SCSSE, University of Wollongong
CSCI124 Data Structures and Algorithms
Autumn 2015

Assignment 1 (Due 11:59 PM Sun 29th March)

Objectives

To revise program design, functional decomposition, structs, file-I/O. To write a program involving multiple source files.

Task

You are to implement a pet lost-and-found directory. The directory should enable people who have lost or found a pet to search for the pet in the directory or add it's details to the directory.

The program is to be implemented in 3 files: **main.cpp** contains the text-menu based user interface. (A starting, but incomplete, version of is provided.) **ass1.h** should contain function prototypes and shared constants (if any) accessible by the main.cpp file. **ass1.cpp** contains the function definitions of the database. Do NOT place everything in the same file.

Step 1

Your first task is to design a suitable data structure to hold the entries. Your struct declaration should be placed in **ass1.h**. Each entry contains all the details relating to a pet along with the location lost/found and a contact phone number. The pet details are name, breed, colour, age in years and months, and microchip number (a 15 digit number). For example:

Status:	lost
Colour:	black
Gender:	male
Breed:	labrador retriever
Age:	2 years 3 months
Chip:	73921013356814
Location:	North Dock, Swansea
Phone:	17921234

Determine suitable data types (and, where necessary, array sizes) and define a structure to hold a single entry. Then define an array which can contain at most 50 entries. This is not very many entries, so make sure your code is written so that it is easy to change this number later.

When deciding what type and array size to make the data fields, look at the test data in pets.txt. You can assume this data is typical.

Note: the array of structs and the number of valid elements it contains should be local to main(), while the struct and array size definitions should be in ass1.h.

Step 2

Write a function to load a set of existing records from the file **pets.dat** into your array. You will need to define appropriate parameters to allow the required data to be passed. For this assignment, declare the file name as a constant in the ass1.cpp file, do not request it from the user. Extend the menu to provide an option load **l** that calls your function and displays how many records have been loaded.

```
Command >  l

There are 44 records in the directory.
```

Take care when using both getline and the stream extraction operator, as sometimes you may need to ignore a newline character.

Step 3

Implement a function to display the records on the screen one at a time until the user chooses `n`.

```
Command >  d

Status          found
Type            dog
Gender          male
Breed           pointer
Age             5 yrs 2 mths
Colour          brown-white
Microchip       739249103312148
Location        corrimal
Phone           42239402
Display next record (y/n)>  y

Status          found
Type            dog
Gender          male
Breed           border collie
Age             1 yrs 0 mths
Colour          black-white
Microchip       183749248347281
Location        dapto
Phone           429486631
Display next record (y/n)>  n
```

Notice that this function does not change anything, so this will affect the way the parameters are defined.

Step 4

Implement a function to add a new record to the directory. For this assignment you can just add it to the end of the array.

```
Command > a

Add Record to Directory
 (Enter your pet's details)
 (Press return if detail is unknown)

Is the pet lost or found?   (l/f) => l
Is the pet a dog or cat?    (d/c) => d
What breed is the pet?            => Border Collie
What age is the pet?     (yy mm ) => 5 6
What colour is the pet?           => black-white
What is the microchip number? => 132271920758392
In what suburb was the pet lost?  => Corrimal
What is your phone number?        => 42123412

The new record has been added to the directory.
There are 45 records in the directory.

Once the record has been added, rewrite the file.
```

Where the user does not enter a value, you will need a suitable default, such as `-1` or `unknown` to represent an unknown value.

Step 5

Write a function to search for records in the directory. If the user enters nothing for a field (just presses enter) that field should be ignored. For example, the search below should find all brown and white pointer dogs. Note that fields with unknown values should be considered to match.

```
Command > s
Search lost or found pets?  (l/f) => l
Search for a dog or cat?    (d/c) => d
What gender is the pet?           =>
What breed is the pet?         => pointer
What age is the pet?   (yy mm) =>
What colour is the pet?        => brown-white
What is the microchip number? =>
In what suburb was the pet found?   =>

There are 2 records that match the criteria:

Status          found
Type            dog
Gender          male
Breed           pointer
```

```
Age             5 yrs 2 mths
Colour          brown-white
Microchip       739249103312148
Location        corrimal
Phone           42239402
Display next record (y/n)>  n
```

Step 6

Write a function to display and then remove the most recently displayed record.

```
Command > r

Last viewed record was:

Status          found
Type            dog
Gender          male
Breed           pointer
Age             5 yrs 2 mths
Colour          brown-white
Microchip       739249103312148
Location        corrimal
Phone           42239402

The record has been removed from the directory.
There are 44 records in the directory.
```

If the most recently viewed record has already been removed, or no records have yet been viewed, this function should not do anything.

```
Command > r

No recently viewed record, please search for a record first.
```

Step 7

Write a function to undo the last removal. This can be done more easily if your remove function swaps the data to the end of the array instead of deleting it completely. Make sure it works even if new records were added after the last removal. Note that it does not have to be reinserted in the same location as before.

```
Command > u

The removed record was:

Status          found
Type            dog
Gender          male
Breed           pointer
Age             5 yrs 2 mths
```

```
Colour          brown-white
Microchip       739249103312148
Location        corrimal
Phone           42239402

The record has been reinserted from the directory.
There are 44 records in the directory.
```

If the most recently viewed record has already been removed, or no records have yet been viewed, this function should not do anything.

```
Command > r

No recently viewed record, please search for a record first.
```

Step 8

Write functions so that the file **pets.dat** is immediately updated after any add, remove or undo. Note that in the case of add and undo, you can just append the new record, but for removal it is easiest to just rewrite the array to file.

Submission instructions

Submit your source code by typing the command:
`submit -u <username> -c CSCI124 -a 1 main.cpp ass1.cpp ass1.h`
from the directory containing your files, where `<username>` is your SOLS username.

1. Late submissions will be marked with a 25% deduction for each day.
2. Submissions more than three days late will not be marked, unless an extension has been granted.
3. If you need an extension apply through SOLS, if possible **before** the assignment deadline.
4. Plagiarism is treated seriously. If we suspect any work is copied, all students involved are likely to receive zero for the entire assignment.