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

Группа ИУ7-35Б

Тема работы **Записи с вариантами, обработка таблиц**

Студент Баранов Николай Алексеевич

Преподаватель Барышникова Марина Юрьевна

2022 г.

Цель работы: приобрести навыки работы с типом данных «запись» («структура»), содержащим вариантную часть, и с данными, хранящимися в таблицах. Оценить относительную эффективность программы по времени и по используемому объёму памяти в зависимости от используемого алгоритма и от объёма сортируемой информации.

1. Описание условия задачи

Создать таблицу, содержащую не менее 40 записей с вариантной частью. Данные: фамилия, имя, группа, пол (м, ж), возраст, средний балл за сессию, дата поступления, адрес (для дома: улица, номер дома, номер квартиры; для общежития: номер общежития, номер комнаты). Ввести общий список студентов (возможность добавления и удаления элемента обязательна). Вывести список студентов указанного года поступления, живущих в общежитии. Упорядочить таблицу, по возрастанию ключей (где ключ – фамилия), используя: а) исходную таблицу; б) массив ключей, используя 2 разных алгоритма сортировки (простой, ускоренный). Оценить эффективность этих алгоритмов (по времени и по используемому объему памяти) при различной реализации программы. Обосновать выбор алгоритмов сортировки.

2. Описание ТЗ

2.1. Описание исходных данных и результатов.

При чтении данных программа сначала принимает на вход натуральное число в диапазоне от 1 до 50: размер массива. Далее программа считывает записи в массив структур.

При чтении записи программа считывает сначала фамилию, имя и группу. Размер входной строки не должен превышать 20 символов. Фамилия и имя могут содержать в себе только английские буквы и тире. Группа также может содержать и цифры. Далее считывается пол (можно ввести только одну цифру, обозначающую нужный пол), затем возраст (натуральное число в диапазоне от 16 до 27), средний балл (число с плавающей точкой в диапазоне от 2 до 5, после точки могут быть не более 2-х цифр, экспоненту писать нельзя, но дробную часть можно опустить), дата поступления (в данном случае только год, целое число в диапазоне от 1961 до 2022), потом определяется где живёт студент – дома или в общежитии (опять можно ввести только одну цифру), после чего вводится либо название улицы, номера дома и квартиры, либо номера общежития и комнаты (улица состоит не более чем из 20 английских букв, тире и цифр, все 4 числа натуральные, номер дома лежит в диапазоне от 1 до 999, номер общежития – от 1 до 20, номер квартиры/комнаты – от 1 до 2000.

На выход программа выдаёт отсортированную таблицу, в строке которой сначала идёт номер элемента в таблице, затем данные из структуры (либо только фамилия и положение элемента в неотсортированной таблице, если выводится таблица ключей). Все поля, кроме места жительства, выводятся в отдельной

ячейке. тип жилья не выводится, а адрес или номер общежития и комната выводятся в одном поле.

Выбор действия происходит в меню, где вводится число – номер действия.

При сравнении эффективности для каждого способа выводится время работы и затраченная память на хранение таблиц и их копирование при более быстром алгоритме сортировке.

2.2. Описание задачи, реализуемой программой.

Программа позволяет считывать таблицу, содержащую до 50 записей, добавлять или удалять записи, сортировать её напрямую или через таблицу ключей, выводить всю таблицу, выводить список студентов указанного года поступления, живущих в общежитии, а также сравнивать эффективность работы 2-х алгоритмов сортировки при сортировки самой таблицы или таблицы ключей.

2.3. Способ обращения к программе.

Запуск программы:

./app.exe

В этом случае программа начнёт выполнять свою задачу. Ввод осуществляется вручную с клавиатуры, но чтение таблицы можно сделать и из файла.

2.4. Описание возможных аварийных ситуаций и ошибок пользователя.

2.4.1. Переполнение буфера. Возникает, если строка при чтении оказалась длиннее чем 20 символов (при чтении чисел буфер уменьшается до 4 символов).

2.4.2. Неожиданный символ. Возникает, если в строке присутствуют символы, не предусмотренные программой, или из-за неправильного формата входных данных (например, лишний пробел, лишняя точка в числе, пробел посреди числа и т. д.).

2.4.3. Число не входит в диапазон. Возникает, если пользователь ввёл число, которое не входит в диапазон допустимых значений.

2.4.4. Ошибка при выделении памяти. В данной программе может возникнуть только при быстром алгоритме сортировки. Практически не зависит от действий пользователя. Возникает только при неудачной попытке выделить память на куче, поэтому вероятность возникновения крайне мала.

2.4.5. Неизвестный аргумент. Возникает при запуске с ключами командной строки.

3. Описание внутренних структур данных.

Для хранения данных о студентах была создана отдельная структура student.

```
#define MAX_STR_SIZE 20

typedef struct
{
    char surname[MAX_STR_SIZE + 1];
    char name[MAX_STR_SIZE + 1];
    bool isMale;
    char group[MAX_STR_SIZE + 1];
    int16_t age;
```

```
        float mark;
        int16_t year;
        enum
        {
            HOME,
            HOSTELL
        } typeofplace;
        union
        {
            struct
            {
                char street[MAX_STR_SIZE + 1];
                int16_t house;
                int16_t flat;
            } home;
            struct
            {
                int16_t num;
                int16_t room;
            } hostel;
        } placeinfo;
    } student;
```

Её поля:

3.1. surname – строка размером 20. Хранит фамилию.

3.2. name – строка размером 20. Хранит имя.

3.3. isMale – логическая переменная. Содержит пол.

3.4. group – строка размером 20. Хранит группу.

3.5. age – целое число. Хранит возраст.

3.6. mark – число с плавающей точкой. Хранит средний балл.

3.7. year – целое число. Хранит год поступления.

3.8. typeofplace – значение, хранящее тип жилища (дом или общежитие).

3.9. placeinfo – объединение, которое можно интерпретировать либо как строку и 2 целых числа, либо как только 2 целых числа.

4. Целесообразность применения типа «запись» с вариантным полем.

Я считаю, что для этой задачи применение типа «запись» с вариантной частью целесообразно, но при этом нецелесообразно реализовывать с использованием объединения. В случае, если студент живёт дома, вариантная часть хранит 2 целых числа и строку, а если в общежитии – то только 2 целых числа. Отличие между ними только в наличии строки в первом случае. Можно было бы хранить 2 целых числа и строку, а для случая, если студент живёт в общежитии, то строку оставлять пустой (ставить в начало символ окончания строки). Это также позволило бы не хранить переменную typeofplace, что также позволило бы ещё и сэкономить немного памяти. Я считаю, что «запись» с вариантным полем через объединение стоит применять только в тех случаях, когда хотя бы у 2 вариантов хранения есть поля, которых нет хотя бы у 1 другого варианта, так как там это действительно может сэкономить память.

5. Сравнительный анализ разных способов сортировки.

Для замера времени и памяти программа запускала 100 итераций, на каждой из которых при этом заново генерировалась таблица. Память во всех случаях для одного размера оставалась неизменной, а время работы выводится среднее за все итерации.

Программа может хранить до 50 записей. В таблице ниже приведены результаты работы сортировок для максимальной для пользователя размерности.

Условия	Среднее время, мкс	Память, байт
Пузырьком, таблица	507	5400
Пузырьком, ключи	190	7000
Слиянием, таблица	379	10800
Слиянием, ключи	130	8600

Как видно из таблицы, уже даже на сравнительно небольшой размерности сортировка слиянием по времени показала себя лучше, чем сортировка пузырьком. На таблицу записей уходит 5400 байт, а на таблицу ключей только 1600 байт. Так как у нас 50 записей, то получаем, что размер одной записи в таблице равен 108 байт, при этом один элемент в таблице ключей равен только 32 байта. Получается, что для сортировки пузырьком при использовании таблицы ключей память возрастёт примерно на 30%. При сортировке слиянием для слияния нужен дополнительный буфер, равный размеру сортируемого массива. Поэтому при сортировке слиянием таблицы придётся использовать на 100% больше памяти чем при сортировке пузырьком таблицы. При этом при сортировке таблицы ключей буфер создаётся именно для таблицы ключей, из-за чего здесь прирост памяти в сравнении с самым оптимальным случаем идёт только на 59%, а в сравнении с сортировкой слиянием всей таблицы расходуется даже на 20% меньше памяти. Если сравнивать время, то при сортировке пузырьком таблица ключей сортируется на 63% быстрее, а при сортировке слиянием – на 66%. Процент прироста памяти получается меньше, чем процент убывания времени работы, а при сортировке слиянием таблица ключей позволяет сэкономить ещё и память в сравнении со случаем сортировки слиянием всей таблицы, из чего следует вывод, что создание таблицы ключей позволяет эффективнее сортировать таблицу, имея преимущество во времени, которое сильнее чем лишние затраты в памяти.

Время работы разных сортировок сравнивать не буду, так как из-за того, что сложность сортировки пузырьком – $O(n^2)$, а сортировки слиянием – $O(n\log_2 n)$, на больших размерностях сортировка слиянием будет только увеличивать свою эффективность относительно сортировки пузырьком, однако на меньших размерностях сортировка пузырьком может оказаться более эффективна. Для этого просто приведу таблицы для размерностей 10 и 2000.

Для таблицы размерностью 10:

Условия	Среднее время, мкс	Память, байт
Пузырьком, таблица	23	1080
Пузырьком, ключи	9	1400
Слиянием, таблица	57	2160
Слиянием, ключи	20	1720

Для таблицы размерностью 2000:

Условия	Среднее время, мкс	Память, 10^3 байт
Пузырьком, таблица	712027	216
Пузырьком, ключи	255469	280
Слиянием, таблица	24291	432
Слиянием, ключи	7996	344

6. Набор тестов.

Тесты описаны только для ввода значений. Отсутствие выходных данных означает в данном случае что тест позитивный.

В меню проверяется только то что после каждого действия запускается нужная функция, а после неверного ввода просит ввести ещё раз.

Вывод таблицы записей не представлен ниже, так как его сложно записать в эту таблицу.

Входные данные	**Выходные данные**	**Что и где проверялось**
a		Ввод строки размером 1
qwertyuiop		Ввод строки, размер которой не граничный
asd-fgh-jkl-zxcv-bnm		Ввод строки размера 20 символов
1		Ввод минимального целого числа
3		Ввод целого числа не на границе
2000		Ввод целого числа максимального размера (в данном случае – для номера квартиры)
0003		Ввод целого числа с ведущими нулями (без переполнения буфера)
2.00		Ввод минимального числа с плавающей точкой
5.00		Ввод максимального числа с плавающей точкой
4.33		2 цифры после разделителя
4.3		1 цифра после разделителя

4.		0 цифр после разделителя
4		Без разделителя
what	Ошибка. Встречен некорректный символ.	Ввод не числа, если нужно целое число.
12345	Ошибка. Переполнение буфера при вводе.	Переполнение буфера (при чтении целого числа).
13	Ошибка. Число не входит в допустимый диапазон значений.	Число не входит в диапазон.
	Ошибка. Пустой ввод	Пустой ввод.
Имя	Ошибка. Встречен некорректный символ.	Ввод недопустимых символов в строку.
ааааааааааааааааааааааа	Ошибка. Переполнение буфера при вводе.	Переполнение буфера при вводе строки.
11	Ошибка. Встречено слишком много символов.	Переполнение буфера при выборе пола студента или места его проживания.

7. Выводы по проделанной работе.

В ходе выполнения работы я познакомился со структурой с вариантной частью и изучил методы работы с ней. Также я познакомился со способами работы с таблицами. Ещё я сравнил время работы и используемую память сортировки таблицы и таблицы ключей пузырьковой сортировкой и сортировкой слиянием. Для малых размеров сортировка пузырьком оказалась самой эффективной, но для больших размеров эффективнее всего оказалась сортировка слиянием таблицы ключей. Из-за таблицы ключей увеличивается расход памяти, но при этом идёт огромное уменьшение времени работы.

8. Ответы на вопросы.

8.1. Как выделяется память под вариантную часть записи.

В языке Си под объединение выделяется один блок памяти, равный самой большой переменной, входящей в него. Далее все переменные могут обращаться к данным, которые лежат в этом участке памяти, из-за чего при изменении одной переменной поменяются значения и у других переменных. Для того, чтобы внутри объединения можно было хранить несколько наборов переменных, используются структуры.

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

В языке Си эта ситуация не должна вызвать ошибки, если правильно выбрать поле или настроить тип указателя, то в объединение можно записать данные любого типа при условии, что они поместятся. Однако проблемы возникнут при попытке взять значение нужного поля, так как оно может оказаться практически любым, в том числе и не входящим в диапазон допустимых значений.

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

Ответственность полностью лежит на программисте. Пользователь скорее всего не знает, что это такое, поэтому именно программист должен дать ему интерфейс работы, который позволяет сделать только корректные действия.

8.4. Что представляет из себя таблица ключей, для чего она нужна?

Таблица ключей содержит в себе ключ – поле, по которому идёт поиск, и адрес ячейки, к которой относится ключ. Она содержит намного меньший объём данных, чем исходная таблица, поэтому её легче сортировать и осуществлять по ней поиск. Она ускоряет время поиска нужной записи, но увеличивает затрачиваемую память.

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

Если размеры таблицы сравнительно небольшие, либо записи в таблице ключей ненамного меньше записей в исходной таблице, или у нас нет памяти на создание массива ключей, то эффективнее обрабатывать данные в самой таблице. Иначе использование таблицы ключей будет более эффективным.

8.6. Какие способы сортировки предпочтительнее и почему?

Про сортировку таблицы и таблицы ключей я ответил в предыдущем вопросе. При выборе алгоритма сортировки, если размер таблицы большой, то лучше брать алгоритм с асимптотикой O(nlogn): быструю сортировку, сортировку слиянием или пирамидальную сортировку. Также важна устойчивость сортировке, а из трёх вышеперечисленных устойчивой является только сортировка слиянием. Однако на небольших размерах таблицы зачастую даже самые простые сортировки оказываются быстрее и эффективнее по памяти, например, сортировка пузырьком, которая тоже является устойчивой.