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 Игнатьев.К.А

Преподаватель Перязева.Ю.В

Санкт-Петербург

2020

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Игнатьев.К.А

Группа 9305

Тема работы: «Разработка электронной картотеки»

Исходные данные:

csv файл, в котором содержится начальная картотека

Содержание пояснительной записки:

«Введение», «Электронная картотека» , «Программная реализация», «Приложения», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:

Не менее 38 страниц.

Дата выдачи задания: 01.04.2020

Дата сдачи реферата: 01.06.2020

Дата защиты реферата: 03.06.2020

Студент _____ Игнатьев К.А.

Преподаватель _____ Перязева Ю.В.

АННОТАЦИЯ

Курсовая работа представляет собой программную реализацию электронной картотеки с добавлением в нее различного функционала (добавление, удаление, сортировку, поиск, изменение элементов картотеки). Картотека считывается из файла и в конце работы программы отредактированная записывается в другой файл. В содержание работы входят: описание картотеки, описание функций, примеры работы программы и блок-схемы, а также ссылка на github с самим кодом.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: написать программу, которая считывает электронную картотеку определенной тематики из файла редактирует ее и записывает в файл.

Задачи:

1.Изучение теории, определение принципа работы редактирования картотеки.

2.Разработка алгоритма, разработка блок-схем, разработка схемы вызова функций.

3.Реализация алгоритма на языке Си.

4.Анализ получившейся программы и ее отладка.

1.ЭЛЕКТРОННАЯ КАРТОТЕКА

1.1.Тематика и структуры

Тематика электронной картотеки: «самые успешные клубы Лиги Чемпионов».

Поля картотеки:

1)Кол-во титулов

2)Название команды

3)Страна

4)Год основания

5)Бюджет

6)Вместительность стадиона

7)Матчи

8)Голы

9)Финалы

Структура данных

```
Struct lnode
{
    int   id;
    int number_of_titles;
    char *team;
    char *country;
    int year;
    float budget;
    float capacity;
    int *matches_goals_finals;
    struct lnode *prev;
    struct lnode *next;
};
```

1.2. Функционал картотеки

1) Справка
2) Добавление элементов
3) Изменение элементов
4) Удаление элементов
5) Вывод элементов
6) Поиск элементов по параметру
7) Сортировка элементов по параметру
8) Сохранение измененной картотеки в файл

2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

2.1. Описание решения

Для хранения информации в электронной картотеке был использован двусвязный кольцевой список.

Для реализации программы были созданы различные функции работы со списками, описанные в файле list.c , также были реализованы функции для работы меню программы, которые находятся в файле menu.c.

Сначала посчитаем кол-во строк в начальном файле и создадим из начального файла двусвязный кольцевой список:

```
n=number_of_structures(df);
p0=make_head();
node_fill2(p0,df,n);
```

Далее вызываем функцию меню, с которой взаимодействует пользователь и после выхода из нее очищаем список и закрываем файл:

```
menu(p0);
clean_node(p0);
fclose(df);
```

Добавление элементов списка производится либо в начало, либо в конец.

Удаление производится по номеру карточки.

Если пользователь выбирает функцию поиска элемента по значению, то в случае, когда поиск производится по полям с целочисленными типами данных, пользователю предлагается ввести значение символьного поля страны дополнительно.

Редактирование работает по номеру карточки для выбранного поля.

Сортировка возможна по всем полям картотеки, кроме вещественных.

2.2. Описание пользовательских полей типов данных

LNode

Поле	Тип	Назначение
number_of_titles	int	кол-во титулов
*team	char*	название команды
*country	char*	страна
year	int	год основания
budget	float	бюджет
capacity	float	вместительность стадиона
*matches_goals_finals	int*	матчи, голы, финалы
*prev	LNode	ссылка на предыдущий элемент
*next	LNode	ссылка на следующий элемент
id	int	номер элемента

LHead

Поле	Тип	Назначение
*first	LNode	указатель на первый элемент
*last	LNode	указатель на последний элемент
cnt	int	кол-во элементов

2.3. Описание функций списка

1.Функция main

Описание:

Является точкой входа в программу. Функция описана в разделе «Постановка задачи и описание решения».

Прототип:

int main()

Пример вызова:

main()

Описание переменных:

Имя переменной	Тип	Назначение
*p0	Head	голова списка
n	int	кол-во строк в файле
*df	FILE	считываемый файл

2.Функция simple_split

Описание:

Алгоритм, реализующий составление массива строк (полей структуры) путем разбиения строк из файла по заданному символу.

Прототип:

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

Пример вызова:

simple_split(s1,slen,sep)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*str	char	строка из файла
формальный аргумент	lenght	int	длина строки
формальный аргумент	sep	char	символ разделитель
локальная	**str_array	char	массив строк
локальная	i	int	индекс массива
локальная	j	int	номер столбца массива
локальная	k	int	номер столбца массива
локальная	m	int	номер строки массива

локальная	key	int	успешно ли выделилась память
локальная	count	int	счетчик успешно выделенной памяти

Возвращаемое значение:

Массив строк

3.Функция ClearStringArray

Описание:

Алгоритм, реализующий очистку памяти строк.

Прототип:

void ClearStringArray(char **str, int n)

Пример вызова:

ClearStringArray(str_array,count);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*str	char	массив строк
формальный аргумент	n	int	длина строки
локальная	i	int	номер строки

Возвращаемое значение:

нет

4.Функция slens

Описание:

Подсчет кол-ва символов в строке.

Прототип:

int slens(char *s1)

Пример вызова:

slen=slens(s1)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение

формальный аргумент	*s1	char	строка символов
локальная	slen	int	кол-во символов в строке

Возвращаемое значение:

Кол-во символов в строке.

5.Функция create_node2

Описание:

Формирование узла списка.

Прототип:

Node *create_node2(char *new_word, int slen)

Пример вызова:

my_node=create_node2(my_word,slen)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	**str	char**	массив строк

Возвращаемое значение:

Ссылка на узел списка

6.Функция number_of_structures

Описание:

Подсчет кол-ва структур в файле.

Прототип:

int number_of_structures(FILE *df)

Пример вызова:

n=number_of_structures(df);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	S1[MAXLEN]	char	строка из файла
локальная	*n	int	кол-во структур

Возвращаемое значение:

Кол-во структур в файле.

7.Функция add_last2

Описание:

Добавление элемента в конец списка.

Прототип:

void add_last2(Head *my_head, Node *new_node, Node *prev_node)

Пример вызова:

add_last2(p0, my_node, p);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*prev_node	Node	последний элемент списка
формальный аргумент	*new_node	Node	новый последний элемент списка
формальный аргумент	*my_head	Head	голова списка

Возвращаемое значение:

Нет

8.Функция make_head

Описание:

Формирование головы списка.

Прототип:

Head *make_head()

Пример вызова:

p0=make_head();

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
локальный аргумент	*ph	Head	голова списка

Возвращаемое значение:

Ссылка на голову списка

9.Функция node_fill2

Описание:

Заполнение узлов списка.

Прототип:

Node *node_fill(Head *p0, FILE *df, int num)

Пример вызова:

node_fill2(p0, df ,n);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка
формальный аргумент	num	int	кол-во элементов списка
формальный аргумент	*df	FILE	файл, откуда считывается список
локальная	*p1	Node	создаваемый узел списка
локальная	*p	Node	ссылка на предыдущий элемент
локальная	**s2	char**	массив строк
локальная	*my_word	Node	элемент списка
локальная	s1	char*	считываемая из файла строка
локальная	slen	int	длина считываемой строки
локальная	t	int	первый ли элемент вводится
локальная	sep	char	разделитель

Возвращаемое значение:

Ссылка на список

10.Функция add_first2

Описание:

Добавление узла списка в начало.

Прототип:

void add_first2(Head *my_head, Node *new_node)

Пример вызова:

add_first2(p0,p1);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*new_node	Node	новый последний элемент списка
формальный аргумент	*my_head	Head	голова списка

Возвращаемое значение:

-

11.Функция clean_node2

Описание:

Очистка списка.

Прототип:

void clean_node(Head *p0)

Пример вызова:

clean_node(p0);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка
локальная	*p	Node	элемент списка
локальная	*p1	Node	временное хранилище

Возвращаемое значение:

-

12.Функция print_node2

Описание:

Вывод списка на экран.

Прототип:

void print_node2(Head *p0)

Пример вызова:

print_node2(p0);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение

формальный аргумент	*p0	Head	голова списка
локальная	*p	Node	элемент списка
локальная	i	int	счетчик

Возвращаемое значение:

-

13.Функция print_header

Описание:

Печать шапки.

Прототип:

void print_header()

Пример вызова:

print_header();

Описание переменных:

-

Возвращаемое значение:

-

14.Функция save_scanf

Описание:

Безопасный ввод.

Прототип:

int safe_scanf()

Пример вызова:

(*new_node).year=safe_scanf();

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
локальная	chooce	int	вводимое значение
локальная	str	char*	проверочная строка

Возвращаемое значение:

Вводимое число.

15.Функция swap

Описание:

Обмен значениями элементов списка.

Прототип:

void swap(Node *tmp, Node *a)

Пример вызова:

swap(p,tmp);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*tmp	Node	заменяемый элемент
формальный аргумент	*a	Node	заменяемый элемент
формальный аргумент	number_of_titles	int	кол-во титулов
локальная	*team	char*	название команды
локальная	*country	char*	страна
локальная	year	int	год основания
локальная	budget	float	бюджет
локальная	capacity	float	вместительность стадиона
локальная	matches	int	матчи
локальная	goals	int	голы
локальная	finals	int	финалы

Возвращаемое значение:

-

16.Функция team_search (country_search – аналогично)

Описание:

Поиск по заданному символьному полю.

Прототип:

void team_search(Head *p0)

Пример вызова:

team_search(p0);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка
локальная	i	int	счетчик
локальная	min	int	минимальное значение элемента поля
локальная	max	int	максимальное значение элемента поля
локальная	len	int	длина вводимого символьного поля
локальная	count	int	счетчик
локальная	s	char*	значение дополнительного поля
локальная	*p	Node	элемент списка

Возвращаемое значение:

-

17.Функция number_of_titles_search (year_search, matches_search, goals_search, finals_search,– аналогично)

Описание:

Поиск по заданному целочисленному полю.

Прототип:

void number_of_titles_search(Head *p0)

Пример вызова:

number_of_titles_search(p0);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка
локальная	i	int	счетчик
локальная	min	int	минимальное значение элемента поля
локальная	max	int	максимальное значение элемента поля
локальная	len	int	длина вводимого символьного поля
локальная	count	int	счетчик

локальная	s	char*	значение дополнительного поля
локальная	*p	Node	элемент списка
локальная	d	int	переменная выбора
локальная	check	int	проверка на корректность ввода

Возвращаемое значение:

-

18.Функция number_of_titles_sorted (year_ sorted, matches_ sorted, goals_ sorted, finals_ sorted– аналогично)

Описание:

Сортировка по заданному полю.

Прототип:

void number_of_titles_ sorted (Head *p0)

Пример вызова:

number_of_titles_ sorted (p0);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка
локальная	sort	int	по возрастанию/убыванию
локальная	p	Node	текущий элемент
локальная	o	Node	элемент для обмена
локальная	tmp	Node	следующий элемент
локальная	i	int	счетчик
локальная	j	int	счетчик

Возвращаемое значение:

-

19.Функция number_of_titles_edit (year_ edit, matches_ edit, goals_ edit, finals_ edit– аналогично)

Описание:

Изменение значения поля.

Прототип:

void number_of_titles_edit(Node *p)

Пример вызова:

number_of_titles_ edit (p0);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка

Возвращаемое значение:

-

20.Функция card_search

Описание:

Функция для поиска по заданному полю.

Прототип:

void card_search(Head *p0,int option)

Пример вызова:

card_search(p0,option);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка
формальный аргумент	option	int	поле для сортировки
локальная	(*kind[7])(Head*,int)	void	массив указателей на функции

Возвращаемое значение:

-

21.Функция card_sorted

Описание:

Функция для сортировки по заданному полю.

Прототип:

void card_sorted(Head *p0,int option)

Пример вызова:

card_sorted(p0,option);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка
формальный аргумент	option	int	поле для поиска
локальная	(*kind[7])(Head*)	void	массив указателей на функции
локальная	check	int	проверка на корректность ввода
локальная	sort	int	по возрастанию/убыванию

Возвращаемое значение:

-

22.Функция edit_card

Описание:

Функция для изменения значения элемента списка по заданному полю.

Прототип:

void edit_card(Head *p0,int num,int option)

Пример вызова:

edit_card(p0, num, option);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка
формальный аргумент	option	int	поле для поиска
формальный аргумент	num	int	номер изменяемого элемента
локальная	(*kind[9])(Node*)	void	массив указателей на функции
локальная	i	int	счетчик
локальная	*p	Node	текущий элемент списка

Возвращаемое значение:

-

23.Функция delete_card

Описание:

Функция для удаления элемента списка.

Прототип:

void delete_card(Head *p0,int n)

Пример вызова:

delete_card(p0,n);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка
формальный аргумент	n	int	номер карты
локальная	i	int	счетчик
локальная	y	int	предел
локальная	*card	Node	текущий элемент списка
локальная	*delete_card	Node	текущий элемент списка

Возвращаемое значение:

-

24.Функция create_node_enter

Описание:

Формирование узла списка(с клавитатуры).

Прототип:

Node *create_node_enter()

Пример вызова:

str0=create_node_enter();

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	**str	char**	массив строк

Возвращаемое значение:

Ссылка на узел списка

2.4. Описание функций меню

1.Функция menu_edit

Описание:

Печать меню для изменения элементов списка.

Прототип:

void menu_edit ()

Пример вызова:

menu_edit ();

Описание переменных:

-

Возвращаемое значение:

-

2.Функция menu_search

Описание:

Печать меню для поиска элементов списка.

Прототип:

void menu_search ()

Пример вызова:

menu_search ();

Описание переменных:

-

Возвращаемое значение:

-

3.Функция menu_sort

Описание:

Печать меню для сортировки элементов списка.

Прототип:

void menu_sort ()

Пример вызова:

menu_sort ();

Описание переменных:

-

Возвращаемое значение:

-

4.Функция print_reference

Описание:

Печать справки.

Прототип:

void print_reference ()

Пример вызова:

print_reference ();

Описание переменных:

-

Возвращаемое значение:

-

5.Функция print_menu

Описание:

Печать меню.

Прототип:

void print_menu ()

Пример вызова:

print_menu ();

Описание переменных:

-

Возвращаемое значение:

-

6.Функция save_scanf

Описание:

Безопасный ввод.

Прототип:

int safe_scanf()

Пример вызова:

(*new_node).year=safe_scanf();

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
локальная	choоce	int	вводимое значение
локальная	str	char*	проверочная строка

Возвращаемое значение:

Вводимое число.

7.Функция save_file

Описание:

Сохранение списка в файл.

Прототип:

void save_file(Head *p0)

Пример вызова:

save_file(p0);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка
локальная	*p	Node	элемент списка
локальная	df2	FILE	сам файл

Возвращаемое значение:

-

8.Функция sub_menu_add

Описание:

Подменю для добавления элемента в список.

Прототип:

void sub_menu_add(Head *p0,int c)

Пример вызова:

sub_menu_add(p0,c);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка
формальный аргумент	c	int	выбор пользователя
локальная	add	int	выбор(в начало, в конец)
локальная	check	int	проверка на корректность ввода

Возвращаемое значение:

-

9.Функция sub_menu_edit

Описание:

Подменю для изменения элемента списка.

Прототип:

void sub_menu_edit(Head *p0,int c)

Пример вызова:

sub_menu_edit(p0,c);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка
формальный аргумент	c	int	выбор пользователя
локальная	num	int	номер изменяемой карты
локальная	option	int	изменяемое поле(номер)
локальная	k	int	проверка на корректность ввода
локальная	p	int	проверка на корректность ввода

Возвращаемое значение:

-

10.Функция sub_menu_delete

Описание:

Подменю для удаления элемента списка.

Прототип:

void sub_menu_delete(Head *p0,int c)

Пример вызова:

sub_menu_delete(p0,c);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка
формальный аргумент	c	int	выбор пользователя
локальная	del	int	номер удаляемой карты
локальная	check2	int	проверка на корректность ввода

Возвращаемое значение:

-

11.Функция sub_menu_output

Описание:

Подменю для вывода списка.

Прототип:

void sub_menu_output(Head *p0)

Пример вызова:

sub_menu_output(p0);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка

Возвращаемое значение:

-

12.Функция sub_menu_search

Описание:

Подменю для поиска элемента списка.

Прототип:

void sub_menu_search(Head *p0,int c)

Пример вызова:

sub_menu_search(p0,c);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка
формальный аргумент	c	int	выбор пользователя
локальная	option	int	поле для поиска
локальная	check	int	проверка на корректность ввода

Возвращаемое значение:

-

13.Функция sub_menu_ sort

Описание:

Подменю для сортировки элементов списка.

Прототип:

void sub_menu_sort(Head *p0,int c)

Пример вызова:

sub_menu_sort(p0,c);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
формальный аргумент	*p0	Head	голова списка
формальный аргумент	c	int	выбор пользователя
локальная	option	int	поле для сортировки
локальная	check	int	проверка на корректность ввода

Возвращаемое значение:

-

2.5. Пример работы программы

```
Menu:

0 - Reference
1 - Adding cards
2 - Editing cards
3 - Deleting cards
4 - Output of the card
5 - Search for cards by parameter
6 - Sorting the cards by parameter
7 - Exit and save card

:
```

```
Where do you want to add the card?
1 - add first
2 - add last
3 - return back
1
Enter number_of_titles:1
Enter team:Chelsea
Enter country:England
Enter year:1905
Enter budget:428.0
Enter capacity:40.000
Enter matches:166
Enter goals:285
Enter finals:2
The card was added successfully!

Press any key to come back to main menu
```

ID	Titles	Team	Country	Year	Budget	Capacity	Matches	Goals	Finals
1	1	Chelsea	England	1905	428.0	40.000	166	285	2
2	2	Benfica	Portugal	1904	147.2	65.647	258	416	7
3	2	Nottingham Forest	England	1856	15.0	30.602	98	110	2
4	5	Barcelona	Spanish	1899	648.3	99.354	316	629	8
5	2	Porto	Portugal	1893	62.0	52.202	245	364	2
6	3	Inter	Italy	1908	185.5	80.018	178	255	5
7	6	Liverpool	England	1892	424.2	54.167	215	406	9
8	3	Manchester United	England	1878	676.3	76.212	279	506	5
9	13	Real Madrid	Spanish	1902	674.6	85.454	437	971	16
10	4	Ajax	Netherlands	1900	138.0	51.324	227	356	6
11	5	Bayern	Germany	1900	587.8	75.021	347	705	10
12	7	Milan	Italy	1899	224.1	80.018	249	416	11
13	2	Juventus	Italy	1897	405.7	41.000	277	439	9

```
Press any key to come back to main menu
```

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была получена программная реализация электронной картотеки, разработана схема вызова функций составлены блок-схемы некоторых функций. После программной реализации картотеки было проведено тестирование программы, в ходе которого были обнаружены ошибки и недочеты в написании алгоритмов, которые были устранены путем отладки. Результат выполнения курсовой работы полностью соответствует поставленной задаче: программа позволяет редактировать исходную картотеку и сохранять ее в файл.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Презентация к лекции Ивана Анатольевича Хахаева «Структуры. Массивы структур.» // moodle.eltech.ru

URL: http://moodle.eltech.ru/pluginfile.php/1946/mod_resource/content/5/lect-10.pdf

2. Презентация к лекции Ивана Анатольевича Хахаева «Перечисления. Указатели на структуры.» // moodle.eltech.ru

URL: http://moodle.eltech.ru/pluginfile.php/2182/mod_resource/content/2/lect-11.pdf

3. Презентация к лекции Ивана Анатольевича Хахаева «Указатели на функции» // moodle.eltech.ru

URL: http://moodle.eltech.ru/pluginfile.php/2197/mod_resource/content/2/lect-12.pdf

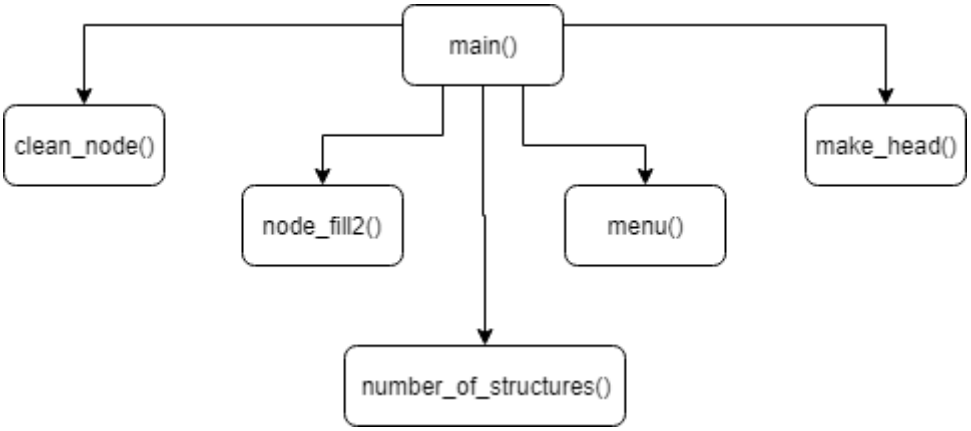
4. Презентация к лекции Ивана Анатольевича Хахаева «Абстрактные типы данных. Списки: линейные односвязные.» // moodle.eltech.ru

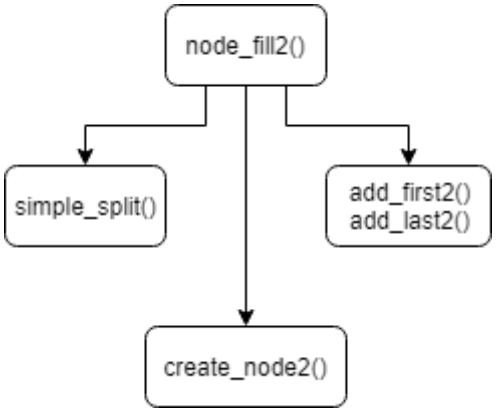
URL: http://moodle.eltech.ru/pluginfile.php/2450/mod_resource/content/5/lect-13-2020.pdf

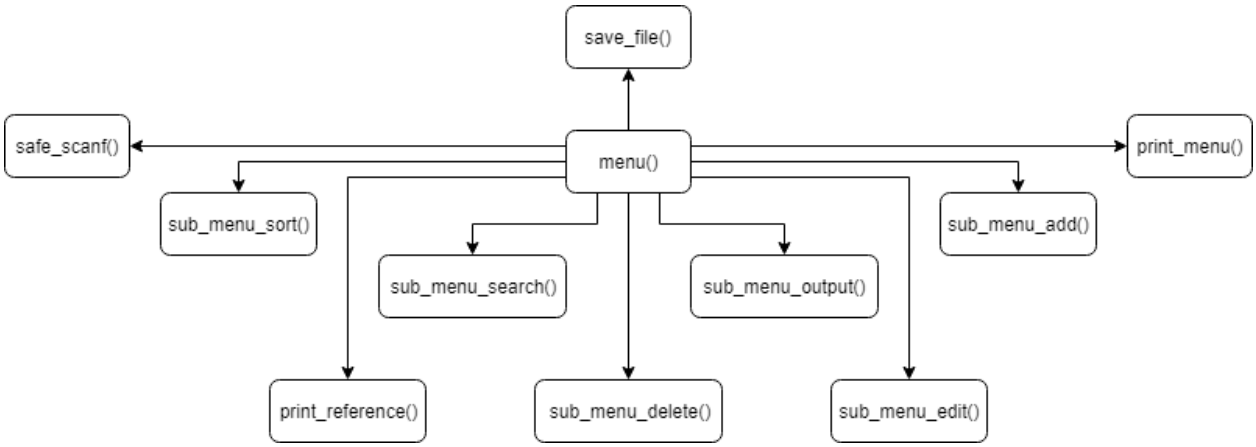
5. Презентация к лекции Ивана Анатольевича Хахаева «Двусвязные списки» // moodle.eltech.ru

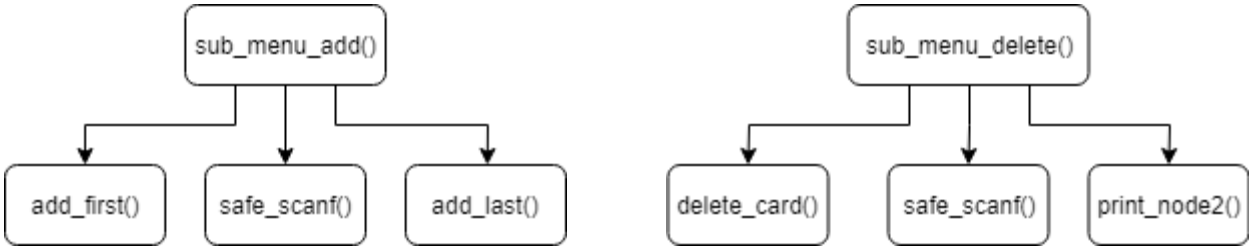
URL: http://moodle.eltech.ru/pluginfile.php/2485/mod_resource/content/3/lect-15-2020.pdf

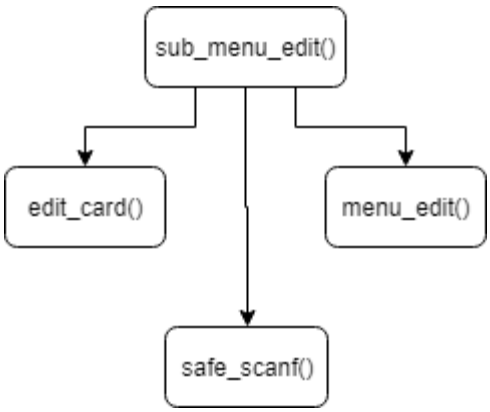
ПРИЛОЖЕНИЕ А
СХЕМЫ ВЫЗОВА ФУНКЦИЙ

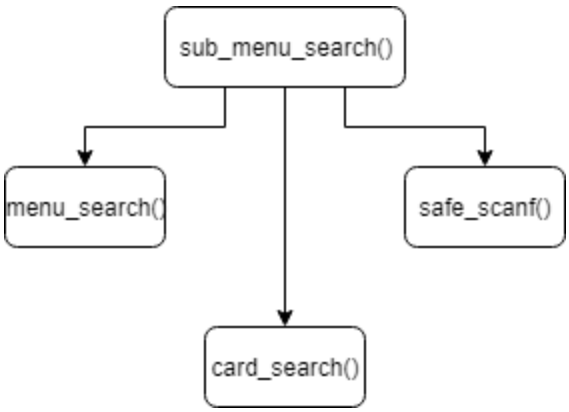


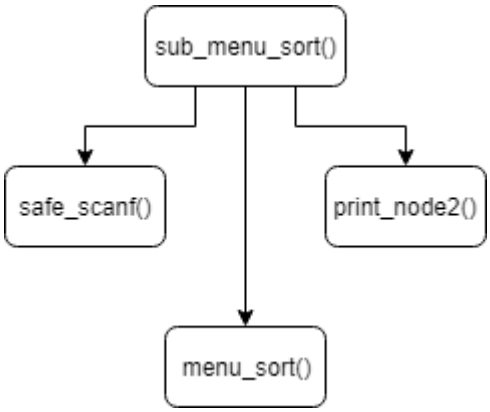


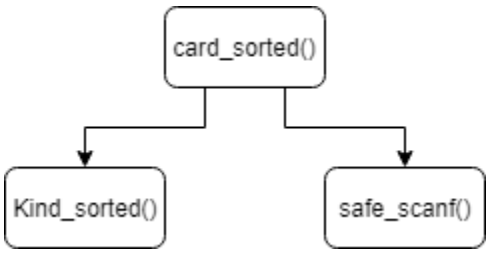


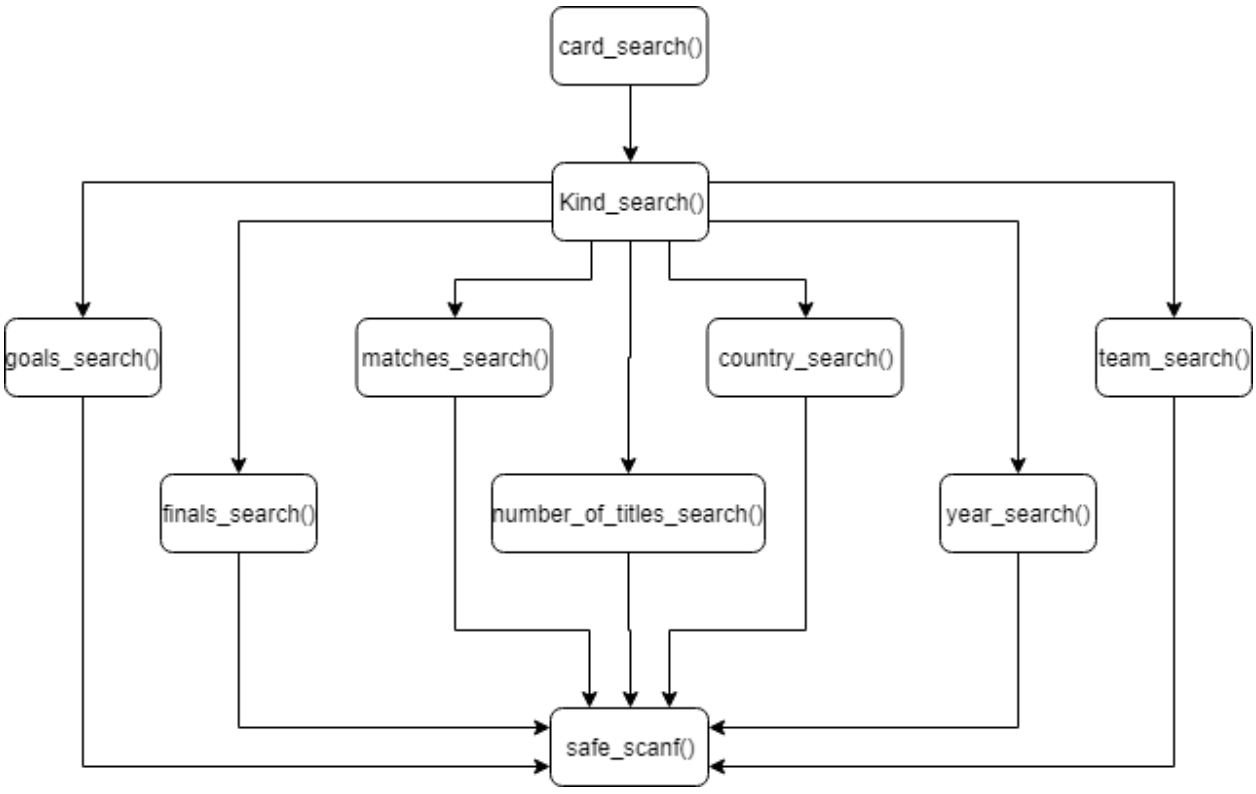


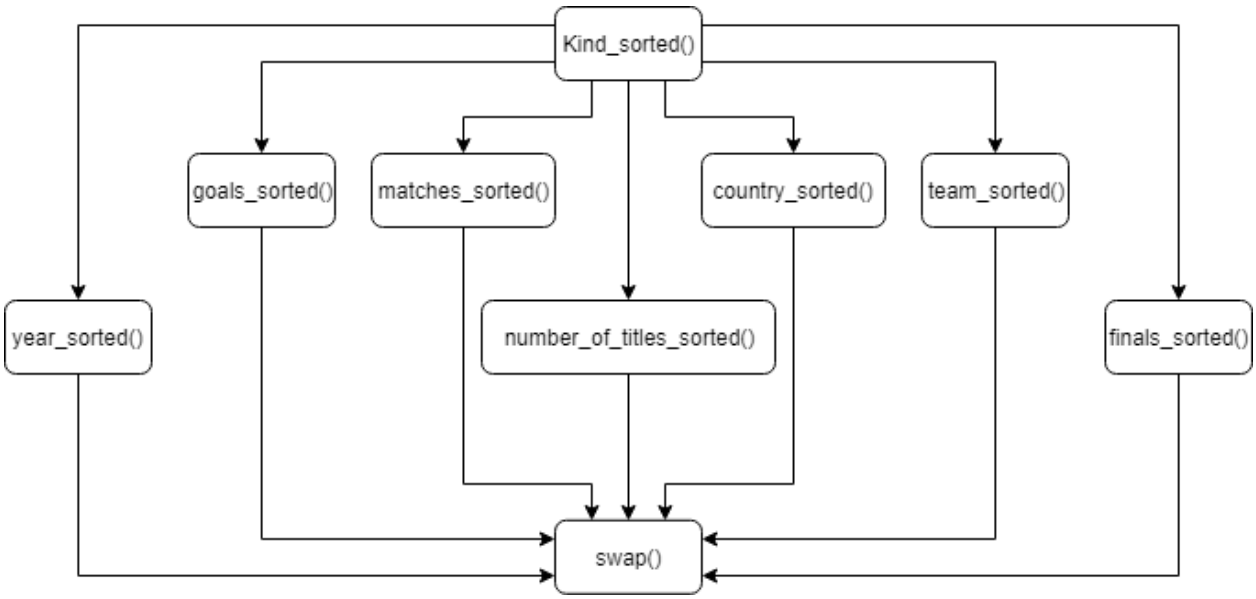

sub_menu_edit()
edit_card()
menu_edit()
safe_scanf()

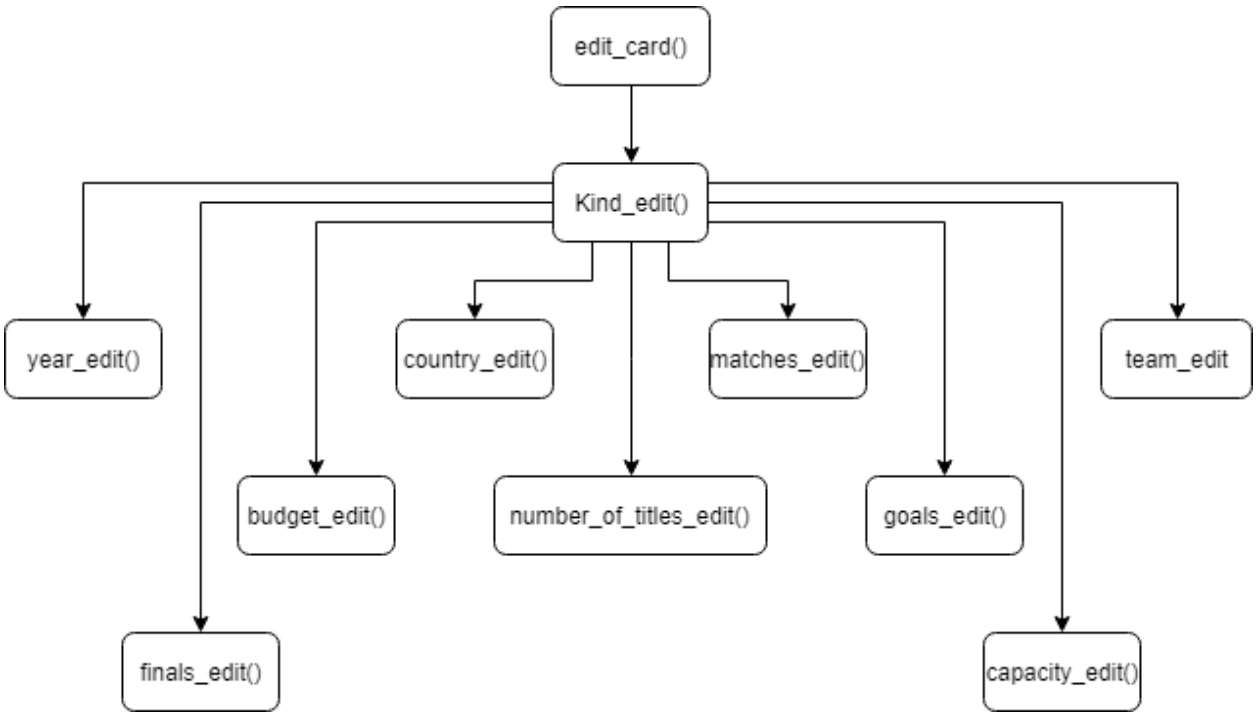

sub_menu_search()
menu_search()
safe_scanf()
card_search()


sub_menu_sort()
safe_scanf()
print_node2()
menu_sort()

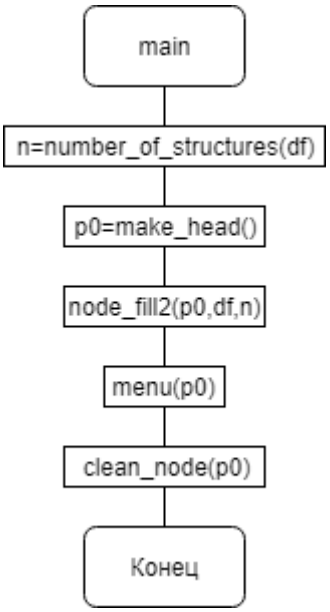

card_sorted()
Kind_sorted()
safe_scanf()

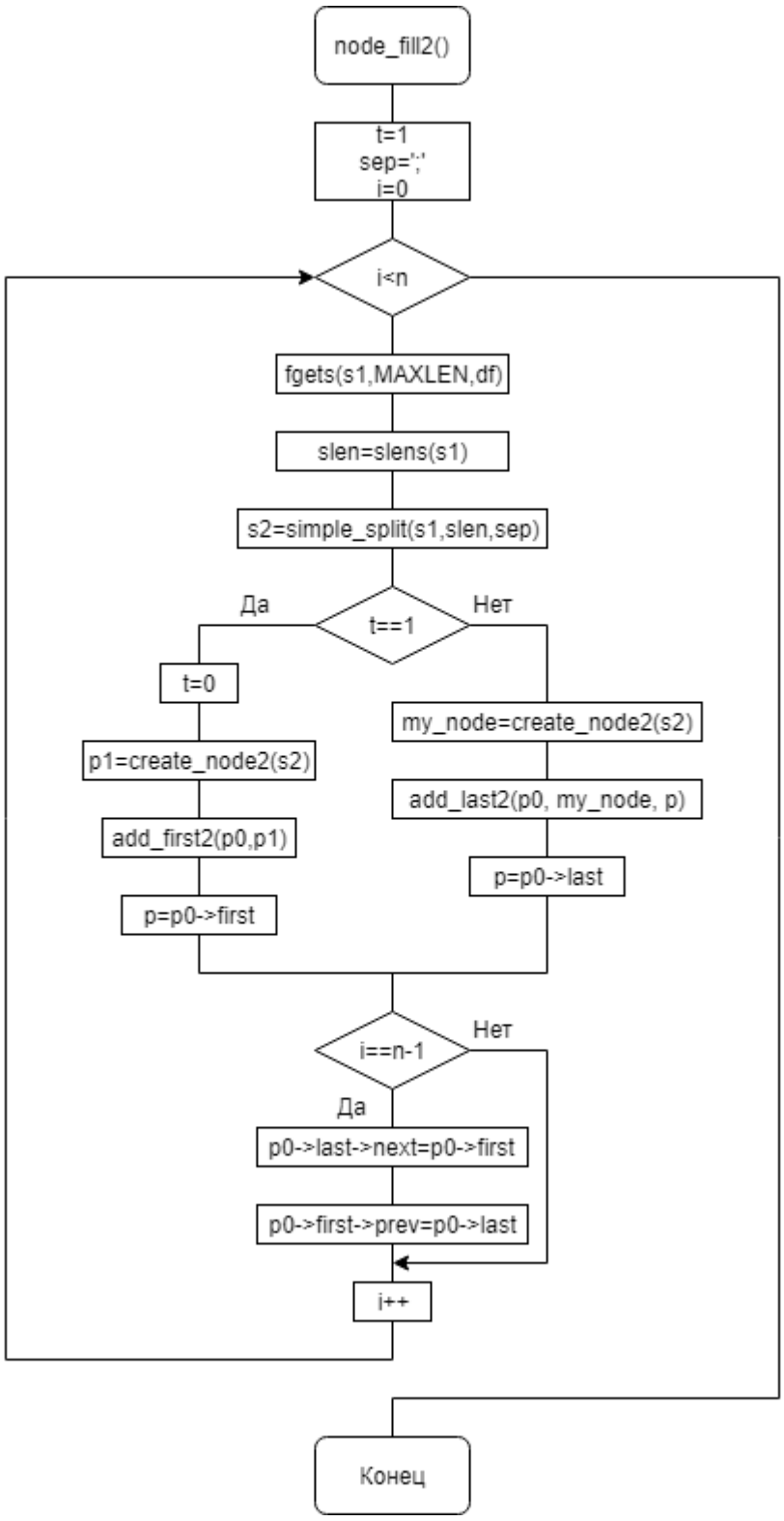

card_search()
Kind_search()
goals_search()
matches_search()
country_search()
team_search()
finals_search()
number_of_titles_search()
year_search()
safe_scanf()

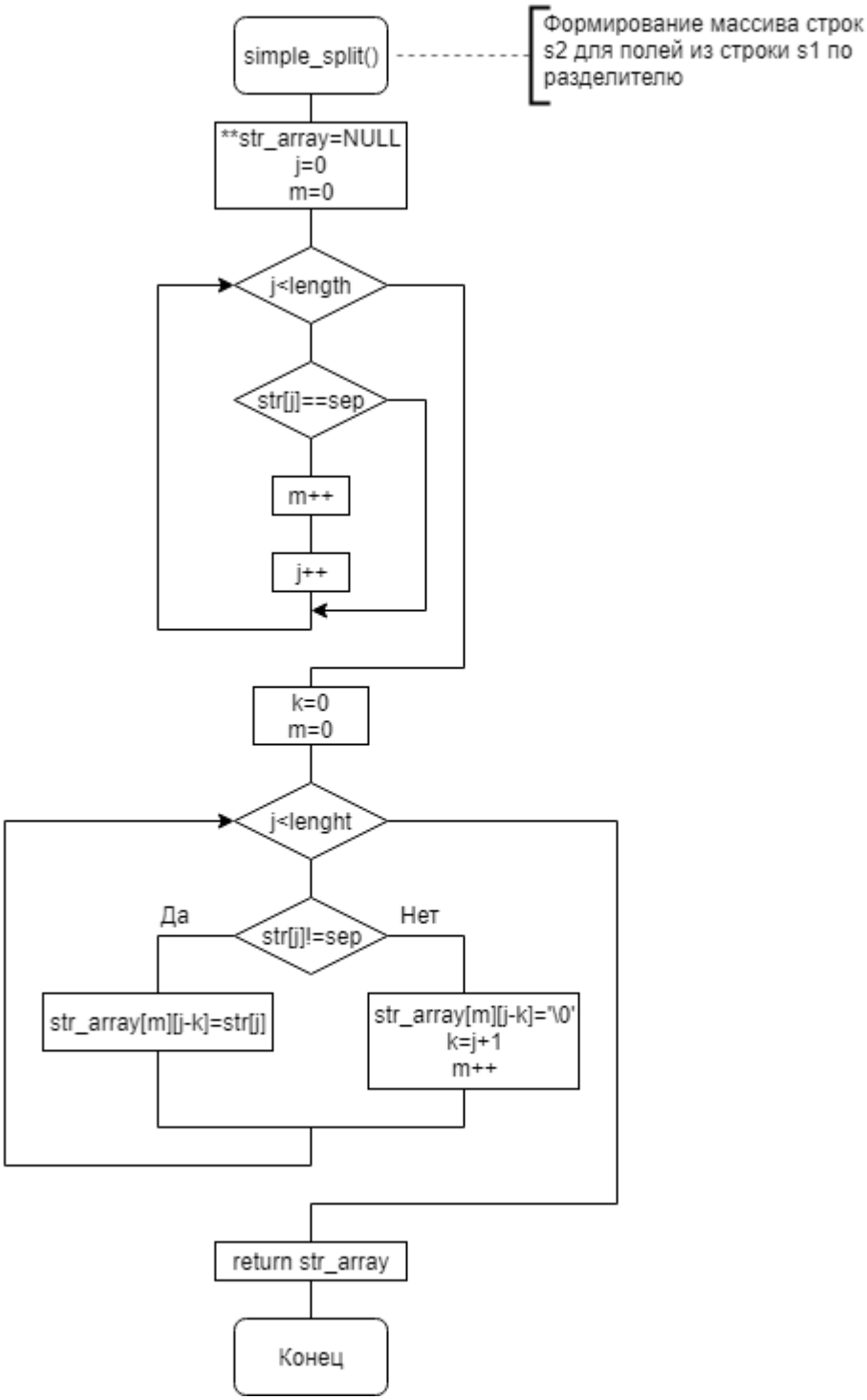

Kind_sorted()
goals_sorted()
matches_sorted()
country_sorted()
team_sorted()
year_sorted()
number_of_titles_sorted()
finals_sorted()
swap()


edit_card()
Kind_edit()
year_edit()
country_edit()
matches_edit()
team_edit
budget_edit()
number_of_titles_edit()
goals_edit()
finals_edit()
capacity_edit()

ПРИЛОЖЕНИЕ В
СХЕМЫ ФУНКЦИЙ




node_fill2()
t=1
sep=';'
i=0
i<n
fgets(s1,MAXLEN,df)
slen=slens(s1)
s2=simple_split(s1,slen,sep)
t==1
Да
Нет
t=0
p1=create_node2(s2)
add_first2(p0,p1)
p=p0->first
my_node=create_node2(s2)
add_last2(p0, my_node, p)
p=p0->last
i==n-1
Нет
Да
p0->last->next=p0->first
p0->first->prev=p0->last
i++
Конец


simple_split()
Формирование массива строк s2 для полей из строки s1 по разделителю
**str_array=NULL
j=0
m=0
j<length
str[j]==sep
m++
j++
k=0
m=0
j<lenght
Да
str[j]!=sep
Нет
str_array[m][j-k]=str[j]
str_array[m][j-k]='\0'
k=j+1
m++
return str_array
Конец

ПРИЛОЖЕНИЕ С
ТЕКСТ ПРОГРАММЫ

Ссылка на github:

https://github.com/kostyan87/myProjects/tree/master/ignatev_kursovaya