

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

ОТЧЕТ
по лабораторной работе № 7
по дисциплине «Программирование»
Тема: УКАЗАТЕЛИ НА СТРУКТУРЫ И ФУНКЦИИ

Студент гр. 3312

Мохно Даниил.

Преподаватель

Аббас Саддам

Санкт-Петербург

2024

Цель работы.

Целью работы является изучение указателей на структуры и функции в языке Си и получение практических навыков в программировании на этом языке.

Задание (вариант 1)

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

Обязательный набор полей:

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

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

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

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

Постановка задачи и описание решения

Для начала объявим структуру. Структура `Smartphone` содержит 9 полей. Два символьных поля: `model` — модель смартфона и `brand` — марка смартфона. Три целочисленных поля: `ram` — объём оперативной памяти, `memory` — объём постоянной памяти, `number_of_cameras` — количество камер в смартфоне. Три поля с плавающей точкой: `screen_size` — размер экрана, `weight` — вес, `price` — цена. И целочисленный массив `camera_resolution`, содержащий разрешения всех камер смартфона. Для удобства выделим структуру в новый тип данных.

В главной функции объявляем переменную `Market` – двойной указатель на `Smartphone`, она будет являться массивом указателей на структуры. Объявляем массив функций `values`, принимающих структуру и возвращающих значение поля или поля с плавающей точкой. Принимаем от пользователя название файла с исходными данными. Заполняем массив указателей на структуры данными из файла, путём получения в переменную `Market` указателя на указатель из функции `create_storage` принимающей указатель на переменную, хранящую размер массива и имя файла. Затем, если размер массива не 0, мы выводим данные из каждой структуры массива в таблицу через цикл и функцию для вывода структуры `print`. Затем заполняем функциями массив функций `values` и вызываем функцию выбора действий из меню – `menu`, куда передаём массив `Market`, указатель на размер массива, и массив `values`. После этого освобождаем место, выделенное под массив. Если же длина массива 0, то выводим сообщение об ошибке.

В функции создания массива указателей на структуры, мы открываем файл с исходными данными. Если он открылся мы циклом считываем каждую строку из файла увеличивая каждый раз размер массива, затем мы сбрасываем указатель текущей позиции файла в начало файла и выделяем место под массив указателей. Далее проверяем чтобы память выделилась, и проходимся циклом до размера массива, передавая каждый раз массив, индекс и файл в функцию добавления элемента. Возвращаем двойной указатель. Если файл открыт не был выводим сообщение. В конце, если двойной указатель не вернулся, возвращаем 0.

В функции добавления новой позиции в массив мы считываем строку из файла/консоли, считаем в ней кол-во разделителей, и если кол-во разделителей равно 7, то мы, когда файл `stdin` пере выделяем память для массива на 1 больше, выделяем память под новый элемент, и вызываем функцию присвоения значений полям в которую передаём указатель на структуру и строку считанную из файла. В случае, когда кол-во разделителей не равно 7, мы выводим сообщение о несовпадении формата строки.

В функции присвоения значений мы выделяем место под строки модели и бренда и строку содержащую подстроку с камерами. Присваиваем значения из строки в поля структуры и строку с камерами путём форматированного сканирования строки функцией `sscanf`. Затем вызываем функцию разделения строки с разрешениями камер и заполнения массива камер. Освобождаем место из-под строки с разрешениями камер.

В функции разделения строки с камерами мы присваиваем длину строки в переменную `len` и перебираем строку считая разделённые числа. Далее выделяем память для массива разрешений камер в структуре и для строки временного хранения отделённых значений. Если кол-во чисел в строке не равно 0, то циклом проходимся по строке. Цикл начинается с инициализации трех переменных: `i`, `j` и `k`, которые используются как счетчики и индексы. Затем цикл выполняется до тех пор, пока `i` не превысит длину строки `len`. Внутри цикла проверяется каждый символ строки `str` по индексу `i`. Если символ не является знаком разделителем и не является символом конца строки (`'\0'`), то он копируется в временную строку по индексу `j`. Если символ равен символу разделителю или является символом конца строки, то производится преобразование временной строки в целое число с помощью `atoi`, которое затем присваивается элементу массива разрешений камер по индексу `k`. После этого временная строка обнуляется, а счетчики `k` и `j` обновляются для следующей итерации. В конце присваиваем полю количества камер кол-во разделённых чисел и освобождаем временную строку.

В функции вывода `print` мы склеиваем значения камер из поля с массивом камер в одну строку и выводим её и другие поля в таблицу. Для формирования строки с разрешениями камер `cameras_res` из поля с массивом со значениями мы объявляем заполняемую строку с разрешениями камер и промежуточную строку для перевода числа в строку `tmp_cam`. Выделяем память для строки `cameras_res`. Затем циклом проходимся по массиву разрешений камер. В нём преобразуем число из массива в строку, записанную в `tmp_cam` с помощью функции `sprintf`. Затем вложенный цикл проходится по символам этой строки,

копируя их в строку `cameras_res` по индексу `j`. Если это не последняя камера, после копирования числа добавляется символ '+' в строку `cameras_res`, и счетчик `j` обновляется. После вывода освобождаем строку `cameras_res`.

В функции меню мы запускаем цикл с пост условием. В нём мы выводим на экран пользователя опции, и запрашиваем номер одной из опций. Далее мы подставляем полученное значение в оператор выбора. Если 1,2,3,4,5 или 6, то мы просто запускаем функцию сортировки передавая ей массив, его размер, и функцию из массива функций получения значений полей по индексу значения введённого пользователем – 1, а за тем выводим таблицу из всех полей структур из массива структур. Если же пользователь ввёл 7, то мы вызываем функцию добавления элемента и передаём в неё массив, указатель на его размер и в качестве файла, стандартный файл ввода `stdin`, за тем увеличиваем значение размера массива на 1. Если же пользователь ввёл 8, то мы вызываем функцию поиска структуры в массиве структур, в которую передаём массив и его размер. Если пользователь ввёл значение 0, цикл завершится.

В функции сортировки мы используем алгоритм сортировки Шелла. Для начала мы устанавливаем шаг `gap` равным половине массива и запускаем цикл, который будет работать пока `gap > 0` и с каждой итерацией будет уменьшать размер шага в 2 раза. Далее запускаем внутренний цикл по массиву `storage`, который начинается с `gap`, а внутри него итерацию по всем элементам с шагом `gap`. Внутренний цикл сравнивает значения элементов с шагом `gap` и, при необходимости, производит обмен элементов. Обмен элементов выполняется с использованием временной переменной `tmp`, которая хранит указатель на структуру `Smartphone`. Этот процесс повторяется для всех элементов подгруппы с заданным шагом. Сравниваемые значения полей структуры, получаем с помощью функции переданной функции получения поля.

В функции поиска структуры в массиве структур по первому слову в поле `model` мы выделяем память под переменную строки, которую введёт пользователь, считываем в неё строку, введённую пользователем в консоль, и записываем её длину в переменную `len`. Устанавливаем флаг найденной строки

равным 0 и проходимся циклом по массиву указателей на структуры, сравнивая каждый раз первые len символов строки, введенной пользователем и поля model. Если они совпадают, то мы выводим на экран поля структуры функцией print и присваиваем флагу найденной строки 1. После цикла проверяем значение флага, если оно равно 0, то выводим сообщение. В конце освобождаем выделенную память.

Описание переменных

Функция – int main():

№	Имя переменной	Тип	Назначение
1	Market	struct Smartphone	Массив указателей на структуры
2	values	float(*Getters)(Smartphone *)	Массив функций получения численных значений полей структуры
3	len	int	Длина названия файла
4	number_of_products	int	Размер массива указателей на структуры
5	filename	char	Название файла

Функция создания массива указателей на структуры – Smartphone create_storage(int *n, char *source_file_name)

№	Имя переменной	Тип	Назначение
1	storage	struct Smartphone	Массив указателей на структуры
2	source	File	Файл с входными данными
3	n	int	Размер массива указателей на структуры
4	source_file_name	int	Название файла
5	tmp_str	char	Временная строка для чтения данных из файла
6	i	int	Итератор

Функция добавления элемента в массив указателей на структуры –
add_new_position(Smartphone **storage, int *index, FILE *file)

№	Имя переменной	Тип	Назначение
1	storage	struct Smartphone	Массив указателей на структуры
2	fiile	File	Файл с входными данными
3	index	int	Указатель на размер массива и индекс добавляемого элемента
4	tmp_str	char	Временная строка для чтения данных из файла
5	len	int	Длина считываемой строки
6	cnt	int	Счётчик точек с запятой
7	i	int	Итератор

Функция изменения размера массива структур – void
resize_storage(Smartphone *storage, int *size)**

№	Имя переменной	Тип	Назначение
1	storage	struct Smartphone	Указатель на массив указателей на структуры
2	size	int	Указатель на размер массива указателей на структуры

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

№	Имя переменной	Тип	Назначение
1	smartphone	struct Smartphone	Указатель на структуру
2	str	char	Строка из файла
3	cameras	char	Подстрока с разрешениями камер

Функция разделения разрешений камер – void split_camera_resolution(Smartphone *smartphone, char *str)

№	Имя переменной	Тип	Назначение
1	smartphone	struct Smartphone	указатель на структуру
2	str	char	Строка с разрешениями камер
3	i	int	итератор
4	j	int	итератор
5	k	int	итератор
6	tmp_str	char	Временная строка для хранения подстрок с числами из строки с расширениями
7	len	int	Длинна строки с расширениями камер
8	number	int	Счётчик чисел в строке разделённых знаком

Функция вывода данных в таблицу – void print_table(Smartphone **storage, int n)

№	Имя переменной	Тип	Назначение
1	storage	struct Smartphone	Массив указателей на структуры
2	n	int	Размер массива
3	i	int	итератор

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

№	Имя переменной	Тип	Назначение
1	smartphone	struct Smartphone	Указатель на структуру
2	i	int	Итератор
3	J	int	Итератор
4	k	int	Итератор
5	tmp_cam	char	Строка для преобразования чисел в строку
6	camera_res	char	Строка для склеивания всех разрешений из массива разрешений камер

Функция выбора пункта меню – void menu(Smartphone **storage, int *n, Getters *get_value)

№	Имя переменной	Тип	Назначение
1	storage	struct Smartphone	Массив указателей на структуры
2	get_values	float(*Getters)(Smartphone *)	Массив функций получения численных значений полей структуры
3	value	int	Значение пункта меню, выбранное пользователем
4	n	int	Указатель на размер массива указателей на структуры

Функция сортировки массива указателей на структуры – void sort_storage(Smartphone **storage, int size, Getters get_value)

№	Имя переменной	Тип	Назначение
1	storage	struct Smartphone	Массив указателей на структуры
2	get_values	float(*Getters)(Smartphone *)	функция получения численных значений полей структуры
3	size	int	Размер массива указателей на структуры
4	gap	int	Шаг сортировки
5	i	int	Итератор
6	k	int	Итератор
7	tmp	Smartphone	Временная переменная для обмена указателями на структуру

Функция поиска нужной структуры по значению строкового поля– void find_element_by_str(Smartphone **storage, int size)

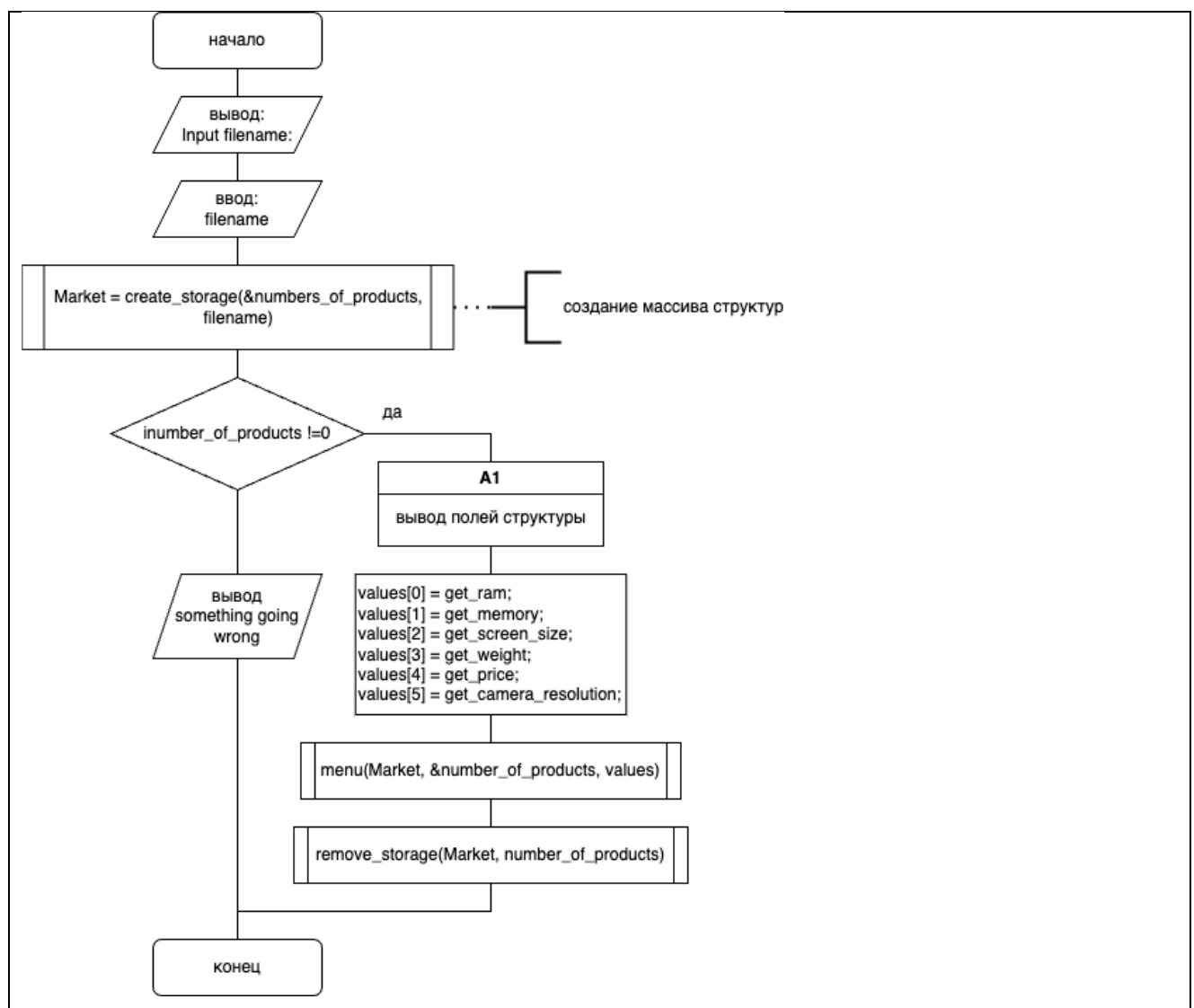
№	Имя переменной	Тип	Назначение
1	storage	struct Smartphone	Массив указателей на структуры
2	size	int	Размер массива указателей на структуры
3	string	char	Строка полученная от пользователя
4	len	int	Длина строки полученной от пользователя
5	i	int	Итератор

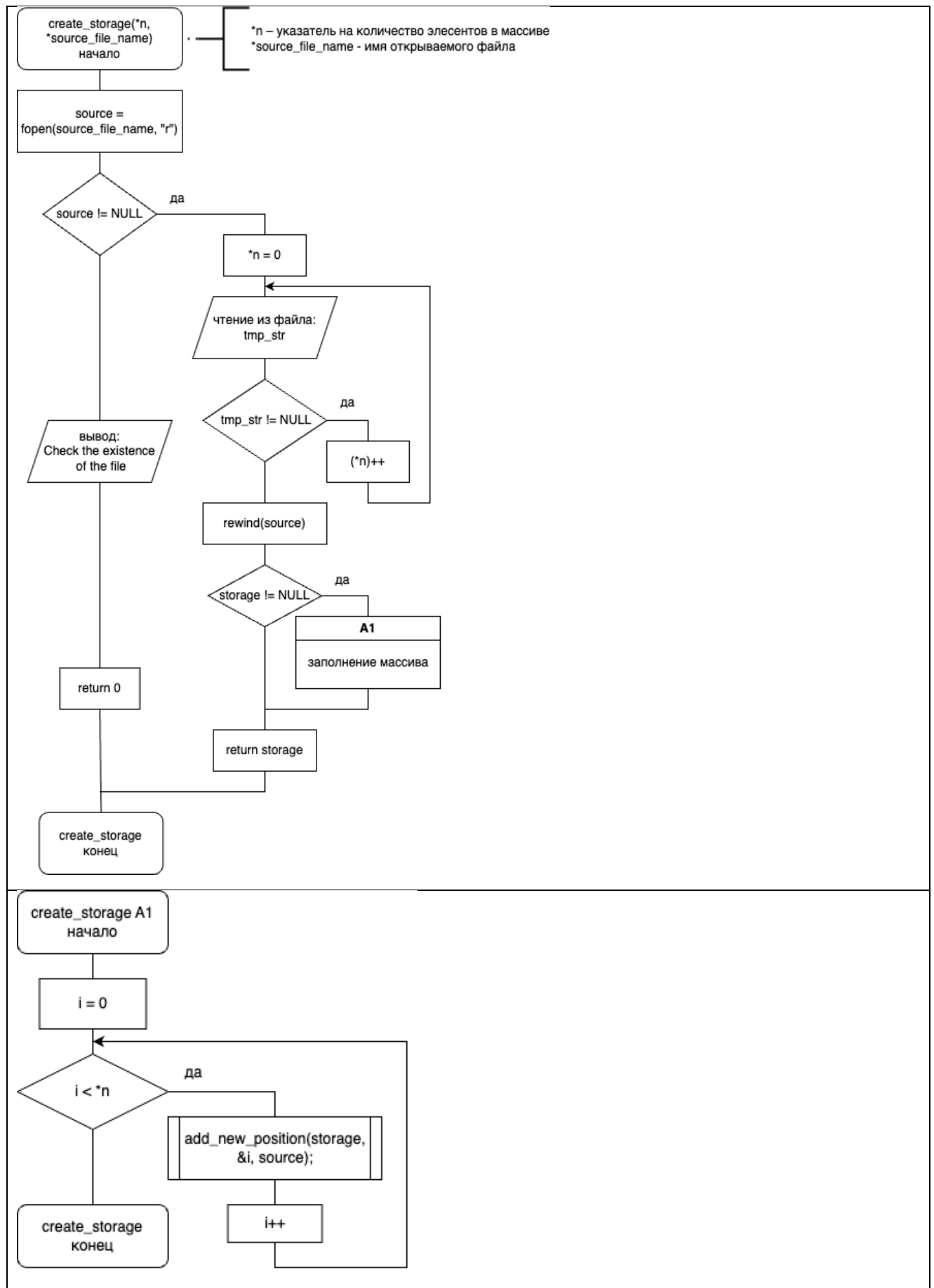
6	found	int	Флаг найденного строки
---	-------	-----	------------------------

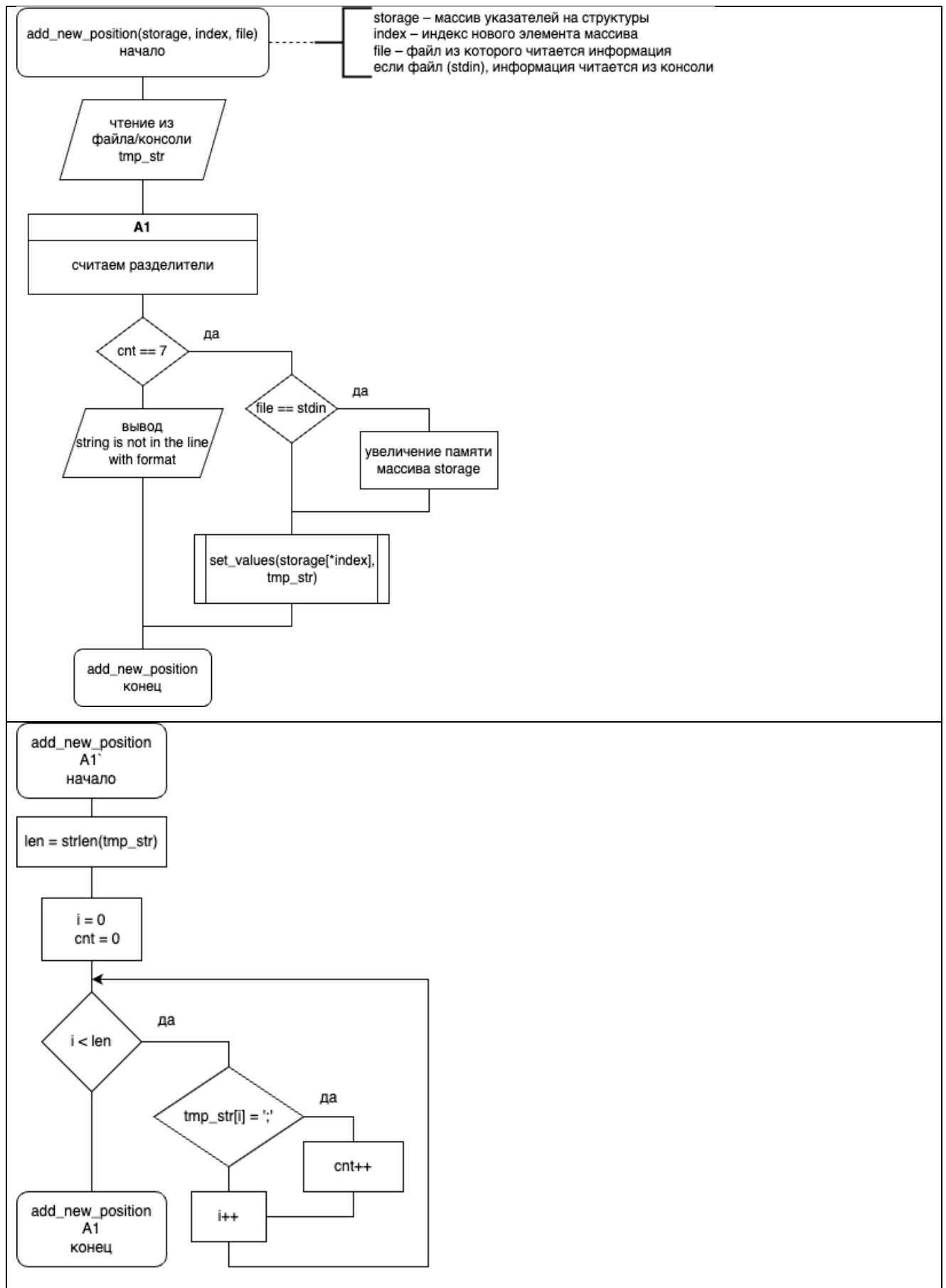
Функция удаления массива указателей на структуры – void remove_storage(Smartphone **storage, int size)

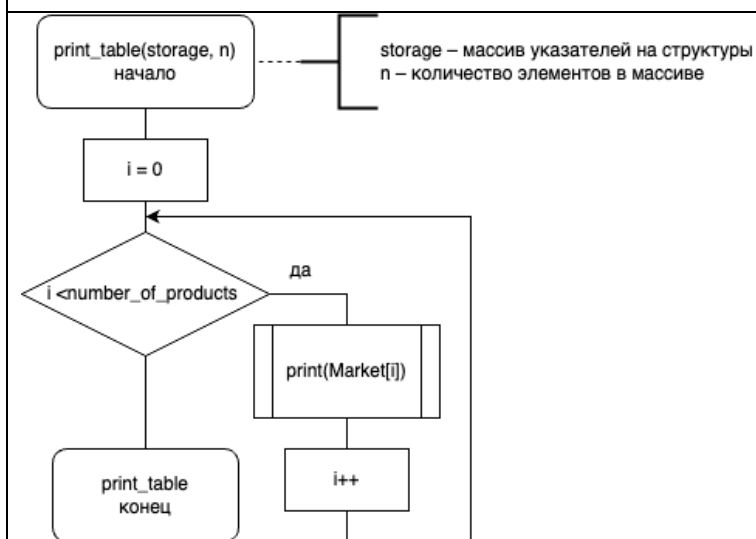
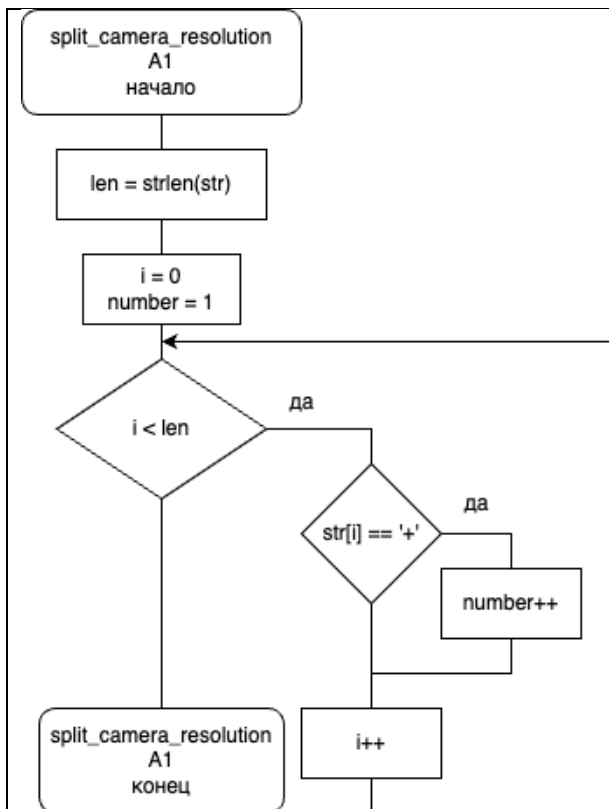
№	Имя переменной	Тип	Назначение
1	storage	struct Smartphone	Массив указателей на структуры
2	size	int	Размер массива
3	i	int	итератор

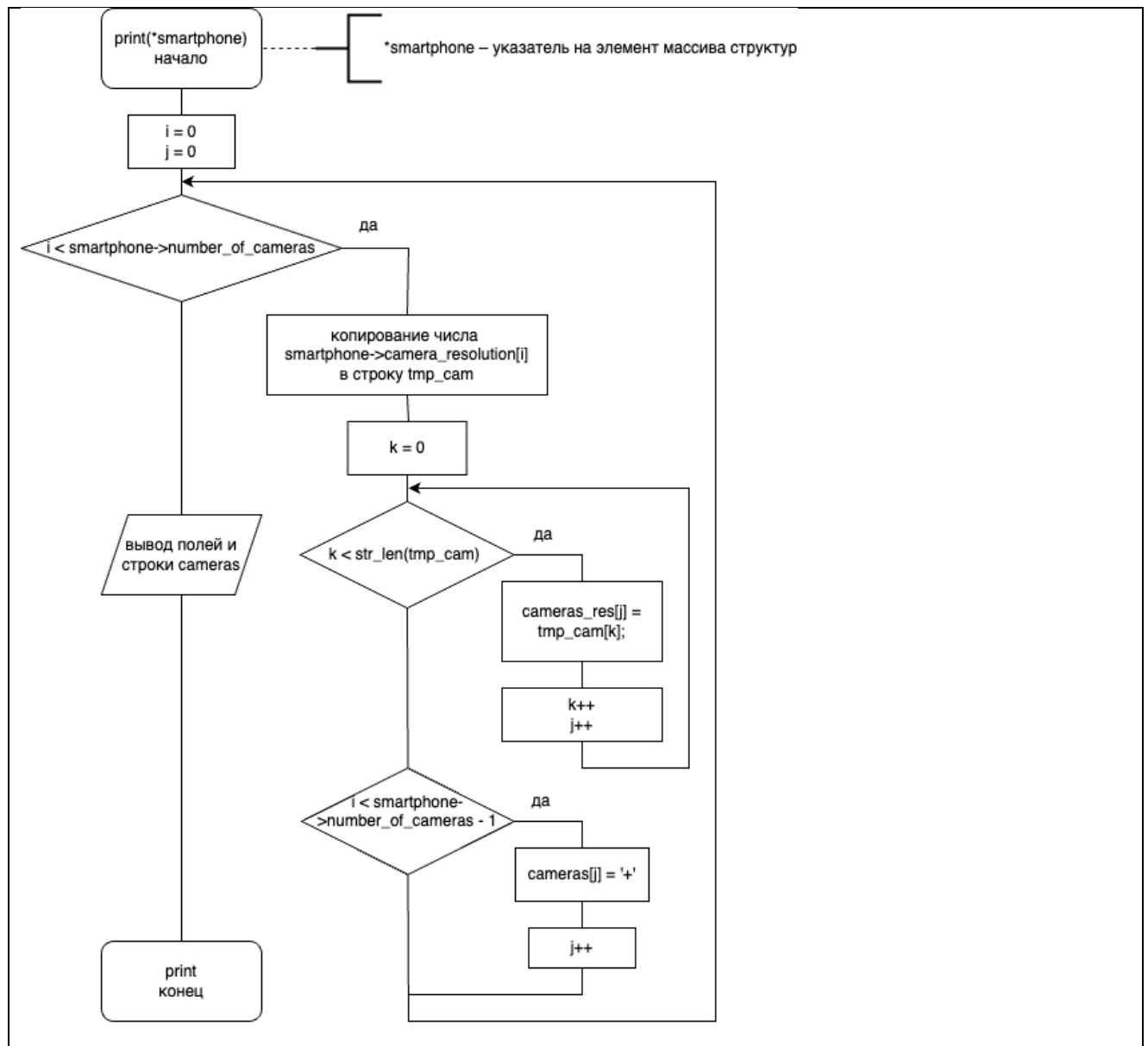
Схема алгоритма

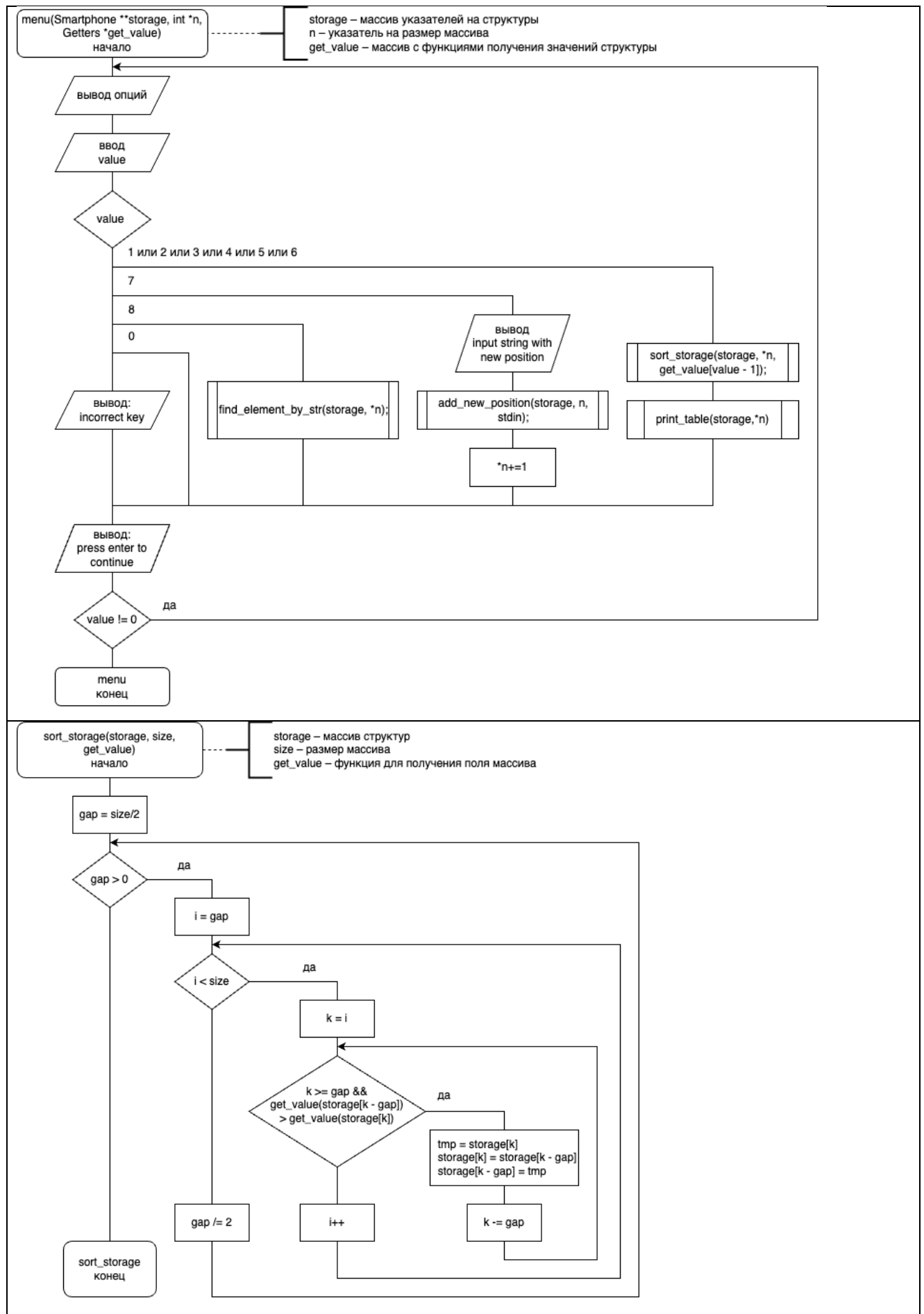


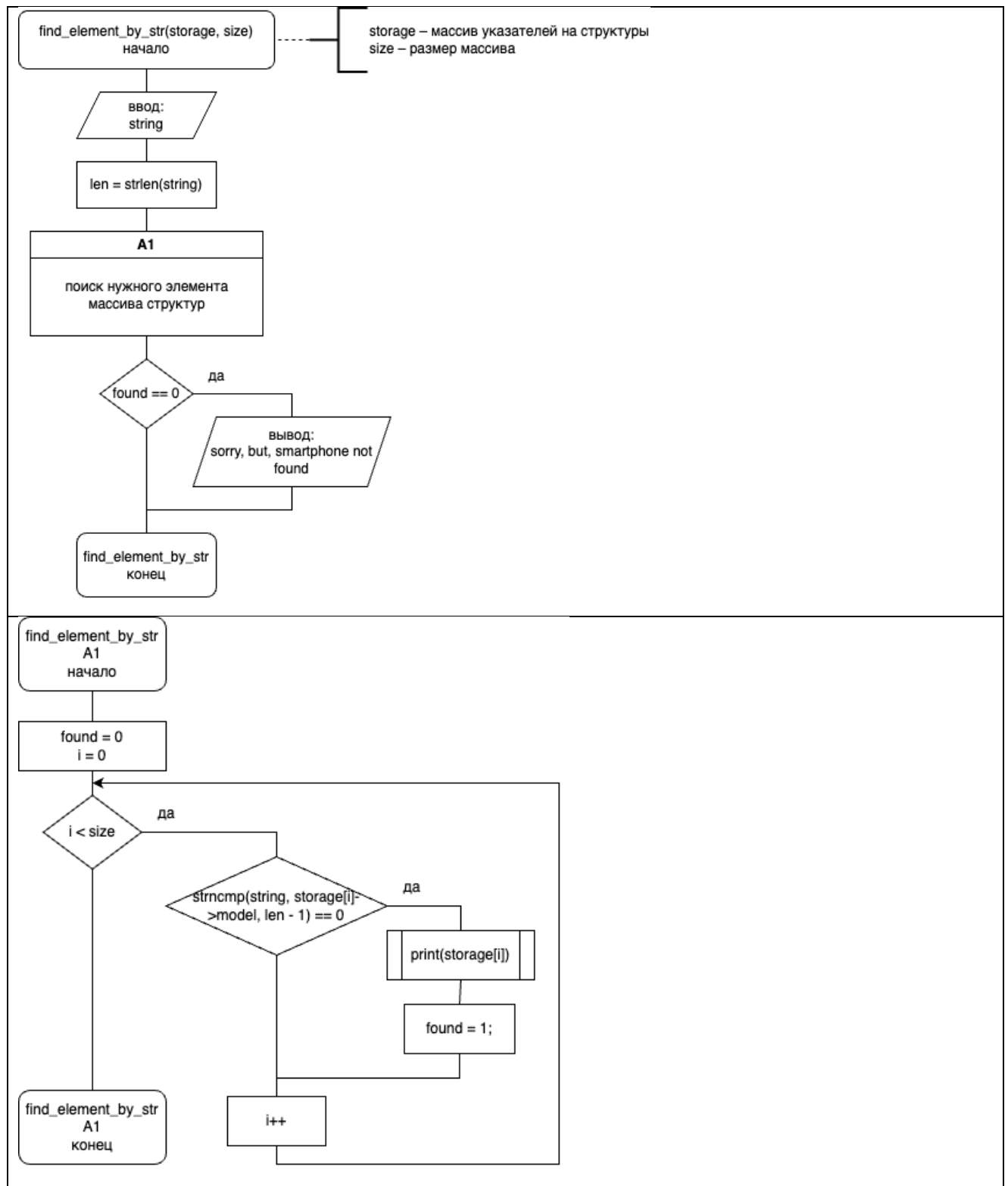












Контрольные примеры

Пример 1:

Содержимое файла Data.csv
iPhone11;Apple;4;128;6.06;194;599;12
iPhone12;Apple;4;256;6.1;215;799;12
Galaxy S20;Samsung;6;128;6.2;163;699;64
OnePlus 8;OnePlus;8;256;6.55;180;699;48
Pixel 5;Google;8;128;6.0;151;699;12
Xperia 1 II;Sony;8;256;6.5;181;1099;12+12+12+12+12
Redmi Note 9;Xiaomi;4;64;6.53;199;229;48+8+2+2
Mate 30 Pro;Huawei;8;256;6.53;198;1099;40+40+8
LG Velvet;LG;6;128;6.8;180;699;48+8+5
Mi 10;Xiaomi;8;256;6.67;208;799;108+13+2+2
Galaxy A51;Samsung;6;128;6.5;172;349;48+12+5+5
Motorola Edge;Motorola;6;256;6.7;188;699;64+16+8
Nokia 9 PureView;Nokia;6;128;5.99;172;699;12+12+12+12+12

Входные данные
Data.csv
6
7
iPhone SE 2020;Apple;3;64;4.7;148;399;12
8
iPhone
0

Выходные данные

Model	Brand	RAM	Storage	Screen	Weight	Price	Camera Resolution
iPhone11	Apple	4 GB	128 GB	6.06 "	194.00g	\$599.00	12mp
iPhone12	Apple	4 GB	256 GB	6.10 "	215.00g	\$799.00	12mp
Galaxy S20	Samsung	6 GB	128 GB	6.20 "	163.00g	\$699.00	64mp
OnePlus 8	OnePlus	8 GB	256 GB	6.55 "	180.00g	\$699.00	48mp
Pixel 5	Google	8 GB	128 GB	6.00 "	151.00g	\$699.00	12mp
Xperia 1 II	Sony	8 GB	256 GB	6.50 "	181.00g	\$1099.00	12+12+12+12+12mp
Redmi Note 9	Xiaomi	4 GB	64 GB	6.53 "	199.00g	\$229.00	48+8+2+2mp
Mate 30 Pro	Huawei	8 GB	256 GB	6.53 "	198.00g	\$1099.00	40+40+8mp
LG Velvet	LG	6 GB	128 GB	6.80 "	180.00g	\$699.00	48+8+5mp
Mi 10	Xiaomi	8 GB	256 GB	6.67 "	208.00g	\$799.00	108+13+2+2mp
Galaxy A51	Samsung	6 GB	128 GB	6.50 "	172.00g	\$349.00	48+12+5+5mp
Motorola Edge	Motorola	6 GB	256 GB	6.70 "	188.00g	\$699.00	64+16+8mp
Nokia 9 PureView	Nokia	6 GB	128 GB	5.99 "	172.00g	\$699.00	12+12+12+12+1mp

Model	Brand	RAM	Storage	Screen	Weight	Price	Camera Resolution
iPhone11	Apple	4 GB	128 GB	6.06 "	194.00g	\$599.00	12mp
iPhone12	Apple	4 GB	256 GB	6.10 "	215.00g	\$799.00	12mp
Xperia 1 II	Sony	8 GB	256 GB	6.50 "	181.00g	\$1099.00	12+12+12+12+12mp
Nokia 9 PureView	Nokia	6 GB	128 GB	5.99 "	172.00g	\$699.00	12+12+12+12+1mp
Pixel 5	Google	8 GB	128 GB	6.00 "	151.00g	\$699.00	12mp
Mate 30 Pro	Huawei	8 GB	256 GB	6.53 "	198.00g	\$1099.00	40+40+8mp
LG Velvet	LG	6 GB	128 GB	6.80 "	180.00g	\$699.00	48+8+5mp
OnePlus 8	OnePlus	8 GB	256 GB	6.55 "	180.00g	\$699.00	48mp
Redmi Note 9	Xiaomi	4 GB	64 GB	6.53 "	199.00g	\$229.00	48+8+2+2mp
Galaxy A51	Samsung	6 GB	128 GB	6.50 "	172.00g	\$349.00	48+12+5+5mp
Galaxy S20	Samsung	6 GB	128 GB	6.20 "	163.00g	\$699.00	64mp
Motorola Edge	Motorola	6 GB	256 GB	6.70 "	188.00g	\$699.00	64+16+8mp
Mi 10	Xiaomi	8 GB	256 GB	6.67 "	208.00g	\$799.00	108+13+2+2mp

Model	Brand	RAM	Storage	Screen	Weight	Price	Camera Resolution
iPhone11	Apple	4 GB	128 GB	6.06 "	194.00g	\$599.00	12mp
iPhone12	Apple	4 GB	256 GB	6.10 "	215.00g	\$799.00	12mp
iPhone SE 2020	Apple	3 GB	64 GB	4.70 "	148.00g	\$399.00	12mp

Пример 2:

Содержимое файла Data2.csv
Galaxy S21;Samsung;8;128;6.2;171;799;64+12+12
iPhone SE 2020;Apple;3;64;4.7;148;399;12
OnePlus 9;OnePlus;12;256;6.55;183;899;48+50+2
Moto G Power;Motorola;4;64;6.4;199;249;16+8+2+2
Redmi K40;Xiaomi;6;128;6.67;196;399;48+8+5
Pixel 6;Google;6;128;6.4;176;699;50+12+16
Galaxy Z Flip 3;Samsung;8;256;6.7;183;1099;12+12+12
iPhone 13 Pro;Apple;6;256;6.1;238;1099;12+12+12
Xperia 5 III;Sony;8;128;6.1;169;899;12+12+12
Redmi Note 10;Xiaomi;6;128;6.43;178;279;48+8+2+2
OnePlus Nord;OnePlus;6;128;6.44;184;399;48+8+5+2
Galaxy A72;Samsung;6;128;6.7;203;449;64+8+12+5
iPhone 12 Mini;Apple;4;64;5.4;133;699;12+12
Mi 11 Lite;Xiaomi;6;128;6.55;157;349;64+8+5
Zenfone 8;Asus;8;256;5.9;169;699;64+12+12

Входные данные
Data2.csv
7
Galaxy M32;Samsung;6;128;6.4;196;249;64+8+2+2
5
8
Galaxy
0

Выходные данные

Model	Brand	RAM	Storage	Screen	Weight	Price	Camera Resolution
Galaxy S21	Samsung	8 GB	128 GB	6.20 "	171.00g	\$799.00	64+12+12mp
iPhone SE 2020	Apple	3 GB	64 GB	4.70 "	148.00g	\$399.00	12mp
OnePlus 9	OnePlus	12 GB	256 GB	6.55 "	183.00g	\$899.00	48+50+2mp
Moto G Power	Motorola	4 GB	64 GB	6.40 "	199.00g	\$249.00	16+8+2+2mp
Redmi K40	Xiaomi	6 GB	128 GB	6.67 "	196.00g	\$399.00	48+8+5mp
Pixel 6	Google	6 GB	128 GB	6.40 "	176.00g	\$699.00	50+12+16mp
Galaxy Z Flip 3	Samsung	8 GB	256 GB	6.70 "	183.00g	\$1099.00	12+12+12mp
iPhone 13 Pro	Apple	6 GB	256 GB	6.10 "	238.00g	\$1099.00	12+12+12mp
Xperia 5 III	Sony	8 GB	128 GB	6.10 "	169.00g	\$899.00	12+12+12mp
Redmi Note 10	Xiaomi	6 GB	128 GB	6.43 "	178.00g	\$279.00	48+8+2+2mp
OnePlus Nord	OnePlus	6 GB	128 GB	6.44 "	184.00g	\$399.00	48+8+5+2mp
Galaxy A72	Samsung	6 GB	128 GB	6.70 "	203.00g	\$449.00	64+8+12+5mp
iPhone 12 Mini	Apple	4 GB	64 GB	5.40 "	133.00g	\$699.00	12+12mp
Mi 11 Lite	Xiaomi	6 GB	128 GB	6.55 "	157.00g	\$349.00	64+8+5mp
Zenfone 8	Asus	8 GB	256 GB	5.90 "	169.00g	\$699.00	64+12+1mp

Model	Brand	RAM	Storage	Screen	Weight	Price	Camera Resolution
Moto G Power	Motorola	4 GB	64 GB	6.40 "	199.00g	\$249.00	16+8+2+2mp
Galaxy M32	Samsung	6 GB	128 GB	6.40 "	196.00g	\$249.00	64+8+2+2mp
Redmi Note 10	Xiaomi	6 GB	128 GB	6.43 "	178.00g	\$279.00	48+8+2+2mp
Mi 11 Lite	Xiaomi	6 GB	128 GB	6.55 "	157.00g	\$349.00	64+8+5mp
Redmi K40	Xiaomi	6 GB	128 GB	6.67 "	196.00g	\$399.00	48+8+5mp
OnePlus Nord	OnePlus	6 GB	128 GB	6.44 "	184.00g	\$399.00	48+8+5+2mp
iPhone SE 2020	Apple	3 GB	64 GB	4.70 "	148.00g	\$399.00	12mp
Galaxy A72	Samsung	6 GB	128 GB	6.70 "	203.00g	\$449.00	64+8+12+5mp
iPhone 12 Mini	Apple	4 GB	64 GB	5.40 "	133.00g	\$699.00	12+12mp
Zenfone 8	Asus	8 GB	256 GB	5.90 "	169.00g	\$699.00	64+12+1mp
Pixel 6	Google	6 GB	128 GB	6.40 "	176.00g	\$699.00	50+12+16mp
Galaxy S21	Samsung	8 GB	128 GB	6.20 "	171.00g	\$799.00	64+12+12mp
OnePlus 9	OnePlus	12 GB	256 GB	6.55 "	183.00g	\$899.00	48+50+2mp
Xperia 5 III	Sony	8 GB	128 GB	6.10 "	169.00g	\$899.00	12+12+12mp
Galaxy Z Flip 3	Samsung	8 GB	256 GB	6.70 "	183.00g	\$1099.00	12+12+12mp
iPhone 13 Pro	Apple	6 GB	256 GB	6.10 "	238.00g	\$1099.00	12+12+12mp

Model	Brand	RAM	Storage	Screen	Weight	Price	Camera Resolution
Galaxy M32	Samsung	6 GB	128 GB	6.40 "	196.00g	\$249.00	64+8+2+2mp
Galaxy A72	Samsung	6 GB	128 GB	6.70 "	203.00g	\$449.00	64+8+12+5mp
Galaxy S21	Samsung	8 GB	128 GB	6.20 "	171.00g	\$799.00	64+12+12mp
Galaxy Z Flip 3	Samsung	8 GB	256 GB	6.70 "	183.00g	\$1099.00	12+12+12mp

Текст программы

```

#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 Smartphone
{
    char *model;
    char *brand;
    int ram;
    int memory;
    float screen_size;
    float weight;
    float price;
    int *camera_resolution;
    int number_of_cameras;
} Smartphone;

typedef float (*Getters)(Smartphone *);

Smartphone **create_storage(int *n, char *source_file_name);

void add_new_position(Smartphone **storage, int *index, FILE *file);

void resize_storage(Smartphone ***storage, int *size);

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

void split_camera_resolution(Smartphone *smartphone, char *str);

```

```

void print_table(Smartphone **storage, int n);

void print_header();

void print(Smartphone *smartphone);

void menu(Smartphone **storage, int *n, Getters *get_value);

void remove_storage(Smartphone **storage, int size);

void sort_storage(Smartphone **storage, int size, Getters get_value);

void find_element_by_str(Smartphone **storage, int size);

float get_ram(Smartphone *smartphone);

float get_memory(Smartphone *smartphone);

float get_screen_size(Smartphone *smartphone);

float get_weight(Smartphone *smartphone);

float get_price(Smartphone *smartphone);

float get_camera_resolution(Smartphone *smartphone);

int main()
{
    Smartphone **Market;
    Getters values[6];
    int number_of_products;
    int len;
    char filename[MAX_FILENAME_LEN];

    printf("Input filename: ");
    fgets(filename, MAX_MODEL_NAME_LEN, stdin);
    len = strlen(filename);
    filename[len - 1] = '\0';

    Market = create_storage(&number_of_products, filename);

    if (number_of_products)
    {
        print_table(Market, number_of_products);

        values[0] = get_ram;
        values[1] = get_memory;
        values[2] = get_screen_size;
        values[3] = get_weight;
        values[4] = get_price;
        values[5] = get_camera_resolution;

        menu(Market, &number_of_products, values);
    }
}

```

```

        remove_storage(Market, number_of_products);
    } else
    {
        printf("something going wrong😞");
    }
    return 0;
}

Smartphone **create_storage(int *n, char *source_file_name)
{
    Smartphone **storage;
    FILE *source;
    char tmp_str[MAX_STR_IN_FILE_LEN];
    int i;

    source = fopen(source_file_name, "r");
    if (source != NULL)
    {
        *n = 0;
        while ((fgets(tmp_str, MAX_STR_IN_FILE_LEN, source)) != NULL) (*n)++;
        rewind(source);
        storage = (Smartphone **) malloc(*n * sizeof(Smartphone *));
        if (storage != NULL)
        {
            for (i = 0; i < *n; i++)
                add_new_position(storage, &i, source);
            return storage;
        }
    } else
    {
        printf("Check the existence of the file\n");
        return 0;
    }
}

void add_new_position(Smartphone **storage, int *index, FILE *file)
{
    char tmp_str[MAX_STR_IN_FILE_LEN];
    int len;
    int cnt;
    int i;
    fgets(tmp_str, MAX_STR_IN_FILE_LEN, file);
    len = strlen(tmp_str);
    tmp_str[len - 1] = '\0';
    for (i = 0, cnt = 0; i < len; i++)
        if (tmp_str[i] == ';')
            cnt++;
    if (cnt == 7)
    {
        if (file == stdin)
            resize_storage(&storage, index); // index incremented
        storage[*index] = malloc(sizeof(Smartphone));
        set_values(storage[*index], tmp_str);
    } else

```



```

{
    printf("string is not in the line with format: %s\n", tmp_str);
}
}

void resize_storage(Smartphone ***storage, int *size)
{
    *storage = realloc(*storage, sizeof(Smartphone *) * (*size));
}

void set_values(Smartphone *smartphone, char *str)
{
    char *cameras;
    cameras = malloc(MAX_MODEL_NAME_LEN * sizeof(char));
    smartphone->model = malloc(sizeof(char) * MAX_MODEL_NAME_LEN);
    smartphone->brand = malloc(sizeof(char) * MAX_MODEL_NAME_LEN);
    sscanf(str, "%[^;];%[^;];%d;%d;%f;%f;%f;%f;%[^;]", smartphone->model, smartphone->brand,
        &(smartphone->ram), &(smartphone->memory), &(smartphone->screen_size),
        &(smartphone->weight), &(smartphone->price), cameras);
    split_camera_resolution(smartphone, cameras);
    free(cameras);
}

void split_camera_resolution(Smartphone *smartphone, char *str)
{
    int i, j, k;
    char *tmp_str;
    int len;
    int number;
    len = strlen(str);
    for (i = 0, number = 1; i < len; i++)
        if (str[i] == '+')
            number++;
    tmp_str = malloc(4 * sizeof(char));
    smartphone->camera_resolution = malloc(sizeof(int) * number);
    if (number != 0)
    {
        for (i = 0, j = 0, k = 0; i <= len; i++, j++)
        {
            if (str[i] != '+' && str[i] != '\0')
            {
                tmp_str[j] = str[i];
            } else
            {
                smartphone->camera_resolution[k] = atoi(tmp_str);
                strcpy(tmp_str, " ");
                k++;
                j = -1;
            }
        }
    }
    smartphone->number_of_cameras = number;
    free(tmp_str);
}

```

```

}

void print_table(Smartphone **storage, int n)
{
    int i;
    print_header();
    for (i = 0; i < n; i++)
        print(storage[i]);
}

void print_header()
{
    printf("| %-20s | %-15s | %-5s | %-5s | %-6s | %-7s | %-8s | %-17s |\n",
           "Model", "Brand", "RAM", "Storage", "Screen", "Weight", "Price", "Camera Resolution");
}

void print(Smartphone *smartphone)
{
    int i, j, k;
    char *cameras_res;
    char tmp_cam[4];

    cameras_res = malloc(smartphone->number_of_cameras * 3 * sizeof(char));
    for (i = 0, j = 0; i < smartphone->number_of_cameras; i++)
    {
        snprintf(tmp_cam, sizeof(tmp_cam), "%d", smartphone->camera_resolution[i]);
        for (k = 0; k < strlen(tmp_cam); k++, j++)
            cameras_res[j] = tmp_cam[k];
        if (i < smartphone->number_of_cameras - 1)
        {
            cameras_res[j] = '+';
            j++;
        }
    }
    printf("| %-20s | %-15s | %-3dGB | %-5dGB | %-5.2f\" | %-6.2fg | $%-7.2f | %15smp |\n",
           smartphone->model, smartphone->brand, smartphone->ram, smartphone->memory,
           smartphone->screen_size, smartphone->weight, smartphone->price, cameras_res);
    free(cameras_res);
}

void menu(Smartphone **storage, int *n, Getters *get_value)
{
    int value;
    do
    {
        puts("\nThere are some options:");
        puts("1 - sort by RAM");
        puts("2 - sort by Storage");
        puts("3 - sort by screen size");
        puts("4 - sort by weight");
        puts("5 - sort by price");
        puts("6 - sort by camera resolution");
        puts("7 - add smartphone");
    }
}

```

```

puts("8 - find element by model or brand");
puts("0 - for EXIT program");
printf("\nEnter option: ");
scanf("%d", &value);
switch (value)
{
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
    case 6:
    {
        CLS;
        sort_storage(storage, *n, get_value[value - 1]);
        print_table(storage, *n);
        getchar();
    }
    break;
    case 7:
    {
        CLS;
        fflush(stdin);
        printf("input string with new position: ");
        add_new_position(storage, n, stdin);
        *n += 1;
    }
    break;
    case 8:
    {
        getchar();
        find_element_by_str(storage, *n);
    }
    break;
    case 0:
    {
        getchar();
    }
    break;
    default:
    {
        puts("Incorrect key!");
        getchar();
    }
}
puts("Press ENTER to continue");
getchar();
CLS;
} while (value != 0);
}

void sort_storage(Smartphone **storage, int size, Getters get_value)
{

```

```

int gap, i, k;
Smartphone *tmp;
for (gap = size / 2; gap > 0; gap /= 2)
{
    for (i = gap; i < size; i++)
    {
        for (k = i; k >= gap && get_value(storage[k - gap]) > get_value(storage[k]); k -=
gap)
        {
            tmp = storage[k];
            storage[k] = storage[k - gap];
            storage[k - gap] = tmp;
        }
    }
}

float get_ram(Smartphone *smartphone)
{
    return (float) smartphone->ram;
}

float get_memory(Smartphone *smartphone)
{
    return (float) smartphone->memory;
}

float get_screen_size(Smartphone *smartphone)
{
    return smartphone->screen_size;
}

float get_weight(Smartphone *smartphone)
{
    return smartphone->weight;
}

float get_price(Smartphone *smartphone)
{
    return smartphone->price;
}

float get_camera_resolution(Smartphone *smartphone)
{
    int i;
    int max;
    max = smartphone->camera_resolution[0];
    for (i = 1; i < smartphone->number_of_cameras; i++)
        if (smartphone->camera_resolution[i] > max)
            max = smartphone->camera_resolution[i];
    return (float) max;
}

```

```

void find_element_by_str(Smartphone **storage, int size)
{
    char *string;
    int i, len, found;
    printf("input: ");
    string = malloc(sizeof(char) * MAX_MODEL_NAME_LEN);
    fgets(string, MAX_MODEL_NAME_LEN, stdin);
    len = strlen(string);
    string[len - 1] = '\0';
    found = 0;
    print_header();
    for (i = 0; i < size; i++)
    {
        if (!strcmp(string, storage[i]->model, len - 1))
        {
            print(storage[i]);
            found = 1;
        }
    }
    if (!found)
    {
        CLS;
        printf("sorry, but, smartphone not found😞\n");
    }
    free(string);
}

void remove_storage(Smartphone **storage, int size)
{
    int i;
    for (i = 0; i < size; i++)
    {
        free(storage[i]->model);
        free(storage[i]->brand);
        free(storage[i]->camera_resolution);
        free(storage[i]);
    }
    free(storage);
}

```

Примеры выполнения программы

Пример 1:

```
Input filename: Data.csv

| Model          | Brand   | RAM | Storage | Screen | Weight | Price  | Camera Resolution |
| iPhone11       | Apple   | 4 GB | 128 GB | 6.06 " | 194.00g | $599.00 | 12mp |
| iPhone12       | Apple   | 4 GB | 256 GB | 6.10 " | 215.00g | $799.00 | 12mp |
| Galaxy S20      | Samsung | 6 GB | 128 GB | 6.20 " | 163.00g | $699.00 | 64mp |
| OnePlus 8       | OnePlus | 8 GB | 256 GB | 6.55 " | 180.00g | $699.00 | 48mp |
| Pixel 5         | Google  | 8 GB | 128 GB | 6.00 " | 151.00g | $699.00 | 12mp |
| Xperia 1 II     | Sony    | 8 GB | 256 GB | 6.50 " | 181.00g | $1099.00 | 12+12+12+12mp |
| Redmi Note 9    | Xiaomi  | 4 GB | 64 GB | 6.53 " | 199.00g | $229.00 | 48+8+2+2mp |
| Mate 30 Pro     | Huawei  | 8 GB | 256 GB | 6.53 " | 198.00g | $1099.00 | 40+40+8mp |
| LG Velvet       | LG       | 6 GB | 128 GB | 6.80 " | 180.00g | $699.00 | 48+8+5mp |
| Mi 10           | Xiaomi  | 8 GB | 256 GB | 6.67 " | 208.00g | $799.00 | 108+13+2+2mp |
| Galaxy A51      | Samsung | 6 GB | 128 GB | 6.50 " | 172.00g | $349.00 | 48+12+5+5mp |
| Motorola Edge   | Motorola | 6 GB | 256 GB | 6.70 " | 188.00g | $699.00 | 64+16+8mp |
| Nokia 9 PureView | Nokia  | 6 GB | 128 GB | 5.99 " | 172.00g | $699.00 | 12+12+12+12+1mp |

There are some options:
1 - sort by RAM
2 - sort by Storage
3 - sort by screen size
4 - sort by weight
5 - sort by price
6 - sort by camera resolution
7 - add smartphone
8 - find element by model or brand
0 - for EXIT program

Enter option: 6

| Model          | Brand   | RAM | Storage | Screen | Weight | Price  | Camera Resolution |
| iPhone11       | Apple   | 4 GB | 128 GB | 6.06 " | 194.00g | $599.00 | 12mp |
| iPhone12       | Apple   | 4 GB | 256 GB | 6.10 " | 215.00g | $799.00 | 12mp |
| Xperia 1 II     | Sony    | 8 GB | 256 GB | 6.50 " | 181.00g | $1099.00 | 12+12+12+12mp |
| Nokia 9 PureView | Nokia  | 6 GB | 128 GB | 5.99 " | 172.00g | $699.00 | 12+12+12+12+1mp |
| Pixel 5         | Google  | 8 GB | 128 GB | 6.00 " | 151.00g | $699.00 | 12mp |
| Mate 30 Pro     | Huawei  | 8 GB | 256 GB | 6.53 " | 198.00g | $1099.00 | 40+40+8mp |
| LG Velvet       | LG       | 6 GB | 128 GB | 6.80 " | 180.00g | $699.00 | 48+8+5mp |
| OnePlus 8       | OnePlus | 8 GB | 256 GB | 6.55 " | 180.00g | $699.00 | 48mp |
| Redmi Note 9    | Xiaomi  | 4 GB | 64 GB | 6.53 " | 199.00g | $229.00 | 48+8+2+2mp |
| Galaxy A51      | Samsung | 6 GB | 128 GB | 6.50 " | 172.00g | $349.00 | 48+12+5+5mp |
| Galaxy S20      | Samsung | 6 GB | 128 GB | 6.20 " | 163.00g | $699.00 | 64mp |
| Motorola Edge   | Motorola | 6 GB | 256 GB | 6.70 " | 188.00g | $699.00 | 64+16+8mp |
| Mi 10           | Xiaomi  | 8 GB | 256 GB | 6.67 " | 208.00g | $799.00 | 108+13+2+2mp |

Press ENTER to continue

There are some options:
1 - sort by RAM
2 - sort by Storage
3 - sort by screen size
4 - sort by weight
5 - sort by price
6 - sort by camera resolution
7 - add smartphone
8 - find element by model or brand
0 - for EXIT program
```

```

Enter option: 7
input string with new position: iPhone SE 2020;Apple;3;64;4.7;148;399;12
Press ENTER to continue

```

```

There are some options:
1 - sort by RAM
2 - sort by Storage
3 - sort by screen size
4 - sort by weight
5 - sort by price
6 - sort by camera resolution
7 - add smartphone
8 - find element by model or brand
0 - for EXIT program

```

```

Enter option: 8

```

```

input: iPhone

```

Model	Brand	RAM	Storage	Screen	Weight	Price	Camera Resolution
iPhone11	Apple	4 GB	128 GB	6.06 "	194.00g	\$599.00	12mp
iPhone12	Apple	4 GB	256 GB	6.10 "	215.00g	\$799.00	12mp
iPhone SE 2020	Apple	3 GB	64 GB	4.70 "	148.00g	\$399.00	12mp

```

Press ENTER to continue

```

```

There are some options:
1 - sort by RAM
2 - sort by Storage
3 - sort by screen size
4 - sort by weight
5 - sort by price
6 - sort by camera resolution
7 - add smartphone
8 - find element by model or brand
0 - for EXIT program

```

```

Enter option: 0

```

```

Press ENTER to continue

```

```

Process finished with exit code 0

```

Пример 2:

```

"/Users/daniilmohno/Library/Mobile Documents/com~apple~CloudDocs/Student-staff/Прора/лабы по проре/2сем/Laboratory_
Input filename: Data2.csv

```

Model	Brand	RAM	Storage	Screen	Weight	Price	Camera Resolution
Galaxy S21	Samsung	8 GB	128 GB	6.20 "	171.00g	\$799.00	64+12+12mp
iPhone SE 2020	Apple	3 GB	64 GB	4.70 "	148.00g	\$399.00	12mp
OnePlus 9	OnePlus	12 GB	256 GB	6.55 "	183.00g	\$899.00	48+50+2mp
Moto G Power	Motorola	4 GB	64 GB	6.40 "	199.00g	\$249.00	16+8+2+2mp
Redmi K40	Xiaomi	6 GB	128 GB	6.67 "	196.00g	\$399.00	48+8+5mp
Pixel 6	Google	6 GB	128 GB	6.40 "	176.00g	\$699.00	50+12+16mp
Galaxy Z Flip 3	Samsung	8 GB	256 GB	6.70 "	183.00g	\$1099.00	12+12+12mp
iPhone 13 Pro	Apple	6 GB	256 GB	6.10 "	238.00g	\$1099.00	12+12+12mp
Xperia 5 III	Sony	8 GB	128 GB	6.10 "	169.00g	\$899.00	12+12+12mp
Redmi Note 10	Xiaomi	6 GB	128 GB	6.43 "	178.00g	\$279.00	48+8+2+2mp
OnePlus Nord	OnePlus	6 GB	128 GB	6.44 "	184.00g	\$399.00	48+8+5+2mp
Galaxy A72	Samsung	6 GB	128 GB	6.70 "	203.00g	\$449.00	64+8+12+5mp
iPhone 12 Mini	Apple	4 GB	64 GB	5.40 "	133.00g	\$699.00	12+12mp
Mi 11 Lite	Xiaomi	6 GB	128 GB	6.55 "	157.00g	\$349.00	64+8+5mp
Zenfone 8	Asus	8 GB	256 GB	5.90 "	169.00g	\$699.00	64+12+1mp

There are some options:

- 1 - sort by RAM
- 2 - sort by Storage
- 3 - sort by screen size
- 4 - sort by weight
- 5 - sort by price
- 6 - sort by camera resolution
- 7 - add smartphone
- 8 - find element by model or brand
- 0 - for EXIT program

Enter option: 7

input string with new position: *Galaxy M32;Samsung;6;128;6.4;196;249;64+8+2+2*

Press ENTER to continue

There are some options:

- 1 - sort by RAM
- 2 - sort by Storage
- 3 - sort by screen size
- 4 - sort by weight
- 5 - sort by price
- 6 - sort by camera resolution
- 7 - add smartphone
- 8 - find element by model or brand
- 0 - for EXIT program

Enter option: 5

Model	Brand	RAM	Storage	Screen	Weight	Price	Camera Resolution
Moto G Power	Motorola	4 GB	64 GB	6.40 "	199.00g	\$249.00	16+8+2+2mp
Galaxy M32	Samsung	6 GB	128 GB	6.40 "	196.00g	\$249.00	64+8+2+2mp
Redmi Note 10	Xiaomi	6 GB	128 GB	6.43 "	178.00g	\$279.00	48+8+2+2mp
Mi 11 Lite	Xiaomi	6 GB	128 GB	6.55 "	157.00g	\$349.00	64+8+5mp
Redmi K40	Xiaomi	6 GB	128 GB	6.67 "	196.00g	\$399.00	48+8+5mp
OnePlus Nord	OnePlus	6 GB	128 GB	6.44 "	184.00g	\$399.00	48+8+5+2mp
iPhone SE 2020	Apple	3 GB	64 GB	4.70 "	148.00g	\$399.00	12mp
Galaxy A72	Samsung	6 GB	128 GB	6.70 "	203.00g	\$449.00	64+8+12+5mp
iPhone 12 Mini	Apple	4 GB	64 GB	5.40 "	133.00g	\$699.00	12+12mp
Zenfone 8	Asus	8 GB	256 GB	5.90 "	169.00g	\$699.00	64+12+1mp
Pixel 6	Google	6 GB	128 GB	6.40 "	176.00g	\$699.00	50+12+16mp
Galaxy S21	Samsung	8 GB	128 GB	6.20 "	171.00g	\$799.00	64+12+12mp
OnePlus 9	OnePlus	12 GB	256 GB	6.55 "	183.00g	\$899.00	48+50+2mp
Xperia 5 III	Sony	8 GB	128 GB	6.10 "	169.00g	\$899.00	12+12+12mp
Galaxy Z Flip 3	Samsung	8 GB	256 GB	6.70 "	183.00g	\$1099.00	12+12+12mp
iPhone 13 Pro	Apple	6 GB	256 GB	6.10 "	238.00g	\$1099.00	12+12+12mp

Press ENTER to continue

There are some options:

- 1 - sort by RAM
- 2 - sort by Storage
- 3 - sort by screen size
- 4 - sort by weight
- 5 - sort by price
- 6 - sort by camera resolution
- 7 - add smartphone
- 8 - find element by model or brand
- 0 - for EXIT program

Enter option: 8

input: *Galaxy*

Model	Brand	RAM	Storage	Screen	Weight	Price	Camera Resolution
Galaxy M32	Samsung	6 GB	128 GB	6.40 "	196.00g	\$249.00	64+8+2+2mp
Galaxy A72	Samsung	6 GB	128 GB	6.70 "	203.00g	\$449.00	64+8+12+5mp
Galaxy S21	Samsung	8 GB	128 GB	6.20 "	171.00g	\$799.00	64+12+12mp
Galaxy Z Flip 3	Samsung	8 GB	256 GB	6.70 "	183.00g	\$1099.00	12+12+12mp

Press ENTER to continue


```
There are some options:
1 - sort by RAM
2 - sort by Storage
3 - sort by screen size
4 - sort by weight
5 - sort by price
6 - sort by camera resolution
7 - add smartphone
8 - find element by model or brand
0 - for EXIT program

Enter option: 0
Press ENTER to continue

Process finished with exit code 0
```

Выводы.

В результате выполнения работы были изучены указатели на структуры и функции в языке Си и получены практические навыки в программировании на этом языке.