|  |  |
| --- | --- |
| 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 **ПРОГРАММНАЯ ИНЖЕНЕРИЯ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04** Программная инженерия

**Отчет**

|  |  |
| --- | --- |
| **По лабораторной работе №** | 1 |

**Название:**

Записи с вариантами, обработка таблиц

**Дисциплина:** Типы и структуры данных

|  |  |  |  |
| --- | --- | --- | --- |
| Студент | ИУ7-31Б |  | Н.Ю. Баринов |
|  | (Группа) |  | (И.О. Фамилия) |
|  |  |  |  |

Москва, 2021

**Описание условия задачи**

Имеются описания: Type жилье = (дом, общежитие); Данные: Фамилия, имя, группа, пол (м, ж), возраст, средний балл за сессию, дата поступления адрес: дом: (улица, №дома, №кв); общежитие: (№общ., №комн.); ввести общий список студентов. Вывести список студентов, указанного года поступления, живущих в общежитии.

**Описание ТЗ**

1. **Описание исходных данных**

Исходными данными является структурированная информация о человеке.

Каждая запись содержит:

1. Фамилия
2. Имя
3. Группа
4. Пол
5. Возраст
6. Средний балл за сессию
7. Дата поступления
8. Тип
9. Улица (если тип - дом)
10. № дома (если тип - дом)
11. № квартиры (если тип - дом)
12. № общежития (если тип - общежитие)
13. № комнаты (если тип - общежитие)

Ввод данных происходит с помощью выбора 1 и 4 пунктов в меню.

Некорректный ввод вызывает завершение программы с ненулевым кодом возврата.

**- Имя не должно содержать цифр. Максимальная длина имени – 30**

**- Фамилия не должна содержать цифр. Максимальная длина фамилии – 30**

**- Группа вводится произвольно. Максимальная длина группы – 9**

**- Пол может быть только m – male или f – female**

**- Возраст должен быть положительным**

**- Средний балл за сессию должен быть больше 0.0 и меньше или равен 5.0**

**- Дата поступления вводится в формате дд.мм.гггг. При этом дата должна быть корректной.**

**- Тип вводится следующим образом:**

home – дом

coliving – общежитие

Ввод недопустимых значений при выборе типа вызывает выход из программы с ненулевым кодом возврата

**- Улица вводится произвольном порядке. Максимальная длина адреса – 30**

**- Номер дома вводится в произвольном порядке. Максимальная длина – 5**

**- Номер квартиры, номер общежития и номер комнаты должны быть целым положительным числом.**

Во всех случаях, когда происходит некорректный ввод, программа завершается с ненулевым кодом возврата.

1. **Описание результатов программы**

Результатом программы будет сформированная база данных с возможными действиями:

0 - Выйти из программы

1 - Прочитать файл

2 - Вывести данные

3 - Вывести таблицу ключей

4 - Добавить запись

5 - Удалить запись

6 - Отсортировать записи по оценке(быстрая, исходная таблица)

7 - Отсортировать записи по оценке(медленная исходная таблица)

8 - Отсортировать таблицу ключей(быстрая)

9 - Отсортировать таблицу ключей(медленная)

10 - Вывести данные по таблице ключей

11 - Вывод таблицы эффективности

12 - Вывести список студентов, указанного года поступления, живущих в общежитии.

**Описание задачи, реализуемой в программе**

Программа реализует обработку массива структур, используя таблицу.

**Способ обращения к программе**

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

исполняемый файл имеет расширение .exe, что предусматривает запуск программе в

ОС Windows

**Описание возможных аварийных ситуаций и ошибок пользователя**

Часть ошибочных ситуаций описана в заголовке Описание исходных данных

ARG\_ERROR 1 – неправильно введены аргументы вызова

OPEN\_FILE\_ERROR 2 – не удалось открыть файл

CHOICE\_ERROR 3 – неправильно выбран пункт меню

EMPTY\_FILE\_ERROR 4 – пустой файл

INPUT\_ERROR 5 – некорректно введены данные

NOT\_FOUND\_ERROR 6 – не нашлось записи для удаления

OVERFLOW\_ERROR 7 – превышение допустимого количества структур

**Описание внутренних структур данных**

Программа содержит в себе структуру, представляющую информацию о человеке

typedef struct

{

    char surname[MAXNAMELEN];

    char name[MAXNAMELEN];

    char group[MAXGROUPLEN];

    gender\_t gen;

    unsigned short int age;

    float score;

    char date[MAXDATELEN];

    live\_kind\_t kind;

    adress\_t adress;

} student\_t;

***Эта структура содердит в себе:***

typedef enum

{

    male = 0,

    female = 1

} gender\_t;

typedef enum

{

    rich = 0,

    poor = 1

} live\_kind\_t;

typedef union

{

    home\_t home;

    coliving\_t coliving;

}adress\_t;

Также есть структура для записи информации о студенте в таблицу ключей

typedef struct

{

    unsigned int position;

    float score;

}key\_t;

**Тесты**

**Положительные тесты**

1. Сортировка одинаковых структур
2. Сортировка отсортированных структур
3. Сортировка одной структуры
4. Добавление структуры с типом «дом»
5. Добавление структуры с типом «общежитие»
6. Удаление единственной структуры

**Негативные тесты**

1. Некорректный ввод имени структуры
2. Некорректный ввод фамилии структуры
3. Некорректный ввод даты поступления
4. Некорректный ввод пола
5. Некорректный ввод типа структуры
6. Некорректный ввод оценки
7. Некорректный ввод возраста
8. Удаления структуры из пустого списка структур
9. Некорректный ввод выбора действий
10. Попытка ввода более 30 символов в поле «Имя»
11. Попытка ввода более 30 символов в поле «Фамилия»
12. Попытка ввода более 9 символов в поле «Группа»
13. Попытка ввода более 30 символов в поле «Улица»
14. Попытка отсортировать пустой список структур
15. Попытка вывести пустой список структур

**Временная эффективность и затраты памяти**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | qsort | insert sort | qsort с таблицей ключей | insert sort с таблицей ключей |
| 3 | 0,00 | 3,00 | 0,00 | 0,00 |
| 10 | 0,00 | 24,00 | 0,00 | 1,00 |
| 100 | 13,00 | 1675,00 | 9,00 | 106,00 |
| 1000 | 175,00 | 151193,00 | 148,00 | 8854,00 |

Для всех тестов было взято среднее время(мс) из 100 изменений.

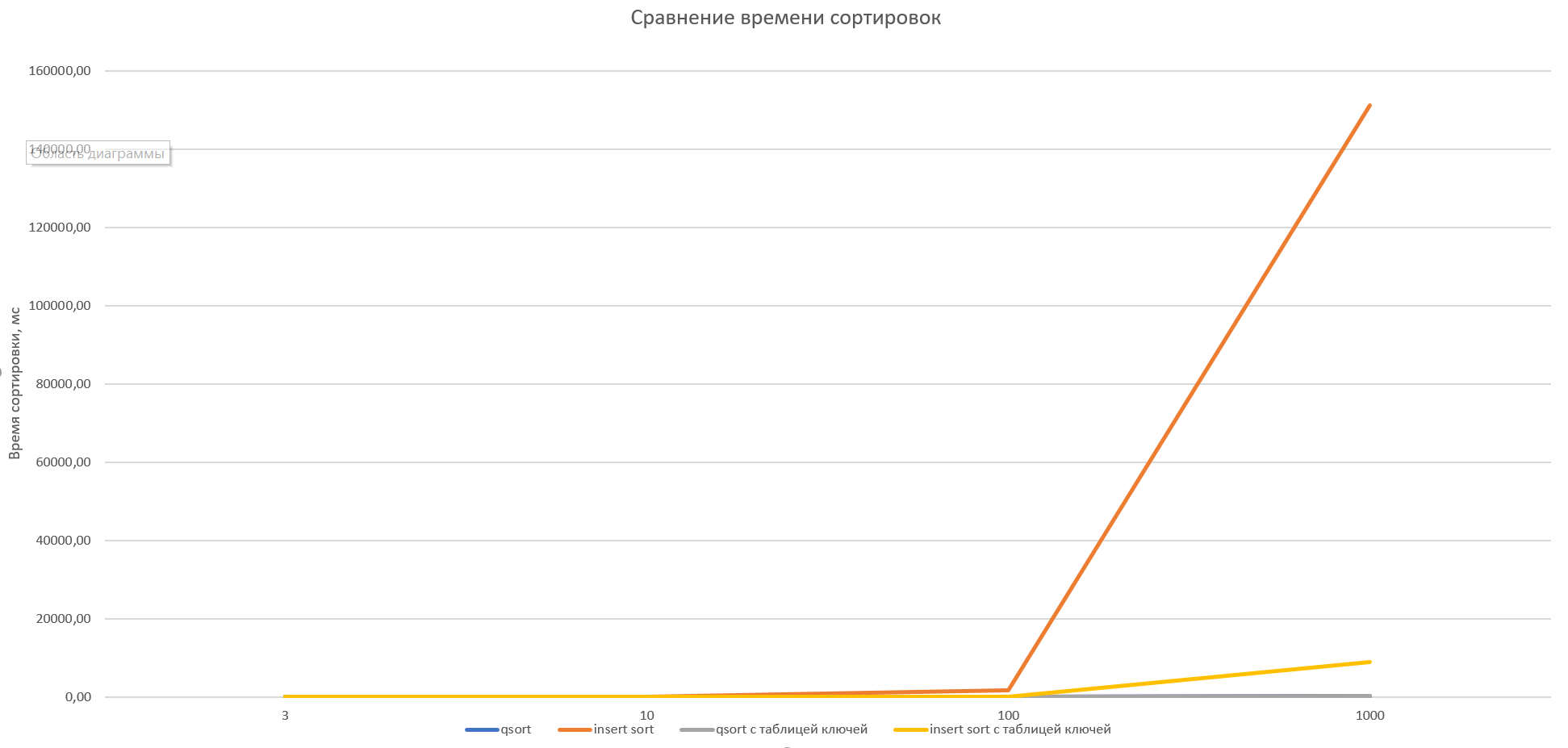
Можно увидеть, что время быстрой сортировки возрастает на 93% (в среднем) и время сортировки вставками – на 99% (в среднем). Использовал формулу – (t1 – t2)/t1 \* 100%

Сравнение использования и неиспользования таблицы ключей:

Посчитал относительное изменение времени работы при использовании таблицы ключей по формуле: (t1 – t2)/t1 \* 100%. Среднее изменение – 55%

Асимптотика быстрой сортировки – O(n\*log(n))

Асимптотика сортировки вставками – O(n^2)



**Затраты по памяти**

При сортировке не используя таблицу ключей:

MAX\_STUDENT\_COUNT \* sizeof(student\_t) = 1000\*140 = 140000 байт

При сортировке используя таблицу ключей:

MAX\_STUDENT\_COUNT \* sizeof(student\_t) + MAX\_STUDENT\_COUNT\*sizeof(key\_t) = 1000\*140 + 1000\*8 = 140000 + 8000 = 148000 байт

**Вывод**

В процессе выполнения данной лабораторной работы я ознакомился с принципами реализации записей с вариантами и способами обработки таблиц.  
Периодически могут возникнуть такие ситуации, когда программисту необходимо будет выбрать наиболее эффективный способ обработки данных вариантных записей.

Затраты по памяти при использовании таблиц оказались на 5% затратнее, но выигрыш по времени оказался на 55% выгоднее.

**Ответы на контрольные вопросы**

1.Как выделяется память под вариативную часть записи?

Объем памяти, необходимый для записи с вариантами складывается из длин полей фиксированной части и максимального по длине поля вариантной части.

Например:

Union test{

int a;

int b;

double c;

}

Под данное объединение будет выделено 8 байт (размер double).

2.Что будет, если в вариативную часть ввести данные, несоответствующие с описанными?

В таком случае данные будут отображаться некорректно.

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

За правильностью выполнения операций над вариативной частью должен следить программист.

4.Что представляет из себя таблица ключей и зачем она нужна?

Таблица ключей представляет из себя структуры, либо массив, каждый элемент которого содержит значение какого-либо ключа (везде одного) и позицию из исходной таблицы.

Таблица ключей нужна повышения эффективности работы с таблицей.

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

В случае, если нам часто приходится сортировать таблицу с большим количеством полей, следует использовать таблицу ключей.

Если же полей немного, то можно обойтись и без выделения дополнительной памяти для таблицы ключей.

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

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