ESIEE Paris

IN3R12 - Programmation C

Projet Vigenere
Documentation développeur

Auteurs :
M. Kévin Ta
M. Antoine Mathis

Professeur :
M. Damien Masson

20 juin 2016

Table des matières

1 Organisation du programme

Le programme a été séparé en plusieurs fichiers afin de faciliter la compréhension du code source. Les fichiers *.c contiennent l'intégralité des fonctions utilisées dans le programme. Les fichiers *.h sont utilisés pour la déclaration des librairies, des constantes, des structures et des fonctions.

3 dossiers bin, doc et src ont été créé à la racine du projet. Le dossier bin contient les binaires générés par le compilateur. Le dossier doc contient la documentation utilisateur et programmeur sous format .PDF. Le dossier src contient les fichiers sources *.c et *.h.

Plus précisément, les fichiers code.c et decode.c représentent les appels aux fonctions afin de coder ou décoder un texte. Le fichier functions.c représente le coeur du programme avec toutes les fonctionnalités.

2 Workflow

Les étapes du programme ont été conçu de manière simple afin de ne pas perdre l'utilisateur. L'étape d'initialisation du programme comprend une vérification des paramètres. Les paramètres sont divisés en 2 parties : une partie option et une partie fichier. Si les options sont renseignées, l'alphabet ou la clé par défaut du programme sera écrasée.

Ensuite, le programme vérifie que les paramètres entrés sont corrects notamment en validant la présence des caractères qui composent la clé dans l'alphabet.

Viens ensuite le calcul et le codage ou décodage du message en fonction de l'alphabet.

Si des fichiers ont été renseignés dans les options, le code sera soit demandé par un prompt, soit tiré d'un fichier, et la sortie se fera soit sur le prompt, soit dans un fichier.

3 Fonctions du programme

3.1 functions.h

3.1.1 code

Cette fonction prend en paramètre 3 pointeurs de Char. Elle permet de coder un message passé en paramètre à l'aide d'une clé passée en paramètre à la longueur du message et d'un alphabet passé en paramètre. Dans la première partie de la fonction, le programme vérifie que le message ne possède pas de caractères spéciaux. Ensuite, les caractères du message sont additionnés avec ceux de la clé pour former une valeur qui correspond à la position d'une lettre dans l'alphabet. Si la valeur est plus grande que la longueur de l'alphabet, la valeur est soustraite de la taille de l'alphabet. Le caractère original du message est remplacé par la nouvelle lettre.

3.1.2 decode

Cette fonction prend en paramètre 3 pointeurs de Char. Elle permet de coder un message passé en paramètre à l'aide d'une clé passée en paramètre à la longueur du message et d'un alphabet passé en paramètre. Les caractères du message sont soustraits avec ceux de la

clé pour former une valeur qui correspond à la position d'une lettre dans l'alphabet. Si la valeur est plus petite que la longueur de l'alphabet, la valeur est additionnée avec la taille de l'alphabet. Le caractère original du message est remplacé par la nouvelle lettre.

3.1.3 getArguments

Cette fonction prend en paramètre une variable de type Integer et un pointeur de pointeur de Char et renvoie un type Arguments. Elle permet de récupérer les différents arguments de notre programme à l'aide de la librairie Getopt. Afin de faciliter le développement du programme, les arguments seront stockés dans une structure. Pour chaque option passée en paramètres, un case est utilisé pour tester les paramètres et exécuter la fonction correspondante. Un message, alphabet et clé sont alloués dans le tas par défaut et seront écrasés ou non en fonction des options.

3.1.4 getFileContent

Cette fonction prend en paramètre 2 pointeurs de Char. Elle permet de récupérer le contenu d'un fichier dont le nom est passé en paramètre et de le stocker dans une chaîne de caractères passée en paramètre.

3.1.5 getValue

Cette fonction prend en paramètre un pointeur de Char et un Char et renvoie un Integer. Elle permet de retourner la valeur d'un caractère passé en paramètre en fonction de sa position dans une chaîne de caractères passée en paramètre.

3.1.6 init

Cette fonction prend en paramètre une variable de type Arguments. Elle permet d'initier notre programme en exécutant les fonctions principales de vérification.

3.1.7 invalid

Cette fonction prend en paramètre 2 pointeurs de Char. Elle permet de vérifier par itération que les caractères d'une chaîne de caractères passée en paramètre sont présents dans une autre chaîne de caractères passée en paramètre.

3.1.8 output

Cette fonction prend en paramètre une variable de type Arguments. Elle permet de renvoyer le résultat de notre codage ou décodage dans un fichier texte ou de l'afficher dans le prompt en fonction des arguments.

3.1.9 removeChar

Cette fonction prend en paramètre un pointeur de Char et un Char et renvoie un pointeur de Char. Elle permet de supprimer un caractère passé en paramètre dans une chaîne de caractères passée en paramètre. La suppression est réalisée par la création par itération d'une chaîne de caractère qui ne contiendra pas le caractère voulue.

3.1.10 repeatkey

Cette fonction prend en paramètre un pointeur de Char et un Integer et renvoie un pointeur de Char. Elle permet de répéter le contenu d'une chaîne de caractères passée en paramètre selon une longueur passée en paramètre.

3.1.11 setFileContent

Cette fonction prend en paramètre 2 pointeurs de Char. Elle permet de récupérer le contenu d'une chaîne de caractères passée en paramètre et de le stocker dans un fichier dont le nom est passé en paramètre.

3.1.12 skip

Cette fonction prend en paramètre 2 pointeurs de Char et renvoie un pointeur de Char. Elle permet de supprimer les caractères d'une chaîne de caractères passée en paramètre qui ne sont pas présents dans une autre chaîne de caractères passée en paramètre. La suppression est réalisée par la création par itération d'une chaîne de caractère qui ne contiendra pas les caractères voulues (grâce à la fonction removeChar).

3.2 Les structures

Une structure a été créé afin de gérer les paramètres. Cette structure a été défini en tant que type Arguments. Elle contient les attributs char * cle, char * alphabet, int skip, char * message et char * sortie. Chaque paramètre correspondant à un argument.

4 Choix algorithmique

Le programme a été conçu de manière à séparer le maximum de fonctionnalités et de factoriser le code. En effet, un code factorisé sera moins complexe, plus facile à reprendre et à débugger. De manière à alléger les ressources du projet, il a été décidé que le programme sera codé de manière dynamique avec un minimum de copie de variables. De plus, afin de faciliter la conception, une structure a été mise en place à l'aide d'une librairie externe pour la gestion des arguments, dans le but d'éviter la répétition de code.

5 Ajout de fonctions

Si vous souhaitez ajouter des fonctions supplémentaires, vous devez dans un premier temps ajouter le prototype de votre fonction dans le fichier functions.h et placer le contenu de votre fonction dans le fichier functions.c. Il est également possible de faire appel à d'autres librairies. Dans ce cas-ci, il faut placer votre include dans le fichier functions.h. N'oubliez pas que grâce à l'implémentation des structures pour la gestion des arguments, il est très facile de récupérer ce que l'utilisateur a entré, et même de pouvoir ajouter de nouvelles options par la suite.