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

«Серверные платформы»

**Расчетно-пояснительная записка**

Разработал студент Е.А. Валуев

Подпись, дата Инициалы, фамилия

Руководитель Н.В. Акамсина

Подпись, дата Инициалы, фамилия

Нормоконтролер Н.В. Акамсина

Подпись, дата Инициалы, фамилия

Защищена \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Воронеж 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

«Серверные платформы». Признаки поиска: «Сокет», «Максимальное

количество CPU». Признаки сортировки: «Производитель», «Форм-фактор

сервера»

Студент группы бТИИ-242

Фамилия, имя, отчество

Технические условия Windows 10, Microsoft Visual Studio 2022,

язык программирования C

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

26 стр, 17 рисунков, 2 табл, 1 приложение

Сроки выполнения этапов:

Анализ и постановка задачи (10.09.24 – 05.10.24);

Разработка пошаговой детализации программы (06.10.24 – 11.11.24);

Реализация программы (11.11.24 – 05.12.24);

Тестирование программы (06.12.24 – 11.12.24);

Оформление пояснительной записки (11.12.24 – 14.12.24).

Срок защиты курсового проекта

Руководитель Н.В. Акамсина

Подпись, дата Инициалы, фамилия

Задание принял студент Е.А. Валуев

Подпись, дата Инициалы, фамилия

**Замечания руководителя**

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc184507785)

[1 Постановка задачи 6](#_Toc184507786)

[2 Реализация программы 10](#_Toc184507787)

[3 Тестирование программы 16](#_Toc184507788)

[ЗАКЛЮЧЕНИЕ 19](#_Toc184507789)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 20](#_Toc184507790)

[ПРИЛОЖЕНИЕ А. Листинг программы 21](#_Toc184507791)

# ВВЕДЕНИЕ

Файловая база данных – файл, в котором хранятся упорядоченные записи данных, описывающих заданную предметную область. Администрирование информации, представленной таким образом довольно затруднительно, поэтому для работы с ними необходимо реализовывать собственные программы.

Целью данного курсового проекта является реализация программы для работы с файловой базой данных «Серверные платформы» на языке программирования C.

Программа должна обеспечивать понятный для пользователя функционал, с помощью которого реализуется возможность выбора функций, обязательной из которых является запись и чтение данных файла. Для проверки работоспособности программы необходимо провести тестирование всех функций.

Разработка программы проходит на языке программирования C на платформе Microsoft Visual Studio 2022.

# 1 Постановка задачи

В программе необходимо работать с данными, описывающими характеристики конкретного предмета, которые необходимо рассматривать как единое целое. Для хранения такой информации следует использовать структуры данных, которые позволяют объединить данные различных типов в единый объект.

Необходимо определиться с полями структуры – характеристиками, описывающими основной элемент базы данных, которым, в нашем случае, является серверная платформа. Описание серверной платформы может задано такими характеристиками, как:

* название платформы;
* производитель;
* форм-фактор сервера;
* сокет сервера;
* максимальное количество CPU;
* вес.

Язык си является строго типизированным языком программирования, т.е. тип каждой переменной отвечает за действия, которые можно над ними выполнять. Каждое поле должно иметь определённый тип данных. В таблице 1 представлено описание полей структуры.

Таблица – Типы данных полей структуры

|  |  |  |
| --- | --- | --- |
| Поле структуры | Тип данных | Описание типа |
| Название платформы | char\* | Строка (массив символов) |
| Производитель | char\* | Строка (массив символов) |
| Форм-фактор сервера | int | Целое число |
| Сокет сервера | char\* | Строка (массив символов) |
| Максимальное количество CPU | int | Целое число |
| Вес | double | Вещественное число |

Для хранения множества данных можно использовать массивы, однако размер массива не должен быть ограничен одним числом. Для этого необходимо динамическое распределение памяти, которое позволит выделять именно столько памяти, сколько требуется на текущий момент.

Для более удобной работы с программой необходимо определить структуру, представляющую собой базу данных, содержащую указатель на начальный элемент массива и его размер. При реализации такой структуры образуется связь между массивом и его размером.

Для организации управления и взаимодействия с программой необходимо реализовать диалог с пользователем, реализованный в виде консольного меню, где при помощи выбора пункта меню выполняется соответствующая этому пункту функция. Также необходимо использовать вложенное меню, уточняющее действие пользователя.

1. Добавление новой записи;
2. Вывод записей в консоль;
3. Поиск записей;
   1. Поиск по сокету;
   2. Поиск по количеству CPU;
4. Запись данных в файл;
5. Чтение данных из файла;
6. Сортировка записей;
   1. Сортировка по производителю;
   2. Сортировка по форм-фактору;
7. Изменение записи.

В зависимости от выбранной функции, программа может потребовать ввода дополнительных параметров, необходимых для ей выполнения. Для каждого ожидаемого ввода необходимо добавить подсказку, описывающую вводимый параметр.

После выполнения функции программа выдаст результат. Результатом может являться вывод данных в консоль, таблица или файл. Более подробное описание данных в функциях представлено в таблице 2.

Таблица – Описание данных в программе

|  |  |  |
| --- | --- | --- |
| Функция | Входные данные | Выходные данные |
| Добавление записи | Данные об объекте, вводимые пользователем | Объект с типом необходимой структуры |
| Вывод записей в консоль | Данные об объектах из базы данных | Консольная таблица объектов базы данных |
| Поиск записей | Признак и параметр поиска, объекты поиска | Таблица найденных объектов из базы данных |
| Запись данных в файл | Название файла, сохраняемые объекты | Файл с данными об объектах |
| Чтение данных из файла | Название файла | Данные об объектах, находящихся в файле |
| Сортировка записей | Признак сортировки, объекты сортировки | Таблица отсортированных объектов |
| Изменение записи | Индекс изменяемой записи, объекты | Измененные данные об указанном объекте |

Выходные данные можно представить более наглядно, используя контрольные примеры – ожидаемые от программы результаты выполнения действий.

Так, контрольный пример консольной таблицы с данными об объектах, содержащихся в базе представлен на рисунке 1.

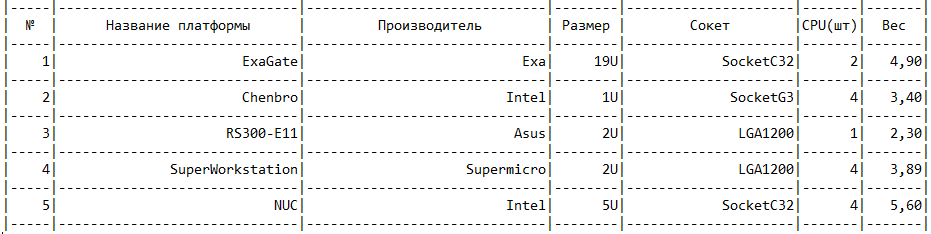


Рисунок – Контрольный пример вывода таблицы объектов базы данных

При совершении поиска в заданной базе данных, пользователю будут показаны искомые объекты. Контрольный пример поиска представлен на рисунке 2.

|  |
| --- |
|  |
| а) |
|  |
| б) |

Рисунок – Контрольный пример поиска

а) – Поиск по сокету; б) – Поиск по количеству CPU

При выполнении сортировке объектов базы данных, пользователю будет представлена таблица отсортированных объектов, контрольный пример которой изображен на рисунке 3.

|  |
| --- |
|  |
| а) |
|  |
| б) |

Рисунок – Контрольный пример сортировки

а) – Сортировка по производителю; б) – Сортировка по форм-фактору

При выполнении функций, связанных с файлом, в папке проекта создаётся, либо открывается новый текстовый файл, название которого указывает пользователь.

# 2 Реализация программы

Работа программы состоит из следующих этапов:

* выбор функции;
* выполнение выбранной функции;
* вывод результатов на экран или запись в файл.

Для использования дополнительных функций необходимо подключить к проекту соответствующие библиотеки с помощью директивы #include:

<stdio.h> – библиотека, реализующая возможности ввода и вывода.

<locale.h> – библиотека для локализации русского языка.

<string.h> – библиотека, содержащая функции для работы со строками.

<malloc.h> – библиотека, содержащая функции распределения памяти.

<stdlib.h> – библиотека дополнительных функций общего назначения.

Элементами базы данных является структура ServerPlatform (ServerPlatform\_type), в которой реализованы поля, описывающие серверную платформу:

struct ServerPlatform

{

char name[40];

char manufacturer[30];

int form\_factor;

char socket[25];

int max\_cpu;

double weight;

};

Для хранения объектов в базе данных была создана отдельная структура DataBase (DataBase\_type), в которой содержится указатель на массив и его размер:

struct DataBase

{

ServerPlatform\_type\* platforms;

int size;

};

Использование такого способа хранения позволяет связать динамически расширяемый массив с его текущим размером.

В программе реализованы функции, позволяющие корректно обрабатывать запросы пользователя. Главной функцией является «int main()». Она вызывается при запуске программы и содержит меню выбора функции, локализацию, диалог с пользователем и вызов других функций. Блок-схема функции «int main()» представлена на рисунке 4.

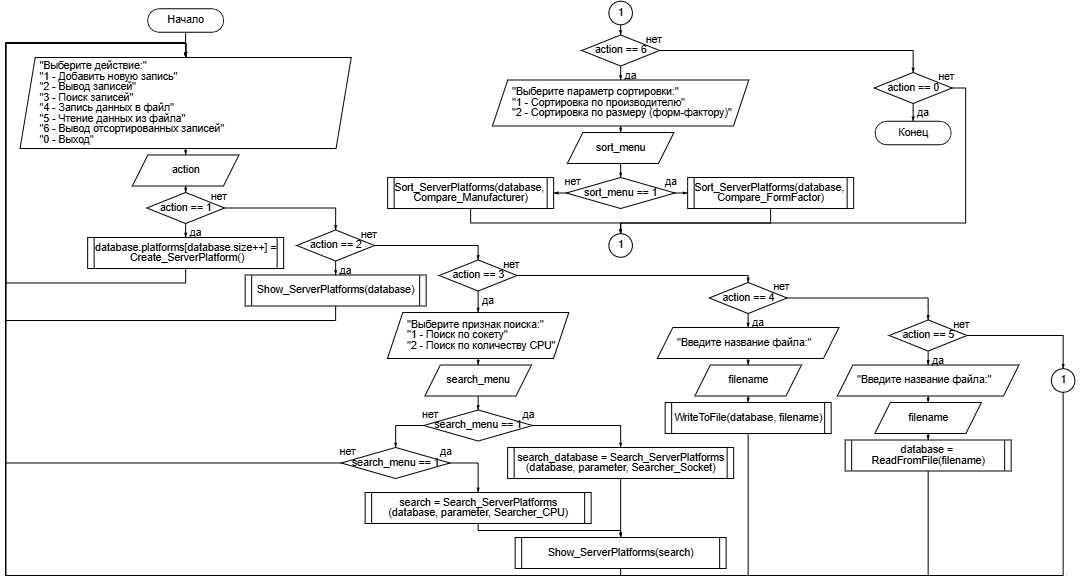


Рисунок – Блок-схема функции «int main()»

При помощи оператора switch определяется пользовательский выбор действия, а благодаря циклу имеется возможность повторения выбора (пока не будет введен «0»).

Для создания объекта, представляющего собой серверную платформу, используется функция «ServerPlatform\_type Create\_ServerPlatform()», В которой реализован последовательный ввод поле структуры, передающихся в основную функцию в качестве возвращаемого параметра – заполненной структуры. Блок-схема функции изображена на рисунке 5.

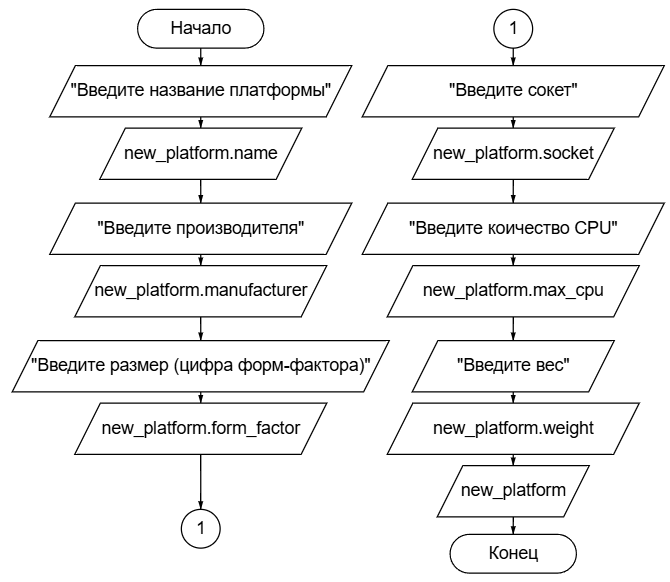


Рисунок 5 – Блок-схема функции «ServerPlatform\_type Create\_ServerPlatform()»

Вывод объектов, содержащихся в базе данных реализован в функции «void Show\_ServerPlatforms(DataBase\_type database)». Функция принимает единственный параметр – базу данных database, которую необходимо вывести, и ничего не возвращает. Блок-схема функции представлена на рисунке 6.

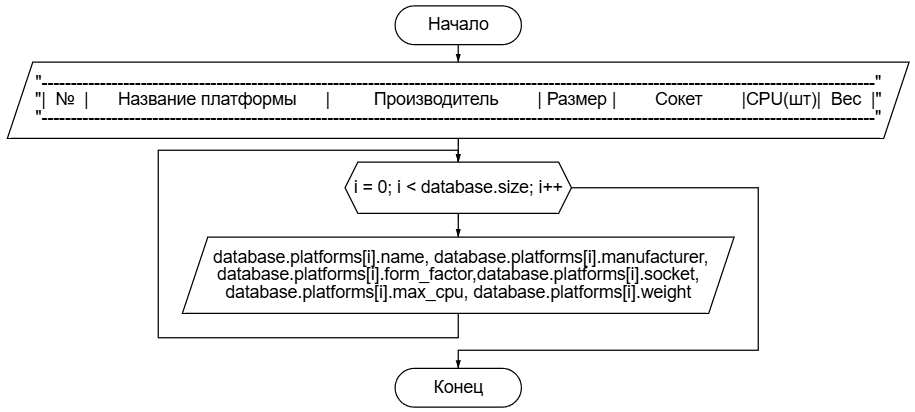


Рисунок 6 – Блок-схема функции «void Show\_ServerPlatforms(DataBase\_type database)»

Поиск объектов в базе данных реализован в функции «DataBase\_type Search\_ServerPlatforms(DataBase\_type database, char\* parameter, int(\*Searcher)(ServerPlatform\_type, char\*))». При помощи функции проверки поиска Searcher, передаваемой в качество параметра, проверяется база данных database, на совпадение с параметром поиска parameter. Функция возвращает структура типа DataBase\_type, содержащую найденные записи. Блок-схема функции представлена на рисунке 7.

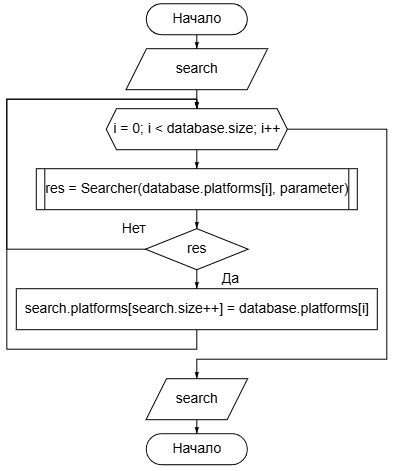


Рисунок 7 – Блок-схема функции «DataBase\_type Search\_ServerPlatforms(DataBase\_type database, char\* parameter, int(\*Searcher)(ServerPlatform\_type, char\*))»

Для поиска реализованы две функции проверки поиска: «int Searcher\_Socket(ServerPlatform\_type platform, char\* search\_socket)» и «int Searcher\_CPU(ServerPlatform\_type platform, char\* search\_cpu)». Обе функции можно передать в функцию поиска, однако они отличаются по признакам сравнения, что можно определить на блок-схемах, изображенных на рисунке 8.

|  |  |
| --- | --- |
|  |  |
| а) | б) |

Рисунок – Блок-схема функций поиска, передаваемых по указателю

а) – «int Searcher\_Socket(ServerPlatform\_type platform, char\* search\_socket)»; б) – «int Searcher\_CPU(ServerPlatform\_type platform, char\* search\_cpu)»

Для сортировки объектов базы данных используется функция «DataBase\_type Sort\_ServerPlatforms(DataBase\_type database, int(\*Compare)(ServerPlatform\_type, ServerPlatform\_type))», в которой при помощи функции сравнения Compare, передаваемой в параметрах, сортируется база данных database. Функция возвращает структуру DataBase\_type, в которой все объекты отсортированы в порядке возрастания. Сортировка происходит пузырьковым методом, блок-схема которой представлена на рисунке 9.

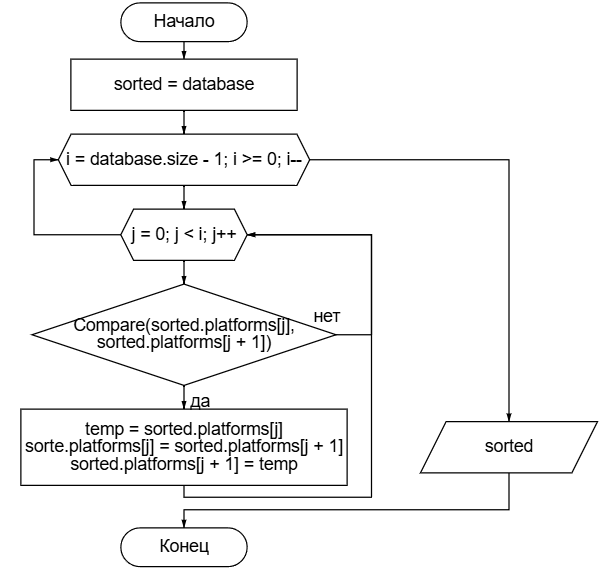


Рисунок 9 – Блок-схема функции «DataBase\_type Sort\_ServerPlatforms(DataBase\_type database, int(\*Compare)(ServerPlatform\_type, ServerPlatform\_type))»

В качестве передаваемой функции для сортировки могут быть использованы функции «int Compare\_Manufacturer(ServerPlatform\_type platform1, ServerPlatform\_type platform2)» и «int Compare\_FormFactor(ServerPlatform\_type platform1, ServerPlatform\_type platform2)». Функции отличаются по признаку сортировки. Блок-схема функций представлена на рисунке 10.

|  |  |
| --- | --- |
|  |  |
| а) | б) |

Рисунок – Блок-схема функций сравнения (компараторов)

а) – «int Compare\_Manufacturer(ServerPlatform\_type pl1, ServerPlatform\_type pl2»;

б) – «int Compare\_FormFactor(ServerPlatform\_type pl1, ServerPlatform\_type pl2»

# 3 Тестирование программы

Проведем тестирование программы с данными, приведенными в качестве контрольного примера. Для начала добавим новые записи. Добавление записей представлено на рисунке 11.

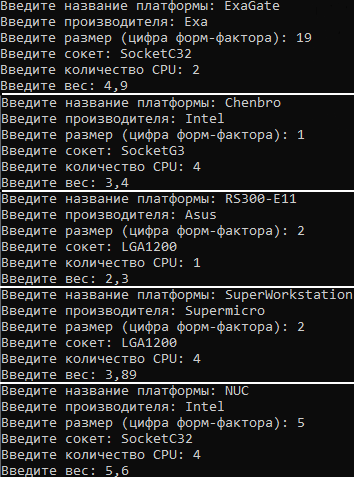


Рисунок – Добавление записей

Выведем добавленные записи. Вывод записей, содержащихся в базе данных представлен на рисунке 12.

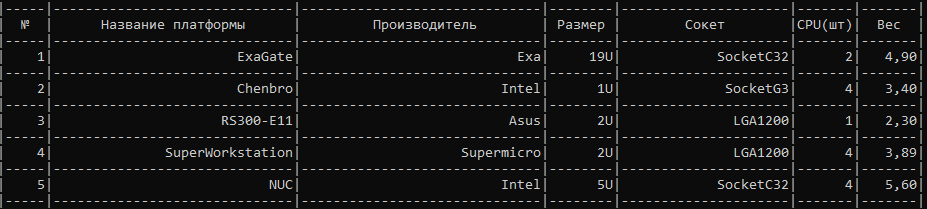


Рисунок – Вывод записей

Выполним поиск по добавленным записям. В качестве значения для поиска по сокетам используем значение SocketC32, а для поиска по количеству CPU – значение 4. Результат поиска по записям представлен на рисунке 13.

|  |
| --- |
|  |
| а) |
|  |
| б) |

Рисунок – Поиск по записям

а) – Поиск по сокетам; б) – Поиск по количеству CPU

Далее выполним сортировку записей по заданным признакам. Результат тестирования сортировки представлен на рисунке 14.

|  |
| --- |
|  |
| а) |
|  |
| б) |

Рисунок – Сортировка записей

а) – Сортировка по производителю; б) – Сортировка по форм-фактору

Далее проверим работоспособность функций по работе с файлом. Сохраним данные в файл «out.txt». Результат сохранения данных в файл представлен на рисунке 15.

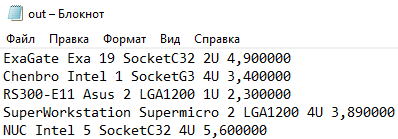


Рисунок – Сохранение в файл

Перезапустим программу и считаем данные с данного файла. Результат чтения данных из файла представлен на рисунке 16.

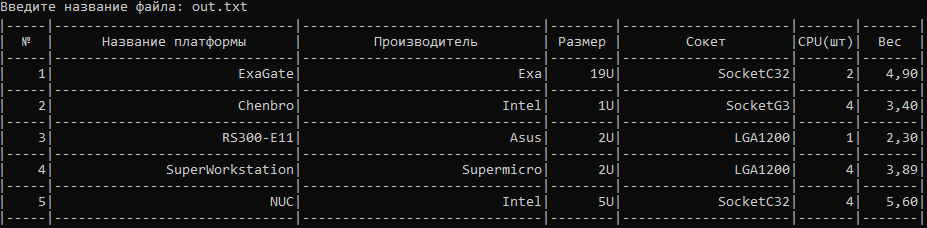


Рисунок – Чтение данных из файла

Протестируем изменение записи по её индексу. Изменим запись под номером два. Результат изменения записи представлен на рисунке 17.

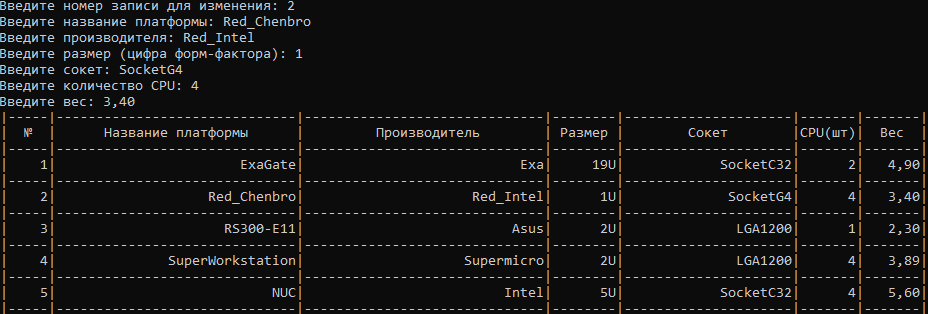


Рисунок – Изменение записи

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была реализована программа, обеспечивающая взаимодействие с файловой базой данных «Серверные платформы». Программа способна:

* добавлять записи в базу данных;
* выводить в консоль данные в виде таблицы;
* осуществлять поиск по сокетам платформ;
* осуществлять поиск по количеству CPU на платформах;
* записывать данные в текстовый файл;
* считывать данные из текстового файла;
* сортировать записи по производителю;
* сортировать записи по форм-фактору;
* изменять записи в базе данных;
* добавлять произвольное количество записей в базу данных.

Программа взаимодействует с текстовым файлом, название которого вводится пользователем. При работе с файлом в каталоге проекта автоматически создается текстовый файл. Так, пользователь может переносить всю сохранённую информацию для работы в других программах.

Благодаря использованию динамического выделения памяти программа способна работать с заранее неизвестным количеством информации, постепенно выделяя память по необходимости.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. О. В. Минакова, О. В. Курипта: «Практикум по си»: [Интернет-ресурс]: https://sites.google.com/view/course-of-study1-c/ [Дата обращения: 29.12.2023];

2. Все о Hi-Tech: [Интернет-ресурс]: <http://all-ht.ru/content.html> [Дата обращения: 26.12.2024];

3. Система вопросов и ответов Stack Overflow: [Интернет-ресурс]: <https://stackoverflow.com/> [Дата обращения: 20.12.2024];

4. Справочник по языку C: [Интернет-ресурс]: [https://learn.microsoft.com/ru-ru/cpp/c-language/](https://learn.microsoft.com/ru-ru/cpp/c-language/c-language-reference?view=%20msvc-160) [Дата обращения: 18.11.2024];

5. Форум программистов CyberForum: [Интернет-ресурс]: <https://www.cyberforum.ru/c/> [Дата обращения: 17.11.2024];

6. Онлайн-редактор блок-схем: [Сайт] – URL: <https://programforyou.ru/block-diagram-redactor>

7. Подбельский В.В., Фомин С.С. Программирование на языке Си, М.: ФиС, 1999-600 с.

8. Сайт о программировании: [Интернет-ресурс]: <https://metanit.com/> [Дата обращения: 26.12.2024]

9. Полный справочник по Си: [Интернет-ресурс]: <https://cpp.com.ru/shildt_spr_po_c/> [Дата обращения: Дата обращения: 21.12.2024]

10. И.С. Солдатенко. Основы программирования на языке Си: Учебное пособие. – Тверь: Твер. гос. ун-т, 2017. – 159 с, ISBN 978-5-7609-1229.

# ПРИЛОЖЕНИЕ А. Листинг программы

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <locale.h>

#include <malloc.h>

#include <string.h>

#include <stdlib.h>

struct ServerPlatform

{

char name[40];

char manufacturer[30];

int form\_factor;

char socket[25];

int max\_cpu;

double weight;

};

typedef struct ServerPlatform ServerPlatform\_type;

struct DataBase

{

ServerPlatform\_type\* platforms;

int size;

};

typedef struct DataBase DataBase\_type;

void Show\_ServerPlatforms(DataBase\_type database);

ServerPlatform\_type Create\_ServerPlatform();

//Поиск

DataBase\_type Search\_ServerPlatforms(DataBase\_type database, char\* parameter, int(\*Searcher)(ServerPlatform\_type, char\*));

int Searcher\_Socket(ServerPlatform\_type platform, char\* search\_socket);

int Searcher\_CPU(ServerPlatform\_type platform, char\* search\_cpu);

//Работа с файлом

DataBase\_type ReadFromFile(char\* filaname);

void WriteToFile(DataBase\_type databaase, char\* filaname);

//Сортировка

DataBase\_type Sort\_ServerPlatforms(DataBase\_type database, int(\*Compare)(ServerPlatform\_type, ServerPlatform\_type));

int Compare\_Manufacturer(ServerPlatform\_type platform\_first, ServerPlatform\_type platform\_second);

int Compare\_FormFactor(ServerPlatform\_type platform\_first, ServerPlatform\_type platform\_second);

int main()

{

setlocale(LC\_ALL, "RUS");

DataBase\_type database = { NULL, 0 };

int action;

do

{

printf("Выберите действие:\n"

"1 - Добавить новую запись\n"

"2 - Вывод записей\n"

"3 - Поиск записей\n"

"4 - Запись данных в файл\n"

"5 - Чтение данных из файла\n"

"6 - Вывод отсортированных записей\n"

"7 - Измнение записи\n"

"8 - Добавление произвольного количества новых записей\n"

"0 - Выход\n"

);

scanf("%d", &action);

getchar();

switch (action)

{

case 1://Добавление новой записи

{

database.platforms = realloc(database.platforms, sizeof(ServerPlatform\_type) \* (database.size + 1));

database.platforms[database.size++] = Create\_ServerPlatform();

break;

}

case 2://Вывод записей

{

Show\_ServerPlatforms(database);

break;

}

case 3://Поиск

{

int search\_menu;

printf("Выберите признак поиска:\n"

"1 - Поиск по сокету\n"

"2 - Поиск по количеству CPU\n");

scanf("%d", &search\_menu);

getchar();

char parameter[30];

printf("Введите параметр поиска: ");

scanf("%s", &parameter);

getchar();

DataBase\_type search\_database = { NULL, 0 };

if (search\_menu == 1)

search\_database = Search\_ServerPlatforms(database, parameter, Searcher\_Socket);

else if (search\_menu == 2)

search\_database = Search\_ServerPlatforms(database, parameter, Searcher\_CPU);

if (search\_database.platforms != NULL)

Show\_ServerPlatforms(search\_database);

free(search\_database.platforms);

break;

}

case 4://Запись в файл

{

char filename[20];

printf("Введите название файла: ");

scanf("%s", &filename);

WriteToFile(database, filename);

break;

}

case 5://Чтение с файла

{

char filename[20];

printf("Введите название файла: ");

scanf("%s", &filename);

free(database.platforms);

database = ReadFromFile(filename);

break;

}

case 6://Сортировка

{

int sort\_menu;

printf("Выберите параметр сортировки:\n"

"1 - Сортировка по производителю\n"

"2 - Сортировка по размеру (форм-фактору)\n");

scanf("%d", &sort\_menu);

getchar();

DataBase\_type sort\_database = { NULL, 0 };

if (sort\_menu == 1)

sort\_database = Sort\_ServerPlatforms(database, Compare\_Manufacturer);

else if (sort\_menu == 2)

sort\_database = Sort\_ServerPlatforms(database, Compare\_FormFactor);

if (sort\_database.platforms != NULL)

Show\_ServerPlatforms(sort\_database);

free(sort\_database.platforms);

break;

}

case 7://Изменение записей

{

int index;

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

scanf("%d", &index);

getchar();

if (index - 1 >= 0 && index - 1 < database.size)

database.platforms[index - 1] = Create\_ServerPlatform();

break;

}

case 8://Добавление произвольного количества новых записей

{

int count\_data;

printf("Введите количество записей: ");

scanf("%d", &count\_data);

getchar();

database.platforms = realloc(database.platforms, sizeof(ServerPlatform\_type) \* (database.size + count\_data));

for (int i = 0; i < count\_data; i++)

database.platforms[database.size++] = Create\_ServerPlatform();

}

}

} while (action != 0);

}

ServerPlatform\_type Create\_ServerPlatform()

{

ServerPlatform\_type new\_platform;

printf("Введите название платформы: ");

scanf("%s", &new\_platform.name);

getchar();

printf("Введите производителя: ");

scanf("%s", &new\_platform.manufacturer);

getchar();

printf("Введите размер (цифра форм-фактора): ");

scanf("%d", &new\_platform.form\_factor);

getchar();

printf("Введите сокет: ");

scanf("%s", &new\_platform.socket);

getchar();

printf("Введите количество CPU: ");

scanf("%d", &new\_platform.max\_cpu);

getchar();

printf("Введите вес: ");

scanf("%lf", &new\_platform.weight);

getchar();

return new\_platform;

}

void Show\_ServerPlatforms(DataBase\_type database)

{

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

printf("| № | Название платформы | Производитель | Размер | Сокет |CPU(шт)| Вес |\n");

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

for (int i = 0; i < database.size; i++)

{

printf("|%5d|%30s|%30s|%7dU|%21s|%7d|%7.2lf|\n", i + 1, database.platforms[i].name, database.platforms[i].manufacturer, database.platforms[i].form\_factor, database.platforms[i].socket, database.platforms[i].max\_cpu, database.platforms[i].weight);

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

}

}

DataBase\_type Search\_ServerPlatforms(DataBase\_type database, char\* parameter, int(\*Searcher)(ServerPlatform\_type, char\*))

{

DataBase\_type search\_platforms = { NULL, 0 };

for (int i = 0; i < database.size; i++)

{

if (Searcher(database.platforms[i], parameter))

{

search\_platforms.platforms = realloc(search\_platforms.platforms, sizeof(ServerPlatform\_type) \* (search\_platforms.size + 1));

search\_platforms.platforms[search\_platforms.size++] = database.platforms[i];

}

}

return search\_platforms;

}

int Searcher\_Socket(ServerPlatform\_type platform, char\* search\_socket)

{

return strcmp(platform.socket, search\_socket) == 0 ? 1 : 0;

}

int Searcher\_CPU(ServerPlatform\_type platform, char\* search\_cpu)

{

return platform.max\_cpu == atoi(search\_cpu)? 1 : 0;

}

DataBase\_type ReadFromFile(char\* filename)

{

FILE\* file\_input = fopen(filename, "r");

DataBase\_type database\_file = { NULL, 0 };

while (!feof(file\_input))

{

database\_file.platforms = realloc(database\_file.platforms, sizeof(ServerPlatform\_type) \* (database\_file.size + 1));

fscanf(file\_input, "%s %s %d %s %dU %lf\n", &database\_file.platforms[database\_file.size].name, &database\_file.platforms[database\_file.size].manufacturer, &database\_file.platforms[database\_file.size].form\_factor, &database\_file.platforms[database\_file.size].socket, &database\_file.platforms[database\_file.size].max\_cpu, &database\_file.platforms[database\_file.size].weight);

database\_file.size++;

}

return database\_file;

}

void WriteToFile(DataBase\_type database, char\* filaname)

{

FILE\* file\_output = fopen(filaname, "w");

for (int i = 0; i < database.size; i++)

fprintf(file\_output, "%s %s %d %s %dU %lf\n", database.platforms[i].name, database.platforms[i].manufacturer, database.platforms[i].form\_factor, database.platforms[i].socket, database.platforms[i].max\_cpu, database.platforms[i].weight);

fclose(file\_output);

}

DataBase\_type Sort\_ServerPlatforms(DataBase\_type database, int(\*Compare)(ServerPlatform\_type, ServerPlatform\_type))

{

DataBase\_type sorted\_database = { malloc(sizeof(ServerPlatform\_type) \* database.size), database.size };

ServerPlatform\_type temp\_platform;

for (int i = 0; i < database.size; i++)

sorted\_database.platforms[i] = database.platforms[i];

for (int i = database.size - 1; i >= 0; i--)

{

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

{

if (Compare(sorted\_database.platforms[j], sorted\_database.platforms[j + 1]) == 1)

{

temp\_platform = sorted\_database.platforms[j];

sorted\_database.platforms[j] = sorted\_database.platforms[j + 1];

sorted\_database.platforms[j + 1] = temp\_platform;

}

}

}

return sorted\_database;

}

int Compare\_Manufacturer(ServerPlatform\_type platform\_first, ServerPlatform\_type platform\_second)

{

if (strcmp(platform\_first.manufacturer, platform\_second.manufacturer) > 0)

return 1;

else if (strcmp(platform\_first.manufacturer, platform\_second.manufacturer) < 0)

return -1;

return 0;

}

int Compare\_FormFactor(ServerPlatform\_type platform\_first, ServerPlatform\_type platform\_second)

{

if (platform\_first.form\_factor > platform\_second.form\_factor)

return 1;

else if (platform\_first.form\_factor > platform\_second.form\_factor)

return -1;

return 0;

}