|  |  |
| --- | --- |
|  | **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”**

Название предмета: Типы и структуры данных

Студент: Малышев Иван Алексеевич

Группа: ИУ7-31Б

*2020г.*

1. **Описание условия задачи**

Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами (объединениями)). Упорядочить данные в ней по возрастанию ключей, двумя алгоритмами сортировки, где ключ – минимальная стоимость представления, используя:

1. саму таблицу,
2. массив ключей.

(Возможность добавления и удаления записей в ручном режиме обязательна). Осуществить поиск информации по варианту.

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.

1. **Техническое задание**
2. ***Описание исходных данных***

*Файл с данными.* Текстовый файл формата .txt. Разделителем в файле является символ новой строки ‘\n’. Каждая новая запись таблицы в обязательном порядке должна находиться на новой строке.

*Целое неотрицательное число.* Представляет собой номер операции*.* Не превышает одиннадцати.

*Дополнительная информация.* Строковое или целочисленное поле, в зависимости от операции.

1. ***Описание результата программы***
2. Полученная таблица (основная или таблица ключей) в отсортированном или неотсортированном виде (в зависимости от выполненной команды).
3. Количественная характеристика сравнения вариантов сортировки таблицы.
4. ***Описание задачи, реализуемой программой***

Программа выводит меню операций, которые она может выполнить, в виде «номер - операция». Она позволяет:

1. Загрузить таблицу из файла
2. Сортировать таблицу
3. Сортировать таблицу по ключу
4. Добавить запись в таблицу
5. Удалить запись из таблицы
6. Создать таблицу по ключу
7. Выводить таблицу по ключу
8. Выводить список ключей
9. Выводить таблицу
10. Совершать поиск по указанным данным
11. Сравнить времени сортировки
12. ***Способ обращения к программе***

Способ обращения к программе – через терминал ОС по её имени «app.exe». Дальнейшие инструкции будут выведены после запуска.

1. ***Описание возможных аварийных ситуаций и ошибок пользователя***

* Некорректный номер операции, т. е. число меньше нуля или больше 11
* Пустой или несуществующий файл
* Невалидные данные в файле
* Введён недопустимый индекс при удалении элемента таблицы
* Выполнение операции над таблицей до загрузки данных, т.е. совершение любой операции, кроме 1 и 11
* Ввод недопустимого типа спектакля
* Таблица ключей не совпадает с исходной таблицей

Во всех указанных случаях программа завершится корректно или сообщит об ошибке.

1. **Описание внутренних структур данных**

Структура **TheatreTable** представляет собой таблицу репертуаров театров с указателем на её начало ptr\_first, размером size и максимальным размером size\_max. Если максимальный размер превышен, то таблица динамически расширяется.

struct TheatreTable

{

struct Theatre \*ptr\_first;

int size;

int size\_max;

};

Структура **Theatre** описывает информацию о репертуаре театра. Длина строковых полей не превышает LEN\_NAME(= 40) + 1 (для терминального нуля). Имеет вариантное поле, которое хранит определённую структуру в зависимости от типа спектакля.

struct Theatre

{

char t\_name[LEN\_NAME + 1]; // название театра

char p\_name[LEN\_NAME + 1]; // название представления

char d\_name[LEN\_NAME + 1]; // фамилия режиссёра

int low\_price; // минимальный диапазон цен

int top\_price; // максимальный диапазон цен

int p\_type; // тип представления

union

{

struct

{

int age; // рекомендованный возраст посещения

int type; // 0 - сказка, 1 - пьеса

} child; // детский

struct

{

int type; // 0 - пьеса, 1 - драма, 2 - комедия

} adult; // взрослый

struct

{

char composer[LEN\_NAME + 1]; // фамилия композитора

char country[LEN\_NAME + 1]; // название страны

int age; // минимальный возраст посещения

double duration; // продолжительность

} musical; // музыкальный

} property; // тип спектакля

};

Структура **KeyTable** представляет собой таблицу репертуаров театров с указателем на её начало ptr\_first и размером n.

struct KeyTable

{

struct Key\* ptr\_first;

int n;

};

Структура **Key** описывает информацию о ключе поля. Ключ формируется по минимальной цене представления.

struct Key

{

int key; // ключ поля

int id; // номер поля в таблице

};

1. **Описание функций**

Функции для работы с репертуарами и таблицей репертуаров:

**void input\_theatre(struct Theatre \*thtr)** – ввод репертуара театра из терминала

**void output\_theatre\_console(struct Theatre thtr)** – печать репертуара в терминал

**int load\_table(FILE \*f, struct TheatreTable \*tbl)** – загрузка таблицы из файла

**int add\_to\_table(struct TheatreTable \*tbl, const struct Theatre \*thtr)** – добавление репертуара в таблицу

**int remove\_from\_table(struct TheatreTable \*tbl, int i)** – удаление репертуара из таблицы

**void clear\_table(struct TheatreTable \*tbl)** – очистка таблицы

**void output\_thTable\_console(struct TheatreTable \*tbl)** – печать таблицы репертуаров в терминал

**int cmp\_low\_price(const void \*a, const void \*b)** – сравнение минимальных стоимостей репертуаров

**void sort\_theatre\_table(struct TheatreTable \*tbl)** – сортировка таблицы репертуаров

**void save\_theatre\_table(const char \*file\_name, struct TheatreTable \*tbl)** – сохранение текущей таблицы в файл

Функции для работы с ключами и таблицей ключей:

**int create\_key\_table(struct TheatreTable\* thtr\_stud, struct KeyTable\* arr\_keys)** – создание таблицы ключей

**void clear\_key\_table(struct KeyTable\* arr\_keys)** – очистка таблицы ключей

**void sort\_key\_table(struct KeyTable\* arr\_keys)** – сортировка таблицы

**void print\_key\_table(const struct KeyTable\* arr\_keys)** – печать таблицы ключей

**int print\_theatre\_table\_by\_key(const struct TheatreTable\* arr\_thtr, const struct KeyTable\* arr\_keys)** – печать таблицы репертуаров по ключу

**int cmp\_key(const void \*key1, const void \*key2)** – сравнение ключей

**int check\_keyTbl(const struct TheatreTable\* arr\_thtr, const struct KeyTable\* arr\_keys)** – проверка актуальности таблицы ключей

Функции перестановок, сортировки и замера времени сортировки:

**unsigned long long tick(void)** – ассемблерная функция для измерения тактов процессора

**void measure\_sorting\_time(void)** – функция сравнения методов сортировок

**void swap\_theatres(struct Theatre \*a, struct Theatre \*b)** – перестановка репертуаров

**void mysort\_thtr(struct Theatre \*base, int nitems)** – собственный метод сортировки(по факту – «пузырёк») исходной таблицы

**void swap\_keys(struct Key \*a, struct Key \*b)** – перестановка ключей

**void mysort\_keys(struct Key \*base, size\_t nitems)** - собственный метод сортировки(по факту – «пузырёк») таблицы ключей

Функции для ввода данных:

**int fread\_line(FILE \*f, char s[], size\_t n)** – чтение строки из файла

**void input\_string(char \*msg, char \*str, int max\_len)** – ввод строки в терминал

**void input\_number(char \*msg, int \*num, int beg, int end)** – ввод числа в терминал

**void input\_bool(char \*msg, int \*num)** – ввод булевого значения(0 или 1) в терминал

**void input\_type(char \*msg, int \*num)** – ввод типа спектакля в терминал

**void input\_dur(char \*msg, double \*num, double beg, double end)** – ввод продолжительности спектакля в терминал

1. **Описание алгоритма**
2. Пользователь вводит номер операции из меню.
3. Пока пользователь не введет 0 (выход из программы), ему будет предложено выполнять действия с таблицей.
4. Данные из файла помещаются таблицу, под которой подразумевается указанная выше структура.
5. При формировании таблицы ключей в качестве ключа используется минимальная цена спектакля, т. е. целое число.
6. При сравнении методов сортировок используются сортировки с вычислительной сложностью O(n2) и O(nlogn).

Формирование ключа по целому числу является более предпочтительным, так как сравнивать целые числа при сортировке или поиске гораздо быстрее, чем строки.

1. **Тестирование**

Позитивные тесты:

|  |  |  |
| --- | --- | --- |
| Описание теста | Входные данные | Результат |
| Ввод записи в таблицу | 4  Корректное поле таблицы | Запись добавлена в таблицу |
| Удаление из таблицы записей по ключу | 513 (при наличии 3 подходящих полей) | Удалено(-а) 3 записей(-ь) |
| Удаление из таблицы записи по ключу | 999 (при отсутствии подходящих полей) | Удалено(-а) 0 записей(-ь) |
| Корректный ввод операции | 4 и далее введено корректное поле | Добавление записи в таблицу |
| Ввод названия непустого файла с валидными данными | 1  Data.txt | Успешное создание таблицы |

Негативные тесты:

|  |  |  |
| --- | --- | --- |
| Описание теста | Входные данные | Результат |
| Некорректная операция | 12 | Некорректный ввод |
| Невалидный файл | Data.ttxxt | Ошибка загрузки |
| Сортировка таблицы до загрузки данных | 2 | Таблица пуста |
| Некорректный ввод типа вариантного поля | …  4 | Ошибка загрузки |
| Создание таблицы ключей до сортировки исходной таблицы | …  6  2  8 | Таблица ключей неактуальна |

1. **Оценка эффективности**

Измерения эффективности сортировок будут производиться в единицах измерения – тактах процессоров. Для измерения была специально написана ассемблерная функция, поэтому погрешность измерений минимальна. При записи результатов использовалось среднее количество тактов, полученное по результатам 10 измерений.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Количество записей | Сортировка пузырьком | | | Быстрая сортировка | |
| Исходная таблица | | Таблица ключей | Исходная таблица | Таблица ключей |
| 50 | | 71086 | 16324 | 31300 | 5777 |
| 100 | | 106632 | 24490 | 45385 | 7510 |
| 150 | | 170611 | 36796 | 65810 | 9763 |
| 200 | | 272977 | 55194 | 95424 | 16501 |
| 250 | | 436764 | 82791 | 138364 | 21455 |
| 300 | | 698823 | 124186 | 200628 | 27891 |

Объём занимаемой памяти (в байтах):

|  |  |  |
| --- | --- | --- |
| Количество записей | Исходная таблица | Таблица ключей |
| 50 | 11450 | 400 |
| 100 | 22900 | 800 |
| 150 | 34350 | 1200 |
| 200 | 45800 | 1600 |
| 250 | 57250 | 2000 |
| 300 | 68700 | 2400 |

|  |  |  |  |
| --- | --- | --- | --- |
| Количество записей | Дополнительная память (%) для хранения массива ключей (по отношению ко всей таблице) | Рост скорости сортировки массива ключей по сравнению с таблицей (сортировка пузырьком) | Рост скорости сортировки массива ключей по сравнению с таблицей (быстрая сортировка) |
| 50 | ~3% | ~4 раз | ~5 раз |
| 100 | ~3% | ~4 раз | ~6 раз |
| 150 | ~3% | ~4 раз | ~6 раз |
| 200 | ~3% | ~4 раз | ~6 раз |
| 250 | ~3% | ~5 раз | ~7 раз |
| 300 | ~3% | ~5 раз | ~7 раз |

1. **Выводы по проделанной работе**

В результате установилась возможность работы с типом данных «запись» (структура), содержащим вариантную часть (объединение, смесь), и с данными, хранящимися в таблицах; произвести сравнительный анализ реализации алгоритмов сортировки; оценить эффективность программы по времени и по используемому объему памяти при использовании различных структур и эффективность использования различных алгоритмов сортировок.

При оценке эффективности сортировок выяснилось, что сортировка таблицы по ключам во обоих случаях оказалась быстрее в несколько раз, причём отношение не постоянно, а растёт, несмотря на то, что занимаемый процент массива ключей постоянен. Это связано с тем, что при сортировке исходной таблицы происходят перестановки строк в структурах, которые тоже влияют на конечное время сортировки.

Также стоит заметить, что в случае быстрой сортировки скорость сортировки по ключам по сравнению с таблицей растёт быстрее, так как в быстрой сортировке совершается меньше перестановок, чем в «пузырьке», поэтому отношение чувствуется сильнее.

1. **Ответы на вопросы**
2. **Как выделяется память под вариантную часть записи?**

Размер памяти, выделяемый под вариантную часть, равен максимальному по длине полю вариантной части. Эта память является общей для всех полей вариантной части записи.

1. **Что будет, если в вариантную часть ввести данные, не соответствующие описанным?**

Тип данных в вариантной части при компиляции не проверяется. Из-за того, что невозможно корректно прочитать данные, поведение будет неопределенным.

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

Контроль за правильностью выполнения операций с вариантной частью записи возлагается на программиста.

1. **Что представляет собой таблица ключей, зачем она нужна?**

Дополнительный массив (структура), содержащий индекс элемента в исходной таблице и выбранный ключ. Она нужна для оптимизации сортировки.

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

В случае, если мы сортируем таблицу ключей, мы экономим время, так как перестановка записей в исходной таблице, которая может содержать большое количество полей, отсутствует. С другой стороны, для размещения таблицы ключей требуется дополнительная память. Кроме того, если в качестве ключа используется символьное поле записи, то для сортировки таблицы ключей необходимо дополнительно обрабатывать данное поле в цикле, следовательно, увеличивается время выполнения. Выбор данных из основной таблицы в порядке, определенном таблицей ключей, замедляет вывод. Если исходная таблица содержит небольшое число полей, то выгоднее обрабатывать данные в самой таблице.

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

Если обработка данных производится в таблице, то необходимо использовать алгоритмы сортировки, требующие наименьшего количества операций перестановки. Если сортировка производится по таблице ключей, эффективнее использовать сортировки с наименьшей сложностью работы.