```
/*
-----------------------------------------------------------------------------
Nom du fichier    : Utilitaires.cpp
Nom du labo       : Labo08_Galton Groupe L
Auteur(s)         : Jeremiah Steiner, Ylli Fazlija
Date creation     : 11.01.2022
Description (But) : Fichier de définition de la librairie Utilitaires. Permets des
                    entrées et sorties basiques.
Remarque(s)       :
Compilateur       : Mingw-w64 g++ 8.1.0
-----------------------------------------------------------------------------
*/

#include <string>
#include <limits>          // Vider le buffer
#include <iostream>
#include "Dictionnaire.h"
#include "Utilitaires.h"

#define VIDER_BUFFER() std::cin.ignore(std::numeric_limits<streamsize>::max(), '\n')

using namespace std;

/// Lis un nombre entré par l'utilisateur, et retourne ledit nombre
/// \param borneMin Valeur minimale acceptable (inclue)
/// \param borneMax Valeur maximale acceptable (inclue)
/// \param msgPrompt Message de demande
/// \param msgErreur Message à Afficher en cas d'erreur
/// \param afficherBornes Est-ce que les bornes sont affichées?
/// \remark Vérification des bornes: [borneMin; borneMax]
/// \return Le nombe récupéré du flux
int LireUnNombre(int borneMin,
                 int borneMax,
                 const string& msgPrompt,
                 const string& msgErreur,
                 bool afficherBornes)
{
   bool erreur;
   int nombreLu;

   do {
      Afficher(msgPrompt, false);

      // Afficher les bornes seulement si on a besoin
      if (afficherBornes)
      {
         Afficher(CARACTERE_ESPACE, false);
         Afficher("["s + to_string(borneMin) + " - " + to_string(borneMax) + "]", false);
      }

      // Afficher fin de prompt et lire nombre
      Afficher(CARACTERE_FIN_PROMPT, false);
      Afficher(CARACTERE_ESPACE, false);
      cin >> nombreLu;

      // Vérification d'erreurs
      erreur = cin.fail() || nombreLu < borneMin || nombreLu > borneMax;
      cin.clear();
      VIDER_BUFFER();

      // La méthode Afficher retourne void, donc on ne peut pas l'utiliser dans la clause
      // du while pour afficher un message en cas d'erreur
      if(erreur)
      {
         Afficher(msgErreur);
      }

   } while(erreur);

   return nombreLu;
}

/// Permet de lire un character, après avoir afficher un message et
/// une liste de char en rapport avec la question, et de le renvoyer
/// \param msgPrompt une string d'affichage
/// \param affichageChars les valeurs prises en compte
/// \param afficherCharsVal boolean, true les char d'information sont affichés
```

```
/// \return le charactere saisi
char LireChar(const string& msgPrompt,
              const vector<char>& affichageChars,
              bool afficherCharsVal)
{
   char charLu = ' ';

   Afficher(msgPrompt, false);
   Afficher(CARACTERE_ESPACE, false);

   // Affiche les réponses possibles.
   if (afficherCharsVal)
   {
      Afficher('[', false);
      Afficher(affichageChars.front(), false);

      for (vector<char>::const_iterator it = affichageChars.begin() + 1;
           it != affichageChars.end();
           ++it)
      {
         Afficher(" / ", false);
         Afficher(*it, false);
      }

      Afficher("] ", false);
   }

   Afficher(CARACTERE_FIN_PROMPT, false);
   Afficher(CARACTERE_ESPACE, false);

   cin >> charLu;
   cin.clear();
   VIDER_BUFFER();

   return charLu;
}

/// Affiche un message dans la console. Fonction surchargée
/// \param message Message à Afficher
/// \param retourLigne Est-ce qu'il faut faire un retour de ligne?
void Afficher(const string& message,
              bool retourLigne)
{
   cout << message;

   if (retourLigne)
   {
      cout << endl;
   }
}

/// Affiche un caractère dans la console. Fonction surchargée
/// \param caractere Caractère à Afficher
/// \param retourLigne Est-ce qu'il faut faire un retour de ligne?
void Afficher(char caractere,
              bool retourLigne)
{
   Afficher(string(1, caractere), retourLigne);
}


/// Surcharge de l'opérateur de flux afin d'afficher un vecteur.
/// \param os Flux de sortie
/// \param v Vecteur à afficher.
/// \return Référence au flux de sortie.
ostream& operator<<(ostream& os,
                    const vector<int>& v)
{
   os << '[';
   for (vector<int>::const_iterator it = v.begin(); it != v.end(); ++it)
   {
      if (it != v.begin())
         os << ", ";
      os << *it;
   }
   return os << ']';
}
```

```
/// Permet de savoir si un char apparait dans une vecteur de char
/// \param c char a tester
/// \param listDeChar un vecteur de char
/// \return true si le char est dans la liste, false sinon
bool estCharDansVect(const char& c,
                     const vector<char>& listDeChar)
{
   return (*find(listDeChar.begin(), listDeChar.end(), c) == c);
}
```