ENONCE DES EXERCICES

Exercice 9.1

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

Variables A, B, C en Numérique

Variables D, E en Caractère

A ← Sin(B)

A ← Sin(A + B * C)

B ← Sin(A) – Sin(D)

D ← Sin(A / B)

C ← Cos(Sin(A)

Exercice 9.2

Ecrivez un algorithme qui demande un mot à l'utilisateur et qui affiche à l'écran le nombre de lettres de ce mot (c'est vraiment tout bête).

Exercice 9.3

Ecrivez un algorithme qui demande une phrase à l'utilisateur et qui affiche à l'écran le nombre de mots de cette phrase. On suppose que les mots ne sont séparés que par des espaces (et c'est déjà un petit peu moins bête).

Exercice 9.4

Ecrivez un algorithme qui demande une phrase à l'utilisateur et qui affiche à l'écran le nombre de voyelles contenues dans cette phrase.

On pourra écrire deux solutions. La première déploie une condition composée bien fastidieuse. La deuxième, en utilisant la fonction Trouve, allège considérablement l'algorithme.

Exercice 9.5

Ecrivez un algorithme qui demande une phrase à l'utilisateur. Celui-ci entrera ensuite le rang d'un caractère à supprimer, et la nouvelle phrase doit être affichée (on doit

réellement supprimer le caractère dans la variable qui stocke la phrase, et pas uniquement à l'écran).

Exercice 9.6 - Cryptographie 1

Un des plus anciens systèmes de cryptographie (aisément déchiffrable) consiste à décaler les lettres d'un message pour le rendre illisible. Ainsi, les A deviennent des B, les B des C, etc. Ecrivez un algorithme qui demande une phrase à l'utilisateur et qui la code selon ce principe. Comme dans le cas précédent, le codage doit s'effectuer au niveau de la variable stockant la phrase, et pas seulement à l'écran.

Exercice 9.7 - Cryptographie 2 - *le chiffre de César*

Une amélioration (relative) du principe précédent consiste à opérer avec un décalage non de 1, mais d'un nombre quelconque de lettres. Ainsi, par exemple, si l'on choisit un décalage de 12, les A deviennent des M, les B des N, etc.

Réalisez un algorithme sur le même principe que le précédent, mais qui demande en plus quel est le décalage à utiliser. Votre sens proverbial de l'élégance vous interdira bien sûr une série de vingt-six "Si...Alors"

Exercice 9.8 - Cryptographie 3

Une technique ultérieure de cryptographie consista à opérer non avec un décalage systématique, mais par une substitution aléatoire. Pour cela, on utilise un alphabet-clé, dans lequel les lettres se succèdent de manière désordonnée, par exemple :

HYLUJPVREAKBNDOFSQZCWMGITX

C'est cette clé qui va servir ensuite à coder le message. Selon notre exemple, les A deviendront des H, les B des Y, les C des L, etc.

Ecrire un algorithme qui effectue ce cryptage (l'alphabet-clé sera saisi par l'utilisateur, et on suppose qu'il effectue une saisie correcte).

Exercice 9.9 - Cryptographie 4 - *le chiffre de Vigenère*

Un système de cryptographie beaucoup plus difficile à briser que les précédents fut inventé au XVIe siècle par le français Vigenère. Il consistait en une combinaison de différents chiffres de César.

On peut en effet écrire 25 alphabets décalés par rapport à l'alphabet normal :

- l'alphabet qui commence par B et finit par …YZA
- l'alphabet qui commence par C et finit par …ZAB

- etc.

Le codage va s'effectuer sur le principe du chiffre de César : on remplace la lettre d'origine par la lettre occupant la même place dans l'alphabet décalé.

Mais à la différence du chiffre de César, un même message va utiliser non un, mais plusieurs alphabets décalés. Pour savoir quels alphabets doivent être utilisés, et dans quel ordre, on utilise une clé.

Si cette clé est "VIGENERE" et le message "Il faut coder cette phrase", on procèdera comme suit :

La première lettre du message, I, est la 9e lettre de l'alphabet normal. Elle doit être codée en utilisant l'alphabet commençant par la première lettre de la clé, V. Dans cet alphabet, la 9e lettre est le D. I devient donc D.

La deuxième lettre du message, L, est la 12e lettre de l'alphabet normal. Elle doit être codée en utilisant l'alphabet commençant par la deuxième lettre de la clé, I. Dans cet alphabet, la 12e lettre est le S. L devient donc S, etc.

Quand on arrive à la dernière lettre de la clé, on recommence à la première.

Ecrire l'algorithme qui effectue un cryptage de Vigenère, en demandant bien sûr au départ la clé à l'utilisateur.

Exercice 9.10

Ecrivez un algorithme qui demande un nombre entier à l'utilisateur. L'ordinateur affiche ensuite le message "Ce nombre est pair" ou "Ce nombre est impair" selon le cas.

Exercice 9.11

Ecrivez les algorithmes qui génèrent un nombre Glup aléatoire tel que …

- 0 =< Glup < 2
- –1 =< Glup < 1
- 1,35 =< Glup < 1,65
- Glup émule un dé à six faces
- –10,5 =< Glup < +6,5
- Glup émule la somme du jet simultané de deux dés à six faces

CORRIGES DES EXERCICES

Exercice 9.1

A ← Sin(B) Aucun problème

A ← Sin(A + B * C) Aucun problème

B ← Sin(A) – Sin(D) Erreur ! D est en caractère

D ← Sin(A / B) Aucun problème… si B est différent de zéro

C ← Cos(Sin(A) Erreur ! Il manque une parenthèse fermante

Exercice 9.2

Vous étiez prévenus, c'est bête comme chou ! Il suffit de se servir de la fonction Len, et c'est réglé :

Variable Mot **en Caractère**
Variable Nb **en Entier**
Debut
Ecrire "Entrez un mot : "
Lire Mot
Nb ← Len(Mot)
Ecrire "Ce mot compte ", Nb, " lettres"
Fin

Exercice 9.3

Là, on est obligé de compter par une boucle le nombre d'espaces de la phrase, et on en déduit le nombre de mots. La boucle examine les caractères de la phrase un par un, du premier au dernier, et les compare à l'espace.

Variable Bla **en Caractère**
Variables Nb, i **en Entier**
Debut
Ecrire "Entrez une phrase : "
Lire Bla
Nb ← 0
Pour i ← 1 à Len(Bla)
 Si Mid(Bla, i , 1) = " " **Alors**
  Nb ← Nb + 1
 FinSi

i **suivant**

Ecrire "Cette phrase compte ", Nb + 1, " mots"

Fin

Exercice 9.4

Solution 1 : pour chaque caractère du mot, on pose une très douloureuse condition composée. Le moins que l'on puisse dire, c'est que ce choix ne se distingue pas par son élégance. Cela dit, il marche, donc après tout, pourquoi pas.

Variable Bla **en Caractère**

Variables Nb, i, j **en Entier**

Debut

Ecrire "Entrez une phrase : "

Lire Bla

Nb ← 0

Pour i ← 1 à Len(Bla)

Si Mid(Bla, i, 1) = "a" ou Mid(Bla, i, 1) = "e" ou Mid(Bla, i, 1) = "i" ou Mid(Bla, i, 1) = "o" ou Mid(Bla, i, 1) = "u" ou Mid(Bla, i, 1) = "y" **Alors**

Nb ← Nb + 1

FinSi

i **suivant**

Ecrire "Cette phrase compte ", Nb, " voyelles"

Fin

Solution 2 : on stocke toutes les voyelles dans une chaîne. Grâce à la fonction Trouve, on détecte immédiatement si le caractère examiné est une voyelle ou non. C'est nettement plus sympathique...

Variables Bla, Voy **en Caractère**

Variables Nb, i, j **en Entier**

Debut

Ecrire "Entrez une phrase : "

Lire Bla

Nb ← 0

Voy ← "aeiouy"

Pour i ← 1 à Len(Bla)

Si Trouve(Voy, Mid(Bla, i, 1)) <> 0 **Alors**

Nb ← Nb + 1

FinSi

i **suivant**

Ecrire "Cette phrase compte ", Nb, " voyelles"

Fin

Exercice 9.5

Il n'existe aucun moyen de supprimer directement un caractère d'une chaîne... autrement qu'en procédant par collage. Il faut donc concaténer ce qui se trouve à gauche du caractère à supprimer, avec ce qui se trouve à sa droite. Attention aux paramètres des fonctions Mid, ils n'ont rien d'évident !

Variable Bla **en Caractère**

Variables Nb, i, j **en Entier**

Début

Ecrire "Entrez une phrase : "

Lire Bla

Ecrire "Entrez le rang du caractère à supprimer : "

Lire Nb

L ← Len(Bla)

Bla ← Mid(Bla, 1, Nb – 1) & Mid(Bla, Nb + 1, L – Nb)

Ecrire "La nouvelle phrase est : ", Bla

Fin

Exercice 9.6

Sur l'ensemble des exercices de cryptographie, il y a deux grandes stratégies possibles :

- soit transformer les caractères en leurs codes ASCII. L'algorithme revient donc ensuite à traiter des nombres. Une fois ces nombres transformés, il faut les reconvertir en caractères.

- soit en rester au niveau des caractères, et procéder directement aux transformations à ce niveau. C'est cette dernière option qui est choisie ici, et pour tous les exercices de cryptographie à venir.

Pour cet exercice, il y a une règle générale : pour chaque lettre, on détecte sa position dans l'alphabet, et on la remplace par la lettre occupant la position suivante. Seul cas particulier, la vingt-sixième lettre (le Z) doit être codée par la première (le A), et non par la vingt-septième, qui n'existe pas !

Variables Bla, Cod, Alpha **en Caractère**

Variables i, Pos **en Entier**

```
Début
Ecrire "Entrez la phrase à coder : "
Lire Bla
Alpha ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
Cod ← ""
Pour i ← 1 à Len(Bla)
 Let ← Mid(Bla, i, 1)
 Si Let <> "Z" Alors
  Pos ← Trouve(Alpha, Let)
  Cod ← Cod & Mid(Alpha, Pos + 1, 1)
 Sinon
  Cod ← Cod & "A"
 FinSi
i Suivant
Bla ← Cod
Ecrire "La phrase codée est : ", Bla
Fin
```

Exercice 9.7

Cet algorithme est une généralisation du précédent. Mais là, comme on ne connaît pas d'avance le décalage à appliquer, on ne sait pas a priori combien de "cas particuliers", à savoir de dépassements au-delà du Z, il va y avoir.
Il faut donc trouver un moyen simple de dire que si on obtient 27, il faut en réalité prendre la lettre numéro 1 de l'alphabet, que si on obtient 28, il faut en réalité prendre la numéro 2, etc. Ce moyen simple existe : il faut considérer le reste de la division par 26, autrement dit le modulo.
Il y a une petite ruse supplémentaire à appliquer, puisque 26 doit rester 26 et ne pas devenir 0.

```
Variable Bla, Cod, Alpha en Caractère
Variables i, Pos, Décal en Entier
Début
Ecrire "Entrez le décalage à appliquer : "
Lire Décal
Ecrire "Entrez la phrase à coder : "
Lire Bla
Alpha ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

```
Cod ← ""
Pour i ← 1 à Len(Bla)
  Let ← Mid(Bla, i, 1)
  Pos ← Trouve(Alpha, Let)
  NouvPos ← Mod(Pos + Décal, 26)
  Si NouvPos = 0 Alors
    NouvPos ← 26
  FinSi
  Cod ← Cod & Mid(Alpha, NouvPos, 1)
i Suivant
Bla ← Cod
Ecrire "La phrase codée est : ", Bla
Fin
```

Exercice 9.8

Là, c'est assez direct.

```
Variable Bla, Cod, Alpha en Caractère
Variables i, Pos, Décal en Entier
Début
Ecrire "Entrez l'alphabet clé : "
Lire Clé
Ecrire "Entrez la phrase à coder : "
Lire Bla
Alpha ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
Cod ← ""
Pour i ← 1 à Len(Bla)
  Let ← Mid(Bla, i, 1)
  Pos ← Trouve(Alpha, Let)
  Cod ← Cod & Mid(Clé, Pos, 1)
i Suivant
Bla ← Cod
Ecrire "La phrase codée est : ", Bla
Fin
```

Exercice 9.9

Le codage de Vigenère n'est pas seulement plus difficile à briser; il est également un peu plus raide à programmer. La difficulté essentielle est de comprendre qu'il faut deux boucles: l'une pour parcourir la phrase à coder, l'autre pour parcourir la clé. Mais quand on y réfléchit bien, ces deux boucles ne doivent surtout pas être imbriquées. Et en réalité, quelle que soit la manière dont on l'écrit, elle n'en forment qu'une seule.

Variables Alpha, Bla, Cod, Clé, Let **en Caractère**
Variables i, Pos, PosClé, Décal **en Entier**
Début
Ecrire "Entrez la clé : "
Lire Clé
Ecrire "Entrez la phrase à coder : "
Lire Bla
Alpha ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
Cod ← ""
PosClé ← 0
Pour i ← 1 à Len(Bla)

On gère la progression dans la clé. J'ai effectué cela "à la main" par une boucle, mais un joli emploi de la fonction Modulo aurait permis une programmation en une seule ligne!

Posclé ← Posclé + 1
Si PosClé > Len(Clé) **Alors**
PosClé ← 1
FinSi

On détermine quelle est la lettre clé et sa position dans l'alphabet

LetClé ← Mid(Clé, PosClé, 1)
PosLetClé ← Trouve(Alpha, LetClé)

On détermine la position de la lettre à coder et le décalage à appliquer. Là encore, une solution alternative aurait été d'employer Mod : cela nous aurait épargné le Si…

Let ← Mid(Bla, i, 1)
Pos ← Trouve(Alpha, Let)
NouvPos ← Pos + PosLetClé
Si NouvPos > 26 **Alors**
NouvPos ← NouvPos – 26
FinSi

```
  Cod ← Cod & Mid(Alpha, NouvPos, 1)
i Suivant
Bla ← Cod
Ecrire "La phrase codée est : ", Bla
Fin
```

Exercice 9.10

On en revient à des choses plus simples...

```
Variable Nb en Entier
Ecrire "Entrez votre nombre : "
Lire Nb
Si Nb/2 = Ent(Nb/2) Alors
  Ecrire "Ce nombre est pair"
Sinon
  Ecrire "Ce nombre est pair"
FinSi
Fin
```

Exercice 9.11

a) Glup ← Alea() * 2

b) Glup ← Alea() * 2 - 1

c) Glup ← Alea() * 0,30 + 1,35

d) Glup ← Ent(Alea() * 6) + 1

e) Glup ← Alea() * 17 – 10,5

f) Glup ← Ent(Alea()*6) + Ent(Alea()*6) + 2