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 **ПРОГРАММНАЯ ИНЖЕНЕРИЯ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04** Программная инженерия

О Т Ч Е Т

По лабораторной работе № 2

Название: Запись с вариантами, обработка таблиц

Дисциплина: Типы и структуры данных

Студент ИУ7-32Б (Группа) С.С. Беляк (И.О. Фамилия)

Преподаватель Барышникова М.Ю.

Москва, 2023

Описание условия задачи

Создать таблицу, содержащую не менее 40-ка записей. Упорядочить данные в ней по возрастанию ключей, двумя алгоритмами сортировки, где ключ – любое невариантное поле (по выбору программиста), используя: а) саму таблицу, б) массив ключей. (Возможность добавления и удаления записей в ручном режиме обязательна). Осуществить поиск информации по варианту. Ввести репертуар театров, содержащий: название театра, спектакль, диапазон цены билета, тип спектакля (1: пьеса, 2: драма, 3: комедия, 4: сказка - возраст (3+, 10+,16+), 5: музыкальный – композитор, страна, тип (балет, опера, мюзикл), возраст (3+, 10+,16+), продолжительность). Вывести список всех балетов для детей указанного возраста с продолжительностью меньше указанной.

Описание ТЗ

1. Описание исходных данных

Исходные данные представляют собой информацию о театральных представлениях. Каждая запись содержит следующие атрибуты:
Название театра
Название представления
Цена билета
Максимальная цена билета
Тип представления
Возрастная категория
Композитор
Страна производства
Тип исполнения
Продолжительность представления

Имеются следующие ограничения:

1. Ограничение по формату ввода данных: Программа может требовать, чтобы данные вводились в определенном формате (например, числа для цены билета, определенные значения для типа спектакля и возрастных категорий).
2. Ограничение по типам спектаклей: Программа может поддерживать только определенные типы спектаклей (пьеса, драма, комедия, сказка, музыкальный), и новые типы могут не добавляться без изменения кода программы.
3. Ограничение по возрастным категориям: Программа может ограничиваться предопределенными возрастными категориями (3+, 10+, 16+), и введенные категории должны соответствовать этим ограничениям.
4. Ограничение по продолжительности: Программа может требовать, чтобы продолжительность спектаклей была представлена в определенном формате (например, часы и минуты), и новые форматы могут не поддерживаться.
5. Ограничение по выводу данных: Программа может выводить список балетов для детей только для указанных возраста и продолжительности, что ограничивает результаты поиска.

Таблица примеров ввода:

Допустимый ввод	**Недопустимый ввод**
Введите цену билета: 500	Введите цену билета: -50
Введите тип спектакля (1: пьеса, 2: драма, 3: комедия, 4: сказка):1	Введите тип спектакля (1: пьеса, 2: драма, 3: комедия, 4: сказка): 6
Введите возрастной диапазон (например, 3+, 10+): 16+	Введите возрастной диапазон (например, 3+, 10+): 20+
Введите имя композитора: Чайковский	Введите имя композитора: Шос5такович
Введите страну: Россия	Введите страну: Россия9
Введите тип спектакля (1: балет, 2: опера, 3: мюзикл): 1	Введите тип спектакля (1: балет, 2: опера, 3: мюзикл): 4
Введите продолжительность: 2.30	Введите продолжительность: 2 часа 30 минута

2. Описание результатов программы

1. Ввод данных: Пользователь может ввести данные о театральных представлениях, указывая все вышеперечисленные атрибуты.
2. Вывод данных: Пользователь может выбрать пункт меню для просмотра данных, которые были введены.
3. Вывод таблицы ключей: Программа предоставляет возможность вывода таблицы ключей, основанной на цене билетов.
4. Добавление записи: Пользователь может добавлять новые записи о театральных представлениях.
5. Удаление записи: Программа позволяет удалять записи о театральных представлениях.
6. Сортировка по цене: предоставляются два варианта сортировки данных по цене билетов - быстрая и медленная.
7. Сортировка таблицы ключей: Программа также предоставляет возможность сортировки таблицы ключей.
8. Вывод данных по таблице ключей: Пользователь может просматривать данные, основанные на таблице ключей.
9. Вывод таблицы эффективности: Программа выводит таблицу, отображающую время выполнения различных операций.
10. Вывод списка балетов для детей: По запросу пользователь может получить список балетов, соответствующих заданным возрастному диапазону и продолжительности.

3. Описание задачи, реализуемой в программе

Цель работы программы заключается в создании инструмента, который позволяет управлять информацией о различных театральных представлениях. Это включает в себя ввод, хранение, обработку и вывод данных, связанных с театральными постановками

4. Способ обращения к программе

Способ обращения к программе пользователем происходит через исполняемый файл app.exe.

5. Описание возможных аварийных ситуаций и ошибок пользователя

1. **ERR_OK**: Этот код обозначает успешное выполнение операции. Используется при успешном завершении программы.
2. **EMPTY_FILE_ERROR**: Ошибка, которая возникает, когда нет данных для вывода или обработки. Например, при попытке вывода данных или таблицы ключей, когда список пуст.
3. **OPEN_FILE_ERROR**: Ошибка, которая возникает при попытке открыть файл и файл не существует или не может быть открыт по какой-либо причине.
4. **OVERFLOW_ERROR**: Ошибка, которая возникает, когда достигнуто максимальное количество записей или операция приводит к переполнению.
5. **ERR_RANGE**: Этот код ошибки используется, когда данные введены в недопустимом диапазоне или не соответствуют ожидаемому формату. Например, при вводе цены билета, типа спектакля или возрастного диапазона в неправильном формате.
6. **CHOICE_ERROR**: Ошибка, которая возникает, когда пользователь сделал неверный выбор в меню (например, ввел число, не соответствующее ни одному из вариантов).

6.Описание внутренних структур данных

Программа содержит в себе структуру, которая используется для хранения информации о театральных представлениях.

```
typedef struct {
    char theater_name[50];
    char play[50];
    float ticket_price;
    float max_price;
    int play_type;
    union {
        struct {
            char age_range[5];
        } fairy_tale;  // Для сказок
        struct {
            char age_range[5];
            char composer[50];
            char country[50];
            int performance_type;
            char duration[10];
        } musical;  // Для музыкальных
спектаклей
    } details;
} TheaterPlay;
```

theater_name[50]: Массив символов, предназначенный для хранения названия театра..
play[50]: Поле для хранения названия спектакля, максимальная длина названия спектакля составляет 50 символов.
ticket_price: Поле с плавающей точкой, предназначенное для хранения цены билета на спектакль.
play_type: Целочисленное поле, которое указывает тип спектакля (1: пьеса, 2: драма, 3: комедия, 4: сказка, 5: музыкальный).
age_range[5]: Поле для хранения возрастного диапазона, к которому относится спектакль. Максимальная длина диапазона составляет 5 символов.
composer[50]: Поле, где можно указать имя композитора спектакля, максимальная длина составляет 50 символов.
country[50]: Поле для указания страны, в которой был создан спектакль. Максимальная длина названия страны составляет 50 символов.
performance_type: Целочисленное поле, которое указывает тип спектакля (1: балет, 2: опера, 3: мюзикл).
duration[10]: Поле для хранения продолжительности спектакля. Максимальная длина продолжительности составляет 10 символов. глобальные переменные:

7. Алгоритм программы

Общий алгоритм программы:

1. Инициализация системы и выделение памяти.
2. Проверка выделения памяти.
3. Основной цикл программы начинается. Пользователю предоставляется меню с различными опциями, и программа ожидает ввода выбора пользователя.
4. В зависимости от выбора пользователя, программа выполняет следующие действия:
    - Выйти из программы
    - Прочитать данные из файла и заполнить массив
    - Вывести данные о театральных постановках
    - Вывести таблицу ключей
    - Добавить новую запись в массив
    - Удалить существующую запись из массива
    - Отсортировать данные по цене с использованием быстрой сортировки
    - Отсортировать данные по цене с использованием медленной сортировки по цене с использованием быстрой сортировки
    - Отсортировать таблицу ключей по цене с использованием медленной сортировки
    - Вывести данные о театральных постановках, используя таблицу ключей
    - Измерить время выполнения различных сортировок для данных и таблицы ключей, а также оценить использование оперативной памяти
    - Вывести список балетов для детей с заданными параметрами
5. После выполнения каждой операции программа возвращает пользователя в главное меню, где он может выбрать следующее действие.
6. При завершении программы освобождается выделенная память и программа завершает свою работу.

Функция containsNumbers:

1. Перебирает каждый символ в строке.
2. Если найдена цифра, возвращает **1**, иначе **0**.

Функция addRecord:

1. Проверяет доступное место.
2. Позволяет пользователю ввести данные.
3. Проверяет введенные значения.
4. Если проверка успешна, добавляет новую запись.

Функция deleteRecord:

1. Пользователь выбирает поле и вводит значение.
2. Поиск записей, соответствующих выбранному полю и значению.
3. Если записи найдены, удаляет их.
4. Выводит результат удаления.
5. Если записи не найдены, сообщает об этом.

Функция quickSortByTicketPrice:

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

Функция slowSortByTicketPrice:

1. Для каждой пары записей сравнивает цену билета.
2. Если цена билета в первой записи больше цены билета во второй, меняет их местами.
3. Повторяет это для всех пар записей.
4. Повторяет сортировку до тех пор, пока массив не будет отсортирован.

Функция compareTheaterByTicketPrice:

1. Получает два указателя на индексы записей.
2. Получает цены билетов для соответствующих записей.
3. Сравнивает цены:
   - Если цена первой записи меньше второй, возвращает **-1**.
   - Если цена первой записи больше второй, возвращает **1**.
   - В противном случае, возвращает **0**.

Функция sortKeysByTicketPrice:

1. Инициализирует таблицу ключей значениями от 0 до **num_records - 1**.
2. Использует функцию **qsort** для сортировки этой таблицы ключей на основе цен билетов.

Функция quickSortByKey:

1. Если left меньше right, выполняет следующее:
2. Устанавливает i в значение left и j в значение right.
3. Задает pivot как цену билетов для элемента theater_plays[left].
4. Внутри цикла:
5. Пока theater_plays[i].ticket_price меньше pivot, выполняет увеличение i.
6. Пока theater_plays[j].ticket_price больше pivot, выполняет уменьшение j.
7. Если i меньше или равно j, выполняет обмен местами элементов theater_plays[i] и theater_plays[j], затем выполняет увеличение i и уменьшение j.
8. Рекурсивно выполняет quickSortByKey для левой и правой частей массива, если left меньше j и i меньше right

Функция slowSortByKey:

Для каждой пары индексов в таблице ключей:

1. Выполняет сравнение цен билетов для двух элементов.
2. Если цена билета в первом элементе больше, чем во втором, выполняет обмен местами этими двумя элементами.

Продолжает этот процесс, пока таблица ключей не будет отсортирована по цене билетов.

Функция printBalletsForChildrenWithDuration:

1. Выводит заголовок таблицы с описанием полей.
2. Для каждой записи в массиве выполняет следующее:
   Проверяет, соответствует ли запись критериям: это балет для детей указанного возраста и его продолжительность меньше указанной.
3. Если критерии соответствуют, выводит данные о спектакле в таблицу, включая название театра, название спектакля, цену билета и другие характеристики.

Функция printTable:

1. Выводит заголовок таблицы с описанием полей.
2. Для каждой записи в массиве выводит данные о спектакле в таблицу, включая название театра, название спектакля, цену билета и другие характеристики

Вывод по алгоритму работающей программы:

Программа представляет собой систему управления информацией о театральных спектаклях. Она обеспечивает возможность добавления, удаления, сортировки и вывода данных о спектаклях. Пользователь может вводить информацию о спектаклях, выполнять поиск и удаление записей по заданным критериям, а также измерять производительность различных алгоритмов сортировки. Программа предоставляет удобный интерфейс для работы с данными о спектаклях и позволяет оценить эффективность сортировки.

Тесты

Таблица положительных тестов.

Номер теста	Описание	Ожидаемый результат
1	Добавление записи и вывод.	Успешное добавление новой записи и вывод на экран.
2	Добавление нескольких записей и вывод.	Успешное добавление нескольких записей и вывод на экран.
3	Чтение данных из файла и вывод.	Успешное считывание данных из файла и вывод на экран.
4	Сортировка и вывод.	Успешная сортировка данных и вывод в правильном порядке.
5	Удаление записи и вывод.	Успешное удаление выбранной записи и вывод обновленной таблицы.
6	Сортировка таблицы ключей и вывод.	Успешная сортировка таблицы ключей и вывод в правильном порядке.
7	Вывод данных по таблице ключей.	Успешный вывод данных о театральных представлениях по таблице ключей.

Таблица негативных тестов.

Номер теста	Описание	Ожидаемый результат
1	Попытка добавления записи с недопустимой ценой билета (отрицательной).	Возврат ошибки ERR_RANGE.
2	Попытка добавления записи с недопустимым типом спектакля (не в диапазоне 1-4).	Возврат ошибки ERR_RANGE.
3	Попытка добавления записи с недопустимым возрастным диапазоном (не "3+", "10+" или "16+").	Возврат ошибки ERR_RANGE.
4	Попытка добавления записи с именем композитора, содержащим числа.	Возврат ошибки ERR_RANGE.
5	Попытка добавления записи с названием страны, содержащим числа.	Возврат ошибки ERR_RANGE.
6	Попытка добавления записи с недопустимым типом спектакля (не в диапазоне 1-3).	Возврат ошибки ERR_RANGE.
7	Попытка добавления записи с недопустимой продолжительностью (неверный формат).	Возврат ошибки ERR_RANGE.
8	Попытка добавления записи при достижении максимального количества записей.	Возврат ошибки OVERFLOW_ERROR.
9	Попытка удаления записи с недопустимым номером (отрицательным).	Возврат ошибки ERR_RANGE.
10	Попытка чтения данных из несуществующего файла.	Возврат ошибки OPEN_FILE_ERROR.

Оценка эффективности

Преимущества использования типа запись с вариативной частью:

1. Структура данных более компактна, так как записи хранятся в одном массиве. Это позволяет экономить оперативную память.
2. Упрощение доступа к данным: для получения доступа к полям записи, не нужно вычислять смещение, как в случае с дополнительными массивами ключей.
3. Записи позволяют легко организовать хранение связанных данных в одной структуре, что может облегчить работу с данными.

Недостатки использования типа "запись" с вариативной частью:

1. Сложность сортировки: сортировка данных внутри записи может быть затруднительной, особенно если данные имеют разные типы и размеры.
2. Удаление и вставка: операции удаления и вставки данных внутри записи могут потребовать дополнительных затрат по времени и ресурсам.

Преимущества использования дополнительного массива ключей:

1. Упрощение сортировки: дополнительный массив ключей позволяет сортировать данные по одному полю (например, по цене билета) без изменения исходных данных.
2. Быстрый доступ: массив ключей обеспечивает быстрый доступ к записям по заданному критерию (например, по цене билета).

Недостатки использования дополнительного массива ключей:

1. Затраты на память: дополнительный массив ключей занимает дополнительное место в памяти.
2. Сложность доступа к данным: для получения данных по ключу требуется двойной доступ к памяти (по ключу, затем к записи).
3. Увеличение сложности кода: требуется дополнительный код для управления массивом ключей и поддержания их актуальности после изменений в данных.
4. Дополнительная нагрузка на процессор: операции с дополнительным массивом ключей могут потреблять процессорное время.

Вывод: Использование записи с вариативной частью имеет свои преимущества, такие как компактность и упрощенный доступ к данным. Однако при работе с большими объемами данных и необходимости выполнения сложных операций сортировки и фильтрации, использование дополнительного массива ключей может оказаться более эффективным.

Измерения времени выполнения сортировок для данных и таблицы ключей показывают следующее:
Быстрая сортировка данных занимает меньше всего времени среди всех алгоритмов сортировки, что делает ее наиболее эффективным методом для сортировки данных в этой программе.
Медленная сортировка данных требует больше времени, чем быстрая сортировка данных, но все равно остается довольно быстрой.
Быстрая сортировка ключей также является относительно быстрым методом, хотя и немного медленнее, чем быстрая сортировка данных.
Медленная сортировка ключей требует больше времени, чем быстрая сортировка ключей, но все равно является относительно быстрым методом.
Сравнивая быстрые и медленные методы для данных и таблицы ключей, видно, что быстрые методы выполняются быстрее, что делает их предпочтительными. Время выполнения всех алгоритмов остается небольшим, что свидетельствует о хорошей производительности программы.
В отношении использования оперативной памяти, данные и таблица ключей занимают небольшой объем памяти.
Таким образом, программа эффективно работает с данными и таблицей ключей, и не требует больших объемов памяти. Быстрые методы сортировки оказываются предпочтительными благодаря их быстрому времени выполнения.

При запуске программы N= 500 раз, были получены следующие данные:

Размер массива	**Быстрая сортировка данных, нс.**	**Быстрая сортировка ключей,нс**	**Медленная сортировка данных,нс**	**Медленная сортировка ключей,нс**	**Размер таблицы ключей, байт**	**Размер данных байт**	**Общий объём памяти байт**
50	3962	2506	3547	3476	164	9512	9676
100	12073	8508	16195	16215	400	23200	23600
200	49494	33962	101693	101667	800	46400	47200
500	216288	146544	917620	915828	2000	116000	118000
1000	228829	153476	1614324	1660185	4000	232000	236000

Находя в процентах отношение по формуле (t1-t2) \ t2 *100%, найдем эффективность:

Размер массива	**Быстрая сортировка, %**	**Медленная сортировка, %**
50	58,1	2.0%
100	41,9	-0,1%
200	45,7	0,02%
500	47,5	0,19%

Вывод

В ходе выполнения лабораторной работы была разработана программа для управления данными о театральных постановках. Программа предоставляет пользователю возможность считывать данные из файла, добавлять новые записи, удалять существующие, а также проводить сортировку данных и анализировать их эффективность .Были реализованы основные функции, такие как считывание и вывод данных, добавление и удаление записей, а также сортировка данных с использованием различных алгоритмов. Программа также поддерживает работу с таблицей ключей, что позволяет ускорить поиск и сортировку данных. В процессе тестирования программы было выявлено, что она успешно справляется с чтением, записью и обработкой данных. Различные сортировки показали разную эффективность, что подтверждено числовыми данными, полученными в результате измерения времени выполнения операций. Использование таблицы ключей оказалось эффективным способом ускорения сортировки и поиска данных. Оценка использования оперативной памяти показала, что программа расходует память в соответствии с размером данных и таблицей ключей, что позволяет контролировать объем используемой памяти. Таким образом, разработанная программа успешно выполняет поставленные задачи по управлению данными о театральных постановках, обеспечивая их эффективную обработку и анализ.

Ответы на контрольные вопросы

1. Как выделяется память под вариативную часть записи?

Память под вариативную часть записи обычно выделяется с использованием указателей или динамических массивов. В основной структуре данных определен указатель, который указывает на область памяти, где хранятся вариативные данные. Эта память выделяется и освобождается по мере необходимости в процессе выполнения программы.

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

Если в вариативную часть введены данные, несоответствующие ожидаемым, это может привести к некорректной работе программы. Например, если ожидается ввод числа, а пользователь вводит текст, это может вызвать ошибку или некорректное поведение программы. Поэтому важно предусмотреть проверки и обработку некорректного ввода.

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

Ответственность за правильность выполнения операций с вариативной частью записи лежит на программисте. Программист должен уделять внимание правильному выделению и освобождению памяти, а также обработке данных в вариативной части, чтобы избежать утечек памяти и ошибок в программе.

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

Таблица ключей представляет собой структуру данных, которая используется для ускорения поиска, сортировки и доступа к данным в основной таблице. Она содержит ссылки (индексы, указатели) на записи в основной таблице. Таблица ключей полезна, когда требуется быстрый доступ к данным, отсортированным по определенному критерию.

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

Эффективность обработки данных в самой таблице или с использованием таблицы ключей зависит от конкретной задачи. Если часто выполняются операции поиска и сортировки данных, то использование таблицы ключей может значительно ускорить выполнение программы. Однако, если требуется постоянный доступ и изменение данных, то работа с данными в основной таблице может быть более эффективной.

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

Выбор способа сортировки зависит от размера данных и требований к производительности. Быстрая сортировка, например, может быть предпочтительнее для больших объемов данных, так как она имеет лучшую временную сложность. Однако, она требует дополнительной памяти для стека вызовов. Медленная сортировка, такая как сортировка пузырьком, может быть менее эффективной, но более простой в реализации и не требует дополнительной памяти. Выбор зависит от конкретной задачи и компромиссов между временем выполнения и потребляемой памятью.