МГТУ им. Н.Э. Баумана

**Дисциплина «Типы и структуры данных»**

**Лабораторная работа №2**

**«Записи с вариантами. Обработка таблиц»**

Работу выполнил:

студент группы ИУ7-31

Сангинов Азамат

Москва, 2018

**Цель работы**: приобрести навыки работы с типом данных «запись» (структура), содержащим вариантную часть (объединение, смесь), и с данными, хранящимися в таблицах, произвести сравнительный анализ реализации алгоритмов сортировки и поиска информации в таблицах, при использовании записей с большим числом полей.

**Условие задачи**: 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.

**1. Введение**

**1.1 Наименование программы**

«Программа для работы с базой данных из литературы, поиска, добавления и удаления».

**1.2 Характеристика области применения**

Данная программа позволяет работать с базой данных, состоящей из литературы. Программа может применяться в любой сфере, где требуется работа со структурами, содержащими несколько полей.

**1.3 Срок выполнения**

2 недели.

**2. Основания для разработки**

**2.1 Заказчик**

Преподаватели по дисциплине «Типы и структуры данных»

**2.2 Исполнитель**

Сангинов Азамат – студент группы ИУ7-31Б

**2.3 Основания для разработки**

Учебный процесс

**3. Назначение разработки**

**3.1 Общая концепция системы**

Данная программа выполняет операцию деления целого числа длиной до 30 цифр на вещественное число с длиной мантиссы до 30 цифр и порядка до 5 цифр.

**3.2 Функциональность**

Программа выполняет операцию деления и только с числами, попадающими в некоторый диапазон. Программа не предназначена для выполнения других операций.

**4. Требования к программе**

**4.1 Требования к информационным структурам и методам решения**

Данные хранятся в структуре с вариантным полем. Структуры оформлены в виде списка, для удобства работы с ними, в частности, для удобства добавления/удаления элемента и сортировки, путем смены указателей.

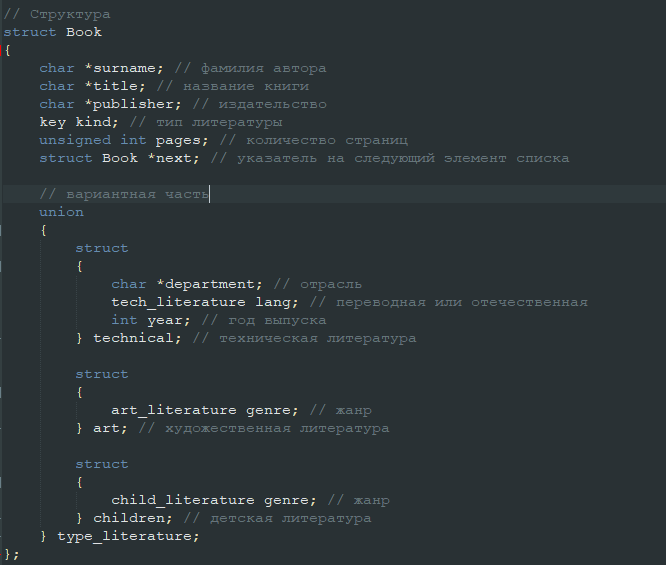
**4.2 Требования к функциональным характеристикам**

Данная программа должна:

1. Вывод таблицы на экран.
2. Добавления записей в конец таблицы и удаления записи по значению указанного поля.
3. Сортировка таблицы по невариантному полю.
4. Поиск в таблице по указанному полю.
5. Вывод результатов использования различных алгоритмов сортировок.

**Описание структур данных**

**Структура:**



**Функции:**

**void info(void) –** справочная информация.

**unsigned long long tick(void) –** измерение времени**.**

**ssize\_t my\_getline(char \*\*lineptr, size\_t \*n, FILE \*stream) –** считывание строки из файла.

**struct Book \*create\_new\_book(char \*surname, char \*title, char \*publisher, key kind, unsigned int pages, char \*department, tech\_literature lang, int year, art\_literature genre0, child\_literature genre1) –** создание нового элемента списка.

**struct Book \*add\_end(struct Book \*head, struct Book \*b) –** Добавление элемента в конец списка.

**void print(struct Book \*head) –** печать всей таблицы.

**void save(struct Book \*head, FILE \*f) –** сохранение в файл.

**void load(struct Book \*\*head, FILE \*f) –** загрузка из файла.

**int add\_el(struct Book \*head) –** добавление нового элемента пользователем.

**struct Book\* delet\_by\_id(struct Book \*head, int id) –** удаление элемента по его номеру.

**int print\_by\_dep(struct Book \*head) –** печать элемента по отрасли.

**void partition(struct Book \*head, struct Book \*\*front, struct Book \*\*back) –** разделение списка пополам.

**struct Book\* mergeLists(struct Book \*a, struct Book \*b) –** слияние списков.

**void mergeSort(struct Book \*\*source) –** сортировка слиянием.

**struct Book\* clear( struct Book \*node ) –** очистка списка.

**void list\_bubble\_sort(struct Book \*\*head) –** сортировка пузырьком.

**Тесты**

1) на работоспособность

**Контрольные вопросы**

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

Выделяется память под самый большой элемент вариантной части.

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

Неопределенное поведение, т.к. компилятор не учитывает тип данных вариантной части.

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

Программист.

**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 запись таблицы состоит из большого количества полей выгоднее использовать таблицу ключей, иначе позволительно использовать саму таблицу.

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

При реализации таблицы в виде односвязного линейного списка предпочтительно использовать сортировку слиянием.

Если не используется таблица ключей, то предпочтительно использовать сортировки, которые предполагают наименьшее количество перемещений элементов. Иначе следует использовать методы сортировки с наименьшей сложностью.