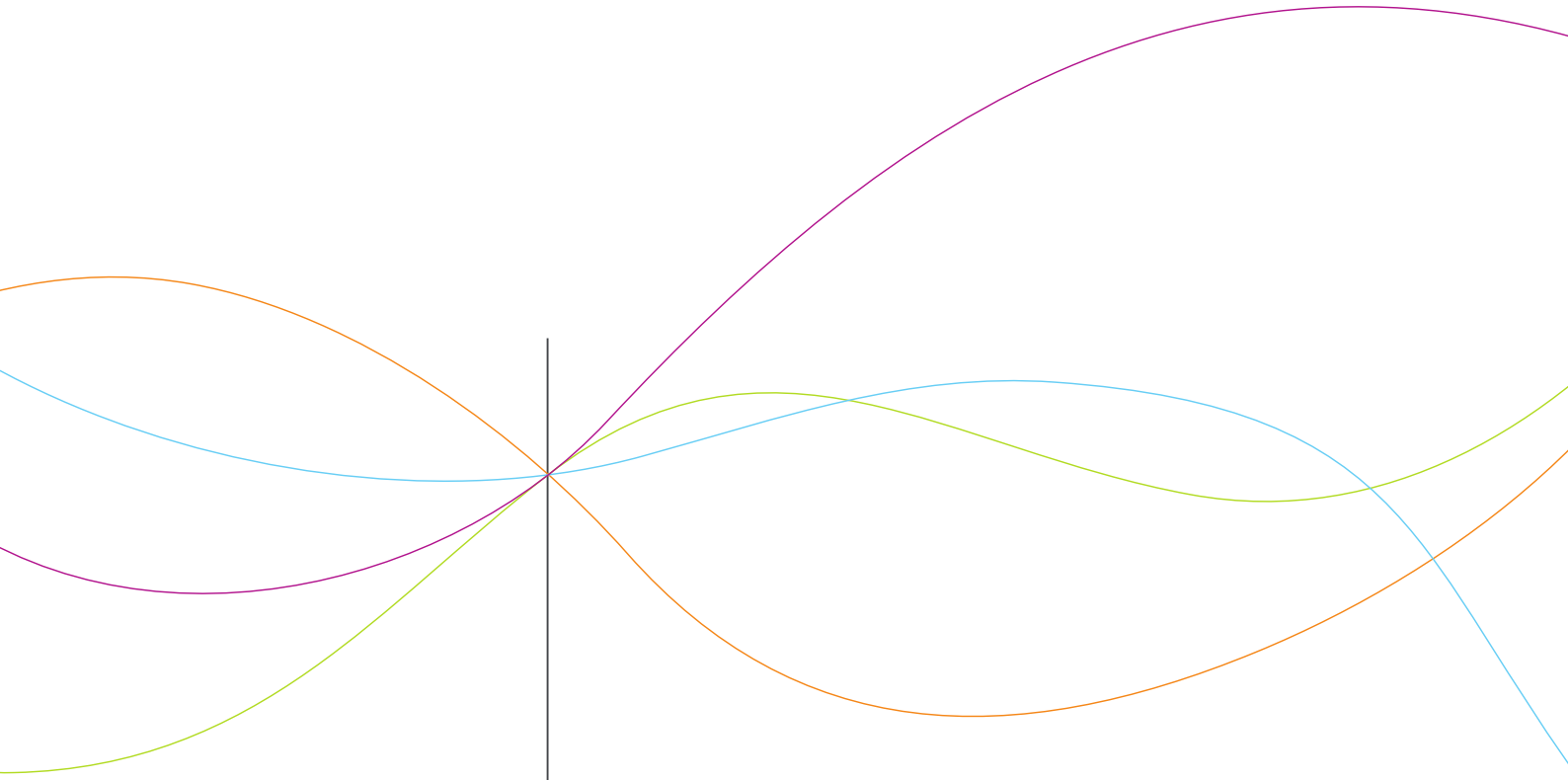
INSA
INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
ROUEN NORMANDIE



Alexis Imbert
Ruth Lioté
Amina Sakouti
Léo Zedek

Correcteur Orthographique
ITI ALGO 2021 2022

Table des matières

1 Analyse

1.1 Présentation des TAD

1.1.1 TAD FichierTexte

Nom: FichierTexte

Utilise: **Chaine de caracteres**, Mode, **Caractere**, **Booleen**

Opérations:

fichierTexte:	**Chaine de caracteres** → FichierTexte
ouvrir:	FichierTexte × Mode ↛ FichierTexte
fermer:	FichierTexte ↛ FichierTexte
estOuvert:	FichierTexte → **Booleen**
Mode:	FichierTexte ↛ Mode
finFichier:	FichierTexte ↛ **Booleen**
ecrireChaine:	FichierTexte × **Chaine de caracteres** ↛ FichierTexte
lireChaine:	FichierTexte ↛ FichierTexte × **Chaine de caracteres**
ecrireCaractere:	FichierTexte × **Caractere** ↛ FichierTexte
lireCaractere:	FichierTexte ↛ FichierTexte × **Caractere**

Sémantiques:

FichierTexte:	creation d'un fichier texte à partir d'un fichier identifié par son nom
ouvrir:	ouvre un fichier texte en lecteur ou ecriture. Si le mode est ecriture et que le fichier existe, alors ce dernier est écrasé
fermer:	ferme un fichier texte
lireCaractere:	lit un caractère à partir de la position courante du fichier
lireChaine:	lit une chaine (jusqu'à un retour à la ligne ou la fin de fichier) à partir de la position courante du fichier
ecrireCaractere:	écrit un caractère à partir de la position courante du fichier
ecrireChaine:	écrit une chaine suivie d'un retour à la ligne à partir de la position courante du fichier

Préconditions:

ouvrir(f):	non estOuvert(f)
fermer(f):	estOuvert(f)
finFichier(f):	mode(f)=lecture
lireXX:	estOuvert(f) et mode(f)=lecture et non finFichier(f)
ecrireXX:	estOuvert(f) et mode(f)=ecriture

1.1.2 TAD Mot

Nom: mot

Utilise: **Chaine de caracteres**, NaturelNonNul, **Caractere**, **Booleen**

Opérations:

estUneLettre:	**Caractere** → **Booleen**
estUnMot:	**Chaine de caracteres** ↛ **Booleen**

creerMot: **Chaine de caracteres** ↛ Mot
estEgal: Mot × Mot → **Booleen**
longueur: Mot → NaturelNonNul
motEnChaine: Mot → **Chaine de caracteres**
remplacerLettre: NaturelNonNul × **Caractere** × Mot ↛ Mot
supprimerLettre: NaturelNonNul × Mot ↛ Mot
insererLettre: **Caractere** × Mot × NaturelNonNul ↛ Mot
inverserLettre: NaturelNonNul × Mot ↛ Mot
decomposerMot: Mot × NaturelNonNul ↛ Mot × Mot

Sémantiques: estUneLettre: renvoie vrai si l'élément pris en entré est une lettre (minuscule ou majuscule)
estUnMot: vérifie que c'est un mot
creerUnMot: Créé un mot à partir d'une chaîne de caractères.
estEgal: retourne vrai si les mots sont identiques (non sensible à la casse)
longueur: calcule la longueur d'un mot
motEnChaine: transforme un mot en chaîne de caractères
remplacerLettre: remplace une lettre à un certain indice du mot
supprimerLettre: supprime une lettre à un certain indice du mot
insererLettre: insère une lettre à un certain indice du mot
inverserLettre: inverse deux lettres consécutifs du mot
decomposerMot: dcompose un mot en deux mots

Préconditions: estUnMot(mot): longueur(mot)>1 ET estUneLettre(mot[i]) ∀ 1≤i≤longueur(mot)
creerMot(c): estUnMot(c)
remplacerLettre(i, car, mot): i≤longueur(mot)
supprimerLettre(i, mot): i≤longueur(mot)
insererLettre(c, mot, i): i<longueur(mot)
inverserLettre(i, mot): estOuvert(f) et mode(f)=lecture et non finFichier(f)
decomposerMot(mot, i): 1 < i < longueur(mot)

1.1.3 TAD Dictionnaire

Nom: Dictionnaire
Utilise: fichierTexte, Dictionnaire, Mot, **Booleen**, mode
Opérations: dictionnaire: → Dictionnaire
estVide: Dictionnaire → **Booleen**
ajouterMot: Dictionnaire × Mot ↛ Dictionnaire
ajouterFichier: Dictionnaire × FichierTexte ↛ Dictionnaire
chargerDictionnaire: **Chaine de caracteres** → Dictionnaire

	sauvegarderDictionnaire:	Dictionnaire → **Booleen**
	estPresent:	Dictionnaire × Mot → **Booleen**
Sémantiques:	dictionnaire:	Créé un dictionnaire vide
	estVide:	vérifie si un dictionnaire est vide
	ajouterMot:	ajoute un mot dans le dictionnaire
	ajouterFichier:	ajoute les mots d'un fichier texte dans le dictionnaire
	chargerDictionnaire:	charge un dictionnaire à partir d'un fichier
	sauvegarderDictionnaire:	créé un fichier texte contenant tous les mots du dictionnaire mis en paramètre.
	estPresent:	indique si un mot est présent dans le dictionnaire
Préconditions:	ajouterFichier(f):	mode(fichierTexte) = lecture
	ajouterMot(m):	non(estPresent(m))
	supprimerMot(m):	estPresent(m)

1.1.4 TAD CorrecteurOrthographique

Nom:	CorrecteurOrthographique	
Utilise:	Mot, Dictionnaire, **Booleen**, NaturelNonNul	
Opérations:	chainesEnMots:	**Chaine de caracteres** → Liste <Mot> × Liste <Naturel>
	sontPresents:	Liste <Mot> × Dictionnaire → Liste <**Booleen**>
	proposerMots:	Mot × Dictionnaire → Ensemble <Mot>
	proposerMotsListe:	Liste <Mot> × Dictionnaire → Liste <Ensemble <Mot» × Liste <**Booleen**>
Sémantiques:	chainesEnMots:	transforme une chaîne de caractères en une liste de Mot et renvoie aussi une liste de la position de chaque mot
	sontPresents:	renvoie une liste de Booleen indiquant la présence ou non des mots de la liste
	proposerMots:	donne un ensemble de Mot correspondant aux corrections possibles d'un mot mal orthographié
	proposerMotsListe:	envoie la liste des ensembles de Mot corrigés possible pour les mots qui ne sont pas dans le dictionnaire

1.2 Analyse Descendante

2 Conception Préliminaire

2.1 Signature FichierTexte

fonction fichierTexte (chaine : **Chaine de caracteres**) **:** FichierTexte
procédure ouvrir (**E/S** fichier : FichierTexte, **E** mode : Mode)

⌊**précondition(s)** non (estOuvert(f))

procédure fermer (**E/S** fichier ; FichierTexte)

⌊**précondition(s)** estOuvert(f)

fonction estOuvert (fichier : fichierTexte) **: Booleen**
fonction Mode (fichier : FichierTexte) **:** Mode
fonction finFichier (fichier : FichierTexte) **: Booleen** mode(fichier)=lecture
procédure ecrireChaine (**E/S** fichier : FichierTexte, **E Chaine de caracteres**)

⌊**précondition(s)** estOuvert(fichier)
mode(fichier)=ecriture

procédure lireChaine (**E/S** fichier : FichierTexte, **S Chaine de caracteres**)

⌊**précondition(s)** estOuvert(fichier)
mode(fichier)=lecture

procédure ecrireCaractere (**E/S** fichier : FichierTexte, **E Caractere**)

⌊**précondition(s)** estOuvert(fichier)
mode(fichier)=ecriture

procédure lireChaine (**E/S** fichier : FichierTexte, **S Caractere**)

⌊**précondition(s)** estOuvert(fichier)
mode(fichier)=lecture

2.2 Signature Mot

fonction estUneLettre (c : **Caractere**) **: Booleen**
fonction longueur (m : Mot) **: NaturelNonNul**
fonction estUnMot (chaine : **Chaine de caracteres**) **: Booleen**

⌊**précondition(s)** longueur(c)>1 ET estUneLettre(c[i]) $\forall 1 \leq i \leq longueur(m)$

fonction creerMot (chaine : **Chaine de caracteres**) **:** m : Mot

⌊**précondition(s)** estUnMot(chaine)

fonction sontEgaux (m1, m2 : Mot) **: Booleen**
fonction motEnChaine (m : Mot) **: Chaine de caracteres**
fonction remplacerLettre (m : Mot, pos : **NaturelNonNul**, c : **Caractere**) **:** Mot

⌊**précondition(s)** $pos \leq longueur(m)$

fonction supprimerLettre (m : mot, pos : NaturelNonNul) **:** Mot

⌊**précondition(s)** $pos \leq longueur(m)$

fonction insererLettre (m : Mot, c : **Caractere**, pos : **NaturelNonNul**) **:** Mot

⌊**précondition(s)** $pos \leq longueur(m)$

fonction inverserLettre (m : Mot, pos : **NaturelNonNul**) **:** Mot

⌊**précondition(s)** $pos < longueur(m)$

fonction decomposerMot (m : Mot, pos : **NaturelNonNul**) **:** Mot, Mot

⌊**précondition(s)** $1 < pos < longueur(m)$

2.3 Signature Dictionnaire

fonction Dictionnaire () **:** Dictionnaire
fonction estVide (dictionnaire : Dictionnaire) **: Booleen**
fonction estPrésent (dictionnaire : Dictionnaire, motChercher : mot) **: Booleen**
procédure ajouterMot (**E**/**S** dictionnaire : Dictionnaire,**E** mot : Mot)

⌊**précondition(s)** non(estPrésent(mot))

procédure ajouterFichier (**E**/**S** dictionnaire : Dictionnaire, **E** fichier : FichierTexte)

⌊**précondition(s)** mode(fichierTexte) = lecture

procédure chargerDictionnaire (**E** chaine : **Chaine de caracteres**, **S** dictionnaire : Dictionnaire)
fonction enregistrerDictionnaire (dictionnaire : Dictionnaire) **:** FichierTexte

2.4 Signature CorrecteurOrhographique

fonction chaineEnMots (c : **Caractere**) **:** Liste<Mot>, Liste<**Naturel**>
fonction sontPresents (mots : Liste<Mot>, d : Dictionnaire) **:** Liste<**Booleen**>
fonction proposerMot (m : Mot, d : Dictionnaire) **:** Ensemble<Mot>
fonction proposerMotListe (liste : Liste<Mot>, d : Dictionnaire) **:** Liste<Ensemble<Mot», Liste<Booleen>

3 Conception Détaillé

3.1 Conception Détaillé du TAD Fichier Texte

3.2 Conception Détaillé du TAD mot

3.3 Conception Détaillé du TAD Dictionnaire

3.4 Conception Détaillé du TAD Correcteur Orhographique

Constante alphabet = 'abcdefghijklmnopqrstuvwxyzéèêëàâäùüûôöïîÿç'

fonction sontPresents (mots : Liste<Mot>, d : Dictionnaire) **:** Liste<Booleen>
  Déclaration i : **Naturel**, resultat : Liste<Booleen>
debut
  pour i ←1 **à** Liste.longueur(mots) **faire**
    ajouter(resultat, estPresent(Liste.obtenirElement(mots, i), d))
  finpour
  retourner resultat
fin

fonction proposerMots (m : Mot, d : Dictionnaire) **:** Ensemble<Mot>
  Déclaration motCorrige, motCorrige2 : Mot, resultat : Ensemble<Mot>
debut
  pour k ←1 **à** Chaine.longueur(alphabet) **faire**
    pour i ←1 **à** Mot.longeur(m) **faire**
      motCorrige <- remplacerLettre(m, alphabet[k], i)

      si Dictionnaire.estPresent(motCorrige, d) **alors**
        ajouter(resultat, motCorrige)
      finsi
    finpour
    pour i ←1 **à** Mot.longeur(m) + 1 **faire**
      motCorrige <- insererLettre(m, alphabet[k], i)

      si Dictionnaire.estPresent(motCorrige, d) **alors**
        ajouter(resultat, motCorrige)
      finsi
    finpour
  finpour
  pour i ←1 **à** Mot.longeur(m) **faire**
    motCorrige <- supprimerLettre(m, i)

```
        si Dictionnaire.estPresent(motCorrige, d) alors
            ajouter(resultat, motCorrige)
        finsi
    finpour
    pour i ←2 à Mot.longeur(m) - 1 faire
        motCorrige, motCorrige2 <- decomposerMot(m, i)

        si Dictionnaire.estPresent(motCorrige, d) alors
            ajouter(resultat, motCorrige)
        finsi
        si Dictionnaire.estPresent(motCorrige2, d) alors
            ajouter(resultat, motCorrige2)
        finsi
    finpour
    retourner resultat
fin

fonction proposerMotsListe (mots : Liste<Mot>, d : Dictionnaire) : Liste< Ensemble<Mot> >
    Déclaration   resultat : Liste< Ensemble<Mot> >
debut
    pour i ←1 à Liste.longeur(mots) faire
        ajouter(resultat, proposerMots(obtenirElements(mots, i), d))
    finpour
fin
```