

Техническое задание:

Задача:

Приобрести навыки работы с типом данных «запись» («структура»), содержащим вариантную часть, и с данными, хранящимися в таблицах. Оценить относительную эффективность программы (в процентах) по времени и по используемому объему памяти в зависимости от используемого алгоритма и от объема сортируемой информации.

Ввести репертуар театров, содержащий: название театра, спектакль, режиссер, диапазон цены билета, тип спектакля: детский – для какого возраста, тип (сказка, пьеса, музыкальный); взрослый – пьеса, драма, комедия); музыкальный – композитор, страна, минимальный возраст, продолжительность). Вывести список всех музыкальных спектаклей для детей указанного возраста с продолжительностью меньше указанной.

Входные данные: файл, содержащий список спектаклей с информацией о них, возможно добавление записей вручную.

Выходные данные: результат поиска спектакля по имени; отсортированная по названиям спектаклей таблица; время на сортировку; поиск музыкальных спектаклей для детей определенного возраста с продолжительностью меньше указанной.

Взаимодействие с программой:

Взаимодействие через консоль по меню:

1. Загрузка данных из файла
2. Сохранение изменений в файл
3. Добавление записи
4. Удаление записи
5. Сортировка таблицы
6. Печать таблицы
7. Сортировка таблицы ключей
8. Печать таблицы ключей
9. Печать таблицы в последовательности ключей
10. Поиск музыкальных спектаклей для детей по возрасту и продолжительности
11. Закреть

Вводится число и выполняется операция.

Аварийные ситуации:

1. Файл не существует. Вывод сообщения об ошибке.
2. Таблица переполнена при добавлении записи. Вывод сообщения об ошибке.
3. Некорректный ввод. Вывод сообщения об ошибке.
4. Таблица не загружена. Вывод сообщения об ошибке.

Используемые структуры:

Таблица, считываемая из файла. Каждая строка содержит информацию о спектакле.

```
struct performance {  
    char theatre[LEN]; // название театра  
    char name[LEN]; // название спектакля  
    char producer[LEN]; // продюсер  
    int begin_price; // минимальная цена билета  
    int end_price; // максимальная цена билета  
    char type[LEN]; // тип  
    union { // может быть детский или взрослый  
        struct { // если детский  
            int age; // минимальный возраст  
            char type_child[LEN]; // тип  
            char composer[LEN]; // если музыкальный указывается композитор  
            char country[LEN]; // страна, продолжительность  
            int duration;  
        } child;  
    };
```

```

    struct { //если взрослый
        char type_adult[LEN]; // тип
    } adult;
};
};

```

Таблица ключей.

```

struct key_performance {
    int number; //идентификационный номер
    char name[LEN]; // название спектакля
};

```

Алгоритмы:

1. Считывание таблицы из файла

```
int download_file(struct performance *poster, struct key_performance *key, int *size)
```

Проверяется файл на существование, если существуют, то данные считываются в структуру до окончания файла, производится подсчет количества записей, вывод сообщения на экран.

Возвращает SUCCESS если таблица считалась, возвращает INCORRECT если файла не существует, вывод сообщения на экран.

2. Сохранение в файл изменений

```
int save_to_file(struct performance *poster, int *size)
```

Вводится имя файла, туда сохраняется отредактированная таблица. Создается таблица ключей.

Возвращает SUCCESS если таблица записалась, возвращает INCORRECT если файла не существует, вывод сообщения на экран.

3. Добавление записи

```
int add_record(struct performance *poster, struct key_performance *key, int *size)
```

Проверяет на существование таблицы и на ее переполнение. Если возможно добавляет запись, проверяет ввод на корректность (цены и возраст должны быть

больше 0), ограничение на ввод строки 50 символов и вариантные поля должны иметь только допустимые значения. Ввод происходит до корректного значения. Ограничение на переполнение 1000 записей. Редактируется таблица ключей.

Возвращает SUCCESS если запись добавлена, возвращает INCORRECT если таблицы не существует или переполнение, вывод сообщения на экран.

4. Удаление записи

```
int delete_record(struct perfomance *poster, struct key_perfomance *key, int *size)
```

Проверяет на существование таблицы и если есть запись с таким именем спектакля удаляет эту запись. Редактируется таблица ключей.

Возвращает SUCCESS если запись удалена, возвращает INCORRECT если таблицы не существует, вывод сообщения на экран.

5. Сортировка таблицы

```
int sort_table(struct perfomance *poster, int *size)
```

Проверяет на существование таблицы. Сортирует пузырьком и быстрой (встроенной) сортировкой. Выводит время работы.

Возвращает SUCCESS если запись отсортирована, возвращает INCORRECT если таблицы не существует, вывод сообщения на экран.

6. Печать таблицы

```
int print_table(struct perfomance *poster, int *size)
```

Проверяет на существование таблицы и выводит ее на экран.

Возвращает SUCCESS если таблица напечатана, возвращает INCORRECT если таблицы не существует, вывод сообщения на экран.

7. Сортировка таблицы ключей

```
int sort_key_table(struct key_perfomance *key, int *size)
```

Проверяет на существование таблицы. Сортирует пузырьком и быстрой (встроенной) сортировкой. Выводит время работы.

Возвращает SUCCESS если отсортированы ключи, возвращает INCORRECT если таблицы не существует, вывод сообщения на экран.

8. Печать таблицы ключей

```
int print_key_table(struct key_perfomance *key, int *size)
```

Проверяет на существование таблицы и выводит ее на экран.

Возвращает SUCCESS если напечатано, возвращает INCORRECT если таблицы не существует, вывод сообщения на экран.

9. Печать таблицы по ключу

```
int print_table_by_key(struct performance *poster, struct key_performance *key, int *size)
```

Проверяет на существование таблицы и выводит ее на экран.

Возвращает SUCCESS если напечатано, возвращает INCORRECT если таблицы не существует, вывод сообщения на экран.

10. Поиск

```
int search(struct performance *poster, int *size)
```

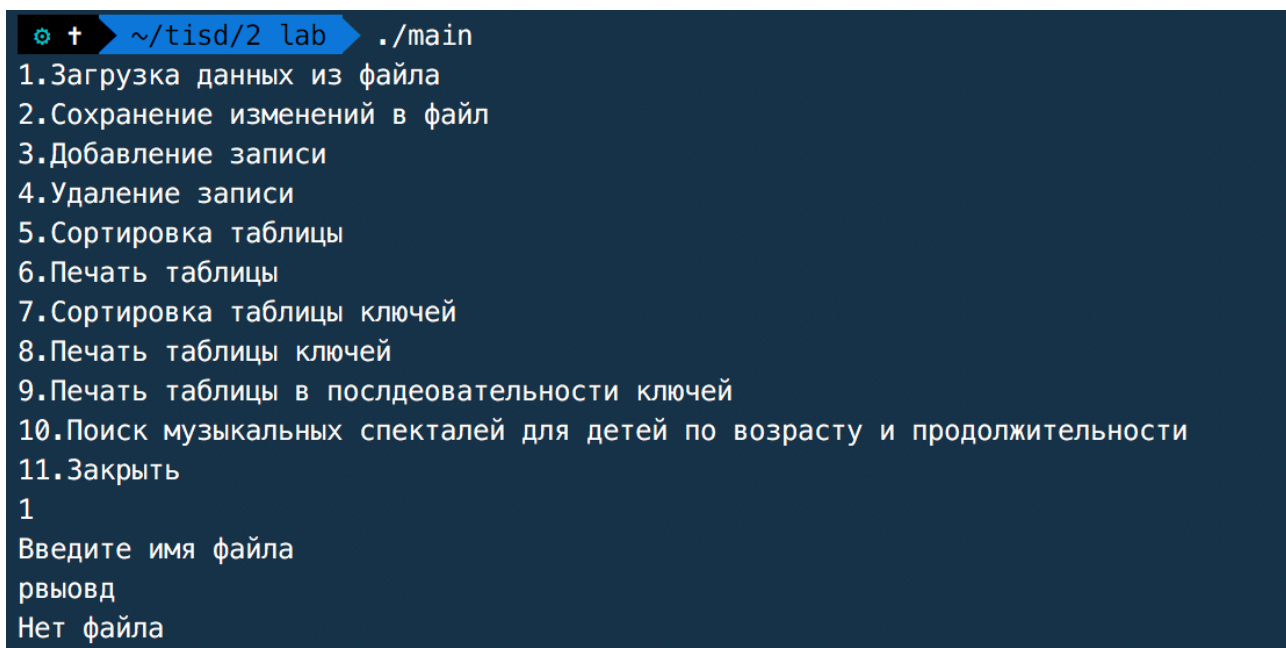
Проверяет на существование таблицы и выполняет поиск музыкальных спектаклей для детей определенного возраста с продолжительностью меньше указанной.

Возвращает SUCCESS если запись найдена, возвращает INCORRECT если таблицы не существует или запись не найдена, вывод сообщения на экран.

Тесты:

Некорректные данные

1. Некорректный файл



```
❏ + ~/tisd/2 lab ./main
1.Загрузка данных из файла
2.Сохранение изменений в файл
3.Добавление записи
4.Удаление записи
5.Сортировка таблицы
6.Печать таблицы
7.Сортировка таблицы ключей
8.Печать таблицы ключей
9.Печать таблицы в последовательности ключей
10.Поиск музыкальных спектаклей для детей по возрасту и продолжительности
11.Закреть
1
Введите имя файла
рвывод
Нет файла
```

2. Не существует таблицы

```
❏ + ~/tisd/2 lab ./main
1.Загрузка данных из файла
2.Сохранение изменений в файл
3.Добавление записи
4.Удаление записи
5.Сортировка таблицы
6.Печать таблицы
7.Сортировка таблицы ключей
8.Печать таблицы ключей
9.Печать таблицы в последовательности ключей
10.Поиск музыкальных спектаклей для детей по возрасту и продолжительности
11.Закрыть
3
Нет загруженной таблицы
```

3 .

Не корректные данные при вводе

```
Файл загружен
1.Загрузка данных из файла
2.Сохранение изменений в файл
3.Добавление записи
4.Удаление записи
5.Сортировка таблицы
6.Печать таблицы
7.Сортировка таблицы ключей
8.Печать таблицы ключей
9.Печать таблицы в последовательности ключей
10.Поиск музыкальных спектаклей для детей по возрасту и продолжительности
11.Закрыть
3
Введите театр большой
Введите название спектакля фигаро
Введите режиссера иванов
Введите минимальную цену -2
Некорректный ввод, попробуйте еще раз 45
Введите максимальную цену 
```

4. Элемента не найдено

Файл загружен

Файл загружен

1. Загрузка данных из файла
 2. Сохранение изменений в файл
 3. Добавление записи
 4. Удаление записи
 5. Сортировка таблицы
 6. Печать таблицы
 7. Сортировка таблицы ключей
 8. Печать таблицы ключей
 9. Печать таблицы в последовательности ключей
 10. Поиск музыкальных спектаклей для детей по возрасту и продолжительности
 11. Закреть
- 10

Введите возраст ребенка 4

Введите максимальную продолжительность 10

Спектаклей по указанным параметрам нет

5. Поиск не дал результатов

6. Таблица переполнена

Файл загружен

1. Загрузка данных из файла
2. Сохранение изменений в файл
3. Добавление записи
4. Удаление записи
5. Сортировка таблицы
6. Печать таблицы
7. Сортировка таблицы ключей
8. Печать таблицы ключей
9. Печать таблицы в последовательности ключей
10. Поиск музыкальных спектаклей для детей по возрасту и продолжительности
11. Закреть

3

Таблица переполнена

Корректные данные:

7. Сортировка

5

Время сортировки пузырьком 66 такт

Время быстрой сортировки 38 такт

8. Добавление записи

11. Закрыть

3

Введите театр(меньше 50 символов) большой

Введите название спектакля(меньше 50 символов) фигаро

Введите режиссера(меньше 50 символов) иванов

Введите минимальную цену 1000

Введите максимальную цену 4000

Введите тип спектакля (взрослый/детский) взрослый

Введите тип спектакля(драма/пьеса/комедия) драма

Запись добавлена

Сравнение времени:

Для 1000 элементов	Таблица	Таблица ключей	Эффективность
Пузырьком	17757	12356	30,4 %
Быстрая	344	120	65 %

Используемая память:

Таблица 382 байт

Ключи прибавляют к памяти еще 58 байт

То есть при использовании ключей проигрыш по памяти 15%.

Контрольные вопросы

1. Как выделяется память под вариантную часть записи?

В С вариантная часть записи определяется с помощью union. Память под вариантную часть выделяется по размеру наибольшего поля вариантной части.

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

Данная возможность в программе должна быть заблокирована. Поля в вариантной части записи расположены в одной области памяти – если данные были записаны туда согласно одному описанию, а обрабатываться будут согласно другому, будут использоваться неправильные значения, некорректная работа с памятью.

3. Кто должен следить за правильностью выполнения операций с вариантной

частью записи?

Контроль за правильностью выполнения возлагается на программиста.

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

Таблица ключей используется для уменьшения времени, затрачиваемого на сортировку таблиц, содержащих большое число строк или столбцов.

Представляет структуру, которая содержит индекс элемента из таблицы и имя ключа.

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

При сортировке таблицы ключей экономится время, так как исходная таблица содержит большее число полей, и на перестановку их значений не тратится время. Хотя для размещения таблицы ключей требуется дополнительная память.

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

При сортировке таблиц, равно как и при любой другой сортировке, оцениваются затраты во времени и памяти. Наиболее эффективной считается сортировка qsort встроенная.

Вывод:

Сортировка по ключам эффективна при больших объемах данных, так как дает больший выигрыш во времени. Но при маленьких данных лучше использовать просто сортировку таблицы без использования дополнительной памяти для хранения ключей.