UVHC
2013
2014

Resolution TP
Projet Tutoré

PARTICIPANTS :
GUEYE Léopold Aziz
SIDIBE Mouhamoudou

PROF ENCADREUR :
DELOT Thierry

GROUP GLASM |Valenciennes

SOMMAIRE

I) ENONCE DU PROBLEME

Le but est de réaliser un programme qui permet de lister le nombre de mots possibles (existent dans le dictionnaire français) a travers une grille fournie en entrée et un dictionnaire (mots français) pour faire les vérifications.

La grille est un fichier .txt et le dictionnaire est un fichier .dic .

Et qui parle de grille, parle forcement de parcours.

Donc a partir de ces informations fournies, nous allons essayer de faire le programme

II) ESPACE DE TRAVAIL

Logiciel de Codage : CodeBlocks

Langage : C

Fichiers : 3 fichiers .c « Grille.c, Dictionnaire.c, Recherche.c » c'est dans ces 3 fichiers qu'on a definit les diiferentes fonctions dont on aura besoin pour pouvoir exécuter le code.

3 fichiers .h « Grille.h, Dictionnaire.h, Recherche.h » pour faire appel aux fonctions créées dans les fichiers .c

1 fichier .c « main.c » c'est la ou on fait appel aux fonctions a éxécutées.

III) LISTE DES GRANDES TACHES A FAIRE

A FAIRE	PERIODE	TEMPS PASSE DSSUS
1 IMPORTATION DE LA GRILLE 2 LECTURE DE LA GRILLE 3 METTRE LES DONNEES LUES DANS UN TABLEAU A 2 DIMENSIONS {18*18}	Le 13 Septembre : Importation : **OK** Lecture : **OK** Affichage : **OK**	De 13H à 17H
4 IMPORTATION DU DICTIONNAIRE 5 FORMER DES MOTS AVEC LA GRILLE	Du 25 Septembre au 15 Octobre : Importation : **OK** Formation : **OK**	- Début import dictionnaire mais petit problème du au chargement dans un tableau. - Faisait des tests
6 VERIFIER SI MOT FORME EST DANS LE DICTIONNAIRE - SI 6 FAIT ALORS - CE QUI IMPLIQUE LA LECTURE DU DICTIONNAIRE - SI 6 EST VRAI ALORS	Du 18 Octobre au 10 Novembre : Avec des fonctions Vérification : **OK** Insertion **: OK**	20 Jours

- METTRE LE MOT DANS UN TABLEAU {a} - ET CREER UN COMPTEUR		
7 SI TOUTES LES VERIFICATIONS SONT TERMINEES	Le 20 Novembre : Vérification : **OK** Fonction Tri : **OK** Affichage **: OK** Test **: OK**	Fonction Tri : 20 mn
8 TRIE LE TABLEAU {a} PAR ORDRE CROISSANT {TAILLE, ALPHABETH}		
9 AFFICHER LE NOMBRE DE MOTS TROUVE		
10 AFFICHER LE TABLEAU {a}		
11 AFFICHER LE TEMPS		
PS : si mot trouvé existe déjà dans le tableau {a} alors ne pas l'enregistrer et ne pas incrementer le compteur de mots trouvés		

IV) PLANNINGS HEBDOMADAIRES

NOM: GUEYE Leopold Aziz && SIDIBE Mouhamadou MOIS: Septembre à Decembre ANNÉE: 2013 / 2014

	VEN : 06/09/13	SAM : 07/09/13	DIM : 08/09/13
SUJET	1er Cour de Projet Tutore : Prof : Thierry DELOT Sujet : presenter une grille avec des lettre (grille fournit en entrée) qui sera en relation avec un dictionnaire qui regroupera un ensemble de mots. But : Afficher l'ensemble des mots possible de la grille et qui existent dans le dictionnaire	Leopold Aziz GUEYE - reflexion sur le sujet donné - Annalyse du problem - Poser le plan - Detailler quelqules plans Heures : De 11h a 16h	Leopold Aziz GUEYE - Debut d'algorithme

	LUN : 09/09/13	MAR : 10/09/13	MER : 11/09/13	JEU : 11/09/13	VEN : 15/10/13
SUJET	- Discussion du sujet entre Leopold Aziz Gueye et Mouhamadou SIDIBE. - Choix binome	Leopold Aziz GUEYE et Mouhamadou SIDIBE - Bilan - Poser le probleme - Analyse du sujet - Choix du langage Heure : De 12h a 13h	Leopold Aziz GUEYE et Mouhamadou SIDIBE - Repartition des taches - Discussion Heure : De 12h a 13h	Liste taches GUEYE Leopold Aziz - Importation grille - Affichage grille - Formation de mots a partir de la grille - Verification - Liaison entre mots formés et le dico - Mots déjà trouvés SIDIBE Mouhamoudou - Importation dico - Compteur de mots trouvés - Trié tableau de mots - afficher	- Compte Rendu de chacun - Importation grille : OK - Affichage : OK - Formation : OK - Importation dico : OK

	VEN : 10/10/13	SAM : 20/10/13	DIM : 01/12/13	JEU : 10/12/13
SUJET	- Compte rendu - Verification : OK - Liaison : OK - Mots déjà trouvvés : OK	- Compte rendu - Compteur : OK - Tri : OK - Affichage : OK	- Debut de redaction de rapport	- Fin de redaction de rapport ! a 15h

V) LES FICHIERS

A) Les Fichiers .c

1) Grille.c

Ce fichier regroupe 2 fonctions, a savoir **ImportationGrille** et **AffichageDeLaGrille** car avec seulement ces 2 fonctions on pourra gérer la grille.

a) ImportationGrille fonction

Dans cette fonction le but est d'importer ce qui se trouve dans le fichier (**ex : grille.txt**) qui se trouve dans notre disque dure, afin de l'implémenter dans notre programme (ex : sous forme de tableau).
Pour ce faire, nous avions déclaré une variable **t1** de type **TabGrille** (tableau) et définit dans Grille.h et qui sera de dimension 2 pour pouvoir représenter la grille sous forme de tableau matricielle.
Ensuite nous avions indiqué le chemin du fichier .Txt avec la variable fichier de type FILE créé.
Nous avions ensuite fait une condition **while** qui a chaque fois vérifie si le fichier a implémenté ne revoit pas du NULL,

- si c'est pas NULL alors il stocke (par caractère car on utilise **fgetc**) ce qu'il a lu du fichier dans le tableau **t1 [i][j]** (i et j varient de 1 à 18, sachant que j est une boucle imbriquée dans i),
- sinon cela que le fichier est vide, et il n'y a plus de lecture donc la fin de la boucle.

Puis on fait ferme le fichier avec fclose (fichier).

b) AffichageDeLaGrille fonction

Dans cette fonction le but est d'afficher ce qu'on a stocké dans le tableau **t1 [i][j]**.
Pour ce faire, nous avions déclaré 2 boucles (**for)** imbriquées qui varient de 1 à 18, sachant que la première boucle gère la variation de **i** et la 2eme celle de **j.**
D'où avec ces 2 variables i et j, nous pouvions faire varier ce qui se trouve dans le tableau **t1 [i][j]** afin de l'afficher.

2) Dictionnaire.c

a) ImportationDicoDico_fonction

Ce fichier contient principalement une seule fonction, à savoir **ImportationDico** qui permet de gérer l'importation du dictionnaire.
Cette fonction prend en paramètre un seul argument ' le nom du fichier à importer et le nom du tableau ou l'implémenter.
Pour l'implémentation, c'est pratiquement la même procédure que celle utilisée dans l'implémentation de la grille, sauf qu'ici, ce qui a été lu dans le fichier (ex : francais.dic) est sauvegardé (par ligne car on utilise **fgets**) dans une variable **tableau de type tab** (tab définit dans le fichier Dictionnaire.h).

3) Recherche.c

C'est dans ce fichier qu'on note plus de fonctions, à savoir : **recherchDansDico**, **RechercheGaucheVersDroite**, **RechercheDroiteVersGauche**, **RechercheHautVersBas**, **RechercheBasVersHaut**, **RechercheDiagonalHautDeGaucheVersDroite**, **RechercheDiagonalHautDeDroiteVersGauche**, **RechercheDiagonalBasDeGaucheVersDroite**, **RechercheDiagonalBasDeDroiteVersGauche** et **Tri_Selection.**

a) RecherchDansDico_fonction

Cette fonction a pour principal objectif de vérifier si un mot qu'on lui donne en argument existe dans le dictionnaire, et en même temps elle vérifie si on n'a pas déjà vérifié ce mot.
Pour ce faire, elle prend en paramètre le mot a vérifié (**chaine**), le tableau ou est stocké le dictionnaire (**tableau**), un compteur de mot (**compt**) et enfin un tableau de type tab (**lesMotTrouve**) ou sera stocké les mots déjà trouvés afin de permettre le non parcours dans le dictionnaire.
Une fois ces paramètres fournis, la fonction vérifie d'abord le nombre de lettre dont compose le mot car on n'a fait une condition pour commencer la recherche à partir de tel nombre de lettres (ex : à partir de 4).
Une fois condition satisfaite, elle vérifie si le mot ne se trouve pas dans le tableau **lesMotTrouve** (qui regroupe l'ensemble des mots déjà parcourus et qui existent dans le dictionnaire grâce au compteur de mots **compt**).
Si il est présent, alors il y'a une variable (**MotInTab**) qui est initialisée a 0 qui va être égale à 1 pour dire qu'il est dans **lesMotTrouve.**
A la sortie de cette boucle, on n'a posé une condition if qui va tester la valeur de **MotInTab**, si elle est égale à 1 alors on la réinitialise

a 0 et on quitte la fonction **recherchDansDico ;** sinon on rechercher le mot au niveau du dictionnaire pour savoir s'il existe ou pas puis on quitte la fonction **recherchDansDico.**
S'il existe alors on l'ajoute dans le tableau **lesMotTrouve** et on incrémente le compteur de mots **compt.**

b) RechercheGaucheVersDroite fonction

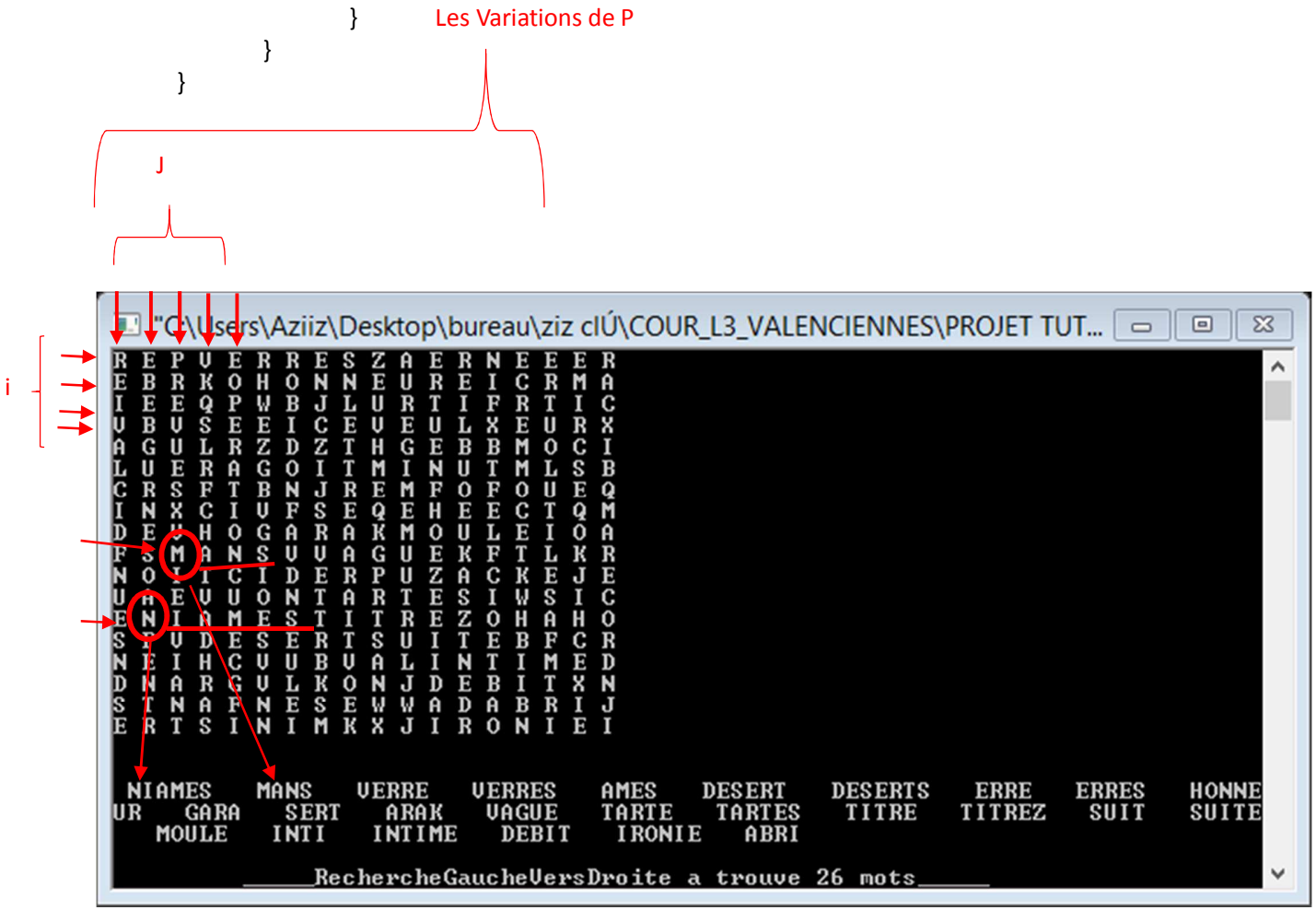
Cette fonction a pour rôle de former des mots à partir de la grille en combinant les lettres de la gauche vers la droite. Elle prend aussi 4 arguments à savoir : le tableau **t** ou est stocké les lettres importées du fichier de la grille pour pouvoir former des mots avec, le tableau **tableau** ou est stocké les mots importés du dictionnaire, le tableau **lesMotTrouve** ou est stocké les mots déjà parcourus et qui existe dans le dictionnaire et enfin un compteur de mots existant **compt.**
Son principe de fonctionnement est le suivant : nous avions déclaré 3 boucles **for** qui sont imbriquées entre elles

- dans la 1ere boucle qui varie de 1 à 17 pour permettre de passer d'une colonne à une autre avec la variable **P**
- dans la 2eme boucle, nous avons une variable **i** qui varie de 1 à 18 aussi, pour permettre de passer à la ligne suivante
- et enfin dans la 3eme boucle, nous avions une variable **j** qui varie en fonction de l'indice de **P** de la 1ere boucle jusqu'à 18 aussi, elle permet aussi de passer de colonnes en colonnes.

C'est dans cette dernière boucle que s'effectuent toutes les opérations concernant la **RechercheGaucheVersDroite**. Dans cette boucle nous avons une variable **chaine** qui à chaque fois concatène la lettre qui se trouve dans **t [i][j]** sachant que i et j varie en fonction de p.
puis on fait appel à la fonction **recherchDansDico** en lui fournissant les arguments.

Le principe :

```
for (p=1; p<17; p++) {
        nb=p;
        for(i=1; i<19; i++){
                k=0;
                for(j=nb; j<19; j++){
                chaine1[k]=t[i][j];
                recherchDansDico(chaine1, tableau, &compt[0], lesMotTrouve);
                k++;
```



Ici nous avions fait la recherche de gauche vers droite des mots de plus de 3 lettres.

c) RechercheDroiteVersGauche_fonction

Cette fonction a le même principe de de fonctionnement que celui de **RechercheGaucheVersDroite**, sauf qu'ici la formation de mots se fait de droite vers gauche. D'où

- la variable **P** varie de 18 à 2 (avec une décrémentation) (1ere boucle),
- celle de **i** de 1 à 18 (donc ne change pas) (2eme boucle),
- et enfin celle de **j** de l'indice de **P** à 2 (avec une décrémentation) (3eme boucle).

d) RechercheHautVersBas_fonction

Cette fonction a le même principe de de fonctionnement que celui de **RechercheGaucheVersDroite**, sauf qu'ici la formation de mots se fait du haut vers le bas. Et

- la 1ere boucle est **P** qui varie de 1 à 17 pour permettre de passer d'une ligne a une autre ligne
- la 2eme boucle est **j** qui varie de 1 à 18 pour permettre de passer d'une colonne à colonne
- et enfin la 3eme boucle est **i** qui varie en fonction de l'indice de **P** de la 1ere boucle jusqu'à 17 aussi, elle permet aussi de passer d'une ligne a une autre ligne.

e) RechercheBasVersHaut_fonction

Cette fonction a le même principe de fonctionnement que celui de **RechercheHautVersBas**, sauf qu'ici la formation de mots se fait de bas vers haut. D'où

- la variable **P** varie de 18 à 2 (avec une décrémentation) (1ere boucle),
- celle de **j** de 1 à 18 (donc ne change pas) (2eme boucle),
- et enfin celle de **i** de l'indice de **P** à 1 (avec une décrémentation) (3eme boucle).

f) RechercheDiagonalHautDeGaucheVersDroite fonction

Cette fonction a le même principe de de fonctionnement que celui de **RechercheGaucheVersDroite**, sauf qu'ici la formation de mots se fait diagonalement en partant du haut de gauche vers droite. D'où

- la variable **P** varie de 1 à 18 (1ere boucle),
- celle de **i** de 2 à 18 (2eme boucle),
- et enfin celle de **j** de l'indice de **P** à la variable **arretboucle** (qui au départ est initialisée a **p+2**, puis incrémenté de 1 à la sortie de la boucle **j**) (3eme boucle).
- la lecture de la grille se fera aussi grâce **t [g][j]** à l'aide de la variable **g**, qui au départ est initialisée à **i** au niveau de la boucle **i** et est incrémentée de 1 au niveau de la boucle **j.**

g) RechercheDiagonalBasDeGaucheVersDroite fonction

Cette fonction a le même principe de de fonctionnement que celui de **RechercheDiagonalHautDeGaucheVersDroite**, sauf qu'ici la formation de mots se fait diagonalement en partant du bas de gauche vers droite. D'où

- la variable **P** varie de 1 à 18 (1ere boucle),
- celle de **i** de 17 à 1 (2eme boucle),
- et enfin celle de **j** de l'indice de **P** à la variable **arretboucle** (qui au départ est initialisée a **p+2**, puis incrémenté de 1 à la sortie de la boucle **j**) (3eme boucle).
- même chose pour la lecture du tableau de la grille

h) RechercheDiagonalHautDeDroiteVersGauche fonction

Cette fonction a le même principe de de fonctionnement que celui de **RechercheDiagonalHautDeGaucheVersDroite**, sauf qu'ici la formation de mots se fait diagonalement en partant du bas de droite vers gauche. D'où

- la 1ere boucle est **P** qui varie de 1 à 17 pour permettre de passer d'une ligne a une autre ligne
- la 2eme boucle est **j** qui varie de 2 à 18 pour permettre de passer d'une colonne à colonne

- et enfin la 3eme boucle est **i** qui varie en fonction de l'indice de **P** à la variable **arretboucle** (qui au départ est initialisée a **p+2**, puis incrémenté de 1 à la sortie de la boucle **i**), elle permet aussi de passer d'une ligne a une autre ligne.
- la lecture de la grille se fera aussi grâce **t [i][g]** à l'aide de la variable **g**, qui au départ est initialisée à **j** au niveau de la boucle **j** et est incrémentée de 1 au niveau de la boucle **i.**

i) RechercheDiagonalBasDeDroiteVersGauche_fonction

Cette fonction a le même principe de de fonctionnement que celui de **RechercheDiagonalHautDeDroiteVersGauche**, sauf qu'ici la formation de mots se fait diagonalement en partant du bas de droite vers gauche. D'où

- la variable **P** varie de 18 à 1 (1ere boucle),
- celle de **j** de 2 à 18 (2eme boucle),
- et enfin celle de **i** de l'indice de **P** à la variable **arretboucle** (qui au départ est initialisée a **p+2**, puis décrémentée de 1 à la sortie de la boucle **i**) (3eme boucle).
- Même chose pour la lecture du tableau de la grille.

j) Tri_Selection_fonction

Cette fonction a pour rôle de triée par ordre décroissant (mots le plus long) les mots qui se trouvent dans le tableau **lesMotTrouve** (liste des mots trouvés et existant dans le dictionnaire). Elle prend comme argument le nom du tableau a trié et le nombre d'éléments à trier (ex : **compt)**. Pour ce faire, on crée 2 boucle for, et on pose une condition pour voir si le mot avec comme indice i et supérieure à celui avec comme indice j puis on fait les permutations nécessaires.

B) Les Fichiers .h

1) Grille.h

Dans ce fichier, nous avions définit un typedef **TabGrille** qui est de type tableau de char de dimension 2, et qui permet de déclarer les tableaux a 2 dimensions.
Il regroupe aussi l'appellation des 2 fonctions définies au niveau du fichier **Grille.c**, à savoir **ImportationGrille** et **AffichageDeLaGrille.**

2) Dictionnaire.h

Dans ce fichier, nous avions définit un typedef chaine qui est de type tableau de char de dimension 1, et qui permet de déclarer les tableaux a 1 dimensions. Nous avions aussi définie une structure ***tab** qui prend aussi comme argument le typedef chaine défini.
Ce fichier regroupe aussi l'appellation de la fonction définie au niveau du fichier **Dictionnaire.c**, à savoir **ImportationDico.**

3) Recherche.h

Dans ce fichier, nous avions définit un typedef **lesMotTrouve** qui est de type tableau de char de dimension 1, et qui permet de déclarer un tableau. Nous avions aussi importé les bibliothèques **#include "Grille.h"** et **#include "Dictionnaire.h"** afin de pouvoir accéder aux structures **TabGrille** et ***tab** qui sont respectivement définies dans ces 2 bibliothèques.
Ce fichier regroupe aussi l'appellation des fonctions définies au niveau du fichier **Recherche.c**, à savoir à savoir : **recherchDansDico**, **RechercheGaucheVersDroite**, **RechercheDroiteVersGauche**, **RechercheHautVersBas**, **RechercheBasVersHaut**, **RechercheDiagonalHautDeGaucheVersDroite**, **RechercheDiagonalHautDeDroiteVersGauche**, **RechercheDiagonalBasDeGaucheVersDroite**, **RechercheDiagonalBasDeDroiteVersGauche** et **Tri_Selection.**

C) Le Fichier main.c

C'est dans ce fichier que se passe tout le fonctionnement du programme.

Tout d'abord, nous avions importé la bibliothèque **recherche.h** qui regroupe en son sein les 2 autres bibliothèques.

Puis nous avions définis quelques variables, ensuite nous avions déclaré les 3 tableaux qui sont :

- le tableau pour l'importation de la grille (**TabGrille**) de type **TabeGrille**
- le tableau pour l'importation du dictionnaire (**TabDico**) de type **tab**
- et enfin le tableau pour stocker les mots déjà trouvés (**TablisteMotTrouve**) de type **tab** aussi

Ensuite suit l'appel des fonctions dans l'ordre suivant :

- ✓ Appel de la fonction **ImportationGrille** en lui indiquant le nom du fichier
- ✓ Appel de la fonction **AffichageDeLaGrille** pour l'afficher
- ✓ Appel de la fonction **ImportationDico** en lui indiquant le nom du fichier
- ✓ Appel de la fonction **RechercheGaucheVersDroite** pour les recherches
- ✓ Appel de la fonction **RechercheDroiteVersGauche**
- ✓ Appel de la fonction **RechercheHautVersBas**
- ✓ Appel de la fonction **RechercheBasVersHaut**
- ✓ Appel de la fonction **RechercheDiagonalHautDeGaucheVersDroite**
- ✓ Appel de la fonction **RechercheDiagonalHautDeDroiteVersGauche**
- ✓ Appel de la fonction **RechercheDiagonalBasDeGaucheVersDroite**
- ✓ Appel de la fonction **RechercheDiagonalBasDeDroiteVersGauche**
- ✓ Appel de la fonction **Tri_Selection** pour trier et afficher l'ensemble des mots trouvés

VI) CONCLUSION

La réalisation de ce projet nous a permis d'acquérir plus de connaissances en pratique. Le fait de le résoudre nous a permis de faire jouer notre imagination en se posant des questions , en faisant des tests , en discutant , en échangeant des idées entre binôme.
Donc en gros sa nous a permis d'avoir une petite idée sur le fait de travailler en groupe car en entreprise, c'est ce qui compte le plus.

On n'a noté aussi que l'implémentation sous tableau mettez trop de temps, c'est pourquoi on note une perte de temps durant l'exécution du programme

Ainsi, après avoir réalisé le programme, nous avions obtenu comme résultat :

En premier résultat

965 mots « avec des mots de 2 lettres et des doublons » en 4 mn

Puis avec des modifications

368 mots « avec les mots de 2 lettres mais sans doublons » en 3 mn

Mais comme le prof a dit que plus le mot est long (taille) , plus c'est bon
Du coup on n'a encore fait des modifications

296 mots « avec les mots de plus 2 lettres et sans doublons » en 2 mn

Et enfin

129 « avec les mots de plus 3 lettres et sans doublons » en 1mn