**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 ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

**По дисциплине «Типы и структуре данных»**

Название: Записи с вариантами, обработка таблиц

Студент Пискунов Панте

Группа ИУ7 – 36Б

Тип лабораторной работы: Учебная

Вариант № 7

Преподаватель Никульшина Татьяна Александровна

*2022 г.*

## Условие задачи

Создать таблицу, содержащую не менее 40 записей с вариантной частью. Произвести поиск информации по вариантному полю. Упорядочить таблицу, по возрастанию ключей (где ключ – любое невариантное поле по выбору программиста), используя: а) исходную таблицу; б) массив ключей, используя 2 разных алгоритма сортировки (простой, ускоренный). Оценить эффективность этих алгоритмов (по времени и по используемому объему памяти) при различной реализации программы, то есть, в случаях а) и б). Обосновать выбор алгоритмов сортировки. Оценка эффективности должна быть относительной (в %).

## Техническое задание

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.

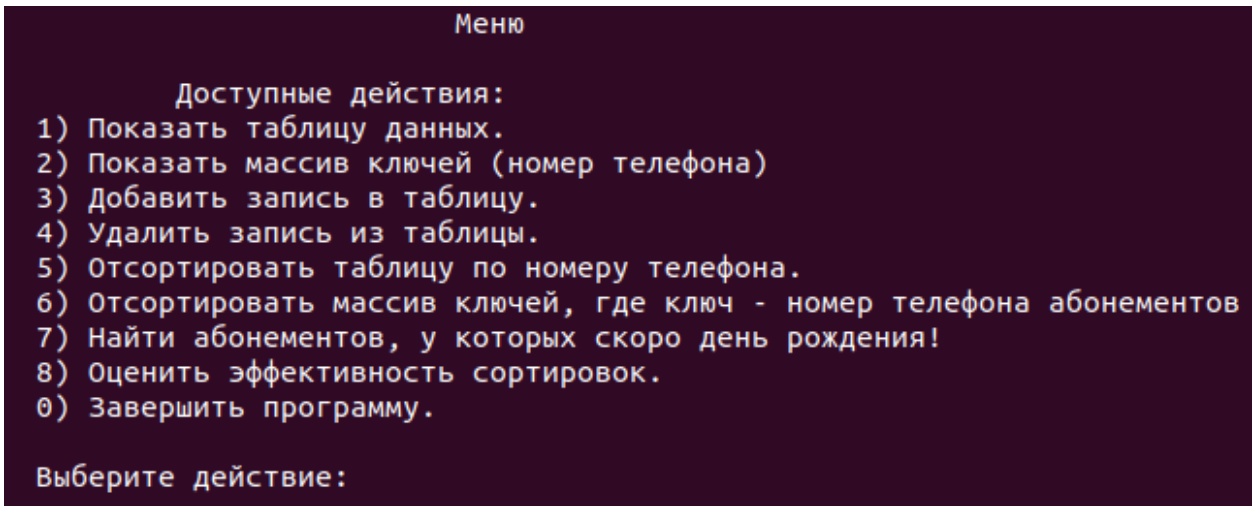
## Описание технического задания

### Входные данные:

Основная структура данных, используемая в программе – массив записей с вариативной частью (таблица)

Запись имеет следующие поля:

1. Фамилия абонемента. (строка)
2. Имя абонемента. (строка)
3. Номер телефона абонемента. (целое неотрицательное число)
4. Название улицы проживания. (строка)
5. Дом проживания. (строка)
6. Статус. (целое число)
7. Если личный статус :
8. День рождения абонемента(день, месяц, год) (целые  неотрицательные числа)
9. Если служебный статус :
10. Должность. (строка)
11. Название организации/компании. (строка)




Тип входных данных

- Цифра, которая указывает на пункт меню, который необходимо выполнить.
- Информационные данные:
  - Строки, определяющие какой-то признак (например время сортировки)
  - Файл, в котором лежит таблица с абонементами(table.txt)

**Выходные данные:**
1 ) Таблица.
2) Таблица ключей (ключ — номер телефона абонементов).
3) Результат эффективности алгоритмов сортировок.
4) Результат поиска дня рождения абонементов в ближайшую неделю.
5) Время сортировки.

**Действие программы:**
Найти день рождения абонементов в ближайшую неделю.

**Способ обращение к программе:**
Запускается через терминал.
(с помощью команды **./app.exe**)

**Сообщения при аварийных ситуациях:**

1 Ошибка: В числе неправильный ввод;
2 Ошибка: Неправильно ввели номер телефона абонемента;
3 Ошибка: Неправильно выбрали сортировку;

4 Ошибка: Неправильно указали статус абонемента;
5 Ошибка: Неправильно ввели название организации/компании, где работает абонемент;
6 Ошибка: Неправильно ввели день рождения абонемента;
7 Ошибка: Неправильно указали название улицы проживания абонемента;
8 Ошибка: Неправильно указали дом проживания абонемента;
9 Ошибка: Неправильно указали фамилию абонемента;
10 Ошибка: Неправильно указали имя абонемента;
11 Ошибка: Таблица полная;
12 Ошибка: Такой абонемент не существует;
13 Ошибка: Такого номера нет;

## Структура данных

В программе используются несколько структур данных :

```
// Структура, показывающая статус абонементов
typedef enum
{
personal, // личный статус
company // службеный статус
}status_t;

// Структура, показывающая дата рождения абонементов
typedef struct data
{
size_t day; // день рождения
size_t month; // месяц рождения
size_t year; // год рождения
}data_t;

// Структура, показывающая службеный статус абонементов
typedef struct company
{
char name[MAX_DATA_SIZE + 1]; // должность абонементов
char position[MAX_DATA_SIZE + 1]; // организация/компания
}company_info_t;

// Структура, показывающая личный статус абонементов
typedef struct personal_info
{
data_t birthday; // дата рождения абонементов
}personal_info_t;

// Структура, объединяющая личный и службеный статус абонементов
```

```
typedef union subscriber_info
{
personal_info_t person_info; // полная личная информация о абонементе
company_info_t company_info; // полная службеная информация о абонементе
}subscriber_info_t;

typedef struct subscriber
{
char first_name[MAX_DATA_SIZE + 1]; // имя абонемента
char last_name[MAX_DATA_SIZE + 1]; // фамилия абонемента
size_t phone[MAX_DATA_SIZE + 1]; // номер телефона абонемента
char street[MAX_DATA_SIZE + 1]; // название улицы
char house[MAX_DATA_SIZE + 1]; // дом проживания
status_t status; // статус абонемента
subscriber_info_t subs_info; // информация о статусе абонемента
}subscriber_t;
```

Примечание: MAX_DATA_SIZE = 100;

## Набор тестов

Тесты для главного меню:

| № | Входные данные | Выходные данные |
|---|---|---|
| 1 | Вводится неправильное число | Ошибка: Неправильный ввод |
| 2 | Вводится символ | Ошибка: Неправильный ввод |
| 3 | Вводится ноль | Программа завершена |
| 4 | Указывается несуществующий файл | Ошибка: Не удалось открыть файл! |
| 5 | Файл пустой | Ошибка: Не удалось прочитать файл |

Тесты для ввода персональную информацию о абонементе

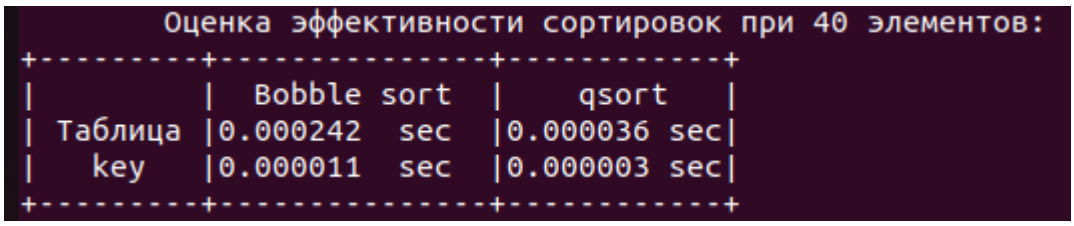
| № | Входные данные | Выходные данные | Проверяется |
|---|---|---|---|
| 1 | Вводится строка(фамилия | Vagapov | Корректная работа |

| | | | |
|---|---|---|---|
| | ) | | |
| 2 | Вводится строка(имя) | David | Корректная работа |
| 3 | Вводится число (номер телефона) | 89779931360 | Корректная работа |
| 4 | Вводится строка(название улицы проживания) | Izmailovskay | Корректная работа |
| 5 | Вводится строка(дом проживания) | 75A | Корректная работа |
| 6 | Вводится число (номер телефона) | asdasdasdasd | Ошибка: Неправильно ввело номер телефона абонемента |
| 7 | Для фамилии абонемента вводится пустая строка | | Ошибка: Неправильно ввели фамилию абонемента! |
| 8 | Для имена абонемента вводится пустая строка | | Ошибка: Неправильно ввели имя абонемента! |
| 9 | Для названии улицы абонемента вводится пустая строка | | Ошибка: Неправильно ввели название улицы абонемента! |
| 10 | Для дома абонемента вводится пустая строка | | Ошибка: Неправильно ввели дом абонемента! |

Тесты для ввода служебную информацию о абонементе:

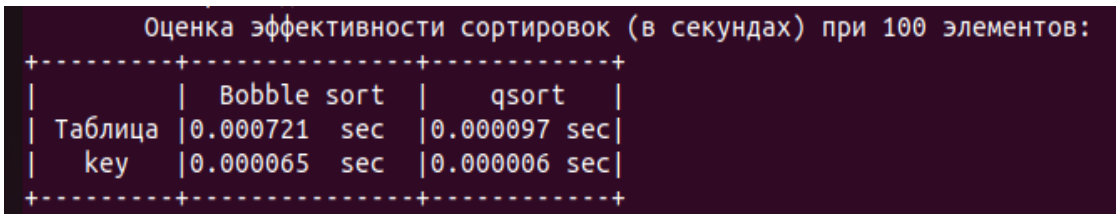
| № | Входные данные | | Проверяется |
|---|---|---|---|
| 1 | Вводится число(статус) | 0/1 | Корректная работа |
| 2 | Если персональная информация: Вводятся целые числа (день месяц и год рождения) | 20 10 2000 | Корректная работа |
| 3 | Если вводится служебная информация: Вводится строка (должность) | programer | Корректная работа |
| 4 | Если вводится служебная информация: Вводится строка (организация) | vk | Корректная работа |
| 5 | Если персональная информация: Вводятся целые числа (день месяц и год рождения) | 16 asd 2000 | Ошибка:  Неправильно ввели день рождения абонемента |
| 6 | Вводится неправильный статус абонемента | | Ошибка: Неправильно указали статус абонемента! |

**Оценка эффективности**



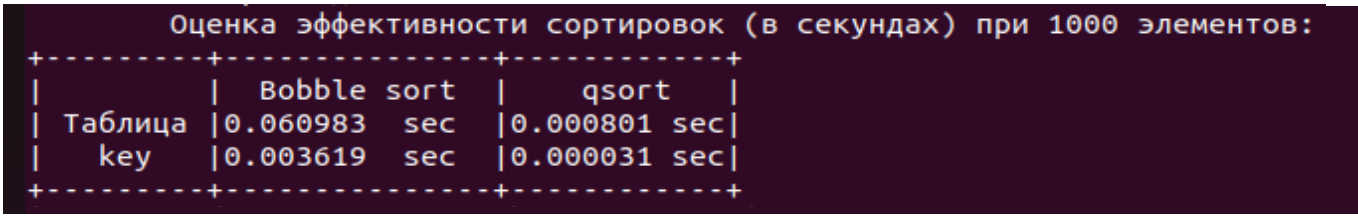
```
        Оценка эффективности сортировок при 40 элементов:
+---------+---------------+------------+
|         |  Bobble sort  |    qsort   |
| Таблица |0.000242  sec  |0.000036 sec|
|   key   |0.000011  sec  |0.000003 sec|
+---------+---------------+------------+
```

Объем памяти при 40 элементов: 320
Разница по времени (Bubble_sort_table) / (Bubble_sort_key): 5.58
Разница по времени (qsort_table) / (qsort_key): 14.67



```
        Оценка эффективности сортировок (в секундах) при 100 элементов:
+---------+--------------+------------+
|         |  Bobble sort |    qsort   |
| Таблица |0.000721  sec |0.000097 sec|
|   key   |0.000065  sec |0.000006 sec|
+---------+--------------+------------+
```

Объем памяти при 100 элементов: 800
Разница по времени (Bubble_sort_table) / (Bubble_sort_key): 11.50
Разница по времени (qsort_table) / (qsort_key): 16.33



```
        Оценка эффективности сортировок (в секундах) при 1000 элементов:
+---------+--------------+------------+
|         |  Bobble sort |    qsort   |
| Таблица |0.060983  sec |0.000801 sec|
|   key   |0.003619  sec |0.000031 sec|
+---------+--------------+------------+
```

Объем памяти при 1000 элементов: 8000
Разница по времени (Bubble_sort_table) / (Bubble_sort_key): 20.15
Разница по времени (qsort_table) / (qsort_key): 32.16

Вывод: По времени qsort в 190 раз эффективнее работает чем bubble_sort(при 100 елементов).

## Вывод

В результате работы были реализованы функции сортировки таблицы (исходной) и таблицы ключей (qsort(сортировка из стандартной библиотеки) и BubbleSort). Также поиск, добавление и удаление номера абонементов в структуре с вариантными полями. Для сортировки использовал таблицу ключей(номер телефона), так как на ее обработку уходит гораздо меньше времени.

## Контрольные вопросы

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

Память под вариативную часть выделяется таким образом, чтобы ее хватало на самую большую возможную структуру.

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

Ничего потому, что они не соответствуют.

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

Программист.

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

Таблица ключей представляет из себя некую структуру, содержащую индекс и значение некоторого выбранного поля таблицы (исходной).

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

Когда много данных, эффективнее обрабатывать данные с использованием ключей. Иначе таблицу.

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

Сортировки, где меньше перестановок**.**