МГТУ им. Н.Э. Баумана

**Дисциплина Типы и Структуры данных.**

**Лабораторный практикум №2**

**по теме:** «Записи с вариантами, обработка таблиц»

Работу выполнил:

студент группы ИУ7-35Б

Прянишников Александр

**Цель работы**: 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

# Условие задачи. Вариант 1

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; 3: детская – сказки, стихи). Вывести список отечественной технической литературы по указанной отрасли.

# Требования к задаче

**Ввод данных**

Первоначальное чтение производится из исходного файла “base.txt”, который расположен в одном каталоге с программой. Каждое поле каждой записи находится на новой строке, программа автоматически считывает данные в массив. Общее количество возможных записи ограничено – 999. На каждое поле также действуют свои ограничения:

1. Длина фамилии автора, названия произведения, издательского дома, а также отрасли среди технических книг, ограничена 30 знаками. Это сделано для того, чтобы таблица вмещалась на полный экран в красивом виде.
2. На поля, значениями которого являются целые числа, назначены ограничения только в порядках здравого смысла: нельзя, например, добавлять книги, которые были написаны после 2020 года.
3. Некоторые поля (язык: отечественный или иностранный) вводятся как целые числа, но у них есть строгий набор дискретных значений, от которых пользователь отступить не может.

**Вывод данных**

Вывод происходит для следующих данных: таблица значений, таблица ключей, сравнение времени сортировок.

Также в программу был добавлен интерфейс, упрощающий работу для пользователей: есть пояснительные надписи, перед вводом данных написано, как правильно вводить данные.

**Описание задачи, реализуемой программой**

Программа реализует базу данных, позволяет хранить в структурируемой форме список книг с вариантными полями. У пользователя есть возможность добавлять новые книги, удалять старые по инвариантным записям (индекс, автор, название книги, количество страниц, издательский дом), производить поиск по инвариантным полям записей, сортировать быстрой сортировкой и сортировкой пузырьком, со вспомогательной таблицей и без неё. Программа также сравнивает все четыре способа, показывает численное значение сортировок, а ещё выводит список отечественной технической литературы по заданной отрасли.

**Способ обращения к программе.**

Программу можно запустить через MSYS либо через QT Creator и любую среду разработки.

Выполняется следующая последовательность команд: *./main.exe*

**Описание возможных аварийных ситуаций и ошибок пользователя.**

**Ошибки пользователя при работе с меню и массивом**

* Выбор отсутствующего пункта меню (от 0 до 12)  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и попросит ввести новое значение пункта меню.*
* Ввод букв при вводе целочисленного значения  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и попросит ввести новое значение пункта меню.*
* Выбор отсутствующего значения в поле записи при строгих дискретных значениях  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и попросит ввести новое значение пункта меню.*
* Пустой ввод  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и попросит ввести новое значение пункта меню.*

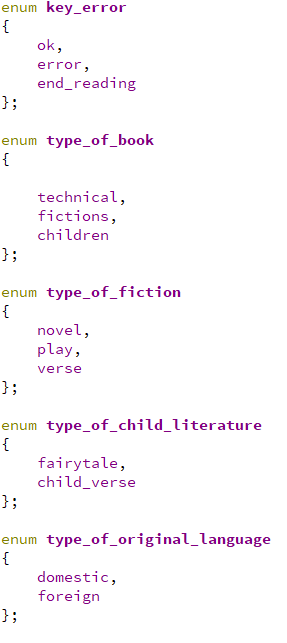
**Предусмотренные аварийные ситуации**

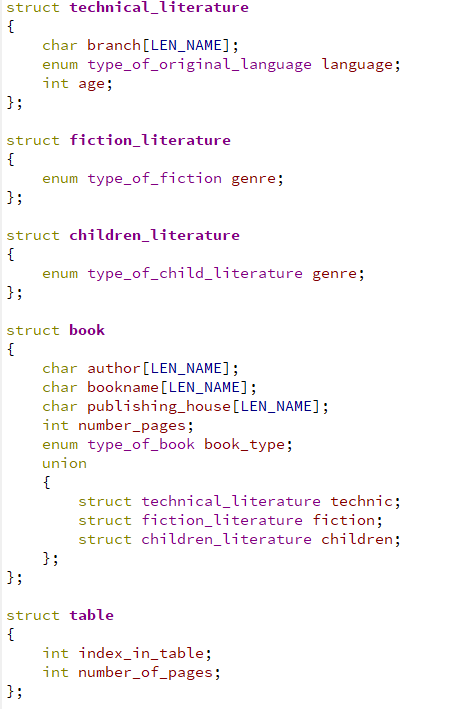
* Попытка добавления новой записи при уже максимальной длине массива  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и выйдет в главное меню*
* Попытка удаления из пустого массива  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и выйдет в главное меню.*
* Отсутствие результата при поиске по полям  
  *Программа сообщит пользователю, что не найдено нужных записей.*

# Описание внутренних структур данных.

Для решения данной задачи мной был выбран вариант хранения данных в структуре, состоящей из инвариантных и вариантных полей, каждая из которых также хранится в структуре. Это позволяет эффективно хранить книги, которые состоят из разных элементов, также это позволяет экономить память, так как используется union для вариантных полей. При этом под таблицу также выделена структура для удобства хранения. Для хранения дискретных значений используется тип данных enum.

Вот так описаны структуры:





# Описание алгоритма

Сначала программа загружает в программу базу данных и преображает её согласно закодированным структурам. Это производится автоматически, после чего на экран выводится полная база со всеми книгами, а также перечень пунктов меню, в котором пользователь выбирает предпочитаемый:

**1 – Добавление в массив**. Пользователь указывает для добавляемой книги автора, название произведения, издательский дом, количество страниц, а затем подбирает нужные вариантные поля. Программа строго следит за правильностью ввода, и если ввод осуществлён неправильно, то пользователю предлагают ввести поле заново с соответствующей надписью.

**2 – Удаление из массива**. Пользователь может удалить любую запись по инвариантному полю: индекс в таблице (первый элемент идёт под первым индексом, это сделано для удобства обычного пользователя), автор, название книги, издательский дом и количество страниц. При этом удаляются все книги, которые соответствуют заданному значению.

**3 – Поиск по массиву.** Пользователь может найти количество вхождений заданного значения из инвариантных полей.

**4 – Вывести массив**. Пользователь может вывести на экран полный массив структур в его текущем состоянии (сортировки не сохраняются).

**5 – Вывести сортированную таблицу**. Пользователь может вывести на экран таблицу отсортированных значений по количеству страниц. Таблица состоит из исходных индексов, индекса в базе и значения количества страниц.

**6 – Вывести результат быстрой сортировки без таблицы**. Пользователь может вывести на экран быстрой сортировки по количеству страниц без использования таблицы.

**7 – Вывести результат быстрой сортировки с таблицей**. Пользователь может вывести на экран быстрой сортировки по количеству страниц с использованием таблицы.

**8 – Вывести результат сортировки пузырьком без таблицы**. Пользователь может вывести на экран сортировки пузырьком по количеству страниц без использования таблицы.

**9 – Вывести результат сортировки пузырьком с таблицей**. Пользователь может вывести на экран быстрой сортировки по количеству страниц с использованием таблицы.

**10 – Сравнить время сортировки различных способов.** Программа выводит результат сравнения по времени сортировки таблицы способы 6–9.

**11 – Вывести список отечественной технической литературы по отрасли.** Пользователь может ввести отрасль, после чего программа, проверив на корректность ввод, выведет на экран требуемый список.

**12 – Сохранение данных.** Программа перезаписывает файл с исходной базой.

**0 – Выход.**

# Тесты

Тестирование происходило по такому принципу: каждый пункт тестировался отдельно, после чего контролировалась работа программы целиком. Поэтому здесь указаны только негативные тесты, которые в целом проверяли работоспособность модулей программы.

|  |  |
| --- | --- |
| **123345fd** | **Буквы при задании числа** |
|  | **Пустой ввод** |
| **–1** | **Выбор неправильного пункта меню** |
| **15** | **Выбор неверного значения поля** |
| **qwertyuiopasdfdsfgkjgfdvcxzvjsdffksdj** | **Слишком длинный ввод поля** |
|  | **Переполнение таблицы при добавлении** |
|  | **Удаление из пустого массива** |
|  | **Отсутствие найденных книг** |

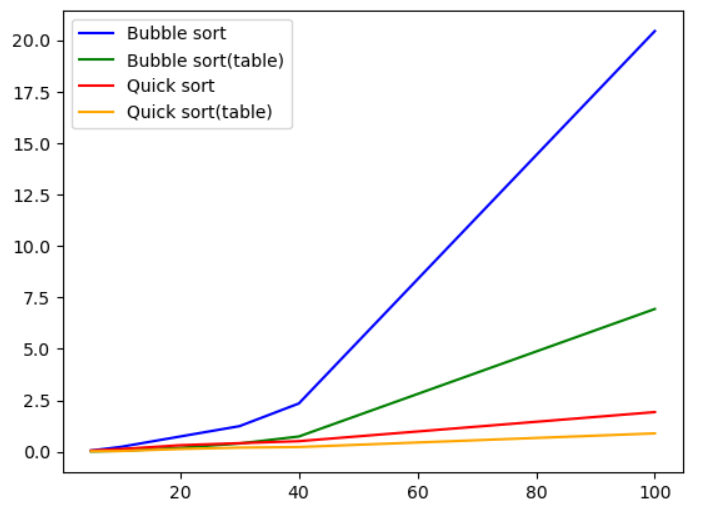
Положительные тесты: вся база данных была вбита в ручном режиме, таким образом была проверена вся программа и каждый её пункт.

# Оценка эффективности

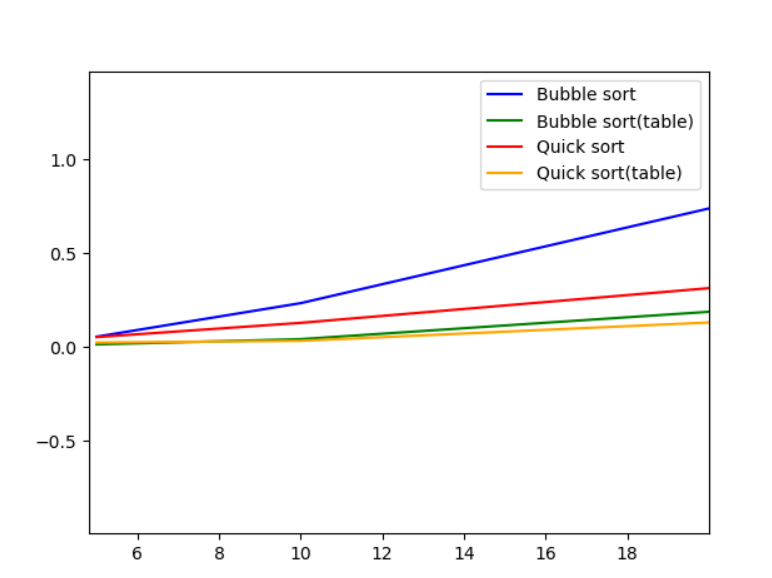
Одна книга занимает в памяти 136 байт, но львиную долю из них занимают строчные значения названия, автора, то есть то, что невозможно оптимизировать. Таблица же занимает всего 8 байт в пересчёте на одного человека, то есть проигрыш по памяти составляет всего 6%. Общее количество занимаемой памяти зависит от N записей в пропорциональной зависимости.

Согласно теоретическим расчётам, быстрая сортировка имеет зависимость NlogN, а сортировка пузырьком – N2. Составим таблицу значений при различных N записей, чтобы проверить закон, а затем визуализируем:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Bubble | Bubble(Table) | Quick | Quick(Table) |
| N = 5 | 0.056 | 0.015 | 0.054 | 0.025 |
| N = 10 | 0.235 | 0.043 | 0.130 | 0.034 |
| N = 20 | 0.740 | 0.189 | 0.315 | 0.132 |
| N = 30 | 1.243 | 0.400 | 0.415 | 0.199 |
| N = 40 | 2.345 | 0.739 | 0.512 | 0.227 |
| N = 100 | 20.460 | 6.942 | 1.927 | 0.891 |



При маленьких N:



Можно сделать несколько выводов. Во–первых, быстрая сортировка действительно гораздо эффективнее, чем сортировка пузырьком, так как уже при N = 100 первая реализует алгоритм в девять раз быстрее, а это небольшая база данных. При этом заметно, что при совсем маленьких N «пузырёк» с таблицей показал себя эффективнее всех, но затем вложенность принесла плоды, и график неизбежно вырос вверх. Теоретические зависимости подтвердились.

Во–вторых, таблица оказалась крайне эффективным средством. В обоих сортировках вариант с таблицей был в 3–4 раза эффективнее, но с увеличением N разница уменьшается, что связано с большим количеством обменов, так что такое поведение естественно.

Подводя итог: таблица сократило время работы программы на 50–70%, а быстрая сортировка позволяет уменьшить время на 40–60%, но проценты растут пропорционально N, поэтому точное число назвать нельзя.

# Выводы по проделанной работе

Для хранения баз данных отлично подойдёт тип данных «запись», но корректную работу должен обеспечивать программист, особенно если дело касается вариантных полей. Получилось, что быстрая сортировка при больших N действительно гораздо эффективнее, чем сортировка пузырьком. Таблица даёт существенный выигрыш при сортировке базы данных, так как экономится огромное количество времени на обмене структур между собой, и чем больше полей, тем существенней проявляет себя экономия. Затраты памяти при этом растут пропорционально N, но они не играют большую роль в оценке эффективности программы.

# Ответы на контрольные вопросы

Как выделяется память под вариантную часть записи?

Вариантное поле в записи может быть только одно и оно всегда последнее. В свою очередь последнее поле в вариантной части может быть вариантным, то есть вариантное поле так же может быть вложенным.

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

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. Если эта ситуация не контролируются, то возможны утечка памяти и аварийное завершение программы.

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

Программист.

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.

Если мы сортируем таблицу ключей, то экономится время, поскольку перестановка записей в исходной таблице, которая иногда может содержать достаточно большое число полей, отсутствует.

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

При больших N и большом количестве полей гораздо выгодней обрабатывать таблицу ключей, в обратном случае затраты на дополнительную память могут быть не оправданы.

Какие способы сортировки предпочтительнее для обработки таблиц и почему?

Самый эффективный способ сортировки – быстрая сортировка, так как зависит от NlogN, при этом использование таблицы ключей позволяет довести временные затраты до минимума.