|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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)**

**Лабораторная работа №2 отчёт**

**Типы и структуры данных**

**«Записи с вариантами, обработка таблиц»**

**Выполнил:**

**Исупов Андрей ИУ7-35Б**

**Описание 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 и массив ключей. Реализовать добавление и удаление записей в таблице.

Ввести список абонентов, содержащий фамилию, имя, телефон, адрес, статус (личный – дата рождения: день, месяц, год; служебный – должность, организация). Найти всех друзей, которых необходимо поздравить с днем рождения в ближайшую неделю.

**Техническое задание:**

**Входные данные:**

* текстовый файл с информацией в несколько строк: имя, фамилия, номер телефона, адрес, тип подписки, если личная, то дата рождения, если служебная, то должность, на следующей организация.
* Номер команды.
* При добавлении поля записи, при удалении выбор поля и его значения.

**Выходные данные:**

* Полученная таблица (основная или таблица ключей) в отсортированном или неотсортированном виде (в зависимости от выполненной команды).
* Количественная характеристика сравнения вариантов сортировки таблицы.

**Функции программы:**

1. Справочная информация о формате ввода.

2. Вывести всех абонентов на экран.

3. Перейти в меню сортировки.

4. Добавить абонента в конец таблицы.

5. Удалить абонента по указанному полю.

6. Закрыть программу.

В качестве сортировки с квадратичной сложностью O(n2) используется сортировка пузырьком, с O(nlogn) – алгоритм быстрой сортировки.

**Обращение к программе:** запускается из терминала.

**Аварийные ситуации:**

* Ошибка открытия файла
* Неправильный ввод поля записи

**Структуры данных:**

**// Структура массива**

typedef struct

{

subscriber sub[MAX\_SUBS\_LENGTH]; // массив записей

unsigned short int length; // длина массива записей

} subscribers;

**// Структура абонента**

typedef struct

{

char surname[MAX\_NAME\_LENGTH]; // фамилия

char name[MAX\_NAME\_LENGTH]; // имя

unsigned long int number; // номер телефона

char address[MAX\_ADDRESS\_LENGTH]; // адрес

char status\_type; // тип статуса

number\_status status; // объединение статусов

} subscriber;

**// Объединение статусов**

typedef union

{

private\_status private\_; // личный статус

service\_status service; // служебный статус

} number\_status;

**// Структура служебного статуса**

typedef struct

{

char position[MAX\_NAME\_LENGTH]; // должность

char organization[MAX\_NAME\_LENGTH]; // организация

} service\_status;

**// Структура личного статуса**

typedef struct

{

date birthday; // структура даты

} private\_status;

**// Структура даты**

typedef struct

{

unsigned short int year; // год

unsigned short int month; // месяц

unsigned short int day; // день

} date;

SYMBOL\_LENGTH 1 длина одного символа

MAX\_NAME\_LENGTH 31 максимальная длина имени (фамилии)

MAX\_ADDRESS\_LENGTH 51 максимальная длина адреса

MAX\_SUBS\_LENGTH 50 максимальное количество записей

MAX\_INPUT\_LENGTH 51 максимальная длина вводимой строки

**Алгоритм**

* Пользователь вводит номер команды из меню.
* Пока пользователь не введет 0 (выход из программы), ему будет предложено выполнять действия с таблицей.

**Тесты**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Тест** | **Пользовательский ввод** | **Результат** |
| **1** | **Некорректный ввод комманды** | **13** | **Введена недопустимая команда! Повторите попытку.** |

**Оценка эффективности**

**Время сортировки**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество записей | Сортировка пузырьком | | Быстрая сортировка | |
| Исходная таблица | Таблица ключей | Исходная таблица | Таблица ключей |
| 50 | 2780268 | 73884 | 1576203 | 25864 |
| 100 | 5174736 | 349872 | 2768808 | 62832 |
| 150 | 8476456 | 539338 | 4037900 | 85500 |
| 200 | 13485184 | 799512 | 6036096 | 116712 |
| 250 | 18818800 | 1198512 | 9117288 | 144192 |
| 300 | 22595916 | 1474588 | 10000881 | 183108 |

**Объём занимаемой памяти (в байтах):**

|  |  |  |
| --- | --- | --- |
| Количество записей | Исходная таблица\* | Таблица ключей |
| 50 | 52200 | 400 |
| 100 | 104400 | 800 |
| 150 | 156600 | 1200 |
| 200 | 208800 | 1600 |
| 250 | 261000 | 2000 |
| 300 | 313200 | 2400 |

\*большой размер обусловлен большим количеством строковых полей, каждое из которых занимает по 256 байт.

|  |  |  |  |
| --- | --- | --- | --- |
| Количество записей | Занимаемый % массива ключей всей таблицы | Рост скорости сортировки массива ключей по сравнению с таблицей (сортировка пузырьком) | Рост скорости сортировки массива ключей по сравнению с таблицей (быстрая сортировка) |
| 50 | ~1% | ~38 раз | ~61 раз |
| 100 | ~1% | ~14 раз | ~44 раза |
| 150 | ~1% | ~15 раз | ~47 раз |
| 200 | ~1% | ~17 раз | ~52 раза |
| 250 | ~1% | ~16 раз | ~63 раза |
| 300 | ~1% | ~15 раз | ~55 раз |

**Контрольные вопросы**

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

Размер памяти, выделяемый под вариантную часть, равен максимальному по длине полю вариантной части.

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

Контроль за правильностью выполнения операций с вариантной частью записи возлагается на программиста.

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

Дополнительный массив (структура), 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?**

В случае, если мы сортируем таблицу ключей, мы экономим время, так как перестановка записей в исходной таблице, которая может содержать большое количество полей, отсутствует. С другой стороны, для размещения таблицы ключей требуется дополнительная память. Кроме того, если в качестве ключа используется символьное поле записи, то для сортировки таблицы ключей необходимо дополнительно обрабатывать данное поле в цикле, следовательно, увеличивается время выполнения. Выбор данных из основной таблицы в порядке, определенном таблицей ключей, замедляет вывод. Если исходная таблица содержит небольшое число полей, то выгоднее обрабатывать данные в самой таблице.

**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 большом количестве полей таблицы более эффективно будет использовать отдельную таблицу ключей для сортировки. Однако минус в таблице ключей в том, что под неё нужно выделять отдельную память. В моём случае она занимала не так много. При большом количестве записей и маленьком количестве полей в таблице будет скорее всего более эффективно будет просто отсортировать саму таблицу, чтобы снизить нагрузку на память.