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 **ПРОГРАММНАЯ ИНЖЕНЕРИЯ (ИУ7)**

О Т Ч Е Т

По лабораторной работе № 2

Название: Записи с вариантами. Обработка таблиц

Дисциплина: Типы и структуры данных

Студент ИУ7-31Б (Группа) Савинова М. Г. (Фамилия И. О.)

Преподаватель Барышникова М. Ю.

Москва, 2022

Оглавление

Цель работы

Приобрести навыки работы с типом данных «запись» («структура») содержащим вариантную часть, и с данными, хранящимися в таблицах. Оценить относительную эффективность программы (в процентах) по времени и по используемому объему памяти в зависимости от используемого алгоритма и от объема сортируемой информации

Условие 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 – любое невариантное поле (по выбору программиста), используя:

а) саму таблицу

б) массив ключей.

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

Ввести список квартир, содержащий адрес, общую площадь, количество комнат, стоимость квадратного метра, первичное жилье или нет (первичное – с отделкой или без нее; вторичное – время постройки, количество предыдущих собственников, количество последних жильцов, были ли животные). Найти все вторичное 2-х комнатное жилье в указанном ценовом диапазоне без животных.

Описание ТЗ

1. Описание входных данных

Исходными данными является структурированная информация о квартире

Каждая запись содержит:

1) Адрес (название улицы, номер дома, этажа, квартиры)
2) Общую площадь
3) Количество комнат
4) Стоимость квадратного метра
5) Тип жилья (первичное/вторичное)
6) Наличие отделки (если первичное жилье)
7) Год постройки (если вторичное жилье)
8) Кол-во предыдущих собственников (если вторичное жилье)
9) Кол-во последних жильцов (если вторичное жилье)
10) Наличие животных в прошлом (если вторичное жилье)

Ввод данных осуществляется с помощью «1» и «4» пунктов меню

Некорректный ввод вызывает завершение программы с ненулевым кодом возврата.

Ограничения:

- Адрес:
  - **Улица:** не содержит цифр, максимальная длина — **20 символов**
  - **Дом:** целое положительное число длиной до **5 цифр**
  - **Этаж:** целое положительное число длиной до **5 цифр**
  - **Квартира:** целое положительное число длиной до **5 цифр**
- Общая площадь: вещественное положительное число длиной до **8 цифр**
- Количество комнат: целое положительное число длиной до **5 цифр**
- Стоимость квадратного метра: вещественное положительное число длиной до **8 цифр**
- Тип жилья: строка *«primary»* или *«secondary»*
- Наличие отделки: символ *«y»* или *«n»*
- Год постройки: целое положительное число длиной до **5 цифр**
- Количество бывших владельцев: целое положительное число длиной до **5 цифр**
- Количество последних жильцов: целое положительное число длиной до **5 цифр**
- Наличие животных в прошлом: символ *«y»* или *«n»*

2. Описание выходных данных

В результате выполнения программы будет сформирована база данных с возможными действиями:

```
1.  Read input file
2.  Output data_table
3.  Output keys_table
4.  Delete record by field "street name"
5.  Add new record
6.  Find record by "precence of trim"
7.  Sort data_table(quick sort)
8.  Sort keys_table(quick sort)
9.  Sort data_table(slow sort)
10. Sort keys_table(slow sort)
11. Output whole data by keys_table
12. Output efficiency_table
13. Output list of secondary 2 rooms flats with pets in
the cost range
14. Write in new file
15. Exit
```

3. Описание задачи, реализуемой в программе

Задача программы состоит из нескольких частей:

1. Работа с данными: добавление в таблицу (из файла или со стандартного потока ввода) и удаление из таблицы;
2. Форматированный вывод основной таблицы, таблицы ключей;
3. Сортировка данных в таблице двумя способами: напрямую и через таблицу ключей;
4. Сравнение эффективности различных методов сортировки таблицы;

4. Способ обращения к программе

Обращение к программе происходит через консоль, путём запуска *.exe файла.

5. Описание возможных аварийных ситуаций и ошибок пользователя

Аварийная ситуация	Код завершения	Сообщение
Неверно выбранный пункт меню	1	`Wrong choice!`
Неверное кол-во аргументов командной строки	2	`Wrong number of arguments!`
Нет входного файла	3	`No avaliable input file!`
Входной файл пустой	4	`Empty input file!`
Пустая исходная таблица	5	`Not avaliable data!`
Некорректно введенные данные	6	`Incorrect input!`
Не найдены необходимые записи при поиске	7	`No avaliable fields :(` `Empty list :(`
Переполнение таблицы	8	`No avaliable space for new structure :(`
Не найдены записи для удаления	9	`Nothing to find :(`

Описание внутренних структур данных

- Структура с данными о квартирах

```
typedef struct
{
```

```
    adress_t adress;      // адрес
    double whole_squares; // кол-во квадратных метров
    int count_rooms;      // кол-во комнат
    double cost_square_m; // стоимость квадратного метра
    housing_kind_t kind;  // тип жилья
    housing_t housing;
} flat_t;
```

- Структура с ключевыми данными

```
typedef struct
{
    int index;       // индекс в основной таблице
    int count_rooms; // кол-во комнат
} keys_t;
```

- Структура, содержащая адрес

```
typedef struct
{
    char street[MAX_STREET]; // имя улицы
    int house;               // номер дома
    int floor;               // номер этажа
    int flat;                // номер квартиры
} adress_t;
```

- Структура для первичного жилья

```
typedef struct
{
    char trim; // наличие отделки
} primary_t;
```

- Структура для вторичного жилья

```
typedef struct
{
    int year_constr;         // год постройки
    int count_former_owners; // кол-во бывших владельцев
    int count_last_roomers;  // кол-во последних жильцов
    char pets;               // были ли животные
} secondary_t;
```

- Перечисляемая переменная для определения типа жилья

```
typedef enum
{
    primary = 1,  // первичное жилье
    secondary = 2 // вторичное жилье
```

```
} housing_kind_t;
```

- Структура, объединяющая данные о типах жилья

```
typedef union // типы жилья
{
    primary_t primary;
    secondary_t secondary;
} housing_t;
```

Тестовые данные

1. Позитивные тесты

1. Сортировка одинаковых структур
2. Сортировка отсортированных структур
3. Сортировка одной структуры
4. Добавление структуры с типом «первичное» жилье
5. Добавление структуры с типом «вторичное» жилье
6. Удаление единственной структуры
7. Поиск квартиры с «отделкой»
8. Поиск квартиры без отделки
9. Запись отсортированной таблицы в выходной файл
10. Вывод неотсортированной таблицы по отсортированной таблице

2. Негативные тесты

1. Неверное кол-во аргументов командной строки

```
./app.exe
Wrong number of arguments!
```

2. Входной файл не существует

```
./app.exe 1.txt 2.txt
No avaliable input file!
```

3. Входной файл пустой

```
./app.exe data.txt 2.txt
Empty input file!
```

4. Добавление записи в переполненную таблицу

```
Input option: 4
No avaliable space for new structure :(
```

5. Сортировка пустой таблицы

```
Input option: 7
Not avaliable data!
```

6. Сортировка пустой таблицы ключей

```
Input option: 8
Not avaliable data!
```

7. Удаление записи в пустой таблице

```
Input option: 5
Not avaliable data!
```

8. Поиск записи в пустой таблице

```
Input option: 6
Not avaliable data!
```

9. Вывод пустой таблицы

```
Input option: 2
Not avaliable data!
```

10. Вывод пустой таблицы ключей

```
Input option: 3
Not avaliable data!
```

11. Отсутствие записей, удовлетворяющих поиску

```
Input option: 13
Costs from: 1
To: 2
Empty list :(
```

```
Input option: 6
Input precence of trim(y/n): n
No avaliable fields :(
```

12. Неверно введенные данные при добавление записи

```
Input option: 4

*Maximim length: 20
Input street name: 2022
Wrong street name!
```

```
*Integer positive value(1, 99999)
Input number of house: -10
Wrong house number!
```

```
*Double positive value(1.0, 99999.9)
Input square meters: 0.9999999
Wrong square meters!
```

13. Неверно введенное значения для поиска по полю

```
Input option: 6
Input precence of trim(y/n): a
Wrong value for presence of trim!
```

Оценка эффективности

Были проведены тесты с таблицами на 10, 100, 500, 1000, 1500 записей. Для каждой размерности таблица перезаполнялась заново во избежание сортировки уже отсортированной таблицы

```
Input option: 12

*Measurements of time in microseconds
*Measurements of memory in bytes

//----------------------/---------/---------/---------/---------/---------/---------//
//          Size        /      10 /     100 /     500 /    1000 /    1500 / Average //
//----------------------/---------/---------/---------/---------/---------/---------//
//Bubble sort: data_table/       7 /     709 /   16121 /   66062 /  156166 /   47813 //
//----------------------/---------/---------/---------/---------/---------/---------//
//Bubble sort: key_table /       2 /     160 /    4402 /   17702 /   41052 /   12663 //
//----------------------/---------/---------/---------/---------/---------/---------//
//Quick sort: data_table /       3 /      37 /     111 /     260 /     359 /     154 //
//----------------------/---------/---------/---------/---------/---------/---------//
//Quick sort: key_table  /       1 /      15 /      89 /     213 /     300 /     123 //
//----------------------/---------/---------/---------/---------/---------/---------//
//Memory of data_table   /     800 /    8000 /   40000 /   80000 /  120000 /---------//
//----------------------/---------/---------/---------/---------/---------/---------//
//Memory of key_table    /     880 /    8800 /   44000 /   88000 /  132000 /---------//
//----------------------/---------/---------/---------/---------/---------/---------//
```

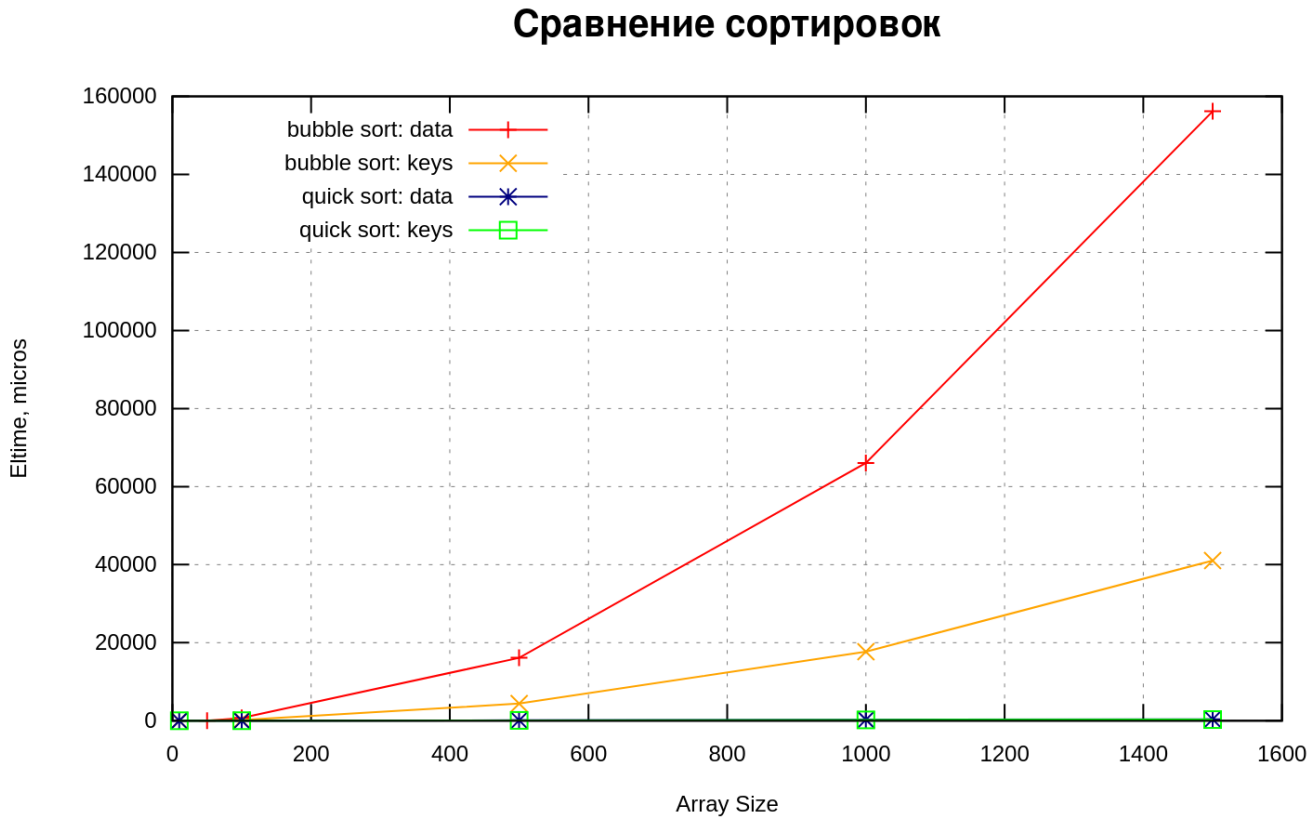
Эффективность по времени:

«Быстрая» сортировка главной таблицы быстрее сортировки модифицированным «пузырьком» в среднем в 310 раз при сортировке основной таблицы, и в 102 раза при сортировке таблицы ключей.

Эффективность по памяти:

Сортировка с таблицей ключей занимает на 10% больше памяти, чем работа без неё.

Асимптотика быстрой сортировки – **O(n*log(n))**
Асимптотика сортировки вставками – **O(n^2)**



Выводы

В ходе выполнения данной работы я ознакомилась с основными принципами реализации записей с вариантами и способами обработки таблиц.

Затраты по памяти при использовании таблиц оказались на 10% больше, но выигрыш по времени оказался в среднем в 200 раз меньше.

*Ответы на контрольные вопросы

1.Как выделяется память под вариантную часть записи?

Особенность этого состоит в том, что все вариантные поля разделяют одну и ту же область памяти. Размер этой области выбирается компилятором по размеру наибольшего вариантного поля.

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

Некорректное отображение данных

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

Программисту необходимо следить за правильностью выполнения

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

При больших размерах таблиц поиск данных и их сортировка может потребовать больших затрат времени. В этом случае можно уменьшить время обработки за счет создания дополнительного массива – таблицы ключей, содержащей индекс элемента в исходной таблице и выбранный ключ.

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

Если мы сортируем таблицу ключей, то экономится время, поскольку перестановка записей в исходной таблице, которая иногда может содержать достаточно большое число полей, отсутствует. Этот выигрыш во времени особенно заметен при большой размерности таблиц и при правильно подобранных ключах. Но для размещения таблицы ключей требуется дополнительная память. А так же, если в качестве ключа используется символьное поле записи, то это влечет за собой необходимость посимвольной обработки данного поля в цикле, и, следовательно, приводит к увеличению времени выполнения любых операций.

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

Предпочтительнее устойчивые, то есть те, которые не переставляют уже отсортированные элементы, так как перестановка элементов таблицы занимает много времени. Например, сортировка пузырька с флагом или метод сортировки вставками.