|  |  |
| --- | --- |
| 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 ???

Группа ИУ7 – 3?Б

Преподаватель Силантьева А.В.

*2021*

**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).

Список студентов; даны фамилия, имя, группа, пол, возраст, дата поступления, средний балл за сессию, место жительства. Если общежитие, то номер общежития и комнаты, если квартира, то улица, номер улицы и квартиры.

Описание технического задания

Входные данные:

1. Файл с данными: текстовый файл. Каждая новая запись таблицы должна находиться на новой строке. В каждой записи хранится информация об одном студенте. Максимальное количество записей 2000.
2. Номер команды: целое число в диапазоне от 0 до 9.

**Выходные данные:**

1. Полученная таблица (основная или таблица ключей) в отсортированном или неотсортированном виде (в зависимости от выполненной команды).
2. Характеристика сравнения вариантов сортировки таблицы.
3. Список студентов после указанных пользователем действий.

Функции программы:

1. Загрузить таблицу из файла.
2. Добавить запись в конец таблицы.
3. Удалить запись из таблицы по оценке ниже введенной.
4. Отсортировать таблицу ключей и вывести ее на экран.
5. Отсортировать таблицу и вывести таблицу на экран.
6. Отсортировать таблицу ключей и вывести исходную таблицу по отсортированной таблице ключей.
7. Напечатать всю таблицу.
8. Напечатать студентов, живущих в общежитие, указанного года поступления.
9. Сравнить время сортировки таблицы сортировками со сложностями O(n^2) и O(n\*log(n)) и сравнение времени обычной сортировки и сортировки таблицы ключей.
10. Завершение программы.

Обращение к программе:

Запускается через терминал.

Аварийные ситуации:

1. Некорректный ввод номера команды.

На входе: число, большее чем 9 или меньшее, чем 0.

На выходе: сообщение «Invalid menu code»

2. Файл пуст.

На входе: пустой файл.

На выходе: сообщение «Sorry, but the file is empty»

3. Выполнение какой-либо команды до выгрузки файла.

На входе: целое число в диапазоне от 3 до 9 (номер команды).

На выходе: сообщение «Please, at first make the 1st or 2nd command»

4. Превышение количества записей в конечной таблице.

На входе: добавление новой записи при максимальном размере таблицы.

На выходе: сообщение «too big data»

5. Неверный ввод строкового поля.

На входе: отрицательное число.

На выходе: сообщение «Invalid score»

6. Ввод недопустимого признака поля.

На входе: дробное число или отрицательное.

На выходе: сообщение «Invalid year»

**Описание структуры данных**

Для хранения таблицы используется структура, содержащая структуры с информацией о студенте и информацией о ключе.

**typedef struct table**

**{**

**students\_t student;**

**keys\_t key;**

**}table\_t;**

Тип **students\_t** используется для хранения информации об одном студенте (информация согласно ТЗ)

**typedef struct students**

**{**

**char surname[100];**

**char name[100];**

**int group;**

**int gender;**

**int age;**

**double avg\_score;**

**int year;**

**int type;**

**housing\_t type\_housing;**

**}students\_t;**

*Поля структуры:*

* **char surname[100] –** фамилия студента;
* **char name[100] —** имя студента;
* **int group –** номер группы;
* **int gender –** пол (М/Ж);
* **int age –** возраст;
* **double avg\_score —** средняя оценка за сессию;
* **int year –** год поступления.
* **int type –** тип жилья
* **housing\_t type\_housing –** описания типа жилья.

Тип **key\_t** используется как таблица ключей:

**typedef struct keys**

**{**

**int age;**

**int id;**

**}keys\_t;**

*Поля структуры:*

* **int age –** возраст
* **int id –** номер ячейки в основной таблице**;**

Тип **housing** **\_t** является вариантной частью:

**typedef union housing**

**{**

**flat\_t flat;**

**dormitory\_t dormitory;**

**}housing\_t;**

* **flat –** тип квартира
* **dormitory** – тип общежитие

Тип для описания квартиры

**typedef struct type\_flat**

**{**

**char street[WORD\_LEN];**

**int house;**

**int room;**

**}flat\_t;**

*Поля структуры:*

* **char street[100]; –** название улицы;
* **int house –** номер дома;
* **int room —** номер квартиры;

Тип для описания общежития

**typedef struct type\_dormitory**

**{**

**int dormitory;**

**int room;**

**}dormitory\_t;**

*Поля структуры:*

* **int dormitory;—** номер общежития
* **int room –** номер комнаты

**Описание алгоритма**

1. Пользователь вводит номер команды, предложенные в меню.
2. Пока пользователь не введет 0 (выход из программы), ему будет предложено вводить номера команд и выполнять действия по выбору.

**Набор тестов**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название теста** | **Пользовательский ввод** | **Результат** |
| 1 | Некорректный ввод команды | 45 | Invalid menu code |
| 2 | Некорректный ввод из файла (файл пустой) | data.txta | Sorry, but file is empty |
| 3 | Добавление записи при максимальном размере таблицы | 2 (Ввод команды на добавление записи) | Too bid data |
| 4 | Некорректный ввод для года | -200 | Invalid year |
| 5 | Некорректный ввод численного поля (отрицательное) | -1.1 | Invalid score |
| 6 | Ввод записей из верного файла (команда 1) | Ввод команды 1 | Таблица заполнена |
| 7 | Добавление верной записи в конец файла (команда 2) | Введена команда 2 и Запись введена верно | Запись добавлена в конец файла |
| 8 | Сортировка таблицы ключей (команда 4) | Команда введена до заполнения таблицы. | At first make the 1 or 2 command |
| 9 | Сортировка таблицы ключей (команда 4) | Команда введена верно. | Таблица ключей отсортирована. |
| 10 | Сортировка таблицы (команда 5) | Команда введена до заполнения таблицы. | At first make the 1 or 2 command |
| 11 | Сортировка таблицы (команда 5) | Команда введена верно, но при выборе сортировки введена неверная цифра (не 1 или 2) | At first make the 1 or 2 command |
| 12 | Сортировка таблицы (команда 5) | Команда введена верно. | Исходная таблица отсортирована. |
| 13 | Вывод таблицы по отсортированной таблицы ключей. (команда 6) | Команда введена до заполнения таблицы. | At first make the 1 or 2 command |
| 14 | Вывод таблицы по отсортированной таблицы ключей. (команда 6) | Команда введена верно. | Таблица выводится в отсортированном виде, причем сама не сортируется |
| 15 | Сравнение сортировок (команда 7) | Команда введена верно. | Выводятся замеры времени различным видов сортировок. |
| 16 | Удаление строки | 3.5 | Из таблицы удалены 10 студентов |
| 17 | Вывод таблицы на экран (команда 9) | Команда введена до заполнения таблицы. | At first make the 1 or 2 command |
| 18 | Вывод таблицы на экран (команда 9) | Команда введена верно. | Таблица выведена на экран. |
| 19 | Выход (команда 0) | Команда введена верно. | Произведен выход. |

# 

**Ответы на контрольные вопросы**

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

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

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

При компиляции тип данных в вариантной части не проверяется. Поведение будет неопределенным из-за того, что невозможно корректно считать данные.

*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 представляет собой дополнительный массив (структура), содержащий индекс исходного элемента в исходной таблице и выбранный ключ.

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

Когда мы сортируем таблицу ключей, мы экономим время, так как перестановка записей в основной таблице (которая может содержать большой количество полей) отсутствует. Минус данного подхода в том, что для размещения таблицы ключей требуется дополнительная память. Кроме того, если использовать в качестве ключа символьное поле, то необходимо будет дополнительно обрабатывать данное поле в цикле, что увеличивает время выполнения, так же выбор данных из основной таблицы в порядке, определенном таблицей ключей, замедляет вывод. Если исходная таблица содержит небольшие число полей, то выгоднее обрабатывать данные в самой таблице.

*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.

**Сравнение сортировок на списке из 2000 студентов**

* Сортировка исходной таблицы:
  + Быстрая сортировка:
    - Время: 4669 мкс
    - Память: 653600 байт
  + Сортировка пузырьком
    - Время: 48010 мкс
    - Память: 653600 байт
* Сортировка таблицы ключей:
  + Быстрая сортировка:
    - Время: 0 мкс
    - Память: 668800 байт
  + Сортировка пузырьком:
    - Время: 8003 мкс
    - Память: 668800 байт

Вывод

При работе с данными разного типа часто нужно пользоваться структурой с вариантной записью. Такой подход позволяет создавать намного более гибкие структуры данных.

При обработке больших данных по определенному признаку полезно создавать отдельный массив ключей для оптимизации времени выполнения. Создание такого массива увеличивает затраты памяти, но зато существенно снижает время сортировки. Следовательно, такой подход необходимо использовать при сортировки таблиц очень большого объема.