

T9 – Un système de saisie prédictive

Le projet

Le but du projet est de réaliser un système de saisie prédictive, de fonctionnement similaire à celui des dictionnaires sur un gsm. Ce système est associé au nom *T9* (Texte sur 9 touches) tel que popularisé par certains grands constructeurs de gsm.

Point de vue de l'utilisateur

Le principe de fonctionnement est bien connu¹ : Afin d'éviter un "MultiTap", le système utilise un dictionnaire pour proposer à l'utilisateur une liste de mots existants et correspondants à la combinaison de touches numériques entrées. Par exemple, la suite 2643 pourrait donner le mot *amie*, mais aussi le mot *ange*. Deux mots obtenus par la même suite numérique seront dits être des *T9-synonymes*.

Avec le dictionnaire qui vous est fourni, la combinaison de touches possédant le plus de *T9-synonymes* est 72737 : Il y a 10 mots correspondants (*papes, parer, pares, raper, rapes, rares, raser, rases, saper, sapes*).

Travail à réaliser

Au minimum, le programme doit permettre à l'utilisateur de saisir au clavier des touches numériques, et lui proposer tous les mots du dictionnaire qui correspondent. Le choix de l'interface utilisateur est laissé libre et n'interviendra pas dans l'évaluation finale du travail.

Point de vue du programmeur

• Structures de données

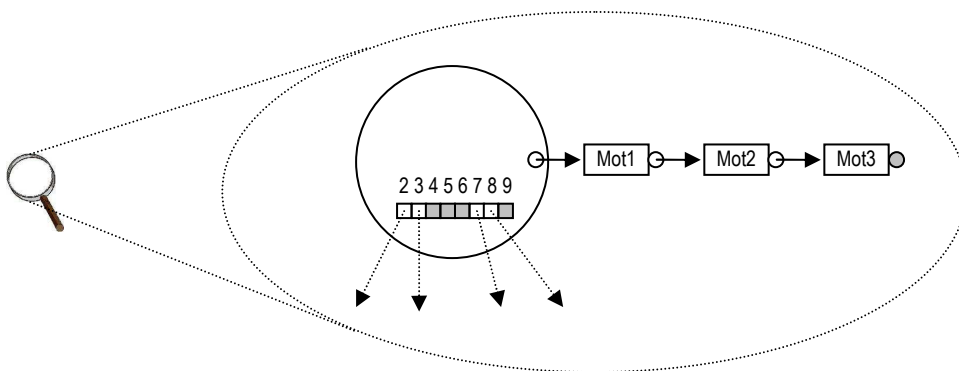
L'implémentation d'un dictionnaire donne lieu à des choix subtils. Dans notre cas, nous ne pouvons pas nous contenter d'un *set* (ensemble) de mots car nous devons, comme stipulé ci-dessus, pouvoir trouver plusieurs mots ayant la même écriture *T9* (la même suite de touches numériques). Par ailleurs, une *table* associant une suite numérique à sa liste de mots ne convient pas non plus car elle ne permet pas, avec suffisamment d'efficacité, une saisie prédictive par laquelle des mots sont proposés dont la suite numérique entrée forme un préfixe (*auto-complétion*). Mon gsm, par exemple, propose le mot *gentils* (7 lettres) quand je n'ai tapé que les 5 caractères de la suite 43684.

La représentation du dictionnaire se fera donc à l'aide d'un arbre (n-aire). Le parcours dans l'arbre est déterminé par le préfixe du mot. Voici, par exemple, un extrait de l'arbre, annoté, de façon incomplète, par les mots du dictionnaire :

¹ http://fr.wikipedia.org/wiki/Text_on_9_keys#T9

Par ailleurs, chaque nœud définit aussi la liste de mots du dictionnaire associés à la suite numérique. Nous voyons donc qu'en suivant les liens marqués par les chiffres 2, 6, 4 et 3, nous arrivons au nœud annoté par les mots *amie* et *ange*.

De plus, donc, chaque nœud doit contenir la liste des mots associés. Cette liste sera représentée de façon dynamique par une liste chaînée.



Ceci nous amène aux déclarations *Pascal* suivantes :

```
Type
  TElem = String[25];

  TListe = ^Cell;

  Cell = Record
    info : TElem;
    svt   : Tliste;
  End;

  TArbre = ^Noeud;

  Tableau = Array['2'..'9'] Of TArbre;

  Noeud = Record
    fils : Tableau;
    mots : TListe;
  End;
```

Vous veillerez, bien sûr, à modulariser votre programme. De fait, ces déclarations seront réparties dans différentes unités.

Si vous décidez d'utiliser un autre langage de programmation, vos déclarations/définitions de type seront équivalentes.

- **Dictionnaire**

Un fichier texte (*dico.txt*) vous est fourni, qui contient environ 110.000 mots de la langue française. Ces mots sont donnés, un mot par ligne, en minuscules et sans diacritique (accents...).

Votre programme doit, au lancement, lire ce fichier et charger son contenu en mémoire dans la structure d'arbre. La lecture du fichier se fera sur base du modèle suivant :

```
Var
  fich : TextFile;
  mot   : String;
// ...
Begin
  If FileExists(NomFichier) Then Begin
    AssignFile(fich, NomFichier);
    Reset(fich);
    While Not Eof(fich) Do Begin
      ReadLn(fich, mot);
      Traiter(mot); // A vous de jouer...
    End;
  End
  Else
    // ... Le fichier n'est pas disponible
  End;
```

Bien sûr, vous adapterez le code ci-dessus pour le langage de programmation choisi.

• Construction de l'arbre

Afin, toujours, d'assurer une bonne modularité par l'abstraction de la représentation des données, vous écrirez deux routines qui masqueront la représentation de l'arbre :

```
Procedure Init(Var t : TArbre);  
// Initialise t à un arbre vide  
  
Procedure AjouterMot(Const mot : String; Var t : TArbre);  
// Ajoute le mot dans l'arbre
```

A nouveau, les entêtes de fonctions seront adaptés (méthodes...) au choix du langage.

Consignes

Le travail est individuel même si vous êtes encouragés à partager vos idées et vos analyses. Partager des idées n'est pas partager du code!

L'état d'avancement du projet doit être validé régulièrement par le professeur.

Vous remettrez le **listing** de votre programme accompagné d'un bref **rapport** et le code source de la totalité des fichiers du projet sous forme électronique (soit sur le serveur de WSP à l'adresse **\\Profs-Public\AS\PRDV\Travaux**, soit par email, à l'adresse **asilovy@ulb.ac.be**).

Le rapport de quelques pages devra contenir une description de la structure des unités du programme et des routines principales.

La date précise de remise du travail vous sera communiquée ultérieurement. Elle se situe vers la mi-janvier soit, une à deux semaines avant l'examen. Les étudiants qui n'auront pas participé aux séances de validation de l'avancement du travail ne pourront pas remettre leur projet ni présenter l'examen!

Pour des raisons de sécurité, ayez, à tout moment, au moins trois copies de votre travail (par exemple une sur le serveur de WSP et deux sur support extérieur : ordinateur personnel, clé USB etc...). Ne vous contentez pas de votre clé USB, ce n'est pas un support de travail adéquat (manque de fiabilité, perte ou oubli ...) !

Evaluation

L'évaluation se fera sur base du travail remis et d'une défense qui aura lieu durant la période des examens de janvier. Elle tiendra compte notamment des critères suivants :

- le respect des règles de bonnes pratiques de la programmation structurée.
- la qualité de votre implémentation
- la qualité de votre présentation et de vos réponses aux questions lors de la défense.

Bonus

Des points de bonus seront attribués pour certaines fonctionnalités optionnelles ajoutées au travail. Par exemple :

- Auto-complétion par des mots fils dans l'arbre
- Saisie de textes complets et pas seulement de mots (ponctuation...)
- Gestion d'un dictionnaire personnel...