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»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

«ЗАПИСИ С ВАРИАНТАМИ. ОБРАБОТКА ТАБЛИЦ»

по курсу «Типы и структуры данных»

Вариант 5

Студент: Писаренко Дмитрий Павлович

Группа: ИУ7-34Б

Студент _____ Писаренко Д.П.

подпись, дата *фамилия, и.о.*

Преподаватель _____ Барышникова М.Ю.

подпись, дата *фамилия, и.о.*

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 %).

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

Техническое задание

Входные данные

Массив структур типа “theatres_t”, максимальный размер – 500 структур, хранится в виде структуры table_t.

```
typedef struct
{
    theatres_t theatres[TABLE_SIZE];
    keys_t keys[TABLE_SIZE];
    int size;
} table_t;
```

```
theatres_t theatres[TABLE_SIZE] – массив репертуаров театров
keys_t keys[TABLE_SIZE] – массив ключей
int size – размер таблицы
```

Описание полей структуры table_t

В свою очередь структура “theatres_t” содержит:

```
typedef struct
{
    char name_theatre[THEATRE_NAME_LEN + 1];
    char performance[PERFORMANCE_LEN + 1];
    char producer[PRODUCER_NAME_LEN + 1];
    int min_price;
    int max_price;
    char str_type_of_performance[TYPE_LEN + 1];
    type_of_performance_t type_of_performance;
} theatres_t;
```

```
char name_theatre[THEATRE_NAME_LEN + 1] – название театра
char performance[PERFORMANCE_LEN + 1] - спектакль
char producer[PRODUCER_NAME_LEN + 1] - режиссер
int min_price – минимальная цена билета
int max_price – максимальная цена билета
char str_type_of_performance[TYPE_LEN + 1] – тип спектакля
type_of_performance_t type_of_performance – объединение трех структур типов
спектакля
```

Описание полей структуры theatres_t

```
typedef union
{
    child_t child;
    adult_t adult;
    music_t music;
} type_of_performance_t;
```

Структуры “child_t”, “adult_t” и “music_t” содержат в себе набор различных полей.

```c
typedef struct
{
    int min_age;
    char type[TYPE_LEN + 1];
} child_t;

typedef struct
{
    char type[TYPE_LEN + 1];
} adult_t;

typedef struct
{
    char composer[COMPOSER_NAME_LEN + 1];
    char country[COUNTRY_LEN + 1];
    int min_age;
    int duration;
} music_t;
```

```
int min_age - минимальный возраст для посещения
char type[TYPE_LEN + 1] - тип спектакля
char composer[COMPOSER_NAME_LEN + 1] - фамилия композитора
char country[COUNTRY_LEN + 1] - страна
int duration - продолжительность
```

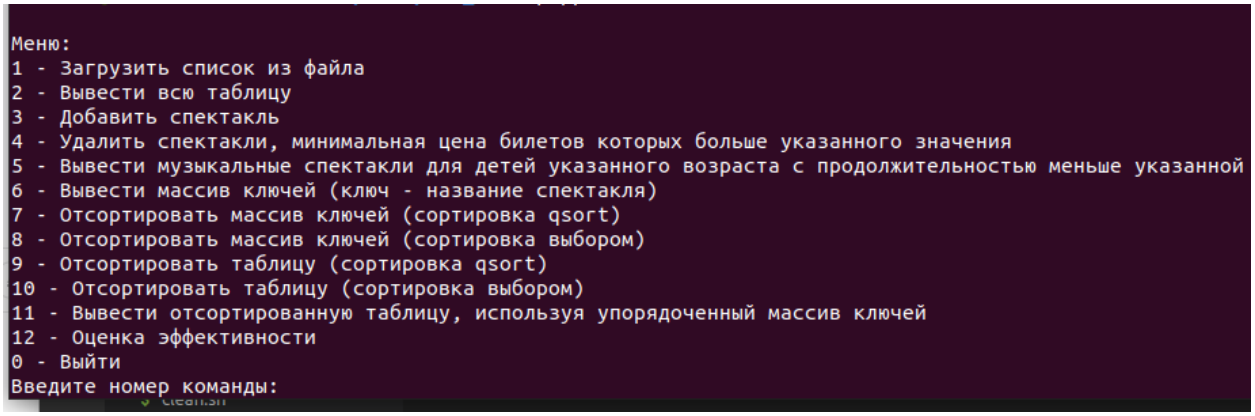
Описание полей структур child_t, adult_t и music_t

Ограничения на входные данные

- Название театра не более 14 символов
- Название спектакля не более 14 символов
- Фамилия режиссера не более 14 символов
- Минимальная и максимальная цены – натуральные числа
- Тип спектакля – “child” / ”adult” / ”music”
- Минимальный возраст – диапазон целых чисел от 0 до 18
- Тип спектакля для “child” – “tale” / “piece”
- Тип спектакля для “adult” – “piece” / “drama” / “comedy”
- Фамилия композитора не более 14 символов
- Продолжительность – натуральное число (в минутах)

Массив может хранить информацию о 500 репертуарах театров.

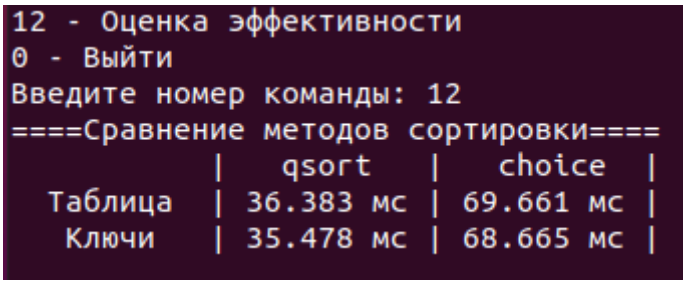
Главное меню



Сортировки

В ходе лабораторной работы измерены скорости двух сортировок (qsort и выбором) на различных количествах записей. Далее будет приведена таблица, в которой время указано в мс, повторений – 1000.

Количество записей	qsort		выбором	
	Исходная таблица	Таблица ключей	Исходная таблица	Таблица ключей
50	36	35	69	68
100	92	72	309	282
150	203	174	953	724



Эксперимент на 50 записях

Расход памяти на 50 структур

Исходная таблица	Таблица ключей
5800 байт	1400 байт

Тестирование

Позитивные тесты

#	Входные данные	Выходные данные	Результат
1	Ключ = 1	Сообщение “Загрузка прошла успешно!”	Ожидание следующего ключа
2	Ключ = 2	На экран выводится таблица спектаклей	Ожидание следующего ключа
3	Ключ = 3 Данные о спектакле	Сообщение “Добавление прошло успешно!”	Ожидание следующего ключа
4	Ключ = 4 Граница цены билета	Сообщение “Количество удаленных спектаклей: N”	Ожидание следующего ключа
5	Ключ = 5 Мин. возраст Макс. длительность	Таблица подходящих спектаклей + сообщение “Количество подходящих спектаклей: N”	Ожидание следующего ключа
6	Ключ = 6	На экран выводится таблица ключей (ключ – название спектакля)	Ожидание следующего ключа
7	Ключ = 7	Сообщение “Сортировка прошла успешно!”	Ожидание следующего ключа
8	Ключ = 8	Сообщение “Сортировка прошла успешно!”	Ожидание следующего ключа
9	Ключ = 9	Сообщение “Сортировка прошла успешно!”	Ожидание следующего ключа
10	Ключ = 10	Сообщение “Сортировка прошла успешно!”	Ожидание следующего ключа
11	Ключ = 11	На экран выводится отсортированная таблица спектаклей	Ожидание следующего ключа

12	Ключ = 12	На экран выводится таблица сравнения методов сортировки и расход памяти	Ожидание следующего ключа
13	Ключ = 0	Поток вывода пустой	Код возврата 0

Негативные тесты

#	Входные данные	Выходные данные	Результат
1	Ключ = 5 В возрасте введены буквы	Сообщение “Возраст введен некорректно”	Код возврата 4
2	Ключ = 5 Длительность меньше 0	Сообщение “Длительность не может быть меньше нуля”	Код возврата 5
3	Ключ = 1 Файл не существует	Сообщение “Неуспешное открытие файла”	Код возврата 1
4	Ключ = 999	Сообщение “Номер команды – от 0 до 12”	Код возврата 14
5	Ключ = 4 В цене билета введены буквы	Сообщение “Неудачное считывание границы минимальной цены билета”	Код возврата 10
6	Ключ = sf	Сообщение “Недопустимый номер команды“	Код возврата 3
7	Ключ = 3 Мин. цена > макс. цена	Сообщение “Максимальная цена билета должна быть больше минимальной”	Код возврата 11

Контрольные вопросы

Как выделяется память под вариантную часть записи?

В языке Си вариативная часть структуры реализована с помощью объединений. Память выделяется в одном “куске” памяти, имеющий размер, который способен вместить наибольшее поле из указанных.

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

Результат будет зависеть от системы, то есть неопределенное поведение. Один из вариантов – приведение типов.

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

Программист.

Что представляет собой таблица ключей, зачем она нужна?

Таблица ключей - таблица, с двумя столбцами: индекс в исходной таблице и значение поля, которое мы выбирали сами (у меня – название спектакля).

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

В самой таблице, когда память важнее времени. В таблице ключей, когда время важнее памяти.

Какие способы сортировки предпочтительнее для обработки таблиц и почему?

Для таблиц из большого количества записей лучше использовать способы сортировки со средним временем обработки O(n*log(n)): merge sort, tree sort, heap sort и т.д.

Если же в таблице небольшое количество записей, то лучше использовать простые алгоритмы сортировки: selection sort, insertion sort и т.д.

Вывод

В процессе выполнения лабораторной работы был получен опыт работы со структурами с вариативной частью. Я узнал, что при работе с большими массивами структур выгоднее хранить таблицу ключей, так как увеличивается эффективность по времени в несколько раз, при этом общая память возрастает всего на ~20 процентов.