Compétences de base et programmation avec un traitement de texte

Sass Ferenc
Service informatique
Fédération de l'Enseignement Secondaire Catholique de Belgique
Mande Saint Etienne, 923
6688 Longchamps
Belgique

1 L'hypothèse de travail

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 intégrés actuellement aux traitements de texte et dès lors de concevoir des algorithmes et programmes traduisibles pour n'importe quel traitement de texte programmable et de déterminer les compétences de base d'un utilisateur de traitement de texte pour l'amener à programmer.

2 Le concept de macro–instruction

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. Le concept de macro–instruction est comparable aux concepts de procédures et fonctions en programmation. Je me permets d'affirmer qu'une macro–instruction est en réalité un programme.

Toutefois, dans l'univers des traitements de texte, 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",

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. Le traitement de texte "enregistre" la séquence d'ordres donnée par l'utilisateur, à la demande de ce dernier.

•les macro–instructions "programme".

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.

Les macro–instruction–programmes sont souvent des programmes dont les instructions ne sont pas facilement générées par le traitement de texte.

3 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. (par exemple, la version 6.0 de Word de Microsoft est livrée avec le langage VisualBasic)

Je distinguerai différentes familles d'actions:

•celles liées aux actions de base du paradigme de programmation procédurale,
_Déclaration de variables globales, locales
_L'affectation
_L'entrée
_Sortie
_Les structures de contrôle
_Séquence
_Alternative
_Répétitives
TANT QUE booléen **FAIRE** actions
REPETER actions **JUSQUE** Booléen
POUR var = valeur_initiale **VERS** valeur_finale **PAS** incrément **FAIRE** actions
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 structure de contrôle répétitive permet d'éviter certains pièges et inconvénients des deux structures Tant Que et Repeter Jusque.

•celles liées aux informations et structures de données,

_les opérateurs et fonctions associés aux types de données: caractère, numérique, logique.
_les variables du système
_Les structures de données
_la structure "texte"
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, ...

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 manipulations des éléments de cette structure

Les manipulations 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, ...

_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, ...

_la structure tableau ou array

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.

_la structure de fichier séquentiel

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.

•celles liées à l'appel des 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.

•celles liées aux actions de base du 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.

_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 déplacements du curseur dans la structure "texte".

Beaucoup de ces actions ne s'appliqueront 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.

_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.

_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.

4 Les domaines d'utilisation

•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–instructions générées.

•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".

•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.

5 Le défi pédagogique sous–jacent: les compétences de base

Si l'univers de la programmation a été progressivement abandonné au profit d'une utilisation raisonnée des logiciels, il pourrait bien faire un retour 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.

•quelles compétences de la part des élèves?

•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?

Les réponses peuvent être suggérées en citant quelques avantages et inconvénients de ces nouveaux langages.

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 programmes 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.

•Il 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.

•Cette 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.

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.

6 Conclusions provisoires

La place manque pour donner une liste d'exercices résolus illustrant tous ces concepts. Un document plus complet peut être obtenu auprès de Sass F.

Ce document comprend

•une partie théorique reprenant l'ensemble des concepts cités dans ce texte.

•la mise en oeuvre de ces concepts avec WordPerfect 6.0 et Word 2.0.

•une liste d'exercices résolus, avec description des algorithmes et leurs traductions pour WordPerfect 6.0 et Word 2.0.

Les exemples choisis montrent que l'hypothèse se vérifie dans la majorité des situations. Un ensemble d'actions de base commun aux langages de programmation des traitements de texte peut être défini, tant pour la partie algorithmique que pour les fonctionnalités de base des traitements de texte. 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 différences apparaissent lors de l'utilisation de focntionnalités avancées tels que la recherche de codes de mise en page, la fusion de documents, la gestion des tableaux, ...

Par contre, l'utilisation du traitement de texte dans le cadre de la résolution de problèmes par l'intermédiaire de l'algorithmique et la programmation doit encore être évaluée.

Dans un milieu de travail, les tentatives d'apprentissage de la création de macro–instructions générées par des secrétaires montrent que:

•elles sont interessées par le concept

•elles n'ont pas le "reflex" d'utiliser le concept. (il faudrait un accompagnement plus intensif durant le travail au quotidien)

•elles acceptent plus facilement d'utiliser des macro–instructions toute prêtes, même si dans un premier temps elles doivent vérifier si "ça marche bien"

•elles ont plus le reflex d'utiliser les balises de mise en page (feuille de styles) que les macro–instructions.

Je n'ai pas fait l'essai de leurs apprendre à programmer réellement, sauf l'utilisation des alternatives dans le cas des fusions de documents.

Il reste à poursuivre une étude auprès des élèves, en milieu scolaire, dont l'objet serait d'évaluer les possibilités de la réintroduction de l'activité algorithmique au travers des langages de programmation intégrés au traitement de texte, mais aussi à d'autres logiciels.