САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра вычислительной техники

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: «Разработка электронной картотеки»

Студент гр. 9305 _____ Любаневич Р.О.

Преподаватель _____ Перязева Ю.В

Санкт-Петербург
2020

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Любаневич Р.О.

Группа 9305

Тема работы: разработка электронной картотеки

Исходные данные:
csv.txt файл

Содержание пояснительной записки:
«Содержание», «Введение», «Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записки:
Не менее 30 страниц.

Дата выдачи задания: 01.04.2020

Дата сдачи реферата: 08.06.2020

Дата защиты реферата: 08.07.2020

Студент _____ Любаневич Р.О.

Преподаватель _____ Перязева Ю.В.

АННОТАЦИЯ

В курсовой работе я работал с двусвязным списком, на котором и была разработана картотека с основными, нужными функциями. А именно: добавление, удаление, редактирование, сортировка, поиск карточки. Код курсовой работы находится на гитхабе, ссылка на который будет прикреплена ниже.

SUMMARY

In the course work, I worked with a doubly linked list, on which a file cabinet was developed with basic, necessary functions. Namely: adding, deleting, editing, sorting, searching for a card. The coursework code is located on the github, a link to which will be attached below.

СОДЕРЖАНИЕ

	Введение	4
1.	Наименования разделов	5
1.1.		0
1.2.		0
2.		0
2.1.		0
2.2.		0
3.		0
3.1.		0
3.2.		0
	Заключение	0
	Список использованных источников	0
	Приложение А. Название приложения	0

ВВЕДЕНИЕ

Цель работы

Написать программу, позволяющую работай с картотекой на определенную тематику.

Задачи

- Изучить поставленную задачу.
- Придумать пути её решения.
- отобразить алгоритм работы картотеки в виде блок-схемы.
- Написать программную реализацию картотеки.
- Проанализировать полученные результаты.

1. КАРТОТЕКА

1.1. Структура и её тематика

Картотека основана на теме животных.

Данные структуры:

1)Название животного

2)Класс животного

3)Длительность жизни

4)Вес

5)Рост

Сама структура:

```
typedef struct animals
{
        char *name;
        char *kind;
        int lifespan;
        float characteristic[2];
}animals;
```

1.2. Функции картотеки

0)Справка

1)Добавление

2)Редактирование

3)Удаление

4)Вывод картотеки

5)Поиск

6)Сортировка

7)Сохранение картотеки в файл

8)Выход

2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

2.1. Описание решения

В качестве основы картотеки я использовал двусвязный список.

Вывод может быть прямой и перевёрнутый.

Редактирование может быть осуществлено для любой структуры и для любого поля.

Сортировка работает только по числовым полям.

Добавление структуры возможно либо в начало, либо в конец.

Поиск работает с каждой переменной структуры.

2.2. Описание структур

Описание структуры данных

animals

Имя поля	**Тип**	**Назначение**
name	char*	Название животного
kind	char*	Царство, к которому оно принадлежит
lifespan	int	Продолжительность жизни(средняя)

weight	float	Средний вес
height	float	Средний рост

Head

Имя поля	**Тип**	**Назначение**
cnt	int	Кол-во элементов
first	Node*	Первый элемент списка
last	Node*	Последний элемент списка

Node

Имя поля	**Тип**	**Назначение**
id	int	Номер элемента
structure	animals*	Структура внутри узла
next	Node*	Следующий элемент списка
prev	Node*	Предыдущий элемент списка

2.3Описание функций

1.fill_node(Head *ddd)
Описание:
По сути тот же мейн, где совершаются все операции, начиная от получения текста из файла, заканчивая выводом списка.

Прототип:

void fill_node(Head *ddd)

Примеры вызова:

fill_node (ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная переменная	node	Node*	Элемент списка
Локальная переменная	p	Node*	Список
Локальная переменная	ddd	Head*	Голова списка
Локальная переменная	slen	int	Длина строки
Локальная переменная	n	int	Кол-во элементов
Локальная переменная	count	int	Номер структуры

Локальная переменная	choice	int	Переменная-выбор
Локальная переменная	s2	char**	Массив строк, каждая из которых служит элементом структуры
Локальная переменная	s1	char	Строка из элементов структуры

2.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

3.print_header

Описание:
Вывод шапки для удобства чтения.

Прототип:

void print_header ()

Пример вызова:

print_header()

Возвращаемое значение: отсутствует

4. menu(Head *ddd)

Описание:
Основная функция, из которой пользователь уже решает, что делать с программой.

Прототип:

void menu(Head *ddd)

Примеры вызова:

menu(ddd)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	ddd	Head*	Голова списка

5. case0(Head *ddd)

Описание:
Выводит справку о программе.

Прототип:

void case0(Head *ddd)

Примеры вызова:

case0(ddd)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	ddd	Head*	Голова списка

6. case1(Head *ddd)

Описание:
Функция, которая даёт право пользователю на добавление карточки.

Прототип:

void case1(Head *ddd)

Примеры вызова:

case1(ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка

7. case2(Head *ddd)

Описание:
Функция, которая даёт право пользователю на редактирование карточки.

Прототип:

void case2(Head *ddd)

Примеры вызова:

case2(ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка

8. case3(Head *ddd)

Описание:
Функция, которая даёт право пользователю на удаление карточки.

Прототип:

void case3(Head *ddd)

Примеры вызова:

case3(ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка

9. case4(Head *ddd)

Описание:
Функция, которая даёт право пользователю на вывод картотеки.

Прототип:

void case4(Head *ddd)

Примеры вызова:

case4(ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка

10. case5(Head *ddd)

Описание:
Функция, которая даёт право пользователю на поиск карточки по параметру.

Прототип:

void case5(Head *ddd)

Примеры вызова:

case5(ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка

11. case6(Head *ddd)

Описание:
Функция, которая даёт право пользователю на поиск сортировку картотеки по параметру.

Прототип:

void case6(Head *ddd)

Примеры вызова:

case6(ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка

12. case7(Head *ddd)

Описание:
Функция, которая даёт право пользователю сохранить картотеку в файл.

Прототип:

void case7(Head *ddd)

Примеры вызова:

case7(ddd)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	ddd	Head*	Голова списка

13. back_to_menu(Head *ddd)

Описание:
Функция, которая даёт право пользователю вернуться в меню, либо выйти из программы.

Прототип:

void back_to_menu(Head *ddd)

Примеры вызова:

back_to_menu (ddd)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	ddd	Head*	Голова списка
Локальная переменная	choice	int	Переменная-выбор

14. correct (Head *ddd, int a, int b)

Описание:
Функция, которая не даёт право пользователю ввести значения, несоответствующее указанным границам.

Прототип:

int correct(Head *ddd, int a, int b)

Примеры вызова:

correct(ddd, a, b)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	ddd	Head*	Голова списка
Формальный аргумент	a	int	Нижняя граница
Формальный аргумент	b	int	Верхняя граница

Возвращаемое значение: Итоговая переменная, передаваемая в меню.

15. create_node(char *str, int i, Node *p, Head *ddd)

Описание:
Создаёт элемент списка из структуры

Прототип:

Node *create_node(char *str, int i, Node *p, Head *ddd)

Примеры вызова:

create_node(str, i, p, ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка
Формальный аргумент	i	int	Номер
Формальный аргумент	p	Node*	Элемент списка
Формальный аргумент	str	char*	Текст для структуры

Возвращаемое значение: Созданный элемент узла.

16. add_first(Head *ddd)

Описание:
Добавляет новый элемент списка в начало.

Прототип:

void add_first(Head *ddd)

Примеры вызова:

add_first(ddd)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	ddd	Head*	Голова списка

17. add_last(Head *ddd)

Описание:
Добавляет новый элемент списка в конец.

Прототип:

void add_last(Head *ddd)

Примеры вызова:

add_last(ddd)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	ddd	Head*	Голова списка

18. fill_node(Head *ddd)

Описание:
Забивает все структуры в список.

Прототип:

void fill_node(Head *ddd)

Примеры вызова:

fill_node(ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка

19. free_node(Head *ddd)

Описание:
Очищает элемент списка.

Прототип:

void free_node(Head *ddd)

Примеры вызова:

free_node(ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка

20. node_out(Head *ddd)

Описание:
Выводит список в консоль.

Прототип:

void node_out(Head *ddd)

Примеры вызова:

node_out(ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка

21. node_out_reverse(Head *ddd)

Описание:
Выводит список в консоль с конца.

Прототип:

void node_out_reverse(Head *ddd)

Примеры вызова:

node_out_reverse(ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка

22. free_list (Head *ddd)

Описание:
Очищает память списка.

Прототип:

void free_list (Head *ddd)

Примеры вызова:

free_list(ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка

23. edit_structure (Head *ddd, int choice, int ch1)

Описание:
Изменяет любой из параметров списка

Прототип:

void edit_structure (Head *ddd, int choice, int ch1)

Примеры вызова:

free_list(ddd, choice, ch1)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка

Формальный аргумент	choice	int	Переменная-выбор
Формальный аргумент	ch1	int	Переменная-выбор

24. struct_out (Node *str0)

Описание:
Вывод структуру списка

Прототип:

void struct_out (Node *str0)

Примеры вызова:

struct_out (str0)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	str0	Node*	Элемент списка

25. find_name (Head *ddd)

Описание:
Находит карточку по имени.

Прототип:

void find_name (Head *ddd)

Примеры вызова:

find_name (ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка
Локальная переменная	ch1	int	Переменная-выбор
Локальная переменная	schet	int	Счётчик
Локальная переменная	ll	int	Длина имени
Локальная переменная	namename	char[]	Имя

26. find_kind (Head *ddd)

Описание:
Находит карточку по классу.

Прототип:

void find_kind (Head *ddd)

Примеры вызова:

find_kind (ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка
Локальная переменная	ch1	int	Переменная-выбор
Локальная переменная	schet	int	Счётчик
Локальная переменная	ll	int	Длина имени
Локальная переменная	namename	char[]	Имя

27. find_lifespan (Head *ddd)

Описание:
Находит карточку по имени.

Прототип:

void find_lifespan (Head *ddd)

Примеры вызова:

find_kind (ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	str0	Node*	Элемент списка
Локальная переменная	poisk	int	Переменная-поиск
Локальная переменная	schet	int	Счётчик
Локальная переменная	temp	Node*	Вспомогательный элемент списка

28. find_weight (Head *ddd)

Описание:
Находит карточку по имени.

Прототип:

void find_weight (Head *ddd)

Примеры вызова:

find_weight (ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка
Локальная переменная	poisk	int	Переменная-поиск
Локальная переменная	schet	int	Счётчик
Локальная переменная	temp	Node*	Вспомогательный элемент списка

29. find_height (Head *ddd)

Описание:
Находит карточку по имени.

Прототип:

void find_height (Head *ddd)

Примеры вызова:

find_height (ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка
Локальная переменная	poisk	int	Переменная-поиск
Локальная переменная	schet	int	Счётчик
Локальная переменная	temp	Node*	Вспомогательный элемент списка

30. sort_by_lifespan (Head *ddd)

Описание:
Сортирует картотеку по имени.

Прототип:

void sort_by_lifespan (Head *ddd)

Примеры вызова:

sort_by_lifespan(ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка

31. sort_by_weight (Head *ddd)

Описание:
Сортирует картотеку по весу.

Прототип:

void sort_by_weight (Head *ddd)

Примеры вызова:

sort_by_weight(ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка

32. sort_by_height (Head *ddd)

Описание:
Сортирует картотеку по росту.

Прототип:

void sort_by_height (Head *ddd)

Примеры вызова:

sort_by_height(ddd)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка

33. text(Head *ddd)

Описание:
Сортирует картотеку по росту.

Прототип:

void sort_by_height (Head *ddd)

Примеры вызова:

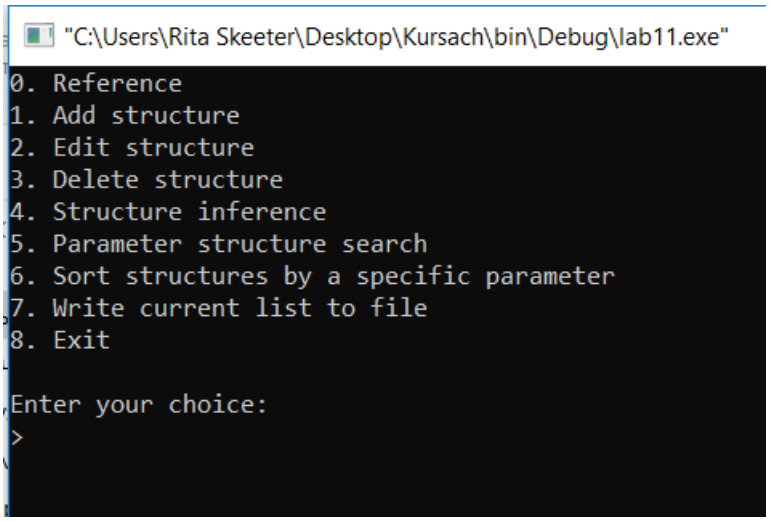
sort_by_height(ddd)

Описание переменных:

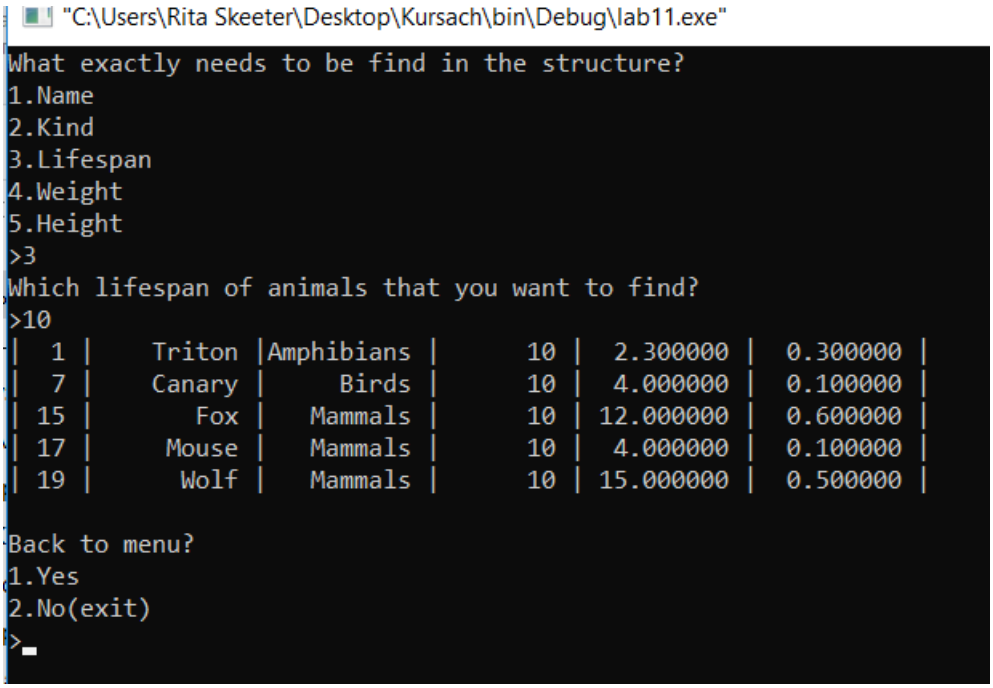
Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	ddd	Head*	Голова списка

2.4Пример работы программы

Меню:



Поиск по продолжительности жизни:



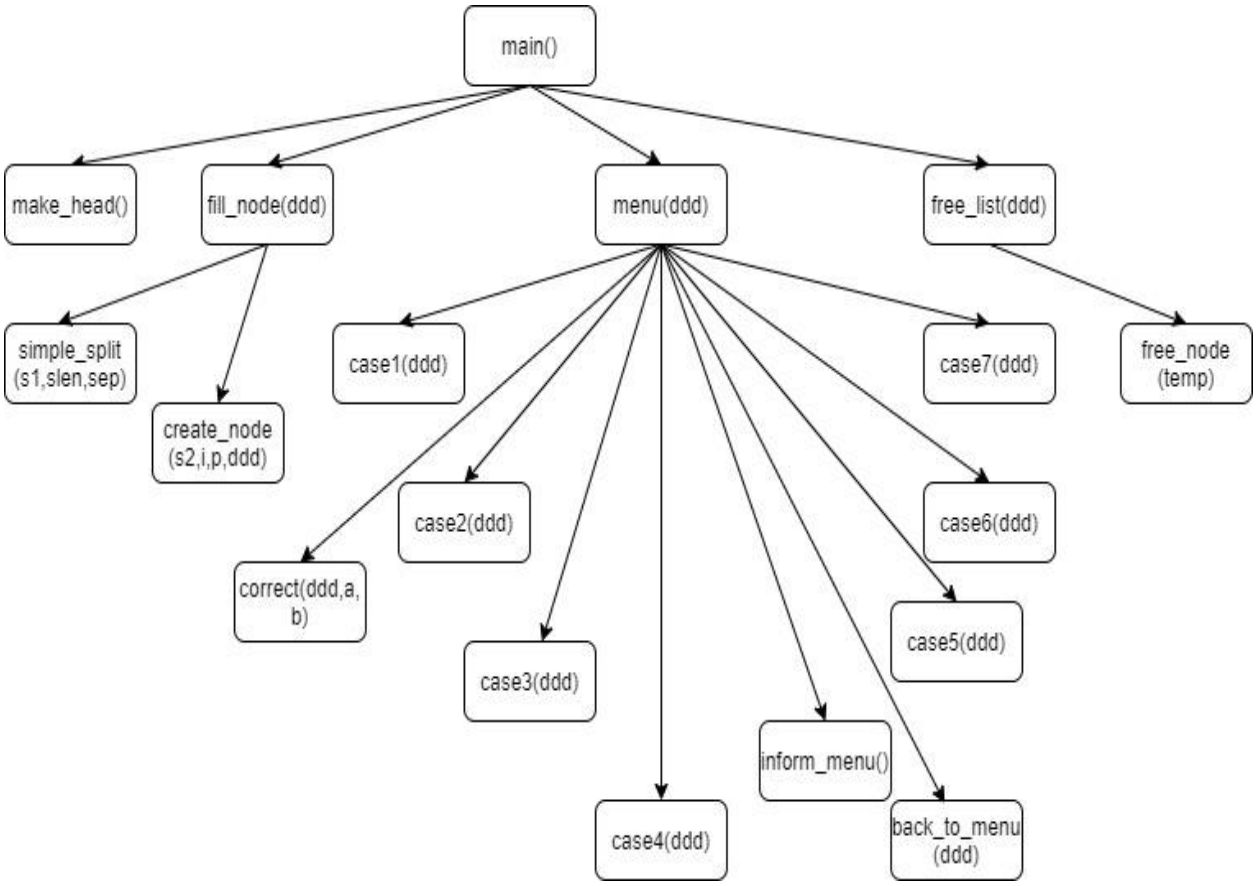
ЗАКЛЮЧЕНИЕ

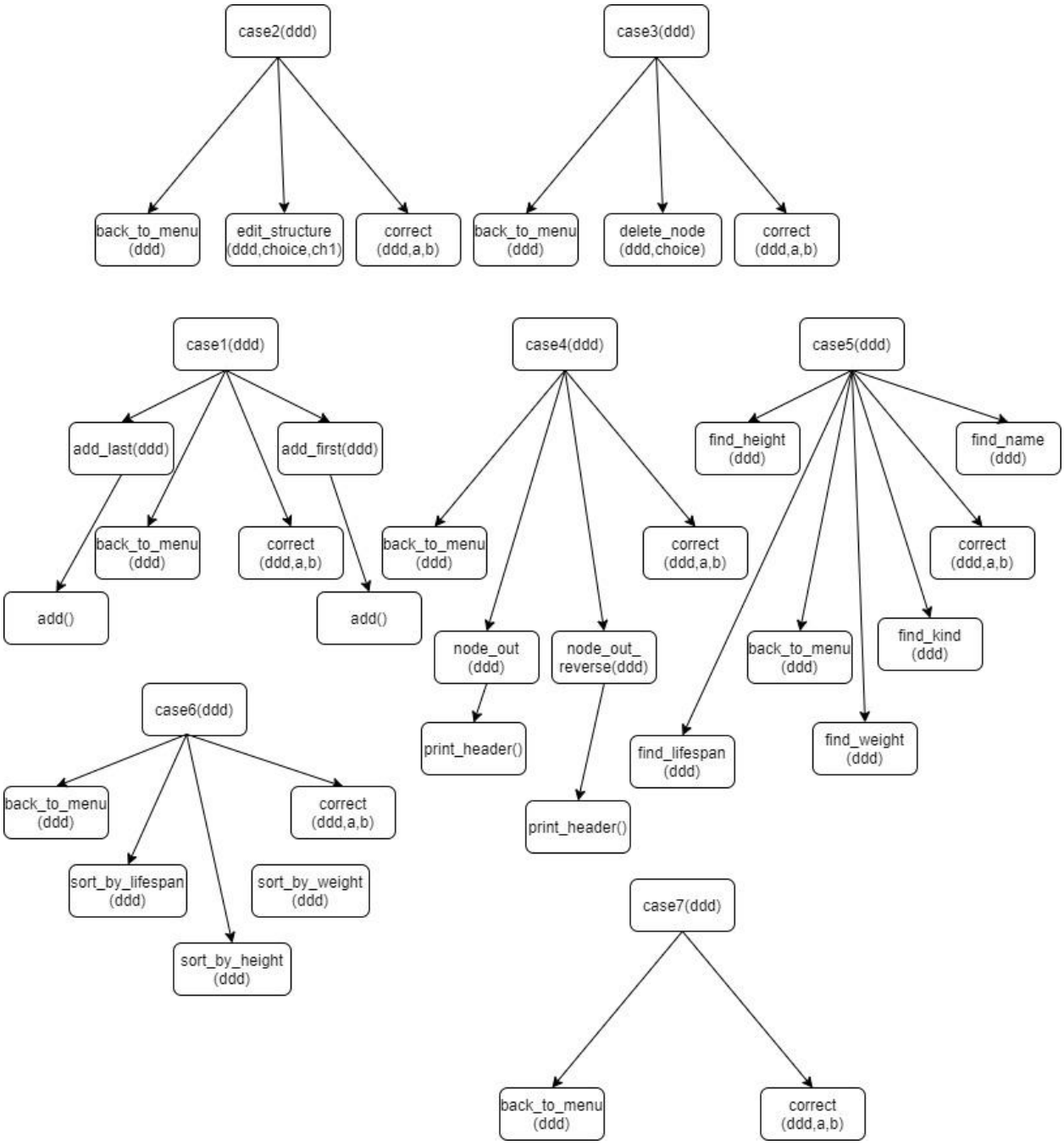
В ходе создания картотеки мною была проведена большая работа, которая оказалась для меня очень интересно, много узнал для себя, окончательно разобрался с двусвязными списками, и понял как использовать их эффективно, хоть моя программа далеко не идеальна, но работает она стабильно, без ошибок. Хотел бы поработать больше над чётким вводом, но пока ограничился только проверкой на числа. Если бы я знал как сделать интерфейс, то возможно использовал бы её в своей жизни.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. http://all-ht.ru/about.html
2. https://prog-cpp.ru/

ПРИЛОЖЕНИЕ А
СХЕМЫ ВЫЗОВА ФУНКЦИЙ


main()
make_head()
fill_node(ddd)
menu(ddd)
free_list(ddd)
simple_split (s1,slen,sep)
create_node (s2,i,p,ddd)
case1(ddd)
case7(ddd)
free_node (temp)
case2(ddd)
case6(ddd)
correct(ddd,a, b)
case5(ddd)
case3(ddd)
inform_menu()
case4(ddd)
back_to_menu (ddd)


case2(ddd)
back_to_menu (ddd)
edit_structure (ddd,choice,ch1)
correct (ddd,a,b)
case3(ddd)
back_to_menu (ddd)
delete_node (ddd,choice)
correct (ddd,a,b)
case1(ddd)
add_last(ddd)
add_first(ddd)
back_to_menu (ddd)
correct (ddd,a,b)
add()
add()
case4(ddd)
back_to_menu (ddd)
correct (ddd,a,b)
node_out (ddd)
node_out_ reverse(ddd)
print_header()
print_header()
case5(ddd)
find_height (ddd)
find_name (ddd)
correct (ddd,a,b)
back_to_menu (ddd)
find_kind (ddd)
find_lifespan (ddd)
find_weight (ddd)
case6(ddd)
back_to_menu (ddd)
correct (ddd,a,b)
sort_by_lifespan (ddd)
sort_by_weight (ddd)
sort_by_height (ddd)
case7(ddd)
back_to_menu (ddd)
correct (ddd,a,b)

ПРИЛОЖЕНИЕ В

Код: https://github.com/RitaStreet/Kursach