Crypto

C++

TOUROLLE Antoine

CRYPTO

TOUROLLE Antoine

I. Fichier de chiffrage :

Utilisation du programme :

Compilation du fichier

```
debian:~/public_html/crypto/code$ c++ -o crypto crypto.cc
```

Entrée des paramètres par l'utilisateur

```
debian:~/public_html/crypto$ ./decrypto
Saisir le nom du fichier à décrypter : cryptage.txt.ch
```

Chiffrement effectué

```
Chiffrement terminé !
```

Partie code :

Déclaration des classes

```
#include <iostream>
#include <fstream>
#include <string>
```

Récupération des paramètres tapés par l'utilisateur

```
// Demande de la clé de chiffrement
// La clé est maintenant une chaîne
char cle[20];
std::cout<<"Saisir la clé de chiffrement : ";
std::cin>>cle;

// demande du fichier à chiffrer
std::string nom_fic;
std::cout<<"Saisir le nom du fichier à crypter : ";
std::cin>>nom_fic;
```

Flux d'entrée/de sortie et ouverture des fichiers

```
// Ouverture des fichiers
std::fstream entree;
entree.open(nom_fic, std::fstream::in);
std::fstream sortie;
sortie.open(nom_fic+".ch", std::fstream::out);
```

Parcours du fichier d'entrée et écriture sur le fichier de sortie

```
// calcul de la taille de la clé
int taille = 0;
while(cle[taille++]);
taille--;

char c;
entree.get(c);
// définition d'un itérateur (au sens large du terme)
int i = 0;
while(!entree.eof())
   {
   sortie.put(c^cle[i++%taille]);
   // dès que i dépasse la taille on revient à 0
   // if(i==taille)
   //    {
   //    i=0;
   //    }
   entree.get(c);
   }
```

Fermeture des fichiers et du programme

```
// Fermeture des fichiers
entree.close();
sortie.close();

// Fin du programme
std::cout<<"Chiffrement terminé !\n";
}
```

II. Fichier de récupération de la clé :

Utilisation du programme :

Compilation du fichier

```
debian:~/public_html/crypto/code$ c++ -o decrypto decrypto.cc
```

Entrée du paramètre correspondant au nom de fichier

```
debian:~/public_html/crypto$ ./decrypto
Saisir le nom du fichier à décrypter : cryptage.txt.ch
```

Proposition retournée par le programme

```
La clef de chiffrement est certainement : sio
```

Dans le cas où l'utilisateur chiffre un fichier avec une clé dont la taille est différente de 3, le programme ne fonctionnera pas correctement :

Chiffrement du fichier avec le mot "toto"

```
debian:~/public_html/crypto/code$ ./crypto

Saisir la clé de chiffrement : toto
Saisir le nom du fichier à crypter : cryptage.txt
Chiffrement terminé !
```

Le programme renvoie une mauvaise clé

```
La clef de chiffrement est certainement : oto
```

Partie code :

Déclaration des classes

```
#include <iostream>
#include <fstream>
```

Récupération des paramètres tapés par l'utilisateur

```
// Demande du fichier à décrypter et on
// met son contenu dans une chaîne suffisament longue
// pour avoir un résultat significatif
char chaine[10000];
std::string nom_fic;
std::cout<<"Saisir le nom du fichier à décrypter : ";
std::cin>>nom_fic;
std::fstream entree;
entree.open(nom_fic, std::fstream::in);
```

```
// On change le type de la clé
char c;
int taille = 0;
entree.get(c);
while(!entree.eof())
   {
   chaine[taille++]=c;
   entree.get(c);
   }
entree.close();
int min = 26000;
// On supprime l'initialisation
char cle[4];
cle[3]=0; // fin de chaîne pour affichage
int x;
std::string cle_trouve;
```

```
// On change la boucle pour faire des mots de 3 lettres en force brute
for(int i=97; i<123; i++)
   {
   cle[0]=i;
   for(int j=97; j<123; j++)
      {
      cle[1]=j;
      for(int k=97; k<123; k++)
         {
         cle[2]=k;
         x = chiffrer(chaine, taille, cle);
         if(x < min)
            {
            min = x;
            cle_trouve = cle;
            }
         // DEBUG
         // std::cout<<cle<<"("<<x<<")\t";
         }
      //std::cout<<"\n";
      }
   }
std::cout<<"La clef de chiffrement est certainement : "<<cle_trouve<<"\n";
}
```

Fréquence d'apparition des lettres de l'alphabet

```
int chiffrer (char * clair, int taille, char * cle)
   {
   int val=0; // cotation du chiffrement effectué

   // fréquence de base d'apparitions des lettres de A à Z
   static int fr[26]={768,80,332,360,1776,106,110,64,723,19,0,589,272,761,534,324,134,681,823,730,605,127,0,54,21,7};
   static char s[10000];
   int i = 0;
```

Chiffrage avec la clé à tester

```
// On chiffre avec la clé à tester
for(int i=0; i<taille; i++)
   {
   s[i]=verif(clair[i]^cle[i%3]);
   }
```

Calcul du nombre d'apparations des caractères

```
// On calcule le nombre d'apparitions de chaque caractère
static int tex[256]; // nombre d'apparitions
// Initialisation du tableau
for(int i=0; i<256; i++)
   {
   tex[i]=0;
   }
// Calcul du nombre d'apparitions
for(int i=0; i<taille; i++)
   {
   tex[s[i]]++;
   }
```

Transformation du nombre d'apparitions en fréquence

```
// On transforme ce nombre d'apparitions en fréquence (*100)
// On ne s'occupe que des lettres de 'A' à 'Z'
// calcul du nombre de lettres
int nb_lettres=1;
for(int i=65; i<91; i++)
   {
   nb_lettres+=tex[i];
   }
```

Calcul des fréquences

```
// calcul des fréquences
// Initialisation du tableau
int fr_tex[26]; // fréquence d'apparitions * 100
for(int i=0; i<26; i++)
   {
   fr_tex[i]=0;
   }
for(int i=0; i<26; i++)
   {
   fr_tex[i]=(tex[i+65]*10000)/nb_lettres;
   val += abs(fr_tex[i] - fr[i]);
   }
return val;
}
```