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»

ОТЧЕТ
Лабораторная работа №2
«Записи с вариантами. Обработка таблиц»

Группа ИУ7-34Б

Дисциплина Типы и структуры данных

Вариант 14

Студент Козлитин Максим Александрович

Преподаватель Силантьева Александра Васильевна

2022г.

Цель работы

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.

Условие задачи

1. Создать таблицу, содержащую 50 записей.

2. Ввести список абонентов, содержащий фамилию, имя, телефон, адрес, статус (личный – дата рождения: день, месяц, год; служебный – должность, организация).

3. Упорядочить данные в ней по возрастанию ключей, двумя алгоритмами сортировки, где ключ – поле, содержащее Фамилию, используя:
   3.1. саму таблицу.
   3.2. массив ключей.

4. Возможность добавления и удаления записей в ручном режиме.

5. Найти всех друзей, которых необходимо поздравить с днем рождения в ближайшую неделю.

Описание исходных данных

1. Выбор действия:

   Целое число от 0 до 18.

2. Фамилия:

   Строка размером 20 символов без пробелов.

3. Имя:

   Строка размером 20 символов без пробелов.

4. Номер телефона:

   Необязательный знак плюс.

   11 цифр - код страны(1 цифра)[знак разделителя]код региона(3 цифры)[знак разделителя]номер(7 цифр).

   В качестве знака разделителя может выступать любые символы, кроме цифр и пробела.

   Пример: +7-111-1234567, 71111234567, 81111234567, 8/111/1234567.

5. Адрес:

   Улица и номер дома через пробел.

   Улица - строка 15 символов без пробелов.

   Номер дома - целое число.

6. Статус:

   Личный - private.

   Служебный - official.

7. Личный (день рождения):

   День, месяц и год - целые числа, разделенные точкой.

   Пример: 14.11.2022

8. Служебный (должность организация):

   Должность и организация через пробел.

   Должность - строка 20 символов без пробелов.

   Организация - строка 20 символов без пробелов.

   Пример: Student BMSTU

9. Абонент

   Через пробел вводятся вся информация.

   Фамилия Имя Номер_телефона Адрес Статус Личный/Служебный

Способ обращения к программе

Работа с программой осуществляется с помощью консоли.

Программа запускается с помощью команды: **./app.exe**

Далее пользователь выполняет взаимодействие с помощью меню.

Описания возможных аварийных ситуаций и ошибок пользователя

1. Ввод не совпадающий с форматом входных данных (описано в Описание исходных данных).
2. Ввод целых чисел превышающих максимальное допустимое значение типа данных int.

Описание внутренних структур данных

Номер телефона

```
typedef struct
{
    char code_country;
    char code_area[CODE_AREA_SIZE];
    char number[NUMBER_SIZE];
} phone_number_t;
```

Код страны - 1 символ.
Код региона - строка из 3 символов.
Номер - строка из 7 символов.

Адрес

```
typedef struct
{
    char street[STREET_LEN];
    int house;
```

```
} address_t;
```

Улица - строка размером 15 символов.
Номер дома - целое число типа int.

Дата

```
typedef struct
{
    int day;
    int month;
    int year;
} date_t;
```

День - целое число типа int.
Месяц - целое число типа int.
Год - целое число типа int.

Личный

```
typedef struct
{
    date_t birthday;
} contact_private_t;
typedef struct
```

День рождения - структура, описанная выше.

Служебный

```
typedef struct
{
    char post[POST_LEN];
    char agency[AGENCY_LEN];
} contact_official_t;
```

Должность - строка размером 20 символов.
Организация - строка размером 20 символов.

Статус

```
typedef union
{
    contact_private_t private;
    contact_official_t official;
} contact_status_t;
```

Личный - структура, описанная выше.
Служебный - структура, описанная выше.

Использование объединения обусловлено экономией памяти, так как размер объединения будет равен максимальному размеру его типов, а не сумме всех.

Абонент

```
typedef struct {
    char surname[SURNAME_LEN];
    char name[NAME_LEN];
    phone_number_t phone_number;
    address_t address;
    char status_type[STATUS_LEN];
    contact_status_t status;
} contact_t;
```

Фамилия - строка размером 20 символов.
Имя - строка размером 20 символов.
Номер телефона - структура, описанная выше.
Адрес - структура - описанная выше.
Тип статуса - строка размером 20 символов.
Статус - объединение, описанное выше.

Описание алгоритма

Удаление по позиции - **contacts_del_by_pos**:

1. Начиная с позиции для удаления запись переносится в конец, с помощью обменов со следующим элементом.
2. Размер массива записей уменьшается на 1.

Удаление по ключу - **contacts_del_by_key**:

1. Проходясь по таблице ключей, выполняется сравнение текущего ключа с искомым. При совпадении выполняется удаление по позиции в исходной таблице.
2. Массив ключей переинициализируется.

Сортировка выбором - **mysort**:

1. Допустим, что подмножество элементов с начала массива и до текущей позиции отсортированный массив.
2. Тогда на каждой итерации на текущую позицию будем ставить минимальный элемент из правой части исходного массива.
3. Таким образом на каждой итерации начиная с начала массива мы будем устанавливать на текущую позицию элемент, который должен там находится в упорядоченном массиве.

Тестовые случаи

Входные данные
Metheringham Lettie 8-187-9348701 Shasta 230 official Environmental Voonte
Boldison Gladys 6-780-3825572 Dixon 503 private 14.11.2023
Wedon Christoph 1-416-2392972 Butternut 976 private 27.06.2023
Pratley Maddy 4-428-3864795 Northport 53 official Nurse Edgepulse
Ruggiero Edithe 9-349-0639692 Randy 271 private 21.05.2022
Lane Merrill 8-789-4024441 Hayes 84306 official Accountant Dablist
Roach Jerrylee 0-735-1927824 Mallard 5 private 31.07.2023
Nutkins Tadeas 2-093-5241994 Brown 9696 private 16.12.2023
Jagger Emanuele 5-439-2941650 Shore 7 official Recruiter Oozz
Probbings Bobina 4-225-5176176 Havey 52230 private 24.09.2022

№	**Негативные тесты**	
	Что проверяется	**Выходные данные**
1	Пустой ввод абонента	Ошибка: Вы ввели некорректную фамилию!
2	Несуществующий файл	Ошибка: Вы ввели некорректное имя файла!
3	Введена только фамилия абонента	Ошибка: Вы ввели некорректное имя!

№	Негативные тесты	
	Что проверяется	**Выходные данные**
4	Введены только фамилия, имя абонента	Ошибка: Вы ввели некорректный номер телефона!
5	Введены только фамилия, имя, номер телефона абонента	Вы ввели некорректную улицу!
6	Введены только фамилия, имя, номер телефона, улица абонента	Ошибка: Вы ввели некорректный номер дома!
7	Введены только фамилия, имя, номер телефона, адрес абонента	Ошибка: Вы ввели некорректный статус!
8	Неправильный статус	Ошибка: Вы ввели некорректный статус!
9	Введены только фамилия, имя, номер телефона, адрес статус - личный абонента	Ошибка: Вы ввели некорректную дату!
10	Введены только фамилия, имя, номер телефона, адрес статус - служебный абонента	Ошибка: Вы ввели некорректную должность!
11	Введены только фамилия, имя, номер телефона, адрес статус - служебный, должность абонента	Ошибка: Вы ввели некорректную организацию!
12	Разделитель не точка перед годом	Ошибка: Вы ввели некорректные символы при указании месяца!
13	Буква вместо года	Ошибка: Вы ввели некорректные символы при указании года!
14	Буква вместо дня	Ошибка: Вы ввели некорректные символы при указании дня!
15	Добавляется 51 абонент	Ошибка: слишком много контактов (> 50)!

Оценка эффективности

50 элементов:

Сортировка	Вывод упорядоченной исходной таблицы, наносекунды	Вывод исходной таблицы по упорядоченной таблице ключей, наносекунды	Разница
Сортировка выбором	13808	8865	35 %
Быстрая сортировка	3159	692	78 %
Разница	77 %	92 %	

10 элементов:

Сортировка	Вывод упорядоченной исходной таблицы, наносекунды	Вывод исходной таблицы по упорядоченной таблице ключей, наносекунды	Разница
Сортировка выбором	1777	633	65 %
Быстрая сортировка	500	231	54 %
Разница	71 %	63 %	

Элементы для данных экспериментов случайным образом сгенерированы с помощью генератора данных.

Алгоритм измерения времени:

1. Измеряется текущее время - начало.
2. Запускается выполнение функции.
3. Измеряется текущее время - конец.
4. Вычисляется количество наносекунд прошедших от начала и до конца.
5. Полученное значение добавляется в сумму.

6. Данное измерение производится 1000 раз, вычисляя итоговую сумму всех запусков.
7. Ответ - среднее арифметическое.

Можем заметить, что время обработки исходной таблицы без таблицы ключей уступает по эффективности второму способу, даже при небольших размерах таблицы разница ощущается достаточной, чтобы использовать данный метод.

Объем занимаемой памяти:

Абонент, байт	Абонент с ключом, байт	Разница, байт
144	172	28

Стоит отметить, что разница по объему занимаемой памяти не столь велика, и в условиях современного мира, когда в среднем рабочие машины пользователей имеют 8 гигабайт ОЗУ, можно сделать выбор в пользу выигрыша по эффективности во времени работы.

Вывод

В ходе проделанной работы, я научился работать с типом данных «запись» (структура), содержащим вариантную часть, и с данными, хранящимися в таблицах. Был произведен 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. Эффективность программы по времени и по используемому объему памяти при использовании различных структур и эффективность использования различных алгоритмов сортировок (быстрая сортировка, сортировка выбором).

В результате было выявлено, что время обработки исходной таблицы без таблицы ключей уступает по эффективности второму способу, разница достаточно ощутимая, чтобы использовать данный метод. Стоит отметить, что разница по объему занимаемой памяти не столь велика, и в условиях современного мира, когда в среднем рабочие машины пользователей имеют 8 гигабайт ОЗУ, можно сделать выбор в пользу выигрыша по эффективности во времени работы.

Таким образом прирост скорости работы составляет **от 35% до 78%**.
Разница по памяти равна 28 байтам.

Подробные результаты и методы оценки эффективности программы можно изучить в пункте «Оценка эффективности».

Контрольные вопросы

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

   Размер union будет равен размеру его максимального поля (плюс, возможно, какие-то дополнительные неиспользуемые байты, добавленные для целей выравнивания). За счет этого мы не можем хранить сразу два типа. Можно использовать только один из возможных.

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

   При компиляции тип данных переданный в вариантную часть записи не

проверяется, поэтому произойдет неопределенное поведение.

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

   Программист

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

   Таблица ключей - массив структуры из двух значений: индекс и значение. Использование таблицы ключей позволяет ускорить скорость некоторых операций над исходной таблицей. Например поиск по ключевому полю и сортировка. Но, стоит отметить, что на хранение таблицы ключей требуется дополнительная память.

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

   Если ключевое поле используется для поиска\добавления и т.п. операциям, где требуется знать значение ключа.

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

   Для сортировки исходной таблицы требуется выбрать алгоритм с наименьшим кол-вом операций над памятью. Если сравнения элементов избежать не получится никак, то стоит выбирать сортировку с наименьшим количеством обменов. Соответственно, предпочтительно выбирать быстрые сортировки, асимптотика которых равна O(n * log(n)).