

N3



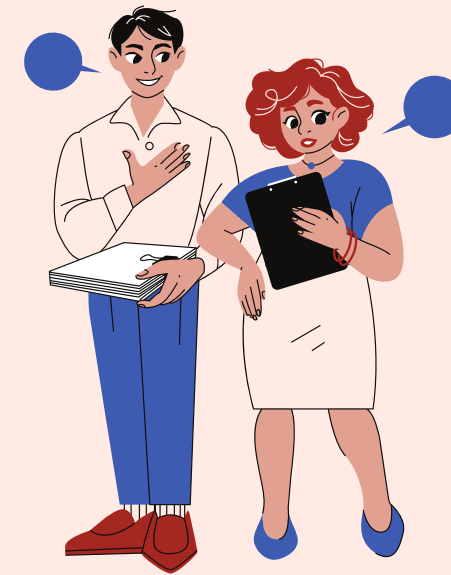
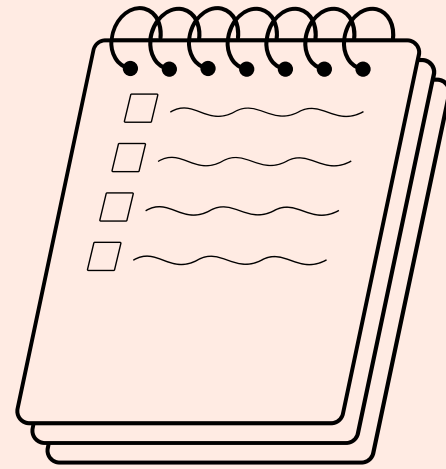
# Sommaire : La compréhension du sujet

1. Cahier des charges
2. Plannification
3. Fonctionnalités et utilisation
4. Le main.c
5. Le MotDeRech.h
6. Démonstration
7. Validation du cahier des charges
8. Difficultés
9. Résolutions
10. Evolutions et améliorations pour la version ALPHA
11. Conclusion



# CAHIER DES CHARGES

Vérifions si les demandes  
ont bien été remplies

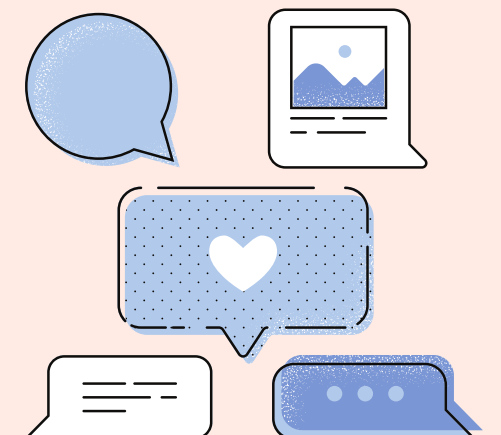


## Exigences techniques minimales

1. Programme en langage C
2. Environnement de développement Linux
3. Programme découpé en plusieurs modules
4. Utilisation de l'allocation dynamique lors des chargements de fichiers CRI.

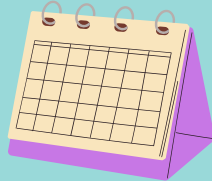
## Exigences fonctionnelles

5. Indexer automatiquement un ensemble de fichier (majuscules, minuscules, singulier, pluriel)
6. Rechercher et classer par pertinence en fonction de la recherche les fichiers qui répondent le mieux aux critères.





# PLANIFICATION

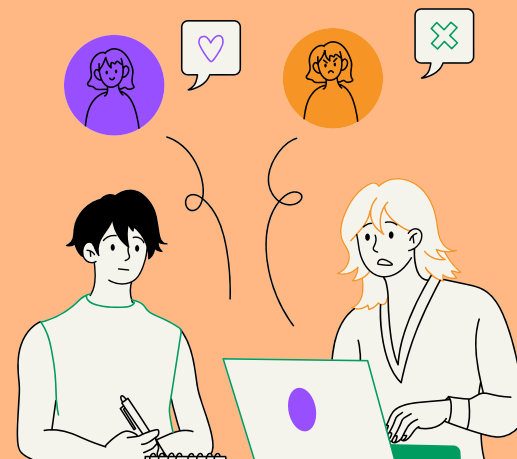


COMPRÉHENSION DU SUJET	PLANIFICATION	CRÉATION DES FICHIERS CRI	RECHERCHE DU MOT CHOISI DANS LES DIFFÉRENTS FICHIERS	PROGRAMME PRINCIPAL
compréhension des objectifs demandé	établir un plan des tâches à réaliser	séparer chaque mot les mettre en minuscule enlever les symboles et la ponctuation	réaliser un code permettant de retourner si le mot qu'on cherche et le mot actuel sont les mêmes	créer l'interface ayant accès à un menu
Identifier les différentes étapes logique du moteur de recherche	mettre à jour le plan des tâches au fil de nos avancées	compter le nombre d'occurrences de chaque mot	fonction permettant de trier le nombres d'occurrences par rapport au fichier	pouvoir réaliser la recherche de/ des mot/s que l'on souhaite
éclairer les doutes en posant les questions aux enseignants	finir toutes les tâches établies	créer et remplir les fichiers CRI pour chaque fichier .txt	fonction permettant de faire des recherches sur plusieurs mots	Obtenir un classement des texte les plus pertinents et le nombre d'occurrences du mot en sortie



## Fonctionnalités

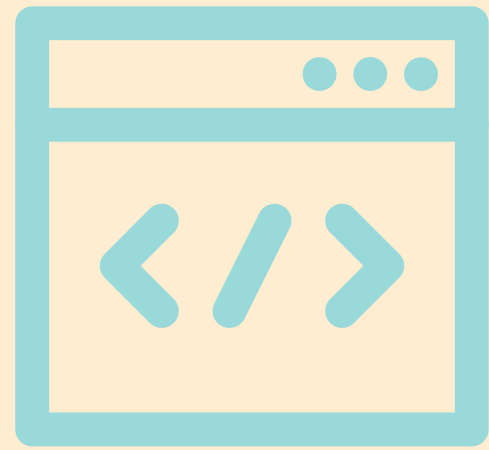
**Le code doit indexer automatiquement un ensemble de fichier et doit rechercher et classer par pertinence en fonction de la recherche les fichiers qui répondent le mieux aux critères.**



## Utilisation

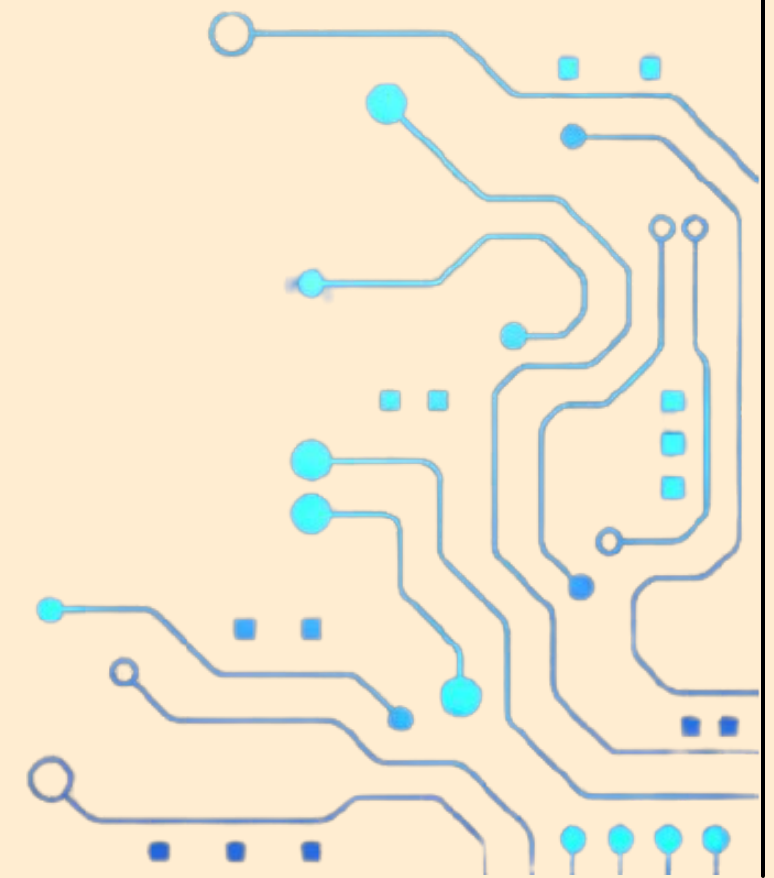
**On peut inscrire plusieurs mots et avoir les listes respectives classées par le nombre d'occurrence du mot dans les textes**






# Les Fonctions créées : main.c

Présentation des différentes fonctions créées.

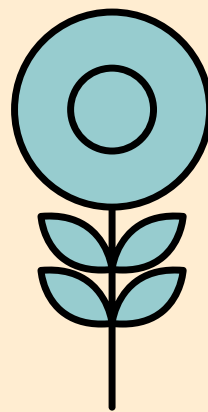
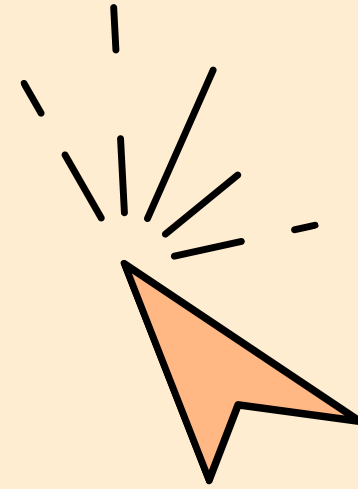


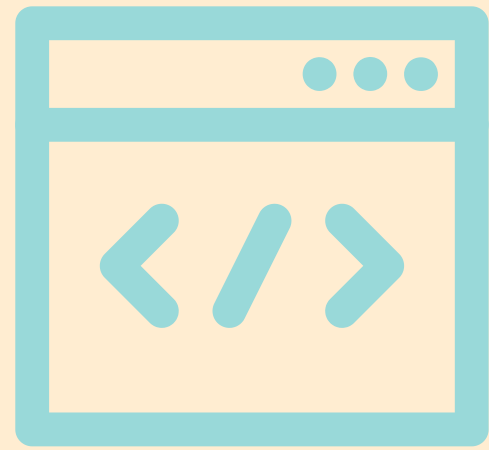
## main.c :

Awesome Web Browser X

← → ↻  <https://www.nosfonctionsontgéniales.fr> ⋮

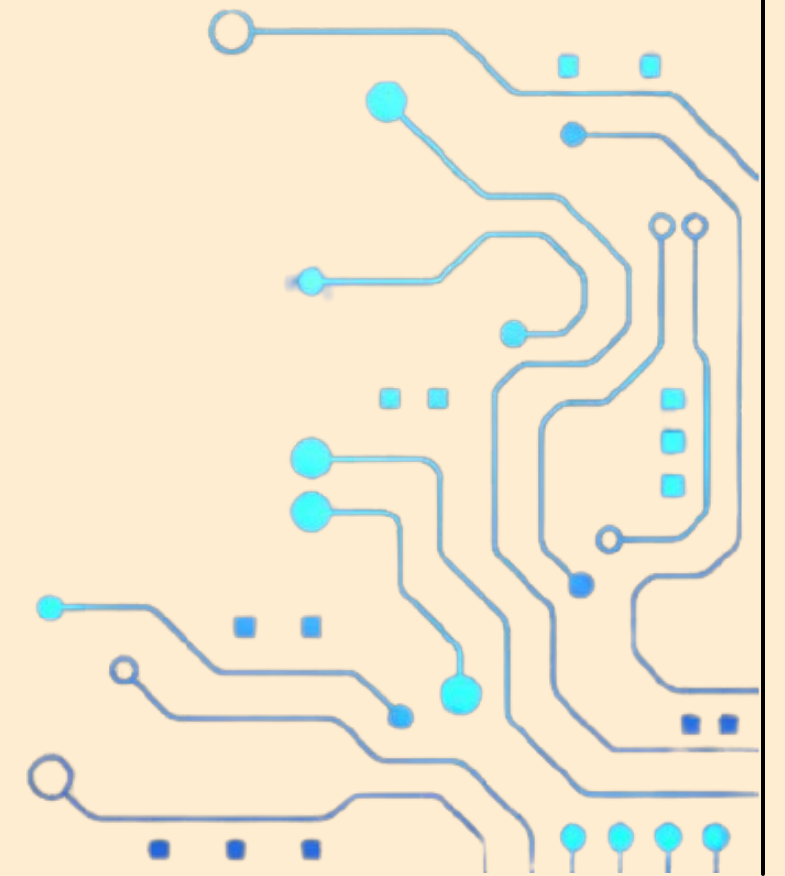
```
1
2 #include "MotDeRech.h"
3
4 int main(int argc, char *argv[])
5 {
6     Créa_glob_CRI();
7     for(int i = 1; i < argc; i++)
8     {
9         Find_glob_Doc(argv[i]);
10    }
11 }
12
```





# Les Fonctions créées : MotDeRech.h

Présentation des différentes fonctions créées.

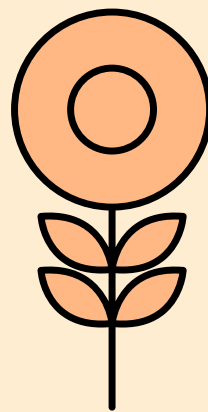
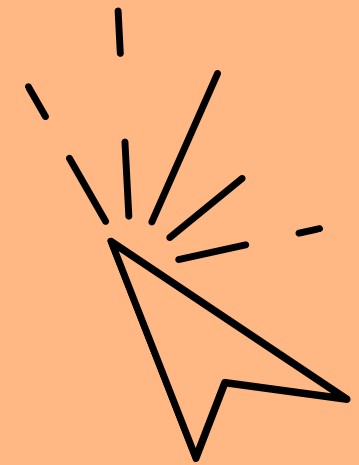




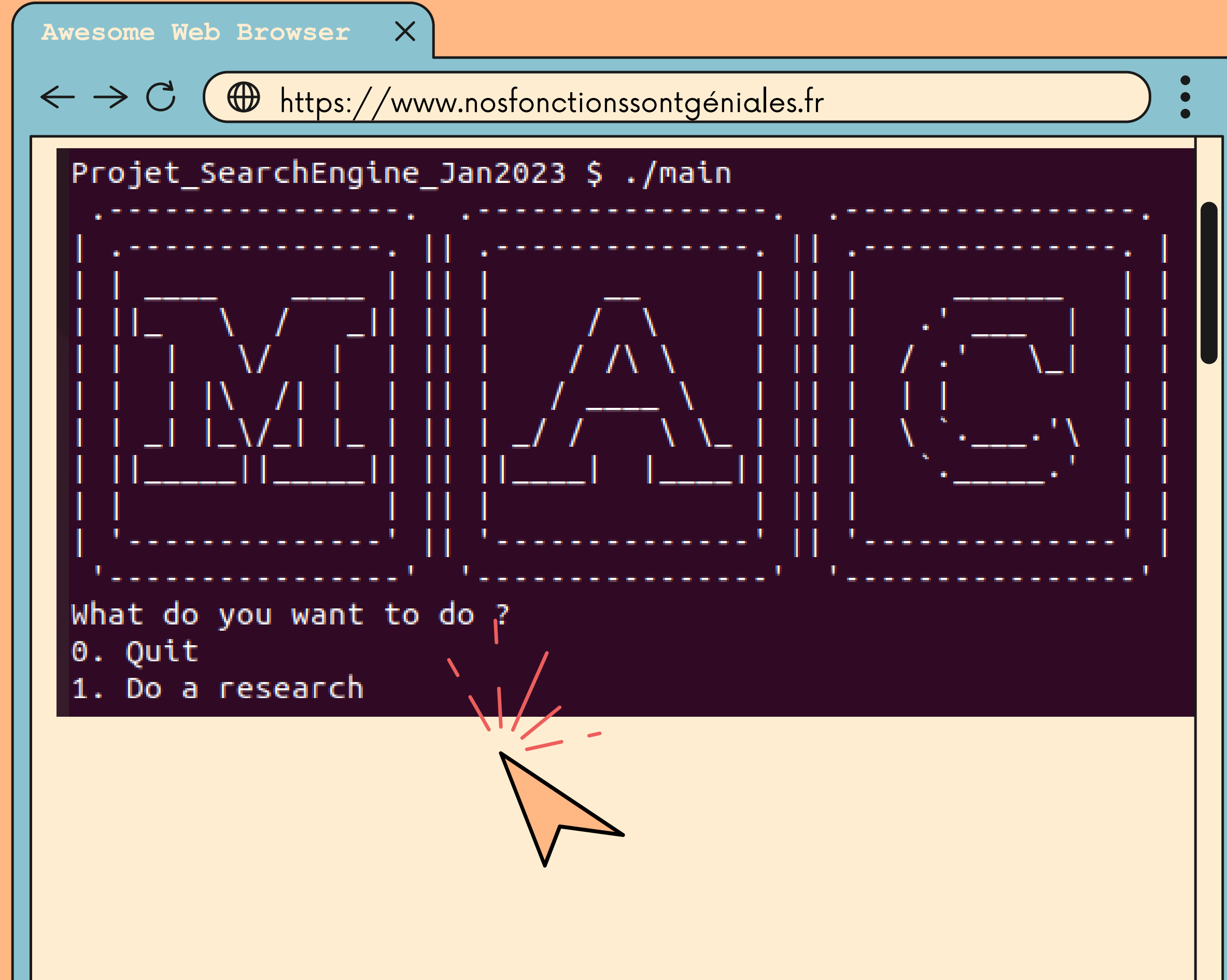
Awesome Web Browser

← → ↻  <https://www.nosfonctionssontgéniales.fr> ⋮

- **void Créa\_glob\_CRI()**
- **void Créa\_CRI(const char \*NomFichier)**
- **void Find\_glob\_Doc(const char \*MotATrouver);**
- **int Find\_Doc(const char \*MotATrouver, const char \*NomFichier);**
- **bool Mot\_Egaux(const char \*Mot1, const char \*Mot2);**
- **bool hasOnlyLetters(const char \*Mot);**
- **char \*transformer\_en\_minuscule\_sans\_ponctuation(const char \*Mot);**
- **char \*SingularTransfo(const char \*Mot);**

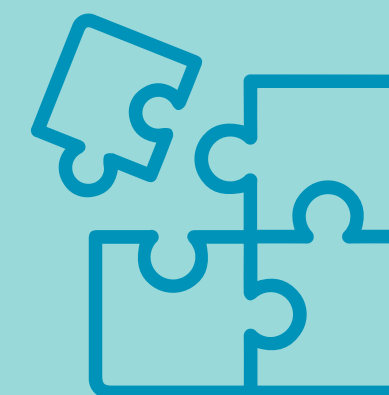
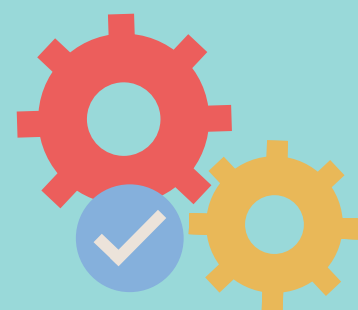


# Affichage:



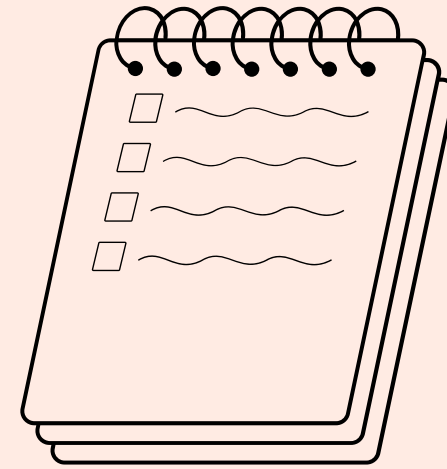


# Demonstration



# CAHIER DES CHARGES

Vérifions si les demandes  
ont bien été remplies



## Exigences techniques minimales



1. Programme en langage C



2. Environnement de développement Linux



3. Programme découpé en plusieurs modules

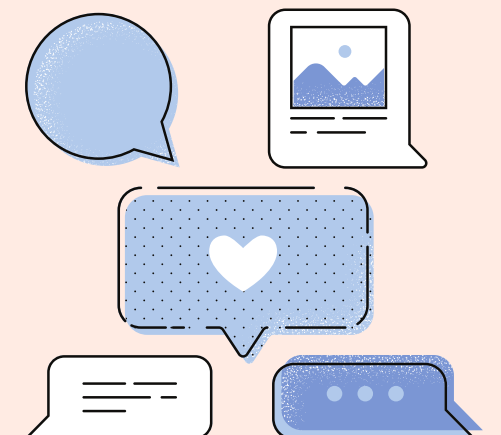


4. Utilisation de l'allocation dynamique lors des  
chargements de fichiers CRI.

## Exigences fonctionnelles

5. Indexer automatiquement un ensemble de fichier  
(majuscules, minuscules, singulier, pluriel)

6. Rechercher et classer par pertinence en fonction de la  
recherche les fichiers qui répondent le mieux aux critères.



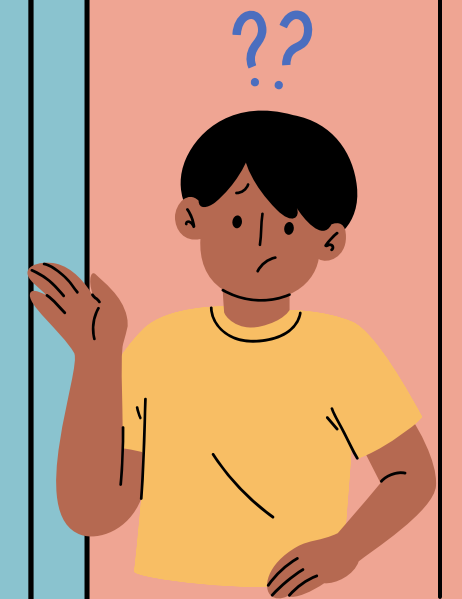
13

## Techniques

- création automatique des CRI
- allocation de la mémoire
- difficultés à utiliser Git

## Relationnelles

- tensions présentes au début du projet car mauvaise communication entre les élèves
- fatigue générale
  - manque de temps



## Techniques

- nous avons utilisés beaucoup de ressources internet afin de nous débloquent
- nous avons demandé de l'aide à des CIN pour se débarrasser d'une erreur
- explication de l'outil et accompagnement pour les élèves en difficulté

## Relationnelles

- nous avons fait une réunion pour exprimer nos inquiétudes et besoins
- nous avons mis en place des meetings journaliers pour exprimer notre progression des fonctions
- nous avons encore plus facilité les tâches en faisant de chaque tâche une étape du programme
- Nous avons mis une heure limite pour ne pas avoir une influence négative sur notre sommeil



15

Awesome Web Browser



<https://www.nosfonctionssontgéniales.fr>

## Exigences fonctionnelles avancées

7. Chargement automatique d'un ensemble de fichiers à partir d'un répertoire: texte initial.

8. En cas d'égalité calcul de la proximité des mots (cas des recherches ET). Attention il faut alors revenir sur le fichier

9. Recherche avec tolérance orthographique (proposition de correction comme sur le moteur google)

10. Classement avec affichage de la partie du texte contenant les mots recherchés

