Cahier Des Specifications

Alabi Steve - Benyamna Younes - Capdenat Nicolas-
Chouipe Thibaut - El Harti Zakaria - Lienhardt Florian

Chef de projet : Benyamna Younes

Sous la direction de Mme Kloul

Outil automatique de décryptage

28 avril 2017

1 Introduction

Dans le cadre de notre troisième année de licence nous devons réaliser une application d'aide au décryptage par la méthode de Substitution ainsi que celle de Vigenère. Nous vous avons donc proposé l'application Dcrypt qui permet de décrypter mais également crypter un text ou même de faire une analyse fréquentielle sur un text. Pour cela, nous avons rédigé un cahier des charges afin de clarifier les objectifs et les exigences de ce projet. Dans la suite du cahier des charges, nous vous proposons donc le cahier des spécifications.

Pour développer notre projet et après étude des besoins les arguments en faveur du langage C sont les suivants :

-Tout d'abord, le C permet d'utiliser des expressions et des opérateurs qui sont très proches du langage machine, il est donc possible de développer des programmes efficients et rapides.

-La portabilité est aussi un argument de choix : En effet, en respectant le standard ANSI-C, il est possible d'utiliser le même programme sur tout autre système (autre hardware, autre système d'exploitation), simplement en le recompilant.

-De plus, pour assurer la maintenance du produit apres sa realisation et ce meme par des developpeurs qui ne sont pas ceux d'origine, il faut un langage des plus utilisés et maitrisés

-Enfin, selon l'IEEE(*), le langage C est depuis 2016 le meilleur langage de programmation(de par sa forte croissance et sa demande par les employeurs). C'est aussi le langage n=1 pour le développement d'applications d'entreprise, de bureau et d'applications scientifiques.

Nous avons également choisi une bibliothèque graphique du langage C : GTK+. Cette dernière nous permet de gérer efficacement la navigation entre les differentes fenêtres ainsi que nos accès memoires pour la sauvegarde et le chargement de nos fichiers. IEEE(*) : Institut des ingénieurs électriciens et électroniciens. C'est la plus grande organisation professionnelle technique du monde de l'évolution de la technologie

2 Organigramme

Interface graphique :
- bouton cryptage
- bouton decryptage
- bouton substitution
- bouton Vigenère
- affichage de text(complet et partiel)
- affichage pour la clé de substitution
- bouton Francais(decryptage)
- bouton Anglais(decryptage)
- affichage pour l'analyse fréquentielle
- charger un fichier text
- sauvegarder un fichier text
- créer un nouveau fichier text(resultats)
- Demander clef de Vigenère

Cryptage Substitution :
-créer une clé aleatoirement
-crypter le message

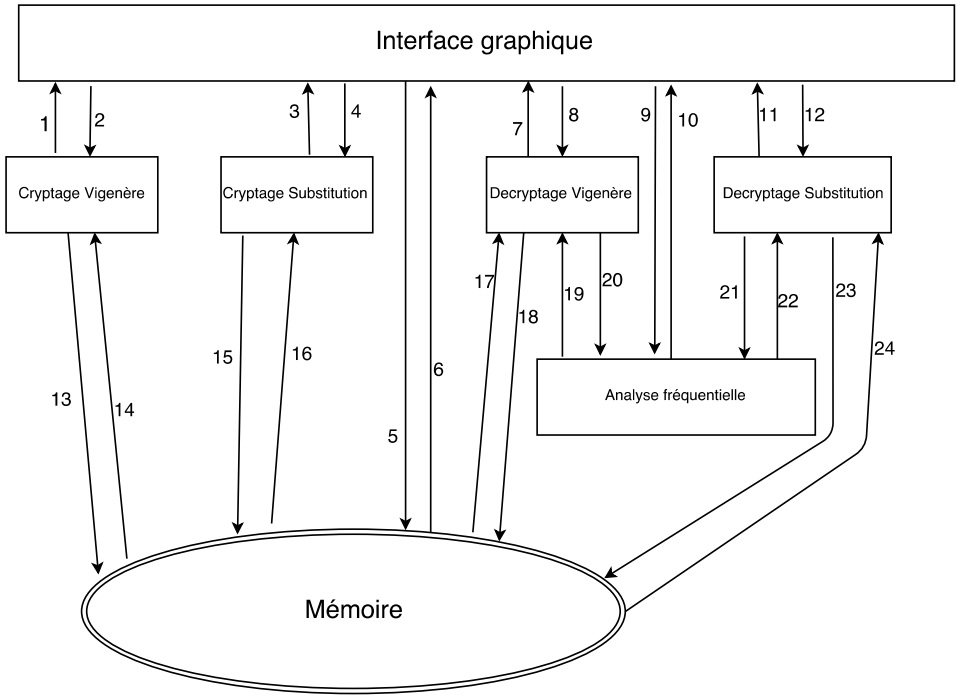
Cryptage Vigenère :
-crypter le message

Decryptage Substitution :
-decrypter le message

Analyse fréquentielle :
-analyse frequentielle sur text donné

Le diagramme résume les fonctions/procédures présentes dans chaque module ainsi que les liens d'inclusion.

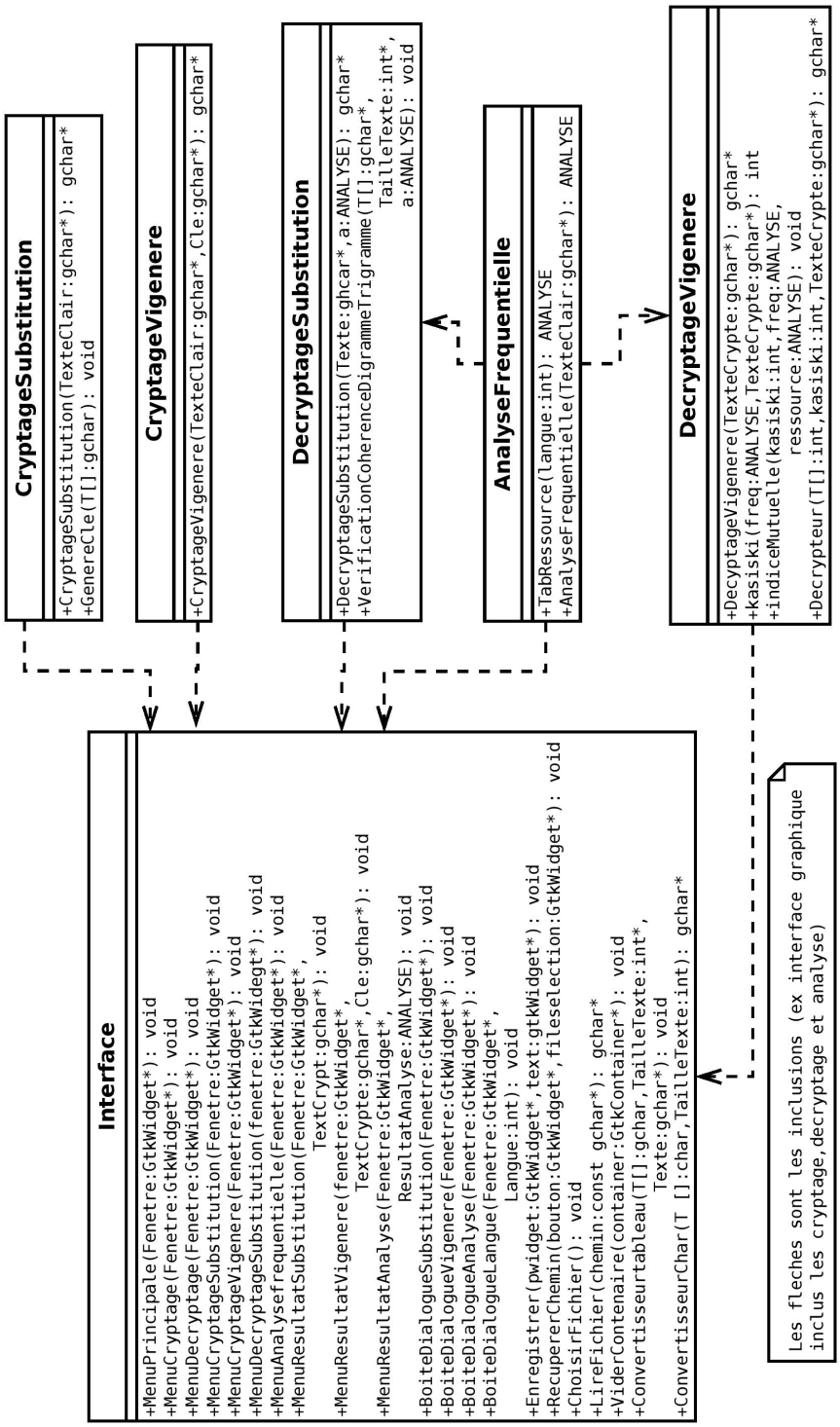
Decryptage Vigenère :
-decrypter le message



1) Une chaine de caractère contenant le texte crypté
2) Une chaine de caractère contenant le texte claire
3) Une chaine de caractère contenant le texte crypté
4) Une chaine de caractère contenant le texte claire
5) Un nom de fichier
6) Fichier texte : clair, crypter, décrypter ou analyse fréquentielle
7) Une chaine de caractère contenant le texte crypté
8) Une chaine de caractère contenant le texte claire
9) Une chaine de caractère contenant le texte à analyser
10) Une structure de données contenant l'analyse fréquentielle
11) Une chaine de caractère contenant le texte crypté
12) Une chaine de caractère contenant le texte claire
13) Un nom de fichier
14) Fichier texte clair
15) Un nom de fichier
16) Fichier texte clair
17) Un fichier texte crypté
18) Un nom de fichier
19) Une structure de données contenant l'analyse fréquentielle
20) Une chaine de caractère contenant le texte à analyser
21) Une chaine de caractère contenant le texte à analyser
22) Un ensemble de données contenant l'analyse fréquentielle
23) Un nom de fichier
24) Un fichier texte crypté

3 Signatures et Explications des methodes

3.1 Diagramme des fonctions des differents modules



3.2 Structures et nouveaux types

Le gchar* est une chaîne de caractères, on utilise pas char* car dans la bibliothèque GTK+ l'affichage d'un text se fait avec gchar*.

les widgets (GTKWidget) sont les boutons, les zones de text, les menus, enfin.. à peu près tout ce qui constitue une interface.

L'argument "gchar* TextClair" que l'on va retrouver par la suite dans plusieurs fonctions designe le texte clair(le texte a crypter).

la seule variable globale utilisée dans notre programme est Fenetre. Elle nous permet de conserver la fenêtre principale de notre application :
GtkWidget *Fenetre;

struct phoneme{
int frequence;
gchar* nom;
}PHONEME;
Cette structure représente les digrammes ou trigrammes et leurs fréquences.
le nom représente le nom du phoneme , la frequence est le nombre d'occurences
de ce dernier.

struct analyse{
int nb //nbre de lettre
float occ[25];
PHONEME di[];
PHONEME tr[];
gchar* pgor;
}ANALYSE;
Cette structure correspond aux caractéristiques d'un text. Elle sera remplie lors de l'appel de la fonction "AnalyseFrequentielle". Ainsi, on connaitra le nombre de caractères dans celui-ci, les digrammes/trigrammes, la plus grande occurence rencontrée et la frequence d'apparition de chaque lettre.
le nb représente le nombre de lettres dans le texte etudié, occ est un tableau qui contient
le nombre d'occurences de chaque lettre (occ[0] pour A etc..) ,di et tr contiennent les digrammes
et les trigrammes dans le texte ainsi que leurs nombre d'occurences.
pgor est la chaine de caracteres qui contiend la Plus Grande Occurance Repetée dans notre texte .

struct ressourceslangue{
float occ[25];
PHONEME di[];
PHONEME tr[];
}RESSOURCESLANGUE;
Cette structure correspond aux caractéristiques d'une langue. On connaitra les digrammes/trigrammes et la fréquence d'apparition de chaque lettre dans la langue choisie.
occ est un tableau qui contiend le nombre d'occurences de chaque lettre (occ[0] pour A etc..)
,di et tr contiennent les digrammes et les trigrammes dans le texte ainsi que leurs nombre d'occurences.

3.3 Signatures

3.3.1 Analyse Frequentielle

ANALYSE TabRessource(int langue);
La fonction prend en entrée un entier correspondant à la langue(francais ou anglais) et va ainsi remplir la structure "ANALYSE" et ses attributs suivants : la fréquence d'apparition de chaque lettre dans la langue

choisie ainsi que les digrammes et les trigrammes.

ANALYSE AnalyseFrequentielle(gchar* TextClair)
La fonction prend en entrée le text clair a analyser et va ainsi remplir la structure "ANALYSE" et ses attributs suivants : le nombre de lettres, la fréquence d'apparition de chaque lettre, les digrammes, les trigrammes et la plus grande occurence rencontrée. Ils seront logiquement remplis en fonction du texte.

3.3.2 Cryptage Substitution

gchar* CryptageSubstitution(gchar* TextClair) ;
Cette fonction permet de crypter un text avec la méthode de Substitution
Pour cela, elle prend en entrée une chaine de caractères qui correspond au text clair.
On utilise ensuite la fonction "ConvertisseurTableau". Le text a crypter se trouve alors dans un tableau que nous allons parcourir. Et pour chaque caractere nous allons lire dans un autre tableau [2][26] ou la premiere ligne du tableau([0][]) correspond a l'alphabet de base et la deuxième ligne ([1][]) celui de substitution(le nouveau).Que l'on aura generé à l'aide de la fonction "GenereCle". Ainsi, nous regardons la lettre dans le premier tableau (celui du texte a crypter), on bascule dans le deuxieme tableau et trouve ce caractere dans la première ligne, puis on regarde sa lettre de substitution (2e ligne), et enfin on remplace dans le premier tableau la lettre de base par ce caractère de substitution. En sortie, le text crypté sera sous forme de chaine de caractères grace a la fonction "ConvertisseurChar".
void GenereCle(gchar T[]) ;
génère une clé(alphabet) aléatoirement.

3.3.3 Decryptage Substitution

void DecryptageSubstitution(gchar* texteCrypté) Cette fonction prend en argument une chaine de caractère contenant le texte crypté. Elle vas commencer par appeler la fonction convertisseurtableau qui vas stocker les caractères du texte crypté dans un tableau. DecryptageSubstitution vas ensuite appeler les fonctions AnalyseFrequencielle et TabRessource. Aprés avoir appellé ces fonctions elle vas, dans une boucle, comparer les resultas de l'analyse frequentielle stocké dans une structure ANALYSE et obtenu par AnalyseFrequencielle aux données stocké dans une autre ANALYSE crée grace à la fonction TabRessource et contenant une analyse fréquentielle sur l'alphabet correspondant à la langue choisit(anglais ou francais). Elle vas alors remplacer dans le tableau une lettre du message crypter par la lettre qu'elle semble crypter grace à une conjecture basé sur les fréquences des lettres les plus utilsé et complété par l'analyse des digrammes et trigrammes, elle vas egalement crée une clef de substitution sous la forme d'un tableau de caractère de la taille de l'alphabet (la premiére case corespondra au caractère décryptant la lettre A, la seconde au caractère décryptant la lettre B ect...) . Cette fonction appellera ensuite la fonction VerificationCoerenceDigrammeTrigramme. La boucle prendra fin lorsque la clef de substitution sera compléte, c'est à dire quand tous les caractère de l'alphabet seron décrypté. Cette fonction ne renverra rien puisque son but vas étre de permettre l'affichage du décryptage et de la clef de décryptage pendant et à la fin de l'opperation. Cette affichage sera appellé dans la fonction VerificationCoerenceDigrammeTrigramme.

void VerificationCoerenceDigrammeTrigramme(gchar T[], int TailleTexte, gchar clef[], int TailleAlphabet, ANALYSE Ressource) Prend en argument un tableau T de la taille du texte crypté et contenant le texte crypté en cours de décryptage, une structure ANALYSE crée grace à la fonction contenant une analyse fréquentielle sur l'alphabet correspondant à la langue choisit(anglais ou francais) ainsi qu'un tableau clef de la taille de l'alphabet et correspondant à la clef de substitution. Cette fonction vas parcourir le texte en cours de décryptage pour vériffier que les letres trouvé ne forme pas de digramme ou trigramme inabituelle. Si c'est le cas elle proposera à l'utilisateur de corriger le décryptage du texte ou le corrigera d'elle même. Pour se faire elle appelera la fonction ConvertisseurChar afin de stocké le résultat du texte partiellement décrypté dans une chaine de caractère permettant son affichage grace à la fonction MenuResultatDecryptagePartiel la fonction appeller pourra également récupérer la clef de substitution final ainsi que le texte final afin de les enregistrer. Ainsi l'utilisation de cette fonction suivra deux cas de figures : soit la fonction ne détécte aucune anomalie est les tableaux en entrée ne sont pas modifiés, soit elle détécte des anomalies et les tableau sont alors modifiés avec l'aide de l'utilisateur.

3.3.4 Decryptage Vignere

gchar* DecryptageVigenere(gchar* textCrypte) ;
Cette fonction prend un texte crypté par le chiffrement de Vignère en argument. Appelle des fonctions AnalyseFrequentielle et TabRessource qui vont initialisé les Valeurs contenue dans la structure ANALYSE. Elle appel la fonction Kasiski qui va conjecturé la taille de la clé. Ensuite, on déclare un tableau cle[taillekasiski] de taille correspondant à la fonction kasiski. On appel la fonction indiceMutuelle qui remplira les valeurs des caractères de la clé dans le tableau cle[].
On retourne finalement la fonction Decrypteur qui retournera le texte en clair, décrypter.

int Kasiski(ANALYSE freq, gchar* texteCrypte) ;
Cette fonction prend une structure ANALYSE en argument afin de connaitre le PGOR (Plus Grande Occurence Rencontrée) et/ou les trigrammes,ainsi que le texteCrypte à décrypter. Elle cherche les différentes distances de PGOR et effectue le PGCD de ces distances. Elle renvoie ce résultat sous forme d'un entier (qui représente la taille du mot clé cherché).

void indiceMutuelle(int cle[],int kasiski, ANALYSE freq, ANALYSE ressource) ;
Cette fonction prend en argument le tableau clé qui contiendra les valeurs de notre mot clé, freq qui contient les fréquences de chacune des lettres du textes et ressource la probabilité d'apparition d'une lettre dans la langue choisie. Cette fonction calcule les indices de coïncidences mutuelles et les enregistrent dans un tableau tab[25][kasiski].
Elle parcourt et compare les 26 valeurs de chaque ligne afin de reperer celles qui sont proches de 0,065 (une par ligne).
Elle modifie le tableau cle[] passé en argument qui contiendra l'indice de chacunes des valeurs choisies dans un tableau à une dimension.

gchar* Decrypteur(int cle[], int kasiski, gchar* texteCrypte) ;
Cette fonction prend en argument le tableau cle ainsi que le texte a décrypter. Elle va soustraire la valeur du mot clef (tableau renvoyé par indiceMutuelle) au texte crypté.
Elle va repeter la suite de nombre composant le mot clef (contenu dans le tableau passé en argument) jusqu'a la fin du texte afin de toujours pouvoir soustraire une valeur du mot clé a une valeur du texte crypté.
Elle renvoie le texte en clair, c'est a dire décrypter.

3.3.5 Cryptage Vignere

gchar* CryptageVigenere(gchar* texteClair, gchar* Cle) ;
Cette fonction permet de crypter un texte avec la methode de Vigenere
Elle prend en argument le texte clair et la clé qui sont tout les deux des chaines de caractères.
On construit un tableau [3][taille texte]. En première ligne([0][]) on trouvera le texte a crypter(ou plutot la valeur en entier de chaque caractere du texte a crypter), en 2e ligne([1][]) la clé (qu'on repetera de facon a couvrir toute la 1e ligne et donc le message a chiffrer). La aussi chaque caractere de la clé sera directement inscrit en entier et enfin en 3e ligne([2][]) le resultat de l'addition des caracteres(entiers) des deux premieres lignes le tout modulo 26. On lira toute cette derniere ligne (l'entier de chaque case) et on ira remplir un second tableau avec directement le caractere correspondant a cet entier. La fonction "ConvertisseurChar" permettra de renvoyer le texte crypté sous forme de gchar*.

3.3.6 Interface Graphique

Dans ce module nous allons utiliser plusieurs menus pour afficher notre application et naviguer entre les menus pour cela nous allons créer plusieurs procédures qui vont afficher les menus

Les Menus

Toutes le procedure suivante ne renvoie rien et elle prennent en argument un conteneur(c'est l'endroit où sera affiché le contenu present dans ce menu)

void MenuPrincipal(GtkWidget *Fenetre) ;
Cette procédure permet d'afficher le menu principal.
Elle prend en argument la fenetre qui sera le contenaire(des boutons et labels).

void MenuCryptage(GtkWidget *Fenetre) ;
Cette procédure permet d'afficher le menu de cryptage.
Elle prend en argument la fenetre qui sera le contenaire(des boutons et labels).

void MenuDecryptage(GtkWidget *Fenetre) ;
Cette procédure permet d'afficher le menu de decryptage.
Elle prend en argument la fenetre qui sera le contenaire(des boutons et labels).

void MenuCryptageSubstitution(GtkWidget *Fenetre) ;
Cette procédure permet d'afficher le menu de cryptage par la méthode de substitution.
Elle prend en argument la fenetre qui sera le contenaire(des boutons et labels).

void MenuCryptageVigenere(GtkWidget *Fenetre) ;
Cette procédure permet d'afficher le menu de cryptage par la méthode de vigenere.
Elle prend en argument la fenetre qui sera le contenaire(des boutons et labels).

void MenuDecryptageSubstitution(GtkWidget *Fenetre) ;
Cette procédure permet d'afficher le menu de decryptage par la méthode de substitution.
Elle prend en argument la fenetre qui sera le contenaire(des boutons et labels).

void MenuDecryptageVigenere(GtkWidget *Fenetre) ;
Cette procédure permet d'afficher le menu de decryptage par la méthode de vigenere.
Elle prend en argument la fenetre qui sera le contenaire(des boutons et labels).

void MenuAnalyseFrequentielle(GtkWidget *Fenetre) ;
Cette procédure permet d'afficher le menu d'analyse frequentielle.
Elle prend en argument la fenetre qui sera le contenaire(des boutons et labels).

void MenuResultatSubstitution(GtkWidget *Fenetre, gchar* textecrypt) ;
Cette procédure permet d'afficher le menu du resultat du cryptage par la methode de substitution.
Elle prend en argument la fenetre qui sera le contenaire(des boutons et labels).

void MenuResultatVigenere(GtkWidget *Fenetre, gchar* textecrypt, gchar* cle) ;
Cette procédure permet d'afficher le menu du resultat du cryptage par la methode de vigenere.
Elle prend en argument la fenetre qui sera le contenaire(des boutons et labels).

void MenuResultatDecryptageSubstitution(GtkWidget *Fenetre, gchar* textecrypt) ;
Cette procédure permet d'afficher le menu du resultat du decryptage par la methode de substitution.
Elle prend en argument la fenetre qui sera le contenaire(des boutons et labels).

void MenuResultatDecryptageVigenere(GtkWidget *Fenetre, gchar* textecrypt, gchar* cle) ;
Cette procédure permet d'afficher le menu du resultat du decryptage par la methode de vigenere.
Elle prend en argument la fenetre qui sera le contenaire(des boutons et labels).

void MenuResultatAnalyse(GtkWidget *Fenetre) ;//rajouer le texte ou tableau
Cette procédure permet d'afficher le menu du resultat de l'analyse frequentielle.
Elle prend en argument la fenetre qui sera le contenaire(des boutons et labels).

void BoiteDialogueSubstitution(GtkWidget *Fenetre) ;
Cette procédure affiche une zone de texte qui nous permet de rentrer le texte à crypter par substitution.

Elle prend en argument la fenêtre.

void BoiteDialogueVigenere(GtkWidget *Fenetre) ;
Cette procédure affiche deux zones de texte qui permettent de rentrer le texte à crypter et la cle qui permet d'effectuer le chiffrement.
Elle prend en argument une fenêtre.

void BoiteDialogueDecryptageSubstitution(GtkWidget *Fenetre) ;
Cette procédure affiche une zone de texte qui nous permet de rentrer le texte à decrypter par substitution.
Elle prend en argument la fenêtre.

void BoiteDialogueDecryptageVigenere(GtkWidget *Fenetre) ;
Cette procédure affiche deux zones de texte qui permettent de rentrer le texte à decrypter par la methode de vigenere.
Elle prend en argument une fenêtre.

void BoiteDialogueAnalyse(GtkWidget *Fenetre) ;
Cette procédure affiche une zone de texte qui permet de rentrer le texte sur lequel effectuer une analyse fréquentielle.
Elle prend en argument une fenêtre.

void BoiteDialogueLangue(GtkWidget *Fenetre,int langue) ;
Cette procédure nous permet de choisir la langue (0 :francais , 1 :anglais).
Elle prend en argument une fenêtre.

nous aurons peut-être besoin d'autres menus au cours de notre application(pour un meilleur affichage ou ameliorations)

Les Enregistrement/Chargements

void Enregistrer (GtkWidget *pwidget, GtkWidget *texte) ;
Cette procédure permet de sauvegarder le texte obtenue.
Elle prend en argument une fenetre et le texte à sauvegarder.
Cette procédure ne renvoie rien.

void RecupererChemin(GtkWidget *bouton, GtkWidget *fileselection) ;
Cette procédure permet de recuperer le chemin d'accés d'un fichier.
Elle prend en argument une fenetre et une deuxieme fenetre (pour les fenetres de selections de fichier)
Cette procédure appel LireFichier,une fonction de Cryptage ou Decryptage et son menu de resultat qui va l'afficher.

void ChoisirFichier() ;
Cette procédure nous permet de selectionner un fichier à charger (fichier qui contient le texte clair ou crypter)

gchar* LireFichier(const gchar* chemin) ;
Cette fonction permet de recuperer le texte qui est dans un fichier, pour cela nous donnons en argument un chemin de fichier.

Autres

void ViderContenaire(GtkContainer * container) ;
Cette procédure nous permet de vider le contenu de la fenetre, cele nous permet de travailler avec une seule fenêtre.
Cette procédure prend en argument la fenêtre à vider.

void ConvertisseurTableau(gchar T[],int *Tailletexte,gchar* texte) ;
Cette procedure permet de transformer une chaine de caractère en tableau de caractère

gchar* ConvertisseurChar(char T[],int Tailletexte) ;
Parcours un tableau donné et le retourne en chaine de caracteres

4 Conclusion

Pour conclure, la rédaction de ce cahier des spécifications est une étape importante pour la réalisation du projet et plus précisément permet d'indiquer comment réaliser le besoin.

Elle se situe après l'étape de conception (cahier des charges) et avant l'étape de réalisation (codage).

Ce cahier permet donc de décrire de manière spécifique les différentes solutions apportées dans la cahier des charges, notamment les données en entrée ou en sortie des fonctions/procédures qui composent nos modules.

Maintenant que tout ces paramètres ont été définis et que l'équipe est d'accord sur l'architecture de l'application, nous allons passer à l'implémentation en respectant tout les règles fixées par le cahier des charges et des spécifications.