# Exercise 1: Building an Asynchronous File Reader

#### Statement:

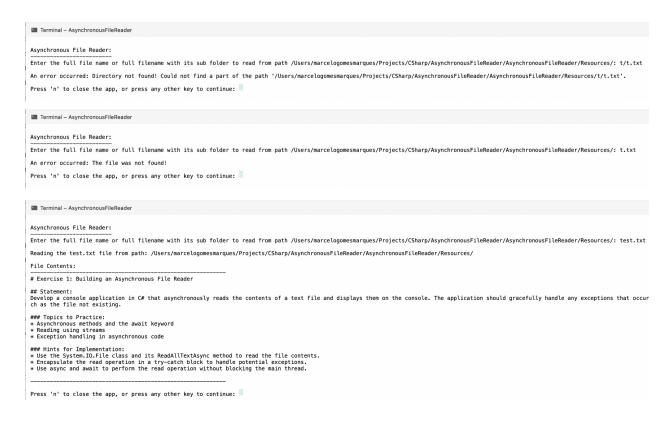
Develop a console application in C# that asynchronously reads the contents of a text file and displays them on the console. The application should gracefully handle any exceptions that occur during the file reading process, such as the file not existing.

## Topics to Practice:

- Asynchronous methods and the await keyword
- · Reading using streams
- Exception handling in asynchronous code

# Hints for Implementation:

- Use the System.IO.File class and its ReadAllTextAsync method to read the file contents.
- Encapsulate the read operation in a try-catch block to handle potential exceptions.
- Use async and await to perform the read operation without blocking the main thread.



# **Exercise 2**: Implementing a Multi-threaded Data Processing Application

#### Statement:

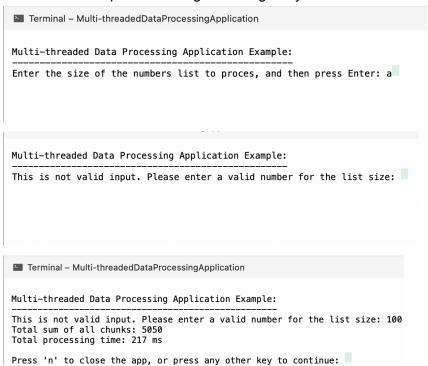
Create a C# console application that performs data processing in parallel. The application should generate a list of integers, divide the list into smaller chunks, and use the Task Parallel Library to process these chunks concurrently. Each task will simulate processing by implementing a delay. After processing, aggregate the results and display the completion time to demonstrate the performance benefit of parallel processing.

#### Topics to Practice:

- Task-based asynchronous programming
- Using Task and Task<T> for parallel operations
- Exception handling in asynchronous code

### Hints for Implementation:

- Utilize the Task.WhenAll method to wait for all processing tasks to complete.
- Consider using System. Diagnostics. Stopwatch to measure and display the processing time.
- Implement a simple processing function that, for example, calculates the sum of integers in each chunk and uses await Task.Delay to simulate work.
- Use exception handling to manage any errors that occur during task execution.



Terminal – Multi-threadedDataProcessingApplication

#### Multi-threaded Data Processing Application Example:

\_\_\_\_\_

Enter the size of the numbers list to proces, and then press Enter: 8

Total sum of all chunks: 36 Total processing time: 102 ms

Press 'n' to close the app, or press any other key to continue:

Terminal – Multi-threadedDataProcessingApplication

#### Multi-threaded Data Processing Application Example:

\_\_\_\_\_

Enter the size of the numbers list to proces, and then press Enter: 877

Total sum of all chunks: 385003 Total processing time: 111 ms

Press 'n' to close the app, or press any other key to continue:

Terminal – Multi-threadedDataProcessingApplication

#### Multi-threaded Data Processing Application Example:

Enter the size of the numbers list to proces, and then press Enter: 123456

A list with 123456 elements is too big to process!

Press 'n' to close the app, or press any other key to continue:

Terminal – Multi-threadedDataProcessingApplication

#### Multi-threaded Data Processing Application Example:

\_\_\_\_\_

Enter the size of the numbers list to proces, and then press Enter: 12345

Total sum of all chunks: 76205685

Total processing time: 169 ms

Press 'n' to close the app, or press any other key to continue: