|  |  |
| --- | --- |
| **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.

**Функции**

**Interface.h**

// Интерфейс главного меню

void print\_main\_menu\_interface();

// Интерфейс меню сортировки

void print\_sort\_menu\_interface();

// Интерфейс меню сравнения сортировок

void print\_sort\_comparison\_interface();

// Интерфейс информации о корректном вводе

void print\_input\_info();

**Sorting.h**

// Перестановка записей

void swap(subscriber \*a, subscriber \*b);

// Перестановка ключей

void swap\_keys(int \*a, int \*b);

// Вывод ключей на экран

Void print\_keys(int \*a, int l);

// Сравнение записей

int subcmp(subscriber \*a, subscriber \*b);

// Сортировка пузырьком без ключей

double bubble\_sort\_no\_keys(subscribers \*subs);

// Сортировка пузырьком с ключами

double bubble\_sort\_with\_keys(subscribers \*subs);

// Быстрая сортировка без ключей

double quick\_sort\_no\_keys(subscriber \*subs, int first, int last);

// Быстрая сортировка с ключпми

void qsort\_with\_keys(subscriber \*subs, int \*keys, int first, int last);

// Запуск быстрой сортировки с ключами

double quick\_sort\_with\_keys(subscribers \*subs);

// Главная функция сортировки

double main\_sort(subscribers \*subs, int choise, int print);

// Меню сортировки

error\_code sort\_subs(subscribers \*subs);

**Subscribers.h**

// Глубокое копирование записи

void sub\_cpy(subscribers \*subs, subscribers \*new);

// Подсчёт полей из файла

size\_t read\_count\_of\_elements(FILE \*file);

// Форматирование строки при вводе

void formating(char \*string, int max\_length);

// Считывание записей из файла

void read\_elements(FILE \*file, subscribers \*subs);

// Считывание записей

error\_code read\_subs(subscribers \*subs);

// Функция вывода на экран

void print\_all\_subscribers(subscribers \*subs);

// Считывание строки

void read\_string(char \*s);

// Функция добавления абонента

error\_code add\_sub(subscribers \*subs);

// Удаление записи

void rm(subscribers \*subs, int index);

// Выбор и удаление записей

void del\_sub(subscribers \*subs);

// Показать пользователей с днём рождения на этой неделе

void show\_birthsday(subscribers \*subs);

**Тесты**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Тест** | **Пользовательский ввод** | **Результат** |
| **1** | **Некорректный ввод строки** | **ОченьдлинноесловоОченьдлинноеслово** | **Неправильный ввод!** |
| **2** | **Некорректный ввод числа** | **строка** | **Неправильный ввод!** |
| **3** | **Некорректный ввод статуса** | **sp** | **Неправильный ввод!** |
| **4** | **Некорректный ввод года** | **0** | **Неправильный ввод!** |
| **5** | **Некорректный ввод**  **года** | **2021** | **Неправильный ввод!** |
| **6** | **Некорректный ввод**  **месяца** | **0** | **Неправильный ввод!** |
| **7** | **Некорректный ввод**  **месяца** | **13** | **Неправильный ввод!** |
| **8** | **Некорректный ввод**  **дня** | **0** | **Неправильный ввод!** |
| **9** | **Некорректный ввод**  **дня** | **32** | **Неправильный ввод!** |

**Оценка эффективности**

**Время сортировки**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество записей | Сортировка пузырьком (с) | | Быстрая сортировка (с) | |
| Исходная таблица | Таблица ключей | Исходная таблица | Таблица ключей |
| 40 | 0.000031 | 0.000025 | 0.000087 | 0.000022 |
| 80 | 0.000064 | 0.000051 | 0.000174 | 0.000024 |
| 120 | 0.000093 | 0.000071 | 0.000258 | 0.000032 |
| 160 | 0.000116 | 0.000098 | 0.000333 | 0.000047 |
| 200 | 0.000172 | 0.000130 | 0.000451 | 0.000047 |
| 240 | 0.000255 | 0.000180 | 0.000564 | 0.000062 |

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

|  |  |  |
| --- | --- | --- |
| Количество записей | Исходная таблица | Таблица ключей |
| 40 | 12320 | 3880 |
| 80 | 24640 | 7760 |
| 120 | 36960 | 11640 |
| 160 | 49280 | 15520 |
| 200 | 61600 | 19400 |
| 240 | 73920 | 23280 |

|  |  |  |  |
| --- | --- | --- | --- |
| Количество записей | Занимаемый % массива ключей всей таблицы | Рост скорости сортировки массива ключей по сравнению с таблицей (сортировка пузырьком) на: | Рост скорости сортировки массива ключей по сравнению с таблицей (быстрая сортировка) на: |
| 40 | 31% | 24% | 400% |
| 80 | 31% | 25% | 700% |
| 120 | 31% | 31% | 800% |
| 160 | 31% | 18% | 700% |
| 200 | 31% | 32% | 900% |
| 240 | 31% | 42% | 900% |

**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 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?**

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 и небольшому размеру поля, по которому происходит сортировка, 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 31% от исходной таблицы, что является достаточно большим размером. При большом количестве записей и маленьком количестве полей в таблице или большом размере поля, по которому происходит сортировка, будет скорее всего более эффективно будет просто отсортировать саму таблицу, чтобы снизить нагрузку на память.