TD Synthèse

Le but de se TP est de couvrir tout ce qui a été vu pour l'instant en C++ : variables, fonctions, structures… L'ordre des questions est important puisque le problème va évoluer en difficulté en mettant en avant les avantages des évolutions.

Objectif du TP

Nous allons essayer aujourd'hui de créer un programme de cryptage. Nous allons commencer par demander à l'utilisateur une phrase que nous afficherons cryptée à l'écran, puis nous essayerons le problème inverse, à savoir on lui donne une phrase cryptée et le programme retrouve la phrase d'origine. Puis nous ferons la même chose sur des fichiers textes et enfin sur n'importe quel type de fichiers (exécutables, images…)

Première partie

Faites un programme qui demande à l'utilisateur de saisir une phrase (pour pouvoir permettre à l'utilisateur de saisir des espaces, il faut utiliser la commande **`getline`**). Sans faire de fonctions, faites les programmes suivants :

- Affichez la phrase à l'envers :
  - Exemple : « **`ma petite phrase`** » doit afficher « **`esarhp etitep am`** »
- Décalez chaque caractère de 2 lettres (**a** devient **c**, **b** devient **d**, **c** devient **e**,…, **y** devient **a** et **z** devient **b**.)
  - Exemple : « **`ma petite phrase`** » doit afficher « **`oc rgvkvg rjtcug`** »

Deuxième partie

Reprenez les deux programmes et utilisez-les pour faire les deux fonctions suivantes :

- string inversePhrase(string phrase) ;
  - Comme vous le voyez, elle a un paramètre de type **`string`** et elle renvoie un **`string`**. Il faut que la variable que vous revoyez soit transformée de la même manière que dans le premier exemple.
- string decalePhrase(string phrase, int nbCaract)
  - Le string qui est renvoyé doit être le 1er paramètre (la phrase) décalé d'autant de caractère que l'indique le 2ème paramètre (suivant le même principe que le second programme). Expliquez pourquoi les lignes suivantes (en commentaire) sont nécessaires (le % est l'opérateur modulo qui permet d'avoir toujours un nombre compris entre 0 et 25) :
  - Si le i^{ème} caractère est compris entre 'A' et 'Z'
    - sortie[i]=(((phrase[i]-65)+nbCaract)%26)+64
  - Si le i^{ème} caractère est compris entre 'a' et 'z'
    - sortie[i]=(((phrase[i]-97)+nbCaract)%26)+96

Troisième partie

Maintenant, on va créer une structure qui nous permet d'enregistrer la phrase d'origine, le code et la phrase cryptée. Nous l'appellerons Crypt. Le principe du cryptage est le suivant : on inverse la phrase puis on décale les caractères de X caractères (X étant le code secret…)

Faites les fonctions suivantes (en expliquant pourquoi il y a un &):

`void crypter(Crypt &donnees)`

Une fois la phrase cryptée, pensez à supprimer la phrase d'origine (mettez la égale à la chaine vide :"") pour éviter qu'il y ait de la triche…

`void decrypter(Crypt &donnees)`

Processus inverse… Essayez d'utiliser les fonctions **`inversePhrase`** et **`décalePhrase`**, il y a moyen de les utiliser tel quel sans modifications pour décrypter. A vous de bien réfléchir à comment utiliser les paramètres.

Test final !

Vous ferez ensuite un **`main`** qui demande à l'utilisateur de saisir une phrase puis un nombre (le code). Vous utiliserez ensuite la fonction **`crypter`**, et afficherez la phrase cryptée.

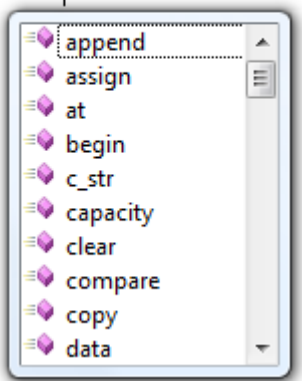
Ensuite, entrez la phrase suivante et décryptez la (sachant que le code est 12 !) :

! eqzgqV eqx ahmdN

Annexe : utilisation du type string

Le type string fournit beaucoup de fonctionnalités. Je vais essayer d'en expliquer les plus importantes. Dans tous les cas, Google est votre ami, ainsi que Visual Studio. En effet, il vous proposera toutes les fonctions possibles à utiliser (voir figure).

```
string phrase;
phrase.
```



Dans les exemples, nous supposerons qu'il y a une variable **phrase** de type **string** :

Connaitre la taille d'une chaine de caractère

Pour connaitre combien il y a de caractère dans une chaine, il suffit de faire :

```
int nbCaract=phrase.length();
```

ou :

```
cout<<phrase.length();
```

Accéder caractère par caractère

Les **string** peuvent s'utiliser de la même manière qu'un tableau. Puisqu'il est possible de connaitre le nombre de caractère, il est possible de connaitre la taille du tableau :

```
int tailleMax=phrase.length();
for(int cpt=0 ;cpt<tailleMax ;cpt++){
     cout<<phrase[cpt] ;
}
```

Chercher un mot

Il est possible de chercher un mot particulier dans la phrase :

```
int i= phrase.find_first_of("jpg");
```

Si **i** est négatif, c'est que **"jpg"** n'existe pas dans la phrase. Sinon, **i** correspond à la position de la première lettre (donc si on affiche **phrase[i]**, on affichera un **j**…)

Il est aussi possible de chercher après un certain nombre de caractère. Dans l'exemple suivant, on demande de chercher à partir du 5ème caractère :

```
int i= phrase.find_first_of("jpg",5);
```

Découper un mot

Il est aussi possible de découper une partie d'un mot. Imaginons qu'il y ait la phrase

« Bonjour Thibault ». Pour récupérer les deux mots, il faut faire comme ça :

```
string mot1= phrase.substr(0,7);//récupère Bonjour
string mot2= phrase.substr(8,8);//récupère Thibault
```

Le premier paramètre est la position où démarrer la copie et le deuxième permet de dire combien de caractères copier.