Quentin DAMOIS
Enzo RIVIERE
Yaël TAHAR

Rapport de projet : générateur automatique de phrases

Promotion 2026 L2-A

Table des matières

Introduction

Le projet que nous avons réalisé, lors du cours d'Algorithmique et structure de données 2, consiste à pouvoir générer des phrases bien accordées automatiquement. Pour cela, nous sommes passés par différentes étapes, telles qu'extraire un mot aléatoirement, ou encore générer des phrases suivant une structure prédéfinie, mais sans respecter les accords.

I. Présentation fonctionnelle du projet

1. Les structures

Les nœuds de l'arbre

Afin de stocker tous les mots contenus dans le dictionnaire, il nous a fallu créer un arbre. Cependant, la structure des nœuds devait être différente de celle vue en classe, car il y a plus d'informations à stocker.

Le stockage des formes fléchies

Les nœuds ne pouvaient pas stocker toutes les informations, alors nous avons créé une autre structure qui stocke les formes fléchies et qui permet de les exploiter facilement.

2. Création de l'arbre

Lecture du dictionnaire

La fonction de lecture du fichier va permettre de lire le fichier ligne par ligne afin de pouvoir par la suite ajouter les mots dans l'arbre.

Vérification du type de mot

Cette fonction va identifier le type de mot à ajouter, si c'est un nom, un verbe, un adjectif, un adverbe.

Ajout d'un mot dans l'arbre

La fonction ajoute un mot dans l'arbre.

Ajout des formes fléchies

Ici, la fonction va permettre d'ajouter les différentes formes fléchies.

3. Générateur de phrases

Génération d'un mot brut

Nous naviguons aléatoirement dans un arbre et nous sortons une variable de type mot contenant la forme de base ainsi que les fléchies et leurs formes grammaticales.

Génération d'une phrase brute

Nous utilisons la génération d'un mot brute pour générer tous les mots de la structure de phrase sélectionnée.

Génération aléatoire d'un fléchie

Nous choisissons aléatoirement un fléchi parmi ceux du mot sélectionné.

Comparaison avec le mot

Nous comparons les genres et nombres d'un mot avec celle d'un fléchi d'un autre.

Accord d'un mot

Nous prenons un mot déjà généré et nous comparons son genre et nombre avec ceux des fléchis d'un autre mot pour trouver le fléchi accorder.

Comparaison d'un mot avec un verbe

On compare un mot déjà généré avec les déclinaisons d'un verbe.

Accord du mot avec un verbe

Nous regardons les déclinaisons d'un verbe et nous sélectionnons celle qui correspond au mot choisi.

Génération d'une phrase accordée

Cette fonction prend les mots générés correctement accordés et les affichent pour faire une phrase correspondant à la structure choisie.

4. Recherche de mots

Vérification de la présence d'une base dans l'arbre

Cette fonction va parcourir lettre par lettre le mot cherché et le comparer avec les lettres de l'arbre afin de savoir s'il existe dans notre arbre.

Recherche de la base

La fonction de recherche de la base va utiliser celle de vérification de la présence d'une base dans l'arbre et afficher son type (un adjectif, un nom, un verbe ou un adverbe).

Trouver le nœud contenant le mot fléchi

Cette fonction parcourt tout l'arbre de manière récursive et recherche la forme fléchie demandée, elle retourne le nœud où est stocké la forme fléchie que l'on recherche.

Trouver la forme fléchie dans la liste

Grace au nœud retourné avant, nous pouvons parcourir sa liste afin de trouver la forme fléchie et son type correspondant.

Afficher la forme fléchie pour chaque arbre

La fonction affiche la forme de base et le type de la forme fléchie trouvée, et ce pour chaque arbre.

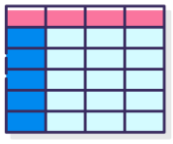
II. Présentation technique du projet

1. Les structures

Les nœuds de l'arbre

La structure des nœuds de nos arbres, aussi appelée p_node, est la suivante :

Caractère : Valeur



Enfants : Tableau dynamique de p_node

Entier : Nombre d'enfants

Entier : vérifie si c'est la dernière lettre d'un mot

Pointeur vers un type « mot » (Stocke les formes fléchies)

Le stockage des formes fléchies

Comme vu précédemment le type mot est utilisé dans la structure d'un p_node, elle est composée des éléments suivent :

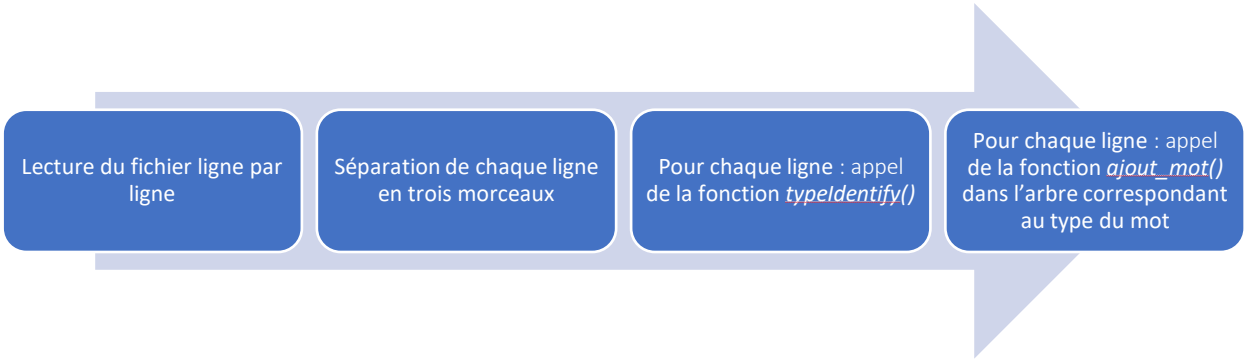
Chaine de caractère : Mot complet sous forme de base

Entier : Nombre de formes fléchies

- Une std_list pour les formes grammaticales
- Une std_list pour les mots fléchis

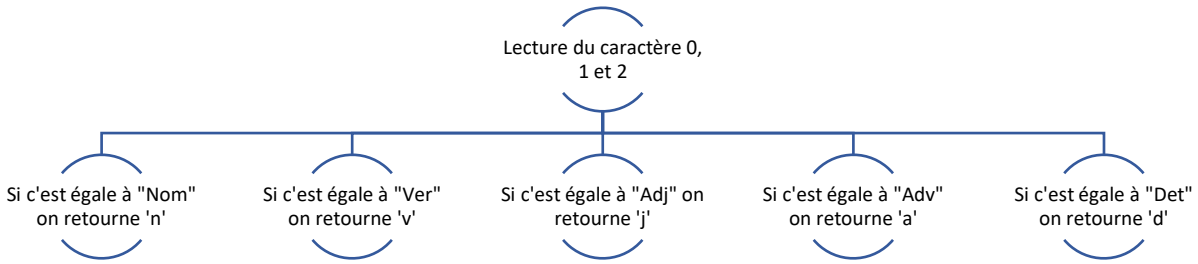
2. La création de l'arbre

Lecture du dictionnaire



Vérification du type de mot

Afin de vérifier le type d'un mot on va alors seulement utiliser la forme grammaticale du mot que l'on stock dans une chaine de caractère.



Ajout d'un mot dans l'arbre

Nous avons pris parti de créer au préalable des arbres avec un premier p_node contenant comme valeur « », afin que ses enfants soient les premières lettres des différents mots du dictionnaire. Ensuite, nous ajoutons simplement un mot de la manière suivante :

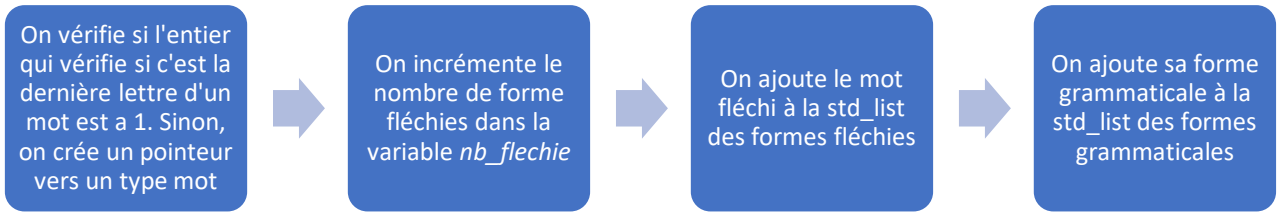
1. On parcourt le mot avec la forme de base lettre par lettre (on s'arrête à l'avant-dernière lettre).
2. On regarde si la lettre est dans la liste d'enfants de la lettre où on est.

- Si l'enfant existe, on se place dessus.
- Sinon on le rajoute avec une fonction *ajout_enfant()* (cette fonction ajoute simplement une place dans le tableau dynamique d'enfant et y rajoute le caractère) et on se place dessus.

3. On fait la même chose avec la dernière lettre de la forme de base du mot et on appelle la fonction *ajout_flech()*.

Ajout des formes fléchies

Cette fonction est appelée lors de l'ajout d'un mot dans un arbre.



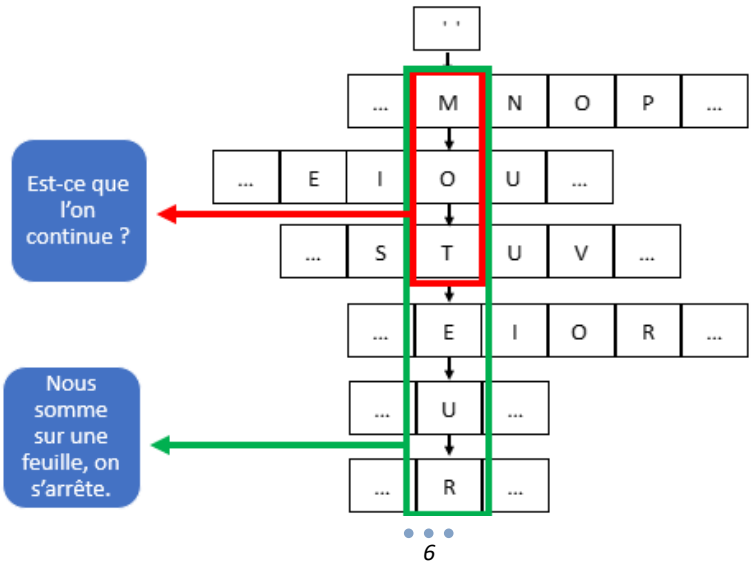
3. Générateur de phrases

Cette partie comporte 9 fonctions différentes.

Génération d'un mot brut

Un mot brut est la forme de base d'un mot sans marque ni du féminin ni du pluriel et sans conjugaison pour les verbes. Suivant le type de mot que nous voulons sélectionner, nous prenons l'arbre approprié en le mettant comme argument de la fonction que nous avons créé.

1. Nous partons du nœud de base de l'arbre et nous passons au premier nœud enfant. Nous choisissons aléatoirement le premier nœud correspondant à la première lettre du mot.
2. À chaque choix de lettre, nous regardons si le mot formé existe.
3. Si c'est le cas et si le nœud a des enfants, nous choisissons alors aléatoirement de continuer ou de s'arrêter. Nous procédons ainsi à chaque nœud sauf si nous arrivons sur une feuille, nous nous arrêtons alors obligatoirement.
4. Quand nous décidons de nous arrêter, nous stockons l'adresse de ce mot dans un pointeur.



Génération d'une phrase brute

Nous affichons la phrase brute sans accord ni conjugaison.

1. Pour afficher la phrase brute suivant la structure choisie, nous actionnons la fonction décrite précédemment pour chaque mot de la phase. Nous utilisons l'arbre correspondant au type de mot comme argument de la fonction.
2. Pour l'affichage, nous regardons l'argument « cas » de la fonction qui indique la structure sélectionnée par l'utilisateur.
3. Nous affichons les mots dans l'ordre approprié et en les séparant par des espaces.

Génération aléatoire du fléchie du premier mot

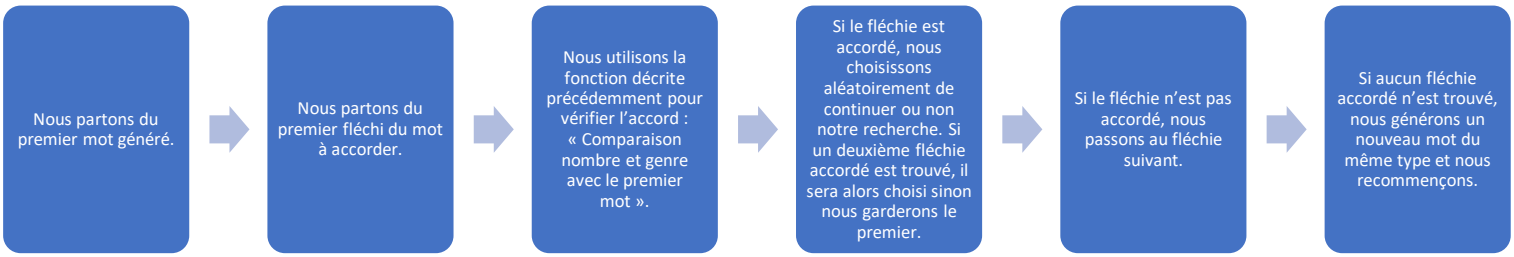
Comparaison nombre et genre avec le premier mot

1. Nous regardons le genre et nombre du premier mot choisi et nous utilisons un pointeur pour chacune de ces informations.
2. Pour un fléchi d'un mot à accorder avec le premier, nous regardons ses genres et nombres.
3. Si le premier mot ou le fléchie est invariable, le genre du deuxième peut être quelconque. Si ce n'est pas le cas, le genre du premier mot et du fléchie doivent être identiques.
4. Pour ce qui du nombre, si le nombre du fléchie ou celle de la déclinaison est invariable alors nous ne regardons pas s'ils ont le même nombre.

Comparaison du verbe avec le mot

Nous faisons la même chose que pour la comparaison nombre et genre, mais nous regardons si le fléchie du verbe est à la troisième personne au lieu de regarder le genre.

Accord du fléchie pour un mot



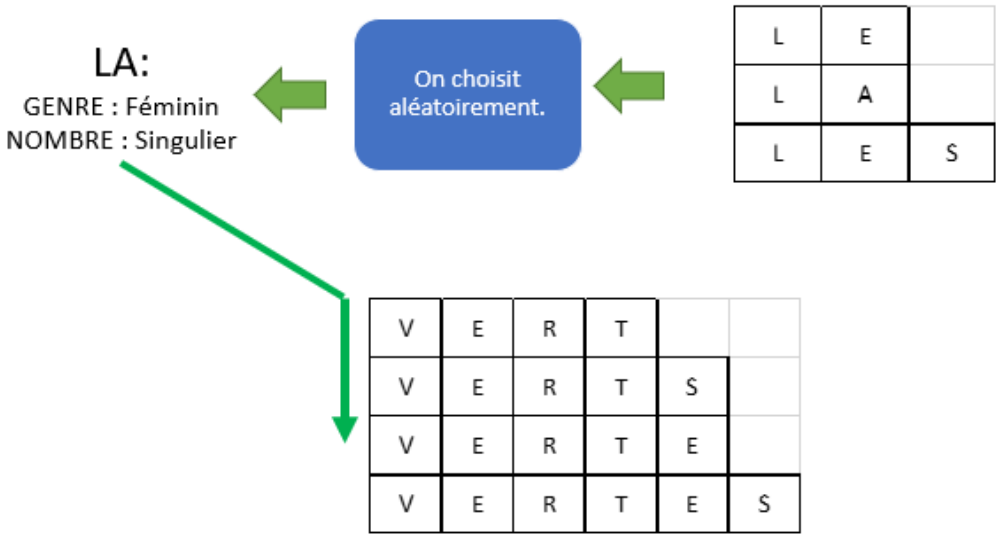
Accord du fléchie pour un Verbe

On fait la même chose que pour l'accord du fléchie pour un mot.

Génération d'une phrase accordée

1. On génère d'abord la forme brute de chaque mot nécessaire à réaliser la structure de phrase choisie.
2. On accorde les mots en considérant le déterminant comme premier mot. On accorde le reste des mots en fonction des genres et nombres du premier mot.

3. Une fois tous les fléchies générés, on les affiche en respectant la structure sélectionnée et en séparant les fléchies par des espaces.

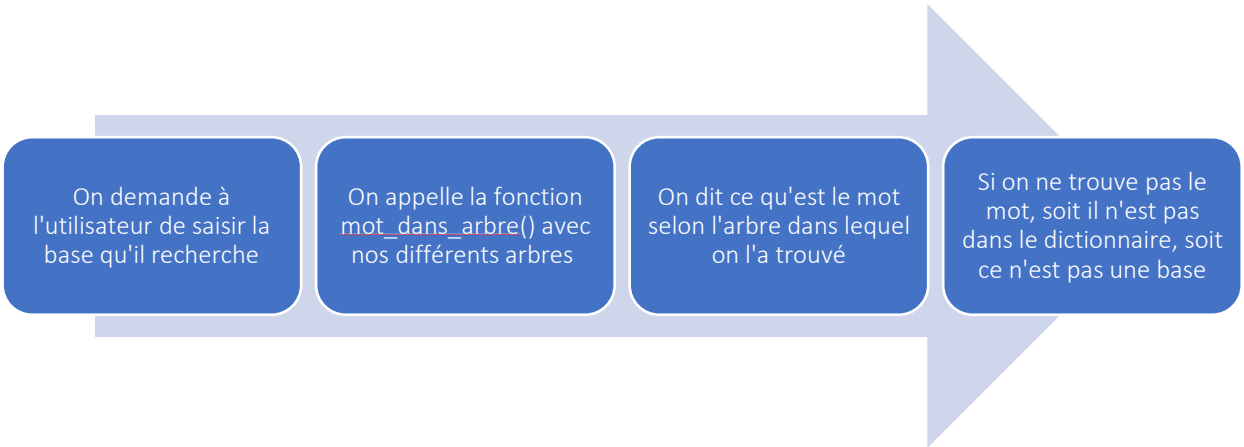


4. Recherche de mots

Vérification de la présence d'une base dans l'arbre

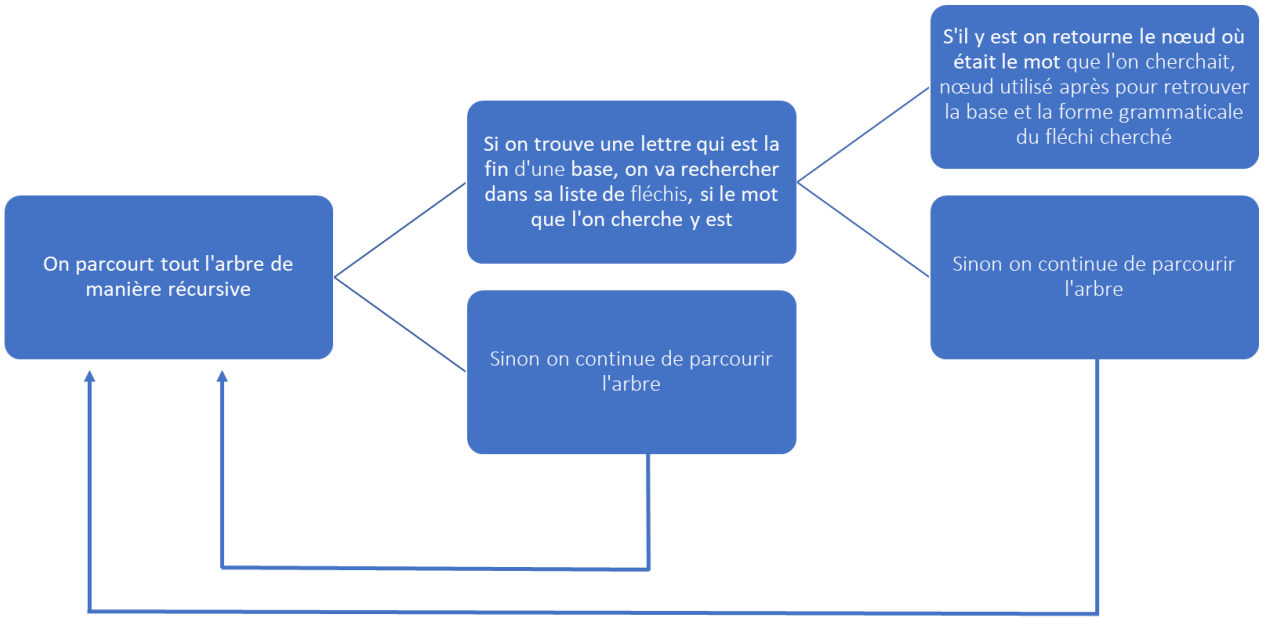
1. On parcourt le mot de base lettre par lettre.
2. On regarde si la lettre est dans le tableau.
   - Si la lettre est dans le tableau : on se place sur cette lettre, on va à son tableau d'enfants et on recommence avec la lettre suivante.
   - Sinon on retourne 0.
3. Si on a retrouvé le mot dans l'arbre, on retourne 1.

Recherche de la base



La fonction de recherche de la base va utiliser celle de vérification de la présence d'une base dans l'arbre dans chacun des arbres que nous avons créés et afficher de quel type de base il s'agit en fonction de cela (un nom, un verbe, un adjectif ou un adverbe).

Trouver le nœud contenant le mot fléchi



Grace au nœud retourné avant, nous pouvons parcourir sa liste afin de trouver la forme fléchie et son type correspondant.

Trouver la forme fléchie dans la liste

1. Si la fonction trouvant le nœud est égale à NULL, on retourne 0.
2. Sinon :
    - On compare le mot que l'on cherche avec chaque mot de la liste (grâce à leurs lettres et leurs tailles afin d'être sûr qu'ils soient vraiment égaux).
    - Dès qu'on l'a trouvé, on le récupère, ainsi que sa position.
    - Grâce à cela, on retrouve son type, qui est à la même position dans une autre liste.
    - Enfin, on affiche sa forme de base, son type et on retourne 1.

Afficher la forme fléchie pour chaque arbre

1. On utiliser notre fonction d'avant pour chercher la forme fléchie dans nos différents arbres (nom, adjectif, adverbe, verbe).
2. Nous allons additionner l'entier que renvoie la fonction dans une variable res.
3. Si à la fin la variable res est toujours égale à 0, on affiche que le mot recherché n'est pas dans le dictionnaire.

5. Problèmes rencontrés

Une première difficulté concerne l'utilisation de la bonne structure d'arbre. Nous avions vu plusieurs types d'arbre et nous avons dû choisir le bon. Sans cela, nous ne pouvions pas générer celui-ci.

La deuxième difficulté concerne la détection des verbes. Les participes présents, par exemple, sont dans les déclinaisons des verbes. Cependant, ils ne s'accordent pas et ne peuvent pas être pris comme un verbe conjugué dans une phrase. Une fois la déclinaison du verbe choisi, il a donc fallu

mettre en place une détection qui vérifiait que nous n'étions pas en présence d'un participe présent en regardant tous les caractères qui décrivaient son temps.

La troisième et dernière difficulté est apparue lorsque nous changions le mot quand il n'avait pas de déclinaison qui correspondait au nom. On appelait récursivement la fonction de recherche dans la recherche. Cette action prenait beaucoup de mémoire vive. Le programme venait donc à saturer la mémoire et l'ensemble s'arrêtait avant la fin. Pour régler le problème, nous avons décidé de procéder itérativement en utilisant une boucle et en mettant en condition de sortie de boucle d'avoir la déclinaison choisie avec le nom.

III. Présentation des résultats

Tout d'abord, un menu s'affiche avec les différentes fonctionnalités proposées à l'utilisateur.

```
Que voulez-vous faire ?
1. Extraire une forme de base au hasard.
2. Generer une phrase automatiquement.
3. Rechercher un mot.
Veuillez saisir une valeur entre 1 et 3.
```

Extraction d'une forme de base au hasard

Si l'utilisateur rentre le nombre 1, alors un mot de base pris au hasard dans l'arbre va alors d'afficher.

```
1
dragueur
```

Génération d'une phrase automatiquement

On propose à l'utilisateur de générer une phrase avec des mots sous forme de base ou alors sous forme fléchie.

```
Quel type de phrase voulez-vous ?
1. Des phrases avec des mots de bases.
2. Des phrases avec des mots flechies.
Veuillez saisir une valeur entre 1 et 2.
```

Dans les deux cas, on lui propose alors 3 types de structures de phrase.

```
Quelle forme de phrase voulez-vous ?
1. nom - adjectif - verbe - nom
2. nom - 'qui' - verbe - verbe - nom - adjectif
3. nom - adjectif - verbe - adverbe
Veuillez saisir une valeur entre 1 et 3.
```

Et cela génère par la suite une phrase avec les paramètres que l'utilisateur a demandé :

```
kwas nippon cagner hi-han.
```

Phrase de la forme 1 avec des mots de base

```
cinquante byssus qui egrisaient kilometrent force wombat jersiais.
```

Phrase de la forme 2 avec des mots sous formes fléchies

Rechercher un mot

Un nouveau menu va alors être proposé à l'utilisateur, pour savoir s'il cherche un mot sous forme de base ou alors un mot sous forme fléchie.

```
Quel type de mot voulez-vous rechercher ?
1. Un mot de base.
2. Un mot flechie.
Veuillez saisir une valeur entre 1 et 2.
```

Si l'utilisateur choisie de chercher un mot sous forme de base :

```
Quel mot de base cherchez-vous ?
escalier
La base escalier est un nom
```

Si l'utilisateur choisie de chercher un mot sous forme fléchie :

```
Quel mot flechie cherchez-vous ?
preneuse
Le mot preneuse vient du mot de base preneur avec la forme grammaticale suivante : Nom:Fem+SG
Le mot preneuse vient du mot de base preneur avec la forme grammaticale suivante : Adj:Fem+SG
```

Revenir au menu principal

Après chaque action, on propose à l'utilisateur de revenir au menu principal où il peut choisir d'effectuer une nouvelle action, ou alors de quitter le programme.

```
Voulez-vous revenir au menu ou arreter ?
1. Revenir au menu
2. Arreter
Veuillez saisir 1 ou 2
```

Conclusion

Lors de ce projet, nous avons appris diverses choses telles que l'importance de la communication dans un groupe, mais aussi à gérer notre temps et à répartir le travail afin de finir le projet dans les temps.

Ce projet nous a aussi permis d'améliorer nos connaissances avec GitHub puisque nous avons utilisé l'intégration de GitHub dans Clion. Nous avons donc pu améliorer notre coordination grâce à cette intégration.

Grâce aux connaissances que nous avons acquises, nous allons pouvoir réaliser nos projets futurs de manière plus efficace.

Notre projet pourrait avoir des améliorations telles que pouvoir générer des phrases avec des pronoms personnels, rajouter les formes fléchies manquantes ou encore d'identifier chaque partie de la forme grammaticale afin de faire une phrase plus jolie à l'utilisateur lors de la recherche d'un mot sous forme fléchie.

Dépôt Github : https://github.com/Enzo-Riviere/Generateur-de-phrases-auto.git