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 Гурова Наталия Алексеевна

Группа ИУ7 – 34Б

Принял Силантьева Александра Васильевна

2021 г.

Оглавление

Описание условия задачи

- Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами). Упорядочить данные в ней по возрастанию ключей, где ключ – любое невариантное поле (по выбору программиста), используя:
  - саму таблицу
  - массив ключей

  (возможность добавления и удаления записей в ручном режиме обязательна).
- Произвести поиск информации по вариантному полю
- Упорядочить таблицу, по возрастанию ключей (где ключ – любое невариантное поле по выбору программиста (в моем случае — площадь квартиры), используя:
  - а) исходную таблицу;
  - б) массив ключей,

  используя 2 разных алгоритма сортировки (простой, ускоренный).
- Оценить эффективность этих алгоритмов (по времени и по используемому объему памяти) при различной реализации программы.
- Интерфейс программы должен быть понятен неподготовленному пользователю. При разработке интерфейса программы следует предусмотреть:
  - указание формата и диапазона данных при вводе и (или) добавлении записей;
  - указание операций, производимых программой;
  - наличие пояснений при выводе результата;
  - возможность добавления записей в конец таблицы и удаления записи по значению указанного поля;
  - просмотр отсортированной таблицы ключей при несортированной исходной таблице;

- вывод упорядоченной исходной таблицы;
- вывод исходной таблицы в упорядоченном виде, используя упорядоченную таблицу ключей
- вывод результатов сравнения эффективности работы программы при обработке данных в исходной таблице и в таблице ключей;
- вывод результатов использования различных алгоритмов сортировок.

- Одним из результатов работы программы должна быть количественная информация (лучше представить в виде таблицы) с указанием времени, затраченного на обработку исходной таблицы и таблицы ключей двумя алгоритмами сортировки (при этом, не забыть оценить так же время выборки данных из основной таблицы с использованием таблицы ключей), а также - объем занимаемой при этом оперативной памяти.
- При тестировании программы необходимо:
  - проверить правильность ввода и вывода данных (в том числе, отследить попытки ввода неверных по типу данных в вариантную часть записи);
  - обеспечить вывод сообщений при отсутствии входных данных («пустой ввод»);
  - проверить правильность выполнения операций;
  - отследить переполнение таблицы. При хранении исходных данных в файлах необходимо также проверить наличие файла и изменения информации в нем при удалении и добавлении данных в таблицу.

Мой вариант данной лабораторной работы:

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.

ОПИСАНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ

Входные данные:

1. **Файл с данными:** текстовый файл формата txt. Разделителем в файле является символ “;”. Каждая новая запись таблицы в обязательном порядке должна находиться на новой строке. Запись содержит :
    - Адрес квартиры (до 25 символов)
    - Площадь квартиры
    - Количество комнат в квартире
    - Цена квадратного метра
    - Признак первичности жилья (0 или 1)
    - Вид жилья
        - первичное
            - с отделкой (0 или 1)
        - вторичное
            - год постройки
            - количество прежних жильцов
            - количество прежних собственников
            - признак наличия животных (0 или 1)

2. **Целое число, представляющее собой номер команды**: целое число в диапазоне от 0 до 12.

Выходные данные:

1. Полученная таблица (основная или таблица ключей) в отсортированном или неотсортированном виде (в зависимости от выполненной команды).
2. Характеристика сравнения вариантов сортировки таблицы.

Функции программы:

1. Добавить квартиру в таблицу
2. Удалить квартиру из таблицы
3. Отсортировать таблицу с данными простой сортировкой
4. Отсортировать таблицу ключей простой сортировкой
5. Отсортировать таблицу с данными ускоренной сортировкой
6. Отсортировать таблицу с ключей ускоренной сортировкой
7. Распечатать таблицу с данными
8. Распечатать таблицу с ключами
9. Распечатать выборку квартир с заданными параметрами
10. Вывести таблицу с данными согласно текущей таблице ключей
11. Считать таблицу из файла заново
12. Распечать отчет о затраченной памяти и времени

Обращение к программе:

Запускается через терминал. Так же можно собрать программу используя makefile и запустить ее с помощью команды release.

Аварийные ситуации:

1. Некорректный ввод номера команды.

   На входе: число, большее чем 12 или меньшее, чем 0.

   На выходе: сообщение «Invalid mode»

2. Некорректный ввод данных с клавиатуры или из файла (data.txt)

ОПИСАНИЕ СТРУКТУРЫ ДАННЫХ

Используемый именной тип данных flat_t представляет собой структуру, определяющую квартиру.

```
typedef struct
{
    char address[MAX_STR];
    int s;
    int count_rooms;
    int price;
    int is_it_old;
    union
    {
        new_flat_t new_flat;
        old_flat_t old_flat;
    } type;
} flat_t;
```

Поля структуры:

- Address – адрес квартиры
- s – площадь квартиры
- count_rooms – количество комнат в квартире
- price – цена квартиры
- is_it_old – флаг, показывает старая квартира или нет
- type – тип квартиры

```
typedef struct
{
    int is_decorate;
} new_flat_t;
```

Поля структуры:

- is_decorate – сделан ли ремонт в квартире

```
typedef struct
{
    int year;
```

```
    int owners;
    int residents;
    int was_animals;
} old_flat_t;
```

Поля структуры:

- year – год постройки
- residents – количество прежних жильцов
- owners – количество прежних собственников
- was_animals – флаг, были ли животные

Все квартиры хранятся в массиве

```
#define MAX_SIZE 1000
flat_t flats[MAX_SIZE];
```

Используемый именной тип данных key_t представляет собой структуру, определяющую ключ для одной квартиры.

```
typedef struct
{
    int ind;
    int s;
} key_t;
```

Поля структуры:

- ind– индекс квартиры
- s – площадь соотв. квартиры

Описание алгоритма

1. Выводится меню данной программы.
2. Пользователь вводит номер команды из предложенного меню.
3. Пока пользователь не введет 0 (выход из программы), ему будет предложено вводить номера команд и выполнять действия по выбору.

Оценка эффективности

В данной лабораторной работе мной были рассмотрены и сравнены два вида сортировки – пузырьком и qsort.

Ключ (поле для сортировки) – площадь квартиры.

Замеры времени (в миллисекундах) были произведены на списке, содержащем 7000 элементов (замеры сделаны 1000 раз):

```
Results (700 flats):

            bub. sort (dataset)     bub. sort (keys)     qsort (dataset)     qsort (keys)
time (ms)   1.538000                0.920000             0.030000            0.011000
memory (B)  44800                   50400                44800               50400
```

Вывод: использование таблицы ключей увеличивает затраты памяти (примерно на 20% для таблицы с 700 записями), но при этом дает качественный выигрыш по времени (примерно в 1,5 раза). Так же уменьшить время работы программы можно используя более эффективные алгоритмы сортировки (в данном случае использование алгоритма q-sort против bubble-sort дает уменьшение времени работы в 3 раза).

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1.Как выделяется память под вариантную часть записи?

Размер памяти, который выделяется 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, несоответствующие описанным?

При компиляции тип данных в вариантной части не проверяется. Поведение будет неопределенным из-за того, что невозможно корректно считать данные.

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

За правильностью выполнения операций с вариантной частью должен следить программист.

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

Таблица ключей представляет собой дополнительный массив (структура), содержащий индекс исходного элемента в исходной таблице и выбранный ключ.

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

Когда мы сортируем таблицу ключей, мы экономим время, так как перестановка записей в основной таблице (которая может содержать большой количество полей) отсутствует. Минус данного подхода в том, что для размещения таблицы ключей требуется дополнительная память. Кроме того, если использовать в качестве ключа символьное поле, то необходимо будет дополнительно обрабатывать данное поле в цикле, что увеличивает время выполнения, так же выбор данных из основной таблицы в порядке, определенном таблицей ключей, замедляет вывод. Если исходная таблица содержит небольшие число полей, то выгоднее обрабатывать данные в самой таблице.

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

Если будет производится сортировка самой таблицы, то необходимо использовать алгоритмы, требующие наименьшее количество операций перестановки. Если же сортировка производится по таблице ключей, то эффективнее использовать сортировки с наименьшей сложностью работы.

Вывод

Одно из преимуществ использования вариантной части записи состоит в том, что тратится меньше памяти, так как один участок памяти (размер равен наибольшему полю в вариантной части) используется сразу для всех значений вариантной части, что экономит память. Один из недостатков — контролировать правильность заполнения такой части должен сам программист, так как компилятор не может отследить ошибку.

При необходимости сортировки больших таблиц данных имеет смысл использовать не массив структур, а массив ключей в случае, если выигрыш по времени больше, чем проигрыш по памяти.

(см вывод по таблице эффективности стр.9)