Traitement des chaînes de caractères

Par
Steve HUSSEY, PDG Alto Stratus LLC
Note technique 4D-200003-10-FR
Version 1

Date 1 Mars 2000

Résumé

4D possède des fonctions basiques de manipulations de chaîne de caractères.
Vous pouvez, à partir de celles-ci créer vos fonctions avancées. En créant ces fonctions avancées vous pourrez créer une bibliothèque de fonctions qui pourra être intégrée dans d'autres bases.

4D Notes techniques

Introduction

La plupart des développements 4D nécessitent l'utilisation de fonctions de manipulation de chaînes de caractères. En voici deux exemples : transformer un mot en majuscule ou enlever les espaces inutiles lors de la saisie. Ces deux exemples peuvent permettre, à partir d'un texte, la création de "mots clés".

Cette note technique présente un ensemble de fonctions, sous forme d'une librairie multi plate-forme, permettant la manipulation de chaînes de caractères.

Les fonctions internes à 4D

4D possède un ensemble de fonctions de manipulation de chaînes de caractères qui permettent la création de fonctions beaucoup plus complexes.

Ci-dessous la liste des symboles d'indice de chaîne permettant de faire des opérations de manipulations sur les chaînes de caractères pour les plates-formes MAC et PC.

Windows			[[ ]]
Mac	≤ ≥	ou	[[ ]]

sur mac, le caractère	"≤"	s'obtient avec les touches "alt + >"
et le caractère	"≥"	avec les touches "alt + shift + >"

Exemple :

```
[contacts]initiale:=[contacts]nom[[1]]
        affecte le premier caractère de [contacts]nom, au champ [contacts]initiale
[contacts]nom[[1]]:=Majusc( [contacts]nom[[1]])
        capitalise la première lettre du nom
```

Voici la liste détaillée des principales fonctions de manipulations de chaînes qu'offre 4D :

Il faut souligner que ce thème est à titre de rappel et que vous trouverez des informations complémentaires dans la documentation sous le thème "Chaînes de caractères". Il est très important de connaître toutes les subtilités de 4D concernant le traitement des chaînes comme il est indiqué dans cette note technique sous la rubrique "précision".

Chaîne

Chaine (expression{; format}) -> Alpha

Paramètre	Type	Description

expression	Alpha	->	Expression à convertir en chaîne (peut être de type Numérique, Entier, Entier long, Date ou Heure)
format	Alpha | Num	->	Format d'affichage
résultat	Alpha	<-	expression convertie en chaîne alphanumérique

La fonction Chaine retourne sous forme de chaîne alphanumérique l'expression de type numérique, Date ou Heure, que vous avez passé dans le paramètre "expression". Il est possible aussi, grâce au "format d'affichage", d'afficher une expression à la place d'une valeur numérique.

exemple : Chaine(Num(1=1);"Vrai;;"Faux") renverra comme résultat la chaîne "Vrai".

Note : reportez vous au manuel pour tous les types de formatage.

Num

Num (expression) -> Numérique

Paramètre	Type		Description
expression	Alpha | Booléen	->	Chaîne à convertir en numérique ou Booléen à convertir en 0 ou 1
résultat	Numérique	<-	Valeur numérique d'une chaîne ou d'un booléen

La fonction Num retourne sous forme de numérique l'expression de type Alpha ou Booléen que vous avez passé dans le paramètre "expression".

Note : Seuls les 32 premiers caractères de "expression" sont pris en compte.

Attention : Il existe trois caractères réservés que Num traite de manière particulière. Il s'agit du séparateur décimal (c'est-à-dire la virgule (,) dans une version française), du tiret (-) et du e (ou E). Ils seront interprétés en tant que caractères de formatage des nombres.

Position

Position (àChercher; **Chaine**) Δ Numérique

Paramètre	Type		Description
àChercher	Alpha	->	Chaîne à rechercher
chaîne	Alpha	->	Chaîne dans laquelle effectuer la recherche

résultat	Numérique	<-	Position de la première occurrence de àChercher

Position retourne la position de la première occurrence de "àChercher" dans chaîne. Si "àchercher" est une chaîne, Position retourne le premier caractère de la première occurrence de cette chaîne. Cette fonction est souvent à la base des fonctions de manipulations de chaînes que vous écrirez.

Précision : Si chaîne ne contient pas "àChercher", la fonction retourne zéro (0).
Si Position trouve une occurrence de "àChercher", la fonction retourne la position du premier caractère de cette occurrence dans "chaîne".
Si vous demandez la position d'une chaîne vide à l'intérieur d'une chaîne vide, Position retourne zéro (0).

Attention : Vous ne pouvez pas utiliser le caractère joker (@) avec Position. Si, par exemple, vous passez "abc@" dans "àChercher", la fonction recherchera effectivement la chaîne "abc@" et non pas "abc suivi de toute valeur".

Sous chaine

Sous chaine (source; àPartirDe{; nbCars}) ->Alpha

Paramètre	Type		Description
source	Alpha	->	Chaîne de laquelle extraire une sous-chaîne
àPartirDe	Numérique	->	Position du premier caractère
nbCars	Numérique	->	Nombre de caractères à extraire
résultat	Alpha	<-	Sous-chaîne de source

La fonction Sous chaine retourne la partie de source délimitée par les paramètres àPartirDe et nbCars.

Précision : Si "nbCars" n'est pas défini ou si le total de "àPartirDe" plus "nbCars" est supérieur au nombre de caractères de la chaîne "source", Sous chaine retourne tous les caractères de la chaîne à partir du caractère spécifié par "àPartirDe". Si "àPartirDe" est supérieur au nombre de caractères de la chaîne "source", Sous chaine retourne une chaîne vide ("").

Longueur

Longueur (**Chaine**) ->Numérique

Paramètre	Type		Description
chaîne	Alpha	->	Chaîne dont vous voulez connaître la

longueur

résultat	Numérique	<-	Nombre de caractères de chaîne

Longueur vous permet d'obtenir la longueur d'une chaîne. Longueur retourne le nombre de caractères alphanumériques contenus dans "chaîne".

Code ascii

Code ascii (**Caractere**) ->Numérique

Paramètre	Type		Description
caractère	Alpha	->	Caractère dont vous voulez obtenir le code ASCII
résultat	Numérique	->	Code ASCII de caractère

La commande Code ascii retourne le code ASCII de caractère. Si la chaîne caractère comporte plus d'un caractère, Code ascii retourne uniquement le code du premier caractère.

Précision : 4D ne respecte pas la casse des caractères c'est-à-dire que pour 4D "a" est égal à "A" dans une comparaison ou une recherche de chaîne. Cependant le code ASCII de "a" est différent de celui de "A". C'est pourquoi vous devez utiliser cette fonction si vous voulez respecter la casse des chaînes de caractères.

Caractere

Caractere (codeASCII) ->**Chaine**

Paramètre	Type		Description
codeASCII	Numérique	->	Code ASCII de 0 à 255
résultat	Chaîne	->	Caractère représenté par codeASCII

La fonction Caractere retourne le caractère dont le code ASCII est "codeASCII".

Astuce : La fonction Caractere est généralement utilisée pour insérer des caractères qui ne peuvent être saisis au clavier ou des caractères de contrôle dans l'éditeur de méthodes.

Important : dans 4D, toutes les valeurs de texte, champs ou variables, utilisent la table ASCII de MacOS, sur les plates-formes Macintosh et Windows ? si aucune conversion vers une autre table ASCII n'a été effectuée.

Majusc

Majusc (**Chaine**) ->Alpha

Paramètre	Type		Description
chaîne	Alpha	->	Chaîne à convertir en majuscules
résultat	Alpha	<-	chaîne en majuscules

Majusc retourne une chaîne de caractères égale à "chaîne" dont tous les caractères alphabétiques ont été convertis en majuscules.

Minusc

Minusc (**Chaine**) ->Alpha

Paramètre	Type		Description
chaîne	Alpha	->	Chaîne à convertir en minuscules
résultat	Alpha	<-	chaîne en minuscules

Minusc retourne une chaîne de caractères égale à "chaîne" dont tous les caractères alphabétiques ont été convertis en minuscules.

Remplacer caracteres

Remplacer caracteres (source; nouveau; **Position**) ->Alpha

Paramètre	Type		Description
source	Alpha	->	Chaîne de départ
nouveau	Alpha	->	Nouveaux caractères
position	Numérique	->	Position de départ du remplacement
résultat	Alpha	<-	Chaîne résultante

Remplacer caracteres retourne une chaîne résultant du remplacement des caractères, dans la chaîne "source", à partir de "position", par la chaîne "nouveau".

Précision : si "nouveau" est une chaîne vide (""), Remplacer caracteres retourne "source" inchangé. Remplacer caracteres retourne toujours une chaîne de la même longueur que "source". Si "position" est inférieure ou supérieure à la longueur de "source", Remplacer caracteres retourne "source".

Inserer chaine

Inserer chaine (source; insertion; **Position**) ->Alpha

Paramètre	Type		Description
source	Alpha	->	Chaîne dans laquelle effectuer l'insertion
insertion	Alpha	->	Chaîne à insérer dans source
position	Numérique	->	Position de l'insertion
résultat	Alpha	<-	Chaîne résultante

Inserer chaine insère la chaîne de caractères alphanumériques "insertion" dans la chaîne "source" à partir de "position" et retourne la chaîne de caractères résultante. La chaîne "insertion" est placée avant le caractère désigné par position.

Précision : si "insertion" est une chaîne vide (""), Inserer chaine retourne "source" inchangé.
Si "position" est supérieure à la longueur de "source", "insertion" est ajouté à la fin de "source".
Si "position" est inférieure à un (1), "insertion" est inséré au début de "source".

Supprimer chaine

Supprimer chaine (source; **Position**; nombreCar) ->Alpha

Paramètre	Type		Description
source	Alpha	->	Chaîne de départ
position	Numérique	->	Premier caractère à supprimer
nombreCar	Numérique	->	Nombre de caractères à supprimer
résultat	Alpha	->	Chaîne résultante

Supprimer chaine supprime "nombreCar" dans "source" à partir de "position" et retourne la chaîne résultante.

Précision : Supprimer chaine retourne la même chaîne que "source" dans les cas suivants.
"source" est une chaîne vide,
"position" est supérieure à la longueur de "source",
"nombreCar" est égal à zéro (0).

Si "position" est inférieure à un (1), les caractères sont supprimés à partir du début de la chaîne.
Si "position" + "nombreCar" est supérieur ou égal à la longueur de "source", les caractères sont supprimés à partir de "position" jusqu'à la fin de "source".

Remplacer chaine

Remplacer chaine (source; obsolète; nouveau{; remplacements}) ->Alpha

Paramètre	Type		Description
source	Alpha	->	Chaîne de départ
obsolète	Alpha	->	Caractère(s) à remplacer
nouveau	Alpha	->	Chaîne de remplacement (si chaîne vide, toutes les occurrences sont effacées)
remplacements	Numérique	->	Nombre de remplacements à effectuer
Résultat	Alpha	<-	Chaîne résultante

Précision : Remplacer chaine retourne une chaîne de caractères résultant du remplacement dans "source" de "obsolète" par "nouveau".
Si "nouveau" est une chaîne vide (""), Remplacer chaine supprime chaque occurrence de "obsolète" dans "source".
Si "remplacements" est spécifié, Remplacer chaine ne remplace que le nombre d'occurrences de "obsolète" spécifié, à partir du premier caractère de "source". Si "remplacements" est omis, toutes les occurrences de "obsolète" seront remplacées.
Si "obsolète" est une chaîne vide, Remplacer chaine retourne "source" inchangé.

Utilisation des fonctions

Le code ci-dessous montre comment mettre en majuscule la première lettre du prénom d'un client.

```
[client]prenom:=Majusc([client]prenom[[1]]) `sous windows
[client]prenom:= Majusc([client]prenom≤1≥) `sous mac
```

Cependant si vous voulez écrire des fonctions plus complexes, il faudra utiliser des méthodes projets. L'exemple ci-dessous montre l'utilisation de la fonction Majusc dans une méthode projet. Cette méthode recevra comme paramètre la chaîne à transformer et retournera la chaîne en majuscule.

Méthode projet EN_MAJUSCULE

```
$0:=Majusc($1[[1]]) `sous Windows

$0:= Majusc($1≤1≥) `sous MacOS
```

Précision : $1 est l'unique paramètre de la fonction qui contient la chaîne à transformer.
$0 contiendra le résultat de la fonction : chaîne en majuscule.

La fonction sera appelée de la manière suivante :
[client]prenom:=*EN_MAJUSCULE*([client]prenom)

Pourquoi utiliser des fonctions ?

Le fait d'écrire des fonctions, rend votre structure plus souple. En effet quand vous écrivez une fonction, ce sont des paramètres que vous lui transmettez et non des valeurs fixes. De plus si vous utilisez 4D insider vous pouvez copier facilement vos méthodes projets d'une base à une autre. Il est même possible, avec 4D insider, de créer une ou des bibliothèques de code que vous pouvez insérer dans tout autre base.

L'autre intérêt majeur est que vous pouvez changer le code à l'intérieur de votre fonction (méthode projet) sans changer la structure de votre base.

Méthodes avancées

Suppression d'espace en début de chaîne :

Pendant la saisie, il arrive parfois que l'utilisateur saisisse des espaces, (involontairement), devant les données. Ceci a pour effet de fausser le résultat des recherches effectuées par la suite car les caractères (espaces) ne font pas partie de la recherche.

La méthode projet *Supp_Blanc_Deb_Ch* permet de supprimer les espaces en tête d'une chaîne.

Précision : $1 est l'unique paramètre de la méthode qui contient la chaîne contenant des espaces en trop (au début) à transformer. La chaîne peut avoir de zéro à un nombre infini d'espaces, dans son en-tête.
$0 contiendra la chaîne avec les espaces supprimés

Cette fonction peut être appelée de la manière suivante :
[contacts]nom:=*Supp_Blanc_Deb_Ch*([contacts]nom)

```
C_TEXTE($0;$1)

C_ENTIER($pos;$longch)
C_BOOLEEN($ok)

$pos:=0
$longch:=Longueur($1)
$ok:=Vrai
Tant que ($ok & ($pos<$longch))
```

```
 Si ($1≤$pos+1≥#" ")
  $ok:=Faux
 Sinon
  $pos:=$pos+1
 Fin de si
Fin tant que

$0:=Sous chaine($1;$pos+1;$longch-$pos)

 `cas particulier si dans l'extraction de la sous chaine "a partir de" est
 `superieur a la longueur de la chaine, 4D renvoie une chaine vide. C'est le cas

 ` quand $1 contient une chaine vide
```

Suppression d'espace en fin de chaîne

Pour les mêmes raisons, la méthode projet Supp_Blanc_Fin_Ch permet de supprimer les espaces en fin de chaîne.

Précision : $1 est l'unique paramètre de la méthode qui contient la chaîne contenant des espaces en trop (à la fin) à transformer. La chaîne peut avoir de zéro à un nombre infini d'espaces en fin de chaîne.
$0 contiendra la chaîne avec les espaces supprimés

Cette fonction peut être appelée de la manière suivante :
[contacts]nom:=Supp_Blanc_Fin_Ch([contacts]nom)

```
C_TEXTE($0;$1)

C_ENTIER($pos)
C_BOOLEEN($ok)

$pos:=Longueur($1)
$ok:=Vrai
Tant que ($ok & ($pos>0))
 Si ($1≤$pos≥#" ")
  $ok:=Faux
 Sinon
  $pos:=$pos-1
 Fin de si
Fin tant que

$0:=Sous chaine($1;1;$pos)

 `cas particulier si dans l'extraction de la sous chaine "a partir de" est
 `superieur a la longueur de la chaine, 4D renvoie une chaine vide. C'est le cas

 ` quand $1 contient une chaine vide
```

Décomposition d'une phrase en mots

La méthode projet Decomp_En_Mot_Ds_Tab permet de décomposer une phrase en mots. Cela peut être très utile, par exemple, si vous voulez importer des mails et les découper sous forme de mots clés pour en faire un index, ou les stocker sous une forme particulière. La difficulté de ce principe est la détection des mots à l'intérieur d'une chaîne. La meilleure façon de résoudre cette difficulté est de détecter les mots puis les stocker dans un tableau.

Précision : $1 correspond à la phrase à découper
$2 est le tableau qui contiendra les mots de la phrase
$3 est le séparateur de mots

Cette fonction peut être appelée de la manière suivante :
Decomp_En_Mot_Ds_Tab (variable1;->Tab_Mots;" ")

```
C_TEXTE($1)

C_POINTEUR($2)
C_ALPHA(1;$3)
C_TEXTE($text)
C_ENTIER($pos_separateur)
C_ENTIER LONG($tt)

TABLEAU TEXTE($2->;0)

$text:=$1

Repeter
 $pos_separateur:=Position($3;$text)
 Si ($pos_separateur#0)
  $tt:=Taille tableau($2->)+1
  INSERER LIGNES($2->;Taille tableau($2->)+1)
  $2->{Taille tableau($2->)}:=Sous chaine($text;1;Position($3;$text)-1)
  $text:=Supprimer chaine($text;1;$pos_separateur)
 Fin de si
Jusque ($pos_separateur=0)

Si ($text#"")
 INSERER LIGNES($2->;Taille tableau($2->)+1)
 $2->{Taille tableau($2->)}:=$text

Fin de si
```

Une fois le tableau construit, il vous est possible de créer l'index. Grâce à ce tableau, il est possible de ne pas prendre en compte tous les éléments. Les éléments non traités lors de la création de l'index pouvant être sauvegardés dans un autre tableau pour un autre traitement.

Respecter la casse

L'une des tâches, couramment effectuée dans le traitement des chaînes de caractères, consiste à mettre la première lettre d'un mot en majuscule. Ceci ne pose pas de problème particulier avec les commandes de 4D.

Cependant certains mots demandent un traitement particulier en termes de transformation de certaines lettres en majuscule .
Par exemple les mots suivants : MacOS, McDougal, Scottish, NATTO, 4D.

Ceci implique la construction d'une fonction de vérifications et de transformation de chaîne en respectant la casse. Mais ceci ne permettra pas à l'utilisateur de conserver les mots déjà traités.
Pour optimiser cette méthode, l'une des solutions serait d'enregistrer les mots "pré-formatés" dans un fichier de paramètres, sous forme de dictionnaire que l'on pourra maintenir, afin de le consulter et d'effectuer des comparaisons.
Pour cet exemple, le dictionnaire contiendra les mots suivants : "McDougal, NATO, MacOS, 4D" dans la table "préférences" et la méthode projet se nomme En_Capital_Avec_Casse.

Précision : $1 correspond à l'ensemble des mots parmi lesquels il faut vérifier la casse.
$0 correspond à la chaîne traitée.

Cette fonction peut être appelée de la manière suivante :
maChaine:=En_Capital_Avec_Casse(variable1)

```
C_TEXTE($0;$1;$text)
C_ENTIER  LONG($trouve;$compteur_mot)

  `première étape transformer les mots de la phrase en minuscules
$text:=Minusc($1)


$0:=""
```

`deuxième étapes découper la phrase à traiter en un tableau de mots : ◊mots
Decomp_En_Mot_Ds_Tab ($text;->◊mots;" ")

```
  `tropisième étape chercher les mots dans le dictionnaire et remplacer

  `les mots par les mots dont la casse est respectée.
Boucle ($compteur_mot;1;Taille  tableau(◊mots))
 $trouve:=Chercher  dans  tableau(◊Mot_En_Capital;◊mots{$compteur_mot})
 S i ($trouve>0)
    `si le mot est trouvé dans le dictionnaire remplacement de celui ci
  ◊Mots{$compteur_mot}:=◊Mot_En_Capital{$trouve}
 Fin  de  si
 $0:=$0+◊Mots{$compteur_mot}+" "
Fin  de  boucle
  `on enleve l'espace situé derrière le dernier mot de la phrase

$0:=Supp_Blanc_Fin_Ch ($0
```

Reconstruction d'un texte sous forme d'un paragraphe

La méthode projet Reconst_Paragraph permet de reconstruire un paragraphe à partir d'un texte contenant des retours à la ligne à l'intérieur du paragraphe logique.
Vous pourrez rencontrer ce type de structure en important, par exemple, des mails. En effet, les messages qui transitent par le WEB (World Wide Web), sont découpés sous forme de petits paquets de textes séparés par un retour chariot. Il faudra donc à la réception dans 4D, reconstruire les paragraphes. Cette technique s'appelle le wrapping.

Cette méthode va permettre à l'utilisateur de sélectionner une partie de texte à l'intérieur d'une zone texte et de la remettre sous la forme d'un paragraphe.

Précision : Le fait d'utiliser la commande "TEXTE SELECTIONNE" vous oblige à utiliser la méthode dans le script de l'objet texte lui-même.

```
C_POINTEUR($1)

C_TEXTE($text;$text_a_traiter)
C_ALPHA(2;$rc)

TEXTE  SELECTIONNE($1->;$debsel;$finsel)
$text_a_traiter:=Sous  chaine($1->;$debsel;($finsel-$debsel))
$text:=Supprimer  chaine($1->;$debsel;($finsel-$debsel))
$rc:="."+Caractere(Retour chariot )
$text_a_traiter:=Remplacer  chaine($text_a_traiter;$rc;Caractere(240))
$text_a_traiter:=Remplacer  chaine($text_a_traiter;Caractere(Retour chariot );" ")
$text_a_traiter:=Remplacer  chaine($text_a_traiter;Caractere(240);$rc)
$text_a_traiter:=Remplacer  chaine($text_a_traiter;(Caractere(Retour chariot )+"
");Caractere(Retour chariot ))

$0:=Inserer  chaine($text;$text_a_traiter;$debsel)
```

Créer un titre à partir des premières lignes d'un texte

La méthode projet *Creation_titre* permet de créer un titre à partir du début d'un texte. En effet, si vous importez un bloc de texte dans un champ, il est probable que vous voudrez renseigner un champ connexe avec un titre résumant le contenu du texte. Voici une méthode simple qui consiste à récupérer la première phrase du texte en détectant le premier retour chariot dans le texte. Éventuellement vous pouvez réduire la longueur du titre.

Précision : cette méthode reçoit un ou deux paramètres
$1 texte à traiter.
$2 longueur maximale du titre (optionnel)
$0 retourne le titre.

```
C_TEXTE($0;$1)

C_ENTIER($2)
C_ENTIER($pos_rc)
```

```
$pos_rc:=Position(Caractere(Retour chariot );$1)
Au cas ou
 : (Nombre de parametres=1)
 Si ($pos_rc>0)
  $0:=Sous chaine($1;1;($pos_rc-1))
 Sinon
  $0:=$1
 Fin de si

 : (Nombre de parametres=2)
 Si (($pos_rc>0) & ($pos_rc<=$2))
  $0:=Sous chaine($1;1;($pos_rc-1))
 Sinon
  $0:=Sous chaine($1;1;$2)
 Fin de si

Fin de cas
```

Les fonctions suivantes sont destinées au traitement statistique des chaînes.

Compter le nombre de phrases dans un texte

La méthode projet Compter_Nb_Phrases permet de compter le nombre de phrases dans un texte. Une phrase étant définie comme suit : un ensemble de mots qui se terminent par un point et suivis par un espace ou un retour chariot.

```
C_TEXTE($1)

C_ENTIER LONG($phrase;$0)
C_ENTIER($pos)

Si (Longueur($1)>0)
 Boucle ($pos;1;(Longueur($1)-1))
  Si ($1≤$pos≥=".")
   Si (($1≤$pos+1≥=Caractere(Espacement )) | ($1≤$pos+1≥=Caractere(Retour chariot )))
    $phrase:=$phrase+1
   Fin de si
  Fin de si
 Fin de boucle
 Si (($1≤Longueur($1)≥)=".")
  $phrase:=$phrase+1
 Fin de si
 $0:=$phrase
Sinon
 $0:=-1

Fin de si
```

Compter le nombre de caractères excepté les espaces

La méthode projet Compter_Car_Dif_Espace compte le nombre de caractères dans une chaîne excepté les espaces.

```
C_TEXTE($1)

C_ENTIER  LONG($nb_car)

$nb_car:=0
Boucle ($pos;1;Longueur($1))
 Si ($1≤$pos≥#Caractere(Espacement ))
  $nb_car:=$nb_car+1
 Fin de si
Fin de boucle

$0:=$nb_car
```

Compter le nombre de mots dans un texte (méthode simple)

La méthode projet Compter_Nb_Mots compte le nombre de mots dans un texte. Un mot étant défini comme suit : un ensemble de caractères entre deus espaces.

```
C_ENTIER  LONG($nb_mots;$0)

C_TEXTE($1;$text)
C_ENTIER($lg_text;$pos)

$text:=$1
$nb_mots:=0
$debut_mot:=Faux
$lg_text:=Longueur($text)

Boucle ($pos;1;$lg_text)
 Si ($debut_mot)
  Si (($text≤$pos≥=Caractere(Espacement )) | ($text≤$pos≥=Caractere(Retour chariot )))
   $debut_mot:=Faux
  Fin de si
 Sinon
  Si ($text≤$pos≥#Caractere(Espacement ))
   $nb_mots:=$nb_mots+1
   $debut_mot:=Vrai
  Fin de si
 Fin de si
Fin de boucle

$0:=$nb_mots
```

Précision : cette méthode ne compte pas le premier mot situé après un retour chariot si le mot situé après le retour chariot n'est pas séparé de celui-ci par un espace. Ce qui signifie que le premier mot de chaque nouveau paragraphe n'est jamais compté. Une manière de contourner le problème est de compter les retours chariots et

les espaces. Il faudra aussi penser à traiter le cas d'espaces ou retours chariots multiples, séparant deux paragraphes.

Compter le nombre de paragraphes

La méthode projet Compter_Nb_Para compte le nombre de paragraphe dans un texte. Un paragraphe étant définie comme suit : un ensemble de mots qui se terminent par un retour chariot à la fin du texte. Le code devra être modifié en conséquence si l'utilisateur intègre plusieurs retours chariots dans un même paragraphe.

```
C_TEXTE($1;$text)
C_ENTIER  LONG($nb_paras)

$text:=$1
Si ($text#"")
 $nb_paras:=0
 Boucle ($pos;1;Longueur($text))
  Si ($text≤$pos≥=Caractere(Retour chariot ))
   $nb_paras:=$nb_paras+1
  Fin de si
 Fin de boucle
 $nb_paras:=$nb_paras+1
 $0:=$nb_paras
Sinon
 $0:=0
Fin de si
```