Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Факультет №8 «Компьютерные науки и прикладная математика»
Кафедра 805 «Прикладная математика»

Курсовой проект
по курсу
«Архитектура ЭВМ, системное программное обеспечение»
2 семестр
Задание 6

Автор работы:
студент 1 курса, гр. М8О-103Б-21
Фадеев Д.В.
Проверил:
Севастьянов В.С.
Дата сдачи:

Москва 2022 г

Содержание:

Задача.

Разработать множество программ на языке Си и структуру данных для взаимодействия с простейшей не реляционной базой данных (далее БД). Записи данной структуры данных в БД хранятся в не текстовом бинарном файле, который генерируется специальной программой, записывающей в него сразу несколько записей. По условию минимальное количество записей - 10. Кроме этого нужно написать программы по распечатке содержимого файла в удобном читаемом формате, добавлению новых записей в БД (а значит и в файл) и выполнению действия из задания с выводом результата.

В моём задании база данных содержит записи о студентах. В записи хранится личная информация о студенте: фамилия, инициалы, пол, номер группы. Также каждый студент имеет информацию о сданных экзаменах (или зачётах) и оценках за них. По заданию мне нужно составить программу, считающую количество девушек студентов с только одной пятёркой за зачёт (или экзамен) из группы, номер которой передан в аргументах программы.

Реализация.

Цель курсового проекта может быть достигнута, если ее разбить на 4 основных блока, или же программы. Каждая из них выполняет действия над БД, описанные в задании. Опишем реализованные исполняемые файлы:

1. create - генерация файла BD.bin, в котором будут храниться БД. Также исполняемый файл заполняет БД одиннадцатью записями о студентах и их экзаменах.
2. modifying - добавление новых записей о студентах.
3. print - распечатка в табличном виде содержимого бинарного файла БД.
4. analysis - анализ записей БД по условию задания.

Написание исходного кода программы и отладка производилась в среде разработки XCode, но для дальнейшего использования программы был написан Makefile для сборки проекта утилитой make.

Сценарий использования программы:

- сгенерировать записи в бинарном файле
- добавить нового студента
- вывести содержимое
- обработать файл по условию с выводом результата

Организация исходного кода.

Все программы, выполняющие действия из задания, взаимодействуют с БД через файл интерфейса (по совместительству «ядра» базы данных). В следующей таблице описаны файлы всего исходного кода.

Файл	Задача
repository.h	Заголовочный файл с описанным интерфейсом взаимолействия с БД
repository.c	Реализация интерфейса взаимодействия
print.c	Печать в консоли содержимого БД
create.c	Создание БД и первично заполнение
modifying.c	Добавление новых студентов в БД
analysis.c	Анализ содержимого БД по условию и вывод результатов
Makefile	Файл сборки

Описание реализации БД.

Начнём разбор реализации проекта с основных двух файлов: repository.h и repository.c.

repository.h

По заданию студент может иметь кроме строго заданной личной информации ещё и неопределённое количество экзаменов, названия которых могут отличаться в зависимости от факультетов и курса. Поэтому в данном файле описаны две структуры - Student и test.

Student обладает полями фамилии, инициалов, группы и пола, а также указатель на массив зачётов и экзаменов, которые он сдавал, а так же их количество.

test - это обобщённая структура для зачётов и экзаменов, которая содержит в себе поля с названим работы и оценкой за неё.

Таким образом получается, что наша БД хранит в себе сразу две сущности, одна из которых привязана к другой.

repository.c

Данный файл реализует методы заголовочного файла, выступая в виде интерфейса взаимодействия с БД, поэтому в данном разделе я буду описывать содержимое их обоих.

str_write и int_write главные методы записи информации в бинарный файл в зависимости от типа передаваемой информации.

str_read и int_read соотвественно в переданный им указатель записывают строку либо число, считанное из файла.

add_student уже более высокоуровневый метод, ставший надстройкой над предыдущими. Он позволяет взаимодействовать с БД на уровне сущностей, а не записи отдельных байтов строк и чисел. Он использует методы записи для сохранения личной информации переданного студента. Если студент уже сдал экзамены (указатель на массив экзаменов не NULL), то вызывается метод add_test. После записи результатов экзаменов ставится знак-разделитель ‘;’. Он позволит находить конец описания экзаменов и начала записей о следующем студенте.

add_test записывает поля переданного экзамена в файл.

get_student в переданный указатель сначала запишет считанные поля из файлов, а затем, пока не обнаружит символ ‘;’, будет считывать методом get_test результаты экзаменов и также их записывать в указатель.

add_test_to_student - это процедура еще более высокого уровня, позволяющая устанавливать связи между структурами студентов и их экзаменов. Он позволяет к уже созданному студенту добавить созданный экзамен.

Описав интерфейс взаимодействия с БД, мы можем приступить к созданию исходного кода программ, использующих методы из repository.h и выполняющих действия из условий курсового проекта.

create.c

Данный файл инициализирует БД, загружает в память в режиме бинарной записи файл. Затем заполняет поля 11 структур, связывает их между друг другом и записывает их методом add_student в файл.

print.c

Задача БД не только сохранять информацию, но и также выводить её в формате, удобном для человека. Данную задачу выполняет программа print.c. Она открывает файл в режиме чтения и в цикле выгружает из него методом get_student информацию о каждом студенте в указатель. Сначала она составляет таблицу со строго описанными полями личной информации о студенте. Затем итеративно проходится по сданным студеном экзаменам, отрисовывая ячейки таблицы в нужном размере и количестве.

modifying.c

Данная программа открывает бинарный файл в режиме добавления новых байтов.

Сначала она выделяет память под указатель структуры студента. Потом, взаимодействуя с пользователем через консольный ввод заполняет поля личной информации и, по желанию, заполняет массив экзаменов, также использую поток ввода. Таким образом в конце мы получаем полноценную сущность, которую легко сохранить методом add_student. Как видите, такой метод надстройки над более простыми методами значительно сокращает количество написанных строк кода.

analysis.c

Алгоритм поиска студентов по заданному условию прост. Сначала я записываю в переменную номер группы, переданный через флаг -p и в зависимости от наличия флага -f делаю булеву переменную fflag истинной или ложной. Если fflag истинно, то мы будем выводить таблицу с найденными студентами по условию.

Программа в цикле проходится по всем студентам. Если студент имеет номер группы как в переменной p и его пол женский, то начинаем проход по экзаменам студентки и считаем количество оценок «отлично». Если в итоге только одна оценка «отлично», то методом print_student печатаем информацию о студенте.

В конце выводим количество всех студентов, которые подошли под условие.

Выводы.

Результатом выполнения курсового проекта стало создание небольшого «комплекса» программок, позволяющих взаимодействовать через описанный интерфейс с БД, записанной в бинарном файле. Таким образом мы изучили самые базовые основы создания БД и взаимодействие с ними. Как и в крупных реляционных БД, моя база имеет файлы интерфейса. Это можно сравнить с JDBC для PostgreSQL.

У созданной программы есть места, в которых её можно улучшить. Например добавить упрощенный интерпретатор SQL запросов, превратить её в настоящую реляционную БД с таблицами и несколькими типами данных. Но создание такое курсового проекта очень по времени затратное задание.

Создание последовательной структуры данных стало для меня ценным опытом по работе с файлами. Мне также очень нравится работать с памятью ПК, указателями и ссылками. Это вносит в процедурное программирование оттенки объектно ориентированного, которое мне очень нравится.