Unsafe Code in .NET: Boosting Performance in a Trading Application

Key Takeaways

Task 1

Title: Setting Up the Project and Implementing Basic Order Book Initialization

- Enable Unsafe Code: Modify your project file to allow unsafe code, enabling the use of pointers for performance optimization.
- Pointer Types: Use pointer types to gain direct access to memory for faster data manipulation.
- Stack Allocation: Utilize stackalloc for efficient temporary storage, reducing heap memory usage.

Task 2

Title: Implementing the Order Book with Pointer Types

- Pointer Arithmetic: Traverse and manipulate data structures efficiently using pointers.
- Native Memory Management: Allocate and free unmanaged memory with NativeMemory for precise control over memory usage.

Task 3

Title: Using Fixed and Moveable Variables for Price Notifications

- Fixed Statement: Use the fixed statement to pin variables in memory and prevent garbage collection from relocating them.
- Memory Safety: Pin variables during critical operations to ensure stable pointers and avoid data corruption.

Task 4

Title: Pointer Conversions and Expressions in Order Fulfillment

- Pointer Conversions: Convert between different pointer types as needed for flexible memory manipulation.
- Pointer Arithmetic: Perform arithmetic operations on pointers to efficiently traverse and manage data.
- Data Integrity: Maintain data integrity by using proper pointer conversions and arithmetic techniques.

Task 5

Title: Implementing Fixed-Size Buffers and Stack Allocation for Order Cancellation

- Fixed-Size Buffers: Declare fixed-size buffers to ensure predictable memory usage and improve performance.
- Manual Memory Management: Optimize performance in high-frequency applications by mastering manual memory management techniques.

Additional Resources:

- C# Unsafe Code
- Pointer Types