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

“Обработка таблиц”

Студент **Зернов Георгий Павлович**

Группа **ИУ7-34Б**

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

Студент _____ **Зернов Г.П.**

Преподаватель _____ **Никульшина Т.А.**

2024

Оглавление

Условие задачи

Создать таблицу, содержащую не менее 40-ка записей. Упорядочить данные в ней по возрастанию ключей, двумя алгоритмами сортировки, где ключ – год поступления, используя: а) саму таблицу, б) массив ключей. Реализовать возможность добавления и удаления записей в ручном режиме, просмотр таблицы, просмотр таблицы в порядке расположения таблицы ключей обязательна. Ввести список студентов, содержащий следующие сведения о них: Фамилия, имя, группа, пол (м/ж), возраст, средний балл за сессию, дата поступления, тип жилья: 1. Дом: (улица, № дома, № квартиры) 2. Общежитие: (№ общежития, № комнаты) 3. Съемное жилье: (улица, № дома, № квартиры, стоимость). Вывести список студентов указанного года поступления, живущих в съемном жилье, стоимостью меньше указанного.

Техническое задание

Реализуемая в программе задача

Цель работы: реализация инструмента для управления информацией о студентах. Это включает в себя загрузку данных из файла, вывод таблицы, добавление записи в конец, удаление по префиксу фамилии, сортировку данных, фильтрацию данных.

Входные данные

В случае загрузки данных из файла – имя файла (строка длиной до 40 символов). В остальных случаях – набор записей о студентах (массив длиной не более 1000 элементов).
Каждая запись содержит поля:

- Фамилия – строка длиной до 15 символов.
- Имя – строка длиной до 15 символов.
- Группа – строка длиной до 10 символов.
- Пол – М\Ж.
- Возраст – целое число от 0 до 100.
- Средняя оценка за сессию – вещественное число от 0 до 100.
- Год поступления – целое число от 1900 до 2024.
- Тип жилья – дом, общежитие, или съёмное жильё.
- Информация о жилье – вариантное поле:
    1. Дом:
        - Адрес – строка до 20 символов.
        - Номер дома – целое число от 0 до 9999.
        - Номер квартиры – целое число от 0 до 9999.
    2. Общежитие:

- Номер общежития – целое число от 0 до 9999.
- Номер комнаты – целое число от 0 до 9999.

3. Съёмное жильё:
    - Адрес – строка до 20 символов.
    - Номер дома – целое число от 0 до 9999.
    - Номер квартиры – целое число от 0 до 9999.
    - Цена аренды – вещественное число большее 0.

Выходные данные

1. Загрузить данные из файла – массив записей о студентах.
2. Вывести таблицу – выведенная на экран таблица данных о студентах.
3. Вывести отсортированную по году поступления таблицу ключей - выведенная на экран таблица ключей (пар: индекс в таблице – год поступления), отсортированная по году поступления.
4. Добавить запись в конец – массив записей о студентах, с добавленной в конец записью.
5. Удалить записи по префиксу фамилии – массив записей о студентах, с удалёнными по префиксу фамилии записями.
6. Сортировка таблицы пузырьком по году поступления – выведенная на экран таблица записей о студентах, отсортированная по году поступления.
7. Быстрая сортировка таблицы по году поступления – выведенная на экран таблица записей о студентах, отсортированная по году поступления.
8. Сортировка с таблицей ключей пузырьком по году поступления – выведенная на экран таблица записей о студентах с таблицей ключей (пар: индекс в таблице – год поступления), отсортированная по году поступления.

9. Быстрая сортировка с таблицей ключей по году поступления – веденная на экран таблица записей о студентах с таблицей ключей (пар: индекс в таблице – год поступления), отсортированная по году поступления.
10. Анализ времени – выведенная на экран таблица, содержащая информацию о затраченных на каждую из сортировок времени и памяти для различных размеров сортируемого массива.
11. Вывести список студентов указанного года поступления, живущих в съёмном жилье, стоимостью меньше указанного – выведенный на экран список вида: “Фамилия Имя Группа”, студентов с указанными параметрами.

Способ обращения к программе

Чтобы обратиться к программе, пользователю необходимо запустить файл app.exe.

Возможные аварийные ситуации или ошибки пользователя

- Невозможность интерпретировать ввод пользователя как валидное значение поля, ввод которого был запрошен – вывод на экран информирующего сообщения о неверном вводе.
- Попытка записать данные в таблицу при максимально возможном количестве записей в таблице – вывод на экран сообщения о невозможности записи данных.

Внутренние структуры и типы данных

Перечисляемый тип, описывающий пол студента.

Листинг структуры:

```
typedef enum
{
    MALE,
    FEMALE
} sex_t;
```

Перечисляемый тип, описывающий тип жилья.

Листинг структуры:

```
typedef enum
{
    HOUSE,
    DORMITORY,
    RENTED
} accomodation_type;
```

Структура, представляющее жильё типа “дом”.

- char street[MAX_STREET_LEN + 1] – адрес (строка длиной не более 40 символов).
- int house – номер дома (целое число от 0 до 9999).
- int flat – номер квартиры (целое число от 0 до 9999).

Листинг структуры:

```
typedef struct
{
    char street[MAX_STREET_LEN + 1];
    int house;
    int flat;
} house_t;
```

Структура, представляющее жильё типа “общежитие”.

- int dormitory – номер общежития (целое число от 0 до 9999).
- int room – номер комнаты (целое число от 0 до 9999).

Листинг структуры:

```
typedef struct
{
    int dormitory;
    int room;
} dormitory_t;
```

Структура, представляющее жильё типа “арендованное жильё”.

- char street[MAX_STREET_LEN + 1] – адрес (строка длиной не более 40 символов).
- int house – номер дома (целое число от 0 до 9999).
- int flat – номер квартиры (целое число от 0 до 9999).
- double price – цена аренды (вещественное число больше 0).

Листинг структуры:

```
typedef struct
{
    char street[MAX_STREET_LEN + 1];
    int house;
    int flat;
    double price;
} rented_accomodation_t;
```

Структура, представляющее вариативное поле описывающее жильё.

Листинг структуры:

```
typedef union
{
    house_t house;
    dormitory_t dormitory;
    rented_accomodation_t rented;
} accomodation_t;
```

Структура, представляющее студента.

- char surname[MAX_SURNAME_LEN + 1] – фамилия (строка длиной не более 15 символов).
- char name[MAX_NAME_LEN + 1] – имя (строка длиной не более 15 символов).
- char group[MAX_GROUP_LEN + 1] – группа (строка длиной не более 10 символов).
- sex_t sex – пол.
- int age – возраст (целое число от 0 до 100).
- double avg_mark – средняя оценка за сессию (вещественное число от 0 до 100).
- int admission_year – год поступления (целое число от 1900 до 2024).
- accomodation_type type – тип жилья.
- accomodation_t accomodation – вариативное поле, описывающее тип жилья.

Листинг структуры:

```
typedef struct
{
    char surname[MAX_SURNAME_LEN + 1];
    char name[MAX_NAME_LEN + 1];
    char group[MAX_GROUP_LEN + 1];
    sex_t sex;
    int age;
    double avg_mark;
    int admission_year;
    accomodation_type type;
    accomodation_t accomodation;
} student_t;
```

Алгоритм программы

- Общий алгоритм программы:

1. Запуск основного цикла программы
2. Вывод меню
3. Запрос у пользователя опции для выполнения
4. Выполнение выбранной опции
5. Переход к новой итерации цикла или завершение основного цикла при соответствующем выборе пользователя.

- Алгоритм считывания из файла:

1. Запрос у пользователя имени файла
2. Проверка валидности имени файла и его доступности для чтения
3. Проверка валидности содержимого
4. Если таблица студентов содержит данные, запрос у пользователя разрешения на перезапись.
5. Считывание студентов из файла в цикле

- Алгоритм добавления студента в конец таблицы:

1. Проверка достижения максимального числа студентов в таблице
2. Запрос у пользователя студента для добавления
3. Запись студента в конец таблицы

- Алгоритм удаления студентов по префиксу фамилии:

1. Запрос у пользователя префикса
2. Итерация по таблице студентов, если фамилия начинается с указанного префикса, увеличение значения сдвига на 1, иначе запись студента на n позиций назад, где n – текущее значение переменной сдвига.

- Алгоритм сортировки таблицы по году поступления с помощью таблицы ключей:

1. Создание таблицы ключей
2. Сортировка таблицы ключей
3. Вывод таблицы по индексам из таблицы ключей

- Алгоритм вывода списка студентов указанного года поступления, живущих в съёмном жилье, стоимостью меньше указанного:

1. Запрос у пользователя ввода поступления и стоимости
2. Итерация по таблице студентов, в случае если студент поступил в указанный год и живёт в съемном жилье стоимостью меньше указанного – вывод имени, фамилии и группы студента на экран.

Функции программы

- int fscan_string(FILE *src_file, char *dst, size_t max_len) – считывает строку из файла src_file в массив символов dst, длиной не более max_len.
- int scan_accomodation(accomodation_t *accomodation, accomodation_type type) – считывает объединения, описывающего жильё, по типу жилья type с приглашением к вводу.
- int scan_student(student_t *student) – считывает студента с приглашением к вводу.
- int fscan_accomodation(FILE *file, accomodation_t *accomodation, accomodation_type type) – считывает объединения, описывающего жильё, по типу жилья type из файла file.
- int fscan_student(FILE *file, student_t *student) – считывает студента с приглашением к вводу.
- int load_data(student_t students[MAX_STUDENTS_NUM], size_t *students_num) – заполняет таблицу студентов students и students_num – число записей в ней из файла по введённому пользователем названию файла.
- int print_students_table(student_t students[MAX_STUDENTS_NUM], size_t *students_num) – выводит таблицу студентов.

- int print_sorted_key_table(student_t students[MAX_STUDENTS_NUM], size_t *students_num) – выводит таблицу ключей по таблице студентов.
- int add_student(student_t students[MAX_STUDENTS_NUM], size_t *students_num) – добавляет вводимого пользователем студента в конец таблицы студентов.
- int del_students_by_surname(student_t students[MAX_STUDENTS_NUM], size_t *students_num) – удаляет студентов из таблицы по введённому пользователем префиксу фамилии.
- int bubble_sort_students(student_t students[MAX_STUDENTS_NUM], size_t *students_num) – сортирует таблицу студентов пузырьком.
- int qsort_students(student_t students[MAX_STUDENTS_NUM], size_t *students_num) – сортирует таблицу студентов быстрой сортировкой.
- int bubble_sort_students_by_key(student_t students[MAX_STUDENTS_NUM], size_t *students_num) – сортирует таблицу студентов по таблице ключей пузырьком.
- int qsort_students_by_key(student_t students[MAX_STUDENTS_NUM], size_t *students_num) – сортирует таблицу студентов по таблице ключей быстрой сортировкой.
- int my_filter(student_t students[MAX_STUDENTS_NUM], size_t *students_num);
- int time_measurement(void) – проводит анализ эффективности алгоритмов сортировки.
- void print_menu(void) – выводит меню.
- int main(void) – исполняет основной алгоритм программы.

Тесты

Тестовый случай	Ожидаемый результат
Выбрана не валидная опция в меню	Сообщение о выборе не валидной опции, программа снова предлагает ввести опцию
Введены не валидные данные на запрос программы	Соответствующее сообщение об ошибке

Файл невозможно открыть на чтение	Соответствующее сообщение об ошибке
В файл содержит не валидные данные	Соответствующее сообщение об ошибке
Загрузка данных из файла	В таблицу студентов успешно загружены данные, сообщение об успехе
Вывод таблицы	Распечатанная на экран таблица студентов
Вывод таблицы ключей	Распечатанная на экран таблица ключей
При добавлении студента в конец в таблице уже максимальное число студентов	Соответствующее сообщение об ошибке
Добавление студента в конец	В конец таблицы добавлен студент, выведено сообщение об успешном добавлении
Удалить студента по префиксу фамилии	Из таблицы удалены подходящие студенты, выведено сообщение о количестве удалённых студентов
Сортировка таблицы пузырьком	Таблица отсортирована, выведены время и затраченная память
Быстрая сортировка таблицы	Таблица отсортирована, выведены время и затраченная память
Сортировка таблицы пузырьком, с помощью таблицы ключей	На экран выведена отсортированная таблица студентов с таблицей ключей, выведены время и затраченная память
Быстрая сортировка таблицы, с помощью таблицы ключей	На экран выведена отсортированная таблица студентов с таблицей ключей, выведены время и затраченная память
Анализ времени	Распечатанная на экран таблица, содержащая время сортировки при различных размерах массива и методах сортировки

Вывести список студентов указанного года поступления, живущих в съёмном жилье, стоимостью меньше указанного	Распечатанный на экран список подходящих студентов вида: фамилия, имя, группа
Выход из программы	Завершение работы программы

Оценка эффективности

Замеры выполняются для числа повторений n = 500:

Кол-во элементов	Сортировка пузырьком		Быстрая сортировка		Размер данных (байт)	Размер таблицы ключей (байт)	Затраченная память (байт)
	Основной таблицы (нс)	С таблицей ключей (нс)	Основной таблицы (нс)	С таблицей ключей (нс)			
10	17979	1003	2488	959	1120	80	1200
50	516147	7029	60668	6645	5600	400	6000
100	2015590	16475	225996	14328	11200	800	12000
500	51154834	104950	5334241	96442	56000	4000	60000
1000	180570146	172557	15959740	155509	112000	8000	120000
2000	837263593	365141	64108977	328338	224000	16000	240000

Эффективность в процентах:

Кол-во элементов	Медленная сортировка, время (%)	Быстрая сортировка, время (%)
10	622	4.588
50	750	5.778
100	791	6.954
500	858	8.822
1000	1031	10.963
2000	1206	11.209

Дополнительные затраты памяти при таблице ключей: 6.667%.

Вывод

При сортировки таблиц использование таблицы ключей является эффективным способом ускорения программы, при этом эффективность значительно больше увеличивается для медленных сортировок.

Контрольные вопросы:

Как выделяется память под вариантную часть записи?

Память под вариантную часть записи выделяется по размеру самого большого поля, входящего в объединение.

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

Если в вариативную часть ввести данные, несоответствующие описанным, то поля объединения будут содержать назначающую последовательность байт, что может привести к неверной работе программы.

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

За правильностью выполнения операций с вариативной частью записи должен следить программист.

Что представляет собой таблица ключей, зачем она нужна?

Таблица ключей – таблица пар типа: индекс – ключ. Таблица ключей нужна для более быстрой обработки данных в исходной таблице, например, при сортировке можно отсортировать таблицу ключей, а затем по ней вывести изначальную таблицу в отсортированном виде.

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

Зависит от конкретной задачи, если размер данных в строке таблице значительно больше длины данных в строке таблице ключей или часто выполняются операции, требующие сравнение по ключу и не требующие изменения исходной таблицы. В случаях, когда требуется изменение исходной таблицы, размер данных в строке сравним с размером пары индекс – ключ или есть ограничения на дополнительную память лучше сортировать исходную таблицу.

Какие способы сортировки предпочтительнее для обработки таблиц и почему?

В зависимости от размера строки таблицы и требований по памяти и времени. Так, быстрая сортировка работает значительно быстрее сортировки пузырьком, но требует больше памяти на стэке. Медленная сортировка не требует дополнительной памяти, но работает медленнее.