МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра вычислительной техники

Отчет по лабораторной работе № 9

по дисциплине «Программирование»

Тема: «Структуры в языке Си».

Студент гр. 9305 Любаневич Р.О.

Преподаватель Перязева Ю. В.

Санкт-Петербург

2020

Содержание

Введение

Получить практические навыки в разработке алгоритма и написании программы на языке Си. Для ознакомления работы со структурами, а также правилами их написания на языке Си.

Задание

Для выбранной предметной области создать динамический массив структур, содержащих характеристики объектов предметной области.

Обязательный набор полей:

- динамический массив символов, включая пробелы (name)
- произвольный динамический массив символов
- числовые поля типов int и float (не менее двух полей каждого типа)
- поле с числовым массивом.

Написать программу, обеспечивающую начальное формирование массива структур при чтении из файла (текст с разделителями — CSV) с последующим возможным дополнением элементов массива при вводе с клавиатуры. Следует использовать указатели на структуры и указатели на функции обработки массива в соответствии с вариантом задания.

Во всех случаях, когда при поиске записей результат отсутствует, следует вывести сообщение.

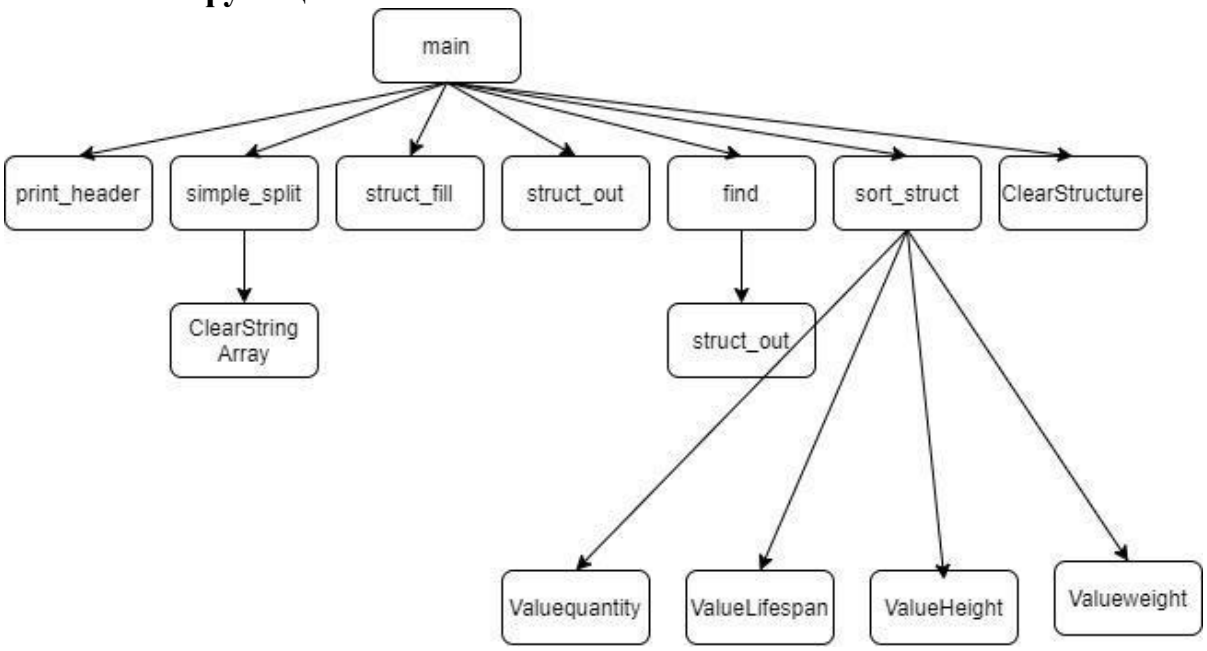
Выбор записей по значению первого слова поля name, сортировка результата по возрастанию значений любого из числовых полей (выбор поля для сортировки — из меню).

Описание структур

Описание структуры данных

Имя поля	**Тип**	**Назначение**
name	char*	Название животного
kind	char*	Царство, к которому оно принадлежит
lifespan	int	Продолжительность жизни(средняя)
quantity	int	Количество особей
weight	float	Средний вес
height	float	Средний рост
numbers	int	Количество видов + сводки по нахождению животных в странах/континентах

Схемы вызова функций



Функции

1. main
Описание:

Прототип:
int main()

Примеры вызова:
main()

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная переменная	stud0	animals**	Двумерный динамический массив структур
Локальная переменная	kind	sorting*	Переменная для сортировки
Локальная переменная	slen	int	Длина строки
Локальная переменная	n	int	Кол-во структур
Локальная переменная	count	int	Номер структуры
Локальная переменная	choice	int	Переменная-выбор
Локальная переменная	sd	int	Кол-во новых структур
Локальная переменная	ch1	int	Переменная-выбор
Локальная переменная	s2	char**	Массив строк, каждая из которых служит элементом структуры
Локальная переменная	names	char**	Массив из имён структур(для удобной сортировки)
Локальная переменная	s1	char	Строка из элементов структуры

Локальная переменная	s11	char	То же, что и s1, но для ввода дополнительных структур
Локальная переменная	sep	char	Символ-разделитель
Локальная переменная	df	FILE*	Переменная для файла
Итератор	i	int	
Итератор	j	int	

Возвращаемое значение: 0

2.ClearStringArray

Описание:
Очистка памяти для разделённого массива строк

Прототип:
void ClearStringArray (char **str, int n)

Примеры вызова:
ClearStringArray(str_array, count);

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Итератор	i	int	
Формальный аргумент	str	char**	Массив строк, каждая из которых элемент структуры
Формальный аргумент	n	int	Кол-во строк

Возвращаемое значение: отсутствует

3.simple_split

Описание:
Функция получающая на вход строку, которую нужно разделить, возвращает массив строк из элементов структуры. Разделение по разделителю

Прототип:
char** simple_split(char *str, int length, char sep)

Примеры вызова
simple_split(s1, slen, sep)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	str	char*	Строка для разделения по разделителю
Формальный аргумент	length	integer	Длина строки
Формальный аргумент	sep	char	Символ-разделитель
Локальная переменная	str_array	char**	Массив строк, получающийся из str
Локальная переменная	k	int	Помощь для цикла
Локальная переменная	m	int	Помощь для цикла
Локальная переменная	key	int	То же, что и булевая переменная
Локальная переменная	count	int	Итератор с сохраняемым значением
Итератор	i	int	
Итератор	j	int	

Возвращаемое значение: Массив строк получающийся из разделённой строки по символу-разделителю

4.print_header

Описание:
Вывод шапки для удобства чтения.

Прототип:
void print_header ()

Пример вызова:
print_header()

Возвращаемое значение: отсутствует

5.struct_out

Описание:
Выводит элементы структуры после шапки.

Прототип:
void struct_out ()

Пример вызова:
struct_out()

Возвращаемое значение: отсутствует.

6.struct_fill

Описание:
Получает на вход разделённые по символам разделителям строки, помещая каждую из них в нужный элемент структуры

Прототип:
animals* struct_fill(char **str)

Пример вызова:
struct_fill(str)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	str	char**	Разделённые по символу-разделителю строки
Локальная переменная	str0	animals*	Возвращаемое значение функции. Структура с элементами

Возвращаемое значение: Заполненная информацией структура

7.Value/Lifespan/Quantity/Weight/Height

Описание:
Возвращает определённый элемент структуры, по которому далее сортируются все структуры.

Прототип:
void Value/Lifespan/Quantity/Weight/Height (animals **str0, int i0)

Пример вызова:
Value/Lifespan/Quantity/Weight/Height(str0, j)

Возвращаемое значение: В зависимости от имени функции возвращается такое же значение

8.SortKind

Описание:
Сортировка структур по одному из признаков

Прототип:
void SortKind(int n, animals **str0, float(*funcName)(animals**, int))

Пример вызова:
SortKind(n, stud0, kind[option - 1])

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	n	int	Количество структур
Формальный аргумент	str0	animals**	Массив структур для сортировки
Формальный аргумент	funcName	float	Тип по которому нужно сортировать
Локальная переменная	tmp_struct	animals*	Вспомогательная переменная для сортировки

Итератор	i	int	
Итератор	j	int	

Возвращаемое значение: отсутствует

9.ClearStructure

Описание:
Производит очистку памяти определённой структуры

Прототип:
void ClearStructure(animals *str0)

Пример вызова:
ClearStructure(stud0[i])

Возвращаемое значение: Отсутствует

10.sort_struct

Описание:
Функция для установки по какому типу нужно отсортировать структуры

Прототип:
void sort_struct(sorting *kind, animals **stud0, int n, int count, int sd)

Пример вызова:
sort_struct(kind, stud0, n, count, sd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	kind	sorting*	Переменная для выбора сортировки
Формальный аргумент	stud0	animals**	Массив структур

Формальный аргумент	n	int	Общее кол-во структур
Формальный аргумент	count	int	Кол-во структур без добавленных
Формальный аргумент	sd	int	Кол-во добавленных структур
Локальная переменная	option	int	Переменная-выбор
Итератор	i	int	

Возвращаемое значение: отсутствует

11. find

Описание:
Функция для поиска структуры по имени

Прототип:
void find(char **names, animals **stud0, int n)

Примеры вызова:
find(names, stud0, n);

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	names	char**	Массив строк из имён
Формальный аргумент	stud0	animals**	Массив структур
Формальный аргумент	n	int	Кол-во структур

Локальная переменная	ch1	int	Переменная-выбор
Локальная переменная	ll	int	Длина имени которое нужно найти
Локальная переменная	schet	int	Переменная для проверки
Локальная переменная	namename	char	Имя, структуру которого нужно найти
Итератор	j	int	
Итератор	i	int	

Возвращаемое значение: отсутствует

Заключение

Выводы:

При выполнении лабораторной работы были получены практические навыки в разработке алгоритма и написании программы на языке Си, а также получена информация о линейных односвязных списках.

Ссылка на github: https://github.com/RitaStreet/labs/tree/master/lab9