EJERCICIO 7

===Arreglo.h===

```
#include <iostream>
#include <vector>
#include <locale.h>

using namespace std;

template <typename T>
class ArregloOps
{
private:
  T sumaRecursiva(const vector<T> &v, int n);

public:
  T suma(const vector<T> &v);
};

template <typename T>
T ArregloOps<T>::sumaRecursiva(const vector<T> &v, int n)
{
  if (n < 0) //CASO BASE
  {
    return T(); // valor neutro (0 para numéricos)
  }
  else
  {
    return v[n] + sumaRecursiva(v, n - 1); //CASO RECURSIVO
  }
}

template <typename T>
T ArregloOps<T>::suma(const vector<T> &v)
{
  return sumaRecursiva(v, v.size() - 1);
}

/* ¿CUÁNDO ESTO SERÍA INFINITO?
    La recursión se volvería infinita si el índice 'n' no se reduce correctamente
en cada llamada.
    Por ejemplo, si se usa 'n + 1' en lugar de 'n - 1', o si se omite el caso base
(n < 0),
    el algoritmo entraría en un ciclo indefinido y provocaría un desbordamiento de
pila.
 ¿POR QUÉ ES UNA SOLUCIÓN NATURAL?
    La suma de los elementos de un vector puede verse como:
    suma(v) = v[n] + suma(v[0..n-1])
    Esta definición se adapta perfectamente a la recursión, ya que el problema se
descompone en una versión más pequeña de sí mismo.*/
```

===main.cpp===

```
#include "Arreglo.h"
#include <iostream>
#include <vector>
#include <locale.h>

using namespace std;

int main(){
  setlocale(LC_ALL, "spanish");
  typedef int Numeros;

  ArregloOps<int> arregloInt;
  Numeros cantidad;

  cout << "¿Cuántos números enteros desea ingresar? ";
  cin >> cantidad;

  if(cantidad <= 0){
    cout << "La cantidad debe ser un número positivo mayor que cero." << endl;
    return 1;
  }

  vector<Numeros> numeros(cantidad);

  cout << "Ingrese " << cantidad << " números enteros:" << endl;
  for(int i = 0; i < cantidad; i++){
    cin >> numeros[i];
  }

  Numeros sumaTotal = arregloInt.suma(numeros);
  cout << "La suma de los números ingresados es: " << sumaTotal << endl;

  return 0;
}
```