```
// LISTA DOBLE
//un ejemplo simple de lista doble para  mostrar el manejo de los punteros
//no es robusto; hay que agregar manejo de eventos  excepcionales

#include<iostream>
using namespace std;

template <class T>class ListaD;

template <class T>
class Nodo
  {
    friend class ListaD <T> ;

    public:
                        Nodo(T _v):_info(_v),_sig(0),_ant(0){}
    private:
        T _info;
        Nodo <T> *_sig, *_ant;
   };


template <class T>
class ListaD
{

  public:
    ListaD( );  //constructor
    ~ListaD( );  //destructor
    void AltaPrin (const T &);  //alta al principio
    void AltaFin (const T &);  //alta al final
    void BajaPrin( );    //borra primer nodo
    void BajaFin( );     //borra ultimo nodo
    bool Vacia( ) const; //retorna true si lista vacia
    void Emitir ( ) const;  //emite la lista
  private:
    Nodo <T> * _principio;
};


template <class T>
ListaD <T>:: ListaD( ):_principio(0){}



template <class T>
ListaD <T>:: ~ListaD( )
{
  Nodo<T> *_aux=_principio;
  while ( _principio)
    {_principio=_principio->_sig;
      delete _aux;
      _aux=_principio;
    }
}
```

```
template <class T>
void ListaD <T>::AltaPrin (const T & _v)
{
  Nodo <T> *_aux;
  _aux= new Nodo <T>(_v);
  _aux->_info=_v;
  _aux->_ant = 0;
  _aux->_sig = _principio;
  if (_principio) _principio->_ant = _aux;
  _principio = _aux;
}


template <class T>
void ListaD <T>::AltaFin (const T & _v)
{
 Nodo <T> *_aux1 , *_aux2 ;
  _aux1= new Nodo <T>(_v);
  _aux1->_info=_v;
  _aux1->_sig= 0;
  if (Vacia())
                  {
                    _principio=_aux1;
                    _principio->_ant=0;
                    }
else
          {
            _aux2=_principio;
            while(_aux2->_sig) _aux2=_aux2->_sig;
            _aux2->_sig= _aux1;
            _aux1->_ant=_aux2;
          }
}

template <class T>
void ListaD <T>::BajaPrin ( )
{
 Nodo<T> *_aux1=_principio;
 if (!Vacia( ))
                  {
                    _principio=_principio->_sig;
                    if (_principio) _principio->_ant=0;
                    delete _aux1;
                    }
}

template <class T>
void ListaD <T>::BajaFin ( )
{
 Nodo<T> *_aux=_principio;
 if (!Vacia( ))
                  {
                  if(_principio->_sig == 0)
                            {
                                                                        delete _principio;
                             _principio =0;
```

```
                    }
                else
                    {
                     while ((_aux->_sig) && (_aux->_sig->_sig)) _aux=_aux->_sig;
                     delete (_aux->_sig);
                     _aux->_sig=0;
                    }
            }

}

template <class T>
void ListaD <T>::Emitir ( ) const
{
 Nodo <T> *_aux =_principio;
 while (_aux)
 {
        cout<< _aux->_info <<endl;
        _aux=_aux->_sig;
 }
}

template <class T>
bool ListaD <T>:: Vacia ( ) const

{
return (_principio ==0);
}


void main()
{

ListaD <int> c1,c2;
c1.AltaPrin(4);
c1.AltaPrin(3);
c1.AltaFin(5);
c1.Emitir();
c2.AltaPrin(2);
c2.AltaPrin(1);
c2.AltaFin(3);
c2.Emitir();
c1.BajaPrin();
c1.Emitir();
c1.BajaFin();
c1.Emitir();

}
```