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 «ЗАПИСИ С ВАРИАНТАМИ, ОБРАБОТКА ТАБЛИЦ»

Студент Тузов Даниил Александрович

Группа ИУ7 – 32Б

Преподаватель Барышникова Марина Юрьевна
Силантьева Александра Васильевна

2023 г.

Оглавление

ОПИСАНИЕ УСЛОВИЯ ЗАДАЧИ.....3

ОПИСАНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ.....3

ОПИСАНИЕ СТРУКТУР ДАННЫХ.....4

ОПИСАНИЕ АЛГОРИТМА.....6

ОПИСАНИЕ ФУНКЦИЙ…...……..6

НАБОР ТЕСТОВ.....8

ЗАМЕРЫ ВРЕМЕНИ РАБОТЫ……10

ОБЪЕМ ИСПОЛЬЗУЕМОЙ ПАМЯТИ..……11

ОЦЕНКА ЭФФЕКТИВНОСТИ……11

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ…….12

ВЫВОД……..12

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:
Тип жилье = (дом, общежитие, съемное);
Данные :
Фамилия, имя, группа, пол (м, ж), возраст, средний балл за сессию, дата поступления
адрес:
дом : (улица, №дома, №кв);
общежитие : (№общ., №комн.)
съемное : (улица, №дома, №кв, стоимость);
Ввести общий список студентов.
Вывести список студентов, указанного года поступления, живущих в съемном жилье.

ОПИСАНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ

Входные данные:
Сначала пользователь выбирает действие — целое число от 0 до 6. В зависимости от выбора действия, пользователь вводит либо название входного/выходного файла, либо данные о студенте, либо строку, которую надо удалить

Выходные данные:
Выводится результат работы при выборе каждого действия. Либо исходная таблица с данными, либо отсортированная.

Описание программы:
Сортировка таблицы данных о студенте по фамилии студента.

Способ обращения к программе:
Взаимодействие через консоль.

Аварийные ситуации:

1. Любой некорректный ввод данных в программе
   Код ошибки — 1
2. Ошибка при вводе слишком длинной строки данных о студенте
   Код ошибки — 2
3. Ошибка открытия файла
   Код ошибки — 3
4. Ошибка в вводе какого-то поля (переполнение ввода)
   Код ошибки — 4
5. Ошибка переполнения таблицы
   Код ошибки — 5

ОПИСАНИЕ СТРУКТУР ДАННЫХ

Структурный тип для квартиры
street – улица
build_num – номер дома
flat_num – номер квартиры

```
typedef struct house {
char street[STREET_LEN + 1];
size_t build_num;
size_t flat_num;
} house_t;
```

Структурный тип для общежития
build_num – номер дома
flat_num – номер квартиры

```
typedef struct dormitory {
size_t build_num;
size_t flat_num;
} dormitory_t;
```

Структурный тип для съемной квартиры
street – улица
build_num – номер дома
flat_num – номер квартиры
cost – стоимость аренды

```
typedef struct rental {
char street[STREET_LEN + 1];
```

```
size_t build_num;
size_t flat_num;
size_t cost;
} rental_t;
```

Структурный тип даты
year – год
month – месяц
day – день

```
typedef struct date {
size_t year;
size_t month;
size_t day;
} date_t;
```

Объединение структурных типов, отвечающее за адрес проживания студента
house – собственная квартира
dorm – общежитие
rent – съемная квартира

```
typedef union adress {
house_t house;
dormitory_t dorm;
rental_t rent;
} adress_t;
```

Перечисляемый тип, тип места жительства
HOUSE – собственная квартира
DORM – общежитие
RENT – съемная квартира

```
typedef enum house_type { HOUSE, DORM, RENT }
house_type_t;
```

Структурный тип с информацией о студенте (основная таблица)
surname – фамилия студента
name – имя студента
group – учебная группа студента
gender – пол студента
age – возраст студента
av_mark – средня оценка за сессию
date – дата поступления
type – тип места жительства
adress – адрес проживания студента

```
typedef struct student {
char surname[SURNAME_LEN + 1];
char name[NAME_LEN + 1];
```

```
char group[GROUP_LEN + 1];
char gender[GENDER_LEN + 1];
size_t age;
double av_mark;
date_t date;
house_type_t type;
adress_t adress;
} student_t;
```

Таблица ключей
surname – фамилия студента
index – индекс поля в исходной таблице

```
typedef struct key {
char surname[SURNAME_LEN + 1];
size_t index;
} key_field_t;
```

ОПИСАНИЕ АЛГОРИТМА

1. Программа выводит пользователю меню
2. Пользователь выбирает действие (1 — загрузить исходную таблицу, 2 — вывести таблицу, 3 — добавить строку в таблицу, 4 — удалить строку из таблицы, 5 — сортировка таблицы, 6 — сортировка таблицы ключей, 0 — выход)
3. В зависимости от выбора пользователя выполняется соответствующая функция
4. Выполняется логика работы соответствующей функции
5. Пользователю снова предоставляется право выбора действия

ОПИСАНИЕ ФУНКЦИЙ

`void print_menu()` - вывод меню для пользователя

`int check_file(FILE **f, char *mods)` – проверка файла f на открытие с соответствующим ключом mods. Возвращает код ошибки

`int check_choice(int *num)` – проверка корректности выбора пользователем действия num. Возвращает код ошибки

`int stud_read(FILE *f, student_t *stud);` – считывание информации о студенте stud из файла f. Возвращает код ошибки

`int stud_read_from_terminal(student_t *stud);` – считывание информации о студенте stud из терминала. Возвращает код ошибки

`void stud_print(FILE *f, student_t *stud);` – вывод информации о студенте stud в файл f

`int upload_table(FILE *f, student_t *table, size_t *n);` – загрузка таблицы из файла f в буфер table размера n. Возвращает код ошибки

`void print_table(FILE *f, student_t *table, size_t n);` – вывод таблицы в файл f, таблица хранится в буфере table размера n

`void print_table_by_keys(FILE *f, student_t *table, key_field_t *key, size_t n);` – печать таблицы table размера n в файл f с помощью таблицы ключей key

`int add_row(student_t *table, size_t *n);` – добавление строки в таблицу table размера n. Возвращает код ошибки

`int delete_row(student_t *table, size_t *n, size_t num_str);` – удаление строки под номером num_str из таблицы table размера n. Возвращает код ошибки

`void create_table_keys(student_t *table, key_field_t *keys, size_t n);` – создание таблицы ключей key на основе таблицы студентов table, длины n

`void print_table_keys(key_field_t *keys, size_t n);` - функция печати таблицы ключей keys размера n

`void bubble_sort_table(student_t *table, size_t n);` – сортировка таблицы table размера n пузырьком

`void bubble_sort_table_keys(key_field_t *keys, size_t n);` – сортировка таблицы ключей keys размера n пузырьком

`int comp_studs(const void *a, const void *b);` – функция сравнения элементов a и b массива студентов по искомому полю. Возвращает код ошибки

`void qsort_table(student_t *table, size_t n);` – сортировка таблицы table размера n быстрой сортирвокой

`int comp_studs_by_keys(const void *a, const void *b);` – функция сравнения элементов a и b массива ключей по искомому полю. Возвращает код ошибки

`void qsort_table_keys(key_field_t *keys, size_t n);` – сортировка таблицы ключей keys размера n быстрой сортировкой

`void filter_output(student_t *table, size_t n, size_t year);` - функция фильтрации таблицы table размера n по году year

НАБОР ТЕСТОВ

№	Описание теста	Ввод пользователя	Ответ программы
1	Выбор действия	1	Загрузка таблицы из файла
2	Ввод из файла	tests/pos1.txt	Таблица загружена
3	Ввод некорректного действия	9	Ошибка
4	Сортировка пустой таблицы при выборе действия	6 или 7	Ошибка
5	Фильтрация пустой таблицы при выборе действия	8	Ошибка
6	Если при фильтрации ничего не нашлось	-1	Сообщение о том, что ничего не нашлось
7	Добавление 10006 записи	Информация о студенте	Ошибка переполнения
8	Удаление записи, которой нет в таблице	100 (таблица размера 40)	Ошибка удаления
9	Вывод таблицы размера 0 при выборе действия	2 или 3	Ошибка
10	Выбор алгоритма сортировки	Число не равное 1 или 2	Ошибка
11	Вывод существующей таблицы	2 или 3	Таблица выводится в файл

12	Вывод в файл	Имя файла	Таблица выводится в файл
13	Ввод имени файла	Имя файла длиной больше 40	Ошибка переполнения
14	Ввод имени студента	Длина строки больше 15	Ошибка переполнения
15	Ввод фамилии студента	Длина строки больше 15	Ошибка переполнения
16	Ввод группы студента	Длина строки больше 10	Ошибка переполнения
17	Ввод пола студента	Длина строки больше 10	Ошибка переполнения
18	Считывание строки с данными о студенте из файла	Длина строки больше 100	Ошибка переполнения
19	Ввод возраста студента	Вводится не положительное число	Ошибка ввода
20	Ввод года поступления студента	Вводится не положительное число	Ошибка ввода
21	Ввод месяца поступления студента	Вводится не положительное число	Ошибка ввода
22	Ввод дня поступления студента	Вводится не положительное число	Ошибка ввода
23	Ввод средней оценки студента	Вводится не вещественное число из диапазона [0; 5]	Ошибка ввода
24	Ввод типа места проживания студента	Вводится не число 1 или 2	Ошибка ввода

		или 3	
25	Ввод номера дома	Вводится не положительное число	Ошибка ввода
26	Ввод номера квартиры	Вводится не положительное число	Ошибка ввода
27	Ввод улицы проживания студента	Длина строки больше 30	Ошибка переполнения
28	Ввод платы за аренду	Вводится не положительное число	Ошибка ввода

ЗАМЕРЫ ВРЕМЕНИ РАБОТЫ

В ходе эксперимента для получения результатов проводилось 20 замеров по времени. Усредненный результат представлен в таблице

Размер таблицы	Пузырек		Быстрая	
	Обычная таблица	Таблица ключей	Обычная таблица	Таблица ключей
40	0.062	0.049	0.028	0.028
100	0.362	0.434	0.076	0.058
500	6.292	6.051	0.357	0.335
1000	32.879	12.347	0.708	0.669
5000	291.494	248.843	3.996	3.592
10000	1236.592	953.213	6.441	7.942

*Замеры времени в таблице представлены в миллисекундах

ОБЪЕМ ИСПОЛЬЗУЕМОЙ ПАМЯТИ

Размер обычной таблицы в байтах – 160, размер таблицы ключей в байтах - 24

Размер таблицы	Обычная таблица	Обычная таблица + таблица ключей
40	6400	7360
100	16000	18400
500	80000	92000
1000	160000	184000
5000	800000	920000
10000	1600000	1840000

*Значения в таблице представлены в байтах

ОЦЕНКА ЭФФЕКТИВНОСТИ

Размер таблицы	Пузырек	Быстрая
40	1.27	1
100	0.83	1.31
500	1.04	1.07
1000	2.66	1.06
5000	1.17	1.11
10000	1.3	0.81

*Числа во втором и третьем столбце определяются отношением времени работы сортировки обычной таблицы к времени работы таблицы ключей

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как выделяется память под вариантную часть записи?
Память выделяется в соответствии с самой большой по размеру структурой.

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

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

4. Что представляет собой таблица ключей, зачем она нужна?
Таблица ключей — таблица вида *индекс-поле*, где *поле* — поле таблицы, а *индекс* — индекс этого поля в таблице. Таблица ключей ускоряет работу сортировки, так как в исходной таблице не приходится менять местами строки, достаточно организовать сортировку таблицы ключей и выводить строки исходной таблицы в соответствии с индексом таблицы ключей.

5. В каких случаях эффективнее обрабатывать данные в самой таблице, а когда – использовать таблицу ключей?
На больших данных однозначно эффективнее таблица ключей. На небольших данных разница несущественна.

6. Какие способы сортировки предпочтительнее для обработки таблиц и почему?
Способы сортировки, в которых используется компаратор, потому что он позволяет сравнивать разные строки таблицы.

ВЫВОД

В рамках этой лабораторной работы я изучил такой тип данных, как запись. Запись реализуется с помощью структур и объединений. Я научился сортировать массив записей по выбранному полю двумя способами: сортировать сразу массив записей, сортировать вспомогательный массив ключей. Так же я сравнил скорости работы этих двух способов для двух сортировок (пузырьком и быстрой сортировкой). Очевидно, что сортировать массив с помощью таблицы ключей более затратно по памяти (согласно таблице «Объем используемой памяти» затраты по памяти на 15% больше), но этот способ дает преимущество по времени (согласно таблице «Замеры времени работы» в среднем на 10% быстрее). Связано это с тем, что при сортировке с помощью ключей мы экономим время при перестановке

элементов, потому что размер элемента в массиве ключей меньше размера элемента в исходной таблице. Соответственно и перестановка происходит быстрее. А также по таблице «Замеры времени работы» можно заметить преимущество во времени при использовании быстрой сортировки.