Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ

КАФЕДРА ИУ7

ОТЧЕТ ПО ЛР2 ТИПОВ И СТРУКТУР ДАННЫХ, Вариант 3

Студент Андреев Александр Алексеевич
фамилия, имя, отчество

Группа ИУ7-34Б

Тип практики учебная

Студент _____ _____
подпись, дата фамилия, и.о.

Преподаватель _____ _____
подпись, дата фамилия, и.о.

Оценка _____

9 октября 2020

Оглавление

1. Цель работы

В качестве цели работы ставится приобретение навыков работы с типом данных “запись” (“структура”), содержащим вариантную часть, и с данными, хранящимися в таблицах. оценить относительную эффективность программы (в процентах) по времени и по используемому объему памяти в зависимости от используемого алгоритма и объема сортируемой информации.

2. Описание условия задачи

Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами (объединениями)). Упорядочить данные в ней по возрастанию ключей, двумя алгоритмами сортировки, где ключ – любое невариантное поле (по выбору программиста), используя: а) саму таблицу, б) массив ключей. (Возможность добавления и удаления записей в ручном режиме обязательна). Осуществить поиск информации по варианту.
Имеются описания:
Type жилье = (дом, общежитие);
Данные : Фамилия, имя, группа, пол (м, ж), возраст, средний балл за сессию, дата поступления
адрес :
дом : (улица, №дома, №кв);
общежитие : (№общ., №комн.);
Ввести общий список студентов.
Вывести список студентов, живущих в общежитии указанного года поступления

3. Описание ТЗ, включающее внешнюю спецификацию

а. Описание исходных данных

Исходные данные программа получает из считанного текстового файла, где

- каждая запись должна находиться в отдельной строке
- в первой строчке файла находится целое число, равное количеству студентов в файле
- каждая запись имеет собственные ограничения по размеру и спецификацию
- каждая запись находится в отдельной строчке
- после последней записи находится пустая строка

Ограничения на записи во входном файле:

1. Длина фамилии не более 256 символов
2. Длина имени не более 256 символов
3. Длина названия улицы адреса дома не более 256 символов
4. Пол студента описывается, как 0/1 (Ж/М)
5. Возраст студента число натуральное не более 100 лет
6. Средний балл студента число вещественное положительное не более 100
7. Номер дома или общежития число натуральное не более 1000
8. Номер комнаты или квартиры число натуральное не более 9999
9. День - число натуральное не более 31

10. Месяц - число натуральное не более 12
11. Год - число натуральное не более 9999

b. Описание задачи, реализуемой программой

Программа должна уметь совершать следующие действия с входящим файлом:

- Закрывать программу по команде **EXIT**
- Выводить список всех студентов по команде - **PRINT_ALL**
- Выводить отфильтрованных студентов по дате зачисления в университет - **PRINT_FILTERED_STUDENTS**
- Сортировать студентов таблицы по ключам (accommodation, surname, name, gender, age, average_score_per_session, street, house_or_campus_number, flat_or_room_number, date_in) и вывести сравнительную информацию по сортировкам по ключам и общей по команде, вывод таблицы ключей - **SORT_TABLE_BY_KEY**
- Добавлять запись в таблицу по команде - **ADD_NOTE**
- Удалять запись из таблица по команде - **DELETE_NOTE**
- Сохранять примененные изменения обратно в файл - **SAVE_TABLE**

c. Способ обращения программы

Скомпилированная программа запускается командой **“./a.out X”** на Unix и **“./a.exe X”** на Windows, где X - это названия входного текстового файла, например **“test_1.txt”**

При запуске программа должна вывести инструкцию по своему применению:
“Программа обработки текстовых данных запущена.
Программа умеет выполнять команды:
EXIT - Закрытие программы
PRINT_ALL - Печать всех студентов
PRINT_FILTERED_STUDENTS - Печать только студентов по ключам
SORT_TABLE_BY_KEY - Сортировка таблицы по ключу, вывод таблицы ключей и сравнение работы алгоритмов
ADD_NOTE - Добавление записи в таблицу
DELETE_NOTE - Удаление записи из таблицы
SAVE_TABLE - Сохранение изменений
”

Далее после ввода соответствующей команды программа должна выполнять команды пользователя.

- **Команда EXIT**
  При вводе команды **EXIT** программа завершает работу без каких-либо предупреждений.
- **Команда PRINT_ALL**
  Программа выводит данные входного файла в виде таблицы в консоль, где в обязательном порядке должны быть отображены все поля стратруктруы и подписаны столбцы.
- **Команда PRINT_FILTERED_STUDENTS**
  Программа должна запросить у пользователя ключ фильтрации “Укажите дату

поступления через пробел (ДД ММ ГГ): “, после ввода которых пользователю выводится на экран отфильтрованная таблица либо сообщается о том, что ничего не найдено.

- **Команда SORT_TABLE_BY_KEY**
  Программа должна вывести список возможных для фильтрации ключей (accommodation, surname, name, gender, age, average_score_per_session, street, house_or_campus_number, flat_or_room_number, date_in), после чего запросить ключ для сортировки, далее программа сортирует таблицу и выведет сравнительную информацию по двум примененным к исходному массиву структур сортировкам. Команда выведет таблицу ключей по отсортированному полю, где в первом столбце номер в исходном и во втором значение.
- **Команда ADD_NOTE**
  Программа должна запросить данные всех полей необходимой для добавления записи в таблицы, после чего должна добавить. Причем запросы по заполнению данных общежития и дома должны отличатся по приглашению ввода.
- **Команда DELETE_NOTE**
  Программа должна запросить данные всех полей необходимой для удаления записи в таблицы, после чего должна сообщить о том, что их не существует. либо удалить их. Причем запросы по удалению данных общежития и дома должны отличатся по приглашению ввода.
- **Команда SAVE_TABLE**
  Программа должна сохранить таблицу в тот файл, из которого ее считала при запуске.

При наличии аварийных ситуаций и ошибок пользователя программа должна вывести соответствующее сообщение и не должна завершить свою работу абортно.

d. Описание возможных аварийных ситуаций и ошибок пользователя

В представленной ниже **Таблице 1** отражены действия программы при различных возможных, допущенных пользователем, при ее использовании ошибках.

Табл.1

Действие программы при различных ошибках		
№	Действие пользователя	Реакция программы
1	Указанного файла пользователем для чтения не существует.	Программа выведет сообщение об ошибке “Указанного файла для чтения не существует.”
2	Указанный файл пользователем оказывается пустым	Программа выведет сообщение об ошибке “Указанный файл пустой.”
3	При чтение данных из файла возникнут ошибки	Программа выведет сообщение об ошибке “При чтение данных

		из файла возникли ошибки”
4	Пользователь вводит некорректные команды (те команды, которые не представлены в списке меню)	Программа выведет сообщение об ошибке “Вы ввели некорректный ключ. Попробуйте заново...”
5	В случае, если программа не нашла записей при фильтрации	Программа выведет сообщение об ошибке “Ничего не найдено….”

4. Описание внутренних СД

Считанные из консоли числа внутри программы хранятся в виде структуры **number,** которая представляет из себя (Рис. 1)

Рис. 1

```
typedef struct date
{
	int day;
	int month;
	int year;
} date;

typedef struct student_personal_information
{
	char surname[INPUT_STRING_MAX_SIZE];
	char name[INPUT_STRING_MAX_SIZE];
	int gender;
	int age;
	double average_score_per_session;
	struct date receipt_date;
} student_personal_information;

typedef struct social_home_address
{
	int house_number;
	int room_number;
} social_home_address;

typedef struct social_home
{
	int is_social_home;
	struct student_personal_information student;
	struct social_home_address 			address;
} social_home;

typedef struct private_home_address
{
	char street[INPUT_TABLE_INFORMATION_MAX_SIZE];
```

```
        int house_number;
        int flat_number;
} private_home_address;

typedef struct private_home
{
        struct student_personal_information student;
        struct private_home_address             address;
} private_home;

typedef union students_accommodation_information
{
        int is_social_home;
        struct social_home  social_home;
        struct private_home private_home;
} students_accommodation_information;

typedef struct sort_key
{
        int position;
        double double_value;
} sort_key;
```

5. Алгоритм

Основная программа

Вывод меню на экран. В случае, если Пользователь выбирает пункт:

EXIT:

программа завершает работу

PRINT_ALL:

функция output_all_students (см. функции)

PRINT_FILTERED_STUDENTS:

функция output_filtered_students (см. функции)

SORT_TABLE_BY_KEY:

функция sort_students_by_key (см. функции)

ADD_NOTE:

функция add_note (см. функции)

DELETE_NOTE:

функция delete_note (см. функции)

SAVE_TABLE:

функция save_information_into_file (см. функции)

Функции

void output_all_students(union students_accommodation_information *input_table_information, int *input_table_information_size);

Функция вывода всех студентов на экран получает на вход массив **union students_accommodation_information *input_table_information** и его размер **int *input_table_information_size**. При положительном размере выводит на экран сначала шапку таблицы, далее в цикле вызывает вывод каждой записи в консоль.

void output_filtered_students(union students_accommodation_information *input_table_information, int input_table_information_size);
Функция вывода всех студентов на экран получает на вход массив **union students_accommodation_information *input_table_information** и его размер **int *input_table_information_size**. При положительном размере функция запрашивает у пользователя дату поступления в университете в указанном выше формате, после чего циклически проходит массив и при совпадении даты выводит информацию на экран.

void sort_students_by_key(union students_accommodation_information *input_table_information, int input_table_information_size);
Функция вывода всех студентов на экран получает на вход массив **union students_accommodation_information *input_table_information** и его размер **int *input_table_information_size**. При положительном размере функция запрашивает у пользователя ключ для сортировка таблицы, после чего создается сортирует массив sort_key key_sort_table[INPUT_TABLE_INFORMATION_MAX_SIZE], в который записываются положение в исходном массиве и его значение, далее происходит сортировка по ключам с засеканием времени и аналогично, но уже с исходным массивом структуры **union students_accommodation_information *input_table_information** по введенному ключу также производится сортировка и засекает время работы. В результате выводится сравнительная информацию по обеим сортировкам. Выводится таблица ключей.

int add_note(union students_accommodation_information *input_table_information, int *input_table_information_size);
Функция вывода всех студентов на экран получает на вход массив **union students_accommodation_information *input_table_information** и его размер **int *input_table_information_size**. При положительном размере функция запрашивает у пользователя данные всех полей:

если Общежитие

Запрашивает Фамилию, Имя, Пол, Возраст, Средний балл за семестр, Номер общежития, Номер комнаты, Даты зачисления.

если Дом

Запрашивает Фамилию, Имя, Пол, Возраст, Средний балл за семестр, Улицу дома, Номер дома, Номер квартиры, Даты зачисления.

После успешного получения данных программа расширяет массив **union students_accommodation_information *input_table_information** и добавляет в него запись.

int delete_note(union students_accommodation_information *input_table_information, int *input_table_information_size);
Функция вывода всех студентов на экран получает на вход массив **union students_accommodation_information *input_table_information** и его размер **int *input_table_information_size**. При положительном размере функция запрашивает у пользователя данные всех полей:

если Общежитие

Запрашивает Фамилию, Имя, Пол, Возраст, Средний балл за семестр, Номер общежития, Номер комнаты, Даты зачисления.

если Дом

Запрашивает Фамилию, Имя, Пол, Возраст, Средний балл за семестр, Улицу

дома, Номер дома, Номер квартиры, Даты зачисления.
После успешного получения данных программа при наличии текущего элемента в массиве **union students_accommodation_information *input_table_information** и добавляет его.

void save_information_into_file(char const *argv[], union students_accommodation_information *input_table_information, int *input_table_information_size);
Функция вывода всех студентов на экран получает на вход аргументы с данными для записи в файл, массив **union students_accommodation_information *input_table_information** и его размер **int *input_table_information_size**. После чего записывает в первую строчку размер массива, далее циклически проходит по массиву и записывает каждую запись в файл с окончание “\n”.

6. Набор тестов с указанием, что проверяется

В представленных ниже **Таблица 2** отражены тестирование устойчивости работы консольного меню программы и демонстрация устойчивости работы программы к различному типу выполняемых с ней операций пользователем соответственно.

Табл. 2

Тестирование устойчивости работы программы			
№	Ввод пользователя	Реакция программы	Что проверяется данной операцией?
1	TEST_MENU	Программа выводит “Вы указали неверную команду. Попробуйте еще раз...”	Устойчивость программы к вводу неверной команды меню
2	При вводе некорректного значения в любом пункте меню, где требуется дополнительный ввод (DELETE_NOTE, ADD_NOTE, SORT_TABLE_BY_KEY, PRINT_FILTERED_STUDENTS)	Программа выводит “Вы указали неверную команду. Попробуйте еще раз...”	Устойчивость программы к вводу неверной команды внутри пунктов меню
3	Пользователем введен	Программа выводит “Введенный файл оказался	Устойчивость к открытию несуществующего файла

	несуществую щий файл	пустым или не может быть прочитан. Исправьте и повторите попытку.”	
4	Введенный пользователем файл оказался пустым	Программа выводит “Введенный файл оказался пустым или не может быть прочитан. Исправьте и повторите попытку.”	Устойчивость к чтению пустого файла
5	Количество аргументов при запуске программы оказалось меньше или больше допустимого	Программа выводит “Допущена ошибка в аргументах при запуске. Исправьте и повторите попытку.”	Устойчивость к запуску программы с неверными аргументами запуска
6	Данные в открытом пользователем файле введены некорректно	Программа выводит “При чтение данных из файла возникли проблемы. Исправьте файл и повторите попытку.”	Корректная работа программы с выводом сообщения об ошибке при неверном содержании открываемого файла
7	Фильтрация студентов дала нулевой результат	Программа выводит “Ничего не найдено...”	Корректная работа при отсутствии найденных студентов во время фильтрации

7. Выводы по проделанной работе

Рис. 2

```
Информация по сортировкам:
1. Сортировка пузырьком
        Сортировка не по ключам: 0.000319(сек)
        Сортировка по ключам: 0.000018(сек)
        Сортировка по ключам быстрее на: 94.357367%
2. Сортировка qsort
        Сортировка не по ключам: 0.000042(сек)
        Сортировка по ключам: 0.000003(сек)
        Сортировка по ключам быстрее на: 92.857143%
```

Подобное решение с использование Union в программе позволяет более структурно подходить к решению задачи. После использование чего я понял, что Union позволит делегировать

написание отдельных модулей обращения со структурой разным программистам, таким образом сократив временные издержки.

На основе Рис. 2, где указан анализ по сравнению сортировок можно сказать, что когда сортируется таблица ключей, то экономится время, так как перестановка записей в исходной таблице отсутствует. Он заметен при большмй размере таблиц. Выбор данных из основной таблицы в порядке, определенном таблицей ключей,также замедляет вывод данных.

8. Контрольные вопросы

1.Как выделяется память под вариантную часть записи?
Размер памяти, который выделяется под вариантную часть представляет из себя максимальный размер одной из объединенных структур. Например, у меня в union находятся две структуры, одна из которых больше другой на 256 Байт (Память, выделенная на поле street). Поэтому при выделение памяти для массива структур размера N будет выделено N * (размер большей структуры)

2.Что будет, если в вариантную часть ввести данные, не соответствующие описанным?
Программа выдаст Undefined behavior. Результат системы будет трудно предсказуем, возможно произойдет приведение типов. Поэтому необходимо делать верификацию каждого введенного параметра, как это я выполнил в работе, чтобы программа работала корректно.

3.Кто должен следить за правильностью выполнения операций с вариантной частью записи?
За правильностью выполнения операций с вариантной частью записи должны следить флаг и функции верификации ввода и обработки.

4.Что представляет собой таблица ключей, зачем она нужна?
Таблица ключей представляет из себя таблицу с номерами записи в исходном массиве и ключа, по которому происходит сортировка. Такое решение ускоряет время работы программы, когда структура большая из-за того, что не происходит постоянной операции перезаписи и сама структура весит значительно меньше, потому что содержит только два поля.

5.В каких случаях эффективнее обрабатывать данные в самой таблице, а когда – использовать таблицу ключей?
Данные эффективнее обрабатывать в таблице, когда размер и память, выделяемая на структуру мала, либо же количество сортируемых записей невелико. Сортировку с таблицей ключей, как мы выяснили в лабораторной, необходимо в тот момент, когда структура большая и невыгодно работать с ее “тяжелыми” полями - это значительно усложняет работу и ее время.

6.Какие способы сортировки предпочтительнее для обработки таблиц и почему?
Конечно, если мы имеет структуру, состоящую из двух элементов, то бесспортно создавать таблицу ключей не имеет смысла. Но если в рабочей структуре 5 и более полей, то таблица ключей позволит ускорить работу программу и облегчить написание кода с той точки зрения,

что не нужно производить процессе переприсваивания всех полей структуры, после каждой удачной операции сравнения элементов.