|  |  |
| --- | --- |
|  | **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)** |

Отчет по лабораторной работе № 2

«Сортировка»

Работу выполнил:

студент группы ИУ7-32Б

Богатырев И.С.

Работу проверил:

Барышникова М.Ю.

Никульшина Т.А.

Москва, 2020 г.

**Условие задачи**

Имеются описания:

Type жилье = (дом, общежитие);

Данные: Фамилия, имя, группа, пол (м/ж),

возраст, средний балл за сессию, дата поступления,

aдрес :

дом :

(улица, No-дома, No-кв );

общежитие :

(Noобщ.,Noкомн.);

Ввести общий список студентов. Вывести список студентов, живущих в общежитииуказанного года поступления.

**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 студентов, параметры добавляемых / удаляемых студентов.

**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

{

student\_t \*\*students;

keys\_t \*keys;

short size;

} table\_t;

**Структура массива ключей**

typedef struct keys

{

int age;

short id;

} keys\_t;

Данные студентов описываются с помощью четырех структур и одной смеси:

typedef struct date

{

unsigned int year;

unsigned short int month;

unsigned short int day;

} date\_t;

typedef struct is\_dormitory

{

unsigned short int dormitory\_num;

unsigned short int room\_num;

} is\_dormitory\_t;

typedef struct is\_house

{

char \*street;

unsigned short int house\_num;

unsigned short int apartment\_num;

} is\_house\_t;

typedef union address

{

is\_dormitory\_t dormitory;

is\_house\_t house;

} address\_t;

typedef struct student

{

char \*surname;

char \*name;

short int group;

bool sex; // 0 - female, 1 - male

short int age;

double average\_score;

date\_t date;

bool is\_dormitory;

address\_t address;

} student\_t;

**Замеры времени**

**Время сортировки**

|  |  |  |  |
| --- | --- | --- | --- |
| Количество записей | Пузырек Таблица | Пузырек Ключи | QuickSort Таблица |
| 10 | 1127 | 1194 | 2808 |
| 50 | 10800 | 9560 | 6288 |
| 250 | 214285 | 163165 | 16255 |
| 500 | 717704 | 623341 | 49453 |
| 1000 | 2892862 | 2460927 | 65301 |

**Объем занимаемой памяти**

|  |  |  |
| --- | --- | --- |
| Количество записей | Таблица | Массив ключей |
| 10 | 640 | 40 |
| 50 | 3200 | 200 |
| 250 | 16000 | 1000 |
| 500 | 32000 | 2000 |
| 1000 | 64000 | 4000 |

**Сравнение в процентах**

|  |  |  |  |
| --- | --- | --- | --- |
| Количество записей | Занимаемый % массива ключей от всей таблицы | % уменьшение времени сортировки массива ключей по сравнению с таблицей (пузырек) | % уменьшение времени сортировки массива ключей по сравнениню с таблицей (QuickSort) |
| 10 | 6.25% | +5.9% | -43.4% |
| 50 | 6.25% | -11.5% | -20.3% |
| 150 | 6.25% | -23.9% | -41.6% |
| 500 | 6.25% | -13.3% | -63.2% |
| 1000 | 6.25% | -14.9% | -53.5% |

**Выводы по проделанной работе**

Сортировка массива ключей подтвердила свою большую эффективность по сравнению с простой сортировкой массива. Чем больше размер матрицы, тем больший выигрыш по времени сортировки дает этот способ. С малыми данными выигрыш минимален (а в одном случае скорость даже совсем несущественно упала), но даже так чаще всего этот способ окупается по времени, хотя и приводит к дополнительной нагрузке по памяти.

В моем случае понадобилось не так много дополнительной памяти под массив ключей, но если бы я в качестве поля выбрал не возраст (тип short int), а, например, вещественное число средней оценки (тип double) или строку хотя бы 10 символов длиной, то затраты на память существенно возросли.

**Контрольные вопросы**

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

В языке C вариативная часть структуры реализована с помощью union. Память выделяется в одном “куске” памяти, имеющий размер бОльшего поля.

**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 – возраст студента).

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

Обрабатывать данные в самой таблице эффективнее, когда время обработки не так важно, как задействованная память. А использование таблицы ключей, наооборот, эффективно, когда нужно быстрое время обработки и не так важна дополнительно 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.