici

Exercice 22 :

Énoncé :
1. Écrire un algorithme en pseudo code qui renvoie le min des éléments d'une liste nommée liste1 ayant n éléments.
2. Écrire une trace d'exécution en prenant la liste1 [7,4,3,8,9] de l'exemple.

Correction :
1. Il suffit de balayer toutes la liste en comparant chaque élément avec le maximum temporaire

```
1 Lire liste1                    // Initialisation par l'utilisateur de la liste listel
2 Lire n                         // Initialisation par l'utilisateur de la taille n de la liste liste1
3 min ← liste1[0]                // Stockage du premier élément de la liste
4 for i←1 to n-1                 // Balayage de tous les éléments de la liste ;le dernier est liste1[n-1].
5       if min>liste1[i]
6               min← liste1[i]   // Nouveau minimum temporaire trouvé
7 ECRIRE min                     // Affichage un minimum trouvé
```

2. Écrire une trace d'exécution en prenant la liste1 de l'exemple.
Correction :

#ligne	min	i	liste[i]	Commentaires
1				Initialisation : liste1=[7,4,3,8,9]
2				Saisie de n=5.
3	7		7	Le premier élément de la liste est liste1[0] soit 7
4		1		première itération avec i=1 :
5			4	7>4 VRAI : on rentre dans la ligne 6
6	4			max←4
4		2		deuxième itération avec i=2 :
5			3	4>3 VRAI : on rentre dans la ligne 6
6	3			max←3
4		3		troisième itération avec i=2 :
5			8	8>3 FAUX : on ne rentre pas dans la ligne 6 : retour à la ligne 4
4		4		dernière itération avec i=4 :
5			9	9>3 FAUX : on ne rentre pas dans la ligne 6
7	3			Affichage du résultat obtenu