МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ “ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

(ВГБОУ ВО “ВГТУ”)

Факультет информационных технологий и компьютерной безопасности

Кафедра Систем управления и информационных технологий в строительстве

КУРСОВОЙ ПРОЕКТ

по дисциплине Основы программирования и алгоритмизации

Тема: Разработка программы для работы с файловой базой данных. “Веб-камеры”

**Расчетно-пояснительная записка**

Разработал студент К.С.Боев

Подпись, дата Инициалы, фамилия

Руководитель Н.В.Акамсина

Подпись, дата Инициалы, фамилия

Нормоконтролер .

Подпись, дата Инициалы, фамилия

Защищена . . JОценка . .

дата

Воронеж

2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ “ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

(ВГБОУ ВО “ВГТУ”)

Кафедра Систем управления и информационных технологий в строительстве

ЗАДАНИЕ

на курсовой проект

по дисциплине Основы программирования и алгоритмизации

Тема: Разработка программы для работы с файловой базой данных.

“Веб-камеры”

Студент группы бТИИ-241 Боев Кирилл Сергеевич

Фамилия, имя, отчество

База данных “Веб-камеры”. Признак поиска: угол обзора, интерфейс подключения. Поля базы данных: разрешение, число мегапикселей, интерфейс, тип подключения, угол обзора, наличие микрофона. Сортировка: разрешение.

Технические условия: Windows 10, Microsoft Visual Studio 2019, Visual Studio Code, язык программирования C.

Содержание и объем проекта (графические работы, расчеты и прочее): стр, \_ рисунков, таб, приложение

Сроки выполнения этапов: 10.09.2024 - 14.12.2024

Срок защиты курсового проекта:

Руководитель Н.В.Акамсина

Подпись, дата Инициалы, фамилия

Задание принял студент К.С.Боев .

Подпись, дата Инициалы, фамилия

ЗАМЕЧАНИЯ РУКОВОДИТЕЛЯ

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 4](#_Toc1)

[ВВЕДЕНИЕ 5](#_Toc2)

[ПОСТАНОВКА ЗАДАЧИ 6](#_Toc3)

[РЕАЛИЗАЦИЯ ПРОГРАММЫ 9](#_Toc4)

[ТЕСТИРОВАНИЕ 18](#_Toc5)

[ЗАКЛЮЧЕНИЕ 26](#_Toc6)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc7)

[ПРИЛОЖЕНИЕ 28](#_Toc8)

# ВВЕДЕНИЕ

Файловая база данных (БД) - это файл, в котором хранятся записи данных, описывающих заданную предметную области. Для работы с ними необходимо реализовать программу на языке C, позволяющую создавать, изменять и выводить записи, а также сортировать объекты по признакам поиска.

Целью работы является разработка такой программы с заданным набором функций для работы с базой данных о веб-камерах.

Для достижения поставленной цели, необходимо решить следующие задачи:

1. Обеспечить программу функцией записи данных о веб-камере и ее сохранении.
2. Создать функции сохранения и загрузки файла формата .txt.
3. Реализовать удобный и понятный интерфейс взаимодействия пользователя с программой, работающей до тех пор, пока пользователь не захочет выйти из нее.
4. Добавить функции изменения существующих данных, сортировки по признакам и их вывода.

# ПОСТАНОВКА ЗАДАЧИ

Для разработки любой программы сначала анализируют поставленную задачу. Как правило, для удобства ее разбивают на подзадачи: одну большую и сложную задачу “дробят” на много, но более простых и небольших подзадач.

Суть метода пошаговой детализации состоит в анализе исходной задачи с целью выяснения подзадач. Пусть ими будут функции. Каждая функция будет выполнять свою задачу. В конце в главной функции (main()) будем вызывать “подзадачи” для выполнении операций.

Разберем функции для работы с базой данных: что будет делать каждая:

* Создавать новую запись о камере (addInfo()),
* Искать запись по заданным пользователем значениям полей (findInfo()),
* Печатать все записи, упорядоченные по определенному критерию (printInfo() - для печати несортированных данных, printSortedInfo() - для сортированных),
* Изменять записи, ранее сохраненных в файле (modInfo()),
* Добавлять произвольное количество новых записей в файл базы данных (addInfo()),
* Записывать и читать все данные из файла (save() и load()).

Для удобства пользователя работать с программой нужно оформить ее интерфейс. Сделаем это в функции main(), при этом он должен иметь следующие возможности:

* Выбор одной из функции программы,
* Ввод значений полей для новой или редактируемой записи,
* Вывод результатов выполнения функции программы,
* Вывод диагностических сообщений в ходе проверки корректности ввода данных,
* Пользовательский выбор полей записи для упорядочивания массива структур (предусмотрено упорядочивание по всем полям),
* Задание пользователем значения одного из двух поисковых полей или обоих сразу.

Таким образом, определяем подзадачи для выполнения основной цели (рис.1):

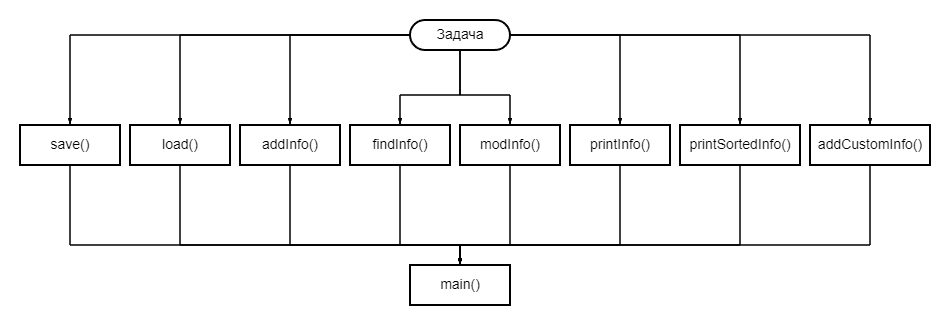


Рис.1. Разбитие задачи на подзадачи

В файл формата .txt заполним информацию о пяти веб-камерах. У каждой будут следующие характеристики:

* Разрешение
* Число мегапикселей
* Наличие микрофона
* Тип подключения
* Угол обзора

Создадим файл cameras.txt в папку “КП” и запишем данные о камерах. Это понадобится для проверки работы функции чтения и перезаписи файла в будущем. Информацию можно взять из любого интернет-магазина. Для правильного чтения данных из файла через пробел запишем каждую характеристику. Оформим запись в виде таблицы (рис.2):

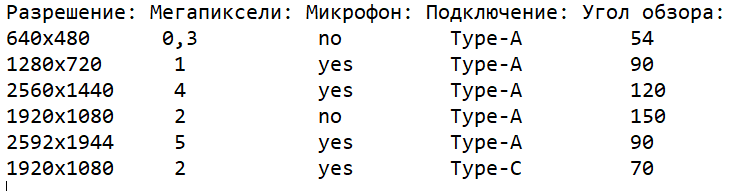


Рис.2. Запись данных о веб-камере в файл вручную

# РЕАЛИЗАЦИЯ ПРОГРАММЫ

Разбитие задач на подзадачи облегчило понимание, что нужно сделать для полноценного функционирования программы.

Продумаем алгоритм работы программы, создав блок-схему для каждой подзадачи. Будем использовать основные блочные символы (таблица 1).

Таблица 1

Основные блочные символы

|  |  |
| --- | --- |
| Обозначение | Функция |
|  | Начало, конец алгоритма |
|  | Процесс, операция, действие |
|  | Ввод, вывод |
|  | Условие, цикл.  В каждом блоке "решение" должны быть указаны вопрос, условие или сравнение, которые он определяет. Имеет один вход и несколько подписанных выходов. В случае,  если блок имеет 2 выхода (соответствует оператору ветвления), на них подписывается результат сравнения — «да/нет». Если из блока выходит большее число линий (оператор выбора), внутри него записывается имя переменной, а на выходящих дугах — значения этой переменной. |

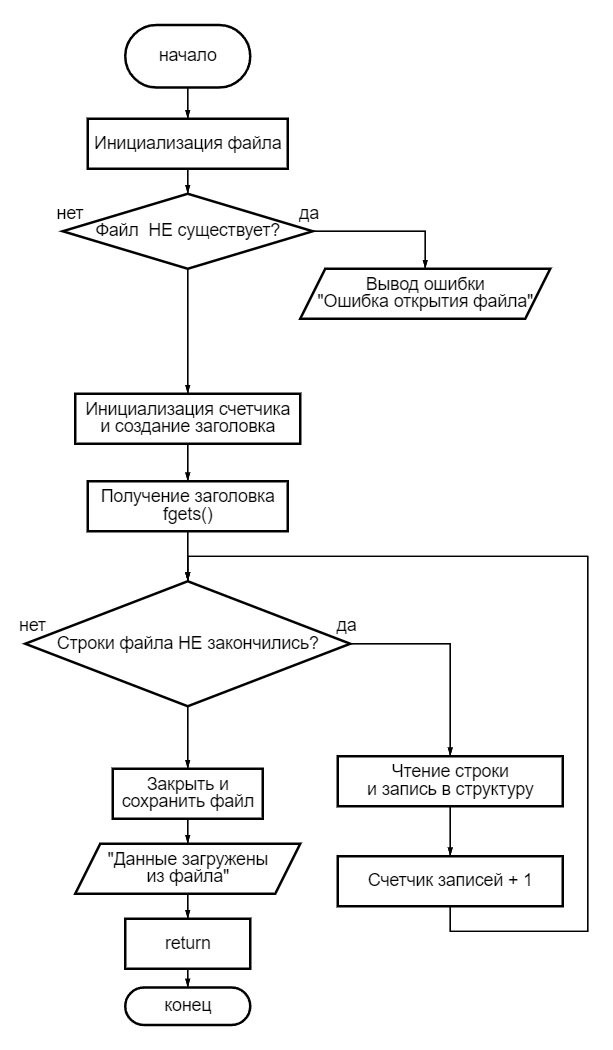


Рис.3. Загрузка данных из файла в программу (схема)

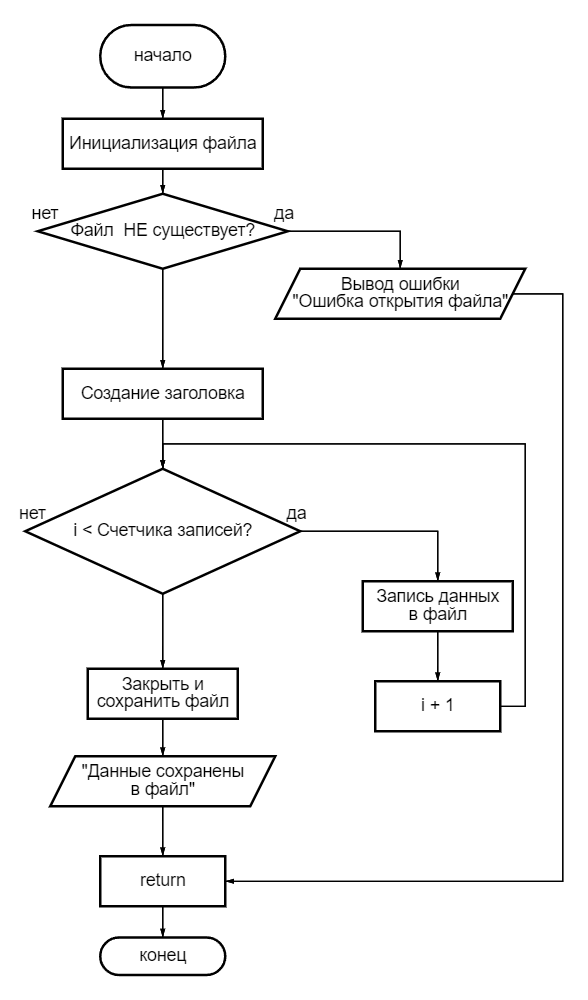


Рис.4. Сохранение файла (схема)

Рис.5. Добавление информации в БД (схема)

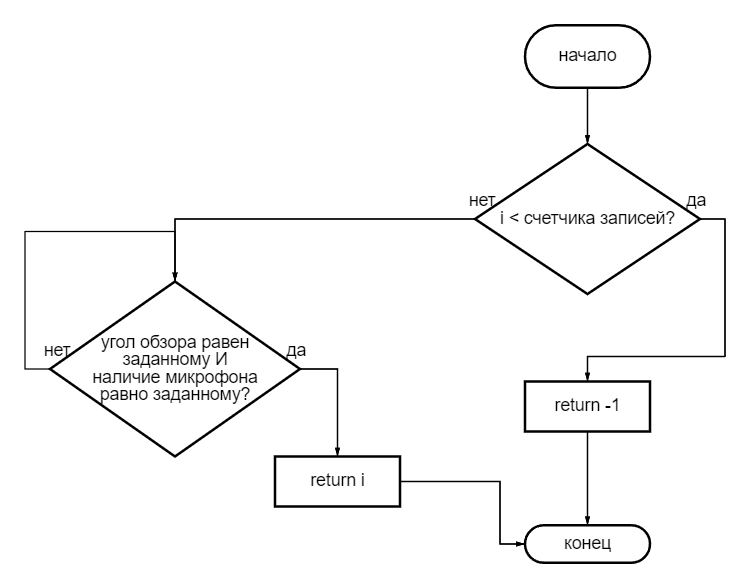
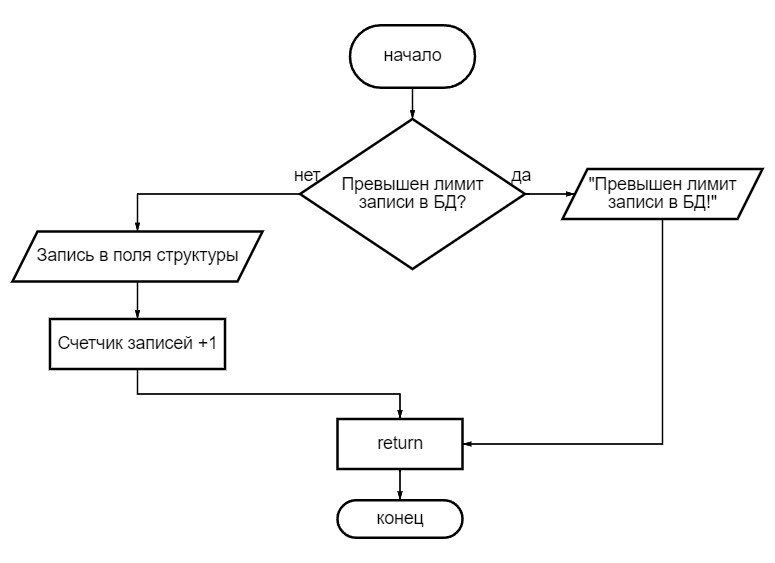


Рис.6. Поиск информации в БД (схема)

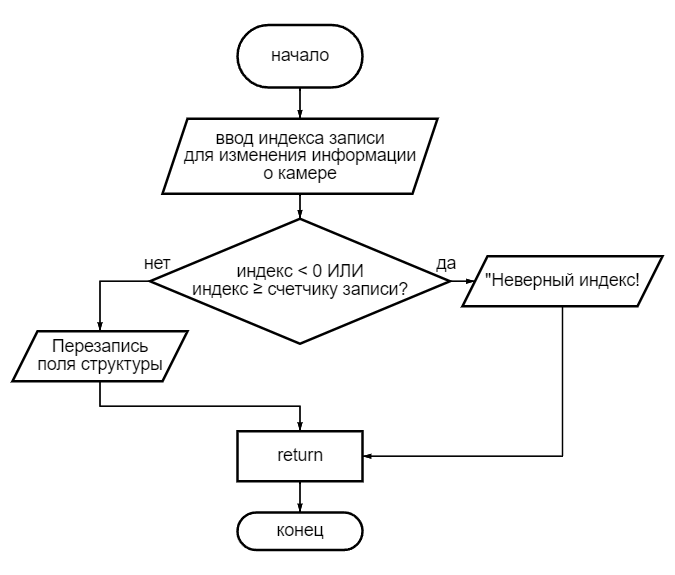


Рис.7. Изменение информации о камере в БД по ее индексу (схема)

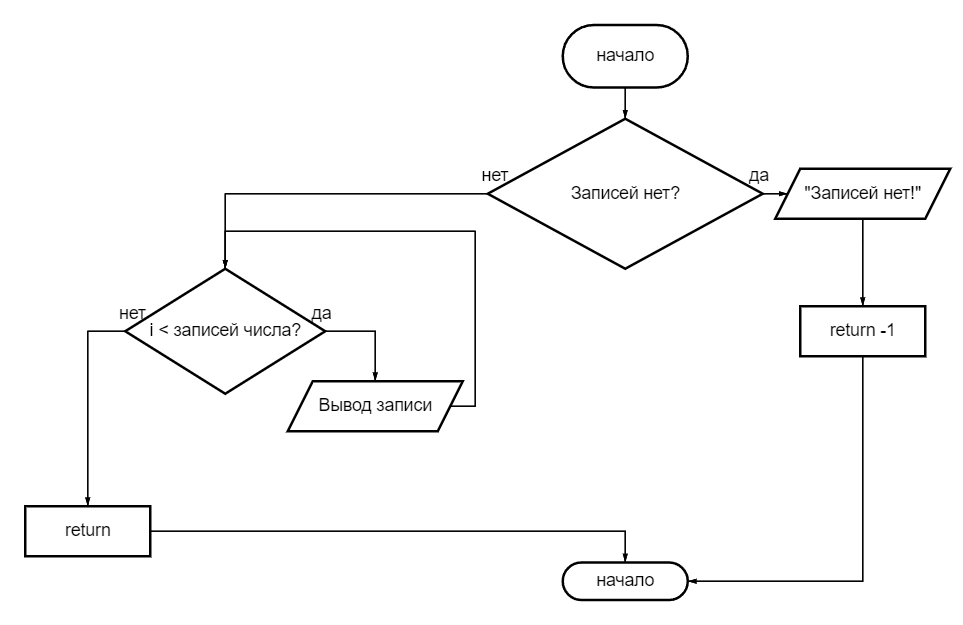


Рис.8. Вывод БД (схема)

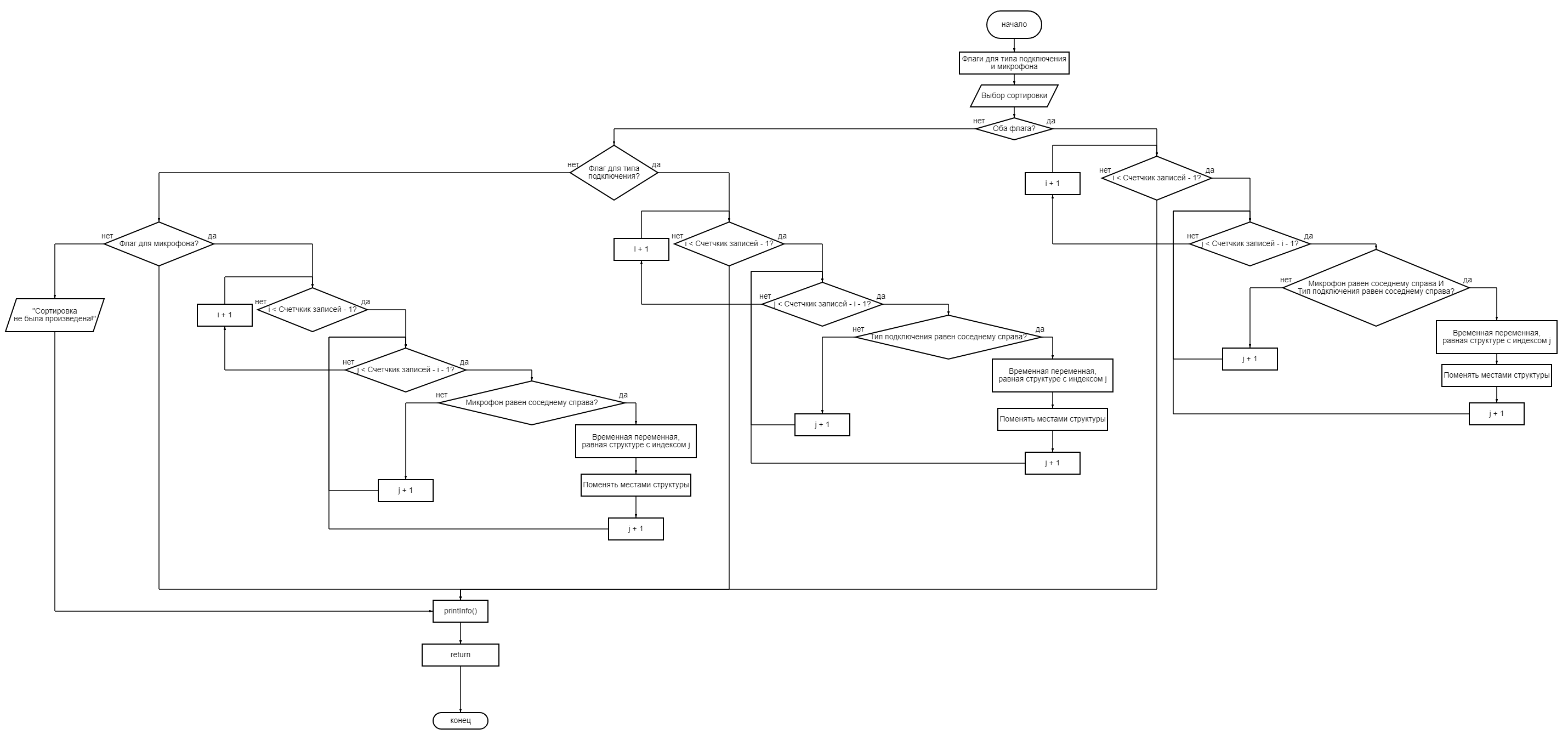


Рис.9. Сортировка БД и ее вывод (схема)

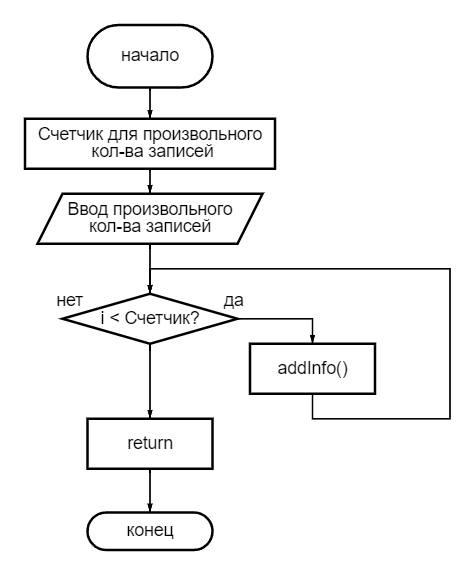
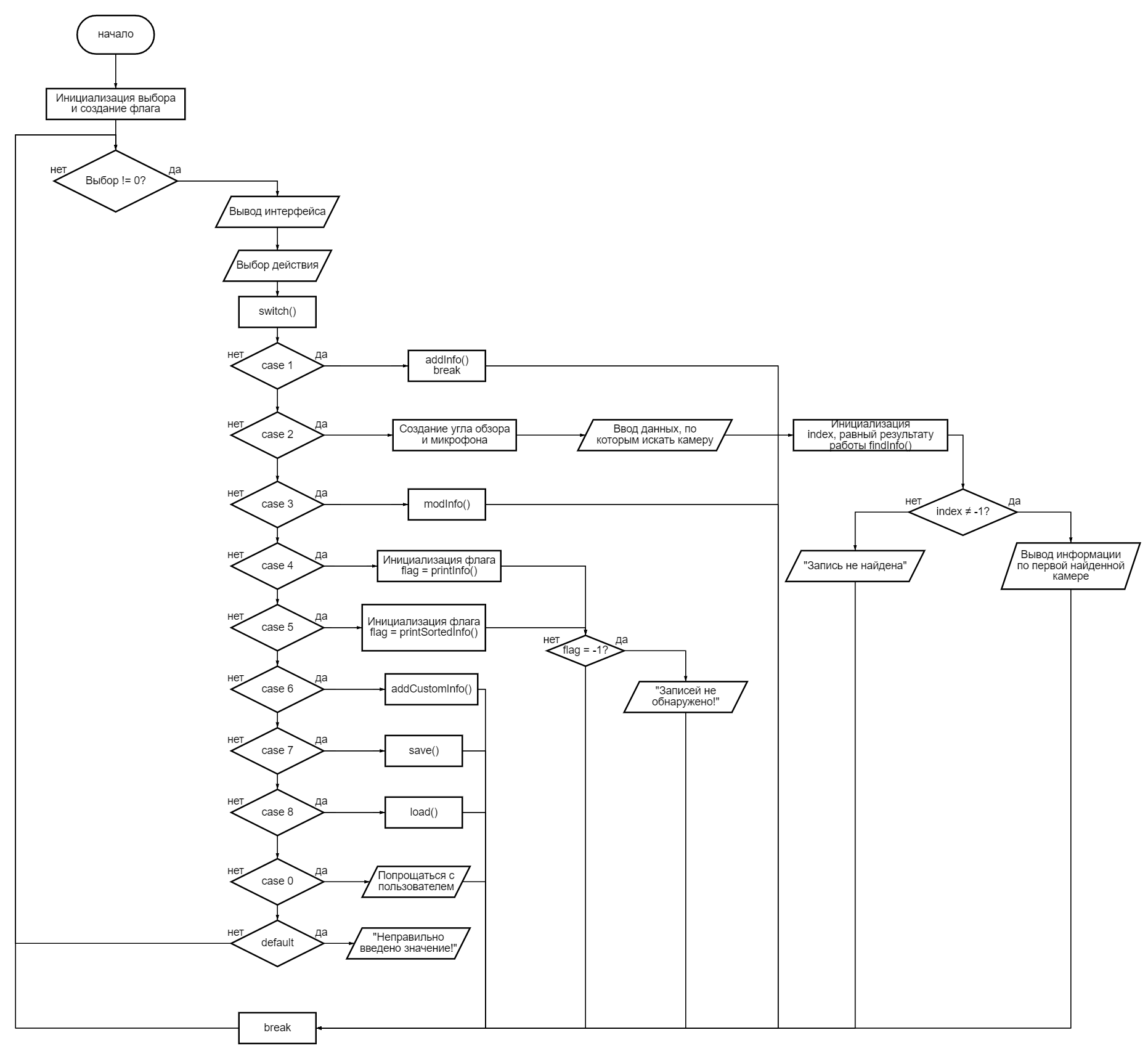


Рис.10. Добавление информации произвольного количества камер (схема)

Рис.11. Основная функция (схема)



По блок-схемам напишем код. Создадим в папке КП (обязательно в той папке, где лежит файл “cameras.txt”!) файл с расширением .C и назовем его “КП\_бТИИ241\_Боев”. В первую очередь начнем работу с глобальными переменными, записью структурного типа данных и прототипы функций. Подключим все нужные заголовочные файлы и идентификатор препроцессора \_CRT\_SECURE\_NO\_WARNINGS, отключающий дополнительную защиту от ошибок.

Теперь сделаем структурный тип данных. Структурный тип данных — это набор переменных с заданными именами и типами из базовых типов языка. Назовем его **Information**. Одна такая структура - это информация о веб-камере, значит, она будет состоять из характеристик камеры: разрешение, число мегапикселей, наличие микрофона, тип подключения и угол обзора. По условию задачи в ней должно быть минимум три различных базовых типа. Тогда для разрешения, наличия микрофона и типа подключения пусть будет тип данных *char,* для числа мегапикселей - *double*, для угла обзора - *int*.

Создадим статический массив длиной в 10000 значений с типом данных *Information* и переменную recordCount, считающую количество записей в БД. Напишем прототипы функций, которые будут использоваться в программе, и начнем писать код, опираясь на работу блок-схем (Рис.3-11).

# ТЕСТИРОВАНИЕ

После завершения написания кода проверим работоспособность программы, найдем ошибки и исправим их. Чтобы успешно найти все недочеты и проблемы, нужно составить список тестов, включающий:

* Запись файла вручную, его загрузка в программу через функцию load() и вывод БД,
* Загрузка готовой БД, запись одного файла программой и сохранение файла,
* Изменение информации по индексу камеры, сохранение файла,
* Все три вида сортировки: только по наличию микрофона, только по типу подключения, по наличию микрофона и типу подключения,
* Выдача ошибки при введения произвольного количества записей в БД большего, чем максимальная длина массива,
* Выдача ошибки, если в запись “Наличие микрофона” пользователь ввел отличное от “yes” или “no” значение,

Протестируем список по порядку. В постановке задач уже была записана ручная БД, так что можно использовать ее (см. Рис.2). Запустим программу. Перед пользователем выйдет интерфейс с выбором действия (Рис.12).

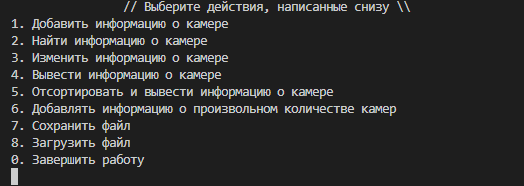


Рис.12. Интерфейс программы

Загрузим файл в программу, выбрав действие 8. В случае удачного чтения файла программа выведет “Данные загружены из файла”, в случае неудачи выдаст ошибку: “Ошибка чтения файла”. Так как файл существует и в нем записаны данные, загрузка информации будет успешна (Рис.13).



Рис.13. Уведомление пользователя об успешной работе программы

Выведем данные, используя действие 4, и сверимся с данными (Рис.14, 15):

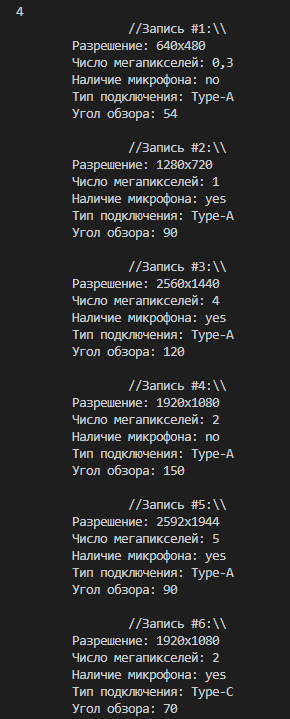


Рис.14. Вывод БД

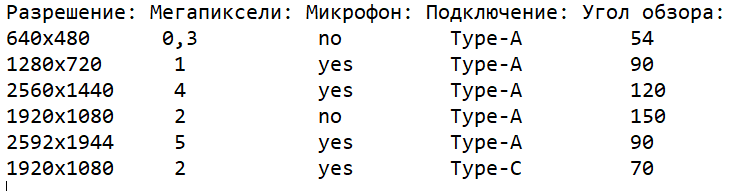


Рис.15. БД, записанная в файле “cameras.txt”

Все характеристики записей из программы совпадают с данными из файла. Первый тест прошел успешно.

Заново запустим программу и загрузим файл. На этот раз сделаем запись новой камеры. Для этого выбираем действие 1. Перед пользователем появляется интерфейс записи камеры (Рис.16).

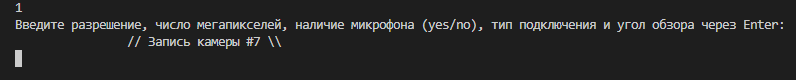


Рис.16. Интерфейс записи информации о новой камере

Запишем камеру, которая существует в реальной жизни (Рис.17). Угол обзора этой камеры равен 58 градусов. Записав все данные в интерфейс, получим (Рис.18)

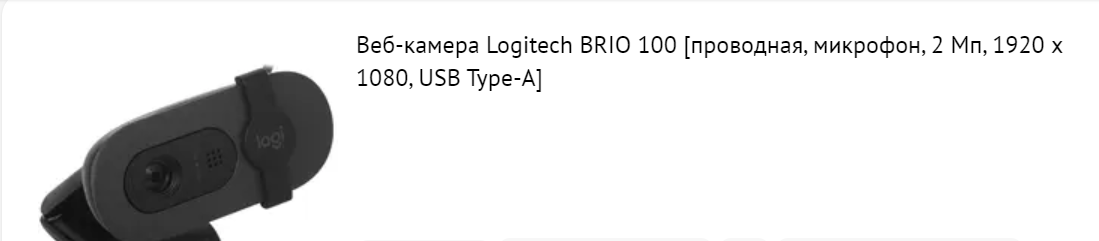


Рис.17. Информация о камере

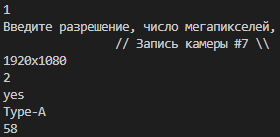


Рис.18. Запись информации о камере

Сохраним файл (действие 7). Об удачном сохранении пользователь будет уведомлен “Данные успешно сохранены”, в ином случае его предупредят об ошибке. Завершим работу, выбрав действие 0. Программа попрощается с пользователем (Рис.19).

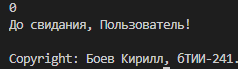


Рис.19. Прощание с пользователем

Откроем файл “cameras.txt” и проверим записи (Рис.20). Действительно, файл записал еще одну камеру.

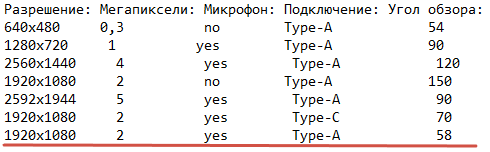


Рис.20. Проверка записи новой камеры в файле

Проверим работу изменения записи файла. Поменяем информацию о последней камере. Оставим разрешение и число мегапикселей неизменным, микрофон будет отсутствовать, тип подключения поменяем на Type-C, а угол обзора будет равен 64 градусам.

Запустим программу, загрузим файл и выберем действие 3: Изменение информации о камере. Проверим, что будет если ввести номер камеры, которой не существует. Всего камер в списке 7, введем 8. Программа выдаст об ошибке и вернет пользователя в главное меню (Рис.21).

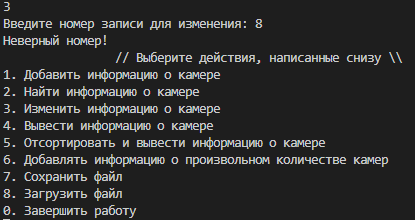


Рис.21. Вывод пользователя в главное меню в случае отсутствия номера камеры

Теперь изменим уже существующую в БД камеру. Ее номер - 7. Если он существует, появится интерфейс программы (Рис.22).

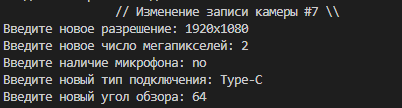


Рис.22. Интерфейс изменения камеры

Изменим данные, после чего сохраним их. Выведем все камеры (действие 4) и проверим, изменились ли данные (Рис.23).

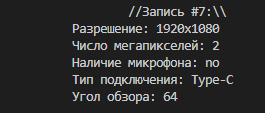


Рис.23. Проверка записи измененной камеры

Как видим, данные камеры #7 действительно поменялись.

Не завершая работу программы, проведем сортировку БД. Сначала отсортируем все камеры по наличию микрофона. Для этого в главном меню выбираем действие 5 (сортировку и вывод БД). После запуска появляются два вопроса: сортировка по типу подключения или наличию микрофона. На первый вопрос отвечаем нулем, на второй - единицей. Таким образом, программа выведет сначала камеры без микрофона, а затем с ним, при этом поменяв их нумера записей и положение в массиве (Рис.24).

Аналогичную сортировку проведем по полю “Тип подключения”. Для этого выберем в меню сортировки “Тип подключения - 1, Наличие микрофона - 0” (Рис.25).

Для сортировки и по типу подключения, и по наличию микрофона, поставим в меню сортировки единицы и в первом вопросе, и во втором (Рис.26).

Рис.24. Вывод БД, сортированного по наличию микрофона

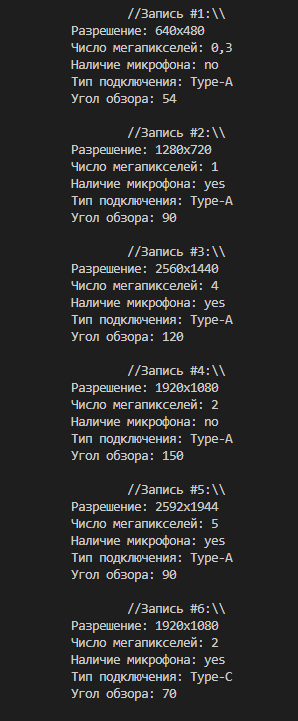
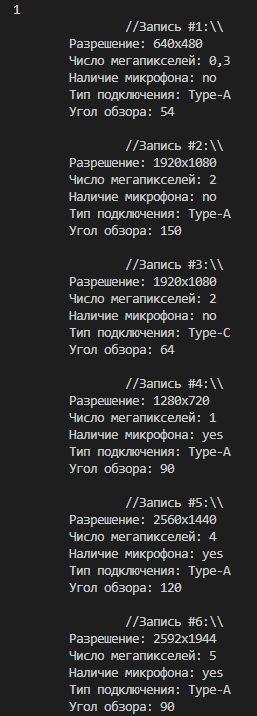


Рис.25. Вывод БД, сортированного по типу подключения

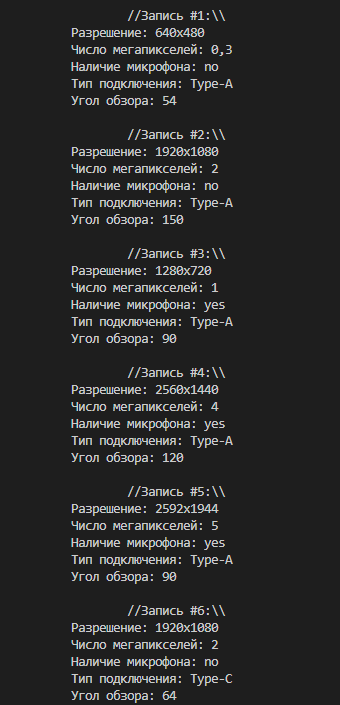


Рис.26. Вывод БД, сортированного и по типу подключения, и по наличию микрофона.

Массив может хранить в себе информацию о 10000 камер. Его размеры менять нельзя, так как он статический. Бывает такое, что пользователь хочет ввести произвольное количество камер, при этом может зайти за границу массива. Если такое происходит, нужно выдать ошибку о превышении лимита записей данных.

Будем отслеживать, если сумма количества записанных в БД камер и произвольного количества желаемых записей меньше, чем максимальная длина, то пропустим пользователя заполнять БД. Иначе выведем ошибку.

Запустим программу, загрузим файл и запишем произвольное количество записей данных для камеры. Пусть будет 9998. При записи такого числа будет превышен лимит в 10000 записей, так как уже существует 7 записей. Если запишем 3000 записей, то программа будет выполнять функцию записи до тех пор, пока эти 3000 камер не будут записаны (Рис.27).

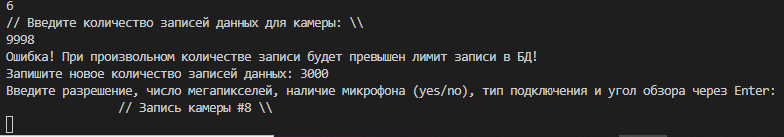


Рис.27. Запись произвольного количества камер.

При заполнении БД бывает такое, что пользователь опечатался и, например, написал вместо “yes” слово “tes”. Программа должна замечать такие ошибки и предложить исправить ошибку пользователя (Рис.28):

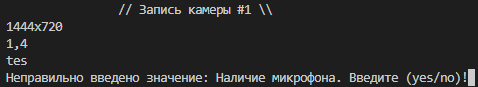


Рис.28. Обнаружение программой ошибки и предложение исправить ее

После тестирования и отладки программы можно полноценно пользоваться ей, составляя базу данных о камерах.

# ЗАКЛЮЧЕНИЕ

Таким образом, была построена постановка задачи, разбив одно большое задание на несколько небольших подзадач. Продумана работа каждой функции через блок-схемы и реализована в коде на языке программирования C. Программа отлажена и протестирована, случайных ошибок со стороны пользователя не должно быть.

Код находится в репозитории GitHub по ссылке: <https://github.com/IceDunTer/LaboratoyWorks> в ветке «main». Там же помимо кода находится файл формата .txt с базой данных.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Практикум по Си - Содержание пояснительной записки / Google Sites // Минакова О.В., Курипта О.В. - URL: https://sites.google.com/view/course-of-study1-c/кп/содержание-пояснительной-записки?authuser=0 (дата обращения: 25.12.2024).
2. Практикум по Си - Оформление пояснительной записки / Google Sites // Минакова О.В., Курипта О.В. - URL: https://sites.google.com/view/course-of-study1-c/кп/оформление-пояснительной-записки?authuser=0 (дата обращения: 25.12.2024).
3. Онлайн-редактор блок-схем | Составить блок-схему онлайн / ProgramForYou // URL: https://programforyou.ru/block-diagram-redactor (дата обращения: 25.12.2024).
4. Сильбершатц, А., Корт, Г. Ф., Сударшан, С. (2010). Концепции систем баз данных (6-е изд.). М.: Вильямс.
5. Рамакришнан, Р., Гехрке, Й. (2003). Основы систем баз данных (4-е изд.). М.: Вильямс.
6. Форта, Б. (2012). SQL за 10 минут (4-е изд.). М.: Вильямс.
7. Клеппман, М. (2017). Разработка приложений с интенсивным использованием данных. М.: Вильямс.
8. Керниган, Б. В., Ритчи, Д. М. (1988). Язык программирования Си (2-е изд.). М.: Вильямс.
9. Кинг, К. Н. (2008). Программирование на Си: современный подход (2-е изд.). М.: Вильямс.
10. Перри, Г. (2012). Программирование на Си для начинающих (3-е изд.). М.: Вильямс.

# ПРИЛОЖЕНИЕ

Структура программы:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <locale.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_LENGTH 10000

typedef struct {

char resolution[MAX\_LENGTH];

double mpixNum;

char microphone[MAX\_LENGTH];

char connectionType[MAX\_LENGTH];

int angle;

} Information;

Information database[MAX\_LENGTH];

int recordCount = 0;

void addInfo();

int findInfo(int angle, const char\* microphone);

void modInfo();

int printInfo();

int printSortedInfo();

int addCustomInfo();

void save();

void load();

int main() {

setlocale(LC\_ALL, "RUS");

int pick = 1, flag;

do {

printf("\t\t// Выберите действия, написанные снизу \\\\\n");

printf("1. Добавить информацию о камере\n");

printf("2. Найти информацию о камере\n");

printf("3. Изменить информацию о камере\n");

printf("4. Вывести информацию о камере\n");

printf("5. Отсортировать и вывести информацию о камере\n");

printf("6. Добавлять информацию о произвольном количестве камер\n");

printf("7. Сохранить файл\n");

printf("8. Загрузить файл\n");

printf("0. Завершить работу\n");

scanf("%i", &pick);

switch (pick) {

case 1:

addInfo();

break;

case 2: {

int angle;

char microphone[MAX\_LENGTH];

puts("Введите угол обзора интересующей Вас камеры:");

scanf("%i", &angle);

puts("Есть ли у камеры микрофон?");

scanf("%s", microphone);

printf("\n//Поиск информации...\\\\\n\n");

int index = findInfo(angle, microphone);

if (index != -1) {

printf("\t\t//Запись найдена:\\\\\n");

printf("\tРазрешение: %s\n", database[index].resolution);

printf("\tЧисло мегапикселей: %i\n", database[index].mpixNum);

printf("\tМикрофон: %s\n", database[index].microphone);

printf("\tТип подключения: %s\n", database[index].connectionType);

printf("\tУгол обзора: %i\n", database[index].angle);

}

else {

printf("Запись не найдена.\n");

}

break;

}

case 3:

modInfo();

break;

case 4:

flag = printInfo();

if (flag == -1) printf("Записей не обнаружено!\n\n");

break;

case 5:

flag = printSortedInfo();

if (flag == -1) printf("Записей не обнаружено!\n\n");

break;

case 7:

save();

break;

case 8:

load();

break;

case 6:

addCustomInfo();

break;

case 0:

pick = 0;

printf("До свидания, Пользователь!\n\nCopyright: Боев Кирилл, бТИИ-241.");

break;

default:

printf("Неправильно введено значение!\n\n");

}

} while (pick);

}

void addInfo() {

int flag = 0;

if (recordCount >= MAX\_LENGTH) {

printf("Превышен лимит записи в базу данных!\n\n");

return;

}

Information\* i = &database[recordCount];

printf("Введите разрешение, число мегапикселей, наличие микрофона (yes/no), тип подключения и угол обзора через Enter:\n");

printf("\t\t// Запись камеры #%i \\\\\n", recordCount + 1);

scanf("%s", i->resolution);

scanf("%lg", &i->mpixNum);

while (!flag) {

scanf("%s", i->microphone);

if (strcmp(i->microphone, "yes") == 0 || strcmp(i->microphone, "no") == 0) flag = 1;

else printf("Неправильно введено значение: Наличие микрофона. Введите (yes/no)!");

}

scanf("%s", i->connectionType);

scanf("%i", &i->angle);

recordCount++;

return;

}

int findInfo(int angle, const char\* microphone) {

for (int i = 0; i < recordCount; i++) {

if (database[i].angle == angle && strcmp(database[i].microphone, microphone) == 0) {

return i;

}

}

return -1;

}

void modInfo() {

int index;

int flag = 0;

printf("Введите номер записи для изменения: ");

scanf("%i", &index);

if (index-1 < 0 || index-1 >= recordCount) {

printf("Неверный номер!\n");

return;

}

Information\* i = &database[index-1];

printf("\t\t// Изменение записи камеры #%i \\\\\n", index);

printf("Введите новое разрешение: ");

scanf("%s", i->resolution);

printf("Введите новое число мегапикселей: ");

scanf("%lg", &i->mpixNum);

printf("Введите наличие микрофона: ");

while (!flag) {

scanf("%s", i->microphone);

if (strcmp(i->microphone, "yes") == 0 || strcmp(i->microphone, "no") == 0) flag = 1;

else printf("Неправильно введено значение: Наличие микрофона. Введите (yes/no)!");

}

printf("Введите новый тип подключения: ");

scanf("%s", i->connectionType);

printf("Введите новый угол обзора: ");

scanf("%i", &i->angle);

return;

}

int printInfo() {

if (recordCount == 0) {

printf("Записей нет!\n");

return -1;

}

for (int i = 0; i < recordCount; i++) {

printf("\t\t//Запись #%i:\\\\\n", i + 1);

printf("\tРазрешение: %s\n", database[i].resolution);

printf("\tЧисло мегапикселей: %lg\n", database[i].mpixNum);

printf("\tНаличие микрофона: %s\n", database[i].microphone);

printf("\tТип подключения: %s\n", database[i].connectionType);

printf("\tУгол обзора: %i\n\n", database[i].angle);

}

return 0;

}

int printSortedInfo() {

int flagConnectionType = 0, flagMicrophone = 0;

puts("// Сортировать по типам подключения? (1 - да/0 - нет) \\");

scanf("%i", &flagConnectionType);

puts("// Сортировать по наличию микрофона? (1 - да/0 - нет) \\");

scanf("%i", &flagMicrophone);

if (flagConnectionType && flagMicrophone) {

for (int i = 0; i < recordCount - 1; i++) {

for (int j = 0; j < recordCount - i - 1; j++) {

if ((strcmp(database[j].microphone, database[j + 1].microphone) > 0) && (strcmp(database[j].connectionType, database[j + 1].connectionType) > 0)) { //Узнать, можно ли поменять поле сортировки, т.к. сортировать по разрешению не очень удобно, хоть и работает.

Information temp = database[j];

database[j] = database[j + 1];

database[j + 1] = temp;

}

}

}

}

else if (flagConnectionType) {

for (int i = 0; i < recordCount - 1; i++) {

for (int j = 0; j < recordCount - i - 1; j++) {

if ((strcmp(database[j].connectionType, database[j + 1].connectionType) > 0)) {

Information temp = database[j];

database[j] = database[j + 1];

database[j + 1] = temp;

}

}

}

}

else if (flagMicrophone) {

for (int i = 0; i < recordCount - 1; i++) {

for (int j = 0; j < recordCount - i - 1; j++) {

if ((strcmp(database[j].microphone, database[j + 1].microphone) > 0)) {

Information temp = database[j];

database[j] = database[j + 1];

database[j + 1] = temp;

}

}

}

}

else printf("Сортировка не была произведена!");

printInfo();

return 0;

}

int addCustomInfo() {

int flag = 0;

int customCount = 0;

puts("// Введите количество записей данных для камеры: \\\\");

while (flag == 0) {

scanf("%i", &customCount);

if (recordCount+customCount < MAX\_LENGTH) flag = 1;

else printf("Ошибка! При произвольном количестве записи будет превышен лимит записи в БД!\nЗапишите новое количество записей данных: ");

}

for (int j = 0; j < customCount; j++) {

addInfo();

}

return 0;

}

void save() {

FILE \*file = fopen("cameras.txt", "w");

if (file == NULL) {

printf("Ошибка открытия файла\n");

return;

}

fprintf(file, "Разрешение: Мегапиксели: Микрофон: Подключение: Угол обзора:\n");

for (int i = 0; i < recordCount; i++) {

fprintf(file, "%s %lg %s %s %i\n", database[i].resolution, database[i].mpixNum, database[i].microphone, database[i].connectionType, database[i].angle);

}

fclose(file);

printf("Данные сохранены в файл.\n\n");

}

void load() {

FILE \*file = fopen("cameras.txt", "r");

if (file == NULL) {

printf("Ошибка открытия файла");

return;

}

recordCount = 0;

char header[1000];

fgets(header, sizeof(header), file);

while (fscanf(file, "%s %lg %s %s %i", database[recordCount].resolution, &database[recordCount].mpixNum, database[recordCount].microphone, database[recordCount].connectionType, &database[recordCount].angle) != EOF) {

recordCount++;

}

fclose(file);

printf("Данные загружены из файла.\n\n");

}