EJERCICIO 1:

===Matematica.h===

```
#pragma once
#include <iostream>
#include <stdexcept>
#include <string>

using namespace std;

template <typename T>
class Matematica {
  private:
  public:
    int calculoFactorial(long numero, int nivel);
};

#include "Matematica.tpp"
```

===Matematica.tpp===

```
#include "Matematica.h"
#include <iostream>
#include <stdexcept>

using namespace std;

template <typename T>
int Matematica<T>::calculoFactorial(long numero, int nivel)
{

  //El numero negativo no existe en factorial lo que ocasionaría errores
  if (numero < 0) {
    throw invalid_argument("El número no puede ser negativo.");
  }
  cout << string(nivel * 2, ' ') << "Nivel " << nivel << ": factorial de " <<
numero << endl;

  if (numero == 0 || numero == 1) { //Casos base: 0! = 1 y 1! = 1
    return 1;
  } else {
    return numero * calculoFactorial(numero - 1, nivel + 1); //Paso recuersivo
  }
}

/*CUANDO PUEDE SER INFINITA?
Cuando el numero no se reduce en cada llamado o si el caso base sea numero < 0
¿Por qué es natural?
El factorial de un número se define como n! = n * (n-1)!. Aquí podemos observar
como el factorial necesita de un factorial para continuar, por lo que se puede
emplear una función recursiva*/
```

```
===main.cpp==
#include "Matematica.h"
#include <iostream>
#include <stdexcept>
#include <string>
#include <locale.h>

using namespace std;

int main(){
  setlocale(LC_ALL, "es_ES.UTF-8"); //Para que lea caracteres latinos
  Matematica<int> matematica;
  typedef long Numero;
  typedef int Enteros;
  Numero numero;

  cout << "Ingrese un número para calcular su factorial: ";
  cin >> numero;
  try {
    Enteros resultado = matematica.calculoFactorial(numero, 0);
    cout << "El factorial de " << numero << " es: " << resultado << endl;
  } catch (const invalid_argument& e) {
    cout << "Error: " << e.what() << endl;
  }
  return 0;
}
```