

# 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

2 mai 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 texte ou même de faire une analyse fréquentielle sur un texte. 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 efficaces 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 après sa réalisation et ce même par des développeurs qui ne sont pas ceux d'origine, il faut un langage des plus utilisés et maîtrisés

-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 différentes fenêtres ainsi que nos accès mémoire 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 texte(complet et partiel)
- affichage pour la clé de substitution
- bouton Français(decryptage)
- bouton Anglais(decryptage)
- affichage pour l'analyse fréquentielle
- charger un fichier texte
- sauvegarder un fichier texte
- créer un nouveau fichier texte(resultats)
- Demander clé de Vigenère

### Decryptage Vigenère :

- decrypter le message

### Cryptage Substitution :

- créer une clé aléatoirement
- crypter le message

### Cryptage Vigenère :

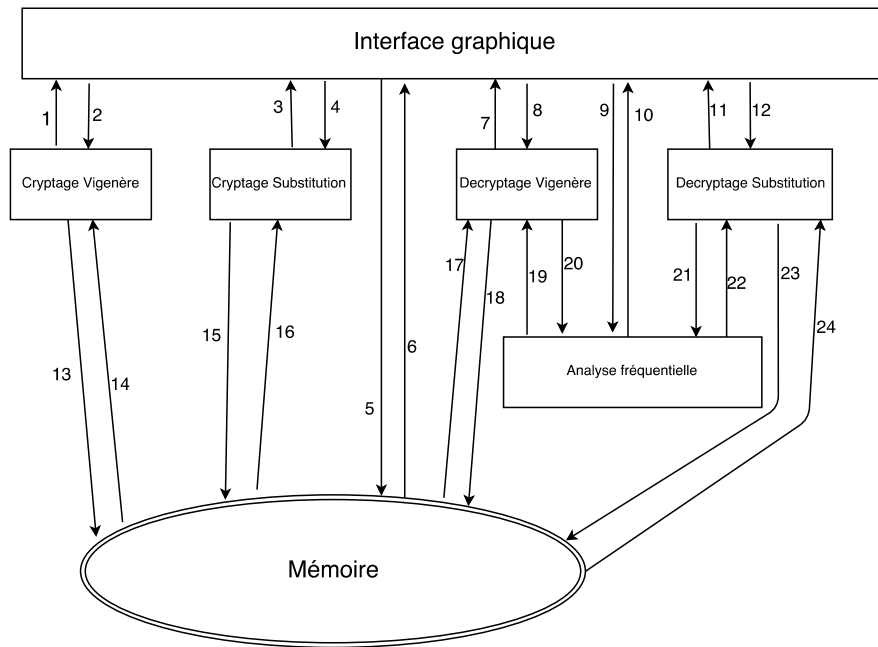
- crypter le message

### Decryptage Substitution :

- decrypter le message

### Analyse fréquentielle :

- analyse fréquentielle sur texte donné

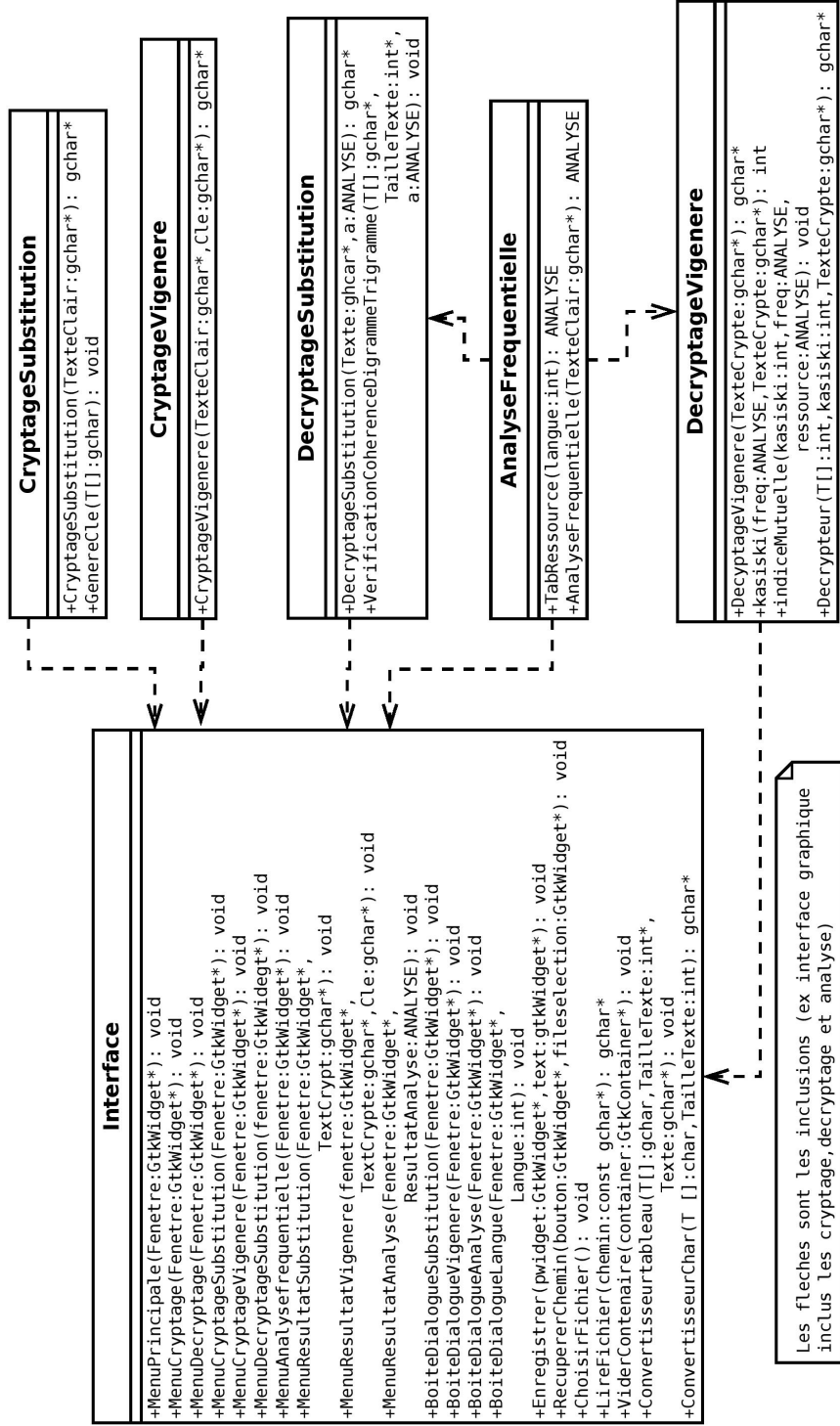


- 1) Une chaîne de caractère contenant le texte crypté
- 2) Une chaîne de caractère contenant le texte claire
- 3) Une chaîne de caractère contenant le texte crypté
- 4) Une chaîne de caractère contenant le texte claire
- 5) Un nom de fichier
- 6) Fichier texte : clair, crypter, décrypter ou analyse fréquentielle
- 7) Une chaîne de caractère contenant le texte crypté
- 8) Une chaîne de caractère contenant le texte claire
- 9) Une chaîne de caractère contenant le texte à analyser
- 10) Une structure de données contenant l'analyse fréquentielle
- 11) Une chaîne de caractère contenant le texte crypté
- 12) Une chaîne 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 chaîne de caractère contenant le texte à analyser
- 21) Une chaîne 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**

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



## 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* TexteClair"` que l’on va retrouver par la suite dans plusieurs fonctions designe le texte clair(le texte lisible).

Les variables globales utilisées dans notre programme sont Fenetre et Langue. Fenetre nous permet de conserver la fenêtre principale de notre application :

```
GtkWidget *Fenetre;
```

Langue permet, quant a elle, de garder la langue utilisée. `int Langue;`

```
typedef struct phoneme{
int frequence;
gchar* nom;
}PHONEME;
```

Cette structure représente les digrammes ou trigrammes et leurs fréquences.

"nom" représente le nom du phoneme et la frequence est le nombre d’occurences de ce dernier.

```
typedef struct analyse{
int nb;
float occ[25];
PHONEME di[20];
PHONEME tr[20];
gchar* pgor;
}ANALYSE;
```

Cette structure correspond aux caractéristiques d’un text. Elle sera remplie lors de l’appel de la fonction "AnalyseFrequentielle". Ainsi, on connaîtra 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 étudié, 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 nombres d’occurences.

pgor est la chaîne de caracteres qui contient la Plus Grande Occurance Rencontrée dans notre texte .

```
typedef struct ressourceslangue{
float occ[25];
PHONEME di[20];
PHONEME tr[20];
}RESSOURCESLANGUE;
```

Cette structure correspond aux caractéristiques d’une langue. On connaîtra les digrammes/trigrammes et la fréquence d’apparition de chaque lettre dans la langue choisie.

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.

## 3.3 Signatures

### 3.3.1 Analyse Frequentielle

```
RESSOURCESLANGUES TabRessource();
```

La fonction recupère un entier correspondant à la langue(francais ou anglais, en variable globale) et va ainsi

remplir la structure "RESSOURCESLANGUES" et ses attributs suivants : la fréquence d'apparition de chaque lettre dans la langue choisie ainsi que les digrammes et les trigrammes. Elle renvoie la structure RESSOURCESLANGUES remplie.

ANALYSE AnalyseFrequentielle(gchar\* TexteCrypté)

La fonction prend en entrée le text crypté à 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 occurrence rencontrée. Ils seront logiquement remplis en fonction du texte. Elle renvoie la structure ANALYSE remplie.

### 3.3.2 Cryptage Substitution

gchar\* CryptageSubstitution(gchar\* TexteClair);

Cette fonction permet de crypter un texte 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 à crypter se trouve alors dans un tableau que nous allons parcourir. Et pour chaque caractere nous allons lire dans un autre tableau [2][26] où la premiere ligne du tableau([0][ ]) correspond à 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 à 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 texte crypté sera sous forme de chaine de caractères grace à la fonction "ConvertisseurChar".

void GenereCle(gchar T[ ]);

Cette procédure permet de générer une clé(alphabet) aléatoirement. On la modifie à l'aide de pointeurs.

### 3.3.3 Decryptage Substitution

gchar\* DecryptageSubstitution( gchar\* texteCrypté ,gchar\*\* clef) Cette fonction prend en argument une chaine de caractères contenant le texte crypté et une chaine de caractères vide. Elle va commencer par appeler la fonction ConvertisseurTableau qui va stocker les caractères du texte crypté dans un tableau. Decryptage-Substitution va ensuite appeler les fonctions AnalyseFrequentielle et TabRessource. Après avoir appelé ces fonctions elle va, dans une boucle comparer les résultats de l'analyse frequentielle stockés dans une structure ANALYSE créée par AnalyseFrequentielle aux données stockées dans une structure RESSOURCESLANGUE créée grace à la fonction TabRessource et contenant une analyse fréquentielle sur l'alphabet correspondant à la langue choisie(anglais ou francais). Elle va alors remplacer dans le tableau une lettre du message crypté par la lettre qu'elle semble crypter grace à une conjecture basée sur les fréquences des lettres les plus utilisées et complétée par l'analyse des digrammes et trigrammes, elle va egalement créer une clef de substitution sous la forme d'un tableau de caractères de la taille de l'alphabet (la première case correspondra au caractère décryptant la lettre A, la seconde au caractère décryptant la lettre B ect...). Cette fonction récupérera la clef de substitution afin de l'enregistrer dans 'clef'. Ainsi l'utilisation de cette fonction suivra deux cas de figures : soit le resultat de la fonction correspond aux attentes de l'utilisateur sinon on rappelle la fonction qui tentera un decryptage different. La boucle prendra fin lorsque la clef de substitution sera partiellement complète, c'est à dire quand 10 caractères de l'alphabet seront décryptés. Cette fonction renverra un texte clair utilisé pour l'affichage.

### 3.3.4 Decryptage Vigenère

gchar\* DecryptageVigenere(gchar\* texteCrypte, gchar\*\* safecle);

Cette fonction prend en argument un texte crypté par le chiffrement de Vigenère ainsi que safecle (chaine de caractères) qui va permettre de sauvegarder la clé. Appels des fonctions AnalyseFrequentielle et TabRessource qui vont initialiser les valeurs contenue dans la structure ANALYSE ainsi que convertisseurtableau qui va stocker le texte crypté dans un tableau. Ensuite appelle la fonction Kasiski qui va conjecturer la taille de la clé. Puis, on déclare un tableau cle[taillekasiski] de taille correspondant à la fonction kasiski. On appelle la fonction indiceMutuelle qui remplira les valeurs des caractères de la clé dans le tableau cle[ ] ainsi que safecle qui contient les caractères mêmes.

On appelle la fonction Decrypteur, cette dernière nous servant à retourner le texte en clair.

```
int Kasiski(ANALYSE freq, gchar* texteCrypte);
```

Cette fonction prend une structure ANALYSE en argument afin de connaître le PGOR (Plus Grande Occurrence 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, gchar** safecle);
```

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 texte et ressource la probabilité d'apparition d'une lettre dans la langue choisie. Cette fonction calcule les indices de coïncidences mutuelles et les enregistre dans un tableau tab[25][kasiski].

Elle parcourt et compare les 26 valeurs de chaque ligne afin de repérer celles qui sont proches de 0,065 (une par ligne).

Elle modifie le tableau cle[] passé en argument qui contiendra l'indice de chacune des valeurs choisies dans un tableau à une dimension et affecte à safecle la clé afin de pouvoir l'afficher dans le menu.

```
gchar* Decrypteur( int cle[], int kasiski, gchar* texteCrypte);
```

Cette fonction prend en argument le tableau cle ainsi que le texte à décrypter. Elle va soustraire la valeur du mot clef (tableau renvoyé par indiceMutuelle) au texte crypté.

Elle va répéter la suite de nombre composant le mot clef (contenue dans le tableau passé en argument) jusqu'à la fin du texte afin de toujours pouvoir soustraire une valeur du mot clé à une valeur du texte crypté.

Elle renvoie le texte en clair, c'est à dire décrypté.

### 3.3.5 Cryptage Vigenère

```
gchar* CryptageVigenere(gchar* texteClair, gchar* Cle);
```

Cette fonction permet de crypter un texte avec la méthode de Vigenère

Elle prend en argument le texte clair et la clé qui sont tout les deux des chaînes de caractères.

On construit un tableau [3][taille texte]. En première ligne([0][]) on trouvera le texte à crypter(ou plutôt la valeur en entier de chaque caractère du texte à crypter), en 2e ligne([1][]) la clé (qu'on répètera de façon à couvrir toute la 1e ligne et donc le message à chiffrer). La aussi chaque caractère de la clé sera directement inscrit en entier et enfin en 3e ligne([2][]) le résultat de l'addition des caractères(entiers) des deux premières lignes le tout modulo 26. On lira toute cette dernière ligne (l'entier de chaque case) et on ira remplir un second tableau avec directement le caractère correspondant à 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 les procédures suivantes ne renvoient rien et elles prennent en argument un conteneur(c'est l'endroit où sera affiché le contenu présent dans ce menu)

```
void MenuPrincipal(GtkWidget *Fenetre);
```

Cette procédure permet d'afficher le menu principal.

Elle prend en argument la fenêtre qui sera le conteneur(des boutons et labels).

```
void MenuCryptage(GtkWidget *Fenetre);
```

Cette procédure permet d'afficher le menu de cryptage.

Elle prend en argument la fenêtre qui sera le conteneur(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 conteneur(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 conteneur(des boutons et labels).

```
void MenuCryptageVigenere(GtkWidget *Fenetre);
```

Cette procédure permet d'afficher le menu de cryptage par la méthode de vigenère.

Elle prend en argument la fenetre qui sera le conteneur(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 conteneur(des boutons et labels).

```
void MenuDecryptageVigenere(GtkWidget *Fenetre);
```

Cette procédure permet d'afficher le menu de decryptage par la méthode de vigenère.

Elle prend en argument la fenetre qui sera le conteneur(des boutons et labels).

```
void MenuAnalyseFrequentielle(GtkWidget *Fenetre);
```

Cette procédure permet d'afficher le menu d'analyse fréquentielle.

Elle prend en argument la fenetre qui sera le conteneur(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 conteneur(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 vigenère.

Elle prend en argument la fenetre qui sera le conteneur(des boutons et labels).

```
void MenuResultatDecryptagePartiel(GtkWidget *Fenetre)
```

Cette procédure permet d'afficher le menu du resultat du decryptage partiel par la methode de substitution.

Ce menu affiche un resultat partiel du decryptage avec des boutons qui permettent d'affiner notre resultat.

Une fois notre resultat obtenu nous passons au MenuResultatDecryptageSubstitution qui lui permet d'afficher le resultat final et la clé de substitution.

Elle prend en argument la fenetre qui sera le conteneur(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 conteneur(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 vigenère.

Elle prend en argument la fenetre qui sera le conteneur(des boutons et labels).

```
void MenuResultatAnalyse(GtkWidget *Fenetre); //rajouter le texte ou tableau
```

Cette procédure permet d'afficher le menu du resultat de l'analyse fréquentielle.

Elle prend en argument la fenetre qui sera le conteneur(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 :français , 1 :anglais),  
en modifiant la variable globale Langue à l'aide de la fonction ModifieLangue.

Elle prend en argument une fenêtre.

```
void modifieLangue(GtkWidget *Fenetre, int lang)
```

Cette fonction va mettre à jour la variable globale Langue.

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'accès 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, cela 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 procédure permet de transformer une chaîne de caractère en tableau de caractère

```
gchar* ConvertisseurChar(char T[],int TailleTexte);
```

Parcours un tableau donné et le retourne en chaîne de caractères

## 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 le cahier des charges, notamment les données en entrée ou en sortie des fonctions/procédures qui composent nos modules.

Maintenant que tous 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 toutes les règles fixées par le cahier des charges et des spécifications.