```
bool Vetor::operator==(Vetor &right )/* determina se dois vetores são
iguais e retorna true, caso contrário retorna false*/
{
   if ( tamanho != right.tamanho )
      return false; // vetores com diferentes números de elementos
   for ( int i = 0; i < tamanho; i++ )
      if ( ptr[ i ] != right.ptr[ i ] )
         return false; // os conteúdos dos vetores não são iguais
   return true; // vetores são iguais
}
int &Vetor::operator[]( int indice ) /* operador de indexação
sobrecarregado para vetores*/
{
   if ( indice < 0 || indice >= tamanho ) /* verifica erro de indice fora do
intervalo*/
   {
      cerr << "\nErro: Indice " << indice << " fora do intervalo." << endl;
      exit( 1 ); // termina o programa indexado fora do intervalo
   }
   return ptr[ indice ]; // retorno da referência
}
```

```
istream &operator>>( istream &input, Vetor &a )// operador de entrada sobrecarregado para a classe Vetor
{
   for ( int i = 0; i < a.tamanho; i++ )
      input >> a.ptr[ i ];
   return input; // permite cin >> x >> y;
}
ostream &operator<<( ostream &output, Vetor &a )// operador de saída sobrecarregado para classe Vetor
{
   int i;
   for ( i = 0; i < a.tamanho; i++ )// gera saída do vetor baseado em ptr
private
   {
      output << setw( 12 ) << a.ptr[ i ];
      if ( ( i + 1 ) % 4 == 0 ) // 4 números por linha de saída
         output << endl;
   }
   if ( i % 4 != 0 ) // termina a última linha de saída
      output << endl;
   return output; // permite cout << x << y;
}
```

```
//conteúdo do arquivo principalVetor.cpp
#include "vetor.h" // definição da classe Vetor
#include <iostream>
using namespace std;
int main()
{
   Vetor inteiros1( 7 ); // Vetor de sete elementos
   Vetor inteiros2; // Vetor de 10 elementos por padrão
   // imprime o tamanho e o conteúdo de inteiros1
   cout << "O tamanho do vetor inteiros1 eh " << inteiros1.getTamanho()
<< endl << "Vetor apos inicializacao: " << endl << inteiros1;
   // imprime o tamanho e o conteúdo de inteiros2
   cout << "O tamanho do vetor inteiros2 eh " << inteiros2.getTamanho()
<< endl << "Vetor apos inicializacao: " << endl << inteiros2;
   // insere e imprime inteiros1 e inteiros2
   cout << "\nEntre com 17 inteiros:" << endl;
   cin >> inteiros1 >> inteiros2;
   cout << endl << "Depois da entrada, os vetores contem:" << endl <<
"inteiros1:" << endl << inteiros1 << "inteiros2:" << endl << inteiros2;
   // utiliza o operador de desigualdade (!=) sobrecarregado
   cout << endl << "Avaliacao: inteiros1 != inteiros2" << endl;
   if ( inteiros1 != inteiros2 )
      cout << "inteiros1 e inteiros2 nao sao iguais" << endl;
```

```
   // cria vetor inteiros3 utilizando inteiros1 como um inicializador
   // imprime tamanho e conteúdo
   Vetor inteiros3( inteiros1 ); // invoca o construtor de cópia
   cout << "O tamanho do vetor inteiros3 eh " << inteiros3.getTamanho()
<< endl << "Vetor apos inicializacao: " << endl << inteiros3;
   // utiliza operador atribuição (=) sobrecarregado
   cout << endl << "Atribuicao de inteiros2 para inteiros1:" << endl;
   inteiros1 = inteiros2; // note que o vetor alvo é menor
   cout << "inteiros1:" << endl << inteiros1 << "inteiros2:" << endl <<
inteiros2;
   // utiliza operador de igualdade (==) sobrecarregado
   cout << endl << "Avaliando: inteiros1 == inteiros2" << endl;
   if ( inteiros1 == inteiros2 )
      cout << "inteiros1 e inteiros2 sao iguais." << endl;
   // utiliza operador de indexação sobrecarregado como rvalue
   cout << endl << "inteiros1[5] eh " << inteiros1[ 5 ];
   // utiliza operador de indexação sobrecarregado como lvalue
   cout << endl << "Atribuindo 1000 para inteiros1[5]" << endl;
   inteiros1[ 5 ] = 1000;
   cout << "inteiros1:" << endl << inteiros1;
   // tentativa de utilizar subscrito fora do intervalo
   cout << endl << "Tentativa de atribuir 1000 para inteiros1[15]" << endl;
   inteiros1[ 15 ] = 1000; // ERRO: fora do intervalo
   return 0;
}
```

Exercício:

Com base no que estudamos, expanda a classe DataSobrecargaOperador sobrecarregando o operador binário +. Você deve possibilitar que o operador + possa ser utilizado para somar um número inteiro a uma data, gerando uma nova data que será constituída da data fornecida deslocada do número de dias, representado pelo operando inteiro.

Observações: garanta a comutatividade da operação e considere os anos bissextos.

Exercício:

Agora, efetuaremos mais uma expansão na classe DataSobrecargaOperador sobrecarregando o operador binário - e operador unário -. Você deve possibilitar que o operador binário - possa ser utilizado para subtrair duas datas, gerando um inteiro que representa a quantidade de dias decorridos entre as datas. Já o operador - unário, ao ser aplicado sobre um objeto da classe DataSobrecargaOperador retorna o número de dias decorridos desde o início do ano contido na data.

Observação: considere os anos bissextos.