TRAITEMENT DE TEXTE
PROGRAMMATION
LE RETOUR

SASS Ferenc
Mande Saint Etienne, 923
6688 Longchamps
Belgique
Tél.: (061) 21.34.16
Fédération de l'Enseignement Secondaire Catholique
Service Informatique
Bruxelles

1 VERS UN NOUVEAU PARADIGME?

1.1 Avertissement

Ce document, sans aucune prétention méthodologique, est destiné à des lecteurs, essentiellement du monde de l'enseignement, qui ont déjà une connaissance du paradigme des langages de programmation procéduraux. Ceci explique que tous les concepts liés à ce paradigme ne seront que cités et non développés dans le texte.

L'hypothèse de travail est d'essayer de montrer qu'il est possible de déterminer un ensemble d'actions de base communes aux langages de programmation intégré actuellement aux traitements de texte et dès lors de concevoir des algorithmes et programmes traduisible pour n'importe quel traitement de texte programmable.

Une première partie fait un relevé des éléments fondamentaux du paradigme des langages de programmation des traitements de texte. Dans une deuxième partie, des exercices illustrant les différents concepts abordés sont proposés.

1.2 Le bricolage mène à tout

Lorsqu'il s'agit de poser un plafond de lambris, vous savez certainement qu'à un moment donné, il faudra répéter les actions ci–dessous:

- prendre un clou,
- placer le clou à la bonne position,
- enfoncer le clou avec un marteau.

Face à cette répétition, je distingue deux attitudes:

- celle du bricoleur *courageux méthodique* qui va répéter des centaines de fois ces actions,

•celle du bricoleur *paresseux raisonné* qui va soit essayer d'inventer soit se procurer une machine qui va prendre en charge certaines actions: le marteau automatique qui prend le clou et l'enfonce, le positionnement du clou restant pour le bricoleur. Pourquoi ne pas imaginer un automate qui enfoncerait un clou tous les 20 cm?

Il existe de nombreux domaines d'activités de l'homme pour lesquelles les actions répétitives sont confiées à une machine. Lors de l'utilisation des outils informatiques, je distinguerai aussi deux type d'utilisateurs:

•le courageux méthodique qui acceptera consciemment ou non de répéter les instructions données au logiciel autant de fois qu'il faut,
•le paresseux raisonné qui tentera de faire faire les tâches répétitives par le logiciel.

Si c'est pour ces derniers que les macro–instructions ont un intérêt majeur, on peut espérer que les courageux méthodiques se transformeront progressivement en paresseux raisonnés.

1.3 Les macro–instructions, un essai de définition

Si on accepte l'idée qu'un logiciel est un exécutant auquel l'utilisateur est amené à donner des ordres, des instructions, sous quelles que formes que ce soit, alors une macro–instruction est une nouvelle instruction, inconnue de l'exécutant et qui contient la description de l'enchaînement d'instructions connues de ce dernier. On peut comparer le concept de macro–instruction aux concepts de procédures et fonctions en programmation.

Toute macro–instruction devra recevoir un nom et un descriptif de l'enchaînement des actions à faire exécuter.

Sans entrer dans des considérations commerciales, publicitaires ou autres, on peut essayer de répartir les nombreux traitements de texte disponibles sur le marché en trois grandes catégories.

•Les traitements de texte avec un compilateur ou interpréteur de commandes,
•les traitements de texte avec un générateur de macro–instructions,
•les traitements de texte qui n'ont ni l'un, ni l'autre.

Dans les pages suivantes, je ne considérerai que les traitements de texte faisant partie des deux premières catégorie. Les plus connus actuellement sont : WordPerfect dans les versions Dos, Windows et MacIntosh, Word dans les versions Dos, Windows et MacIntosh, Ami–Pro dans sa version Windows.

Si on définit un invariant comme l'ensemble des concepts valables dans le temps pour les logiciels actuellement disponibles, j'essayerais, dans la mesure du possible, de déterminer l'ensemble des invariants du paradigme de programmation généré par ce type de traitement de texte.

1.4 Il y a macro–instruction et macro–instruction

Pour le concepteur d'une macro–instruction, les étapes d'élaboration sont les suivantes:

•déterminer ce que la macro instruction devra faire. C'est l'énoncé du travail à faire faire. Cet énoncé devra toujours être éclairci,
•élaborer l'enchaînement des actions à faire faire par le traitement de texte lors de la demande d'exécution de la macro–instruction. Par définition même, cette exécution est différée.

•traduire cet enchaînement dans le langage propre au traitement de texte. On obtient de la sorte, un texte,
•encoder le texte ainsi obtenu,
•donner l'ordre d'exécution de la macro–instruction. Lors de cette demande, l'interpréteur, ou le compilateur, se charge de la traduction de la macro–instruction dans un langage exécutable par la machine.
•vérifier l'adéquation du résultat obtenu à l'énoncé du travail. Normalement, cette étape devrait être inutile, si l'enchaînement a été correctement prévu.
•ne pas avoir peur de recommencer

Le lecteur averti observera que ces quelques principes sont aussi applicables à l'élaboration d'un programme. On peut affirmer qu'une macro–instruction est en réalité un programme.
Toutefois, on peut distinguer deux types de macro–instructions, en fonction de la méthode d'encodage de l'enchaînement des actions. Le noms attribués à ces types varient d'un traitement de texte à l'autre, et j'adopterai les conventions suivantes:
•les macro–instructions "générées",
•les macro–instructions "programme".

1.4.1 Les macro–instructions générées

Lors de l'encodage de l'enchaînement des instructions de la macro–instruction, le concepteur ne doit pas connaître et utiliser le langage de programmation du traitement de texte. Il suffit de signaler au traitement de texte que
•toutes les instructions données via les différents périphériques d'entrée doivent être enregistrées,
•signaler quand cet enregistrement se termine,
•le traitement de texte générera le texte correspondant aux ordres donnés.

Description des actions	Consignes avec WP 6.0	Consignes avec Word 2.0
Signaler le début de l'enregistrement	Ctrl–f10	Alt, O, R
	il faut fournir le nom de la macro–instruction	
A partir de ce moment, toutes les instructions données au traitement de texte sont enregistrées, qu'elles soient "bonnes" ou "mauvaises" par rapport au travail à faire faire.		
Signaler la fin de l'enregistrement	Ctrl–f10	Alt, O, R

Exemple
les traitements de texte utilisent de plus en plus des interfaces graphiques FIMS (Fenêtres, Icônes, Menus, Souris) pour dialoguer avec l'utilisateur. Si ce type de dialogue est souvent convivial, il n'en nécessite pas moins une série de sélections de menus, icônes, boutons et autres pour faire exécuter une action par le traitement de texte. Il est alors utile de remplacer cette enchaînement de sélections par une macro–instruction générée:
•texte d'entête de page,
•texte de pied de page,
•encadrer un paragraphe, ...

1.4.2 Les macro–instructions programmes

Contrairement aux macro–instructions générées, l'encodage des instructions des macro–instructions programme est effectué à l'aide d'un éditeur de texte, comme pour la majorité des langages de programmation. L'éditeur de texte est ici le traitement de texte lui–même. Le nombre d'instructions, leur variété et leur syntaxe souvent lourde a forcé les concepteurs à fournir des aides directes lors de l'encodage. Le texte encodé sera interprété ou compilé.

Description des actions	Consignes avec WP 6.0	Consignes avec Word 2.0
Signaler le début de l'encodage	Ctrl–f10	Alt, O, M
	il faut fournir le nom de la macro–instruction	
Signaler la fin de l'encodage et l'enregistrement de la macro–instruction	F7 ou F10	ALT, F, F

Les macro–instruction–programmes sont souvent des programmes dont les instructions ne sont pas facilement générées par le traitement de texte. Des exemples de ce type de macro–instructions sont données dans les exercices.
Le traitement de texte pourra aussi être utilisé pour corriger le texte d'une macro–instruction générée.

1.5 Exécution des macro–instructions

Lors de la demande d'exécution des macro–instructions, elles sont soient interprétées soit compilées par l' interpréteur/compilateur intégré au traitement de texte. L'ordre d'exécution est souvent donné par l'appel de la macro–instruction par son nom, comme une procédure en programmation, soit par un raccourci clavier correspondant à cet appel.

Description des actions	Consignes avec WP 6.0	Consignes avec Word 2.0
Demande d'exécution de la macro–instruction	Alt–f10 ou un raccourci–clavier	Alt,O,M,E ou un raccourci–clavier
	il faut fournir le nom de la macro–instruction	

1.6 Le paradigme de programmation

D'une manière générale, les langages de programmation des traitements de texte font parties des langages dit procéduraux, même si dans les environnements graphiques, les instructions peuvent parfois adresser des messages à des objets définis. J'ai imaginé un langage formel de description d'algorithme construit à partir du langage formel habituel pour les algorithmes du paradigme procédural. Il est tout à fait imaginable d'utiliser des GNS ou d'autres langages de description. Dans la mesure du possible, je reprendrai la traduction de cette description dans les deux langages de programmation WordPerfect 6.0 pour DOS et Word 2.0 pour Windows, sans donner les détails de syntaxe. Les cellules vides dans les cadres indiquent, à ma connaissance, l'abscence de l'instruction, sauf information contraire.

Je distinguerai différentes familles d'actions:

- •celles liées aux actions de base du paradigme de programmation procédurale,
- •celles liées aux informations et structures de données,
- •celles liées à l'appel des procédures et fonctions,

•celles liées aux actions de base du traitement de texte

1.6.1 Les actions de base liées au paradigme de programmation

Déclaration de variables globales, locales

Description de l'action	Consignes pour son exécution avec WP 6.0	Consignes pour son exécution avec Word 2.0
Déclaration d'une variable locale	Local(noms des variables)	
Déclaration d'une variable globale	Global(noms des variables)	Dim Shared noms des variables

L'affectation

_VAR <– valeur

Description de l'action	Consignes pour son exécution avec WP 6.0	Consignes pour son exécution avec Word 2.0
Affectation d'une valeur à une variable	Assign(variable,valeur) ou variable = valeur	Let variable = valeur

L'entrée

_LIRE VAR

La traduction de cette action dépend du type de valeur et du périphérique d'entrée utilisé.

Description de l'action	Consignes pour son exécution avec WP 6.0	Consignes pour son exécution avec Word 2.0
Lecture d'une valeur à partie du clavier	Getstring(variable,message,tit re) Getnumber(variable,message, titre) Getunits(variable,message,titr e) Input(message)	Input Inputbox$ Lineinput Input$
Lecture d'une valeur à partir d'un support disque	Fileretrieve Fileopen	Insertfile

Sortie

_ECRIRE valeur, VAR

Les résultats des traitements sont souvent insérés dans le texte sur lequel la macro–instruction agit, même si dans certains cas, les valeurs peuvent être envoyées vers un périphérique de sortie(imprimante, fichier séquentiel, ...).

Description de l'action	Consignes pour son exécution avec WP 6.0	Consignes pour son exécution avec Word 2.0
Insérer la valeur d'une variable dans le texte qui est en mémoire centrale (voir à ce sujet les structures de données)	Type(variable)	Insert variable
Ecrire vers un support externe	Filesave Appendtofile	Filesave Print

Les actions d'entrée et sortie sont souvent utilisées pour dialoguer avec un utilisateur. Les traitements de texte intègrent dans leur langage de programmation un ensemble de primitives de gestion de boîtes de dialogue qui améliorent l'ergonomie générale de l'interface homme–machine. Ces boîtes de dialogue sont généralisées avec les logiciels fonctionnant sous des systèmes d'exploitation avec interface graphique tels Window ou System Apple. Je ne reprendrai pas ici toutes les possibilités de gestion de ces boîtes de dialogue.

1.6.2 les opérateurs et fonctions associés aux types de données: caractère, numérique, logique.

Description de l'action	Consignes pour son exécution avec WP 6.0	Consignes pour son exécution avec Word 2.0
Numérique	espace pour séparer les opérandes et opérateurs	
+ addition	+	+
– soustraction	–	–
* multiplication	*	*
/ division	/	/
^ exponentiation		^

fonctions mathématiques diverses	La liste est très variable, allant des fonctions goniométriques aux fonctions logarithmes et exponentielles, en passant par les calculs de valeurs d'intérêts, ...	
Chaîne de caractères	espace pour séparer les opérandes et opérateurs	
+ (concaténation)	+	+
longueur	Strlen(chaîne)	Len(chaîne)
Extraction de sous–chaîne	Substr(chaîne,pos,long)	Mid$(chaîne,pos,long)
Valeur(chaîne)	Strnum(chaîne)	Val(chaîne)
Convertir_en_majuscules	Toupper(chaîne)	Changecase(1), Ucase$(chaîne)
Convertir_en_minuscules	Tolower(chaîne)	Changecase(0)
Convertir_en_capitale	Caps(chaîne)	Changecase(2)
Booléen		
Non	NOT	NOT
Et	AND	AND
Ou	OR, XOR	OR
Opérateurs de comparaison		
plus grand	>	>
plus petit	<	<
égal	=	=
différent	<>	<>
plus grand ou égal	>=	>=
plus petit ou égal	<=	<=

1.6.3 les variables du système

Les développeurs des traitements de texte utilisent un nombre important de variables dans leurs programmes. La majorité des valeurs de ces variables sont accessibles, voire modifiables par les macro–instruction–programmes conçus par le programmeur de la macro–instruction. Ces variables sont appelées les variables système. Leur nombre est tellement élevé que le tableau ci– dessous ne reprendra, à titre d'exemples, qu'une liste réduite de ces variables.

Description de l'action variable système	Consignes pour son exécution avec WP 6.0	Consignes pour son exécution avec Word 2.0
valeur_margegauche(n)	MarginLeft	FormatPageSetup
valeur_margedroite	MarginRight	FormatPageSetup
valeur_margesupérieure	Margintop	FormatPageSetup
valeur_margeinférieure	MarginBottom	FormatPageSetup
valeur_interligne	LineSpacing	FormatParagraph
valeur_interparagraphe	ParagraphSpacing	FormatParagraph
valeur_numeropage	Pagenumber	FormatPagenumber, InsertPageNumber
valeur_taillepolice	FontSize	FontSize
valeur_nompolice	Font	Font
valeur_style	StyleOn	FormatStyle

Certaines variables systèmes sont utilisées par l'interpréteur/compilateur et ont une influence non négligeable sur l'exécution des macro–instruction.

Description de l'action variable système	Consignes pour son exécution avec WP 6.0	Consignes pour son exécution avec Word 2.0
Trouvé? (valeur logique, elle indique si la recherche d'une chaîne a abouti oui ou non)	?NotFound NorFound(On/Off)	EditionRechercherTrouver()

1.6.4 Les structures de données

Je ne considérerai que trois structures de données, même si d'autres structures peuvent être envisagées.

•la structure "texte"

•la structure tableau ou array

•la structure de fichier séquentiel

La structure "texte"

Les programmes travailleront essentiellement sur un texte, qui devra être envisagé comme une structure de données particulière, dénommée ici *s*tructure–texte.

Les éléments de la structure

Suivant les situations, le programmeur pourra considérer le texte comme

•une suite de caractères,

•use suite de mots,

•une suite de paragraphes,

•une suite de lignes,

•une suite de pages, ...

Accès aux différents éléments de la structure.

On ne peut pas insérer cette structure de données dans une des structures connues. Le texte peut être perçu comme

•une structure de fichier séquentiel de caractères, mots, ...

•une structure avec accès direct aux informations. Les fonctionnalités offertes par le langage de programmation permettent des accès directs aux éléments du texte et la structure s'apparente aux structures de type tableaux, sans utilisation d'indices.

Dans cette structure de données, l'accès aux différents éléments de la structure se font essentiellement à l'aide des fonctions de déplacement d'un pointeur, plus connu sous le nom de curseur ou point d'insertion. Les déplacements (accès) sont repris dans le tableau ci–dessous.

Description de l'action	Consignes pour son exécution avec WP 6.0	Consignes pour son exécution avec Word 2.0
curseur_début	PosDocTop	StartOfDocument
curseur_fin	PosDocBottom	EndOfDocument

curseur_car_gauche	PosCharPrevious	CharLeft
curseur_car_droite	PosCharNext	CharRight
curseur_ligne_haut	PosLineUp	LineUp
curseur_ligne_bas	PosLineDown	Linedown
curseur_para_précédent	ParagraphUp	ParaUp
curseur_para_suivant	ParagraphDown	ParaDown
curseur_page_précédente	PosPagePrevious	PrevPage
curseur_page_suivante	PosPageNext	NextPage
curseur_mot_suivant	PosWordNext	WordRight
curseur_mot_précédent	PosWordPrevious	WordLeft
rechercher_avant(chaîne)	SearchString	Editfind
rechercher_arrière(chaîne)	SearchString	EditFind

Les manipulations des éléments de cette structure

Les manipulations reprises ci–dessous concernent l'ajout, la suppression des éléments de la structure. Les éléments à ajouter sont insérés à la position du curseur. Leur provenance peut être diverse: valeur renvoyée par une fonction, valeur d'une variable, texte provenant d'un fichier enregistré sur support externe, valeur directement encodée au clavier, ...

Description de l'action	Consignes pour son exécution avec WP 6.0	Consigne pour son exécution avec Word 2.0
Insérer		
insère_valeur(VAR ou fonction)	Type	Insert
insère_fichier(Fichier)	FileRetrieve	InsertFile
insère_clavier	{Clavier}	{demander var}
Supprimer		

efface_cara_gauche	DeleteCharPrevious	EditionEffacer
efface_cara_droite	DeleteCharNext	EditionEffacer
efface_paragraphe	BlockOn(ParagrapheMode!) BlockDelete	
efface_page	BlockOn(PageMode!)	
efface_ligne	DeleteToEndofLine	
efface_mot	DeleteWord	DeleteWord, DeleteBackWord
efface_texte	ClearDoc	FichierFermer
efface_sélection	BlockDelete	
Enregistrer/ouvrir		
Ouvrir_texte(nom du fichier)	FileOpen	FichierOuvrir
Enregistrer_texte(nom du fichier)	FileSave	FichierFermer

_On peut considérer que le programmeur peut adapter la perception qu'il a de cette structure en fonction du travail à faire faire. Il pourra y retrouver des structures plus classiques en restreignant les actions d'accès aux informations et de leur manipulation: fichier séquentiel, tableau, pile, ...

Les tableaux

Il s'agit ici du concept de tableau en tant que structures de données et non d'un texte présenté sous forme de tableau. On retrouvera, comme pour la majorité des langages de programmation, les tableaux à une ou deux dimensions (vecteurs et matrices) bien connues des programmeurs.

Description de l'action	Consignes pour son exécution avec WP 6.0	Consignes pour son exécution avec Word 2.0
Déclaration d'une variable de type tableau		Dim variable(n)

Les fichiers séquentiels

Il s'agit des fichiers séquentiels au sens algorithmique du terme. Les éléments du fichier sont des enregistrements (données composées). Les champs des différents enregistrements sont souvent identifiés par un nom de variable. L'utilisation des fichiers séquentiels est souvent le mailing, mais on peut imaginer d'autres applications comme vous le verrez dans les exercices proposés.

Description de l'action	Consignes pour son exécution avec WP 6.0	Consignes pour son exécution avec Word 2.0
Déclaration d'un enregistrement	MergefileType(TextData!) MergeCode(Fieldnames; noms des champs; ...)	FileCreateDataFile Fieldnames ...
Encodage des informations, des enregistrements d'un fichier	valeur Endfields valeur Endfields ... Valeur Endfields EndRecord ...	Encodage dans un tableau ligne = enregistrements colonne = champs.
Aller au début de fichier	PosDocTop	
Passer à l'enregistrement suivant	Enr Suivant	Suivant
Fin de fichier	un prédicat indiquant la fin d'un fichier, d'un texte n'existe pas, il faut le simuler	Eof pour les fichiers séquentiels, pas pour le texte en mémoire centrale.

1.6.5 Les structures de contrôle

Il s'agit ici des structures de contrôles du paradigme des langages de programmation procéduraux. Les explications et l'apprentissage de l'utilisation de ces concepts sortent du cadre de ce document. La description proposée est celle des langage de description des algorithmes.

_Séquence

Début

action–1

action–2

action–n

Fin

Pour les deux langages abordés, il n'y a pas de délimiteurs pour la séquence.

_Alternative

SI booléen **ALORS** actions_1 **SINON** actions_2

En dehors de l'alternative, on retrouve le "sélectif", (le célèbre CASE OF.)

Description de l'action	Consignes pour son exécution avec WP 6.0	Consignes pour son exécution avec Word 2.0
Alternative	If ... Then ... Else ... Endif	If ... Then ... Else ... Endif
Conditionnel	If ... Then ... Endif	If ... Then ... Endif
Sélectif	Case Caseof Caseof ... Default Endcase	Select Case Case ... Else Endselect

_Répétitives

Les trois structures de contrôle répétitives habituelles sont:

•**TANT QUE** booléen **FAIRE** actions

•**REPETER** actions **JUSQUE** Booléen

•POUR var = valeur_initiale VERS valeur_finale PAS incrément FAIRE actions

Description de l'action	Consignes pour son exécution avec WP 6.0	Consignes pour son exécution avec Word 2.0
Tant que	While ... Endwhile	While ... Wend
Répéter	Repeat ... Until	
Pour	For ... To ... Next	For ... To ... Next

Le description et la mise en oeuvre de ces structures répétitives sont bien connues et ne feront pas l'objet d'une description détaillée dans ce document. Toutefois, une autre structure répétitive "Itérer", plus générale peut être utilisée grâce à l'instruction "SORTIRSI booléen". Cette structure peut d'ailleurs être considérée comme le "père" des deux autres structures TANT QUE et REPETER.

_ITERER

ITERER

actions_1

SORTIRSI booléen

actions_2

FINITERER

Si actions_1 est vide, on retrouve le tant que, et si actions_2 est vide on retrouve le répéter jusque. Cette structure est rarement implémentée dans les langages et peut être simulée à l'aide d'un répétition "tant que" pseudo infinie. Cette simulation est possible avec WordPerfect. Avec Word uniquement avec l'utilisation du "Goto" volontairement rejeté dans le cadre d'une programmation structurée.

Cette structure de contrôle répétitive permet d'éviter certains pièges et inconvénients des deux structures Tant Que et Repeter Jusque.

exemple

Procédure de lecture de nombres uniquement positifs avec validation et message d'erreur destiné à l'encodeur.

Avec tant que	avec iterer
océdure saisie e valeur nt que valeur <0 rire "nombre positif SVP!" e valeur tantque n Procédure	océdure Saisie rer e valeur rtirsi valeur >=0 rire "nombre positif SVP!" itérer n Procédure

Description de l'action	Consignes pour son exécution avec WP 6.0	Consignes pour son exécution avec Word 2.0
Sortirsi booléen	If booléen Break Endif	En utilisant l'instruction Goto

1.6.6 Les procédures et fonctions

Le programmeur peut définir aussi bien des procédures que des fonctions. Ces dernières pouvant générer des valeurs qui pourront être, par exemple, directement insérées dans le texte. Dans le langage de description des algorithmes, l'appel d'une fonction ou d'une procédure se fera par son nom.

Description de l'action	Consignes pour son exécution avec WP 6.0	Consignes pour son exécution avec Word 2.0
Définition de procédure	PROCEDURE(arguments) ... RETURN	SUB ... ENDSUB
Définition de fonction	FUNCTION(arguments) ... RETURN(valeur)	FUNCTION ... ENDFUNCTION
Appel de procédure/fonction	nom de la procédure, CALL, NESTMACRO	nom de la procédure, CALL

1.7 Les actions de base liées au traitement de texte

L'ensemble de toutes les fonctionnalités du traitement de texte sont exécutables à partir d'un programme. Ces actions correspondent à l'appel de procédures et fonctions directement implémentées comme instructions du langage de programmation. Si on pense qu'un traitement de texte un peu sérieux offre plus d'un millier de fonctionnalités, vous pouvez facilement imaginer qu'il est impossible de les recenser toutes. Je propose plutôt une liste des instructions qui m'apparaissent suffisantes pour aborder la conception des premiers programmes écrits à l'aide d'un traitement de texte. Le langage proposé est formel et son utilisation doit être envisagée dans le cadre de la description des algorithmes de programmation. Elles viennent compléter le langage de description des algorithmes. Pour faciliter la lecture, ces instructions seront réparties dans les catégories décrites ci–après.

1.7.1 gestion du texte

Ce sont toutes les actions qui permettent la gestion de la structure "texte" telle qu'elle a été abordée précédemment. Les actions supplémentaires sont celles qui permettent la copie, le déplacement de parties de texte définies par le programmeur. Ces actions sont utilisées conjointement avec les actions de déplacement du curseur dans la structure "texte".

Beaucoup de ces actions ne s'appliquerons qu'à des morceaux de texte. Ces derniers sont habituellement définis de manière interactive par l'utilisateur, mais devront être définis de manière formelle par le programmeur, en utilisant des fonctions de positionnement du curseur.

Description de l'action	Consignes pour son exécution avec WP 6.0	Consignes pour son exécution avec WORD 2.0
Selection	BlockOn(CharMode!)	ExtendSelection
Couper_selection	Cut	EditCut
Copier_selection	Copy	Editcopy
Coller_selection	Paste	EditPaste
Copier_Paragraphe	BlockOn(ParagraphMode!) Copy	
Couper_paragraphe	BlockOn(ParagraphMode!) Cut	
Coller_paragraphe	BlockOn(ParagraphMode!) Paste	
Copier_page	BlockOn(PageMode!) Copy	

Coller_page	BlockOn(PageMode!) Paste	
Couper_page	BlockOn(PageMode!) Cut	
Remplacer_tout(chaîne1,chaîn e2)	ReplaceString	Editreplace
Remplacer_certains(chaîne1,c haîne2)	ReplaceConfirm	EditReplace

1.7.2 Mise en page du texte du point de vue caractère

Ce sont toutes les actions de mise en page qui gèrent les caractères: changement de police, changement de taille, application d'un style donné. Ces actions agissent souvent sur une partie sélectionnée de la structure texte en modifiant les valeurs de variables systèmes abordées dans les pages précédentes.

1.7.3 Mise en page du texte du point de vue espacement

Ce sont toutes les actions qui gèrent la répartition des blancs sur la page imprimée: marges, interlignes, interparagraphe, intermot, crénage, ... en modifiant les valeurs de variables systèmes.

1.8 Les domaines d'utilisation

La programmation avec les langages de programmation des traitements de texte offre un éventail d'applications très large qui peut être réparti en trois grandes familles.

1.8.1 Macro–instruction pour raccourcir les manipulations

Ce sont toutes les macro–instructions qui remplaceront les manipulations fréquemment utilisées lors de la manipulation du traitement de texte. Ce sont souvent des macro–instruction générées.

- insérer une tête de page, un pied de page,
- sélectionner un format de papier,
- faire afficher la liste des noms dans un répertoire,
- convertir en majuscules, minuscules, etc ...

1.8.2 Sans parcourir la structure "texte"

Ce sont les applications qui généreront du texte, des informations à insérer dans un texte existant. Elles ne gèrent pas la structure de données "texte". On pourrait mettre dans cette famille tous les énoncés de travail du type:

- calculs divers à partir d'informations entrées au clavier ou provenant d'un texte sur un support extérieur,
- génération de tables diverses: table de multiplication, de conjugaison, ...
- établissement de documents "type", avec des valeurs calculées: l'exemple classique étant la facture, le devis ou assimilé,
- etc ...

1.8.3 En parcourant la structure "texte"

Ce sont tous les programmes qui manipuleront la structure de données "texte", en utilisant toutes les actions de gestion et de navigation dans le structure. On pourrait mettre dans cette famille tous les énoncés de travail du type

- toilettage du texte: suppression de caractères inutiles, encombrants,
- comptages divers,
- génération de textes divers à partir de textes existants,
- etc ...

1.9 Le défi pédagogique sous–jacent

Si l'univers de la programmation a été progressivement abandonné au profit d'une utilisation raisonnée des logiciels, il pourrait bien faire un retour remarqué au travers des langages de programmation intégrés à certains logiciels, qu'il s'agissent de traitements de texte, tableurs ou gestionnaires de fichiers. Le défi à relever peut être résumé par les questions ci–dessous.

- Comment utiliser ce nouvel univers de programmation, quel que soit le traitement de texte, pour développer les facultés de raisonnement des élèves sur des problèmes concrets? On retrouve le concept de "problèmes à résoudre" cher à l'algorithmique et à la programmation.
- Quelles méthodologies mettre en oeuvre pour l'apprentissage?
- Les méthodologies développées dans le cadre de l'apprentissage de l'algorithmique et de la programmation sont–elles applicables?
- Quelles activités proposer aux élèves?

Sans répondre totalement à ces questions, j'ai pu relever quelques avantages et inconvénients de cet univers de programmation:

Avantages

- le générateur de programme,
- la possibilité de corriger le programme ainsi généré,
- la facilité relative de mise en oeuvre pour des petits programmes,

- l'écriture de programme concrets, qui on une utilité directe pour l'utilisateur–programmeur,
- approche en douceur de l'univers de programmation,
- apparition progressive de langages de programmation utilisant des objets,
- réconciliation entre les tenants de l'algorithmique "pure et dure", et les utilisateurs de logiciels.

Inconvénients

- Il faut bien connaître les possibilités du traitement de texte. L'apprentissage est plus long que dans l'approche habituelle d'un langage de programmation. Le nombre de primitives à manipuler est plus important.
- Il faut créer le "reflex" du programmeur. Il n'est pas acquis que les utilisateurs acceptent de passer à la programmation, même de macro–instructions générées.
- l faut deux apprentissages en parallèle ou en séquence: l'algorithmique et l'utilisation du traitement de texte.
- Illusion de facilité. La programmation reste toujours, même dans ce contexte une activité complexe.
- Il n'y a pas de langage standard pour les traitements de texte. Les syntaxes sont souvent lourdes, surtout pour les primitives du traitement de texte.
- Les langages ne sont pas des langages de développement au même titre que les autres langage de programmation.
- ette activité de programmation demande une compétence supérieure de la part de l'utilisateur de logiciels.
- Peu d'enseignants sont actuellement formés à ce type de programmation.
- Un apprentissage de l'algorithmique semble incontournable.
- Les applications doivent être trouvées, proposées et publiées. Il faut observer à ce sujet que le niveau de difficulté dépend du traitement de texte. S'il est possible de dégager des algorithmes indépendant du logiciel, il en existent d'autres qui sont liés au traitement de texte. Un travail aisé avec un logiciel peut devenir très complexe avec un autre et vice et versa.

Les pages suivantes contiennent des propositions d'activités illustrant les différents concepts abordés. Chaque énoncé est accompagné de la description de l'algorithme correspondant ainsi que de sa traduction dans les langages de programmation de WordPerfect 6.0 DOS, et Word 2.0 Windows. Le temps m'a manqué pour entrer dans le nouveau langage de programmation de Word 6.0. La traduction des algorithmes dans ce langage ou un autre est la bienvenue.

Vous pouvez faire parvenir vos suggestions de programmes, mais aussi d'énoncés que je me ferai un plaisir de partager avec tous les enseignants, à l'adresse suivante: