МИНОБРНАУКИ РОССИИ
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. 2305 _____ Коленко М.А.

Преподаватель _____ Хахаев И.А.

Санкт-Петербург

2023

Введение:

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). Выбранная мной предметная область картотеки была магазин игрушек (название игрушки, название производителя, количество игрушек на складе, количество вариаций игрушки, средний объём продаж игрушки в месяц, цена и ID вариаций).

Задание:

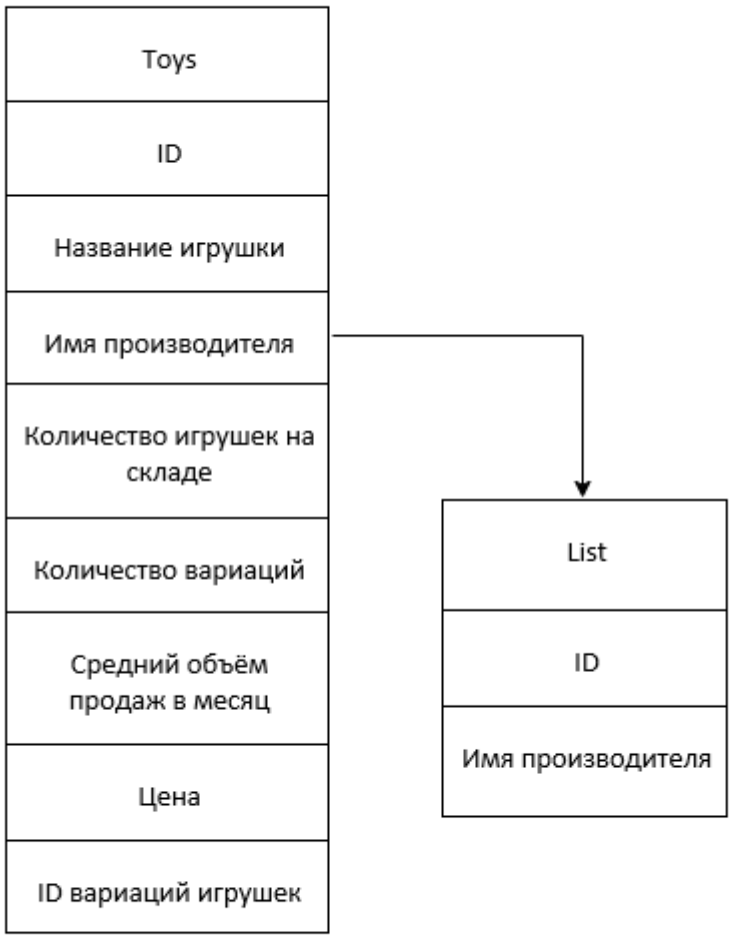
Программа должна выполнять следующие действия:

- Выводить справку (вывод на экран краткой документации по пунктам меню).
- Добавлять новые карточки (внесения нового элемента).
- Редактирование карточки (изменение элемента, находящегося в списке)
- Удаление карточки (удаление элемента).
- Вывод картотеки (вывод на экран списка).
- Поиск карточки по параметру (поиск по введённым данным в заданном пользователем поле).
- Сортировка картотеки по параметру (сортировка по заданному пользователем полю)
- Выход (сохранение изменённой картотеки и завершение работы программы).

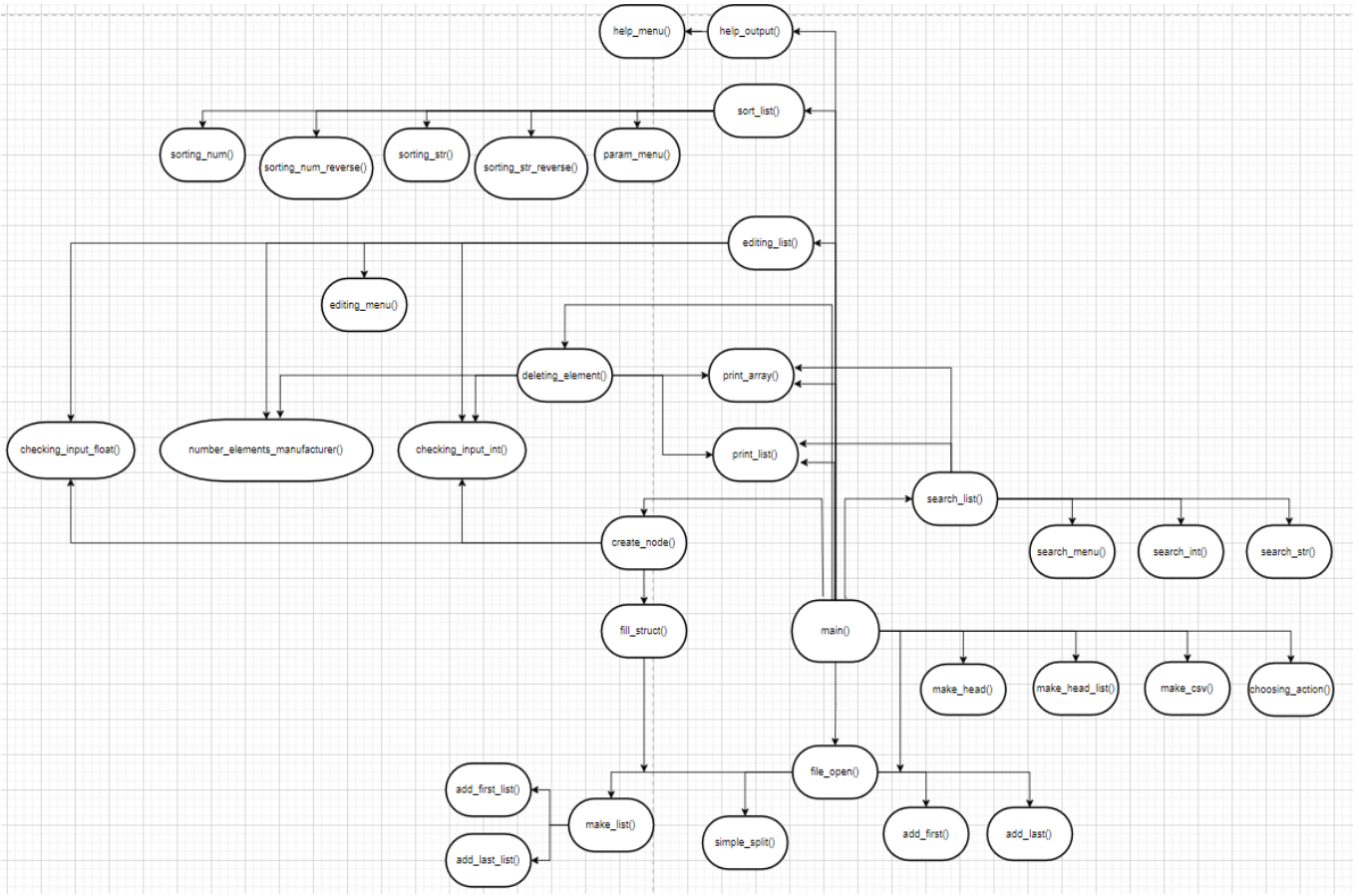
Описание общей архитектуры данных:

Работа с меню находится в основном теле программы (main), все основные действия (пункты меню) вынесены в отдельные функции: справка, добавление, редактирование, удаление, вывод, сортировка, поиск. Сохранение картотеки происходит после того, как пользователь захочет выйти. Так же после выхода происходит очистка памяти.

Сущности:



Связь функций:



Описание переменных структур:
Переменные prod элемент дополнительного двусвязного списка

№	Имя переменной	Тип	Назначение
1	id	int	Индивидуальный идентификатор
2	name	char*	Имя производителя
3	next, prev	struct prod*	Указатель на следующий (предыдущий) элемент в списке

Переменные head голова дополнительного двусвязного списка

№	Имя переменной	Тип	Назначение
1	cnt	int	Количество внесённых элементов
2	first	struct prod*	Указатель на первый элемент в списке
3	last	struct prod*	Указатель на последний элемент в списке

Переменные apartments элемент основного списка

№	Имя переменной	Тип	Назначение
1	id	int	Индивидуальный идентификатор
2	name	char*	Название игрушки
3	producer	struct prod*	Указатель на структуру с именами производителей
4	toys_count	int	Количество игрушек на складе
5	variations	int	Количество вариаций игрушки
6	sales_month	float	Среднее количество продаж в месяц
7	price	float	Цена игрушки
8	id_var	int*	Массив id вариаций игрушки
9	next, prev	struct toys	Указатель на следующий и предыдущий элементы списка

Переменные lhead голова основного списка

№	Имя переменной	Тип	Назначение
1	cnt	Int	Количество внесённых элементов
2	first	struct toys *	Указатель на первый элемент в списке
3	last	struct toys *	Указатель на последний элемент в списке

Описание переменных функций:

Переменные функции main()

№	**Название**	**Тип**	**Описание**
1	*l0	lh1	Указатель на голову структуры list в которой хранятся указатели на первый и последний элемент, и количество элементов
2	*p0	lh	Указатель на голову структуры lh в которой хранятся указатели на первый и последний элемент, и количество элементов
3	*p1,*p2	cr	Указатели на структуру toys
4	*l1, *l2	list	Указатели на структуру list

5	key	char	Введенный номер элемента

Переменные функции make_head()

№	Название	тип	Описание
1	*ph	lh	Указатель на голову структуры toys

Переменные функции make_head_list()

№	Название	тип	Описание
1	*ph	lh1	Указатель на голову структуры list

Переменные функции add_last ()

№	Название	тип	Описание
1	n	int	Счётчиковая переменная

Переменные функции create_node()

№	Название	тип	Описание
1	node	cr	Указатель на структуру toys

Переменные функции fill_struct()

№	Название	тип	Описание
1	j	int	Счётчиковая переменная
2	*p1	list	Указатель на структуру list
3	string[inf]	char	Массив для занесения новых имен

Переменные функции file_open()

№	Название	тип	Описание
1	j,i,n	int	Счётчиковые переменные
2	slen	int	Переменная для длин строк при считывании файла
3	*p1,*p2	cr	Указатели на структуру toys
4	**s2	char	Указатель на двумерный массив для занесения данных из файла в структуру
5	sep	char	Знак-разделитель
6	s1[maxlen]	char	Массив для считывания строк из файла

Переменные функции clear_arr ()

№	Название	тип	Описание
1	i	int	Номер элемента

Переменные функции simple_split()

№	название	тип	описание
1	i,j	int	Счётчиковые-переменная
3	k,m	char	Переменные для занесения в 2-мерный массив элементы строки
5	key	int	Проверка на выделение памяти
6	count	int	Для очистки массива после передачи его в main

Переменные функции make_list ()

№	название	тип	описание
1	*node	list	Указатель на структуру list
2	flag	int	Переменная для проверки на наличие

Переменные функции print_array ()

№	название	тип	описание
1	*p	cr	Указатель на структуру toys
2	i	int	Счетчиковая переменная

Переменные функции print_list ()

№	название	тип	описание
1	*p	list	Указатель на структуру list

Переменные функции choosing_action(), param_menu(), editing_menu(), search_menu(), help_menu()

№	название	тип	описание
1	a	char	Переменная для считывания номера
2	b	char	Переменная для присвоения введеного номера

Переменные функции deleting_element()

№	название	тип	описание
1	a	char	Переменная для считывания номера
2	b	char	Переменная для присвоения введенного номера

3	*p1,*p2	cr	Указатели на структуру toys
4	*l1, *l2	list	Указатели на структуру list
5	line[maxlen]	char	Массив для удаления элементов
6	num	int	Номер элемента

Переменные функции sorting_num (), sorting_str(), sorting_num_reverse(), sorting_str_reverse()

№	**название**	**тип**	**описание**
1	*cur, *root, *newroot, *node	cr	Указатели на структуру toys

Переменные функции editing_list()

№	**Название**	**тип**	**Описание**
1	*p1,*p2	cr	Указатели на структуру toys
2	*l1, *l2	list	Указатели на структуру list
3	key	char	Введенный номер элемента
4	string[inf]	char	Массив для ввода имени или же числа
5	j,i	int	Счётчиковые переменные
6	num	int	Переменная для номера элемента

Переменные функции make_csv()

№	**Название**	**тип**	**Описание**
1	*p	cr	Указатель на структуру toys
2	i	int	Счётчиковая переменная

3	*df	FILE	Проверка на открытие файла

Переменные функции search_str ()

№	Название	тип	Описание
1	*p1,*node	cr	Указатели на структуру toys
2	i,j	int	Счётчиковые переменные
3	string[inf]	char	Массив для ввода того что будет искаться
4	str[inf],pstr[inf]	char	Массивы для перевода в нижний регистр

Переменные функции search_int()

№	Название	тип	Описание
1	*p1,*node	cr	Указатели на структуру toys
2	i,j	int	Счётчиковые переменные
3	string[inf]	char	Массив для ввода того что будет искаться

Переменные функции search_list ()

№	Название	тип	Описание
1	key	char	Номер элемента
2	i	int	Счётчиковая переменная

Переменные функции number_elements_manufacturer()

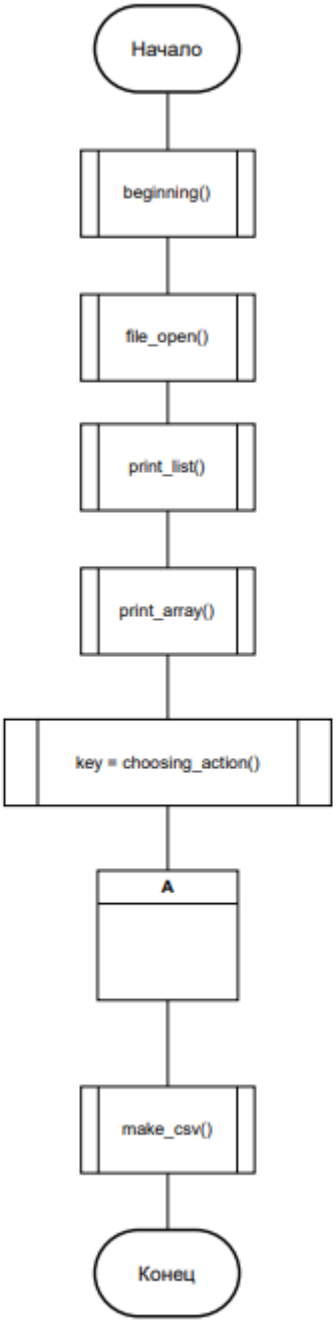
№	Название	тип	Описание

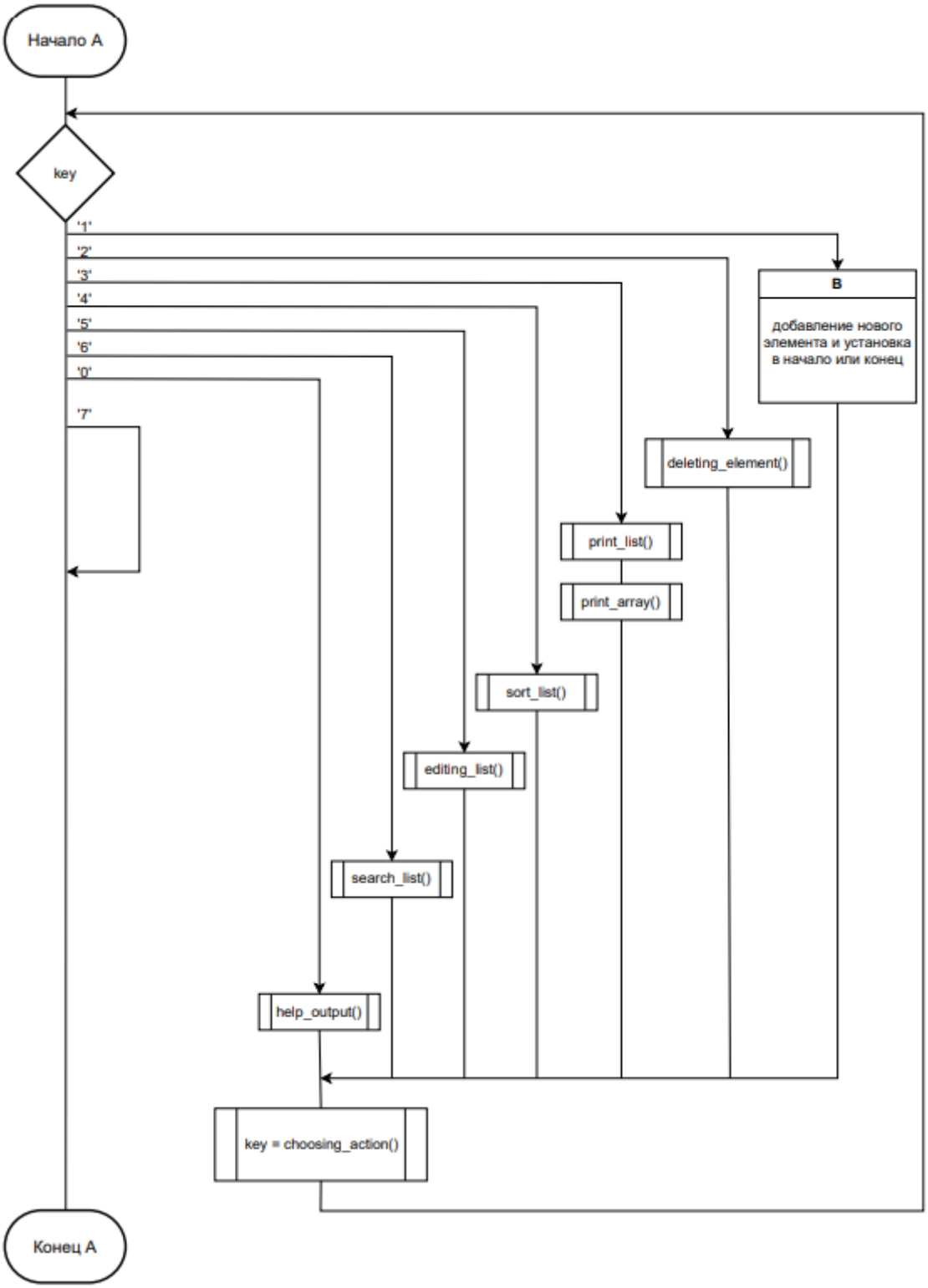
1	*p1,*node	cr	Указатели на структуру toys
2	count	int	Счётчиковая переменная

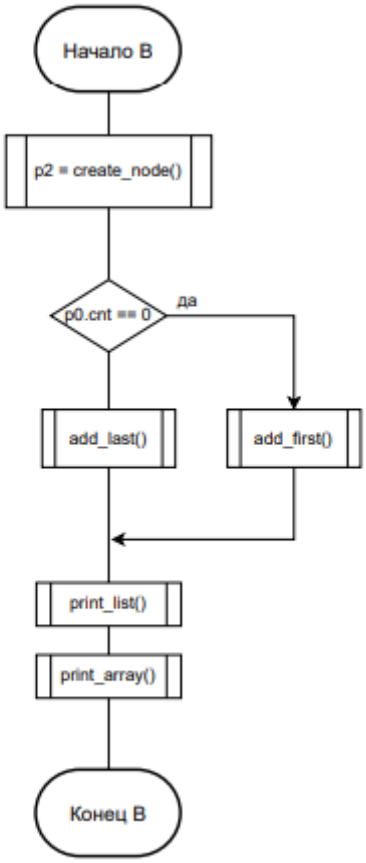
Переменные функции help_output()

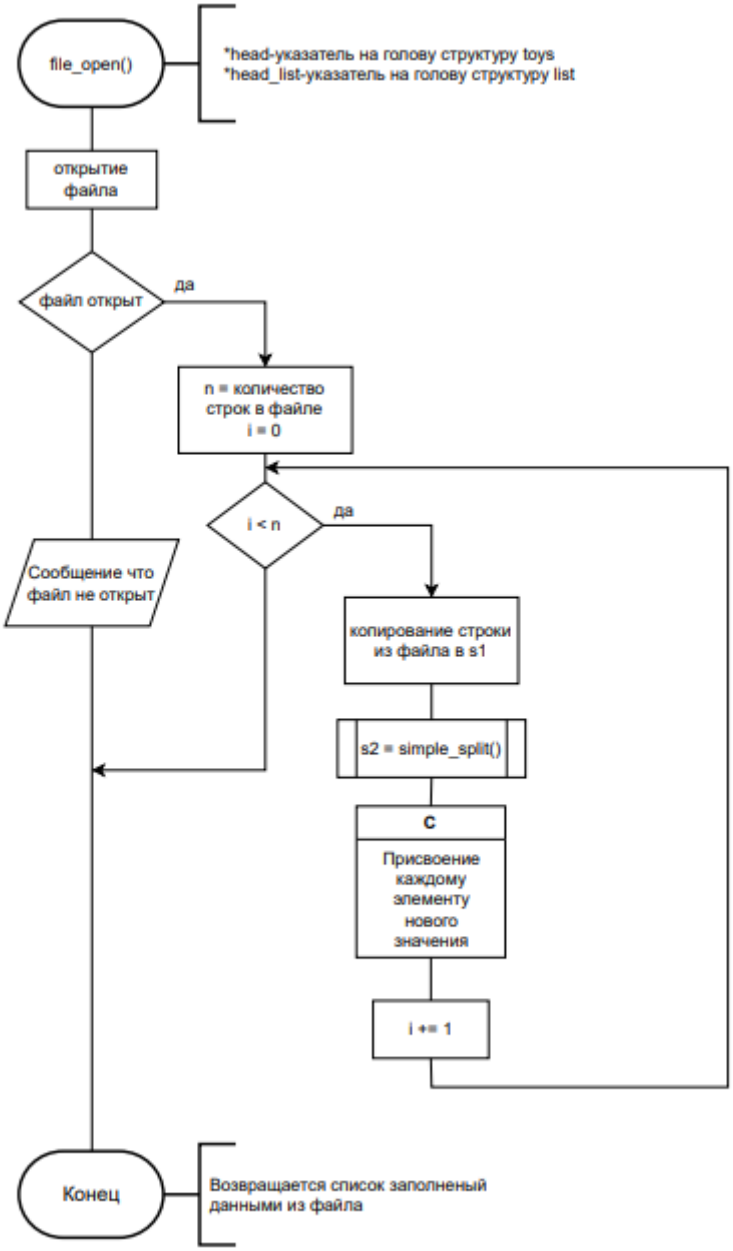
№	**Название**	**тип**	**Описание**
1	key	char	Номер элемента

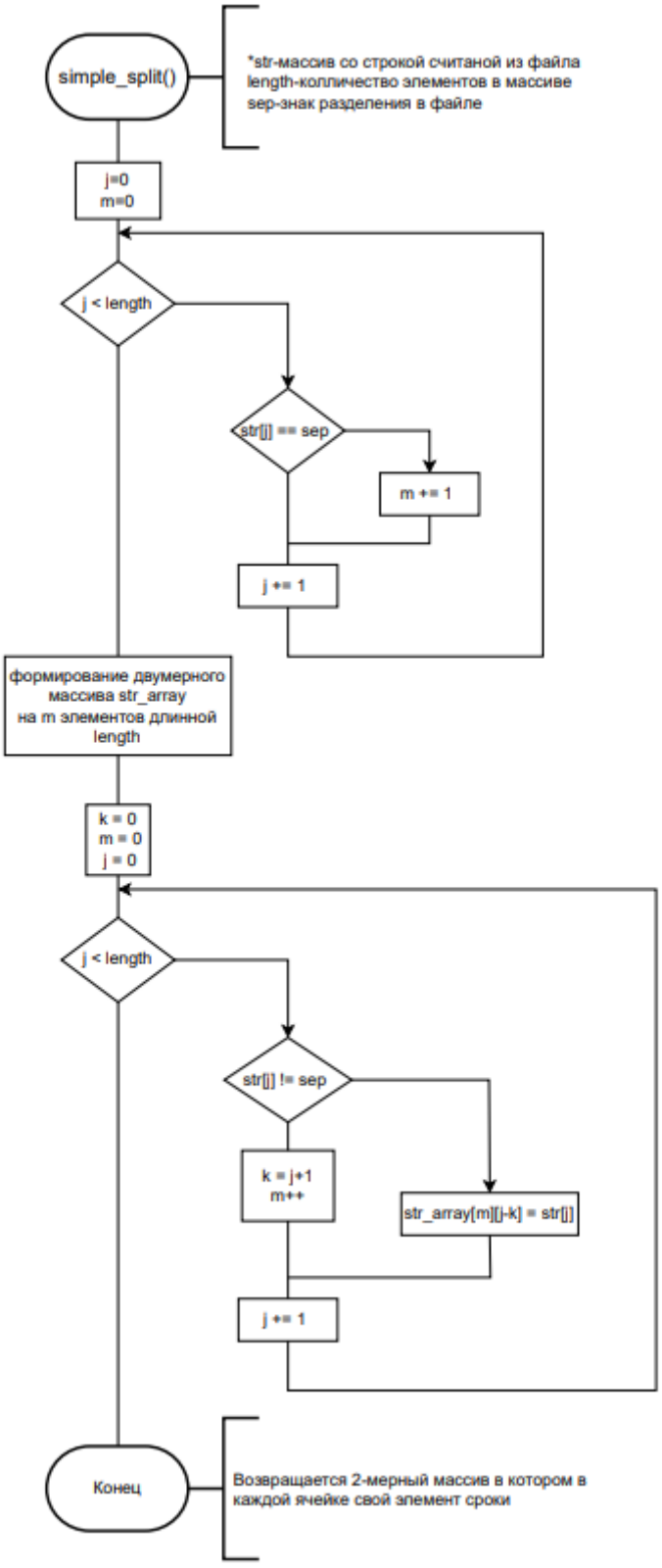
Схема алгоритма:

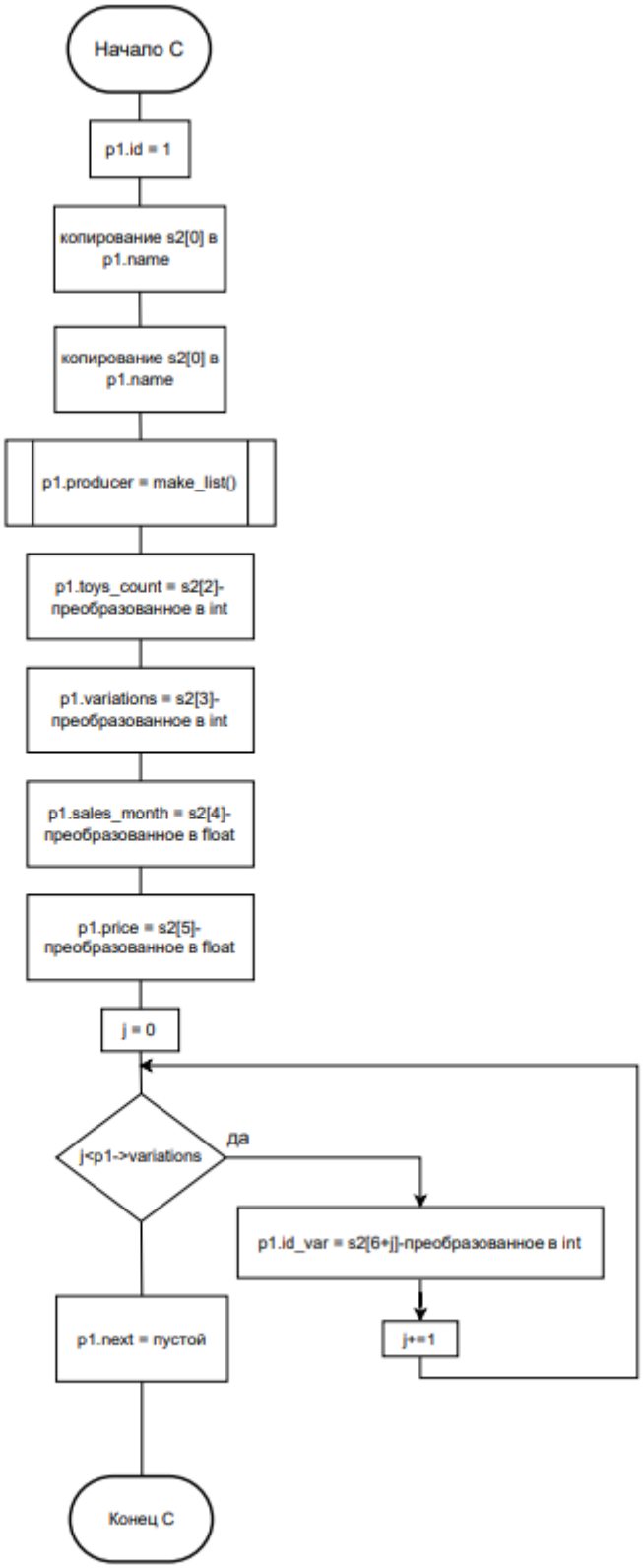


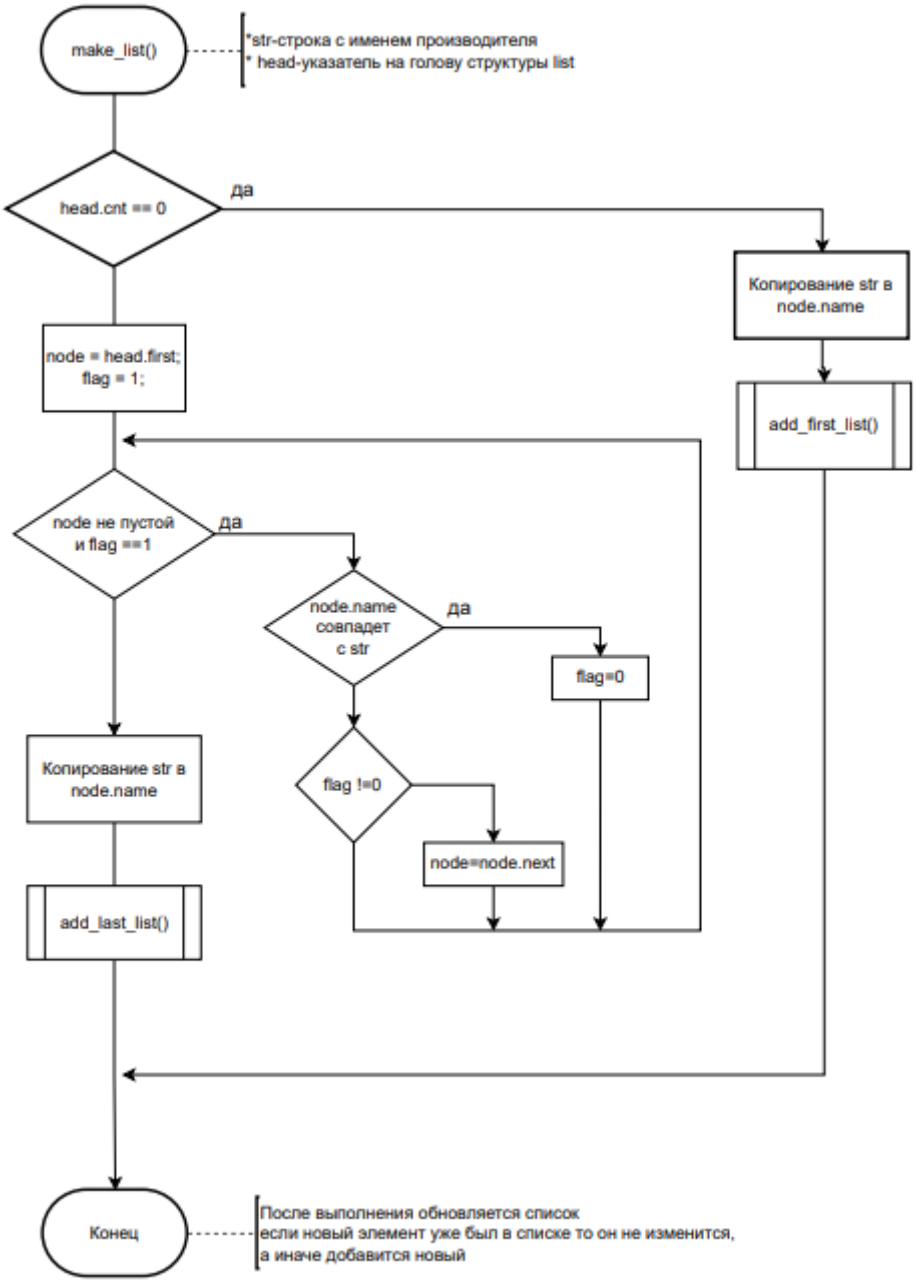

Начало A
key
'1'
'2'
'3'
'4'
'5'
'6'
'0'
'7'
B
добавление нового элемента и установка в начало или конец
deleting_element()
print_list()
print_array()
sort_list()
editing_list()
search_list()
help_output()
key = choosing_action()
Конец A

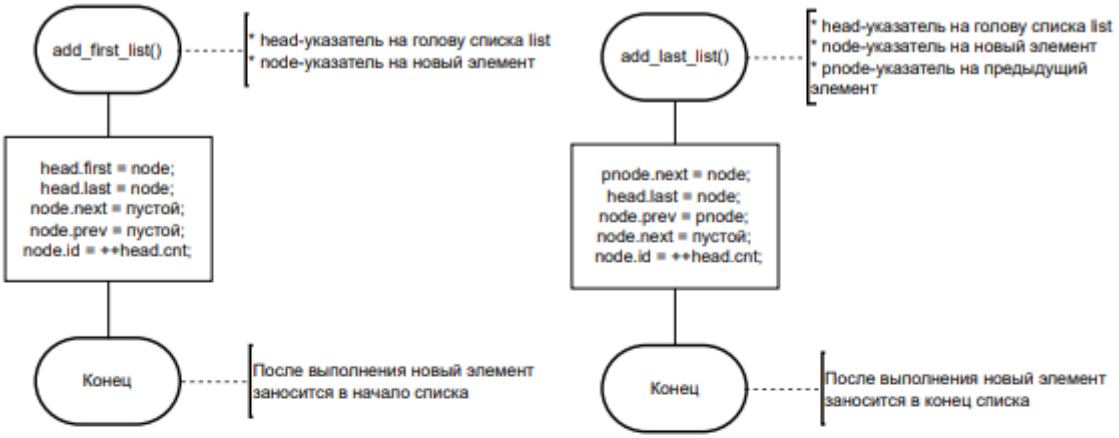

Начало B
p2 = create_node()
p0.cnt == 0
да
add_last()
add_first()
print_list()
print_array()
Конец B

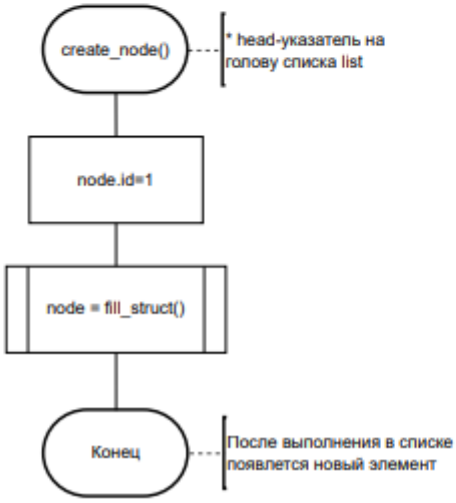

file_open()
*head-указатель на голову структуру toys
*head_list-указатель на голову структуру list
открытие файла
файл открыт
да
n = количество строк в файле
i = 0
i < n
да
Сообщение что файл не открыт
копирование строки из файла в s1
s2 = simple_split()
C
Присвоение каждому элементу нового значения
i += 1
Конец
Возвращается список заполненый данными из файла

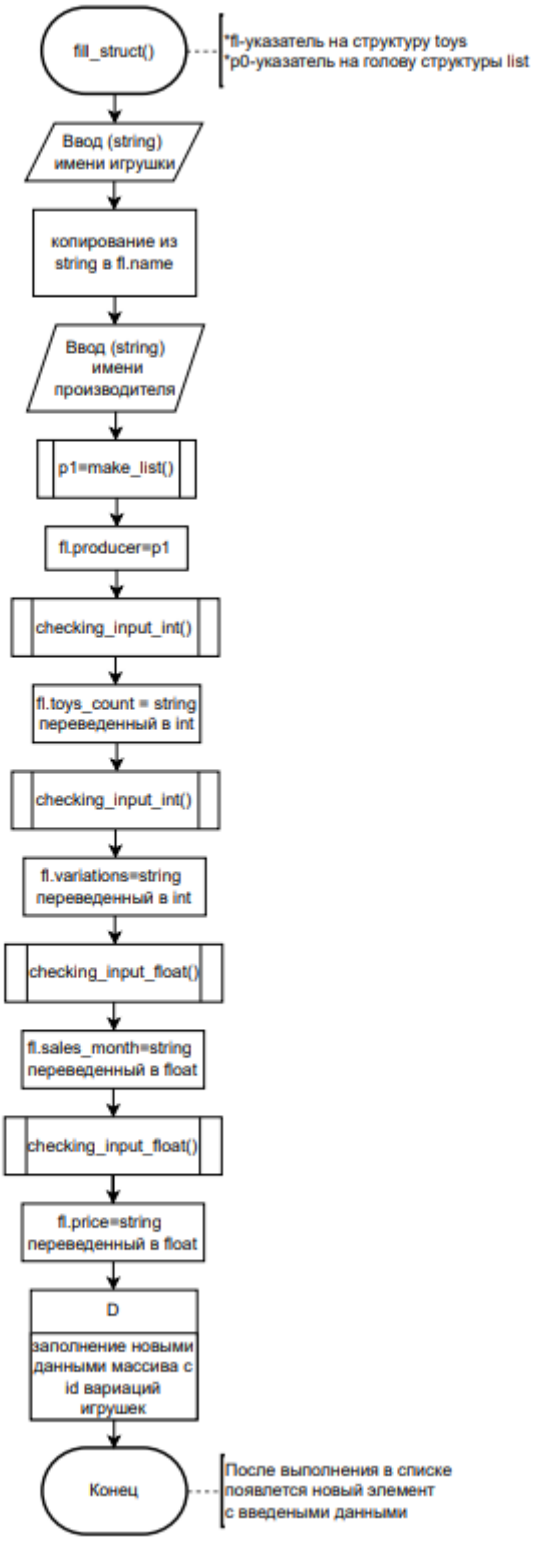

simple_split()
*str-массив со строкой считаной из файла
length-колличество элементов в массиве
sep-знак разделения в файле
j=0
m=0
j < length
str[j] == sep
m += 1
j += 1
формирование двумерного массива str_array на m элементов длинной length
k = 0
m = 0
j = 0
j < length
str[j] != sep
k = j+1
m++
str_array[m][j-k] = str[j]
j += 1
Конец
Возвращается 2-мерный массив в котором в каждой ячейке свой элемент сроки

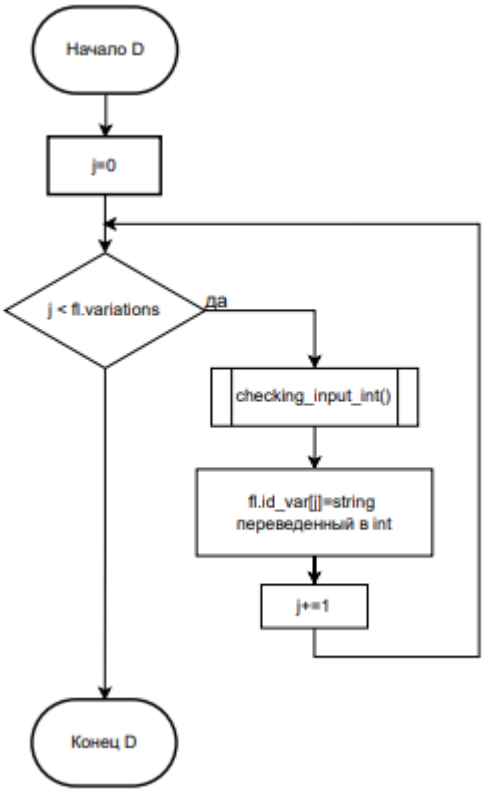

Начало С
p1.id = 1
копирование s2[0] в p1.name
копирование s2[0] в p1.name
p1.producer = make_list()
p1.toys_count = s2[2]-преобразованное в int
p1.variations = s2[3]-преобразованное в int
p1.sales_month = s2[4]-преобразованное в float
p1.price = s2[5]-преобразованное в float
j = 0
j<p1->variations
да
p1.id_var = s2[6+j]-преобразованное в int
j+=1
p1.next = пустой
Конец С

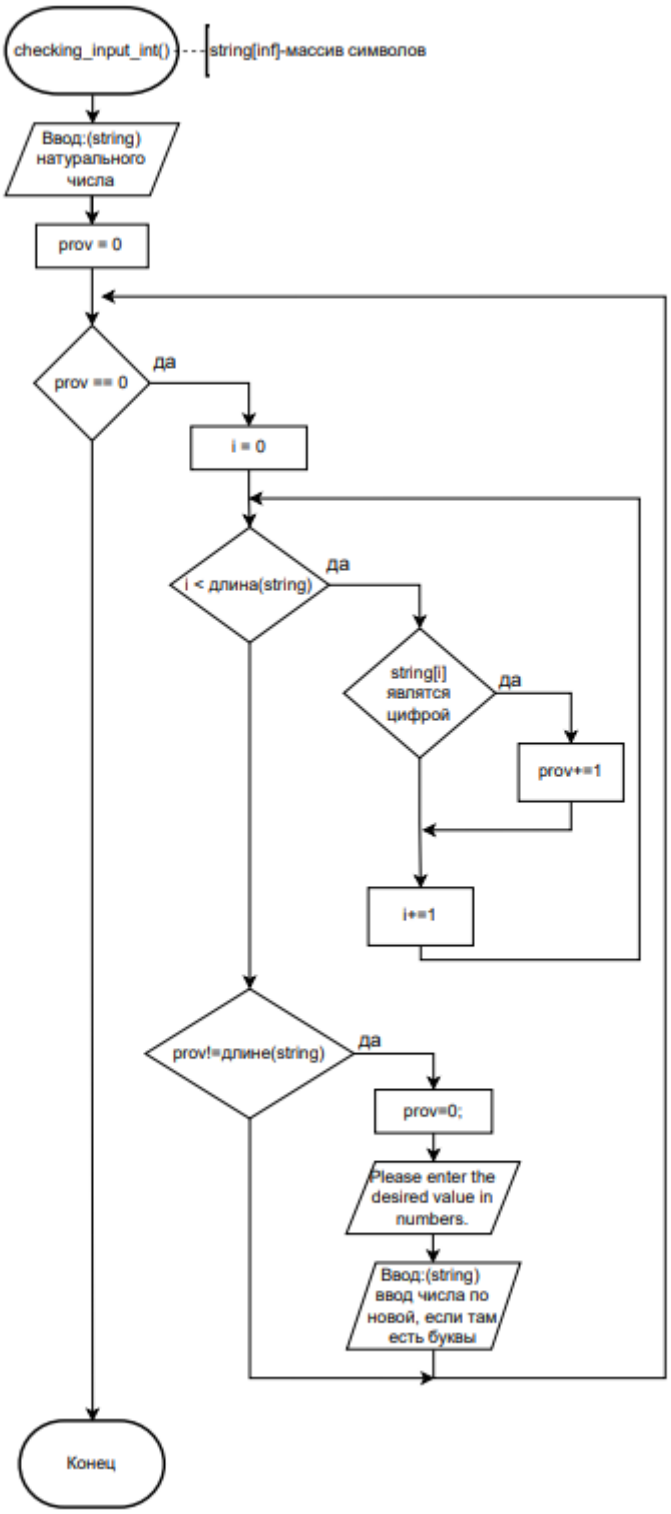

make_list()
*str-строка с именем производителя
* head-указатель на голову структуры list
head.cnt == 0
да
Копирование str в node.name
add_first_list()
node = head.first;
flag = 1;
node не пустой и flag ==1
да
node.name совпадет с str
да
flag=0
flag !=0
node=node.next
Копирование str в node.name
add_last_list()
Конец
После выполнения обновляется список
если новый элемент уже был в списке то он не изменится,
а иначе добавится новый

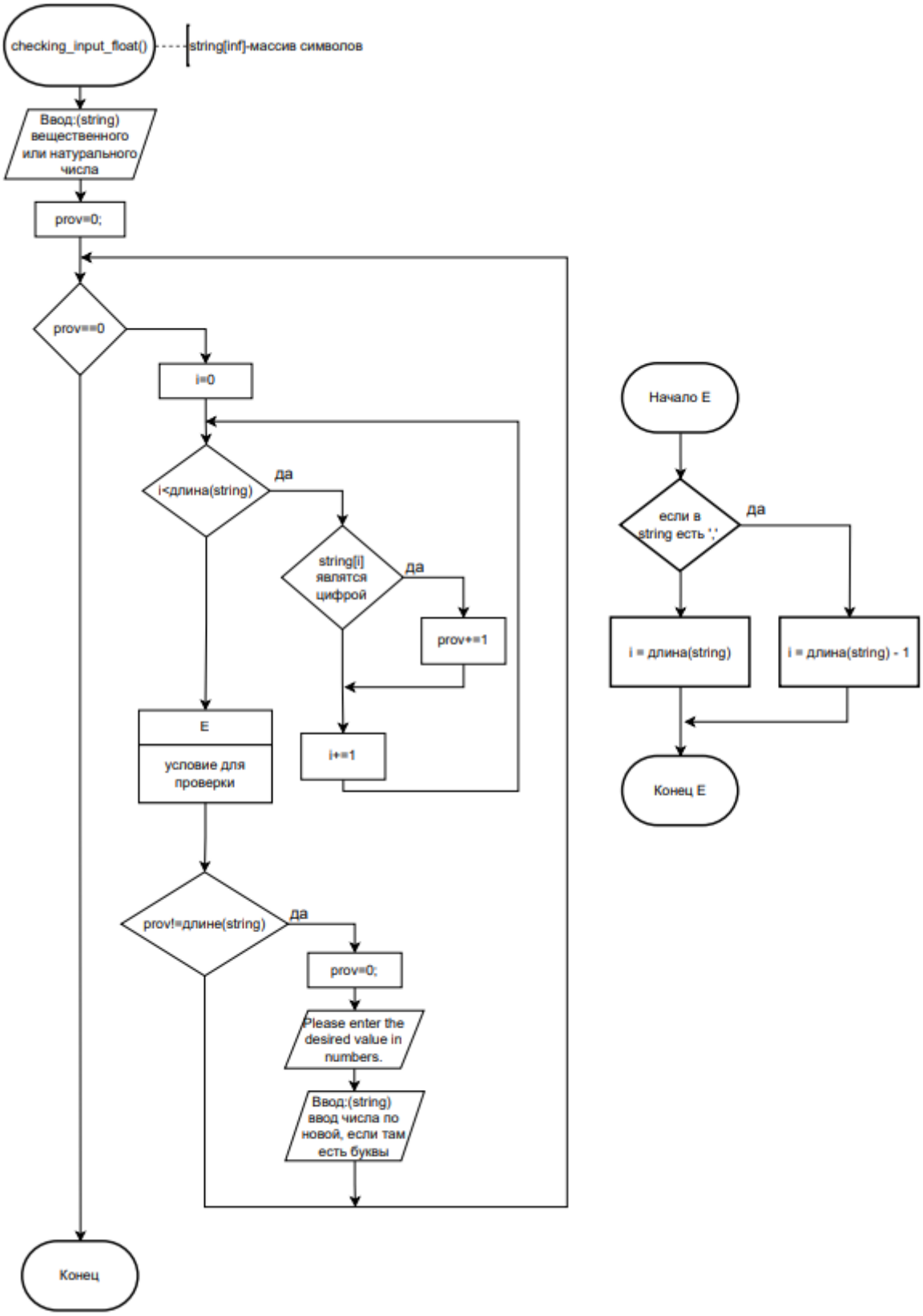

add_first_list()
* head-указатель на голову списка list
* node-указатель на новый элемент
head.first = node;
head.last = node;
node.next = пустой;
node.prev = пустой;
node.id = ++head.cnt;
Конец
После выполнения новый элемент заносится в начало списка
add_last_list()
* head-указатель на голову списка list
* node-указатель на новый элемент
* pnode-указатель на предыдущий элемент
pnode.next = node;
head.last = node;
node.prev = pnode;
node.next = пустой;
node.id = ++head.cnt;
Конец
После выполнения новый элемент заносится в конец списка

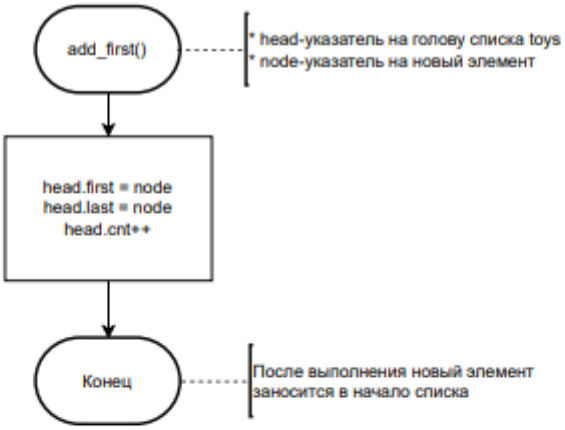

create_node()
* head-указатель на голову списка list
node.id=1
node = fill_struct()
Конец
После выполнения в списке появлется новый элемент

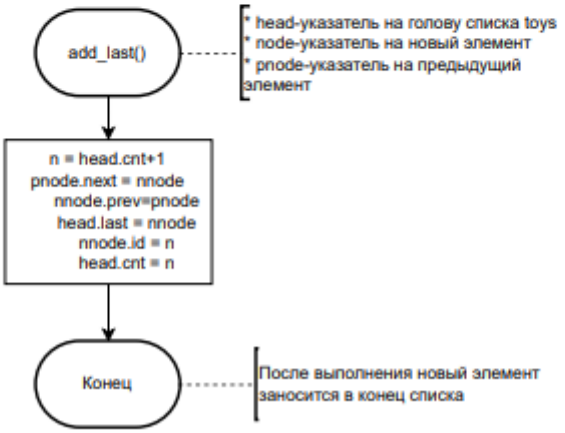

fill_struct()
*fl-указатель на структуру toys
*p0-указатель на голову структуры list
Ввод (string) имени игрушки
копирование из string в fl.name
Ввод (string) имени производителя
p1=make_list()
fl.producer=p1
checking_input_int()
fl.toys_count = string переведенный в int
checking_input_int()
fl.variations=string переведенный в int
checking_input_float()
fl.sales_month=string переведенный в float
checking_input_float()
fl.price=string переведенный в float
D
заполнение новыми данными массива с id вариаций игрушек
Конец
После выполнения в списке появлется новый элемент с введеными данными

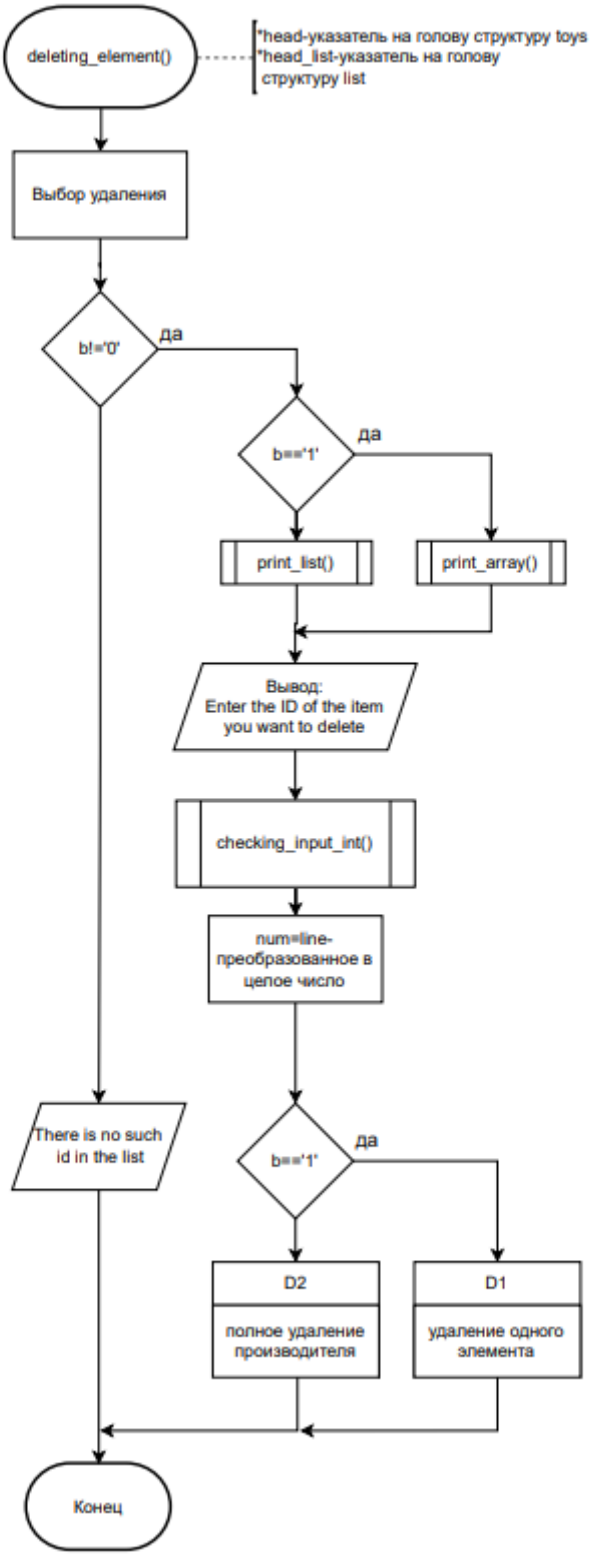

Начало D
j=0
j < fl.variations
да
checking_input_int()
fl.id_var[j]=string
переведенный в int
j+=1
Конец D

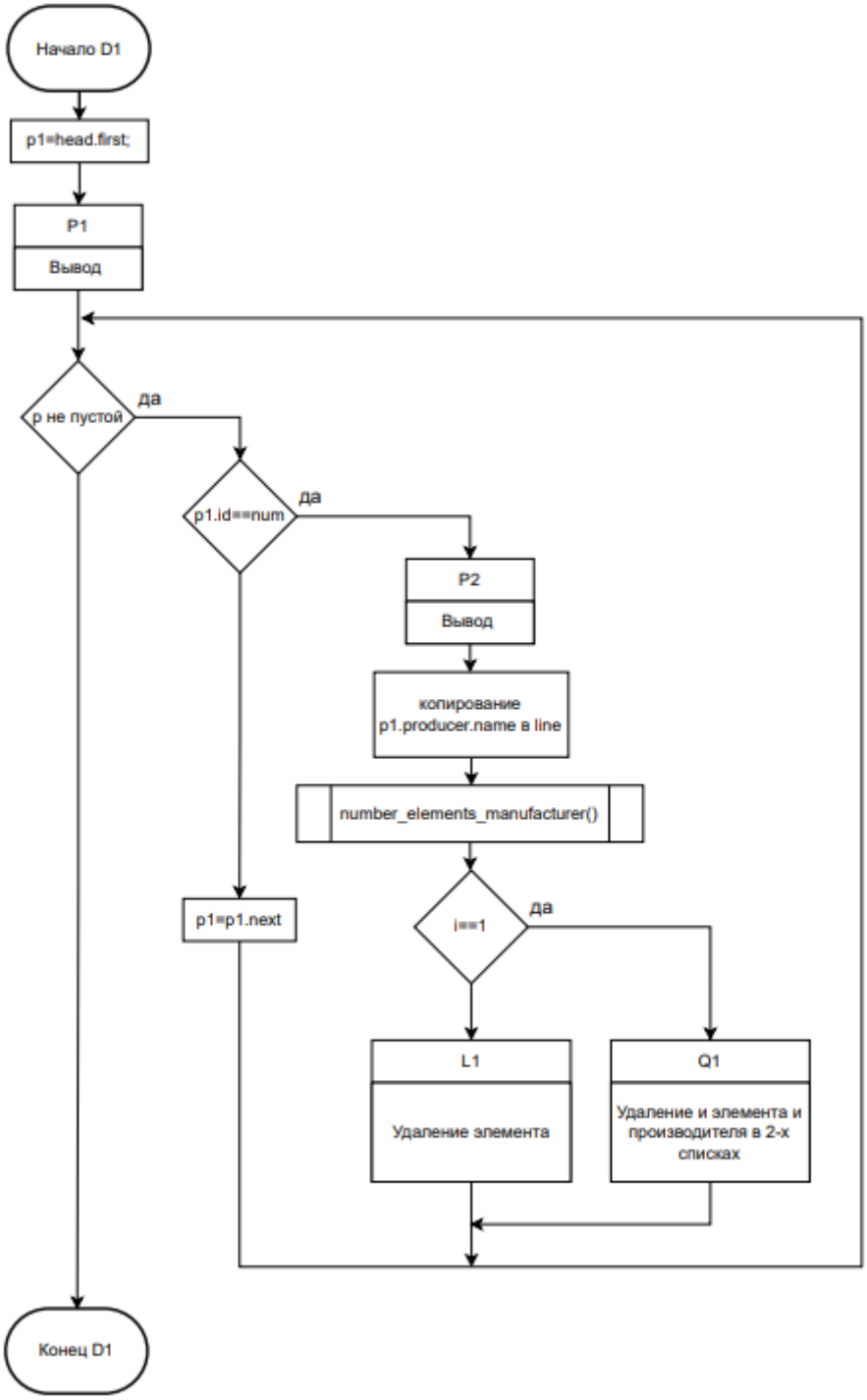

checking_input_int()
string[inf]-массив символов
Ввод:(string) натурального числа
prov = 0
prov == 0
да
i = 0
i < длина(string)
да
string[i] являтся цифрой
да
prov+=1
i+=1
prov!=длине(string)
да
prov=0;
Please enter the desired value in numbers.
Ввод:(string) ввод числа по новой, если там есть буквы
Конец

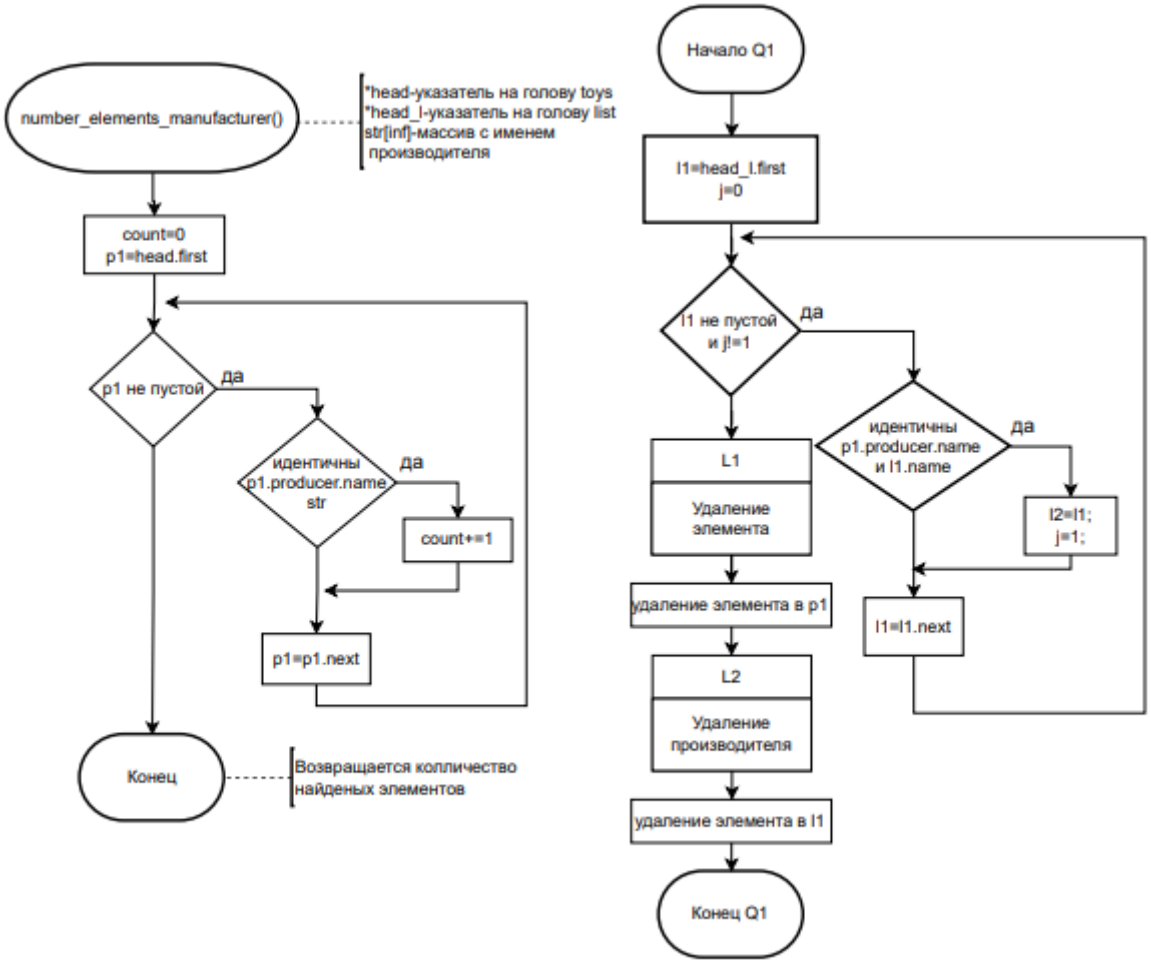

checking_input_float()
string[inf]-массив символов
Ввод:(string) вещественного или натурального числа
prov=0;
prov==0
i=0
i<длина(string)
да
string[i] являтся цифрой
да
prov+=1
i+=1
E
условие для проверки
prov!=длине(string)
да
prov=0;
Please enter the desired value in numbers.
Ввод:(string) ввод числа по новой, если там есть буквы
Конец
Начало E
если в string есть ','
да
i = длина(string)
i = длина(string) - 1
Конец E

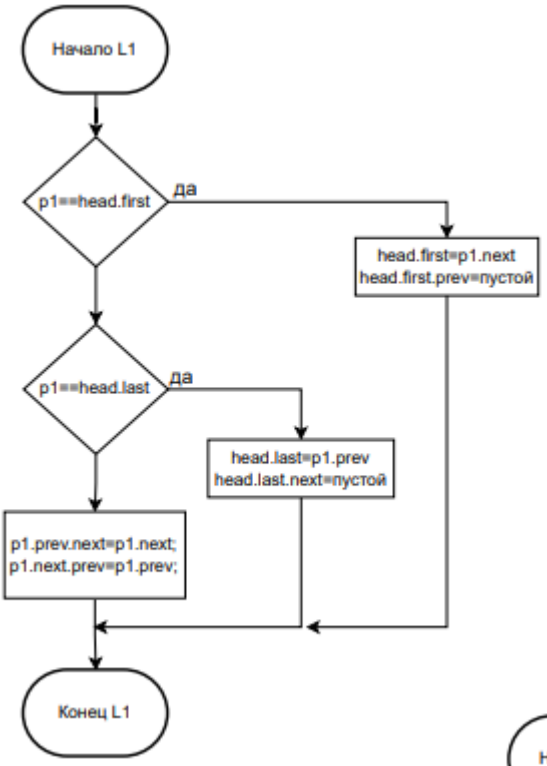

add_first()
* head-указатель на голову списка toys
* node-указатель на новый элемент
head.first = node
head.last = node
head.cnt++
Конец
После выполнения новый элемент заносится в начало списка

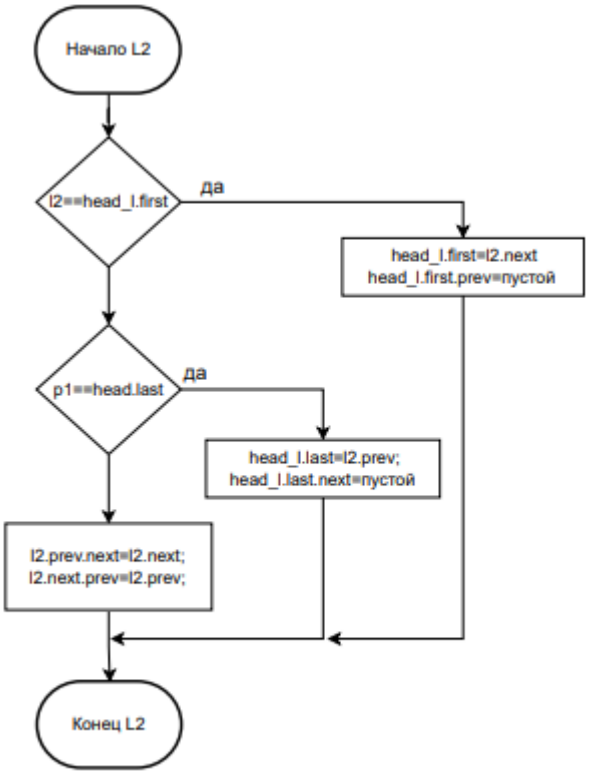

add_last()
* head-указатель на голову списка toys
* node-указатель на новый элемент
* pnode-указатель на предыдущий элемент
n = head.cnt+1
pnode.next = nnode
nnode.prev=pnode
head.last = nnode
nnode.id = n
head.cnt = n
Конец
После выполнения новый элемент заносится в конец списка

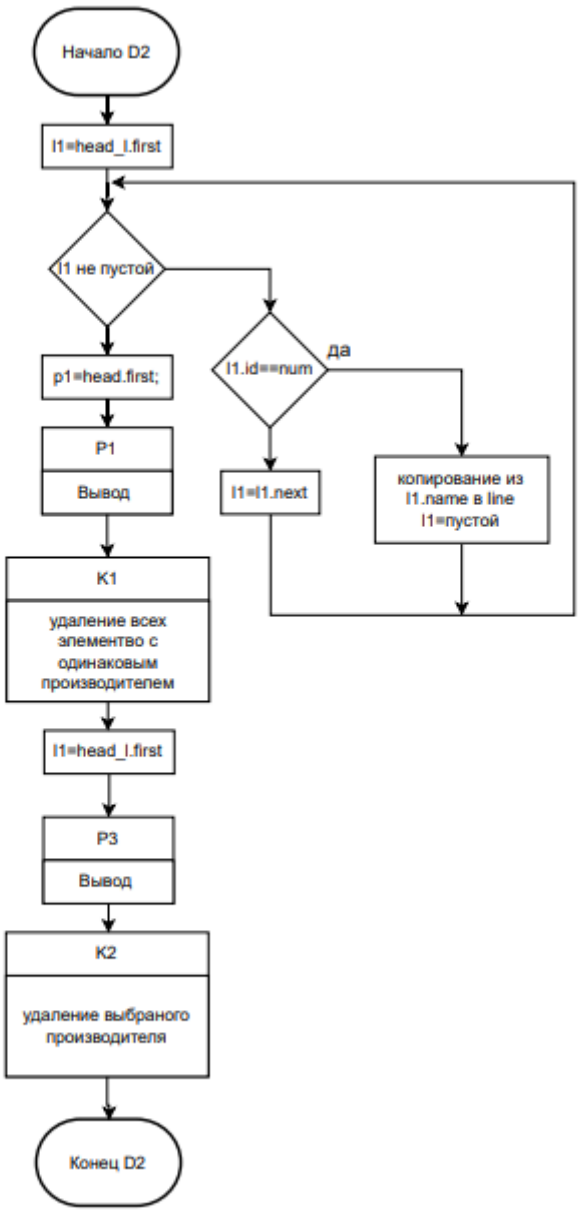

deleting_element()
*head-указатель на голову структуру toys
*head_list-указатель на голову структуру list
Выбор удаления
b!='0'
да
b=='1'
да
print_list()
print_array()
Вывод:
Enter the ID of the item you want to delete
checking_input_int()
num=line-преобразованное в целое число
There is no such id in the list
b=='1'
да
D2
полное удаление производителя
D1
удаление одного элемента
Конец

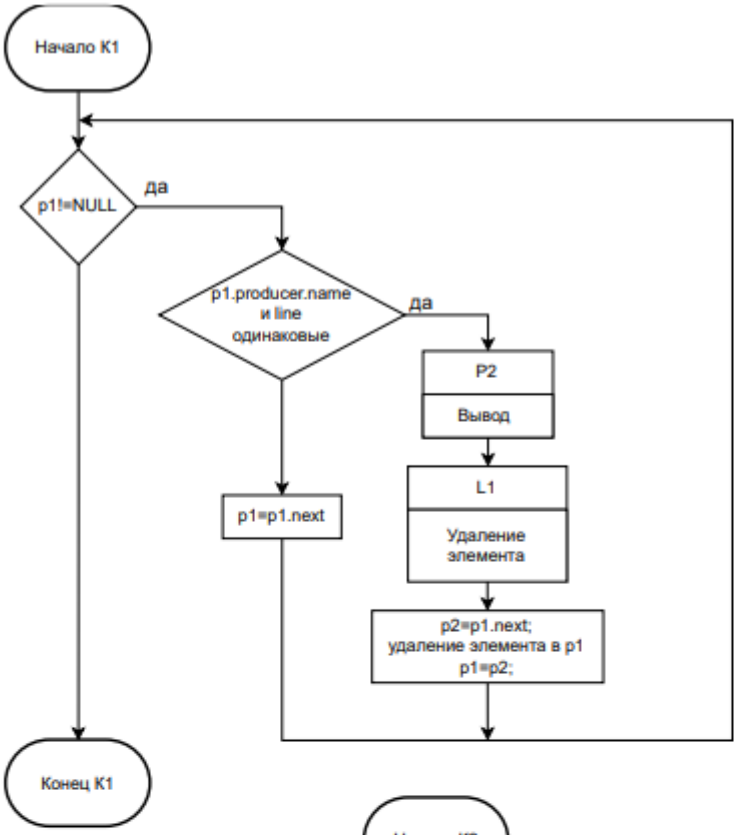

Начало D1
p1=head.first;
P1
Вывод
p не пустой
да
p1.id==num
да
P2
Вывод
копирование p1.producer.name в line
number_elements_manufacturer()
i==1
да
p1=p1.next
L1
Удаление элемента
Q1
Удаление и элемента и производителя в 2-х списках
Конец D1

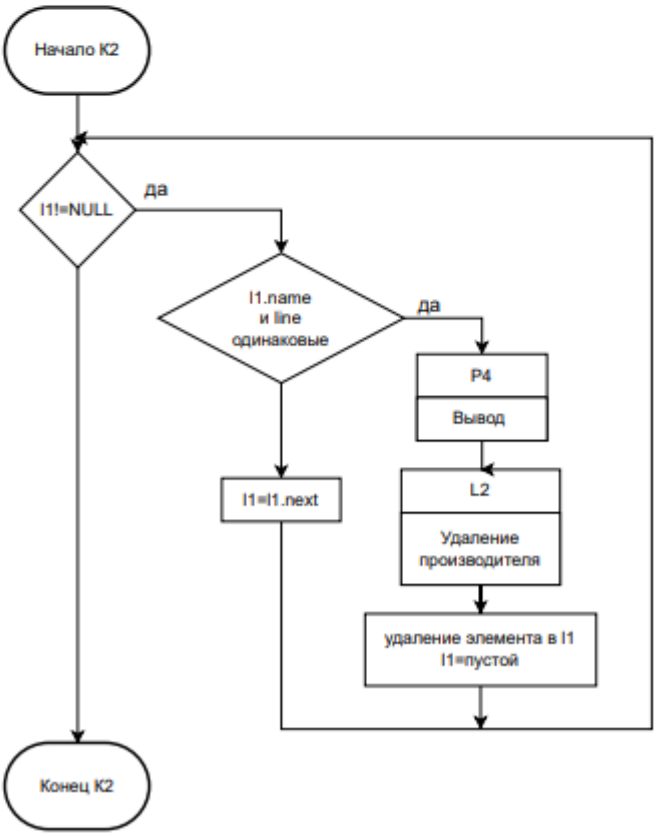

number_elements_manufacturer()
*head-указатель на голову toys
*head_l-указатель на голову list
str[inf]-массив с именем производителя
count=0
p1=head.first
p1 не пустой
да
идентичны p1.producer.name str
да
count+=1
p1=p1.next
Конец
Возвращается колличество найденых элементов
Начало Q1
l1=head_l.first
j=0
l1 не пустой и j!=1
да
идентичны p1.producer.name и l1.name
да
L1
Удаление элемента
l2=l1;
j=1;
l1=l1.next
удаление элемента в p1
L2
Удаление производителя
удаление элемента в l1
Конец Q1

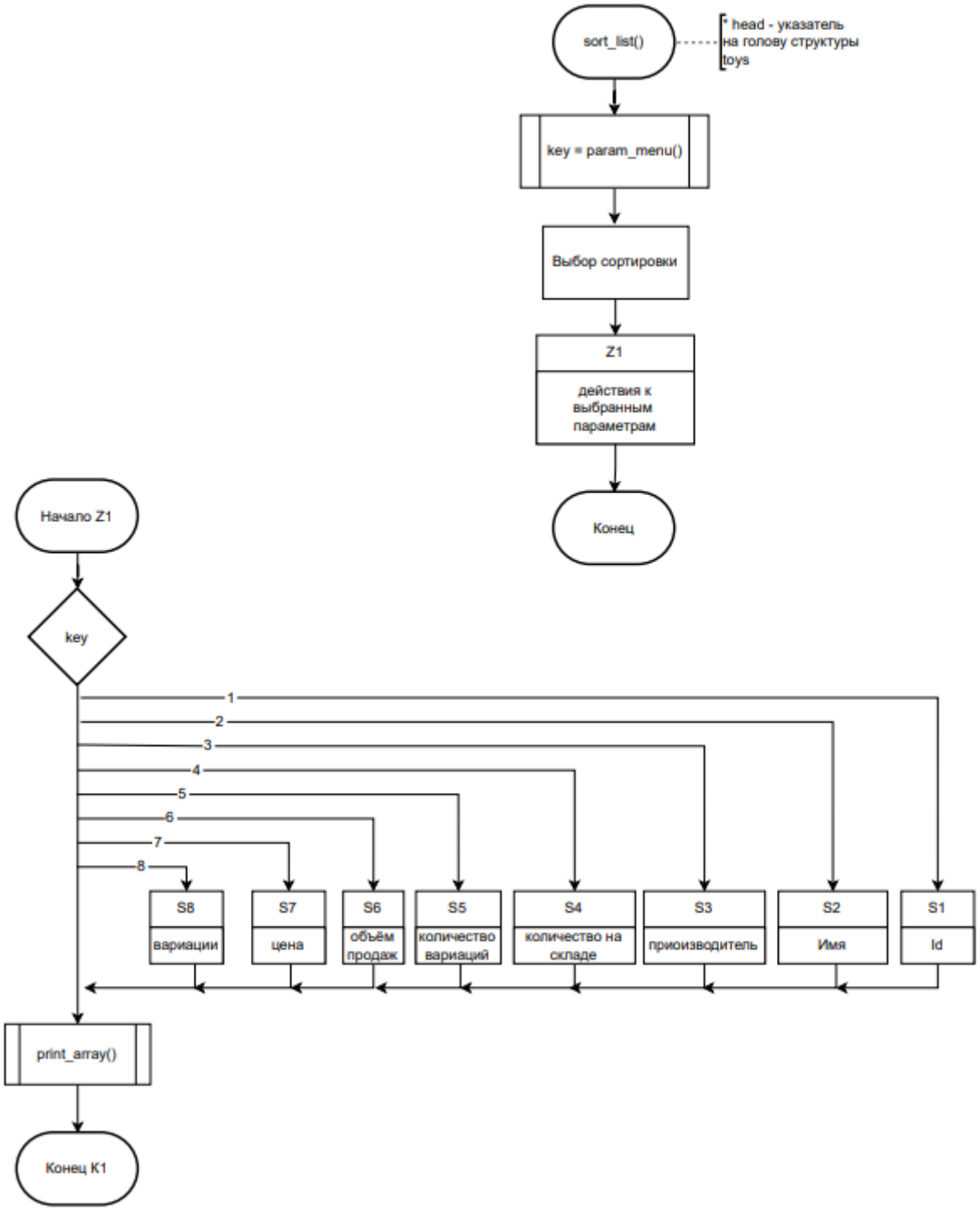

Начало L1
p1==head.first
да
head.first=p1.next
head.first.prev=пустой
p1==head.last
да
head.last=p1.prev
head.last.next=пустой
p1.prev.next=p1.next;
p1.next.prev=p1.prev;
Конец L1

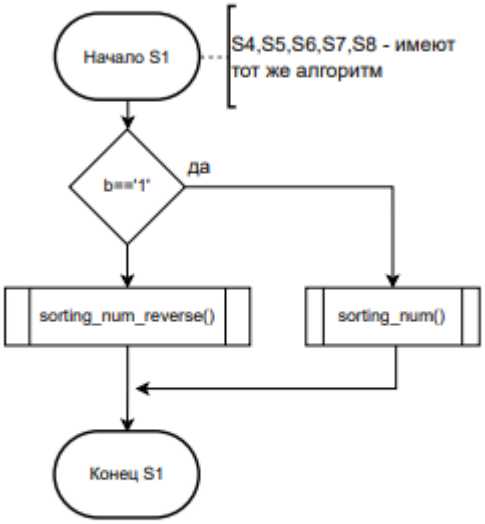

Начало L2
l2==head_l.first
да
head_l.first=l2.next
head_l.first.prev=пустой
p1==head.last
да
head_l.last=l2.prev;
head_l.last.next=пустой
l2.prev.next=l2.next;
l2.next.prev=l2.prev;
Конец L2

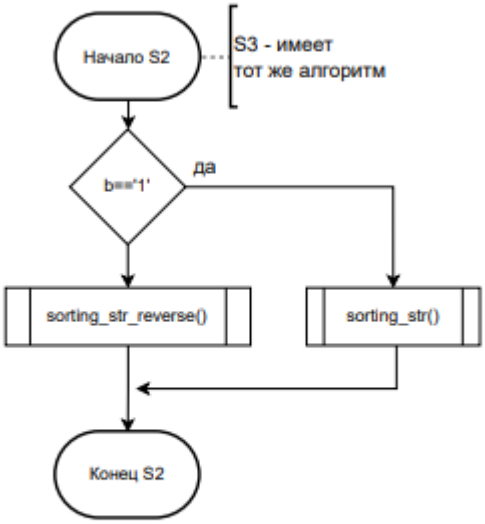

Начало D2
l1=head_l.first
l1 не пустой
p1=head.first;
P1
Вывод
K1
удаление всех элементво с одинаковым производителем
l1=head_l.first
P3
Вывод
K2
удаление выбраного производителя
Конец D2
l1.id==num
да
l1=l1.next
копирование из l1.name в line
l1=пустой

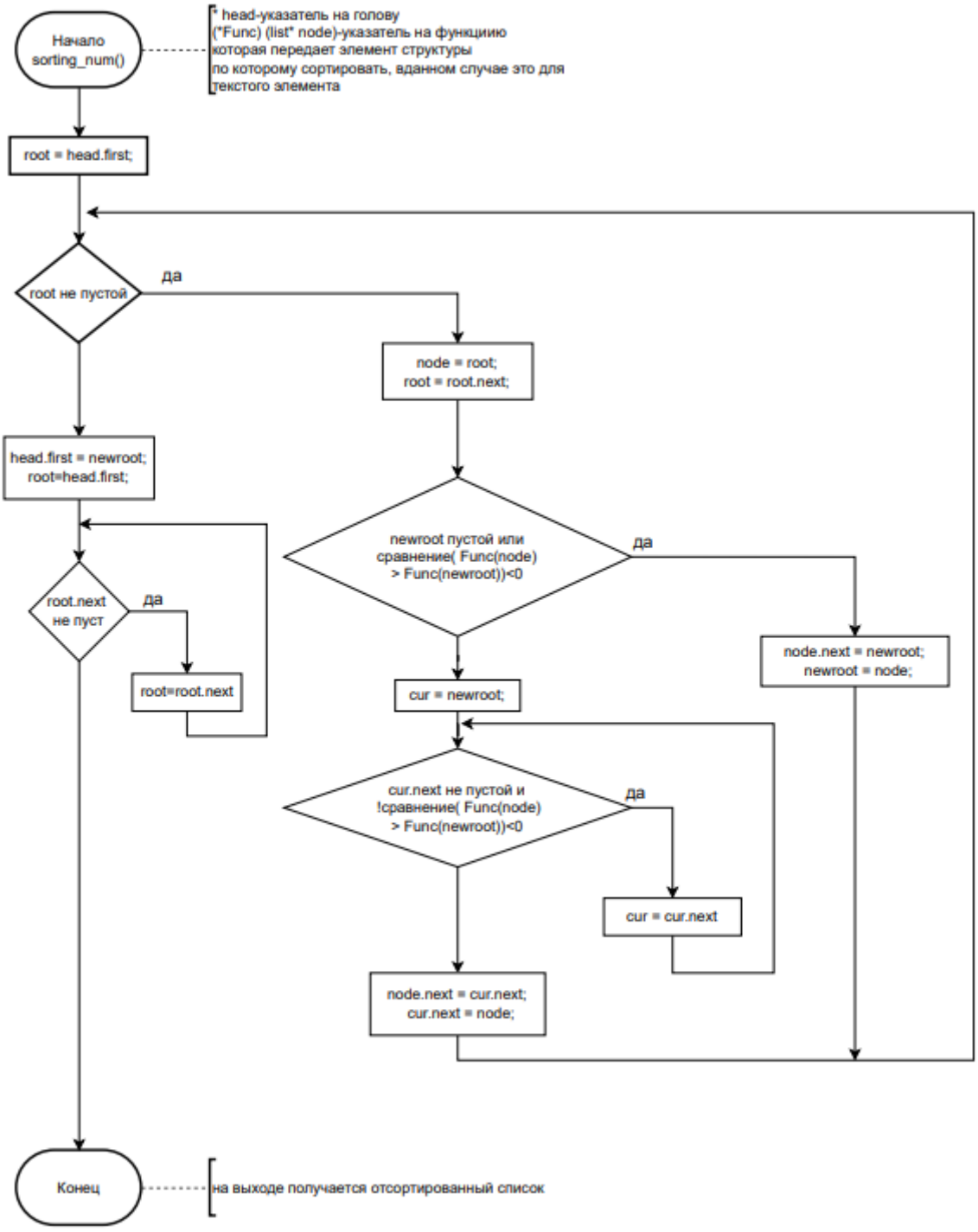

Начало К1
p1!=NULL
да
p1.producer.name
и line
одинаковые
да
P2
Вывод
L1
Удаление
элемента
p2=p1.next;
удаление элемента в p1
p1=p2;
p1=p1.next
Конец К1

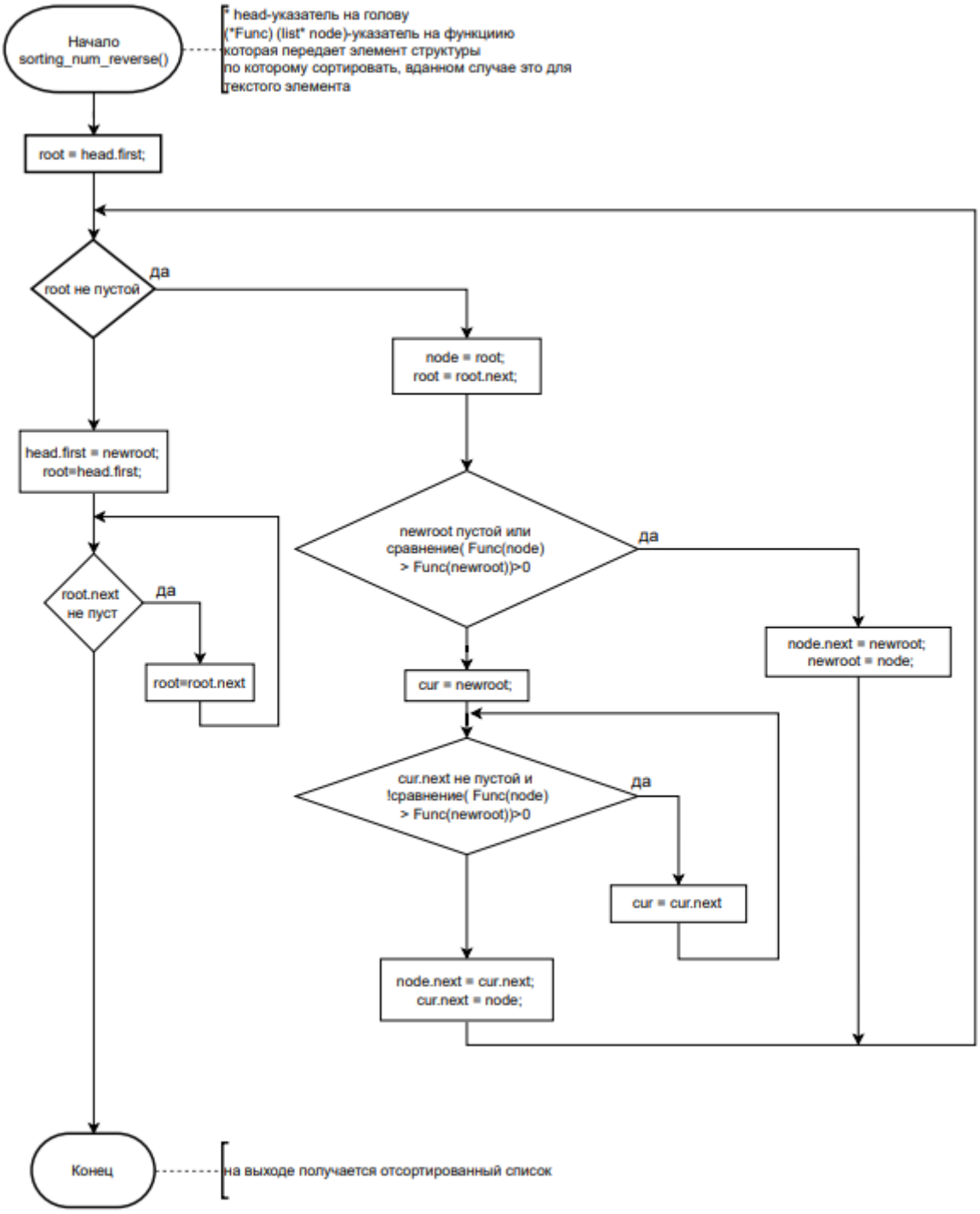

Начало К2
l1!=NULL
да
l1.name
и line
одинаковые
да
P4
Вывод
L2
Удаление
производителя
удаление элемента в l1
l1=пустой
l1=l1.next
Конец К2

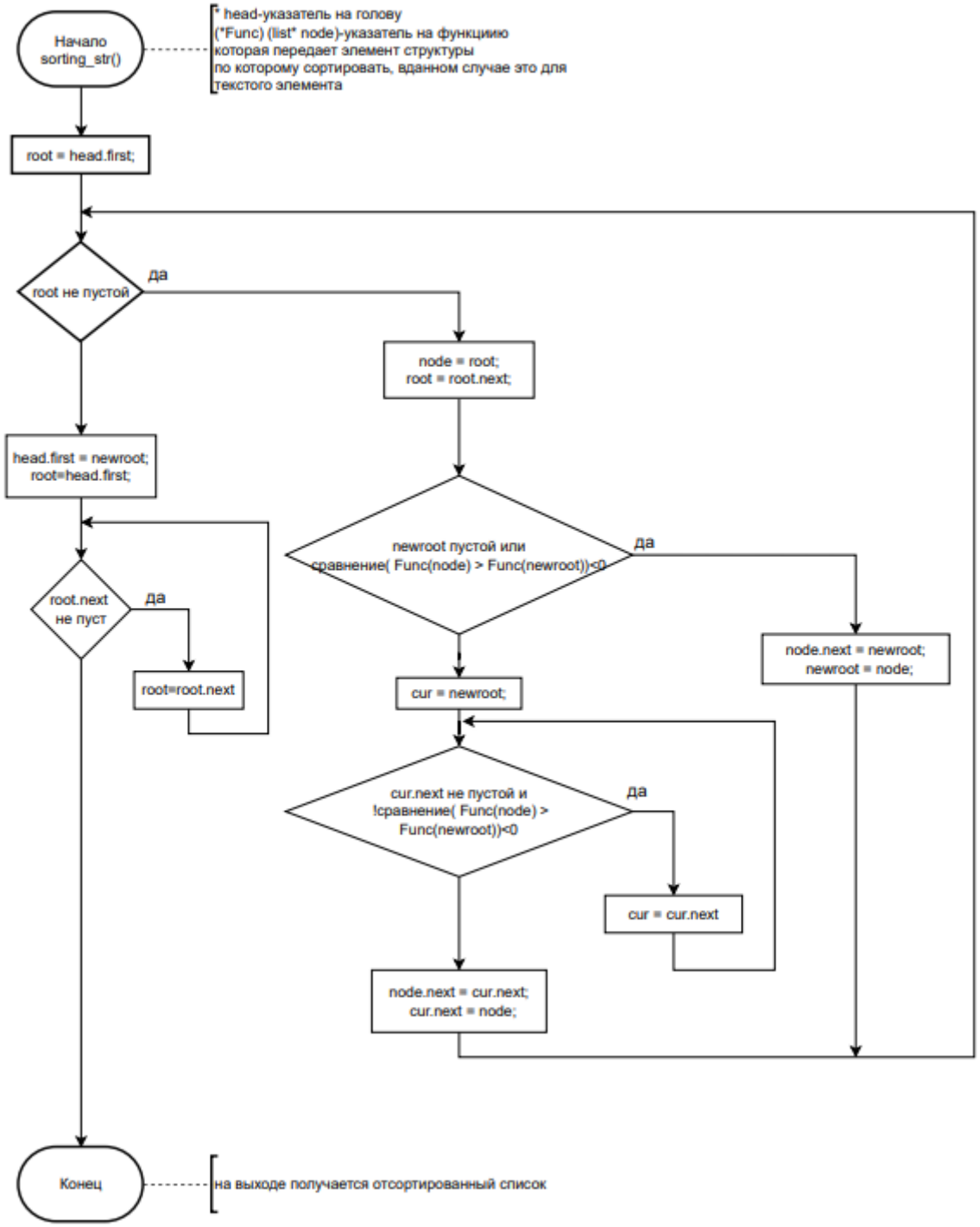

sort_list()
* head - указатель на голову структуры toys
key = param_menu()
Выбор сортировки
Z1
действия к выбранным параметрам
Конец
Начало Z1
key
1
2
3
4
5
6
7
8
S8
вариации
S7
цена
S6
объём продаж
S5
количество вариаций
S4
количество на складе
S3
приоизводитель
S2
Имя
S1
Id
print_array()
Конец К1

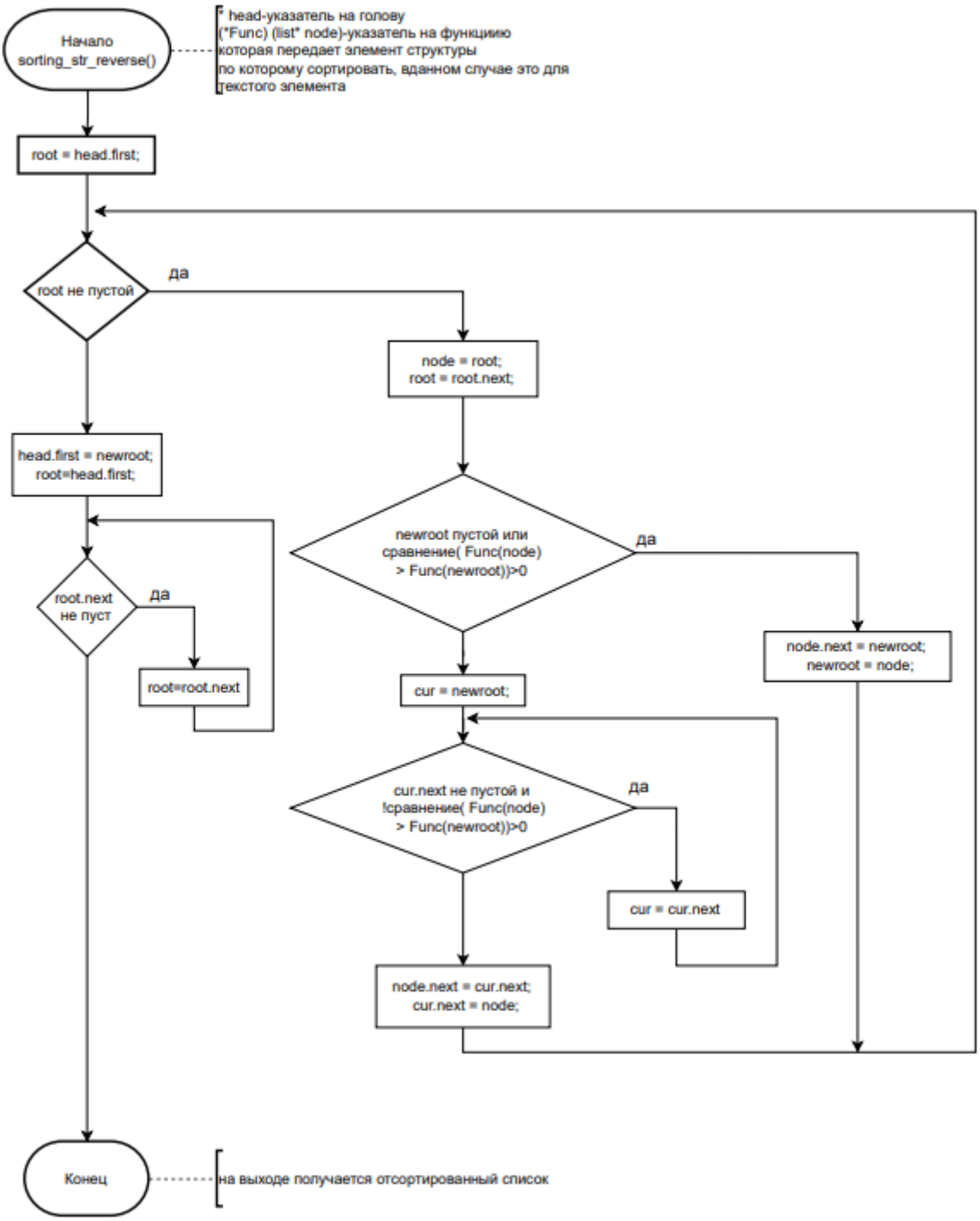

Начало S1
S4,S5,S6,S7,S8 - имеют
тот же алгоритм
b=='1'
да
sorting_num_reverse()
sorting_num()
Конец S1

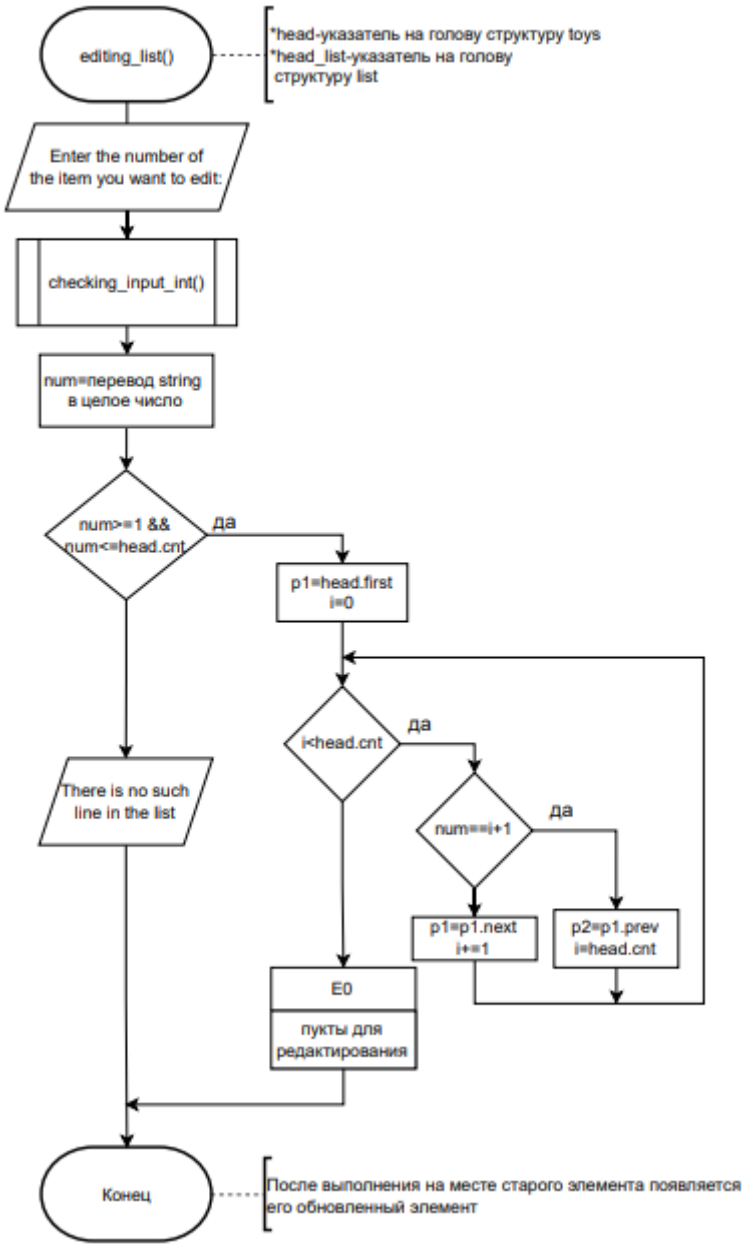

Начало S2
S3 - имеет
тот же алгоритм
b=='1'
да
sorting_str_reverse()
sorting_str()
Конец S2


Начало
sorting_num()
* head-указатель на голову
(*Func) (list* node)-указатель на функциию
которая передает элемент структуры
по которому сортировать, вданном случае это для
текстого элемента
root = head.first;
root не пустой
да
node = root;
root = root.next;
head.first = newroot;
root=head.first;
newroot пустой или
сравнение(Func(node)
> Func(newroot))<0
да
root.next
не пуст
да
root=root.next
node.next = newroot;
newroot = node;
cur = newroot;
cur.next не пустой и
!сравнение(Func(node)
> Func(newroot))<0
да
cur = cur.next
node.next = cur.next;
cur.next = node;
Конец
на выходе получается отсортированный список

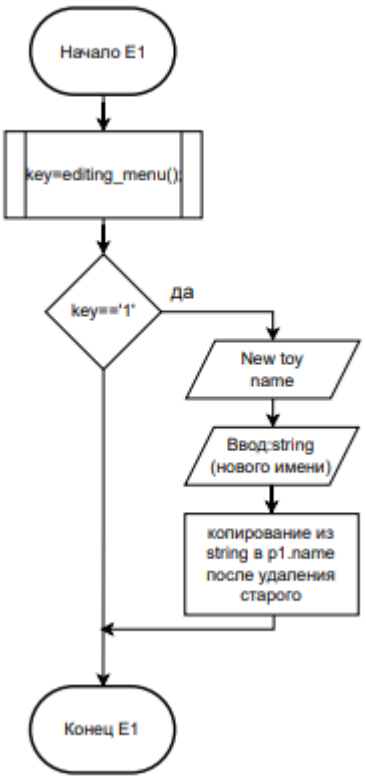

Начало
sorting_num_reverse()
* head-указатель на голову
(*Func) (list* node)-указатель на функциию
которая передает элемент структуры
по которому сортировать, вданном случае это для
текстого элемента
root = head.first;
root не пустой
да
node = root;
root = root.next;
head.first = newroot;
root=head.first;
newroot пустой или
сравнение(Func(node)
> Func(newroot))>0
да
root.next
не пуст
да
root=root.next
node.next = newroot;
newroot = node;
cur = newroot;
cur.next не пустой и
!сравнение(Func(node)
> Func(newroot))>0
да
cur = cur.next
node.next = cur.next;
cur.next = node;
Конец
на выходе получается отсортированный список

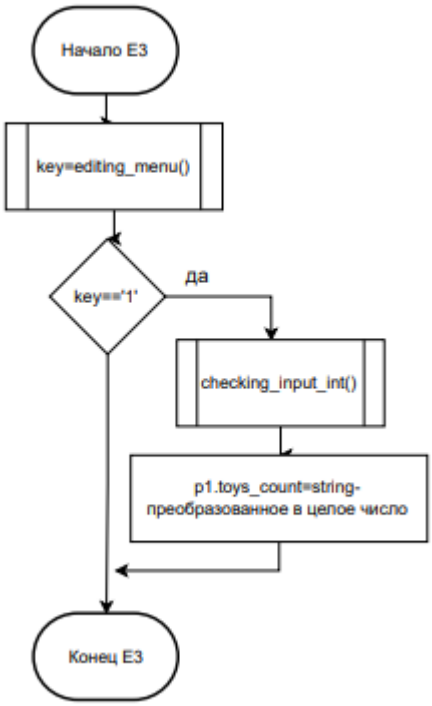

Начало
sorting_str()
* head-указатель на голову
(*Func) (list* node)-указатель на функциию
которая передает элемент структуры
по которому сортировать, вданном случае это для
текстого элемента
root = head.first;
root не пустой
да
node = root;
root = root.next;
head.first = newroot;
root=head.first;
newroot пустой или
сравнение(Func(node) > Func(newroot))<0
да
root.next
не пуст
да
root=root.next
node.next = newroot;
newroot = node;
cur = newroot;
cur.next не пустой и
!сравнение(Func(node) >
Func(newroot))<0
да
cur = cur.next
node.next = cur.next;
cur.next = node;
Конец
на выходе получается отсортированный список

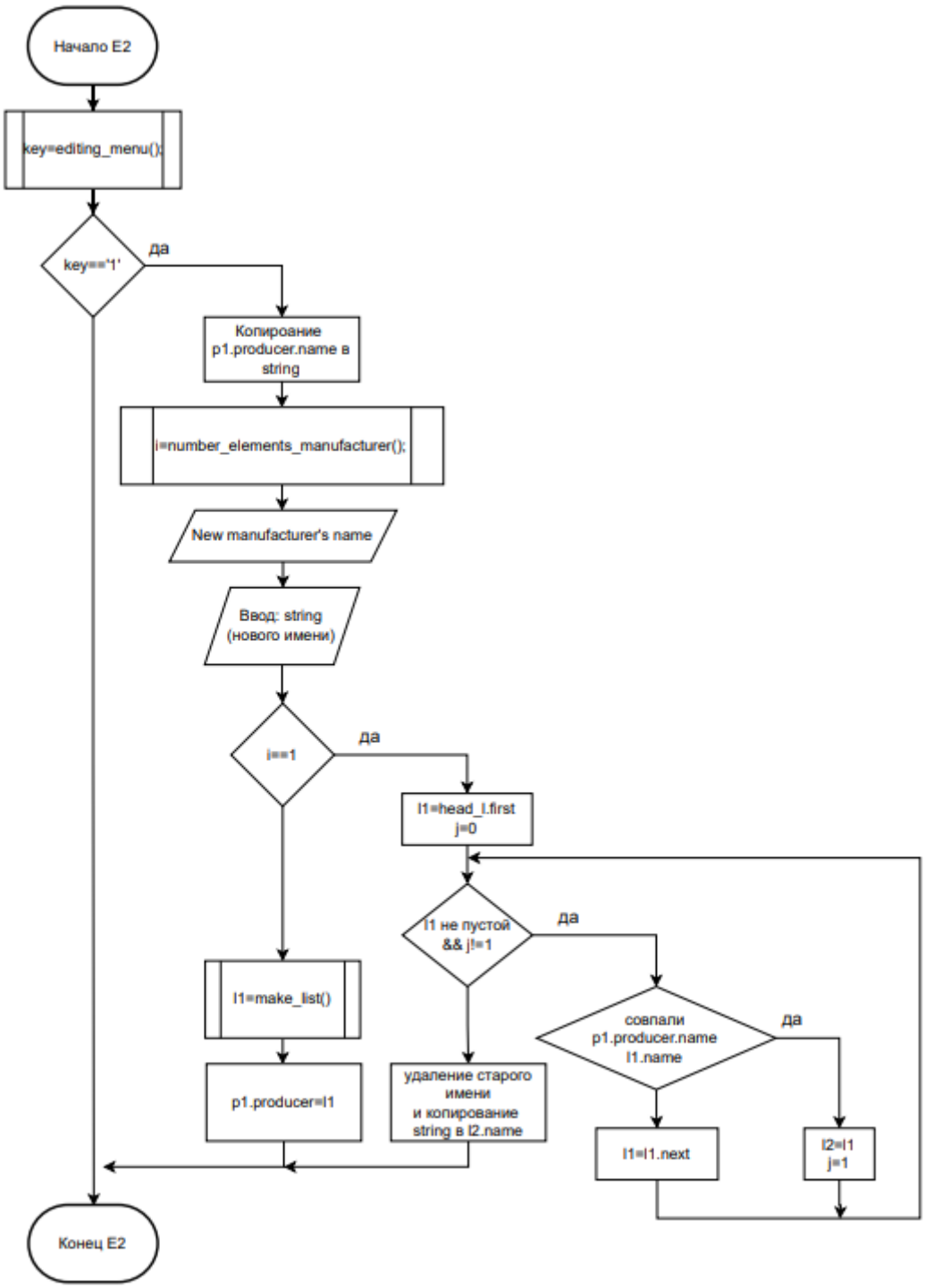

Начало
sorting_str_reverse()
* head-указатель на голову
(*Func) (list* node)-указатель на функциию
которая передает элемент структуры
по которому сортировать, вданном случае это для
текстого элемента
root = head.first;
root не пустой
да
node = root;
root = root.next;
head.first = newroot;
root=head.first;
newroot пустой или
сравнение(Func(node)
> Func(newroot))>0
да
root.next
не пуст
да
root=root.next
node.next = newroot;
newroot = node;
cur = newroot;
cur.next не пустой и
!сравнение(Func(node)
> Func(newroot))>0
да
cur = cur.next
node.next = cur.next;
cur.next = node;
Конец
на выходе получается отсортированный список

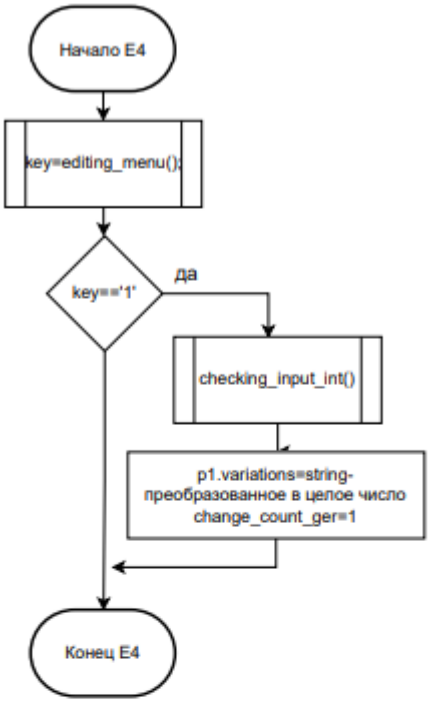

editing_list()
*head-указатель на голову структуру toys
*head_list-указатель на голову структуру list
Enter the number of the item you want to edit:
checking_input_int()
num=перевод string в целое число
num>=1 && num<=head.cnt
да
p1=head.first
i=0
i<head.cnt
да
num==i+1
да
p1=p1.next
i+=1
p2=p1.prev
i=head.cnt
There is no such line in the list
E0
пукты для редактирования
Конец
После выполнения на месте старого элемента появляется его обновленный элемент

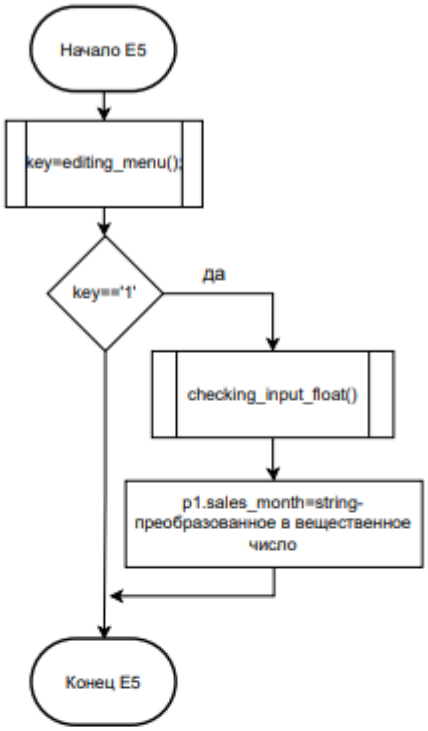

Начало E0
E1
редактирование имени
E2
редактирование производителя
E3
редактирование количества игрушек
E4
редактирование количества вариаций
E5
редактирование объёма продаж
E6
редактирование цены
E7
редактирование id каждой вариации
Конец E0

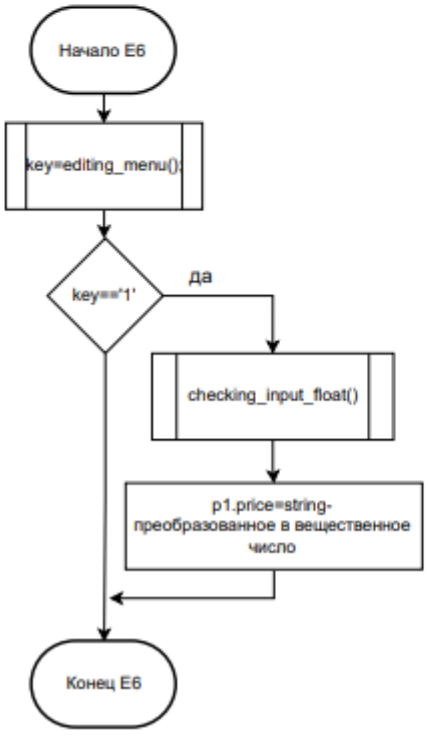

Начало E1
key=editing_menu()
key=='1'
да
New toy name
Ввод:string (нового имени)
копирование из string в p1.name после удаления старого
Конец E1

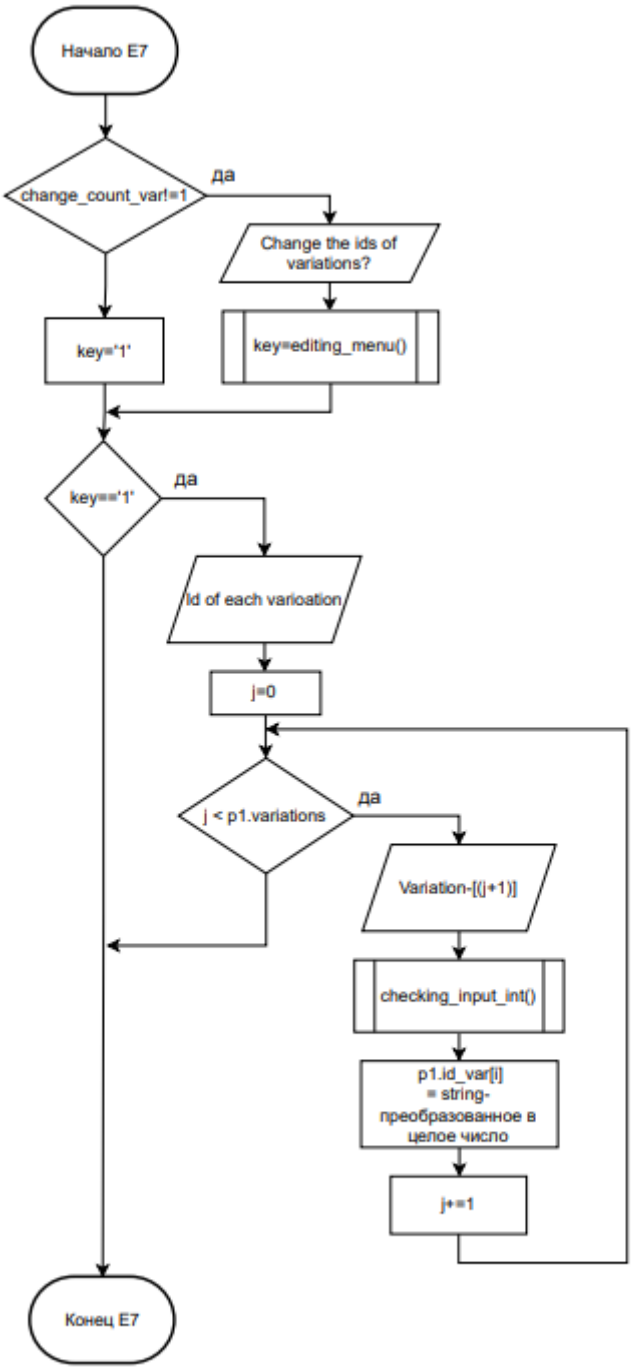

Начало E3
key=editing_menu()
key=='1'
да
checking_input_int()
p1.toys_count=string-преобразованное в целое число
Конец E3

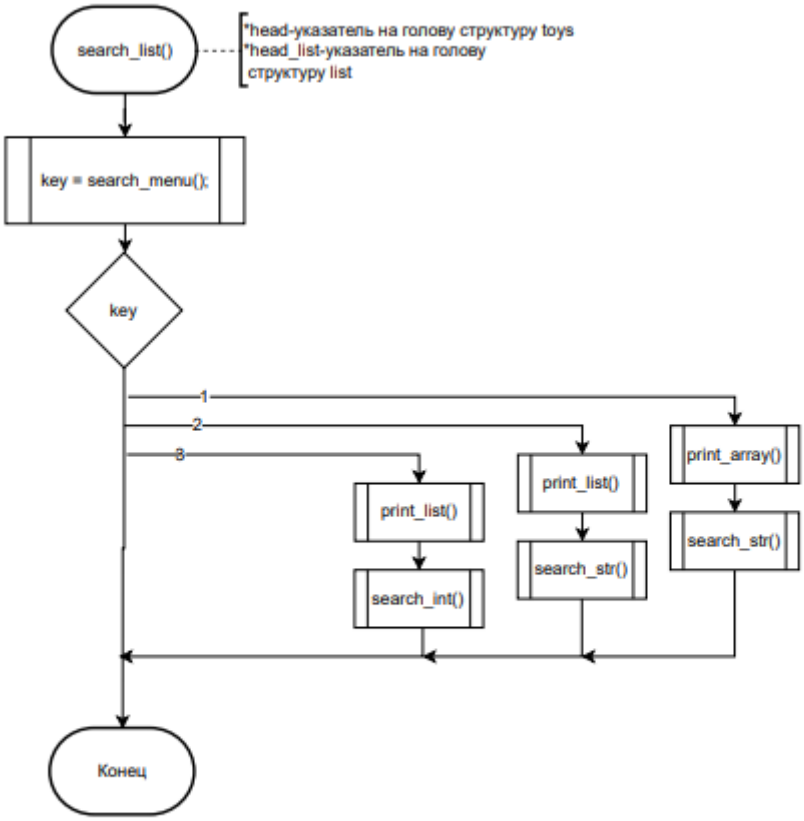

Начало E2
key=editing_menu();
key=='1'
да
Копирование p1.producer.name в string
i=number_elements_manufacturer();
New manufacturer's name
Ввод: string (нового имени)
i==1
да
l1=head_l.first
j=0
l1 не пустой && j!=1
да
l1=make_list()
p1.producer=l1
удаление старого имени и копирование string в l2.name
совпали p1.producer.name l1.name
да
l1=l1.next
l2=l1
j=1
Конец E2

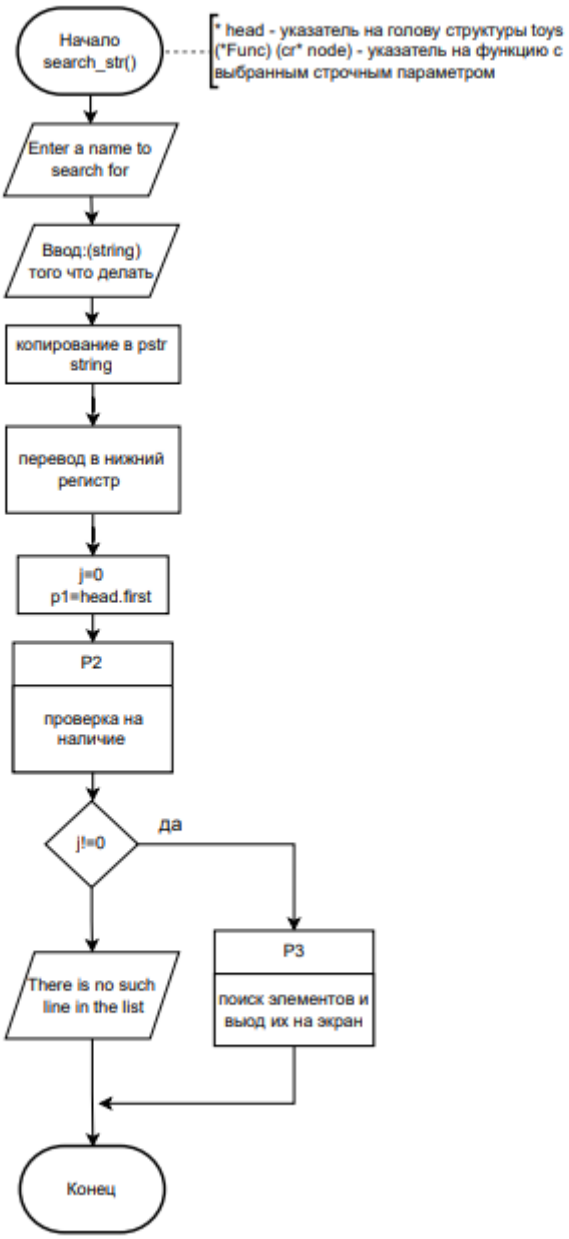

Начало E4
key=editing_menu();
key=='1'
да
checking_input_int()
p1.variations=string-преобразованное в целое число
change_count_ger=1
Конец E4

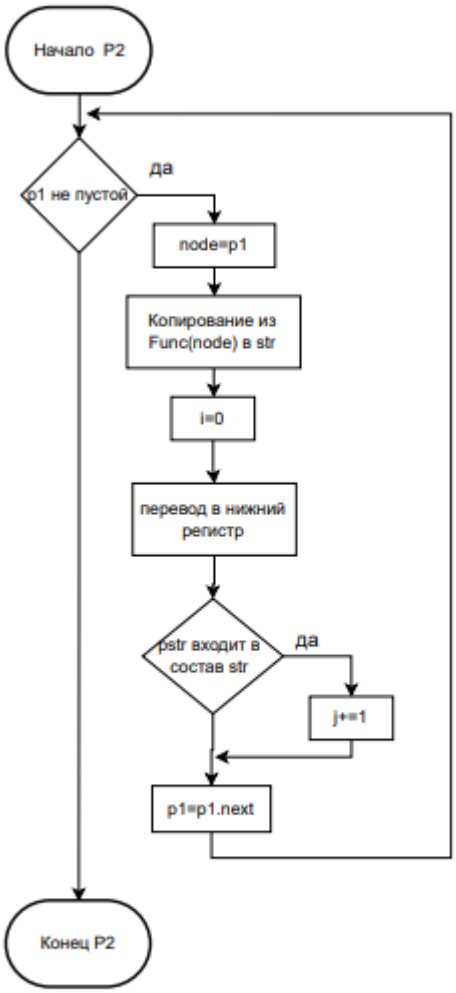

Начало E5
key=editing_menu();
key=='1'
да
checking_input_float()
p1.sales_month=string-преобразованное в вещественное число
Конец E5

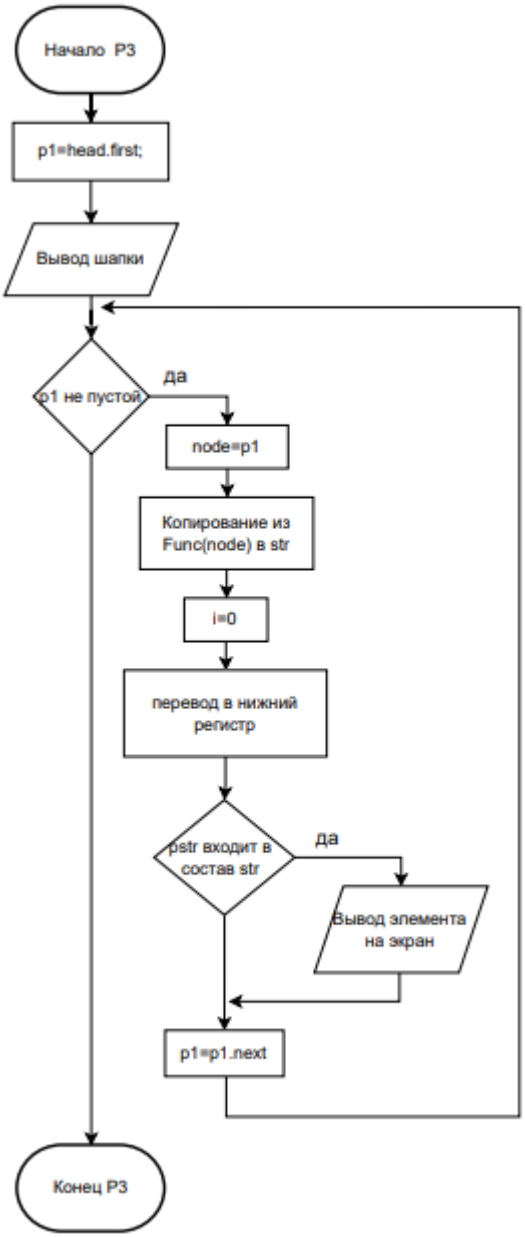

Начало E6
key=editing_menu();
key=='1'
да
checking_input_float()
p1.price=string-преобразованное в вещественное число
Конец E6

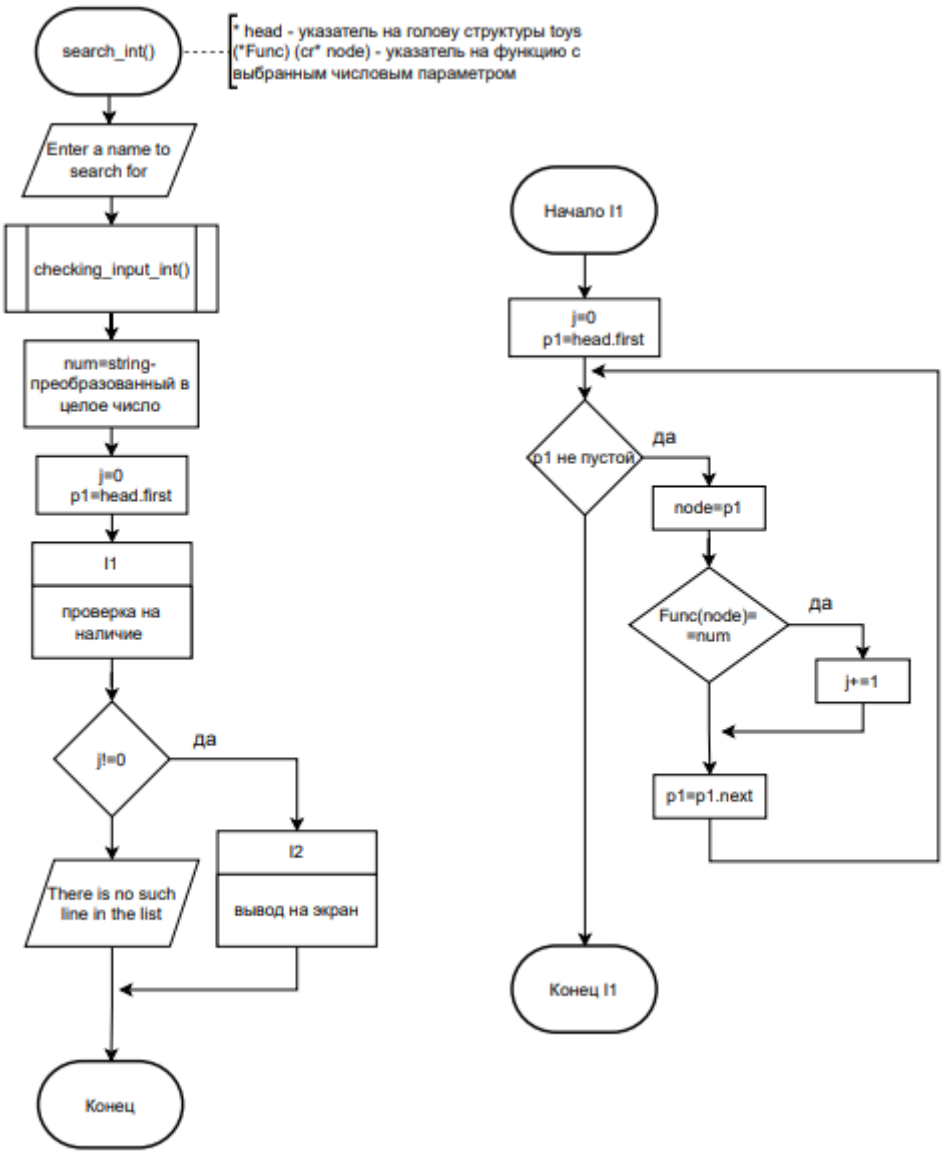

Начало E7
change_count_var!=1
да
Change the ids of variations?
key='1'
key=editing_menu()
key=='1'
да
Id of each varioation
j=0
j < p1.variations
да
Variation-[(j+1)]
checking_input_int()
p1.id_var[i] = string-преобразованное в целое число
j+=1
Конец E7

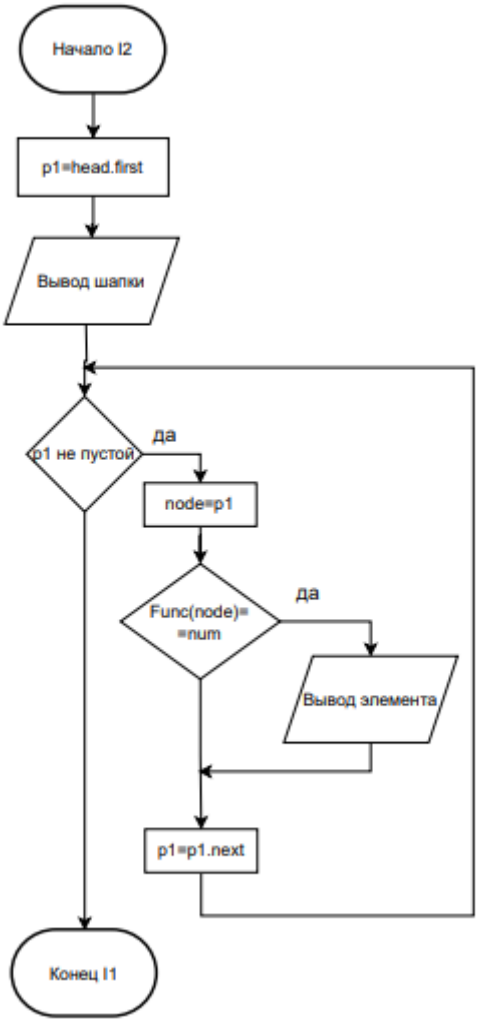

search_list()
*head-указатель на голову структуру toys
*head_list-указатель на голову
структуру list
key = search_menu();
key
1
2
3
print_array()
print_list()
print_list()
search_str()
search_str()
search_int()
Конец

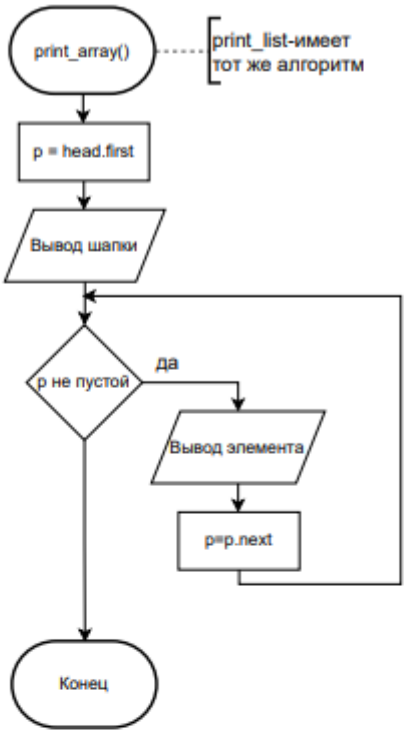

Начало search_str()
* head - указатель на голову структуры toys
(*Func) (cr* node) - указатель на функцию с выбранным строчным параметром
Enter a name to search for
Ввод:(string) того что делать
копирование в pstr string
перевод в нижний регистр
j=0
p1=head.first
P2
проверка на наличие
j!=0
да
There is no such line in the list
P3
поиск элементов и выод их на экран
Конец

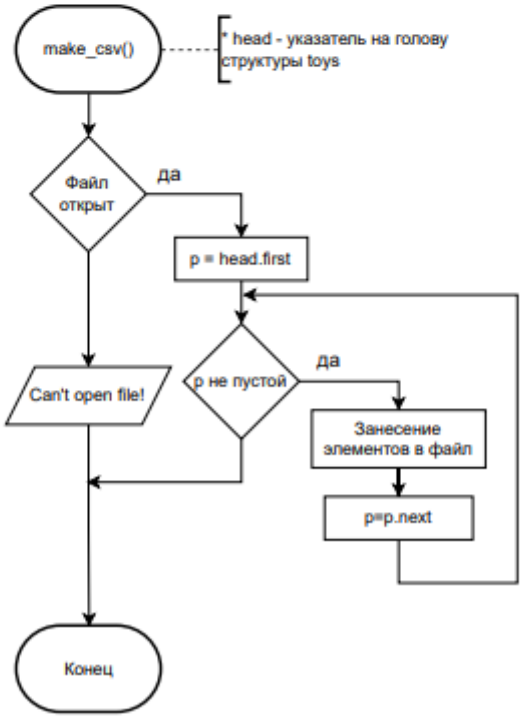

Начало P2
p1 не пустой
да
node=p1
Копирование из Func(node) в str
i=0
перевод в нижний регистр
pstr входит в состав str
да
j+=1
p1=p1.next
Конец P2


Начало P3
p1=head.first;
Вывод шапки
p1 не пустой
да
node=p1
Копирование из Func(node) в str
i=0
перевод в нижний регистр
pstr входит в состав str
да
Вывод элемента на экран
p1=p1.next
Конец P3


search_int()
* head - указатель на голову структуры toys
(*Func) (cr* node) - указатель на функцию с
выбранным числовым параметром
Enter a name to search for
checking_input_int()
num=string-преобразованный в целое число
j=0
p1=head.first
I1
проверка на наличие
j!=0
да
There is no such line in the list
I2
вывод на экран
Конец
Начало I1
j=0
p1=head.first
p1 не пустой
да
node=p1
Func(node)==num
да
j+=1
p1=p1.next
Конец I1


Начало l2
p1=head.first
Вывод шапки
p1 не пустой
да
node=p1
Func(node)==num
да
Вывод элемента
p1=p1.next
Конец l1


print_array()
print_list-имеет тот же алгоритм
p = head.first
Вывод шапки
p не пустой
да
Вывод элемента
p=p.next
Конец


make_csv()
* head - указатель на голову структуры toys
Файл открыт
да
p = head.first
Can't open file!
p не пустой
да
Занесение элементов в файл
p=p.next
Конец

Контрольные примеры:

Helicopter;LEGO;215;5;52.3;9.99;12;15;16;18;21 NERF;Hasbro;420;6;84.1;14.99;101;102;107;111;128;142 MONOPOLY;Hasbro;123;3;15.7;14.99;1001;1002;1054 HotWheels;MATTEL;250;6;80.9;12.49;1;5;6;11;26;27 Spider-Man;Hasbro;149;4;44.4;7.49;121;165;186;188 Minecraft Fox;LEGO;49;2;14.1;24.99;1598;1599 Furby;Hasbro;35;4;38.7;24.49;1005;1006;1011;10171 Boat;Lego;233;5;44.2;13.99;11;12;18;41;42	NERF;Hasbro;420;6;84.1;14.99;101;102;107;111;128;142 MONOPOLY;Hasbro;123;3;15.7;14.99;1001;1002;1054 HotWheels;MATTEL;250;6;80.9;12.49;1;5;6;11;26;27 Spider-Man;Hasbro;149;4;44.4;7.49;121;165;186;188 Furby;Hasbro;35;4;38.7;24.49;1005;1006;1011;1017 Boat;Lego;233;5;44.2;13.99;11;12;18;41;42
Helicopter;LEGO;215;5;52.3;9.99;12;15;16;18;21 NERF;Hasbro;420;6;84.1;14.99;101;102;107;111;128;142 MONOPOLY;Hasbro;123;3;15.7;14.99;1001;1002;1054 HotWheels;MATTEL;250;6;80.9;12.49;1;5;6;11;26;27 Spider-Man;Hasbro;149;4;44.4;7.49;121;165;186;188 Minecraft Fox;LEGO;49;2;14.1;24.99;1598;1599 Furby;Hasbro;35;4;38.7;24.49;1005;1006;1011;1017 Boat;Lego;233;5;44.2;13.99;11;12;18;41;42 2 2	NERF;Hasbro;420;6;84.1;14.99;101;102;107;111;128;142 MONOPOLY;Hasbro;123;3;15.7;14.99;1001;1002;1054 HotWheels;MATTEL;250;6;80.9;12.49;1;5;6;11;26;27 Spider-Man;Hasbro;149;4;44.4;7.49;121;165;186;188 Furby;Hasbro;35;4;38.7;24.49;1005;1006;1011;1017 Boat;Lego;233;5;44.2;13.99;11;12;18;41;42 4 1 1

Текст программы:

```
#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#define inf 256

#define maxlen 128

/*

Examples for adding new items to a list


Boat

Lego

233

5

44.2

13.99

11

12

18

41

42


Tranformers

Hasbro

166

6

7.99

34.8

21

22

64

69

77

78


*/

/* Structure of the list of manufacturers */

struct prod {

    int id;

    char *name;

    struct prod* next;

    struct prod* prev;
```

```
};
typedef struct prod list;
/* The head of the structure of the list of manufacturers */
struct head{
   int cnt;
   list* first;
   list* last;
};
typedef struct head lh1;
/* Structure of the list of toys*/
struct toys
{
      int id;
      char *name;
      list *producer;
      int toys_count;
      int variations;
      float sales_month;
      float price;
      int *id_var;
      struct toys* next;
      struct toys* prev;
};
typedef struct toys cr;
/* Head of toy list structure*/
struct lhead
{
   int cnt;
   cr* first;
   cr* last;
};
typedef struct lhead lh;
/* Creating the head of the list of toys (memory allocation for data)*/
lh* make_head();
/* Creating the head of the list of manufacturers (memory allocation for data)*/
lh1* make_head_list();
/* Creating an element to fill in from the keyboard*/
cr* create_node(lh1* head);
/* Filling in from the keyboard*/
cr* fill_struct(cr* fl,lh1 *p0);
/* opening a file and entering data into lists*/
void file_open(lh *head,lh1 *head_list);
/* creating a list of the 2nd list of manufacturers*/
list* copy_create_node(list *p1);
```

```
/* Reading data from a file*/
char **simple_split(char *str, int length, char sep);
/* putting elements at the end  for the list of manufacturers*/
void add_last_list(lh1* head, list* node, list* pnode);
/* putting an element in the first position if the list is empty  for the list of manufacturers*/
void add_first_list(lh1* head, list* node);
/* distribution of elements by position*/
list* make_list(char* str, lh1* head);
/* clearing the list of manufacturers*/
void clear_list(list* fl);
/* clearing the list of toys*/
void clear_struct(cr* fl);
/* clearing an array to read data from a file*/
void clear_arr (char** s2, int num);
/* putting an element in the first position if the list is empty for the list of toys*/
void add_first(lh* head, cr* node);
/* putting elements at the end for the list of toys*/
void add_last(lh* head, cr* nnode, cr* pnode);
/* placing items at the end for a formative list based on a list of manufacturers*/
void add_copy_last_list(lh1* head, list* node, list* pnode);
/* output a list of toys*/
void print_array(lh* head);
/* output a list of manufacturers*/
void print_list(lh1* head);
/* menu*/
char choosing_action();
/* deleting elements by an introductory substring*/
void deleting_element(lh* head,lh1* head_l);
/* output of a request to increase the console*/
void beginning();
/* pointer to a numeric structure parameter*/
float ID(cr* node);
/* pointer to a string structure parameter*/
char *NAME(cr* node);
/*pointer to a string structure parameter*/
char *PRODS(cr* node);
/* pointer to a numeric structure parameter*/
float T_COUNT(cr* node);
/* pointer to a numeric structure parameter*/
float VAR(cr* node);
/* pointer to a numeric structure parameter*/
float SALES_M(cr *node);
/* pointer to a numeric structure parameter*/
float PRICE(cr *node);
```

```
/* pointer to a numeric structure parameter*/
float VAR_ID(cr * node);
/* sorting by numbers*/
void sorting_num(lh* head, float (*Func) (cr* node));
/* sorting by lines*/
void sorting_str(lh* head, char* (*Func) (cr* node));
/* reverse sorting by numbers*/
void sorting_num_reverse(lh* head, float (*Func) (cr* node));
/* reverse sorting by lines*/
void sorting_str_reverse(lh* head, char* (*Func) (cr* node));
/* search by lines*/
void search_str(lh* head, char* (*Func) (cr* node));
/* searches by numbers*/
void search_int(lh* head, float (*Func) (cr* node));
/* sorting menu*/
char param_menu();
/* actions for sorting depending on the selected number*/
void sort_list(lh* head);
/* editing elements*/
void editing_list(lh *head ,lh1 *head_l);
/* saving to a file*/
void make_csv(lh* head);
/* search menu*/
void search_list(lh* head,lh1 *head_l);
/* checking for the number of elements*/
int number_elements_manufacturer(lh *head,lh1 *head_l,char str[inf]);
/* output the help text for each function*/
void help_output();
/* menu for reference*/
char help_menu();
/* checking for the input of an integer*/
void checking_input_int(char string[inf]);
/* checking for the input of a real number*/
void checking_input_float(char string[inf]);
/* ---------------------------------------- */
int main(){
    lh *p0=NULL;
    lh1 *l0=NULL;
    cr *p1=NULL,*p2=NULL;
    list *l1=NULL,*l2=NULL;
    char key;
    beginning();
    system("@cls||clear");
    p0=make_head();
```

```
if (p0!=NULL){
    l0=make_head_list();
    if (l0!=NULL){
        file_open(p0,l0);
        print_array(p0);
        print_list(l0);
        system("@cls||clear");
        key=choosing_action();
        while(key!='7'){

            if(key=='1'){
                p2=create_node(l0);
                if(p0->cnt==0){
                    add_first(p0, p2);
                }
                else{
                    add_last(p0, p2, p0->last);
                }
                system("@cls||clear");
                print_array(p0);
                print_list(l0);
            }
            else if (key=='2'){
                system("@cls||clear");
                deleting_element(p0,l0);
            }
            else if (key=='3'){
                print_array(p0);
                puts("");
                print_list(l0);
            }
            else if(key=='4'){
                sort_list(p0);
                system("@cls||clear");
            }
            else if(key=='5'){
                editing_list(p0,l0);
                system("@cls||clear");
            }
            else if(key=='6'){
                search_list(p0,l0);
                system("@cls||clear");
            }
```

```
    else if(key=='0'){
        help_output();
        system("@cls||clear");
    }
    system("@cls||clear");
    key = choosing_action();
}
make_csv(p0);
p1 = p0->first;
while (p1)
{
    p1->producer = NULL;
    p1 = p1->next;
}
if (l0)
{
    if (l0->cnt != 0)
    {
        l1 = l0->first;
        l2 = l0->first;
        while(l1) {
            l2 = l1->next;
            clear_list(l1);
            l1 = l2;
        }
        l1 = NULL;
        l2 = NULL;
        free(l0);
    }else free(l0);
}
if (p0)
{
    if (p0->cnt != 0)
    {
        p1 = p0->first;
        p2 = p0->first;
        while(p2->next !=NULL)
        {
            p2 = p2->next;
            clear_struct(p1);
            p1 = p2;
        }
        clear_struct(p1);
        p1 = NULL;
```

```
                    p2 = NULL;
                    free(p0);
                } else free(p0);
            }
    }
    else{
        free(l0);
        puts("Error with memory!");
    }
}
else{
    free(p0);
    puts("Error with memory!");
}
return 0;
}
/* ---------------------------------------- */
void beginning(){
    printf("In order to see the result of the work, please enlarge the window.\n");
    printf("To continue, click enter.\n");
    getchar();
}
/* ---------------------------------------- */
lh* make_head(){
    lh *ph = NULL;
    ph =(lh*) malloc(sizeof(lh));
    if (ph)
    {
        ph->cnt = 0;
        ph->first = NULL;
        ph->last=NULL;
    }
    else{
         free(ph);
        puts("Error with memory!");
    }
    return ph;
}
/* ---------------------------------------- */
lh1* make_head_list(){
    lh1* ph = NULL;
    ph =(lh1*) malloc(sizeof(lh1));
    if (ph)
    {
```

```
        ph->cnt = 0;
        ph->first = NULL;
        ph->last=NULL;
    }
    else{
         free(ph);
        puts("Error with memory!");
    }
    return ph;
}
/* ------------------------------------------ */
void add_first(lh* head, cr* node)
{
    if(head&&node)
    {
        head->first = node;
        head->last = node;
        node->next=NULL;
        node->prev=NULL;
        head->cnt++;
    }
}
/* ------------------------------------------ */
void add_last(lh* head, cr* nnode, cr* pnode)
{
    int n;
    if(head&&nnode&&pnode)
    {
        n = head->cnt+1;
        pnode->next = nnode;
        nnode->prev=pnode;
        nnode->next=NULL;
        head->last = nnode;
        nnode->id = n;
        head->cnt = n;
    }

}
/* ------------------------------------------ */
cr* create_node(lh1* head){
    cr* node = NULL;
    node = (cr*)malloc(sizeof(cr));
    if (node)
    {
```

```
        node->id = 1;
        node = fill_struct(node,head);
        if (node) {
            node->next=NULL;
            node->prev=NULL;
        }
    }
    else{
        free(node);
        puts("Error with memory!");
    }
    return node;
}
/* ---------------------------------------- */
cr* fill_struct(cr* fl,lh1 *p0)
{
    int j;
    list *p1;
    char string[inf];
    printf("Toy name: ");
    fgets(string, inf, stdin);
    string[strlen(string)-1] = '\0';
    fl->name = (char*) malloc((strlen(string)+1)*sizeof(char));
    if (fl->name)
    {
        strcpy(fl->name, string);
        printf("Manufacturer's name: ");
        fgets(string, inf, stdin);
        string[strlen(string)-1] = '\0';
        p1=make_list(string,p0);
        if(p1!=NULL){
                fl->producer=p1;
                printf("Toy count at stock: ");
                checking_input_int(string);
                fl->toys_count=atoi(string);
                printf("Number of variations: ");
                checking_input_int(string);
                fl->variations=atoi(string);
                printf("Average toy sales per month: ");
                checking_input_float(string);
                fl->sales_month=atof(string);
                printf("Price in USD: ");
                checking_input_float(string);
                fl->price=atof(string);
```

```
            fl->id_var = (int*) malloc((fl->variations)*sizeof(int));
            if(fl->id_var)
            {
               printf("Id of each variation.\n");
               for(j=0; j < fl->variations; j++)
               {
                  printf("Variation-[%d]: ", (j+1));
                  checking_input_int(string);
                  fl->id_var[j]=atoi(string);
               }
            } else
            {
               puts("Error with memory!");
               clear_struct(fl);
            }
         }
         else
         {
            puts("Error with memory!");
            clear_struct(fl);
         }
      } else
      {
         puts("Error with memory!");
         clear_struct(fl);
      }
      return fl;
   }
   /* ---------------------------------------- */
   void file_open(lh *head,lh1 *head_list){
      int i, j, n,slen;
      cr *p1=NULL,*p2=NULL;
      char **s2;
      char s1[maxlen],sep;
      FILE *df;
      df=fopen("kolenko_course.txt","r");
      n=0;
      sep=';';
      if (df!=NULL){
         while(fgets(s1, maxlen, df) != NULL) n++;
         rewind(df);

         for (i=0;i<n;i+=1){
            fgets(s1,maxlen,df);
```

```
    s1[strlen(s1)-1]='\0';
    slen=strlen(s1);
    s2=simple_split(s1,slen,sep);
    p1=(cr*)malloc(sizeof(cr));
    if (s2!=NULL && p1!=NULL){

        p1->name=(char*)malloc((strlen(s2[0])+1)*sizeof(char));
        p1->producer=make_list(s2[1],head_list);
        p1->id_var=(int*)malloc((atoi(s2[3])+1)*sizeof(int));

        if ((p1->name!=NULL)&&(p1->id_var!=NULL)){

            p1->id = 1;
            strcpy(p1->name,s2[0]);
            p1->toys_count=atoi(s2[2]);
            p1->variations=atoi(s2[3]);
            p1->sales_month=atof(s2[4]);
            p1->price=atof(s2[5]);
            for(j=0;j<p1->variations;j+=1){
                p1->id_var[j]=atoi(s2[j+6]);
            }
            p1->next=NULL;
        }
        else{
            clear_struct(p1);
        }
        clear_arr(s2,p1->variations+6);

    }
    else{
        i=n;
        p1=NULL;
        free(p1);
        printf("Error with memory!");
    }

    if (i==0){
        add_first(head,p1);
    }
    else{
        add_last(head,p1,p2);
    }
    p2=p1;
}
```

```
        fclose(df);
    }
    else{
        puts("File opening error.");
        printf("To continue, click enter");
        getchar();
    }
}
/* ---------------------------------------- */
void clear_arr (char** s2, int num){
    int i;
    for(i=0; i < num; i++) free(s2[i]);
    free(s2);
}
/* ---------------------------------------- */
char **simple_split(char *str, int length, char sep)
{
    char **str_array=NULL;
    int i,j,k,m;
    int key,count;
    for(j=0,m=0;j<length;j++)
    {
        if(str[j]==sep) m++;
    }
    key=0;
    str_array=(char**)malloc((m+1)*sizeof(char*));
    if(str_array!=NULL)
    {
        for(i=0,count=0;i<=m;i++,count++)
        {
            str_array[i]=(char*)malloc(length*sizeof(char));
            if(str_array[i]!=NULL) key=1;
            else
            {
                key=0;
                i=m;
            }
        }
        if(key)
        {
            k=0;
            m=0;
            for(j=0;j<length;j++)
            {
```

```
            if(str[j]!=sep) str_array[m][j-k]=str[j];
            else
            {
               str_array[m][j-k]='\0';
               k=j+1;
               m++;
            }
         }
      }
      else
      {
         clear_arr(str_array,count);
      }
   }

   return str_array;
}
/* ---------------------------------------- */
list* make_list(char* str, lh1* head)
{
   list *node;
   int flag;

   if (head->cnt == 0)
   {
      node = (list*) malloc (sizeof(list));
      if (node)
      {
         node->name = (char*) malloc ((strlen(str)+1)*sizeof(char));
         if (node->name)
         {
            strcpy(node->name, str);
            add_first_list(head, node);
         } else clear_list(node);
      }
      else{
         free(node);
         puts("Error with memory!");
      }
   }else
   {
      node = head->first;
      flag = 1;
      while ((node != NULL) && (flag == 1))
```

```
        {
            if (strcmp(node->name, str) == 0) flag = 0;
            if (flag) node = node->next;
        }
        if(flag)
        {
            node = (list*) malloc (sizeof(list));
            if (node)
            {
                node->name = (char*) malloc ((strlen(str)+1)*sizeof(char));
                if (node->name)
                {
                    strcpy(node->name, str);
                    add_last_list(head, node, head->last);
                } else clear_list(node);
            }
            else{
            free(node);
            puts("Error with memory!");
            }
        }
    }
    return node;
}
/* ---------------------------------------- */
void add_first_list(lh1* head, list* node)
{
    if (head&&node)
    {
        head->first = node;
        head->last = node;
        node->next = NULL;
        node->prev = NULL;
        node->id = ++head->cnt;
    }
}
/* ---------------------------------------- */
void add_last_list(lh1* head, list* node, list* pnode)
{
    if (head&&node)
    {
        pnode->next = node;
        head->last = node;
        node->prev = pnode;
```

```
        node->next = NULL;
        node->id = ++head->cnt;
    }
}
/* ---------------------------------------- */
void clear_list(list* fl)
{
    if (fl->name) free(fl->name);
    fl->name=NULL;
    fl->next=NULL;
    fl->prev=NULL;
    free(fl);
    fl=NULL;
}
/* ---------------------------------------- */
void clear_struct(cr* fl)
{
    if (fl->name) free(fl->name);
    if (fl->id_var) free(fl->id_var);
    fl->name=NULL;
    fl->id_var=NULL;
    fl->next = NULL;
    fl->prev=NULL;
    free(fl);
    fl=NULL;
}
/* ---------------------------------------- */
void print_array(lh* head)
{
    cr* p = NULL;
    int i;

    p = head->first;
    puts("Catalog of toys:");
    printf("+----+-------------------------+-------------------------+-----------------+---------------+----------------------+----------------------+-------------------------------+\n");
    printf("|%4s|%27s|%27s|%10s|%16s|%25s|%25s||%35s |","ID","Toy name","Toy manufacturer","Toy count at stock","Count variations","Average sales per month","Price in USD","ID of variation");
    printf("\n+----+-------------------------+-------------------------+-----------------+---------------+----------------------+----------------------+-------------------------------+\n");
    while (p!=NULL)
    {
       printf("|%4d|%27s|%27s|%18d|%16d|%25g|%25g||",p->id,p->name,p->producer->name,p->toys_count,p->variations, p->sales_month, p->price);
        for(i = 0; i < p->variations; i++)  printf("%5d|",p->id_var[i]);
        printf("|");
```

```
        printf("\n+----+-------------------------+-------------------------+-----------------+---------------+-----------------------+-----------------------+-------------------------
----------+\n");

        p = p->next;

    }

    printf("To continue, click enter");

    getchar();

}

/* ---------------------------------------- */

void print_list(lh1* head)

{

    list* p = NULL;


    p = head->first;

    puts("List of manufacturers:");

    printf("+------+------------------+\n");

    printf("| %4s | %15s |\n", "ID", "Toy manufacturer");

    printf("+------+------------------+\n");

    while (p)

    {

        printf("| %4i | %15s  |\n", p->id, p->name);

        printf("+------+------------------+\n");

        p = p->next;

    }

    printf("To continue, click enter");

    getchar();

}

/* ---------------------------------------- */

char choosing_action()

{

    char a, b;

    b = ' ';

    while ((b != '1')&& (b != '2') && (b != '0')&& (b != '3') && (b!='4')&& (b!='5')&& (b!='6')&& (b!='7'))

    {

    a = ' ';

    puts("Please,select mode:");

    puts("Enter 0 - reference");

    puts("Enter 1 - add new element");

    puts("Enter 2 - delete element");

    puts("Enter 3 - print lists");

    puts("Enter 4 - sorting list");

    puts("Enter 5 - editing an entry");

    puts("Enter 6 - search in the list");

    puts("Enter 7 - exit");

    while (a != '\n')
```

```
    {
        printf("Input: ");
        a = getc(stdin);
        if (a == '1') b='1';
        else if (a == '2') b='2';
        else if (a == '3') b='3';
        else if (a == '4') b='4';
        else if (a == '5') b='5';
        else if (a == '6') b='6';
        else if (a == '7') b='7';
        else if (a == '0') b='0';
    }
    system("@cls||clear");
    }
    return b;
}
/* ---------------------------------------- */
void deleting_element(lh* head,lh1* head_l)
{
   cr* p1=NULL,*p2=NULL;
   list *l1=NULL,*l2=NULL;
   int num,i,j;
   char a,b;
   char line[maxlen];
   b = ' ';
   while ((b != '1')&& (b != '2') && (b !='0')){
       a = ' ';
       puts("Select in which list you want to delete:");
       puts("Enter 1 - list of toys");
       puts("Enter 2 - list of manufacturers");
       puts("Enter 0 - exit");
       while (a != '\n'){
           printf("Input: ");
           a = getc(stdin);
           if (a == '1') b='1';
           else if (a == '2') b='2';
           else if (a == '0') b='0';
       }
       system("@cls||clear");
   }
   if(b!='0'){
       if (b=='1'){
           print_array(head);
       }
```

```
        else{
            print_list(head_l);
        }
        printf("Enter the ID of the item you want to delete: ");
        checking_input_int(line);
        num=atoi(line);
        system("@cls||clear");
        if (num>=1 && num<=head->cnt){
            if (b=='1'){
                p1=head->first;
                puts("The item to be deleted:");
                printf("+----+-------------------------+-------------------------+-----------------+---------------+------------------------+------------------------+-------------------------------+\n");
                printf("|%4s|%27s|%27s|%10s|%16s|%25s|%25s||%35s |","ID","Toy name","Toy manufacturer","Toy count at stock","Count variations","Average sales per month","Price in USD","ID of variation");
                printf("\n+----+-------------------------+-------------------------+-----------------+---------------+------------------------+------------------------+-------------------------------+\n");
                while(p1!=NULL){
                    if (p1->id==num){
                        printf("|%4d|%27s|%27s|%18d|%16d|%25g|%25g||",
                                p1->id,p1->name,p1->producer->name,p1->toys_count,p1->variations, p1->sales_month, p1->price);
                        for(i = 0; i < p1->variations; i++)  printf("%5d|",p1->id_var[i]);
                        printf("|");
                        printf("\n+----+-------------------------+-------------------------+-----------------+---------------+------------------------+------------------------+-------------------------------+\n");
                        strcpy(line,p1->producer->name);
                        i=number_elements_manufacturer(head,head_l,line);
                        if (i==1){
                            l1=head_l->first;
                            j=0;
                            while(l1!=NULL && j!=1){
                                if (strcmp(l1->name,p1->producer->name)==0){
                                    l2=l1;
                                    j=1;
                                }
                                l1=l1->next;
                            }
                            if (p1==head->first){
                                head->first=p1->next;
                                head->first->prev=NULL;
                            }

                            else if (p1==head->last){
                                head->last=p1->prev;
                                head->last->next=NULL;
                            }
```

```
    else{
        p1->prev->next=p1->next;
        p1->next->prev=p1->prev;
    }
    p1->producer=NULL;
    clear_struct(p1);
    p1=NULL;

    if (l2==head_l->first){
        head_l->first=l2->next;
        head_l->first->prev=NULL;
    }
    else if (l2==head_l->last){
        head_l->last=l2->prev;
        head_l->last->next=NULL;
    }
    else{
        l2->prev->next=l2->next;
        l2->next->prev=l2->prev;
    }
    clear_list(l2);

}
else{
    if (p1==head->first){
        head->first=p1->next;
        head->first->prev=NULL;
    }

    else if (p1==head->last){
        head->last=p1->prev;
        head->last->next=NULL;
    }

    else{
        p1->prev->next=p1->next;
        p1->next->prev=p1->prev;
    }
    p1->producer=NULL;
    clear_struct(p1);
    p1=NULL;

}
```

```
            }
            else{
                p1=p1->next;
            }
        }
        print_array(head);

    }
else{
        l1=head_l->first;
        while(l1!=NULL){
            if (l1->id==num){
                strcpy(line,l1->name);
                l1=NULL;
            }
            else{
                l1=l1->next;
            }

        }
        p1 = head->first;
        puts("The item to be deleted:");
        printf("+----+--------------------------+--------------------------+-----------------+---------------+------------------------+------------------------+-------------------------------+\n");
        printf("|%4s|%27s|%27s|%10s|%16s|%25s|%25s||%35s |","ID","Toy name","Toy manufacturer","Toy count at stock","Count variations","Average sales per month","Price in USD","ID of variation");
        printf("\n+----+--------------------------+--------------------------+-----------------+---------------+------------------------+------------------------+-------------------------------+\n");
        while(p1!=NULL){
            if (strstr(p1->producer->name,line)!=NULL){
                printf("|%4d|%27s|%27s|%18d|%16d|%25g|%25g||",
                        p1->id,p1->name,p1->producer->name,p1->toys_count,p1->variations, p1->sales_month, p1->price);
                for(i = 0; i < p1->variations; i++)  printf("%5d|",p1->id_var[i]);
                printf("|");
                printf("\n+----+--------------------------+--------------------------+-----------------+---------------+------------------------+------------------------+-------------------------------+\n");
              if (p1==head->first){
                  head->first=p1->next;
                  head->first->prev=NULL;
                }

                else if (p1==head->last){
                  head->last=p1->prev;
                  head->last->next=NULL;
                }
```

```
        else{
            p1->prev->next=p1->next;
            p1->next->prev=p1->prev;
        }
        p2=p1->next;
        p1->producer=NULL;
        clear_struct(p1);
        p1=p2;
    }
    else{
        p1=p1->next;
    }
}
print_array(head);

l1=head_l->first;
printf("The item to be deleted:\n");
printf("+----+----------------------------+\n");
printf("|%4s|%27s|","ID","Toy manufacturer");
printf("\n+----+----------------------------+\n");
while(l1!=NULL){
    if (strstr(l1->name,line)!=NULL){
        printf("|%4d|%27s|",l1->id,l1->name);
        printf("\n+----+----------------------------+\n");
        if (l1==head_l->first){
            head_l->first=l1->next;
            head_l->first->prev=NULL;

        }
        else if (l1==head_l->last){
            head_l->last=l1->prev;
            head_l->last->next=NULL;

        }
        else{
            l1->prev->next=l1->next;
            l1->next->prev=l1->prev;
        }

        clear_list(l1);
        l1=NULL;
    }
    else{
        l1=l1->next;
```

```
                }
            }
            print_list(head_l);
        }
    }
    else{
        printf("There is no such id in the list\n");
        printf("To continue, click enter");
        getchar();
    }
}
}
/* ---------------------------------------- */
float ID(cr *node){
    return (node->id);
}
/* ---------------------------------------- */
char *NAME(cr *node){
    return (node->name);
}
/* ---------------------------------------- */
char *PRODS(cr *node){
    return (node->producer->name);
}
/* ---------------------------------------- */
float T_COUNT(cr *node){
    return (node->toys_count);
}
/* ---------------------------------------- */
float VAR(cr *node){
    return (node->variations);
}
/* ---------------------------------------- */
float SALES_M(cr *node){
    return(node->sales_month);
}
/* ---------------------------------------- */
float PRICE(cr *node){
    return(node->price);
}
/* ---------------------------------------- */
float VAR_ID(cr *node){
    int i;
    float sum;
```

```
    sum=0;
    for (i=0;i<=node->variations;i+=1){
        sum+=node->id_var[i];
    }
    sum/=node->variations;
    return sum;
}
/* ---------------------------------------- */
void sorting_num(lh* head, float (*Func) (cr* node))
{
    cr *cur, *root, *newroot, *node;
    root = NULL;
    newroot = NULL;
    node = NULL;
    root = head->first;
    while (root)
    {
        node = root;
        root = root->next;
        if (newroot == NULL || Func(node) < Func(newroot))
        {
            node->next = newroot;
            newroot = node;
        } else
        {
            cur = newroot;
            while (cur->next != NULL && !(Func(node) < Func(cur->next))) cur = cur->next;
            node->next = cur->next;
            cur->next = node;
        }
    }
    head->first = newroot;
    root=head->first;
    while(root->next!=NULL){
        root=root->next;
    }
    head->last=root;
}
/* ---------------------------------------- */
void sorting_str(lh* head, char* (*Func) (cr* node))
{
    cr *cur, *root, *newroot, *node;
    root = NULL;
    newroot = NULL;
```

```
    node = NULL;
    root = head->first;
    while (root)
    {
        node = root;
        root = root->next;
        if (newroot == NULL || strcmp(Func(node), Func(newroot)) < 0)
        {
            node->next = newroot;
            newroot = node;
        } else
        {
            cur = newroot;
            while (cur->next != NULL && !(strcmp(Func(node),Func(cur->next)) < 0)) cur = cur->next;
            node->next = cur->next;
            cur->next = node;
        }
    }
    head->first = newroot;
    root=head->first;
    while(root->next!=NULL){
        root=root->next;
    }
    head->last=root;
}
/* ---------------------------------------- */
void sorting_num_reverse(lh* head, float (*Func) (cr* node)){
    cr *cur, *root, *newroot, *node;
    root = NULL;
    newroot = NULL;
    node = NULL;
    root = head->first;
    while (root)
    {
        node = root;
        root = root->next;
        if (newroot == NULL || Func(node) > Func(newroot))
        {
            node->next = newroot;
            newroot = node;
        } else
        {
            cur = newroot;
            while (cur->next != NULL && !(Func(node) > Func(cur->next))) cur = cur->next;
```

```
            node->next = cur->next;
            cur->next = node;
        }
    }
    head->first = newroot;
    root=head->first;
    while(root->next!=NULL){
        root=root->next;
    }
    head->last=root;
}
/* ---------------------------------------- */
void sorting_str_reverse(lh* head, char* (*Func) (cr* node))
{
    cr *cur, *root, *newroot, *node;
    root = NULL;
    newroot = NULL;
    node = NULL;
    root = head->first;
    while (root)
    {
        node = root;
        root = root->next;
        if (newroot == NULL || strcmp(Func(node), Func(newroot)) > 0)
        {
            node->next = newroot;
            newroot = node;
        } else
        {
            cur = newroot;
            while (cur->next != NULL && !(strcmp(Func(node),Func(cur->next)) > 0)) cur = cur->next;
            node->next = cur->next;
            cur->next = node;
        }
    }
    head->first = newroot;
    root=head->first;
    while(root->next!=NULL){
        root=root->next;
    }
    head->last=root;
}
/* ---------------------------------------- */
char param_menu()
```

```
{
    char a, b;

    b = ' ';
    while ((b != '1') && (b != '2') && (b != '3') && (b != '4') && (b != '5') && (b != '6') && (b != '7') && (b != '8') && (b != '0'))
    {
        a = ' ';
        puts("Please, select parameter:");
        puts("Enter 1 - id ");
        puts("Enter 2 - name");
        puts("Enter 3 - manufacturer");
        puts("Enter 4 - toy count at stock");
        puts("Enter 5 - number of variations");
        puts("Enter 6 - average sales per month");
        puts("Enter 7 - price of the toy");
        puts("Enter 8 - all ids of variations");
        puts("Enter 0 - exit");
        printf("Input: ");
        while (a != '\n')
        {
            a = getc(stdin);
            if (a == '1') b = '1';
            else if (a == '2') b = '2';
            else if (a == '3') b = '3';
            else if (a == '4') b = '4';
            else if (a == '5') b = '5';
            else if (a == '6') b = '6';
            else if (a == '7') b = '7';
            else if (a == '8') b = '8';
            else if (a == '0') b = '0';
        }
        system("@cls||clear");
    }
    return b;
}
/* ---------------------------------------- */
void sort_list(lh* head)
{
    char key;
    char a,b;
    key = param_menu();
    system("@cls||clear");
    b = ' ';
    while ((b != '1') && (b != '2') && (key!='0'))
```

```
{
    a = ' ';
    puts("Choose the way how to sort:");
    puts("Enter 1- sort in direct order");
    puts("Enter 2- sort in reverse order");
    printf("Input: ");
    while (a != '\n')
    {
        a = getc(stdin);
        if (a == '1') b = '1';
        else if (a == '2') b = '2';
    }
    system("@cls||clear");
}
switch(key){

    case('1'):
        {
            if(b=='1'){
                sorting_num(head, ID);
                break;
            }
            else {
                sorting_num_reverse(head, ID);
                break;
            }
        }
    case ('2'):
        {
            if(b=='1'){
                sorting_str(head, NAME);
                break;
            }
            else{
                sorting_str_reverse(head, NAME);
                break;
            }
        }
    case('3'):
        {
            if (b=='1'){
                sorting_str(head, PRODS);
                break;
            }
```

```
        else{
            sorting_str_reverse(head, PRODS);
            break;
        }
    }
case('4'):
    {   if (b=='1'){
            sorting_num(head, T_COUNT);
            break;
        }
        else{
            sorting_num_reverse(head, T_COUNT);
            break;
        }
    }
case('5'):
    {
        if (b=='1'){
            sorting_num(head, VAR);
            break;
        }
        else{
            sorting_num_reverse(head, VAR);
            break;
        }
    }
case('6'):
    {
        if (b=='1'){
            sorting_num(head, SALES_M);
            break;
        }
        else{
            sorting_num_reverse(head, SALES_M);
            break;
        }
    }
case('7'):
    {
        if (b=='1'){
            sorting_num(head, PRICE);
            break;
        }
        else{
```

```
                sorting_num_reverse(head, PRICE);
                break;
            }
        }
    case('8'):
        {
            if (b=='1'){
                sorting_num(head, VAR_ID);
                break;
            }
            else{
                sorting_num_reverse(head, VAR_ID);
                break;
            }
        }
        print_array(head);
    }

}
/* ---------------------------------------- */
char editing_menu()
{
    char a, b;
    b = ' ';
    while ((b != '1') && (b != '2'))
    {
        a = ' ';
        puts("Enter 1 - edit an element");
        puts("Enter 2 - leave it unchanged and move on to the next one");
        printf("Input: ");
        while (a != '\n')
        {
            a = getc(stdin);
            if (a == '1') b = '1';
            else if (a == '2') b = '2';
        }
        system("@cls||clear");
    }
    return b;
}
/* ---------------------------------------- */
void editing_list(lh *head ,lh1 *head_l)
{
    int j,i,num;
```

```
cr *p1=NULL;
list *l1=NULL,*l2=NULL;
char string[inf];
char key;
int change_count_var;
printf("Enter the number of the item you want to edit: ");
checking_input_int(string);
num=atoi(string);
system("@cls||clear");
if (num>=1 && num <=head->cnt){
   p1=head->first;
   for (i=0;i<head->cnt;i+=1){
      if (num==i+1){
         p1=p1->prev;
         i=head->cnt;
      }
      else{

         p1=p1->next;
      }
   }
   system("@cls||clear");
   puts("Change the toy name?");
   key=editing_menu();
   if (key=='1'){
         free(p1->name);
         p1->name=NULL;
         printf("Toy name: ");
         fgets(string, inf, stdin);
         string[strlen(string)-1] = '\0';
         while(key!='2'){
            p1->name = (char*) malloc((strlen(string)+1)*sizeof(char));
            if (p1->name!=NULL){
               strcpy(p1->name, string);
               key='2';
            }
            else{
               puts("Error with memory!");
               free(p1->name);
               p1->name=NULL;
            }
         }
   }
   system("@cls||clear");
```

```
puts("Change the manufacturer's name?");
key=editing_menu();
if (key=='1'){
   strcpy(string,p1->producer->name);
   i=number_elements_manufacturer(head,head_l,string);
   printf("Manufacturer's name: ");
   fgets(string, inf, stdin);
   string[strlen(string)-1] = '\0';
   if (i==1){
      l1=head_l->first;
      j=0;
      while(l1!=NULL && j!=1){
         if (strcmp(l1->name,p1->producer->name)==0){
            l2=l1;
            j=1;
         }
         l1=l1->next;
      }
      p1->producer=NULL;
      free(l2->name);
      l2->name=NULL;
      while(key!='2'){
         l2->name = (char*) malloc((strlen(string)+1)*sizeof(char));
         if (l2->name!=NULL){
            strcpy(l2->name,string);
            p1->producer=l2;
            key='2';
         }
         else{
            free(l2->name);
            l2->name=NULL;
         }
      }

   }
   else{
      p1->producer=NULL;
      l1=make_list(string,head_l);
      p1->producer=l1;
   }
}
system("@cls||clear");
puts("Change the value of the toy count at stock?");
key=editing_menu();
```

```
if(key=='1'){
    printf("Toy count at stock: ");
    checking_input_int(string);
    p1->toys_count=atoi(string);
}
system("@cls||clear");
puts("Change the count of variations?");
key=editing_menu();
if (key=='1'){
    printf("Count of variations: ");
    checking_input_int(string);
    p1->variations=atoi(string);
    change_count_var=1;

}
system("@cls||clear");
puts("Change the average sales per month?");
key=editing_menu();
if (key=='1'){
    printf("Average sales per month: ");
    checking_input_float(string);
    p1->sales_month=atof(string);
}
system("@cls||clear");
puts("Change the price of the toy in (USD)?");
key=editing_menu();
if (key=='1'){
    printf("Price in USD: ");
    checking_input_float(string);
    p1->price=atof(string);
}
system("@cls||clear");
if(change_count_var!=1){
    puts("Change the ids of variations?");
    key=editing_menu();
}
else{
    key='1';
}
if (key=='1'){
    while(key!='2'){
        free(p1->id_var);
        p1->id_var=NULL;
        p1->id_var = (int*) malloc((p1->variations)*sizeof(int));
```

```
            if (p1->id_var!=NULL){
               key='2';
               printf("Id of each variation.\n");
               for(j=0; j < p1->variations; j++){
                  printf("Variation-[%d]: ", (j+1));
                  checking_input_int(string);
                  p1->id_var[i]=atoi(string);

               }
            }
         }
      }
      print_array(head);
   }
   else{
      puts("There is no such line in the list");
      printf("To continue, click enter");
      getchar();
   }
}
/* ---------------------------------------- */
void checking_input_int(char string[inf])
{
   int i,prov;
   fgets(string, inf, stdin);
   string[strlen(string)-1] = '\0';
   prov=0;
   while(prov==0){
      system("@cls||clear");
      for(i=0;i<strlen(string);i+=1){
         if(isdigit(string[i])){
            prov+=1;
         }
      }
      if(prov!=strlen(string)){
         prov=0;
         system("@cls||clear");
         printf("Please enter the desired value in numbers.\n");
         printf("Input: ");
         fgets(string, inf, stdin);
         string[strlen(string)-1] = '\0';
      }
   }
}
```

```
/* ---------------------------------------- */
void checking_input_float(char string[inf])
{
    int i,prov;
    fgets(string, inf, stdin);
    string[strlen(string)-1] = '\0';
    prov=0;
    while(prov==0){
        system("@cls||clear");
        for(i=0;i<strlen(string);i+=1){
            if(isdigit(string[i])){
                prov+=1;
            }
        }
        if(strchr(string,'.')!=NULL) i=strlen(string)-1;
        else i=strlen(string);

        if(prov!=i){
            prov=0;
            system("@cls||clear");
            printf("Please enter the desired value in numbers.\n");
            printf("Input: ");
            fgets(string, inf, stdin);
            string[strlen(string)-1] = '\0';
        }
    }
}
/* ---------------------------------------- */
void make_csv(lh* head)
{
    cr* p;
    int i;
    FILE *df;

    df = fopen("kolenko_course.txt", "w");
    if (df)
    {
        p = head->first;
        while(p!=NULL)
        {
            fprintf(df, "%s;%s;%i;%i;%f;%f;",p->name, p->producer->name, p->toys_count,
                    p->variations, p->sales_month, p->price);
            for(i=0; i < p->variations; i++)
            {
```

```
                fprintf(df, "%d", p->id_var[i]);
                if (i != (p->variations - 1)) fprintf(df, ";");
                else if(p->next != NULL) fprintf(df, "\n");
            }
            p = p->next;
        }
        fclose(df);
    }else puts("Can't open file!");
}
/* ---------------------------------------- */
void search_str(lh* head, char* (*Func) (cr* node)){
    cr *p1=NULL,*node=NULL;
    int i,j;
    char string[inf];
    char str[inf],pstr[inf];
    puts("Enter a name to search for");
    printf("Input: ");
    fgets(string,inf,stdin);
    string[strlen(string)-1]='\0';
    strcpy(pstr,string);
    for (i=0;i<strlen(pstr);i+=1){
        pstr[i]=tolower(pstr[i]);
    }
    system("@cls||clear");
    j=0;
    p1=head->first;
    while(p1!=NULL){
        node=p1;
        strcpy(str,Func(node));
        for (i=0;i<strlen(str);i+=1){
            str[i]=tolower(str[i]);
        }
        if(strstr(str,pstr)!=NULL) j++;
        p1 = p1->next;

    }
    if (j!=0){
        p1=head->first;
        printf("+----+-------------------------+-------------------------+-----------------+----------+----------------------+----------------------+-------------------------------+\n");
        printf("|%4s|%27s|%27s|%10s|%11s|%25s|%25s||%35s |","ID","Toy name","Toy manufacturer","Toy count at stock","Count variations","Average sales per month","Price in USD","ID of variation");
        printf("\n+----+-------------------------+-------------------------+-----------------+----------+----------------------+----------------------+-------------------------------+\n");
        while(p1!=NULL){
```

```
            node=p1;
            strcpy(str,Func(node));
            for (i=0;i<strlen(str);i+=1){
               str[i]=tolower(str[i]);
            }
            if(strstr(str,pstr)!=NULL){
               printf("|%4d|%27s|%27s|%18d|%11d|%25g|%25g||",p1->id,p1->name,p1->producer->name,p1->toys_count,p1->variations, p1->sales_month, p1->price);
               for(i = 0; i < p1->variations; i++)  printf("%5d|",p1->id_var[i]);
               printf("|");
               printf("\n+----+--------------------------+--------------------------+-----------------+-----------+------------------------+------------------------+-------------------------------+\n");
            }
            p1 = p1->next;

        }
    }
    else{
        puts("There is no such line in the list");
    }
    printf("To continue, click enter");
    getchar();
}
/* ---------------------------------------- */
void search_int(lh* head, float (*Func) (cr* node)){
    cr *p1=NULL,*node=NULL;
    int i,j;
    char string[inf];
    int num;
    puts("Choose the number of variations for the toy");
    printf("Input: ");
    checking_input_int(string);
    num=atoi(string);
    system("@cls||clear");
    j=0;
    p1=head->first;
    while(p1!=NULL){
        node=p1;
        if(Func(node) == num) j++;
        p1 = p1->next;

    }
    if (j!=0){
        p1=head->first;
        printf("+----+--------------------------+--------------------------+-----------------+-----------+------------------------+------------------------+-------------------------------+\n");
```

```
        printf("|%4s|%27s|%27s|%10s|%11s|%25s|%25s||%35s |","ID","Toy name","Toy manufacturer","Toy count at stock","Count variations","Average sales per month","Price in USD","ID of variation");

        printf("\n+----+---------------------------+---------------------------+-----------------+-----------+-------------------------+-------------------------+-----------------------------------+\n");

        while(p1!=NULL){

            node=p1;

            if(Func(node)==num){

                printf("|%4d|%27s|%27s|%18d|%11d|%25g|%25g||",p1->id,p1->name,p1->producer->name,p1->toys_count,p1->variations, p1->sales_month, p1->price);

                for(i = 0; i < p1->variations; i++)  printf("%5d|",p1->id_var[i]);

                printf("|");

                printf("\n+----+---------------------------+---------------------------+-----------------+-----------+-------------------------+-------------------------+-----------------------------------+\n");

            }

            p1 = p1->next;



        }

    }

    else{

        puts("There is no such line in the list");

    }

    printf("To continue, click enter");

    getchar();

}

/* ---------------------------------------- */

char search_menu()

{

    char a, b;



    b = ' ';

    while ((b != '1') && (b != '2') && (b != '3') && (b != '0'))

    {

        a = ' ';

        puts("Please, select parameter:");

        puts("Enter 1 - name");

        puts("Enter 2 - manufacturer");

        puts("Enter 3 - count of variations");

        puts("Enter 0 - exit");

        printf("Input: ");

        while (a != '\n')

        {

            a = getc(stdin);

            if (a == '1') b = '1';

            else if (a == '2') b = '2';

            else if (a == '3') b = '3';

            else if (a == '0') b = '0';
```

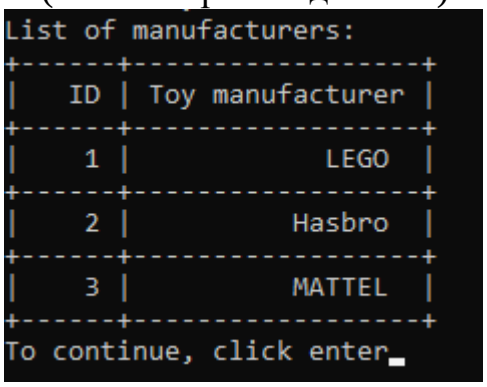
```
        }
        system("@cls||clear");
    }
    return b;
}
/* ---------------------------------------- */
void search_list(lh* head,lh1 *head_l)
{
    char key;
    key = search_menu();
    switch(key){
        case('1'):
            {
                print_array(head);
                search_str(head, NAME);
                break;
            }
        case ('2'):
            {
                print_list(head_l);
                search_str(head, PRODS);
                break;

            }
        case('3'):
            {
                print_array(head);
                search_int(head,VAR);
                break;
            }
    }
}
/* ---------------------------------------- */
int number_elements_manufacturer(lh *head,lh1 *head_l,char str[inf]){
    cr* p1=NULL;
    int count;
    count=0;
    p1=head->first;
    while(p1!=NULL){
      if (strcmp(p1->producer->name,str)==0){
          count+=1;
      }
      p1=p1->next;
    }
```

```
    return count;
}
/* ---------------------------------------- */
void help_output(){
    char key;
    key=help_menu();
    while(key!='0'){
        if (key=='1'){
            printf("The function of adding new items to the list.\nThis function allows you to add a new entry to the end of the list.\n");
            puts("To use it, the user enters the number 1 in the main menu in the input field.");
            puts("Next, you need to enter:\nThe name of the new toy.\nThe name of its manufacturer.\nToy count at stock.\nThe number of variations of toy.\nThe average sale per month of the toy.\nIts price on the market. \nAnd the ids of variations.");
            getchar();
        }
        else if(key=='2'){
            printf("Delete function.\nThe user is given the choice to delete one item\nor all the elements with some manufacturer.");
            puts("In the first variant, an element with a specific identifier is deleted.");
            puts("And if the item is the only one with a particular manufacturer and it is deleted,");
            puts("then the manufacturer will also be removed from the list of manufacturers.");
            puts("In the second case, the user selects a manufacturer with a specific identifier,");
            puts("too, and all elements with this manufacturer in the key itself will be deleted.");
            getchar();
        }
        else if(key=='3'){
            printf("List output function.\n");
            puts("When you select it, two lists will be displayed on the screen.");
            puts("The first list of toys, which contains information about each model.");
            puts("The second is a list of manufacturers, the list in which the existing manufacturers are listed.");
            getchar();
        }
        else if(key=='4'){
            printf("The function of sorting the list by parameters.\n");
            puts("To use it, you need to select the parameter by which the sorting and order will be.");
            puts("After its operation, a list will be displayed on the screen.");
            getchar();
        }
        else if(key=='5'){
            printf("List item editing function.\n");
            puts("For it to work, you need to enter the ID of the element to be edited.");
            puts("Next, the user chooses for each item to edit or not.");
            puts("At the end of the function, a list is displayed.");
            getchar();
        }
        else if(key=='6'){
```

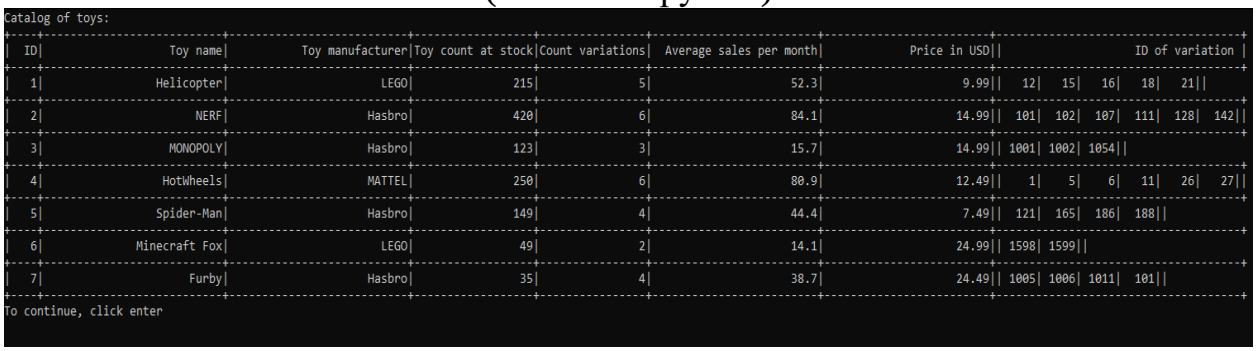
```
            printf("Search function in the list by parameters.\n");
            puts("For this function to work, you need to select the parameter by which the search will be performed.");
            puts("If the name of the toy or manufacturer is selected, you must enter the name of the toy or manufacturer.");
            puts("If the count of variations you need to enter the number of variations with which you want to see the result.");
            getchar();
        }
        key=help_menu();
    }

}
/* ---------------------------------------- */
char help_menu(){
    char a, b;
    b = ' ';
    system("@cls||clear");
    while ((b != '1') && (b != '2') && (b != '3') && (b != '4') && (b != '5') && (b != '6') && (b != '0'))
    {
        a = ' ';
        puts("Please select a page:");
        puts("Enter 1 - first function ");
        puts("Enter 2 - second function");
        puts("Enter 3 - third function");
        puts("Enter 4 - fourth function");
        puts("Enter 5 - fifth function");
        puts("Enter 6 - sixth function");
        puts("Enter 0 - exit");
        printf("Input: ");
        while (a != '\n')
        {
            a = getc(stdin);
            if (a == '1') b = '1';
            else if (a == '2') b = '2';
            else if (a == '3') b = '3';
            else if (a == '4') b = '4';
            else if (a == '5') b = '5';
            else if (a == '6') b = '6';
            else if (a == '0') b = '0';
        }
        system("@cls||clear");
    }
    return b;
}
/* ---------------------------------------- */
```

Примеры выполнения программы:
Данные полученные из файла

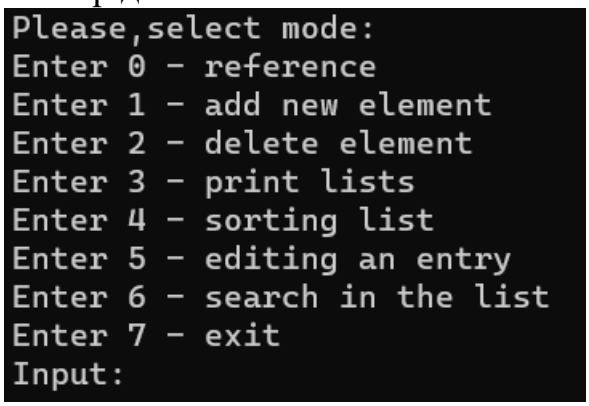
(Список производителей)



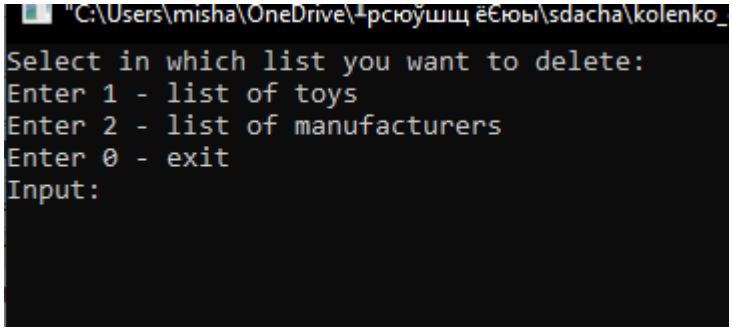
(Список игрушек)



Выбор действия



Выбор: 2 (удаление производителя)



(Удаление из списка игрушек с производителем LEGO)

Catalog of toys:

ID	Toy name	Toy manufacturer	Toy count at stock	Count variations	Average sales per month	Price in USD	ID of variation
1	NERF	Hasbro	420	6	84.1	14.99	101, 102, 107, 111, 128, 142
2	MONOPOLY	Hasbro	123	3	15.7	14.99	1001, 1002, 1054
3	HotWheels	MATTEL	250	6	80.9	12.49	1, 5, 6, 11, 26, 27
4	Spider-Man	Hasbro	149	4	44.4	7.49	121, 165, 186, 188
5	Furby	Hasbro	35	4	38.7	24.49	1005, 1006, 1011, 1

To continue, click enter

(Удаление из списка производителей LEGO)

List of manufacturers:

ID	Toy manufacturer
1	Hasbro
2	MATTEL

To continue, click enter

Заключение

Используемые заголовочные файлы стандартной библиотеки:

1. stdio.h:
    1.1. fopen – для открытия файла.
    1.2. fclose – для закрытия файла.
    1.3. rewind – для сброса курсора в файле.
    1.4. fgets – для считывания строк.
    1.5. getchar – для “паузы” между переключениями окон.
    1.6. getc – для считывания символа.
    1.7. printf – для вывода в консоль информации.
    1.8. fprintf – для записи информации в файл.
    1.9. puts – для вывода информации в консоль.
2. stdlib.h:
    2.1. atof – для получения вещественного числа из строки.
    2.2. atoi – для получение целого числа из строки
    2.3. malloc – для выделения памяти.
    2.4. free – для освобождения памяти.
3. string.h:
    3.1. strcmp – для сравнения строк.
    3.2. strstr – для проверки на вхождение
    3.3. strcpy – для копирования строк.
    3.4. strlen – для получения длины строки.
4. ctype.h
    4.1. tolower – для перевода в нижний регистр

В результате выполнения работы изучена работа со структурами в языке Си и получены практические навыки в создании электронных картотек. Цель работы считается достигнутой.