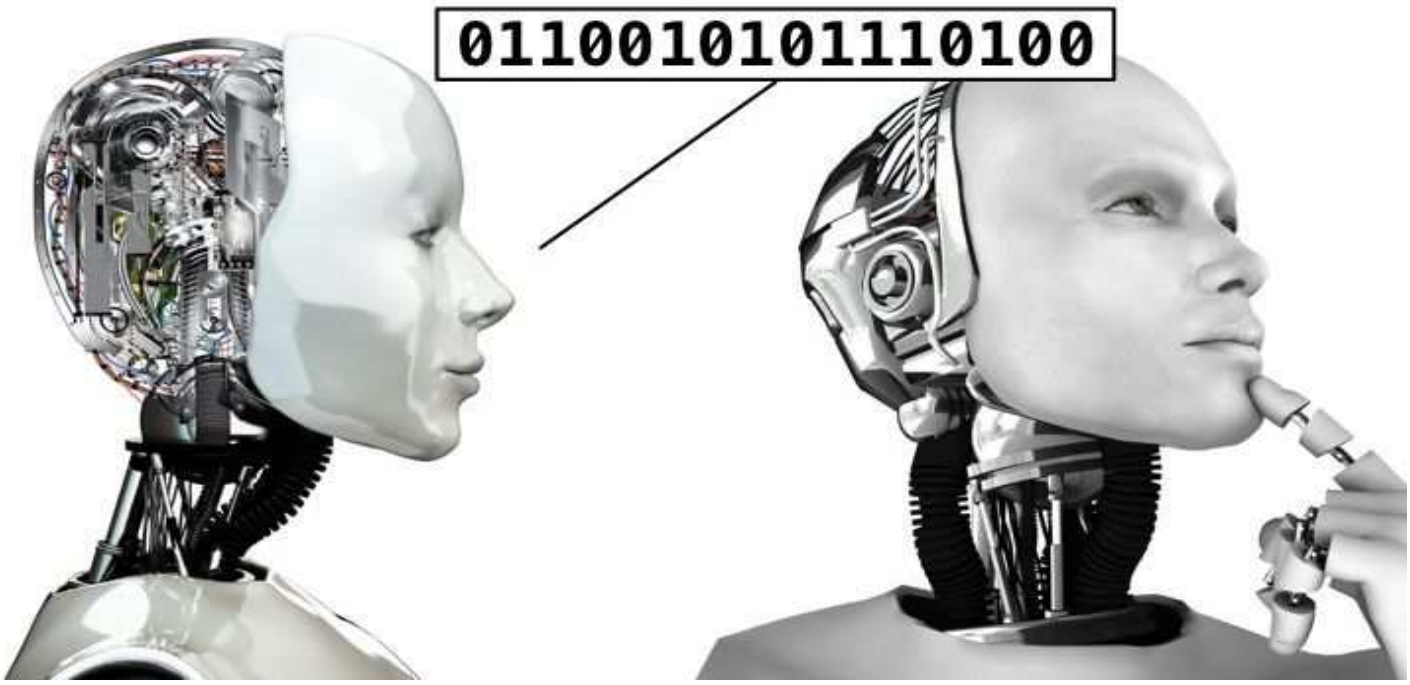
Niveau 4

TP
S4. Développement logiciel
S4.1. Principes de base : Gestion mémoire, allocation dynamique
Flux d'entrée et de sortie de base : fichiers
S4.3. Structure et gestion des données : Formats de fichier : texte, binaire, multimédia

Alice à Bob, Ô !

TABLE DES MATIÈRES



1 - PRÉSENTATION

Mademoiselle Moitrankil veut faire une blague à son meilleur ami, Bob. Elle lui transmet par courriel les paroles de sa chanson préférée, avec pour seul message "Bonjour". Mais Alice a trouvé plus amusant de chiffrer la pièce jointe !

Lorsque Monsieur Slègue reçoit le fichier, il est bien embêté... il ne comprend rien à ce qui se trouve dans le fichier...

2 - LE CHIFFREMENT D'ALICE

Le principe utilisé par Alice pour coder sa pièce jointe est relativement simple. Il repose sur un opérateur binaire réversible : le OU-EXCLUSIF !

En effet, lorsqu'une valeur (par exemple la lettre 'A') est combinée par un ou-exclusif avec une autre valeur appelée **clef** (mettons un 'B'), on obtient une troisième valeur (dans notre cas la lettre de code ASCII 0x03).

'A'	0100 0001	(0x41)
'B' (clef)	0100 0010	(0x42)
'A' XOR 'B' =	0000 0011	**0x03**

Maintenant, si nous reprenons cette valeur (0x03) et que nous lui appliquons la même opération avec la même valeur de **clef** 'B', nous allons retrouver le 'A' initial !

0x03	0000 0011		
'B' (clef)	0100 0010	(0x42)	
0x03 XOR 'B' =	0100 0001	**0x41**	**'A'**

De cette façon, le chiffrement et le déchiffrement partagent un même algorithme, qui consiste à combiner chaque lettre d'un texte avec une clef par un ou-exclusif.

Pour compliquer un peu le décryptage par quelqu'un de malveillant, il suffit de changer la clef à chaque caractère du texte initial.

Alice, est maline : elle a fournit la clef "Bonjour" à Bob !

La première lettre du texte est combinée avec le 'B', la seconde avec 'o', la troisième avec 'n'... la septième avec 'r'. Ici la clef est terminée ! L'opération continue alors en reprenant la clef à la première lettre : la huitième lettre du message est combinée avec le 'B', la neuvième avec 'o', etc. Par exemple :

```
'U'  'n'  ' '  't'  'o'  'u'  'r'  ' '  'd'  'e'  ' '  'c'…
'B'  'o'  'n'  'j'  'o'  'u'  'r'  'B'  'o'  'n'  'j'  'o'…
```

Le chiffrement est terminé lorsque toutes les lettres du message ont été codées, même si la clef n'est pas terminée !

Dans un nouveau projet bien structuré et **commenté**, de type application console, sous Qt...

2.1 - LECTURE DU FICHIER ORIGINAL

1) Dans le `main()`, créez une variable chaîne de caractères de type `QString`, et initialisez-là avec le nom du fichier fourni : `chanson.txt`.
Pour plus de facilité dans le traitement, mettez ce fichier à la racine de votre disque dur !

2) Créez les fichiers source et en-tête pour la gestion des `Fichiers`.

3) Ajoutez la déclaration et la définition d'une fonction `Get_nTaille()`, qui prend en paramètre le nom du fichier à ouvrir du type `QString`, et qui retourne la taille du fichier demandé selon le type `ifstream::pos_type`. Il vous faudra trouver les fichiers en-têtes spécifiques à inclure et comment utiliser la variable de type `QString`.
Le prototype à respecter est le suivant :
```
ifstream::pos_type     Get_nTaille(QString sNomFichier) ;
```

4) Testez cette fonction dans le `main()` : vous devez trouver une taille identique à celle indiquée par l'explorateur de Windows (soit `1988 octets`) !

5) Ajoutez les déclaration et définitions de la fonction `Lire()`, qui reçoit en paramètre le nom du fichier à lire (`QString`), qui retourne le contenu du fichier dans un tableau 1D créé dynamiquement (`char *`) ainsi que le nombre d'octets lus (`streamsize`) dans ce fichier. Le tableau dynamique est retourné directement tandis que le nombre d'octets lus fera l'objet d'un passage par référence.
Vous pouvez réutiliser les fonctions de création dynamiques de tableau 1D déjà étudiées dans un précédent TP.
Le prototype à respecter est le suivant :
```
char *  Lire(QString sNomFichier, streamsize & nNbreLu) ;
```

6) Complétez le `main()` de façon à pouvoir appeler correctement cette fonction. Affichez le contenu récupéré dans le fichier ainsi que le nombre d'octets réellement lus (il doit être identique à la taille du fichier).

2.2 - CHIFFREMENT

7) Ajoutez à votre projet les fichiers d'en-tête et application pour la gestion du `Chiffrement`.

8) Codez les déclaration et définition de la fonction `Chiffrer()`, qui reçoit en paramètre un tableau de caractères rempli à coder (`sOriginal`), le nombre d'octets de ce tableau (`nTaille`) et une chaîne de caractères représentant la clef à utiliser pour le chiffrement (`sClef`). Cette fonction va créer dynamiquement un autre tableau qui récupère le résultat du chiffrement (voir plus haut). Le tableau chiffré est alors retourné directement.
Le prototype à respecter est le suivant :
```
char *  Chiffrer(char * sOriginal, int nTaille, const char * sClef) ;
```

9) Adaptez le code du `main()` et testez votre fonction.

2.3 - SAUVEGARDE DU FICHIER CHIFFRÉ

10) Dans les fichiers du projet gérant les `Fichiers`, ajoutez les déclaration et définition de la fonction `Ecrire()` qui va permettre de sauvegarder dans un fichier (dont le nom `sNomFichier` est passé en paramètre, en type `QString`), le contenu d'un tableau d'octet (`sContenu` passé toujours en paramètre) et de taille connue (`nNbreAEcrire` de type `streamsize` et passé en paramètre aussi). Cette fonction retourne directement une valeur booléenne en fonction du résultat de l'écriture... On se concentrera alors sur les erreurs d'écritures... On testera le `rdstate()` du fichier avec le `ifstream::badbit` !
Pour une fois, Gogole sera votre ami !
Le prototype à respecter est le suivant :
```
bool    Ecrire(QString sNomFichier, char * sContenu, streamsize nNbreAEcrire) ;
```

11) Dans le `main()`, ajoutez le code nécessaire pour tester cette fonction. Le fichier chiffré devra avoir le même nom que le fichier original, mais avec une extension `.bin` ! Vous devrez trouver comment générer le nom de ce fichier automatiquement.
Vous obtenez un fichier correspondant sur votre disque dur !

12) Complétez le `main()` de façon à ce qu'il libère les tableaux qui ont été dynamiquement alloués dans les fonctions précédentes (`Lire()` et `Chiffrer()`)...

3 - LE DÉCHIFFRAGE DE BOB

13) Ajoutez, dans les fichiers de gestion du `Chiffrement`, une fonction `Dechiffrer()`, ayant le même prototype que la fonction `Chiffrer()`. Cette fonction devant réaliser le même traitement que `Chiffrer()`, il vous est imposé de la coder en seulement **trois (3) lignes** de code maximum (en comptant les déclarations de variables éventuelles et le retour !
Le prototype à respecter est le suivant :
```
char *  Dechiffrer(char * sOriginal, int nTaille, const char * sClef) ;
```

14) Dans le `main()`, ajoutez le code pour charger le fichier chiffré dans un nouveau tableau.

15) Déchiffrez le contenu de ce tableau en appelant la fonction `Dechiffrer()`.

16) Sauvegardez le résultat du déchiffrement dans un nouveau fichier qui portera le même nom que le fichier chiffré suivi des caractères `_2` puis de l'extension `.txt`, et comparez ce fichier avec le fichier original dans l'explorateur du système d'exploitation : la taille et le contenu doivent être identiques !

4 - BONUS

17) Si vous n'avez pas repris les fonctions du TP adéquate, déportez la création et libération des tableaux dynamiques dans des fichiers en-tête et application appropriés (`TableauDynamique`).

18) Dans un second projet (copie du projet initial), changez de méthode de cryptage, en proposant une nouvelle méthode (cryptage et décryptage) personnelle...

5 - PROTOTYPES UTILES

Pour les flux de fichiers `std::fstream`, `std::ifstream` et/ou `std::ofstream` :

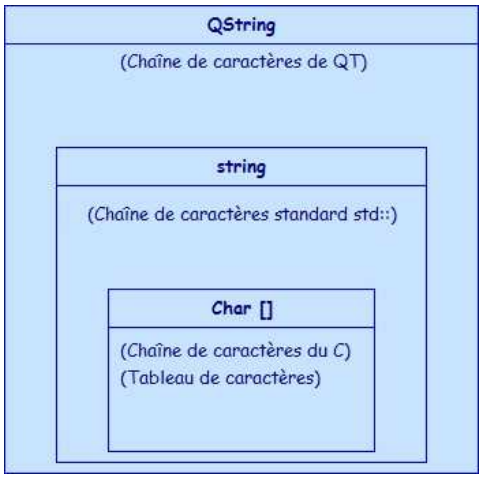
```
streamsize          std::fstream::gcount()  const ;
ifstream::pos_type  std::fstream::tellg() ;
```

Pour les chaînes de caractères standards `std::string` et de Qt `QString` :

```
            QString::QString(const char * str) ;

std::string QString::toStdString()  const ;
char *      std::string::c_str() const ;

int         QString::length()     const ;
```



Remarque :
Le flux standard de sortie écran (cout) ne peut afficher que des chaînes de caractères de type C (char *)