IntVector.h

```
#ifndef __INTVECTOR_H
#define __INTVECTOR_H


class CIntVector {
private:
      enum { DEFAULT_SIZE = 10, GROW_STEP = 5 };

      int * m_iVector;
      int   m_iSize;

public:
      CIntVector( int iSize = DEFAULT_SIZE );
      CIntVector( const CIntVector& v );
      ~CIntVector();

      bool update( int index, int  value );
      bool index(  int index, int& value );
      int  search( int value );
      void showAll( void );

      int  size() { return m_iSize; }
      bool grow( int iTimes = 1, int iStep = GROW_STEP );
      void reset();
};


#endif
```

IntVector.cpp

```
#include <iostream.h>
#include "IntVector.h"


/* construtor por defeito */
CIntVector::CIntVector( int iSize /* = DEFAULT_SIZE */ )
{
	m_iVector = new int[ iSize ];

	if ( m_iVector != NULL )
		m_iSize = iSize;
	else
		m_iSize = 0;
}


/* construtor cópia */
CIntVector::CIntVector( const CIntVector& v )
{
	m_iVector = new int[ v.m_iSize ];

	if ( m_iVector != NULL ) {

		m_iSize = v.m_iSize;

		for ( int i = 0; i < m_iSize; i++ )
			m_iVector[ i ] = v.m_iVector[ i ];
	}
	else
		m_iSize = 0;
}

/* destrutor */
CIntVector::~CIntVector()
{
	reset();
}

/* liberta a memória do vector */
void CIntVector::reset()
{
	if ( m_iVector != NULL )
		delete [] m_iVector;

	m_iSize = 0;
}
```

```
/* aumentar a dimensão do vector em função de um factor multiplicado por um
tamanho */
bool CIntVector::grow( int iTimes /* = 1 */, int iStep /* = GROW_STEP */ )
{
      int  new_size = m_iSize + iStep * iTimes;

      int * pv = new int[ new_size ];

      if ( pv == NULL ) return false;

      for ( int i = 0; i < m_iSize; i++ )
            pv[ i ] = m_iVector[ i ];

      m_iSize = new_size;

      delete [] m_iVector;

      m_iVector = pv;

      return true;
}


/* alterar o valor de uma determinada posição do vector */
bool CIntVector::update( int index, int value )
{
      if ( index < 0 || index >= m_iSize )
            return false;

      m_iVector [ index ] = value;

      return true;
}


/* obtem o valor de uma posição do vector */
bool CIntVector::index(  int index, int& value )
{
      if ( index < 0 || index >= m_iSize )
            return false;

      value = m_iVector [ index ];

      return true;
}

/* procura pela primeira ocorrência de um valor no vector e retorna o índice do
vector */
int CIntVector::search( int value )
{
      for ( int index = 0; index < m_iSize; index++ )
            if ( m_iVector[ index ] == value ) return index;

      return -1;
}


/* visualiza o conteúdo do vector */
void CIntVector::showAll( void )
{
      for ( int index = 0; index < m_iSize; index++ )
            cout << index << ":" << m_iVector[ index ] << endl;
}
```