Отчет по 2 лабораторной работе
По дисциплине «Типы и структуры данных»

Подготовил Пересторонин Павел
Группа ИУ7-33Б
Вариант 15

07.09.2019

1. Цель работы.

Цель работы: приобрести навыки работы с типом данных «запись» (структура), содержащим вариантную часть (объединение, смесь), и с данными, хранящимися в таблицах, произвести сравнительный анализ реализации алгоритмов сортировки и поиска информации в таблицах, при использовании записей с большим числом полей, и тех же алгоритмов, при использовании таблицы ключей; оценить эффективность программы по времени и по используемому объему памяти при использовании различных структур и эффективность использования различных алгоритмов сортировок.

2. Техническое задание.

Введение.

Наименование программы: Записи с вариантами. Обработка таблиц.
Характеристика области применения: Применяется в учебных целях в МГТУ им. Н.Э.Баумана.
Сроки исполнения: 02.09.2019 – 16.09.2019 (2 недели)

Основание для разработки.

Заказчик: Силантьева Александра Васильевна (Преподаватель по дисциплине “Типы и Структуры Данных” в МГТУ им. Н.Э.Баумана).
Исполнитель: Пересторонин Павел (студент группы ИУ7-33Б МГТУ им. Баумана).
Основание: Учебный процесс.

Назначение разработки.

Общая концепция системы: Пользователи – студенты и (-или) сотрудники МГТУ им. Н.Э.Баумана.

Функциональное назначение: Программа предоставляет возможность вести информационную таблицу о студентах 2 типов: проживающих в общежитии или живущих дома.

Требования к программе.

1) Указание формата и диапазона данных при вводе и (или) добавлении записей;

2) Указание операций производимых программой;

3) Наличие пояснений при выводе результата;

4) Возможность добавления записей в конец таблицы и удаления записи по значению указанного поля.

5) Просмотр **отсортированной** таблицы ключей при **несортированной** исходной таблице;

6) Вывод упорядоченной исходной таблицы;

7) Вывод исходной таблицы в упорядоченном **виде,** используя **упорядоченную** таблицу ключей

8) Вывод результатов сравнения эффективности работы программы при обработке данных в исходной таблице и в таблице ключей;

9) Вывод результатов использования различных алгоритмов сортировок.

Требования к функциональным характеристикам.

1) Все логически завершенные фрагменты алгоритма должны быть оформлены в виде подпрограмм.

Внешняя спецификация программы.

Тип входных данных: входные данные могут быть двух видов:
1) цифра, которая указывает на пункт меню, который необходимо выполнить.
2) входные данные подпрограммы, которая была выбрана из основного меню (либо поля студента, либо цифра, определяющая какой-то признак (например, поле для сортировки); подробности см. в описании алгоритма)

Тип выходных данных: 3 типа:
1) таблица со студентами
2) информационные данные (сравнение алгоритмов сортировки и т.п.)
3) данные о каком-либо действии (например, об успешном добавлении студента в список)
(подробный список входных и выходных данных см. в описании алгоритма)

Задачи программы: хранить таблицу студентов и производить с ней действия, запрошенные пользователем и определенные в пункте “Требования к программе” (2 вида сортировок таблицы, сортировка ключей, вывод по ключам, простой вывод таблицы, добавление/удаление студента, сравнение алгоритмов сортировок)

Способ обращения к программе: Запуск программы из папки хранения и ввод определенных в меню целых чисел для входа в определенный режим работы, а также следование выводимым на экран в этом режиме инструкциям.

Описание возможных аварийных ситуаций и ошибок пользователя.

1) Конец файла при вводе нового студента (Нажатие комбинации клавиш ctrl+D)

2) Некорректный файл при создании начальной таблицы (преднамеренное нарушение структуры имеющегося файла)

Структура данных.

Основная структура данных, используемая в программе – массив записей с вариативной частью (таблица)
Запись, в свою очередь, имеет следующие поля:
1. тип жилья, в котором живет студент (по-простому, это можно назвать “признаком” или “флажком” (2 варианта допустимо: общежитие или дом)
2. фамилия студента (статическая строка фиксированного размера)
3. имя студента (статическая строка фиксированного размера)
4. пол студента (тот же самый “признак” или “флажок” с 2 значениями)
5. средний балл по учебе (рациональное число от 1 до 5 точности 2)
6. год поступления (в моем случае установлены доп. Ограничения: так как программа пишется для использования в МГТУ им. Баумана, то минимальный год поступления = 1830 (год создания МГТУ им. Баумана), а максимальный год – 2019 (год написания программы))
7. вариативная часть:
7.1. Если студент живет в общежитии:
1) номер общежития (беззнаковое целое число)
2) номер комнаты (беззнаковое целое число)
7.2. Если студент живет в доме
1) Улица (статическая строка фиксированной длины)
2) Дом (беззнаковое число)
3) Квартира (беззнаковое число)

Минимальный возможный размер максимально упакованной записи студента:
1 байт (под 2 “флажка”)
~35 байт (2 статические строки символов: фамилия (20 символов) и имя (15 символов))
1 байт (под год, если уместить промежуток 1830-2019 в 0-189)
~19 байт (под вариативную часть: на общежитие нужно ~4 байта (номер общежития и номер комнаты в пределах [0;2^16] лежат), а на дом нужно то же, только со строкой под имя функции (15 символов в нашем случае))

~(40-50) байт/запись

Структура массива ключей такова: каждый ключ ссылается на индекс записи в таблице, таким образом представляет из себя простой целочисленный массив (причины такого выбора объясняются в алгоритме)

Алгоритм (полный алгоритм работы программы).
При запуске программы в первую очередь инициализируется таблица (считывается из заранее подготовленного или оставенного после работы предыдущего пользователя файла). Далее в бесконечном цикле пользователю выводится меню и предлагается выбрать один из пунктов, указанных там (всего пунктов 12 штук, функционал каждого будет описан ниже). В случае ввода пользователем некорректного значения программа предложит ввести нужный пункт еще раз до тех пор, пока он не будет введен корректно.
Предоставляемые в меню возможности:

0. Выход
Из названия очевидно, что введя “0” вы получите сообщение о выходе из программы и выйдете (в данном пункте высвобождаются все используемые ресурсы и заканчивается работа программы)

1. Вывод списка студентов группы.
Данная опция выводит список студентов, исходя из их номера строки в таблице. Список выводится путем вывода в ряд всех полей записи каждого студента (а так же индекса каждого студента)

2. Вывод таблицы в порядке ключей.
Данная опция также выводит список всех студентов, однако она опирается на список ключей, который изначально установлен так, что совпадает со списком записей (то есть ключ с индексом 1 указывает на запись с индексом 1, 2 на 2 и т.д.). Однако, если вы использовали сортировку с использованием ключей, то будет выводится другой список, ведь положение в списке ключей станет отличным от положения в таблице (которое не изменится)

3. Вывод студентов, поступивших в определенном году и живущих в общежитии.
Данная опция обеспечивает все тот же вывод списка студентов, но с наложением ограничения: пользователю предлагается ввести год, и в зависимости от введеного года ему будет выведен отфильтрованный список студентов.

4. Сортировка массива ключей.
Сортировка массива ключей проходит следующим образом: пользователю предлагается выбрать поле в записи, по которому он хочет сортировать (можно

выбрать любое, присущее всем студентам) и после этого начинается сортировка. Выбранное поле записи участвует в операции сравнения элементов и дает основную информацию о том, какие элементы нужно менять, а какие нет. Как я уже описал ранее, в моей программе массив ключей указывает на индексы записей в таблице и сделано это по следующим причинам: во-первых, если бы массив ключей указывал на строчный тип данных то и объем хранимых данных, и время доступа к элементу значительно бы увеличились. Таким образом становится ясно, что лучше пользоваться числами. Также весьма рационально использовать некоторое поле, которое можно рассматривать как некоторое индуктивное множество (некоторый набор уникальных элементов, как функция от натурального аргумента), однако, что очевидно, 100% уникальным будет только множество индексов студентов (подобно уникальному идентификатору). Ключи сортируются быстрой сортировкой, однако при обмене элементов они сравниваются и при равенстве не меняются, что добавляет лишнего времени, но делает сортировку устойчивой.
Еще один не обговоренный мною момент – зачем использовать массив ключей, если мы все равно опираемся на массив записей? Ответ прост: когда мы сравниваем элементы, мы считываем одно поле напрямую из записи по индексу, и не сильно теряем по времени (если вообще теряем) относительно ситуации, если бы данное поле считывалось из самого массива ключей (RAM, машина Фон-Неймана, все такое). Основная экономия времени просматривается в обмене элементами. Чтобы поменять местами 2 записи нужно переписать в буфер запись, размером 40-50 байт, вставить на ее место другую запись такого же размера и потом вставить первую запись на место второй. Таким образом имеем 3 операции записи с объемом данных 40-50 байт. В случае же, когда мы меняем местами ключи в таблице размером до 2^16, нужно перезаписать 3 раза всего по 2 байта информации, что значительно экономит время.
(из вышесказанного видно, что массив ключей вынуждает тратить всего на 4-5% больше памяти)

5. Сортировка таблицы пузырьком.
В данном случае сортируются сами записи. Пользователю так же предлагается выбрать поле, по которому требуется отсортировать данные. Данная сортировка – самая долгая из всех, что имеются в данной работе, это так же видно из сравнения алгоритмов сортировки (см. Пункт 9)

6. Быстрая сортировка таблицы.
Данный Алгоритм сортировки имеет меньшую среднюю сложность относительно алгоритма, описанного в пункте 5, однако вследствие того, что сортируются сами записи, метод (не алгоритм, потому что алгоритм в в пункте 4 такой же, но при этом обмен меняются не элементы таблицы, а ключи) сортировки все же медленнее, чем представленный в пункте 4.

7. Ввод нового студента.

Данная опция позволяет пользователю ввести данные о новом студенте и занести их в таблицу. Подробная информация о том, что и как вводить выдается программой.

8. Просмотр таблицы ключей.
Данная опция позволяет вывести таблицу в соответствии с таблицей ключей, а не по порядку индекса.
9. Вывод результатов сравнения сортировок.
Данная опция позволяет сравнить эффективности методов и алгоритмов сортировок. Выводит сравнения использованных в сессии программы алгоритмов и методов сортировок.

10. Удаление студента.
С помощью этого пункта можно удалить студента по ключу или по индексу (при выборе данного пункта, будут выведены соответствующие инструкции.

11. Выгрузка списка студентов из файла заново.
Данный пункт в меню позволяет выгрузить таблицу в том виде, в котором она была первоначально.

Тесты.

Тесты для главного меню:
1. Ввод символов.
2. Ввод числа, выходящего за предлагаемый диапазон (не из 0 – 11).
3. Ввод каждого числа из диапазона (0 – 11).

Тесты для вывода всех студентов:
1. В таблице нет студентов.
2. В таблице один студент.
3. В таблице 2+ студентов.

Тесты для вывода таблицы в порядке ключей:
1. В таблице нет студентов.
2. В таблице один студент.
3. В таблице 2+ студентов.

Тесты для вывода студентов, поступивших в определенном году и живущих в общежитии:
1. Ввод неверного года (не лежащего в диапазоне возможных годов поступления).
2. Ввод верного года.
3. Ввод года, в котором не было поступающих, поселившихся в общежитии.

Тесты для функции считывания признака сравнения:

1. Ввод символов.
2. Ввод числа не из требуемого диапазона (не из 1-8).
3. Ввод числа из требуемого диапазона (1-8).

Тесты для ввода нового студента:

1. Ввод типа места жительства (выбор из 2 вариантов, ввод цифры 1 или 2):

1.1. Ввод буквы.
1.2. Ввод неверной цифры.
1.3. Ввод верной цифры (1 или 2).

2. Ввод фамилии:

2.1. Ввод строки, содержащей некорректные символы (что-либо кроме букв и ‘-’).
2.2. Ввод строки больше 40 символов.
2.3. Ввод корректной строки (<40 символов и только корректные символы).

3. Ввод имени:

3.1. Ввод строки, содержащей некорректные символы (что-либо кроме букв и ‘-’).
3.2. Ввод строки больше 20 символов.
3.3. Ввод корректной строки (<20 символов и только корректные символы).

4. Ввод пола (выбор из 2 вариантов, ввод цифры 1 или 2):

4.1. Ввод буквы.
4.2. Ввод неверной цифры.
4.3. Ввод верной цифры (1 или 2).

5. Ввод среднего балла за сессию:

5.1. Ввод буквы.
5.2. Ввод некорректного числа (не в диапазоне 1-5).
5.3. Ввод числа из требуемого диапазона (целого/нецелого).

6. Ввод года поступления:

6.1. Ввод буквы.
6.2. Ввод числа не из диапазона (1830 – 2019)
6.3. Ввод числа из требуемого диапазона.

7. Ввод улицы:

7.1. Ввод строки, содержащей некорректные символы (что-либо кроме букв и ‘-’).
7.2. Ввод строки больше 25 символов.
7.3. Ввод корректной строки (<25 символов и только корректные символы).

8. Ввод номера дома:

8.1. Ввод буквы.
8.2. Ввод числа не из диапазона (1 - 10000)
8.3. Ввод числа из требуемого диапазона.

9. Ввод номера квартиры:

9.1. Ввод буквы.

9.2. Ввод числа не из диапазона (1 - 1000)
9.3. Ввод числа из требуемого диапазона.
10. Ввод номера общежития:
10.1. Ввод буквы.
10.2. Ввод числа не из диапазона (1 - 1000)
10.3. Ввод числа из требуемого диапазона.
11. Ввод номера комнаты:
11.1. Ввод буквы.
11.2. Ввод числа не из диапазона (1 - 1000)
11.3. Ввод числа из требуемого диапазона.

Тесты для функции просмотра таблицы ключей:
1. Пустая таблица.
2. 1 студент в таблице.
3. 2+ студентов в таблице.

Тесты для вывода сравнения алгоритмов сортировки:
1. Сортировок не было в программе.
2. Была 1 сортировка.
3. Было 2 сортировки.
4. Было 3 сортировки.

Тесты для удаления студента:
1. Список пустой.
2. Ввод неверного ответа в выборе метода удаления.
3. Ввод неверного ключа/индекса (слишком большого).
4. Ввод верных данных (первого, последнего и среднего студентов).

Итоги и расчеты.

Сравнение времени работы различных методов сортировки показано ниже (первый пример на таблице из 47 записей, второй – из 141 записи):

```
Быстрая сортировка ключей: 0.000092;
быстрая сортировка таблицы: 0.000137;
разница (ключи относительно быстрой сортировки таблицы) = 48.913043%
Быстрая сортировка ключей: 0.000092;
Сортировка таблицы пузырьком: 0.000661;
разница (ключи относительно пузырька) = 618.478261%
Быстрая сортировка таблицы: 0.000137;
Сортировка таблицы пузырьком: 0.000661;
разница (быстрая сортировка таблицы относительно пузырька) = 382.481752%
```

```
Быстрая сортировка ключей: 0.000330;
Быстрая сортировка таблицы: 0.000646;
разница (ключи относительно быстрой сортировки таблицы) = 95.757576%
Быстрая сортировка ключей: 0.000330;
Сортировка таблицы пузырьком: 0.010685;
разница (ключи относительно пузырька) = 3137.878788%
Быстрая сортировка таблицы: 0.000646;
Сортировка таблицы пузырьком: 0.010685;
разница (быстрая сортировка таблицы относительно пузырька) = 1554.024768%
```

Таким образом видно, что одинаковый алгоритм сортировки при разных “перемещениях” дает разные результаты. Скорость сортировки на относительно небольших данных в 2 раза меньше (разница показывает, на сколько процентов худший алгоритм хуже более быстрого, относительно более быстрого (то есть 1500% == выполнялся еще 15 времен лучшего алгоритма == в 16 раз дольше)). Как уже было рассчитано, упакованная запись занимает 40-50 байт. Если ограничить размер таблицы числом 256, то на каждый ключ из массива ключей можно отвести по 1 байту. Таким образом дополнительная память = 4-5%. Выигрыш по времени = 50% (на небольших данных). Рациональность использования очевидна.

Вывод.

В данной работе мы научились работать с такой структурой данных, как таблица, а также поэкспериментировали с вспомогательным массивом ключей. После проведения экспериментов очевидно, что для операций сортировки гораздо выгоднее ввести вспомогательный массив (см. Итоги и расчеты).

Ответы на вопросы

1. Как выделяется память под вариантную часть записи?

Память под вариативную часть выделяется таким образом, чтобы ее хватало на самую большую возможную структуру или тип.

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

Эти данные обработаются как некоторая последовательность байт и в зависимости от ОС (little endian, big endian) заполнится пространство этими байтами начиная с конца (little endian) или начала (big endian). Конечно же, если данные не соответствуют нужному нам формату и виду, мы не можем оперировать с данным объектом, рассчитывая на корректную работу программы.

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

Ответственность за правильность выполнения операций с вариативной частью целком и полностью лежит на плечах программиста.

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

Таблица ключей — некоторая вспомогательная таблица к некоторой имеющейся таблице записей, используемая для повышения эффективности выполнения операций «перемещения» записей, например, при сортировке данных.

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

Когда требуется просто доступ к элементу таблицы или же требуется полная информация об элементе таблицы — эффективнее обращаться к самой таблице (если время доступа к ней такое же, как к массиву ключей). В случае, когда записи таблицы требуется многократно перемещать (сортировка), выгоднее использовать таблицу.

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

Для обработки таблиц предпочтительнее использовать устойчивые сортировки, чтобы можно было сортировать по нескольким полям сразу (то есть, например, если у 2 людей одинаковые фамилии, то в таблице они будут отсортированы по имени), а так же чтобы уменьшить количество операций перемещения записей.