Cycle préparatoire 1ère année

Algorithmique - Projet de fin de semestre

Florent Devin, Peio Loubière

Matière : **Informatique**	*Date* : **Janvier 2018**
	Durée : **4 semaines**
	Nombre de pages : **8**

Table des matières

1 Contexte général

Le projet est un travail à réaliser en groupes de deux personnes (du même groupe de TD), éventuellement trois personnes (si le nombre d'élèves d'un groupe est impair, vous validerez cela avec votre professeur). À la fin du projet, vous devrez rendre une archive (au format `tgz` [1]) contenant :

— un rapport [2] sur le projet (au format PDF) ;
— vos codes sources (et uniquement les codes sources) ;
— un fichier README.md (contenant une rapide aide sur l'utilisation de votre projet).

Cette archive doit s'appeler `<nom1>_<nom2>.tgz`, où `<nom1>` et `<nom2>` sont à remplacer par vos noms respectifs (dans l'ordre alphabétique).

Lisez entièrement le sujet avant de commencer, réfléchissez, analysez avant tout. Les différentes parties du sujet n'étant pas toutes indépendantes les une des autres, il se peut qu'une partie influence vos choix sur une autre partie.

Toutes les parties sont à réaliser. Veuillez respecter à 100% les consignes données.

Par exemple si l'on vous dit :

Écrivez la fonction `calculerValeurCellule` qui prend en arguments :

— `grille` : une grille ;
— `x` : une abscisse ;
— `y` : une ordonnée.

1. Ce format est généré par la commande `tar`
2. Vous trouverez en annexe des conseils pour la rédaction de ce type de rapport.

et qui retourne la nouvelle valeur de la case `(x, y)` de `grille` calculée en fonction des règles régissant les cellules.

Cela signifie que vous devez faire une fonction qui aura le prototype suivant :

```
FUNCTION calculerValeurCellule(grille: TypeGrille; x,y: Integer): Integer;
```

Vous n'avez pas d'autre possibilité. Le non respect de ces consignes pourra entraîner des pénalités. Si l'on vous donne un prototype de fonction, il vous faut le respecter.

Par ailleurs, vous devez fournir un code correct et lisible. Cela signifie que vous devez documenter votre code, utiliser des noms de variables explicites, et ne faire ***aucune*** fonction/procédure qui dépasse 25 lignes (y compris le programme principal).

Enfin, veuillez noter que le plagiat est interdit. Il existe des outils permettant de détecter automatiquement le plagiat (pour le code, comme pour les documents écrits). Toute forme de plagiat sera sanctionné sévèrement, et pourra le cas échéant faire l'objet d'un conseil de discipline. Le plagiat peut être assimilé à une contrefaçon, punie par la loi.

Quelques conseils : Pour simplifier vos traitements, nous vous recommandons de commencer les tableaux à l'indice 0. Pour vous permettre d'effacer la console, vous pourriez avoir besoin de la fonction `ClrScr;`. Cette fonction fait partie de la librairie `Crt`. Pour utiliser cette librairie, il faut mettre `USES Crt;` juste après la ligne qui commence par `PROGRAM`. Cette librairie, vous offre aussi la possibilité d'utiliser la fonction `Delay(millisecond:Integer)`. Cependant, l'utilisation de cette librairie empêche l'utilisation de la séquence `Ctrl-C`...

2 Sujet

Le sujet de ce projet est inspiré du blog : "La science étonnante", et plus particulièrement sa page "la machine à inventer des mots". L'objectif est de réaliser un algorithme capable d'inventer des mots, et des phrases simples. Comme évoqué, sur le site web, ainsi que dans la vidéo [3], il existe plusieurs façons de générer aléatoirement des mots.

Pour rappel, la langue française est composée de 42 lettres, les 26 lettres de l'alphabet latin : 13 voyelles accentuées, le graphème ç, ainsi que les deux ligatures (æ, œ), et aussi le symbole `-`. Afin de pouvoir utiliser un encodage correct des chaînes de caractères [4] dans pascal, il faut que vous ajoutiez les directives de compilation, ainsi que la librairie comme indiqué sur le listing 1, et utiliser le type `WideString` à la place de `String`.

```
.../...
PROGRAM Projet;
{$mode objfpc}{$H+}
{$codepage UTF8}
{$I-}

USES cwstring, crt;
.../...
```

Listing 1: Directives et unités à inclure

Enfin, vous veillerez à ce que votre programme traite la ligne de commande [5], pour lancer la méthode et pour charger un dictionnaire. Pour ce faire, vous utiliserez `ParamCount` et `ParamStr`. Utilisé

3. Je vous conseille vivement de regarder cette vidéo. N'hésitez pas à en regarder d'autres...
4. Y compris pour les caractères spéciaux
5. À la manière d'une commande UNIX

sans paramètre, votre programme devra afficher l'aide [6]. La syntaxe générale du programme devra respecter la syntaxe représentée par le listing 2.

```
NAME
        projet - la machine à inventer des mots

SYNOPSIS
       projet [OPTION]... FILE

DESCRIPTION
       Génère des mots ou des phrases à partir du dictionnaire FILE

       -a     utilise la méthode aléatoire pour générer les mots

       -d     utilise la méthode des digrammes pour générer les mots

       -t     utilise la méthode des trigrammes pour générer les mots

       -p     génère une phrase (en utilisant la méthode des
        ↪  trigrammes)

       -n NB  génère NB mots (par défaut génère 100 mots)

       -s NB  affiche uniquement des mots de NB caractères

       -h     affiche cette aide et quitte

AUTHORS
       Écrit par F. Devin, P. Loubière.
```

Listing 2: Utilisation du programme (présenté comme `man`)

2.1 Aléatoire

Comme il est évoqué dans la vidéo, nous pouvons générer un mot totalement aléatoire (en tirant aléatoirement des lettres). Une telle méthode génère par exemple les mots : `xyêùzûwùr, jdzeqjhgh, œlwqècüts`. Ce qui ne produit pas des mots très lisibles…

Créez une fonction qui permet de créer un mot de manière totalement aléatoire. Cette fonction doit avoir le prototype suivant :

```
FUNCTION creeMotAleatoire (taille : Integer) : WideString;
```

Listing 3: Prototype de la fonction `creeMotAleatoire`

Vous pouvez définir si besoin la constante suivante :

```
CONST   alphabet : WideString  = 'abcdefghijklmnopqrstuvwxyzàâéèêëîïôùûüÿæœç-';
```

Listing 4: Constante pour les lettres de l'alphabet français

6. À la manière d'un `ls -help`

2.2 Enchaînement de digrammes

Afin de générer des mots plus plausibles, à la lecture, nous orientons le choix de chaque lettre selon des règles probabilistes d'enchaînement de deux lettres. La génération de mot par digrammes est plus complexe que la précédente. Pour pouvoir générer des mots à partir de digrammes, il faut dans un premier temps disposer d'un dictionnaire. Puis pour chaque mot de ce dictionnaire, pour chaque lettre, regarder quelle est la lettre suivante, et mettre à jour la table de probabilité. Il peut être utile de mémoriser quelles lettres commencent un mot, et lesquelles terminent un mot.

Par exemple, avec le mot `informatique`, on extrait la table de probabilités suivante :

	a	e	f	i	m	n	o	q	r	t	u	*fin*
début	_	_	_	*1*	_	_	_	_	_	_	_	_
a	_	_	_	_	_	_	_	_	_	1	_	_
e	_	_	_	_	_	_	_	_	_	_	_	*1*
f	_	_	_	_	_	_	1	_	_	_	_	_
i	_	_	_	_	_	0.5	_	0.5	_	_	_	_
m	1	_	_	_	_	_	_	_	_	_	_	_
n	_	_	1	_	_	_	_	_	_	_	_	_
o	_	_	_	_	_	_	_	_	1	_	_	_
q	_	_	_	_	_	_	_	_	_	_	1	_
r	_	_	_	_	1	_	_	_	_	_	_	_
t	_	_	_	1	_	_	_	_	_	_	_	_
u	_	1	_	_	_	_	_	_	_	_	_	_

TABLE 1 – Table de probabilités du mot `informatique`

Une fois la table complète de probabilités calculée, il suffit de partir d'une lettre, et d'enchaîner les lettres en fonction de la table de probabilité. En suivant la table 1, et en partant de la lettre `t`, on génère le mot `tique`. Bien évidemment, cela n'a aucun sens de faire une table de probabilités sur un seul mot. Plus on prend de mots en compte, plus la table de probabilités reflète correctement l'enchaînement des lettres dans la langue. En utilisant entièrement [7] cette table de probabilités, on ne peut générer que deux mots : `informatique`, `ique`.

Créez une fonction qui permet de générer un mot en utilisant la méthode des digrammes. Il faut donc trouver une méthode afin de créer des mots différents en partant d'une même lettre. Il ne suffit pas de prendre la probabilité la plus élevée, pour trouver la lettre suivante. Il faut trouver un moyen astucieux pour ne pas prendre toujours la probabilité la plus élevée, tout en évitant les probabilités trop faibles. Par ailleurs, certaines lettres sont très peu probables comme première lettre, c'est le cas de `ô` qui ne commence que le mot ôter (et ses formes conjuguées), ou le `ç` qui commence uniquement les mots `ça`, `çà` et `çruti`.

2.3 Enchaînement de trigrammes

La méthode des trigrammes suit la même procédure que la méthode des digrammes, mais au lieu de ne considérer qu'une seule lettre pour regarder la suivante, on considère les deux précédentes. Cela crée une table de probabilités plus conséquente, mais beaucoup plus fine pour la création de mot. En utilisant cette méthode, il faut également réfléchir à la façon de débuter un mot (qui ne peut pas être totalement aléatoire).

2.4 Génération de phrases

L'objectif de cette partie est de créer des phrases simples, en utilisant des mots générés aléatoirement. La langue française étant très complexe, la génération sera limitée au schéma :

7. c'est-à-dire en utilisant les informations de *début* et de *fin*

— Sujet + verbe.
— Sujet + verbe + adjectif.
— Sujet + verbe + adverbe + adjectif.

Dans ce projet, nous limiterons les sujets à la forme : nom propre ou article + nom commun. Si l'article est pluriel [8], il faut accorder le nom commun en ajoutant un `s`. Les verbes se conjuguent [9] [10]. Les adverbes ne s'accordent pas. Les adjectifs s'accordent.

Pour ce faire, vous disposez de 6 dictionnaires : un pour les adjectifs, un pour les adverbes, un pour les verbes (à l'infinitif), un pour les articles (singulier, et pluriel), un pour les noms communs, et enfin un pour les noms propres.

Les phrases que vous générez doivent être "grammaticalement" correctes. De plus, vous devez pouvoir générer les trois formes possibles. Pour chaque catégorie, vous devez générer [11] des mots adaptés, et les assembler correctement afin de faire une phrase.

2.5 Bonus

Pour une meilleure note, et seulement si les parties précédentes ont été traitées, vous pouvez réaliser des bonus, parmi ceux proposés (ou d'autres si vous voulez), vous pouvez :

— Étendre la génération de phrases à d'autres formes.
— Étendre la conjugaison à d'autres forme/groupe.
— Générer des mots avec une autre méthode.
— …

3 Notation

Vous trouverez dans cette partie, une indication sur la façon dont vous serez noté. Vous trouverez dans la table 2 un barème, donné à titre indicatif. Les points sont attribués si la partie évaluée est terminée et correctement traitée (à l'appréciation de vos professeurs). La qualité du code sera évaluée par vos professeurs. De cette qualité découlera un pourcentage qui sera appliqué à l'ensemble des points obtenus. Un code qui ne compile pas se voit attribuer la note de 0.

3.1 Qualité du code

La qualité du code s'évalue selon plusieurs critères :

— Les fonctions / procédures / programme ne doivent pas faire plus de 25 lignes. Le nombre de lignes est calculé à partir du premier `BEGIN`, jusqu'au dernier `END` de la structure concernée. Par exemple, dans le listing 5, la procédure `hello` compte 6 lignes.
— Il n'y a aucune variable globale. Les constantes sont bien évidemment autorisées.
— La compilation se déroule sans `Warning`.
— La compilation se déroule sans `Note`.
— Les variables sont correctement nommées.
— Les commentaires de fonction / procédure / programme existent, et sont correctement remplis.
— Le code est lisible.
— …

Chaque "infraction" est répercutée sur l'indicateur qualité de votre code.

8. On considérera les articles pluriels, tous les articles qui se terminent pas un `s` ou un `x`
9. La conjugaison des verbes du premier et du second groupe est facile
10. Dans un premier temps, on se limitera à la conjugaison au présent
11. en utilisant la méthode de trigrammes

```pascal
(*
 ------------------------------------------------------------------------
 -- PROCEDURE         : hello
 -- Auteur            : Florent Devin <fd@eisti.eu>
 -- Date de creation  : Thu Nov  1 16:43:41 2018
 --
 -- But               : Dire bonjour à quelqu'un
 -- Remarques         : Aucune
 -- Pré conditions    : Préconditions
 -- Post conditions   : Dire bonjour à quelqu'un
 ------------------------------------------------------------------------*)
PROCEDURE hello;
VAR
   nom : String;
BEGIN
   REPEAT
      WRITELN('Entrez votre nom : ');
      READLN(nom);
   UNTIL (nom <> '');
   WRITELN('Hello ', nom);
END; { Hello : 6 lignes }
```

Listing 5: Exemple de code

3.2 Barème indicatif

Objet	Partie évaluée	Points
Ligne de commande	Sans argument	0.5
	Gestion des commutateurs `-a`, `-d`, `-t`, `-h`	1
	Gestion des options `-s`, `-n`	1
	Utilisation du dictionnaire en paramètre	0.5
Génération aléatoire		1
Méthode des digrammes	Lecture du dictionnaire	1
	Construction de la table des fréquences	0.5
	Génération des mots	1
Méthode des trigrammes	Lecture du dictionnaire	0.5
	Construction de la table des fréquences	0.5
	Génération des premières lettres	1
	Génération des mots	0.5
Génération de phrases	Lecture des dictionnaires	2
	Génération des mots	1
	Accord du sujet	1
	Conjugaison du verbe	1
	Accord de l'adjectif	1
	Génération des phrases	2
Rapport	Respect des règles	3
Bonus		0 – 3
Qualité du code		0 - 100%
Total		20

TABLE 2 – Grille de notation

A Rédaction du rapport

Vous trouverez ici quelques conseils pour la rédaction d'un rapport de projet informatique. La rédaction du rapport est trop souvent reportée à la fin du projet, c'est une erreur. Vous devez essayer de le rédiger au fur et à mesure de la réalisation du projet.

A.1 Objectifs d'un rapport de projet

Le rapport de projet sert avant tout à informer les professeurs du travail accompli, il sert donc principalement au correcteur. Il faut donc y travailler dès le début afin que le correcteur puisse y trouver les informations utiles afin d'évaluer correctement le travail.

Un rapport de projet a aussi pour objectif d'apprendre à rédiger un rapport technique. À ce titre, vous devez supposer que le lecteur ne connaît pas le projet. Le rapport doit donc contenir tout ce qu'il faut pour permettre une bonne compréhension du travail accompli. Le correcteur attend de fait un rapport qui présente le projet (sujet, mise en œuvre, difficultés, ...) ainsi qu'une documentation (guide d'utilisation, environnement d'exécution, environnement de développement, ...)

A.2 Structure du rapport

Un rapport, comme tout texte scientifique/technique doit contenir (dans l'ordre) :

— un titre, le nom des auteurs, la date (à minima) ; on pourra éventuellement créer une page de garde comportant ces éléments ;
— une table des matières ;
— une introduction ;
— un développement ;
— une conclusion ;
— une bibliographie ;
— éventuellement des annexes.

Vous trouverez, ci-dessous, pour chaque élément une description.

A.2.1 Table des matières

Une table des matières se place au début d'un document, et contient les références aux chapitres, sections, ... accompagnées d'un numéro de page.

A.2.2 Introduction

Le but de l'introduction est d'avoir une vue générale de ce qu'il fallait faire, et du travail réalisé. L'introduction sert à donner envie au lecteur de lire le rapport. Une mauvaise introduction n'incitera pas le lecteur à poursuivre sa lecture. Vous devez situer brièvement le sujet, et annoncer la problématique étudiée. Si le rapport est long, vous devez aussi présenter les parties suivantes, en y indiquant ce que le lecteur trouvera dans ces différentes parties.

A.2.3 Développement

C'est la partie la plus conséquente du rapport. Il est judicieux de commencer cette partie en même temps que votre projet. Vous noterez au fur et à mesure vos idées, et si éventuellement vous remettez en cause une idée, vous pourrez indiquer pourquoi, et ainsi mieux justifier vos choix.

Vous devez y présenter votre travail (les points essentiels). Les codes sources ne suffisent pas, au contraire dans le rapport vous devez présenter l'analyse du problème, sa réalisation, et éventuellement, les tests que vous avez effectués. Il est intéressant de décrire votre démarche, comment vous avez segmenté le problème en sous problèmes (et pourquoi) ; détailler les difficultés (algorithmiques) rencontrées ; présenter les structures de données utilisées (si cela est nécessaire).

Expliquer un algorithme c'est décrire les étapes qui ont mené à celui-ci, et aussi le critiquer, le détailler (comportement global). Décrire c'est :

— utiliser des exemples illustratifs ;
— créer des schémas pour aider à la compréhension ;
— justifier la cohérence (pré-conditions, post-conditions, invariants, ...).

Expliquer un algorithme n'est pas donner des détails techniques (pas de code source, ni des détails d'implémentation).

A.2.4 Conclusion

La conclusion d'un rapport est essentielle, elle reprend les éléments importants de l'introduction, et répond aux questions qui y ont été posées. Le problème a-t-il été traité dans son intégralité, aurait-on pu faire autrement, que reste-t-il à faire. Il faut aussi prendre du recul par rapport à ce que le projet vous a apporté (connaissances, expérience, ...).

A.2.5 Bibliographie

Une bibliographie contient tous les ouvrages que vous avez lus pour réaliser votre projet, et que vous *citez* dans votre rapport. Vous ne devez pas mettre des références dans une bibliographie, si vous n'y faites jamais mention dans votre rapport (utilisez la commande `\cite` de LaTeX, et n'utilisez jamais la commande `\nocite`).

Une bibliographie devrait contenir majoritairement des références à des ouvrages écrits et très peu de références web. En effet, une référence web est une référence qui a été publiée sans relecture, et la confiance en celle-ci est restreinte.

A.2.6 Annexe

En annexe, vous pouvez mettre tout ce qui vous semble utile pour comprendre le travail, mais qui n'est pas indispensable à la compréhension du rapport, lors de la lecture de celui-ci.

A.3 Conseils pour une meilleure rédaction

— Un titre ne comporte pas d'article, sauf cas très particulier. Par exemple, on dit : "Conclusion" pas "La conclusion".
— L'emploi des majuscules doit respecter les règles d'écriture et d'orthographe en vigueur dans la langue de votre rapport.
— Pour être compris, utilisez des phrases courtes.
— Pour être lu, rédigez sans fautes.
— Évitez les phrases négatives.
— Visez la concision (vous ne faites pas un roman, préférez le style direct et les formulations neutres).
— Le plagiat est interdit [12].
— Si vous recopiez du texte (ou le traduisez), mettez le texte entre guillemets et citez le texte et vos sources.
— Ne mettez pas de code dans le rapport (ou alors en annexe).
— Mettez des titres à *toutes* vos figures, tables, images, ...
— Utilisez une police de taille minimum 10 (la taille standard étant 12).

12. Le plagiat est une faute d'ordre moral, civil ou commercial, qui peut être sanctionnée au pénal (d'après wikipédia)