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

 But         : Implémentation de la classe Temps.

 Remarque(s) : La majorité des calculs se fait en convertissant le temps en secondes.
               Si les valeurs membres ne sont pas comprises dans les bornes [0-23] pour les
               heures, [0-59] pour les minutes et les secondes, certaines fonctions vont
               redimentionner ces valeurs dans leurs bornes respectives, sans toutefois
               garantir la cohérance des résultats.

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

#include "Temps.h"
#include <iomanip>

using namespace std;

unsigned int Temps::NB_SECONDES_JOUR = 86400; // = 60*60*24

Temps::Temps() : heures{0}, minutes{0}, secondes{0} {}

Temps::Temps(unsigned int heures, unsigned int minutes, unsigned int secondes)
      : heures{heures}, minutes{minutes}, secondes{secondes} {}


Temps::Temps(const time_t &tempsCourante) {
   tm *now  = localtime(&tempsCourante);
   heures   = (unsigned int) now->tm_hour;
   minutes  = (unsigned int) now->tm_min;
   secondes = (unsigned int) now->tm_sec;
}

unsigned int Temps::getHeures() const {
   return heures;
}

unsigned int Temps::getMinutes() const {
   return minutes;
}

unsigned int Temps::getSecondes() const {
   return secondes;
}

void Temps::setHeures(unsigned int heures) {
   this->heures = heures;
}

void Temps::setMinutes(unsigned int minutes) {
   this->minutes = minutes;
}

void Temps::setSecondes(unsigned int secondes) {
   this->secondes = secondes;
}

bool operator==(const Temps &lhs, const Temps &rhs) {
   return lhs.tempsEnSecondes() == rhs.tempsEnSecondes();
}

bool operator!=(const Temps &lhs, const Temps &rhs) {
   return !(rhs == lhs);
}

bool operator<(const Temps &lhs, const Temps &rhs) {
   return lhs.tempsEnSecondes() < rhs.tempsEnSecondes();
}

bool operator>(const Temps &lhs, const Temps &rhs) {
   return rhs < lhs;
```

```
 78  }
 79
 80  bool operator<=(const Temps &lhs, const Temps &rhs) {
 81     return !(rhs < lhs);
 82  }
 83
 84  bool operator>=(const Temps &lhs, const Temps &rhs) {
 85     return !(lhs < rhs);
 86  }
 87
 88  Temps &Temps::operator+=(const Temps &rhs) {
 89     return secondesEnTemps(tempsEnSecondes() + rhs.tempsEnSecondes());
 90  }
 91
 92  Temps operator+(Temps lhs, const Temps &rhs) {
 93     lhs += rhs;
 94     return lhs;
 95  }
 96
 97  Temps &Temps::operator-=(const Temps &rhs) {
 98     unsigned sec1 = tempsEnSecondes(),
 99              sec2 = rhs.tempsEnSecondes();
100     if (sec1 > sec2) {
101        return secondesEnTemps(sec1 - sec2);
102     }
103     return secondesEnTemps(NB_SECONDES_JOUR + sec1 - sec2);
104  }
105
106  Temps operator-(Temps lhs, const Temps &rhs) {
107     lhs -= rhs;
108     return lhs;
109  }
110
111  //setfill('0') << setw(2) permet de mettre un zero si le nombre est plus petit que 10
112  ostream &operator<<(ostream &os, const Temps &temps) {
113     os << setfill('0') << setw(2) << temps.heures  << ":"
114        << setfill('0') << setw(2) << temps.minutes << ":"
115        << setfill('0') << setw(2) << temps.secondes;
116     return os;
117  }
118
119  Temps &Temps::operator++() {
120     return secondesEnTemps(tempsEnSecondes() + 1);
121  }
122
123  Temps Temps::operator++(int) {
124     Temps temps = *this;
125     ++(*this);
126     return temps;
127  }
128
129  Temps &Temps::operator--() {
130     unsigned int sec = tempsEnSecondes();
131     return secondesEnTemps( (sec ? sec : NB_SECONDES_JOUR) - 1);
132  }
133
134  Temps Temps::operator--(int) {
135     Temps temps = *this;
136     --(*this);
137     return temps;
138  }
139
140  Temps::operator double() const {
141     return (double) heures + (double) minutes * 1 / 60. + (double) secondes * 1 / 3600.;
142  }
143
144  unsigned int Temps::tempsEnSecondes() const {
145     //converti Temps en nombre de secondes compris entre 0 et NB_SECONDES_JOUR
146     return (heures * 3600 + minutes * 60 + secondes) % NB_SECONDES_JOUR;
147  }
148
149  Temps &Temps::secondesEnTemps(unsigned int tempsEnSecondes) {
150     heures   = tempsEnSecondes / 3600 % 24;   //converti en heures [0-23]
151     minutes  = tempsEnSecondes / 60 % 60;     //converti en minutes [0-59]
152     secondes = tempsEnSecondes % 60;          //converti en secondes [0-59]
153     return *this;
154  }
```