*Государственное образовательное учреждение высшего профессионального образования*

*«Московский Государственный Технический Университет имени Н. Э. Баумана»*

ОТЧЕТ

По лабораторной работе №2

По курсу «Типы и структуры данных»

Тема: Вариантные записи

Вариант №6

|  |  |
| --- | --- |
| Студент: | Щербатюк Д.С. |
| Группа: | ИУ7-34 |
| Преподаватель: | Силантьева А. В. |

Москва, 2016

**Техническое задание**

**Задача**: 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: а) саму таблицу, б) массив ключей (возможность добавления и удаления записей в ручном режиме обязательна).

Ввести список квартир, содержащий адрес, общую площадь, количество комнат, этаж, стоимость квадратного метра (первый – есть ли подвал; последний – наличие чердака, его площадь и возможность приватизации; другой – наличие балкона и его площадь). Найти все квартиры, с возможностью приватизации чердака.

**Входные данные**: файл, содержащий список квартир и информацию о них; возможно добавление квартир в список вручную.

**Выходные данные**: результат поиска квартир, с возможностью приватизации чердака; отсортированная по этажам таблица; время на сортировку и поиск.

**Функция программы**: реализация поиска в таблице, сортировка как исходной таблицы, так и таблицы ключей.

**Аварийные ситуации**

1) Файл не существует.

Будет выведено сообщение об ошибке.

2) Файл повреждён или содержит некорректные данные.

Будет выведено соответствующее сообщение, с указанием типа ошибки.

3) Попытка добавить квартиру без указания площади\комнат\цены.

Будет выведено соответствующее сообщение.

4) Поиск или сортировка несформированной таблицы.

Будет выведено сообщение об ошибке.

**Используемые структуры**

struct balcony //структура, отвечающая за хар-ки балкона

{

char isB;

float squareB;

};

struct cellar //структура, отвечающая за хар-ки подвал

{

char isC;

};

struct attic //структура, отвечающая за хар-ки чердака

{

char isA;

float squareA;

char isPrivate;

};

struct flat { //структура квартиры

int index;

char address[20];

float square;

int rooms;

float price;

int floor;

union

{

struct cellar cell;

struct balcony balc;

struct attic att;

} ad;

};

struct key //структура одного ключа из таблицы

{

int index;

int floor;

};

**Функции и алгоритмы**

**int read(FILE \*f, struct flat table[], struct key key[]);**

Чтение таблицы из файла “in.txt” и заполнение таблицы ключей.

**void private\_attic(struct flat table[]);**

Определяет, возможна ли приватизация чердака. Проходится по всем строкам в таблице и смотрит значение параметра .ad.att.isPrivate, если он равен ‘y’,то выводит строку в таблицу.

**int add(struct flat \*table, struct key \*key);**

Добавляет еще одну квартиру в конец, спрашивает дополнительную информацию в зависимости от этажа.

**int del(struct flat \*table, struct key \*key);**

Удаляет элемент, указанный пользователем по номеру в исходной таблице.

**void print(struct flat table[], struct key key[]);**

Печатает таблицу квартир и таблицу ключей.

**void shell\_key (struct key key[]);**

Сортирует таблицу ключей сортировкой Шелла.

**void shell\_table (struct flat table[]);**

Сортирует таблицу квартир сортировкой Шелла.

**void insert\_key(struct key key[]);**

Сортирует таблицу ключей сортировкой вставками.

**void insert\_table(struct flat table[]);**

Сортирует таблицу квартир сортировкой вставками.

**void print\_table(struct flat table[]);**

Печатает таблицу квартир.

**void print\_key(struct key key[]);**

Печатает таблицу ключей.

**Тесты**

1. **Открытие несуществующего файла**



1. **Ошибка чтения файла/Некорроектные данные**



1. **Этаж указан больше максимального.**



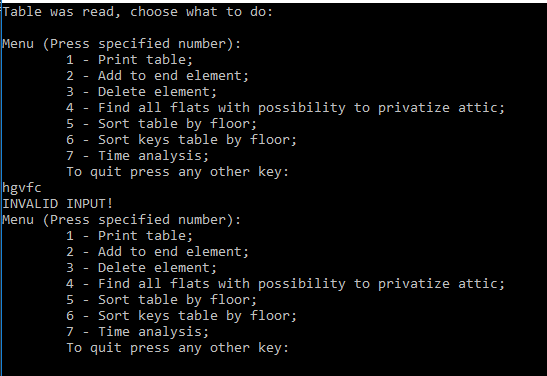
1. **Неполные данные**



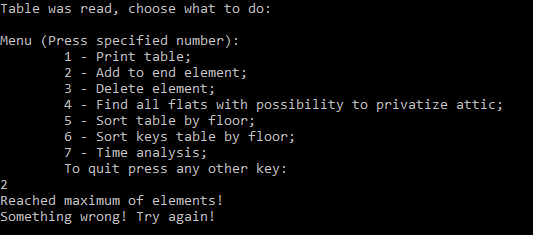
1. **Вывод времени сортировки прежде самой сортировки.**



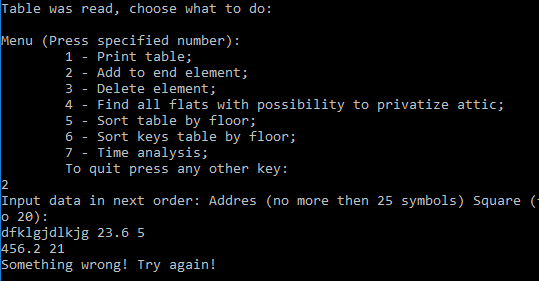
1. **Неверный ввод**



1. **Добавление записи в заполненную таблицу**



1. **Неверные данные при добавлении.**



**Сравнение времени**

SHELL for 10000 elements.



Сортировка таблицы ключей в случае сортировки Шелла на 86% эффективнее, однако требует некоторых затрат памяти (см. ниже). Рассмотрим сортировку вставками.

INSERT for 10000 elements.



В этом случае таблица ключей оказывается на 63% эффективнее и данная тенденция сохраняется.

Вот затраты по памяти для заполненной таблицы из 40 строк.



Контрольные вопросы

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

В С вариантная часть записи определяется с помощью union; в Delphi с помощью case <переменная> of. 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 заблокирована. Поля в вариантной части записи расположены в одной области памяти – если данные были записаны туда согласно одному описанию, а обрабатываться будут согласно другому, неизбежно будут использоваться неправильные значения (например, в качестве 1-байтового char будет использоваться первый байт от double).

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

Тип данных в вариантной части при компиляции не проверяется, контроль за правильностью ее использования 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 собой массив, каждый элемент которого содержит индекс элемента из таблицы. При сортировке происходит перестановка строк таблицы ключей, а не строк исходной таблицы.

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

При сортировке таблицы ключей экономится время - исходная таблица содержит бОльшее число полей, и на перестановку их значений не тратится время. Этот выигрыш во времени особенно заметен при большой размерности таблиц и при правильно подобранных ключах. Тем не менее, для размещения таблицы ключей требуется дополнительная память. Кроме того, выигрыш во времени может быть меньше, если сортировка идёт по ключу типа «строка».

*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. Таблицы могут занимать чрезвычайно большие объёмы памяти (например, крупные базы данных, или таблицы изображений) – может оказаться выгоднее алгоритм сортировки, занимающий большее время, но при этом требующий меньшее количество дополнительной памяти.