UNIVERSIDAD TECNOLÓGICA DE TIJUANA

DESARROLLO DE SOFTWARE

PRÁCTICA 2. RECURSIVIDAD.

ACKERMANN

ELABORADO POR:

CUEVAS ZAPATA JUAN ANDRÉS

GONZALEZ GONZALEZ BRANDON

MARTIN ROJAS CARLOS ARIEL

TOLEDO HERRERA NEYZER JOEL

SALAZAR URTUZUASTEGUI YADIRA GUADALUPE

EN LA MATERIA DE: ESTRUCTURA DE DATOS APLICADAS

IMPARTIDA POR EL MAESTRO: CLEOTILDE TENORIO HERNANDEZ

GRUPO:4C

Tijuana, Baja California a 19 de Enero del 2025

Descripción

La función Ackermann es una función matemática no primitiva recursiva que crece extremadamente rápido. Toma dos números enteros no negativos, m y n, como entrada y devuelve un único valor entero como salida. Es famoso por su rápida tasa de crecimiento y por ser un ejemplo de recursión profunda.

En este ejercicio se buscó implementar dicha función en C# por medio de métodos, utilizando la recursividad. Además considerando las características de Ackermann, se delimitaron los rangos de m y n a números enteros del 0 al 3, para obtener datos representables, por ello se realizaron diversas validaciones para evitar posibles inconvenientes en el código. A su vez, se trató de realizar una interfaz amigable e intuitiva.

TDA Ackermann

Los datos necesarios son:

- Dos valores enteros no negativos con un rango de 0 a 3

Con este TDA se debe ser capaz de realizar las siguientes operaciones:

- calcularAckermann
- validarNumero
- validarContinuar

Paso 1: Elementos que lo conforman (tipo de dato)

Elemento	Tipo de dato
m	Número Entero
n	Numero Entero

Paso 2: Tipo de organización en el que se guardan los elementos

Lineal

Paso 3: Dominio de la estructura (rango de valores)

Elemento	Valores
m	Número que representa el nivel o complejidad de recursión
n	Número que representa la cantidad de pasos por cada iteración

Paso 4: Operaciones

- calcularAckermann
- imprimirAckermann
- validarNumero
- validarContinuacion

Operación 1	validarNumero
Objetivo	Validar que los parámetros para calcular la función de Ackermann no provoquen un desbordamiento de pila.
Pre condición	Que los valores ingresados sean números enteros.
Entrada	Valores asignados a ‘m’ y ‘n'.
Salida	Retornar los valores válidos (0 al 3) para la función Ackermann.
Post Condición	Que proceda con el cálculo de la función Ackermann.

Operación 2	calcularAckermann
Objetivo	Calcular el resultado de la función de Ackermann.
Pre condición	Que m y n estén validados con la Operación 1.
Entrada	• Número entero m • Número entero n
Salida	Devolver el resultado de la función Ackermann.
Post Condición	Que el resultado sea correcto.

Operación 3	imprimirAckerman
Objetivo	Pedir los valores al usuario, validarlos e imprimir el resultado de la función de Ackermann.
Pre condición	Que funcionen correctamente la operación 1 y 2.
Entrada	Los valores numéricos por medio del teclado.
Salida	Despliegue del resultado de manera agradable.
Post Condición	Que los datos hayan sido calculados y desplegados correctamente.

Operación 4	validarContinuacion
Objetivo	Leer si el usuario confirma que se vuelva a repetir una operación nuevamente o que se detenga
Pre condición	Que ya haya una operación antes ejecutar esta misma
Entrada	Una letra (“S” o “N”).
Salida	Mensaje de confirmación.
Post Condición	Se repita o detenga la operación correctamente.

Código en C#

```
class Program
{
    0 references
    static void Main(String[] args)
    {
        Console.Title = "Ackermann";
        bool bandera;

        do
        {
            Console.Clear();
            Console.WriteLine("\n\t\tCalculadora Ackermann\n");
            ImprimirAckerman();
            bandera = ValidarContinuacion();
        } while (bandera);

        Console.WriteLine("\nAdiós");
    }
```

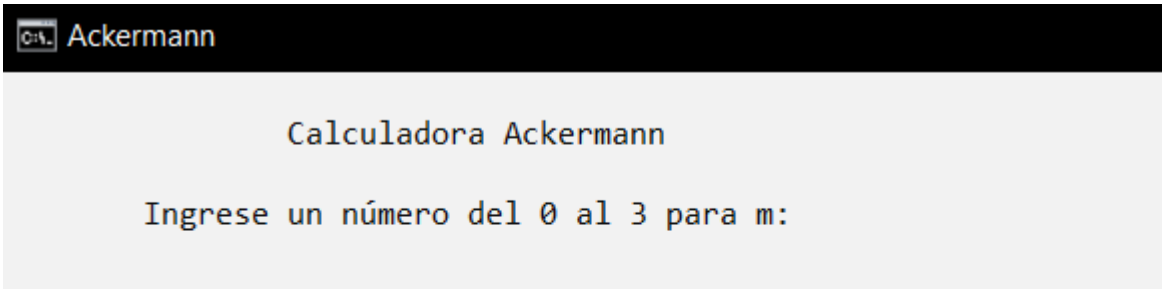
```
static void ImprimirAckerman()
{
    int m = ValidarNumero("\tIngrese un número del 0 al 3 para m: ");
    int n = ValidarNumero("\tIngrese un número del 0 al 3 para n: ");

    //$"" permite interpolacion, es decir,
    //permite concatenar variables dentro del texto
    Console.WriteLine($"\n\tAckerman({m},{n}) = " + Ackerman(m, n));
}
4 references
static int Ackerman(int m, int n)
{
    //si m = 0 : n +1
    if (m == 0 ) return n + 1;
    //si m > 0 y n = 0 : acker([m-1], 1)
    if(m > 0 && n == 0) return Ackerman((m - 1),1);
    //si m > 0 y n > 0 : acker( [m-1], acker(m,[n-1]) )
    return Ackerman((m - 1), Ackerman(m ,(n - 1)));
}
```

```
//El parametro texto es el mensaje que se mostrara
//al usuario antes de leer el numero
2 references
static int ValidarNumero(String texto)
{
    int num = 0;

    Console.Write(texto);

    // guardar posicion del cursor antes de leer el numero
    int posx = Console.CursorLeft;
    int posy = Console.CursorTop;

    do
    {
        // borrar lo que se haya escrito por teclado
        PrintXY(posx, posy, "                                          ");
        PrintXY(posx, posy, "");

        try {
            num = Int32.Parse(Console.ReadLine());

            //si se ingresa un numero valido, se rompe el ciclo
            if(num >= 0 && num  <= 3)
                break;

        } catch(Exception){
            //Borra el mensaje de error anterior para evitar sobreposicion
            PrintXY(0, posy+1, "                                                    ");
            PrintXY(4, posy+1, "");
            Console.WriteLine("\tFavor de Ingresar Un Número Válido");
            continue; //Continua el ciclo
        }
        Console.WriteLine("\tFavor de Ingresar Un Número entre 0 y 3");

    } while (true);

    //Borra la linea de abajo (donde se imprimen los errores)
    PrintXY(0, posy+1, "\t                                                  ");
    PrintXY(0, posy+1, "");

    return num;
}
```

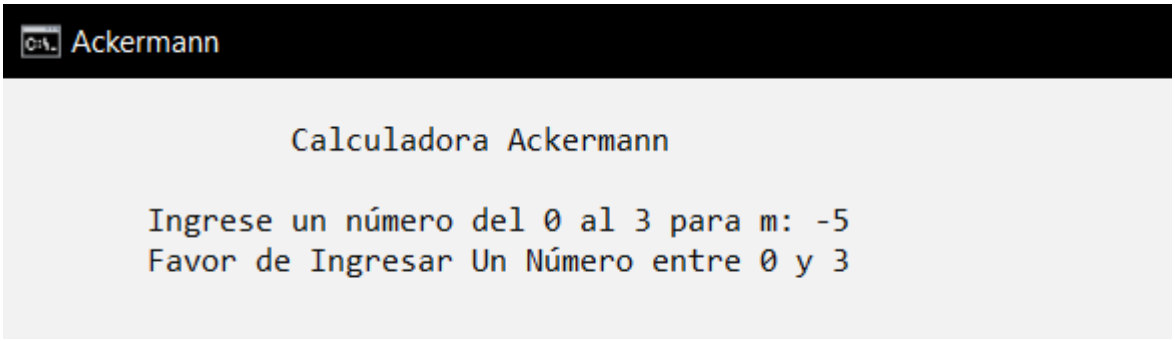
```
static bool ValidarContinuacion()
{
    String respuesta;
    bool validacion;

    Console.Write("\n\t¿Desea Continuar? (S/n): ");

    // guardar posicion del cursor antes de leer el numero
    int posx = Console.CursorLeft;
    int posy = Console.CursorTop;

    do{
        // borrar lo que se haya escrito por teclado
        PrintXY(posx, posy, "                                                ");
        PrintXY(posx, posy, "");
        respuesta = Console.ReadLine().ToLower();

        //validar que escribio S o N
        validacion = respuesta.Equals("s") || respuesta.Equals("n");
    } while (!validacion);

    return respuesta.Equals("s");
}


    static void PrintXY(int x, int y, string text)
    {
        Console.SetCursorPosition(x,y);
        Console.Write(text);
    }

}
```

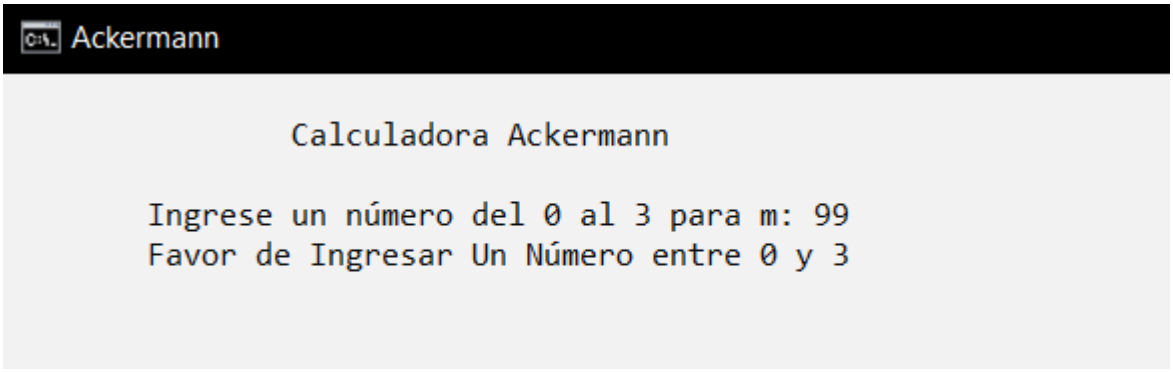
Imágenes de los resultados obtenidos

Pantalla principal del programa donde solicita ingresar un valor de m.

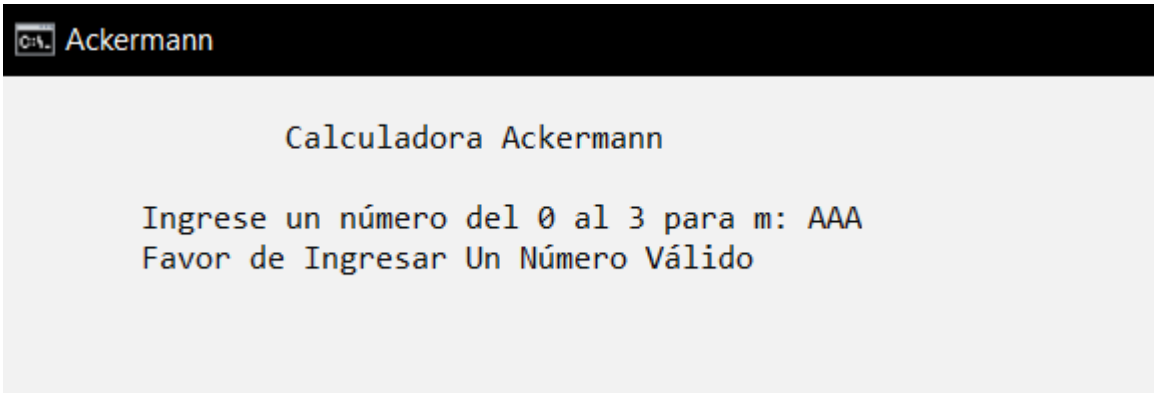


Si el usuario introduce un número fuera del rango, se le es notificado y se le solicita ingresar el valor de nuevo.

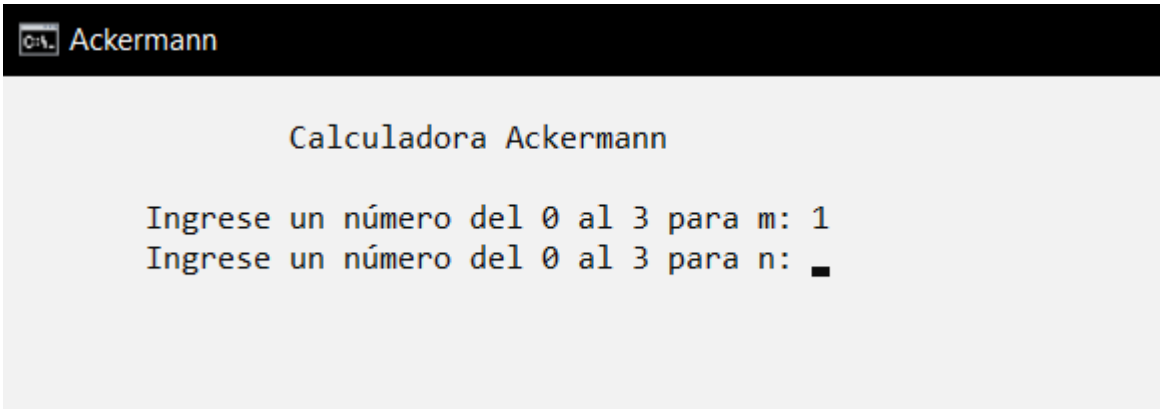




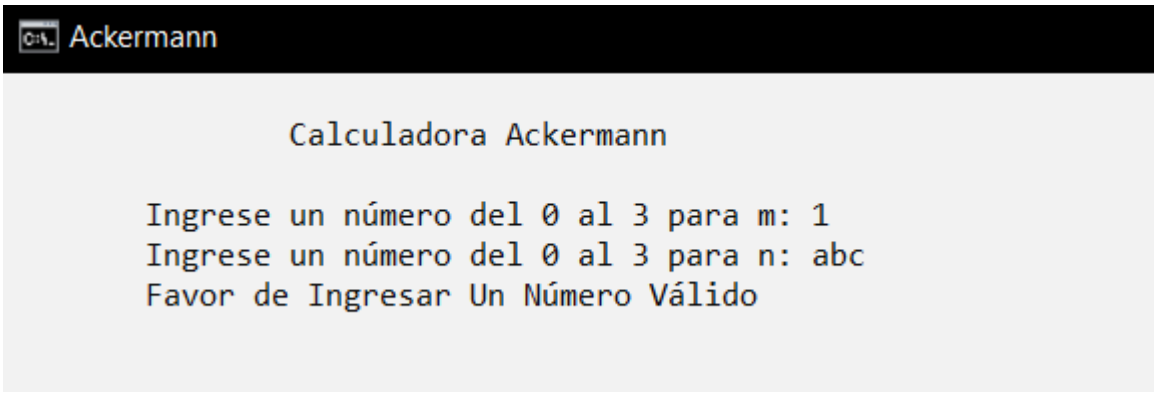
De igual forma, si intenta ingresar un valor que no es numérico (una cadena o un decimal), se le notifica y se le solicita ingresar un número válido.



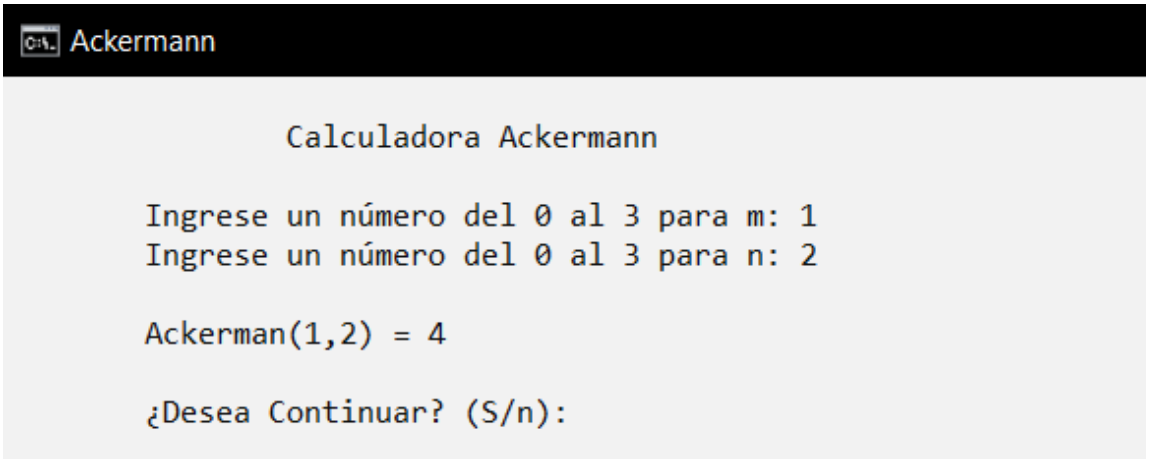
Si el usuario introduce un valor dentro del rango, el programa procede a pedir el valor de n.



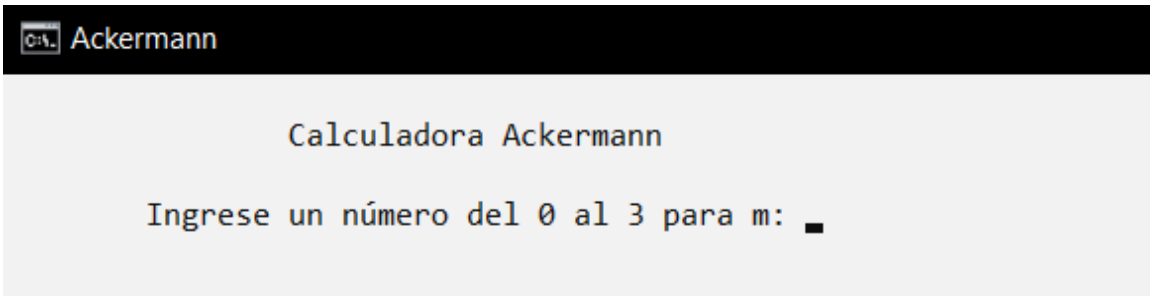
Esta lectura también cuenta con validaciones y mensajes de advertencia.



En caso de ingresar 2 valores válidos, el programa calcula la fórmula de Ackermann y lo despliega en pantalla. Seguido de esto, se le pregunta si desea continuar.



Si escribe “s” o “S”, el programa pedirá de nuevo los valores para m y n.



En caso de escribir “n” o “N”, el programa terminará su ejecución.