Codage Morse TP SN1

On se propose de réaliser une librairie de fonctions permettant de générer le code Morse équivalent à une chaîne de caractères et de retrouver le texte originel à partir d'un code morse.

En toute fin de TP, un code secret est proposé : essayez de retrouver cette phrase célèbre !

Pour cela, au moins deux fonctions sont nécessaires :

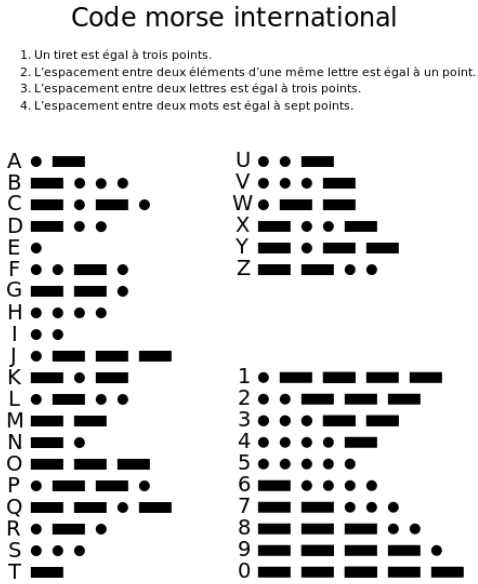
```
//Fonction renvoyant le code morse correspondant a un texte passe en
argument
string textToMorse(string);

//Fonction renvoyant le texte correspondant au code morse passe en argument
string morseToText(string);
```

Une troisième fonction peut s'avérer utile : c'est une fonction de recherche utilisée par les deux fonctions précédentes :

```
//Fonction recherchant dans un code une lettre. Elle renvoie la lettre
trouvee
string lookforcode(string);
```

Code Morse



Idée : On sauvegarde dans un tableau de *string* les équivalents morse de chaque lettre de l'alphabet :

```
//tableau des lettres
const string TMorse[]={".-","-...","-.-.","-..",".","..-.","--
.","....","..",".---",
                       "-.-",".-..","--","-.","---",".--.","--.-",".-
.","...","-","..-",
                       "...-",".--","-..-","-.--","--..", "/", ".----."};

//Tableau des chiffres
const string TMorseC[]={"-----",".----","..---","...--","....-",".....","-
....","--...","---..","----."};
```

Les tableaux TMorse[] et TMorseC[] sont des tableaux constant de *string*

Si on accède à TMorse[0] on obtient l'équivalent morse de la lettre 'A'

Si on accède à TMorse[1] on obtient l'équivalent morse de la lettre 'B'

Etc..

1. On va s'intéresser à la fonction *string textToMorse(string text)* :

Cette fonction prend en paramètre un objet de la classe *string* et renvoie l'équivalent morse.

Voici le code de la fonction à compléter :

```
string textToMorse(string text)
{
    string codemorse="";
    char c;

    for (unsigned int i=0; i<text.size();i++)
    {
        ..................

        if ((c>='a')&&(c<='z'))
        {
            ..................
        }

        if ((c>='A')&&(c<='Z'))
        {
            ..................
            codemorse+=" ";
        }

        if ((c>='0')&&(c<='9'))
        {
            ..................
            codemorse+=" ";
        }

        if (c==' ')
        {
            ..................
            codemorse+=" ";
        }
    }
    return codemorse;
}
```

Explications : on parcourt la chaine *texte* passee en argument. Chaque caractère est analysé pour savoir si c'est une lettre minuscule ou majuscule ou bien si c'est un chiffre. Si c'est une lettre, on accède au tableau *TMorse*. Si c'est un chiffre, on accède au tableau *TMorseC* pour trouver l'équivalent morse.

2. Fonction *string morseToText(string morse)* :

Cette fonction est un peu plus compliquée puisqu'il faut parcourir la chaine morse pour retrouver chaque lettre. Mais une lettre peut être codée en trois symboles ou quatre voire cinq. C'est là que les espaces sont importants. La chaine morse doit contenir des séparateurs (ici espace) pour pouvoir retrouver la lettre ou le chiffre correspondant.

Compléter la fonction *string morseToText(string morse)* en vous appuyant sur le site :

https://www.cplusplus.com/reference/string/string/?kw=string

Observer surtout les exemples proposés sur le site.

```
string morseToText(string morse)
{
    string sscar;
    string text="";
    int poscourant = 0;
    size_t oldfound;
    //On cherche la premiere occurrence d'un espace
    size_t found = morse.find_first_of(....);

    //Tant qu'on n'est pas a la fin
    while (found!=string::npos)
    {
      sscar = morse.substr(.......... , .........);
      poscourant = found+1;

      //On va cherche dans le tableau TMorse ou TMorseC la sous chaine
sscar
      text += ...................;

      oldfound = found;
      found=morse.find_first_of(" ",found+1);

    }

    ..........++;
    sscar = morse.substr(..................., morse.length()-..........);

    text += lookforcode(sscar);

    return text;
}
```

```
//lookforcode recherche dans les tableaux TMorse et TMorseC
//les caracteres correspondant aux codes morses passés en argument a la
fonction
string lookforcode(string sschaine)
{
    string letexte;

    for (int l=0; l<28;l++)
    {
        if (sschaine == TMorse[l])
        {
            if (l==26)
            {
                letexte = 0x20; // ' '
            }
            else if (l==27)
                 {
                    letexte = 0x27; //'
                 }
                 else
                 {
                   letexte = 0x41+l;
                 }
        }
    }
    for (int c=0; c<10;c++)
    {
        if (sschaine == TMorseC[c])
        {
            letexte = 0x30+c;
        }
    }

    return letexte;
}
```

Travail demandé :

- Créer un projet *non Qt project / Plain C++*
- Ajouter un fichier « morse.h » qui contiendra les trois prototypes des trois fonctions
- Ajouter un fichier « morse.cpp » qui contiendra le code des trois fonctions
- Compléter les deux fonctions dans le fichier « morse.cpp »
- Dans la fonction *main()*, demander à l'utilisateur de saisir un texte. Le programme affiche alors le code morse correspondant
- Un code secret à décoder est proposé :
  #define CODESECRET "..- -. / .--. . - .. - / .--. .- ... / .--. --- ..- .-. / .-.. .----. .... --- -- -- . / ..- -. / --. .-. .- -. -.. / .--. .- ... / .--. --- ..- .-. / .-.. .----. .... ..- -- .- -. .. - ."
  Modifier la fonction *main()* afin de décoder cette phrase.

NB : la saisie avec juste *cin* ne permet pas de prendre en compte du texte avec des espaces. Il y a une solution. A la place de *cin*, je vous propose d'appeler une fonction existante dans *std* :

std::getline(std::*cin*, *texte*);

Cela permet de saisir du texte contenant des espaces !