```cpp
/*
-----------------------------------------------------------------------------
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
```

```
77  /// \param afficherCharsVal boolean, true les char d'information sont affichés
78  /// \return le charactere saisi
79  char LireChar(const string& msgPrompt,
80                    const vector<char>& affichageChars,
81                    bool afficherCharsVal)
82  {
83      char charLu = ' ';
84
85      Afficher(msgPrompt, false);
86      Afficher(CARACTERE_ESPACE, false);
87
88     // Affiche les réponses possibles.
89      if (afficherCharsVal)
90      {
91          Afficher('[', false);
92          Afficher(affichageChars.front(), false);
93
94          for (vector<char>::const_iterator it = affichageChars.begin() + 1;
95                it != affichageChars.end();
96                ++it)
97          {
98              Afficher(" / ", false);
99              Afficher(*it, false);
100         }
101
102         Afficher("] ", false);
103     }
104
105     Afficher(CARACTERE_FIN_PROMPT, false);
106     Afficher(CARACTERE_ESPACE, false);
107
108     cin >> charLu;
109     cin.clear();
110     VIDER_BUFFER();
111
112     return charLu;
113 }
114
115 /// Affiche un message dans la console. Fonction surchargée
116 /// \param message Message à Afficher
117 /// \param retourLigne Est-ce qu'il faut faire un retour de ligne?
118 void Afficher(const string& message,
119                   bool retourLigne)
120 {
121     cout << message;
122
123     if (retourLigne)
124     {
125         cout << endl;
126     }
127 }
128
129 /// Affiche un caractère dans la console. Fonction surchargée
130 /// \param caractere Caractère à Afficher
131 /// \param retourLigne Est-ce qu'il faut faire un retour de ligne?
132 void Afficher(char caractere,
133                   bool retourLigne)
134 {
135     Afficher(string(1, caractere), retourLigne);
136 }
137
138
139 /// Surcharge de l'opérateur de flux afin d'afficher un vecteur.
140 /// \param os Flux de sortie
141 /// \param v Vecteur à afficher.
142 /// \return Référence au flux de sortie.
143 ostream& operator<<(ostream& os,
144                           const vector<int>& v)
145 {
146    os << '[';
147    for (vector<int>::const_iterator it = v.begin(); it != v.end(); ++it)
148     {
149       if (it != v.begin())
150          os << ", ";
151       os << *it;
152    }
```

```
153       return os << ']';
154   }
155
156   /// Permet de savoir si un char apparait dans une vecteur de char
157   /// \param c char a tester
158   /// \param listDeChar un vecteur de char
159   /// \return true si le char est dans la liste, false sinon
160   bool estCharDansVect(const char& c,
161                        const vector<char>& listDeChar)
162   {
163       return (*find(listDeChar.begin(), listDeChar.end(), c) == c);
164   }
```