Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
| Информатика |
| кафедра |

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №4**

|  |
| --- |
| Управление файлами в OC GNU/LINUX |
| Тема |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  | А. Н. Пупков |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ21-16/1б, 032155832 |  |  |  | Н. А. Терентьев |
|  | номер группы, зачётной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2023

**СОДЕРЖАНИЕ**

1 Цели 3

2 Задачи 3

3 Описание варианта задания 3

4 Ход выполнения 3

4.1 Структура данных 3

4.2 Основные функции 4

4.3 Doxygen документация 5

5 Выводы 6

Список использованных источников 7

Приложение А 8

**1 Цель**

Программная реализация обработки текстовой информации, хранящейся во внешней памяти, с использованием системных вызовов низкоуровневого ввода-вывода.

**2 Задачи**

Для выполнения практической работы необходимо выполнить  
следующие задачи:

* ознакомиться с краткими теоретическими сведениями по управлению файлами в ОС GNU/Linux;
* получить у преподавателя индивидуальный вариант задания на разработку программы, позволяющей выполнять операции над файлами с записями фиксированной длины;
* используя изученные механизмы, разработать и отладить программу, выполняющую заданную работу. Обеспечить сборку программы с использованием инструментального набора GNU Autotools;
* составить общее описание результатов, инструкции по сборке и использованию программ, а также инструкцию по получению документации, сформировать архив формата tar.gz и представить на проверку с исходными текстами программы.

**3 Описание варианта задания**

Работа выполнена в соответствии с вариантом 9.

Структура данных: предприятие-изготовитель; тип; цена; производительность. Создать два запроса, позволяющих найти список предприятий с наилучшим соотношением цена/производительность и диапазон цен.

**4 Ход выполнения**

**4.1 Структура данных**

В файле structure.h была написана структура, необходимая для выполнения работы. Код структуры представлен в листинге 1.

Листинг 1 – Содержимое файла structure.h

typedef struct Enterprise Enterprise; //!< Структура предприятия

/\*!

\* \brief Предприятие-изготовитель

\*/

struct Enterprise

{

char company[MAX\_LEN]; /\*!< Предприятие изготовитель \*/

char type[MAX\_LEN]; /\*!< Тип \*/

double price; /\*!< Цена \*/

double performance; /\*!< Производительность \*/

};

**4.2 Основные функции**

В файле functions.c были реализованы функции по добавлению одного предприятия в файл, модификации предприятия в файле, удаления предприятия из файла, чтения одного предприятия из файла, отображения содержимого файла. Также были написаны две функции для нахождения предприятий с наилучшим соотношением цена/производительность в файле и диапазон цен. Работа с программой представлена на рисунке 1.

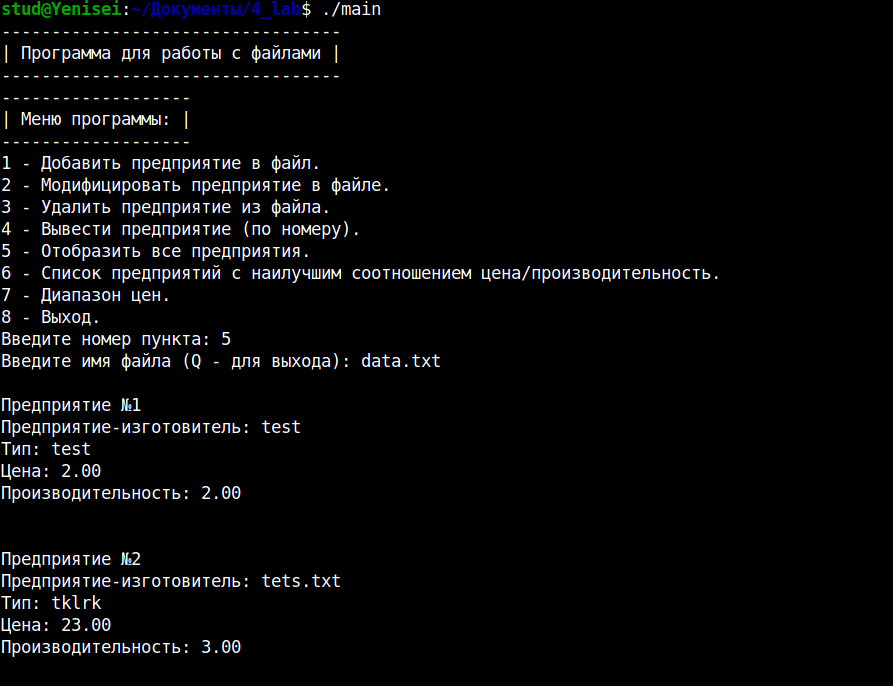


Рисунок 1 – Работа с программой

**4.3 Doxygen документация**

Все файлы содержат комментарии в стиле системы doxygen. Благодаря конфигурационному файлу была сгенерирована документация. Её внешний вид представлен на рисунке 2.

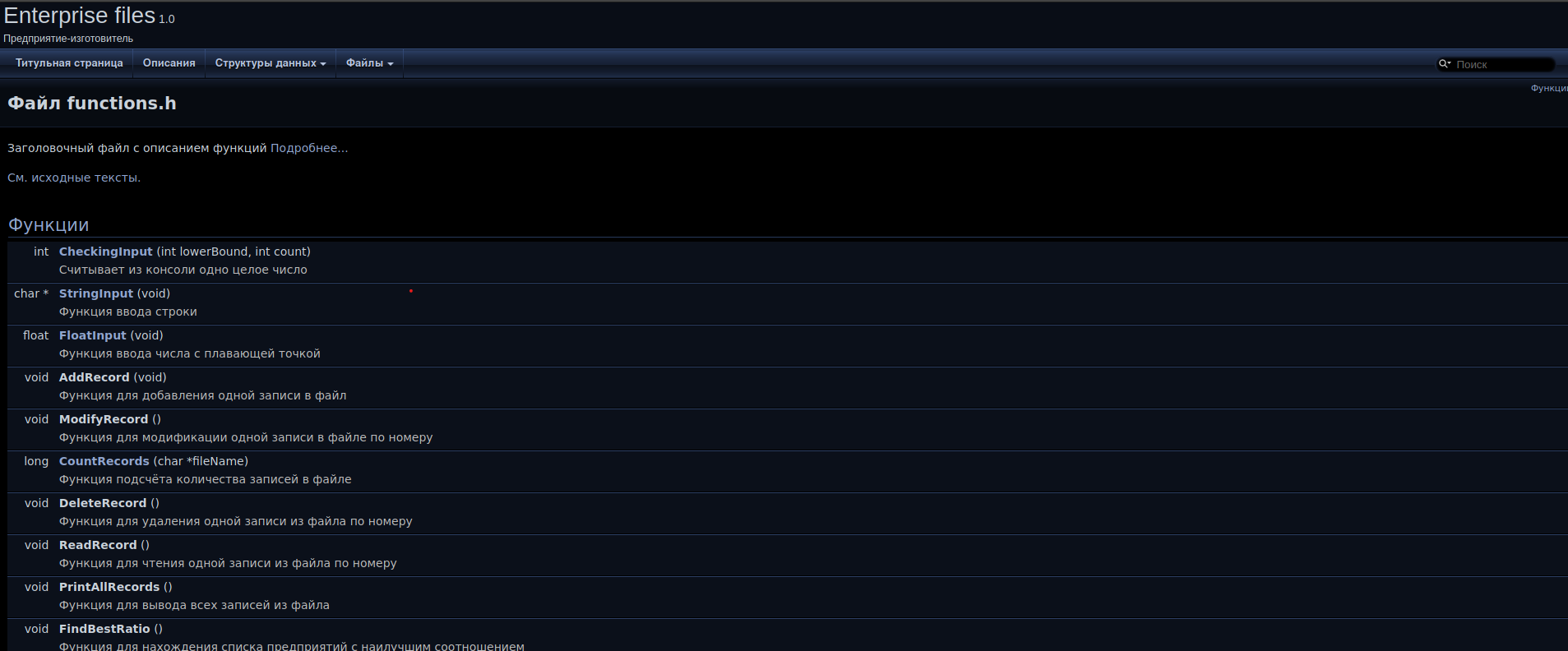


Рисунок 2 – Сгенерированная документация

**5** **Выводы**

В ходе работы был изучено управление файлами в ОС GNU/Linux. Была разработана программа в соответствии с вариантом.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 СТО 4.2-07-2014 Система менеджмента качества. Общие требования  
к построению, изложении и оформлению документов учебной деятельности.  
Дата введения – 30.12.2-13.

**ПРИЛОЖЕНИЕ А**

**Исходный код файла functions.c**

/\*! Функции программы \*/

#include <stdio.h>

#include <stdlib.h>

#include <sys/uio.h>

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

#include "functions.h"

#include "structure.h"

#define CORRECT\_INPUT 2

#define END\_STRING '\n'

#define BACKSPACE\_KEY 8

#define START\_CHAR\_RANGE 32

#define END\_CHAR\_RANGE 126

#define MAX\_LEN 100 // фиксированная длина записей

int CheckingInput(int lowerBound, int count)

{

int userInput = 1;

char inputChar = '\0';

int input = scanf("%d%c", &userInput, &inputChar);

if (count)

{

while (input != CORRECT\_INPUT || inputChar != END\_STRING

|| userInput < lowerBound || userInput > count)

{

if (userInput >= lowerBound && userInput <= count)

{

while ((inputChar = getchar()) != '\n');

}

userInput = 1;

printf("Неверный ввод. Попробуйте снова.\nВведите номер: ");

input = scanf("%d%c", &userInput, &inputChar);

}

}

else

{

while (input != CORRECT\_INPUT || inputChar != END\_STRING

|| userInput <= lowerBound)

{

if (userInput > lowerBound)

{

while ((inputChar = getchar()) != '\n');

}

userInput = 1;

printf("Неверный ввод. Попробуйте снова.\nВведите номер: ");

input = scanf("%d%c", &userInput, &inputChar);

}

}

return userInput;

}

char\* StringInput(void)

{

char\* userStr = (char\*)malloc(1 \* sizeof(char));

userStr[0] = '\0';

char curChar = 0;

int curSize = 1;

while(curChar != '\n')

{

curChar = getchar();

int deltaVal = 0; // Определяет, на сколько изменится длина массива

int lengthDif = 0;

// Если мы стираем символы, а не пишем их,

if (curChar == BACKSPACE\_KEY)

{

deltaVal = -1; // то будем уменьшать длину массива

lengthDif = 1; // и копировать строку до предпоследнего символа

}

// Иначе проверяем, входит ли введённый символ в диапазон печатных

else

{

if (curChar >= START\_CHAR\_RANGE && curChar <= END\_CHAR\_RANGE)

{

deltaVal = 1; // Если да, то будем увеличивать длину на 1

lengthDif = 2; // Не заполняем последние 2 символа -

// оставляем место для введённого символа и \0

}

else

{

continue; // Если это не печатный символ, то пропускаем его

}

}

// Если стирать больше нечего, но пользователь

// всё равно жмёт Backspace.

int newSize = curSize + deltaVal;

if (newSize == 0)

{

continue; // то мы переходим на следующую итерацию - ждём '\n'

}

char\* tmpStr = (char\*)malloc(newSize \* sizeof(char));

if (tmpStr) // Проверяем, выделилась ли память

{

// Идём до предпоследнего символа, т.к. надо в конец записать '\0'

for (int i = 0;

i < newSize - lengthDif;

++i)

{

tmpStr[i] = userStr[i];

}

if (curChar != BACKSPACE\_KEY) // Если введён печатный символ,

{

tmpStr[newSize - 2] = curChar; // Добавляем его в строку

tmpStr[newSize - 1] = '\0';

}

free(userStr);

userStr = tmpStr;

curSize = newSize;

}

else

{

printf("Не могу выделить память под обновлённую строку!");

break;

}

}

return userStr;

}

float FloatInput(void)

{

float number = 0;

while (number == 0 || number < 0)

{

char\* InputValue = StringInput();

number = strtof(InputValue, NULL);

if (number == 0 || number < 0)

{

printf("Неверный ввод. Попробуйте снова.\n"

"Повторите ввод: ");

}

free(InputValue);

}

return number;

}

void AddRecord(void)

{

Enterprise tmpEnterprise; // Cтруктура для хранения данных о продукте.

struct iovec iov[4]; // Массив структур iovec для векторной записи

char fileName[MAX\_LEN];

long count; // Переменная для хранения количества записей

printf("Введите название предприятия-изготовителя: ");

scanf("%s", tmpEnterprise.company);

printf("Введите тип предприятия: ");

scanf("%s", tmpEnterprise.type);

getchar();

printf("Введите цену: ");

tmpEnterprise.price = FloatInput();

printf("Введите производительность: ");

tmpEnterprise.performance = FloatInput();

iov[0].iov\_base = tmpEnterprise.company; // указатель на данные о предприятии-изготовителе

iov[0].iov\_len = MAX\_LEN; // размер данных о предприятии-изготовителе в байтах

iov[1].iov\_base = tmpEnterprise.type; // указатель на данные о типе

iov[1].iov\_len = MAX\_LEN; // размер данных о типе в байтах

iov[2].iov\_base = &tmpEnterprise.price; // указатель на данные о цене

iov[2].iov\_len = sizeof(double); // размер данных о цене в байтах

iov[3].iov\_base = &tmpEnterprise.performance; // указатель на данные о производительности

iov[3].iov\_len = sizeof(double); // размер данных о производительности в байтах

int fileDescriptor = -1;

while (fileDescriptor == -1)

{

printf("Введите имя файла: ");

scanf("%s", fileName);

int len = strlen(fileName); // Получаем длину имени файла

char \*ext = fileName + len - 4; // Получаем указатель на последние четыре символа

if (strcmp(ext, ".txt") == 0 || strcmp(ext, ".bin") == 0)

{

fileDescriptor = open(fileName, O\_WRONLY | O\_APPEND

| O\_CREAT, 0666); // Открываем файл для записи в конец или создаём

// его при необходимости с правами 0666

}

else

{

printf("Неверное расширение файла! Попробуйте снова\n");

}

}

if (writev(fileDescriptor, iov, 4) == -1)

{

// Выполняем векторную запись из массива структур iovec в файл и

// проверяем успешность операции

perror("Ошибка записи в файл");

exit(1);

}

close(fileDescriptor); // Закрываем файл

printf("Запись успешно добавлена.\n");

}

void ModifyRecord()

{

int number; // Номер записи для модификации

int choice; // Выбор пользователя для модификации полей

Enterprise tmpEnterprise; // Структура для хранения данных о предприятии

struct iovec iov[4]; // Массив структур iovec для векторной записи

int fileOpen = 0;

int fileDescriptor = -1;

char fileName[MAX\_LEN];

while (fileDescriptor == -1)

{

printf("Введите имя файла (Q - для выхода): ");

scanf("%s", fileName);

int len = strlen(fileName);

char\* ext = fileName + len - 4;

if (strcmp(ext, ".txt") == 0 || strcmp(ext, ".bin") == 0)

{

// Открываем файл для чтения и записи

fileDescriptor = open(fileName, O\_RDWR);

long tmpCount = CountRecords(fileName);

if (tmpCount == 0)

{

printf("Файл пустой.\n");

fileOpen = -1;

}

if (fileDescriptor == -1)

{

printf("Такого файла нет! Попробуйте снова.\n");

}

}

else

{

if (strcmp(fileName, "Q") == 0)

{

fileOpen = -1;

break;

}

printf("Неверное расширение файла! Попробуйте снова\n");

}

}

if (fileOpen == 0)

{

long count = CountRecords(fileName);

printf("Всего в файле %ld предприятий/я.\n", count);

printf("Введите номер предприятия: ");

number = CheckingInput(1, count); // Номер записи для чтения

iov[0].iov\_base = tmpEnterprise.company; // Указатель на данные о предприятии-изготовителе

iov[0].iov\_len = MAX\_LEN; // Размер данных о предприятии-изготовителе в байтах

iov[1].iov\_base = tmpEnterprise.type; // Указатель на данные о типе

iov[1].iov\_len = MAX\_LEN; // Размер данных о типе в байтах

iov[2].iov\_base = &tmpEnterprise.price; // Указатель на данные о цене

iov[2].iov\_len = sizeof(double); // Размер данных о цене в байтах

iov[3].iov\_base = &tmpEnterprise.performance; // Указатель на данные о производительности

iov[3].iov\_len = sizeof(double); // Размер данных о производительности в байтах

if (lseek(fileDescriptor, (number - 1) \*

(MAX\_LEN \* 2 + sizeof(double) \* 2),

SEEK\_SET) == -1)

{

// Перемещаем указатель позиции в файле на начало нужной записи

// и проверяем успешность операции

perror("Ошибка перемещения в файле");

exit(1);

}

if (readv(fileDescriptor, iov, 4) == -1)

{

// Выполняем векторное чтение из файла в массив структур iovec

// и проверяем успешность операции

perror("Ошибка чтения из файла");

exit(1);

}

close(fileDescriptor); // Закрываем файл

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

printf("| Параметры: |\n");

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

printf("1 - Предприятие-изготовитель\n");

printf("2 - Тип\n");

printf("3 - Цена\n");

printf("4 - Производительность\n");

printf("5 - Все поля\n");

printf("Введите номер пункта: ");

choice = CheckingInput(1, 5);

switch(choice)

{

case 1: // Модификация поля предприятие-изготовитель

printf("Введите новое название предприятия-изготовителя: ");

scanf("%s", tmpEnterprise.company);

iov[0].iov\_base = tmpEnterprise.company; // Указатель на данные о предприятие-изготовителе

iov[0].iov\_len = MAX\_LEN; // Размер данных о предприяте-изготовителе в байтах

break;

case 2: // Модификация поля тип

printf("Введите новый тип предприятия: ");

scanf("%s", tmpEnterprise.type);

iov[1].iov\_base = tmpEnterprise.type; // Указатель на данные о типе

iov[1].iov\_len = MAX\_LEN; // Размер данных о типе в байтах

break;

case 3: // Модификация поля цена

printf("Введите новую цену: ");

tmpEnterprise.price = FloatInput();

iov[2].iov\_base = &tmpEnterprise.price; // Указатель на данные о цене

iov[2].iov\_len = sizeof(double); // Размер данных о цене в байтах

break;

case 4: // Модификация поля производительность

printf("Введите новую производительность: ");

tmpEnterprise.performance = FloatInput();

iov[3].iov\_base = &tmpEnterprise.performance; // Указатель на данные о производительности

iov[3].iov\_len = sizeof(double); // Размер данных о производительности в байтах

break;

case 5: // Модификация всех полей

printf("Введите новое название предприятия-изготовителя: ");;

scanf("%s", tmpEnterprise.company);

printf("Введите новый тип предприятия: ");

scanf("%s", tmpEnterprise.type);

getchar();

printf("Введите новую цену: ");

tmpEnterprise.price = FloatInput();

printf("Введите новую производительность: ");

tmpEnterprise.performance = FloatInput();

iov[0].iov\_base = tmpEnterprise.company; // Указатель на данные о предприятие-изготовителе

iov[0].iov\_len = MAX\_LEN; // Размер данных о предприяте-изготовителе в байтах

iov[1].iov\_base = tmpEnterprise.type; // Указатель на данные о типе

iov[1].iov\_len = MAX\_LEN; // Размер данных о типе в байтах

iov[2].iov\_base = &tmpEnterprise.price; // Указатель на данные о цене

iov[2].iov\_len = sizeof(double); // Размер данных о цене в байтах

iov[3].iov\_base = &tmpEnterprise.performance; // Указатель на данные о производительности

iov[3].iov\_len = sizeof(double); // Размер данных о производительности в байтах

break;

default: // Неверный выбор пользователя

printf("Неверный выбор.\n");

return;

}

fileDescriptor = open(fileName, O\_WRONLY); // Открываем файл для записи

if (fileDescriptor == -1)

{

// Проверяем успешность открытия файла

perror("Ошибка открытия файла");

exit(1);

}

if (lseek(fileDescriptor, (number - 1) \*

(MAX\_LEN \* 2 + sizeof(double) \* 2), SEEK\_SET) == -1)

{

// Перемещаем указатель позиции в файле на начало нужной записи

// и проверяем успешность операции

perror("Ошибка перемещения в файле");

exit(1);

}

if (writev(fileDescriptor, iov, 4) == -1)

{

// Выполняем векторную запись из массива структур iovec в

// файл и проверяем успешность операции

perror("Ошибка записи в файл");

exit(1);

}

close(fileDescriptor); // Закрываем файл

printf("Предприятие успешно изменено.\n");

}

}

long CountRecords(char\* fileName)

{

long count; // Переменная для хранения количества записей

int fileDescriptor = -1;

fileDescriptor = open(fileName, O\_RDONLY);

count = lseek(fileDescriptor,0,SEEK\_END)/(MAX\_LEN \*2 + sizeof(double)\*2);

close(fileDescriptor);

return count;

}

void DeleteRecord()

{

int number;

long count;

char buffer[MAX\_LEN \* 2 + sizeof(double) \* 2]; // Буфер для хранения данных о предприятии

struct iovec iov[1]; // Массив структур iovec для векторной чтения/записи

int fileOpen = 0;

int fileDescriptor = -1;

char fileName[MAX\_LEN];

while (fileDescriptor == -1)

{

printf("Введите имя файла (Q - для выхода): ");

scanf("%s", fileName);

int len = strlen(fileName);

char\* ext = fileName + len - 4;

if (strcmp(ext, ".txt") == 0 || strcmp(ext, ".bin") == 0)

{

fileDescriptor = open(fileName, O\_RDWR);

long tmpCount = CountRecords(fileName);

if (tmpCount == 0)

{

printf("Файл пустой.\n");

fileOpen = -1;

}

if (fileDescriptor == -1)

{

printf("Такого файла нет! Попробуйте снова.\n");

}

}

else

{

if (strcmp(fileName, "Q") == 0)

{

fileOpen = -1;

break;

}

printf("Неверное расширение файла! Попробуйте снова\n");

}

}

if (fileOpen == 0)

{

count = CountRecords(fileName);

printf("Всего в файле %ld предприятий/я.\n", count);

printf("Введите номер предприятия, которое вы хотите удалить: ");

number = CheckingInput(1, count); // Номер записи для чтения

for(int i=number;i<count;i++)

{

// Цикл по всем последующим записям

if (lseek(fileDescriptor,i\*(MAX\_LEN\*2+sizeof(double)\*2),SEEK\_SET)==-1)

{

// Перемещаем указатель позиции в файле на начало текущей записи

// и проверяем успешность операции

perror("Ошибка перемещения в файле");

exit(1);

}

iov[0].iov\_base = buffer; // Указатель на данные в буфере

iov[0].iov\_len = MAX\_LEN\*2+sizeof(double)\*2; // Размер данных в буфере в байтах

if(readv(fileDescriptor,iov,1)==-1)

{

// Выполняем векторное чтение из файла в массив структур iovec

// и проверяем успешность операции

perror("Ошибка чтения из файла");

exit(1);

}

if(lseek(fileDescriptor,(i-1)\*(MAX\_LEN\*2+sizeof(double)\*2),SEEK\_SET)==-1)

{

// Перемещаем указатель позиции в файле на начало предыдущей

// позиции и проверяем успешность операции

perror("Ошибка перемещения в файле");

exit(1);

}

if(writev(fileDescriptor,iov,1)==-1)

{

// Выполняем векторную запись из массива структур iovec в файл

// и проверяем успешность операции

perror("Ошибка записи в файл");

exit(1);

}

}

if(ftruncate(fileDescriptor,(count-1)\*(MAX\_LEN\*2+sizeof(double)\*2))==-1)

{

// Уменьшаем размер файла на одну запись и проверяем успешность операции

perror("Ошибка изменения размера файла");

exit(1);

}

close(fileDescriptor); // Закрываем файл

printf("Предприятие успешно удалено.\n");

}

}

void ReadRecord()

{

Enterprise tmpEnterprise;

struct iovec iov[4];

int fileOpen = 0;

int fileDescriptor = -1;

char fileName[MAX\_LEN];

while (fileDescriptor == -1)

{

printf("Введите имя файла (Q - для выхода): ");

scanf("%s", fileName);

int len = strlen(fileName);

char\* ext = fileName + len - 4;

if (strcmp(ext, ".txt") == 0 || strcmp(ext, ".bin") == 0)

{

fileDescriptor = open(fileName, O\_RDONLY);

long tmpCount = CountRecords(fileName);

if (tmpCount == 0)

{

printf("Файл пустой.\n");

fileOpen = -1;

}

if (fileDescriptor == -1)

{

printf("Такого файла нет! Попробуйте снова.\n");

}

}

else

{

if (strcmp(fileName, "Q") == 0)

{

fileOpen = -1;

break;

}

printf("Неверное расширение файла! Попробуйте снова\n");

}

}

if (fileOpen == 0)

{

long count = CountRecords(fileName);

printf("Всего в файле %ld предприятий/я.\n", count);

printf("Введите номер предприятия: ");

int number = CheckingInput(1, count);

if (lseek(fileDescriptor, (number - 1) \* (MAX\_LEN \* 2 + sizeof(double) \* 2), SEEK\_SET) == -1)

{

perror("Ошибка перемещения в файле");

exit(1);

}

iov[0].iov\_base = tmpEnterprise.company; // указатель на данные о предприятие-изготовителе

iov[0].iov\_len = MAX\_LEN; // размер данных о предприятие-изготовителе в байтах

iov[1].iov\_base = tmpEnterprise.type; // указатель на данные о типе

iov[1].iov\_len = MAX\_LEN; // размер данных о типе в байтах

iov[2].iov\_base = &tmpEnterprise.price; // указатель на данные о цене

iov[2].iov\_len = sizeof(double); // размер данных о цене в байтах

iov[3].iov\_base = &tmpEnterprise.performance; // указатель на данные о производительности

iov[3].iov\_len = sizeof(double); // размер данных о производительности в байтах

if (readv(fileDescriptor, iov, 4) == -1)

{

perror("Ошибка чтения из файла");

exit(1);

}

close(fileDescriptor);

printf("\n");

printf("Предприятие №%d\n", number);

printf("Предприятие-изготовитель: %s\n", tmpEnterprise.company);

printf("Тип: %s\n", tmpEnterprise.type);

printf("Цена: %.2f\n", tmpEnterprise.price);

printf("Производительность: %.2f\n", tmpEnterprise.performance);

printf("\n");

}

}

void PrintAllRecords() {

Enterprise tmpEnterprise; // структура для хранения данных о продукте

struct iovec iov[4]; // массив структур iovec для векторного чтения

int count = 0; // счетчик записей

int fileDescriptor = -1;

char fileName[MAX\_LEN];

int fileOpen = 0;

while (fileDescriptor == -1)

{

printf("Введите имя файла (Q - для выхода): ");

scanf("%s", fileName);

int len = strlen(fileName);

char\* ext = fileName + len - 4;

if (strcmp(ext, ".txt") == 0 || strcmp(ext, ".bin") == 0)

{

fileDescriptor = open(fileName, O\_RDONLY);

if (fileDescriptor == -1)

{

printf("Такого файла нет! Попробуйте снова.\n");

}

}

else

{

if (strcmp(fileName, "Q") == 0)

{

fileOpen = -1;

break;

}

printf("Неверное расширение файла! Попробуйте снова\n");

}

}

if (fileOpen == 0)

{

while (1)

{ // бесконечный цикл

iov[0].iov\_base = tmpEnterprise.company; // указатель на данные о предприятие-изготовителе

iov[0].iov\_len = MAX\_LEN; // размер данных о предприяте-изготовителе в байтах

iov[1].iov\_base = tmpEnterprise.type; // указатель на данные о типе

iov[1].iov\_len = MAX\_LEN; // размер данных о типе в байтах

iov[2].iov\_base = &tmpEnterprise.price; // указатель на данные о цене

iov[2].iov\_len = sizeof(double); // размер данных о цене в байтах

iov[3].iov\_base = &tmpEnterprise.performance; // указатель на данные о производительности

iov[3].iov\_len = sizeof(double); // размер данных о производительности в байтах

// Выполняем векторное чтение из файла в массив структур iovec и

// сохраняем количество прочитанных байтов

ssize\_t bytesRead = readv(fileDescriptor, iov, 4);

if (bytesRead == -1)

{ // Проверяем успешность операции чтения

perror("Ошибка чтения из файла");

exit(1);

}

if (bytesRead == 0)

{ // Проверяем конец файла

break;

}

count++; // Увеличиваем счетчик записей

printf("\n");

printf("Предприятие №%d\n", count);

printf("Предприятие-изготовитель: %s\n", tmpEnterprise.company);

printf("Тип: %s\n", tmpEnterprise.type);

printf("Цена: %.2f\n", tmpEnterprise.price);

printf("Производительность: %.2f\n", tmpEnterprise.performance);

printf("\n");

}

close(fileDescriptor);

if (count == 0)

{

// Если счетчик равен нулю, то сообщаем об отсутствии записей

printf("Файл пуст.\n");

}

}

}

void FindBestRatio()

{

Enterprise tmpEnterprise;

struct iovec iov[4];

int fileDescriptor = -1;

char fileName[MAX\_LEN];

int fileOpen = 0;

double bestRatio = 0.0; // Лучшее соотношение

while (fileDescriptor == -1)

{

printf("Введите имя файла (Q - для выхода): ");

scanf("%s", fileName);

int len = strlen(fileName);

char\* ext = fileName + len - 4;

if (strcmp(ext, ".txt") == 0 || strcmp(ext, ".bin") == 0)

{

fileDescriptor = open(fileName, O\_RDONLY);

long tmpCount = CountRecords(fileName);

if (tmpCount == 0)

{

printf("Файл пустой.\n");

fileOpen = -1;

}

if (fileDescriptor == -1)

{

printf("Такого файла нет! Попробуйте снова.\n");

}

}

else

{

if (strcmp(fileName, "Q") == 0)

{

fileOpen = -1;

break;

}

printf("Неверное расширение файла! Попробуйте снова\n");

}

}

if (fileOpen == 0)

{

while (1)

{ // бесконечный цикл

iov[0].iov\_base = tmpEnterprise.company; // указатель на данные о предприятие-изготовителе

iov[0].iov\_len = MAX\_LEN; // размер данных о предприяте-изготовителе в байтах

iov[1].iov\_base = tmpEnterprise.type; // указатель на данные о типе

iov[1].iov\_len = MAX\_LEN; // размер данных о типе в байтах

iov[2].iov\_base = &tmpEnterprise.price; // указатель на данные о цене

iov[2].iov\_len = sizeof(double); // размер данных о цене в байтах

iov[3].iov\_base = &tmpEnterprise.performance; // указатель на данные о производительности

iov[3].iov\_len = sizeof(double); // размер данных о производительности в байтах

ssize\_t bytesRead = readv(fileDescriptor, iov, 4); // выполнить векторное чтение из файла в массив структур iovec и сохранить количество прочитанных байтов

if (bytesRead == -1)

{

perror("Ошибка чтения из файла");

exit(1);

}

if (bytesRead == 0)

{

break;

}

double ratio = tmpEnterprise.price /

tmpEnterprise.performance; // Вычисление соотношения

if (ratio < bestRatio || bestRatio ==

0.0)

{ // Если соотношение лучше или еще не определено

bestRatio = ratio; // Обновляем лучшее соотношение

}

}

close(fileDescriptor);

printf("Лучшее соотношение цена/производительность: %.2f\n",

bestRatio);

printf("Предприятия с таким коэффициентом:\n");

fileDescriptor = open(fileName, O\_RDONLY);

while (1)

{ // бесконечный цикл

iov[0].iov\_base = tmpEnterprise.company; // указатель на данные о предприятие-изготовителе

iov[0].iov\_len = MAX\_LEN; // размер данных о предприяте-изготовителе в байтах

iov[1].iov\_base = tmpEnterprise.type; // указатель на данные о типе

iov[1].iov\_len = MAX\_LEN; // размер данных о типе в байтах

iov[2].iov\_base = &tmpEnterprise.price; // указатель на данные о цене

iov[2].iov\_len = sizeof(double); // размер данных о цене в байтах

iov[3].iov\_base = &tmpEnterprise.performance; // указатель на данные о производительности

iov[3].iov\_len = sizeof(double); // размер данных о производительности в байтах

ssize\_t bytesRead = readv(fileDescriptor, iov, 4); // выполнить векторное чтение из файла в массив структур iovec и сохранить количество прочитанных байтов

if (bytesRead == -1)

{ // проверить успешность операции чтения

perror("Ошибка чтения из файла");

exit(1);

}

if (bytesRead == 0)

{ // проверить конец файла

break;

}

double ratio = tmpEnterprise.price /

tmpEnterprise.performance;

if (ratio == bestRatio)

{ // Если соотношение совпадает с лучшим

printf("%s\n",

tmpEnterprise.company); // Выводим название предприятия

}

}

close(fileDescriptor); // закрыть файл

}

}

void FindPriceRange()

{

double minPrice = 0.0; // Минимальная цена

double maxPrice = 0.0; // Максимальная цена

Enterprise tmpEnterprise; // структура для хранения данных о продукте

struct iovec iov[4]; // массив структур iovec для векторного чтения

int fileDescriptor = -1;

char fileName[MAX\_LEN];

int fileOpen = 0;

double bestRatio = 0.0; // Лучшее соотношение

while (fileDescriptor == -1)

{

printf("Введите имя файла (Q - для выхода): ");

scanf("%s", fileName);

int len = strlen(fileName);

char\* ext = fileName + len - 4;

if (strcmp(ext, ".txt") == 0 || strcmp(ext, ".bin") == 0)

{

fileDescriptor = open(fileName, O\_RDONLY);

long tmpCount = CountRecords(fileName);

if (tmpCount == 0)

{

printf("Файл пустой.\n");

fileOpen = -1;

}

if (fileDescriptor == -1)

{

printf("Такого файла нет! Попробуйте снова.\n");

}

}

else

{

if (strcmp(fileName, "Q") == 0)

{

fileOpen = -1;

break;

}

printf("Неверное расширение файла! Попробуйте снова\n");

}

}

if (fileOpen == 0)

{

while (1)

{ // бесконечный цикл

iov[0].iov\_base = tmpEnterprise.company; // указатель на данные о предприятие-изготовителе

iov[0].iov\_len = MAX\_LEN; // размер данных о предприяте-изготовителе в байтах

iov[1].iov\_base = tmpEnterprise.type; // указатель на данные о типе

iov[1].iov\_len = MAX\_LEN; // размер данных о типе в байтах

iov[2].iov\_base = &tmpEnterprise.price; // указатель на данные о цене

iov[2].iov\_len = sizeof(double); // размер данных о цене в байтах

iov[3].iov\_base = &tmpEnterprise.performance; // указатель на данные о производительности

iov[3].iov\_len = sizeof(double); // размер данных о производительности в байтах

ssize\_t bytesRead = readv(fileDescriptor, iov, 4);

if (bytesRead == -1)

{

perror("Ошибка чтения из файла");

exit(1);

}

if (bytesRead == 0)

{

break;

}

double price = tmpEnterprise.price;

if (price < minPrice ||

minPrice == 0.0)

{

minPrice = price;

}

if (price > maxPrice ||

maxPrice == 0.0)

{

maxPrice = price;

}

}

close(fileDescriptor);

printf("Диапазон цен от %.2f до %.2f\n", minPrice, maxPrice);

}

}