



UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE
ÁREA DE CIENCIAS DE LA COMPUTACION
ESTRUCTURA DE DATOS

NRC:23230

Fecha: 5 de junio de 2025

Autor: Esteban Quiroga

MANUAL DE EJERCICIOS ESTRUCTURA DE DATOS EN C++
Ejercicios Extras

Enunciado

1) Realice un programa para calcular una serie geométrica, en el que se pida el primer término, la razón y el número total de términos.

$$\sum_{n=0}^{\infty} ar^n = a + ar + ar^2 + ar^3 + \dots$$

1.1 Objetivo

Calcular la serie geométrica dados los parámetros ingresados por el usuario.

1.2 Procedimiento

- Pedir al usuario los parámetros: primer término, razón, número de términos a calcular.
- Calcular la serie con una función recursiva.

1.3 Ejemplos

Primer término a: 2

Razón r: 2

Número de términos n: 3

$$\sum_{n=0}^3 2 \cdot 2^n = 2 + 4 + 8 + 16 = 30$$

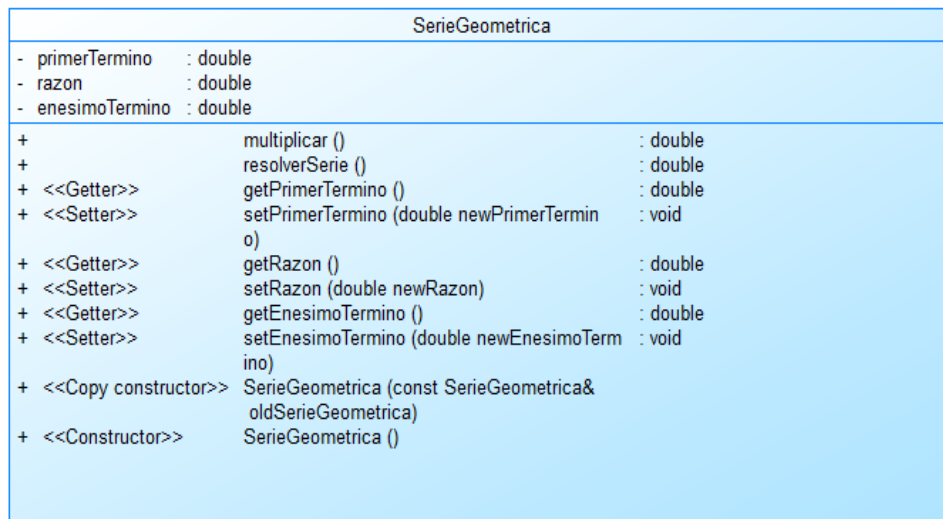
Primer término a: 3

Razón r: 2

Número de términos n: 3

$$\sum_{n=0}^3 3 \cdot 2^n = 3 + 6 + 12 + 24 = 45$$

1.4 Diagrama de clases (Power Designer)



1.5 Clase SerieGeometrica.h

```
1 #pragma once
2
3 class SerieGeometrica
4 {
5 public:
6     SerieGeometrica(double primerTermino, double razon, double enesimoTermino);
7     double multiplicar(int exponente);
8     double resolverSerie(int);
9     void setPrimerTermino(double valor);
10    void setRazon(double valor);
11    void setEnesimoTermino(double valor);
12    double getPrimerTermino() const;
13    double getRazon() const;
14    double getEnesimoTermino() const;
15
16 protected:
17 private:
18     double primerTermino;
19     double razon;
20     double enesimoTermino;
21 };
```

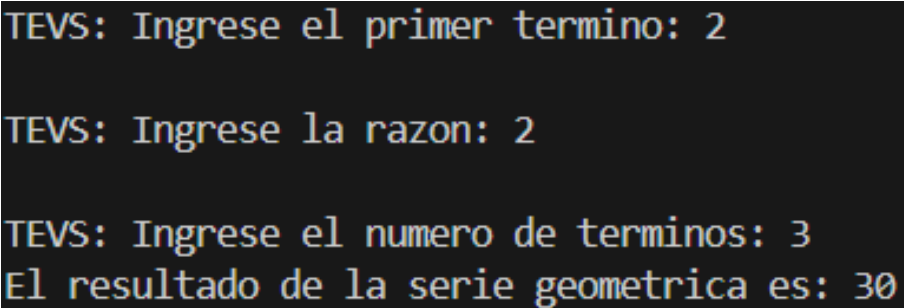
1.6 Clase SerieGeometrica.cpp

```
1 #include "SerieGeometrica.h"
2 #include <cmath>
3
4 SerieGeometrica::SerieGeometrica(double primerTermino, double razon, double
   enesimoTermino)
5 : primerTermino(primerTermino), razon(razon), enesimoTermino(enesimoTermino){}
6
7 double SerieGeometrica::multiplicar(int exponente)
8 {
9     return primerTermino * pow(razon, exponente);
10 }
11
12 double SerieGeometrica::resolverSerie(int exponente)
13 {
14     if(exponente == enesimoTermino){
15         return multiplicar(exponente);
16     } else {
17         return multiplicar(exponente) + resolverSerie(exponente + 1);
18     }
19 }
20
21 void SerieGeometrica::setPrimerTermino(double valor) {
22     primerTermino = valor;
23 }
24
25 void SerieGeometrica::setRazon(double valor) {
26     razon = valor;
27 }
28
29 void SerieGeometrica::setEnesimoTermino(double valor) {
30     enesimoTermino = valor;
31 }
32
33 // Getters
34 double SerieGeometrica::getPrimerTermino() const {
35     return primerTermino;
36 }
37
38 double SerieGeometrica::getRazon() const {
39     return razon;
40 }
41
42 double SerieGeometrica::getEnesimoTermino() const {
43     return enesimoTermino;
44 }
45 }
```

1.7 Clase main.cpp

```
1 #include <iostream>
2 #include "SerieGeometrica.h"
3 #include "../ValidarDatos.h"
4
5 int main() {
6     double primerTermino, razon, enesimoTermino;
7     primerTermino = ValidarDatos::validarDouble("Ingrese el primer termino: ");
8     razon = ValidarDatos::validarDouble("Ingrese la razon: ");
9     enesimoTermino = ValidarDatos::validarEntero("Ingrese el numero de terminos: ");
10    SerieGeometrica serie(primerTermino, razon, enesimoTermino);
11    double resultado = serie.resolverSerie(0);
12    std::cout << "El resultado de la serie geometrica es: " << resultado << std::endl;
13    return 0;
14 }
```

1.8 Ejecución del Programa



```
TEVS: Ingrese el primer termino: 2
TEVS: Ingrese la razon: 2
TEVS: Ingrese el numero de terminos: 3
El resultado de la serie geometrica es: 30
```

Figura 1: Ejecución de la serie armónica en consola

Ejercicio 2: Serie Armónica

2.1 Enunciado

Realice un programa para calcular la suma de la serie armónica, en el que se pida el número total de términos a calcular.

$$\sum_{n=1}^N \frac{1}{n} = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{N}$$

2.2 Objetivo

Calcular la suma de la serie armónica dado el número de términos ingresado por el usuario.

2.3 Procedimiento

- Pedir al usuario el número total de términos a calcular.
- Calcular la suma de la serie armónica con una función recursiva.

2.4 Ejemplos

Número de términos n: 3

$$\sum_{n=1}^3 \frac{1}{n} = 1 + \frac{1}{2} + \frac{1}{3} \approx 1,833$$

Número de términos n: 5

$$\sum_{n=1}^5 \frac{1}{n} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} \approx 2,283$$

2.5 Diagrama de clases (Power Designer)

SerieArmonica		
-	enesimoTermino	: double
+	resolverSerie (int contador)	: double
+	<<Getter>> getEnesimoTermino ()	: double
+	<<Setter>> setEnesimoTermino (double newEnesimoTerm ino)	: void
+	<<Copy constructor>> SerieArmonica (const SerieArmonica& oldSerieArmonica)	

2.6 Clase SerieArmonica.h

```

1 #pragma once
2
3 class SerieArmonica
4 {
5 public:
6     SerieArmonica(double enesimoTermino);
7     double resolverSerie(int contador);
8     void setEnesimoTermino(double valor);
9     double getEnesimoTermino() const;
10
11 protected:
12 private:
13     double enesimoTermino;
14 };

```

2.7 Clase SerieArmonica.cpp

```

1 #include "SerieArmonica.h"
2 #include <iostream>
3
4 using namespace std;
5
6 SerieArmonica::SerieArmonica(double enesimoTermino)
7 : enesimoTermino(enesimoTermino) {}
8
9 double SerieArmonica::resolverSerie(int contador) {
10     if (contador == enesimoTermino) {
11         return 1.0 / contador;
12     } else if (contador < enesimoTermino) {
13         return 1.0 / contador + resolverSerie(contador + 1);
14     }
15     return 0; // en caso de error o fin de recursi n
16 }
17
18 void SerieArmonica::setEnesimoTermino(double valor) {
19     enesimoTermino = valor;
20 }
21
22 double SerieArmonica::getEnesimoTermino() const {
23     return enesimoTermino;
24 }

```

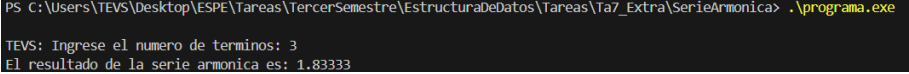
2.8 Clase main.cpp

```

1 #include <iostream>
2 #include "SerieArmonica.h"
3 #include "../ValidarDatos.h"
4
5 int main() {
6     double enesimoTermino;
7     enesimoTermino = ValidarDatos::validarEntero("Ingrese el numero de terminos: ");
8     SerieArmonica serie(enesimoTermino);
9     double resultado = serie.resolverSerie(1);
10     std::cout << "El resultado de la serie armonica es: " << resultado << std::endl;
11     return 0;
12 }

```

2.9 Ejecución del Programa



```
PS C:\Users\TEVS\Desktop\ESPE\Tareas\TercerSemestre\EstructuraDeDatos\Tareas\Ta7_Extra\SerieArmonica> .\programa.exe
TEVS: Ingrese el numero de terminos: 3
El resultado de la serie armonica es: 1.83333
```

Figura 2: Ejecución de la serie armónica en consola

Ejercicio 3: Serie Aritmético-Geométrica

3.1 Enunciado

Realice un programa para calcular la suma de una serie aritmético-geométrica, en la que se pida el número total de términos a calcular.

$$\sum_{n=1}^N n \cdot r^n = 1 \cdot r^1 + 2 \cdot r^2 + 3 \cdot r^3 + \dots + N \cdot r^N$$

3.2 Objetivo

Calcular la suma de una serie aritmético-geométrica dado el número de términos ingresado por el usuario.

3.3 Procedimiento

- Pedir al usuario el número total de términos a calcular.
- Calcular la suma de la serie aplicando recursividad.

3.4 Ejemplo

Número de términos n: 3, razón r = 2

$$\sum_{n=1}^3 n \cdot 2^n = 1 \cdot 2 + 2 \cdot 4 + 3 \cdot 8 = 2 + 8 + 24 = 34$$

3.5 Diagrama de clases (Power Designer)

SerieAritmeticoGeometrica		
-	enesimoTermino	: double
+	resolverSerie (enesimoTermino int)	: double
+	<<Constructor>> SerieAritmeticoGeometrica ()	
+	<<Getter>> getEnesimoTermino ()	: double
+	<<Setter>> setEnesimoTermino (double newEnesimoTerm ino)	: void

3.6 Clase SerieAritmeticoGeometrica.h

```

1 #pragma once
2
3 class SerieAritmeticoGeometrica
4 {
5 public:
6     SerieAritmeticoGeometrica(double enesimoTermino);
7     double resolverSerie(int);
8     void setEnesimoTermino(double valor);
9     double getEnesimoTermino() const;
10
11 protected:
12 private:
13     double enesimoTermino;
14 };

```

3.7 Clase SerieAritmeticoGeometrica.cpp

```

1 #include "SerieAritmeticoGeometrica.h"
2 #include <iostream>
3 #include <cmath>
4
5 using namespace std;
6
7 SerieAritmeticoGeometrica::SerieAritmeticoGeometrica(double enesimoTermino)
8 : enesimoTermino(enesimoTermino) {}
9
10 double SerieAritmeticoGeometrica::resolverSerie(int contador) {
11     if (contador == enesimoTermino) {
12         return contador * pow(2, contador);
13     } else if (contador < enesimoTermino) {
14         return contador * pow(2, contador) + resolverSerie(contador + 1);
15     }
16     return 0;
17 }
18
19 void SerieAritmeticoGeometrica::setEnesimoTermino(double valor) {
20     enesimoTermino = valor;
21 }
22
23 double SerieAritmeticoGeometrica::getEnesimoTermino() const {
24     return enesimoTermino;
25 }

```

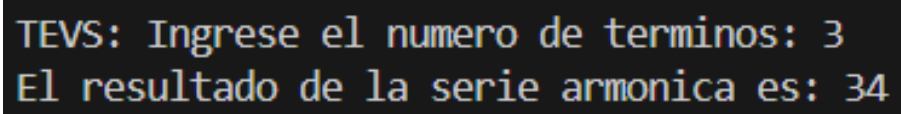
3.8 Clase main.cpp

```

1 #include <iostream>
2 #include "SerieAritmeticoGeometrica.h"
3 #include "../ValidarDatos.h"
4
5 int main() {
6     double enesimoTermino;
7     enesimoTermino = ValidarDatos::validarEntero("Ingrese el numero de terminos: ");
8     SerieAritmeticoGeometrica serie(enesimoTermino);
9     double resultado = serie.resolverSerie(1);
10     std::cout << "El resultado de la serie aritmetico geometrica es: " << resultado <<
11         std::endl;
12     return 0;
13 }

```

3.9 Ejecución del Programa



```
TEVS: Ingrese el numero de terminos: 3
El resultado de la serie armonica es: 34
```

Figura 3: Ejecución de la serie aritmético-geométrica en consola

Ejercicio 4: Serie de Fibonacci

4.1 Enunciado

Realice un programa que imprima los términos de la serie de Fibonacci hasta una cantidad dada por el usuario.

$$F(n) = \begin{cases} 0, & \text{si } n = 0 \\ 1, & \text{si } n = 1 \\ F(n-1) + F(n-2), & \text{si } n > 1 \end{cases}$$

4.2 Objetivo

Mostrar en pantalla los primeros n términos de la serie de Fibonacci, donde n es proporcionado por el usuario.

4.3 Procedimiento

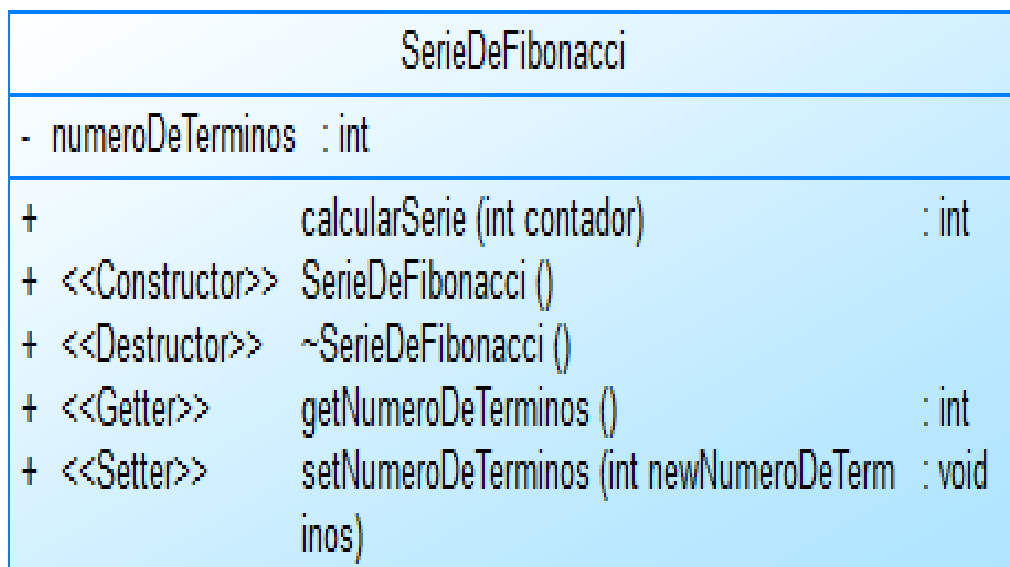
- Solicitar al usuario el número de términos a imprimir.
- Generar e imprimir la serie de Fibonacci desde el primer término hasta el n -ésimo.

4.4 Ejemplo

Número de términos n : 7

- Salida esperada: 0, 1, 1, 2, 3, 5, 8

4.5 Diagrama de clases (Power Designer)



4.6 Clase SerieDeFibonacci.h

```
1 #if !defined(__DiagramasTa7_SerieDeFibonacci_h)
2 #define __DiagramasTa7_SerieDeFibonacci_h
3
4 class SerieDeFibonacci
```

```

5 {
6 public:
7     void calcularSerie(int a = 0, int b = 1, int count = 0);
8     SerieDeFibonacci(int newNumeroDeTerminos);
9     ~SerieDeFibonacci();
10    int getNumeroDeTerminos(void);
11    void setNumeroDeTerminos(int newNumeroDeTerminos);
12
13 protected:
14 private:
15     int numeroDeTerminos;
16 };
17
18 #endif

```

4.7 Clase SerieDeFibonacci.cpp

```

1 #include "SerieDeFibonacci.h"
2 #include <iostream>
3 using namespace std;
4
5 void SerieDeFibonacci::calcularSerie(int a, int b, int count)
6 {
7     if (count < numeroDeTerminos) {
8         cout << a << " ";
9         calcularSerie(b, a + b, count + 1);
10    }
11 }
12
13 SerieDeFibonacci::SerieDeFibonacci(int newNumeroDeTerminos)
14 : numeroDeTerminos(newNumeroDeTerminos)
15 {
16 }
17
18 SerieDeFibonacci::~SerieDeFibonacci()
19 {
20     // TODO : implement
21 }
22
23 int SerieDeFibonacci::getNumeroDeTerminos(void)
24 {
25     return numeroDeTerminos;
26 }
27
28 void SerieDeFibonacci::setNumeroDeTerminos(int newNumeroDeTerminos)
29 {
30     numeroDeTerminos = newNumeroDeTerminos;
31 }

```

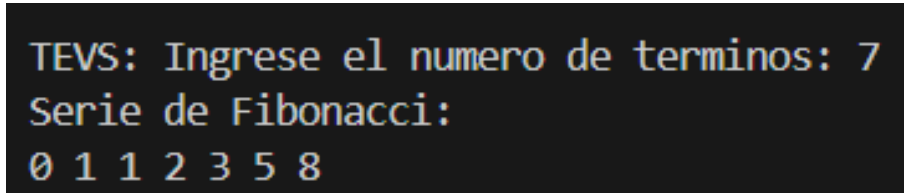
4.8 Clase main.cpp

```

1 #include "SerieDeFibonacci.h"
2 #include "../ValidarDatos.h"
3 #include <iostream>
4
5 using namespace std;
6
7 int main()
8 {
9     int numeroDeTerminos;
10    numeroDeTerminos = ValidarDatos::validarEntero("Ingrese el numero de terminos: ");
11
12    SerieDeFibonacci serie(numeroDeTerminos);
13    cout << "Serie de Fibonacci: " << endl;
14    serie.calcularSerie();
15    return 0;

```

4.9 Ejecución del Programa



```
TEVS: Ingrese el numero de terminos: 7
Serie de Fibonacci:
0 1 1 2 3 5 8
```

Figura 4: Ejecución de la serie de Fibonacci en consola

Ejercicio 5: Serie Logarítmica

5.1 Enunciado

Realice un programa que calcule la suma de una serie logarítmica, solicitando al usuario el número de términos y el valor de a .

$$a \sum_{i=1}^n \lg i$$

5.2 Objetivo

Calcular la suma de una serie logarítmica definida por el usuario, proporcionando la cantidad de términos y el valor de a .

5.3 Procedimiento

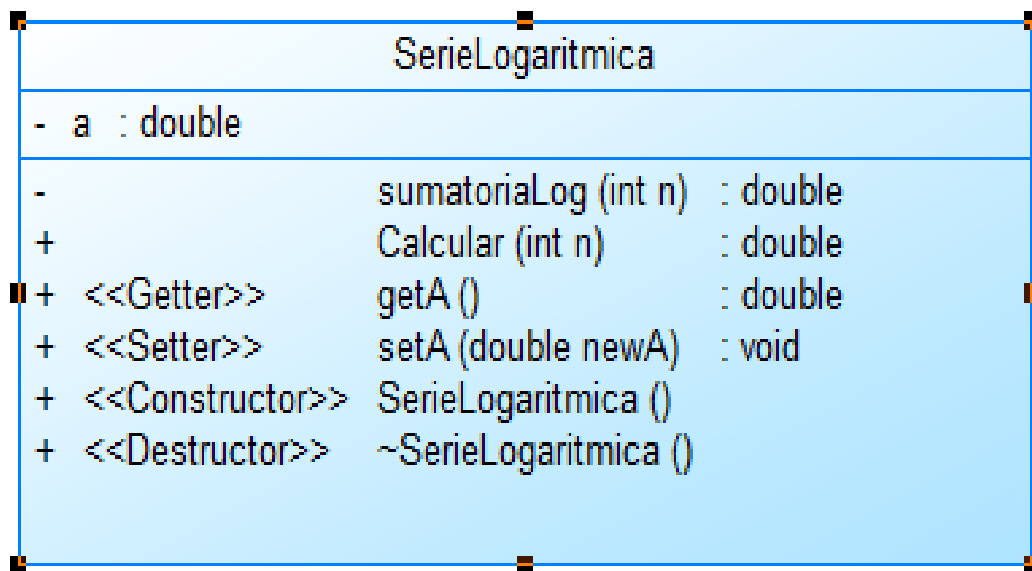
- Solicitar al usuario el número de términos n , donde $n > 0$.
- Solicitar el valor de a .
- Calcular la suma de la serie logarítmica con los datos proporcionados.

5.4 Ejemplo

Entrada: $n = 5$, $a = 2$

- Salida esperada: El resultado de la serie logarítmica es aproximadamente 4.15836

5.5 Diagrama de clases (Power Designer)



5.6 Clase SerieLogaritmica.h

```

1 #if !defined(__Class_Diagram_2_SerieLogaritmica_h)
2 #define __Class_Diagram_2_SerieLogaritmica_h
3
4 class SerieLogaritmica
5 {
6 public:
7     double Calcular(int n);
8     double getA(void);
9     void setA(double newA);
10    SerieLogaritmica();
11    ~SerieLogaritmica();
12
13 protected:
14 private:
15     double sumatoriaLog(int n);
16
17     double a;
18 };
19
20 #endif

```

5.7 Clase SerieLogaritmica.cpp

```

1 #include "SerieLogaritmica.h"
2 #include <cmath>
3
4 double SerieLogaritmica::sumatoriaLog(int n)
5 {
6     if (n == 1)
7         return log10(1);
8     else
9         return log10(n) + sumatoriaLog(n - 1);
10 }
11
12 double SerieLogaritmica::Calcular(int n)
13 {
14     return a * sumatoriaLog(n);
15 }
16
17 double SerieLogaritmica::getA(void)
18 {
19     return a;
20 }
21
22 void SerieLogaritmica::setA(double newA)
23 {
24     a = newA;
25 }
26
27 SerieLogaritmica::SerieLogaritmica() {}
28
29 SerieLogaritmica::~SerieLogaritmica() {}

```

5.8 Clase main.cpp

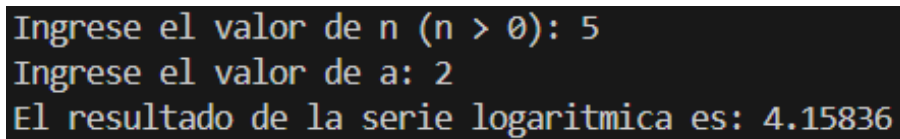
```

1 #include <iostream>
2 #include "SerieLogaritmica.h"
3 using namespace std;
4
5 int main() {
6     SerieLogaritmica serie;
7     int n;
8     double a;
9
10    cout << "Ingrese el valor de n (n > 0): ";

```

```
11     cin >> n;
12     if (n <= 0) {
13         cout << "Error: n debe ser mayor que 0." << endl;
14         return 1;
15     }
16
17     cout << "Ingrese el valor de a: ";
18     cin >> a;
19     serie.setA(a);
20
21     double resultado = serie.Calcular(n);
22     cout << "El resultado de la serie logaritmica es: " << resultado << endl;
23
24     return 0;
25 }
```

5.9 Ejecución del Programa



Ingrese el valor de n (n > 0): 5
Ingrese el valor de a: 2
El resultado de la serie logaritmica es: 4.15836

Figura 5: Ejecución de la serie logarítmica en consola

Ejercicio 6: Integral Exponencial

6.1 Enunciado

Realice un programa que calcule el valor de la siguiente integral definida:

$$\int_0^n e^{ax} dx$$

6.2 Objetivo

Calcular la integral definida de la función exponencial e^{ax} desde 0 hasta un valor n , con a ingresado por el usuario.

6.3 Procedimiento

- Solicitar al usuario el valor del límite superior n .
- Solicitar el valor del coeficiente a .
- Evaluar la integral utilizando la fórmula:

$$\int_0^n e^{ax} dx = \frac{1}{a}(e^{an} - 1)$$

6.4 Ejemplo

Entrada: $n = 2$, $a = 3$

- Resultado esperado: $\frac{1}{3}(e^6 - 1) \approx 134,142$

6.5 Diagrama de clases (Power Designer)

IntegralExponencial		
-	a : double	
-	n : double	
+	<<Constructor>> IntegralExponencial ()	
+	<<Destructor>> ~IntegralExponencial ()	
+	setDatos ()	: void
+	resolver ()	: double

6.6 Clase IntegralExponencial.h

```

1 #ifndef INTEGRALEXPONENCIAL_H
2 #define INTEGRALEXPONENCIAL_H
3
4 class IntegralExponencial {
5 private:
6     double a, n;
7 public:
8     IntegralExponencial();
9     ~IntegralExponencial();
10    void setDatos(double, double);
11    double resolver();
12 };
13
14 #endif

```

6.7 Clase IntegralExponencial.cpp

```

1 #include "IntegralExponencial.h"
2 #include <cmath>
3
4 IntegralExponencial::IntegralExponencial() : a(0), n(0) {}
5
6 IntegralExponencial::~IntegralExponencial()
7 {
8 }
9
10 void IntegralExponencial::setDatos(double a, double n) {
11     this->a = a;
12     this->n = n;
13 }
14
15 double IntegralExponencial::resolver() {
16     return (1.0 / a) * (exp(a * n) - 1);
17 }

```

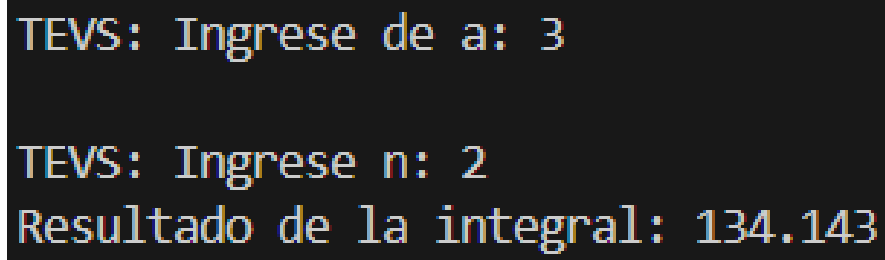
6.8 Clase main.cpp

```

1 #include <iostream>
2 #include "IntegralExponencial.h"
3 #include "../ValidarDatos.h"
4
5 int main() {
6     IntegralExponencial integral;
7     double a, n;
8
9     a = ValidarDatos::validarDouble("Ingreso de a: ");
10    n = ValidarDatos::validarDouble("Ingreso n: ");
11
12    integral.setDatos(a, n);
13    std::cout << "Resultado de la integral: " << integral.resolver() << std::endl;
14
15    return 0;
16 }

```

6.9 Ejecución del Programa



```
TEVS: Ingrese de a: 3
TEVS: Ingrese n: 2
Resultado de la integral: 134.143
```

Figura 6: Ejecución del cálculo de la integral exponencial en consola

Ejercicio 7: Integral de Potencia

7.1 Enunciado

Realice un programa que calcule el valor de la siguiente integral definida:

$$\int_0^n x^k dx = \frac{1}{k+1} n^{k+1}$$

7.2 Objetivo

Calcular la integral definida de la función potencia x^k desde 0 hasta un valor n , con k ingresado por el usuario.

7.3 Procedimiento

- Solicitar al usuario el valor del límite superior n .
- Solicitar el valor del exponente k .
- Evaluar la integral utilizando la fórmula:

$$\int_0^n x^k dx = \frac{1}{k+1} n^{k+1}$$

7.4 Ejemplo

Entrada: $n = 2$, $k = 3$

- Resultado esperado: $\frac{1}{4}(2)^4 = \frac{16}{4} = 4$

7.5 Diagrama de clases (Power Designer)

IntegralPotencia		
-	<code>n</code> : double	
-	<code>k</code> : int	
+	<<Constructor>> IntegralPotencia ()	
+	<<Destructor>> ~IntegralPotencia ()	
+	setDatos ()	: void
+	resolver ()	: double

7.6 Clase IntegralPotencia.h

```

1 #ifndef INTEGRALPOTENCIA_H
2 #define INTEGRALPOTENCIA_H
3
4 class IntegralPotencia {
5 private:
6     double n;
7     int k;
8 public:
9     IntegralPotencia();
10    ~IntegralPotencia();
11    void setDatos(double, int);
12    double resolver();
13 };
14
15 #endif

```

7.7 Clase IntegralPotencia.cpp

```

1 #include "IntegralPotencia.h"
2 #include <cmath>
3
4 IntegralPotencia::IntegralPotencia() : n(0), k(0) {}
5
6 IntegralPotencia::~~IntegralPotencia()
7 {
8 }
9
10 void IntegralPotencia::setDatos(double n, int k) {
11     this->n = n;
12     this->k = k;
13 }
14
15 double IntegralPotencia::resolver() {
16     return (1.0 / (k + 1)) * pow(n, k + 1);
17 }

```

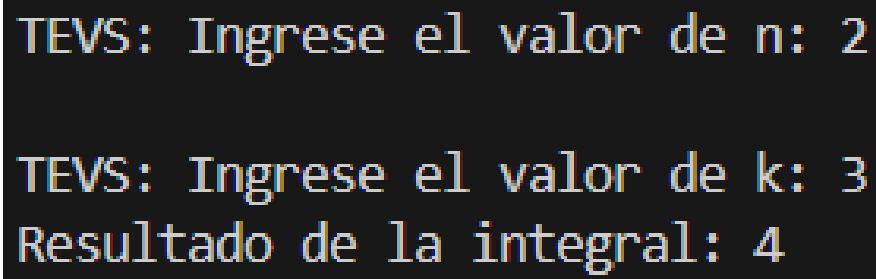
7.8 Clase main.cpp

```

1 #include <iostream>
2 #include "IntegralPotencia.h"
3 #include "../ValidarDatos.h"
4
5 int main() {
6     IntegralPotencia integral;
7     double n;
8     double k;
9
10    n = ValidarDatos::validarDouble("Ingrese el valor de n: ");
11    k = ValidarDatos::validarDouble("Ingrese el valor de k: ");
12
13    integral.setDatos(n, k);
14    std::cout << "Resultado de la integral: " << integral.resolver() << std::endl;
15
16    return 0;
17 }

```

7.9 Ejecución del Programa



```
TEVS: Ingrese el valor de n: 2
TEVS: Ingrese el valor de k: 3
Resultado de la integral: 4
```

Figura 7: Ejecución del cálculo de la integral de potencia en consola

Ejercicio 8: Sumatoria Aritmética con 3 Métodos

8.1 Enunciado

Realice un programa que calcule la suma de los números enteros desde 1 hasta n , ingresado por el usuario, utilizando tres métodos: fórmula, ciclo y recursividad.

$$\sum_{i=1}^n i = \frac{n(n+1)}{2}$$

8.2 Objetivo

Comparar tres métodos para calcular la sumatoria aritmética desde 1 hasta n : usando fórmula directa, iteración y recursividad.

8.3 Procedimiento

- Solicitar al usuario un número entero positivo n .
- Calcular la sumatoria mediante la fórmula: $\sum_{i=1}^n i = \frac{n(n+1)}{2}$
- Calcular la sumatoria mediante un ciclo.
- Calcular la sumatoria mediante recursividad.
- Mostrar los tres resultados y compararlos.

8.4 Ejemplo

Entrada: $n = 5$

- Por fórmula: $\frac{5(5+1)}{2} = 15$
- Por sumatoria: $1 + 2 + 3 + 4 + 5 = 15$
- Por recursividad: $1 + 2 + 3 + 4 + 5 = 15$

8.5 Diagrama de clases (Power Designer)

SerieAritmetica		
-	n	: int
+	calcularSumatoriaFormula ()	: int
+	calcularSumatoria ()	: int
+	calcularSumatoriaRecursividad ()	: int
+	<<Getter>> getN ()	: int
+	<<Setter>> setN (int newN)	: void

8.6 Clase Serie.h

```
1 #pragma once
2
3 class Serie
4 {
5 public:
6     int getN(void);
7     void setN(int newN);
8     Serie();
9     ~Serie();
10    int CalcularSumatoriaFormula(void);
11    int CalcularSumatoria(void);
12    int CalcularSumatoriaRecursividad(int param);
13
14 protected:
15 private:
16     int n;
17 };
```

8.7 Clase Serie.cpp

```
1 #include "Serie.h"
2
3 int Serie::getN(void) {
4     return n;
5 }
6
7 void Serie::setN(int newN) {
8     n = newN;
9 }
10
11 Serie::Serie() {}
12
13 Serie::~Serie() {}
14
15 int Serie::CalcularSumatoriaFormula(void){
16     return n * (n + 1) / 2;
17 }
18
19 int Serie::CalcularSumatoria(void){
20     int aux = 0;
21     for (int i = 1; i <= n; i++){
22         aux = aux + i;
23     }
24     return aux;
25 }
26
27 int Serie::CalcularSumatoriaRecursividad(int param) {
28     if (param == 1) {
29         return 1;
30     } else {
31         return param + CalcularSumatoriaRecursividad(param - 1);
32     }
33 }
```

8.8 Clase main.cpp

```
1 #include <iostream>
2 #include "Serie.h"
3 using namespace std;
4
5 int main() {
6     Serie serie;
7     int numero;
8 }
```



```

9      cout << "Ingrese un numero entero positivo: ";
10     cin >> numero;
11
12     serie.setN(numero);
13     int resultado = serie.CalcularSumatoriaFormula();
14     int result = serie.CalcularSumatoria();
15     int resultadoRecursivo = serie.CalcularSumatoriaRecursividad(numero);
16
17     cout << "La suma de los enteros del 1 al " << numero << " por formula es: " <<
18         resultado << endl;
19     cout << "\nLa suma de los enteros del 1 al " << numero << " por sumatoria es: " <<
20         result << endl;
21     cout << "\nLa suma de los enteros del 1 al " << numero << " por recursividad es: " <<
22         resultadoRecursivo << endl;
23
24     return 0;
25 }

```

8.9 Ejecución del Programa

```

Ingrese un numero entero positivo: 5
La suma de los enteros del 1 al 5 por formula es: 15

La suma de los enteros del 1 al 5 por sumatoria es: 15

La suma de los enteros del 1 al 5 por recursividad es: 15

```

Figura 8: Ejecución del cálculo de la sumatoria aritmética por tres métodos

Ejercicio 9: Serie Polinómica

9.1 Enunciado

Realice un programa que calcule la suma de una serie polinómica de la forma $\sum_{i=1}^n i^k$, donde n y k son ingresados por el usuario.

9.2 Objetivo

Calcular la suma de los números elevados a una potencia k , desde 1 hasta n , usando una función recursiva.

9.3 Procedimiento

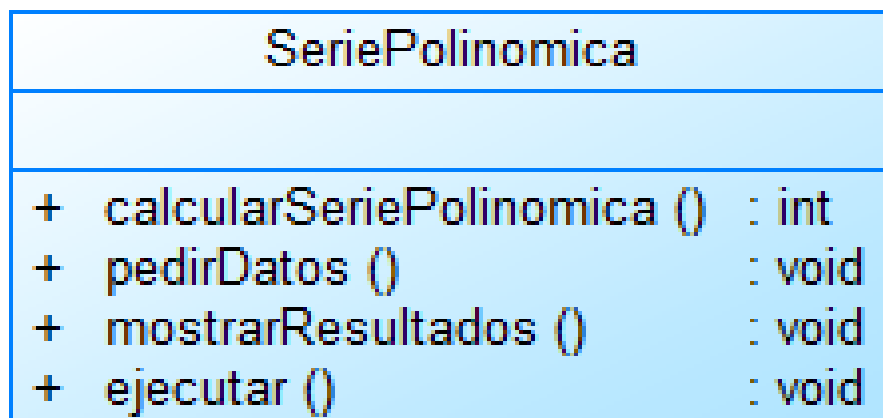
- Solicitar al usuario el valor de n (hasta dónde se suma).
- Solicitar el valor de k (potencia de cada i).
- Calcular la suma $\sum_{i=1}^n i^k$.
- Mostrar el resultado.

9.4 Ejemplo

Entrada: $n = 5$, $k = 2$

- Resultado esperado: $1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 55$

9.5 Diagrama de clases (Power Designer)



9.6 Clase SeriePolinomica.h

```
1 #pragma once
2 class SeriePolinomica
3 {
4 public:
```

```

5     void ejecutar(void);
6
7 protected:
8 private:
9     int calcularSeriePolinomial(int n, int k);
10    void pedirDatos(int& n, int& k);
11    void mostrarResultados(int resultado);
12 };

```

9.7 Clase SeriePolinomial.cpp

```

1 #include "../Model/SeriePolinomial.h"
2 #include <iostream>
3 #include <cmath>
4 using namespace std;
5
6 void SeriePolinomial::ejecutar(void)
7 {
8     int n, k;
9     pedirDatos(n, k);
10    int resultado = calcularSeriePolinomial(n, k);
11    mostrarResultados(resultado);
12 }
13
14 int SeriePolinomial::calcularSeriePolinomial(int n, int k)
15 {
16     if (n == 0) {
17         return 0;
18     }
19     return potencia(n, k) + calcularSeriePolinomial(n - 1, k);
20 }
21
22 void SeriePolinomial::pedirDatos(int& n, int& k)
23 {
24     cout << "Ingrese el valor de n (hasta donde se suma): ";
25     cin >> n;
26     cout << "Ingrese el valor de k (potencia de cada i): ";
27     cin >> k;
28 }
29
30 void SeriePolinomial::mostrarResultados(int resultado)
31 {
32     cout << "El resultado de la serie polinomial es: " << resultado << endl;
33 }
34
35 int SeriePolinomial::potencia(int base, int exponente) {
36     int resultado = 1;
37     for (int i = 0; i < exponente; ++i) {
38         resultado *= base;
39     }
40     return resultado;
41 }

```

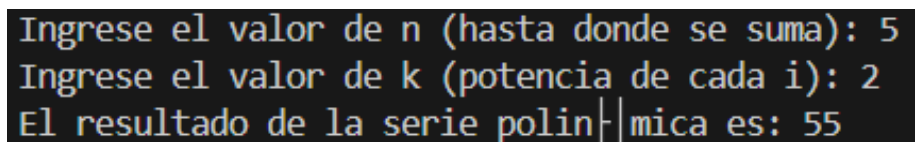
9.8 Clase main.cpp

```

1 #include "SeriePolinomial.h"
2
3 int main() {
4     SeriePolinomial serie;
5     serie.ejecutar();
6     return 0;
7 }

```

9.9 Ejecución del Programa



```
Ingrese el valor de n (hasta donde se suma): 5
Ingrese el valor de k (potencia de cada i): 2
El resultado de la serie polinómica es: 55
```

Figura 9: Ejecución del cálculo de la serie polinómica en consola

Ejercicio 10: Serie de potencias de dos

10.1 Enunciado

Realice un programa que calcule la suma de una serie de potencias de dos, según la siguiente fórmula:

$$\sum_{i=0}^k 2^i = 2^{k+1} - 1$$

10.2 Objetivo

Evaluar una serie de potencias de 2 desde el exponente 0 hasta k , ingresado por el usuario, y mostrar el resultado, aplicando una función recursiva.

10.3 Procedimiento

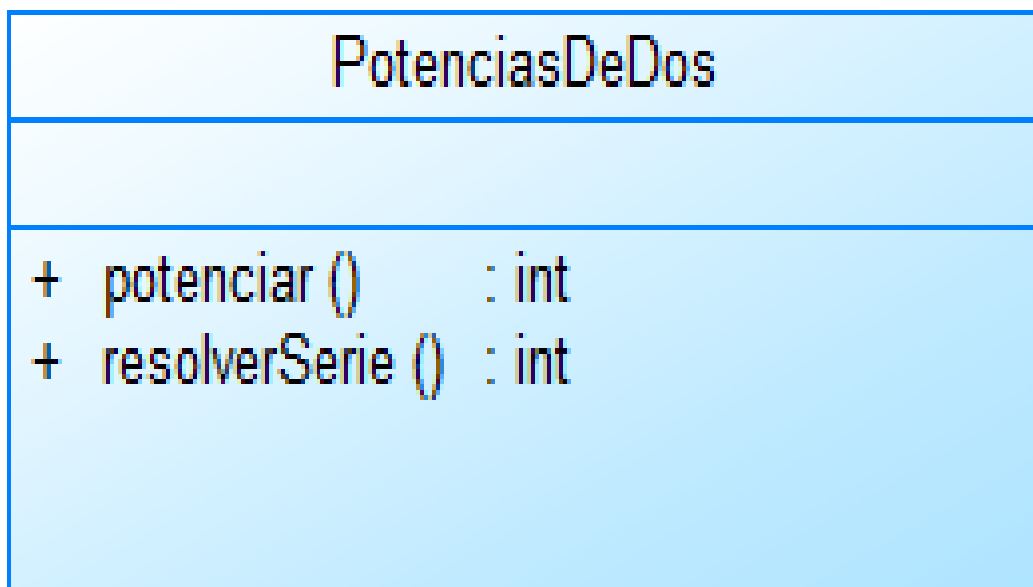
- Solicitar al usuario el valor de k (índice final).
- Calcular la suma de potencias de dos: $\sum_{i=0}^k 2^i$.
- Mostrar el resultado obtenido.

10.4 Ejemplo

Entrada: $k = 4$

- Resultado esperado: $2^0 + 2^1 + 2^2 + 2^3 + 2^4 = 1 + 2 + 4 + 8 + 16 = 31$
- Fórmula directa: $2^{4+1} - 1 = 32 - 1 = 31$

10.5 Diagrama de clases (Power Designer)



10.6 Clase PotenciasDeDos.h

```

1 #pragma once
2 class PotenciasDeDos
3 {
4 public:
5
6     static int potenciar(int, int);
7     static int resolverSerie(int);
8
9 protected:
10 private:
11
12
13 };

```

10.7 Clase PotenciasDeDos.cpp

```

1 #include "../Model/PotenciasDeDos.h"
2
3
4 int PotenciasDeDos::potenciar(int potencia, int base)
5 {
6     int result = 1;
7     for(int i = 0; i < potencia; i++){
8         result = base*result;
9     }
10    return result;
11 }
12
13 int PotenciasDeDos::resolverSerie(int indiceFinal)
14 {
15     if (indiceFinal == 0)
16         return potenciar(0, 2);
17     else
18         return potenciar(indiceFinal, 2) + resolverSerie(indiceFinal - 1);
19 }

```

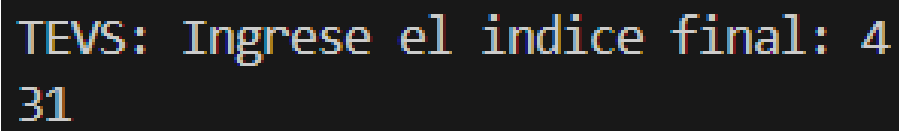
10.8 Clase main.cpp

```

1 #include <iostream>
2 #include "../Model/PotenciasDeDos.h"
3 #include "../Model/ValidarDatos.h"
4
5
6 using namespace std;
7
8 int main()
9 {
10     int indiceFinal = ValidarDatos::validarNumeros("Ingrese el indice final: ");
11     int resultado = PotenciasDeDos::resolverSerie(indiceFinal);
12
13     cout<<resultado;
14 }

```

10.9 Ejecución del Programa



```
TEVS: Ingrese el indice final: 4
31
```

Figura 10: Ejecución del cálculo de la serie de potencias de dos

Anexo: Clase ValidarDatos

La clase ValidarDatos es utilizada en todos los ejercicios anteriores para validar las entradas del usuario.

0.1. ValidarDatos.h

```
1 #ifndef ValidarDatos_h
2 #define ValidarDatos_h
3
4 class ValidarDatos {
5 public:
6     static int validarEntero(char msj[20]);
7     static double validarDouble(char msj[20]);
8 };
9
10 #endif
```

0.2. ValidarDatos.cpp

```
1 #include <iostream>
2 #include <stdio.h>
3 #include <conio.h>
4 #include <stdlib.h>
5 #include "ValidarDatos.h"
6
7
8 int ValidarDatos::validarEntero(char msj[20]){
9     char c;
10    int valor,i=0;
11    char* dato = new char[10];
12    printf("\nTEVS: %s",msj);
13    while((c=getch())!=13){
14        if(c>='0' && c<='9'){
15            *(dato + i++)=c;
16            printf("%c",c);
17        }
18    }
19    *(dato + i)='\0';
20    valor=atoi(dato);
21    printf("\n");
22    delete[] dato;
23    return valor;
24 }
25
26 double ValidarDatos::validarDouble(char msj[20]){
27     char c;
28     bool puntoUsado = false;
29     int valor,i=0;
30     char* dato = new char[10];
31     printf("\nTEVS: %s",msj);
32     while((c=getch())!=13){
33         if(c == '.' && !puntoUsado) {
34             *(dato + i++)=c;
35             puntoUsado = true;
36             printf("%c", c);
37         } else
38             if(c>='0' && c<='9'){
39                 dato[i++] = c;
40                 printf("%c",c);
41             }
42     }
43     *(dato + i)='\0';
44     valor=atoi(dato);
45     printf("\n");
46     delete[] dato;
```



```
47     return valor;  
48 }
```