ANIMAL SHELTER REPORT

Animal Shelter

Overview

The Animal Shelter Management System is designed to help manage animals within a shelter. It includes functionalities to add, remove, sort, and find animals by their names. The system also provides functionality to save animal data to a file and read animal data from a file, ensuring data persistence.

File Structure

- **animal.h:** Contains the definition of the **`ANIMAL`** structure and enumeration for species.
- **administration.c:** Implements the core functionalities to manage animals (add, remove, sort, and find).
- **file_element.c:** Provides functionalities to save and read animal data to and from a file.
- **animal_shelter.c:** The main application file that provides a menu-driven interface to interact with the animal shelter functionalities.
- **administration_test.c:** Contains unit tests for the functions implemented in `administration.c`.

Functionalities

Administration.c

```c
int addAnimal(const ANIMAL* animalPtr, ANIMAL* animalArray, int position){
    // Check if the specified position is within valid bounds
    if (position < 0 || position >= MAXIMUM_ANIMALS)
    {
        // Invalid position, return an error code
        return -1;
    } else {
        animalArray[position] = *animalPtr;
        return 0;
    }
}
```

1. **addAnimal**
    - Description: Adds an animal to the specified position in the animal array.
    - Parameters:
        - `const ANIMAL* animalPtr`: Pointer to the animal to be added.
        - `ANIMAL* animalArray`: Array where the animal will be added.

        - `int position`: Position in the array to add the animal.
    - Return Value: Returns `0` if the animal is added successfully, otherwise returns `-1`.
2. **removeAnimal**

```
int removeAnimal(const char *name, ANIMAL *animalArray, int number)
{

    int counter = 0;
    int writeIndex = 0;
    for (int readIndex = 0; readIndex < number; readIndex++)
    {
        // check if name in given size matches
        if (strcmp(animalArray[readIndex].Name, name) != 0)
        {
            animalArray[writeIndex++] = animalArray[readIndex];
        }
        else
        {
            counter++;
        }
    }

    return counter;
}
```

    - Description: Removes animals with the specified name from the array.
    - Parameters:
        - `const char* name`: Name of the animal to be removed.
        - `ANIMAL* animalArray`: Array of animals.
        - `int number`: Number of animals in the array.
    - Return Value: Returns the number of animals removed.
3. **sortAnimalsByAge**

```c
int sortAnimalsByAge(ANIMAL *animalArray, int animalArrayLength)
{
    // no input in the array
    if (animalArray == NULL || animalArrayLength <= 0)
    {
        return -1;
    }

    int i, j;
    for (i = 0; i < animalArrayLength - 1; i++)
    {
        for (j = 0; j < animalArrayLength - 1 - i; j++)
        {
            if (animalArray[j].Age > animalArray[j + 1].Age)
            {
                // Swap the animals
                ANIMAL temp = animalArray[j];
                animalArray[j] = animalArray[j + 1];
                animalArray[j + 1] = temp;
            }
        }
    }

    return 0; // Return 0 on success
}
```

- Description: Sorts the animals in the array by age in ascending order.
- Parameters:
  - `ANIMAL* animalArray`: Array of animals.
  - `int animalArrayLength`: Length of the animal array.
- Return Value: Returns `0` if sorting is successful, otherwise returns `-1`.

4. **findAnimalByName**

```c
int findAnimalByName(const char *name, const ANIMAL *animalArray,
                     int animalArrayLength, ANIMAL *animalPtr)
{
    for (int i = 0; i < animalArrayLength; i++)
    {
        if (strcmp(name, animalArray[i].Name) == 0)
        {
            *animalPtr = animalArray[i];
            return 1;
        }
    }
    return 0;
}
```

- Description: Finds an animal by name in the array.
- Parameters:
  - `const char* name`: Name of the animal to find.
  - `const ANIMAL* animalArray`: Array of animals.
  - `int animalArrayLength`: Length of the animal array.
  - `ANIMAL* animalPtr`: Pointer to the animal found.
- Return Value: Returns 1 if the animal is found, otherwise returns `0`.

file_element.c

1. **saveAnimalsToFile**

```c
int saveAnimalsToFile(const ANIMAL *animalArray, int size)
{
    FILE *file = fopen(FILENAME, "w");
    if (file == NULL)
    {
        return 0;
    }

    for (int i = 0; i < size; i++)
    {
        fprintf(file, "%s %d %d\n", animalArray[i].Name, animalArray[i].Species, animalArray[i].Age);
    }

    fclose(file);
    return 1;
}
```

- Description: Saves the animals in the array to a file.
- Parameters:
  - `const ANIMAL* animalArray`: Array of animals to save.
  - `int size`: Number of animals in the array.

- Return Value: Returns 1 if saving is successful, otherwise returns `0`.

2. **readAnimalsFromFile**

```c
int readAnimalsFromFile(ANIMAL *animalArray, int maxSize, int *currentSize)
{
    FILE *file = fopen(FILENAME, "r");
    if (file == NULL)
    {
        return 0;
    }

    *currentSize = 0;
    while (*currentSize < maxSize && fscanf(file, "%s %d %d", animalArray[*currentSize].Name,
                                            (int *)&animalArray[*currentSize].Species,
                                            &animalArray[*currentSize].Age) == 3)
    {
        (*currentSize)++;
    }

    fclose(file);
    return 1;
}
```

- Description: Reads animals from a file into the array.
- Parameters:
  - `ANIMAL* animalArray`: Array to store the read animals.
  - `int maxSize`: Maximum size of the animal array.
  - `int* currentSize`: Pointer to store the number of animals read.
- Return Value: Returns 1 if reading is successful, otherwise returns `0`.

Main Application (`animal_shelter.c`)

The main application provides a menu-driven interface to interact with the animal shelter functionalities. Users can perform the following actions:

1. **Show all animals.**

```c
// Function to display details of all animals in the array
void showAnimals(ANIMAL* animals, int currentSize) {
    if (currentSize == 0) {
        printf("No animals in the shelter.\n");
    } else {
        printf("Animals in the shelter:\n");
        for (int i = 0; i < currentSize; i++) {
            printf("Name: %s, Species: %s, Age: %d\n", animals[i].Name, SpeciesNames[(animals[i].Species)], animals[i].Age);
        }
    }
}
```

2. **Add a new animal.**

```c
case 2: // Add Animal
    if (currentSize >= MAXIMUM_ANIMALS)
    {
        printf("Error: Cannot add more animals, array is full.\n");
        break;
    }
    ANIMAL animal;

    printf("Enter Name: ");
    scanf("%s", animal.Name);
    printf("Enter Species (0: Cat, 1: Dog, 2: GuineaPig, 3: Parrot): ");
    int species;
    scanf("%d", &species);

    // Validate species
    if (species < 0 || species > 3)
    {
        printf("Error: Invalid species. Please enter a valid species (0: Cat, 1: Dog, 2: GuineaPig, 3: Parrot).\n");
        break;
    }
    else
    {
        animal.Species = (SPECIES)species;
    }
    printf("Enter Age: ");
    // Validate age input by checking if  number entered is negative and input of user is
    while (scanf("%d", &animal.Age) != 1 || animal.Age < 0)
    {
        printf("Invalid input. Please enter a valid age as a positive number: ");
        while (getchar() != '\n')
            ; // Clear the input buffer
    }
    int result = addAnimal(&animal, animals, currentSize);
    if (result == -1)
    {
        printf("Error: Cannot add more animals, array is full.\n");
        break;
    }
    else
    {
        printf("Animal added successfully.\n");
        currentSize++;
    }
    saveAnimalsToFile(animals, currentSize);
    break;
```

3. **Remove an animal by name.**

```c
case 3: // Remove Animal
        // checks if there is no animal in the array
    if (currentSize == 0)
    {
        printf("Error: No animals to remove.\n");
    }
    else
    {
        // if there is animal in array it prompt user to input animal name
        char removeName[20];
        printf("Enter the name of the animal to remove: ");
        scanf("%s", removeName);
        // declare an integer to store number of element going to be removed
        int removedCount = removeAnimal(removeName, animals, currentSize);
        // if that inteeger is grater than zero means there element to be removes
        if (removedCount > 0)
        {
            // so the current size of array reduced
            currentSize -= removedCount;
            printf("Successfully removed %d animal(s) with the name '%s'.\n", removedCount, removeName);
        }
        else
        {
            // message when name you write does not includ in array
            printf("No animals were removed. No animal with the name '%s' found.\n", removeName);
        }
    }
    saveAnimalsToFile(animals, currentSize);
    break;
```

4. **Sort animals by age.**

```c
case 4: // Sort Animal By Age
        // sort animal by age it has two parameters first one help us to know number of element in the array and animal is pointer to
    if (sortAnimalsByAge(animals, currentSize) == 0)
    {
        printf("Animals sorted by age successfully.\n");
    }
    else
    {
        printf("Error: Could not sort animals bcz there is no animal.\n");
    }
    saveAnimalsToFile(animals, currentSize);
    break;
```

5. **Find an animal by name.**

```
case 5: // find animal by name
{
    char findName[25];
    printf("Enter the name of the animal to find: ");
    scanf("%s", findName);

    ANIMAL foundAnimal;
    int resultAnimal = findAnimalByName(findName, animals, currentSize, &foundAnimal);

    if (resultAnimal == 1)
    {
        printf("Found animal:\n");
        printf("Name: %s, Species: %s, Age: %d\n", foundAnimal.Name, SpeciesNames[(int)foundAnimal.Species], foundAnimal.Age);
    }
    else
    //(resultAnimal == 0 && currentSize == 0);
    {
        printf("No animal found with the name '%s'.\n", findName);
    }

}
break;
```

6. **Quit the application.**

```
case 0:
    saveAnimalsToFile(animals, currentSize);
    break;
```

Compiling the Program

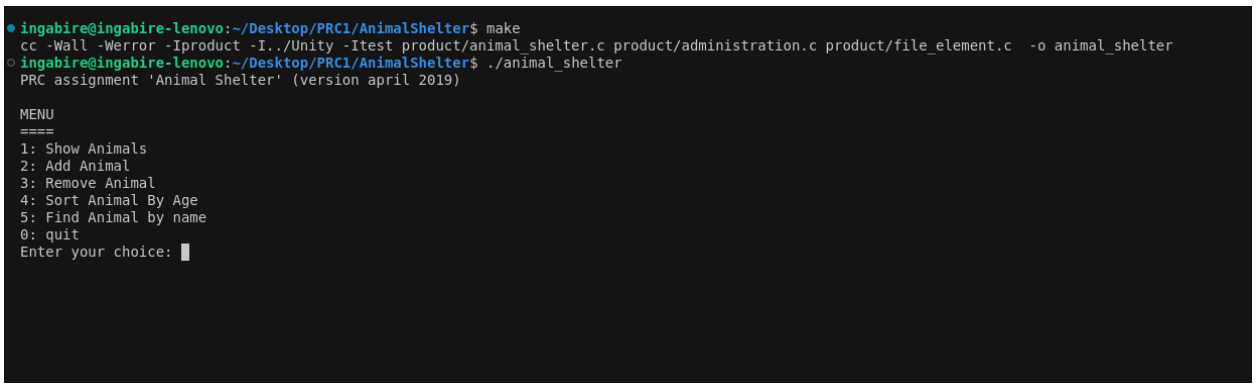
To compile the main application and the test suite, use the following commands in the terminal:

1. **Compile Main Application**
   make all or make

2. **Run the Main Application**

   ```
   ./animal_shelter
   ```

   After it prompt you the following message:



3. **make adminTest**

./administrationTest