Условие 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.

Техническое задание

Приобрести навыки работы с типом данных «запись» (структура), содержащим вариантную часть (объединение, смесь), и с данными, хранящимися в таблицах, произвести сравнительный анализ реализации алгоритмов сортировки и поиска информации в таблицах, при использовании записей с большим числом полей, и тех же алгоритмов, при использовании таблицы ключей; оценить эффективность программы по времени и по используемому объему памяти при использовании различных структур и эффективность использования различных алгоритмов сортировок.

Входные данные

Пункт меню (число от 0 до 9), файл с записями квартир, параметры добавляемой / удаляемой квартиры.

Выходные данные

Текущее состояние таблицы, результаты сравнения эффективности сортировок, результаты поиска по заданным полям.

Возможные аварийные ситуации

Некорректный ввод данных

Структуры данных

Структура самой таблицы:

```
typedef struct table
{
   appartments_t *appartments;
   keys_t *keys;
   short size;
   short maxsize;
}
```

Структура массива ключей:

```
typedef struct keys
{
   int area;
   short id;
} keys_t;
```

Сами квартиры описываются с помощью трех структур и одной смеси:

```
typedef struct appartments
{
   char address[ADDRESS_MAX_SIZE];
   int area;
   short rooms;
   int square_meter_price;
   bool is_primary;
   primary_t flat;
}
```

```
typedef union primary
{
    is_primary_t primary;
    not_primary_t secondary;
} primary_t;

typedef struct is_primary
{
    bool is_decoration;
} is_primary_t;

typedef struct not_primary
{
    short build_time;
    short previous_count;
    short last_count;
    bool is_animals;
} not_primary_t;
```

Замеры времени

Время сортировки (в тактах процессора, процессор со средней частотой ~ 2.5gHz)

Количество записей	Пузырек		QuickSort	
	Таблица	Массив ключей	Таблица	Массив ключей
10	12084	6475	30030	25956
50	175794	91551	52497	37902
150	1502460	483906	208017	126576
500	20992533	6382914	394554	240063

Объем занимаемой памяти:

Количество записей	Таблица	Массив ключей
10	1320	80
50	6600	400
150	19800	1200
500	66000	4000

Количество записей	Занимаемый % массива ключей от всей таблицы	% роста скорости сортировки массива ключей по сравнению с таблицей (пузырек)	% роста скорости сортировки массива ключей по сравнениню с таблицей (QuickSort)
10	~6%	~200%	~115%
50	~6%	~192%	~138%
150	~6%	~310%	~164%
500	~6%	~328%	~165%

Выводы по проделанной работе

Чем больше размерой самой таблицы, тем более эффективнее становится сортировка массива ключей, но даже на маленьких размерах таблицы, сортировка массива ключей происходит быстрее, чем сортировка самой таблицы. Однако, для хранения массива ключей нужна дополнительная память. В моем случае понадобилось не так много дополнительной памяти под массив ключей, но, если бы я в качествев поля выбрал бы не площадь квартиры (тип int, 4б), а, например, строку, длиной хотя бы 10 символов, то затраты на память возросли. Стоит отметить, что использование массива ключей неэффективно при небольших размерах самой таблицы. В таком случае эффективнее просто отсортировать таблицу, так как разница во времени будет несущественна, а затраты на память сократятся.

Контрольные вопросы

Как выделяется память под вариантную часть записи?

В языке си, вариативная часть структуры реализована с помощью union. Память выделяется в одном “куске” памяти, имеющий размер, который способен вместить наибольшее поле из указанных.

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

Результат будет системно-зависимым и трудно предсказуемым. Возможно, произойдет приведение типов.

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

Ответственность за правильность проведения операций целиком и полностью лежит на программисте. *“Следить и помнить, какие именно данные были помещены в объединение, - это забота программиста”* — Брайан Керниган, Деннис Ритчи. Язык программирования Си.

Что представляет собой таблица ключей, зачем она нужна?

Таблица ключей представляет собой таблицу, в которой находится два столбца: номер ячейки в исходной таблице и значение выбранного программистом поля исходной таблицы для этой ячейки (в моем случае – площадь квартиры).

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

Обрабатывать данные в самой таблице эффективнее использовать когда время обработки не так важно, как задействованная память. А использование таблицы ключей, наоборот, эффективно когда нужно быстрое время обработки и не так важна дополнительная задействованная память. Так же, использование таблицы неэффективно, когда сама таблица состоит из маленького количества полей, например, таблица, имеющая два поля: “Ученик” и “Оценка”. В таком случае, таблица ключей будет лишь занимать дополнительное место в памяти и не даст никакой выгоды во времени.

Какие способы сортировки предпочтительнее для обработки таблиц и почему?

Для таблиц из большого количества записей предпочитетльно использовать стандартные и устойчивые способы сортировки, со средним временем обработки O(n*log n), такие как QuickSort, MergeSort и т.д. Если же в таблице не так много записей, то предпочтительнее использовать простые алгоритмы сортировки, например,сортировку пузырьком.