**Workshop Handout: Managing Arrays and Enums in C#**

**Overview**

In this workshop, we will explore how to use arrays and enums in C#. We will create a console application that captures and displays pet details, including their category, using structured data types and enums.

**Key Concepts**

1. **Structs in C#**:
   * Structs are value types that can encapsulate small groups of related variables.
   * They are useful for representing lightweight objects such as a Pet in this example.
2. **Arrays in C#**:
   * Arrays are collections of variables of the same type stored in contiguous memory locations.
   * They allow storage and manipulation of multiple values efficiently.
3. **Enums in C#**:
   * Enums (enumerations) represent a group of named constants.
   * They enhance code readability and maintainability by assigning symbolic names to sets of values.
4. **User Input and Loops**:
   * We'll use loops to capture user input and populate the array.
   * A **for** loop will gather pet information, and a **foreach** loop will display it.

**Program Structure**

**1. Struct Definition**

* Define a **Pet** struct to hold properties like ID, name, and category.

**2. Enum Definition**

* Define an enum **PetCategory** with values: Dog, Cat, Bird, Fish, Reptile, and Other.

**3. Main Method Steps**

* **Prompt for Number of Pet Names**:
  + Ask the user how many pets they want to add to the array.
* **Populate the Array**:
  + Use a **for** loop to get pet details, including ID, name, and category.
  + Utilize a **do-while** loop and a **switch** statement to validate and set the pet category.
* **Display Pet Details**:
  + Use a **foreach** loop to iterate through the array and display the details of each pet.

**Detailed Steps**

1. **Define Struct and Enum**:
   * Create a struct **Pet** with fields for ID, name, and category.
   * Define an enum **PetCategory** with different pet categories.
2. **Get User Input for Array Size**:
   * Prompt the user to specify how many pets they want to add.
3. **Enter Pet Details**:
   * Use a loop to input details for each pet.
   * Assign a unique ID to each pet.
   * Prompt the user to enter the pet's name.
   * Prompt the user to choose a pet category from a list of options.
   * Use a **switch** statement to set the pet category based on user input.
4. **Display Stored Pet Details**:
   * Use a loop to iterate through the array.
   * Display the details of each pet, including ID, name, and category.

**Execution Flow**

1. **Prompt for Number of Pets**:
   * The program begins by asking the user how many pets they want to add.
2. **Input Pet Details**:
   * For each pet, the user is prompted to enter the name and select a category.
   * The program assigns an ID to each pet and stores the details in an array.
3. **Display Pet Details**:
   * After all details are entered, the program displays the information for each pet.

**Important Notes**

* **Array Size Limitation**:
  + Ensure the user does not enter more pets than the array can hold.
  + This example uses a fixed array size of 10.
* **Validation and Error Handling**:
  + The **do-while** loop ensures that the user selects a valid category.
  + Proper validation helps prevent runtime errors and enhances user experience.
* **Loops for Efficiency**:
  + The **for** loop is used for inputting pet details.
  + The **foreach** loop is used for displaying the details, making the code cleaner and more readable.
* **Enums for Readability**:
  + Using enums for pet categories makes the code more understandable and reduces the risk of errors from using magic numbers.

**Conclusion**

By the end of this workshop, you should understand how to use structs, arrays, and enums in C# to manage and display collections of related data. Arrays provide an efficient way to store multiple items, while enums enhance code clarity by representing fixed sets of related constants. This exercise demonstrates the practical use of these concepts in a user-interactive console application.