Лабораторная работа №3 по дисциплине «Типы и структуры данных»

Записи с вариантами, обработка таблиц

Цель работы:

Приобрести навыки работы с типом данных «запись», содержащим вариантную часть, и с данными, хранящимися в таблицах. Сравнить несколько различных алгоритмов сортировок массива при использовании таблиц записей с большим числом полей и таблиц ключей. Оценить относительную эффективность программы (в процентах) по времени и по используемому объему памяти

Задание (Вариант 1):

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 — выбрано поле Количество страниц), используя: а) саму таблицу, б) массив ключей.

Ввести список литературы, содержащий фамилию автора, название книги, издательство, количество страниц, вид литературы (1: техническая – отрасль, отечественная, переводная, год издания; 2: художественная – роман, пьеса, стихи). Вывести список всей переводной лит-ры по указанной отрасли с годом издания не позднее указанного.

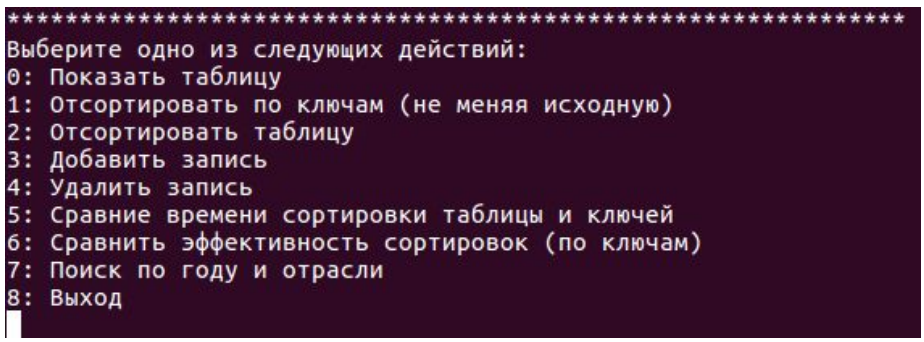
Исходные данные:

На вход программы подаются данные трех типов.

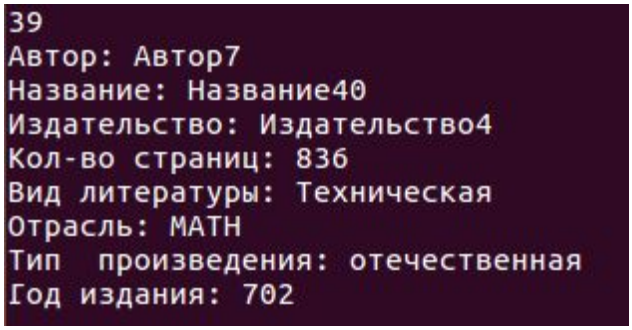
1) Выбор поля, указанного в подсказке. Необходимо ввести одно из указанных значений, никакие другие числа недопустимы.
2) Слова, длина слов не должна превышать 150 символов.
3) Целые числа. Для поля количество страниц должно быть строго больше 0.

Интерфейс программы:

1) Главное меню



2) Распечатанный фрагмент таблицы



Возможные ошибки пользователя:

1) При некорректном вводе или выборе меню пользователь получит следующее сообщение об ошибке **"Ошибочный ввод"**
2) При превышение количества элементов таблицы: **"Переполнение массива записей"**
3) При недоступности файла с данными: **"Файл не может быть открыт!"**

Структура данных для хранения записей таблицы

```
//Только для Художественной
//роман, пьеса, стихи
enum kind_art { novels, plays, poems };

//Только для технической
enum kind_localization { patriotic, translated };

enum type_book {technical,art};

struct technical_book
{
  char industry[LEN_STRING];
  kind_localization locate;
  int year;
  void create_technical_book();
  void show();

};

struct art_book
{
  kind_art art;
  void create_art_book();
  void show();
};


union kind_book {
  technical_book  tech;
  art_book art;
};

class Book {

private:

  string author_name;
  string book_title;
  string publisher;
  int pages;
  type_book book_t;
  kind_book book_record;
}
```

Структура данных для хранения ключей

```
class Table_key {
  int id;
  int pages;
}
```

Сравнение эффективности сортировок (по ключам)

Кол-во эл-тов	Время работы быстрой сортировки, мс	Время работы сортировки пузырьком, мс
20	10	13
40	13	37
80	32	179
160	33	253
200	40	375

Быстрая сортировка заметно быстрее пузырьковой. Это становится особенно заметно с ростом количества элементов массива.

Сравнить эффективность сортировок (по ключам)

Кол-во эл-тов	Время работы для таблицы ключей, мс	Время работы для таблицы записей, мс
20	5	127
40	8	263
80	14	401
160	33	839
200	40	1128

Каждая запись таблицы использует 264 байта памяти и дополнительных 8 байт для хранения таблицы ключей. Увеличение используемой памяти на 3% позволяет ускорить сортировку в 25-30 раз.
Значительное ускорение сортировки достигается отсутствием необходимости двигать значительный объем памяти для перестановки элементов таблицы, а также использованием эффективных алгоритмов сортировки.

Вывод:

Во время работы было проведено сравнение времени работы быстрой и пузырьковой сортировок. Сортировка пузырьком имеет более высокую алгоритмическую сложность, что приводит к ее медлительности.

(пузырек - O(n^2), быстрая — O(NlogN)).

Для хранения информации о различных видах книг была использована вариативная запись.Однако необходимость отслеживания правильности хранения и обработки данных, усложняет программу, хоть и дает небольшой выигрыш по памяти.

Хранение дополнительной таблицы ключей значительно уменьшает время сортировки, хоть и требует незначительного дополнительного объема памяти для ее хранения.

Контрольные вопросы

1. Как выделяется память под вариантную часть записи?

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

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

Тип данных в вариантной части при компиляции не проверяется, поэтому, контроль за правильностью ее использования возлагается на программиста. Поэтому может произойти что угодно.

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

Контроль за правильностью ее использования возлагается на программиста

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

При больших размерах таблиц поиск данных, имеющих указанный ключ, может потребовать больших затрат времени. Если же помимо поиска требуется произвести сортировку данных, то временные затраты

многократно возрастут, так как потребуется осуществлять их перестановку (перемещение). В этом случае можно уменьшить время обработки за счет создания дополнительного массива – таблицы ключей, содержащей индекс элемента в исходной таблице и выбранный ключ.

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

Если каждая запись таблицы состоит из небольшого кол-ва полей, то полезно обрабатывать данные в таблице. В других случаях выгоднее использовать таблицу ключей.

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

Если обработка данных производится в таблице, то необходимо использовать алгоритмы сортировки, предполагающие наименьшее кол-во операций перестановки.

Если сортировка производится по таблице ключей, эффективнее использовать сортировки с наименьшей сложностью работы.