**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по курсовой работе № 2**

**по дисциплине «Программирование»**

Тема: **Разработка электронной картотеки**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0310 |  | Савенко Н.С |
| Преподаватель |  | Хахаев И.А |

Санкт-Петербург

2021**Введение.**

**Цель работы.**

Полное решение содержательной задачи (содержательная и формальная постановка задачи, спецификация, включая описание диалога, выбор метода решения и структур данных, разработка алгоритма, программная реализация, тестирование и отладка, документирование).

**Задание**

Создать электронную картотеку уток, хранящуюся на диске, и программу, обеспечивающую взаимодействие с ней.

Программа должна выполнять следующие действия:

занесение данных в электронную картотеку;

внесение изменений (исключение, корректировка, добавление);

поиск данных по различным признакам;

сортировку по различным признакам;

вывод результатов на экран и сохранение на диске.

Выбор подлежащих выполнению команд должен быть реализован с помощью основного меню и вложенных меню.

Задача должна быть структурирована и отдельные части должны быть оформлены как функции.

Исходные данные должны вводиться с клавиатуры. В процессе обработки картотека должна храниться в памяти компьютера в виде списков и массивов структур, связанных указателями. Типы списков и структур выбираются исходя из предметной области.

Картотека составляется по выбранной предметной области.

В программе должно быть реализовано простейшее меню. Выполнение программы должно быть многократным по желанию пользователя. Данные первоначально считываются из файла (файлов), в процессе работы данные вводятся с клавиатуры.

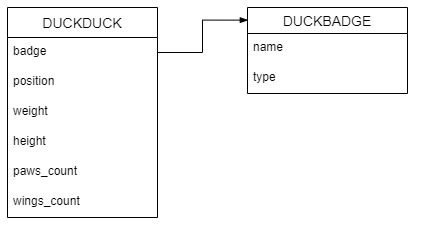
**Описание общей архитектуры данных**

Данные хранятся в виде структуры DUCKDUCK, которая является составной и хранит в себе DUCKBADGE.

DUCKDUCK является объектом утки, которая обладает: биркой (DUCKBADGE), позицией, весом, ростом, количеством лап и крыльев.

DUCKBADGE является объектом бирки, хранит имя и тип утки.

ER-диаграмма:



**Описание структур.**

DUCKDUCK

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | badge | DUCKBADGE | Бирка |
| 2 | position | Int[2] | Координаты |
| 3 | weight | float | Вес |
| 4 | height | float | Рост |
| 5 | paws\_count | int | Количество лап |
| 6 | wings\_count | int | Количество крыльев |

DUCKBADGE

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | name | char\* | Имя |
| 2 | type | char\* | Тип(порода) |

**Описание функций.**

*void show\_info();*

Позволяет отобразить вводную информацию о программе.

*int load\_data(duck \*\*ducksDest, char \*\*filename);*Позволяет загрузить массив уток по имени файла и получить их количество.

*void save\_data(duck \*\*ducks, int count);*Позволяет сохранить массив уток в файл.

*int add\_data(duck \*\*ducks, int count, char \*templates\_file);*Позволяет добавить утку в массив. Для упрощения добавления можно передать указатель на имя файла с шаблонами.

*char \*\*simple\_split(char \*str, int length, char sep);*Отвечает за разделение входной строки. Возвращает массив поделенных строк.

*void ClearStringArray(char \*\*str, int n);*Отвечает за очистку массива строк.

*int new\_gets(char \*s, int lim, char endSymbol);*Позволяет получить от пользователя строку.

*void show\_data(duck \*\*ducks, int count);*Отображает массив уток с заданным количеством.

*void edit\_data(duck \*\*ducks, int count);*Позволяет изменить элемент в массиве уток с заданным количеством.

*int delete\_data(duck \*\*ducks, int count);*Позволяет удалить элемент в массиве уток с заданным количеством.

*void search\_data(duck \*\*ducks, int count);*Позволяет найти элемент в массиве уток по определенному параметру.

*void sort\_data(duck \*\*ducks, int count);*Сортирует массив уток с заданным количеством.

*duck \*get\_new\_duck();*Позволяет получить от пользователя новую утку. Возвращает указатель на нее.

*void free\_duck(duck \*target);*Позволяет высвободить ресурсы, занимаемые конкретной уткой.

*void swap\_data(duck \*\*ducks, int count, int index0, int index1);*Меняет местами два элемента с заданными индексами в передаваемом массиве.

*void print\_duck(duck \*target\_duck);*Выводит данные о переданной утке.

*void free\_data(duck \*\*ducks, int count);*

Позволяет высвободить ресурсы, занимаемые массовом уток.

**Описание переменных**

**main()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | ducks | int | Массив уток |
| 2 | ducks\_count | int | Количество элементов |
| 3 | database\_filename | char[] | Имя файла базы данных |
| 4 | samples\_filename | Char[] | Имя файла базы шаблонов |
| 5 | state | int | Состояние |

**load\_data (duck \*\*ducksDest, char \*\*filename)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | ducksDest | duck\*\* | Указатель на массив уток |
| 2 | filename | Char\*\* | Указатель на имя файла |
| 3 | s2 | char[] | Массив сырых значений |
| 4 | string | Char[] | Текст из файла |
| 5 | sep | Char | Символ делитель |
| 6 | df | FILE\* | Дескриптор файла |
| 7 | slen | int | Длина строки |
| 8 | i/j | int | Итераторы |
| 9 | count | int | Количество элементов |

**add\_data (duck \*\*ducks, int count, char \*templates\_file)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | ducks | duck\*\* | Указатель на массив уток |
| 2 | local | duck\* | Буферный элемент |
| 3 | samples | duck\* | Указатель на массив шаблонов |
| 4 | templates\_file | Char\*\* | Указатель на имя файла |
| 5 | i | int | Итератор |
| 6 | samples\_count | int | Количество элементов шаблонов |
| 7 | count | int | Количество элементов массива |

**save\_data(duck \*\*ducks, int count)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | ducks | duck\*\* | Указатель на массив уток |
| 2 | count | int | Количество элементов массива |
| 2 | buffer\_len | int | Длина буфера |
| 3 | buffer | char\* | Буффер |
| 4 | df | FILE\* | Дескриптор файла |
| 5 | filename | Char\* | Имя файла |
| 6 | i | int | Итератор |

**char \*\*simple\_split(char \*str, int length, char sep)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | str | char\* | Исходная строка |
| 2 | length | int | Длина строки |
| 3 | sep | char | Символ делитель |
| 4 | str\_array | char\*\* | Конечный массив строк |
| 5 | key | int | Маркер успешности |
| 6 | i, j, k, m | int | Итераторы |

**ClearStringArray(char \*\*str, int n)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | str | char\*\* | Массив строк |
| 2 | n | int | Количество строк |
| 2 | i | int | Итератор |

**new\_gets(char \*s, int lim, char endSymbol)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | i | int | Счетчик символов |
| 2 | lim | int | Ограничение на количество символов |
| 3 | s | char[] | Текст из консоли |
| 4 | c | char | Текущий символ |
| 5 | endSymbol | Char | Символ окончания текста |

**show\_data(duck \*\*ducks, int count)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | ducks | duck\*\* | Указатель на массив уток |
| 2 | count | int | Количество элементов массива |
| 3 | i | int | Итератор |

**edit\_data(duck \*\*ducks, int count)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | ducks | duck\*\* | Указатель на массив уток |
| 2 | count | int | Количество элементов массива |
| 3 | choosen | int | Выбранное значение |
| 4 | new\_duck | duck\* | Указатель на новую утку. |

**int delete\_data(duck \*\*ducks, int count)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | ducks | duck\*\* | Указатель на массив уток |
| 2 | count | int | Количество элементов массива |
| 3 | choosen | int | Выбранное значение |

**search\_data(duck \*\*ducks, int count)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | ducks | duck\*\* | Указатель на массив уток |
| 2 | count | int | Количество элементов массива |
| 3 | choosen | int | Выбранное значение |
| 4 | value\_int | int | Выбранное целое значение |
| 5 | value\_float | float | Выбранное вещественное значение |
| 6 | string | char\* | Выбранное строковое значение |
| 7 | i | int | Итератор |

**sort\_data(duck \*\*ducks, int count)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | ducks | duck\*\* | Указатель на массив уток |
| 2 | count | int | Количество элементов массива |
| 3 | choosen | int | Выбранное значение |
| 4 | vector | int | Выбранное целое значение |
| 5 | step | int | Значение шага |
| 6 | i | int | Итератор |

**duck \*get\_new\_duck()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | new\_duck | duck\* | Указатель на новую утку |

**free\_duck(duck \*target)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | target | duck\* | Указатель на утку для высвобождения |

**swap\_data(duck \*\*ducks, int count, int index0, int index1)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | ducks | duck\*\* | Указатель на массив уток |
| 2 | count | int | Количество элементов массива |
| 3 | index0 | int | Индекс первого элемента |
| 4 | Index1 | int | Индекс второго элемента |
| 5 | temp | duck | Буферная утка |

**print\_duck(duck \*target\_duck)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | target\_duck | duck\* | Указатель на утку для отображения |

**free\_data(duck \*\*ducks, int count)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | ducks | duck\*\* | Указатель на массив уток |
| 2 | count | int | Количество элементов массива |
| 3 | i | int | Итератор |

**Схема алгоритма:**

См директорию schemes

**Текст программы**

См. файл Savenko-kurs2-main.c

**Заключение**:

**Использованные заголовочные файлы:**

**stdio.h**

puts() – вывод в консоль

scanf() – получение данных из консоли

getchar() – получение символа из консоли

fopen() – открытие файла

fseek() установка положения в файле

fgets () – получение строки из файла

fclose() – закрытие файла

**stdlib.h**

malloc() – выделение памяти

realloc() – довыделение памяти

free() – высвобождение пам

system() - передача команды командному процессору операционной системы

atoi() – приведение к целому числу

atof() – приведение к вещественному числу **string.h**

strlen() – получение длины строки

strcpy() – копирование

strcmp() – сравнение строк

**Вывод:**

Научился делать законченное полное решение содержательной задачи (содержательная и формальная постановка задачи, спецификация, включая описание диалога, выбор метода решения и структур данных, разработка алгоритма, программная реализация, тестирование и отладка, документирование).