```
/*

  Programma : cppClassTemplateNumero

  Autore : Riontino Raffaele classe 4 informatici (serale)
           ITIS Molinari - Milano 17/01/2011

  Funzionalità : la classe template è una classe generica che
                 viene specializzata in fase di utilizzo.
                 Questa classe permette di creare ed utilizzare
                 oggetti che hanno le stesse caratteristiche dei
                 tipi di variabili che gestiscono numeri come
                 int, float, double...

  aggiornamento : 18/01/2011 -> aggiunto il metodo 'operator !='.
*/

#include <iostream>
using namespace std;

template <class T>
class Numero
      {
       public :
               Numero (); //costruttore senza parametro
               Numero (T); //costruttote con parametro
               void out (); //metodo che visualizza il contenuto
               void input (); //metodo che consente di introdurre un parametro
               T getValue (); //metodo che ritorna il parametro
               void setValue (T); //metodo che cambia il parametro
               Numero operator + (Numero); //metodo per la somma
               Numero operator - (Numero); //metodo per la sottrazione
               Numero operator * (Numero); //metodo per la moltriplicazione
               Numero operator / (Numero); //metodo per la divisione
               void operator += (Numero); //metodo per la somma
               void operator -= (Numero); //metodo per la sottrazione
               void operator *= (Numero); //metodo per la moltriplicazione
               void operator /= (Numero); //metodo per la divisione
               bool operator > (Numero); //metodo per confrontare se maggiore
               bool operator < (Numero); //metodo per confrontare se minore
               bool operator >= (Numero); //metodo per confrontare se maggiore-uguale
               bool operator <= (Numero); //metodo per confrontare se minore-uguale
               bool operator == (Numero); //metodo per confrontare se uguale
               bool operator != (Numero); //metodo per confrontare se diversi
               void operator = (Numero); // medoto di assegnazione di un valore
               Numero operator % (Numero); //metodo che titorna il resto della divisione
               void operator ++ (int); //metodo che incrementa di un elemento
               void operator -- (int); //metodo che decrementea di un elemento
               ~Numero (); //distruttore della classe
       private :
                T dato;
      };

 template<class T>
 Numero<T>::Numero ()
    {
     this->dato = 0;
    }

 template<class T>
 Numero<T>::Numero (T temp)
    {
     this->dato = temp;
    }

 template<class T>
```

```
void Numero<T>::out ()
   {
    cout << this->dato << endl;
   }

template<class T>
void Numero<T>::input ()
   {
    cin >> this->dato;
   }

template<class T>
T Numero<T>::getValue ()
   {
    return this->dato;
   }

template<class T>
void Numero<T>::setValue (T temp)
   {
    this->dato = temp;
   }

template<class T>
Numero<T> Numero<T>::operator + (Numero temp)
   {
    return this->dato + temp.dato;
   }

template<class T>
Numero<T> Numero<T>::operator - (Numero temp)
   {
    return this->dato - temp.dato;
   }

template<class T>
Numero<T> Numero<T>::operator * (Numero temp)
   {
    return this->dato * temp.dato;
   }

template<class T>
Numero<T> Numero<T>::operator / (Numero temp)
   {
    return this->dato / temp.dato;
   }

template<class T>
void Numero<T>::operator += (Numero temp)
   {
    this->dato += temp.dato;
   }

template<class T>
void Numero<T>::operator -= (Numero temp)
   {
    this->dato -= temp.dato;
   }

template<class T>
void Numero<T>::operator *= (Numero temp)
   {
    this->dato *= temp.dato;
   }

template<class T>
```

```
void Numero<T>::operator /= (Numero temp)
   {
    this->dato /= temp.dato;
   }

template<class T>
bool Numero<T>::operator > (Numero temp)
   {
    if (this->dato > temp.dato) return true;
    else return false;
   }

template<class T>
bool Numero<T>::operator < (Numero temp)
   {
    if (this->dato < temp.dato) return true;
    else return false;
   }

template<class T>
bool Numero<T>::operator >= (Numero temp)
   {
    if (this->dato >= temp.dato) return true;
    else return false;
   }

template<class T>
bool Numero<T>::operator <= (Numero temp)
   {
    if (this->dato <= temp.dato) return true;
    else return false;
   }

template<class T>
bool Numero<T>::operator == (Numero temp)
   {
    if (this->dato == temp.dato) return true;
    else return false;
   }

template<class T>
bool Numero<T>::operator != (Numero temp)
   {
    if (this->dato != temp.dato) return true;
    else return false;
   }

template<class T>
void Numero<T>::operator = (Numero temp)
   {
    this->dato = temp.dato;
   }

template<class T>
Numero<T> Numero<T>::operator % (Numero temp)
   {
    return (this->dato % temp.dato);
   }

template<class T>
void Numero<T>::operator ++ (int)
   {
    this->dato ++;
   }

template<class T>
```

```
 void Numero<T>::operator -- (int)
    {
     this->dato -- ;
    }

 template<class T>
 Numero<T>::~Numero()
   {
   }


    //fine classe

int main()
    {
     Numero<int> a; //dichiarazione di un oggetto di tipo intero
     Numero<int> b(4); //dichiarazione di un oggetto di tipo intero ed assegnazione di un
valore
     Numero<int> ris;
     cout << "\n\tAlcuni esempi di utilizzo della classe Numero\n\n";
     ris = a + b; //somma dei due oggetti
     cout << "\nint\nris = a + b -> ris = " << a.getValue() << " + " << b.getValue() << "
= " << ris.getValue();
     Numero<float> a1, b1, ris1; //dichiarazione di tre oggetti di tipo float
     cout << "\n\nfloat\ninserisci il valore di a1 : ";
     a1.input();
     cout << "inserisci il valore di b1 : ";
     b1.input();
     ris1 = a1 / b1;
     cout << "ris1 = a1 / b1 -> " << ris1.getValue();
     Numero<double> a2 , b2;
     cout << "\n\ndouble\ninserisci il valore di a2 : ";
     a2.input();
     cout << "inserisci il valore di b2 : ";
     b2.input();
     if (a2 > b2) cout << "a2 > b2\n";
     else if (a2 < b2) cout << "a2 < b2\n";
          else if (a2 == b2) cout << "a2 == b2\n";
     system("pause");
     return 1;
    }
```