Phase 1: C# Basics & Core Concepts – Full Notes

---

1. Variables & Data Types

 Value Types

- Stored in the **stack**, they are faster and directly contain their data.
- Examples: `int`, `float`, `bool`, `char`

 Reference Types

- Stored in the **heap**. Variables store references (addresses) to the actual data.
- Examples: `string`, `array`, `class`, `List<T>`

2. Control Flow

Conditional Statements:

- `if`, `else`, `else if`
- `switch`: Good for fixed-value branching
- Ternary operator: `condition ? valueIfTrue : valueIfFalse`

Loops:

- `for`: Use when count is known
- `while`: Use for unknown loop count
- `foreach`: Use to iterate collections
- `do-while`: Runs at least once

Practice: FizzBuzz logic using `for` loop and conditions.

---

3. Arrays

Key Methods:

- `arr[0]`: Access first element
- `arr.Length`: Get size of array
- `Array.Sort(arr)`: Sort in ascending order

Practice Problems:

- Reverse an array
- Find max and min element
- Count even numbers in an array

---

4. Strings

Key Methods:

- `Length`: Number of characters
- `Substring(start, length)`: Get substring
- `Replace(old, new)`: Replace characters
- `Split(' ')`: Tokenize string into array of words
- `ToCharArray()`: Convert to character array

Practice Problems:

- Reverse a string manually
- Check if a string is a palindrome
- Count vowels in a string

---

5. List

Key Methods:

- `Add(item)`: Adds element
- `Remove(item)`: Removes first occurrence
- `Sort()`: Sorts list
- `Insert(index, item)`: Inserts at specific index
- `Contains(item)`: Checks existence

Practice Problems:

- Remove duplicates
- Merge two sorted lists
- Remove all even numbers from a list

---

6. Dictionary (HashMap)

Key Methods:

- `dict[key] = value`: Add or update entry
- `ContainsKey(key)`: Check for existence
- `Remove(key)`: Delete key-value pair
- `TryGetValue(key, out value)`: Safer access

Practice Problems:

- Character frequency in a string
- Most frequent number in array
- Group words by their length

---

7. HashSet

Key Methods:

- `Add(item)`: Adds unique item

- `Contains(item)`: Checks presence
- `Remove(item)`: Deletes from set
- `Count`: Number of unique items

Practice Problems:

- Remove duplicates from an array
- Check if two arrays share elements
- Detect repeated characters

---

8. Stack & Queue

Stack:

- `Push(item)`: Add to top
- `Pop()`: Remove from top
- `Peek()`: View top without removing

Queue:

- `Enqueue(item)`: Add to end
- `Dequeue()`: Remove from front
- `Peek()`: View front without removing

Practice Problems:

- Check for valid parentheses (Stack)
- Reverse a queue
- First non-repeating character using queue

---

9. Custom Classes & OOP

Key Concepts:

- **Encapsulation:** Use of `private` and `public` to protect data
- **Constructors:** Special method to initialize objects
- **Inheritance:** One class derives from another (`: base`)
- **Polymorphism:** Override methods in child classes

Practice Problems:

- Create `Book` class with Title, Author, Display()
- Inherit `Animal` -> `Dog`, `Cat`
- Constructor for initializing object values

---

10. LINQ Essentials

 Filtering:

- `.Where(x => x > 5)`: Filter elements by condition

 Mapping:

- `.Select(x => x * 2)`: Transform each item

 Aggregation:

- `.Sum()`, `.Count()`, `.Min()`, `.Max()`

 Grouping:

- `.GroupBy(x => x.Length)`: Group by key (e.g., string length)

 Sorting:

- `.OrderBy(x => x)`: Ascending
- `.OrderByDescending(x => x)`: Descending

 Set Operations:

- `.Distinct()`: Remove duplicates
- `.Intersect()`: Common items
- `.Union()`: Combine with uniqueness
- `.Except()`: Difference between two sequences

Practice Problems:

- Count word frequency in a list using `GroupBy`
- Get top 3 largest numbers
- Find longest string using `OrderByDescending().First()`

Bonus: Practice Set Summary

Topic	Problem
Array	Rotate array left by k steps
String	Check if two strings are anagrams
List	Merge 2 sorted lists
Dictionary	First non-repeating character
Stack	Valid parentheses check
Queue	Simulate task scheduler
HashSet	Find duplicates in array
OOP	Create Employee class and calculate net salary
LINQ	Count how many even numbers are in a list using `.Where().Count()`

End of Phase 1 Notes.