|  |  |
| --- | --- |
| Изображение выглядит как герб, эмблема, символ, нашивка  Автоматически созданное описание | **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 № \_\_**

**ПО ДИСЦИПЛИНЕ «ТИПЫ И СТРУКТУРЫ ДАННЫХ»**

***<Тема лабораторной работы*>**

**Вариант № 7 (порядковый номер в журнале – 14)**

Студент: **Коротков Е.Д.**

Группа: **ИУ7-34Б**

Преподаватель: **Никульшина Т.А.**

**2025 г.**

**Описание условия задачи**

Ввести список абонентов, содержащий фамилию, имя, номер телефона, адрес (улица, дом), статус абонента:

1. Друзья:

a. Дата рождения: день, месяц, год

2. Коллеги:

a. должность

b. организация

Вывести список всех друзей, которых необходимо поздравить с днем рождения в ближайшую неделю (текущая дата вводится пользователем).

Упорядочить данные в таблице по возрастанию ключей, двумя алгоритмами сортировки, где ключ –любое невариантное поле, используя:

а) саму таблицу,

б) массив ключей.

Реализовать возможность добавления и удаления записей в ручном режиме, просмотр таблицы, просмотр таблицы в порядке расположения таблицы ключей.

**Техническое задание**

**Исходные данные:**

**Ввод номера пункта меню**: число – от 1 до 12

**Текстовый файл с данными:** файл, содержащий поля таблицы структур (разделены пробелом)

**Данные, запрашиваемые в ходе выполнения программы:** данные, совпадающие с соответствующими полями структуры, имя входного и выходного файлов.

**Выходные данные:**

Исходная таблица структур, отсортированная таблица структур, исходная и отсортированная таблицы ключей, поля структуры, удовлетворяющие условию задачи (список всех друзей, которых необходимо поздравить с днем рождения в ближайшую неделю (текущая дата вводится пользователем)).

**Реализуемые задачи:**

Пункт меню 1 – ввод таблицы структур из файла

Пункт меню 2 – ввод поля (полей) структуры из стандартного потока (с клавиатуры)

Пункт меню 3 – вывод таблицы структур в файл

Пункт меню 4 – вывод таблицы структур на экран (в консоль)

Пункт меню 5 – поиск и вывод всех друзей, которых необходимо поздравить с днем рождения в ближайшую неделю

Пункт меню 6 – сортировка таблицы структур методом пузырька

Пункт меню 7 – сортировка таблицы структур методом быстрой сортировки (quick sort)

Пункт меню 8 – сортировка ключей методом пузырька

Пункт меню 9 - сортировка ключей методом быстрой сортировки (quick sort)

Пункт меню 10 – сравнение сортировок различными методами (таблица ключей/структур) и алгоритмами (bubble sort/quick sort)

Пункт меню 11 – Выход из программы

Пункт меню 12 – Вывод таблицы ключей

**Способ обращения к программе:**

Запуск в CLI командой ./a.exe

После запуска работа с программой происходит путём введения номеров пунктов меню в консоль.

**Аварийные ситуации:**

1. Memory allocation error – ошибка выделения динамической памяти для массива структур.

Остальные ошибочные ситуации в процессе работы программы не являются аварийными и вызывают повторный ввод данных или выход в меню

**Внутренние структуры данных**

Описание структур, находящихся в файле struct.h

typedef struct

{

    int date;

    int month;

    int year;

    int date\_since\_new\_year;

} date\_t;

Структура для даты. Содержит следующие поля:

Date – день

Month – месяц

Year – год

Date\_since\_new\_year – количество дней, прошедших с начала года

typedef struct

{

    date\_t birth\_date;

} friend\_t;

Структура для абонента «друг». Birth\_date содержит дату рождения.

typedef struct

{

    char job[JOB\_LEN+1];

    char org[ORG\_LEN+1];

} colleague\_t;

Структура для абонента «коллега». Поле Job – содержит должность и поле org – место работы.

typedef enum

{

    FRIEND=1,

    COLLEAGUE=2

} type\_status\_t;

Перечисляемый тип для определения статуса абонента: «друг» (FRIEND) или «коллега» (COLLEAGUE)

typedef union

{

    colleague\_t colleague;

    friend\_t friend;

} subs\_status\_t;

Объединение, которое содержит либо поле friend – информация о абоненте «друг», либо поле colleague информация о абоненте «коллега»

typedef struct

{

    char name[NAME\_LEN+1];

    char surname[SURNAME\_LEN+1];

    char phone[PHONE\_LEN+1];

    char address[ADDRESS\_LEN+1];

    type\_status\_t status;

    subs\_status\_t subscriber;

} subscriber\_t;

Структура, содержащая полную информацию о абоненте:

Поле name – имя

Поле surname – имя

Поле phone – номер телефона

Поле address – адрес

Поле status содержит статус абонента: друг или коллега

Поле subscriber – вариантное поле, в зависимости от значения поля status принимает одно из полей объединения subs\_status\_t friend или colleague

typedef struct

{

    int index;

    char surname[SURNAME\_LEN+1];

} key\_table\_t;

Структура для хранения одного элемента таблицы ключей. Содержит поля:

Index – индекс элемента в изначальной таблице

Surname – фамилия (значение, по которой производится сортировка)

**Описание алгоритма**

Пункт 1: Ввод из файла

1. Запрос имени файла у пользователя
2. Попытка чтения данных из указанного файла
3. При успехе - загрузка данных в массив абонентов
4. При ошибке - предложение повторить с другим файлом или вернуться в меню
5. Отображение количества загруженных записей

Пункт 2: Ввод с клавиатуры

1. Последовательный запрос данных для каждого абонента

* Фамилия, имя, телефон, адрес
* Статус (друг/коллега)
* В зависимости от статуса:
* Для друга: дата рождения
* Для коллеги: должность и организация

1. После ввода каждой записи - предложение продолжить или остановиться
2. При ошибке ввода - предложение повторить

Пункт 3: Вывод в файл

1. Проверка наличия данных в таблице
2. Запрос имени файла для сохранения
3. Запись всех данных в указанный файл
4. При ошибке - предложение повторить с другим именем файла

Пункт 4: Вывод на экран

1. Проверка наличия данных в таблице
2. Вывод всех записей в консоль в табличном формате

Пункт 5: Поиск дней рождений

1. Проверка наличия данных в таблице
2. Запрос текущей даты у пользователя
3. Расчет номера дня в году для текущей даты
4. Поиск среди абонентов-друзей тех, у кого день рождения в ближайшие 7 дней
5. Вывод найденных записей с указанием, через сколько дней день рождения

Пункт 6: Сортировка таблицы (пузырьком)

1. Проверка наличия данных в таблице
2. Сортировка основного массива абонентов по фамилии алгоритмом пузырьковой сортировки
3. Предложение вывести отсортированную таблицу на экран
4. Вывод таблицы, в случае согласия пользователя в пункте 3

Пункт 7: Сортировка таблицы (qsort)

1. Проверка наличия данных в таблице
2. Сортировка основного массива абонентов по фамилии с помощью стандартной функции qsort
3. Предложение вывести отсортированную таблицу на экран

Пункт 8: Сортировка через ключи (пузырьком)

1. Проверка наличия данных в таблице
2. Создание таблицы ключей (если не создана ранее)
3. Сортировка таблицы ключей по фамилии алгоритмом пузырьковой сортировки
4. Вывод основной таблицы через отсортированные ключи

Пункт 9: Сортировка через ключи (qsort)

1. Проверка наличия данных в таблице
2. Создание таблицы ключей (если не создана ранее)
3. Сортировка таблицы ключей по фамилии с помощью стандартной функции qsort
4. Вывод основной таблицы через отсортированные ключи

Пункт 10: Сравнение производительности сортировок

1. Проверка наличия данных в таблице
2. Выполнение четырех видов сортировок по 10 итераций каждая:

* Таблица пузырьком
* Таблица qsort
* Ключи пузырьком
* Ключи qsort

1. Расчет среднего времени выполнения для каждого метода
2. Вывод сравнительной таблицы с временами выполнения и коэффициентами ускорения
3. Вывод информации об использовании памяти

Пункт 11: Выход

1. При наличии данных - предложение сохранить их перед выходом
2. При согласии - запрос имени файла и сохранение данных
3. Завершение программы

Пункт 12: Печать таблицы ключей

1. Проверка наличия данных в таблице
2. Создание таблицы ключей (если не создана ранее)
3. Вывод таблицы ключей в формате "Индекс | Фамилия"

**Основные функции:**

Описание функций по модулям:

**Модуль: menu.h**

int menu\_handler(subscriber\_t \*subscribers);

Назначение: Основная функция обработки меню программы

Входные параметры: Указатель на массив абонентов subscriber\_t \*subscribers

Возвращаемое значение: Целое число - код возврата программы (1 при успешном завершении)

**Модуль: io.h**

int get\_single\_subscriber\_from\_stdin(subscriber\_t \*sub);

Назначение: Ввод данных для одного абонента

Входные параметры: subscriber\_t \*sub - указатель на структуру для заполнения

Возвращаемое значение: Целое число (1 - успех, 0 - ошибка)

int get\_data\_from\_stdin(subscriber\_t \*data, int \*len);

Назначение: Интерактивный ввод данных о абонентах с клавиатуры

Входные параметры:

subscriber\_t \*data - указатель на массив для хранения данных

int \*len - указатель на переменную для хранения количества введенных записей

Возвращаемое значение: Целое число (1 - успех, 0 - ошибка)

int get\_based\_info(subscriber\_t \*sub, FILE\* file);

Назначение: Чтение основной информации о абоненте из файла

Входные параметры:

subscriber\_t \*sub - указатель на структуру абонента

FILE\* file - указатель на файл для чтения

Возвращаемое значение: Целое число (1 - успех, 0 - ошибка)

int get\_status\_info(subscriber\_t \*sub, FILE\* file);

Назначение: Чтение вариантной информации о абоненте из файла

Входные параметры:

subscriber\_t \*sub - указатель на структуру абонента

FILE\* file - указатель на файл для чтения

Возвращаемое значение: Целое число (1 - успех, 0 - ошибка)

int get\_data\_from\_file(subscriber\_t \*data, const char \*filename, int

\*len);

Назначение: Чтение данных об абонентах из файла

Входные параметры:

subscriber\_t \*data - указатель на массив для хранения данных

const char \*filename - имя файла для чтения

int \*len - указатель на переменную для хранения количества загруженных записей

Возвращаемое значение: Целое число (1 - успех, 0 - ошибка)

int write\_data\_to\_file(subscriber\_t \*data, int count, const char \*filename);

Назначение: Запись данных о подписчиках в файл или на экран

Входные параметры:

subscriber\_t \*data - указатель на массив данных

int count - количество записей для сохранения

const char \*filename - имя файла или "stdout" для вывода на экран

Возвращаемое значение: Целое число (1 - успех, 0 - ошибка)

void print\_keys\_table(key\_table\_t \*table, int len);

Назначение: Вывод таблицы ключей в формате "Индекс | Фамилия"

Входные параметры:

key\_table\_t \*table - указатель на таблицу ключей

int len - количество элементов

Возвращаемое значение: нет (void)

**Модуль date.h**

int get\_day\_since\_new\_year(int day, int month, int year);

Назначение: Вычисление порядкового номера дня в году

Входные параметры: День, месяц, год (целые числа)

Возвращаемое значение: Целое число (1-365)

void search\_date(subscriber\_t \*subs, int date, int len);

Назначение: Поиск абонентов-друзей с днями рождения в ближайшие 7 дней

Входные параметры:

subscriber\_t \*subs - указатель на массив абонентов

int date - текущий день года

int len - количество абонентов

Возвращаемое значение: нет (void)

**Модуль sort.h**

void bubble\_sort(void \*base, int nmemb, int size, int (\*compar)(const void\*, const void\*));

Назначение: Универсальная сортировка пузырьком для любого типа данных

Входные параметры:

void \*base - указатель на начало массива

int nmemb - количество элементов

int size - размер одного элемента

int (\*compar)(const void\*, const void\*) - указатель на функцию сравнения

Возвращаемое значение: Нет (void)

int compare\_surname(const void \*a, const void \*b);

Назначение: Сравнение двух подписчиков по фамилии

Входные параметры: Два указателя на элементы типа subscriber\_t

Возвращаемое значение: Целое число (значение strcmp: <0, 0, >0)

int compare\_keys(const void \*a, const void \*b);

Назначение: Сравнение двух ключей по фамилии

Входные параметры: Два указателя на элементы типа key\_table\_t

Возвращаемое значение: Целое число (значение strcmp: <0, 0, >0)

void create\_key\_table(subscriber\_t \*table, key\_table\_t \*keys, int len);

Назначение: Создание таблицы ключей на основе основной таблицы

Входные параметры:

subscriber\_t \*table - указатель на основную таблицу

key\_table\_t \*keys - указатель на массив ключей для заполнения

int len - количество элементов в таблице

Возвращаемое значение: Нет (void)

void print\_via\_keys(subscriber\_t \*table, key\_table\_t \*keys, int len);

Назначение: Вывод основной таблицы в порядке, заданном отсортированной таблицей ключей

Входные параметры:

subscriber\_t \*table - указатель на основную таблицу

key\_table\_t \*keys - указатель на отсортированную таблицу ключей

int len - количество элементов

Возвращаемое значение: Нет

void measure\_sorts(subscriber\_t \*table, int len);

Назначение: Сравнительный анализ производительности различных методов сортировки

Входные параметры:

subscriber\_t \*table - указатель на таблицу для тестирования

int len - количество элементов

Возвращаемое значение: Нет

**Тесты**

**Работа меню**

|  |  |  |
| --- | --- | --- |
| **Тест** | **Ввод** | **Вывод** |
| Отрицательное число | **-10** | Invalid mode (-10). Please choose a number between 1 and 12.  Available options:  1-Input from file, 2-Input from keyboard, 3-Output to file, 4-Output to screen  5-Search birthdays, 6-Sort table (bubble), 7-Sort table (qsort)  8-Sort via keys (bubble), 9-Sort via keys (qsort), 10-Compare sorts  11-Exit, 12-Print keys table |
| Положительное число, большее максимума (12) | **15** | Аналогично строке выше |
| Максимальный пункт | **12** | Вывод таблицы ключей |
| Минимальный пункт | **1** | Select mode: 1  Enter filename: test\_100.txt  Data loaded successfully. Records: 100 |
| Некорректный меню | **a** | Input error! Please enter a number between 1 and 12. |
| Пункт меню при пустой таблице | **5** | Table is empty! Nothing to search |

**Чтение файла**

|  |  |  |  |
| --- | --- | --- | --- |
| Тест | ввод | Содержание файла | вывод |
| Несуществующий файл | Data.txt | - | Enter filename: data.txt  Error while opening file  Failed to load data from file: data.txt  Try again with different file? (y/n): |
| Пустой файл |  |  | File is empty |
| Файл с 1 корректной структурой | test1.txt | Ivanov Ivan 79161234567 Lenina5 1 15 5 1990 | Data loaded successfully. Records: 1 |
| Файл с некорректной структурой | Error.txt | Ivanov Ivan 79161234567 Lenina5 2 12 12 12 | Failed to load data from file: error.txt |
| Файл с слишком большим количеством структур | File.txt | 10001 абонент | File reading error or too many records. |

**Ввод с клавиатуры**

|  |  |  |
| --- | --- | --- |
| **Тест** | **Поле** | **Результат** |
| Некорректный статус | 123 | Invalid status, try again: 1 - friend, 2 - colleague |
| Корректный ввод друга | 15 5 1950 | Успешное добавление записи |
| Корректный ввод коллеги | Director BMSTU | Успешное добавление записи |

**Поиск дня рождения**

|  |  |  |  |
| --- | --- | --- | --- |
| **Тест** | **Текущие данные** | **Текущая дата** | **Результат** |
| Пустая таблица | - | 03 10 2025 | Table is empty! Nothing to search |
| Нет подходящих дней рождений | друг: 01.01.1990 | 15.05.2024 | "No birthdays found in the next 7 days" |
| День рождения сегодня | друг: 15.05.1990 | 15.05.2024 | Вывод записи: "(in 0 days)" |
| День рождения через 3 дня | друг: 18.05.1990 | 15.05.2024 | Вывод записи: "(in 3 days)" |
| День рождения через неделю | друг: 22.05.1990 | 15.05.2024 | Вывод записи: "(in 7 days)" |
| День рождения через 8 дней | друг: 23.05.1990 | 15.05.2024 | "No birthdays found in the next 7 days" |
| Переход через год | друг: 01.01.1990 | 30.12.2024 | Вывод записи: "(in 2 days)" |
| Только коллеги | коллега | 15.05.2024 | "No birthdays found in the next 7 days" |

Тестирование сортировок

|  |  |  |  |
| --- | --- | --- | --- |
| Тест | Данные | Алгоритм (пункт меню) | Результат |
| Сортировка пустой таблицы | 0 записей | 6 | Table is empty! Nothing to sort |
| Сортировка одной записи | Ivanov | 6 | Table successfully sorted by surname (bubble sort) |
| Сортировка нескольких записей | Petrov, Ivanov, Sidorov | 6 | Отсортированный порядок: Ivanov, Petrov, Sidorov |
| Сортировка через ключи | Petrov, Ivanov, Sidorov | 8 | Вывод в отсортированном порядке через ключи |
| Сравнение сортировок | 100 записей | 10 | Таблица с временами выполнения 4 методов сортировки |

**Работа с ключами**

|  |  |  |
| --- | --- | --- |
| **Тест** | **Данные** | **Результат** |
| Вывод изначальных ключей | Williams87  Johnson100  Smith73  Williams75  Smith25  Smith77  Johnson36 | 0 Williams87  1 Johnson100  2 Smith73  3 Williams75  4 Smith25  5 Smith77  6 Johnson36 |
| Вывод изначальной таблицы | Williams87  Johnson100  Smith73  Williams75  Smith25  Smith77  Johnson36 | Williams87  Johnson100  Smith73  Williams75  Smith25  Smith77  Johnson36 |
| Вывод таблицы ключей после сортировки | Williams87  Johnson100  Smith73  Williams75  Smith25  Smith77  Johnson36 | 1 Johnson100  6 Johnson36  4 Smith25  2 Smith73  5 Smith77  3 Williams75  0 Williams87 |
| Создание таблицы ключей с пустой структурой | - | Table is empty! No keys to display |

**Результаты сравнения**

Для сравнения произведен замер скорости сортировки при размерах таблицы в 100, 500, 1000 и 10000 элементов (среднее значение за 10 сортировок). Конфигурация тестирующей системы:

ЦП: AMD Ryzen 5 5600x

ГП: NVIDIA RTX 3060TI

ОЗУ: 16 гб

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размер | Bubble sort таблица (сек) | Bubble sort ключи (сек) | Quick sort таблица (сек) | Quick sort ключи (сек) |
| 100 | 0.0002 | 0.000008 | 0.0001 | 0.000007 |
| 500 | 0.003700 | 0.000100 | 0.001800 | 0.000100 |
| 1000 | 0.014300 | 0.000300 | 0.006900 | 0.000100 |
| 10000 | 1.485300 | 0.007000 | 0.709700 | 0.002400 |

Выигрыш по времени: (принимаем Bubble sort таблицы за 1, то есть 100%)

|  |  |  |  |
| --- | --- | --- | --- |
| Размер | Bubble sort ключи | Quick sort таблица | Quick sort ключи |
| 500 | 105% | 3600% | 3600% |
| 1000 | 107.2% | 4666.7% | 14200.0% |
| 10000 | 109.3% | 21118.6% | 61787.5% |

Дополнительные затраты по памяти, в каждом случае, составляли 21.6%

**Вывод:**

При выполнении над большими таблицами с данными операций, например, таких, как сортировка, эффективнее использовать сортировку не всей таблицы, а только таблицы ключей. Это позволяет уменьшить время сортировки ~ в 2 раза при сортировке пузырьком и в зависимости от размера (от 30 до 600 раз) при сортировке quick sort. Достигается это за счёт обмена во время сортировки элементов меньшего размера. Но таблица ключей приводит к дополнительным затратам памяти: ~21% дополнительно к размеру структуры.

**Ответы на контрольные вопросы**

1. Как выделяется память под вариантную часть записи?

Память под вариантную часть выделяется через объединение (union). В структуре subscriber\_t используется объединение subs\_status\_t, которое содержит два поля: colleague\_t и friend\_t. Размер памяти выделяется равным размеру наибольшего элемента объединения. В данном случае:

colleague\_t занимает: 25 (job) + 25 (org) = 50 байт

friend\_t занимает: размер структуры date\_t (4 int × 4 байта = 16 байт)

Таким образом, под вариантную часть выделяется 50 байт - размер наибольшего элемента (colleague\_t).

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

Если ввести данные, не соответствующие текущему статусу записи, это приведет к интерпретации неправильных данных, или же ошибке (если тип сканируемого поля не соответствует вводимому, например, при вводе организации (у коллеги) в поле даты (у друга))

3. Кто должен следить за правильностью выполнения операций с вариантной частью записи? За правильностью операций с вариантной частью должны следить программист (при разработке функций ввода/вывода должен обеспечивать проверку соответствия статуса и вводимых данных) и функции валидации (в программе должны быть реализованы проверки перед записью в вариантную часть)

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

Таблица ключей представляет собой массив пар индекс (в исходной таблице)-ключ. Она позволяет ускорить операции, которые требуют перемещение элементов таблицы (например сортировку)

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

Использовать таблицу ключей выгоднее и эффективнее тогда, когда размер её элементов значительно меньше размера структуры (таблицы).

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

Для обработки таблиц предпочтительнее использовать быстрые сортировки, так как это позволяет ускорить сортировку. Если размер ключа значительно меньше размера всей записи, то имеет смысл сортировать таблицу ключей, а не саму таблицу.