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 Информатика и системы управления (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 «ЗАПИСИ С ВАРИАНТАМИ, ОБРАБОТКА ТАБЛИЦ»

Студент, группа **Буланый К., ИУ7-36Б**

2020 г.

Описание условия задачи

Ввести список квартир, содержащий адрес, общую площадь, количество комнат, стоимость квадратного метра, первичное жилье или нет (первичное–с отделкой или без нее; вторичное – время постройки, количество предыдущих собственников, количество последних жильцов, были ли животные). Найти все вторичное 2-х комнатное жилье в указанном ценовом диапазоне без животных.

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 – площадь квартиры, используя:

- саму таблицу;
- массив ключей.

Возможность добавления и удаления записей в ручном режиме обязательна.

Техническое задание

Входные данные:

Файл, в нем: количество квартир в таблице, а затем сама таблица квартир. Аргументы вызова.

Выходные данные:

Файл с таблицей квартир.

Функция программы: Ввод в файл и чтение из файла таблицы квартир, добавление и удаление квартир из таблицы, сортировка таблицы по площади, поиск квартир по параметрам, указанным в задании.

Обращение к программе: запускается из терминала с определенными ключами.

Аварийные ситуации:

Некорректные аргументы запуска, сообщение «Unknown arguments.».
Некорректный ввод данных квартиры, сообщение «Failure during input.»
Невозможно открыть файл для чтения, сообщение «Input file cannot be opened.».
Невозможно открыть файл для записи, сообщение «Output file cannot be opened.».
Некорректный индекс для удаления из таблицы, сообщение «Failure during deletion.»
Неудача при выделении памяти под таблицу, сообщение в зависимости от функции, в которой это произошло.

Допущения

Длина адреса квартиры не более 31 символов, но рекомендуется не вводить более 16 символов, в противном случае таблица будет неровной.
Все числовые поля целые.

Структуры данных

Для хранения данных о квартире используется запись flat_t :

```
typedef struct flat
{
   char adr[MAXN];
   int area;
   int price_per_m2;
   long price;
   int room_cnt;
   bool is_new;
   type_t type;
} flat_t;
```

- adr – адрес квартиры, где MAXN – максимальное число символов (31).
- area – площадь квартиры.
- price_per_m2 – стоимость квадратного метра.
- price – стоимость квартиры.
- room_cnt – количество комнат.
- is_new – флаг, показывающий новая ли квартира и как обращаться к вариантной части.
- type – вариантная часть.

Вариантная часть представлена объединением type_t:

```
typedef union type
{
   bool is_trim;
   old_t old;
} type_t;
```

- is_trim – если квартира первичная, то программа обращается к этому полю, которое показывает, есть ли отделка.
- old – если квартира вторичная, то программа обращается к этому полю, в котором записаны данные, описанные ниже.

Поле old представлено записью old_t:

```
typedef struct old
{
   int year;
   int owner_cnt;
   int lst_rsdnt_cnt;
   bool were_anmls;
} old_t;
```

- year – год, в котором квартира была построена.
- owner_cnt – количество предыдущих владельцев квартиры.
- lst_rsdnt_cnt – количество последних жителей квартиры.
- were_anmls – поле, показывающее, были ли у этих жителей животные.

Вспомогательная запись key с индексами квартир и их площадями:

```
typedef struct key
{
   int index;
   int area;
} key_t;
```

- index – индекс квартиры в таблице.
- area – площадь этой квартиры.

Алгоритм

1. При создании нового файла с таблицей.

Программа запускается с соответствующими ключами.
Открывается файл, проверяется успешность.
Вводится количество квартир, проверяется корректность ввода.
Вводятся данные каждой квартиры, проверяется корректность ввода.
Таблица записывается в файл.

2. При чтении таблицы из файла.

Программа запускается с соответствующими ключами.
Открывается файл, проверяется успешность.
Проверяется корректность файла, считывается таблица.
Таблица выводится на экран.

3. При удалении квартиры по индексу.

Программа запускается с соответствующими ключами.
Открываются файлы, проверяется успешность.
Проверяется корректность файла ввода, считывается таблица.
Проверяется корректность индекса.
Удаляется соответствующая квартира из таблицы.

Таблица записывается в файл вывода.

4. При добавлении квартиры в таблицу.

Программа запускается с соответствующими ключами.
Открываются файлы, проверяется успешность.
Проверяется корректность файла ввода, считывается таблица.
Считываются данные новой квартиры, проверяется корректность.
Добавляется квартира в таблицу.
Таблица записывается в файл вывода.

5. При сортировке.

Программа запускается с соответствующими ключами.
Открываются файлы, проверяется успешность.
Проверяется корректность файла ввода, считывается таблица.
Таблица сортируется.
Результат записывается в файл вывода.

6. При поиске по параметрам.

Программа запускается с соответствующими ключами.
Открываются файлы, проверяется успешность.
Проверяется корректность файла ввода, считывается таблица.
Считается количество подходящих квартир.
Записываются индексы подходящих квартир в массив.
Подходящие квартиры записываются в файл вывода.

Тесты

Тест	Ввод	Результат
Некорректные аргументы	./app.exe a	Unknown arguments.
Некорректный ввод файла для чтения	./app.exe -o dat.ttt	File cannot be opened.
Некорректный ввод данных квартиры (напри мер, площади)	asdwe	Failure during input.
Неудача при открытии файла	-	File cannot be opened.
Некорректный файл при выводе таблицы	-	Failure during reading.
Неудача при выполнении ввода, вывода, сортировки, поиска, связанная с выделением памяти.	-	В зависимости от задачи
Пустой файл	-	Failure during reading.
Создание нового файла	./app.exe -i out.txt	Файл с таблицей
Вывод таблицы из файла	./app.exe -o in.txt	Таблица

Сортировка	./app.exe -mrg/-ins [-k] in.txt out.txt	Array was sorted.
Удачный поиск	./app.exe -f in.txt out.txt	Flats were found.
Неудачный поиск	./app.exe -f in.txt out.txt	Flats were not found.

Оценка эффективности

Измерение эффективности сортировок производились в микросекундах. Для этого использовалась библиотека sys/time.h

*Сортировка вставками не меняет местами элементы упорядоченного массива, следовательно, упорядоченные массивы такая сортировка сортирует крайне быстро, поэтому в некоторых случаях мне не удалось достоверно измерить время.

Записи, упорядоченные по возрастанию*

Кол-во записей	Сортировка вставками		Сортировка слиянием	
	Исходная таблица	Таблица ключей	Исходная таблица	Таблица ключей
10000	?	?	4573	1125
20000	?	?	7521	2735
30000	?	?	12040	5264

Записи, упорядоченные по убыванию

Кол-во записей	Сортировка вставками		Сортировка слиянием	
	Исходная таблица	Таблица ключей	Исходная таблица	Таблица ключей
10000	808028	242710	5119	998
20000	3355599	937084	7979	2611
30000	7361173	2213953	12885	4989

Записи в случайном порядке

Кол-во записей	Сортировка вставками		Сортировка слиянием	
	Исходная таблица	Таблица ключей	Исходная таблица	Таблица ключей
10000	413442	132563	4026	1387
20000	1767618	456355	7201	3382
30000	3828660	1026055	12755	5843

Объем занимаемой памяти в байтах:

Количество записей	Исходная таблица	Таблица ключей
10000	720000	80000
20000	1440000	160000

30000	2160000	240000

Записи, упорядоченные по возрастанию*

Количество записей	Занимаемый % массива ключей всей таблицы	Рост скорости сортировки массива ключей по сравнению с таблицей (сортировка вставками)	Рост скорости сортировки массива ключей по сравнению с таблицей (сортировка слиянием)
10000	~11%	?	~4 раза
20000	~11%	?	~2.7 раз
30000	~11%	?	~2.2 раз

Записи, упорядоченные по убыванию

Количество записей	Занимаемый % массива ключей всей таблицы	Рост скорости сортировки массива ключей по сравнению с таблицей (сортировка вставками)	Рост скорости сортировки массива ключей по сравнению с таблицей (сортировка слиянием)
10000	~11%	~3.3 раз	~5 раз
20000	~11%	~3.6 раз	~3 раза
30000	~11%	~3.3 раз	~2.5 раза

Записи в случайном порядке

Количество записей	Занимаемый % массива ключей всей таблицы	Рост скорости сортировки массива ключей по сравнению с таблицей (сортировка вставками)	Рост скорости сортировки массива ключей по сравнению с таблицей (сортировка слиянием)
10000	~11%	~3.1 раз	~2.9 раз
20000	~11%	~3.8 раз	~2 раза
30000	~11%	~3.7 раз	~2.2 раза

Особенно учитывая, что сортировка по возрастанию массива по убыванию для сортировки вставками – худший случай, сортировка ключей показывает значительный рост скорости.

Контрольные вопросы

1. **Как выделяется память под вариантную часть записи?**

Размер памяти, выделяемый под вариантную часть, равен максимальному по длине полю вариантной части. Эта память является общей для всех полей вариантной части записи.

2. **Что будет, если в вариантную часть ввести данные, не соответствующие описанным?**

Тип данных в вариантной части при компиляции не проверяется. Из-за того, что невозможно корректно прочитать данные, поведение будет неопределенным.

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

Программист.

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

Дополнительный массив, содержащий индекс элемента в исходной таблице и выбранный ключ. Она нужна для оптимизации сортировки.

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

На большом объеме данных выгоднее использовать массивы ключей. На небольших – напротив.

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

Если обработка данных производится в таблице, то необходимо использовать алгоритмы сортировки, требующие наименьшее количество операций перестановки. Если сортировка производится по таблице ключей, эффективнее использовать сортировки с наименьшей сложностью работы.

Вывод

При выполнении лабораторной работы была реализована обработка и сортировка таблиц. При сортировке больших таблиц эффективно использовать массив индексов и ключей. Это увеличивает скорость сортировки в 2-4 раза в зависимости от исходной таблицы.