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 Равашдех Фадей Хешамович

Группа ИУ7 – 35Б

Преподаватель Никульшина Т. А.

2023 г.

Оглавление

Описание условия задачи.....3
Описание технического задания.....4
Описание структур данных.....6
Описание функций.....8
Описанние алгоритма.....17
Набор тестов.....18
Оценка эффективности.....19
Ответы на контрольные вопросы.....21
Вывод.....22

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.

Описание студента:

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:

Файл с данными: каждая строка содержит значения полей, разделённые пробелом. Если информация в поле отсутствует, оно обозначается прочерком (символом «–»).

Целое число: задаёт пункт меню.

Строковые и числовые данные: для поиска, добавления, удаления записей.

Выходные данные:

Таблица, отсортированная таблица ключей, сравнение эффективности сортировок, результат поиска по таблице, вывод таблицы по ключам.

Функции меню программы:

1. Информация о программе
2. Вывести общий список студентов
3. Вывести список студентов, указанного года поступления, живущих в съемном жилье
4. Добавить запись о студенте в конец таблицы
5. Удалить запись о студенте из таблицы по фамилии
6. Удалить запись о студентах из таблицы по фамилии
7. Вывести отсортированную по фамилии студента таблицу ключей
8. Вывести студентов упорядоченных по фамилии
9. Вывести студентов упорядоченных по фамилии, используя таблицу ключей

10. Вывести результаты сравнения эффективности работы программы при обработке данных в исходной таблице и таблице ключей
11. Выйти
12. Сохранить и выйти

Обращение к программе:

Запускается через терминал: ./app.exe *имя_файла_с_таблицей*

Аварийные ситуации:

Неверные аргументы командной строки.

Файл не найден или программа не имеет прав на чтение.

Действие с введенным пунктом отсутствует.

Введен символ окончания ввода EOF. Завершение программы.

Прочитана запись из файла, содержащая неверное количество полей.

Поле не прошло проверку формата.

Поле адреса не соответствует ни одному типу адреса.

Некорректный ввод. Количество введенных символов равно нулю или превысило ожидаемое.

Таблица заполнена, добавление записей невозможно.

Некорректная дата.

Невозможно сохранить данные в файл. Возможно нет прав на запись.

Описание структур данных

Постоянные, определяющие размеры стуктур:

```
#define LAST_NAME_LEN 20
#define FIRST_NAME_LEN 12
#define GROUP_LEN 9
#define GENDER_LEN 1
#define AGE_LEN 3
#define GPA_LEN 4
#define YEAR_LEN 4
#define DATE_LEN 10
#define ADDRESS_TYPE_LEN 14
#define STREET_NAME_LEN 20
#define CHAR3 3
#define CHAR4 4
#define CHAR6 6
#define MAX_TABLE_LEN 1000
```

Типы жилья:

```
typedef struct
{
    char street[STREET_NAME_LEN+1];
    char house_num[CHAR3+1];
    char flat_num[CHAR4+1];
} house_t;

typedef struct
{
    char dorm_num[CHAR3+1];
    char room_num[CHAR4+1];
} dormitory_t;

typedef struct
{
    char street[STREET_NAME_LEN+1];
    char house_num[CHAR3+1];
    char flat_num[CHAR4+1];
    char cost[CHAR6+1];
} rental_place_t;
```

Объединение, использующееся для вариативного поля адреса:

```
typedef union
{
    house_t house;
    dormitory_t dormitory;
    rental_place_t rent;
```

```
} address_t;
```

Структура студента:

```
typedef struct
{
    char last_name[LAST_NAME_LEN+1];        // Фамилия
    char first_name[FIRST_NAME_LEN+1];      // Имя
    char group[GROUP_LEN+1];                // Группа
    char gender;                            // Пол
    char age[AGE_LEN+1];                    // Возраст
    char gpa[GPA_LEN+1];                // Средняя оценка за сессию
    char enrollment_date[DATE_LEN+1];       // Дата поступления
    char address_type[ADDRESS_TYPE_LEN+1];  // Тип адреса
    address_t address;                  // Вариативное поле адреса
} student_t;
```

Таблица студентов:

```
typedef struct
{
    student_t table[MAX_TABLE_LEN];
    size_t n;
} student_table_t;
```

Ключ-фамилия и индекс студента:

```
typedef struct
{
    char last_name[LAST_NAME_LEN+1];
    size_t index;
} student_key_t;
```

Таблица ключей:

```
typedef struct
{
    student_key_t key_table[MAX_TABLE_LEN];
    size_t n;
} student_key_table_t;
```

Описание функций

Функции главного файла программы:

```
/**
* @brief Проверка аргументов
*
* @param argc
* @return int
*/
int check_args(int argc);

/**
* @brief Контроллер
*
* @param option выбранный пользователем пункт меню
* @param filename имя файла
* @param table таблица
* @param key_table таблица ключей
* @return int код возврата
*/
int controller(int option, char *filename, student_table_t *table,
student_key_table_t *key_table);
```

Функции модуля ввода/вывода:

```
/**
* @brief Очищение буфера ввода
*
*/
void clear_buff(void);

/**
* @brief Существует ли файл
*
* @param file_name имя файла
* @return int код возврата
*/
int is_file_exist(char *file_name);

/**
* @brief Прочитать запись
*
* @param f файловый дескриптор
* @param fields поля записи
* @return int код возврата
*/
int read_record(FILE *f, char fields[TABLE_FIELDS]
[MAX_FIELD_LEN+1]);
```

```
/**
* @brief Печать контура заголовка таблицы
*
* @param fields поля
*/
void print_table_header_lines(char fields[TABLE_FIELDS+1]
[MAX_FIELD_LEN]);

/**
* @brief Печать заголовка таблицы
*
* @param fields поля
*/
void print_table_header_formatter(char fields[TABLE_FIELDS+1]
[MAX_FIELD_LEN]);

/**
* @brief Печать заголовка таблицы
*
*/
void print_table_header(void);

/**
* @brief Вывод студента в табличном формате
*
* @param stud студент
*/
void print_record(student_t stud);


/**
* @brief Печать контура заголовка таблицы ключей
*
* @param fields поля
*/
void print_key_table_header_lines(char fields[KEY_TABLE_FIELDS+1]
[MAX_FIELD_LEN]);


/**
* @brief Печать заголовка таблицы ключей
*
* @param fields поля
*/
void print_key_table_header_formatter(char
fields[KEY_TABLE_FIELDS+1][MAX_FIELD_LEN]);

/**
* @brief Печать заголовка таблицы ключей
*
```

```
*/
void print_key_table_header(void);

/**
* @brief Вывод ключа в табличном формате
*
* @param stud_key запись таблицы ключей
*/
void print_key_record(student_key_t stud_key);

/**
* @brief Вывод меню программы
*
*/
void print_menu(void);

/**
* @brief Вывод информации о программе
*
*/
void print_info(void);

/**
* @brief Чтение строки определенной длины из стандартного потока
ввода
*
* @param pch указатель на принимаемую строку
* @param n длина строки
* @return int код возврата
*/
int get_input(char pch[], size_t n);

/**
* @brief Получение выбранного пользователем пункта
*
* @param option выбранный пункт
* @return int код возврата
*/
int get_option(size_t *option);

/**
* @brief Вывод сообщения, поясняющего ошибку
*
* @param rc код возврата
*/
void print_error_msg(int rc);

/**
* @brief Вывод сообщения, поясняющего ошибку, и возврат того же
кода возврата
*
```

```
* @param rc код возврата
* @return int код возврата
*/
int print_error(int rc);
```

Функции модуля сортировок:

```
/**
* @brief Двоичный поиск для вставки элемента
*
* @param base Указатель на начало сортируемой области
* @param x вставляемый элемент
* @param nmemb количество элементов
* @param size размер элемента
* @param compar указатель на функцию сравнения элементов
* @return void* указатель на место вставки элемента
*/
void *binary_search(void *base, void *x, size_t nmemb, size_t
size, int (*compar)(const void*, const void*));

/**
* @brief Поиск от конца к началу для вставки элемента
*
* @param base Указатель на начало сортируемой области
* @param x вставляемый элемент
* @param nmemb количество элементов
* @param size размер элемента
* @param compar указатель на функцию сравнения элементов
* @return void* указатель на место вставки элемента
*/
void *end_to_beg_search(void *base, void *x, size_t nmemb, size_t
size, int (*compar)(const void*, const void*));

/**
* @brief Циклический сдвиг вправо
*
* @param pb Указатель на начало области
* @param pe Указатель на конец области
* @param size размер элемента (или же размер сдвига в char'ax)
*/
void right_cycle_shift(void *pb, void *pe, size_t size);

/**
* @brief Функция сортировки бинарными вставками
*
* @param base Указатель на начало сортируемой области
* @param nmemb количество элементов
* @param size размер элемента
* @param compar указатель на функцию сравнения элементов
*/
```

```
void binary_insertion_sort(void *base, size_t nmemb, size_t size,
int (*compar)(const void*, const void*));

/**
* @brief Функция сортировки бинарными вставками
*
* @param base Указатель на начало сортируемой области
* @param nmemb количество элементов
* @param size размер элемента
* @param compar указатель на функцию сравнения элементов
*/
void insertion_sort(void *base, size_t nmemb, size_t size, int
(*compar)(const void*, const void*));
```

Функции модуля для работы с таблицами:

```
/**
* @brief Проверка даты
*
* @param date строка даты в формате DD.MM.YYYY
* @return int результат (0 - если корректная дата)
*/
int is_date_valid(char date[]);

/**
* @brief Проверка полей
*
* @param fields поля записи
* @return int код возврата
*/
int check_fields(char fields[TABLE_FIELDS][MAX_FIELD_LEN+1]);

/**
* @brief Перевод массива строк в стуктуру студента
*
* @param fields массив строк
* @return student_t структура студента
*/
student_t fields_to_student(char fields[TABLE_FIELDS]
[MAX_FIELD_LEN+1]);

/**
* @brief Запись строк записи в структуру студента с проверкой
корректности строк
*
* @param fields массив строк
* @param student указатель на структуру студента
* @return int код возврата
*/
```

```
int process_student(char fields[TABLE_FIELDS][MAX_FIELD_LEN+1],
student_t *student);

/**
* @brief Чтение таблицы из файла
*
* @param file_name имя файла
* @param students таблица студентов
* @return int код возврата
*/
int read_table(char *file_name, student_table_t *students);

/**
* @brief Вывод таблицы студентов
*
* @param students таблица студентов
*/
void print_table(student_table_t students);


/**
* @brief Запрашивает field_name, длиной n. Записывает в field и
проверяет mask
*
* @param field строка для записи
* @param n размер поля
* @param field_name название поля
* @param mask маска регулярного выражения
* @return int код возврата
*/
int get_field(char field[], size_t n, char field_name[], char
mask[]);

/**
* @brief Поиск студентов определенного года поступления и живущих
в съемном жилье
*
* @param students таблица студентов
* @return int код возврата
*/
int print_filtered_table(student_table_t students);

/**
* @brief Добавить запись в таблицу
*
* @param table таблица
* @return int код возврата
*/
int add_record(student_table_t *table);

/**
```

```
* @brief Удалить студента с введенной фамилией из таблицы
*
* @param table таблица
* @return int код возврата
*/
int del_record(student_table_t *table);

/**
* @brief Удалить всех студентов с введенной фамилией из таблицы
*
* @param table таблица
* @return int код возврата
*/
int del_records(student_table_t *table);

/**
* @brief Функция сравнения двух студентов по фамилии
*
* @param p указатель на первого студента
* @param q указатель на второго студента
* @return int результат
*/
int compare_student_last_names(const void *p, const void *q);

/**
* @brief Функция сравнения двух студентов по фамилии
*
* @param p указатель на первого студента
* @param q указатель на второго студента
* @return int результат
*/
int compare_student_key_last_names(const void *p, const void *q);

/**
* @brief Заполнение таблицы ключей
*
* @param table таблица
* @param key_table таблица ключей
*/
void fill_student_key_table(student_table_t table,
student_key_table_t *key_table);

/**
* @brief Вывод табилцы ключей
*
* @param key_table таблица ключей
*/
void print_key_table(student_key_table_t key_table);

/**
* @brief Вывод
```

```
*
* @param key_table таблица ключей
*/
void sort_key_table_and_print(student_key_table_t *key_table);

/**
* @brief Вывод отсортированной таблицы
*
* @param table таблица
*/
void print_sorted_table(student_table_t *table);

/**
* @brief Вывод отсортированной таблицы используя таблицу ключей
*
* @param table таблица
* @param key_table таблица ключей
*/
void print_table_using_sorted_key_table(student_table_t table,
student_key_table_t *key_table);

// Функция для сортировки таблицы (передача структуры по значению,
чтобы не менять исходную таблицу)
void table_insertion_sort(student_table_t table);

// Функция для сортировки таблицы ключей (передача структуры по
значению, чтобы не менять исходную таблицу)
void table_key_insertion_sort(student_key_table_t table);

// Функция для сортировки таблицы (передача структуры по значению,
чтобы не менять исходную таблицу)
void table_binary_insertion_sort(student_table_t table);

// Функция для сортировки таблицы ключей(передача структуры по
значению, чтобы не менять исходную таблицу)
void table_key_binary_insertion_sort(student_key_table_t table);

/**
* @brief Сравнение функций сортировок и отсутствие/наличие таблицы
ключей
*
* @param filename имя файла
* @return int код возврата
*/
int compare(char *filename);

/**
* @brief Перевод структуры студента в массив строк
*
* @param stud структура студента
* @param fields массив строк
```

```
*/
void student_to_str(student_t stud, char fields[TABLE_FIELDS]
[MAX_FIELD_LEN+1]);

/**
* @brief Запись таблицы в файл
*
* @param filename имя файла
* @param table таблица
* @return int код возврата
*/
int write_table(char *filename, student_table_t table);
```

Описанние алгоритма

1. Программа запускается с одним ключом – именем файла-таблицы.

2. Происходит считывание данных из файла в таблицу. Если считывание успешно, то выводится сообщение об успешности считывания. Затем считываются данные из таблицы в таблицу ключей.

3. Пользователю выводится меню программы и предлагается выбрать и ввести пункт меню с необходимым действием.

4. Если действие предполагает ввод дополнительных данных (например, функции добавления / удаления / поиска записи в таблице), то пользователю предлагается ввести эти данные.

5. Если действие предполагает вывод, то выводятся данные (таблица / таблица ключей / таблица сравнения сортировок).

6. В случае возникновнеие аварийной ситуации, пользователь получит сообщение, раскрывающее суть ошибки.

НАБОР ТЕСТОВ

№	Название теста	Ввод	Вывод
1	Некорректный запуск программы	./app.exe	«Ошибка: Неверные аргументы командной строки.» «Команда вызывается из командной строки так: ./app.exe table_file.txt»
2	Некорректный ввод пункта меню	112	«Ошибка: Действие с введенным пунктом отсутствует.»
3	Некорректный ввод пункта меню	afaeFefWfcWV	«Ошибка: Действие с введенным пунктом отсутствует.»
4	Просмотр таблицы	2	Таблица
5	Вывести список студентов, указанного года поступления, живущих в съемном жилье	3 2020	Вывод список студентов, указанного года поступления, живущих в съемном жилье в виде таблицы
6	Добавление записи	4 Ab Ba IU7-35B m 19 5.0 29.02.2000 dormitory 1 1	«Запись успешно добавлена.»
7	Некорректное удаление записи	Ab Ba IU7-35B m 19 5.0 29.02.1900 dormitory 1 1	«Ошибка: Некорректная дата.»
8	Запрос на удаление несуществующего студента.	5 A	«В таблице нет подходящих записей.»
9	Запрос на удаление несуществующих студентов.	6 A	«В таблице нет подходящих записей.»

Оценка эффективности

Замеры времени (в секундах) выполнения сортировок 1000 раз:

Кол-во записей	Сортировка вставками		Сортировка вставками с бинарным поиском	
	Таблица	Таблица ключей	Таблица	Таблица ключей
40	0.002673	0.002305	0.001826	0.001417
100	0.015218	0.013016	0.007588	0.006048
200	0.054028	0.046776	0.022995	0.018498
500	0.346513	0.312171	0.156377	0.118733
1000	1.303090	1.129123	0.524000	0.380556

Оценка памяти:

Структура таблицы, содержащая статический массив из 1000 записей студентов, занимает 117008 байт. Структура таблицы ключей, содержащая статический массив из 1000 пар ключ-индекс, занимает 32008 байт. Суммарно структуры занимают 149016 байт.

Получается, что использование таблицы ключей занимает на 27,35% больше памяти, по сравнению с использованием обычной таблицы.

Оценка эффективности:

Кол-во записей	Прирост эффективности при соритировке вставками таблицы ключей относительно соритровки обычной таблицы, %	Прирост эффективности при соритировке вставками с бинарным поиском таблицы ключей относительно соритровки обычной таблицы, %
40	15.965	28.864
100	16.918	25.463
200	15.504	24.311
500	11.001	31.705
1000	15.407	37.693

Как видно из таблицы эффективности, при использовании таблицы ключей и сортировки вставками с бинарным поиском, прирост в скорости соритровки более 24%, что при дополнительных затратах памяти в 27% вполне целесообразно использовать. Также можно заметить, что по мере увеличения записей увеличивается и прирост в скорости вплоть до 37% при 1000 записях.

Ответы на контрольные вопросы

1. Как выделяется память под вариантную часть записи?

Выделяется такое количество байт, чтобы хватило на самую большую структуру. В данном случае, самая большая структура rental_place_t занимает 37 байт, поэтому весь union address_t занимает 37 байт.

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

В таком случае в память вариативной части будут записаны введенные данные.

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

Программист должен написать программу так, что при неверном вводе, программа завершит только текущую операцию и выведет на экран сообщение об ошибке, при этом программа не завершится. Пользователю тоже следует внимательно следить за правильностью ввода, потому что в противном случае данные придется вводить заново.

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

Таблица ключей представляет собой список пар ключ-индекс и длина этого списка. Пара ключ-индекс представляет собой ключ, в данном случае фамилию студента, и индекс, под которым в основной таблице находится этот студент. Таблица ключей нужна ускорения сортировки таблицы, также можно выводить таблицу в упорядоченном виде, не упорядочивая ее.

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

Использовать таблицу ключей эффективно, когда используется какое-то одно поле таблицы, в данном случае фамилия студента. Использование таблицы ключей показало прирост в скорости сортировки.

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

У каждой сортировки есть свои характеристики скорости и затрат памяти. Сортировки, требующие много памяти, будут занимать еще больше места, так как будут работать с записями таблиц, большинство полей которых не понадобятся в сортировке, поэтому лучше выбирать сортировки с затратами памяти O(1). Также будет лучше выбрать соритровки с меньшим количеством перестановок элементов, т. к. в данном случае элементами выступают записи таблиц, т. е. большие блоки данных, и их перестановка будет более трудозатратной.

Вывод

В результате выполнения этой лабораторной работы, я улучшил свои навыки работы с типом данных стркутура, в том числе со структурой с вариативным полем, с типом данных объединение.

В результате оценки эффективности, я пришёл к выводу, что при отсутствии ограничений по памяти, стоит использовать таблицу ключей, так как это дает прирост в эффективности. Также использование ускоренной сортировки показало еще большую эффективность при работе с таблицами ключей, прирост скорости в процентах превысил процент дополнительной памяти, выделяемой на таблицу ключей.