```
/*
 ------------------------------------------------------------------------------------
 Laboratoire : Labo1
 Fichier     : Temps.h
 Auteur(s)   : Ganguillet, Parrino et Scharwath
 Date        : 02.03.2020

 But         : Cette librairie met à disposition une classe Temps permettant d'effectuer
               différentes opérations sur les heures telles que :
               - comparaisons
               - addition et soustraction (de deux objet Temps)
               - post et pré incrémentation et décrémentation
               - affichage dans un ostream au format hh:mm:ss
               - getters et setters pour l'heure, les minutes, les secondes
               - convertion en temps décimal (double) exprimé en heures

 Remarque(s) : Il n'y a pas de vérification des valeurs entrées dans les setteurs et
               dans le constructeur. La cohérence des résultats n'est pas garantie si les
               valeurs entrées ne sont pas comprises dans les bornes [0-23] pour les heures,
               [0-59] pour les minutes et les secondes.

 Compilateur : MinGW-g++ 6.3.0
 ------------------------------------------------------------------------------------
*/

#ifndef LABO1_TEMPS_H
#define LABO1_TEMPS_H

#include <ctime>
#include <iostream>

class Temps {
   /**
    * Operateur d'égalité
    * @param lhs
    * @param rhs
    * @return true si égal
    */
   friend bool operator==(const Temps &lhs, const Temps &rhs);

   /**
    * Operateur de différence
    * @param lhs
    * @param rhs
    * @return true si différent
    */
   friend bool operator!=(const Temps &lhs, const Temps &rhs);

   /**
    * Opérateur strictement inférieur
    * @param lhs
    * @param rhs
    * @return true si strictement inférieur
    */
   friend bool operator<(const Temps &lhs, const Temps &rhs);

   /**
    * Opérateur strictement supérieur
    * @param lhs
    * @param rhs
    * @return true si strictement supérieur
    */
   friend bool operator>(const Temps &lhs, const Temps &rhs);

   /**
    * Opérateur inférieur
    * @param lhs
    * @param rhs
    * @return true si inférieur
    */
   friend bool operator<=(const Temps &lhs, const Temps &rhs);

   /**
    * Opérateur supérieur
    * @param lhs
    * @param rhs
    * @return true si supérieur
```

```
78       */
79      friend bool operator>=(const Temps &lhs, const Temps &rhs);
80
81      /**
82       * Opérateur d'addtion
83       * @param lhs
84       * @param rhs
85       * @return Temps additionné
86       */
87      friend Temps operator+(Temps lhs, const Temps &rhs);
88
89      /**
90       * Opérateur de soustraction
91       * @param lhs
92       * @param rhs
93       * @return Temps soustrait
94       */
95      friend Temps operator-(Temps lhs, const Temps &rhs);
96
97      /**
98       * surcharge du flux format (hh:mm:ss)
99       * @param os
100      * @param temps
101      * @return ostream& référence au flux de sortie
102      */
103     friend std::ostream &operator<<(std::ostream &os, const Temps &temps);
104
105 public:
106 //---------------------------------CONSTRUCTEUR-------------------------------------//
107     /**
108      * Constructeur par défaut où tout est initialisé à 0
109      */
110     Temps();
111
112     /**
113      * initialisation avec un objet de type time_t
114      * @example Temps t1 = time(NULL);
115      *          Temps t1(time(NULL));
116      * @param heureCourante
117      */
118     Temps(const time_t &heureCourante);
119
120     /**
121      * initialisation avec heures, minutes, secondes optionnel
122      * @param heures
123      * @param minutes
124      * @param secondes
125      */
126     Temps(unsigned int heures, unsigned int minutes, unsigned int secondes = 0);
127 //----------------------------------------------------------------------------------//
128
129 //-----------------------------------GETTER-----------------------------------------//
130     /**
131      * getter des heures
132      * @return l'heure
133      */
134     unsigned int getHeures() const;
135
136     /**
137      * getter des minutes
138      * @return les minutes
139      */
140     unsigned int getMinutes() const;
141
142     /**
143      * getter des secondes
144      * @return les secondes
145      */
146     unsigned int getSecondes() const;
147 //----------------------------------------------------------------------------------//
148
149 //---------------------------------SETTER-------------------------------------------//
150     /**
151      * setter des heures
152      * @param heures
153      */
154     void setHeures(unsigned int heures);
```

```
155
156      /**
157       * setter des minutes
158       * @param minutes
159       */
160      void setMinutes(unsigned int minutes);
161
162      /**
163       * setter des secondes
164       * @param secondes
165       */
166      void setSecondes(unsigned int secondes);
167   //-------------------------------------------------------------------------------//
168
169   //-----------------------------FONCTION MEBRES-----------------------------------//
170      /**
171       * converti en temps décimal exprimé en heures
172       * @return double heures en décimal
173       */
174       explicit operator double() const;
175
176      /**
177       * surcharge de l'opérateur +=
178       * @param rhs Temps à rajouter
179       * @return Temps
180       */
181      Temps &operator+=(const Temps &rhs);
182
183      /**
184       * surcharge de l'opérateur -=
185       * @param rhs Temps à soustraire
186       * @return Temps
187       */
188      Temps &operator-=(const Temps &rhs);
189
190      /**
191       * pre-incrementation de Temps
192       * @return Temps
193       */
194      Temps &operator++();
195
196      /**
197       * post-incrémentation de Temps
198       * @return Temps
199       */
200      Temps operator++(int);
201
202      /**
203       * pre-decrementation de Temps
204       * @return Temps
205       */
206      Temps &operator--();
207
208      /**
209       * post-decrementation de Temps
210       * @return Temps
211       */
212      Temps operator--(int);
213
214   //-------------------------------------------------------------------------------//
215
216   private:
217
218      unsigned int heures;
219      unsigned int minutes;
220      unsigned int secondes;
221
222      static unsigned int NB_SECONDES_JOUR; // nombre de secondes dans une journée
223
224      /**
225       * calcule le Temps en secondes
226       * @return nombre de secondes
227       */
228      unsigned int tempsEnSecondes() const;
229
230      /**
231       * actualise les champs membres de Temps
```

```
232         * @param tempsEnSecondes
233         * @return Temps actualisé
234         */
235        Temps &secondesEnTemps(unsigned int tempsEnSecondes);
236
237    };
238
239
240    #endif //LABO1_TEMPS_H
241
```