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»

Лабораторная Работа №2 «Записи с вариантами, обработка таблиц»

Вариант №1

Студент **Поляков Андрей Игоревич**

Группа **ИУ7-32Б**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент **Поляков А.И.**

Оценка _____

Описание условия задачи

Приобрести навыки работы с типом данных «запись» («структура») содержащим вариантную часть, и с данными, хранящимися в таблицах. Оценить относительную эффективность программы (в процентах) по времени и по используемому объему памяти в зависимости от используемого алгоритма и от объема сортируемой информации.

Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами (объединениями)). Упорядочить данные в ней по возрастанию ключей, двумя алгоритмами сортировки, где ключ – любое невариантное поле, 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, поэзия; 3: детская – минимальный возраст, сказки, стихи). Вывести список всех романов указанного автора.

Техническое задание

Исходные данные:

Файл с данными о книгах, каждое поле записано с новой строки.

Пункты меню, выраженные целыми числами 0-12. Внутри некоторых пунктов вводятся строки.

1) Считать массив из файла

2) Добавить запись в конец

3) Удалить запись

4) Вывести массив структур

5) Вывести массив структур с помощью массива ключей

6) Вывести массив ключей

7) Отсортировать массив структур

8) Отсортировать массив ключей

9) Вывести список всех романов указанного автора

10) Вывести результаты использования различных алгоритмов сортировок

11) Сохранить массив в файл

12) Вывести информацию о программе

0) Выход

Результат:

Таблица структур, таблица ключей, таблица структур, выведенная с помощью массива ключей, выборка романов указанного автора, результаты замеров времени работы алгоритмов и оценки их эффективности

Описание задачи:

Чтение, сохранение, вывод, сортировка массива структур, вывод всех романов указанного автора, оценка эффективности алгоритмов.

Способ обращения к программе:

Запуск с помощью ./app.exe

Аварийные ситуации и ошибки:

1. ERR_IO (Ошибка ввода): Введенные данные не соответствуют ожидаемому формату.

2. ERR_RANGE (Ошибка размера): Значение выходит за допустимые пределы.

3. ERR_OVERFLOW (Переполнение): Произошло переполнение.

4. ERR_ARGS (Ошибка аргументов командной строки): Ошибка в аргументах, переданных через командную строку.

5. ERR_NO_FILE (Файл отсутствует): Указанный файл не найден.

6. ERR_EMPTY (Массив пуст): Попытка обработать пустой массив.

7. ERR_NOT_FOUND (Структура не найдена): Запрашиваемая структура не найдена.

8. ERR_EXIT: Программа завершает выполнение.

9. ERR_WRONG_COMMAND (Некорректная команда): Введена некорректная команда.

10. ERR_NO_SPEC (Нет романов данного автора): Не найдено романов от указанного автора.

Описание внутренних структур данных

```c
#define SUR_SIZE 50
#define TITLE_SIZE 50
#define PUBLISHER_SIZE 50
struct book_t
{
    char surname[SUR_SIZE + 1];
    char title[TITLE_SIZE + 1];
    char publisher[PUBLISHER_SIZE + 1];
    int page_cnt;
    lit_types type;
    union type_chars_u
    {
        struct eng_chars_t eng;
        struct fict_chars_t fict;
        struct child_chars_t child;
    } chars;
};
```

Структура представляет книгу, с полями для фамилии автора, названия, издательства, количества страниц, типа литературы и объединения chars, которое может содержать характеристики разных типов книг в зависимости от lit_types.

- `surname` - фамилия автора (строка, размером до `SUR_SIZE + 1` символов)
- `title` - название книги (строка, размером до `TITLE_SIZE + 1` символов)
- `publisher` - издательство (строка, размером до `PUBLISHER_SIZE + 1` символов)
- `page_cnt` - количество страниц (целое число)
- `type` - тип литературы (перечисление `lit_types`)
- `chars` - объединение (`union`), содержащее характеристики книги в зависимости от типа:
    - Если `type` равен `ENG`, используются характеристики из `struct eng_chars_t`.
    - Если `type` равен `FICTION`, используются характеристики из `struct fict_chars_t`.
    - Если `type` равен `CHILD`, используются характеристики из `struct child_chars_t`.

```
typedef enum {TALES = 1, POEMS} child_types;

struct child_chars_t
{
    int min_age;
    child_types type;
};
```

Структура содержит характеристики детской литературы.

- `min_age` - минимальный возраст читателя (целое число)
- `type` - тип (перечисление `child_types`)

```
typedef enum {NOVEL = 1, PLAY, POETRY} fict_types;

struct fict_chars_t
{
    fict_types type;
};
```

Структура, представляющая характеристики художественной литературы.

- `type` - тип (перечисление `fict_types`)

```
#define IND_SIZE 50

typedef enum {ENG = 1,  FICTION, CHILD} lit_types;

typedef enum {RUS = 1, TRANSLATED} eng_lang;

struct eng_chars_t
{
    char industry[IND_SIZE + 1];
    eng_lang lang;
    int year;
};
```

Структура, представляющая характеристики технической литературы:

- `industry` - отрасль (строка, размером до `IND_SIZE + 1` символов)

- `lang` - язык (перечисление `eng_lang`)
- `year` - год издания (целое число)

Описание алгоритма

1. Отобразить пользователю список команд и дождаться ввода номера нужной команды.
2. При выполнении команды на добавление или удаление элемента из массива, проверить его размеры и выполнить операцию, если размеры соответствуют требованиям.
3. Команда считывания из файла считывает массив структур из файла.
4. Команда записи в файл записывает массив структур в файл.
5. Если выбрана команда сортировки, отсортировать массив.
6. Если выбрана команда сортировки массива ключей, отсортировать массив ключей.
7. При выборе замеров, создать копии массивов, отсортировать их, измерив время каждой сортировки. Оценить относительную эффективность программы (в процентах) по времени и по используемому объему памяти в зависимости от используемого алгоритма и от объема сортируемой информации.
8. При выборе команды “Вывести список всех романов указанного автора”, пройтись по массиву, выводя только нужные книги.
9. При выборе вывода информации о программе, вывести всю необходимую иныормацию.
10. При выборе выхода, закрыть файл и завершить программу.

Тестовые данные

Позитивные тесты

Тест	**Входные данные**	**Результат**
Добавить элемент в пустой массив	Пустой массив - книга	Массив с одной книгой
Добавить элемент в массив	Массив книг - книга	Массив книг + 1
Вывести пустой массив	Пустой массив	Пустая таблица
Вывести не пустой массив	Массив	Таблица книг

Вывести массив структур с помощью массива ключей	Массив книг, массив ключей	Таблица книг
Вывести массив ключей	Массив ключей	Таблица ключей
Удалить элемент в массиве из одного элемента	Массив с одной книгой, индекс	Пустой массив
Удалить элемент	Массив книг, индекс	Массив книг – 1
Вывести список всех романов указанного автора	Массив книг, фамилия автора	Список всех романов указанного автора
Прочитать массив из файла	Файл	Массив книг
Вывести массив в файл	Массив, файл	Заполненный файл
Отсортировать случайный массив структур	Случайный массив структур	Отсортированный массив структур
Отсортировать отсортированный массив структур	Отсортированный массив структур	Такой же массив
Отсортировать отсортированный в обратном порядке массив структур	Неотсортированный массив структур	Отсортированный массив структур
Отсортировать массив из одного элемента	Массив из одного элемента	Такой же массив
Отсортировать массив ключей	Массив ключей	Отсортированный массив ключей
Вывести результаты использования различных алгоритмов сортировок	Неотсортированный массив структур	Результаты замеров

Негативные тесты

Неверная команда в меню	23	Некорректная комманда
Некорректное невариантное	Кол-во страниц - abc	Ошибка ввода

поле		
Некорректное вариантное поле	Год издания - abc	Ошибка ввода
Слишком длинная строка	Фамилия длины 60	Ошибка размера
Файл не существует	Попытка прочитаь не существующий файл	Файл отсутствует
Превышение размера массива	Добавление 71 элемента в массив	Переполнение

Замеры сортировок

Программа замеряет время сортировки массива структур и массива ключей сортировкой пузырьком и быстрой сортировкой. Для каждого из тестов программа выполняет многочисленные замеры, пока их RSE не становится <5. Затем все тестовые случаи сравниваются и оценивается их эффективность. Оценивается также время выборки данных из основной таблицы с использованием таблицы ключей.

Сортировка пузырьком массива структур:

Размер массива	Время, нс	Кол-во итераций	RSE	Быстрее на n%
150	58692.30	20	4.92	19.9
300	281596.70	20	3.54	17.47
600	1060787.90	10	3.05	20.36

Сортировка пузырьком массива ключей:

Размер массива	Время, нс	Кол-во итераций	RSE
150	73365.75	10	3.12
300	341200.10	10	1.69
600	1331879.30	10	1.87

Сортировка массива структур в среднем быстрее на **19.24**%

Быстрая сортировка массива структур:

Размер массива	Время, нс	Кол-во итераций	RSE
150	6613.68	40	4.23
300	44377.50	40	4.53
600	101518.95	20	3.62

Быстрая сортировка массива ключей:

Размер массива	Время, нс	Кол-во итераций	RSE	Быстрее на n%
150	3841.45	40	4.59	41.92
300	24977.50	30	4.78	43.72
600	56299.93	30	3.56	44.54

Сортировка с помощью ключей в среднем быстрее на **43.39**%

Объем памяти массива ключей составляет 4% от объема массива структур.

Время выборки данных из основной таблицы с использованием таблицы ключей в среднем составляет 0.02% от времени сортировки.

Ответы на вопросы

1.Как выделяется память под вариантную часть записи?

Вариантная часть записи выделяется в памяти так, чтобы она имела достаточно места для хранения наибольшего варианта данных. В этом случае можно использовать указатель на вариантную часть, который будет указывать на начало выделенной области памяти.

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

Если данные в вариантной части не соответствуют описанному формату или структуре, это может привести к непредсказуемому поведению программы, включая ошибки, падения или утечки памяти.

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

Ответственность за правильность операций с вариантной частью лежит на программисте. Программист должен убедиться, что все операции с вариантной частью соответствуют её описанию и формату данных.

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

Таблица ключей представляет собой структуру данных, в которой хранятся ключи (например, уникальные идентификаторы) и связанные с ними указатели на соответствующие записи в таблице данных. Таблица ключей используется для эффективного поиска, сортировки и доступа к записям.

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

- Обработка данных в самой таблице эффективнее, когда операции требуют обращения к большому объему данных, и когда необходимо часто осуществлять поиск, обновление и вставку записей.

- Использование таблицы ключей становится предпочтительным, когда операции поиска и сортировки выполняются часто, а объем данных велик. Таблица ключей позволяет быстро находить нужные записи по ключам без необходимости просматривать все данные.

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

- Если таблица отсортирована редко, а часто происходят вставки и удаления, сортировка вставками может быть предпочтительнее, так как она эффективна при небольших объемах данных и обладает небольшой временной сложностью для частично отсортированных данных.

- Быстрая сортировка (quick sort) обычно предпочтительна для сортировки больших массивов, так как имеет хорошую временную сложность в среднем случае.

Вывод

Время от времени программисту приходится иметь дело с большими объемами данных, поэтому важно знать, как оценить различные алгоритмы и ускорить обработку данных. Например, можно использовать таблицу ключей или более эффективные методы обработки, такие как применение алгоритма Хоара для сортировки.