Calculadora

Este es un programa simple en C# que realiza operaciones básicas de suma, resta, multiplicación, división y módulo (residuo) entre dos números ingresados por el usuario.

1. **Declaración de variables:**

NUM1, NUM2 y RESUL son variables de tipo float, que almacenan números con punto decimal.

linea es una variable de tipo string que se utilizará para leer las entradas del usuario.

|  |
| --- |
| float **NUM1, NUM2, RESUL**; |
| string linea; |

1. **Solicitud de los números al usuario:**

Se pide al usuario que ingrese el primer número y se almacena en la variable NUM1.

Se pide al usuario que ingrese el segundo número y se almacena en la variable NUM2.

|  |
| --- |
| Console.WriteLine("PRIMER NUMERO"); |
| linea = Console.ReadLine(); |
| NUM1 = float.Parse(linea); |
| Console.WriteLine("SEGUNDO NUMERO"); |
| linea = Console.ReadLine(); |
| NUM2 = float.Parse(linea); |

1. **Operaciones aritméticas:**

Se realizan las operaciones de suma, resta, multiplicación, división y módulo con los números ingresados y se almacenan en la variable RESUL.

Cada resultado se imprime en la consola.

|  |
| --- |
| RESUL = NUM1 + NUM2; |
| Console.WriteLine("LA SUMA ES = {0} ", RESUL); |
| RESUL = NUM1 - NUM2; |
| Console.WriteLine("LA RESTA DE {0} - {1} ES = {2}", NUM1, NUM2, RESUL); |
| RESUL = NUM1 \* NUM2; |
| Console.WriteLine("LA MULTIPLICACION ES = " + RESUL); |
| RESUL = NUM1 / NUM2; |
| Console.WriteLine("LA DIVISIÓN ES = " + RESUL); |
| RESUL = NUM1 % NUM2; |
| Console.WriteLine("EL RESIDUO ES = " + RESUL); |

1. **Espera de entrada:**

El programa espera a que el usuario presione una tecla antes de cerrarse.

|  |
| --- |
| Console.Write("Pulse una Tecla:"); |
| Console.ReadLine(); |

Devuelve datos

Este programa en C# es un ejemplo simple que ilustra cómo definir y llamar a un método en C#.

1. **Definición del método Calculations:**

Se define un método llamado Calculations que toma dos parámetros a y b, ambos enteros.

Este método devuelve un entero que es la suma de a y b.

|  |
| --- |
| int Calculations(int a, int b) |
| { |
| return a + b; |
| } |

1. **Llamada al método desde el método Main:**

Dentro del método Main, se llama al método Calculations y se le pasan dos números como argumentos.

El resultado de la llamada al método se almacena en una variable llamada sum.

Finalmente, se imprime el valor de sum en la consola.

|  |
| --- |
| int sum = Calculations(10, 20); |
| Console.WriteLine(sum); |

En resumen, el programa ejecuta el método Calculations con los argumentos 10 y 20. Este método devuelve la suma de estos dos números, que luego se imprime en la consola.

Captura de datos para devolver un resultado capturado

Este programa en C# solicita al usuario que ingrese un número del 1 al 3, y luego muestra un mensaje dependiendo del número ingresado. Aquí está su explicación paso a paso:

1. **Declaración de variables:**

Se declara una variable llamada *opcion* de tipo entero para almacenar la opción ingresada por el usuario.

int opcion;

1. **Solicitud de entrada al usuario:**

Se muestra un mensaje pidiendo al usuario que ingrese un número del 1 al 3.

La entrada del usuario se lee con Console.ReadLine() y se convierte de cadena a entero con int.Parse(), luego se asigna a la variable *opcion*.

|  |
| --- |
| Console.WriteLine("Del 1 al 3 ingrese un numero "); |
| opcion = int.Parse(Console.ReadLine()); |

1. **Estructura switch:**

Se utiliza una estructura de control switch para evaluar el valor de la variable *opcion* y realizar diferentes acciones según el valor.

Cada case representa una opción válida que el usuario puede ingresar.

Si el valor de *opcion* coincide con uno de los casos, se ejecuta el código correspondiente dentro de ese caso.

Si no hay coincidencia con ninguno de los casos, se ejecuta el default.

|  |
| --- |
| switch (opcion) |
| { |
| case 1: |
| Console.WriteLine("Has seleccionado la opción 1"); |
| break; |
| case 2: |
| Console.WriteLine("Has seleccionado la opción 2"); |
| break; |
| case 3: |
| Console.WriteLine("Has seleccionado la opción 3"); |
| break; |
| default: |
| Console.WriteLine("Opción no válida"); |
| break; |
| } |

1. **Mensaje de salida:**

Se muestra en la consola el mensaje correspondiente a la opción seleccionada por el usuario o el mensaje de "Opción no válida" si la opción no coincide con ninguno de los casos.

Por ejemplo, si el usuario ingresa 2, el programa imprimirá: lo que contenga el número 2 en este caso nada.

Ciclo For

Se solicita al usuario que ingrese 10 números, luego calcula cuántos están por encima de 7, cuántos están por debajo de 7 y calcula el promedio de todos los números ingresados. Aquí está el análisis detallado:

1. **Declaración de variables:**

Se declaran las variables num1, num2, num3, num4, num5, num6 y num7.

num3, num4, num5 y num6 se inicializan en 0.

|  |
| --- |
|  |
| int num1, num2, num3, num4, num5, num6, num7; |
| num3 = 0; |
| num4 = 0; |
| num5 = 0; |
| num6 = 0; |

1. **Bucle FOR para solicitar los números:**

Se utiliza un bucle for para solicitar al usuario que ingrese 10 números.

num1 actúa como un contador en este bucle.

Dentro del bucle, se muestra el mensaje "Ingrese un número" y se lee la entrada del usuario con Console.ReadLine(). Esta entrada se convierte a un número entero utilizando int.Parse() y se almacena en la variable num2.

Se suma el valor de num2 a num6, que se utilizará para calcular el promedio más tarde.

Se comprueba si num2 es mayor o igual a 7. Si es así, se incrementa num4 en 1 (esto contará cuántos números están por encima de 7).

Si num2 es menor que 7, se incrementa num5 en 1 (esto contará cuántos números están por debajo de 7).

|  |
| --- |
| for (num1 = 0; num1 < 10; num1++) |
| { |
| Console.WriteLine("Ingrese un número"); |
| num2 = int.Parse(Console.ReadLine()); |
| num6 = num2 + num6; |
| if (num2 >= 7) |
| { |
| num4++; |
| } |
| else if (num2 < 7) |
| { |
| num5++; |
| } |
| } |

1. **Cálculo y presentación de los resultados:**

Después del bucle, se muestra el número de personas que aprobaron (números mayores o iguales a 7) y el número de personas que reprobaron (números menores a 7).

Se calcula el promedio dividiendo la suma total num6 entre 10 (la cantidad de números ingresados).

|  |
| --- |
| Console.WriteLine("Las personas que se encuentran por encima de 7 son: {0} ", num4); |
| Console.WriteLine(""); |
| Console.WriteLine("Las personas que se encuentran debajo de 7 son: {0} ", num5); |
| Console.WriteLine(""); |
| num7 = num6 / 10; |
| Console.WriteLine("El promedio general es: {0} ", num7); |

Matrices

Este código en C# solicita al usuario que ingrese 5 números y luego muestra los números ingresados junto con la posición en la que se almacenaron en el arreglo.

1. **Importación del espacio de nombres System:**

El programa comienza con using System;, lo que permite el uso de clases y métodos definidos en el espacio de nombres System.

|  |
| --- |
| using System; |

1. **Declaración de un arreglo y una variable:**

Se declara un arreglo de enteros llamado numeros con una longitud de 5. Este arreglo se usará para almacenar los números ingresados por el usuario.

Se declara una variable cont e inicializa en 0. Esta variable se utilizará para llevar el control del número de veces que se solicita al usuario ingresar un número.

|  |
| --- |
| int[] numeros = new int[5]; |
| int cont = 0; |

1. **Bucle for para solicitar los números:**

Se utiliza un bucle for para iterar sobre cada elemento del arreglo numeros.

En cada iteración, se incrementa cont en 1 y se muestra un mensaje solicitando al usuario que ingrese un número.

Se lee la entrada del usuario con Console.ReadLine() y se convierte a un número entero utilizando int.Parse(), luego se asigna al elemento correspondiente del arreglo numeros.

|  |
| --- |
| for (int i = 0; i < numeros.Length; i++) |
| { |
| cont++; |
| Console.WriteLine("Ingrese número a guardar en {0}", cont); |
| numeros[i] = int.Parse(Console.ReadLine()); |
| } |

1. **Bucle for para mostrar los números ingresados junto con sus posiciones:**

Se utiliza otro bucle for para iterar sobre cada elemento del arreglo numeros.

En cada iteración, se incrementa cont en 1 y se muestra un mensaje que indica la posición en la que se almacenó el número en el arreglo, así como el número mismo.

|  |
| --- |
| Console.WriteLine(""); |
| for (int i = 0; i < numeros.Length; i++) |
| { |
| cont++; |
| Console.WriteLine("El primer número guardado en {0} y era {1}", cont, numeros[i]); |
| } |

Métodos void

Este código realiza la suma acumulada de un número elevado a una potencia ingresada por el usuario. Aquí tienes una explicación detallada:

1. **Declaración de variables:**

num, contador, exponente, a y b son variables enteras que se utilizarán en el programa.

respuesta es una cadena de texto que almacena la respuesta del usuario para continuar o no con el cálculo.

|  |
| --- |
| int num, contador, exponente, a, b; |
| string respuesta; |
| contador = 0; |

1. **Solicitud del número y la potencia al usuario:**

Se inicia un bucle do-while que solicitará al usuario que ingrese un número (a) y la potencia a la que desea elevarlo (b).

Luego se calcula la suma acumulada utilizando la función sumaAcumulada(a, b).

Se muestra el resultado de la suma acumulada.

|  |
| --- |
| do |
| { |
| Console.WriteLine("Ingrese un número"); |
| a = int.Parse(Console.ReadLine()); |
| Console.WriteLine("¿A qué potencia desea aumentarlo?"); |
| b = int.Parse(Console.ReadLine()); |
| Console.WriteLine("El número {0} elevado a {1} es igual a {2}", a, b, sumaAcumulada(a, b)); |
|  |
| contador++; |
| Console.WriteLine("\n¿Desea calcular la suma acumulada para otro número (s/n)? "); |
| respuesta = Console.ReadLine(); |
| } while (respuesta == "s" || respuesta == "S"); |

1. **Mensaje de cantidad de operaciones realizadas:**

Se verifica si se realizó solo una operación o si se realizaron múltiples operaciones.

Se muestra un mensaje indicando cuántas operaciones se realizaron.

|  |
| --- |
| if (contador == 1) |
| { |
| Console.WriteLine("Se realizó esta operación una vez"); |
| } |
| else |
| { |
| Console.WriteLine("Se realizaron esta operación {0} veces", contador); |
| } |

1. **Definición de la función sumaAcumulada como método void:**

Se define una función llamada sumaAcumulada que toma dos parámetros enteros a y b.

Dentro de esta función, se calcula la suma acumulada del número a elevado a la potencia b.

La función devuelve la suma acumulada

|  |
| --- |
| int sumaAcumulada(int a, int b) |
| { |
| int suma = 1; |
| for (int i = 1; i <= b; i++) |
| { |
| suma = suma \* a; |
| } |
| return suma; |
| } |

Matriz que captura datos para almacenar

Este código implementa el algoritmo de búsqueda binaria para encontrar un elemento específico en un arreglo ordenado. Aquí tienes una explicación paso a paso:

1. **Inicialización del arreglo y el objetivo:**

Se declara un arreglo ordenado arr que contiene algunos números.

Se declara una variable target que representa el elemento que estamos buscando en el arreglo.

|  |
| --- |
| int[] arr = { 2, 5, 8, 12, 16, 23, 38, 56, 72, 91 }; |
| int target = 23; |

1. **Llamada a la función de búsqueda binaria:**

Se llama a la función BinarySearch pasando el arreglo arr y el elemento target.

La función retornará el índice del elemento si se encuentra en el arreglo o -1 si no se encuentra.

|  |  |
| --- | --- |
| int result = BinarySearch(arr | target); |

1. **Mostrar el resultado:**

Se verifica si el resultado de la búsqueda binaria es diferente de -1.

Si el resultado es diferente de -1, se imprime un mensaje indicando que el elemento se encontró en el índice result del arreglo.

Si el resultado es -1, se imprime un mensaje indicando que el elemento no se encontró en el arreglo.

|  |
| --- |
| if (result != -1) |
| Console.WriteLine($"El elemento {target} se encuentra en el índice {result} del arreglo."); |
| else |
| Console.WriteLine($"El elemento {target} no se encuentra en el arreglo."); |

1. **Implementación de la búsqueda binaria:**

Se define la función BinarySearch que toma el arreglo arr y el elemento target como parámetros.

Se inicializan dos variables left y right que representan los límites izquierdo y derecho del rango de búsqueda.

Se utiliza un bucle while para realizar la búsqueda binaria:

Se calcula el índice medio mid del rango actual.

Si el elemento en la posición mid es igual al objetivo, se devuelve mid.

Si el objetivo es menor que el elemento en la posición mid, se actualiza right para buscar en la mitad izquierda del rango.

Si el objetivo es mayor que el elemento en la posición mid, se actualiza left para buscar en la mitad derecha del rango.

Si el elemento no se encuentra en el arreglo, se devuelve -1.

|  |
| --- |
| static int BinarySearch(int[] arr, int target) |
| { |
| int left = 0; |
| int right = arr.Length - 1; |
| while (left <= right) |
| { |
| int mid = left + (right - left) / 2; |
|  |
| if (arr[mid] == target) |
| return mid; |
|  |
| if (target < arr[mid]) |
| right = mid - 1; |
| else |
| left = mid + 1; |
| } |
| return -1; |
| } |

Arrays

Un arreglo en C# es una colección de elementos del mismo tipo, que están almacenados en posiciones consecutivas de memoria y son accedidos mediante un índice. En este código, se declara un arreglo de enteros llamado numeros con una longitud de 5. Aquí está el funcionamiento del código paso a paso:

1. **Declaración del arreglo:**

Se utiliza la siguiente sintaxis para declarar un arreglo:

|  |
| --- |
| tipo\_de\_dato[] nombre\_del\_arreglo = new tipo\_de\_dato[tamaño\_del\_arreglo]; |

tipo\_de\_dato indica el tipo de elementos que se almacenarán en el arreglo (pueden ser int, string, double, etc.).

nombre\_del\_arreglo es el nombre que le das al arreglo.

tamaño\_del\_arreglo es el número de elementos que el arreglo puede contener.

|  |
| --- |
| int[] numeros = new int[5]; |

1. **Asignación de valores a los elementos del arreglo:**

Los elementos del arreglo se pueden asignar individualmente utilizando el nombre del arreglo seguido de corchetes [] y el índice del elemento.

El índice comienza en 0 y termina en tamaño\_del\_arreglo - 1.

En este código, se asignan valores a los elementos del arreglo numeros.

numeros[0] = 43;

numeros[1] = 42;

numeros[2] = 41;

numeros[3] = 40;

numeros[4] = 39;

1. **Acceso a los elementos del arreglo mediante un bucle for:**

Se utiliza un bucle for para recorrer cada elemento del arreglo.

La propiedad Length de un arreglo devuelve el número de elementos en el arreglo, por lo que se utiliza en la condición del bucle.

En cada iteración, se imprime el elemento del arreglo en la posición i.

|  |
| --- |
| for (int i = 0; i < numeros.Length; i++) |
| { |
| Console.WriteLine(numeros[i]); |
| } |

1. **En este caso, el código imprimirá:**

43

42

41

40

39