ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**ОТЧЕТ ПО ЛАБОРОТОРНОЙ РАБОТЕ № 2**

**«ОБРАБОТКА ТАБЛИЦ»**

Студент Байрамгалин Ярослав Ринатович

Группа ИУ7-33Б

1. Описание условия задачи

Создать таблицу, содержащую не меньше сорока записей (тип – запись с вариантами). Упорядочить данные в ней по возрастанию ключей, двумя алгоритмами сортировки, где ключ – любое невариантное поле (по выбору программиста), используя:

* саму таблицу;
* массив ключей.

Возможность удаления и добавления записей обязательна.

Вывести 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.

1. Описание технического задания

***Входные данные:***

Любые входные данные могут содержать только символы латинского алфавита, цифры и знаки пунктуации, если не указано иное. При вводе кириллических символов поведение программы не определено.

*Имя файла*: строка, содержащая не более 29 символов. При существовании файла с указанным именем он будет открыт. При отсутствии будет создан новый файл. При открытии файла неверной структуру (созданного не этой программой) поведение программы не определено.

*Пункт меню:* строка, не превышающая 29 символов. Для взаимодействия с программой необходимо указать число соответствующее пункту меню:

* 1 – для добавления записи в таблицу;
* 2 – для удаления записи из таблицы;
* 3 – для вывода всех абонентов из таблицы, не сортируя;
* 4 – для вывода всех абонентов, отсортированных по фамилии;
* 5 – для вывода всех ключей при сортировке абонентов по имени;
* 6 – для вывода абонентов, имеющих день рождения в течение 7 дней;
* 0 – для завершения работы программы.

При выборе иного пункта меню будет выведено сообщение об ошибке.

*Добавление записи в таблицу:*

* ввод имени (строка не более 29 символов)
* ввод фамилии (строка не более 29 символов)
* ввод номера телефона (строка не более 29 символов)
* ввод города (строка не более 29 символов)
* ввод улицы (строка не более 29 символов)
* ввод дома (строка не более 29 символов)
* ввод квартиры (строка не более 29 символов)
* ввод статуса (строка не более 29 символов) – может принимать значение *personal* или *company*, при вводе иного будет выведено сообщение об ошибке
* если при вводе статуса указано personal:
  + ввод дня даты рождения (целое число от 1 до 31)
  + ввод месяца даты рождения (целое число от 1 до 12)
  + ввод года даты рождения (целое число от 1900 до 2021)
* если при вводе статуса указано company:
  + ввод названия компании (строка не более 29 символов)
  + ввод должности (строка не более 29 символов)

При успешном добавлении записи в таблицы будет выведено соответствующее сообщение.

*Удаление записи из таблицы:* необходимо указать индекс соответствующей записи (можно посмотреть при выводе без сортировки).

При успешном удалении записи из таблицы будет выведено соответствующее сообщение.

*Просмотр записи по ключам:* для просмотра отсортированных по ключам записей после выбора пункта 6 меню будет предложено показать отсортированные по ключам позиции. Для показа необходимо ввести «Д» или «д».

***Выходные данные:***

При открытии файла: сообщение о успешно выполненном открытии или сообщение об ошибке.

При удалении: сообщение о успешно выполненном удалении или сообщение об ошибке.

При добавлении записи: сообщение о успешно добавлении записи или сообщение об ошибке.

При сортировке по фамилии: список абонентов и всех данных о них, отсортированный по фамилии.

При сортировке по имени: значения массива ключей, при сортировке исходного списка абонентов по имени. Далее будет выведено предложение вывести абонентов в порядке, соответствующем ключам. При положительном ответе будет выведен список абонентов и всех данных о них, отсортированный по имени.

При поиске ближайших дней рождения: будет выведен список абонентов, у которых день рождения выпадает на ближайшие 7 дней.

***Действие программы:***

Организация работы с таблицей абонентов с полями, указанными выше.

***Обращение к программе:***

Запускается командой через терминал, находясь в директории, содержащей программу.

***Аварийные ситуации***

1. Введенная строка превышает 29 символов.
2. При указании даты рождения введены числа, выходящие за границы значений, указанных в спецификации.
3. Превышена допустимая длина строки.
4. Пустой ввод.
5. Файл не может быть открыт.
6. Ввод пустой.
7. Описание структуры данных

Для хранения данных об абонентах была разработана структура данных subscriber\_t.

*Примечание:* MAX\_NAME\_LNG *=* 30, MAX\_PHONE\_LNG = 15.

Ниже приведено описание типа subscriber\_t.

typedef struct {  
 char first\_name[MAX\_NAME\_LNG];  
 char last\_name[MAX\_NAME\_LNG];  
 char phone[MAX\_PHONE\_LNG];  
 address\_t address;  
 status\_t status;  
 subscriber\_info\_t subscriber\_info;  
} subscriber\_t;

Поля структуры:

* first\_name – имя;
* last\_name– фамилия;
* phone – телефон;
* address – адрес;
* status – статус;
* subscriber\_info – информация об абоненте;
* key – индекс.

Для хранения адреса была разработана структура address\_t.

*Примечание:* MAX\_ADDRESS\_FIELD\_LNG *=* 30.

Ниже приведено описание типа address\_t.

typedef struct {  
 char city[MAX\_ADDRESS\_FIELD\_LNG];  
 char street[MAX\_ADDRESS\_FIELD\_LNG];  
 char house[MAX\_ADDRESS\_FIELD\_LNG];  
 char apartment[MAX\_ADDRESS\_FIELD\_LNG];  
} address\_t;

Поля структуры:

* city – город;
* street – улица;
* house –дом;
* apartment – квартира.

Для хранения данных об статусе абонента была разработана структура данных status\_t.

Ниже приведено описание типа subscriber\_t.

typedef enum {  
 personal,  
 company  
} status\_t;

Поля структуры:

* personal – статус - личный;
* company – статус - служебный.

Для хранения вариативных данных информации о абонементе было реализовано объединение subscriber\_info\_t.

Ниже приведено описание типа subscriber\_info\_t.

typedef union {  
 personal\_info\_t person\_info;  
 company\_info\_t company\_info;  
} subscriber\_info\_t;

Поля объединения:

* person\_info – информация об абоненте, если его статус - личный;
* company\_info – информация об абоненте, если его статус - служебный.

Для хранения данных о компании была реализована структура company\_info\_t.

Ниже приведено описание типа company\_info\_t.

typedef struct {  
 char name[MAX\_COMPANY\_INFO\_LNG];  
 char position[MAX\_COMPANY\_INFO\_LNG];  
} company\_info\_t;

Поля структуры:

* name – название компании;
* position – должность.

Для хранения личных данных абонента была разработана структура данных personal\_info\_t.

Ниже приведено описание типа subscriber\_t.

typedef struct {  
 date\_t birth\_date;  
} personal\_info\_t;

Поля структуры:

* date\_t – дата рождения.

Для хранения данных о дате рождения была реализована структура date\_t.

Ниже приведено описание типа date\_t.

typedef struct {  
 int day;  
 int month;  
 int year;  
} date\_t;

Поля структуры:

* day– день;
* month–месяц;
* year –год.

1. Описание алгоритма
2. Пользователь вводит пункт меню.
3. Пользователь выбирает соответствующий его желанию пункт меню.
4. При вводе «0» программа завершается.
5. Набор тестов

|  |  |  |  |
| --- | --- | --- | --- |
|  | Описание теста | Ввод | Ожидаемый вывод |
| 1 | Выбран некорректный пункт меню | uuuu | Неверный ввод |
| 2 | Превышено наибольшее возможное число записей в файле | ../too\_many\_elems | Слишком много элементов |
| 3 | Превышена допустимая длина строки (в любом месте) | Aaaaa3718937129371293719371923791237912379217319823712931728371293218779 | Превышена допустимая длина строки |
| 4 | Недопустимое значение дня рождения | 35 | Ошибка ввода |
| 5 | Недопустимое значение месяца рождения | 14 | Ошибка ввода |
| 6 | Недопустимое значение года рождения | 1000 | Ошибка ввода |

1. Оценка эффективности

При сортировке массива с массивом ключей требуется дополнительная память в размере sizeof(int) \* sz, где sz – размер массива. Однако выигрыш по времени является существенным и увеличивается с увеличением размера сортируемой структуры. Таким образом сортировка по ключам может помочь оптимизировать работу с памятью.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| COUNT | QSORT, мс | BUBBLE, мс | QSORT KEY, мс | BUBBLE KEY, мс |
| 45 | 16.8 | 31 | 3.3 | 37 |
| 100 | 69.6 | 157 | 6.3 | 150 |
| 500 | 1850 | 4547 | 55 | 1978 |
| 1000 | 7826 | 18379 | 145 | 7689 |

1. Выводы

Использование массива ключей для сортировки требует некоторое количество дополнительной памяти, однако при большом размере сортируемой структуры размер дополнительной памяти sizeof(int) / sizeof(struct my\_struct)

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

Память под вариативную часть в Си выделяется так, чтобы помещалось поле, занимающее наибольшее место. Наиболее удобный способ реализации – используя объединения (union).

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

Поведение программы не будет определенным. Именно поэтому необходимо контролировать любые данные, поступающее от пользователя.

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

Ответственность за правильность проведения операций возлагается на программиста.

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

Таблица ключей представляет собой массив, содержащий индексы в исходной таблице. Для получения выигрыша по времени (оптимизации объема переставляемой местами информации) при необходимости сортировки таблица ключей представляет более оптимальное решение.

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

В самой таблице эффективнее обрабатывать данные, если необходимо минимизировать затраты по памяти. Использование таблицы ключей же дает выигрыш во времени, но задействует O(n) дополнительной памяти.

* 1. ***Какие способы сортировки предпочтительнее для обработки таблиц и почему?***

Выбор сортировки зависит от конкретной задачи. Одним из самых быстрых по среднему времени работы в Си является встроенный алгоритм qsort. Однако и он не лишен недостатков, так например, он не является устойчивым. При необходимости можно (например, если в данных встречается много повторяющихся элементов) имплементировать собственную реализацию qsort, которая будет являться устойчивой.