Etapa 2

Creando Interfaces Graficas

John Brown Nope Mayorga

204024_4

DIRECTOR DE CURSO

FELIPE HERNAN ORTIZ ROJAS

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA (UNAD).

ESCUELA DE CIENCIAS BÁSICAS TECNOLOGÍA E INGENIERA

LENGUAJES DE PROGRAMACIÓN

BOGOTÁ

19 de marzo del 2023

Contenido

¿Qué tipo de datos maneja C#?

C# es un lenguaje POO que puede manejar varios tipos de datos. Algunos de estos son:

a. Numéricos: int, long, float, double, decimal...

b. De caracteres y cadenas: char, string.

c. Booleanos: bool.

d. Fecha y hora: DateTime.

e. Objetos: class, struct, enum.

f. Punteros: unsafe.

g. Colecciones: arrays, listas, diccionarios.

C# también permite la creación de tipos de datos personalizados mediante la definición de clases y estructuras.

¿Qué estructuras condicionales maneja C#?

C# tiene varios tipos de condicionales, como, por ejemplo:

1. if: Es una declaración condicional simple que permite ejecutar un bloque de código si se cumple una condición determinada:

```
if (edad >= 18)
{
Console.WriteLine("Eres mayor de edad");
}
```

2. if-else: Permite ejecutar un bloque de código si se cumple una condición y otro bloque si no se cumple:

```
if (edad >= 18)
{
Console.WriteLine("Eres mayor de edad");
}
else
{
Console.WriteLine("Eres menor de edad");
```

```
}
```

3. else if: Se utiliza para evaluar múltiples condiciones:

```
if (x > y)
{
Console.WriteLine("x es mayor que y");
}
else if (x < y)
{
Console.WriteLine("y es mayor que x");
}
else
{
Console.WriteLine("x e y son iguales");
}
```

4. switch: Permite seleccionar uno de varios bloques de código para ejecutar, dependiendo del valor de una variable determinada:

```
switch (opcion)
{
case 1:
Console.WriteLine("Opción 1 seleccionada");
break;
case 2:
Console.WriteLine("Opción 2 seleccionada");
break;
default:
Console.WriteLine("Opción no válida");
break;
}
```

¿Qué estructuras cíclicas maneja C#?

C# tiene varios tipos de ciclos o bucles que permiten repetir un conjunto de instrucciones mientras se cumple una condición. Ejemplo:

1. while: Repite un conjunto de instrucciones mientras se cumple una condición. Por ejemplo:

```
int i = 0;

while (i < 10)

{

Console.WriteLine(i);

i++;

}
```

2. do while: También repite un conjunto de instrucciones mientras se cumple una condición, pero la condición se evalúa después de la primera iteración. Por lo tanto, el conjunto de instrucciones se ejecuta al menos una vez. Por ejemplo:

```
int i = 0;

do

{

Console.WriteLine(i);

i++;

}

while (i < 10);
```

3. for: Es un ciclo controlado por contador que repite un conjunto de instrucciones un número específico de veces:

```
for (int i = 0; i < 10; i++)

{

Console.WriteLine(i);

}
```

4. foreach: Se utiliza para recorrer los elementos de una colección, como una matriz o una lista. Por ejemplo:

```
int[] numeros = { 1, 2, 3, 4, 5 };

foreach (int numero in numeros)

{
```

```
Console.WriteLine(numero);
}
```

¿Qué tipo de arreglos maneja C#?

Un arreglo es una estructura de datos que almacena un conjunto de elementos del mismo tipo. Los arreglos se utilizan comúnmente para almacenar y manipular grandes cantidades de datos.

Para declarar un arreglo en C#, se utiliza la siguiente sintaxis:

```
tipoDeDato[] nombreDelArreglo = new tipoDeDato[tamañoDelArreglo];
```

1. tipoDeDato es el tipo de dato de los elementos que se van a almacenar en el arreglo.

2. nombreDelArreglo es el nombre que se le da al arreglo.

3. tamañoDelArreglo es el número de elementos que se pueden almacenar en el arreglo.

Por ejemplo, el siguiente código declara un arreglo de enteros con un tamaño de 5 elementos:

```
int[] numeros = new int[5];
```

Para acceder a un elemento específico del arreglo, se utiliza el índice del elemento entre corchetes. Por ejemplo, para asignar el valor 10 al primer elemento del arreglo, se utiliza:

```
numeros[0] = 10;
```

Los índices de los elementos en un arreglo comienzan en cero y van hasta el tamaño del arreglo menos uno. Por ejemplo, si un arreglo tiene un tamaño de 5 elementos, los índices válidos son 0, 1, 2, 3 y 4.

C# tiene métodos para trabajar con arreglos, como el método Length que devuelve el tamaño del arreglo, el método Sort que ordena los elementos del arreglo y el método Copy que copia los elementos del arreglo a otro arreglo.

También se puede declarar arreglos de múltiples dimensiones en C#. Por ejemplo, el siguiente código declara un arreglo de dos dimensiones con tres filas y dos columnas:

```
int[,] matriz = new int[3, 2];
```

Los arreglos en C# son estructuras de datos útiles para almacenar y manipular grandes cantidades de datos del mismo tipo.

¿Qué es una excepción y como se implementan en C#?

Las excepciones se utilizan para manejar errores y problemas que pueden surgir durante la ejecución de un programa, como errores de sintaxis, divisiones por cero, acceso a datos no válidos o problemas de comunicación con otros sistemas.

Una excepción es un objeto que se lanza cuando ocurre un error durante la ejecución de un programa. Cuando se produce una excepción, el programa busca una sección de código capaz de manejar la excepción o, si no se encuentra ninguna, el programa se detiene y se muestra un mensaje de error.

En C#, las excepciones se manejan con bloques try-catch. El bloque try se utiliza para encapsular el código que puede lanzar una excepción, mientras que el bloque catch se utiliza para manejar la excepción lanzada. Por ejemplo:

```
try
{
// código que puede lanzar una excepción
}
catch (TipoDeExcepcion excepcion)
{
// código que maneja la excepción lanzada
}
```

El bloque catch se ejecuta sólo si se produce una excepción en el bloque try. El tipo de excepción que se puede manejar se especifica en el parámetro TipoDeExcepcion. Por ejemplo, el siguiente código maneja una excepción DivideByZeroException que se lanza cuando se intenta dividir un número por cero:

```
try
{
int resultado = 10 / 0;
}
catch (DivideByZeroException excepcion)
{
Console.WriteLine("Error: división por cero");
}
```

C# también proporciona los bloques finally y throw para trabajar con excepciones.

1. El bloque finally se utiliza para ejecutar código después de que se haya completado el bloque try o catch, independientemente de si se produjo una excepción o no.

2. El bloque throw se utiliza para lanzar una excepción manualmente.

Las excepciones en C# se utilizan para manejar errores y problemas durante la ejecución de un programa. Los bloques try-catch se utilizan para manejar excepciones, mientras que los bloques finally y throw se utilizan para trabajar con excepciones.