Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное 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 СТРУКТУРЫ ДАННЫХ»

Записи с вариантами, обработка таблиц

Вариант № 5

Студент: Бондарева **В. А.**

Группа: ИУ**7-34Б**

Преподаватель: **Никульшина Т. А.**

2025 г.

Условие задачи

Создать таблицу, содержащую не менее сорока записей о театральных постановках, реализовать два алгоритма сортировки, один из который предполагает сортировку массива структур, а второй – сортировку массива ключей, где ключ – любое невариантное поле. Реализовать добавление и удаление записей из таблицы. Реализовать вывод списка всех балетов для детей указанного возраста, продолжительностью меньше указанной. Реализовать вывод таблицы значений (массива структур) в упорядоченном виде.

Описание ТЗ

Исходные данные

Исходные данные – информация о театральных представлениях. Исходные данные вводятся пользователем с клавиатуры или читаются из указанного файла в зависимости от выбранной пользователем опции меню. Каждая запись представлена следующей структурой:

1. Пьеса
    a. Драма
    b. Комедия
    c. Сказка
        i. Возраст: 3+, 10+, 16+
2. Мюзикл
    a. Композитор
    b. Страна
    c. Вид: балет, опера, мюзикл
    d. Возраст: 3+, 10+, 16+
    e. Продолжительность

Таблица 1. Структура театрального представления.

На исходные данные (вводимые с клавиатуры или считываемые из файла) накладывается ряд ограничений:

1. Формат ввода данных: данные должны вводиться в определенном установленным и указанным программистом формате.
2. Формат чтения данных: данные должны содержаться в файлах в определенном установленным и указанным программистом формате.
3. Вариативность полей структуры: такие поля как тип постановки, значение возрастного ограничения, вид жанра пьесы/мюзикла являются строго вариативными.

Результат

Результатом выполнения программы являются:

1. Таблица записей.
2. Таблица ключей.
3. Таблица записей, выведенная в соответствии с таблицей ключей
4. Выборка определенного списка записей, а именно всех балетов, для детей указанного возраста, продолжительностью меньше указанной.
5. Таблица сравнения эффективности для двух методов сортировки .

Способ обращения к программе

Пользователь обращается к программе при помощи исполняемого файла app.exe.

Возможные аварийные ситуации и ошибки пользователя

Все возможные аварийные ситуации и ошибки пользователя описаны типом status_t, включающим в себя коды возврата программы:

1. Неверно указано количество чего-либо
2. Ошибка при работе с указателем
3. Ошибка ввода (некорректный и/или не валидный ввод)
4. Ошибка выделения памяти
5. Ошибка ввода вариативного поля
6. Ошибка работы с файлом, связанная с невозможностью его открыть
7. Ошибка чтения файла
8. Ошибка обработки массива
9. Переполнение таблицы с театральными постановками
10. Ошибка удаления театральной постановки
11. Ошибка сортировки таблицы

Описание внутренних структур данных

Для реализации необходимого по условию работы функционала программа использует следующую структуру данных:

```
typedef struct
{
    char theater_name[MAX_STR_LEN];
    char play_name[MAX_STR_LEN];
    double ticket_price;
    double max_ticket_price;
    play_type_t play_type;
    age_rating_t age_rating;
    union
    {
        struct
        {
            piece_genre_t piece_genre;
```

```
        } piece_info;
        struct
        {
            char composer[MAX_STR_LEN];
            char country[MAX_STR_LEN];
            musical_genre_t musical_genre;
            int duration;
        } musical_info;
    } play_data;
} theater_play_t;
```

Листинг 1. Структура театральной постановки.

- theater_name[MAX_STR_LEN]: строка (максимальная длина MAX_STR _LEN), содержащая название театра

- play_name[MAX_STR_LEN]: строка (максимальная длина MAX_STR_LEN), содержащая название постановки

- ticket_price: ЧПТ, минимальная цена билета

- max_ticket_price: ЧПТ, максимальная цена билета

- play_type: вариантное поле, тип театральной постановки (пьеса или мюзикл)

- age_rating: вариантное поле, возрастной рейтинг (3+, 10+, 16+)

- piece_genre: вариантное поле, жанр пьесы (драма, комедия, сказка)

- composer[MAX_STR_LEN]: строка (максимальная длина MAX_STR_LEN), содержащая имя композитора

- country[MAX_STR_LEN]: строка (максимальная длина MAX_STR_LEN), содержащая страну происхождения

- musical_genre: вариантное поле, тип мюзикла (балет, опера, музыкальное шоу)

- duration: целое число, продолжительность постановки в минутах

Алгоритм

```
// сортировка
status_t quick_sort_by_ticket_price(theater_play_t *theater_plays_arr, int
left, int right);
status_t slow_sort_by_ticket_price(theater_play_t *theater_plays_arr, size_t
*theater_plays_q);
status_t quick_sort_by_keys(theater_play_t *theater_plays_arr, int
*theater_plays_keys, int left, int right);
status_t slow_sort_by_keys(theater_play_t *theater_plays_arr, int
*theater_plays_keys, size_t *theater_plays_q);

// чтение файла
status_t read_file(theater_play_t *theater_plays_arr, int
*theater_plays_keys, size_t *theater_plays_q);

// работа с таблицей записей
```

```
status_t allocate_memory(void **elems_arr, size_t elems_quantity, size_t
elem_size);
status_t process_choice(choice_t choice, bool *program_running,
theater_play_t *theater_plays_arr, int *theater_plays_keys, size_t
*theater_plays_q);
status_t clean_arr(theater_play_t *theater_plays_arr, size_t
*theater_plays_q);
status_t add_play(theater_play_t *theater_plays_arr, int
*theater_plays_keys, size_t *theater_plays_q);
status_t input_string_to_delete(char *target_string);
status_t shift_array_with_delete_one_string_elem(theater_play_t
*theater_plays_arr, int *theater_plays_keys, size_t *theater_plays_q, size_t
*current_pos, char *string_to_delete, char *source_string_to_compare);
status_t shift_array_with_delete_one_integer_elem(theater_play_t
*theater_plays_arr, int *theater_plays_keys, size_t *theater_plays_q, size_t
*current_pos, int int_to_delete, int source_int_to_delete);
status_t delete_play(theater_play_t *theater_plays_arr, int
*theater_plays_keys, size_t *theater_plays_q);
status_t rebuild_keys_table(int *theater_plays_keys, size_t
*theater_plays_q);

// печать меню, инструкций и предупреждений
status_t print_menu(void);
status_t print_read_file_comments(void);
status_t print_delete_instructions(void);

// вывод таблиц
status_t print_theater_plays_table(theater_play_t *theater_plays_arr, int
*theater_plays_keys, size_t theater_plays_q, bool print_by_keys);
status_t print_keys_table(theater_play_t *theater_plays_arr, int
*theater_plays_keys, size_t theater_plays_q);
status_t print_balets_by_conditions(theater_play_t *theater_plays_arr,
size_t theater_plays_q, age_rating_t target_age_rating, int
target_duration);
status_t print_efficiency_table(long long average_quick_sort_data_time, long
long average_slow_sort_data_time, long long average_quick_sort_key_time,
long long average_slow_sort_key_time);
status_t print_memory_data(size_t memory_used, size_t memory_used_keys);
status_t print_status_message(status_t rc);
```

Таблица 2. Сигнатуры некоторых функций

Помимо основного алгоритма программы (main.c), основополагающими алгоритмами для реализации заданного функционала являются:

- добавление записи о театральной постановке в таблицу

- удаление из таблицы запись о театральной постановке

- разные алгоритмы сортировки (по ключам и обычная, быстрая и медленная)

- алгоритм для измерения эффективности разных видов сортировки

Рассмотрим каждый алгоритм.

```
#include "status.h"
#include "data.h"
#include "color.h"
#include "input.h"
#include "process.h"
```

```c
#include "print_instructions.h"
#include <stdbool.h>

static theater_play_t *theater_plays_arr = NULL;
static int *theater_plays_keys = NULL;
static size_t theater_plays_q = 0;

int main(void)
{
    status_t exit_code = SUCCCESS_CODE;
    bool program_running = false;
    choice_t current_choice;

    exit_code = allocate_memory((void **)&theater_plays_arr,
MAX_PLAYS_QUANTITY, sizeof(theater_play_t));
    if (exit_code == SUCCCESS_CODE)
        exit_code = allocate_memory((void **)&theater_plays_keys,
MAX_PLAYS_QUANTITY, sizeof(int));

    if (exit_code == SUCCCESS_CODE)
        program_running = true;

    while (program_running && exit_code == SUCCCESS_CODE)
    {
        exit_code = print_menu();
        exit_code = input_choice(&current_choice);
        if (exit_code == SUCCCESS_CODE)
            process_choice(current_choice, &program_running,
theater_plays_arr, theater_plays_keys, &theater_plays_q);
    }

    // освобождаем выделенную память
    if (theater_plays_arr)
        free(theater_plays_arr);

    return exit_code;
}
```

Листинг 2. Основной алгоритм программы.

Псевдокод основного алгоритма:

1. Выделяется память для массива структур театральных постановок (размер MAX_PLAYS_QUANTITY).
2. Дальнейшие действия выполняются циклически, пока корректен ввод опций меню и пользователь не вышел из программы сам.
3. Печатается меню с возможными действиями пользователя и приглашение к вводу.
4. В зависимости от указанной опции программа выполняет конкретные действия и обработку.
5. Печатается результат выполнения указанного действия.
6. После завершения цикла, если массив структур театральных постановок был инициализирован, он освобождается.
7. Программа завершается, возвращая код статуса своего выполнения.

Листинг 3. Псевдокод основного алгоритма программы.

1. Проверка валидности переданных в функцию аргументов.

2. Ввод названия театра.
3. Ввод названия спектакля.
4. Ввод минимальной и максимальной цен билета с соответствующей проверкой диапазона значений.
5. Ввод типа спектакля.
6. Ввод остальных вариантных полей постановки в зависимости от выбранного типа спектакля.
7. Если в процессе ввода данных не возникла ошибка, добавляет новую запись.

Листинг 4. Псевдокод функции добавления записи о театральной постановке.

1. Проверка валидности переданных в функцию аргументов.
2. Печать инструкций по удалению записи из таблицы.
3. Ввод поля, по которому будет происходить удаление, и его значения
4. Поиск записей, соответствующих пользовательскому вводу.
5. Удаление найденных записей (при наличии).

Листинг 5. Псевдокод функции удаления записи о театральной постановке.

1. Если левая граница меньше правой:
    a. Инициализирует индексы i и j на левой и правой границах.
    b. Задает pivot как опорный элемент (цена билета).
    c. Пока i не превышает j выполняет:
        i. Поиск элемента слева, который больше или равен pivot.
        ii. Поиск элемента справа, который меньше или равен pivot.
        iii. Если такие элементы найдены, меняет их местами.
    d. Рекурсивно применяет функцию для левой и правой частей массива.

Листинг 6. Псевдокод быстрой сортировки массива структур.

1. Пока массив не будет отсортирован:
    a. Для каждой последовательной пары записей сравнивается цена билета.
    b. Если цена билета первой записи больше цены билета второй:
        i. Меняет записи местами

Листинг 7. Псевдокод медленной сортировки массива структур.

1. На основе значения ключа находит соответствующую ему постановку.
2. Если левая граница меньше правой:
    a. Инициализирует индексы i и j на левой и правой границах.
    b. Задает pivot как опорный элемент (цена билета).
    c. Пока i не превышает j выполняет:
        i. Поиск элемента слева, который больше или равен pivot.
        ii. Поиск элемента справа, который меньше или равен pivot.
        iii. Если такие элементы найдены, меняет ключи местами.
    d. Рекурсивно применяет функцию для левой и правой частей массива.

Листинг 8. Псевдокод быстрой сортировки по ключу.

1. На основе значения ключа находит соответствующую ему постановку.
2. Пока массив не будет отсортирован:
    a. Для каждой последовательной пары записей сравнивается цена билета.
    b. Если цена билета первой записи больше цены билета второй:
        i. Меняет ключи местами

Листинг 9. Псевдокод медленной сортировки по ключу.

1. Открывается указанный файл БД, содержащий N записей
2. 500 раз повторяется цикл:
    a. Для каждого типа сортировки:
        i. Файл считывается заново.
        ii. Производится замер для конкретного типа сортировки
3. Считается среднее арифметическое для каждого значения.
4. Файл БД закрывается.

Листинг 10. Псевдокод алгоритма сравнения разных видов сортировок на заданном массиве структур.

Тесты

Позитивные тесты

Номер теста	Описание	Ожидаемый результат
1	Корректное добавление записи.	Успешное добавление записи.
2	Корректное удаление записи.	Успешное удаление записи.
3	Чтение данных из файла.	Успешное чтение данных из файла.
4	Сортировка таблицы значений.	Успешная сортировка таблицы значений.
5	Сортировка таблицы ключей.	Успешная сортировка таблицы ключей.
6	Вывод данных из таблицы.	Успешный вывод таблицы значений.
7	Вывод данных по таблице ключей.	Успешный вывод таблицы значений по таблице ключей.

Таблица 3. Позитивные тесты.

Негативные тесты

Номер теста	Описание	Ожидаемый результат
1	Попытка добавления не валидной записи.	Возврат ошибки и вывод соответствующего сообщения об ошибке.
2	Попытка удаления записи, некорректные входные данные, например, буквы в продолжитель-ности.	Возврат ошибки и вывод соответствующего сообщения об ошибке.
3	Попытка чтения из несуществующего файла.	Возврат ошибки и вывод соответствующего сообщения об ошибке.

4	Попытка выбора несуществующей опции.	Возврат ошибки и вывод соответствующего сообщения об ошибке.

Таблица 4. Негативные тесты.

Оценка эффективности

Преимущества использования типа «запись» с вариантной частью

1. Упрощенный доступ к данным. Нет необходимости вычислять смещение.
2. Компактная структура данных.

Недостатки использования типа «запись» с вариантной частью

1. Сложность и затратность сортировки. Операция «перестановки двух структур в памяти» является затратной.
2. Сложность удаления и добавления новых записей.

Преимущества использования массива ключей для массива «записей»

1. Быстрый доступ. Массив ключей позволяет быстрее получить доступ к конкретной записи по заданному полю.
2. Оптимизация сортировки. Массив ключей позволяет эффективнее сортировать данные по значению какого-либо поля, т.к. позволяет избежать затратной операции «перестановки двух структур в памяти».
3. Оптимизация удаления и добавления новых записей.

Недостатки использования массива ключей для массива «записей»

1. Необходимость дополнительных затрат на память. Хранение массива ключей предполагает выделение памяти под него.

Сравнение времени сортировки и затрачиваемой памяти для 100 записей

Сортировка	Наносекунды
Быстрая сортировка	11987
Медленная сортировка	95894
Быстрая сортировка по ключу	884
Медленная сортировка по ключу	6260

Таблица 5. Эффективность для 100 элементов.

Использование оперативной памяти:
Размер данных: 54400 байт
Размер таблицы ключей: 400 байт
Общий объем используемой памяти: 54800 байт

Сравнение времени сортировки и затрачиваемой памяти для 500 записей

Сортировка	Наносекунды
Быстрая сортировка	85147
Медленная сортировка	2406587
Быстрая сортировка по ключу	6034
Медленная сортировка по ключу	149224

Таблица 6. Эффективность для 500 элементов.

Использование оперативной памяти:
Размер данных: 272000 байт
Размер таблицы ключей: 2000 байт
Общий объем используемой памяти: 274000 байт

Сравнение времени сортировки и затрачиваемой памяти для 1000 записей

Сортировка	Наносекунды
Быстрая сортировка	198277
Медленная сортировка	9914933
Быстрая сортировка по ключу	13463
Медленная сортировка по ключу	608812

Таблица 7. Эффективность для 1000 элементов.

Использование оперативной памяти:
Размер данных: 544000 байт
Размер таблицы ключей: 4000 байт
Общий объем используемой памяти: 548000 байт

Анализ полученных значений

Измерение времени продолжительности разных видов сортировок для разных способов обращения к данным показало, что:

- алгоритм быстрой сортировки является наиболее эффективным алгоритмом сортировки по сравнению с алгоритмом медленной сортировки
- таблица ключей занимает сравнительно небольшой объем памяти относительно массива структур (примерно 0.7% от объема массива структур)
- быстрая сортировка массива ключей в среднем выполняется в 14 раз быстрее, чем быстрая сортировка массива структур

Вывод

Сортировка и обработка массива «записей» выполняется эффективнее при использовании дополнительного массива ключей. Относительная эффективность использования того или иного метода работы с массивом записей определяется объемом этих данных. Чем больше объем обрабатываемых данных, тем эффективнее

будет использование дополнительного массива ключей по сравнению с обычной сортировкой массива значений. Вариантная часть записи эффективна в ситуациях, когда требуется описать объекты с различными наборами характеристик, но создание единой универсальной структуры нерационально. Использование вариантной части записи является гибким и эффективным решением для определенных случаев, однако требует особого внимания к контролю работы с данными.

Контрольные вопросы

1. *Как выделяется память под вариантную часть записи?*
   Память под вариантную часть записи выделяется с помощью объединения (union). В структуре театральной постановки theater_play_t используется объединение play_data, которое содержит поля musical_info и piece_info. Память под вариантную часть записи выделяется в соответствии с размером памяти, занимаемой бóльшим элементом объединения.
2. *Что будет, если в вариантную часть ввести данные, несоответствующие описанным?*
   Если в вариантную часть ввести данные, не соответствующие описанным, это приведет к неопределенному поведению или некорректной работе программы. Компилятор не проверяет соответствие типов данных в вариантной части во время выполнения, так как эта ответственность лежит на программисте. В этом случае программа может аварийно завершиться или повредить данные.
3. *Кто должен следить за правильностью выполнения операций с вариантной частью записи?*
   Ответственность за правильность выполнения операций с вариантной частью лежит на программисте.
4. *Что представляет собой таблица ключей? Зачем она нужна?*
   Таблица ключей – вспомогательная таблица, представленная в виде массива записей, где каждая запись содержит одно или несколько ключевых полей. Ключевое поле используется для идентификации и поиска элементов таблицы по их значению. Такая таблица облегчает быструю и эффективную обработку и поиск данных.
5. *В каком случае эффективнее обрабатывать данные в самой таблице, а когда – использовать таблицу ключей?*
   Если таблица небольшая или операции обработки не являются частыми, эффективнее работать с основной таблицей напрямую. При больших таблицах, над которыми часто совершаются операции сортировки или поиска, выгоднее использовать таблицу ключей для снижения временных затрат на перестановку данных.

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

   Предпочтительнее использовать алгоритмы сортировки с более низкой временной сложностью и высокой эффективностью, например, быструю сортировку, поскольку они снижают время обработки больших таблиц по сравнению с простыми алгоритмами.