

ФАКУЛЬТЕТ: Информатика и системы управления

КАФЕДРА: Программное обеспечение ЭВМ и информационные технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

**«Записи с вариантами, обработка таблиц»**

Студент: Зайцева Алена Андреевна

Группа ИУ7 – 32Б

*2020 г.*

Оглавление

Оглавление

[1. Описание условия задачи 3](#_Toc52606500)

[2. Описание ТЗ 4](#_Toc52606501)

[1) Описание исходных данных: 4](#_Toc52606502)

[2) Описание результатов: 5](#_Toc52606503)

[3) Описание задачи, реализуемой программой: 5](#_Toc52606504)

[4) Способ обращения к программе: 6](#_Toc52606505)

[5) Описание возможных аварийных ситуаций и ошибок пользователя. 7](#_Toc52606506)

[3. Описание внутренних структур данных 7](#_Toc52606507)

[4. Описание алгоритма 8](#_Toc52606508)

[5. Набор тестов с указанием, что проверяется 9](#_Toc52606509)

[1) Негативные тесты на неверный ввод команды 9](#_Toc52606510)

[2) Негативные тесты на ввод полей записи 9](#_Toc52606511)

[3) Позитивные тесты при вводе полей записи 11](#_Toc52606512)

[4) Переполнение таблицы: 12](#_Toc52606513)

[5) Неверная последовательность команд: 12](#_Toc52606514)

[6) Тестирование отдельных команд: (при исключении вышеперечисленных ошибок) 12](#_Toc52606515)

[6. Оценка эффективности 14](#_Toc52606516)

[7. Выводы по проделанной работе 16](#_Toc52606517)

[8. Ответы на вопросы 17](#_Toc52606518)

# Описание условия задачи

* Создать таблицу, содержащую не менее 40 записей с вариантной частью (*список литературы, содержащий фамилию автора, название книги, издательство, количество страниц, вид литературы (1: техническая – отрасль, отечественная, переводная, год издания; 2: художественная – роман, пьеса, стихи; 3: детская – сказки, стихи))*
* Произвести поиск информации по вариантному полю. (*Вывести список отечественной технической литературы по указанной отрасли)*
* Упорядочить таблицу, по возрастанию ключей (где ключ – любое невариантное поле по выбору программиста *(мой выбор – фамилия автора)*), 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, то есть, в случаях а) и б). Обосновать выбор алгоритмов сортировки. Оценка эффективности должна быть относительной (в %).
* Интерфейс программы должен быть понятен неподготовленному пользователю. При разработке интерфейса программы следует предусмотреть:

∙ указание формата и диапазона данных при вводе и (или) добавлении записей;

∙ указание операций, производимых программой;

∙ наличие пояснений при выводе результата;

∙ возможность добавления записей в конец таблицы и удаления записи по значению указанного поля;

∙ просмотр отсортированной таблицы ключей при несортированной исходной таблице;

∙ вывод упорядоченной исходной таблицы;

∙ вывод исходной таблицы в упорядоченном виде, используя упорядоченную таблицу ключей

∙ вывод результатов сравнения эффективности работы программы при обработке данных в исходной таблице и в таблице ключей;

∙ вывод результатов использования различных алгоритмов сортировок.

Одним из результатов работы программы должна быть количественная информация (лучше представить в виде таблицы) с указанием времени, затраченного на обработку исходной таблицы и таблицы ключей двумя алгоритмами сортировки (при этом, не забыть оценить так же время выборки данных из основной таблицы с использованием таблицы ключей), а также - объем занимаемой при этом оперативной памяти.

При тестировании программы необходимо:

∙ проверить правильность ввода и вывода данных (в том числе, отследить попытки ввода неверных по типу данных в вариантную часть записи);

∙ обеспечить вывод сообщений при отсутствии входных данных («пустой ввод»);

∙ проверить правильность выполнения операций;

∙ отследить переполнение таблицы. При хранении исходных данных в файлах (что приветствуется) необходимо также проверить наличие файла и изменения информации в нем при удалении и добавлении данных в таблицу.

# Описание ТЗ

Внешняя спецификация:

## Описание исходных данных:

Программа работает со списком литературы (до 5000 записей), каждая запись содержит информацию о книге:

* фамилия автора (до 25 символов);
* название книги (до 25 символов);
* название издательства (до 25 символов);
* количество страниц (от 1 до 9999);
* вид литературы: (1, или 2, или 3)

1. техническая

* отрасль (до 15 символов);
* происхождение: (1 или 2)
  + 1 - отечественная
  + 2 - переводная;
* Год издания (от 0 до 2100);

1. художественная

* Вид: (1, или 2, или 3)
  + 1 - роман
  + 2 - пьеса
  + 3 - стихи;

1. детская

* вид: (1 или 2)
  + - * + 1 - сказки
        + 2 - стихи;

Список литературы можно ввести из специально подготовленного файла literature.txt или создать собственный. Также можно добавить записи к уже созданному списку.

## Описание результатов:

В зависимости от команды программа может вывести руководство по пользованию программой, список литературы в указанном формате, определенную запись/записи из списка, количество элементов в обрабатываемом списке литературы, таблицу ключей, сравнительную таблицу эффективности использования различных алгоритмов сортировки и подходов к использованию таблиц.

## Описание задачи, реализуемой программой:

Программа позволяет обрабатывать список литературы (таблицу, тип записей в которой – запись с вариантами (объединениями)). Программа, в зависимости от команды, может:

* показать информацию о программе и список возможных команд (*help*);
* показать информацию о формате и диапазоне вводимых значений (*how*);
* показать количество записей в существующем списке литературы (*amount*);
* очистить существующий в программе список литературы (*clean*);
* удалить записи по значению фамилии автора (*delete*);
* ввести в программу подготовленный список литературы из файла literature.txt (*input\_prepared*);
* ввести в программу ваш собственный список литературы (*input\_my*);
* добавить одну книгу в конец существующего списка литературы (*input\_one*);
* показать сравнительную таблицу эффективности использования различных алгоритмов сортировок и подходов к использованию таблицы (*show\_compare*);
* вывести список отечественной технической литературы по указанной отрасли (*show\_tech*);
* показать текущее состояние таблицы ключей (*show\_key\_table*);
* показать текущее состояние списка литературы (*show\_main\_table*);
* отсортировать список литературы по возрастанию количества страниц с использованием алгоритма "пузырек" и вывести отсортированный список (*sort\_main\_bubble*);
* отсортировать список литературы по возрастанию количества страниц с использованием алгоритма "быстрой сортировки" и вывести отсортированный список (*sort\_main\_quick*);
* вывести список литературы в отсортированном виде путем сортировки таблицы ключей по возрастанию количества страниц с использованием алгоритма "пузырек" (*sort\_key\_bubble*);
* вывести список литературы в отсортированном виде путем сортировки таблицы ключей по возрастанию количества страниц с использованием алгоритма " быстрой сортировки " (*sort\_key\_quick*);
* завершить выполнение программы (*exit*).

## Способ обращения к программе:

Для сборки исполняемого файла в папке есть файл makefile. Чтобы собрать исполняемый файл app.exe нужно написать команду “make app.exe”. Программа запускается из папки, содержащей исполняемый файл app.exe, командой “./app.exe”.

## Описание возможных аварийных ситуаций и ошибок пользователя.

Возможные ошибки пользователя:

1. Неверная команда
2. Использование не английского языка
3. Попытка обработать список до его ввода или создания
4. Неверный ввод записи или поля записи
5. Пустой ввод
6. При вызове команды для вывода списка отечественной технической литературы по указанной отрасли не найдено ни одной подходящей записи.
7. При вызове команды для удаления записи по значению фамилии автора не найдено ни одной подходящей записи.

Возможные аварийные ситуации:

1. при вводе числового поля пользователь ничего не вводит – программа будет ждать ввода и не завершится, пока не встретит хоть один непробельный символ.
2. В папке нет подготовленных мной файлов literature.txt, 5000\_books.txt

# Описание внутренних структур данных

1. Для представления книги (запись с вариантами (объединениями)) используется следующая структура:

typedef struct

{

char author[26];

char name[26];

char pub[26];

int pages;

int type\_l; //1-technical, 2 - fictional, 3- childish

union {

struct {

char field[16];

int origin; //1-domestic, 2 - translated

int year;

} tech;

int fict; //1-novel, 2 - play, 3 - poems

int child; // 1 - fairytale, 2 - poems

} type\_l\_inf;

} book\_struct;

* Для представления вариантной части выбран тип объединение, чтобы разделить между тремя возможными вариантами заполнения одну часть памяти;
* Для представления полей фамилия автора, название книги, название издательства и отрасль (для технической литературы) выбраны строки (массивы символов, оканчивающиеся нуль-символом), так как там будут храниться строки.
* Для представления полей количество страниц и год издания (для технической литературы) выбран целочисленный тип, так как это самый маленький по занимаемой памяти тип, который вместит допустимые значения
* Для представления полей тип литературы/произведения и происхождения – целый тип, так как он более удобный в пользовании, чем перечислимый тип, а строки проигрывают ему в объеме занимаемой памяти.

2) Для представления таблицы ключей выбрана целочисленная матрица, где первый столбец - индекс записи в исходной таблице, второй столбец - количество страниц в книге по этому индексу.

## Описание алгоритма

1. Выводится общая информация о программе, о формате используемых данных и списке возможных команд
2. Циклически:

* считывается команда;
* выполняется соответствующее действие или сообщается о неверно введенной команде;
* выводится результат выполнения действия или сообщение об ошибке,

пока пользователь не введет команду для выхода из программы

1. Осуществляется выход из программы

Код программы содержит комментарии по используемым в программе подпрограммам.

# Набор тестов с указанием, что проверяется

## Негативные тесты на неверный ввод команды

|  |  |  |
| --- | --- | --- |
| Что проверяет | Команда | Результат |
| Пустой ввод | ENTER | ERROR: unknown command  и  список возможных команд |
| Слишком длинная команда | Qwertyuiopasdfghjklzxcvbnm |
| Неизвестная команда | unknown |

## Негативные тесты на ввод полей записи

(общий набор тестов для команд *delete, input\_my, input\_one, show\_tech*, так как они используют один алгоритм ввода)

|  |  |  |
| --- | --- | --- |
| Что проверяет | Тест | Результат |
| Фамилия автора | ENTER | Сообщение об ошибке |
| Toolongmorethan25exactly26 |
| toooooooooooooolongggggggggggg |
| Название книги | ENTER |
| Toolongmorethan25exactly26 |
| toooooooooooooolongggggggggggg |
| Название издательства | ENTER |
| Toolongmorethan25exactly26 |
| toooooooooooooolongggggggggggg |
| Количество страниц | -10 |
| 0 |
| 10000 |
| 500000 |
| . |
| a |
| 1.1 |
| Вид литературы | -1 |
| 0 |
| 4 |
| technical |
| . |
| a |
| 1.1 |
| Отрасль (для технической литературы) | ENTER |
| More15exactlyy16 |
| toooooooooooooolongggggggggggg |

|  |  |  |
| --- | --- | --- |
| Происхождение (для технической литературы) | -1 | Сообщение об ошибке |
| 0 |
| 3 |
| domestic |
| . |
| a |
| 1.1 |
| Год издания (для технической литературы) | -1 |
| 2200 |
| . |
| a |
| 2001.1 |
| Вид (для художественной литературы) | -1 |
| 0 |
| 4 |
| . |
| a |
| 1.1 |
| Вид (для детской литературы) | -1 |
| 0 |
| 3 |
| . |
| a |
| 1.1 |

## Позитивные тесты при вводе полей записи

(общий набор тестов для команд *delete, input\_my, input\_one, show\_tech*, так как они используют один алгоритм ввода)

|  |  |  |
| --- | --- | --- |
| Что проверяет | Тест | Результат |
| Фамилия автора | A | Программа продолжит выполнение команды |
| Exactly25symbols789012345 |
| With tabs |
| Название книги | A |
| Exactly25symbols789012345 |
| With tabs |
| Название издательства | A |
| Exactly25symbols789012345 |
| With tabs |
| Количество страниц | 1 |
| 1000 |
| 9999 |
| Вид литературы | 1 |
| 2 |
| 3 |
| Отрасль (для технической литературы) | A |
| Exactly15symbol |
| With tabs |
| Происхождение (для технической литературы) | 1 |
| 2 |
| Год издания (для технической литературы) | 1 |
| 2100 |
| 2010 |
| Вид (для художественной литературы) | 1 |
| 2 |
| 3 |
| Вид (для детской литературы) | 1 |
| 2 |

## Переполнение таблицы:

* 1. При вызове команды *input\_my* на вопрос о количестве элементов указано, что их будет больше 5000 – сообщение об ошибке
  2. На момент вызова команды *input\_one* в списке уже содержится 5000 записей - сообщение об ошибке

## Неверная последовательность команд:

* Если команды *сlean, delete, input\_one, show\_tech, show\_key\_table, show\_main\_table, sort\** вызваны до команд ввода списка литературы (*input\_prepared, input\_my*) – выводится сообщение об ошибке - обрабатываемая таблица пуста.
* Команды ввода списка литературы (*input\_prepared, input\_my*) вызваны, когда в программе уже существует непустой список – выводится сообщение об ошибке и предложение очистить существующую таблицу.

## Тестирование отдельных команд: (при исключении вышеперечисленных ошибок)

|  |  |  |
| --- | --- | --- |
| Команда | Условия | Результат |
| *help* | любые | показана информация о программе и список возможных команд |
| *how* | любые | показана информация о формате и диапазоне вводимых значений |
| *amount* | любые | показано количество записей в списке литературы |
| *clean* | любые | очищен существующий в программе список литературы и таблица ключей |
| *delete* | Не найдено ни одной подходящей записи (нет совпадений по фамилии) | Сообщение об ошибке |
| Удалено n записей  (даже если удалены все записи) | Сообщение о том, что удалено n записей, и том, сколько записей осталось в таблице. Изменилась как исходная таблица, так и таблица ключей |
| *input\_prepared* | Файл literature.txt находится в папке, он не пустой и не поврежден | Сообщение о том, что успешно введено 48 элементов |
| В папке отсутствует файл literature.txt, или он пустой, или он поврежден | Сообщение о об ошибке |
| *input\_my* | Любые | Сообщение о том, что успешно введено n записей. Изменилась как исходная таблица, так и таблица ключей |
| *input\_one* | Любые | Сообщение о том, что запись успешно введена. Изменилась как исходная таблица, так и таблица ключей |
| *show\_compare* | Файл 5000\_books.txt находится в папке, он не пустой и не поврежден | Показана сравнительная таблица эффективности использования различных алгоритмов сортировок и подходов к использованию таблицы |
| В папке отсутствует файл 5000\_books.txt, или он пустой, или он поврежден | Сообщение о об ошибке |
| *show\_tech* | Не найдено ни одной подходящей записи | Сообщение об ошибке |
| Найдено n подходящих записей | Выведены n подходящих записей |
| *show\_key\_table* | Любые | показано текущее состояние таблицы ключей |
| *show\_main\_table* | Любые | показано текущее состояние списка литературы |
| *sort\_main\_bubble* | Любые | список литературы отсортирован по возрастанию количества страниц с использованием алгоритма "пузырек" и выведен на экран. Таблица ключей также изменилась |
| *sort\_main\_quick* | Любые | список литературы отсортирован по возрастанию количества страниц с использованием алгоритма "быстрая сортировка" и выведен на экран. Таблица ключей также изменилась |
| *sort\_key\_bubble* | Любые | Таблица ключей отсортирована по возрастанию количества страниц с использованием алгоритма "пузырек" и выведена на экран. Исходная таблица выведена на экран в отсортированном виде, но в программе не изменилась |
| *sort\_key\_quick* | Любые | Таблица ключей отсортирована по возрастанию количества страниц с использованием алгоритма "быстрая сортировка " и выведена на экран. Исходная таблица выведена на экран в отсортированном виде, но в программе не изменилась |
| *exit* | Любые | Сообщение о завершении работы программы, возврат нулевого кода |

# Оценка эффективности

Оценим эффективность различных алгоритмов сортировок (сортировки пузырьком и быстрой сортировки) и различных подходов к обработке таблиц (работа напрямую с исходной таблицей и с использованием таблицы ключей).

Обоснование выбора алгоритмов сортировки: были выбраны один из самых медленных на больших массивах алгоритмов – пузырьком, и один из самых быстрых – быстрая сортировка, чтобы замеры на наших данных были наиболее показательными

Средняя сложность алгоритма сортировки пузырьком – n\*n, быстрой сортировки – n\*log(n).

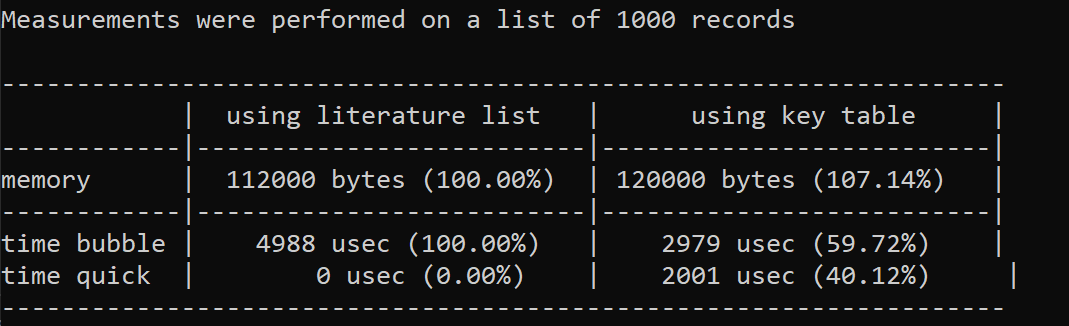
Память, занимаемая одной записью основной таблицы – 112 байт, таблицы ключей – 8 байт.

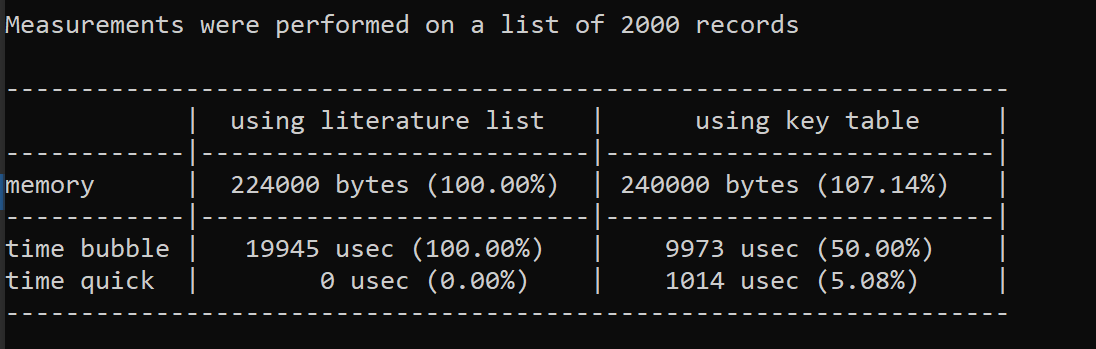
Оценивать будем в процентном отношении:

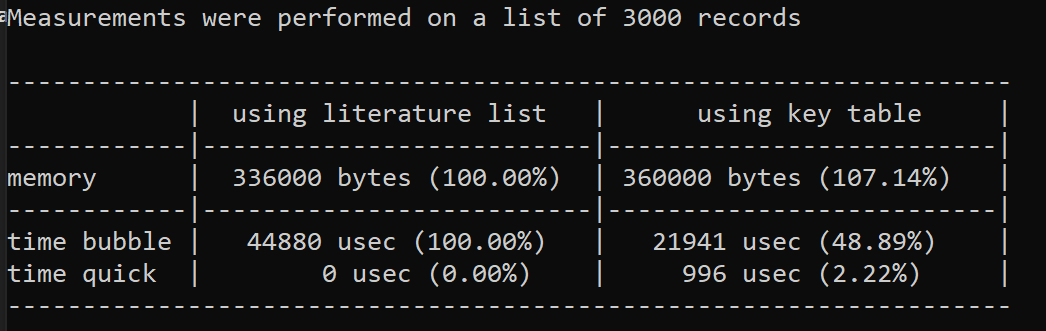
* по памяти: память, занимаемую только основной таблицей считаем за 100%.
* по времени: время, затраченное на сортировку исходной таблицы напрямую пузырьком считаем за 100%.

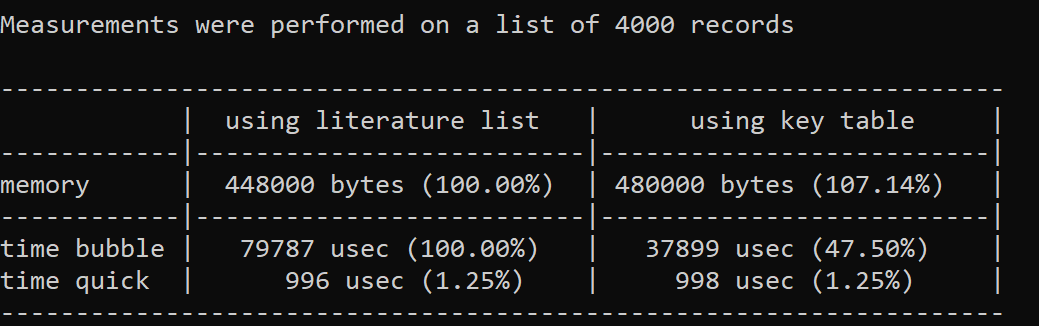
Проведем замеры для таблиц, с количеством записей от 1000 до 5000 с шагом 1000

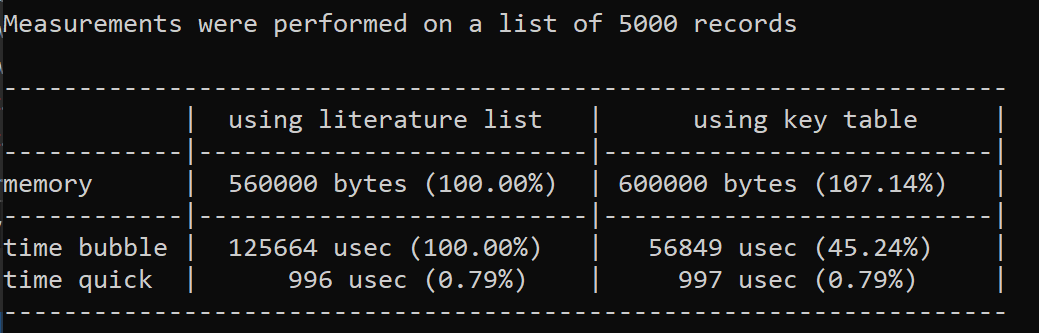
Примечание: при использовании таблицы ключей учитывается время выборки данных из основной таблицы с использованием таблицы ключей.











# Выводы по проделанной работе

* Итак, преимущество использования типа «запись» с вариантной частью в том, что один участок памяти может быть разделен между всеми возможными значениями вариантной части, что значительно экономит оперативную память. Недостаток такого подхода - в том, что программист должен сам контролировать правильность обработки данных, хранящихся в вариантной части, что может повлечь трудноуловимые ошибки.
* Преимущество использования дополнительной таблицы при обработке объемных таблиц – в сокращении времени обработки исходной таблицы (несмотря на дополнительное время на выборку данных из основной таблицы с использованием таблицы ключей), так как перестановка или изменение объемных данных – записей таблицы – долгая операция. Однако у этого подхода есть и недостаток – под хранение дополнительной таблицы ключей также нужно дополнительное место в памяти.
* Как и ожидалось, чем длиннее сортируемый массив, тем лучше себя показывает “быстрая сортировка” (в сравнении с пузырьковой). Но на коротких массивах пузырек может показывать себя даже лучше быстрой сортировки. Это связано с вычислительной сложностью алгоритмов.

Для принятия решения о том, стоит ли использовать подход, который подразумевает использование дополнительной таблицы ключей для обработки объемных таблиц, стоит оценить, насколько много записей будет храниться в основной таблице, а также количество полей в одной записи (так как при большом их количестве время перестановки или изменения записей будет достаточно большим). Как показали мои измерения в пункте 6, отношение памяти, требуемой для размещения таблицы ключей, к памяти, необходимой для основной таблицы, постоянно. А вот с ростом числа записей время обработки таблицы с использованием описанного подхода все больше выигрывает у времени обработки основной таблицы без использования таблицы ключей.

# Ответы на вопросы

1. Как выделяется память под вариантную часть записи?

под вариантную часть записи выделяется один участок памяти, размер которого равен размеру максимального поля, которое может быть записано в вариантную часть.

1. Что будет, если в вариантную часть ввести данные, не соответствующие описанным?

Тип данных в вариантной части при компиляции не проверяется, поэтому, контроль за правильностью ее использования возлагается на программиста. То есть, если в вариантную часть ввести данные, не соответствующие описанным, то память будет заполнена некорректно, и в дальнейшем правильно использовать эти данные в программе будет невозможно.

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

Программист.

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

Таблица ключей - это дополнительный массив (матрица), содержащей индекс элемента в исходной таблице и соответствующий выбранный ключ (интересующее нас поле). Она нужна для уменьшения времени обработки исходной таблицы.

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

Если мы сортируем таблицу ключей, то экономится время, поскольку перестановка записей в исходной таблице, которая иногда может содержать достаточно большое число полей, не выполняется. Этот выигрыш во времени особенно заметен при большой размерности таблиц и при правильно подобранных ключах. При этом, правда, надо помнить, что для размещения таблицы ключей требуется дополнительная память. Выбор данных из основной таблицы в порядке, определенном таблицей ключей, замедляет вывод этих данных.

Таким образом, с ростом количества записей в таблице, все более эффективным становится использование таблицы ключей, так как отношение памяти, необходимой для хранения ключей, к памяти, отведенной под хранение исходной таблицы, не меняется, а выигрыш по времени обработки (сортировки, поиска и т.д.), увеличивается.

1. Какие способы сортировки предпочтительнее для обработки таблиц и почему.

Так как в записях таблицы хранится много данных, некоторые из них занимают большой объём данных, а изменение или перестановка записей - затратная по времени операция, то предпочтительнее использовать сортировки, в которых происходит меньше всего перестановок записей.

При большом количестве записей также эффективней применить подход с использованием таблицы ключей, так как перестановка записей (а перестановки в любом случае потребуются) в этой вспомогательной таблице будет выполняться значительно быстрее, чем в исходной.