

# TP POO C++ : La surcharge des opérateurs

---

© 2013-2017 tv <tvaira@free.fr> - v.1.2

<b>Travail demandé</b>	<b>2</b>
La classe Temps . . . . .	2
La surcharge des opérateurs de flux « et » . . . . .	3
La surcharge des opérateurs arithmétiques . . . . .	5
La surcharge des opérateurs logiques . . . . .	7

# TP POO C++ : La surcharge des opérateurs

Les objectifs de ce TP sont de mettre en œuvre la surcharge des opérateurs de flux « et » et de quelques opérateurs arithmétiques en C++.

## Travail demandé

On va modéliser une classe `Temps` qui aura les caractéristiques suivantes :

- la valeur du temps sera conservée sous forme d'un entier long exprimant des secondes
- on doit pouvoir initialiser un objet `Temps` soit avec un nombre de secondes soit avec un nombre d'heures, minutes et secondes
- on doit obtenir soit le nombre de secondes soit le nombre d'heures, minutes et secondes d'un objet `Temps`

```
class Temps
{
private:
    long valeur; // la valeur du temps en secondes

public:
    Temps();
    Temps(int secondes);
    Temps(int heure, int minute, int seconde);

    int getSeconde() const;
    int getMinute() const;
    int getHeure() const;
    int getValeur() const;
};
```

*Temps.h*

On vous fournit un programme `testTemps.cpp` où vous devez décommenter progressivement les parties de code source correspondant aux questions posées.

## La classe Temps

**Question 1.** Définir la classe `Temps` en complétant le fichier `Temps.cpp` afin de pouvoir instancier divers objets `Temps`. Décommenter les parties de code de cette question dans le fichier `testTemps.cpp` et tester. Vérifier que vous obtenez les résultats ci-dessous.

```
/* Question 1 */
cout << "Question 1 : " << endl;
Temps t1;
cout << "Temps : " << t1.getValeur() << " secondes\n";
cout << "Temps (HH:MM:SS) : " << setfill('0') << setw(2) << t1.getHeure() << ":" << setfill(
    '0') << setw(2) << t1.getMinute() << ":" << setfill('0') << setw(2) << t1.getSeconde()
    << endl;
```

```
Temps t2(3670);
cout << "Temps : " << t2.getValeur() << " secondes\n";
cout << "Temps (HH:MM:SS) : " << setfill('0') << setw(2) << t2.getHeure() << ":" << setfill(
    '0') << setw(2) << t2.getMinute() << ":" << setfill('0') << setw(2) << t2.getSeconde()
    << endl;

Temps t3(1, 60, 30);
cout << "Temps : " << t3.getValeur() << " secondes\n";
cout << "Temps (HH:MM:SS) : " << setfill('0') << setw(2) << t3.getHeure() << ":" << setfill(
    '0') << setw(2) << t3.getMinute() << ":" << setfill('0') << setw(2) << t3.getSeconde()
    << endl;

cout << endl;
```

Ce qui doit donner :

Question 1 :

```
Temps : 0 secondes
Temps (HH:MM:SS) : 00:00:00
Temps : 3670 secondes
Temps (HH:MM:SS) : 01:01:10
Temps : 7230 secondes
Temps (HH:MM:SS) : 02:00:30
```

## La surcharge des opérateurs de flux « et »

*Rappels* : Un **flot** est un **canal recevant (flot d'« entrée »)** ou **fournissant (flot de « sortie ») de l'information**. Ce canal est associé à un périphérique ou à un fichier.

Un **flot d'entrée** est un objet de type `istream` tandis qu'un **flot de sortie** est un objet de type `ostream`.



Le flot `cout` est un flot de sortie prédéfini connecté à la sortie standard `stdout`. De même, le flot `cin` est un flot d'entrée prédéfini connecté à l'entrée standard `stdin`.

On surchargera les opérateurs de flux « et » pour une classe quelconque, sous forme de **fonctions amies**, en utilisant ces « canevas » :

```
ostream & operator << (ostream & sortie, const type_classe & objet1)
{
    // Envoi sur le flot sortie des membres de objet en utilisant
    // les possibilités classiques de << pour les types de base
    // c'est-à-dire des instructions de la forme :
    // sortie << ..... ;

    return sortie ;
}

istream & operator >> (istream & entree, type_de_base & objet)
{
    // Lecture des informations correspondant aux différents membres de objet
    // en utilisant les possibilités classiques de >> pour les types de base
```

```
// c'est-à-dire des instructions de la forme :  
// entree >> ..... ;  
  
return entree ;  
}
```

**Question 2.** Compléter les fichiers `Temps.cpp` et `Temps.h` fournis afin d'implémenter la surcharge de l'opérateur de flux de sortie « pour qu'il affiche un objet `Temps` de la manière suivante : `HH:MM:SS`. Décommenter les parties de code de cette question dans le fichier `testTemps.cpp` et tester. Vérifier que vous obtenez les résultats ci-dessous.

```
/* Question 2 */  
cout << "Question 2 : " << endl;  
cout << "Temps t1 : " << t1 << endl;  
cout << "Temps t2 : " << t2 << endl;  
cout << "Temps t3 : " << t3 << endl;  
cout << endl;
```

Ce qui doit donner :

```
Question 2 :  
Temps t1 : 00:00:00  
Temps t2 : 01:01:10  
Temps t3 : 02:00:30
```



Vous utiliserez les fonctions `setfill()` et `setw()` pour formater l'affichage souhaité. Pour disposer de ces fonctions, il vous faudra inclure le fichier d'en-tête `iomanip`.

**Question 3.** Compléter les fichiers `Temps.cpp` et `Temps.h` fournis afin d'implémenter la surcharge de l'opérateur de flux d'entrée » pour qu'il réalise la saisie d'un objet `Temps` à partir d'un nombre de secondes. Décommenter les parties de code de cette question dans le fichier `testTemps.cpp` et tester. Vérifier que vous obtenez les résultats ci-dessous.

```
cout << "Question 3 : " << endl;  
Temps t4;  
  
cin >> t4;  
if (! cin)  
{  
    cout << "Erreur de lecture !\n";  
    cout << endl;  
}  
else  
{  
    cout << "Temps t4 : " << t4.getValeur() << " secondes\n";  
    cout << "Temps t4 : " << t4 << endl;  
    cout << endl;  
}
```

Bonus : modifier la surcharge de l'opérateur de flux d'entrée » pour qu'il réalise aussi la saisie d'un objet `Temps` de la manière suivante : `HH:MM:SS`.

**Question 4.** L'opérateur de flux » fonctionne aussi avec des fichiers. Décommenter les parties de code de cette question dans le fichier `testTemps.cpp` et tester. Vérifier que vous obtenez les résultats ci-dessous.

```
fichier = new fstream("chronos.txt", fstream::in);

cout << "Natation > Championnats du monde 2017 : 1500m nage libre Homme" << endl;

*fichier >> nom;
while (! fichier->eof())
{
    *fichier >> temps;
    cout << nom << " -> " << temps << endl;
    while (fichier->get() != '\n');
    *fichier >> nom;
}

fichier->close();
delete fichier;
cout << endl;
```

Ce qui doit donner :

```
Question 4 :
Natation > Championnats du monde : 1500m nage libre Homme
Paltrinieri -> 00:14:35
Romanchuk -> 00:14:37
Horton -> 00:14:47
Detti -> 00:14:52
Christiansen -> 00:14:54
Frolov -> 00:14:55
Wojdak -> 00:15:01
Micka -> 00:15:09
```

**Question 5.** L'opérateurs de flux « fonctionne aussi avec des fichiers. Compléter le programme `testPoint.cpp` afin d'écrire les temps `t1`, `t2` et `t3` dans un fichier texte `resultats.txt`.

Vous devez obtenir le contenu suivant :

```
$ cat resultats.txt
00:00:00
01:01:10
02:00:30
```

## La surcharge des opérateurs arithmétiques

**Question 6.** Compléter les fichiers `Temps.cpp` et `Temps.h` fournis afin d'implémenter la surcharge des opérateurs utilisés dans le code source fourni. Décommenter les parties de code de cette question dans le fichier `testTemps.cpp` et tester. Vérifier que vous obtenez les résultats ci-dessous.

```
/* Question 6 */
cout << "Question 6 : " << endl;
```

```
cout << "Temps t1 : " << t1 << endl;
t1 += 10;
cout << "Temps t1 + 10 : " << t1 << endl << endl;

cout << "Temps t1 : " << t1 << endl;
t1 -= 10;
cout << "Temps t1 - 10 : " << t1 << endl << endl;

cout << "Temps t2 : " << t2 << endl;
cout << "Temps t3 : " << t3 << endl;
t2 += t3;
cout << "Temps t2 += t3 : " << t2 << endl << endl;

cout << "Temps t2 : " << t2 << endl;
cout << "Temps t3 : " << t3 << endl;
t2 -= t3;
cout << "Temps t2 -= t3 : " << t2 << endl << endl;

cout << "Temps t2 : " << t2 << endl;
cout << "Temps t3 : " << t3 << endl;
t1 = t2 + t3;
cout << "Temps t2 + t3 : " << t1 << endl << endl;

cout << "Temps t2 : " << t2 << endl;
cout << "Temps t3 : " << t3 << endl;
t1 = t3 - t2;
cout << "Temps t3 - t2 : " << t1 << endl << endl;
```

Ce qui doit donner :

Question 6 :

Temps t1 : 00:00:00

Temps t1 + 10 : 00:00:10

Temps t1 : 00:00:10

Temps t1 - 10 : 00:00:00

Temps t2 : 01:01:10

Temps t3 : 02:00:30

Temps t2 += t3 : 03:01:40

Temps t2 : 03:01:40

Temps t3 : 02:00:30

Temps t2 -= t3 : 01:01:10

Temps t2 : 01:01:10

Temps t3 : 02:00:30

Temps t2 + t3 : 03:01:40

Temps t2 : 01:01:10

Temps t3 : 02:00:30

Temps t3 - t2 : 00:59:20

Bonus : ajouter la surcharge des opérateurs d'incréméntation (++) et décrémentation (--).

## La surcharge des opérateurs logiques

**Question 7.** Compléter les fichiers `Temps.cpp` et `Temps.h` fournis afin d'implémenter la surcharge des opérateurs utilisés dans le code source fourni. Décommenter les parties de code de cette question dans le fichier `testTemps.cpp` et tester. Vérifier que vous obtenez les résultats ci-dessous.

```
/* Question 7 */
cout << "Question 7 : " << endl;
Temps t5(1, 1, 10); // 01:01:10

if(t2 == t5) // t2 : 01:01:10
{
    cout << "t2 égal à t5\n";
}
else
{
    cout << "t2 différent de t5\n";
}

if(t3 != t5) // t3 : 02:00:30
{
    cout << "t3 différent de t5\n";
}
else
{
    cout << "t3 égal à t5\n";
}
```

Ce qui doit donner :

t2 égal à t5

t3 différent de t5