EJERCICIO 5

===Factorial.h===

```
#include <iostream>
#include <locale>
#include <string>
#include <stdexcept>

using namespace std;

class Factorial
{
public:
  long factorial(long n, int nivel = 0);
};

long Factorial::factorial(long n, int nivel)
{

  if (n < 0)
  {
    throw std::invalid_argument("El número no puede ser un negativo.");
  }

  cout << string(nivel * 2, ' ') << "Ingresando el factorial de:(" << n << ")" <<
endl;

  long resultado;
  if (n == 0 || n == 1) //CASO BASE: tanto 0 y 1 comparten un mismo resultado
  {
    resultado = 1;
  }
  else
  {
    resultado = n * factorial(n - 1, nivel + 1); //Paso recursivo
  }

  // Mostrar salida de la función
  cout << string(nivel * 2, ' ') << "Resultado del factorial de : " << n << " = "
<< resultado << endl;
  return resultado;
}

/* ¿CUÁNDO ESTO SERÍA INFINITO?
  La recursión se volvería infinita si el valor de 'n' no se aproxima al caso base
en cada llamada.
  Por ejemplo, si se incrementa en lugar de disminuir, o si se omite la condición
de parada,
  el algoritmo entraría en un ciclo indefinido y provocaría un desbordamiento de
pila.
```

```
  ¿POR QUÉ ES UNA SOLUCIÓN 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 "Factorial.h"
#include <iostream>
#include <locale.h>
#include <stdexcept>

using namespace std;

int main(){
  setlocale(LC_ALL, "es_ES.UTF-8");

  Factorial factorial;
  long numero;

  cout << "Ingrese un número para calcular su factorial: ";
  cin >> numero;

  try{
    long resultado = factorial.factorial(numero);
     cout << "El factorial de " << numero << " es: " << resultado << endl;
  } catch (const invalid_argument& e) {
     cout << "Error: " << e.what() << endl;
  }

  return 0;
}
```