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

ОТЧЕТ

по лабораторной работе № 10

по дисциплине «Программирование»

Тема: Кольцевые списки

Студент гр. 3312 _____ Мохно Даниил.

Преподаватель _____ Аббас Саддам

Санкт-Петербург

2024

Цель работы.

Целью работы является изучение односвязных кольцевых списков в языке Си и получение практических навыков в программировании на этом языке.

Задание (вариант 5)

Разработать подалгоритм добавления элемента в односвязный кольцевой список после заданного по номеру элемента и в «конец» списка, если такого элемента нет.

Постановка задачи и описание решения

Для начала объявим связные списки. Структура Brand содержит в себе поля name – название брэнда и next – указатель на следующий элемент списка брэндов. Структура Smartphone содержит 10 полей. Символьное поле: model – модель смартфона, Указатель на элемент списка с брэндами brand – марка смартфона. Три целочисленных поля: ram – объём оперативной памяти, memory – объём постоянной памяти и index – индекс. Три поля с плавающей точкой: screen_size – размер экрана, weight – вес, price – цена. Указатель на такую же структуру next – следующий элемент списка. Затем объявим структуру для головы двусвязного списка. Структура головы двусвязного списка Storage содержит целочисленное поле size хранящее кол-во элементов в списке и указатели на первый и последний элементы списка first_pos и last_pos. Для удобства выделим структуры в новые типы данных.

В главной функции объявляем переменную Market – указатель на Storage, она будет являться головой односвязного списка и указатель brands – указатель на первый элемент связного списка брэндов. Затем запрашиваем у пользователя имя файла, выделяем память в Market, и присваиваем значение размера списка 0 и значение наибольшего элемента 0. Далее вызываем функцию заполнения списка из файла, куда передаём Market, имя файла и brands. Теперь проверяем чтобы размер списка был больше 0 и вызываем функцию вставки элемента по

выбору индекса которой передаём Market и указатель на brands. В конце освобождаем память из-под списков Market и brands.

В функции заполнения списка мы открываем файл, запускаем цикл, пока из файла читаются строки, в цикле создаём новую позицию, передав в функцию создания позиции прочитанную строку, индекс, и указатель на указатель на первый элемент списка брэндов. Если это первая итерация, мы устанавливаем указатели головы списка на новый элемент, а размер и максимальный индекс ставим 1, в остальных случаях, вызываем функцию добавления элемента в список после текущего, куда передаём указатель на голову списка, указатель на последний элемент списка и новый элемент.

В функции создания позиции мы выделяем место под новый элемент, устанавливаем переданный индекс в поле индекс, а поле next в NULL. Далее присваиваем значения из строки в поля структуры и строку с названием брэнда путём форматированного сканирования строки функцией sscanf. Далее вызываем функцию присвоения боля брэнда и добавления нового узла в список брэндов которой передаём созданный элемент, строку с названием брэнда, и указатель на указатель на первый элемент списка брэндов.

В функции присвоения боля брэнда и добавления нового узла в список брэндов мы присваиваем в указатель на текущий бренд, указатель на первый узел из списка брэндов. Затем пока мы не пройдём до конца списка, мы проверяем совпадает ли строка из бренда с переданной строкой, и, если совпадает мы присваиваем указателю на бренд в структуре, адрес текущего указателя на бренд. За тем присваиваем указателю на предыдущий бренд текущий бренд, а текущий в текущий бренд присваиваем следующий бренд. Далее, если мы не нашли на предыдущем шаге нужной структуры, значит ещё надо добавить. Проверяем чтобы поле брэнд в структуре было NULL, выделяем паять в текущий узел бренда, в поле name этого узла записываем переданную строку. Проверяем существовал ли предыдущий узел бренда, если да, то

присваиваем в поле next предыдущего бренда текущий элемент, если нет, то это первый элемент и мы присваиваем значением переданному указателю на указатель текущее значение бренда. Затем присваиваем в поле бренд структуры указатель на текущий элемент. В случае, если нужный узел с брендом у нас уже был, мы просто освобождаем память из-под строки.

В функции добавления элемента после текущего мы увеличиваем размер списка, а затем проверяем равен ли узел последнему элементу. Если да, то в поле next текущего элемента мы записываем новый элемент. В последний элемент мы записываем новый элемент, а в поле next нового элемента мы записываем адрес первого элемента списка. Если текущий элемент не последний, то в next нового элемента записываем next текущего элемента. А в поле next текущего элемента записываем новый элемент.

В функции выбора индекса элемента для вставки элемента мы запускаем цикл с пост условием. В нём, если размер списка больше 0, то мы вызываем функцию вывода данных в таблицу которой передаём указатель на голову списка. Далее мы считываем индекс, по которому вставится новый элемент. Затем если индекс, который ввёл пользователь больше 0, считываем строку с данными, разделёнными разделителями и вызываем функцию вставки по индексу, в которую передаём указатель на голову списка, строку, полученное значение индекса и указатель на указатель на первый элемент списка брендов. Цикл завершается, когда введённое пользователем значение будет 0.

В функции вывода таблицы мы присваиваем указателю на текущий элемент адрес первого элемента. А затем запускаем цикл, в котором выводим ткущий элемент, а затем присваиваем текущему элементу следующий элемент из его поля next.

В функции вставки элемента по переданному индексу, мы объявляем флаг найденного элемента равным 1. Создадим новый элемент, передав в функцию создания нового элемента полученную строку, индекс и указатель на первый

элемент списка брендов. Затем, если переданный индекс больше или равен размеру списка, то в текущий элемент помещаем указатель на последний элемент списка и вызываем функцию добавления элемента после текущего, а размер инкрементируем. В ином случае мы присваиваем в текущий элемент первый элемент списка, и запускаем цикл с пост условием, пока индекс текущего элемента меньше, чем размер списка. В цикле в текущий элемент присваиваем следующий, если индекс текущего элемента равен искомому, мы вызываем функцию вставки элемента после текущего, индекс текущего элемента инкрементируем, в текущий элемент присваиваем его поле next, то есть следующий элемент и флаг устанавливаем 1. В ином случае, если флаг равен 0, инкрементируем индекс текущего элемента.

Описание переменных

Функция – int main():

№	Имя переменной	Тип	Назначение
1	Market	struct Storage	Указатель на голову связного списка
2	brands	struct Brands	Указатель на первый элемент списка с названиями брэндов
3	len	int	Длинна названия файла
4	filename	char	Название файла

Функция создания головы списка – Storage *create_storage():

№	Имя переменной	Тип	Назначение
1	storage	struct Storage	Указатель на голову связного списка

Функция заполнения списка из файла – void fill_storage(Storage *storage, char *filename, Brands **brands):

№	Имя переменной	Тип	Назначение
1	storage	struct Storage	Указатель на голову связного списка
2	filename	char	Название файла
3	source	FILE	Указатель на файл
4	new_pos	struct Smartphone	Указатель на новый элемент списка
5	tmp_str	char	Строка из файла
6	i	int	Итератор
7	brands	struct Brands	Указатель на указатель на первый элемент списка

Функция создания узла списка – Smartphone create_position(Storage *storage, char *filename, Brands **Brands):

№	Имя переменной	Тип	Назначение
1	position	struct Smartphone	Указатель на элемент списка
2	string	char	Строка из файла
3	index	int	Порядковый номер в списке
4	brands	struct Brands	Указатель на указатель на первый элемент списка

Функция присвоения значений полям структуры – void set_values(Smartphone *smartphone, char *str, Brands **brands)

№	Имя переменной	Тип	Назначение
1	smartphone	struct Smartphone	Указатель на структуру
2	str	char	Строка из файла
3	brands	struct Brands	Указатель на указатель на первый элемент списка
4	brand_name	char	Подстрока с названием брэнда

Функция присвоения боля брэнда и добавления нового узла в список брэндов – void set_brand(Smartphone *smartphone, char *brand, Brands **brands)

№	Имя переменной	Тип	Назначение
1	smartphone	struct Smartphone	Указатель на структуру
2	str	char	Строка из файла
3	brands	struct Brands	Указатель на указатель на первый элемент списка
4	brand	char	Строка с названием брэнда
5	cur_brand	struct Brands	Указатель на текущий брэнд из списка брэндов
6	prev_brand	struct Brands	Указатель на предыдущий брэнд из списка брэндов

Функция добавления первого элемента в список – void add_first(storage, new_position)

№	Имя переменной	Тип	Назначение
1	storage	struct Storage	Указатель на голову связного списка
2	new_position	struct Smartphone	Указатель на новый элемент списка

Функция добавления элемента после текущего элемента в список – void add(Storage *storage, Smartphone *cur_position, Smartphone *new_position)

№	Имя переменной	Тип	Назначение
1	storage	struct Storage	Указатель на голову связного списка
2	cur_position	struct Smartphone	Указатель на текущий элемент списка
3	new_position	struct Smartphone	Указатель на новый элемент списка

Функция выбора индекса вставляемого элемента– void choose_to_insert(storage)

№	Имя переменной	Тип	Назначение
1	storage	struct Storage	Указатель на голову связного списка
2	val	int	Индекс элемента, выбранного пользователем
3	string	char	Строка, полученная от пользователя
4	brands	struct Brands	Указатель на указатель на первый элемент списка

Функция вывода таблицы – void print_table(Storage *storage)

№	Имя переменной	Тип	Назначение
1	storage	struct Storage	Указатель на голову связного списка
2	cur	struct Smartphone	Указатель на текущий элемент связного списка

Функция вывода полей структуры – void print(Smartphone *smartphone)

№	Имя переменной	Тип	Назначение
1	smartphone	struct Smartphone	Указатель элемент связного списка

Функция вставки элемента в список – void insert_selected(Storage *storage, char *string, int index)

№	Имя переменной	Тип	Назначение
1	storage	struct Storage	Указатель на голову связного списка
2	index	int	Индекс элемента, выбранного пользователем
3	string	char	Строка, полученная от пользователя
4	new	struct Smartphone	Вставляемый элемент
5	cur	struct Smartphone	Текущий элемент
6	brands	struct Brands	Указатель на указатель на первый элемент списка

Функция освобождения памяти узла списка – void delete_position(storage)

№	Имя переменной	Тип	Назначение
1	posititon	struct Smartphone	Указатель на узел связного списка

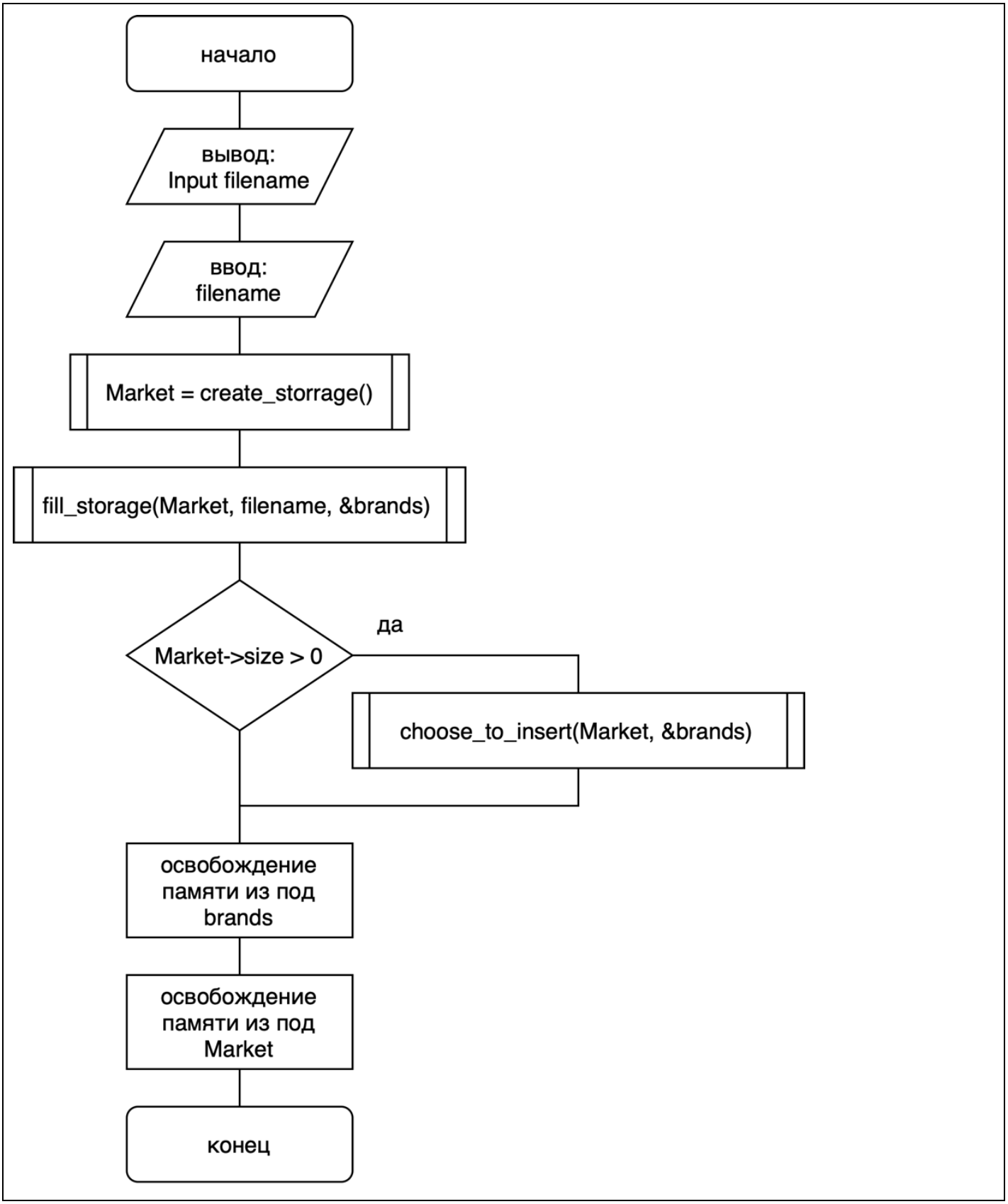
Функция удаления списка брэндов– void delete_brands(Brands *brand)

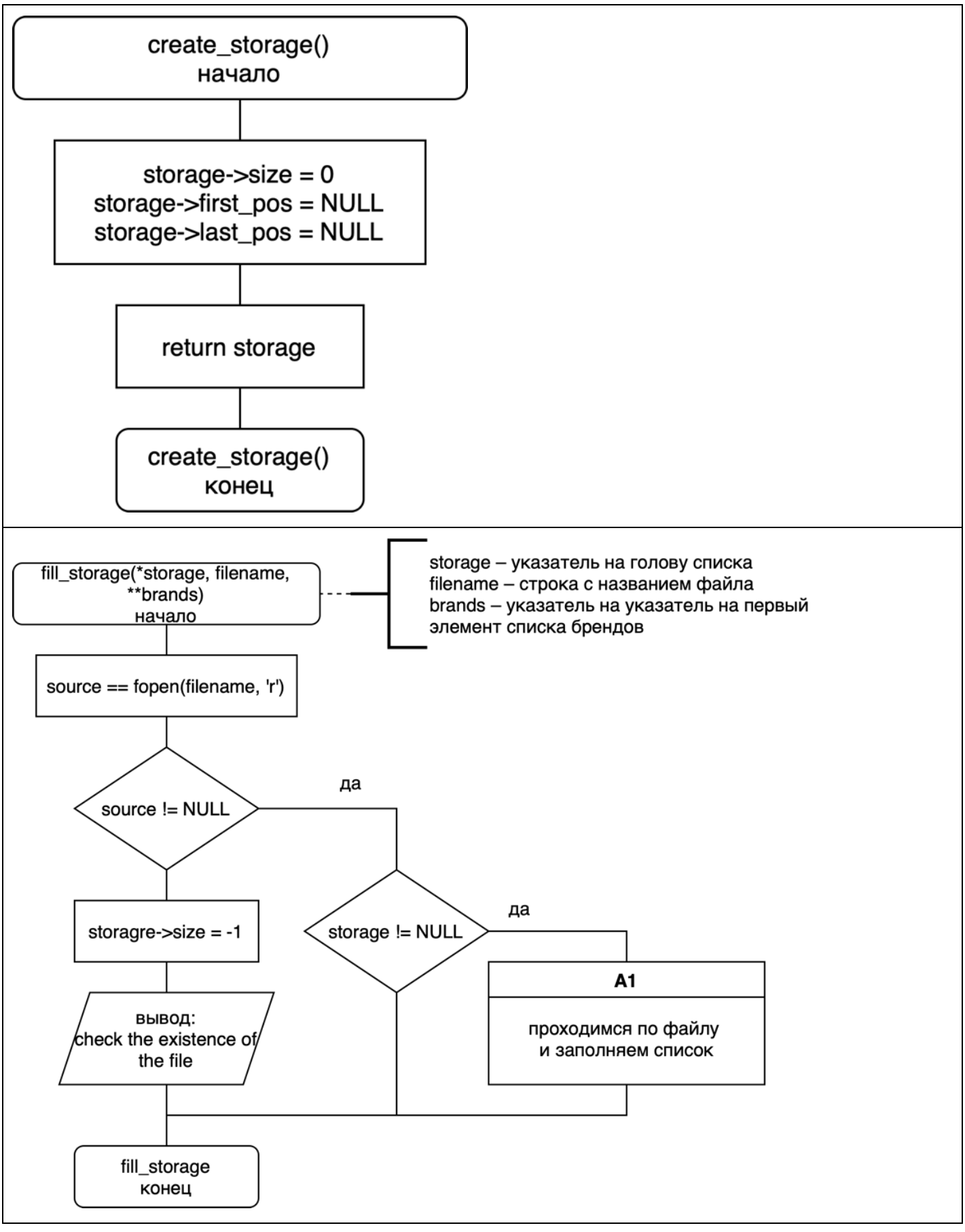
№	Имя переменной	Тип	Назначение
1	brands	struct Brands	Указатель на элемент списка с названиями брэндов
2	tmp_brand	struct Brands	Дополнительный указатель на элемент списка с названиями брэндов

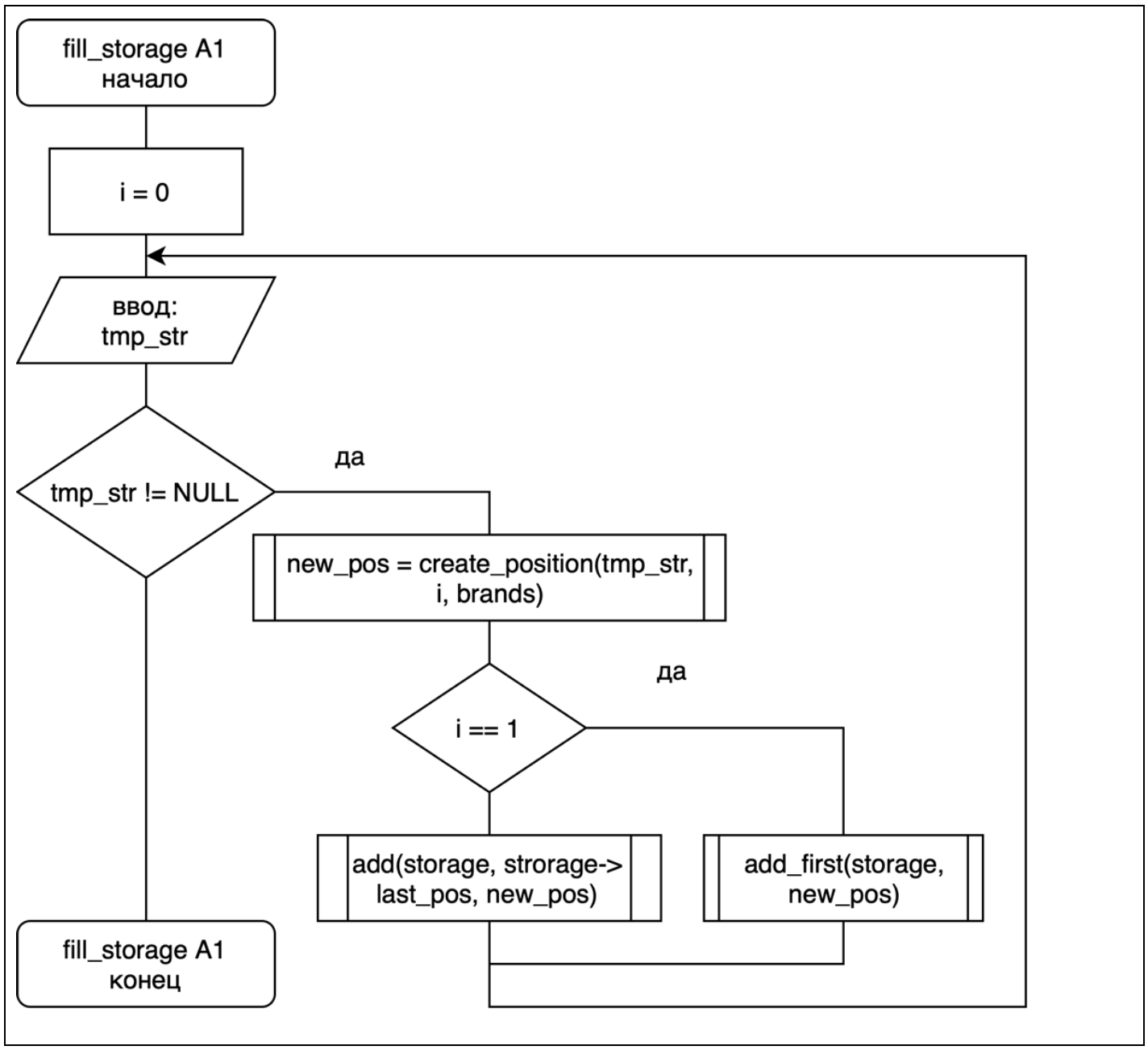
Функция удаления списка– void delete_storage(storage)

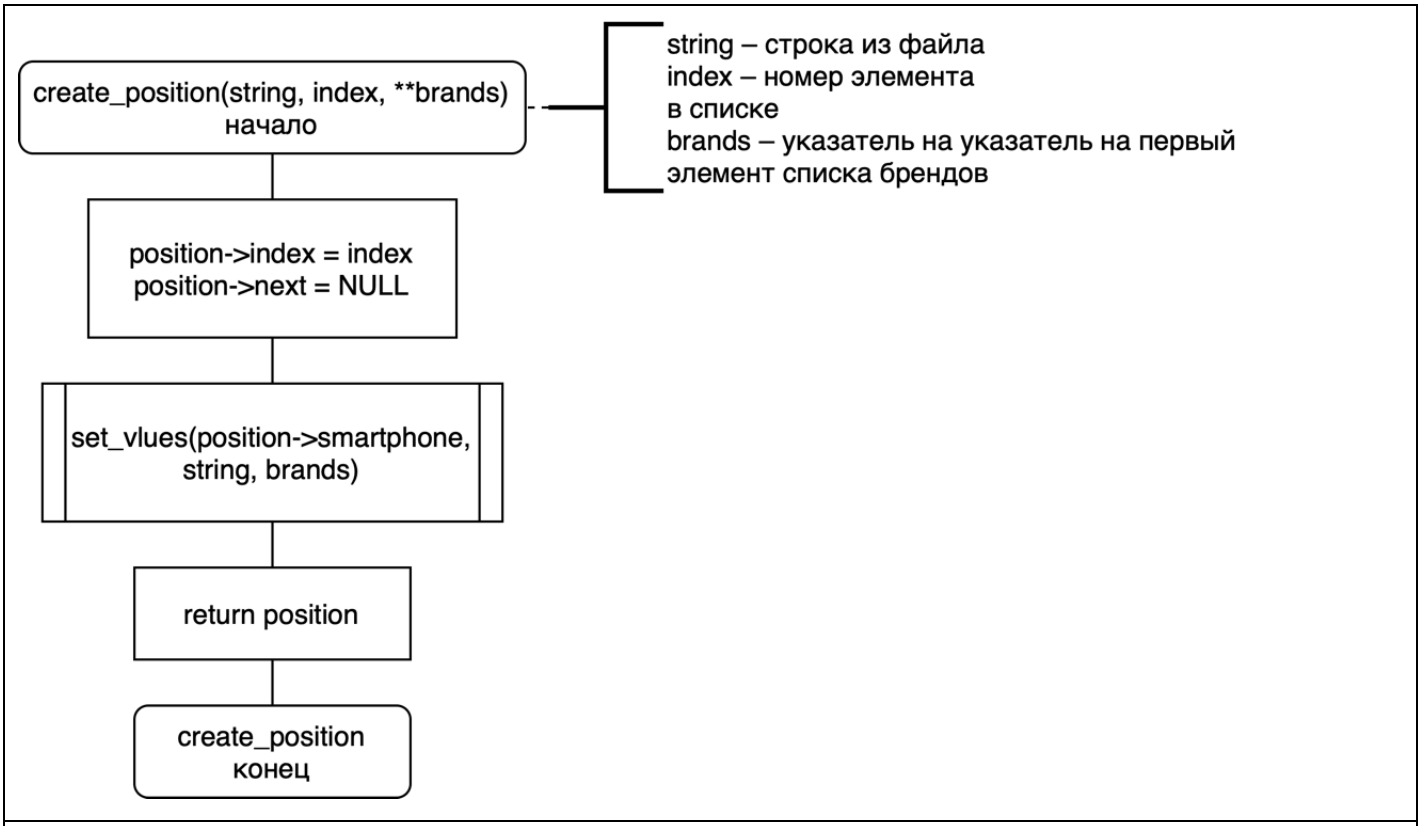
№	Имя переменной	Тип	Назначение
1	storage	struct Storage	Указатель на голову связного списка
2	cur	struct Smartphone	Указатель на текущий элемент списка, который нужно удалить.
3	next	struct Smartphone	Указатель на следующий элемент, который нужно, не потерять при удалении текущего элемента.

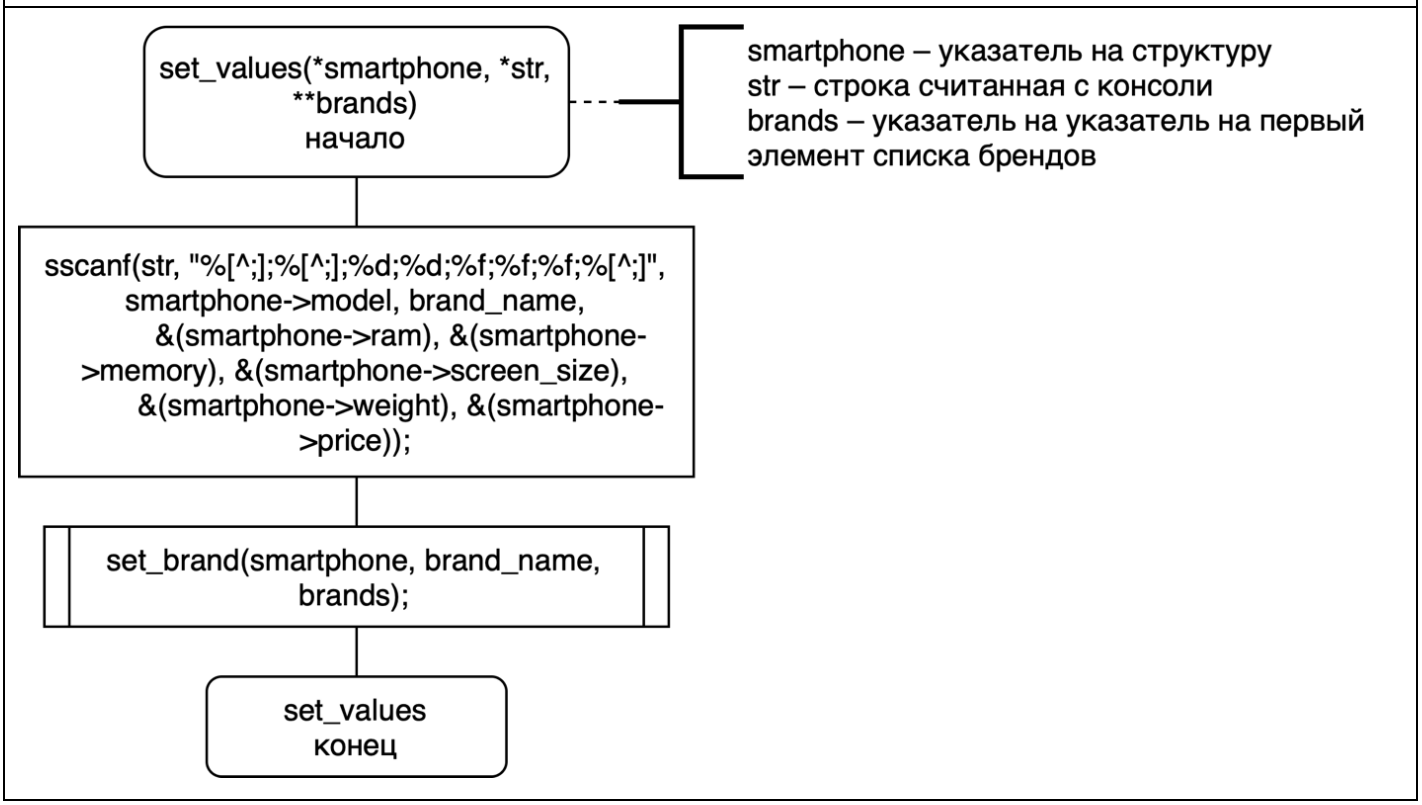
Схема алгоритма

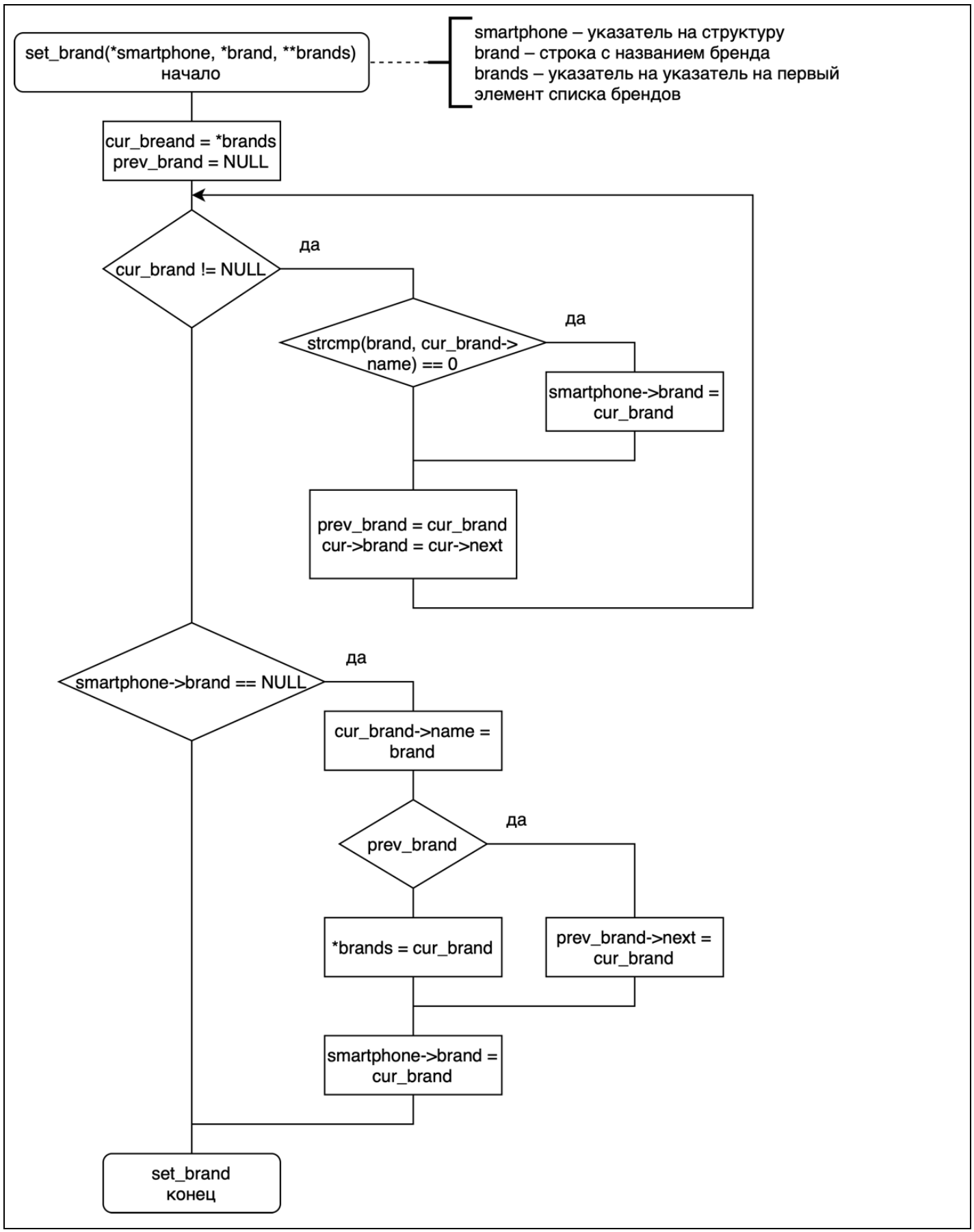


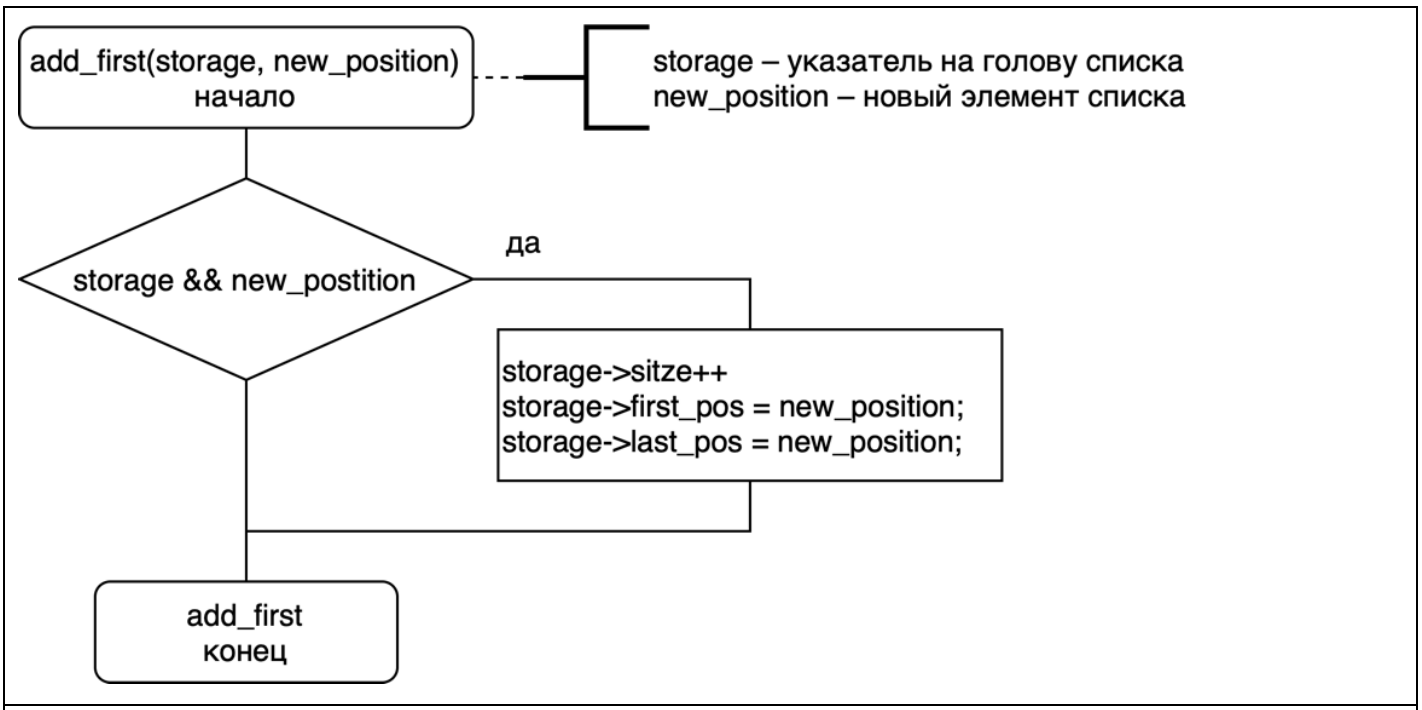

create_storage()
начало
storage->size = 0
storage->first_pos = NULL
storage->last_pos = NULL
return storage
create_storage()
конец
fill_storage(*storage, filename, **brands)
начало
storage – указатель на голову списка
filename – строка с названием файла
brands – указатель на указатель на первый элемент списка брендов
source == fopen(filename, 'r')
source != NULL
да
storagre->size = -1
вывод:
check the existence of the file
storage != NULL
да
A1
проходимся по файлу и заполняем список
fill_storage
конец

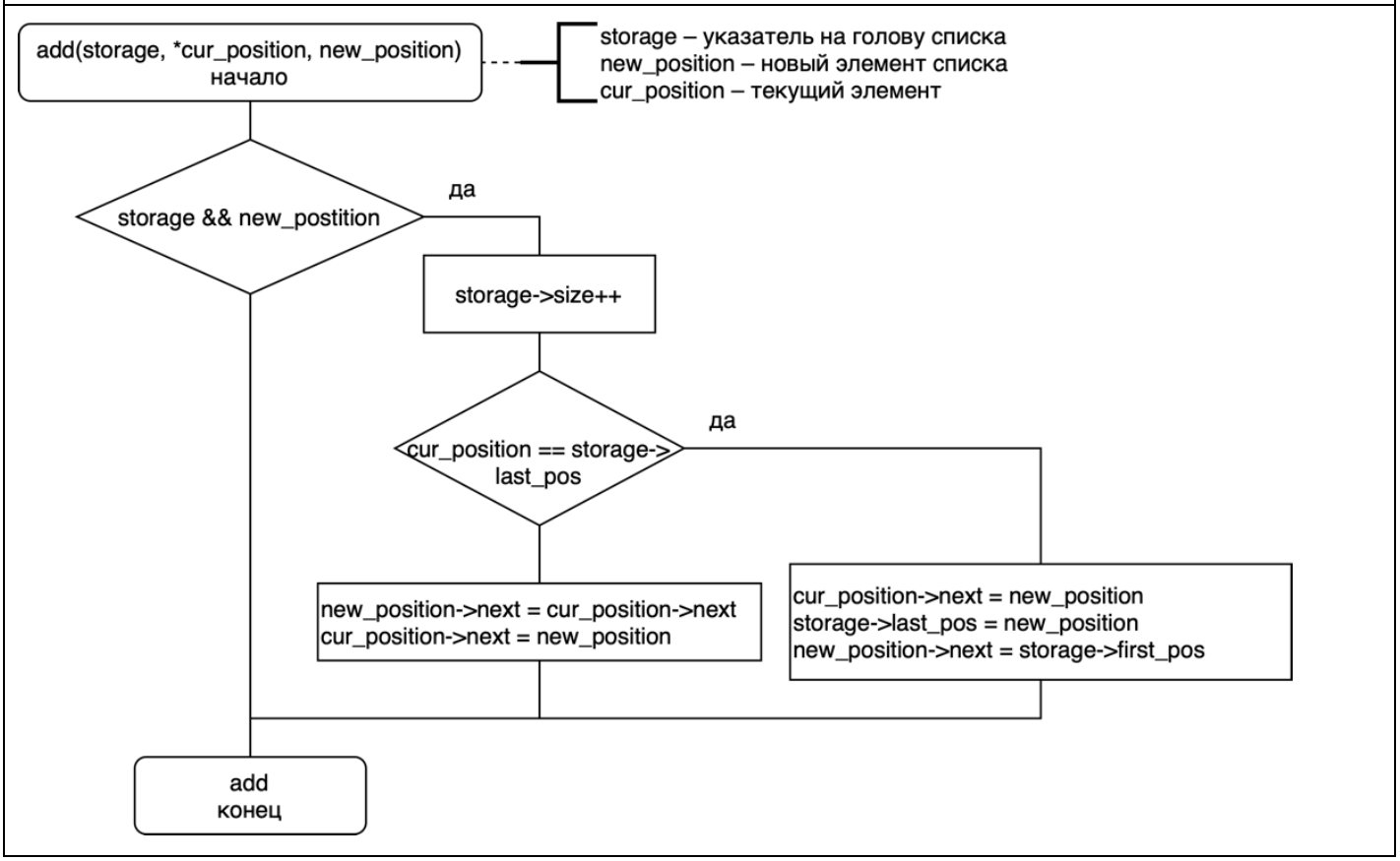

fill_storage A1
начало
i = 0
ввод:
tmp_str
tmp_str != NULL
да
new_pos = create_position(tmp_str,
i, brands)
да
i == 1
add(storage, strorage->
last_pos, new_pos)
add_first(storage,
new_pos)
fill_storage A1
конец

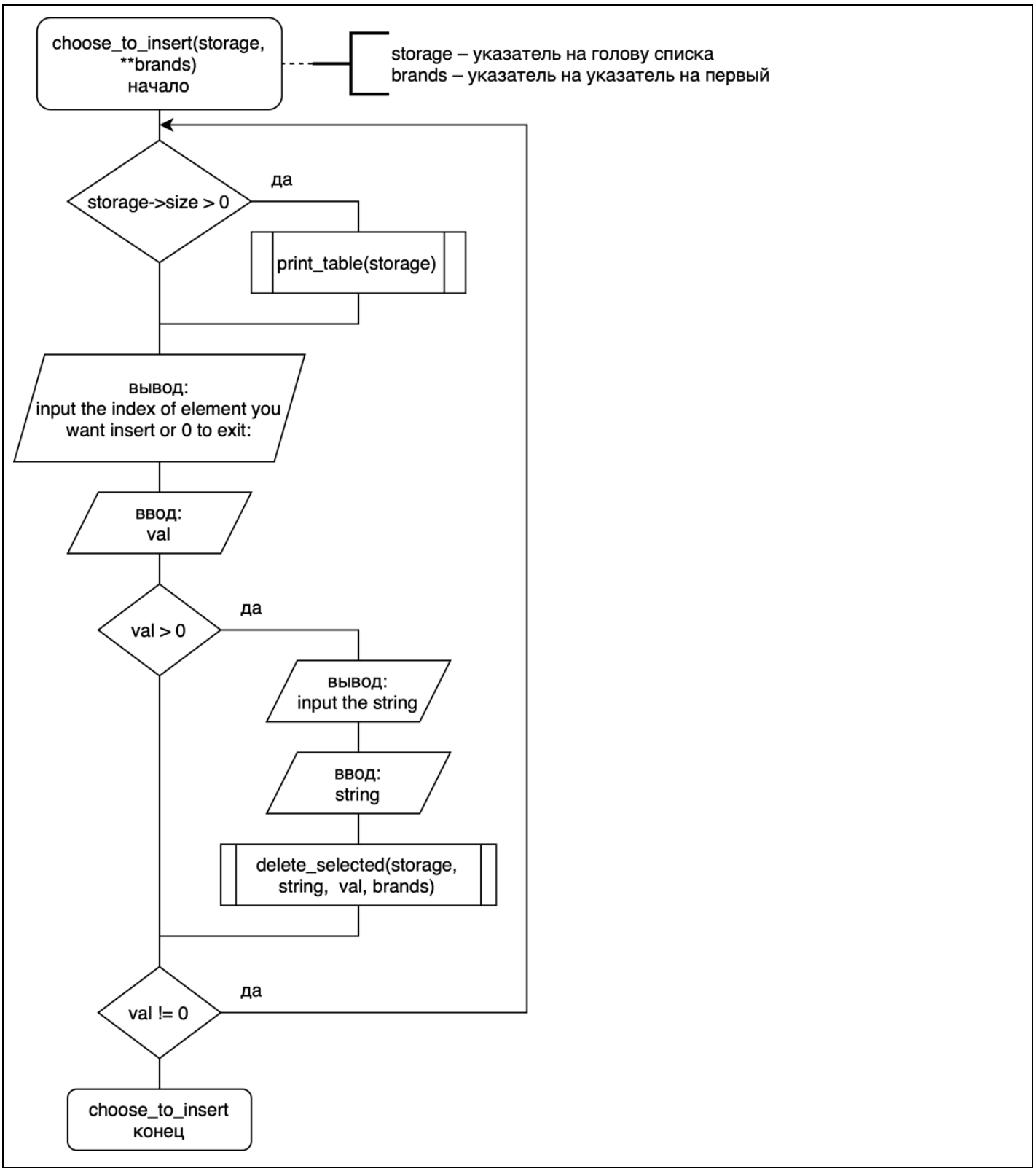

create_position(string, index, **brands)
начало
string – строка из файла
index – номер элемента
в списке
brands – указатель на указатель на первый
элемент списка брендов
position->index = index
position->next = NULL
set_vlues(position->smartphone,
string, brands)
return position
create_position
конец

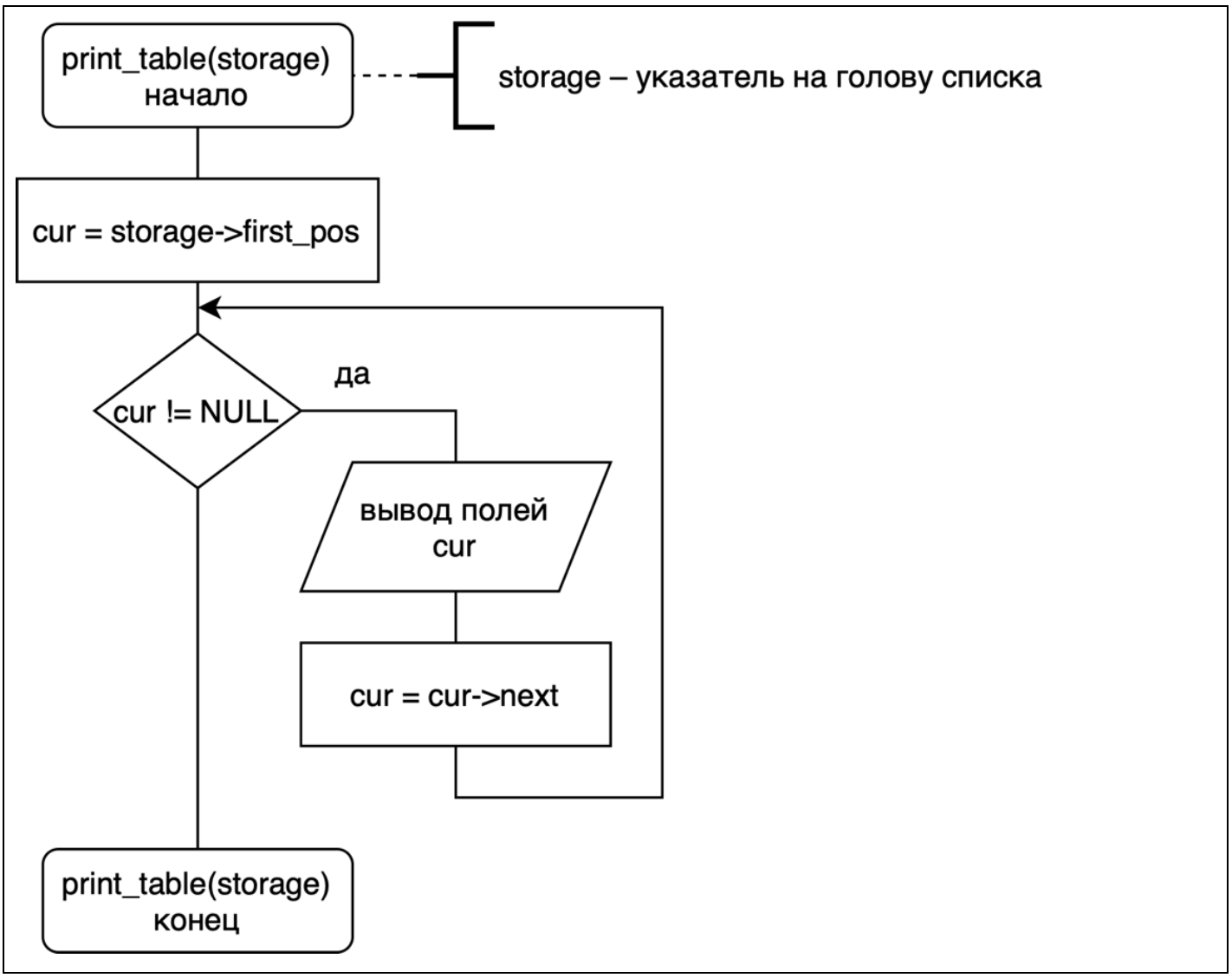

set_values(*smartphone, *str,
**brands)
начало
smartphone – указатель на структуру
str – строка считанная с консоли
brands – указатель на указатель на первый
элемент списка брендов
sscanf(str, "%[^;];%[^;];%d;%d;%f;%f;%f;%[^;]",
smartphone->model, brand_name,
&(smartphone->ram), &(smartphone-
>memory), &(smartphone->screen_size),
&(smartphone->weight), &(smartphone-
>price));
set_brand(smartphone, brand_name,
brands);
set_values
конец

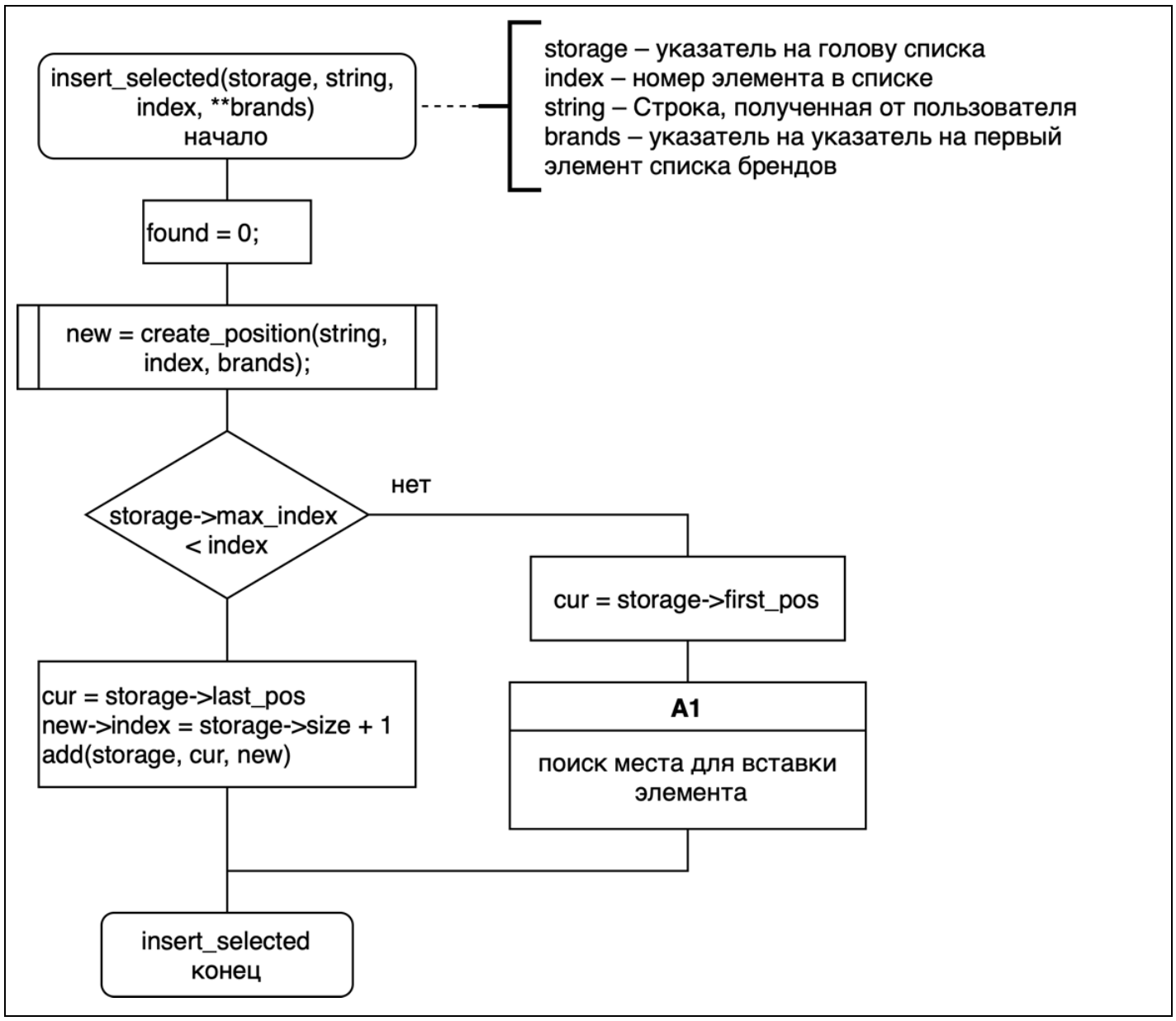

set_brand(*smartphone, *brand, **brands)
начало
smartphone – указатель на структуру
brand – строка с названием бренда
brands – указатель на указатель на первый элемент списка брендов
cur_breand = *brands
prev_brand = NULL
cur_brand != NULL
да
strcmp(brand, cur_brand->name) == 0
да
smartphone->brand = cur_brand
prev_brand = cur_brand
cur->brand = cur->next
smartphone->brand == NULL
да
cur_brand->name = brand
prev_brand
да
*brands = cur_brand
prev_brand->next = cur_brand
smartphone->brand = cur_brand
set_brand
конец

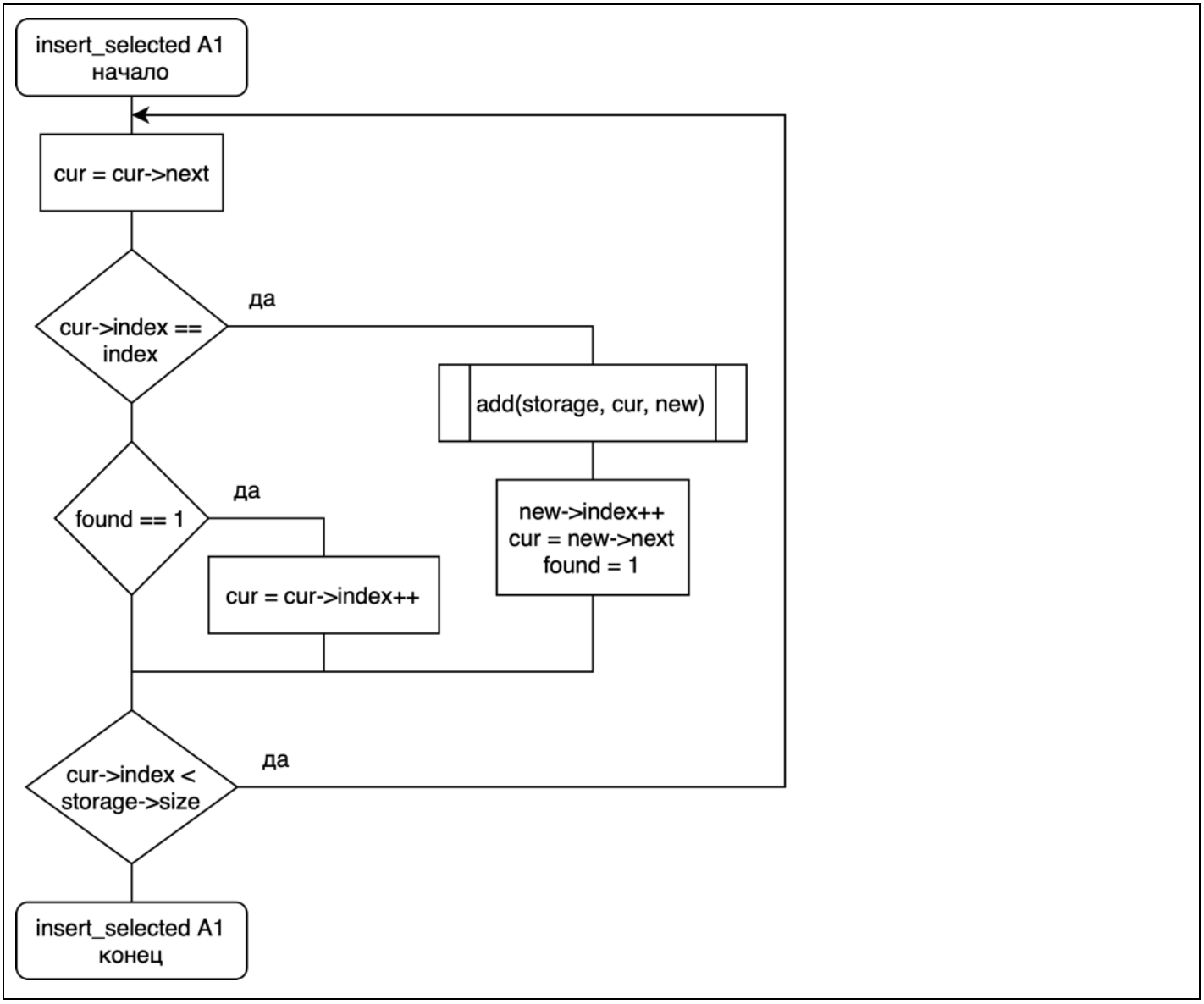

add_first(storage, new_position)
начало
storage – указатель на голову списка
new_position – новый элемент списка
storage && new_postition
да
storage->sitze++
storage->first_pos = new_position;
storage->last_pos = new_position;
add_first
конец


add(storage, *cur_position, new_position)
начало
storage – указатель на голову списка
new_position – новый элемент списка
cur_position – текущий элемент
storage && new_postition
да
storage->size++
cur_position == storage->
last_pos
да
new_position->next = cur_position->next
cur_position->next = new_position
cur_position->next = new_position
storage->last_pos = new_position
new_position->next = storage->first_pos
add
конец


choose_to_insert(storage, **brands)
начало
storage – указатель на голову списка
brands – указатель на указатель на первый
storage->size > 0
да
print_table(storage)
вывод:
input the index of element you want insert or 0 to exit:
ввод:
val
val > 0
да
вывод:
input the string
ввод:
string
delete_selected(storage, string, val, brands)
val != 0
да
choose_to_insert
конец


print_table(storage)
начало
storage – указатель на голову списка
cur = storage->first_pos
cur != NULL
да
вывод полей
cur
cur = cur->next
print_table(storage)
конец


insert_selected(storage, string, index, **brands)
начало
storage – указатель на голову списка
index – номер элемента в списке
string – Строка, полученная от пользователя
brands – указатель на указатель на первый элемент списка брендов
found = 0;
new = create_position(string, index, brands);
storage->max_index < index
нет
cur = storage->first_pos
A1
поиск места для вставки элемента
cur = storage->last_pos
new->index = storage->size + 1
add(storage, cur, new)
insert_selected
конец



Контрольные примеры

Пример 1:

Содержимое файла Data.csv
iPhone11,Apple,4,128,6.06,194,599
iPhone12,Apple,4,256,6.1,215,799
Galaxy S20,Samsung,6,128,6.2,163,699
OnePlus 8,OnePlus,8,256,6.55,180,699
Pixel 5,Google,8,128,6.0,151,699
Xperia 1 II,Sony,8,256,6.5,181,1099

Входные данные

```
Data.csv
2
iPhone SE 2020,Apple,3,64,4.7,148,399
0
```

Выходные данные

```
+-----+----------------------+----------------+-------+---------+--------+---------+----------+
| №   | Model                | Brand          | RAM   | Storage | Screen | Weight  | Price    |
+-----+----------------------+----------------+-------+---------+--------+---------+----------+
|   1 | iPhone11             | Apple          | 4  GB | 128  GB | 6.06 " | 194.00g | $599.00  |
|   2 | iPhone12             | Apple          | 4  GB | 256  GB | 6.10 " | 215.00g | $799.00  |
|   3 | Galaxy S20           | Samsung        | 6  GB | 128  GB | 6.20 " | 163.00g | $699.00  |
|   4 | OnePlus 8            | OnePlus        | 8  GB | 256  GB | 6.55 " | 180.00g | $699.00  |
|   5 | Pixel 5              | Google         | 8  GB | 128  GB | 6.00 " | 151.00g | $699.00  |
|   6 | Xperia 1 II          | Sony           | 8  GB | 256  GB | 6.50 " | 181.00g | $1099.00 |
+-----+----------------------+----------------+-------+---------+--------+---------+----------+

+-----+----------------------+----------------+-------+---------+--------+---------+----------+
| №   | Model                | Brand          | RAM   | Storage | Screen | Weight  | Price    |
+-----+----------------------+----------------+-------+---------+--------+---------+----------+
|   1 | iPhone11             | Apple          | 4  GB | 128  GB | 6.06 " | 194.00g | $599.00  |
|   2 | iPhone12             | Apple          | 4  GB | 256  GB | 6.10 " | 215.00g | $799.00  |
|   3 | iPhone SE 2020       | Apple          | 3  GB | 64   GB | 4.70 " | 148.00g | $399.00  |
|   3 | Galaxy S20           | Samsung        | 6  GB | 128  GB | 6.20 " | 163.00g | $699.00  |
|   5 | OnePlus 8            | OnePlus        | 8  GB | 256  GB | 6.55 " | 180.00g | $699.00  |
|   6 | Pixel 5              | Google         | 8  GB | 128  GB | 6.00 " | 151.00g | $699.00  |
|   7 | Xperia 1 II          | Sony           | 8  GB | 256  GB | 6.50 " | 181.00g | $1099.00 |
+-----+----------------------+----------------+-------+---------+--------+---------+----------+

size: 7
```

Пример 2:

Содержимое файла Data2.csv

```
Galaxy S21,Samsung,8,128,6.2,171,799
iPhone SE 2020,Apple,3,64,4.7,148,399
OnePlus 9,OnePlus,12,256,6.55,183,899
```

Входные данные

```
Data2.csv
45
OnePlus 8,OnePlus,8,256,6.55,180,699
1
iPhone12,Apple,4,256,6.1,215,799
0
```

Выходные данные

```
+-----+---------------------+----------------+-------+---------+--------+---------+----------+
| №   | Model               | Brand          | RAM   | Storage | Screen | Weight  | Price    |
+-----+---------------------+----------------+-------+---------+--------+---------+----------+
|   1 | Galaxy S21          | Samsung        | 8  GB | 128  GB | 6.20 " | 171.00g | $799.00  |
|   2 | iPhone SE 2020      | Apple          | 3  GB | 64   GB | 4.70 " | 148.00g | $399.00  |
|   3 | OnePlus 9           | OnePlus        | 12 GB | 256  GB | 6.55 " | 183.00g | $899.00  |
+-----+---------------------+----------------+-------+---------+--------+---------+----------+

+-----+---------------------+----------------+-------+---------+--------+---------+----------+
| №   | Model               | Brand          | RAM   | Storage | Screen | Weight  | Price    |
+-----+---------------------+----------------+-------+---------+--------+---------+----------+
|   1 | Galaxy S21          | Samsung        | 8  GB | 128  GB | 6.20 " | 171.00g | $799.00  |
|   2 | iPhone SE 2020      | Apple          | 3  GB | 64   GB | 4.70 " | 148.00g | $399.00  |
|   3 | OnePlus 9           | OnePlus        | 12 GB | 256  GB | 6.55 " | 183.00g | $899.00  |
|   4 | OnePlus 8           | OnePlus        | 8  GB | 256  GB | 6.55 " | 180.00g | $699.00  |
+-----+---------------------+----------------+-------+---------+--------+---------+----------+

+-----+---------------------+----------------+-------+---------+--------+---------+----------+
| №   | Model               | Brand          | RAM   | Storage | Screen | Weight  | Price    |
+-----+---------------------+----------------+-------+---------+--------+---------+----------+
|   1 | Galaxy S21          | Samsung        | 8  GB | 128  GB | 6.20 " | 171.00g | $799.00  |
|   2 | iPhone12            | Apple          | 4  GB | 256  GB | 6.10 " | 215.00g | $799.00  |
|   2 | iPhone SE 2020      | Apple          | 3  GB | 64   GB | 4.70 " | 148.00g | $399.00  |
|   4 | OnePlus 9           | OnePlus        | 12 GB | 256  GB | 6.55 " | 183.00g | $899.00  |
|   5 | OnePlus 8           | OnePlus        | 8  GB | 256  GB | 6.55 " | 180.00g | $699.00  |
+-----+---------------------+----------------+-------+---------+--------+---------+----------+

size: 7
```

Текст программы

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#ifdef WIN32
#define CLS system("cls")
#else
#define CLS system("clear")
#endif

enum
{
    MAX_STR_IN_FILE_LEN = 200,
    MAX_MODEL_NAME_LEN = 30,
    MAX_FILENAME_LEN = 20
};

typedef struct Brand
{
    char *name;
    struct Brand *next;
} Brands;

typedef struct Smartphone
```

```
{
    char *model;              /* Model */
    Brands *brand;            /* Brand */
    int ram;                  /* RAM size, GB */
    int memory;               /* Storage capacity, GB */
    float screen_size;        /* Screen size, inches */
    float weight;             /* Weight, grams */
    float price;              /* Price, dollars */

    int index;                /* Index */
    struct Smartphone *next; /* Pointer to the next element in the list */
} Smartphone;

typedef struct Storage
{
    int size;
    Smartphone *first_pos;
    Smartphone *last_pos;
} Storage;

Storage *create_storage();

void fill_storage(Storage *storage, char *filename, Brands **brands);

Smartphone *create_position(char *string, int index, Brands **brands);

void set_values(Smartphone *smartphone, char *str, Brands **brands);

void set_brand(Smartphone *smartphone, char *brand, Brands **brands);

void add_first(Storage *storage, Smartphone *new_position);

void add(Storage *storage, Smartphone *cur_position, Smartphone *new_position);

void choose_to_insert(Storage *storage, Brands **brands);

void print_header();

void print_table(Storage *storage);

void print(Smartphone *smartphone);

void insert_selected(Storage *storage, char *string, int index, Brands **brands);

void delete_position(Smartphone *position);

void delete_brands(Brands *brand);

void delete_storage(Storage *storage);

int main()
{
    Storage *Market;
    Brands *brands = NULL;
    int len;
    char filename[MAX_FILENAME_LEN];

    /* ------------get filename--------------- */
    printf("Input filename: ");
    fgets(filename, MAX_FILENAME_LEN, stdin);
    len = (int) strlen(filename);
    filename[len - 1] = '\0';
    /* ------------------------------------- */

    Market = create_storage();
```

```
    fill_storage(Market, filename, &brands);

    if (Market->size > 0)
        choose_to_insert(Market, &brands);

    printf("size: %i\n", Market->size);

    delete_brands(brands);

    delete_storage(Market);
    return 0;
}

Storage *create_storage()
{
    Storage *storage = NULL;
    storage = malloc(sizeof(Storage));
    if (storage)
    {
        storage->size = 0;
        storage->first_pos = NULL;
        storage->last_pos = NULL;
    }
    return storage;
}

void fill_storage(Storage *storage, char *filename, Brands **brands)
{
    FILE *source;
    Smartphone *new_pos;
    char tmp_str[MAX_STR_IN_FILE_LEN];
    int i;
    source = fopen(filename, "r");
    if (storage != NULL)
    {
        if (source != NULL)
        {
            for (i = 1; fgets(tmp_str, MAX_STR_IN_FILE_LEN, source); i++)
            {
                new_pos = create_position(tmp_str, i, brands);
                if (i == 1) add_first(storage, new_pos);
                else add(storage, storage->last_pos, new_pos);
            }
            fclose(source);
        } else
        {
            storage->size = -1;
            printf("Check the existence of the file\n");
        }
    }
}

Smartphone *create_position(char *string, int index, Brands **brands)
{
    Smartphone *position = NULL;
    position = (Smartphone *) malloc(sizeof(Smartphone));
    if (position)
    {
        position->index = index;
        position->next = NULL;
        set_values(position, string, brands);
    }
    return position;
}

void set_values(Smartphone *smartphone, char *str, Brands **brands)
```

```
{
    char *brand_name;
    smartphone->model = malloc(sizeof(char) * MAX_MODEL_NAME_LEN);
    smartphone->brand = NULL;
    brand_name = malloc(sizeof(char) * MAX_MODEL_NAME_LEN);
    sscanf(str, "%[^,],%[^,],%d,%d,%f,%f,%f", smartphone->model, brand_name,
           &(smartphone->ram), &(smartphone->memory), &(smartphone->screen_size),
           &(smartphone->weight), &(smartphone->price));
    set_brand(smartphone, brand_name, brands);
}

void set_brand(Smartphone *smartphone, char *brand, Brands **brands)
{
    Brands *cur_brand, *prev_brand;
    cur_brand = *brands;
    prev_brand = NULL;

    while (cur_brand != NULL)
    {
        if (strcmp(brand, cur_brand->name) == 0)
            smartphone->brand = cur_brand;
        prev_brand = cur_brand;
        cur_brand = cur_brand->next;
    }
    if (smartphone->brand == NULL)
    {
        cur_brand = malloc(sizeof(Brands));
        cur_brand->name = brand;
        if (prev_brand)
            prev_brand->next = cur_brand;
        else
            *brands = cur_brand;
        smartphone->brand = cur_brand;
    } else free(brand);
}

void add_first(Storage *storage, Smartphone *new_position)
{
    if (storage && new_position)
    {
        storage->first_pos = new_position;
        storage->last_pos = new_position;
        storage->size++;
    }
}

void add(Storage *storage, Smartphone *cur_position, Smartphone *new_position)
{
    if (storage && new_position && cur_position)
    {
        storage->size++;
        if (cur_position == storage->last_pos)
        {
            cur_position->next = new_position;
            storage->last_pos = new_position;
            new_position->next = storage->first_pos;
        } else
        {
            new_position->next = cur_position->next;
            cur_position->next = new_position;
        }
    }
}

void choose_to_insert(Storage *storage, Brands **brands)
{
```

```
    int val;
    char string[MAX_STR_IN_FILE_LEN];
    do
    {
        if (storage->size > 0)
            print_table(storage);
        printf("input the index of element you want insert or 0 to exit: ");
        scanf("%d", &val);
        if (val > 0)
        {
            getchar();
            printf("input the string: ");
            fgets(string, MAX_STR_IN_FILE_LEN, stdin);
            insert_selected(storage, string, val, brands);
        }

    } while (val != 0);
}

void print_header()
{
    printf("+-----+----------------------+----------------+-------+---------+--------+---------+-
---------+\n");
    printf("| %-5s | %-20s | %-15s | %-5s | %-5s | %-6s | %-7s | %-8s |\n",
           "№", "Model", "Brand", "RAM", "Storage", "Screen", "Weight", "Price");
    printf("+-----+----------------------+----------------+-------+---------+--------+---------+-
---------+\n");
}

void print_table(Storage *storage)
{
    Smartphone *cur;
    cur = storage->first_pos;
    print_header();
    do
    {
        print(cur);
        cur = cur->next;
    } while (cur != storage->first_pos);
    printf("+-----+----------------------+----------------+-------+---------+--------+---------+-
---------+\n");
}

void print(Smartphone *smartphone)
{
    printf("| %3i | %-20s | %-15s | %-3dGB | %-5dGB | %-5.2f\" | %-6.2fg | $%-7.2f |\n",
           smartphone->index, smartphone->model, smartphone->brand->name, smartphone->ram,
smartphone->memory,
           smartphone->screen_size, smartphone->weight, smartphone->price);
}

void insert_selected(Storage *storage, char *string, int index, Brands **brands)
{
    Smartphone *new, *cur;
    int found;

    found = 0;
    new = create_position(string, index, brands);
    if (storage->size <= index)
    {
        cur = storage->last_pos;
        new->index = storage->size + 1;
        add(storage, cur, new);
    } else
    {
        cur = storage->last_pos;
```

```
        do
        {
            cur = cur->next;
            if (cur->index == index)
            {
                add(storage, cur, new);
                new->index++;
                cur = new->next;
                found = 1;
            } else if (found) cur->index++;
        } while (cur->index < storage->size);
    }
}

void delete_position(Smartphone *position)
{
    free(position->model);
    position->next = NULL;
    free(position);
}

void delete_brands(Brands *brand)
{
    Brands *tmp_brand;
    while (brand)
    {
        free(brand->name);
        tmp_brand = brand;
        brand = brand->next;
        free(tmp_brand);
    }
}

void delete_storage(Storage *storage)
{
    Smartphone *cur, *next;
    cur = storage->first_pos;
    do
    {
        next = cur->next;
        delete_position(cur);
        cur = next;
    }
    while (cur != storage->first_pos);
    free(storage);
}
```

Примеры выполнения программы

Пример1:

```
Input filename: data.csv
+-----+----------------------+-----------------+-------+---------+--------+---------+----------+
| №   | Model                | Brand           | RAM   | Storage | Screen | Weight  | Price    |
+-----+----------------------+-----------------+-------+---------+--------+---------+----------+
|   1 | iPhone11             | Apple           | 4  GB | 128  GB | 6.06 " | 194.00g | $599.00  |
|   2 | iPhone12             | Apple           | 4  GB | 256  GB | 6.10 " | 215.00g | $799.00  |
|   3 | Galaxy S20           | Samsung         | 6  GB | 128  GB | 6.20 " | 163.00g | $699.00  |
|   4 | OnePlus 8            | OnePlus         | 8  GB | 256  GB | 6.55 " | 180.00g | $699.00  |
|   5 | Pixel 5              | Google          | 8  GB | 128  GB | 6.00 " | 151.00g | $699.00  |
|   6 | Xperia 1 II          | Sony            | 8  GB | 256  GB | 6.50 " | 181.00g | $1099.00 |
+-----+----------------------+-----------------+-------+---------+--------+---------+----------+
input the index of element you want insert or 0 to exit: 2
input the string: iPhone SE 2020,Apple,3,64,4.7,148,399
+-----+----------------------+-----------------+-------+---------+--------+---------+----------+
| №   | Model                | Brand           | RAM   | Storage | Screen | Weight  | Price    |
+-----+----------------------+-----------------+-------+---------+--------+---------+----------+
|   1 | iPhone11             | Apple           | 4  GB | 128  GB | 6.06 " | 194.00g | $599.00  |
|   2 | iPhone12             | Apple           | 4  GB | 256  GB | 6.10 " | 215.00g | $799.00  |
|   3 | iPhone SE 2020       | Apple           | 3  GB | 64   GB | 4.70 " | 148.00g | $399.00  |
|   3 | Galaxy S20           | Samsung         | 6  GB | 128  GB | 6.20 " | 163.00g | $699.00  |
|   5 | OnePlus 8            | OnePlus         | 8  GB | 256  GB | 6.55 " | 180.00g | $699.00  |
|   6 | Pixel 5              | Google          | 8  GB | 128  GB | 6.00 " | 151.00g | $699.00  |
|   7 | Xperia 1 II          | Sony            | 8  GB | 256  GB | 6.50 " | 181.00g | $1099.00 |
+-----+----------------------+-----------------+-------+---------+--------+---------+----------+
input the index of element you want insert or 0 to exit: 0
size: 7
Process finished with exit code 0
```

Пример 2:

```
Input filename: data2.csv
+-----+----------------------+-----------------+-------+---------+--------+---------+----------+
| №   | Model                | Brand           | RAM   | Storage | Screen | Weight  | Price    |
+-----+----------------------+-----------------+-------+---------+--------+---------+----------+
|   1 | Galaxy S21           | Samsung         | 8  GB | 128  GB | 6.20 " | 171.00g | $799.00  |
|   2 | iPhone SE 2020       | Apple           | 3  GB | 64   GB | 4.70 " | 148.00g | $399.00  |
|   3 | OnePlus 9            | OnePlus         | 12 GB | 256  GB | 6.55 " | 183.00g | $899.00  |
+-----+----------------------+-----------------+-------+---------+--------+---------+----------+
input the index of element you want insert or 0 to exit: 45
input the string: OnePlus 8,OnePlus,8,256,6.55,180,699
+-----+----------------------+-----------------+-------+---------+--------+---------+----------+
| №   | Model                | Brand           | RAM   | Storage | Screen | Weight  | Price    |
+-----+----------------------+-----------------+-------+---------+--------+---------+----------+
|   1 | Galaxy S21           | Samsung         | 8  GB | 128  GB | 6.20 " | 171.00g | $799.00  |
|   2 | iPhone SE 2020       | Apple           | 3  GB | 64   GB | 4.70 " | 148.00g | $399.00  |
|   3 | OnePlus 9            | OnePlus         | 12 GB | 256  GB | 6.55 " | 183.00g | $899.00  |
|   4 | OnePlus 8            | OnePlus         | 8  GB | 256  GB | 6.55 " | 180.00g | $699.00  |
+-----+----------------------+-----------------+-------+---------+--------+---------+----------+
input the index of element you want insert or 0 to exit: 1
input the string: iPhone12,Apple,4,256,6.1,215,799
+-----+----------------------+-----------------+-------+---------+--------+---------+----------+
| №   | Model                | Brand           | RAM   | Storage | Screen | Weight  | Price    |
+-----+----------------------+-----------------+-------+---------+--------+---------+----------+
|   1 | Galaxy S21           | Samsung         | 8  GB | 128  GB | 6.20 " | 171.00g | $799.00  |
|   2 | iPhone12             | Apple           | 4  GB | 256  GB | 6.10 " | 215.00g | $799.00  |
|   2 | iPhone SE 2020       | Apple           | 3  GB | 64   GB | 4.70 " | 148.00g | $399.00  |
|   4 | OnePlus 9            | OnePlus         | 12 GB | 256  GB | 6.55 " | 183.00g | $899.00  |
|   5 | OnePlus 8            | OnePlus         | 8  GB | 256  GB | 6.55 " | 180.00g | $699.00  |
+-----+----------------------+-----------------+-------+---------+--------+---------+----------+
input the index of element you want insert or 0 to exit: 0
size: 5
```

Выводы.

В результате выполнения работы были изучены линейные кольцевые списки в языке Си и получены практические навыки в программировании на этом языке.