Programming in C: Animal Shelter

Table of Contents

Introduction

Please use the Assignments/AnimalShelter directory for this assignment.

An animal shelter wants you to write a program for administrating its animals. You will have to provide a program that records information per animal. The program will use an in-memory list to store the animals of the shelter.

The application must for instance be able to add, search of remove animals from the animal administration. For using the application a simple command line interface is used as can be seen in figure 4.1.

Figure 4.1: the command line interface of the 'main' program.

See the section 'Build instructions' for more information on using the menu. For storing the animals a data structure called 'ANIMAL' will be used as can be seen in the figure below.

Figure 4.2: struct used for storing animal information (see animal.h for more details).

Source code

In Assignments/AnimalShelter you will find the following directories and files:

The following files are **not used** in this assignment (available as optional extension using file IO):

shared/file_element.c
shared/file_element.h
test/file_element_test.c

IMPORTANT NOTE:
This assignment uses a resource checker module that is located in your Code tree. You need not worry about how it works: it only checks if you free all claimed resources before your program finishes (and notifies you if you didn't).

To make sure the resource checker can work properly, you do need to:
include "resource_detector.h" in all your c files, **make sure it is the last include in the file!**
use the full prototype for main: int main (int argc, char * argv[])
(rather than the shortcut version: int main(void))
If you forget one of these points, your project will not compile!

Build instructions

Open a terminal and go to the directory that contains the Makefile.
The table below provides a overview of the possible build commands:

Build instructions	Description
make	Builds the executable ‘build/main’. You can run it from a terminal with ./build/main. It will display a menu that can be used to manipulate an in-memory animal administration.
make test	Builds and runs an executable ‘build/main_test’ containing the unit-tests for the ‘administation.c’ module. You can run it from a terminal with ./build/main_test.

Submission: Administration module

Only the following files of the administration module need to be implemented and submitted
(no more, no less)

- administration.c
- administration_test.c

Method information

A short description on all methods of the administration module can be found in 'administration.h'. Here you will found more information on the:

- pre- and post-conditions
- the expected return values

In the assignments below you will show with unit test that your implementation are in agreement.

Part-1: Administration: adding animals

Prerequisite knowledge

- pointers
- arrays
- unit-tests
- structs (only basic knowledge on how to use them).

Tips

```
//You can access 'Age' of an animal by using `.`:
Animal dog;
dog.Age = 8;

//For an animal-pointer you can access 'Age' by using `->`:
Animal* dogPtr = &dog;
dogPtr->Age = 9;

//Or if you like:
(*dogPtr).Age = 9;
```

Assignment

Please implement the following function in shared/administration.c:

- addAnimal

You will of course also make unit tests. These are implemented in test/administration_test.c.

Part-2: Administration: removing and finding animals

Assignment

Your administration still misses an implementation for the following methods:

- removeAnimal
- findAnimalById

Please implement them (it goes without saying that you'll make unit tests as well).

Part-3 (optional): Administration: sorting animals

Prerequisite knowledge

- function pointers

Tips

The easiest way to sort is by using qsort (see film about function pointers or man page). Please note: you are of course also allowed to implement your own sorting algorithm, as long as you implement 1 generic sorting method that has a function pointer to a compare function as one of its parameters.

Assignment

Your administration still misses an implementation for the remaining sort methods of the animal.c module:

- sortAnimalsByAge
- sortAnimalsByYearFound
- sortAnimalsBySex

For the sort functions: the goal is not sorting itself, but exercising with **function pointers**.