🔹 Input & Output

- `Console.ReadLine()` is used to take input from the user.
- `Console.WriteLine()` is used to print output.
- Input needs to be **parsed** (converted) to numeric types when necessary using `int.Parse`, `Convert.ToInt32`, etc.

1. Input Handling & Parsing

✓ `Console.ReadLine()`

- Always returns a `string?` (nullable string) → **must validate null or whitespace** before parsing.
- Safer alternative: `int.TryParse()` avoids exceptions and improves robustness.

✓ Parsing Methods:

- `int.Parse(string)` → **throws** on bad format or null.
- `Convert.ToInt32(string)` → returns `0` on `null`, but throws on bad format.
- `int.TryParse(string, out result)` → safest; returns `true` / `false`.

🔹 Error Handling

- Use `try-catch` to handle runtime exceptions, like `FormatException` when parsing invalid strings.
- Example: Trying to convert `"123abc"` to an integer throws a `FormatException`.

🔹 Data Types & Operations

- **Value Types** (`int`, `float`) are copied by value. Changing one doesn't affect the other.
- **Reference Types** (`arrays`, `class`) are copied by reference. Changes via one reference affect the other.
- `Math.Max` and `Math.Min` help in comparing numbers.

🔹 String Operations

- Use `.Substring(startIndex, length)` to extract parts of a string.
- Strings can be concatenated using `+`.
- `string interpolation` with `$"..."` is a cleaner alternative to formatting strings.

🔹 Arithmetic

- Floating-point numbers (e.g., `float`, `double`) can be used for decimal calculations like:
  - BMI: `weight / (height * height)`
  - Simple Interest: `(principal * rate * time) / 100`

🔹 Conditions

- Use `if-else`, ternary (`?:`), or `switch` for decision-making.
- Ternary is great for short condition evaluations.

🔹 Character & String Checks

- Use `.Contains(char)` to check if a letter is a vowel.
- Convert input char to lowercase using `Char.ToLower()` for uniform comparison.

🔹 Miscellaneous

- `DateTime.DaysInMonth(year, month)` returns the number of days in any month.
- Always validate user input length or nulls to avoid runtime errors (e.g., check array length after `.Split()`).
-