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

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

Название предмета: Типы и структуры данных

Студент: Мицевич Максим Дмитриевич

Группа: ИУ7-31Б

2020г.

I. Описание условия задачи

Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами (объединениями)). Упорядочить данные в ней по возрастанию ключей, двумя алгоритмами сортировки, где ключ – фамилия абонента, используя: а) саму таблицу, б) массив ключей. (Возможность добавления и удаления записей в ручном режиме обязательна).

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.

II. Техническое задание

1. *Описание исходных данных*

Все входные данные являются строками. Изначально все данные читаются в таблицу из файла „in.txt”, далее в цикле вводятся с клавиатуру числа от 0 до 8, которые определяют дальнейшие поведение программы.

Формат ввода:

Возможные пункты меню, которые можно выбрать, введя число с клавиатуры:

0 — завершение программы

1 — вывод таблицы абонентов

2 — вывод таблицы ключей

3 — добавление абонента

4 — удаление абонента по его индексу в таблице

5 — вывод всех абонентов, день рождения у которых будет в ближайшую неделю

6 — сортировка пузырьком таблицы элементов

7 — сортировка пузырьком таблицы при помощи таблицы ключей

8 — результаты времени работы двух разных алгоритмов и методов сортировки

9 — быстрая сортировка таблицы элементов

10 — быстрая сортировка таблицы при помощи таблицы ключей

При удалении абонента дополнительно требуется ввести его индекс в таблице. При добавлении абонента нужно заполнить все его поля. Фамилия состоит из любых символов, кроме пробела и перевода на новую строку, ввод фамилии завершается пробелом, символом новой строки или символом конца файла . Имя, пост, организация - вводятся аналогично фамилии. Номер абонента содержит только цифры и при необходимости знак „+“ на первой позиции, ввод номера завершается аналогично фамилии. Адрес состоит из любых символов, кроме перевода на новую строку и точки, ввод адреса завершается символом новой строки, точки или символом конца файла. Дата рождения читается в формате YYYY MM DD, числа должны отделяться пробельными символами. Статус читается как целое число (0 — личный, 1 - официальный)

Ограничения:

- Длинна фамилии, номера, и названия организации должна быть не нулевой и не больше 15 символов;
- Длинна имени, и занимаемой должности должна быть не нулевой и не больше 10 символов;
- Длинна адреса должна быть не нулевой и не больше 30 символов;

- Статус может быть только 0 или 1
- Дата должна быть реальной и год должен быть не меньше 0 и не больше 2020
- Количество записей в таблице не меньше нуля и не больше 5 тысяч

2. ***Описание результата программы***

При корректных данных будет напечатана таблица, добавлен или удален элемент или напечатана таблица времени работы сортировок, в ином случае будет выведено сообщение об ошибке.

Формат вывода:

<индекс><фамилия><имя><номер телефона><адрес><статус><значение вариативного поля>

Ограничения:

- Время работы сортировки содержит максимум 7 значащих цифр

3. ***Описание задачи, реализуемой программой***

Программа выполняет обработку таблицы абонентов двумя методами (обработка самой таблицы и таблицы ключей (фамилий)). Возможен вывод таблиц элементов и ключей, возможно добавление и удаление элементов из таблицы абонентов, сортировка непосредственно таблицы абонентов и таблицы ключей, получение результатов работы этих сортировок.

4. ***Способ обращения к программе***

Обращение к программе происходит с помощью стандартных потоков ввода и вывода.

5. ***Описание возможных аварийных ситуаций и ошибок пользователя***

- невозможность открытия файла с данными
- превышение количества допустимых символов при вводе строк
- использование некорректных символов при вводе номера телефона
- неправильный ввод статуса абонента
- несуществующая дата или дата, выходящая за рамки ограничений.

Во всех указанных случаях программа завершится корректно и сообщит об ошибке.

III. Описание внутренних структур данных

Для ввода, хранения и вывода данных было организовано 2 структуры данных:

Представление абонента было организовано структурой, в качестве полей которой выступают:

1. Массив из 16 символов для записи фамилии;
2. массив из 11 символов для записи имени;
3. массив из 16 символов для записи номера телефона;
4. массив из 31 символа для записи адреса абонента;
5. перечисление 0 или 1 для хранения статуса абонента;
6. Объединение, содержащее структуру для хранения даты и структуру для хранения должности.

Описание структуры на языке С выглядит следующим образом:

```
typedef struct
{
   char surname[SURNAME_LEN + 1];
   char name[NAME_LEN + 1];
   char phone_number[NUMBER_LEN + 1];
   char addres[ADDRES_LEN + 1];
   status_t status;
   union
   {
      date_t birh_date;
      official_title_t official;
   }extra_inf;
} abonent_t;
```

Представление даты организовано структурой, в качестве полей которой выступают:

1. целое число для хранения года;
2. перечисление 1-12 для хранения месяца;
3. целое число для хранения дня месяца;

Описание структуры на языке С выглядит следующим образом:

```
typedef struct
{
   int year;//0-2020
   month_t month;//1-12
   int day;//1-31
} date_t;
```

Представление должности организовано структурой, в качестве полей которой выступают:

1. массив из 11 символов для хранения занимаемого поста;

2. массив из 16 символов для хранения названия организации;

Описание структуры на языке С выглядит следующим образом:

```
typedef struct
{
   char post[POST_LEN + 1];
   char organization[ORGANIZATION_LEN + 1];
}official_title_t;
```

Для создания таблицы ключей был создана структура, полями которой являются:

1. целое беззнаковое для хранения индекса элемента в исходной таблице;
2. Массив из 16 символов для хранения ключа (фамилии абонента).

Описание структуры на языке С выглядит следующим образом:

```
typedef struct
{
   size_t index;
   char key[SURNAME_LEN + 1];
}abonent_key_t;
```

Таблица абонентов представляет собой массив из элементов типа структуры абонентов.

```
typedef abonent_t abonent_table_t[TABLE_LEN];
```

Таблица ключей представляет собой массив из элементов типа структуры ключа абонента.

```
typedef abonent_key_t abonent_key_table_t[TABLE_LEN];
```

IV. Описание алгоритма

1. Данные из файла „in.txt” читаются в таблицу элементов, далее создается таблица ключей по прочитанной таблице абонентов.
2. Читается команда пользователя и выполняется.
3. При печати таблицы производится печать каждого ее элемента на новой строке.
4. При добавлении в таблицу элемента производится проверка размера таблицы, чтение элемента. Если размер позволяет добавить абонента и он прочитан корректно, то он добавляется в конец таблицы абонентов, создается ключ по этому абоненту, который добавляется в конец таблицы ключей, и размер таблицы увеличивается на 1.
5. При удалении элемента из таблицы производится чтение индекса элемента, который нужно удалить. Если индекс является валидным, то все элементы в таблице абонентов, стоящие за ним, сдвигаются на 1 влево, производится поиск ключа в таблице ключей с введенным индексом и аналогичное смещение, размер таблицы уменьшается на 1.
6. При поиске абонентов, у которых день рождения будет в ближайшую неделю, происходит обход всех абонентов из таблицы абонентов. Для каждого абонента создается копия даты его рождения, в которой в качестве года выступает текущей год. Полученная дата сравнивается с текущей и если разница между ними меньше или равна 7 дням, то происходит печать абонента на экран.
7. Сортировка таблиц ключей и абонентов происходит с помощью алгоритмов сортировки пузырьком и быстрой сортировки. Если сортировка происходит с использованием таблицы ключей, то сортируется эта таблица, а после печатается таблица абонентов в порядке индексов ключей, полученных после сортировки.
8. Подсчета времени, потраченного на сортировку. Засекается время, которое требуется на копирование таблиц элементов и ключей. Для каждого метода и алгоритма сортировки производится подсчет времени в следующем цикле: копирование, выполнение сортировки. Из полученного времени, деленного на количество итераций цикла, вычитается время, потраченное на копирование.

V. Тестирование

Позитивные тесты:

Входные данные (мантисса, порядок, целое)	Описание теста	Результат
1	Печать таблицы элементов	Таблица элементов в читаемом формате
2	Печать таблицы ключей	Таблица ключей в читаемом формате
3 M m +7856 addres. 0 2002 04 03	Добавление элемента	Элемент с указанными полями добавляется в конец таблицы ключей и таблицы абонентов
4 0	Удаление	Из таблицы абонентов удален элемент с индексом 0, из таблицы ключей удален ключ с индексом 0, а индексы всех остальных ключей уменьшены на 1
5	Поиск	Вывод всех абонентов, день рождения которых будет в ближайшую неделю
6	Сортировка исходной таблицы	Исходная таблица отсортирована по фамилиям абонентов. Таблица ключей переписана под исходную
7	Сортировка с использованием таблицы ключей	Таблица ключей отсортирована по ключу, исходная таблица не изменена
8	Информация о времени выполнения сортировок	Исходная таблица и таблица ключей не изменены, на экран выведены результаты времени работы сортировок

Негативные тесты:

Входные данные (мантисса, порядок, целое)	Описание теста	Результат
	Пустой исходный файл	Empty file.
M m mm	Неправильная структура абонента в файле (в номере телефона буква)	Wrong file.
4 -1	Неправильный индекс	Wrong index
3 M mmmmmmmmmm	Превышение размеров имени	Wrong input
3 M m +7852 m. 0 2019 02	Несуществующая дата	Wrong input

29		
3 M m 8745 addres. 2	Неправильный статус	Wrong input

VI. Выводы по проделанной работе

В работе применялась запись с вариантным полем, что сильно помогло сэкономить память, так как вместо выделения памяти под две различные структуры данных и использования только одной из них, мы выделяем память только под одну структуру наибольшую по размеру.

Результаты времени сортировок для 100 элементов:

	buble	quick
table	0.000541s	0.000048s
keys	0.000039s	0.000011s

buble by keys / buble = 0.072078

Выигрыш за счет использования таблицы ключей для сортировки пузырьком получился 93%

quick by keys / quick = 0.229070

Выигрыш за счет использования таблицы ключей для быстрой сортировки получился 78%

quick / buble = 0.088714

Выигрыш за счет выбора метода быстрой сортировки, а не пузырька для исходной таблицы получился 92%

quick by keys / buble by keys = 0.281941

Выигрыш за счет выбора метода быстрой сортировки, а не пузырька для таблицы ключей получился 72%

Результаты времени сортировок для 1000 элементов:

	buble	quick
table	0.087036s	0.001074s
keys	0.005235s	0.000225s

buble by keys / buble = 0.060146

Выигрыш за счет использования таблицы ключей для сортировки пузырьком получился 94%

quick by keys / quick = 0.209393

Выигрыш за счет использования таблицы ключей для быстрой сортировки получился 80%

quick / buble = 0.012338

Выигрыш за счет выбора метода быстрой сортировки, а не пузырька для исходной таблицы получился 99%

quick by keys / buble by keys = 0.042954

Выигрыш за счет выбора метода быстрой сортировки, а не пузырька для таблицы ключей получился 96%

Память, занимаемая одним элементом исходной таблицы: 101 байт

Память, занимаемая одним элементом таблицы ключей: 24 байта

Память, выделяемая под хранение структуры данных увеличилась на 24 процента.

Благодаря использованию таблицы ключей мы получили сильный выигрыш во времени, особенно при сортировке больших таблиц. При обработке таблиц небольших размерностей выигрыш во времени не такой существенный и целесообразнее будет сортировать исходную и не выделять дополнительную память. Выигрыш от метода сортировки больше выражается при сортировке таблиц с большим количеством полей.

VII. Ответы на вопросы

1. Как выделяется память под вариантную часть записи?

   Сравнивается память, занимаемая каждым из полей объединения, и выделяется память под максимальное из них.

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

   Неопределенное поведение.

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

   За правильностью выполнения операций с вариантной частью записи должен следить программист.

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

   Таблица ключей представляет собой массив записей, созданный на основе исходной таблицы. Полями записи являются индекс элемента в исходной таблице и ключ, по которому идет сортировка. Таблица ключей позволяет сэкономить время, затрачиваемое на сортировку. Происходит это благодаря уменьшения времени, затрачиваемого на обмен элементов местами.

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

   Если размер таблицы небольшой, то лучше обрабатывать данные в самой таблице, иначе лучше создавать таблицу ключей.

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

   Для сортировки таблиц предпочтительно использовать устойчивые сортировки, чтобы уменьшить количество перестановок