МГТУ им. Н. Э. Баумана

Отчет по лабораторной работе №2 по предмету «Типы и структуры данных»
«Записи с вариантами. Обработка таблиц.»

Выполнила Параскун София,
ИУ7-34Б
Проверила
...

Москва, 2020 г.

1. Описание условия задачи

Создать таблицу, содержащую не менее 40 записей (тип — запись с объединениями). Упорядочить данные в ней по возрастанию ключей, двумя алгоритмами сортировки, где ключ — любое невариативное поле (по выбору программиста), используя: а) саму таблицу, б) массив ключей. Осуществить поиск информации по варианту.

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);

Вывести общий список студентов. Вывести список студентов, живущих в общежитии указанного года поступления.

2. Описание ТЗ, включающего внешнюю спецификацию

a) описание исходных данных и результатов

Изначально мы имеем базу данных, состоящую из 40 записей и хранящуюся в файле in_out.txt. Каждое поле каждой записи записывается с новой строки.

Результатами программы в некоторых режимах является выведенная в консоль таблица, содержащая все данные или ключи со значением определенного поля, или же сообщение о статусе выполнения действия.

b) Допущения

Все данные в исходном файле являются корректными и в проверке не нуждаются. Также при добавлении новых записей в файл гарантируется их корректность многочисленными проверками данных, введенных пользователем.

При вводе строк можно использовать пробелы, при этом они запоминаются как значащие символы.

Поиск осуществляется по полю «Дата поступления», а именно по году поступления; сортировка осуществляется по полю «Фамилия»; удаление записи осуществляется по полю «Группа».

c) Способ обращения к программе

Программа запускается через консоль, после чего можно увидеть меню программы, состоящее из 7 пунктов:
1 - Вывод всех данных,
2 - Вывод студентов, проживающих в общежитии определенного года поступления,
3 - Добавить данные в конец списка,
4 - Удалить одну запись по полю «Группа»,
5 - Отсортировать таблицу ключей по полю «Фамилия» без сортировки данных,
6 - Вывод отсортированных данных по полю «Фамилия» по сортированным ключам,
7 - Вывод отсортированных данных по полю «Фамилия» без использования ключей,
8 - Вывод результатов сравнения эффективности сортировки таблицы и сортировки по ключам,
9 - Вывод результатов сравнения эффективности сортировки вставками с барьером и быстрой сортировки,
0 - Выход из программы.

Все вводы и выводы также осуществляются в консоли.

d) Описание возможных аварийных ситуаций и ошибок пользователя

№ ситуации	Аварийная ситуация	Реакция программы
1	Ввод некорректного пункта в главном меню	Прекращение работы и сообщение «Its end of working the program!»
2	Ввод некорректного по подсказке поля записи	Сообщение об ошибке, возврат в главное меню
3	Отсутствие записи для удаления	Сообщение «Record not found!», возврат в главное меню

3. Описание внутренних СД

Необходимо обеспечить хранение данных с множественными полями и вариативной частью. Для записей необходимо хранить следующие данные каждого студента: фамилия, имя, группа, пол(м или ж), возраст, средний балл, дату поступления, тип жилья (дом или общежитие) и само место проживания, задаваемое либо улицей, №дома и №квартиры или же №общежития, №комнаты.

Значения используемых ниже констант:

```
#define MAX_LEN 21
#define MAX_LEN_GROUP 11
#define MAX_DATE_LEN 11
#define MAX_ARR 80
```

Основной является структура student.

Она содержит поля: surname, отвечающее за фамилию; name, отвечающее за имя; group, отвечающее за название группы; gender, отвечающее за пол (содержит «F» и «M» и нулевой символ); age, отвечающее за возраст; average_score, отвечающее за средний балл студента; date, отвечающее за дату поступления; housing, отвечающее за тип жилья, который впоследствии будет использован; объединение adress, используемое как вариативная часть и состоящее из нижеописанных структур.

```
struct student
{
    char surname[MAX_LEN];
    char name[MAX_LEN];
    char group[MAX_LEN_GROUP];
    char gender[2];
    int age;
    float average_score;
    char date[MAX_DATE_LEN];
    char housing[MAX_LEN];
    union adress{
        struct house home;
        struct hostel sweet_home;
    }adress;
};
```

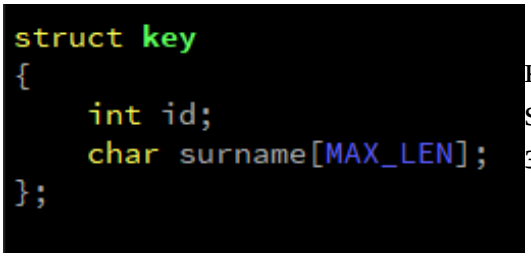
```
struct house
{
    char street[MAX_LEN];
    int house_number;
    int apart_number;
};
```

Структура house отвечает за тип жилья «дом» и состоит из следующих полей: street, отвечающее за улицу проживания; house_number, отвечающее за дом проживания; apart_number, отвечающее за квартиру проживания.

Структура hostel отвечает за тип жилья «общежитие» и состоит из следующих полей: hostel_number, отвечающее за номер общежития; room_number, отвечающее за номер комнаты.

```
struct hostel
{
    int hostel_number;
    int room_number;
};
```

Также внутри программы используется вспомогательная структура key, используемся при сортировке данных по ключам по полю «Фамилия».



В этой структуре поле id отвечает за индекс конкретной записи в массиве записей, а поле surname содержит фамилию соответствующей записи.

4. Описание алгоритма

Внутри основной функции осуществляется выбор пользователем режима работы программы из предложенного меню. Пользователь вводит целое число, соответствующее номеру режима. При вводе 0 или неверного значения работа программы завершается с выводом «Its end of working the program!».

Перед первым вводом пользователя происходит считывание уже имеющихся данных из файла «in_out.txt». Это осуществляется функцией read_struct, которая производит считывание и запись данных, пока не обнаружится конец файла.

При выборе 1 пункта меню «Вывод всех данных» выводятся уже считанные данные в формате таблицы с помощью функции output_data_all, которая в свою очередь вызывает функцию out_hint для вывода шапки таблицы.

При выборе 2 пункта меню «Вывод студентов, проживающих в общежитии определенного года поступления» внутри функции output_data_certain запрашивается у пользователя желаемый год поступления, если он находится в диапазоне 1900-2020, то происходит вывод соответствующих данных в виде таблицы, или же сообщение «It can’t be!» и возврат в меню.

При выборе 3 пункта меню «Добавить данные в конец списка» вызывается функция add_data. Пользователю поочередно предлагается заполнить поля записи, при вводе некорректных данных выводится соответствующее сообщение и происходит возврат в меню. Если все данные корректны, выводится сообщение «Added successfully!», происходит добавление данных в массив записей и в файл, содержащий все записи.

При выборе 4 пункта меню «Удалить одну запись по полю «Группа»» вызывается функция delete_by_field. Она запрашивает у пользователя значение поля, если оно не корректно, выводит сообщение «Wrong group!» и возвращает в меню. Если данные корректны, начинается поиск первой по вхождению записи, значение поля «Группа» которой совпадает с введенным пользователем значением. Если таких записей не найдено — выводится сообщение «Records not found!» и происходит

возврат в меню. Иначе функция delete_rec осуществляет удаление найденной записи по индексу.

При выборе 5 пункта меню «Отсортировать таблицу ключей по полю «Фамилия» без сортировки данных» сначала вызывается функция key_init, которая заполняет массив структур key соответствующими индексами и фамилиями записей. Далее происходит сортировка ключей по полю «Фамилия» функцией sort_key_by_surname. После чего осуществляется вывод таблицы ключей, уже отсортированной по определенному полю.

При выборе 6 пункта меню «Вывод отсортированных данных по полю «Фамилия» по сортированным ключам» аналогично пункту 5 производится инициализация ключей и их сортировка, но выводится в конце уже не таблица ключей, а сама таблица с записями, вывод организован с использованием отсортированной таблицы ключей функцией output_data_by_key.

При выборе 7 пункта меню «Вывод отсортированных данных по полю «Фамилия» без использования ключей» производится сортировка данных функцией sort_data_by_surname, вывод отсортированной таблицы.

При выборе 8 пункта меню «Вывод результатов сравнения эффективности сортировки таблицы и сортировки по ключам» функция key_efficiency производит замер времени сортировки по значениям и по ключам, вычисляет среднее значение и относительную эффективность метода сортировки по ключам.

При выборе 9 пункта меню «Вывод результатов сравнения эффективности сортировки вставками с барьером и быстрой сортировки» функция type_of_sorting_efficiency проводит аналогичные 8 пункту замеры и вычисления но для сортировки ключей быстрой сортировкой относительно сортировки ключей вставками с барьером.

5. Набор тестов, с указанием, что проверяется

Сначала рассмотрим ситуации в главном меню, когда пользователь выбирает режим работы.

№ теста	Входные данные	Что проверяется?	Реакция программы
1	fwhf	Не является целым числом	Сообщение «its end of working the program!»
2	13	Не является режимом программы	Сообщение «its end of working the program!»
3	0	Ожидается выход из	Сообщение «its end of working the

		программы	program!»
4	4	Ожидается удаление записи по полю	Подсказка к удалению записи

Теперь рассмотрим работу каждого отдельного режима. Режим 1 - «Вывод всех данных». Так как работа режима не зависит от пользователя, программа всегда выдает таблицу, содержащую исходные данные.

Режим 2 - «Вывод студентов, проживающих в общежитии определенного года поступления». От пользователя требуется ввести год поступления студента.

№ теста	**Входные данные**	**Что проверяется?**	**Реакция программы**
1	аоал	Не является числом	Сообщение «Incorrect data!»
2	1023	Слишком маленький год	Сообщение «It can’t be!»
3	2092	Слишком большой год	Сообщение «It can’t be!»
4	2019	Корректные данные	Вывод соответствующих данных

Режим 3 - «Добавить данные в конец списка». В этом режиме от пользователя требуется ввести много данных, каждые из которых требуют проверки.

№ теста	**Входные данные**	**Что проверяется?**	**Реакция программы**
1		Пустая фамилия	Сообщение «Empty surname!»
2	Aaaaaaaaaaaaaaaaaaaaa	Слишком длинная фамилия (21 символ)	Сообщение «So long surname!»
3	Lapaev	Корректная фамилия	*продолжение ввода*
4		Пустое имя	Сообщение «Empty name!»
5	Aaaaaaaaaaaaaaaaaaaaa	Слишком длинное имя (21 символ)	Сообщение «So long name!»
6	Igor	Корректное имя	*продолжение ввода*
7		Пустая группа	Сообщение «Empty group!»
8	HHFWO4828591	Слишком длинная группа (12 символов)	Сообщение «So long group!»
9	IU7-34B	Корректная группа	Продолжение ввода
10		Пустой пол	Сообщение «Wrong gender!»
11	L	Несуществующий пол	Сообщение «Wrong gender!»
12	m	Пол с маленькой буквы	Сообщение «Wrong gender!»
13	M	Корректный пол	*продолжение ввода*
14	fafl	Не целое число	Сообщение «Wrong age!»
15	14	Слишком маленький возраст	Сообщение «Wrong age!»

		(<16)	
16	41	Слишком большой возраст (>40)	Сообщение «Wrong age!»
17	19	Корректный возраст	*продолжение ввода*
18	fjsfa	Не число	Сообщение «Wrong average score!»
19	1.832	Слишком низкий балл (<3)	Сообщение «Wrong average score!»
20	6.2	Слишком высокий балл (>5)	Сообщение «Wrong average score!»
21	4.38	Корректный балл	*продолжение ввода*
22		Пустая дата	Сообщение «Wrong date!»
23	16.08.2019	Некорректная дата	Сообщение «Wrong date!»
24	16/08/2019	Корректная дата	*продолжение ввода*
25		Пустой тип жилья	Сообщение «Wrong housing type!»
26	Fhsfd;la	Некорректный тип жилья	Сообщение «Wrong housing type!»
27	house	Корректный тип жилья	*продолжение ввода адреса дома*
28	hostel	Корректный тип жилья	*продолжение ввода общежития*

Дальше ввод разделяется на ввод информации по дому

№ теста	Входные данные	Что проверяется?	Реакция программы
1		Пустая улица	Сообщение «Wrong street!»
2	Aaaaaaaaaaaaaaaaaaaaa	Слишком длинная улица (21 символ)	Сообщение «Wrong street!»
3	Parkovaya	Корректная улица	*продолжение ввода*
4	hsfik	Не является числом	Сообщение «Wrong house number!»
5	-16	Отрицательный номер дома	Сообщение «Wrong house number!»
6	21	Корректный номер дома	*продолжение ввода*
7	рада	Не является числом	Сообщение «Wrong apartment number!»
8	0	Нулевой номер квартиры	Сообщение «Wrong apartment number!»
9	101	Корректный номер квартиры	Сообщение «Added successfully!»

И ввод информации по общежитию

№ теста	Входные данные	Что проверяется?	Реакция программы
1	hfaf	Не является числом	Сообщение «Wrong hostel

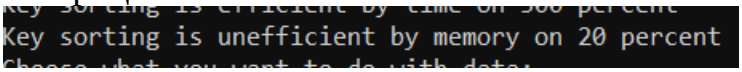
			number!»
2	-1	Отрицательный номер общежития	Сообщение «Wrong hostel number!»
3	23	Слишком большой номер общежития	Сообщение «Wrong hostel number!»
4	5	Корректный номер общежития	*продолжение ввода*
5	fkafo	Не является числом	Сообщение «Wrong room number!»
6	0	Нулевой номер комнаты	Сообщение «Wrong room number!»
7	819	Корректный номер комнаты	Сообщение «Added successfully!»

Режим 4 - «Удалить одну запись по полю «Группа»». От пользователя требуется ввести значение поля «Группа».

№ теста	**Входные данные**	**Что проверяется?**	**Реакция программы**
1		Пустая группа	Сообщение «Wrong group!»
2	NFDIQ68LS90	Слишком длинная группа (11 символ)	Сообщение «Wrong group!»
3	IU7-35M	Корректная группа, имеются записи с таким значением поля	Сообщение «Records not found!»
4	IU7-35M	Корректная группа, не имеется записей с таким значением поля	Сообщение «Record deleted successfully!»

Режим 5 «Отсортировать таблицу ключей по полю «Фамилия» без сортировки данных» и режим 6 «Вывод отсортированных данных по полю «Фамилия» по сортированным ключам», а также режим 7 «Вывод отсортированных данных по полю «Фамилия» без использования ключей» не зависит от данных, вводимых пользователем, программа всегда выдает отсортированную таблицу ключей (режим 5) или отсортированную по ключам таблицу со всеми данными (режим 6) или отсортированную без ключей таблицу со всеми данными (режим 7).

Режим 8 «Вывод результатов сравнения эффективности сортировки таблицы и сортировки по ключам» не зависит от пользователя, но зависит от количества записей. В случае эффективности по памяти мы всегда будем получать один и тот же результат, так как размер структуры student равен 124 б, а размер структуры key составляет 25 б. Отсюда следует, что использование ключей проигрывает в эффективности на 20 процентов



Сравнение эффективности по времени работы представлено в таблице ниже (время работы приведено в тактах). Ключи и таблица сортировались методом вставок с барьером.

Кол-во записей	**Время по ключам**	**Время без ключей**	**Эффективность**
40	445206	1709080	На 283%
100	3485018	12469114	На 257%
200	17225044	21703956	На 226%
1000	61309840	77981512	На 227%

На повторных прогонах время уменьшалось, поэтому в таблицу были внесены первые замеры.

Режим 9 «Вывод результатов сравнения эффективности сортировки вставками с барьером и быстрой сортировки» не зависит от пользователя, но так же как режим 8 зависит от количества записей. Ниже приведена таблица сравнения эффективности сортировок записей по ключам(время работы указано в тактах).

Кол-во записей	**Время сортировки вставками**	**Время быстрой сортировки**	**Эффективность**
40	356291	7479	На 4663%
100	3507096	33208	На 10460%
200	12705299	68149	На 18543%
1000	20109924	907632	На 2315%

А эта таблица сравнения эффективности сортировок записей по значениям.

Кол-во записей	**Время сортировки вставками**	**Время быстрой сортировки**	**Эффективность**
40	1775531	32332	На 5391%
100	16380431	98548	На 16521%
200	20816016	161753	На 12969%
1000	38593868	597694	На 6557%

6. Выводы по проделанной работе

В ходе этой лабораторной удалось поработать с таким типом данных как запись с вариативной частью. Для этого типа было организовано считывание, хранение, вывод в формате таблицы, сортировка и удаление по значению конкретного поля. Использование вариативной части значительно упрощает хранение данных, так как при ее отсутствии пришлось бы создавать отдельные структуры, что не так удобно в обработке внутри программы.

Также была произведена оценка эффективности сортировки записей с большим количеством полей по ключам и по самим значениям, сравнение алгоритмов быстрой сортировки и сортировки вставками с барьером на ранее используемых данных. При этом учитывались затрачиваемое время и затрачиваемый объем памяти.

Сортировка по ключам эффективнее по времени в среднем на 230%, что является большим плюсом при обработке большого количества записей, однако памяти на эту реализацию требуется на 20% больше. При обработке малого количества записей в использовании ключей нет смысла, так как машина обрабатывает данные почти моментально, но при большом количестве записей стоит пренебречь затратами памяти и сделать упор на сокращение времени.

Сравнивая быструю сортировку и сортировку вставками с барьером нетрудно заметить, что последняя сильно проигрывает по времени. Это значит, что независимо от количества записей и независимо от способа сортировки (по ключам или по значениям) стоит использовать именно быструю сортировку для сильной экономии времени.

Ответы на контрольные вопросы

1. Как выделяется память под вариативную часть записи?

Память для вариативной части выделяется по максимальной длине вариативного поля.

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

В случае ввода в вариативную часть несоответствующих описанию данных может произойти следующее: если форматы не совпадают (например, int и char), то программа завершится аварийно; если форматы совпадают, то данные запишутся не в ту вариативную часть и доступ к ним будет потерян.

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

Исключительно разработчик.

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

Таблица ключей представляет собой индексы и определенные поля соответствующих элементов. Используется для оптимизации работы в тех случаях, когда можно обойтись без использования всех данных, ускоряет процесс.

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

Таблицу ключей стоит использовать в том случае, когда данные имеют большое количество полей, в обратном случае это не имеет смысла.

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

При сортировке таблиц значительно удобнее работать с таблицей ключей, потому что это может значительно сэкономить время. Что касается вида сортировка, удобнее всего использовать встроенную qsort, результаты эффективности которой в сравнении с сортировкой вставками с барьером были приведены выше. Она производит O(n * log n) обменов в лучшем случае и O(n^2) в худшем. Но еще лучше использовать сортировки, которые совершают не более O(n * log n) обменов в любой ситуации — это сортировки слиянием и кучей. Они являются наиболее оптимальными для общего назначения.