МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ 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 «Мониторы»

Расчетно-пояснительная записка

Разработал студент А. О. Анохин

Подпись, дата Инициалы, фамилия

Руководитель Н. В. Акамсина

Подпись, дата Инициалы, фамилия

Нормоконтролер Н. В. Акамсина

Подпись, дата Инициалы, фамилия

Защищена Оценка

дата

2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ 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 «Мониторы»

Студент группы бТИИ-241 Анохин А. О.

Фамилия, имя, отчество

База данных «Мониторы»; Признак поиска: Дигональ в см или дюймах, тип матрицы; Поля: Производитель, диагональ, разрешение, матрица, изогнутый экран (да+\ нет-), разъем HDMI; Вариант сортировки: По производителю.

Технические условия Windows 11, MicrosoftVisualStudio2022, язык программирования C

Содержание и объем проекта (графические работы, расчеты и прочее):

Сроки выполнения этапов анализ и постановка задачи (10.9-5.10.24); разработка пошаговой детализации программы (6.10 -11.11.24); реализация программы (11.11-5.12.24); тестирование программы (6.12-11.12.24); оформление пояснительной записки (11.12-14.12.24).

Срок защиты курсового проекта

Руководитель Н. В. Акамсина

Подпись, дата Инициалы, фамилия

Задание принял студент А. О. Анохин

Подпись, дата Инициалы, фамилия

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc10586)

[ПОСТАНОВКА ЗАДАЧИ 5](#_Toc19390)

[РЕАЛИЗАЦИЯ ПРОГРАММЫ 7](#_Toc29893)

[ТЕСТИРОВАНИЕ ПРОГРАММЫ 14](#_Toc29128)

[ЗАКЛЮЧЕНИЕ 18](#_Toc22630)

[СПИСОК ЛИТЕРАТУРЫ 19](#_Toc31179)

[ПРИЛОЖЕНИЕ 20](#_Toc2895)

# ВВЕДЕНИЕ

База данных — это организованная совокупность информации, которая хранится и управляется с помощью системы управления базами данных (СУБД). Она позволяет эффективно хранить, извлекать, изменять и управлять данными.

Базы данных необходимы для:

- Хранения информации: структурирование данных в одном месте для удобного доступа.

- Удобства работы с данными: возможность быстрого поиска и фильтрации информации.

- Безопасности: защита данных от несанкционированного доступа и потери.

- Совместной работы: обеспечение одновременного доступа к данным для многих пользователей.

- Анализа данных: помощь в принятии обоснованных решений на основе собранной информации.

Целью является написание программы, предназначенной для работы с записями данных различного типа предметной области «Мониторы», содержащимися в файле.

Согласно заданию, интерфейс созданной базы данных должен включать:

1. Добавление новой записи

2. Поиск по признаку

3. Выведение базы данных на экран

4. Сортировка данных по производителю

5. Изменение/удаление записи

6. Добавление произвольного числа записей

7. Запись и чтение из файла

Таким образом, есть 7 задач, которые необходимо реализовать.

# ПОСТАНОВКА ЗАДАЧИ

База данных будет состоять из структуры, включающей в себя записи:

1. Название монитора

2. Производитель

3. Диагональ экрана

4. Разрешение экрана

5. Тип матрицы

6. Изогнутый ли экран

7. Есть ли разъём HDMI

Данные будут оформлены в виде таблицы для удобства чтения пользователем. Стоить рассмотреть каждый элемент поподробнее.

Название — это название конкретной модели монитора. Оно обычно включает в себя производителя и разрешение, либо диагональ. Название зачастую является самой длинной записью, так что при выведении данных этому элементу будет предоставлено место в таблице длинной 30 символов.

Производитель — компания, которая произвела монитор. Названия компаний обычно недлинный, так что их место в таблице ограничится 10 символами.

Диагональ экрана — длина диагонали экрана. Может быть представлена в сантиметрах или дюймах. Ограничится 8 символами.

Разрешение экрана — количество пикселей на экране. Представлено в виде умножения длины на ширину экрана (например 1920х1080). Ограничится 10 символами.

Тип матрицы — тип матрицы экрана, т.е. по какой технологии она сделана. Не превышает 5 символов.

Изогнутый ли экран и есть ли разъём HDMI содержат информацию о том, есть ли данная характеристика у монитора или нет. Ограничим 4 символами.

Также программа будет содержать интерфейс, позволяющий взаимодействовать с базой данных по пунктам, описанным в ведении. Интерфейс будет на английском языке для предотвращения проблем с локализацией.

Необходимой задачей является создание файла для хранения базы данных. Всего будет два файла: текстовый и бинарный. Текстовый нужен для отображения базы данных в отдельном файле. Бинарный будет хранить в себе двоичную информацию о базе данных (в данном случае структуру, количество записей и массив сортировки). Именно он будет читаться программой.

Для осуществления поставленных задач будут составлены блок схемы, объясняющие, по какому алгоритму работают функции программы.

Для тестирования программы нужно взять список с несколькими мониторами, которые будут внесены в базу. Список:

1) Xiaomi\_Redmi\_23.8" | Manufacturer: Xiaomi | Diagonal: 23.8" | Resolution: 1920x1080 | Matrix: IPS | Curved screen: - | HDMI connector: +

2) Samsung\_Odyssey\_G3 | Manufacturer: Samsung | Diagonal: 24" | Resolution: 1920x1080 | Matrix: \*VA | Curved screen: - | HDMI connector: +

3) Lime\_E215Y | Manufacturer: Lime | Diagonal: 21.5" | Resolution: 1920x1080 | Matrix: \*VA | Curved screen: - | HDMI connector: +

4) ASUS\_ROG\_Strix | Manufacturer: ASUS | Diagonal: 49" | Resolution: 5120x1440 | Matrix: MVA | Curved screen: + | HDMI connector: +

5) MSI\_MPG321URX | Manufacturer: MSI | Diagonal: 32" | Resolution: 3840x2160 | Matrix: OLED | Curved screen: + | HDMI connector: +

Этих пяти мониторов достаточно для тестирования программы и её функций.

# РЕАЛИЗАЦИЯ ПРОГРАММЫ

Код программы пишется на языке C. Для начала добавим нужные библиотеки: <stdio.h>, <string.h>. Далее необходимо создать структуру, содержащую семь массивов символов (строк). Каждый массив назовём в соответствии с данными, который он будет хранить (рисунок 1).

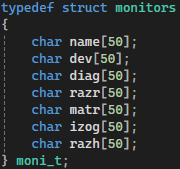


Рисунок 1 — структура

Также нужен указатель на структуру, который назовём st. Ему можно задать любой размер, это будет определять максимальное количество записей в базе данных. Для тестирования программы ограничимся 30. Далее, после объявления функций, которые будут расписаны ниже, идёт тело программы, которое будет работать по следующему алгоритму:

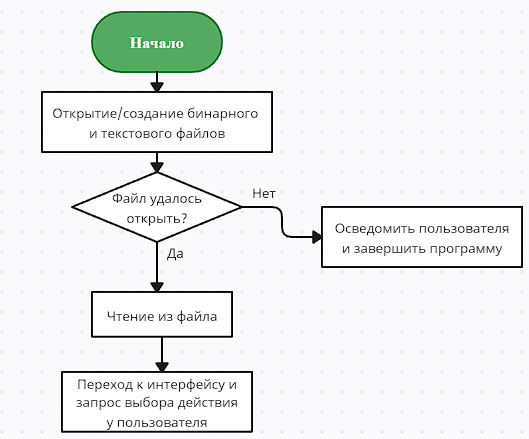


Рисунок 2 — блок-схема тела программы

Открытие файлов происходит в режимах ab+ (для чтения и записи бинарного файла) и w (для перезаписывания текстового файла). Также эти режимы позволяют создать новый файл, если такого не было найдено.

Создадим массив с наименованиями записей для удобства будущего выведения базы данных:

char name[8][20] = {"Name: ","Manufacturer: ","Diagonal: ","Resolution: ","Matrix: ","Curved screen: ","HDMI connector: "};

Также нужна переменная символьного типа, обозначающая, хочет ли пользователь продолжить использовать программу или нет:

char ans = 'y';

После каждого действия пользователю будет предоставлен выбор: продолжить программу или нет. Если ответ пользователя равен ‘y’, программа продолжается. По сути вся программа — один большой цикл, работающий, если значение ans равно ‘y’. Именно поэтому изначально переменная и равна ‘y’, чтобы программа началась.

Сам интерфейс выводит доступные действия и запрашивает номер одного из них, необходимого пользователю. Далее в дело вступает функция switch, которая, в зависимости от введённого значения, переходит к соответствующему блоку кода. После каждого из блоков будет стоять break, чтобы программа не переходила к следующему после завершения одного из блоков. Наконец, реализуем функции программы.

**Добавление записи.** Для добавления записи создадим функцию input\_moni, которая будет работать по следующему алгоритму:

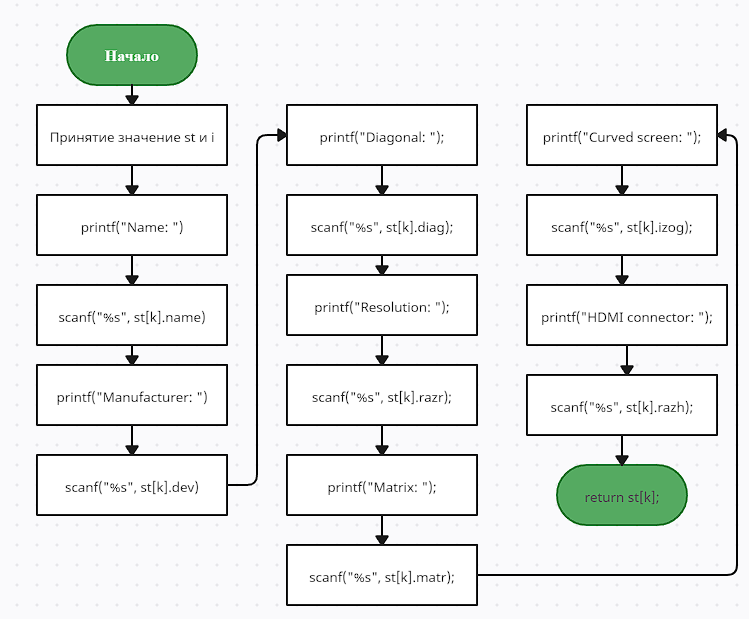


Рисунок 3 — блок-схема функции input\_moni

Проще говоря, функция будет сканировать введённые пользователем элементы записи. Стоит уточнить, что переменная i обозначает количество записей в структуре, поэтому после выполнения функции input\_moni к i будет прибавляться 1. Также элемент массива sort[i] будет приравниваться к i. Это необходимо для последующей сортировки.

**Вывод базы данных.** База данных будет выводиться не только при выполнении действия *Вывести базу*. Выводиться база данных будет и при изменении/удалении записи, чтобы пользователь смог выбрать необходимую запись. Для этого необходим цикл for, который будет повторяться столько раз, сколько записей есть в базе (т.е. число i). Вывод осуществляется в виде таблицы, каждый элемент будет длинной в столько символов, сколько было указано выше. Для этого используем printf:

printf("\n%d) %s%30s | %s%10s | %s%8s | %s%10s | %s%5s | %s%3s | %s%4s",j+1, name[0], st[j].name, name[1], st[j].dev, name[2], st[j].diag, name[3], st[j].razr, name[4], st[j].matr, name[5], st[j].izog, name[6], st[j].razh);

Это функция для вывода одной записи. Сначала выводиться номер записи, затем название монитора, производитель, диагональ, разрешение, изогнутый экран, разъём HDMI. Данные берутся из ранее созданного массива с наименованиями и из структуры. Благодаря циклу, будет выведена каждая запись.

**Поиск по признаку.** Необходимо выполнить поиск по матрице или по диагонали. Если в случае диагонали есть всего два варианта: в сантиметрах или в дюймах, то для поиска матрицы нужно запросить у пользователя ключ, по которому будет произведён поиск. Будет создана функция search, принмающая в качетсве аргументов moni\_t\*st, char name[8][20], int i. В начале пользователю предоставляется выбор осуществления поиска: по диагонали, по матрице, или по двум сразу. Далее работает тот же оператор switch, запускающий нужный блок. Для выведения соответствующих данных программа должна сравнивать строку с введённым значением, а точнее искать это значение внутри строки. Для этого отлично подходит функция strstr, сравнивающая строки. С помощью цикла проверяется каждую строку, если значение strstr не равно NULL (то есть первая строка есть во второй), то программа выводит эту строку, иначе продолжает цикл:

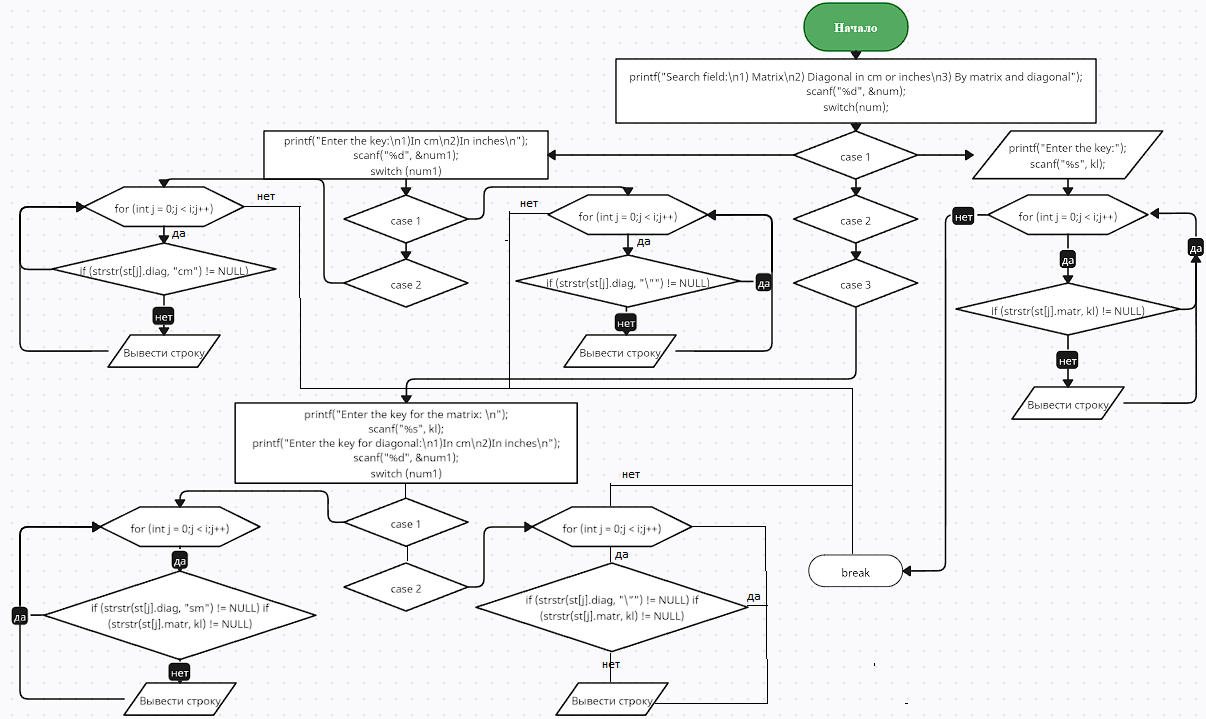


Рисунок 4 — блок-схема функции search

**Сортировка по производителю.** Пожалуй, самое сложное действие в интерфейсе из-за специфики массивов на Си. Для этого создадаётся функция sorti, берущая значения int\* sort1, int\* sort, moni\_t\* st, int i. Переменная sort1 это пока пустой массив чисел длинной 30. В самой функции введём еще три переменные:

1. int frst = 0. Эта переменная обозначает позицию первого элемента списка. Позже она будет приравниваться к другим элементам.

2. char pz[15]. Массив символов, будет содержать название производителями определённой строки в структуре.

3. char dev[50][20]. Двумерный массив символов, будет содержать имена всех производителей, которые есть в структуре.

Первым шагом переносятся все значения sort и st.dev в переменные sort1 и dev. С помощью цикла будет заменён каждый элемент переменных на значения соответствующих элементов исходных массивов. В случае с sort1 просто приравнивается к sort. Для dev потребуется функция strcpy. Сама сортировка происходит по циклу for (int j = 0;j < i;j++) и выполняется, пока все элементы не будут рассмотрены. Сначала в цикл вводится целочисленная переменная result, которая приравнивается к сравнению строк strcmp(dev[j], dev[j + 1]). Если результат больше нуля, проиходит замена первой строки на вторую, а первая встаёт на место второй. Значение j уменьшается на 1 и цикл начинается сначала. Если же результат не больше нуля, цикл просто продолжается, увеличивая j на 1.

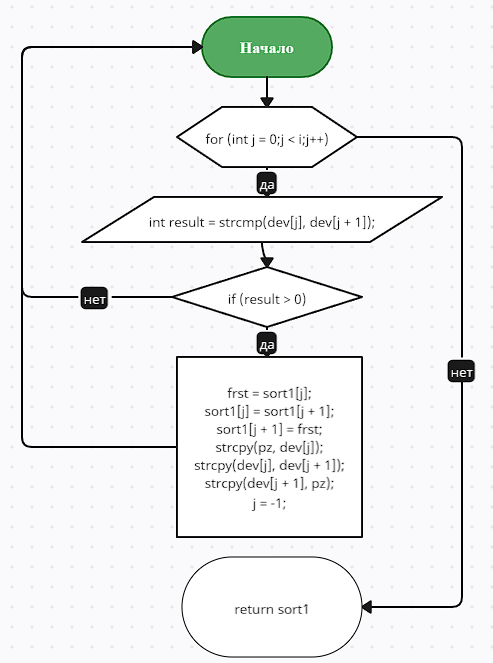


Рисунок 5 — блок-схема цикла функции sorti

Введённые переменные и массивы нужны для хранения элементов, которые менялись в списке местами. Иначе не получилось бы поставить первый элемент на место второго, ведь второй бы просто стирал его. Сама функция возвращает sort1, который хранит в себе номера элементов в порядке сортировки по производителю. Остается только вывести их описанным выше способом, только вместо привычного указателя на список в структуре использовать sort1[m] (m — переменная цикла, увеличивающаяся на 1 после каждого выполнения).

**Изменение/удаление записи.** Для изменения записи создана функция edit\_moni. Она считывает аргумент в виде структуры, запрашивает у пользователя номер записи для изменения. Затем происходит заполнение этой записи как в функции input\_moni, только вместо добавления новой записи изменяется выбранная. Функция возвращает структуру. Удаление записи происходит по той же схеме. Пользователь выбирает номер записи, затем программа заменяет его на следующую запись, следующую запись заменяет на позаследующую и так далее. Это цикл, сдвигающий все элементы, стоящие справа от выбранного, на одну позицию влево. Число i уменьшается на 1. Стоит уточнить, что перед выполнением каждой из этих операций пользователю выводится база данных, чтобы он видел номера записей перед глазами.

**Множественное добавление записей.** Самый простой для реализации пункт интерфейса. У пользователя запрашивается число, сколько записей нужно добавить. Затем функция input\_moni выполняется столько раз, сколько задал пользователь. Число i увеличивается на это число.

**Чтение из файла/запись в файл.** Чтение из файла происходит в самом начале. Хотя открываются и текстовый, и бинарный файл, читается только бинарный, так как именно он содержит всю необходимую информацию (расписано выше). С этим справляются функции fread(&st, sizeof(st), 1, fl); fread(&i, sizeof(i), 1, fl); fread(&sort, sizeof(sort), 1, fl). При сохранении данных бинарный файл открывается не в режиме ab+, как в начале, а в режиме wb+. Это позволяет перезаписать данные в файле. Текстовый открывается в том же режиме w. Для записи в бинарный файл используются функции fwrite(&st, sizeof(st), 1, fl); fwrite(&i, sizeof(i), 1, fl); fwrite(&sort, sizeof(sort), 1, fl). Запись в текстовый происходит точно так же, как и выведение базы данных на экран. Только функция printf заменяется на fprintf, а в начале аргументов добавляется указатель на файл txt. В конце пользователю выводится сообщение, что данные сохранены успешно.

Программа реализована. Необходимо протестировать её работу.

# ТЕСТИРОВАНИЕ ПРОГРАММЫ

При запуске программы пользователю выводится меню для выбора действия:

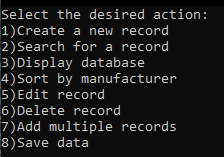


Рисунок 6 — меню интерфейса

Начнём с первого пункта и введём 1. Начинается заполение полей для новой записи:

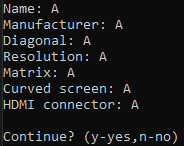


Рисунок 7 — заполнение новой записи

Далее, как и после каждого пункта, программа спрашивает, хочет ли пользователь продолжить. Для продолжения введём y. Для проверки, выводится ли база данных и добавилась ли в неё новая запись, выберем пункт номер 3:

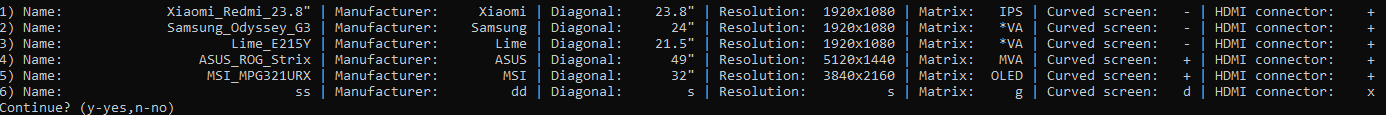


Рисунок 8 — вывод базы данных

Программа успешно добавила новую запись в базу. Далее проведём поиск по критерию. Выберем поиск по матрице и введём ключ IPS:



Рисунок 9 — поиск по матрице

Также произведём поиск по матрице и диагонали, введя ключ VA и выбрав дюймы:

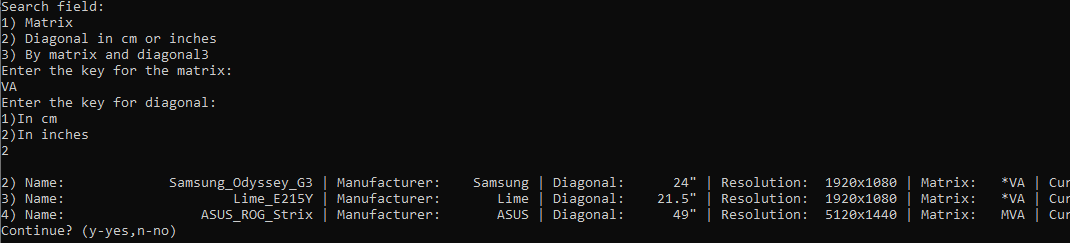


Рисунок 10 — поиск по матрице и диагонали

Поиск работает успешно. Перейдём к сортировке и введём 4:

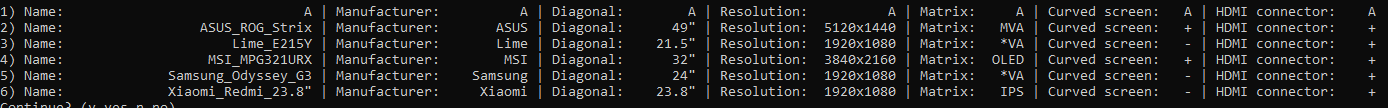


Рисунок 11 — отсортированная база данных

Как видим, программа успешно отсортировала базу, а также учла ранее введённую запись. Далее проверим изменение записи. Вводим 5 и выбираем, например, запись номер 3. Заполняем её по новой и выводим результат:

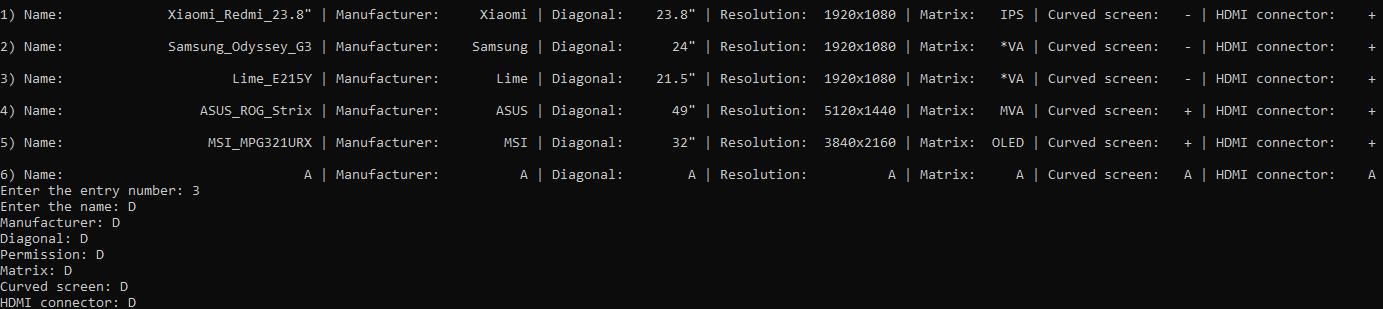


Рисунок 12 — изменение записи

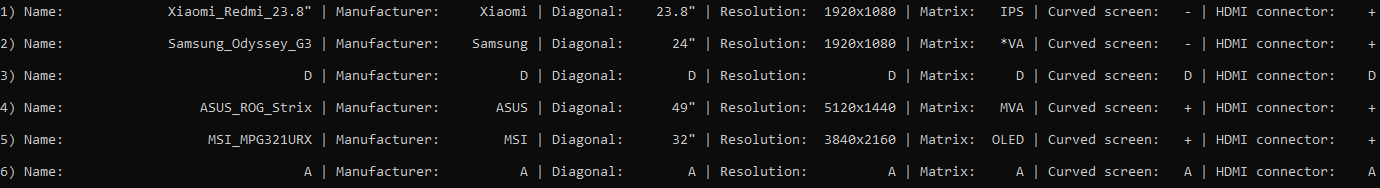


Рисунок 13 — база с изменённой записью

Изменение записи работает успешно. Проверим удаление записи. Вводим 6 и выбираем запись номер 6:

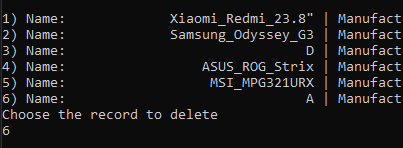


Рисунок 14 — удаление записи

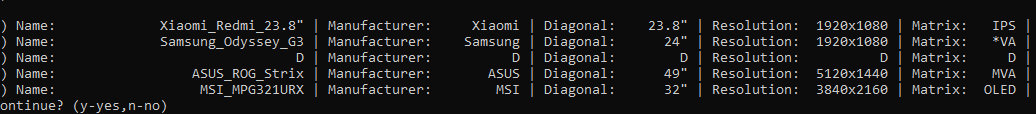


Рисунок 15 — база после удаления записи

Удаление работает успешно. Проверим добавление множества записей, добавим, например, 3 записи:

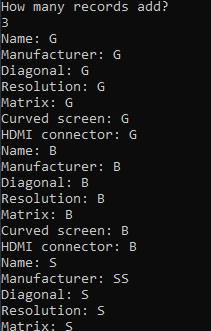


Рисунок 16 — добавление трёх записей

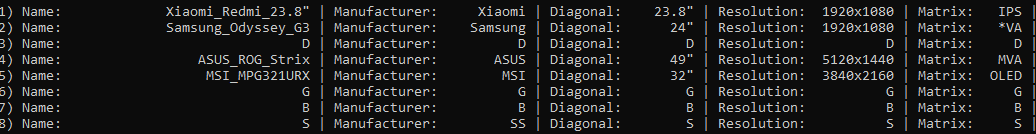


Рисунок 17 — база после добавления записей

Множественное добавление работает. Остаётся проверить сохранение в файл. Вводим 8. Дата сохранена успешно, остаётся это проверить. Запустим программу заново и осуществим, к примеру, сортировку:

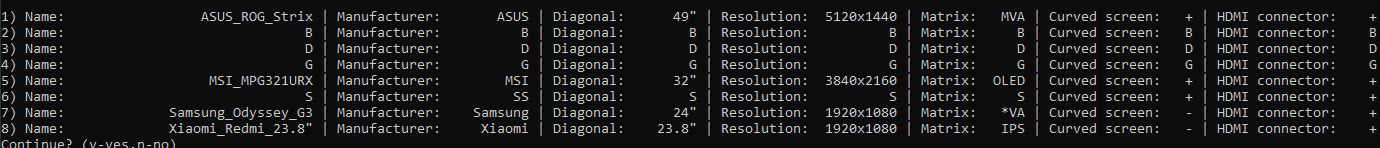


Рисунок 18 — отсортированная база из файла

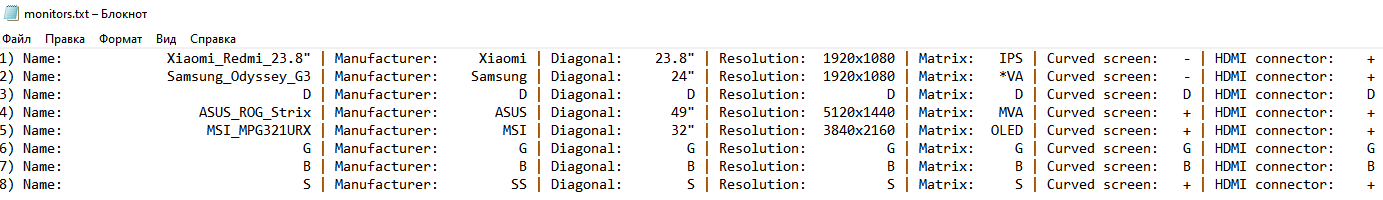


Рисунок 19 — сохранённая база в текстовом файле

Таким образом, тестирование программы не вывело ошибок. Она работает исправно и готова к использованию.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения проекта была реализована и протестирована программа для работы с файловой базой данных. Все поставленные задачи выполнены, программа работает без ошибок. Также были усвоены навыки по работе с файлами, структурами, функциями. Проект не только научил создавать собственные базы данных на языке Си, но и закрепил навыки программирования.

# СПИСОК ЛИТЕРАТУРЫ

1. Громов Ю.Ю. Программирование на языке Си: учебное пособие / Ю.Ю. Громов, С.И. Татаренко – Тамбов, 1995. – 169 с.

2. Дейтел, Х. М., Дейтел, П. Дж. Как программировать на Си++. 1999. — 1000 с.

3. Подбельский, В. В., Фомин, С. С. Программирование на языке Си: 2004. — 600 с.

4. Трой, Д. Программирование на языке Си для персонального компьютера IBM PC: 1991. — 432 с.

# ПРИЛОЖЕНИЕ

Github — https://github.com/Proffesor228/Lab\_Anokhin/tree/master

Листинг программы

#include <stdio.h>

#include <string.h>

typedef struct monitors

{

char name[50];

char dev[50];

char diag[50];

char razr[50];

char matr[50];

char izog[50];

char razh[50];

} moni\_t;

moni\_t input\_moni(moni\_t\* st, int k);

int sorti(int\* sort1,int\* sort, moni\_t\* st, int i);

moni\_t edit\_moni(moni\_t\* st);

void savetxt(FILE\* txt, moni\_t st[], char\* name, int i);

void savefl(FILE\* fl, moni\_t\* st, int\* sort, int i);

void search(moni\_t\* st, int i, char\* name, size\_t n);

void search(moni\_t\* st, char name[8][20], int i);

void main()

{

FILE\* fl;

FILE\* txt;

fl = fopen("monitors.bin", "ab+");

txt = fopen("monitors.txt", "w");

if (fl == NULL || txt == NULL) {

perror("Error occured while opening file");

return 1;

}

char ans = 'y';

char kl[40];

char name[8][20] = { "Name: ","Manufacturer: ","Diagonal: ","Resolution: ","Matrix: ","Curved screen: ","HDMI connector: " };

int num, i = 0, num1;

int sort[50], sort1[50];

char dev[50][20];

char poz[15];

int first = 0;

moni\_t st[30];

fread(&st, sizeof(st), 1, fl);

fread(&i, sizeof(i), 1, fl);

fread(&sort, sizeof(sort), 1, fl);

fclose(fl);

while (ans == 'y')

{

printf("Select the desired action:\n1)Create a new record\n2)Search for a record\n3)Display database\n4)Sort by manufacturer\n5)Edit record\n6)Delete record\n7)Add multiple records\n8)Save data");

scanf("%d", &num);

if (num > 8 || num < 1) printf("\nThe required action was not found.\n");

switch (num)

{

case 1:

input\_moni(st, i);

sort[i] = i;

i++;

break;

case 2:

search(st, name, i);

break;

case 3:

for (int j = 0;j < i;j++)

{

printf("\n%d) %s%30s | %s%10s | %s%8s | %s%10s | %s%5s | %s%3s | %s%4s",j+1, name[0], st[j].name, name[1], st[j].dev, name[2], st[j].diag, name[3], st[j].razr, name[4], st[j].matr, name[5], st[j].izog, name[6], st[j].razh);

}

break;

case 4:

sorti(sort1,sort, st, i);

for (int m = 0; m < i; m++)

{

printf("\n%d) %s%30s | %s%10s | %s%8s | %s%10s | %s%5s | %s%3s | %s%4s",m+1, name[0], st[sort1[m]].name, name[1], st[sort1[m]].dev, name[2], st[sort1[m]].diag, name[3], st[sort1[m]].razr, name[4], st[sort1[m]].matr, name[5], st[sort1[m]].izog, name[6], st[sort1[m]].razh);

}

break;

case 5:

for (int j = 0;j < i;j++)

{

printf("\n%d) %s%30s | %s%10s | %s%8s | %s%10s | %s%5s | %s%3s | %s%4s", j + 1, name[0], st[j].name, name[1], st[j].dev, name[2], st[j].diag, name[3], st[j].razr, name[4], st[j].matr, name[5], st[j].izog, name[6], st[j].razh);

}

edit\_moni(st);

break;

case 7:

printf("How many records add?\n");

scanf("%d", &num1);

for (int j = 0;j < num1;j++)

{

input\_moni(st, i);

i++;

sort[i] = i;

}

break;

case 8:

fl = fopen("monitors.bin", "wb+");

txt = fopen("monitors.txt", "w");

fwrite(&st, sizeof(st), 1, fl);

fwrite(&i, sizeof(i), 1, fl);

fwrite(&sort, sizeof(sort), 1, fl);

for (int j = 0;j < i;j++)

{

fprintf(txt, "%d) %s%30s | %s%10s | %s%8s | %s%10s | %s%5s | %s%3s | %s%4s\n", j+1, name[0], st[j].name, name[1], st[j].dev, name[2], st[j].diag, name[3], st[j].razr, name[4], st[j].matr, name[5], st[j].izog, name[6], st[j].razh);

}

printf("\nData saved succesfully\n");

break;

case 6:

for (int j = 0;j < i;j++)

{

printf("\n%d) %s%30s | %s%10s | %s%8s | %s%10s | %s%5s | %s%3s | %s%4s", j + 1, name[0], st[j].name, name[1], st[j].dev, name[2], st[j].diag, name[3], st[j].razr, name[4], st[j].matr, name[5], st[j].izog, name[6], st[j].razh);

}

printf("Choose the record to delete\n");

scanf("%d", &num1);

for (int z = num1-1;z < i;z++) st[z] = st[z+1];

i -= 1;

break;

}

getchar();

printf("\nContinue? (y-yes,n-no)\n");

scanf("%c", &ans);

}

}

moni\_t input\_moni(moni\_t\* st, int k)

{

printf("Name: ");

scanf("%s", st[k].name);

printf("Manufacturer: ");

scanf("%s", st[k].dev);

printf("Diagonal: ");

scanf("%s", st[k].diag);

printf("Resolution: ");

scanf("%s", st[k].razr);

printf("Matrix: ");

scanf("%s", st[k].matr);

printf("Curved screen: ");

scanf("%s", st[k].izog);

printf("HDMI connector: ");

scanf("%s", st[k].razh);

return st[k];

}

int sorti(int sort1[50], int sort[50], moni\_t\* st, int i)

{

int frst = 0;

char pz[15];

char dev[50][20];

for (int z = 0;z < i;z++)

{

sort1[z] = sort[z];

}

for (int z = 0;z < i;z++)

{

strcpy(dev[z], st[z].dev);

}

for (int j = 0;j < i;j++)

{

int result = strcmp(dev[j], dev[j + 1]);

if (result > 0) {

frst = sort1[j];

sort1[j] = sort1[j + 1];

sort1[j + 1] = frst;

strcpy(pz, dev[j]);

strcpy(dev[j], dev[j + 1]);

strcpy(dev[j + 1], pz);

j = -1;

}

}

return sort1;

}

moni\_t edit\_moni(moni\_t\* st)

{

int num1;

printf("Enter the entry number: ");

scanf("%d", &num1);

printf("Enter the name: ");

scanf("%s", st[num1 - 1].name);

printf("Manufacturer: ");

scanf("%s", st[num1 - 1].dev);

printf("Diagonal: ");

scanf("%s", st[num1 - 1].diag);

printf("Permission: ");

scanf("%s", st[num1 - 1].razr);

printf("Matrix: ");

scanf("%s", st[num1 - 1].matr);

printf("Curved screen: ");

scanf("%s", st[num1 - 1].izog);

printf("HDMI connector: ");

scanf("%s", st[num1 - 1].razh);

return st[num1-1];

}

void savetxt(FILE\*txt,moni\_t st[], char\* name, int i)

{

for (int j = 0;j < i;j++)

{

fprintf(txt, "%d) %s%30s | %s%10s | %s%8s | %s%10s | %s%5s | %s%3s | %s%4s\n", j, name[0], st[j].name, name[1], st[j].dev, name[2], st[j].diag, name[3], st[j].razr, name[4], st[j].matr, name[5], st[j].izog, name[6], st[j].razh);

}

printf("\nData saved succesfully\n");

}

void savefl(FILE\*fl,moni\_t\*st,int\*sort,int i)

{

fl = fopen("monitors.bin", "wb+");

fwrite(&st, sizeof(st), 1, fl);

fwrite(&i, sizeof(i), 1, fl);

fwrite(&sort, sizeof(sort), 1, fl);

printf("\nData saved succesfully\n");

}

void search(moni\_t\* st, char name[8][20], int i)

{

int num = 0;

char kl[30];

int num1 = 0;

printf("Search field:\n1) Matrix\n2) Diagonal in cm or inches\n3) By matrix and diagonal");

scanf("%d", &num);

switch (num)

{

case 1:

printf("Enter the key:");

scanf("%s", kl);

for (int j = 0; j < i; j++)

{

if (strstr(st[j].matr, kl) != NULL) printf("\n%s%30s | %s%10s | %s%8s | %s%10s | %s%5s | %s%3s | %s%4s", name[0], st[j].name, name[1], st[j].dev, name[2], st[j].diag, name[3], st[j].razr, name[4], st[j].matr, name[5], st[j].izog, name[6], st[j].razh);

}

break;

case 2:

printf("Enter the key:\n1)In cm\n2)In inches\n");

scanf("%d", &num1);

switch (num1)

{

case 1:

for (int j = 0; j < i; j++)

{

if (strstr(st[j].diag, "cm") != NULL) printf("\n%s%30s | %s%10s | %s%8s | %s%10s | %s%5s | %s%3s | %s%4s", name[0], st[j].name, name[1], st[j].dev, name[2], st[j].diag, name[3], st[j].razr, name[4], st[j].matr, name[5], st[j].izog, name[6], st[j].razh);

}

break;

case 2:

for (int j = 0; j < i; j++)

{

if (strstr(st[j].diag, "\"") != NULL) printf("\n%s%30s | %s%10s | %s%8s | %s%10s | %s%5s | %s%3s | %s%4s", name[0], st[j].name, name[1], st[j].dev, name[2], st[j].diag, name[3], st[j].razr, name[4], st[j].matr, name[5], st[j].izog, name[6], st[j].razh);

}

break;

default:

printf("\nNo records found");

break;

}

break;

case 3:

printf("Enter the key for the matrix: \n");

scanf("%s", kl);

printf("Enter the key for diagonal:\n1)In cm\n2)In inches\n");

scanf("%d", &num1);

switch (num1)

{

case 1:

for (int j = 0; j < i; j++)

{

if (strstr(st[j].diag, "sm") != NULL) if (strstr(st[j].matr, kl) != NULL) printf("\n%d) %s%30s | %s%10s | %s%8s | %s%10s | %s%5s | %s%3s | %s%4s", j + 1, name[0], st[j].name, name[1], st[j].dev, name[2], st[j].diag, name[3], st[j].razr, name[4], st[j].matr, name[5], st[j].izog, name[6], st[j].razh);

}

break;

case 2:

for (int j = 0; j < i; j++)

{

if (strstr(st[j].diag, "\"") != NULL) if (strstr(st[j].matr, kl) != NULL) printf("\n%d) %s%30s | %s%10s | %s%8s | %s%10s | %s%5s | %s%3s | %s%4s", j + 1, name[0], st[j].name, name[1], st[j].dev, name[2], st[j].diag, name[3], st[j].razr, name[4], st[j].matr, name[5], st[j].izog, name[6], st[j].razh);

}

break;

default:

printf("\nNo records found");

break;

}

}

}