|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

**«ЗАПИСИ С ВАРИАНТАМИ, ОБРАБОТКА ТАБЛИЦ»**

Студент Нисуев Нису Феликсович

Группа ИУ7 – 32Б

Преподаватель Барышникова Марина Юрьевна

2023г.

**Оглавление**

[**Оглавление** 2](#_Toc147521631)

[**Описание условия задачи** 3](#_Toc147521632)

[**Описание технического задания** 3](#_Toc147521633)

[**Описание структуры данных** 5](#_Toc147521634)

[**Описание алгоритма** 6](#_Toc147521635)

[**Набор тестов** 6](#_Toc147521636)

[**Сравнительная характеристика сортировок** 8](#_Toc147521637)

[**Ответы на контрольные вопросы** 9](#_Toc147521638)

[**Вывод** 10](#_Toc147521639)

# **Описание условия задачи**

Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами). Упорядочить данные в ней по возрастанию ключей, где ключ – любое невариантное поле (по выбору программиста), используя:

* саму таблицу
* массив ключей

(возможность добавления и удаления записей в ручном режиме обязательна).

Ввести репертуар театров, содержащий: название театра, спектакль, режиссер, диапазон цены билета, тип спектакля: пьеса, драма, комедия, сказка – возраст (3+, 10+,16+); 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 технического задания**

Входные данные:

1. Файл с данными: текстовый файл формата TXT. Разделителем в файле является символ пробел “|”. Каждая новая запись таблицы обязательно должна находиться на новой строке.
2. Номер команды: целое число в диапазоне {-1} ∪ [1; 10].
3. Дополнения к таблице: строковое или целочисленное поле (в зависимости от команды)

Выходные данные:

1. Полученная таблица в отсортированном или неотсортированном виде (в зависимости от выполненной команды).
2. Таблица из отобранных по заданию записей основной таблицы.
3. Характеристика сравнения вариантов сортировки таблицы.

Функции программы:

1. Загрузить таблицу из файла (./data/theatres.txt)
2. Добавить запись в конец таблицы.
3. Удалить запись из таблицы по названию спектакля.
4. Вывести таблицу на экран.
5. Отсортировать таблицу (сортировка пузырьком с флагом по названию театра) и вывести ее на экран.
6. Отсортировать таблицу ключей (сортировка пузырьком с флагом по названию театра) и вывести ее на экран.
7. Вывести на экран таблицу из всех балетов для детей указанного возраста с продолжительностью меньше указанной.
8. Сравнить сортировку таблицы и таблицы ключей по времени и памяти
9. Сравнить быструю сортировку и сортировку пузырьком с флагом по времени и памяти
10. Очистка таблицы

-1. Завершить программу.

Обращение к программе:

Запуск через терминал (./build/app.exe).

Аварийные ситуации:

1. Некорректный ввод номера команды: число не входит в диапазон или введено не число.

Сообщение на выходе: *«ERROR: Incorrect action»*

1. Некорректный файл.

Сообщение на выходе: *«ERROR: Incorrect file»*

1. Некорректный ввод строка.

Сообщение на выходе: *«ERROR: Incorrect input»*

# **Описание структуры данных**

|  |
| --- |
| //----------------------------  #define MAX\_THEATRE\_NAME 30    /// @def MAX\_THEATRE\_NAME - максимальная длина названия театра  #define MAX\_PERFOMACE\_NAME 30  /// @def MAX\_PERFOMACE\_NAME - максимальная длина названия представления  #define MAX\_COMPOSER\_NAME 30   /// @def MAX\_COMPOSER\_NAME - максимальная длина имени композитора  #define MAX\_COUNTRY\_NAME 30    /// @def MAX\_COUNTRY\_NAME - максимальная длина названия страны  #define RANGE\_PRICE 2          /// @def RANGE\_PRICE - длина массива диапозона цен  //----------------------------  // @enum play\_type\_t - перечисление типов спектаклей  typedef enum {      PIESA,    // Пьеса      DRAMA,    // Драма      COMEDY,   // Комедия      FAIRYTALE // Сказка  } play\_type\_t;  // @enum musicaly\_type\_t - перечисление типов музыкальных представлений  typedef enum {      BALLET,   // Балет      OPERA,    // Опера      MUSICAL   // Мюзикл  } musicaly\_type\_t;  // @struct play\_t - структура спектакля  typedef struct {      play\_type\_t type; // Тип спектакля      bool for\_kids;    // Детский ли спектакль      uint16\_t age;     // Проходной возраст спектакля  } play\_t;  // @struct musicaly\_t - структура музыкального представления  typedef struct {      char composer\_name[MAX\_COMPOSER\_NAME + 1]; // Имя композитора      char country\_name[MAX\_COUNTRY\_NAME + 1];   // Страна      musicaly\_type\_t type;                      // Тип музыкального представления      uint16\_t age;                              // Проходной возраст      uint32\_t duration;                         // Продолжительность  } musicaly\_t;  // @union perfomance\_t - представление  typedef union {      play\_t play;         // Структура спектакля      musicaly\_t musical;  // Структура музыкального представления  } perfomance\_t;  // @enum perfomance\_type\_t - типы представлений  typedef enum {      PLAY,     // Спектакль      MUSICALY  // Музыкальное представление  } perfomance\_type\_t;  // @struct theatre\_t - структура репертуара театра  typedef struct {      char theatre\_name[MAX\_THEATRE\_NAME + 1];      // Название театра      char perfomance\_name[MAX\_PERFOMACE\_NAME + 1]; // Название представления      uint32\_t price\_range[RANGE\_PRICE];            // Диапазон цены      perfomance\_type\_t type;                       // Тип представления      perfomance\_t perfomance;                      // Структура представления  } theatre\_t;  // @struct theatre\_table\_t - таблица репертуаров театра  typedef struct {      size\_t size;          // Размер      theatre\_t \*theatres;  // Массив репертуаров структур  } theatre\_table\_t;  // @struct key\_t - структура ключа  typedef struct {      char theatre\_name[MAX\_THEATRE\_NAME + 1]; // Название театра      size\_t ind;                              // Индекс в таблице репертуаров театров  } key\_t;  // @struct key\_table\_t - таблица ключей  typedef struct keys\_table{      key\_t \*keys;                                                      // Массив ключей      size\_t size;                                                      // Размер      void (\*key\_table\_gen)(struct keys\_table (\*) ,theatre\_table\_t);     // Генератор таблицы ключей      void (\*free\_key\_table)(struct keys\_table (\*));                     // Освобождение таблицы ключей} key\_table\_t; |

# **Описание алгоритма**

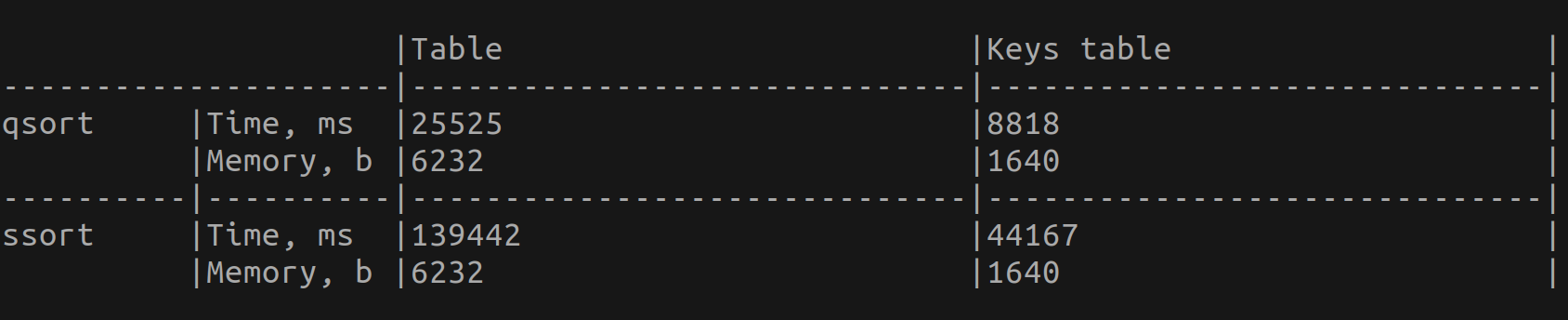
1. Ввод команды (возможные команды представлены в меню).
2. Пока пользователь не введет -1 (выход из программы), ему будет предложено вводить номера команд и выполнять действия по выбору.

# **Набор тестов**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Название теста** | **Пользовательский ввод** | **Результат** |
| **Позитиивные тесты** | | | |
| 1 | Добавление элемента в таблцу | 2  {корректный ввод} | Data added successfully |
| 2 | Загрузка данных из непустого файла | {корректный файл} | Data loaded successfully from file \(filename) |
| 3 | Загрузка данных из пустого файла | {корректный файл} | No data in \(filename) |
| 4 | Вывод непустой таблицы | 4 | {Таблица} |
| 5 | Вывод пустой таблицы | 4 | No loaded data |
| 6 | Сортировка таблицы | 5 | {Отсортированная таблица} |
| 7 | Сортировка таблицы ключей с выводом таблицы ключей | 6  К | { Отсортированная таблица ключей} |
| 8 | Сортировка таблицы ключей с выводом таблицы | 6  Т | { Отсортированная таблица} |
| 9 | Вывод искомых балетов  (Балеты найдены) | 7  {Корректные данные} | {Таблица искомых балетов} |
| 10 | Вывод искомых балетов  (Балеты не найдены) | 7  {Корректные данные} | No matching ballets found |
| 11 | Вывод таблицы сравнения таблиц | 8 | {Таблица сравнения таблиц} |
| 12 | Вывод таблицы сравнения сортировок | 9 | {Таблица сравнения сортировок} |
| 13 | Очистка таблицы | 10 | Data successfully cleaned |
| 14 | Выход из программы | -1 | Exit the application… |
| **Негативные тесты** | | | |
| 1 | Некорректное число действия | 23456 | ERROR: Incorrect action |
| 2 | Некорректное действие | Ф | ERROR: Incorrect action |
| 3 | Некорректный файл | notfile | ERROR: Incorrect file |
| 4 | Некорректный ввод | 2  {33 буквы ‘d’} | ERROR: Incorrect input |
| 5 | Некорректный ввод | 2  {Возраст: -13 } | ERROR: Incorrect input |
| 6 | Некорректный ввод | 2  {Продолжительность: -1} | ERROR: Incorrect input |
| 7 | Ввод литерала вместо числа | 2  {продолжительность: ф} | ERROR: Incorrect input |

# **Сравнительная характеристика сортировок**

В лабораторной работе были рассмотрены и сравнены два вида сортировки – пузырек с флагом и быстрая сортировка. Замеры были проведены на списке структур, содержащем 40+ записей. Время в миллисекундах, память в байтах. Проводилось 1000 измерений и бралось среднее.



Вывод: таблица ключей уменьшает время сортировки на 68,3% в случае сортировки пузырьком и на 65,5% в случае быстрой сортировки. Память под таблицу ключей выделяется на 26,3% больше но это сильно ускоряет сортировку.

# **Ответы на контрольные вопросы**

*1.Как выделяется память под вариантную часть записи?*

Выделяется общий блок памяти для всех полей вариантной части. Размер памяти равен максимальному по длине полю вариантной части.

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

Невозможно найти подходящий тип из встречающих в записи вариантной части, поэтому считать данные невозможно. Поведение не определено.

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

За правильностью выполнения операций с вариантной частью должен следить программист.

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

Таблица ключей – массив или структура, содержащая индексы определенного элемента каждой записи исходной таблицы. Таблица ключей позволяется сократить время при сортировке и поиске записей в исходной таблице, так как хранит только один параметр.

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

Таблица ключей позволяет сэкономить время при сортировке таблицы, так как перестановка записей исходной таблицы не происходит. Данный подход имеет минусы: для размещения таблицы ключей требуется дополнительная память, а если ключом является символьное поле, его обработка требует дополнительных временных затрат (в цикле). Обработка данных происходит менее ресурсозатратно при работе с самой таблицей (а не таблицей ключей), если она содержит небольшое число записей.

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

Если будет производится сортировка самой таблицы, то необходимо использовать алгоритмы, требующие наименьшее количество операций перестановки. Если же сортировка производится по таблице ключей, то эффективнее использовать сортировки с наименьшей сложностью работы.

# 

# **Вывод**

Программистам часто приходится работать с большими объемами данных, и поэтому оптимизация обработки этих данных становится ключевой задачей. Одним из способов решения этой задачи является объединение, так как оно экономит память. Также использование таблицы ключей и выбор подходящего алгоритма сортировки существенно ускоряют обработку данных, что является критически важным аспектом при работе с большими объемами данных. Даже при дополнительном использовании памяти эти оптимизации позволяют значительно повысить скорость и эффективность обработки данных.