МГТУ им. Н.Э.Баумана

Отчет по Лабораторной работе №2

по теме “Записи с вариантами. Обработка таблиц”.

Работу выполнил:

Елгин И. Ю.

Группа ИУ7-34Б

**Заданча***:*

Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами). Упорядочить данные в ней по возрастанию ключей, где ключ – любое невариантное поле (по выбору программиста), используя: а) саму таблицу, б) массив ключей (возможность добавления и удаления записей в ручном режиме обязательна). Ввести репертуар театров, содержащий: название театра, спектакль, режиссер, диапазон цены билета, тип спектакля: детский – для какого возраста, тип (сказка, пьеса, музыкальный); взрослый - тип (пьеса, драма, комедия); музыкальный – композитор, страна, минимальный возраст, продолжительность. Вывести список всех музыкальных спектаклей для детей указанного возраста с продолжительностью меньше указанной.

**Формат входные данных:**

• Файл с записанной заранее таблицей.

(файл содержит поля записи в необходимой форме, каждое поле записи записано в новой строке)

• Номер пункта меню.

(Возможные номера указываются при вводе)

\*\*\*\*\*\*\*\*\*\*\*\*\* меню \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1 - вывести таблицу на экран

2 - добавить запись

3 - удалить запись по индексу

4 - отсортировать ключи по меньшей цене билетов и вывести индексы

5 - отсортировать таблицу по меньшей цене билетов и вывести записи

6 - отсортировать ключи по меньшей цене билетов и вывести таблицу

7 - сравнить сортировку по ключам и сортировку таблицы

8 - сравнить быструю сортировку и сортировку пузырьком

9 - найти записи по указанному возрастному ограничению и продолжительности

0 – выход

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

• Строка или целое число, в зависимости от выбранного пункта меню (максимальная длина строки или интервал ввода чисел обозначаются при вводе).

**Формат выходные данных:**

• Таблица ключей

• Таблица записей

• Таблица записей по ключам

• Вывод результатов поиска

• Таблица результатов эффективности сортировок

**Аварийное завершение программ:**

При пустом вводе, или вводе буквы в числовое поле, или вводе числа, выходящего за допустимый интервал программа выводит сообщение “Некорректный ввод, повторите” и продолжает запрашивать у пользователя ввод поля.

При вводе количества символов больше разрешённого программа выводит сообщение “Некорректный ввод, повторите” и продолжает запрашивать у пользователя ввод поля.

При вызове функции “Добавить запись”, когда массив заполнен, программа выводит сообщение “Достигнут максимум записей” и продолжает работу.

При вызове функций сортировки, удаления или сравнения, когда массив пуст, программа выводит сообщение “В таблице нет записей” и продолжает работу.

**Обращение к программе:**

Исполняемый файл main.exe. Запускается из командной строки, аргументы строки не требуются.

**Структуры данных:**

Для работы с записями используется 2 массива, массив структур и массив ключей.

*#define THEATER 70*

*#define NAME\_LEN 80*

*#define ARR\_LEN 350*

*struct performance array[ARR\_LEN];*

Массив структур для хранения записей максимальная длина массива 350 записей.

*int keys[ARR\_LEN][2];*

Массив ключей включает 2 поля индекс записи и цену. 1 элемент занимает 8 байт.

*struct performance*

*{*

*char theater[THEATER + 1];*

*char perf\_name[NAME\_LEN + 1];*

*char director[NAME\_LEN+1];*

*int low\_prise;*

*int high\_prise;*

*int type\_spc;*

*union*

*{*

*struct v1*

*{*

*int age;*

*int type;*

*} child;*

*struct v2*

*{*

*int type;*

*} adult;*

*struct v3*

*{*

*char composer[NAME\_LEN + 1];*

*char country[THEATER + 1];*

*int age;*

*int duration;*

*} musical;*

*} spct;*

*};*

Структура содержит в себе 3 массива символов для хранения строк,3 целых числа для хранения цен и варианта спектакля, объединение размером 160 байт. Размер одной записи – 405 байт

**Функции:**

*void fill\_keys(int keys[ARR\_LEN][2], struct performance \*array, int count);*

- Функция реализует заполнение массива ключей.

- Параметры на вход: keys – массив ключей, array – массив записей, count-количество элементов.

- Результат работы массив ключей.

*void print\_keys(int keys[ARR\_LEN][2], int count);*

- Функция выводит массив ключей.

- Параметры на вход: keys – массив ключей, count-количество элементов.

- ничего не возвращает.

*void add\_struct(struct performance \*array, int i);*

- Функция добавляет запись в массив.

- Параметры на вход: array – массив записей, i-количество элементов.

- Результат работы массив записей.

*void remove\_struct(struct performance \*array, int \*count);*

- Функция реализует удаление записи из массива.

- Параметры на вход: array – массив записей, count-количество элементов.

- Результат работы array – массив записей, count-количество элементов.

*void print\_array\_by\_keys(struct performance \*array, int keys[ARR\_LEN][2], int count);*

- Функция реализует ввод массива по ключам.

- Параметры на вход keys – массив ключей, array – массив записей, count-количество элементов.

-Ничего не возвращает.

*void print(struct performance \*array, int count);*

- Функция выводит поле структуры.

- Параметры на вход: array – массив записей, count-количество элементов.

-Ничего не возвращает.

*void fill\_file(struct performance \*array, int \*count);*

- Функция производит чтение из файла.

- Параметры на вход: array – массив записей, count-количество элементов.

- Результат работы массив записей, количество элементов.

*void fill\_manually(struct performance \*array, int \*count);*

- Функция производит чтение с консоли.

- Параметры на вход: array – массив записей, count-количество элементов.

- Результат работы массив записей, количество элементов.

*void sort1 (int keys[ARR\_LEN][2], int count);*

- Функция реализует быструю ключей пузырьком.

- Параметры на вход: keys – массив ключей, count-количество эллементов.

- Результат работы отсортированный массив

*void sort\_array\_qs(struct performance \*array, int first, int last);*

Функция реализует быструю сортировку массива.

- Параметры на вход: array – массив, first-левая граница сортировки, last – правая.

- Результат работы отсортированный массив

*void sort\_array\_sort1 (struct performance \*array, int count);*

- Функция реализует сортировку массива пузырьком.

- Параметры на вход: array- массив, count – количество эллементов.

- Результат работы отсортированный массив

*void search(struct performance \*array, int count);*

- Функция реализует поиск записей по параметрам и вывод их на экран.

- Параметры на вход array – массив записей, count-количество.

- Ничего не возвращает.

*void qs(int keys[ARR\_LEN][2], int first, int last);*

- Функция реализует быструю сортировку ключей.

- Параметры на вход: keys – массив ключей, first-левая граница сортировки, last – правая.

- Результат работы отсортированный массив.

*void time\_sort\_keys( struct performance \*array, int keys[ARR\_LEN][2], int count);*

- Функция выводит время сортировки массива ключей.

- Параметры на вход: keys – массив ключей, array – массив записей, count-количество элементов.

- Ничего не возвращает

*void time\_sort\_array( struct performance \*array, int keys[ARR\_LEN][2], int count);*

- Функция выводит время сортировки массива записей.

- Параметры на вход: keys – массив ключей, array – массив записей, count-количество элементов.

- Ничего не возвращает

**Алгоритм:**

* Заполнение таблицы
* Запуск рабочего цикла программы, работает пока пользователь не выберет пункт меню выход
* Вывод меню
* По выбору пользователя программа выполняет одну из следущих команд:
* Добавление записи

-последовательно ввести значения всех полей добавляемой записи, в случае некорректных данных – вывести сообщение

-увеличить счётчик количества записей.

* Удаление записи

-Прочитать индекс удаляемой записи. В случае недопустимого индекс вернуть ошибку.

-Сдвинуть все элементы после данного на один назад

-Уменьшить счётчик количества записей;

* Поиск музыкальных детских постановок указанного возраста и продолжительности

-Ввести минимальный допустимый возраст и максимальную продолжительность, в случае некорректных данных вернуть сообщение

-Последовательно в таблице сравнивать полученные значения со значениями полей записей

-В случае соответствия вывести запись

-Если не было найдено соответствующих записей, вывести сообщение

* Сортировка

-Если следует выполнять сортировку по ключам, составить таблицу ключей на основе исходной таблицы, записывая индексы и значения максимальной цены за билет

*Набор тестов:*

Заполнение таблицы:

|  |  |  |
| --- | --- | --- |
| Ввод | Случай | Вывод программы |
| Режисcер: Ааа..аа(более 40) | Ввод строки длиннее разрешённой | Некорректный ввод, повторите |
| Театр: Аааа..аа(более 35) | Ввод строки длиннее разрешённой | Некорректный ввод, повторите |
|  | Пустой ввод | Некорректный ввод, повторите |

Заполнение запроса:

|  |  |  |
| --- | --- | --- |
| Ввод | Случай | Вывод программы |
| Возрастное ограничение: ааа | Буквы вместо цифр | Некорректный ввод, повторите |
| Возрастное ограничение: 30 | Выход за пределы разрешённого интервала | Некорректный ввод, повторите |
| Продолжительность: -5 | Выход за пределы разрешённого интервала | Некорректный ввод, повторите |
|  | Пустой ввод | Некорректный ввод, повторите |
| Выберите действие (1,2,0): 3 | Ввод несоответствующий  ожидаемому | Некорректный ввод, повторите |

Другие возможные ошибки:

|  |  |  |
| --- | --- | --- |
| Ввод | Случай | Вывод программы |
| Сортировка ключей (массив пуст) | Пустой массив | В таблице нет записей |
| Вывод таблицы (массив пуст) | Пустой массив | В таблице нет записей |
| Удаление записи (массив пуст) | Пустой массив | В таблице нет записей |
| Поиск(нет подходящих записей) | Нулевой поиск | Записей не найдено |
| Добавление записи (массив полон) | Переполнение массива | Достигнут максимум записей |

**Сравнение сортировок:**

**40 записей.**

|  |  |  |
| --- | --- | --- |
| Тип сортировки | Сортировка ключей | Сортировка таблицы |
| Пузырьком | 18700 | 29340 |
| Быстрая | 7590 | 17604 |

Быстрая сортировка быстрее сортировки пузырьком на 60 % и в случае сортировки таблицы, и в случае сортировки ключей.

Сортировка ключей быстрее сортировки таблицы на 63%, при этом матрица ключей занимает лишь 1.9 % памяти относительно таблицы структур.

**160 записей.**

|  |  |  |
| --- | --- | --- |
| Тип сортировки | Сортировка ключей | Сортировка таблицы |
| Пузырьком | 220796 | 134780 |
| Быстрая | 27707 | 80868 |

Быстрая сортировка быстрее сортировки пузырьком на 87 % и в случае сортировки таблицы, и в случае сортировки ключей.

Сортировка ключей быстрее сортировки таблицы на 67%, при этом матрица ключей занимает лишь 1.9 % памяти относительно таблицы структур.

**250 записей.**

|  |  |  |
| --- | --- | --- |
| Тип сортировки | Сортировка ключей | Сортировка таблицы |
| Пузырьком | 220796 | 134780 |
| Быстрая | 117645 | 431881 |

Быстрая сортировка быстрее сортировки пузырьком на 86 % и в случае сортировки таблицы, и в случае сортировки ключей.

Сортировка ключей быстрее сортировки таблицы на 82%, при этом матрица ключей занимает лишь 1.9 % памяти относительно таблицы структур.

*Вывод:* Использование объединений помогает сэкономить память при вариантной записи. Быстрая сортировка эффективнее сортировки пузырьком на 77%. А сортировка массива ключей более эффективным способом дает уменьшение времени на 70% по сравнению с сортировкой таблицы исходным способом, с ростом числа записей эффективность возрастает и стремится к 90%. Эффективнее использовать быстрою сортировку ключей.

**Ответы на вопросы:**

*1.Как выделяется память под вариантную часть записи?*

Память под вариантную часть записи выделяется по наибольшему элементу объединения.

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

Они введутся, но не будет корректного доступа к ним, ввод данных должен в вариантную часть контролироваться программистом.

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

За правильностью выполнения любых операций с вариантной частью следит программист, так как компилятор это не отслеживает.

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

Таблица ключей – массив содержащий 2 поля: индекс и поле сортировки (в данной программе цену).

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

При малом объёме, занимаемом таблицей, таблицы эффективнее обрабатывать данные в ней, при большом количестве записей в ней и при большом количестве полей в записи, эффективнее использовать таблицу ключей.

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

При сортировке таблиц предпочтительнее использовать сортировки с наименьшим количеством перестановок, так как для перестановки необходимо копирование, а копирование записи занимает достаточно большое время.