Федеральное государственное бюджетное

образовательное учреждение

высшего образования

**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ»**

(национальный исследовательский университет)

Факультет № 8 «Компьютерные науки и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

**Задание № 6**

**«Обработка последовательной файловой структуры   
на языке Си»**

по дисциплине «Практикум на ЭВМ»

2 семестр

|  |  |
| --- | --- |
| Студент | Мезенин О.А. |
| Группа | М8О-106Б-21 |
| Преподаватель | Дубинин А.В. |
| Оценка |  |

Оглавление

[Оглавление 2](#_Toc98705859)

[Введение 3](#_Toc98705860)

[Процесс и методы обработки файлов 4](#_Toc98705861)

[Представление файловой структуры в памяти 4](#_Toc98705862)

[Методы для работы с потоками 6](#_Toc98705863)

[Системный вызов и файловый дескриптор 7](#_Toc98705864)

[Передача аргументов программе 8](#_Toc98705865)

[Практика 9](#_Toc98705866)

[Описание программ 9](#_Toc98705867)

[Тесты 11](#_Toc98705868)

[Выводы 12](#_Toc98705869)

[Список источников 13](#_Toc98705870)

Введение

Для написания программ пользуются не только атомарными значениями базовых типов, но и применяют структурные значения, которые представляют собой организованную совокупность других элементов. Такими значениями могут быть, например, вектор, матрица или информация о реальном объекте, состоящая из полей разного типа. Также структурами являются файлы, использующиеся, в первую очередь, для долгосрочного хранения данных на диске.

Целью курсовой работы является знакомство с процессом и методами последовательной обработки файловой структуры на языке Си, а также с представлением файловой структуры в памяти компьютера.

Процесс и методы обработки файлов

Представление файловой структуры в памяти

Рассматривая организацию данных на внешнем носителе, различают логическое и физическое представление данных, логическую и физическую структуры файла.

*Логическое представление* определяет, как используются данные в прикладных программах, какие логические (причинно-следственные связи) существуют между отдельными элементами данных. *Физическое представление* определяет хранение данных на физическом носителе (во внешней памяти).

Логическая запись представляет собой совокупность элементов данных, воспринимаемую и обрабатываемую как единое целое. С точки зрения пользователя прикладные программы работают именно с логическими записями. Обычно логические записи физически размещаются отдельно в рабочей области памяти. Совокупность логических записей образует *логический файл.* Обработка логических файлов осуществляется прикладными программными средствами.

Физическая запись представляет собой совокупность элементов данных, размещаемых на внешнем носителе, которая может быть считана (записана) одной командой ввода-вывода. Совокупность физических записей образует *физический файл.*

По отношению к *способам доступа* к логическим записям файла (или *режимы обработки данных в файловых структурах)* выделяют:

1. *Последовательный доступ.* В этом случае обработка (запись или чтение) логических записей осуществляются последовательно: из ПЗУ в оперативную память (и наоборот) в соответствии с порядком, в котором они размещены на носителе. Такой тип доступа применяется для неструктурированных файлов, длина и формат записей которых могут меняться, например, для текстовых файлов.

2. *Произвольный (прямой) доступ.* В этом случае обработка логических записей выполняется в соответствии с логикой и алгоритмами, реализованными в конкретной программе. Данные считываются и формируются записями фиксированной длины, к которым можно обратиться по номерам машинных слов. Подобный способ доступа применяется при обработке структурированных файлов, например, двоичных файлов.

Логическая организация файлов может быть различна. Современные ОС и используемые ими файловые системы поддерживают широко распространенные *типы схем логической организации файлов* или *логической структуризации данных в файлах*.

1. В виде записей фиксированной длины.



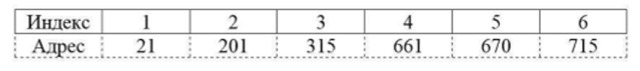
Память, выделяемая для каждой из записей подобного типа, всегда имеет фиксированный размер, известный заранее.

1. Записей переменной длины.



Перед *байтами содержания* (данными) располагаются *байты длины,* в которых хранится непосредственная длина содержания.

1. Индексной логической организации.



Для обработки файлов подобного типа строится специальная *индексная таблица,* в которой каждому значению ключа записи поставлен в соответствие адрес внешней памяти. Индексная таблица не содержит данных, но в ней хранятся номера *(индексы)* записей исходного файла. Ведением индексных таблиц занимается файловая система.

Выделяют два основных формата файлов: двоичные и текстовые.

*Двоичные файлы* обычно состоят из последовательностей байт, сгруппированных в записи фиксированной длины. В файлах такого типа хранятся исполняемые программы и данные в двоичном представлении. Без специальных средств (программного обеспечения) увидеть содержимое двоичного файла в понятном для человека виде невозможно.

*Текстовый файл* состоит из последовательностей символов, которые сгруппированы в строки переменной длины. Каждая такая строка является логической записью, состоящей из текстовых символов и заканчивающейся специальным маркером конца строки. Их последовательность выводится на экран или печатающее устройство в понятном для человека виде. В виде текстовых файлов могут быть представлены исходные тексты программ на некотором алгоритмическом языке, таблицы, исходные данные, результаты и т.д.

Методы для работы с потоками

В библиотеке <stdio.h> есть функции для работы с потоками. Основные из них:

FILE \*fopen(const char \*pathname, const char \*mode) - открывает файл, имя которого указано в параметре pathname и связывает его с потоком, который может быть идентифицирован для выполнения различных операций с файлом.

size\_t fread(void \*ptr, size\_t size, size\_t nmemb, FILE \*stream)

size\_t fwrite(const void \*ptr, size\_t size, size\_t nmemb, FILE \*stream)

Эти функции позволяют читать и записывать блоки данных любого типа.

ptr - указатель на блок памяти, размер которого должен быть минимум  (size\* nmemb) байт.

size - размер в байтах каждого считываемого элемента.

nmemb - количество элементов, каждый из которых имеет размер size байт.

stream - указатель на объект типа FILE, который связан с потоком ввода.

Функции fread и fwrite возвращают количество прочитанных/записанных элементов.

int fprintf(FILE \*stream, const char \*format, ...)

int fscanf(FILE \*stream, const char \*format, ...)

Аналогичны printf и scanf, но могут работать с указанным потоком.

int fseek(FILE \*stream, long offset, int whence) – перемещает указатель позиции в потоке.

stream - указатель на объект типа FILE, идентифицируемый поток.

offset - количество байт для смещения.

whence - позиция указателя, относительно которой будет выполняться смещение. Такая позиция задаётся одной из следующих констант:

SEEK\_SET – начало файла

SEEK\_CUR – текущее положение файла

SEEK\_END – конец файла

int feof(FILE \*stream) -  проверяет, достигнут ли конец файла, связанного с потоком.

int fclose(FILE \*stream) - закрывает и разъединяет файл stream, связанный с потоком.

Системный вызов и файловый дескриптор

Когда программе нужна операция, требующая доступ к оборудованию (например, доступ к файлам) или доступ к другим процессам, она обращается к ядру операционной системы. Такое обращение называется *системным вызовом*.

*Файловый дескриптор* – это неотрицательное число, которое является идентификатором потока ввода-вывода. Такие числа находятся в таблице файловых дескрипторов, которая в свою очередь находится в ядре ОС.

Обычно файловые дескрипторы выделяются последовательно. Например, когда создаётся или открывается файл, ядро связывает его с файловым дескриптором. Первые три дескриптора зарезервированы:

0 — стандартный ввод (stdin).

1 — стандартный вывод (stdout).

2 — стандартный поток ошибок (stderror).

В языке Си файловый дескриптор записывается в структуру FILE, содержащую информацию о файле. Указатель на FILE возвращает, например, функция fopen.

Передача аргументов программе

Для того, чтобы программа могла получать аргументы с командной строки, необходимо в функции main указать два параметра: int argc – количество аргументов, char \*argv[] – массив из самих аргументов. Первый аргумент argv[0] указывает на имя самой программы.

Практика

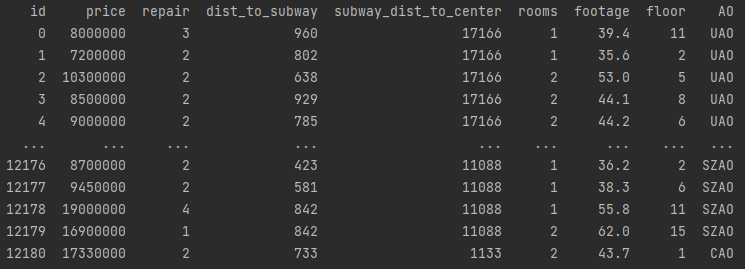
Требуется выбрать предметную область и подготовить файл с данными, а также написать две программы:

1) Программа переводит исходный файл в текстовом формате в бинарный.

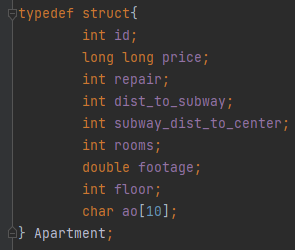
2) Программа читает получившийся бинарный файл и работает с данными, сохранёнными в нём.

Описание программ

В качестве файла с данными был выбран датасет с сайта kaggle.com с квартирами в Москве за ноябрь 2020 года.

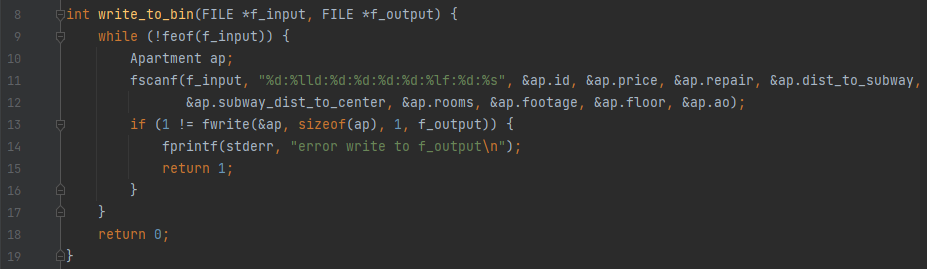


Структура с информацией о квартире записана в файле ma\_struct.h:

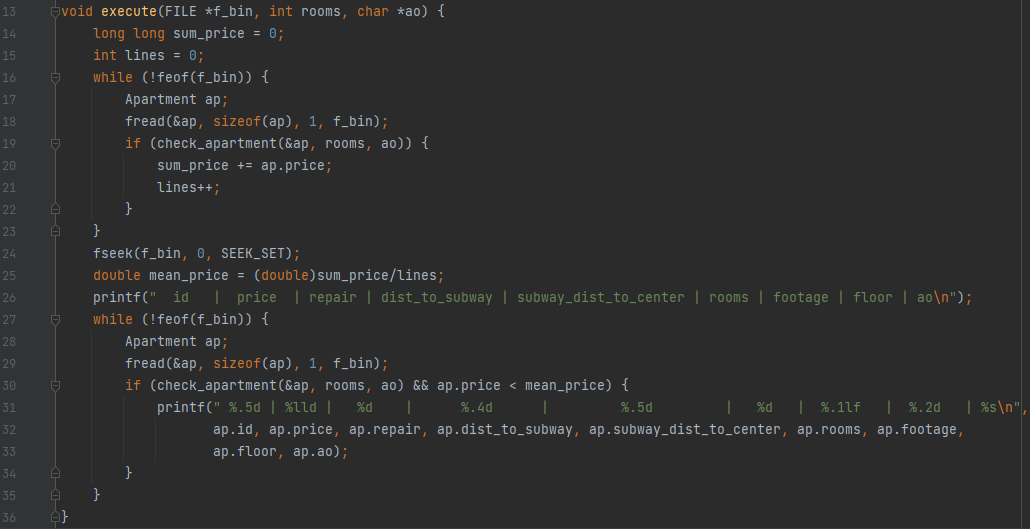


Программа перевода исходного файла в текстовом формате в бинарный to\_bin.c принимает путь к исходному файлу. Также можно указать название выходного файла.

Так как файл может быть очень большим и не помещаться в оперативную память, функция записи читает файл по одной строке.

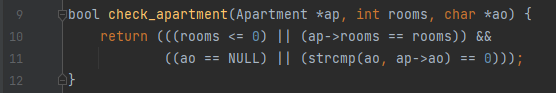


Основная программа main.c принимает путь к бинарному файлу, а также аргументы rooms и ao. Функция execute выводит все квартиры в административном округе ao с указанными количеством комнат rooms, цена которых ниже средней.

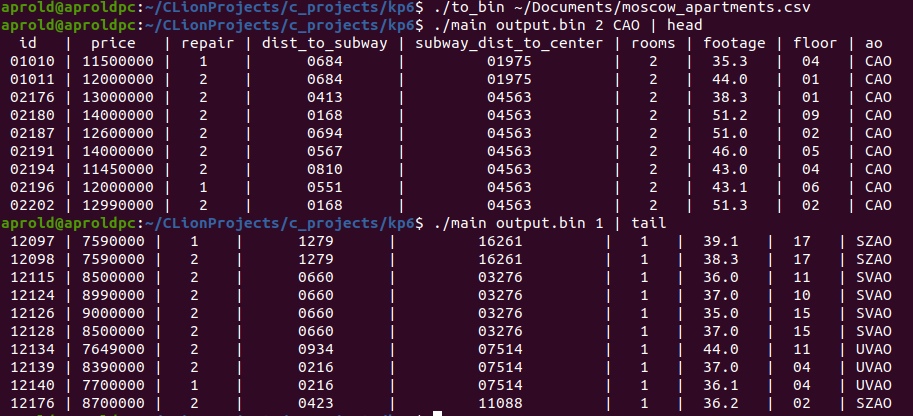


Функция проходит по файлу два раза: в первый ищет среднюю цену квартир, подходящих по настройкам, во второй – выводит подходящие квартиры ниже средней цены.

Функция check\_apartment проверяет, подходит ли квартира по заданным параметрам:



Тесты



Выводы

Файлы предназначены для хранения потенциально бесконечной информации. Для хранения всего файла оперативной памяти может не хватить, поэтому читать файл нужно по частям.

Файл может быть текстовым и бинарным. Информация, содержащаяся в файле текстового формата, предназначена для интерпретации человеком. Файл в бинарном формате может быть прочитан программой быстрее, т.к. он состоит из последовательности байт данных и пользоваться дополнительными преобразованиями для интерпретации информации не нужно.

В ходе работы были изучены функции для работы с потоками, являющиеся системными вызовами – обращениями к ядру ОС. Файловый дескриптор является идентификатором потока ввода-вывода, хранящийся в таблице файловых дескриптором, которая находится в ядре.

Список источников

1. Структуры хранения данных в памяти - <https://studme.org/192642/informatika/struktury_hraneniya_dannyh_pamyati>

2. Что такое файловый дескриптор простыми словами - <https://timeweb.com/ru/community/articles/chto-takoe-faylovyy-deskriptor-prostymi-slovami>

3. Moscow Apartment Listings w/prices - https://www.kaggle.com/alexeyleshchenko/moscow-apartment-listings