2 QUELQUES EXERCICES

Tous les énoncés repris dans la liste ci–après proviennent d'applications concrètes utilisées essentiellement en bureautique. Ils ont été adaptés, généralisés et parfois simplifiés.

Les exercices nécessitent les prérequis et les logiciels suivants:

•prérequis

_une connaissance avancée des actions de base d'un traitement de texte,

_une connaissance des concepts de mise en page à l'aide des balises (feuille de styles),

_une connaissance des concepts de macro–instructions enregistrées

_une connaissance des concepts des macro–instructions "programme",

_une connaissance des actions de base et des structures de contrôle du paradigme des langages de programmation procéduraux,

_Tous les énoncés ne nécessitent pas ces prérequis.

•logiciels de traitement de texte avec

_macro–commandes enregistrables,

_un module de langage de programmation intégré ou associé,

_un module de gestion de fichiers pour la fusion "des documents–type" avec les enregistrements d'un ou plusieurs fichiers de données,

_les fichiers de données sont intégrés au traitement de texte, ou proviennent d'un autre logiciel de gestion de fichiers.

_Si on considère que les macro–instructions et programmes sont des extensions de capacités des traitements de texte, leur conception ne se justifie que si le travail demandé dans l'énoncé ne fait pas partie des fonctionnalités de base du logiciel. En d'autres termes, si l'énoncé du travail à faire faire correspond à une action de base du traitement de texte, il n'y a pas d'algorithme et de macro–instructions à concevoir.

2.1

Tous les énoncés sont accompagnés d'une ébauche d'analyse, d'une suggestion d'algorithme, de la traduction de cet algorithme pour WordPerfect 6.0 et pour Word 2.0. Ces propositions ne représentent pas la seule et unique manière de concevoir les algorithmes, et je serai très heureux de recevoir d'autres suggestions d'algorithmes et de programmes pour ces énoncés. Le principe essentiel est d'utiliser au maximum les possibilités du traitement de texte, en évitant le parcours séquentiel caractère par caractère du texte, pour élaborer l'algorithme.

_Mon hypothèse de travail a été d'essayer de vérifier s'il est possible de dégager des algorithmes identiques pour construire les programmes, indépendamment du traitement de texte. Les exemples choisis montrent que cette hypothèse se vérifie dans la majorité des situations. L'utilisation de fonctions existant dans un langage et pas dans un autre induit une élaboration de cette fonction dans le langage qui le nécessite.

Les algorithmes sont volontairement simplifiés. Les validations des valeurs entrées par l'utilisateur, les cas limites, la gestion de l'interface homme–machine,... n'ont pas été traités pour ne pas allonger les algorithmes et programmes.

2.1.1 Faire enregistrer une macro–instruction qui permette d'inverser deux caractères consécutifs.

Ce type de macro–instruction est très facile à faire enregistrer. Il faut répondre à quelques questions avant de se lancer dans son enregistrement.

Quelles sont les conditions initiales?

- le curseur est–il sur le premier des deux caractères à permuter ou sur le deuxième. Le choix est d'autant plus important, que l'enregistrement de la macro–instruction, mais surtout son fonctionnement dans tous les cas envisageables doit être correct. (un seul caractère dans le texte). Dans le cas présent, j'opterai pour le curseur à gauche du deuxième caractère.

Ce type de macro–instruction n'est intéressante que si son exécution est obtenue par un raccourci clavier, par exemple ALT–C

Dans ce cas l'algorithme est simple: sélectionner le caractère à droite du curseur, donner l'ordre de le déplacer (le curseur est alors à droite du premier caractère), faire déplacer le curseur d'un caractère vers la gauche, coller le caractère déplacé. Le cas où le texte ne comporte qu'un caractère n'est pas envisagé.

Algorithme	
Programme macro_1 lection rseur_car_droite uper_sélection rseur_car_gauche ller_sélection **fin**	
Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**
SPLAY(Off!) ockOn(CharMode!) sCharNext t sCharPrevious ste Return	b MAIN endreSélection rDroite 1, 1 litionCouper rGauche 1 litionColler d Sub

2.1.2 Faire enregistrer une macro–instruction qui permette d'inverser deux paragraphes consé– cutifs.

Préliminaires

Même remarque que pour l'exercice 1, en remplaçant caractère par paragraphe.

Algorithme	
ogramme macro2 uper_paragraphe rseur_par_précédent ller_paragraphe n	
Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**
SPLAY(Off!) ockOn(ParagraphMode!) t ragraphUp ste turn	b MAIN radown(1, 1) litionCouper raup(1, 0) litionColler d Sub

2.1.3 Faire enregistrer une macro–instruction qui permette d'appliquer un style donné à un paragraphe. Par exemple la macro–instruction «t» qui applique le style «titre» à un paragraphe.

Ce type de macro–instruction sert à accélérer les manipulations du traitement de texte. Elle correspond à l'enregistre–ment de manipulations souvent exécutées par l'utilisateur. Elle ne peut être conçue que si le traitement de texte offre la possibilité de faire rechercher et appliquer les styles enregistrés dans une feuille de style.

Algorithme	
Il n'y a pas d'algoritmme à prorement parlé. Les balises s'appliquant à un paragraphe, l'appel de l'instruction d'utilsation de la balise suffit.	
Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**
SPLAY(Off!) yleOn("titre") turn	b Main rmatStyle .Nom = "Titre", .Appliquer d Sub

2.1.4 Faire enregistrer une macro–instruction qui mette la première lettre d'un paragraphe en majuscule.

Cette action est de plus en plus une action de base des traitements de texte, pour les mots mais pas pour les paragraphes. La macro–instruction devra faire faire le travail à partir de n'importe quelle position du curseur dans le paragraphe.

L'algorithme est une simple séquence: curseur au début du paragraphe, sélection de la première lettre, convertion de cette lettre en majuscule.

<table>
<tr><th colspan="2">Algorithme</th></tr>
<tr><td colspan="2">Programme macro4

rseur_para_précédent
lection
rseur_car_droite
ajuscules
rseur_para_suivant
n</td></tr>
<tr><th>Traduction pour WordPerfect 6.0</th><th>Traduction pour Word 2.0</th></tr>
<tr><td>SPLAY(Off!)
ragraphUp
ockOn(CharMode!)
sCharNext
nvertCaseUppercase
ockOff
ragraphDown
turn</td><td>b MAIN
raVersHaut 1
endreSÉlection
rDroite 1, 1
ajusculeMinuscule
raVersBas 1
d Sub</td></tr>
</table>

2.1.5 Ecrire un programme pour faire mettre la première lettre de tous les paragraphes en majuscule.

Ce travail est une généralisation du travail précédent. Il est manifeste que si le nombre de paragraphe est connu, il suffit de faire répéter la macro–instruction précédente un certain nombre de fois. L'intéressant est de concevoir une macro–instruction en considérant que le nombre de paragraphes est inconnu.

Une répétition "Tant que" des instructions de l'algorithme précédent convient pour cet algorithme. Le curseur est d'office positionné au début de chaque paragraphe. Le prédicat "findetexte" est simulé par le caractère "*" qui doit être obligatoirement présent uniquement à la fin du texte..

Algorithme	
Programme macro_5 urseurDébut nt que non(findetexte) lection rseur_car_droite ajuscules rseur_para_suivant nTantque **n**	
Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**
SPLAY(Off!) sDocTop hile(?RightChar <>"*") ockOn(CharMode!) sCharNext nvertCaseUppercase ockOff ragraphDown dwhile turn	b MAIN butDocument hile SÉlection$() <> "*" endreSÉlection rDroite 1, 1 rmatCaract_re .Majuscules = 1 rDroite 1 raVersBas 1 end d Sub

2.1.6 Faire enregistrer une macro–instruction qui transforme la première lettre d'un para–graphe en une lettrine

Cette action est de plus en plus une action de base des traitements de texte. L'algorithme est proposé pour les traitements de texte qui ne disposeraient pas de cette action. La lettrine proposée ici est simplement la première lettre du paragraphe dans une police de taille 36.

L'algorithme est une simple séquence: curseur au début du paragraphe, sélection de la première lettre, changement de taille de la police de cette lettre.

<table>
<tr><th colspan="2">Algorithme</th></tr>
<tr><td colspan="2">Programme macro_6

rseur_para_précédent
lection
rseur_car_droite
leur_taillepolice(36)
rseur_para_suivant
n</td></tr>
<tr><th>Traduction pour WordPerfect 6.0</th><th>Traduction pour Word 2.0</th></tr>
<tr><td>SPLAY(Off!)
ragraphUp
ockOn(CharMode!)
sCharNext
ntSize(36p)
ockOff
ragraphDown
turn</td><td>b MAIN
raVersHaut 1
endreSÉlection
rDroite 1, 1
rmatCaract_re .Police = "Tms Rmn", .Points = "36", .Gras =
0, .Italique = 0, .BarrÉ = 0, \ .CachÉ = 0,
.PetitesMajuscules = 0, .Majuscules = 0, .SoulignÉ =
.Couleur = 0, .Position = \ "0 pt", .Espacement = "0 p
rDroite 1
raVersBas 1
d Sub</td></tr>
</table>

2.1.7 Ecrire un programme pour transformer la première lettre de tous les paragraphes en une lettrine

Ce travail est une généralisation du travail précédent. Il est manifeste que si le nombre de paragraphes est connu, il suffit de faire répéter la macro–instruction précédente un certain nombre de fois. L'interessant est de concevoir une macro–instruction en considérant que le nombre de paragraphes est inconnu.

Une répétition Tant que convient pour cet algorithme. Le prédicat "findetexte" est simulé par le caractère "*" qui doit être obligatoirement présent uniquement à la fin du texte..

Algorithme	
Programme macro_7 rseurDébut nt que non(findetexte) lection rseur_car_droite leur_taillepolice(36) rseur_para_suivant tantque n	
Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**
SPLAY(Off!) sDocTop ile(?RightChar <>"*") ockOn(CharMode!) sCharNext ntSize(36p) ockOff ragraphDown dwhile turn	ébutDocument hile SÉlection$() <> "*" endreSÉlection rDroite 1, 1 rmatCaract_re .Police = "Tms Rmn", .Points = "36", .Gras = 0, .Italique = 0, .BarrÉ = 0, \ .CachÉ = 0, .PetitesMajuscules = 0, .Majuscules = 0, .SoulignÉ = .Couleur = 0, .Position = \ "0 pt", .Espacement = "0 p rDroite 1 raVersBas 1 end d Sub

2.1.8 Faire enregistrer une macro–instruction qui permette de supprimer tous les espaces multiples (plusieurs caractères espaces qui se suivent) dans un texte.

Ce travail est surtout utile pour toiletter les textes provenant d'autres traitements de texte, ou d'utilisateur, qui gèrent la mise en page en utilisant le caractère espace. Cet algorithme permet de montrer les différences entre les algorithmes avec parcours séquentiel et ceux qui utilisent les possibilités du traitement de texte.

L'algorithme présenté ici utilise les fonctions de remplacement des traitements de texte. L'algorithme de parcours séquentiel du texte est laissé à titre d'exercice. L'idée de l'algorithme est de faire remplacer les doubles espace par un seul dans tout le document, tant qu'il y a des doubles espaces dans le texte.

Algorithme

Programme macro_8

urseurDébut
chercher_avant(" ")
nt que trouvé?
urseur_Début
mplacer_tout(" "," ")
chercher_avant(" ")
tantque

Version de l'algorithme utilisant le répétitif Itérer et sortirsi. Pour les exercices suivants, les deux versions ne seront plus d'office présentées.

ogramme macro_8_bis
rer
urseurDébut
chercher_avant(" ")
rtirSi non(trouvé?)
urseurDébut
mplacer_tout(" "," ")
intérer

Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**

```
SPLAY(Off!)
 = 1    simulation d'itérer
hile(ok = 1)            avec while
sDocTop
archString("  ")
rchForward
?NotFound)
eak
dif
sDocTop
placeString(" ")
placeForward()
dwhile
turn
```

```
b MAIN
butDocument
itionRechercher .Rechercher = "  "
hile EditionRechercherTrouver()
itionRemplacer .Rechercher = "  ", .Remplacer = " ",
        .MotEntier = 0, .Casse = 0, \ .Format = 0,
        .RemplacerTout = 1
butDocument
itionRechercher .Rechercher = "  "
end
d Sub
```

2.1.9 Faire enregistrer une macro–instruction qui permette de supprimer tous les sauts de paragraphes forcés multiples. (plusieurs caractères «RETURN» consécutifs dans un texte)

Ce travail est surtout utile pour toiletter les textes provenant d'autres traitements de texte, ou d'utilisateur, qui gèrent la mise en page en utilisant le caractère RETURN. Cet algorithme permet de montrer les différences entre les algorithmes avec parcours séquentiel et ceux qui utilisent les possibilités du traitement de texte.

L'algorithme présenté ici utilise les fonctions de remplacement des traitements de texte. L'utilisation de caractères plus particulier tels que les "return" lors du remplacement n'est pas toujours faisable avec les traitements de texte. L'algorithme de parcours séquentiel du texte est laissé à titre d'exercice.

Algorithme	
L'algorithme est identique à celui du remplacement des espaces. ("espace" devient "return")	
Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**
SPLAY(Off!) = 1 ile(ok = 1) sDocTop archString("[RT][RT]") archForward ?notfound) eak dif dwhile sDocTop placeString("[RT]") placeForward() turn	b MAIN butDocument itionRechercher .Rechercher = "^p^p" hile EditionRechercherTrouver() itionRemplacer .Rechercher = "^p^p", .Remplacer = "^p", .RemplacerTout = 1 DébutDocument itionRechercher .Rechercher = "^p^p" end d Sub

2.1.10 Ecrire un programme qui permette de nettoyer un texte des espaces multiples consécutifs et des sauts de paragraphes forcés multiples consécutifs.

Ce travail est réalisé en utilisant les deux macro–instructions précédentes.

Cet algorithme illustre l'appel de procédures dans un algorithme. Les programmes sont suffisamment explicites; Les partisans d'une programmation structurée et une analyse descendante y trouveront leur compte.

<table>
<tr><th colspan="2">Algorithme</th></tr>
<tr><td colspan="2">ogramme Macro_20
tespace
tretour
n</td></tr>
<tr><th>Traduction pour WordPerfect 6.0</th><th>Traduction pour Word 2.0</th></tr>
</table>

```
)TFOUND(off!)
 SPLAY(Off!)
tespace()
tretour()
urn

ocedure netespace()
 = 1
ile(ok = 1)
sDocTop
archString("  ")
rchForward
?NotFound)
eak
dif
sDocTop
placeString(" ")
placeForward()
dwhile
dproc



ocedure netretour()
 = 1
ile(ok = 1)
sDocTop
archString("[RT][RT]")
archForward
?notfound)
eak
dif
dwhile
sDocTop
placeString("[RT]")
placeForward()
dproc
```

```
b Main
tespace
tretour
d sub

b netespace
butDocument
itionRechercher .Rechercher = "  "
hile EditionRechercherTrouver()
itionRemplacer .Rechercher = "  ", .Remplacer = " ",
        .MotEntier = 0, .Casse = 0, \ .Format = 0,
        .RemplacerTout = 1
butDocument
itionRechercher .Rechercher = "  "
end
d Sub

b netretour
butDocument
itionRechercher .Rechercher = "^p^p"
hile EditionRechercherTrouver()
itionRemplacer .Rechercher = "^p^p", .Remplacer = "^p",
        .RemplacerTout = 1 DébutDocument
itionRechercher .Rechercher = "^p^p"
end
d Sub
```

2.1.11 Dans un texte, les règles de ponctuation dactylographique n'ont pas été toujours respectées: parfois un espace après la virgule, parfois aucun, parfois deux espaces, etc ... pour les autres caractères. Ecrire un programme qui modifie le texte pour que toutes les règles de ponctuation dactylographique soient respectées.

Cet énoncé est d'application pour des textes encodés par des personnes ne connaissant pas ou n'appliquant pas les règles des signes de ponctuation. L'algorithme correspondant peut être envisagé en utilisant un seul parcours séquentiel du texte, ou en utilisant les possibilités de remplacement offertes par le traitement de texte.

Les possibilités de remplacement sont utilisées à outrance dans cette séquence. Tous les cas de figure n'ont pas été repris, mais l'algorithme peut être facilement complèté.

<table>
<tr><th colspan="2">Algorithme</th></tr>
<tr><td colspan="2">L'algorithme est une séquence d'appels des procédures et fonctions de remplacement remplace_–
tout(chaine1,chaine2).</td></tr>
<tr><th>Traduction pour WordPerfect 6.0</th><th>Traduction pour Word 2.0</th></tr>
</table>

```
 DISPLAY(Off!)


sDocTop
archString(".")
placeString(".  ")
placeForward()
sDocTop
archString(",")
placeString(", ")
placeForward()
sDocTop
archString("?")
placeString("? ")
placeForward()
sDocTop
archString("!")
placeString("! ")
placeForward()
sDocTop
archString(":")
placeString(": ")
placeForward()
sDocTop
archString(".  .  .")
placeString("...")
placeForward()
sDocTop
archString(" :")
placeString(":")
placeForward()
```

```
b MAIN
ebutDocument
itionRemplacer .Rechercher = ".", .Remplacer = ".  ",
        .RemplacerTout = 1
ebutDocument
itionRemplacer .Rechercher = ",", .Remplacer = ", ",
        .RemplacerTout = 1
ebutDocument
itionRemplacer .Rechercher = "?", .Remplacer = "? ",
        .RemplacerTout = 1
ebutDocument
itionRemplacer .Rechercher = "!", .Remplacer = "! ",
        .RemplacerTout = 1
ebutDocument
itionRemplacer .Rechercher = ":", .Remplacer = ": ",
        .RemplacerTout = 1
ebutDocument
itionRemplacer .Rechercher = ". . .", .Remplacer = "...",
        .RemplacerTout = 1
ebutDocument
itionRemplacer .Rechercher = " :", .Remplacer = ":",
        .RemplacerTout = 1
ebutDocument
itionRemplacer .Rechercher = " ,", .Remplacer = ",",
        .RemplacerTout = 1
ebutDocument
 ...
d Sub
```

```
sDocTop
archString(" ,")
archNext()
archString(" ,")
placeString(",")
placeForward()
sDocTop
archString(" ?")
archNext()
sDocTop
archString(" ?")
placeString("?")
placeForward()
sDocTop
archString(" !")
archNext()
archString(" !")
placeString("!")
placeForward()
  etc ...

  Return
```

2.1.12 Ecrire un programme qui harmonise la présentation de tous les tableaux présents dans le texte de la manière suivante: tous les tableaux auront un cadre avec des lignes double, les lignes intérieures seront toutes simples, les titres des colonnes de tous les tableaux sont centrés dans la cellule, horizontalement et verticalement et seront dans un style gras.

TITRE	**TITRE**	**TITRE**

Cet énoncé s'applique dans le cas où l'harmonisation de la présentation n'a pas été prévue au départ. Le traitement de texte doit permettre de faire faire une recherche sur les tableaux du texte.

Cet algorithme fait apparaître les différences entre deux langages. WordPerfect permettant une recherche sur les codes définissant les tableaux et Word pas. L'algorithme en Word doit simuler cette recherche en parcourant tous les paragraphes séquentiellement et en vérifiant leur caractéristiques à l'aide de la fonction SélInfo. Dans l'algorithme WordPerfect, Itérer et Sortirsi sont utilisés pour éviter certains inconvénients du tant que.

Algorithme	
Programme macro_12 urseurDébut rer chercher(tableau) rtirSi non(trouvé?) pliquer_caractéristiques procédure à développer en fonction du traitement de texte intérer n	
Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**

```
SPLAY(Off!)
OTFOUND(off!)
sDocTop
 = 1
hile(ok = 1)
archString("[Déf Tbl]")
archNext()
?notfound)              s'il n'y a plus de tableau
eak
dif
bleEdit
bleBlockOn
sTableEnd
bleDefaultLine(SingleLine!)
bleBorderEditBegin
bleBorder(DoubleBorder!)
bleBorderEditEnd(Save!)
bleBlockOff
sTableBeg
bleBlockOn
sTableRowEnd
bleCellVerticalAlignment(Center!)
bleCellAttributeOn(Bold!)
bleCellJustification(Center!)
dwhile
turn
```

```
b MAIN
ébutDocument
hile Sélection$() <> "*"        tant que pas à la
SélInfo(12) = - 1 Then          fin du texte (*)
bleauSélectionnerLigne
raCentrÉ
as 1
bleauSÉlectionnerTout
rmatBordure .DuTexte = "", .AppliquerA = 3, .Ombre = 0,
        .BordureHaut = 4, \ .BordureGauche = 4, .BordureBa
        = 4, .BordureDroite = 4, .BordureHoriz = 0, \
        .BordureVert = 0, .CouleurHaut = 0, .CouleurGauche
        0, .CouleurBas = 0, \ .CouleurDroite = 0,
        .CouleurHoriz = 0, .CouleurVert = 0, .Motif = 0,
        .PremierPlan = 0, \ .Arri_rePlan = 0
gneVersBas 1
d If
raVersBas
end
d Sub
```

2.1.13 Un texte est structuré de la manière suivante:

expression 1: définition de l'expression définition de l'expression définition de l'expression définition de l'expression définition de l'expression définition de l'expression.

expression 2: définition de l'expression définition de l'expression définition de l'expression définition de l'expression définition de l'expression définition de l'expression.

expression 3: définition de l'expression définition de l'expression définition de l'expression définition de l'expression définition de l'expression définition de l'expression.

expression 4: définition de l'expression définition de l'expression définition de l'expression définition de l'expression définition de l'expression définition de l'expression.

etc ...

Faire enregistrer une macro–instruction qui met l«'expression n°» en style gras.

Cet algorithme attire l'attention sur la nécessité de disposer d'une structuration des données. L'idée est de sélectionner le texte "expression n_" et de lui faire apliquer le style gras. Cet sélection ne peut être effectuée que si le texte est structuré d'une certaine manière. Cete structure est obtenue grâce au ":" qui permet de délimiter formellement l'"expression.".

L'algorithme est une simple séquence illustrant l'utilisation de la fonction de marquage d'un texte et de la recherche d'un caractère. Le curseur est supposé être sur le premier caractère du paragraphe.

Algorithme	
Programme macro_13 lection chercher(":") le(gras) rseur_para_suivant **n**	
Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**
SPLAY(Off!) ockOn(CharMode!) sCharacter(":") tributeAppearanceOn(Bold!) ockOff ragraphDown turn	b MAIN endreSÉlection endreSÉlection \ as 1 raVersBas 1 d Sub

2.1.14 Ecrire un programme qui met toutes les «expression n°» en style gras.

Ce travail est une généralisation du travail précédent. Il est manifeste que si le nombre de paragraphes est connu, il suffit de faire répéter la macro précédente un certain nombre de fois. L'intéressant est de concevoir une macro en considérant que le nombre de paragraphes est inconnu. L'utilisation de "Itérer sortirsi" pour WordPerfect et "Tant que" pour Word illustre certains avantages d'"itér–er Sortirsi"

Algorithme	
Programme macro_14 urseurDébut rer lection chercher(":") rtirsi non(trouvé?) yle(gras) urseur_para_suivant intérer n	
Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**
SPLAY(Off!) tfound(off!) sDocTop = 1 hile(ok = 1) ockOn(CharMode!) archString(":") archNext() ?notfound) ockOff eak dif tributeAppearanceOn(Bold!) ockOff ragraphDown dwhile sDocTop turn	ébutDocument endreSÉlection itionRechercher .Rechercher = ":", .MotEntier = 0, .Casse = 0, .Vers = 1, .Format = 0 hile EditionRechercherTrouver() And fini = 0 as 1 raVersBas 1 SÉlection$() <> "*" Then endreSÉlection itionRechercher .Rechercher = ":", .MotEntier = 0, .Casse = 0, .Vers = 1, .Format = 0 Else i = 1 dIf end d Sub

2.1.15 Ecrire un programme qui à la demande de l'utilisateur, permet de crypter un texte ou de décrypter un texte. Le nom sous lequel le texte a été enregistré sera fourni par l'utilisa–teur.

Le cryptage /décryptage d'un texte nécessite un algorithme de cryptage. J'ai choisi un cryptage des plus primitifs pour illustrer l'algorithme: le nouveau caractère est obtenu en augmentant le code ascii du caractère en cours de 10. L'idée est de parcourir le texte du début jusqu'à la fin et de remplacer chaque craactère par un nouveau, en utilisant la séquence insérer/suprimer. Des variables sont utilisées dans cet algorithme.

<table>
<tr><th colspan="2">Algorithme</th></tr>
<tr><td colspan="2">Programme macro_15

éclaration des variables: fichier: caractères, nouveau: entier
rire"Nom du fichier"
re fichier)
uvrir_texte(fichier)
rseurDébut
nt que non(findetexte)
uveau = code_ascii(car_droite)+10
face_cara_droite
sère_valeur(car_ascii(nouveau))
tantque
n</td></tr>
<tr><th>Traduction pour WordPerfect 6.0</th><th>Traduction pour Word 2.0</th></tr>
<tr><td>SPLAY(Off!)
tstring(fichier;"Nom du fichier";)
leOpen(fichier)
sDocBottom
pe("*")
sDocTop
hile(?rightchar<>"*")
uveau = cton(?rightcode) + 10
leteCharNext
pe(ntoc(nouveau))
dwhile
urn</td><td>b MAIN
mfichiers$=inputbox$("nom du fichier")
hierouvrir .nom=nomfichier$
document
sérer "*"
ébutDocument
hile Sélection$() <> "*"
uveau$ = Chr$(Asc(Sélection$()) + 10)
litionEffacer
sérer nouveau$
end
d Sub</td></tr>
<tr><td colspan="2">Cet algorithme illustre l'insertion du caractère "*" en fin de texte pour simuler le prédicat "findetexte".</td></tr>
</table>

2.1.16 Ecrire un programme qui permet de compter et insèrer à la fin du texte le nombre de lettres, de mots, de ligne, de paragraphes d'un texte. Le nom sous lequel le texte a été enregistré sera fourni par l'utilisateur.

A partir des valeurs du comptage, on peut faire des calculs de prix de traduction par exemple et insérer leurs résultats. On peut aussi imaginer les calculs statistiques plus élaborés sur le texte. Ces actions font partie des possibilités des traitements de texte élaborés. Il faut cependant vérifier le comptage des mots en fonction de leur définition formelle: Madame , Monsieur 3 ou 2 mots?

Les traitements de texte intègrent des fonctions de comptage qui ont été délibérément ignorées. L'algori–thme simple proposé ici ne fait que le comptage des caractère, y compris les espaces. Il peut être généralisé au comptage des autres éléments du texte.

Algorithme

Programme macro_16

éclaration des variables: fichier: caractère, nblettre: réel
rire"Nom du fichier"
re fichier)
uvrir_texte(fichier)
rseurDébut
lettre := 0
t que non(findetexte)
lettre = nblettre + 1
rseur_cara_droite
tantque
sérer_valeur("Nombre de lettres",nblettre)

Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**
SPLAY(Off!) sDocTop lettre = 0 ile(?rightchar<>"*") lettre = nblettre + 1 sCharNext dwhile t e("lettres:") e(nblettre) urn	ébutDocument lettre = 0 hile SÉlection$() <> "*" lettre = nblettre + 1 rDroite end SérerPara Sérer "Nombre de lettres : " Sérer Str$(nblettre) d Sub

2.1.17 A partir d'un texte donné, faire un texte lacunaire en remplaçant dans le texte, les mots d'une liste de mots fournie par l'utilisateur, par autant d'astérisques que le mot à remplacer compte de lettres.

L'algorithme dépend de l'interface utilisateur envisagé. Dans la solution proposée, les chaînes à remplacer sont encodées puis remplacées, avant de passer à la chaîne suivante. L'idée est d'utiliser une fonction qui retourne la valeur de la chaîne encodée par l'utilisateur. La longueur maximale de la chaîne ne peut dépasser 26 caractères.

Algorithme	
Programme Macro_17 éclaration des variables: chaîne, asterisque: caractère terisque = "**********" aîne = àremplacer() nt que chaîne <> "" mplacer_tout(chaîne, sous–chaîne(asterisque,1,long(chaîne)) aîne = àremplacer() tantque **n** **nction àremplacer()** claration des varaibles locales: x: caractère e x nt que long(x) > 26 e x tantque **etourne(x)**	
Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**

```
splay(off!)
terisque = "**************************"
 = 1
hile(ok = 1)
aine = àremplacer()
chaine = "")
eak
dif
sdoctop
archString(chaine)
placeString(substr(asterisque;1;strlen(chain e)))
placeForward()
dwhile
urn
UNCTION àremplacer()
 = 1
hile(ok=1)
tstring(x;"mot à remplacer";)
strlen(x)<=26)
eak
dif
dwhile
turn(x)
dfunc
```

```
b MAIN
terisque$ = "**************************"
aine$ = àremplacer$
hile chaine$ <> ""
ebutDocument
litionRemplacer .Rechercher = chaine$, .Remplacer =
id$(asterisque$, 1, Len(chaine$–)), .RemplacerTout
aine$ = àremplacer$
end
d Sub

nction àremplacer$
 = InputBox$("chaine à remplacer")
hile Len(x$) > 26
 = InputBox$("chaine à remplacer")
end
emplacer$ = x$
d Function
```

2.1.18 Dans un texte donné, insérer la conjugaison d'un verbe donné par l'utilisateur à toutes les personnes, pour un temps choisi par l'utilisateur. Les temps proposés dépenderont du programmeur.

C'est le type de travail dont l'utilité n'est pas certaine, quoique? Il est proposé à titre indicatif, en espérant qu'il donnera naissance à d'autres énoncés peut–être plus utiles.

Cet algorithme illustre que de nombreux énoncés de problèmes destinés aux langages de programmation plus traditionnels sont adaptables aux traitements de texte. Dans cet algorithme de conjugaison, seul l'indic–atif du présent des verbes régulier en "er" est envisagé. Aucune validation des données n'est effectuée, il serait interessant de prévoir celle–ci en liaison avec les dictionnaires de vérification orthographique intégrés aux traitements de texte. C'est l'occasion de montrer l'utilisation de variables de type tableau.

Algorithme	
Programme macro_18 éclaration des variables: personne, present: tableau(6); radical, verbe: caractère;i: entier rsonne[1] = "Je" rsonne[2] = "Tu" rsonne[3] = "Il/Elle" rsonne[4] = "Nous" rsonne[5] = "Vous" rsonne[6] = "Ils/Elles" esent[1] = "e" esent[2] = "es" esent[3] = "e" esent[4] = "ons" esent[5] = "ez" esent[6] = "ent" rire "Verbe à conjuguer" e verbe dical = sous_chaîne(verbe,1,longeur(verbe)–2) sère_valeur("conjugaison du verbe",verbe,"a l'indicatif présent" ur i = 1 vers 6 sère_valeur(personne[i],radical+present[i]) pour n	
Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**

```
SPLAY(Off!)
rsonne[1] = "Je"
rsonne[2] = "Tu"
rsonne[3] = "Il/Elle"
rsonne[4] = "Nous"
rsonne[5] = "Vous"
rsonne[6] = "Ils/Elles"
esent[1] = "e"
esent[2] = "es"
esent[3] = "e"
esent[4] = "ons"
esent[5] = "ez"
esent[6] = "ent"
tstring(verbe;"Verbe à conjuguer";"Conjugaison")
dical = substr(verbe;1;strlen(verbe) – 2)
be("Conjugaison du verbe ")
ributeappearanceon(bold!)
be(verbe)
ributeappearanceoff(bold!)
be(" au présent")
t
t
rnext(i;1;6)
be(personne[i]+ " ")
be(radical)
ributeappearanceon(bold!)
be(present[i])
ributeappearanceoff(bold!)
t
dfor
turn
```

```
b MAIN
m personne$(6), present$(6)
rsonne$(1) = "Je"
rsonne$(2) = "Tu"
rsonne$(3) = "Il/Elle"
rsonne$(4) = "Nous"
rsonne$(5) = "Vous"
rsonne$(6) = "Ils/Elles"
esent$(1) = "e"
esent$(2) = "es"
esent$(3) = "e"
esent$(4) = "ons"
esent$(5) = "ez"
esent$(6) = "ent"
rbe$ = InputBox$("Verbe Ó conjuguer")
dical$ = Mid$(verbe$, 1, Len(verbe$) - 2)
Sérer "Conjugaison du verbe "
as 1
Sérer verbe$
as 0
Sérer " au prÉsent"
SérerPara
SérerPara
r i = 1 To 6
Sérer personne$(i) + " " + radical$
as 1
Sérer present$(i)
as 0
SérerPara
xt
d Sub
```

2.1.19 Les lignes d'un texte représente les enregistrements d'un fichier. Malheureusement, il n'y a aucun séparateur de champ, le caractère «return» (fin de paragraphe) est le séparateur d'enregistrement. Cependant, on sait que le premier champ compte 4 caractères, le deuxième 18, le troisième 5, le quatrième 8. Ecrire un programme qui insère le séparateur de champs «#,#» entre chaque champ.

le texte peut être représenté de la manière suivante, A,B,C et D pouvant représenter n'importe quel caractère:

AAAABBBBBBBBBBBBBBBBCCCCCDDDDDDDD

AAAABBBBBBBBBBBBBBBBCCCCCDDDDDDDD

AAAABBBBBBBBBBBBBBBBCCCCCDDDDDDDD

AAAABBBBBBBBBBBBBBBBCCCCCDDDDDDDD

AAAABBBBBBBBBBBBBBBBCCCCCDDDDDDDD

etc ...

doit devenir

#AAAA#,#BBBBBBBBBBBBBBBB#,#CCCCC#,#DDDDDDDD# etc..

Ce type de travail est utile lorsqu'il est question d'échanger des fichiers de diverses origines. Il est présenté pour son intérêt théorique même s'il a été réellement utilisé dans une pratique de bureautique sur un fichier dont les enregistrements avaient une longeur de 985 carcatères (sic.) sans séparateurs.

Un simple parcours séquentiel de chaque ligne (paragraphe) tant que la fin du texte n'est pas atteinte. (simulé par "*" dans les langages)

Algorithme
Programme macro_19 rseur_début nt que non(findetexte) sère_valeur("#") seur_droite(4) sère_valeur("#,#") rseur_droite(18) sère_valeur("#,#") rseur_droite(8) sère_valeur("#") rseur_para_suivant tantque n

Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**
SPLAY(Off!) sDocTop ile(?rightchar<>"*") e("#") peatValue(4) sCharNext pe("#,#") peatValue(18) sCharNext pe("#,#") peatValue(5) sCharNext pe("#,#") peatValue(8) sCharNext pe("#") ragraphDown dwhile turn	b MAIN butDocument hile SÉlection$() <> "*" Sérer "#" rDroite 4 Sérer \ ,#" rDroite 18 Sérer "#,#" rDroite 5 Sérer "#,#" rDroite 8 Sérer "#" raVersBas 1 end d Sub

2.1.20 On dispose de feuilles cartonnées aux dimensions A4 pour faire imprimer des cartes de visite. Ecrire un programme interactif qui permette à un utilisateur d'encoder ses coordonnées et le nombre de cartes de visites qu'il souhaite, et de faire imprimer ces cartes. La feuille A4 contiendra 10 cartes de visites réparties sur 2 colonnes de 5 cartes. Le nombre de cartes demandé devra être arrondi pour que toutes les feuilles A4 imprimées soient bien remplies. La définition du contenu des cartes est laissée au concepteur du programme.

Cet énoncé permet d'illustrer les concepts d'entrée et de sortie des informations à partir d'un dialogue avec l'utilisateur du traitement de texte.

La difficulté essentielle des algorithmes est liée au spécifictés des deux traitements de texte. WordPerfect permet de facilement diviser une page physique en pages logiques. Je n'ai pas trouvé d'équivalent avec Word. Un utilisateur averti de Word pourrait–il me faire parvenir une solution élégante?

L'algorithme est simplifié. Il est évident que le dialogue et la saisie des données peuvent être améliorés en utilisant les possibilités de gestion des boîtes de dialogues et autres objets de l'univers des interfaces graphiques.

Algorithme

Programme macro_20

éclaration des variables:nombre,i,reste: entier;nom;prenom;adresse;cp;localite;telephone: caractère
isie
pression

rocèdure saisie
e nom
e prenom
e adresse
e cp
e localite
e telephone
e nombre
ste = nombre modulo 10
reste <> 0
mbre = (quotient(nombre/10) + 1)*10
si
procédure

rocédure impression
ur i = 1 vers nombre
sère_valeur(nom,prenom)
sère_valeur(adresse)
sère_valeur(cp,localite)
sére_valeur(telephone)
pour
procédure

Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**

```
LO–BAL(nombre ;nom ;prenom ;adresse ;cp ;localite; tele–
        phone)
isie()
arginLeft(1c)
arginRight(1c)
arginBottom(1c)
arginTop(1c)
bdividePage(2;5)
pression()
urn

ocedure saisie()
tstring(nom;"nom")
tstring(prenom;"prénom")
tstring(adresse;"adresse")
tstring(cp;"code postal")
tstring(localite;"Localité")
tstring(telephone;"Téléphone")
tnumber(nombre;"nombre de cartes")
ste = nombre%10
reste <> 0)
mbre = ((nombre div 10) + 1)*10
dif
dproc

ocedure impression()
rnext(i;1;nombre)
pe(nom + " " + prenom)
pe(adresse)
pe(cp + " " + localite)
pe("Tél.: " + telephone)
i < nombre)
rdpagebreak
dif
dfor
 endproc
```

```
m Shared nom$, prenom$, adresse$, cp$, localite$,
        telephone$, nombre Sub MAIN
isie
pression
d Sub

b saisie
m$ = InputBox$("Nom")
enom$ = InputBox$("prÉnom")
resse$ = InputBox$("Adresse")
$ = InputBox$("Code postal")
calite$ = InputBox$("LocalitÉ")
ephone$ = InputBox$("TÉlÉphone")
mbre = Val(InputBox$("Nombre"))
ste = nombre Mod 10
reste <> 0 Then
mbre =(Int(nombre / 10) + 1) * 10
d If
d Sub

b impression
r i = 1 To nombre
Sérer nom$ + " " + prenom$
SérerPara
Sérer adresse$
SérerPara
Sérer cp$ + " " + localite$
SérerPara
Sérer telephone$
s_reSautPage
xt
d Sub
```

2.1.21 On dispose d'un texte réparti sur trois colonnes sur taquets de tabulation. Ecrire un programme qui mette le texte de la première colonne en gras, et celui de la deuxième en italique.

On suppose que chaque colonne ne contient qu'un seul mot (pas d'espace). Les colonnes étant des colonnes sur tabulation, chaque ligne est un paragraphe. Les actions à répéter sont: délimiter le premier mot, lui appliquer le style gras, délimiter le mot suivant, lui appliquer le style italique, passer au début du paragraphe suivant.

Algorithme	
Programme macro_21 rseurDébut nt que non(findetexte) lection seur_mot_suivant le(gras) lection rseur_mot_suivant le(italique) rsuer_para_suivant tantque	
Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**
SPLAY(Off!) tfound(off!) sDocBottom pe("*") = 1 ile(ok = 1) ?rightchar = "*") eak dif ockOn(CharMode!) sWordNext tributeAppearanceOn(Bold!) ockOff ockOn(CharMode!) sWordNext tributeAppearanceOn(Italics!) ockOff ragraphDown dwhile leteCharNext turn	b MAIN nDocument sérer "*" ébutDocument hile SÉlection$() <> "*" endreSÉlection endreSÉlection \ r$(9) as 1 rDroite 1 endreSÉlection endreSÉlection Chr$(9) lique 1 raVersBas 1 end d Sub

2.1.22 On dispose d'un fichier d'enregistrements structurés de la manière suivante (noms des champs):

TYPE,COL1,COL2,COL3,COL4,COL5

Le champ TYPE peut contenir 4 valeurs différentes: E,1,2, ou C.

On dispose dans une feuille de styles de 3 styles, appelés respectivement TITRE 1, TITRE 2 ET CORPS.

On demande d'écrire un programme qui génère un état imprimé sur 5 colonnes, des enregistrements du fichier, en sachant que:

•si le champ TYPE contient E, le contenu des champs COL1, COL2 et COL3 doivent être le contenu de la tête de page,

•si le champ TYPE contient 1, le contenu des champs COL1 et COL2 doivent être le titre général des colonnes dans le style TITRE 1,

•si le champ type contient 2, le contenu du champ COL1 doit être le sous–titre général des colonnes dans le style TITRE 2,

•si le champ type contient C, le contenu des champs COL1 à COL5 doivent être imprimés dans 5 colonnes en utilisant le style CORPS. Le contenu des champs peut dépasser la largeur prévue des colonnes, il faut alors que le texte s'imprime sur plusieurs lignes dans la colonne,

•Les enregistrements du fichier sont triés de la manière suivante:

Type: E,1,2,C,C,C,C,C,2,C,C,C,C,C,1,2,C,,C,C,E,1,C,C,C,E,1,2,C,C,C,C, etc...

Cet énoncé illustre le fonctionnement de ce qui est souvent appelé la fusion de document– type avec un fi–chier de données. La difficulté provient plus de l'élaboration des documents–type que de l'algorithme lui–même. La résolution du problème passe par la création de 5 documents type et d'un fichier de données. On suppose que l'insertion des noms des champs dans un document type est connu.

Il montre comment les documents–type peuvent contenir des structures de contrôle gérant la fusion des documents.

Algorithme

tête.txt

ntient les champs col1, col3 et col 4 en entête de page
re1.txt
ntient les champs col1 et col2 avec le style titre1.
re2.txt
ntient le champ col1 avec le style titre1.
rps.txt
ntient les champs col1 à col 5 répartis sur cinq colonnes définies au préalable, avec le style corps.
sion.txt
ntient une séquence de si alors.
type = "E", fusionner le document tête.txt
type = "1", fusionner le document titre1.txt
type = "2", fusionner le document titre2.txt
type = "C", fusionner le document corps.txt
nnées.txt
ntient les enregistrements du fichier.

Traduction Pour WordPerfect 6.0	**Traduction pour Word 2.0**

e.txt DL1 DL2 DL3 re1.txt DL1 DL2 re2.txt DL1 rps.txt DL1 DL2 DL3 DL4 DL5 sion.txt pe="E" e.txttype="A "titre1.txttype ="B"titre2.txtt ype="C"corps.txt	e.txt CHAMPFUSION COL1} CHAMPFUSION COL2} CHAMPFUSION COL3} re1.txt CHAMPFUSION COL1} CHAMPFUSION COL2} re2.txt CHAMPFUSION COL1} rps.txt CHAMPFUSION COL1} CHAMPFUSION COL2} CHAMPFUSION COL3} CHAMPFUSION COL4} CHAMPFUSION COL5} sion.txt {SI{CHAMPFUSION type} = "E" {INCLURE tête.txt}} {SI{CHAMPFUSION type} = "1" {INCLURE titre1.txt}} {SI{CHAMPFUSION type} = "2" {IN–CLURE titre2.txt}} {SI{CHAMPFUSION type} = "C" {IN–CLURE corps.txt}}

2.1.23 On dispose d'un fichier d'enregistrements structurés de la manière suivante:

TYPE, DENOMINATION

le champ TYPE peut contenir deux valeurs: école, élèves.

Les enregistrements sont triés dans l'ordre suivants:

école,Institut Pécé

élève,Dijkstra

élève,Newton

...

école,Institut Mac

éléve,Job

élève,Gates

...

On demande d'écrire un programme qui permettent de faire imprimer la liste des élèves d'une école dans un tableau, chaque école commençant sur une nouvelle page.

Ce problème est une application directe du problème précédent à une situation plus concrète. La détermination des documents–type et du fichier de données est laissée à titre d'exercice.

2.1.24 On dispose d'un texte contenant plusieurs tableaux. Ecrire un programme qui génère un nouveau texte ne contenant que les tableaux du texte, précédés du numéro de la page où le tableau commence.

Une autre version de cet énoncé: même demande, mais les tableaux à insérer dans le nouveau texte le seraient à la demande de l'utilisateur.

Ce problème illustre les différences entre les deux traitements de texte. WordPerfect disposant d'une fonction d'ajout d'un texte marqué à la fin d'un fichier existant, la procédure de création du deuxième document est immédiate. Word nécessite l'utilisation d'un deuxième fenêtre et de l'enregistrement du contenu de cette fenêtre en fin d'algorithme. La version de Word peut être adaptée pour WordPerfect si on ne veut pas utiliser la fonction "ajouter à". L'algorithme est la répétition de la recherche d'un tableau et de son ajout à la fin du deuxième texte, directemnt sur disque ou dans un texte ouvert dans une autre fenêtre.

Algorithme	
Programme macro_24 **urseurDébut** **rer** **chercher(tableau)** **rtirsi non(trouvé?)** **sère_valeur(numéro_page)** la valeur est insérée dans le document courant **lectionner_tableau** **puter_sélection("doc_tab")** cette procédure doit être définie en fonction du traitement de texte **itérer** **n**	
Traduction pour WordPerfect 6.0	**Traduction pour Word 2.0**

```
SPLAY(Off!)
sdoctop
tfound(off!)
 = 1
hile(ok=1)
archString("[Déf Tbl]")
archNext()
?notfound)
eak
dif
be("page ")
be(?Page)
archString("[Déf Tbl]")       Sélection du tableau
archPrevious()
sCharPrevious
ockOn(CharMode!)
archString("[Tbl dés]")
archNext()
ppendToFile("doc_tab")        ajout du tableau
ockOff
dwhile
  Return
```

```
b Main
oc$=InputBox$("Nom du document dont il faut copier les
        tableaux")
n$=Inputbox$("Nom du document dans lequel copier les
        tableaux?")
chierNouveauDocument .modèle="NORMAL"
ihierEnregistrerSous .nom=fen$
chierOuvrir .Nom=Doc$
hile CmpSignets("\Sel","\EndOfDoc")
raVersBas 1,1
SélInfo(12) = –1     sélection du tableau
hile SélInfo(12) = –1
gneVersBas 1,1
end
rGauche 1,1
litionCopier
=SélInfo(3)
tiver fen$ ajout du tableau
Sérer Str$(pa)
SérerPara
litionColler
tiver doc$
d if
bleauSélectionnerLigne
raVersBas
end
tiver fen$
chierEnregistrer
ocFermeture
dSub
```

2.1.25 Une disquette contient des fichiers de texte. Ces textes ont été enregistrés avec un autre traitement de texte au format ASCII. Ecrire un programme qui convertit et enregistre ces fichiers au format du traitement de texte utilisé dans un sous–répertoire appelé CONVERTI. Les noms des fichiers à convertir son entrés par l'utilisateur via le clavier.

Ce type de situation se présente pour les personnes qui sont amenées à gérer des texte rédigés avec différents traitements de texte. Ce type de manipulation n'est utile que si les conversions doivent être faites en une fois. Dans les autres cas, les traitements de texte permettent de convertir lors de l'ouverture d'un texte.

Les détails sur les différents formats de fichiers sont omis dans la description de l'algorithme. Cet exemple permet de nouveau d'illustrer la différence entre itérer et tant que.

Algorithme

Programme macro_25

```
rer
rire "nom du fichier"
e fichier
rtirsi fichier = ""
vrir_texte(fichier)
registrer_texte(converti,fichier)
face_texte
itérer
```

Traduction pour WordPerfect 6.0

```
splay(Off!)
 = 1
hile(ok=1)
tstring(fichier;"nom du fichier";)
fichier="")
eak
dif
leOpen("A:" + fichier;ASCIISrt!)
leSave("C:\CONVERTI\" + fichier;WordPer– fect_60!)
earDoc
dwhile
turn
```

Traduction pour Word 2.0

```
b MAIN
mfichier$ = InputBox$("nom du fichier")
hile nomfichier$ <> ""
chierOuvrir .Nom = nomfichier$, .LectureSeule = 0
mfichier$ = "c:\converti\" + nomfichier$
chierEnregistrerSous .Nom = nomfichier$, .Format = 0, \
errouillerSaufAnnotations = 0, .MotDePasse = ""
chierFermer
mfichier$ = InputBox$("nom du fichier")
end
d Sub
```