Inicio del programa.

Al principio del programa coloque que se pudiera crear dos carpetas automáticamente y están direccionadas hasta la carpeta principal  la creación de esas 2 carpetas las coloque con Directory.CreateDirectory que es un método estáticos que lo que hace es crear una carpeta pero si la carpeta ya existe no modifica nada se queda como esta y no se agrega otra extra.

```
Directory.CreateDirectory("C:/LaboratorioAvengers/Backup");
Directory.CreateDirectory("C:/LaboratorioAvengers/ArchivosClasificados");
```

Menu principal:

Para crear es menú principal cree un switch porque me permite ejecutar diferentes bloques de Código según el valor de una variable, como igual coloque un do-while para un ciclo para que se repita hasta que se cumpla la condición que en ese caso seria que cierre el y como igual coloque un if-else  para comprobar la validación del usuario al ingresar un numero que sea valido.

```
do
{
    Console.WriteLine("\t\u001b[97m==============================\u001b[0m");
    Console.WriteLine("\t\u001b[91m ¿Qué deseas hacer hoy\u001b[0m?");
    Console.WriteLine("\t==============================\n");
    Console.WriteLine("1. Crear Archivo");
    Console.WriteLine("2. Agregar inventario");
    Console.WriteLine("3. Leer documento línea por línea");
    Console.WriteLine("4. Leer todo el texto");
    Console.WriteLine("5. Copiar Archivo");
    Console.WriteLine("6. Mover Archivo");
    Console.WriteLine("7. Crear carpeta");
    Console.WriteLine("8. Listar archivos y carpetas");
    Console.WriteLine("9. Salir");
    Console.Write("\n Elige una opción: ");

    // Validamos que el usuario ingrese un número
    if (int.TryParse(Console.ReadLine(), out opcion))
    {
        Console.WriteLine("-----------------------------------------------");

        switch (opcion)
        {
            case 1:
                CrearArchivo();
                Console.WriteLine("Archivo creado con éxito.");
                break;

            case 2:
                AgregarInventos();
                Console.WriteLine("Inventos agregados correctamente.");
                break;

            case 3:
                LeerLineaPorLinea();
                Console.WriteLine("Lectura completada.");
                break;

            case 4:
                leerTodoElTexto();
                Console.WriteLine("Texto leído completamente.");
```

```
    else
    {
        Console.WriteLine("Por favor, ingresa un número válido.");
        opcion = 0; // Evita que el programa termine si el usuario ingresa texto
    }

while (opcion != 9); // Repite hasta que el usuario elija salir
```

El primer void que creé fue para **crear un archivo**. En él, definí **dos variables de tipo string**:

1. Una para almacenar la **ruta** donde se guardará el archivo.
2. Otra para el **contenido** que se escribirá en el archivo.

Luego, usé **File.WriteAllText**, que **crea** el archivo y escribe en él. Si el archivo ya existe, **sobrescribe** su contenido.

```
        void CrearArchivo()
{
    string path = "C:/LaboratorioAvengers/Inventos.txt";
    string contenido = "Archivo creado para Tony Stark\n";
    File.WriteAllText(path, contenido);
    Console.WriteLine("El archivo fue creado con éxito\t");
}
```

En el segundo void que creé, permití que el usuario ingresara inventos.
Al igual que en el primero, declaré **dos variables**:

1. Una para la **ruta del archivo** donde se guardarán los inventos.
2. Otra para almacenar el **contenido ingresado por el usuario**.

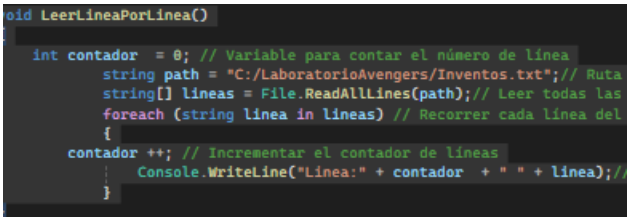
Para hacer el proceso más cómodo, agregué un **ciclo for**, que se repite la cantidad de veces que el usuario elija.
Además, utilicé File.AppendAllText(path, contenido + Environment.NewLine);, que agrega texto al final del archivo sin sobrescribir su contenido.

```
void AgregarInventos()
{
    Console.WriteLine("¿Cuántos inventos desea agregar?");
    int cantidad = int.Parse(Console.ReadLine()); // Convertir la entrada a número

    if (cantidad < 1)
    {
        Console.WriteLine(" No se pueden agregar 0 inventos.");
        return;
    }

    string path = "C:/LaboratorioAvengers/Inventos.txt";

    for (int i = 0; i < cantidad; i++)  // se creo un blucle for para poder pedirle al us
    {
        Console.Write($"Ingrese el invento {i + 1}: ");
        string contenido = Console.ReadLine();
        File.AppendAllText(path, contenido + Environment.NewLine);
    }

    Console.WriteLine(" Los inventos se agregaron con éxito.");
}
```

En el tercer void que creé, implementé la lectura línea por línea de un archivo.
En este caso, creé tres variables:

1. contador: Esta variable sirve para llevar el número de la línea mientras se recorre el archivo.
2. path: Es la ruta del archivo que se va a leer, en este caso "C:/LaboratorioAvengers/Inventos.txt".
3. lineas: Es un arreglo de tipo string[], donde se almacenan todas las líneas del archivo que se leen con File.ReadAllLines(path).

El ciclo foreach recorre cada línea del archivo y, por cada línea, se incrementa el contador para llevar el control del número de línea. Finalmente, se imprime en la consola el número de línea y su contenido.



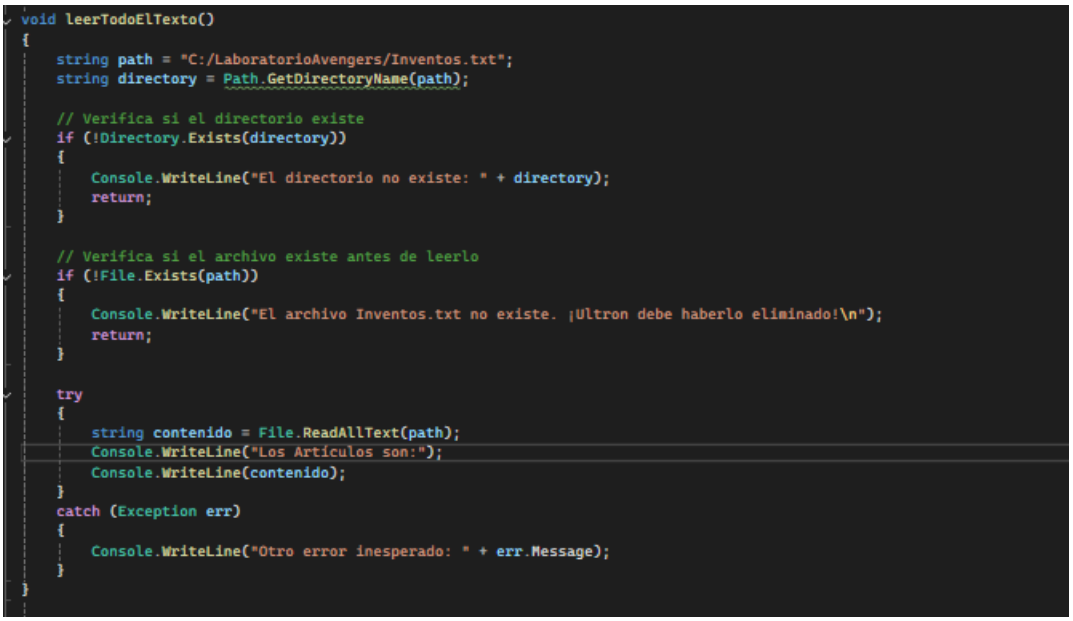
```
oid LeerLineaPorLinea()

  int contador  = 0; // Variable para contar el número de línea
        string path = "C:/LaboratorioAvengers/Inventos.txt";// Ruta
        string[] lineas = File.ReadAllLines(path);// Leer todas las
        foreach (string linea in lineas) // Recorrer cada línea del
        {
    contador ++; // Incrementar el contador de líneas
          Console.WriteLine("Linea:" + contador  + " " + linea);//
        }
```

En el cuarto void que creé, implementé la función para leer todo el archivo.
Para ello, creé dos variables:

1. La variable path: Que almacena la ruta del archivo que quiero leer.
2. La variable directory: Que obtiene la ruta del directorio donde se encuentra el archivo.
   Además, añadí varias condiciones para asegurarme de que:

El directorio exista antes de intentar acceder al archivo.

El archivo exista, de lo contrario, mostrar un mensaje que indica que el archivo no se encuentra.



```
void leerTodoElTexto()
{
    string path = "C:/LaboratorioAvengers/Inventos.txt";
    string directory = Path.GetDirectoryName(path);

    // Verifica si el directorio existe
    if (!Directory.Exists(directory))
    {
        Console.WriteLine("El directorio no existe: " + directory);
        return;
    }

    // Verifica si el archivo existe antes de leerlo
    if (!File.Exists(path))
    {
        Console.WriteLine("El archivo Inventos.txt no existe. ¡Ultron debe haberlo eliminado!\n");
        return;
    }

    try
    {
        string contenido = File.ReadAllText(path);
        Console.WriteLine("Los Artículos son:");
        Console.WriteLine(contenido);
    }
    catch (Exception err)
    {
        Console.WriteLine("Otro error inesperado: " + err.Message);
    }
}
```

En el quinto void que creé, implementé la función para copiar y eliminar un archivo.
Para esto, creé dos variables:

1. La variable origen: Esta variable contiene la ruta del archivo que quiero copiar.
2. La variable destino: Esta variable contiene la ruta de destino donde quiero que se copie el archivo.

Proceso:

- Comprobación de la existencia del archivo: Utilizo File.Exists(origen) para verificar si el archivo de origen existe. Si el archivo no existe, el proceso no continuará.
- Copiar el archivo: Si el archivo de origen existe, File.Copy(origen, destino, true) copia el archivo desde la ruta de origen hacia la ruta de destino. El parámetro true asegura que, si el archivo de destino ya existe, se sobrescriba.
- Eliminar el archivo original: Finalmente, después de copiar el archivo, utilizamos File.Delete(origen) para eliminar el archivo original.

```
void copiarArchivo()

{

    string origen = "C:/LaboratorioAvengers/inventos.txt";
    string destino = "C:/Backup/inventos.txt(copi)";

    if (File.Exists(origen))
    {
        // Copiar el archivo a la nueva ubicación
        File.Copy(origen, destino, true);
        Console.WriteLine("Archivo copiado exitosamente a 'Backup'.");

        // Eliminar el archivo original después de copiarlo
        File.Delete(origen);
        Console.WriteLine(" Archivo original eliminado.");
    }
    else
    {
        Console.WriteLine("Peligro el archivo 'inventos.txt' No esta creado todavia.");
    }
}
```

**En el sexto void que creé, implementé la función para mover un archivo.**
Para esto, utilicé **dos variables**:

1. **La variable origen**: Que contiene la **ruta del archivo** que quiero mover.
2. **La variable destino**: Que contiene la **ruta a la cual quiero mover el archivo**.
   Después realice Usé **File.Move(origen, destino)** para mover el archivo de la ruta de origen a la ruta de destino. Después de mover el archivo, se muestra un mensaje indicando que el archivo fue movido correctamente.

```
void MoverArchivo ()
{
    string origen = "C:/LaboratorioAvengers/inventos.txt";
    string destino = "C:/ArchivosClasificados/inventos.txt";
    File.Move(origen, destino);
    Console.WriteLine("Archivo movido exitosamente a la carpeta ArchivosClasificados.\n");
}
```

**En el séptimo void que creé, implementé la función para crear una carpeta.**
Es una función bastante simple, donde utilizo **una variable**:

1. **La variable path**: Esta variable almacena la **ruta completa** donde quiero que se cree la carpeta.

- Utilizo **Directory.CreateDirectory(path)** para crear la carpeta en la ubicación especificada por la variable path.
- Si la carpeta ya existe, este método no hace nada, pero si no existe, la crea sin problemas.

```
void CrearCarpeta()
{
    string path = "C:/LaboratorioAvengers/ProyectosSecretos";
    Directory.CreateDirectory(path);
    Console.WriteLine("la carpeta fue creada con existente\n");

}
```

En el octavo void que creé, implementé una función para listar las carpetas y archivos de una carpeta principal.
Para esto, utilicé una variable y dos métodos:

1. La variable path: Almacena la dirección de la carpeta principal donde se guardan todas las carpetas y archivos.

- Primero, uso Directory.Exists(path) para verificar si la carpeta principal realmente existe.
- Si la carpeta existe, creo dos variables:
    - Directory.GetDirectories(path): Esta variable obtiene todas las carpetas dentro de la ruta especificada.
    - Directory.GetFiles(path): Esta variable obtiene todos los archivos dentro de la ruta especificada.
    - Usa Path.GetFileName(archivo) para mostrar solo el nombre del archivo o carpeta, sin la ruta completa.

```
void ListadoArchivosYCarpetas()
{
    string path = "C:/LaboratorioAvengers";

    if (Directory.Exists(path))
    {
        // Obtener todas las carpetas dentro del directorio
        string[] carpetas = Directory.GetDirectories(path);
        // Obtener todos los archivos dentro del directorio
        string[] archivos = Directory.GetFiles(path);

        Console.WriteLine("Contenido de la carpeta 'LaboratorioAvengers :");

        // Mostrar las carpetas
        Console.WriteLine("\nCarpetas:");
        foreach (string carpeta in carpetas)
        {
            Console.WriteLine( Path.GetFileName(carpeta));
        }

        // Mostrar los archivos
        Console.WriteLine("\n Archivos:");
        foreach (string archivo in archivos)
        {
            Console.WriteLine( Path.GetFileName(archivo));
        }
    }
    else
    {
        Console.WriteLine("La carpeta 'LaboratorioAvengers' no existe.");
    }
}
```