Министерство науки и высшего образования 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 «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 ПО ДИСЦИПЛИНЕ: ТИПЫ И СТРУКТУРЫ ДАННЫХ

Записи с вариантами. Обработка таблиц

Студент **Попов Ю.А.**

Группа **ИУ7-32Б**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент _____ **Попов Ю.А.**

Преподаватель _____ **Барышникова М.Ю.**

2024

Оглавление

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, просмотр таблицы, просмотр таблицы в порядке расположения таблицы ключей.

Осуществить поиск информации по варианту. Вывести список студентов указанного года поступления, живущих в съемном жилье, стоимостью меньше указанного

Описание технического задания

Файл с данными:

В файле последовательно хранятся записи, каждая запись начинается с новой строки. Поля записи содержат в себе информацию для заполнения структуры, разделитель между полями - символ ;.

Входные данные:

Пользователь вводит целое число-команду от 0 до 11. Программа выполняет операции, в зависимости от выбранной опции. Чтобы корректно завершить программу, необходимо воспользоваться соответствующей командой. Необязательным параметром выступает аргумент для запуска приложения, который указывает программе на файл с данными.

Выходные данные:

В зависимости от выбранного действия, пользователь получает разные выходные данные. Общими остаются уведомления об успешности или ошибки выполнения действия, и вывод информации на экран.

Действие программы:

Программа работает до тех пор, пока пользователь не введет число 0. Программное обеспечение позволяет ввести команды от 0 до 11, в случае если команда выходит за этот диапазон, или не подходит по типу данных, выводится соответствующее сообщение, и программа завершает работу. Разработанное программное обеспечение работает с таблицами и позволяет провести с ними такие действия, как: сортировка различными методами, поиск по ключу, добавление и удаление записей.

Обращение к программе:

Программа запускается через терминал, в командной строке, с помощью команды ./app.exe. При обращении с программой, пользователь может использовать параметр запуска, в котором нужно указать путь до файла с данными. Если параметров нет, то программа будет использовать файл по умолчанию. После запуска. пользователю будет доступно меню программы.

Аварийные ситуации:

1. ERR_FILENAME – Ошибка в имени файла. Возможно файла не существует. Происходит при безуспешном открытии файла.
   Код ошибки – 2
2. ERR_UNKNOWN – Неизвестная ошибка.

Код ошибки – 100

3. ERR_OPERATION_COUNT – Ошибка при выборе операции, выход за допустимый диапазон значений.

   Код ошибки – 4.

4. ERR_OPERATION_INPUT – Ошибка при выборе операции, ошибка считывания или текстовый ввод.

   Код ошибки – 3.

5. ERR_STRING_OVERFLOW – Переполнение строки, ошибка возникает при некорректном вводе строки, если ввод длиннее буфера.

   Код ошибки – 5.

6. ERR_EMPTY_DATABASE – Пустой файл с данными, ошибка возникает при инициализации переменных с таблицами, когда данные отсутствуют.

   Код ошибки – 6.

7. ERR_MEMORY_ALLOCATION – Ошибка выделения памяти.

   Код ошибки – 7.

8. ERR_FILE_ORDER_OVERFLOW – Ошибка в записи выделения памяти, запись больше буффера.

   Код ошибки – 9.

9. ERR_TOO_LONG_FIELD – Одно из полей в записи длиннее ограничения .

   Код ошибки – 10.

10. ERR_CONVERTATION_DOUBLE – Ошибка при считывании числа. Может произойти из-за неверной записи числа.

    Код ошибки – 11.

11. ERR_UNKNOWN_TYPE – Ошибка при считывании типа жилья. Случается, когда тип не равен M, H или R.

    Код ошибки – 12.

12. ERR_UNKNOWN_GENDER – Пол записан с ошибкой. Можно ввести только M - мужской и F - женский.

Код ошибки – 13.

13. ERR_TOO_MANY_STUDENTS – Слишком длинный файл с данными, программа допускает ввод до 5000 записей.

Код ошибки – 14.

14. ERR_FILE_SAVE – Ошибка при сохранении файла.

Код ошибки – 15.

15. ERR_EMPTY_OUTPUT– Пустой вывод, может случиться, когда пользователь пытается удалить последнюю запись.

Код ошибки – 18.

Во всех нештатных ситуациях, программа уведомляет о месте, где произошла ошибка.

Описание структуры данных

Информация о студентах, проживающих в квартире хранится в структуре flat_t. Ее поля:

street – улица проживания

house_number – дом

flat_number – дом

```
struct flat_t
        {
            char street[MAX_STREET_LEN];
            short house_number;
            short flat_number;
        };
```

листинг 1. Структура flat_t

Информация о студентах, проживающих в общежитии хранится в структуре hostel_t. Ее поля:

hostel_number – номер общежития

fhostel_number – номер комнаты

```
struct hostel_t
        {
            short hostel_number;
            short hostel_flat;
        };
```

листинг 2. Структура hostel_t

Информация о студентах, проживающих в арендной квартире хранится в структуре hostel_t. Ее поля:

hostel_number – номер дома

fhostel_number – номер квартиры

cost – стоимость аренды

```
struct rental_t
        {
            char street[MAX_STREET_LEN];
            short house_number;
            short flat_number;
```

```
            int cost;
        };
```

листинг 3. Структура rental_t

Информация о вариантной части хранится в объединении apartaments_type . Его поля:

flat – структурная переменная типа flat_t

hostel – структурная переменная типа hostel_t

rental – структурная переменная типа rental_t

```
union apartaments_type
    {
        struct flat_t flat;
        struct hostel_t hostel;
        struct rental_t rental;
    }:
```

листинг 4. Объединение apartaments_type

Основная структура программы - students_t – содержит в себе, всю информации о студенте.

Описание полей:

surname – фамилия студента

name – имя студента

group – группа обучения

gender – пол студента

average_score – средний балл

admission_year – год поступления

type – тип жилья

apartaments_type – Объединение с информацией о вариантной части.

```
typedef struct
{
    char surname[MAX_NAME_LEN];
    char name[MAX_NAME_LEN];
    char group[MAX_GROUP_NAME];
    char gender; // M - Male, F - Female
    double average_score;
    int admission_year;
```

```
    char type; // F - flat, H - hostel, R - rental
    union apartaments_type aparts;
} students_t;
```

Листинг 5. Структура students_t

Структура для таблицы ключей приведена на листинге 6.

Описание полей:

index_src - Индекс в исходной таблице

surname - Фамилия студента (ключ сортировки)

```
 typedef struct
{
    size_t index_src;
    char *surname;

} table_t;
```

Листинг 6. Структура table_t

Структура для проведения замерного эксперимента приведена на листинге 7.

Описание полей:

size – длина сортируемого массива

time_def_mysort – время сортировки исходной таблицы методом модифицированной сортировки пузырьком

time_def_qsort – время сортировки исходной таблицы встроенным методом быстрой сортировки

time_key_mysort – время сортировки таблицы ключей методом модифицированной сортировки пузырьком

time_key_qsort – время сортировки таблицы ключей встроенным методом быстрой сортировки

```
typedef struct
{
    size_t size;
    long long time_def_mysort;
    long long time_def_qsort;
```

```
    long long time_key_mysort;
    long long time_key_qsort;
} measuring;
```

Листинг 7. Структура `measuring`

Описание алгоритма

1. После запуска программы выводит приглашение к вводу и меню.
2. Программа принимает номер операции, и запускает соответствующие действия в соответствии с номер команды
3. Если пункт связан с записью/вводом данных, то происходит валидация данных и выполняется запрос
4. Если пункт связан с сортировкой исходной таблицей, то таблица сортируется методом быстрой сортировки и выводится на экран
5. Если пункт связан с сортировкой таблицы ключей, то сначала создается таблица ключей, после этого она сортируется, самостоятельно-реализованным методом, а после этого выводится на экран исходная таблица, отсортированная с помощью таблицы ключей.
6. Если пользователь вводит 0, то выполняется сохранение таблицы в текстовый файл.

Описание основных функций

```
int input_string(char *string)
```

Функция реализует ввод строки из stdin.

Входные параметры: указатель на строку.

Выходные параметры: измененная строка.

Возвращаемый результат: код возврата.

```
int database_import_students(FILE *file, students_t *students, size_t
*count)
```

Функция реализует запись из текстового файла в массив структур

Входные параметры: указатель на файловую переменную, указатель на массив структур типа students_t, размер массива.

Выходные параметры: заполненный массив students_t.

Возвращаемый результат: код возврата.

```
int input_operation(operations_t *operation)
```

Функция реализует ввод операции

Входные параметры: указатель на enum-переменную, содержащая в себе, наименование операции.

Выходные параметры: номер операции

Возвращаемый результат: код возврата.

```
void database_all_print(students_t *array_students, size_t count)
```

Функция выводит таблицу студентов на экран

Входные параметры: массив студентов, размер массива.

Выходные параметры: вывод таблицы на экран.

```
int database_append(students_t *array_students, size_t *count)
```

Функция реализует ручное добавление студента в массив студентов.

Входные параметры: массив структур студентов, указатель на длину массива.

Выходные параметры: измененный массив структур студентов.

Возвращаемый результат: код возврата.

```
int database_delete_student(students_t *array_students, size_t *count)
```

Функция удаляет запись о студентах, чьи имена и фамилии равны запросу

Входные параметры: массив студентов, размер массива.

Выходные параметры: вывод измененной таблицы на экран.

Возвращаемый результат: код возврата.

```
int database_search(students_t *array_students, size_t count)
```

Функция поиска по таблице. Выводит список студентов указанного года поступления, живущих в съемном жилье, стоимостью меньше указанного.

Входные параметры: массив студентов, размер массива.

Выходные параметры: вывод соответствующих записей на экран.

Возвращаемый результат: код возврата.

void create_key_table(table_t *key_table, students_t *array, size_t count)

Функция создает таблицу ключей по исходной таблице

Входные параметры: указатель на таблицу ключей, массив студентов, размер массива.

Выходные параметры: заполненная таблица ключей

void key_table_print(table_t *key_table, size_t count)

Функция выводит на экран таблицу ключей

Входные параметры: указатель на таблицу ключей, размер таблицы.

void mysort(void *arr, size_t number, size_t width, int (*compare)(const void *, const void *))

Функция сортировки, аналогичная сигнатуре qsort. Реализует метод модифицированной сортировки пузырьком.

Входные параметры: массив, длина массива, объем памяти, занимаемым одним элементов, указатель на функцию сравнения двух элементов.

Выходные параметры: отсортированный массив.

void print_table_with_keys(students_t *students_arr, table_t *table, size_t count)

Вывод таблицы с помощью таблицы ключей.

Входные параметры: массив студентов, таблица ключей, длина массива.

Выходные параметры: выведенный на экран отсортированный массив студентов.

Набор тестов

№	Описание теста	Пользовательский ввод	Вывод
1	Некорректный ввод операции	abc	Ошибка при выборе операции
2	Некорректный ввод операции. Выход за диапазон	10000	Ошибка, введите число от 1 до 12
3	Некорректный ввод операции. Отрицательное число	-100	Ошибка, введите число от 1 до 12
4	Корректный файл		Вывод пользовательского меню
5	Корректное добавление структуры	Nikiforov;Vladimir;IU7-73B;M;4.75;2023;H;13;5072	Успешное добавление
6	Некорректное добавление структуры	Nikiforov;Vladimir;IU7-73B;MMMMM;4.75;2023;H;13;5072	"Ошибка в написании пола. Можно указать только M - Мужской и F - Женский"
7	Удаление последней структуры		Ошибка, пустой вывод
8	Переполнение строки при пользовательском вводе	Trofimova;Marina;IU1-72BBBBBBB;F;5.5;2024;H;8;2169	Ошибка, одно из полей слишком длинное
9	Некорректный файл, слишком много студентов		Ошибка, максимальное количество студентов - 1000

10	Неизвестный тип жилья при вводе студента	Trofimova;Marina;IU1-72B;U;5.5;2024;H;8;2169	Ошибка при вводе типа жилья. Можно записать только “F”/”R”/”H”

Временные замеры

Временные замеры осуществлялись с помощью библиотеки time.h. Каждый запуск, таблица заново считывалась из памяти. Каждый размер запускался 10 раз, в память помещалось среднее значение этих измерений. В таблице 1 представлено время работы (в наносекундах) каждого алгоритма.

Размер файла	Быстрая сортировка		Модифицированная сортировка пузырьком	
	Исходная таблица	Таблица ключей	Исходная таблица	Таблица ключей
16	100	100	4100	800
32	400	300	29100	5400
64	2100	500	90700	18900
128	2400	700	248300	46700
256	5100	1100	529400	104600
512	17400	3900	1785300	398300
1024	34700	9800	6934700	1561300
2048	57000	21400	29487700	6649800

Таблица 1. Сравнение времени работы исследуемых алгоритмов.

Объем занимаемой памяти (байты)

Объем, занимаемый структурой представлен в таблице 2.

Размер файла	Исходная таблица	Таблица ключей
16	1664	384
128	13312	3027
256	26624	6144

512	53248	12228
1024	106496	24576
2048	212992	49152

Таблица 2. Оценка занимаемой памяти, относительно размера.

Оценка эффективности

Для того, чтобы оценить эффективность исследуемого метода, сравним его эффективность относительно обычной сортировки таблицы. Сравнение представлено в таблице 3.

Размер	Отношение быстрой сортировки основной таблицы к таблице ключей	Отношение модифицированной сортировки основной таблицы к таблице ключей
16	1.000	4.556
32	1.333	5.389
64	4.200	4.799
128	3.428	5.317
256	4.636	5.061
512	4.461	4.482
1024	3.541	4.442
2048	2.664	4.435

Таблица 3. Отношение времени сортировки таблицы студентов ко времени сортировки таблицы ключей

Ответы на контрольные вопросы

1. Как выделяется память под вариантную часть записи?

Память выделяется под самую большую структуру объединения, в моем случае это rental_t.

2. Что будет, если в вариантную часть ввести данные, несоответствующие описанным?

Ничего не произойдет, так как программа проверяет все данные на корректность.

3. Кто должен следить за правильностью выполнения операций с вариантной частью записи?

Программист, потому что только он обладает достаточным объемом знаний, для определения ошибок в вариантной части. Компилятор так возможностью не обладает

4. Что представляет собой таблица ключей, зачем она нужна?

Таблица ключей представляет собой совокупность индекса элемента в исходной таблице, индекса элемента в таблице ключей и ключ поиска (в моем случае - фамилия студента)

5. В каких случаях эффективнее обрабатывать данные в самой таблице, а когда – использовать таблицу ключей?

Использование эффективно, когда размер структуры таблицы ключей кратно меньше размера исходной структуры, а также при достаточное длине массива.

6. Какие способы сортировки предпочтительнее для обработки таблиц и почему?

Как всегда наиболее предпочтительные методы сортировки - сортировки, имеющие наименьшую вычислительную сложности. В лучшем случае использовать quick_sort со сложностью до O(N^2) или merge sort с константной сложностью O(n * log(n)).

ВЫВОД

Объединение в языке СИ помогает уменьшать размеры типов и структур данных. В ходе работы, было установлено, что сортировка ключей эффективнее сортировки самой таблицы, если ключ достаточно мал и быстро сравним. То есть выделяя дополнительную память можно достичь серьезных прибавок к производительности. Выбор сортировки существенно влияет на скорость работы программы.