> **Ce cours est une mise à jour du cours C++ de OpenClassRoom pour le mettre à jour pour le C++11/14. Le cours d'origine est consultable sur la page suivante : Programmez avec le langage C++, par Mathieu Nebra et Matthieu Schaller. Ce cours est sous licence CC-BY-NC-SA.**

Utilité de ce chapitre si on mets des exos et TP dans chaque chapitre ?

TP : le mot mystère

Depuis le début de ce cours sur le C++, vous avez découvert de nombreuses notions : le compilateur, l'IDE, les variables, les fonctions, les conditions, les boucles... Vous avez pu voir des exemples d'utilisation de ces notions au fur et à mesure mais est-ce que vous avez pris le temps de créer un vrai programme pour vous entraîner ? Non ? Eh bien c'est le moment de s'y mettre !

On trouve régulièrement des TP au milieu des cours du Site du Zéro. Celui-ci ne fait pas exception. Le but ? Vous forcer à vous lancer « pour de vrai » dans la programmation !

Le sujet de ce TP n'est pas très compliqué mais promet d'être amusant : nous allons mélanger les lettres d'un mot et demander à un joueur de retrouver le mot « mystère » qui se cache derrière ces lettres (figure suivante).

Préparatifs et conseils

Le jeu que nous voulons réaliser consiste à retrouver un mot dont les lettres ont été mélangées. C'est simple en apparence mais il va nous falloir utiliser des notions que nous avons découvertes dans les chapitres précédents :

- les variables `string` ;
- les fonctions ;
- les structures de contrôle (boucles, conditions…).

N'hésitez pas à relire rapidement ces chapitres pour bien être dans le bain avant de commencer ce TP !

Principe du jeu « Le mot mystère »

Nous voulons réaliser un jeu qui se déroule de la façon suivante :

1. Le joueur 1 saisit un mot au clavier ;
2. L'ordinateur mélange les lettres du mot ;
3. Le joueur 2 essaie de deviner le mot d'origine à partir des lettres mélangées.

Voici un exemple de partie du jeu que nous allons réaliser :

```
Saisissez un mot
MYSTERE

Quel est ce mot ? MSERETY
RESEMTY
Ce n'est pas le mot !

Quel est ce mot ? MSERETY
MYRESTE
Ce n'est pas le mot !

Quel est ce mot ? MSERETY
MYSTERE
Bravo !
```

1. Dans cette partie, le joueur 1 choisit « MYSTERE » comme mot à deviner.
2. L'ordinateur mélange les lettres et demande au joueur 2 de retrouver le mot qui se cache derrière « MSERETY ».

3. Le joueur 2 essaie de trouver le mot. Ici, il y parvient au bout de 3 essais :
    1. RESEMTY : on lui dit que ce n'est pas cela
    2. MYRESTE : là non plus
    3. MYSTERE : là on lui dit bravo car il a trouvé, et le programme s'arrête.

Bien sûr, en l'état, le joueur 2 peut facilement lire le mot saisi par le joueur 1. Nous verrons à la fin du TP comment nous pouvons améliorer cela.

Quelques conseils pour bien démarrer

Quand on lâche un débutant dans la nature la première fois, avec comme seule instruction « Allez, code-moi cela », il est en général assez désemparé. « Par quoi dois-je commencer ? », « Qu'est-ce que je dois faire, qu'est-ce que je dois utiliser ? ». Bref, il ne sait pas du tout comment s'y prendre et c'est bien normal vu qu'il n'a jamais fait cela.

Mais moi, je n'ai pas envie que vous vous perdiez ! Je vais donc vous donner une série de conseils pour que vous soyez préparés au mieux. Bien entendu, ce sont juste des conseils, vous en faites ce que vous voulez.

Repérez les étapes du programme

Je vous ai décrit un peu plus tôt les 3 étapes du programme :

1. Saisie du mot à deviner ;
2. Mélange des lettres ;
3. Boucle qui se répète tant que le mot mystère n'a pas été trouvé.

Ces étapes sont en fait assez indépendantes. Plutôt que d'essayer de réaliser tout le programme d'un coup, pourquoi n'essayeriez-vous pas de

faire chaque étape indépendamment des autres ?

1. L'étape 1 est très simple : l'utilisateur doit saisir un mot qu'on va stocker en mémoire (dans une variable de type `string`, car c'est le type adapté). Si vous connaissez `cout` et `cin`, vous ne mettrez pas plus de quelques minutes à écrire le code correspondant.
2. L'étape 2 est la plus complexe : vous avez un `string` qui contient un mot comme « MYSTERE » et vous voulez aléatoirement mélanger les lettres pour obtenir quelque chose comme « MSERETY ». Comment faire ? Je vais vous aider un peu pour cela car vous devez utiliser certaines choses que nous n'avons pas vues.
3. L'étape 3 est de difficulté moyenne : vous devez créer une boucle qui demande de saisir un mot et qui le compare au mot mystère. La boucle s'arrête dès que le mot saisi est identique au mot mystère.

Créez un canevas de code avec les étapes

Comme vous le savez, tous les programmes contiennent une fonction `main()`. Écrivez dès maintenant des commentaires pour séparer les principales étapes du programme. Cela devrait donner quelque chose de comparable au squelette ci-dessous :

```
int main()
{
    //1 : On demande de saisir un mot

    //2 : On mélange les lettres du mot

    //3 : On demande à l'utilisateur quel est le mot mystère

    return 0;
}
```

À vous de réaliser les étapes ! Pour y aller en difficulté croissante, je vous conseille de faire d'abord l'étape 1, puis l'étape 3 et enfin l'étape 2.

Lorsque vous aurez réalisé les étapes 1 et 3, le programme vous demandera un mot et vous devrez le ressaisir. Ce ne sera pas très amusant mais, de cette manière, vous pourrez valider les premières étapes ! N'hésitez donc pas à y aller pas à pas !

Ci-dessous un aperçu du programme « intermédiaire » avec seulement les étapes 1 et 3 réalisées :

```
Saisissez un mot
MYSTERE

Quel est ce mot ?
RESEMTY
Ce n'est pas le mot !

Quel est ce mot ?
MYRESTE
Ce n'est pas le mot !

Quel est ce mot ?
MYSTERE
Bravo !
```

Comme vous le voyez, le programme ne propose pas encore le mot avec les lettres mélangées, mais si vous arrivez déjà à faire cela, vous avez fait 50 % du travail !

Un peu d'aide pour mélanger les lettres

L'étape de mélange des lettres est la plus « difficile » (si je puis dire !) de ce TP. Je vous donne quelques informations et conseils pour réaliser cette fameuse étape 2.

Tirer un nombre au hasard

Pour que les lettres soient aléatoirement mélangées, vous allez devoir tirer un nombre au hasard. Nous n'avons pas appris à le faire auparavant, il faut donc que je vous explique comment cela fonctionne.

- vous devez inclure `ctime` et `cstdlib` au début de votre code source pour obtenir les fonctionnalités de nombres aléatoires ;
- vous devez appeler la fonction `srand(time(0));` une seule fois au début de votre programme (au début du `main()`) pour initialiser la génération des nombres aléatoires ;
- enfin, pour générer un nombre compris entre 0 et 4 (par exemple), vous écrirez : `nbAleatoire = rand() % 5;` (Oui oui, on écrit « 5 » pour avoir un nombre compris entre 0 et 4.).

donne du code sans explications. Parler aussi des nouvelles fonctions de stl et de algorithm

Un exemple qui génère un nombre entre 0 et 4 :

main.cpp

```cpp
#include <iostream>
#include <ctime> // Obligatoire
#include <cstdlib> // Obligatoire

using namespace std;

int main()
{
    srand(time(0));

    int const nbAleatoire = rand() % 5;

    return 0;
}
```

Tirer une lettre au hasard

Tirer un nombre au hasard c'est bien mais, pour ce programme, j'ai besoin de tirer une lettre au hasard pour mélanger les lettres ! Imaginons que vous ayez un string appelé `motMystere` qui contient le mot « MYSTERE ». Vous avez appris que les string pouvaient être considérés comme des tableaux, souvenez-vous ! Ainsi, `motMystere.at(0)` correspond à la première lettre, `motMystere.at(1)` à la deuxième lettre, etc.

Il suffit de générer un nombre aléatoire entre 0 et le nombre de lettres du mot (qui nous est donné par `motMystere.size()`) pour tirer une lettre au hasard ! Une petite idée de code pour récupérer une lettre au hasard :

main.cpp

```cpp
#include <iostream>
#include <string>
#include <ctime>
#include <cstdlib>

using namespace std;

int main()
{
    string const motMystere { "MYSTERE" };

    srand(time(0));

    int const position = rand() % motMystere.size();

    cout << "Lettre au hasard :" << motMystere.at(position);

    return 0;
}
```

Retirer une lettre d'un string

Pour éviter de tirer 2 fois la même lettre d'un mot, je vous conseille de retirer au fur et à mesure les lettres qui ont été piochées. Pour ce faire, on va faire appel à `erase()` sur le mot mystère, comme ceci :

```
motMystere.erase(4, 1);  // Retire la lettre n°5
```

Il y a 2 paramètres :

- le numéro de la lettre à retirer du mot (ici 4, ce qui correspond à la 5ème lettre car on commence à compter à partir de 0) ;
- le nombre de lettres à retirer (ici 1).

Créez des fonctions !

Ce n'est pas une obligation mais, plutôt que de tout mettre dans le `main()`, vous pourriez créer des fonctions qui ont des rôles spécifiques. Par exemple, l'étape 2 qui génère un mot dont les lettres ont été mélangées mériterait d'être fournie sous forme de fonction.

Ainsi, on pourrait appeler la fonction comme ceci dans le `main()` :

```
motMelange = melangerLettres(motMystere);
```

On lui envoie le `motMystere`, elle nous renvoie un `motMelange`.

Bien entendu, toute la difficulté consiste ensuite à coder cette fonction `melangerLettres`. Allez, au boulot !

Correction

C'est l'heure de la correction !

Vous avez sûrement passé du temps à réfléchir à ce programme. Cela n'a peut-être pas toujours été facile et vous n'avez pas forcément su tout

faire. Ce n'est pas grave ! Ce qui compte, c'est d'avoir essayé : c'est comme cela que vous progressez le plus !

Normalement, les étapes 1 et 3 étaient assez faciles pour tout le monde. Seule l'étape 2 (mélange des lettres) demandait plus de réflexion : je l'ai isolée dans une fonction `melangerLettres` comme je vous l'ai suggéré plus tôt.

Le code

Sans plus attendre, voici la correction :

main.cpp

```cpp
#include <iostream>
#include <string>
#include <ctime>
#include <cstdlib>

using namespace std;

string melangerLettres(string mot)
{
    string melange {};

    //Tant qu'on n'a pas extrait toutes les lettres du mot
    while (mot.size() != 0)
    {
        // On choisit un numéro de lettre au hasard dans le
mot
        int const position = rand() % mot.size();

        // On ajoute la lettre dans le mot mélangé
        melange += mot.at(position);

        // On retire cette lettre du mot mystère
        // Pour ne pas la prendre une deuxième fois
        mot.erase(position, 1);
    }
```

```
    //On renvoie le mot mélangé
    return melange;
}

int main()
{
    //Initialisation des nombres aléatoires
    srand(time(0));

    //1 : On demande de saisir un mot
    string motMystere {};
    cout << "Saisissez un mot" << endl;
    cin >> motMystere;

    //2 : On récupère le mot avec les lettres mélangées dans
motMelange
    string const motMelange = melangerLettres(motMystere);

    //3 : On demande à l'utilisateur quel est le mot mystère
    string motUtilisateur {};
    do
    {
        cout << endl << "Quel est ce mot ? " << motMelange
<< endl;
        cin >> motUtilisateur;

        if (motUtilisateur == motMystere)
        {
            cout << "Bravo !" << endl;
        }
        else
        {
            cout << "Ce n'est pas le mot !" << endl;
        }
    } while (motUtilisateur != motMystere);
    // On recommence tant qu'il n'a pas trouvé

    return 0;
}
```

Ne vous laissez pas surprendre par la « taille » du code (qui n'est d'ailleurs pas très gros) et soyez méthodiques en le lisant : commencez

par lire le `main()` et non la fonction `melangerLettres()`. Regardez les différentes étapes du programme une par une : isolées, elles sont plus simples à comprendre.

Des explications

Voici quelques explications pour mieux comprendre le programme, étape par étape.

Étape 1 : saisir un mot

C'était, de loin, l'étape la plus simple : un `cout` pour afficher un message, un `cin` pour récupérer un mot que l'on stocke dans la variable `motMystere`. Facile !

Étape 2 : mélanger les lettres

Plus difficile, cette étape est réalisée en fait dans une fonction `melangerLettres` (en haut du programme). Le `main()` appelle la fonction `melangerLettres()` en lui envoyant le mot mystère. Le rôle de la fonction est de renvoyer une version mélangée des lettres, que l'on stocke dans `motMelange`.

Analysons la fonction `melangerLettres`. Elle extrait une à une, aléatoirement, les lettres du mot et recommence tant qu'il reste des lettres à extraire dans le mot :

```
while (mot.size() != 0)
{
    position = rand() % mot.size();
    melange += mot.at(position);
    mot.erase(position, 1);
}
```

À chaque passage de boucle, on tire un nombre au hasard compris entre 0 et le nombre de lettres qu'il reste dans le mot. On ajoute ces lettres piochées aléatoirement dans un `string melange` et on retire les lettres du mot d'origine pour ne pas les piocher une deuxième fois.

Une fois toutes les lettres extraites, on sort de la boucle et on renvoie la variable `melange` qui contient les lettres dans le désordre.

Étape 3 : demander à l'utilisateur le mot mystère

Cette étape prend la forme d'une boucle do ... while, qui nous permet de nous assurer que nous demandons bien au moins une fois quel est le mot mystère.

L'utilisateur saisit un mot grâce à cin et on compare ce mot avec le motMystere qu'il faut trouver. On continue la boucle tant que le mot n'a pas été trouvé, d'où la condition :

```
while (motUtilisateur != motMystere); // On recommence tant qu'il n'a pas trouvé
```

On affiche un message différent selon qu'on a trouvé ou non le mot mystère. Le programme s'arrête dès qu'on sort de la boucle, donc dès qu'on a trouvé le mot mystère.

Vous pouvez ainsi tester le programme et éventuellement vous en servir comme base par la suite pour réaliser les améliorations que je vais vous proposer (si vous n'avez pas réussi à faire le programme vous-mêmes, bien entendu !).

Aller plus loin

Notre programme est terminé... mais on peut toujours l'améliorer. Je vais vous présenter une série de suggestions pour aller plus loin, ce qui vous donnera l'occasion de travailler un peu plus sur ce petit jeu.

Ces propositions sont de difficulté croissante :

- Ajoutez des sauts de ligne au début : lorsque le premier joueur saisit le mot mystère la première fois, vous devriez créer plusieurs sauts de lignes pour que le joueur 2 ne voie pas le mot qui a été saisi, sinon c'est trop facile pour lui. Utilisez plusieurs `endl`, par exemple, pour créer plusieurs retours à la ligne.
- Proposez au joueur de faire une nouvelle partie. Actuellement, une fois le mot trouvé, le programme s'arrête. Et si vous demandiez « Voulez-vous faire une autre partie ? (o/n) ». En fonction de la réponse saisie, vous reprenez au début du programme. Pour ce faire, il faudra créer une grosse boucle do ... while qui englobe les 3 étapes du programme.
- Fixez un nombre maximal de coups pour trouver le mot mystère. Vous pouvez par exemple indiquer « Il vous reste 5 essais » et lorsque les 5 essais sont écoulés, le programme s'arrête en affichant la solution.
- Calculez le score moyen du joueur à la fin du programme : après plusieurs parties du joueur, affichez-lui son score, qui sera la moyenne des parties précédentes. Vous pouvez calculer le nombre de points comme vous le voulez. Vous devrez sûrement utiliser les tableaux dynamiques `vector` pour stocker les scores de chaque partie au fur et à mesure, avant d'en faire la moyenne.
- Piochez le mot dans un fichier-dictionnaire : pour pouvoir jouer seul, vous pourriez créer un fichier contenant une série de mots (un par ligne) dans lequel le programme va piocher aléatoirement à chaque fois. Voici un exemple de fichier-dictionnaire :

```
MYSTERE
XYLOPHONE
ABEILLE
PLUTON
MAGIQUE
AVERTISSEMENT
```

Au lieu de demander le mot à deviner (étape 1) on va chercher dans un fichier comme celui-ci un mot aléatoire. À vous d'utiliser les fonctionnalités de lecture de fichiers que vous avez apprises ! Allez, puisque vous m'êtes sympathiques, je vous propose même de télécharger un fichier-dictionnaire tout prêt avec des dizaines de milliers de mots ! Merci qui ?! Télécharger le fichier

Si vous avez d'autres idées, n'hésitez pas à compléter encore ce programme ! Cela vous fera beaucoup progresser, vous verrez.

Chapitre précédent	Sommaire principal	Chapitre suivant

Cours, C++