Institut National Polytechnique
Félix HOUPHOUET-BOIGNY

LARIT

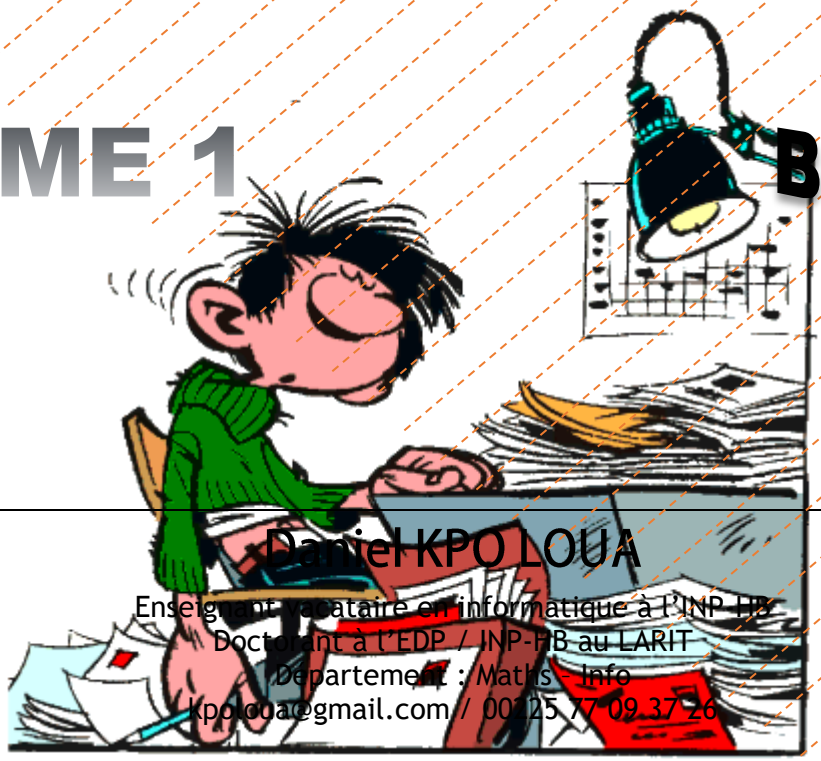
Multi-Algo Prépa

M.P.S.I / B.C.P.S.T 1

MP

BCPST 2

TOME 1



Auteur

Daniel KPO LOUA

Enseignant vacataire en informatique à l'INP-HB
Doctorant à l'EDP / INP-HB au LARIT
Département : Maths - Info
kpolouad@gmail.com // 00225 77 09 37 26

Cahier D'activités

18 Décembre 2018

A Celui qui peut tout
Et qui me rend capable de tout,
Mon bien-aimé Père !

Table des matières

- Concernant ce cahier d'activité 4

CHAPITRE 1 : Algorithmes linéaires 5

Introduction 5

Exercice 001 (corrigé) 5
Exercice 002 (corrigé) 5
Exercice 003 (corrigé) 5
Exercice 004 (corrigé) 6
Exercice 005 (corrigé) 6
Exercice 006 (corrigé) 6
Exercice 007 (corrigé) 6
Exercice 008 (corrigé) 7
Exercice 009 (corrigé) 7
Exercice 010 (corrigé) 7
Exercice 011 (corrigé) 7
Exercice 012 (corrigé) 8
Exercice 013 (corrigé) 8
Exercice 014 (corrigé) 8

Les instructions de lecture et d'écriture 8

Exercice 015 (corrigé) 8
Exercice 016 (corrigé) 8
Exercice 017 (corrigé) 9
Exercice 018 (corrigé) 9

CHAPITRE 2 : Les structures de contrôle 10

- Les tests ou les structures conditionnelles 10

Exercice 019 (corrigé) 10
Exercice 020 (corrigé) 10
Exercice 021 (corrigé) 10
Exercice 022 (corrigé) 10
Exercice 023 (corrigé) 10
Exercice 024 (corrigé) 11
Exercice 025 (corrigé) 11
Exercice 026 (corrigé) 11
Exercice 027 (corrigé) 11

Exercice 028 (corrigé) 11
Exercice 029 (corrigé) 11
Exercice 030 (corrigé) 12
Exercice 031 (corrigé) 12
Exercice 032 (corrigé) 12
Exercice 033 (corrigé) 13
- Schéma répétitif 13

Exercice 034 (corrigé) 13
Exercice 035 (corrigé) 13
Exercice 036 (corrigé) 13
Exercice 037 (corrigé) 13
Exercice 038 (corrigé) 13
Exercice 039 (corrigé) 14
Exercice 040 (corrigé) 14
Exercice 041 (corrigé) 14
Exercice 042 (corrigé) 14
Exercice 043 (corrigé) 14
Exercice 044 (corrigé) 14
Exercice 045 (corrigé) 15
Exercice 046 (corrigé) 15
Exercice 047 (corrigé) 15
Exercice 048 (corrigé) 15
Exercice 049 (corrigé) 15
Exercice 050 (corrigé) 16
Exercice 051 (corrigé) 16
Exercice 052 (corrigé) 16

CHAPITRE 4 : Les sous-algorithmes 17

- Les procédures et fonctions 17

Exercice 053 (corrigé) 17
Exercice 054 (corrigé) 17
Exercice 055 (corrigé) 17
Exercice 056 (corrigé) 17
Exercice 057 (corrigé) 17
Exercice 058 (corrigé) 17
Exercice 059 (corrigé) 18

K.L.

Exercice 060 (corrigé) 18
Exercice 061 (corrigé) 18
Exercice 062 (corrigé) 19
Exercice 063 (corrigé) 19
Exercice 064 (corrigé) 19
Exercice 065 (corrigé) 19
Exercice 066 (corrigé) 19
Exercice 067 (corrigé) 19
Exercice 068 (corrigé) 19
Exercice 069 (corrigé) 19
Exercice 070 (corrigé) 19

CHAPITRE 5 : Les types construits.....20

- Les tableaux.....20

Exercice 071 (corrigé) 20
Exercice 072 (corrigé) 20
Exercice 073 (corrigé) 20
Exercice 074 (corrigé) 20
Exercice 075 (corrigé) 21
Exercice 076 (corrigé) 21
Exercice 077 (corrigé) 21
Exercice 078 (corrigé) 22
Exercice 079 (corrigé) 22
Exercice 080 (corrigé) 22
Exercice 081 (corrigé) 22
Exercice 082 (corrigé) 22
Exercice 083 (corrigé) 23
Exercice 084 (corrigé) 23
Exercice 085 (corrigé) 23
Exercice 086 (corrigé) 23
Exercice 087 (corrigé) 23
Exercice 088 (corrigé) 23
Exercice 089 (corrigé) 24
Exercice 090 (corrigé) 24
Exercice 091 (corrigé) 24
Exercice 092 (corrigé) 24

- Les enregistrements....24

Exercice 093 (corrigé) 24
Exercice 094 (corrigé) 25
Exercice 095 (corrigé) 25
Exercice 096 (corrigé) 25
Exercice 097 (corrigé) 25
Exercice 098 (corrigé) 26
Exercice 099 (corrigé) 27
Exercice 100 (corrigé) 28

Les corrections....28

- Schéma itératif 43
- Les procédures et fonctions 51
- Les tableaux....60
- Les enregistrements....69

Concernant ce cahier d'activité

Ce cahier d'activité à vue le jour pour répondre aux différents besoins de mise à niveau exprimés par des étudiants en 2ième année de prépa (MP et BCPST 2) de l'ESETEC (l'école supérieure de l'enseignement technique et commercial). Il propose 100 algorithmes corrigés (sauf certains) en LDA. La majeure partie de son contenu a été le fruit de profondes analyses et de recherche sur l'internet. Ce qui fait de lui un outil très important dans l'apprentissage des étudiants.

Pour ma part, j'ose croire que l'étudiant qui parviendra à traiter rigoureusement tous ces exercices minutieusement sélectionnés aura un avenir radieux dans le domaine de la programmation.

Ce document n'est pas exigé lors des cours et ne s'impose à personne. Cependant il n'est pas gratuit et son prix de vente est fixé à 3 000 f CFA payable uniquement avec l'autorisation de l'auteur.

L'auteur interdit **strictement les étudiants de photocopier** ce document ou même de le numériser (prendre en photo) partiellement ou totalement.

Quant aux enseignants, l'autorisation leur est accordée de photocopier partiellement des portions pour les devoirs et interrogations.

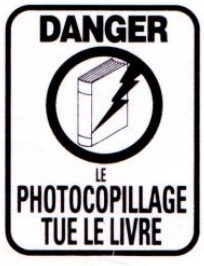
Ce document vise à :

- Renforcer les TD de classe ;
- Amener l'étudiant à travailler plus en vue d'avoir la maitrise parfaite des algorithmes ;
- Renforcer les capacités de l'étudiant dans la rédaction du corps d'un algorithme à travers les corrections bien rédigées ;
- Etc…

N'y aurait-il pas une erreur dans le corrigé de tel ou tel exercice ?

Si, il peut en avoir malgré les lectures déjà faites. Mais si vous en trouvez ou si vous pensez en trouver, veillez s'il vous plait me le signaler en classe ou par mon mail *kpoloua@gmail.com* en précisant votre nom, votre école et votre classe.

Bon courage et puisse Dieu bénir le fruit de vos efforts !
L'auteur.



K.L.

CHAPITRE 1 : Algorithmes linéaires

Introduction

Exercice 001 (corrigé)

Questions du cours

1. Donner la définition d'un algorithme ;
2. Donner la composition d'un algorithme ;
3. Donner la définition d'une variable ;
4. Donner les quatre éléments caractérisant une variable ;
5. Comment déclarant-on une variable ?
6. Donner les types de variables utilisés en algorithmique (LDA);
7. Lors de la déclaration d'une variable, cette dernière possède-t-elle une valeur ?
8. Que permet de faire une instruction d'affectation ?
9. Que font les instructions de lecture et d'écriture ? Donner un exemple pour chacun ;
10. Le nom d'une variable doit respecter certaines normes, lesquelles ?

Exercice 002 (corrigé)

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Var A, B : Entier
Debut
  A ← 1
  B ← A + 3
  A ← 3
Fin
```

Exercice 003 (corrigé)

Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

```
Var A, B, C : Entier
Debut
  A ← 5
  B ← 3
  C ← A + B
  A ← 2
  C ← B - A
Fin
```

Exercice 004 (corrigé)

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Var A, B : Entier
Debut
  A ← 5
  B ← 2
  A ← B
  B ← A
Fin
```

Moralité : les deux dernières instructions permettent-elles d'échanger les deux valeurs de B et A ? Si l'on inverse les deux dernières instructions, cela change-t-il quelque chose ?

Exercice 005 (corrigé)

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Var A, B : Entier
Debut
  A ← 5
  B ← A + 4
  A ← A + 1
  B ← A - 4
Fin
```

Exercice 006 (corrigé)

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Var A, B :Entier
Debut
  A ← 5
  B ← A % 2
  A ← A + 1
  B ← A - 4
Fin
```

Exercice 007 (corrigé)

1. Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Algorithme Calcul
Var A, B : Entier
Debut
      A ← 2
      B ← A+5
      A ← A+B
      B ← B+2
      A ← B – A
  Fin
```

Exercice 008 (corrigé)

Quelles seront les valeurs des variables A, B, C, D et E après exécution des instructions suivantes ?

```
Algorithme Calcul
Var A, B, C : Entier
     D, E : Reel
 Debut
      A ← 2
      B ← 5
      C ← B /A
      D ← C % A
      E ← B /A
Fin
```

Exercice 009 (corrigé)

Que fait l'algorithme suivant :

```
Algorithme Calcul
Var A, B : Entier
Debut
      A ← A+B
      B ← A– B
      A ← A – B
Fin
```

Exercice 010 (corrigé)

Écrivez un algorithme permettant de produire le même résultat, mais sans faire des opérations arithmétiques.

Exercice 011 (corrigé)

Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

```
Var A, B, C : Entier
Debut
  A ← 3
  B ← 10
  C ← A + B
  B ← A + B
  A ← C
Fin
```

Exercice 012 (corrigé)

Plus difficile, mais c'est un classique absolu, qu'il faut absolument maîtriser : écrire un algorithme permettant d'échanger les valeurs de deux variables A et B, et ce quel que soit leur contenu préalable.

Exercice 013 (corrigé)

Une variante du précédent : on dispose de trois variables A, B et C. Ecrivez 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).

Exercice 014 (corrigé)

Que contient la variable C à la fin de cet algorithme ?

```
Var A, B, C : Caracteres
Debut
  A ← "423"
  B ← "12"
  C ← A + B
Fin
```

Les instructions de lecture et d'écriture

Exercice 015 (corrigé)

Soit deux entiers a et b. On suppose que a=7 et b= 4. Quel est le résultat des instructions suivantes : Ecrire (a < b) ; Ecrire ("a < b") ; Ecrire ('a' < 'b') ;

Exercice 016 (corrigé)

Quel résultat produit le programme suivant ?

```
Var Val, Double : Reel
Debut
   Val ← 231
   Double ← Val * 2
   Ecrire(Val)
   Ecrire(Double)
Fin
```

Exercice 017 (corrigé)

Ecrire un programme qui demande un nombre à l'utilisateur, puis qui calcule et affiche le carré de ce nombre.

Exercice 018 (corrigé)

Ecrire un programme qui demande son prénom à l'utilisateur, et qui lui réponde par un charmant « Bonjour » suivi du prénom. On aura ainsi le dialogue suivant :

- *machine* : Quel est votre prénom ?
- *utilisateur* : Marie-Cunégonde
- *machine* : Bonjour, Marie Cunégonde !

CHAPITRE 2 : Les structures de contrôle

Les tests ou les structures conditionnelles

Un test est la vérification d'une condition logique. Le test est soit Vrai ou Faux, il est donc booléen. Il serait préférable de connaitre les différentes opérations ou conditions possibles.

ET	VRAI	FAUX
VRAI	.....	.....
FAUX	.....	.....

OU	VRAI	FAUX
VRAI	.....	.....
FAUX	.....	.....

Exercice 019 (corrigé)

Ecrire un algorithme qui permet de discerner une mention à un étudiant selon la moyenne de ses notes :

- "Très bien" pour une moyenne comprise entre 16 et 20 (16<= moyenne <=20)
- "Bien" pour une moyenne comprise entre 14 et 16 (14<= moyenne <16)
- "Assez bien" pour une moyenne comprise entre 12 et 14 (12<= moyenne <14)
- "Passable" pour une moyenne comprise entre 10 et 12 (10<= moyenne <12)

Exercice 020 (corrigé)

Ecrire un algorithme qui demande un nombre à l'utilisateur, et l'informe ensuite si ce nombre est positif ou négatif (on laisse de côté le cas où le nombre vaut zéro).

Exercice 021 (corrigé)

Ecrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si leur produit est négatif ou positif (on laisse de côté le cas où le produit est nul). Attention toutefois : on ne doit pas calculer le produit des deux nombres.

Exercice 022 (corrigé)

Ecrire un algorithme qui demande trois nombres à l'utilisateur et l'informe ensuite s'ils sont rangés ou non dans l'ordre croissant.

Exercice 023 (corrigé)

Ecrire un algorithme qui demande un nombre à l'utilisateur, et l'informe ensuite si ce nombre est positif ou négatif (on inclut cette fois le traitement du cas où le nombre vaut zéro).

Exercice 024 (corrigé)

Ecrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si le produit est négatif ou positif (on inclut cette fois le traitement du cas où le produit peut être nul). Attention toutefois, on ne doit pas calculer le produit !

Exercice 025 (corrigé)

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

Peut-on concevoir plusieurs algorithmes équivalents menant à ce résultat ?

Exercice 026 (corrigé)

Formulez un algorithme équivalent à l'algorithme suivant :

```
Si (Tutu > Toto + 4) OU (Tata == "OK" )Alors
  Tutu ← Tutu + 1
Sinon
  Tutu ← Tutu - 1
Finsi
```

Exercice 027 (corrigé)

Cet algorithme est destiné à prédire l'avenir, et il doit être infaillible !
Il lira au clavier l'heure et les minutes, et il affichera l'heure qu'il sera une minute plus tard. Par exemple, si l'utilisateur tape 21 puis 32, l'algorithme doit répondre : "Dans une minute, il sera 21 heure(s) 33".
NB : on suppose que l'utilisateur entre une heure valide. Pas besoin donc de la vérifier.

Exercice 028 (corrigé)

De même que le précédent, cet algorithme doit demander une heure et en afficher une autre. Mais cette fois, il doit gérer également les secondes, et afficher l'heure qu'il sera une seconde plus tard.
Par exemple, si l'utilisateur tape 21, puis 32, puis 8, l'algorithme doit répondre : "Dans une seconde, il sera 21 heure(s), 32 minute(s) et 9 seconde(s)".
NB : là encore, on suppose que l'utilisateur entre une date valide.

Exercice 029 (corrigé)

Un magasin de reprographie facture 0,10 E les dix premières photocopies, 0,09 E les vingt suivantes et 0,08 E au-delà. Ecrivez un algorithme qui demande à l'utilisateur le nombre de photocopies effectuées et qui affiche la facture correspondante.

Exercice 030 (corrigé)

Les habitants de Kabakouma 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

Le programme demandera donc l'âge et le sexe du Kabakoumanois, et se prononcera donc ensuite sur le fait que l'habitant est imposable.

Exercice 031 (corrigé)

Les élections législatives, en Guignolerie Septentrionale, obéissent à la règle suivante :

- Lorsque l'un des candidats obtient plus de 50% des suffrages, il est élu dès le premier tour.
- En cas de deuxième tour, peuvent participer uniquement les candidats ayant obtenu au moins 12,5% des voix au premier tour.

Vous devez écrire un algorithme qui permette la saisie des scores de quatre candidats au premier tour. Cet algorithme traitera ensuite le candidat numéro 1 (et uniquement lui) : il dira s'il est élu, battu, s'il se trouve en ballottage favorable (il participe au second tour en étant arrivé en tête à l'issue du premier tour) ou défavorable (il participe au second tour sans avoir été en tête au premier tour).

Exercice 032 (corrigé)

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

- un conducteur de moins de 25 ans et titulaire du permis depuis moins de deux ans, se voit attribuer le tarif rouge, si toutefois il n'a jamais été responsable d'accident. Sinon, la compagnie refuse de l'assurer.
- un conducteur de moins de 25 ans et titulaire du permis depuis plus de deux ans, ou de plus de 25 ans mais titulaire du permis depuis moins de deux ans a le droit au tarif orange s'il n'a jamais provoqué d'accident, au tarif rouge pour un accident, sinon il est refusé.
- un conducteur de plus de 25 ans titulaire du permis depuis plus de deux ans bénéficie du 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. Avant de se lancer à corps perdu dans cet exercice, on pourra réfléchir un peu et s'apercevoir qu'il est plus simple qu'il n'en a l'air (cela s'appelle faire une analyse !)

Exercice 033 (corrigé)

Ecrivez un algorithme qui après avoir demandé un numéro de jour, de mois et d'année à l'utilisateur, renvoie s'il s'agit ou non d'une date valide.

Cet exercice est certes d'un manque d'originalité affligeant, mais après tout, en algorithmique comme ailleurs, il faut connaître ses classiques ! Et quand on a fait cela une fois dans sa vie, on apprécie pleinement l'existence d'un type numérique « date » dans certains langages…).

Il n'est sans doute pas inutile de rappeler rapidement que 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. Ouf !

Un dernier petit détail : vous ne savez pas, pour l'instant, exprimer correctement en pseudo-code l'idée qu'un nombre A est divisible par un nombre B. Aussi, vous vous contenterez d'écrire en bons télégraphistes que A divisible par B se dit « A dp B ».

Schéma répétitif

Exercice 034 (corrigé)

Ecrire un algorithme qui affiche exactement 100 fois la phrase : "je dois absolument passer l'examen de TP qui compte 25% de la note finale".

Exercice 035 (corrigé)

Écrire un algorithme qui affiche les entiers de 1 à 100.

Exercice 036 (corrigé)

Écrire un algorithme qui affiche les entiers pairs de 1 à 100.

Exercice 037 (corrigé)

Ecrire un algorithme qui calcule et affiche la somme des n premiers nombres entiers positifs. L'algorithme demandera à l'utilisateur d'entrer la valeur de n.

Exercice 038 (corrigé)

Exécuter l'algorithmes suivants :

```
Vari i, j : Entier
Debut
      Pour i ← 1 a 2 Faire
          Ecrire(" i= ", i)
          Pour j ← de 1 a 3 Faire
              Ecrire("Le produit de",i," et ",j," est:",i*j)
         FinPour
      FinPour
Fin
```

Exercice 039 (corrigé)

Exécuter l'algorithmes suivants :

```
Var i, j : Entier
Debut
      Pour i ←  1 a 2 Faire
           Ecrire(" i= ", i)
      FinPour
      Pour j ← 1 a 3 Faire
           Ecrire("le produit de",i," et ",j," est:",i*j)
      FinPour
Fin
```

Exercice 040 (corrigé)

Écrivez un algorithme qui permet de compter le nombre de bits nécessaires pour coder en binaire un entier n.

Exercice 041 (corrigé)

Ecrire un algorithme qui compte le nombre de 1 dans la représentation binaire de l'entier n.

Exercice 042 (corrigé)

Ecrire un algorithme qui demande un nombre compris entre 10 et 20 à l'utilisateur, jusqu'à ce que la réponse convienne. En cas de réponse supérieure à 20, on fera apparaitre un message : *Plus petit !* et inversement, *Plus grand !* si le nombre est inférieur à 10.

Exercice 043 (corrigé)

Ecrire un algorithme qui demande à l'utilisateur un nombre compris entre 1 et 3 jusqu'à ce que la réponse convienne.

Exercice 044 (corrigé)

Ecrire un algorithme (sans utiliser la boucle **Pour)** qui demande un nombre de départ, et qui ensuite affiche les dix nombres suivants. Par exemple, si l'utilisateur entre le nombre 17, le programme affichera les nombres : 18 19 20 21 22 23 24 25 26 27.

Exercice 045 (corrigé)

Réécrire l'algorithme précédent, en utilisant cette fois l'instruction **Pour**

Exercice 046 (corrigé)

Ecrire un algorithme qui demande un nombre de départ, et qui ensuite écrit la table de multiplication de ce nombre, présentée comme suit (cas où l'utilisateur entre le nombre 7) :

```
Table de 7 :
7 x 1 = 7
7 x 2 = 14
…
7 x 10 = 70
```

Exercice 047 (corrigé)

Ecrire un algorithme qui demande un nombre 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
```

NB : on souhaite afficher uniquement le résultat, pas la décomposition du calcul.

Exercice 048 (corrigé)

Ecrire un algorithme qui demande un nombre de départ, et qui calcule sa factorielle.
NB : la factorielle de 8, notée 8 ! vaut 1 x 2 x 3 x 4 x 5 x 6 x 7 x 8

Exercice 049 (corrigé)

Ecrire un algorithme qui demande successivement 20 nombres à l'utilisateur, et qui lui dit ensuite quel était le plus grand parmi ces 20 nombres :

```
Entrez le nombre numéro 1 : 12
Entrez le nombre numéro 2 : 14
etc.
```

```
Entrez le nombre numéro 20 : 6
Le plus grand de ces nombres est : 14
```

Modifiez ensuite l'algorithme pour que le programme affiche de surcroît en quelle position avait été saisie ce nombre :

```
C'était le nombre numéro 2
```

Exercice 050 (corrigé)

Réécrire l'algorithme précédent, mais cette fois-ci on ne connaît pas d'avance combien l'utilisateur souhaite saisir de nombres. La saisie des nombres s'arrête lorsque l'utilisateur entre un zéro.

Exercice 051 (corrigé)

Lire la suite des prix (en euros entiers et terminée par zéro) des achats d'un client. Calculer la somme qu'il doit, lire la somme qu'il paye, et simuler la remise de la monnaie en affichant les textes "10 Euros", "5 Euros" et "1 Euro" autant de fois qu'il y a de coupures de chaque sorte à rendre.

Exercice 052 (corrigé)

Écrire un algorithme qui permette de connaître ses chances de gagner au tiercé, quarté, quinté et autres impôts volontaires.
On demande à l'utilisateur le nombre de chevaux partants, et le nombre de chevaux joués. Les deux messages affichés devront être :

```
Dans l'ordre : une chance sur X de gagner
Dans le désordre : une chance sur Y de gagner
```

X et Y nous sont donnés par la formule suivante, si n est le nombre de chevaux partants et p le nombre de chevaux joués (on rappelle que le signe ! signifie "factorielle", comme dans l'exercice 5.6 ci-dessus) :

```
X = n ! / (n - p) !
Y = n ! / (p ! * (n – p) !)
```

NB : cet algorithme peut être écrit d'une manière simple, mais relativement peu performante. Ses performances peuvent être singulièrement augmentées par une petite astuce. Vous commencerez par écrire la manière la plus simple, puis vous identifierez le problème, et écrirez une deuxième version permettant de le résoudre.

CHAPITRE 4 : Les sous-algorithmes

Les procédures et fonctions

Exercice 053 (corrigé)

Parmi ces affectations (considérées indépendamment les unes des autres), lesquelles provoqueront des erreurs, et pourquoi ?

```
Var A, B, C : Reel
Var D : Caractere
Debut
  A ← Sin(B)
  A ← Sin(A + B * C)
  B ← Sin(A) - Sin(D)
  C ← Sin(A / B)
  C ← Cos(Sin(A)
Fin
```

Exercice 054 (corrigé)

Ecrivez un algorithme qui demande un mot à l'utilisateur et qui affiche à l'écran le nombre de lettres de ce mot.

Exercice 055 (corrigé)

Écrivez une fonction qui renvoie la somme de cinq nombres fournis en argument.

Exercice 056 (corrigé)

Ecrire un traitement qui informe si un tableau envoyé en argument est formé ou non d'éléments tous rangés en ordre croissant.

Exercice 057 (corrigé)

Ecrire un traitement qui inverse le contenu de deux valeurs passées en argument.

Exercice 058 (corrigé)

Créer un petit ensemble de procédures et de fonctions permettant de manipuler facilement les heures et les minutes et composé de :

a) La fonction `Minutes`, qui calcule le nombre des minutes correspondant à un nombre d'heures et un nombre de minutes donnés.

b) La fonction ou la procédure `HeuresMinutes` qui réalise la transformation inverse de la fonction Minute. Pour `HeuresMinutes`, il y a deux résultats à fournir, une fonction ne peut convenir, il faut donc écrire une procédure comportant trois paramètres : La durée (entrée), l'heure (sortie) et les minutes (sortie).

c) La procédure `AjouteTemps` qui additionne deux couples de données heures et minutes en utilisant les deux fonctions précédentes. La procédure `AjouteTemps` reçoit quatre paramètres en entrée, fournit deux paramètres en sortie. La variable locale `MinuteEnTout` sert à stocker un résultat intermédiaire, mais elle n'est pas indispensable.

Exercice 059 (corrigé)

Cet exercice permet de compléter les procédures et fonctions de l'exercice précédent

1. Créer une fonction qui permet de dire si un mois a 30 jours ou non. Cette fonction renverra 1 si c'est le cas et 0 sinon.

1	2	3	4	5	6	7	8	9	10	11	12
Jan	Fev	Mars	Avril	Mai	Juin	Juil	Aout	Sept	Oct	Nov	Dec
31	28/ 29	31	30	31	30	31	31	30	31	30	31

2. Créer une fonction qui permet de dire si un mois a 31 jours ou non. Cette fonction renverra 1 si c'est le cas et 0 sinon.

3. Créer une fonction qui permet de dire si une année est bissextile ou non. Cette fonction renverra 1 si c'est le cas et 0 sinon. Pour qu'une année soit bissextile, il suffit que l'année soit un nombre divisible par 4 et non divisible par 100, ou alors qu'elle soit divisible par 400.

4. Utiliser ces fonctions pour écrire une fonction `NombreDeJour()` retournant le nombre de jours pour un mois et une année donnée.

5. Écrire un programme principal permettant à l'utilisateur d'entrer un numéro de mois (entre 1 et 12) et une année (entre 1995 et 2100), qui seront ensuite passés en paramètres à la fonction `NombreDeJour()`. Il faut tester la validité des mois et années.

Exercice 060 (corrigé)

Ecrire une fonction ou procédure qui calcule la partie entière d'un nombre positif.

Exercice 061 (corrigé)

Ecrire une fonction ou procédure qui affiche le tableau de multiplication d'un entier positif x.

Exercice 062 (corrigé)

Ecrire une fonction ou procédure qui permet de lire deux nombres, calculer la somme et le produit et affiche si ces derniers sont positifs ou négatifs.

Exercice 063 (corrigé)

Ecrire une fonction ou procédure qui permet de lire une liste de nombres entiers dont la dernière valeur = -1 et affiche le nombre d'entiers pairs et leur pourcentage par rapport au nombre d'entiers donnés.

Exercice 064 (corrigé)

Ecrire une fonction ou procédure qui permet d'entrer deux valeurs M et N et d'afficher toutes les valeurs paires entre M et N si M < N.

Exercice 065 (corrigé)

Ecrire une fonction ou procédure qui affiche si un nombre est premier ou non

---

Exercice 066 (corrigé)

Ecrire une fonction ou procédure qui affiche tous les nombres pairs compris entre deux valeurs entières positives lue x et y

---

Exercice 067 (corrigé)

Ecrire une fonction ou procédure qui permet d'entrer la date d'aujourd'hui puis demande le nom de la personne ; si ce nom = Lutetia il y a affichage de "Bienvenue Lutetia » puis lui demande sa date d'anniversaire et la compare à la date d'aujourd'hui si c'est la même il y a affichage de "Joyeux Anniversaire Lutetia" sinon il y a affichage "erreur de personne !"

Exercice 068 (corrigé)

Ecrire une fonction ou procédure qui permet de calculer la multiplication de deux nombres A et B entiers en utilisant l'addition.

Exercice 069 (corrigé)

Ecrire une fonction ou procédure qui permet d'avoir un nombre entier positif et afficher son image miroir. Exemple le nombre est 3524, on doit afficher 4253.

Exercice 070 (corrigé)
Ecrire un algorithme (en utilisant fonction et/ou procédure) qui permet de calculer le cosinus de x € [0, Π/2] sachant que :

```
Cos(W) = 1 - P(W, 2)/2! + P(W,4)/4! - P(W,6)/6! …
```

CHAPITRE 5 : Les types construits

Les tableaux

Définition

Un ensemble de valeurs portant le même nom de variable et repérées par un nombre, s'appelle un tableau, ou encore une variable indicée.

Tab :

45	58	96	0	14	2	3	75	56	5
0	**1**	2	3	4	5	6	7	8	9

Le nombre qui, au sein d'un tableau, sert à repérer chaque valeur s'appelle **l'indice.** Chaque fois que l'on doit désigner un élément du tableau, on fait figurer le nom du tableau, suivi de l'indice de l'élément, entre croché.

Exercice 071 (corrigé)

Ecrire un algorithme qui déclare et remplisse un tableau de 7 valeurs numériques en les mettant toutes à zéro.

Exercice 072 (corrigé)

Ecrire un algorithme qui déclare et remplisse un tableau contenant les six voyelles de l'alphabet latin.

Exercice 073 (corrigé)

Ecrire un algorithme qui déclare un tableau de 9 notes, dont on fait ensuite saisir les valeurs par l'utilisateur.

Exercice 074 (corrigé)

Que produit l'algorithme suivant ?

```
Var i en Entier
    Tableau Nb[5] De Entier
Debut
    Pour i ← 0 a 5 Faire
      Nb[i] ← i * i
    Pour i ← 0 a 5 Faire
     Ecrire(Nb[i])
Fin
```

Peut-on simplifier cet algorithme avec le même résultat ?

Exercice 075 (corrigé)

Que produit l'algorithme suivant ?

```
Var i, k : Entier
     Tableau Nb[6] en Entier
Debut
     N[0] ← 1
     Pour k ← 1 a 6 Faire
        N[k] ← N[k-1] + 2

      Pour i ← 0 a 6 Faire
         Ecrire (N[i])
Fin
```

Peut-on simplifier cet algorithme avec le même résultat ?

Exercice 076 (corrigé)

Que produit l'algorithme suivant ?

```
Var i : Entier
     Tableau Suite[7] De Entier
Debut
     Suite[0] ← 1
     Suite[1] ← 1
     Pour i ← 2 a 7 Faire
        Suite[i] ← Suite[i-1] + Suite[i-2]

     Pour i ← 0 a 7 Faire
        Ecrire Suite[i]
Fin
```

Exercice 077 (corrigé)

Ecrivez la fin de l'algorithme de *l'exercice 073* afin que le calcul de la moyenne des notes soit effectué et affiché à l'écran.

Exercice 078 (corrigé)

Ecrivez un algorithme permettant à l'utilisateur de saisir un nombre quelconque de valeurs, qui devront être stockées dans un tableau. L'utilisateur doit donc commencer par entrer le nombre de valeurs qu'il compte saisir. Il effectuera ensuite cette saisie. Enfin, une fois la saisie terminée, le programme affichera le nombre de valeurs négatives et le nombre de valeurs positives.

Exercice 079 (corrigé)

Ecrivez un algorithme calculant la somme des valeurs d'un tableau (on suppose que le tableau a été préalablement saisi).

Exercice 080 (corrigé)

Ecrivez un algorithme constituant un tableau, à partir de deux tableaux de même longueur préalablement saisis. Le nouveau tableau sera la somme des éléments des deux tableaux de départ.

Tableau 1 :

4	8	7	9	1	5	4	6

Tableau 2 :

7	6	5	2	1	3	7	4

Tableau à constituer :

11	14	12	11	2	8	11	10

Exercice 081 (corrigé)

Toujours à partir de deux tableaux précédemment saisis, écrivez un algorithme qui calcule le schtroumpf des deux tableaux. Pour calculer le schtroumpf, il faut multiplier chaque élément du tableau 1 par chaque élément du tableau 2, et additionner le tout. Par exemple si l'on a :

Tableau 1 :

4	8	7	12

Tableau 2 :

3	6

Le Schtroumpf sera :
3 * 4 + 3 * 8 + 3 * 7 + 3 * 12 + 6 * 4 + 6 * 8 + 6 * 7 + 6 * 12 = 279

Exercice 082 (corrigé)

Ecrivez un algorithme qui permette la saisie d'un nombre quelconque de valeurs, sur le principe de *l'exercice 078* Toutes les valeurs doivent être ensuite augmentées de 1, et le nouveau tableau sera affiché à l'écran.

Exercice 083 (corrigé)

Ecrivez un algorithme permettant, toujours sur le même principe, à l'utilisateur de saisir un nombre déterminé de valeurs. Le programme, une fois la saisie terminée, renvoie la plus grande valeur en précisant quelle position elle occupe dans le tableau. On prendra soin d'effectuer la saisie dans un premier temps, et la recherche de la plus grande valeur du tableau dans un second temps.

Exercice 084 (corrigé)

Toujours et encore sur le même principe, écrivez un algorithme permettant, à l'utilisateur de saisir les notes d'une classe. Le programme, une fois la saisie terminée, renvoie le nombre de ces notes supérieures à la moyenne de la classe.

Exercice 085 (corrigé)

Une fonction qui permet de renvoyer la position d'un élément dans un tableau déjà saisie

---

Exercice 086 (corrigé)

Ecrire une procédure qui permet de fusionner deux tableaux de tailles différentes et déjà saisies. Dans le tableau final, on ne souhaiterait pas voir un élément se répété.

Exercice 087 (corrigé)

Ecrivez un algorithme qui permette de saisir un nombre quelconque de valeurs, et qui les range au fur et à mesure dans un tableau. Le programme, une fois la saisie terminée, doit dire si les éléments du tableau sont tous consécutifs ou non.
Par exemple, si le tableau est :

12	13	14	15	16	17	18

ses éléments sont tous consécutifs. En revanche, si le tableau est :

9	10	11	15	16	17	18

ses éléments ne sont pas tous consécutifs.

Exercice 088 (corrigé)

Ecrivez un algorithme qui trie un tableau dans l'ordre décroissant.

Vous écrirez bien entendues deux versions de cet algorithme, l'une employant le tri par sélection, l'autre le tri à bulles.

Exercice 089 (corrigé)

Ecrivez un algorithme qui inverse l'ordre des éléments d'un tableau dont on suppose qu'il a été préalablement saisi (« les premiers seront les derniers… »)

Exercice 090 (corrigé)

Ecrivez un algorithme qui permette à l'utilisateur de supprimer une valeur d'un tableau préalablement saisi. L'utilisateur donnera l'indice de la valeur qu'il souhaite supprimer. Attention, il ne s'agit pas de remettre une valeur à zéro, mais bel et bien de la supprimer du tableau lui-même ! Si le tableau de départ était :

12	8	4	45	64	9	2

Et que l'utilisateur souhaite supprimer la valeur d'indice 4, le nouveau tableau sera :

12	8	4	45	9	2

Exercice 091 (corrigé)

Ecrivez l'algorithme qui recherche un mot saisi au clavier dans un dictionnaire. Le dictionnaire est supposé être codé dans un tableau préalablement rempli et trié.

Exercice 092 (corrigé)

Écrivez un algorithme qui fusionne deux tableaux (déjà existants) dans un troisième, qui devra être trié.
Attention ! On présume que les deux tableaux de départ sont préalablement triés : il est donc irrationnel de faire une simple concaténation des deux tableaux de départ, puis d'opérer un tri : comme quand on se trouve face à deux tas de papiers déjà triés et qu'on veut les réunir, il existe une méthode bien plus économique (et donc, bien plus rationnelle...)

Les enregistrements

Exercice 093 (corrigé)

Déclarer des types qui permettent de stocker :

1. Un joueur de basket caractérisé par son nom, sa date de naissance, sa nationalité, et son sexe
2. Une association de joueurs de basket

K.L.

Exercice 094 (corrigé)

Un algorithme qui permet de comparer deux temps et afficher le plus grand.

Exercice 095 (corrigé)

a. Construire un type **Complexe** qui représente le type des nombres complexes.
b. Ecrire une procédure ou fonction **Conj()** qui prend en paramètre un nombre complexe puis renvoie son **conjugué.**
c. Ecrire une procédure ou fonction **Module()** qui prend en paramètre un nombre complexe puis renvoie son **module**.
d. Ecrire une procédure ou fonction **Som2()** qui prend en paramètre deux nombres complexes et qui permet de renvoie leur **somme.**
e. Ecrire une procédure ou fonction **Prod2()** qui prend en paramètre deux nombres complexes et qui permet de renvoie leur **Produit.**

Exercice 096 (corrigé)

Nous allons utiliser un type construit pour faciliter la solution de *l'exercice 059* page 17-18

a) Créer un type `Temps` qui représente le temps en heure H et minute M
b) La fonction `Minutes`, qui calcule le nombre des minutes correspondant à un nombre d'heures et un nombre de minutes donnés.
c) La fonction `HeuresMinutes` qui réalise la transformation inverse de la fonction Minute.
d) La fonction `AjouteTemps` qui additionne deux couples de données heures et minutes en utilisant les deux fonctions précédentes.

Exercice 097 (corrigé)

Cet exercice permet de compléter les procédures et fonctions de l'exercice précédent mais cette fois-ci en utilisant les types construits. Voir *exercice pp 18*

1. On considère qu'un mois possède un numéro d'ordre numMois, d'un nom nomMois et du nombre de jour NbJrMois. Créer le type `Mois` correspondant.

1	2	3	4	5	6	7	8	9	10	11	12
Jan	Fev	Mars	Avril	Mai	Juin	Juil	Aout	Sept	Oct	Nov	Dec
31	28/ 29	31	30	31	30	31	31	30	31	30	31

2. Créer une fonction qui permet de dire si un mois a 30 jours ou non. Cette fonction renverra Vrai si c'est le cas et Faux sinon.

3. Créer une fonction qui permet de dire si un mois a 31 jours ou non. Cette fonction renverra Vrai si c'est le cas et Faux sinon.

4. Créer une fonction qui permet de dire si une année est bissextile ou non. Cette fonction renverra Vrai si c'est le cas et Faux sinon. Pour qu'une année soit bissextile, il suffit que l'année soit un nombre divisible par 4 et non divisible par 100, ou alors qu'elle soit divisible par 400.

5. Utiliser ces fonctions pour écrire une fonction `NombreDeJour()` retournant le nombre de jours pour un mois et une année donnée.

6. Écrire un programme principal permettant à l'utilisateur d'entrer un numéro de mois (entre 1 et 12) et une année (entre 1990 et 2065), qui seront ensuite passés en paramètres à la fonction `NombreDeJour()`. Il faut tester la validité des mois et années.

Exercice 098 (corrigé)

L'idée de base du *comb sort* ou du *tri à peigne* est de comparer un élément avec un autre élément plus lointain, espacé de celui-ci d'un certain intervalle. Au départ, cet intervalle est relativement grand, puis on le réduit progressivement à chaque passe de l'algorithme jusqu'à ce qui ne soit plus que de 1 en fin de traitement. Le tri à bulles peut être considéré comme une variante du comb sort dans lequel l'intervalle est toujours de 1.

Autrement dit, la boucle interne du tri à bulle (celle qui fait la comparaison de deux éléments voisins et l'échange des valeurs si nécessaire) est modifiée pour comparer d'abord des éléments distants de la valeur de l'intervalle, puis, à chaque passe, cet intervalle est divisé par un certain coefficient, le facteur de réduction jusqu'à ce qu'il tombe à 1.

La valeur d'origine de l'intervalle est généralement la taille de la liste à trier divisée par le facteur de réduction. Des essais empiriques ont montré que la valeur idéale de ce facteur de réduction est voisine de 1,3.

Exemple :

Soit la liste présente à trier :

14	21	8	15	35	59	63	9	42	69

Ici N = 10, on prend la partie entière de 10/3 soit 7.

14	21	8	15	35	59	63	**9**	42	69

Seuls 14 et 9 sont à permuter, on calcule la partie entière de 7/1.3 soit 5, on a alors :

9	21	8	15	35	**59**	63	14	42	69

Aucun élément n'est à permuter, on calcule la partie entière de 5/1.3 soit 3, on a alors en ne visualisant que les paires à permuter

9	21	8	15	**35**	**59**	63	**14**	42	69

Deux paires sont à permuter, on calcule la partie entière de 3/1.3 soit 2, on a alors :

9	21	**8**	15	14	42	63	35	59	69

Quatre paires sont à permuter, on calcule 2/1.3, on prend 1, on a alors en ne visualisant que les paires à permuter :

8	**15**	**9**	21	14	35	59	42	63	69

Trois paires sont à permuter, on observe que l'on peut encore permuter deux nombres consécutifs :

8	9	**15**	**14**	21	35	42	59	63	69

On obtient la liste triée :

8	9	14	15	21	35	42	59	63	69

Ecrire l'algorithme du comb sort !!!!

Exercice 099 (corrigé)

Zéro d'une fonction (4 méthodes)

On recherche le zéro d'une fonction f continue sur un intervalle [a, b] telle que f(a)*f(b) < 0 ; il existe donc une racine de f dans]a, b[que nous supposerons unique.

1. Ecrire un algorithme qui détermine le zéro de cos(x) dans [1, 2] selon la méthode par dichotomie.

Indications : on pose x1 = a, x2 = b et x = (x1 + x2)/2. Si f(x1)*f(x) < 0, la racine est dans]x1, x[et on pose x2 = x ; sinon la racine est dans]x, x2[et on pose x1 = x. Puis on réitéré le procédé, la longueur de l'intervalle ayant été divisée par deux. Lorsque x1 et x2 seront suffisamment proches, on décidera que la racine est x.

2. Ecrire un algorithme qui détermine le zéro de cos(x) dans [1, 2] selon la méthode des tangentes.

Indications : soit xn une approximation de la racine c recherchée :
f(c) = f(xn) + (c − xn)f 0 (xn) ; comme f(c) = 0, on a : c = xn − f(xn)/f0 (xn).
Posons xn+1 = xn − f(xn)/f0 (xn) : on peut considérer que xn+1 est une meilleure approximation de c que xn. On recommence le procédé avec xn+1 et ainsi de suite jusqu'à ce que |xn+1 − xn| soit inférieur à un certain seuil s.

3. Ecrire un algorithme qui détermine le zéro de cos(x) dans [1, 2] selon la méthode des sécantes.

Indications : reprendre la méthode des tangentes en effectuant l'approximation suivante : f0(xn) = (f(xn) – f(xn–1))/(xn – xn–1).

4. Ecrire un algorithme qui détermine le zéro de cos(x) dans [1, 2] selon la méthode des cordes.

Indications : reprendre la méthode par dichotomie en prenant pour x le point d'intersection de la corde AB et de l'axe des abscisses :
x = (x2f(x1) – x1f(x2))/(f(x1) – f(x2)), c'est-à-dire le point obtenu par la méthode des sécantes.

Exercice 100 (corrigé)

Soit *(E) : $aX^2 + bX + c = 0$* où a, b, c sont des réels.

Ecrire un algorithme qui résout dans **C** l'équation (E).

Les corrections

Exercice 001 *(énoncé)*

Voir Cours « L'algo en Prépa »

K.L.

Exercice 002 (énoncé)

Après	La valeur des variables est :	
A ← 1	A = 1	B = ?
B ← A + 3	A = 1	B = 4
A ← 3	**A = 3**	**B = 4**

Exercice 003 (énoncé)

Après	La valeur des variables est :		
A ← 5	A = 5	B = ?	C = ?
B ← 3	A = 5	B = 3	C = ?
C ← A + B	A = 5	B = 3	C = 8
A ← 2	A = 2	B = 3	C = 8
C ← B - A	**A = 2**	**B = 3**	**C = 1**

Exercice 004 (énoncé)

Après	La valeur des variables est :	
A ← 5	A = 5	B = ?
B ← 2	A = 5	B = 2
A ← B	A = 2	B = 2
B ← A	**A = 2**	**B = 2**

Les deux dernières instructions ne permettent donc pas d'échanger les deux valeurs de B et A, puisque l'une des deux valeurs (celle de A) est ici écrasée. Si l'on inverse les deux dernières instructions, cela ne changera rien du tout, hormis le fait que cette fois c'est la valeur de B qui sera écrasée.

Exercice 005 (énoncé)

Après	La valeur des variables est :	
A ← 5	A = 5	B = ?
B ← A + 4	A = 5	B = 9
A ← A + 1	A = 6	B = 9
B ← A - 4	**A = 6**	**B = 2**

Exercice 006 (énoncé)

Après	La valeur des variables est :	
A ← 5	A = 5	B = ?
B ← A % 2	A = 5	B = 1
A ← A + 1	A = 6	B = 1
B ← A - 4	**A = 6**	**B = 2**

Exercice 007 (énoncé)

A=0 ET B=9

Exercice 008 *(énoncé)*

A= 2, B=5, C=2, D=0 ET E=2,5

Exercice 009 *(énoncé)*

Échanger les valeurs des variables A et B.

Exercice 010 (énoncé)

```
Algorithme echange
Var A,B,C: Entier
Debut
     C ← A
     A ← B
     B ← C
Fin
```

Exercice 011 *(énoncé)*

Après	La valeur des variables est :		
A ← 3	A = 3	B = ?	C = ?
B ← 10	A = 3	B = 10	C = ?
C ← A + B	A = 3	B = 10	C = 13
B ← A + B	A = 3	B = 13	C = 13
A ← C	**A = 13**	**B = 13**	**C = 13**

Exercice 012 *(énoncé)*

```
Debut
…
  C ← A
  A ← B
  B ← C
Fin
```

On est obligé de passer par une variable dite temporaire (la variable C).

Exercice 013 *(énoncé)*

```
Debut
…
  D ← C
  C ← B
  B ← A
  A ← D
Fin
```

En fait, quel que soit le nombre de variables, une seule variable temporaire suffit…

Exercice 014 *(énoncé)*

La variable C contient "42312"

Exercice 015 *(énoncé)*

Soit deux entiers a et b. On suppose que a=7 et b= 4. Quel est le résultat des instructions suivantes :

Ecrire (a < b)

Faux (car a < b est une expression dont la valeur est faux à cause du 4 <7)

Ecrire ("a < b")

a<b (chaîne de caractères)

Ecrire ('a' < 'b')

Vrai (le codage par exemple ascii de a est inférieur à celui de b.
Les entiers sont codés par des entiers dans la plupart des langages)

Exercice 016 *(énoncé)*

```
231
462
```

Exercice 017 *(énoncé)*

```
Var nb, carr : Entier
Debut
  Ecrire("Entrez un nombre :")
  Lire(nb)
  carr ← nb * nb
  Ecrire("Son carré est : ", carr)
Fin
```

En fait, on pourrait tout aussi bien économiser la variable carr en remplaçant les deux avant-dernières lignes par :

```
Ecrire("Son carré est : ", nb*nb)
```

C'est une question de style ; dans un cas, on privilégie la lisibilité de l'algorithme, dans l'autre, on privilégie l'économie d'une variable.

Exercice 018 *(énoncé)*

```
Var prenom : Caractere
Debut
  Ecrire("Quel est votre prenom ?")
  Lire(Prenom)
  Ecrire("Bonjour ", Prenom, " !")
Fin
```

Exercice 019 (énoncé)

```
Algorithme Moyenne
Var moy : Reel
Debut
     Ecrire ("Saisir la moyenne")
     Lire(moy)
     Si ((moy >=16) ET (moy<=20))
         Ecrire("Très bien")
     Fin Si
     Si ((moy >=14) ET (moy<16))
         Ecrire("Bien")
     Fin Si
     Si ((moy >=12) ET (moy< 14))
         Ecrire("Assez bien")
     Fin Si
     Si ((moy >=10) ET (moy< 12))
         Ecrire("Passable")
    Fin Si
Fin
```

Version améliorée ou imbriquée (fortement conseillée)

```
Algorithme Moyenne
Variables moy : Reel
Debut
      Ecrire ("Saisir la moyenne")
      Lire(moy)
      Si ((moy >=16) ET (moy<=20))
          Ecrire("Très bien")
      Sinon
           Si ((moy >=14) ET (moy<16))
              Ecrire("Bien")
           Sinon
              Si ((moy >=12) ET (moy< 14))
                  Ecrire("Assez bien")
              Sinon
                  Si ((moy >=10) ET (moy< 12))
                      Ecrire("Passable")
                  Fin Si
               Fin Si
            Fin Si
      Fin Si
Fin
```

Exercice 020 (énoncé)

```
Var n : Entier
Debut
  Ecrire("Entrez un nombre : ")
  Lire(n)
```

K.L.

```
  Si(n > 0) Alors
    Ecrire("Ce nombre est positif")
  Sinon
    Ecrire("Ce nombre est négatif")
  Finsi
Fin
```

Exercice 021 (énoncé)

```
Var m, n : Entier
Debut
  Ecrire("Entrez deux nombres : ")
  Lire(m, n)
  Si ((m > 0 ET n > 0) OU (m < 0 ET n < 0)) Alors
    Ecrire("Leur produit est positif")
  Sinon
    Ecrire("Leur produit est négatif")
  Finsi
Fin
```

Exercice 022 (énoncé)

```
Var a, b, c : Chaine
Debut
  Ecrire("Entrez successivement trois noms : ")
  Lire(a, b, c)
  Si( (a < b) ET (b < c)) Alors
    Ecrire("Ces noms sont classés alphabétiquement")
  Sinon
    Ecrire("Ces noms ne sont pas classés")
  Finsi
Fin
```

Exercice 023 (énoncé)

```
Var n : Entier
Debut
  Ecrire("Entrez un nombre : ")
  Lire(n)
  Si (n < 0) Alors
    Ecrire("Ce nombre est négatif")
  Sinon
    Si (n == 0) Alors
      Ecrire("Ce nombre est nul")
    Sinon
      Ecrire("Ce nombre est positif")
  Finsi
Fin
```

Exercice 024 (énoncé)

```
Var m, n : Entier
Debut
  Ecrire("Entrez deux nombres : ")
  Lire(m, n)
  Si((m == 0) OU (n == 0)) Alors
    Ecrire("Le produit est nul")
  Sinon
    Si ((m < 0 ET n < 0) OU (m > 0 ET n > 0)) Alors
      Ecrire("Le produit est positif")
    Sinon
      Ecrire("Le produit est négatif")
  Finsi
Fin
```

Si on souhaite simplifier l'écriture de la condition lourde du SinonSi, on peut toujours passer par des variables booléennes intermédiaires. Une astuce de sioux consiste également à employer un Xor (c'est l'un des rares cas dans lesquels il est pertinent)

Exercice 025 (énoncé)

```
Var age : Entier
Debut
  Ecrire("Entrez l'âge de l'enfant : ")
  Lire(age)
  Si (age >= 12) Alors
    Ecrire("Catégorie Cadet")
  Sinon
    Si (age >= 10) Alors
      Ecrire("Catégorie Minime")
    Sinon
      Si (age >= 8) Alors
        Ecrire("Catégorie Pupille")
      Sinon
        Si(age >= 6) Alors
          Ecrire("Catégorie Poussin")
  FinSi
Fin
```

On peut évidemment écrire cet algorithme de différentes façons, ne serait-ce qu'en commençant par la catégorie la plus jeune.

Exercice 026 (énoncé)

Aucune difficulté, il suffit d'appliquer la règle de la transformation du OU en ET vue en cours (loi de Morgan). Attention toutefois à la rigueur dans la transformation des conditions en leur contraire...

```
Si (Tutu <= Toto + 4 ET Tata <> "OK") Alors
  Tutu ← Tutu - 1
Sinon
  Tutu ← Tutu + 1
Finsi
```

Exercice 027 (énoncé)

```
Var h, m : Reel
Debut
  Ecrire("Entrez les heures, puis les minutes : ")
  Lire(h, m)
  m ← m + 1
  Si (m == 60) Alors
    m ← 0
    h ← h + 1
  FinSi
  Si(h == 24) Alors
    h ← 0
  FinSi
  Ecrire("Dans une minute il sera ", h, "heure(s) ", m,
"minute(s)")
Fin
```

Exercice 028 (énoncé)

```
Var h, m, s :Reel
Debut
  Ecrire("Entrez les heures, puis les minutes, puis les
secondes : ")
  Lire(h, m, s)
  s ← s + 1
  Si (s == 60) Alors
    s ← 0
    m ← m + 1
  FinSi
  Si(m == 60) Alors
    m ← 0
    h ← h + 1
  FinSi
  Si(h == 24) Alors
    h ← 0
  FinSi
  Ecrire("Dans une seconde il sera ", h, "h", m, "m et ", s,
"s")
Fin
```

Exercice 029 (énoncé)

```
Var n, p : Reel
Debut
     Ecrire ("Nombre de photocopies : ")
     Lire(n)
     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
     Ecrire("Le prix total est: ", p)
Fin
```

Exercice 030 *(énoncé)*

```
Var  sex : Caractere
     age : Entier
     C1, C2 : Booleen
Debut
     Ecrire("Entrez le sexe (M/F) : ")
     Lire(sex)
     Ecrire("Entrez l'âge: ")
     Lire(age)
     C1 ← sex == "M" ET age > 20
     C2 ← sex == "F" ET (age > 18 ET age < 35)
     Si (C1 ou C2) Alors
         Ecrire("Imposable")
     Sinon
         Ecrire("Possible")
     FinSi
Fin
```

Exercice 031 *(énoncé)*

Cet exercice, du pur point de vue algorithmique, n'est pas très méchant. En revanche, il représente dignement la catégorie des énoncés piégés.

En effet, rien de plus facile que d'écrire : si le candidat a plus de 50%, il est élu, sinon s'il a plus de 12,5 %, il est au deuxième tour, sinon il est éliminé. Hé hé hé... mais il ne faut pas oublier que le candidat peut très bien avoir eu 20 % mais être tout de même éliminé, tout simplement parce que l'un des autres a fait plus de 50 % et donc qu'il n'y a pas de deuxième tour !...
Moralité : ne jamais se jeter sur la programmation avant d'avoir soigneusement mené l'analyse du problème à traiter.

```
Algorithme CategorieElu
Var A, B, C, D : Reel
    C1, C2, C3, C4 : Booleen
Debut
     Ecrire("Entrez les scores des quatre prétendants :")
     Lire(A, B, C, D)
     C1 ← A > 50
     C2 ← B > 50 ou C > 50 ou D > 50
     C3 ← A >= B et A >= C et A >= D
```

Exercice 032 *(énoncé)*

Là encore, on illustre l'utilité d'une bonne analyse. Je propose deux corrigés différents. Le premier suit l'énoncé pas à pas. C'est juste, mais c'est vraiment lourd. La deuxième version s'appuie sur une vraie compréhension d'une situation pas si embrouillée qu'elle n'en a l'air.
Dans les deux cas, un recours aux variables booléennes aère sérieusement l'écriture.
Donc, premier corrigé, on suit le texte de l'énoncé pas à pas :

```
Algorithme Tarif
Var age, perm, acc, assur : Entier
          C1, C2, C3 : Booleen
          situ : Chaine
Debut
     Ecrire("Entrez l'âge: ")
     Lire (age)
     Ecrire ("Entrez le nombre d'années de permis: ")
     Lire (perm)
     Ecrire( "Entrez le nombre d'accidents: ")
     Lire (acc)
     Ecrire ("Entrez le nombre d'années d'assurance: ")
     Lire (assur)
     C1 ← age >= 25
     C2 ← perm >= 2
     C3 ← assur > 5
     Si Non(C1) et Non(C2) Alors
       Si acc == 0 Alors
         situ ← "Rouge"
       Sinon
         situ ← "Refusé"
       FinSi
     Sinonsi ((Non(C1) et C2) ou (C1 et Non(C2)) Alors
       Si acc == 0 Alors
         situ ← "Orange"
       SinonSi acc == 1 Alors
         situ ← "Rouge"
       Sinon
         situ ← "Refusé"
       FinSi
     Sinon
       Si acc == 0 Alors
         situ ← "Vert"
       SinonSi acc == 1 Alors
         situ ← "Orange"
       SinonSi acc == 2 Alors
         situ ← "Rouge"
       Sinon
         situ ← "Refusé"
       FinSi
     FinSi
     Si C3 Alors
       Si situ == "Rouge" Alors
         situ ← "Orange"
       SinonSi situ == "Orange" Alors
         situ ← "Vert"
       SinonSi situ == "Vert" Alors
         situ ← "Bleu"
       FinSi
```

```
            situ ← "Bleu"
          FinSi
        FinSi
        Ecrire ("Votre situation : ", situ)
Fin
```

Vous trouvez cela compliqué ? Oh, certes oui, ça l'est ! Et d'autant plus qu'en lisant entre les lignes, on pouvait s'apercevoir que ce galimatias de tarifs recouvre en fait une logique très simple : un système à points. Et il suffit de comptabiliser les points pour que tout s'éclaire... Reprenons juste après l'affectation des trois variables booléennes C1, C2, et C3. On écrit :

```
      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 ← "Bleu"
      SinonSi P == 0 Alors
        situ ← "Vert"
      SinonSi P == 1 Alors
        situ ← "Orange"
      SinonSi P == 2 Alors
        situ ← "Rouge"
      Sinon
        situ ← "Refusé"
      FinSi
      Ecrire("Votre situation : ", situ)
Fin
```

Cool, non ?

Exercice 033 *(énoncé)*

En ce qui concerne le début de cet algorithme, il n'y a aucune difficulté. C'est de la saisie bête et même pas méchante :

```
Algorithme DateValide
Var J, M, A, JMax : Reel
    VJ, VM, B : Booleen
Debut
     Ecrire("Entrez le numéro du jour")
     Lire(J)
```

```
Ecrire("Entrez le numéro du mois")
Lire (M)
Ecrire( "Entrez l'année")
Lire (A)
```

C'est évidemment ensuite que les ennuis commencent… La première manière d'aborder la chose consiste à se dire que fondamentalement, la structure logique de ce problème est très simple. Si nous créons deux variables booléennes VJ et VM, représentant respectivement la validité du jour et du mois entrés, la fin de l'algorithme sera d'une simplicité biblique (l'année est valide par définition, si on évacue le débat byzantin concernant l'existence de l'année zéro) :

```
Si VJ et VM alors
    Ecrire ("La date est valide")
Sinon
   Ecrire ("La date n'est pas valide")
FinSi
```

Toute la difficulté consiste à affecter correctement les variables VJ et VM, selon les valeurs des variables J, M et A. Dans l'absolu, VJ et VM pourraient être les objets d'une affectation monstrueuse, avec des conditions atrocement composées. Mais franchement, écrire ces conditions en une seule fois est un travail de bénédictin sans grand intérêt. Pour éviter d'en arriver à une telle extrémité, on peut sérier la difficulté en créant deux variables supplémentaires :

B : variable booléenne qui indique s'il s'agit d'une année bissextile
JMax : variable numérique qui indiquera le dernier jour valable pour le mois entré.

Avec tout cela, on peut y aller et en ressortir vivant. On commence par initialiser nos variables booléennes, puis on traite les années, puis les mois, puis les jours.

```
B ← (A % 400 == 0) ou ((non(A % 100 ==0) et A % 4==0))
Jmax ← 0
VM ← M >= 1 et M =< 12
Si VM Alors
  Si M == 2 et B Alors
    JMax ← 29
```

```
  SinonSi M == 2 Alors
    JMax ← 28
  SinonSi M == 4 ou M == 6 ou M == 9 ou M == 11 Alors
    JMax ← 30
  Sinon
    JMax ← 31
  FinSi
  VJ ← J >= 1 et J =< Jmax
FinSi
```

Cette solution a le mérite de ne pas trop compliquer la structure des tests, et notamment de ne pas répéter l'écriture finale à l'écran. Les variables booléennes intermédiaires nous épargnent des conditions composées trop lourdes, mais celles-ci restent néanmoins sérieuses.

Une approche différente consisterait à limiter les conditions composées, quitte à le payer par une structure beaucoup plus exigeante de tests imbriqués. Là encore, on évite de jouer les extrémistes et l'on s'autorise quelques conditions composées lorsque cela nous simplifie l'existence. On pourrait aussi dire que la solution précédente "part de la fin" du problème (la date est-elle valide ou non ?), alors que celle qui suit "part du début" (quelles sont les données entrées au clavier ?) :

```
Si M < 1 ou M > 12 Alors
  Ecrire ("Date Invalide")
SinonSi M = 2 Alors
  Si A % 400 ==0 Alors
    Si J < 1 ou J > 29 Alors
      Ecrire ("Date Invalide")
    Sinon
      Ecrire("Date Valide")
    FinSi
  SinonSi (A %100==0) Alors
    Si J < 1 ou J > 28 Alors
      Ecrire ("Date Invalide")
    Sinon
      Ecrire ("Date Valide")
    FinSi
  SinonSi A dp 4 Alors
    Si J < 1 ou J > 29Alors
      Ecrire ("Date Invalide")
    Sinon
      Ecrire ("Date Valide")
    FinSi
  Sinon
    Si J < 1 ou J > 28 Alors
      Ecrire ("Date Invalide")
    Sinon
      Ecrire ("Date Valide")
```

```
        Si J < 1 ou J > 28 Alors
          Ecrire ("Date Invalide")
        Sinon
          Ecrire ("Date Valide")
        FinSi
      FinSi
    SinonSi M == 4 ou M == 6 ou M == 9 ou M == 11 Alors
      Si J < 1 ou J > 30 Alors
        Ecrire ("Date Invalide")
      Sinon
        Ecrire ("Date Valide")
      FinSi
    Sinon
      Si J < 1 ou J > 31 Alors
        Ecrire ("Date Invalide")
      Sinon
        Ecrire ("Date Valide")
      FinSi
    FinSi
```

On voit que dans ce cas, l'alternative finale (Date valide ou invalide) se trouve répétée un grand nombre de fois. Ce n'est en soi ni une bonne, ni une mauvaise chose. C'est simplement une question de choix stylistique. Personnellement, j'avoue préférer assez nettement la première solution, qui fait ressortir beaucoup plus clairement la structure logique du problème (il n'y a qu'une seule alternative, autant que cette alternative ne soit écrite qu'une seule fois).

Il convient enfin de citer une solution très simple et élégante, un peu plus difficile peut-être à imaginer du premier coup, mais qui avec le recul apparaît comme très immédiate. Sur le fond, cela 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. Ainsi :

```
    B ← (A % 400 == 0) ou ((non(A % 100 ==0) et A % 4==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
```

Tout est alors réglé avec quelques variables booléennes et quelques conditions composées, en un minimum de lignes de code.

K.L.

La morale de ce long exercice - et non moins long corrigé, c'est qu'un problème de test un peu compliqué admet une pléiade de solutions justes...
...Mais que certaines sont plus astucieuses que d'autres !

Schéma itératif

Exercice 034 (énoncé)

```
Algorithme Repeter100Fois
Variable i : Entier
Debut
     Pour i ← 1 a 100 Faire
          Ecrire("je dois absolument passer l'examen de TP
          qui compte 25% de la note finale")
Fin
```

Exercice 035 (énoncé)

```
Algorithme Affiche1A100
Variable i : Entier
Debut
     Pour i ← 1 a 100 Faire
          Ecrire(i)
Fin
```

Exercice 036 (énoncé)

```
Algorithme EntierPaire1A100
Variable i : Entier
Debut
     Pour i ← 1 a 100 Faire
          Si(i%2 == 0)Alors
               Ecrire(i)
Fin
```

Exercice 037 (énoncé)

```
Algorithme SommeDeN
Variable n,SomN : Entier
Debut
     Ecrire("Saisir le nombre n svp : ")
     Lire(n)
     SomN ← n
     Pour i ← 1 a n Faire
          SomN ← SomN + 1
     Ecrire("la somme est : " ,SomN)
Fin
```

Exercice 038 (énoncé)

L'algorithme affiche les phrases suivantes :

```
i = 1
Le produit de 1 et 1 est : 1
Le produit de 1 et 2 est : 2
Le produit de 1 et 3 est : 3
i = 2
Le produit de 2 et 1 est : 2
Le produit de 2 et 2 est : 4
Le produit de 2 et 3 est : 6
```

Exercice 039 (énoncé)

```
i = 1
i = 2
Le produit de 2 et 1 est : 2
Le produit de 2 et 2 est : 4
Le produit de 2 et 3 est : 6
```

Exercice 040 (énoncé)

```
Algorithme Nombre_de_bits
Variable N,nbit,q : Entier
Debut
     Ecrire("Entrer le nombre N : ")
     Lire(N)
     q ← 1
     nbit ← 0
     TantQue N >= 1 Faire
          q ← N % 2
          N ← N // 2
          nbit ← nbit + 1
     FinTantQue
     Ecrire(("Le nombre de bits est : ", nbit)
Fin
```

Exercice 041 (énoncé)

```
Algorithme Nombre_de_1
Variable N,nb1,q : Entier
Debut
     Ecrire("Entrer le nombre N : ")
     Lire(N)
     q ← 1
     nb1 ← 0
     TantQue N >= 1 Faire
          q ← N % 2
          N ← N // 2
          Si (q == 1) Alors
               Nb1 ← nb1 + 1
     FinTantQue
     Ecrire(("Le nombre de bits est : ", nbit)
Fin
```

Exercice 042 (énoncé)

```
Algorithme BonneReponse
Variable N : Entier
Debut
     N ← 0
     Ecrire "Entrez un nombre entre 10 et 20"
     TantQue N < 10 ou N > 20
       Lire N
       Si N < 10 Alors
         Ecrire "Plus grand !"
       SinonSi N > 20 Alors
         Ecrire "Plus petit !"
       FinSi
     FinTantQue
Fin
```

Exercice 043 (énoncé)

```
Algorithme BonneReponse
Variable N : Entier
Debut
     N ← 0
     Ecrire ("Entrez un nombre entre 1 et 3")
     Repeter
       Lire (N)
         Si N < 1 ou N > 3 Alors
           Ecrire("Saisie erronée. Recommencez")
         FinSi
     JusquA( N <= 3 ou N >= 1 )
Fin
```

On convient le mieux d'utiliser la boucle Repeter.

K.L.

Exercice 044 (énoncé)

On peut imaginer deux variantes, strictement équivalentes :

```
Algorithme LaSuite
Var N, i : Entier
Debut
     Ecrire ("Entrez un nombre : ")
     Lire (N)
     Stop ← N+10
     Ecrire ("Les 10 nombres suivants sont : ")
     TantQue N < Stop Faire
        N ← N+1
        Ecrire (N)
     FinTantQue
Fin
```

Ou bien :

```
Algorithme LaSuite
Variables N, i en Entier
Debut
     Ecrire ("Entrez un nombre : ")
     Lire (N)
     i ← 0
     Ecrire ("Les 10 nombres suivants sont : ")
     TantQue i < 10 Faire
        i ← i + 1
        Ecrire (N + i)
     FinTantQue
Fin
```

Exercice 045 (énoncé)

Là encore, deux variantes, correspondant trait pour trait à celles du corrigé précédent :

```
Algorithme LaSuite
Variables N, i : Entier
Debut
     Ecrire("Entrez un nombre : ")
     Lire(N)
     Ecrire("Les 10 nombres suivants sont : ")
     Pour i ← N + 1 a N + 10 Faire
       Ecrire(i)
Fin
```

Ou bien :

```
Algorithme LaSuite
Var N, i : Entier
Debut
     Ecrire("Entrez un nombre : ")
     Lire(N)
     Ecrire("Les 10 nombres suivants sont : ")
     Pour i ← 1 a 10 Faire
       Ecrire(N + i)
Fin
```

Exercice 046 (énoncé)

```
Algorithme TableMultiplication
Var N, i : Entier
Debut
     Ecrire("Entrez un nombre : ")
     Lire(N)
     Ecrire("La table de multiplication de ce nombre est :")
     Pour i ← 1 a 10 Faire
       Ecrire(N, " x ", i, " = ", n*i)
Fin
```

Exercice 047 (énoncé)

```
Algorithme Somme
Var N, i, Som : Entier
Debut
     Ecrire("Entrez un nombre : ")
     Lire (N)
     Som ← 0
     Pour i ← 1 a N Faire
       Som ← Som + i
     Ecrire("La somme est : ", Som)
Fin
```

Exercice 048 (énoncé)

```
Algorithme Factorielle
Variables N, i, F : Entier
Debut
     Ecrire ("Entrez un nombre : ")
     Lire (N)
     F ← 1
     Pour i ← 2 a N Faire
       F ← F * i
     Ecrire ("La factorielle est : ", F)
Fin
```

Exercice 049 (énoncé)

```
Algorithme PlusGrand
Var N, i, PG : Entier
Debut
     PG ← 0
     Pour i ← 1 a 20 Faire
       Ecrire("Entrez un nombre : ")
       Lire (N)
       Si i == 1 ou N > PG Alors
         PG ← N
       FinSi
     Ecrire("Le nombre le plus grand était : ", PG)
Fin
```

En ligne 3, on peut mettre n'importe quoi dans PG, il suffit que cette variable soit affectée pour que le premier passage en ligne 7 ne provoque pas d'erreur.

Pour la version améliorée, cela donne :

```
Algorithme PlusGrand
Var N, i, PG, IPG : Entier
Debut
     PG ← 0
     Pour i ← 1 a 20
       Ecrire("Entrez un nombre : ")
       Lire(N)
       Si i == 1 ou N > PG Alors
         PG ← N
         IPG ← i
       FinSi
     Ecrire("Le nombre le plus grand était : ", PG)
     Ecrire("Il a été saisi en position numéro ", IPG)
Fin
```

Exercice 050 (énoncé)

```
Algorithme PlusGrandPosition
Var N, i, PG, IPG : Entier
Debut
     N ← 1
     i ← 0
     PG ← 0
     TantQue N <> 0 Faire
       Ecrire("Entrez un nombre : ")
       Lire(N)
       i ← i + 1
       Si i = 1 ou N > PG Alors
         PG ← N
         IPG ← i
       FinSi
     FinTantQue
     Ecrire("Le nombre le plus grand était : ", PG)
     Ecrire("Il a été saisi en position numéro ", IPG)
Fin
```

Exercice 051 (énoncé)

```
Algorithme SommeDue
Var E, somdue, M, Reste, Nb10E, Nb5E : Entier
Debut
     E ← 1
     somdue ← 0
     TantQue E <> 0 Faire
       Ecrire("Entrez le montant : ")
       Lire(E)
       somdue ← somdue + E
     FinTantQue
     Ecrire("Vous devez :", somdue, " euros")
     Ecrire("Montant versé :")
     Lire(M)
     Reste ← M - somdue
     Nb10E ← 0
     TantQue Reste >= 10 Faire
       Nb10E ← Nb10E + 1
       Reste ← Reste - 10
     FinTantQue
     Nb5E ← 0
     Si Reste >= 5
       Nb5E ← 1
       Reste ← Reste - 5
     FinSi
     Ecrire("Rendu de la monnaie :")
     Ecrire("Billets de 10 E : ", Nb10E)
```

```
     Ecrire("Billets de  5 E : ", Nb5E)
     Ecrire("Pièces de 1 E : ", reste)
Fin
```

Exercice 052 *(énoncé)*

Spontanément, on est tenté d'écrire l'algorithme suivant :

```
Algorithme ChanceTierce
Variables N, P, i, Nume, Deno1, Deno2 : Entier
Debut
     Ecrire("Entrez le nombre de chevaux partants : ")
     Lire(N)
     Ecrire("Entrez le nombre de chevaux joués : ")
     Lire(P)
     Nume ← 1
     Pour i ← 2 a N Faire
       Nume ← Nume * i

     Deno1 ← 1
     Pour i ← 2 à N-P Faire
       Deno1 ← Deno1 * i

     Deno2 ← 1
     Pour i ← 2 a P Faire
       Deno2 ← Deno2 * i
     Ecrire("Dans l'ordre, une chance sur ", Nume / Deno1)
     Ecrire("Dans le désordre, une sur ", Nume / (Deno1 *
     Deno2)
Fin
```

Cette version, formellement juste, comporte tout de même deux faiblesses.

La première, et la plus grave, concerne la manière dont elle calcule le résultat final. Celui-ci est le quotient d'un nombre par un autre ; or, ces nombres auront rapidement tendance à être très grands. En calculant, comme on le fait ici, d'abord le numérateur, puis ensuite le dénominateur, on prend le risque de demander à la machine de stocker des nombres trop grands pour qu'elle soit capable de les coder. C'est d'autant plus fou que rien ne nous oblige à procéder ainsi : on n'est pas obligé de passer par la division de deux très grands nombres pour obtenir le résultat voulu.

La deuxième remarque est qu'on a programmé ici trois boucles successives. Or, en y regardant bien, on peut voir qu'après simplification de la formule, ces trois boucles comportent le même nombre de tours ! (si vous ne me croyez pas, écrivez un exemple de calcul et biffez les nombres identiques au numérateur et au dénominateur). Ce

triple calcul (ces trois boucles) peut donc être ramené(es) à un(e) seul(e). Et voilà le travail, qui est non seulement bien plus court, mais aussi plus performant :

```
Algorithme ChanceTierce
Var N, P, i, A, B: Reel
Debut
     Ecrire("Entrez le nombre de chevaux partants : ")
     Lire(N)
     Ecrire("Entrez le nombre de chevaux joués : ")
     Lire(P )
     A ← 1
     B ← 1
     Pour i ← 1 a P Faire
       A ← A * (i + N - P)
       B ← B * i
     FinPour
     Ecrire("Dans l'ordre, une chance sur ", A )
     Ecrire("Dans le désordre, une chance sur ", A / B)
Fin
```

Les procédures et fonctions

Exercice 053 (énoncé)

```
A ← Sin(B)              #Aucun problème
A ← Sin(A + B * C)      #Aucun problème
B ← Sin(A) - Sin(D)     #Erreur ! D est en caractère
C ← Sin(A / B)          #Aucun problème… si B est différent de
zéro
C ← Cos(Sin(A)          #Erreur ! Il manque une parenthèse
fermante
```

Exercice 054 (énoncé)

```
Algorithme CompteCar
Var c : Caractere
     iC : Entier
Debut
     Ecrire("Entrez votre mot svp : ")
     iC ← 0
     Repeter
          Lire(c)
          iC ← iC + 1
     JusquA(c =="")
     Ecrire("Le nombre de caractères de votre mot est : ", iC)
Fin
```

Exercice 055 (énoncé)

Voilà un début en douceur…

```
Fonction Sum(a, b, c, d, e) : Reel
  Retourner a + b + c + d + e
FinFonction #Facultatif
```

Exercice 056 (énoncé)

```
Fonction TableauCroissant(Tableau T[] De Reel, n :
Entiere) : Booleen
Var i : Entier
      Flag : Booleen
Debut
     Flag ← Vrai
     i ← 0
     TantQue Flag Et i < n-1 Faire
        Flag ← T[i] < T[i+1]
        i ← i+1
     FinTantQue
     Retourner Flag
FinFonction
```

Exercice 057 (énoncé)

```
Procedure Inversion(X,y : Reel)
Var Temp : Reel
Debut
     Temp ← X
     X ← Y
     Y ← Temp
FinProcedure
```

Exercice 058 (énoncé)

a.

```
Fonction Minutes(M,H : Entier) : Entier
     Retourner H*60 + M
FinFonction
```

b.

```
Procédure HeuresMinutes(Duree,H,M)
Debut
     H ← Duree Div 60 # division entière
     M ← Durée - 60 * H
Fin
```

c.

```
Procédure AjouteTemps (H1, M1, H2, M2 : Entier)
Var MinuteEnTout, Hsomme , Msomme : Entier
Debut
     MinuteEnTout ← Minutes(H1, M1) + Minutes(H2, M2)
     HeuresMinutes(MinuteEnTout, Hsomme , Msomme)
Fin
```

Exercice 59 *(énoncé)*

1.

```
Fonction EstUnMoisDeTrenteJours(mois : Entier) :Entier
Debut
     Si (mois == 4 ou mois == 6 ou mois == 9 ou mois ==
11)alors
          Retourner 1
     Sinon
          Retourner 0
     Finsi
Fin
```

2.

```
Fonction EstUnMoisDeTrenteEtUnJours (mois :Entier) :Entier
Debut
     Si (mois == 1 ou mois == 3 ou mois == 5 ou mois == 7 ou
     mois == 8 ou mois == 10 ou mois == 12) alors
          Retourner 1
     Sinon
          Retourner 0
     Finsi
Fin
```

3.

```
Fonction EstUneAnneeBissextile ( annee : Entier) :Entier
Debut
     Si ((annee Mod 4 == 0 et annee % 100 <> 0) ou (annee %
     400 == 0)) alors #Année bissextile
          Retourner 1
     Sinon
          Retourner 0
     Finsi
Fin
```

Si l'année est bissextile alors le mois de février à 29 jours. Il en a 28 sinon

4.

```
Fonction Nombre_de_jours (mois ,annee :Entier) : Entier
Variable est31, est30 : Entier
Debut
     est31 ← EstUnMoisDeTrenteEtUnJours(mois)
     est30 ← EstUnMoisDeTrenteJours(mois)
     Si (est31 == 1) alors
          Retourner 31
     Sinon
          Si (est30 == 1) alors
               Retourner 30
          Sinon
          # mois de février
          # regarder si bissextile
               Si(EstUneAnneeBissextile(annee) == 1)alors
                    Retourner 29
               Sinon
                    Retourner 28
               FinSi
          FinSi
     FinSi
Fin
```

5. Ecrire un programme principal permettant à l'utilisateur d'entrer un numéro de mois (entre 1 et 12) et une année (entre 1990 et 2065), qui seront ensuite passés en paramètres à la fonction `Nombre_de_jours()`. Il faut tester la validité des mois et années.

Exercice 060 *(énoncé)*

```
Fonction PartieEntiere (x : Reel) : Entier
Var y : Entier
Debut
     y ← 0
     TantQue y < x Faire
          y ← y + 1
     FinTantQue
     Retourner y
Fin
```

Exercice 061 *(énoncé)*

```
Procedure TableMultipl (x: Entier) ;
Var a, b : Entier
Debut
     b ← 0
     Pour a ← 1 a 10 Faire
          b ← b + x
          Ecrire( x, " X " ,a, " = ", b)
     FinPour
Fin
```

Exercice 062 (énoncé)

```
Procedure calcul()
Var a, b, som, prod : Reel ;
Debut
     Lire (a, b)
     som ← a + b
     srod ← a*b
     Si(som > = 0)Alors
          Ecrire ('la somme est positive')
     Sinon
          Ecrire ('la somme est négative')
     Si(prod >= 0)Alors
          Ecrire (' et le produit est positif')
     Sinon
          Ecrire (' et le produit est négatif')
Fin
```

Exercice 063 (énoncé)

```
Procedure Nombres()
Var x, cop, co : Entier;
     pourcent : Reel ;
Debut
     cop ← 0
     co ← 0
     Repeter
          Lire(x)
          co ← co + 1
          Si(x % 2 == 0)Alors
               cop ← cop + 1
     Jusqua( x == -1)
     Pourcent ← cop*100/co
     Ecrire ("Nombre de valeurs paires = ", cop, "et leur
pourcentage = ", pourcent)
Fin
```

Exercice 064 (énoncé)

```
Procedure Nombres()
Variable M, N : Entier;
Debut
     Lire(M,N)
     Si(M >= N)Alors
          Ecrire("Pas d'affichage")
     Sinon
          TantQue M < N Faire
               Si M % 2 == 0 Alors
                    Ecrire(M)
               M ← M + 1
          FinTantQue
     FinSi
Fin
```

Exercice 065 (énoncé)

```
Procedure NombrePremier (a : Entier)
Var b : Booleen
     d : Entier
Debut
     b ← Vrai
     d ← 2
     TantQue (d <= a/2 Et b == Vrai) Faire
          Si (a % d == 0)Alors
               b ← Faux
          Sinon
               d ← d + 1
     FinTantQue
     Si(b == Vrai)Alors
          Ecrire(a, "est premier")
     Sinon
          Ecrire(a, "n'est pas premier")
Fin
```

Exercice 066 (énoncé)

```
Procedure NombrePaire()
Var x, y, z :Entier
Debut
     Lire(x,y)
     Si(x > y)Alors
          z ← x
          x ← y
          y ← z
     FinSi
     TQ(x < = y) Faire
          Si( x % 2 == 0)Alors
               Ecrire(x)
          x ← x + 1
     FinTQ
Fin
```

Exercice 067 (énoncé)

```
Procedure Bienvenue()
Var jj, mm, aa, jl, ml, al : Entier
     nom : Chaine
Debut
     Ecrire ("SVP donnez la date d'aujourd"hui")
     Lire (jj, mm, aa)
     Ecrire ("SVP quel est votre nom ? ")
     Lire (nom)
     Si nom == "Lutetia" Alors
          Ecrire ("Bienvenue Lutetia")
          Ecrire("quelle est la date de votre anniversaire ?
     ")
          Lire (jl, ml, al)
          Si (jl == jj) et (ml == mm) et (al == aa) Alors
               Ecrire ("Joyeux Anniversaire Lutetia")
     FinSi
     Sinon
          Ecrire ("Erreur de personne")
     FinSi
Fin
```

Exercice 068 (énoncé)

```
Fonction multiple (A, B : Entier) : Entier
Var Res, Y : Entier
Debut
     Res ← 0
     Si(B < 0)Alors
          Y ← -B
     Sinon
          Y ← B
     FinSi
     TQ(Y > 0)Faire
          Res ← Res + A
          Y ← Y - 1
     FinTQ
     Si(B < 0)Alors
          Res ← -Res
     Retourner Res
Fin
```

Exercice 069 (énoncé)

```
Procedure Miroir (x : Entier)
Var a, b : Entier
Debut
     a ← 0
     TQ (x <>0)Faire
          a ← x % 10
          Ecrire(a)
          x ← x // 10
     FinTQ
Fin
```

Exercice 070 (énoncé)

Pour réaliser cette fonction nous allons écrire deux fonctions générales Fact et P qui réalisent respectivement le calcul de la fonctionnelle d'un nombre entier et celle qui produit X^Y, X étant un réel et Y un entier

Pour la fonction Factorielle `Facto()` Voir cours !

```
Fonction P (X : Reel, Y : Entier) : Reel
Var Z : Entier
     R : Reel
Debut
     Si X == 0 Alors
          R ← 0
     Sinon
          R  ← 1
          Si Y < 0 Alors
               Z ← -1
          Sinon
               Z ← 1
          TQ( Y > 0) Faire
               R ← R*X
               Y ← Y - 1
          FinTQ
          Si (Z < 0)Alors
               R ← 1/R
     Retourner R
Fin
```

```
Fonction Cosinus (W : Reel, Nb : Entier) : Reel
Var F, Co, F1 : Entier
     Res, Y : Reel
Debut
     Res ← 1
     F1 ← -1
     Pour Co ← 1 a Nb Faire
          F ← Facto(Co)
          Y ← P(W,Co)
          Res ← Res + (Y/F)*F1
          F1 ← -F1
     FinPour
     Retourner Res
Fin
```

Les tableaux

Exercice 071 (énoncé)

```
Algorithme RemplirTableau
Variable i : Entier
     Tableau Truc[6] De Reel
Debut
     Pour i ← 0 a 6 Faire
        Truc(i) ← 0
Fin
```

Exercice 072 (énoncé)

```
Algorithme RemplirTableauVoyelles
Var Tableau Truc[5] : Caractere
Debut
     Truc[0] ← "a"
     Truc[1] ← "e"
     Truc[2] ← "i"
     Truc[3] ← "o"
     Truc[4] ← "u"
     Truc[5] ← "y"
Fin
```

Exercice 073 (énoncé)

```
Algorithme TableauDeNotes
Var i : Entier
     Tableau Notes[8] : Reel
Debut
     Pour i ← 0 a 8 Faire
          Ecrire("Entrez la note numéro ", i + 1)
          Lire Notes[i]
     Fin
```

Exercice 074 (énoncé)

Cet algorithme remplit un tableau avec six valeurs : 0, 1, 4, 9, 16, 25.
Il les écrit ensuite à l'écran. Simplification :

```
Algorithme TableauDeValeurs
Variable i Entier
     Tableau Nb[5] De Entier
Debut
     Pour i ← 0 à 5
          Nb(i) ← i * i
          Ecrire(Nb[i])
Fin
```

K.L.

Exercice 075 (énoncé)

Cet algorithme remplit un tableau avec les sept valeurs : 1, 3, 5, 7, 9, 11, 13.
Il les écrit ensuite à l'écran. Simplification :

```
Algorithme TableauDeValeurs
Var i, k : Entier
     Tableau N[6] De Entier
Debut
     N[0] ← 1
     Ecrire( N[0])
     Pour k ← 1 a 6 Faire
       N[k] ← N[k-1] + 2
       Ecrire (N[k])
     FinPour
Fin
```

Exercice 076 (énoncé)

Cet algorithme remplit un tableau de 8 valeurs : 1, 1, 2, 3, 5, 8, 13, 21

Exercice 077 (énoncé)

```
Algorithme CalculMoyenne
Var S : Reel
     Tableau Notes[8] De Reel
Debut
     s ← 0
     Pour i ← 0 a 8 Faire
       Ecrire("Entrez la note n° ", i + 1)
       Lire(Notes[i])
       s ← s + Notes[i]
     FinPour
     Ecrire("Moyenne :", s/9)
Fin
```

Exercice 078 (énoncé)

```
Algorithme Nb_Positif_Negatif
Var Nb, Nbpos, Nbneg : Reel
Tableau T[] De Reel
Debut
     Ecrire("Entrez le nombre de valeurs :")
     Lire(Nb)
     Redim T[Nb-1]
     Nbpos ← 0
     Nbneg ← 0
     Pour i ← 0 a Nb - 1 Faire
       Ecrire("Entrez le nombre n° ", i + 1)
       Lire (T[i])
       Si T[i] > 0 alors
         Nbpos ← Nbpos + 1
       Sinon
         Nbneg ← Nbneg + 1
       Finsi
     FinPour
     Ecrire("Nombre de valeurs positives : ", Nbpos)
     Ecrire("Nombre de valeurs négatives : ", Nbneg)
Fin
```

Exercice 079 (énoncé)

```
Algorithme SommeDesValeurs
Var i, Som, N : Entier
     Tableau T[] De Reel
Debut
# on ne programme pas la saisie du tableau, dont on suppose qu'il compte N éléments
     Redim T[N-1]
     …
     Som ← 0
     Pour i ← 0 a N - 1 Faire
       Som ← Som + T[i]
     Ecrire("Somme des éléments du tableau : ", Som)
     Fin
```

L'auteur

K.L.

Exercice 080 (énoncé)

```
Algorithme SommeDeuxTableaux
Var i, N : Entier
     Tableaux T1[], T2[], T3[] De Reel
Debut
# on suppose que T1 et T2 comptent N éléments, et qu'ils sont déjà saisis
     Redim T3[N-1]
     …
     Pour i ← 0 a N - 1 Faire
       T3[i]← T1[i] + T2[i]
Fin
```

Exercice 081 (énoncé)

```
Algorithme SommeDeuxTableaux
Var i, j, N1, N2, S : Entier
     Tableaux T1(), T2() : Reel
Debut
#On ne programme pas la saisie des tableaux T1 et T2.
#On suppose que T1 possède N1 éléments, et que T2 en possède T2)
     …
     S ← 0
     Pour i ← 0 a N1 - 1 Faire
       Pour j ← 0 a N2 - 1 Faire
         S ← S + T1[i] * T2[j]
     Ecrire("Le schtroumpf est : ", S)
Fin
```

Exercice 082 (énoncé)

```
Algorithme SommeDeuxTableaux
Var Nb, i : Entier
     Tableau T[] De Reel
Debut
     Ecrire("Entrez le nombre de valeurs : ")
     Lire(Nb)
     Redim T[Nb-1]
     Pour i ← 0 a Nb - 1 Faire
       Ecrire("Entrez le nombre n° ", i + 1]
       Lire (T[i])
     FinPour
     Ecrire("Nouveau tableau : ")
     Pour i ← 0 a Nb - 1 Faire
       T[i] ← T[i] + 1
       Ecrire T[i]
     FinPour
Fin
```

K.L.

Exercice 083 (énoncé)

```
Algorithme PlusGrandeValeur
Var Nb, Posmaxi : Entier
     Tableau T[] De Entier
Debut
     Ecrire("Entrez le nombre de valeurs :")
     Lire(Nb)
     Redim T[Nb-1]
     Pour i ← 0 a Nb - 1 Faire
       Ecrire("Entrez le nombre n° ", i + 1)
       Lire(T[i])
     FinPour
     Posmaxi ← 0
     Pour i ← 0 a Nb - 1 Faire
       Si T[i] > T[Posmaxi] alors
         Posmaxi ← i
       Finsi
     Ecrire("Element le plus grand : ", T[Posmaxi])
     Ecrire("Position de cet élément : ", Posmaxi)
Fin
```

Exercice 084 (énoncé)

```
Algorithme PlusGrandeValeurNote
Variables Nb, i, Som, Moy, Nbsup : Reel
     Tableau T[] De Reel
Debut
     Ecrire("Entrez le nombre de notes à saisir : ")
     Lire(Nb)
     Redim T[Nb-1]
     Pour i ← 0 a Nb - 1 Faire
       Ecrire("Entrez le nombre n° ", i + 1)
       Lire(T[i])
     FinPour
     Som ← 0
     Pour i ← 0 a Nb - 1 Faire
       Som ← Som + T[i]
     Moy ← Som / Nb
     NbSup ← 0
     Pour i ← 0 a Nb - 1 Faire
       Si T(i) > Moy Alors
         NbSup ← NbSup + 1
       FinSi
     Ecrire(NbSup, " élèves dépassent la moyenne de la
     classe")
Fin
```

Exercice 085 (énoncé)

```
Fonction PositionValeur(Tableau T[n] : Entier,
Val :Entier) : Entier
Var i : Entier
      Vu : Booleen
Debut
      Vu ← Faux
      i ← 1
      TantQue Non Vu Et i <= n Faire
            Si Val == T[i] Alors
                  Vu ← Vrai
                  posI ← i
            FinSi
            i ← i +1
      FinTantQue
      Si(Vu)Alors
            Retourner posI
Fin
```

Exercice 086 (énoncé)

```
Procedure Fusion(Tableau S[n],R[m], T[m+n] : De Entier) :Tableau
Var i : Entier
Debut
      Pour i ← 1 a n Faire
            Si(Present(S[i],T)== Faux)Alors
                  T[i] ← S[i]
      Pour i ← n+1 a n+1+m Faire
            Si(Present(R[i],T)== Faux)Alors
                  T[i] ← S[i]
Fin
# Cette fonction vérifie la présence d'une valeur dans un
# Tableau
Fonction Present(Val : Entier, Tableau T[n] : De Entier) : Booleen
Var i : Entier
      Vu : Booleen
Debut
      Vu ← Faux
      i ← 1
      TantQue Non Vu Et i <= n Faire
            Si Val == T[i] Alors
                  Vu ← Vrai
            FinSi
            i ← i +1
      FinTantQue
      Retourner Vu
Fin
```

Exercice 087 (énoncé)

```
Algorithme Consecutifs
Var Nb, i : Entier
     Flag : Booleen
     Tableau T[] : Entier
Debut
     Ecrire("Entrez le nombre de valeurs :")
     Lire(Nb)
     Redim T[Nb-1]
     Pour i ← 0 a Nb - 1 Faire
       Ecrire("Entrez le nombre n° ", i + 1)
       Lire(T[i])
     FinPour
     Flag ← Vrai
     Pour i ← 1 a Nb - 1 Faire
       Si T[i] <> T[i - 1] + 1 Alors
         Flag ← Faux
       FinSi
     Si Flag Alors
       Ecrire("Les nombres sont consécutifs")
     Sinon
       Ecrire("Les nombres ne sont pas consécutifs")
     FinSi
Fin
```

Cette programmation est sans doute la plus spontanée, mais elle présente le défaut d'examiner la totalité du tableau, même lorsqu'on découvre dès le départ deux éléments non consécutifs. Aussi, dans le cas d'un grand tableau, est-elle dispendieuse en temps de traitement. Une autre manière de procéder serait de sortir de la boucle dès que deux éléments non consécutifs sont détectés. La deuxième partie de l'algorithme deviendrait donc :

```
     i ← 1
     TantQue T[i] == T[i - 1] + 1 et i < Nb - 1 Faire
       i ← i + 1
     FinTantQue
     Si T[i] == T[i - 1] + 1 Alors
       Ecrire("Les nombres sont consécutifs")
     Sinon
       Ecrire("Les nombres ne sont pas consécutifs")
     FinSi
```

Exercice 088 (énoncé)

On suppose que N est le nombre d'éléments du tableau. Tri par insertion :

```
…
Pour i ← 0 a N – 2 Faire
  posmaxi ← i
  Pour j ← i + 1 a N – 1 Faire
    Si t[j] > t[posmaxi] alors
      posmaxi ← j
    Finsi
  temp ← t[posmaxi]
  t[posmaxi] ← t[i]
  t[i] ← temp
FinPour
Fin
```

Tri à bulles :

```
…
Yapermut ← Vrai
TantQue Yapermut Faire
  Yapermut ← Faux
  Pour i ← 0 a N – 2 Faire
    Si t[i] < t[i + 1] Alors
      temp ← t[i]
      t[i] ← t[i + 1]
      t[i + 1] ← temp
      Yapermut ← Vrai
    Finsi
FinTantQue
Fin
```

Exercice 089 (énoncé)

On suppose que n est le nombre d'éléments du tableau préalablement saisi

```
…
Pour i ← 0 a (N-1)/2 Faire
  Temp ← T[i]
  T[i] ← T[N-1-i]
  T[N-1-i] ← Temp
FinPour
Fin
```

Exercice 090 (énoncé)

```
      …
      Ecrire("Rang de la valeur à supprimer ?")
      Lire(S)
      Pour i ← S a N-2 Faure
        T[i] ← T[i+1]
      i suivant
      Redim T[N-1]
```

Exercice 091 (énoncé)

N est le nombre d'éléments du tableau Dico[], contenant les mots du dictionnaire, tableau préalablement rempli.

```
Algorithme RechercheMot
Var Sup, Inf, Comp : Entier
     Fini : Booleen
Debut
     Ecrire("Entrez le mot à vérifier")
     Lire(Mot)
#On définit les bornes de la partie du tableau à considérer
     Sup ← N - 1
     Inf ← 0
     Fini ← Faux
     TantQue Non Fini Faire
#Comp désigne l'indice de l'élément à comparer. En bonne
rigueur, il faudra #veiller à ce que Comp soit bien un
nombre entier, ce qui pourra s'effectuer de #différentes
manières selon les langages.
          Comp ← (Sup + Inf)/2
#Si le mot se situe avant le point de comparaison, alors la
borne supérieure #change, la borne inférieure ne bouge pas.
          Si Mot < Dico[Comp] Alors
               Sup ← Comp - 1
          #Sinon, c'est l'inverse
           Sinon
               Inf ← Comp + 1
          FinSi
          Fini ← Mot == Dico[Comp] ou Sup < Inf
     FinTantQue
     Si Mot == Dico[Comp] Alors
       Ecrire( "le mot existe")
     Sinon
       Ecrire( "Il n'existe pas")
     Finsi
Fin
```

Exercice 092 (énoncé)

Les deux tableaux de départ, A[m] et B[n}, sont déjà triés : pas question donc de les empiler simplement pour se relancer dans un (long) tri. On prend simplement les deux tableaux, et on avance dans l'un puis dans l'autre selon celui des deux éléments auquel on est parvenu est le plus petit (il suffit de s'imaginer devant deux tas de papiers triés par date, et de vouloir constituer un tas unique, pour comprendre ce qu'on va faire). Le truc est qu'on ne sait pas par avance où on va en être à un moment donné dans un tableau et dans l'autre : il nous faut donc deux compteurs différents pour noter notre position dans chacun des deux tableaux. On appelle C le tableau de destination, et ic la variable qui indique où on en est dans celui-ci.

```
Debut
      (...)
      Afini ← faux
      Bfini ← faux
      ia ← 0
      ib ← 0
      ic ← -1
      TantQue Non(Afini) ou Non(Bfini) Faire
         ic ← ic + 1
         Redim C[ic]
         Si Afini ou A[ia]>B[ib] Alors
            C[ic] ← B[ib]
            ib ← ib + 1
            Bfini ← ib > n
         Sinon
            C[ic] ← A[ia]
            ia ← ia + 1
            Afini ← ia > m
         FinSi
      FinTantQue
Fin
```

Les enregistrements

Exercice 093 *(énoncé)*

1.

```
Type :
      Structure JoueurBasket
            Nom : Chaine
            DateNaissance : Chaine
            Nationalite : Chaine
            Sexe : Chaine
      FinStructure
```

2.

```
Type :
     Structure Association
          Tableau Asso [] : De JoueurBasket
     FinStructure
```

Exercice 094 (énoncé)

```
Algorithme ComparerTemps
Type :
     Structure Temps
          H : Entier
          M : Entier
     FinStructure

Var X,Y :Temps

Fonction CompareTemps(T,t : Temps) :Temps
Debut
     Si(T.H < t.H)Alors
          Retourner t
     Sinon
          Si(T.H > t.H)Alors
               Retourner T
          Sinon
               Si(T.M < t.M)Alors
                    Retourner t
               Sinon
                    Retourner T
               FinSi
          FinSi
     FinSi
Fin

Debut
     Ecrire("Entrez le temps 1 : ")
     Lire(X.H,X.M)
     Ecrire("Entrez le temps 2 : ")
     Lire(Y.H,Y.M)
     Ecrire("Le temps le plus grand est : ",CompareTemps(T,t))
Fin
```

Exercice 095 (énoncé)

a.

```
Type :
     Structure Complexe
          Re : Reel
          Im : Reel
     FinStructure
```

b.

```
Fonction Conj(Z : Complexe) :Complexe
Debut
     Z.Im ← -Z.Im
     Retourner Z
Fin
```

c.

```
Fonction Module(Z : Complexe) : Reel
Debut
     Retourner Sqrt(Z.Re**2 + Z.Im**2)
Fin
```

d.

```
Fonction Som2(Z,W : Complexe) :Complexe
Var X : Complexe
Debut
     X.Re ← Z.Re + W.Re
     X.Im ← Z.Im + W.Im
     Retourner X
Fin
```

e.

```
Fonction Prod2(Z,W : Complexe) : Complexe
Var X : Complexe
Debut
     X.Re ← Z.Re*W.Re - Z.Im*W.Im
     X.Im ← Z.Re*W.Im + Z.Im*W.Re
     Retourner X
Fin
```

Pour afficher un complexe :

```
     Ecrire(X.Re, " + ",X.Im, "i")
```

Utiliser un Si pour bien afficher le cas où la partie imaginaire est positive ou négative.

Exercice 096 (énoncé)

a) Créer un type `Temps` qui représente le temps en heure H et minute M

```
Type :
     Structure Temps
          H : Entier
          M : Entier
     FinStructure
```

b) La fonction `Minutes`, qui calcule le nombre des minutes correspondant à un nombre d'heures et un nombre de minutes donnés.

```
Fonction Minutes(T :Temps) :Entier
     Retourner T.H*60 + T.M
```

c) La fonction `HeuresMinutes` qui réalise la transformation inverse de la fonction Minute.

```
Fonction HeuresMinutes(Duree :Entier) :Temps
Var T : Temps
Debut
     T.H ← Duree//60 #Division entière
     T.M ← Duree % 60
     Retourner T
Fin
```

d) La fonction `AjouteTemps` qui additionne deux couples de données heures et minutes en utilisant les deux fonctions précédentes.

```
Fonction AjouteTemps(T1, T2 : Temps) :Temps
Var Hminutes : Temps
     mins : Entier
Debut
     mins ← Minutes(T1) + Minutes(T2)
     Hminutes ← HeuresMinutes(mins)
     Retourner Hminutes
Fin
```

Exercice 097 (énoncé)

1. Créer le type `Mois` correspondant.

```
Type :
     Structure Mois
          numMois : Entier
          nomMois : Chaine
          NbJrMois : Entier
     FinStructure
```

2.

```
Fonction EstUnMoisDeTrenteJours(mois : Mois) :Booleen
Debut
     Si(mois.numMois == 4 ou mois.numMois == 6 ou
     mois.numMois == 9 ou mois.numMois == 11)alors
          Retourner Vrai
     Sinon
          Retourner Faux
     Fsi
Fin
```

3.

```
Fonction EstUnMoisDeTrenteEtUnJours (mois :Mois) :Booleen
Debut
     Si (mois.numMois == 1 ou mois.numMois == 3 ou
     mois.numMois == 5 ou mois.numMois == 7 ou mois.numMois
     == 8 ou mois.numMois == 10 ou mois.numMois == 12) alors
          Retourner Vrai
     Sinon
          Retourner Faux
     Fsi
Fin
```

4.

```
Fonction EstUneAnneeBissextile ( annee : Entier) :Entier
Debut
     Si ((annee Mod 4 == 0 et annee % 100 <> 0) ou (annee %
     400 == 0)) alors #Année bissextile
          Retourner Vrai
     Sinon
          Retourner Faux
     Fsi
Fin
```

5.

```
Fonction Nombre_de_jours (mois : Mois,annee :Entier) : Entier
Variable est31, est30 : Booleen
Debut
     est31 ←EstUnMoisDeTrenteEtUnJours(mois)
     est30 ←EstUnMoisDeTrenteJours(mois)
     Si (est31) alors
          Retourner 31
     Sinon
          Si (est30) alors
               Retourner 30
          Sinon
          # mois de février
          # regarder si bissextile
               Si(EstUneAnneeBissextile(annee))alors
                    Retourner 29
               Sinon
                    Retourner 28
               FinSi
          FinSi
     FinSi
Fin
```

6. Ecrire un programme principal permettant à l'utilisateur d'entrer un numéro de mois (entre 1 et 12) et une année (entre 1990 et 2065), qui seront ensuite passés en paramètres à la fonction `Nombre_de_jours()`. Il faut tester la validité des mois et années.

Exercice 098 *(énoncé)*

```
Fonction combsort(Tableau liste[n] :De Entier) : Tableau De Entier
Var i, intervalle
       echange : Booleen
Debut
    intervalle ← n #initialisation de l'intervalle
    TantQue intervalle > 1 OU echange == Vrai Faire
        intervalle ← Entier(intervalle / 1.3) #Partie entière
        Si intervalle < 1 Alors
          intervalle ← 1
        FinSi
        i ← 0
        echange ← faux

        TantQue i ≤ n - intervalle Faire
            Si liste[i] > liste[i + intervalle] Alors
```

```
                Echanger(liste[i], liste[i + intervalle])
                echange ← vrai
            FinSi
            i ← i + 1
        FinTantQue
    FinTantQue
    Retourner liste
Fin
```

Exercice 099 (énoncé)

1. Méthode par dichotomie

```
Fonction RechercheDicotoCosinus(x1, x2,s : Reel) :Reel
Var x : Reel
Debut
     x ← (x1 + x2)/2
     s ← 0.0000000001
     TQ x2 - x1 > s Faire
          Si Cos(x)*Cos(x1) < 0 Alors
               x2 ←  x
          Sinon
               x1 ← x
          x ← (x1 + x2)/2
     FinTQ
     Retourner x
Fin
# RechercheDicotoCosinus(1,2,0.000000001)
```

2. Méthode des tangentes

```
Fonction TangenteCosinus(x1, x2, s: Reel) :Reel
Var x : Reel
Debut
     x ← x2 Cos(x2)/(-Sin(x2)) # f(x)=cos(x) df(x) = sin(x)
     TQ abs(x - x2) > s Faire
          x2 ←  x
          x ← x - Cos(x)/(-Sin(x))
     FinTQ
     Retourner x
Fin
```

3. Method des secants

```
Fonction TangenteCosinus(x1, x2, s: Reel) :Reel
Var x : Reel
Debut
     df ← (Cos(x2) - Cos(x1))/(x2 - x1)
     x ← x2 - Cos(x2)/df
     TQ abs(x - x2) > s Faire
          x2 ←  x
          df ← (Cos(x2) - Cos(x1))/(x2 - x1)
          x ← x - Cos(x)/df
     FinTQ
     Retourner x
Fin
```

4. Méthode des cordes

```
Fonction TangenteCosinus(x1, x2, s: Reel) :Reel
Var x : Reel
Debut
     x ← (x2*Cos(x1)-x1*Cos(x2))/(Cos(x1)-Cos(x2))
     TQ abs(x2 - x1) > s Faire
          Si (Cos(x1)*Cos(x)<0 Alors
               x2 ← x
          Sinon
               x1 ← x
          x ← (x2*Cos(x1)-x1*Cos(x2))/(Cos(x1)-Cos(x2))
     FinTQ
     Retourner x
Fin
```

Exercice 100 (énoncé)

```
Algorithme EquationDegre2C
Var a,b,c,Dis,Re,Im : Reel
Debut
     Ecrire("Entrer les trois coefficients svp : ")
     Lire(a,b,c)
     Si(a==0)Alors
          Si(b==a)Alors
               Ecrire("Pas de solution")
          Sinon
               Ecrire("la solution est :",-c/b)
          FinSi
     Sinon
          Dis ← b**2 - 4*a*c
          Re ← -b/(2*a)
          Si(Dis < 0)Alors
               Im ← Sqrt(-Dis)/(2*a)
               Ecrire("la solution 1 : ", Re, "+",Im, "i")
               Ecrire("la solution 2 : ", Re,Im, "i")
          Sinon
               Si(Dis == 0)Alors
                    Ecrire("la solution est double : ",Re)
               Sinon
                    Im ← Sqrt(Dis)/(2*a)
                    Ecrire("la solution 1 : ", Re, "+",Im)
                    Ecrire("la solution 2 : ", Re,Im)
               FinSi
          FinSin
     FinSi
Fin
```