У меня есть мечта…

В течение трёх лет делался и тестировался на разных книгах набор скриптов для подготовки индексных указателей. Первая версия появилась в 2020 году. Всего сразу не предусмотришь, поэтому были и доработки скриптов, и создание новых. В процессе этой работы к концу 2022 года стало ясно, что в основной таблице для подготовки указателя надо добавить ещё два столбца, чтобы заметно расширить возможности по подготовке указателей.

Но это не самое главное озарение, гораздо важнее была другая мысль: *важно вернуть редактора в активный процесс работы над указателем*.

То, что было сделано в плане скриптов и безупречно работает — это всё инструменты для верстальщика. В данном случае *для меня лично* было большим интересом иметь такой набор. Но совершенно очевидно, что далеко не все верстальщики хотят делать указатели. Во-первых, эта работа должна оплачиваться отдельно от вёрстки, а работодатель может не понимать, что это совсем другая работа. Во-вторых, тут часто надо принимать решения по оформлению указателя, что и как включать в него — те решения, которые по делу должен бы делать редактор. Но в индизайне нет удобного инструмента взаимодействия верстальщика и редактора на этапе работы с указателем.

А ведь именно редактор, его знание и понимание книги может и должно быть основой качественной подготовки книжных указателей.

На мой взгляд, основная причина, по которой сейчас нет таких отлично сделанных указателей, какие были в книгах, сделанных ручным набором, — это невозможность активного участия редакторов в их подготовке. Раньше работа с гранками предполагала только знание материала, а потом наборщик готовил указатель. А в технологии компьютерной вёрстки редактору и автору в процессе работы с указателем места нет.

Я мечтаю, чтобы в книгах, свёрстанных в индизайне, были такие же проработанные редактором указатели, как это было в книгах ручного набора. И главным инсайтом, пришедшим в процессе тестирования первой версии (2020-2022), стало понимание *как организовать взаимодействие редактора и верстальщика в работе над указателем*.

Это сотрудничество в работе над индексом можно сделать в виде использования таблиц, и обычных текстовых файлов, в которых будут отражаться результаты постепенного создания указателя. Редактор (или автор) может не знать InDesign, но он должен знать Word и Акробат.

Такое решение вернёт редакторов в активный процесс подготовки указателей, и есть надежда, что в книгах снова появятся добротные полноценные указатели, а не бесполезные для работы сделанные для галочки списки отдельных терминов, которые есть в тексте в именительном падеже.

Дальше об этом взаимодействии специалистов рассказывается подробно, прямо по шагам, кто что делает. Вы удивитесь, как всё это просто.

Взаимодействие редактора и верстальщика

Совместная работа этих специалистов над указателем должна быть организована так, как описано ниже. В такой группе руководителем будет редактор, он принимает все решения по оформлению указателя.

1. Подготовка таблицы для получения указателя

1.1. Редактор готовит таблицу-двухколоник (с. 6, 7).

1.2.Верстальщик скриптом **UseReadyTable.jsx** (с. 8) превращает таблицу в файл IndexList-nnn.indd.

1.3. Верстальщик скриптом **TermCompare.jsx** (с. 9) обрабатывает этот файл. Редактору передаётся файл-рабочая таблица IndexList@TermCompare.txt и PDF-файл рабочей таблицы IndexList-nnn.indd.

1.4. Редактор должен убедиться, что в таблице:

- поиск терминов определён так, как требуется: или только в именительном или во всех падежах (это видно по запросам);
- поиск фамилий не оформлен как поиск слов;
- предложенные в файле IndexList@TermCompare.txt названия символьных стилей есть в четвёртой колонке рабочей таблицы.

П.п. 1.1, 1.3, 1.4 — тут используются обычные текстовые файлы и PDF, редактору не надо знать индизайн, чтобы начать руководить процессом подготовки указателя.

1.5. Если редактор решает, что в таблицу надо добавить строки с новыми терминами, он готовит обычный текстовый файл, в котором определяет, что это за термины — слова или фамилии, в каких падежах, инициалы и пр. (с. 9-10), а таблицу обновляет верстальщик при помощи скрипта **GetInfoForSearch.jsx**. После этого снова пункты 1.3 и 1.4.

П. 1.5 — это снова руководящая работа редактора, и это опять просто работа с текстом.

1.6. После этого верстальщик запускает скрипт **ShowFoundByGrep.jsx** и отдаёт редактору текстовый файл ...@foundByGrep.txt. (с. 11). Полезность этой процедуры в том, что выполняется тот же поиск, что будет при подготовке указателя, но не тратится время на сохранение информации в указателе, т.к. сейчас важно только проверить, что всё будет правильно найдено.

Редактор проверяет правильность работы grep-запросов, в случае необходимости какие-то запросы в рабочей таблице изменяются.

П. 1.6 — редактор в получает для проверки текст с всеми словами, которые программа готова поместить в указатель. Формат каждой информационной единицы в этом файле очень простой:

термин — слова, найденные для него.

Это именно редакторская работа убедиться, что всё найдено верно.

1.7. Когда редактор решает, что файл IndexList-nnn.indd соответствует поставленной задаче, верстальщик скриптами **ApplySpecialStyles.jsx** и **MarkSameQueries.jsx** переносит стилевую разметку, указанную в четвёртой колонке рабочей таблицы, в файл вёрстки (с. 12-14).

1.8. Если это один из нескольких указателей, то по указанию редактора изменяется название файла IndexList-nnn.indd, это делает верстальщик скриптом **AddMarker.jsx** (с. 21).

П. 1.8 — редактор выбором строчных маркирующих букв определяет, в какой очерёдности будут размещаться конкретные указатели.

1.9. Если в работе несколько указателей, то там возможны совпадения терминов. Скрипт **FindSameItems.jsx** собирает информацию о таких совпадениях в текстовый файл ...@SameTermsInfo.txt (это файл для редактора), скрипт **ColorSimilarLines.jsx** отмечает их цветом.

Надо отметить такие термины уникальными символьными стилями (с. 22).

2. Получение начального варианта указателя

2.1. Верстальщик скриптом **ProcStoryOrDoc.jsx** (с. 15) готовит указатель, собирает его на отдельной странице [Указатель > Построить указатель], после этого скриптом **ProcNumberLines.jsx** приводит в порядок номера страниц (с. 23, 24).

2.2. Этот указатель в виде PDF-файла отдаётся редактору. Также редактор должен получить от верстальщика текстовый лог-файл обработки grep-запросов.

2.3. Редактор должен убедиться, что нет потерянных ссылок, для этого надо попробовать в лог-файле найти текст не удалось поместить в индексный указатель (с. 20).

П.п. 2.2 и 2.3 — это понятная редактору работа с PDF и обычным текстовым файлом.

3. Оформление указателя

3.1. Если требуется преобразовать глухой указатель в аннотированный, то редактор должен подготовить текстовый файл аннотационных данных, это двухколонник, в левой колонке термины, в правой аннотационные данные к ним. Эта таблица отдаётся верстальщику, и он скриптом **AddAnnotationData.jsx** переделывает указатель в аннотированный (с. 25).

Редактор также должен определить, как оформлять этот указатель (с. 25).

3.2. Дальше начинается тонкая настройка указателя или указателей: если они двухуровневые, то можно сделать ссылку "см." для всех терминов второго уровня: скрипт **AddSeeTopic.jsx** (с. 26, 27). Если надо фамилии в скобках сделать ссылками, для этого есть скрипт **NameAndNick.jsx** (с.27, 28).

3.3. Многолетняя, если не сказать многовековая традиция оформления указателей предписывает заменять на прочерки совпадающие слова в соседних строках. Раньше это было кропотливой редакторской работой, сейчас данную задачу за редактора выполняет скрипт **DashInsteadWord.jsx** (с.28, 29).

3.4. И конечно, редактор должен обязательно внимательно посмотреть PDF-файл вёрстки, где найденные термины отмечены цветом. В акробате есть хорошо продуманный инструментарий добавления комментариев, и это тоже вариант взаимодействия редактора и верстальщика. Там можно отметить какие-то случайно попавшие в указатель слова, чтобы верстальщик их удалил скриптом **DeleteUnnecessarySign.jsx** (с. 23).

П.п. 3.1, 3.2 и 3.4 — это живая редакторская работа с текстом: доведение до ума подготовленного по предоставленному словнику указателя.

3.5. Особенностей указателей может быть много, всего в данной инструкции не предусмотришь, но наступит момент, когда указатель будет готов. И после этого его надо сделать алфавитным, чтобы читателю с ним было удобно работать.

Редактор определяет, как будут оформлены буквы, и верстальщик запуском скрипта **AddLetter.jsx** завершает работу над указателем (с. 30, 31).

Вот три этапа подготовки указателя. Данное описание помещено в начале инструкции, это текст в первую очередь для редакторов, кто тоже хочет, чтобы в книгах были удобные для читателя, хорошо сделанные подробные указатели. Название программ в описании может смутить, поэтому синим цветом и курсивом выделено объяснение именно для редакторов — какова их задача на каждом шаге, и показано, что для них это привычная работа с обычным текстовым файлом или PDF-файлом, а не в программе вёрстки.

Описание этапов очень краткое, поэтому и редактору, и верстальщику надо обязательно прочесть этот документ полностью. В описании программ есть акценты на работе для каждого специалиста.

И что касается индексирования, программы проверены не на одной книге, всё работает.

Я начал заниматься решением проблемы отсутствия хорошего инструмента для создания указателей в русских книгах в 2006 году, сперва это были макросы в Word, потом скрипты в индизайне. И вот к концу 2022 года всё, что нужно для этого, сделано. Если нужна помощь в освоении этих программ, обращайтесь.

Михаил Юрьевич Иванюшин
dotextok@gmail.com | dotextok.ru

Программы

AddAnnotationData.jsx
AddLetter.jsx
AddMarker.jsx
AddSeeTopic.jsx
ApplySpecialStyles.jsx
ColorSimilarLines.jsx
DashInsteadWord.jsx
DeleteUnnecessarySign.jsx
FindSameItems.jsx
ForIndex.jsxinc
GetInfoForDearch.jsx
LogFileUsage.jsx
MarkSameQueries.jsx
NameAndNick.jsx
ProcNumberLines.jsx
ProcStoryOrDoc.jsx
ShowFoundByGrep.jsx
TermCompare.jsx
UseReadyTable.jsx
WorkLoad.jsx

Рабочие папки

AddLinesToIndex
Info
sets
Стили индекса (Index Styles)

Создание в индизайне указателей разных видов

В прошлом веке, во время использования горячего набора, указатели были в большинстве исторических и технических книг. Это было в порядке вещей, несмотря на то что требовало недели если не месяцы кропотливой ручной работы с текстом гранок.

Сейчас вёрстка компьютерная, и вроде как работать с текстом можно эффективнее. Это верно, но книги с указателями не стало особо легче делать.

Программа индизайн, работа которой тут будет обсуждаться, позволяет делать индексы, но интерфейс ввода терминов для разных уровней индексации продуман только для случаев, когда в словах нет окончаний. Это нормально для английского, немецкого, но с нашими падежными формами такой подход не годится. В результате в русских книгах предметные, именные и прочие указатели иногда есть, но если термины только те, что в книге есть в именительном падеже, и ссылки только на эти страницы, то такой указатель не особо полезен.

Единственное потенциально работающее решение — использовать grep-запросы, имитирующие изменение слов по падежам. Но и тут не всё просто.

Над указателем должны вместе работать редактор и верстальщик. Редактор определяет, что должно быть в указателе, а верстальщик должен понимать возможные проблемы создания конкретного варианта указателя и уметь их решать.

Проблемы создания указателей

Будем называть терминами отдельные записи указателя, неважно какой он по сути — предметный, именной, географический, и т.д.

Проблемы создания русскоязычных указателей следующие.

1) Совпадение слов в разных терминах. Например, в словнике могут быть термины *Сопротивление* и *Удельное сопротивление*. Если сперва искать в тексте падежные формы термина *Сопротивление*, то в выборку попадут и слова, перед которыми есть слово *удельное*.

2) Другой примечательный пример при работе с именными указателями — фамилия *Александрович*. Она всегда в первых строках словника, и если она уйдёт в обработку первой, то отметится во всех отчествах *Александрович*, что абсолютно неправильно. Это тоже задача пользователя — обработать все термины с отчеством *Александрович* до того как будет обработана фамилия *Александрович* — т.е. подтверждение того, что специалист, собирающий индекс, должен знать текст.

3) Одинаковые названия, имеющие разное значение. Вот пример из словника географических терминов:

Брянск I, станция, Брянская обл., РСФСР
Брянск II, пос., Брянская обл., РСФСР
Брянск, г., РСФСР
или

Биржайская волость, Литовская ССР
Биржайский уезд, Литовская ССР

Надо иметь возможность при поиске отличать Брянск I, Брянск II и Брянск город. Тоже самое с волостью и уездом.

4) Разные написания названия города или фамилии в рамках одного документа.

Вот примеры городов:

Йоэнсуу (Иоэнсуу), г., Финляндия
Святнаволок (Свят–Наволок), с., КФССР

А это варианты фамилий:

Саррай (Саррайль) Морис
Хаккы–паша (Хакки–паша) Ибрагим

Тут не принимаются глубокомысленные сентенции де «документы должны быть оформлены единообразно...», если это оцифрованные сканы бумаг, то там важно оставить всё так, как было при создании данного документа. Но у нас должен быть такой инструмент, для которого не проблема искать термины с разными вариантами написания.

5) Содержащиеся в термине **фамилия имя отчество** обычно бесполезны для поиска. В подавляющем большинстве случаев в тексте фигурируют только фамилии. Как в таких случаях искать именно по фамилиям, не помещая в запрос поиска имя и отчество?

Здесь даже не упомянуто, что для учёта падежей должна быть грамотная генерация grep-запросов, исходя из того, на какую букву заканчивается слово. Это само собой разумеющееся требование.

Как правильно подойти к решению

Можно добавлять термины в указатель так — идти по тексту и выбирать нужные слова. Наверное, это приемлемо, когда, например, в тексте какое-то слово встречается много раз, а для индекса надо только несколько этих слов из конкретных абзацев.

Решение, реализованное тут, берет на себя поиск всех терминов, во всех падежных формах. Но и описанный в предыдущем абзаце подход тоже реализован как частный случай (с. 12).

Нужно, чтобы редактор книги подготовил отдельный текстовый файл со всеми терминами, которые должны быть в указателе, и отметил, на каком уровне иерархии каждый из них.

Например, для книги по астрономии в этом файле может быть такой текст:

Планеты
  \Венера
  \Земля
  \Марс
  ...
Созвездия
  \Большая Медведица
  \Кассиопея
  \Лебедь
  ...
Звёзды с именами
  \Альтаир
  \Вега
  \Денеб
  ...

Тут термины **Планеты, Созвездия, Звёзды с именами** — это первый уровень, а названия астрономических объектов — второй. Редактор должен отметить термины второго уровня обратной наклонной чертой. И для терминов второго уровня будут искаться номера страниц.

Вот пример именного указателя, в котором все записи первого уровня:

Авксентьев Николай Дмитриевич
Аксельрод Александр Ефремович
Алексеев Михаил Васильевич
Андроников Михаил Михайлович
Антонов-Овсеенко Владимир Александрович
Астафьев Николай Анатольевич
Багратуни Яков Герасимович
Балабанов Маврикий Леонтьевич
....

Но просто списка терминов недостаточно, даже если отметить обратной наклонной чертой термины второго и следующих уровней. В реальной работе термин в том виде, как он есть в указателе, очень часто в самом тексте отсутствует. Поэтому для использования этого набора скриптов редактор должен подготовить таблицу-двухколонник: в левой колонке указать термины, как они должны выглядеть в указателе, а в правой возможные варианты этих терминов в книге.

Проблема разметки: совпадение слов разных терминов

Допустим, есть гипотетическая книга о Пушкине и Грибоедове. У них имя-отчество совпадают. В тексте могут быть как фамилии, так и имя-отчество, так и просто имя. В указателе надо для них отметить, что один был поэтом, а второй — поэтом и дипломатом. И таблица для работы программы подготовки указателя будет, как на рис. справа.

Тут Пушкин А. С. и Грибоедов А. С. это термины первого уровня, а поэт и дипломат — это термины второго уровня. Знак обратной наклонной черты — указатель, что это термин следующего уровня. Две последних строки этой таблицы содержат одинаковые фамилии: для понимания особенностей работы этого набора скриптов предположим, что в этой книге есть упоминания о двух малоизвестных поэтах-однофамильцах пушкинской поры — Василии Ивановиче Туманском и Фёдоре Антоновиче Туманском, но по какой-то причине их имена в индекс не попали.

Пушкин А. С.	Пушкин
Пушкин А. С.	Александр
Пушкин А. С.	Александр Сергеевич
\поэт	
Грибоедов А. С.	Грибоедов
Грибоедов А. С.	Александр
Грибоедов А. С.	Александр Сергеевич
\дипломат	
\поэт	
Туманский	
Туманский	

Данная таблица — учебный вариант вполне реального словника, но даже не стоит пытаться решить задачу создания указателя по этому списку, используя возможности индизайна, вы потратите очень большое число часов, чтобы пытаться решить имеющимся в индизайне инструментом данную задачу.

Если в исходном двухколонник правая ячейка пустая, то скрипт скопирует в неё текст из левой ячейки.

Вот какие с точки зрения возможностей штатного инструмента тут есть проблемы:

- в этом словнике есть совпадающие слова для поиска — Александр и Александр Сергеевич — относящиеся к разным терминам. Тут это, конечно, надуманный случай, но существуют реальные тексты, например, в индийских эпосах есть разные персонажи с одинаковыми именами, но мне тут важно на простом обозримом примере показать, как обрабатываются одинаковые термины разных уровней;
- есть термины второго уровня — поэт и дипломат, при этом термин поэт относится к разным терминам первого уровня — Пушкин А. С. и Грибоедов А. С.;
- есть два совпадающих термина;
- искомые слова могут быть в разных падежах. Это, пожалуй, самое проблемное условие, ставящее жирный крест на использовании штатного инструмента. Никто в здравом уме не будет терять время на подготовку слов для их поиска во всех возможных падежных формах в том неприспособленном для этого окне.

Таблица «Термин – grep»

Двуколонник отдаётся верстальщику. В папке со скриптами в каталоге **Стили индекса** есть файл **Стили индекса.idml**, его надо перенести в рабочую папку проекта. В этом файле есть группа абзацных стилей **#IndexStyles**, в ней четыре абзацных стиля **#Level1** – **#Level4** и служебный стиль **#PageShow**. Потом открыть этот idml-файл и поместить в него таблицу-двухколонник.

Теперь можно запустить скрипт **UseReadyTable.jsx**, он из этой таблицы сделает файл **IndexList-nnn.indd**.

Внизу результат работы скрипта, в левой колонке синим цветом отмечены термины первого уровня, чёрным — второго. Одинаковые термины теперь отличаются, скрипт добавил индексы _1 и _2.

Во второй колонке № или # для слов, которые надо искать в тексте. Если ячейка пустая, слово не ищется.

В третьей колонке grep-запросы для поиска слов, помещённых в правую колонку. Неважно, сколько слов в крайней правой ячейке, grep-запрос найдёт этот текст во всех падежных формах. Все запросы — не конкретно именно для данного слова, они обоб-

Пушкин А. С.	№	\bПушкин[мауоыеё]*\b		Пушкин
Пушкин А. С.	№	\bАлександр[мауоыеё]*\b		Александр
Пушкин А. С.	№	\bАлександр[мауоыеё]*\b\h\bСергеевич[мауоыеё]*\b		Александр Сергеевич
поэт_1	№	\bпоэт[мауоыеё]*\b		поэт
Грибоедов А. С.	№	\bГрибоедов[мауоыеё]*\b		Грибоедов
Грибоедов А. С.	№	\bАлександр[мауоыеё]*\b		Александр
Грибоедов А. С.	№	\bАлександр[мауоыеё]*\b\h\bСергеевич[мауоыеё]*\b		Александр Сергеевич
дипломат	№	\bдипломат[мауоыеё]*\b		дипломат
поэт_2	№	\bпоэт[мауоыеё]*\b		поэт
Туманский_1	№	\bТуманск[ийайяогюемуивхё]+\b		Туманский
Туманский_2	№	\bТуманск[ийайяогюемуивхё]+\b		Туманский

щённые, и делаются для похожих слов. Например, наш поэт Пушкин и шотландский писатель Кронин. По правилам русской грамматики творительный падеж для русских и иностранных фамилий не совпадает: Пушкиным, но Кронином. Таковы правила, но в grep-запрос не встроишь выяснение гражданства. Проще в запросе иметь обе буквы. Поэтому запрос такой:

\bПушкин[мауоыеё]*\b

Буква ё добавляется во все наборы возможных букв, на всякий случай. Подробно о создании запросов читайте в главе **Опыт настройки grep-запросов.**

Четвёртая колонка для помещения имён уникальных символьных стилей, которые надо использовать, если разные термины имеют совпадающие grep-запросы. Сейчас она пустая, но совпадения будут, т.к. одинаковое имя и имя-отчество Пушкина и Грибоедова и термин второго уровня — слово поэт, поэтому для этих случаев надо иметь отдельные символьные стили. И поиск совпадений должен делать не редактор или верстальщик, а программа, этот скрипт называется TermCompare.jsx.

Таблица сохраняется с именем **IndexList-nnn.indd**, тут **nnn** — это порядковый номер. Если, например, в одной работе два вида указателей, то таблицы для географического и именного указателей будут иметь разные номера.

Уточнение grep-запросов

При первом создании таблицы IndexList все grep-запросы делаются в предположении, что это обычные слова и нужны запросы для всех падежей. Но это могут быть не обычные слова, а фамилии, и для них другие правила падежных склонений. Так, например, слова молоко, танки изменяются по падежам, а похожие по написанию фамилии Руденко, Бианки не склоняются.

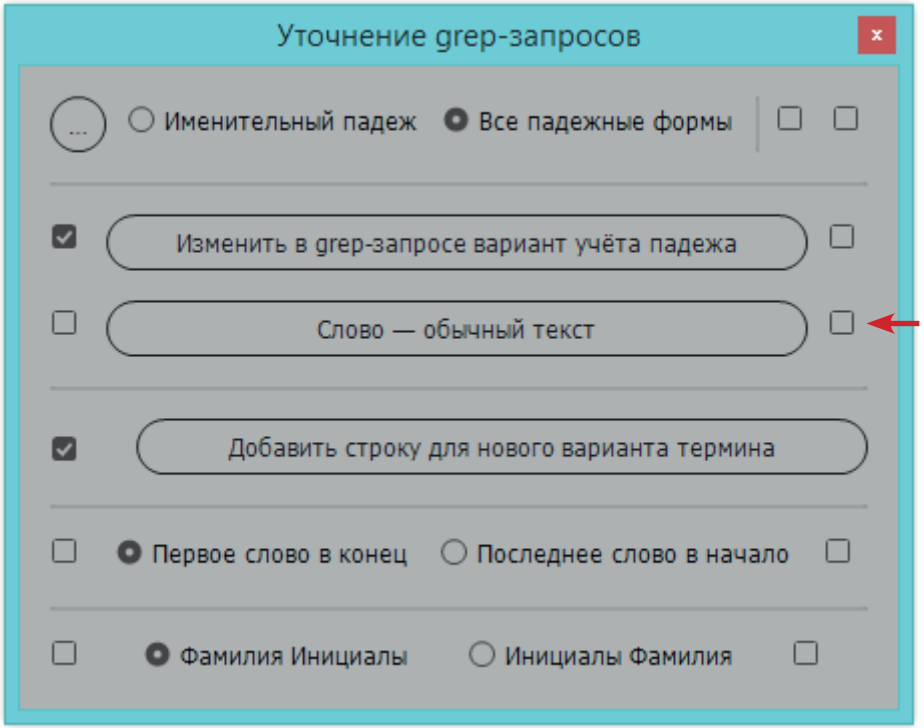
Слова для создания grep-запроса берутся из правой крайней колонки.

Иногда требуется иметь grep-запросы только в именительном падеже. Редко, но были случаи, когда требовалось сохранить для поиска знаки пунктуации и знаки ударения. Для этого добавлен флажок справа от кнопки **Изменить в grep-запросе вариант учёта падежа.**

Очерёдность инициалов и фамилий в поиске может отличаться от того, что есть в термине готового указателя: иногда надо оставить только фамилию, иногда надо поменять местами слова; может потребоваться в фамилии-имени-отчестве оставить только инициалы. В общем, практика такова, что термин в указателе и слова в тексте, относящиеся к этому термину, могут очень сильно различаться. Все эти задачи решаются скриптом GetInfoForSearch.jsx, всплывающие подсказки к кнопкам и флажкам достаточно информативны.

Совпадение искомых слов мы уже видели в том примере, где Пушкина и Грибоедова предполагалось искать по их одинаковым именам и одинаковым именам-отчествам. Предполагается, что это готовит редактор до начала вёрстки, но опыт показывает, что иногда надо расширить поиск в процессе работы.

Эта задача решается кнопкой **Добавить строку для нового варианта**



термина. В добавленной строке в первой ячейке останется тот же термин. Если флажок слева от этой кнопки установлен, то текст в левой ячейке не будет виден. В правой ячейке надо поместить новые варианты поиска этого термина.

Например, в книге о полководцах в правой колонке текст для поиска термина Барклай-де-Толли был такой:

- Барклай-де-Толли
- Барклай
- Михаил Богданович

и уже скрипт помещал в ячейки третьей колонки grep-запросы для поиска во всех падежах.

Это очень мощное решение — возможность искать термин, используя разные варианты текста!

Тут можно быстро подготовить разные варианты для поиска, этот скрипт делался в первую очередь для работы с фамилиями и именами. В словнике принято указывать первым фамилию, потом имя и отчество, а в тексте обычно только фамилия. Для таких случаев предусмотрен флажок-кнопка, оставляющий в тексте третьей ячейки только первое слово. На картинке он справа от кнопки **Слово — это фамилия** (отмечен красной стрелкой).

Поменять в новом варианте термина имя и фамилию местами, поставить фамилию после имени-отчества, превратить имя-отчество в инициалы — все эти операции подготовки данных для grep-поиска выполняются кнопками окна уточнения grep-запросов.

Учёт в grep-запросе знаков ударения и учёта регистра

Бывают тексты, в которых термин может быть указан с ударением. Это случаи достаточно редкие, чтобы учитывать их во всех grep-запросах, и вместе с тем когда в слове есть знак ударения, это знак с кодом 0x301, то слово найдено не будет, если данного кода в запросе нет. Флажок добавления кода ударения в grep-запрос расположен в окне уточнения grep-запросов, вверху второй справа.

Флажок-кнопка учёта регистра находится в том же окне вверху, крайний справа.

Важная задача подготовки текста перед получением указателя — отметить разными символьными стилями одинаковые термины. "Одинаковые" означает, что их grep-запросы совпадают.

Пушкин А. С.	№	\bПушкин[мауоыеё]*\b		Пушкин
Пушкин А. С.	№	\bАлександр[мауоыеё]*\b	Пушкин А. С.	Александр
Пушкин А. С.	№	\bАлександр[мауоыеё]*\b\h\bСергеевич[мауоыеё]*\b	Пушкин А. С.	Александр Сергеевич
поэт_1	№	\bпоэт[мауоыеё]*\b	поэт_1_Пушкин А. С.	поэт
Грибоедов А. С.	№	\bГрибоедов[мауоыеё]*\b		Грибоедов
Грибоедов А. С.	№	\bАлександр[мауоыеё]*\b	Грибоедов А. С.	Александр
Грибоедов А. С.	№	\bАлександр[мауоыеё]*\b\h\bСергеевич[мауоыеё]*\b	Грибоедов А. С.	Александр Сергеевич
дипломат	№	\bдипломат[мауоыеё]*\b		дипломат
поэт_2	№	\bпоэт[мауоыеё]*\b	поэт_2_Грибоедов А. С.	поэт
Туманский_1	№	\bТуманск[ийайяогюемуивхё]+\b	Туманский_1	Туманский
Туманский_2	№	\bТуманск[ийайяогюемуивхё]+\b	Туманский_2	Туманский

Поиск совпадающих grep-запросов

Возможна ситуация, когда разные термины имеют одинаковые grep-запросы — как в учебном примере, разные термины Грибоедов и Пушкин имеют одинаковые слова для поиска — Александр Сергеевич. О таких случаях специально говорилось, когда объяснялось создание рабочей таблицы IndexList-nnn.indd. Надо такие случаи найти, собрать информацию о них в текстовом файле и предложить понятные названия символьных стилей для оформления таких текстов. Эти задачи выполняет скрипт **TermCompare.jsx**.

При обнаружении совпадающих запросов он помещает в четвёртую ячейку строк с совпавшими терминами название символьных стилей. Информация о совпадениях в файле IndexList-nnn.indd помещается в файл IndexList@TermCompare.txt, он в той же папке, где IndexList-nnn.indd.

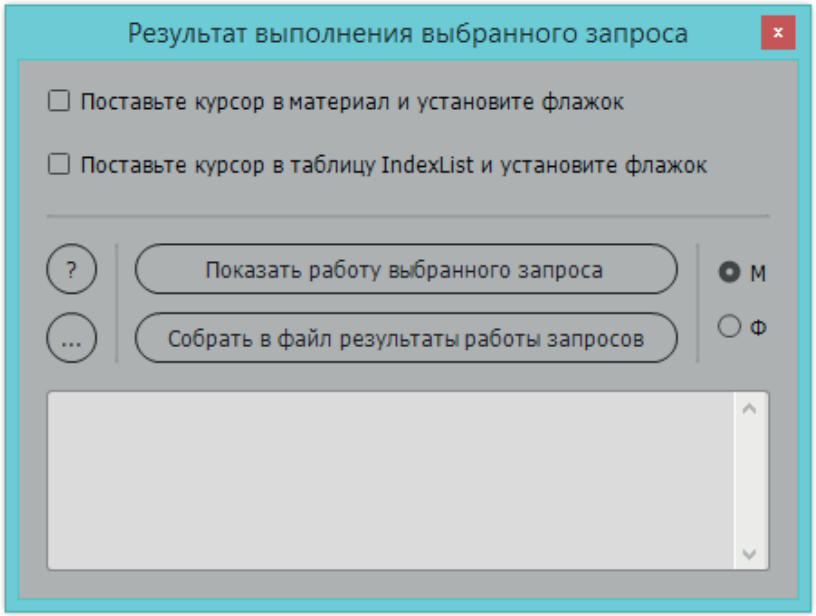
Учебный файл IndexList-nnn.indd, обработанный этим скриптом, показан вверху.

Проверка правильности работы запросов

В первой версии были скрипты проверки правильности выполнения запросов. Они хорошо справлялись с задачей, но это было действие *после того, как указатель уже собран*. И если находились ошибки, то надо было переделывать какие-то строки в файле IndexList, удалять неверные результаты, и обрабатывать исправленные строки заново. В первых книгах таких ошибок было много, но после того, как русские слова были разделены по группам в зависимости от окончаний, ошибки фактически исчезли. Об этом подробно написано в главе **Опыт настройки grep-запросов.**

Но хотелось иметь возможность проверять, верно ли работают запросы, *до начала сбора указателя*. И в конце концов был придуман скрипт, решивший этот вопрос. Он называется **ShowFoundByGrep.jsx**.

Должны быть открыты файл вёрстки и таблица IndexList-nnn.indd. Выбираете запрос, нажимаете кнопку **Показать работу выбранного запроса**, и в окне будут выведены всё слова, найденные в вёрстке с использованием этого запроса. Радиокнопки: **М** — поиск в материале, **Ф** — поиск в файле.



Нажатие на кнопку **Собрать в файл результаты работы запросов** сохранит обычный текстовый файл, в котором будет такая информация:

Термин указателя | искомый текст | grep-запрос
Результат поиска

Если вёрстка называется kniga.indd, то этот файл будет называться kniga@foundByGrep.txt.

И потом этот файл надо отдать на проверку редактору. Тут нужен свежий глаз, ведь как ни старайся делением слов на группы сузить область поиска, ошибки возможны. В частности, в этом учебном примере стандартный grep-запрос для слова поэт во всех падежах

\bпоэт[мауоыеё]*\b

неожиданно нашёл слово поэтому. Конечно, редактор это увидит, сравнивать искомые слова с тем, что реально найдено, проще, чем просматривать всю вёрстку.

Что касается этого grep-запроса, то исправим его на

\bпоэт[мауоыеё]{0,2}\b

и такой ошибки не будет.

Оформление уникальных слов своими стилями

Одна из хлопотных задач разметки текста перед сбором указателя — *отметить одинаковым символьным стилем разные слова* или *одно слово, встречающееся в разных местах*.

Как учебный пример можно привести задачу сбора всех упоминаний братьев в сказке П. П. Ершова "Конёк-Горбунок". Пример одного слова, встречающегося на разных страницах — слово "старче" в Сказке о рыбаке и рыбке" А.С. Пушкина. С помощью скриптов UseReadyTable.jsx и GetInfoForSearch.jsx мы подготовим таблицу IndexList-nnn.indd, укажем в четвёртой колонке название символьного стиля. Но как создать этот стиль в тексте вёрстки, как найти эти слова и оформить данным стилем?

Для этого есть скрипт ApplySpecialStyles.jsx. Должны быть открыты два файла — IndexList-nnn.indd и файл вёрстки. Пользователь определяет область обработки, файл или материал, дальше скрипт добавляет нужные символьные стили, применяет их к найденным словам. После обработки название стиля в четвёртой колонке и слово в пятой колонке будут зачёркнуты. Это сделано для того, чтобы исключить попадание их в обработку скриптом MarkSameQueries.jsx, описанным дальше.

Оформление одинаковых слов разными стилями

Если говорить о совпадении запросов поиска имени Александр, их имени-отчества, слова поэт в книге о Пушкине и Грибоедове, или где в тексте какой поэт Туманский упоминается, то задача верстальщика найти все эти имена и слова, и каждое из них отметить нужным символьным стилем.

И для этого тоже есть скрипт **MarkSameQueries.jsx** (стартовое окно показано слева на следующей странице), он запускается после того как программа **TermCompare.jsx** поместит в четвёртую колонку файла IndexList.indd предлагаемые имена символьных стилей. Это вторая задача стилевого оформления текста, и надо сперва выполнить первую — запустить программу **ApplySpecialStyles.jsx**.

Должно быть открыто два файла — файл вёрстки, где надо найти совпадающие термины и отметить их символьными стилями, и файл IndexList-nnn.indd с именами символьных стилей в четвёртой колонке.

Грибоедов А. С.	\bАлександр[мауоыеё]*\b	Грибоедов А. С.	Александр
Пушкин А. С.	\bАлександр[мауоыеё]*\b	Пушкин А. С.	Александр
Грибоедов А. С.	\bАлександр[мауоыеё]*\b\h\bСергеевич[мауоыеё]*\b	Грибоедов А. С.	Александр Сергеевич
Пушкин А. С.	\bАлександр[мауоыеё]*\b\h\bСергеевич[мауоыеё]*\b	Пушкин А. С.	Александр Сергеевич
Туманский_1	\bТуманск[ийайяогюемуивхё]+\b	Туманский_1	Туманский
Туманский_2	\bТуманск[ийайяогюемуивхё]+\b	Туманский_2	Туманский
поэт_1	\bпоэт[мауоыеё]*\b	поэт_1_Пушкин А. С.	поэт
поэт_2	\bпоэт[мауоыеё]*\b	поэт_2_Грибоедов А. С.	поэт

Программа сперва попросит указать, где файл вёрстки, а где таблица. Затем этот скрипт сделает копию файла IndexList, файл с именем IndexList@MarkSameQueries.indd в нём будет сделано следующее: а) оставлены только строки, в которых в четвёртой ячейке есть название символьного стиля и которые не обрабатывались скриптом **ApplySpecialStyles.jsx**, б) строки сортируются по содержимому колонки с запросами, чтобы термины с одинаковыми запросами оказались в соседних строках, в) вторая колонка удалена. Для нашего примера такой файл показан вверху. Из всех строк таблицы оставлены только те, в которых есть совпадающие слова и символьные стили для их оформления. И с этой таблицей дальше будет работа.

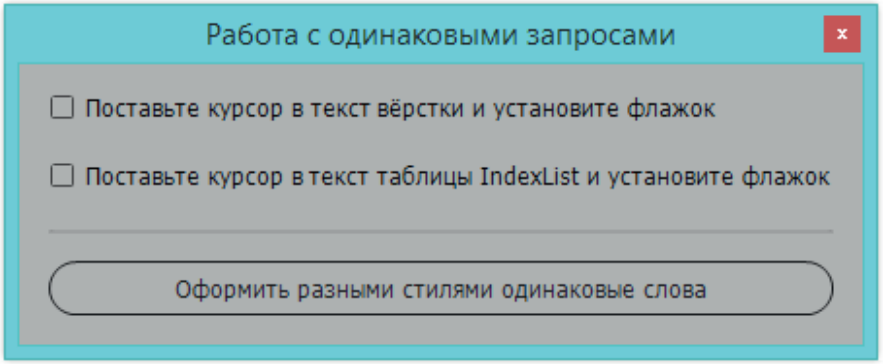
Итак, что есть в этой таблице:

левая колонка — термины, как они должны быть написаны в указателе. Сохранено стилевое оформление разных уровней терминов. Служебные номера _1, _2 удалит верстальщик после сбора указателя.

вторая колонка — grep-запросы. Строки этой таблицы упорядочены именно по этим запросам: одинаковые запросы стоят рядом.

третья колонка — предлагаемые названия символьных стилей для одинаковых запросов. Пока не приступили к маркировке, эти стили ещё не созданы и предложенные названия можно изменить. Например, убрать инициалы А. С. из них.

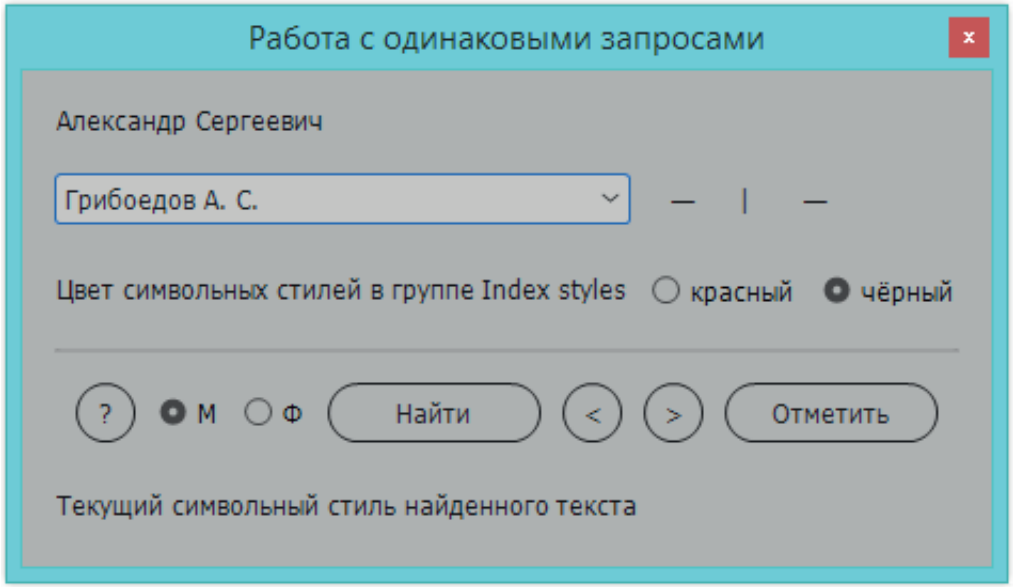
правая колонка — слова, для которых сделаны grep-запросы. Эти слова будет искать верстальщик, определять, к какому термину найденное слово относится, искать в тексте и указывать, каким символьным стилем его отметить.



В первых книгах, где использовался этот набор программ, поиск одинаковых слов, относящихся к разным терминам, отнимал заметно много времени. Да, в штатных возможностях индизайна есть grep-поиск, но когда надо просматривать текст, чтобы понять, какой символьный стиль нужен, и всякий раз подставлять его в поле замены, то это уже тягомотный процесс, убивающий на корню всю радость творчества во время вёрстки. Вот как эта хлопотная задача поиска таких слов решается скриптом **MarkSameQueries.jsx**.

Допустим, в этой книге надо найти все падежные формы слов Александр Сергеевич, и если речь о Грибоедове, то оформить эти слова символьным стилем Грибоедов А. С,, а если повествование о Пушкине, то использовать символьный стиль Пушкин А. С. Эти две строки с совпадающими запросами отмечены на предыдущей странице жёлтым цветом. Выделяем данные строки, нажимаем на кнопку **Оформить разными стилями одинаковые слова.**

Текущее окно исчезнет и появится другое.
Над выпадающим списком указан искомый термин, в выпадающем списке символьные стили для оформ-



ления этого текста. Нажатие на кнопку Найти откроет файл вёрстки, будут найдены все слова. Теперь можно по ним перемещаться кнопками < и > , выбирать в выпадающем меню нужный символьный стиль и отмечать им слово. На кнопке ? есть подробная справка.

Возможный конфликт символьных стилей

ApplySpecialStyles.jsx и **MarkSameQueries.jsx** прикладывают к найденным словам символьные стили. Но теоретически эти слова могли быть до этого оформлены символьными стилями. Поскольку мы готовим вёрстку для получения указателя, то у слов будет оформление, сделанное этими скриптами. Но визуальный вид этих слов не изменится — шрифт, кегль, интерлиньяж, привязка к базовой сетке и выключка будут сохранены.

Основная программа для индексирования

Итак, словник превращён в рабочую таблицу, для каждого термина есть grep-запрос его поиска, слова отмечены нужными символьными стилями. Как этим распорядиться для получения указателя?

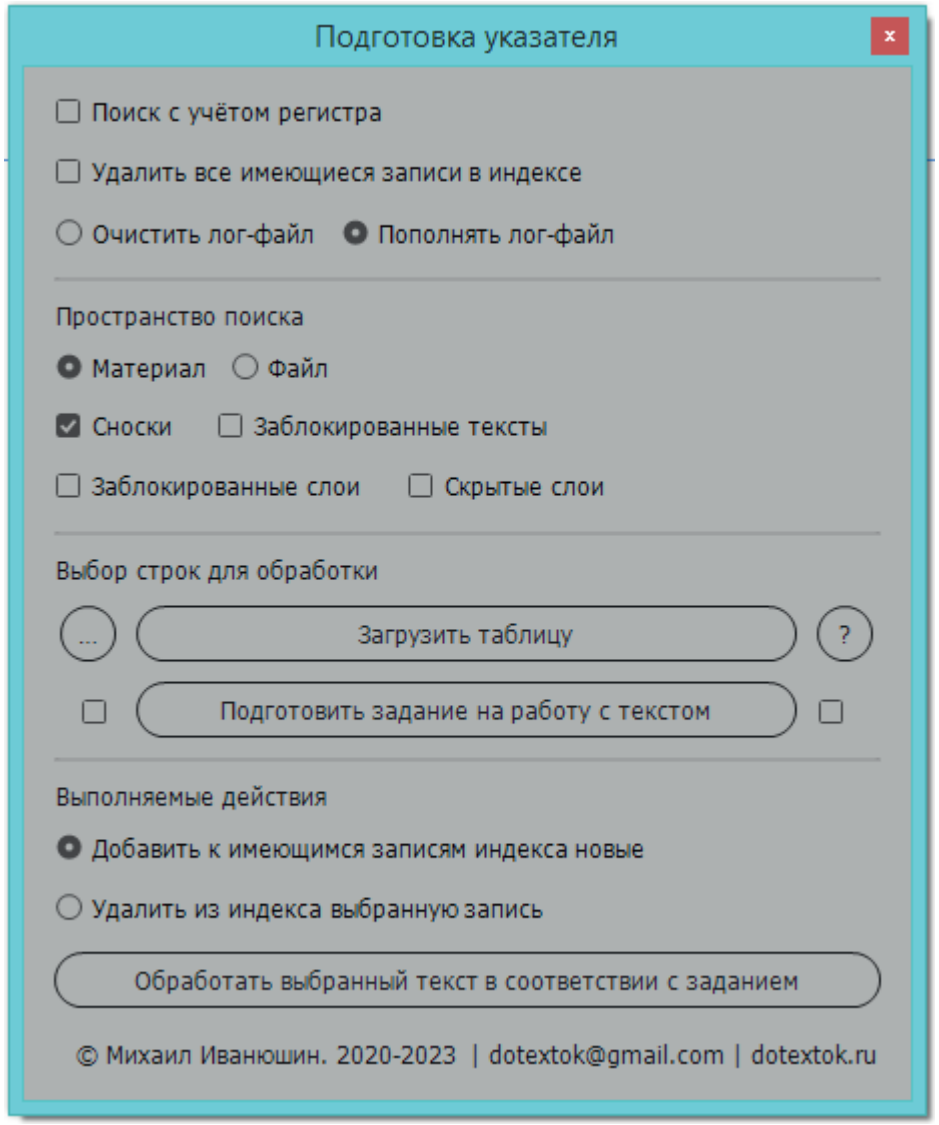
Все перечисленные ранее проблемы составления разных видов указателей на русском языке можно решить, если иметь возможность выбирать одну или несколько строк из таблицы, определять диапазон поиска — вся статья / документ или только выделенная область. Все эти возможности собраны в скрипте **ProcStoryOrDoc.jsx**, его рабочее окно справа.

Назначение кнопок понятно из их названия.

Первое, с чего начинается подготовка указателя, это загрузка таблицы (кнопка **Загрузить таблицу**) и выбор в ней нужных строк для обработки. Потом нажимается кнопка **Подготовить задание на работу с текстом.**

Для поиска иногда важно учитывать или нет регистр при поиске. Это определяет пользователь флажком **Поиск с учётом регистра**, и он должен быть установлен до нажатия кнопки **Подготовить задание на работу с текстом.** Как работает флажок **Поиск с учётом регистра**, объяснено во врезке на следующей странице. Учёт регистра можно в grep-запросе можно изменить правым верхним флажком-кнопкой в окне уточнения grep-запросов.

Надо определить пространство поиска — материал, файл и пр., — и нажать на кнопку **Обработать выбранный текст в соответствии с заданием.** В панели **Указатель (Index**, если используется английская версия индизайна) появятся найденные термины. Выбранный вариант остаётся активен, пока в таблице не будут выбраны другие строки. *Это сделано для тех, кто предпочитает выделить в таблице термин, и потом идти по книге и отмечать его только в конкретных абзацах.*



Очерёдность обработки строк

В документации к первой версии отмечено, что если есть частично совпадающие термины, например, сопротивление и удельное сопротивление, то надо начинать поиск с более длинных терминов, чтобы все их правильно обработать. И был флажок, предписывавший скрипту ProcStoryOrDoc.jsx,сперва отсортировать термины в порядке убывания длины grep-запросов. Сейчас эта сортировка стала обязательной частью алгоритма работы, и отдельный флажок не нужен.

Предварительная стилевая разметка текста при помощи скрипта MarkSameQueries.jsx обусловила необходимость учитывать в сортировке не только длину grep-запросов, но и есть ли символьный стиль для поиска по этому запросу. Вот реальная ситуация, которая потребовала проверку, есть ли при поиске символьный стиль. В индийском эпосе Рамаяна есть два персонажа с одинаковым именем Джанака. Один встречается в тексте 80 раз, второй два раза. Очевидно, что проще отметить символьным стилем второго, а первого искать grep-запросом без уточнения стиля. Но не всё так просто: если сперва будет поиск Джанаки-80, то этот поиск найдёт и все случаи Джанаки-2, и отметит их цветом. В результате при поиске Джанаки-2 ничего найдено не будет. Поэтому алгоритм сортировки по убыванию длины сделан так, что сперва идут строки, в которых есть учёт символьного стиля.

Обрабатывать только один раз

Чтобы исключить выбор уже обработанной строки, надо изменить цвет текста в третьей ячейке (с grep-запросами). Если он не чёрный, то строка пропускается. Сделайте привычкой сразу менять этот цвет, чтобы не терять время на попытки понять, что не так в указателе, повторная обработка уже обработанного текста может дать неожиданный результат.

ⓘ В программе подготовки указателя для изменения текста в ячейке с grep-запросом предусмотрен флажок-кнопка справа от кнопки **Подготовить задание на работу с текстом**. Работает, когда на экране файл с таблицей.

Поиск с учётом регистра

В приводившихся примерах словников есть слова с прописными буквами, но будут они учитываться или нет, определяет флажок **Поиск с учётом регистра** на вкладке **Работа с указателем**. Вот как он работает.

Допустим, есть текст «Белый снег и белый мел, белый заяц тоже бел».

Тогда, если флажок установлен, в строке grep-поиска это будет обозначаться оператором **(?-i)**, поиск **(?-i)Белый** найдёт только одно слово «Белый». Запрос **(?-i)белый** в той же строке найдёт два слова «белый».

Если флажок сброшен, т.е. ищем без учёта регистра, это grep-оператор **(?i)**, то любой из запросов **(?i)Белый** или **(?i)белый**, и даже **(?i)БЕЛЫЙ** найдёт и «Белый», и «белый».

Опыт настройки grep-запросов

В первой версии диапазон возможных букв в конце слова определялся так: [а-я]. И с таким диапазоном была головная боль, что ищешь **Петров[а-я]{0,2}**, а по факту результат такого grep-запроса были Петрович, Петровна. С **Павлов[а-я]{0,2}** была та же проблема: находились Павловна, Павлович. В результатах обработки запроса **Корнилов[а-я]{0,2}** нашёлся Корниловец.

Проблема появления Петровны, Павловича, Корниловца и пр. обусловлена большой информационной избыточностью набора [а-я], ну действительно, тут все буквы алфавита. В поиске решения я разделил слова на группы по вариантам окончания, чтобы у каждой группы были в наборе только те буквы, которые могут появиться. Вот какие группы получились:

Обозначение трёх последних букв слова: **s** — согласная, **g** — гласная, **m** — мягкий знак, **t** — твёрдый знак.

ssg = мечта ‖ **мечТА: \bмеч[тайоувамыихея]+\b**
ssm = честь лесть ‖ **честЬ: \bчест[ьиюяем]+\b**
gsm = тень день ‖ **дЕНЬ: \bд[еньиюяеём]+\b**
msg = кольцо ‖ **колЬЦО: \bкол[ьцоаыеуовмийх]+\b**
smg = листья ладья ‖ **ладЬЯ: \bлад[ьяиеюёйявмх]+\b**
sgg = милая ‖ **милАЯ: \bмил[аяйяогюемуивхё]+\b**
gsg = олово вода ‖ **воДА: \bво[даейямихаыуо]+\b**
smg = семья ‖ **семьЯ: \bсемь[яейиюё]+\b**
sgs = Петров ‖ **\bПетров[мыаоуе]*\b**
sms = Гарольд ‖ **\bГарольд[ауоме]*\b**
mgs = батальон ‖ **\bбатальон[ауоме]*\b**
gss = пациент торт ‖ **\bторт[аоуме]*\b**
sss = контекст текст ‖ **\bтекст[аоуме]*\b**
ggs = Малеев ‖ **\bМалеев[мыауое]*\b**
tgs = объём подъём ‖ **\bобъём[ауоме]*\b**

Группы имеют свои *наборы возможных последних букв*, они заключены в квадратные скобки, и в них букв намного меньше, чем в стандартном [а-я]. Приведённая выше информация, какие буквы входят в эти группы, не самая последняя. Эти данные периодически обновляются, и последняя версия содержится в файле настроек ForIndex.jsxinc. По ходу проверки слов меняется набор возможных последних букв для **sgs**-слов, оканчивающихся на -ел/-ёл, -ок, -ец (осёл, ребенок, отец). **ssg**-слова, оканчивающиеся на -че (отче, старче), не имеют падежных форм.

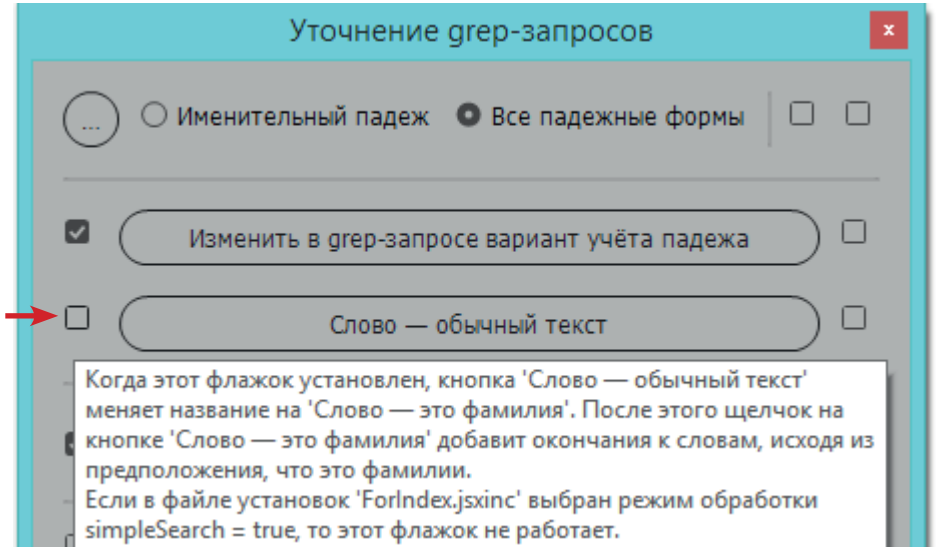
Поиск фамилий

В главе **Опыт настройки grep-запросов** рассказано о подготовке grep-запросов, учитывающих особенности русского языка, и это решение заметно помогло в поиске терминов в разных падежах.

Но оно потребовало доработки, если искать этими запросами фамилии. Надо было учесть два момента: во-первых, есть фамилии, которые не имеют падежных форм, во-вторых, есть двойные фамилии. И эти ограничения были сняты.

В файле ForIndex.jsxinc есть и grep-запросы, и код для обеих ситуаций, когда в тексте для указателя обычные слова, и когда фамилии. Но как сообщить программе уточнения grep-запросов, какой из вариантов исполь-

зовать? Для решения этого вопроса в интерфейс программы GetInfoForSearch.jsx добавлен флажок, определяющий, какой это текст:



Скрипт проверяет этот флажок и использует один из вариантов в зависимости от его установки.

В этом подходе заключена та важная гибкость, требующаяся для подготовки любого указателя: когда флажок сброшен, мы можем искать в тексте одно-два-три слова в разных падежах, ну в общем, все появления многословного термина. А если в словнике фамилия имя отчество, то в большинстве случаев результативен только поиск по фамилии, и запрос этого поиска должен быть правильным. И задача создания запроса, учитывающего все особенности фамилий на русском языке, тут решена, пожалуй, для всех случаев. Из известных мне не охвачен вариант, когда фамилия кончается на -ких, -пых, -тых (Мягких, Тупых, Толстых), мне кажется, что они тоже не склоняются независимо от того кто это — мужчина или женщина. Но если это не так, то grep-запрос для этих случаев несложно добавить в код файла ForIndex.jsxinc.

Ещё вопрос — фамилия Павел (или Орел). Мужчина по имени Павел, и фамилия такая же, как *фамилия имя* должны быть написаны в родительном падеже? **Певела Павла** или **Павла Павла**? Тему падежей фамилий, оканчивающихся на -ел, оставим филологам, а скрипт находит оба варианта.

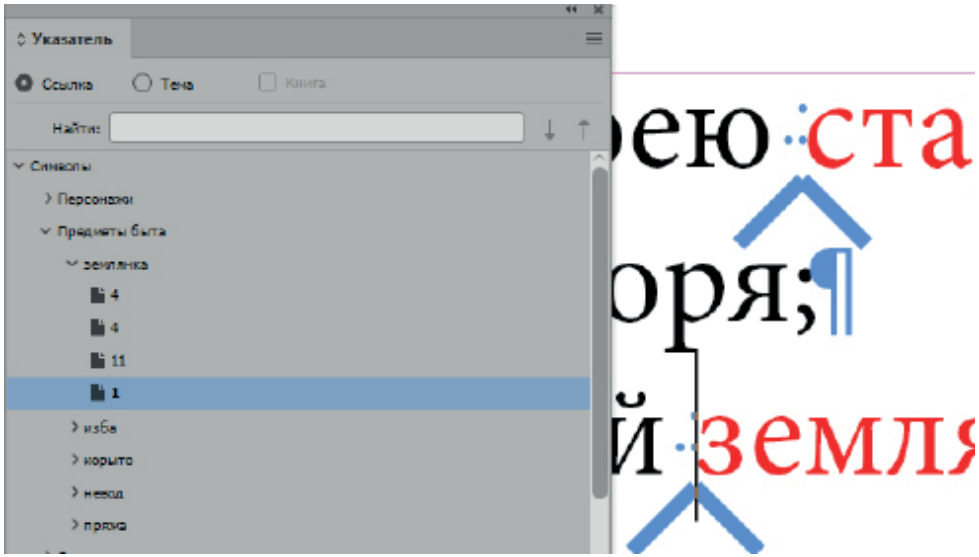
Особые случаи

Это в большей мере касается именных указателей, если там есть фамилии, оканчивающиеся на -е, -и, -о, -у, -э, -ю (Гаридзе, Иоселиани, Лещенко, Рытхэу, Друцэ, Цзю), то в большинстве случаев они не имеют падежных форм, и это надо учесть — поправить grep-запрос. Но бывают исключения, тут опять повторюсь, что редактор должен быть в теме, например, в документах первых лет после революции фамилия Родзянко почему-то нередко склонялась: Родзянке, Родзянкой.

Результаты индексирования желательно просматривать, благо что они окрашены красным цветом. Могут быть неожиданности. Например, в словнике была *Франц Мария*. Учёл, что фамилия не склоняется. В grep-поиске **\bФранц\b**, т.е. ищется граница слова: после буквы пробел, перевод строки или знак пунктуации. И после обработки текста неожиданно вижу: Франц[узский] посол [Ж.] Нуланс не уезжает — искомое слово и открывающая квадратная скобка, хотя в запросе

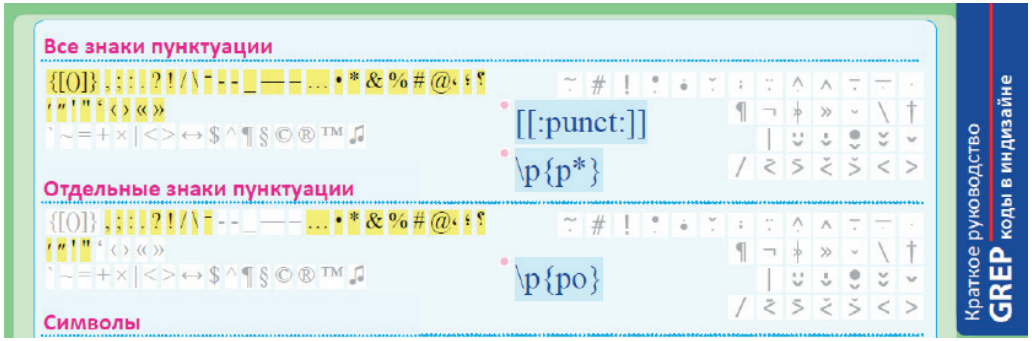
Информация о термине в тексте

Слева от слова, попавшего в индекс, всегда есть маркер. Если его выделить, то в панели **Указатель** (**Index**) откроется ветка всей иерархии указателей для этого термина, и будет отмечена страница, где этот маркер был выделен. Это очень полезная опция, особенно для случаев, когда видишь ошибку применения grep-запроса. С её помощью можно быстро разобраться, что надо исправить.



Чтобы после выделения в указателе страницы термина перейти в текст к маркеру используйте команду **Перейти к выделенному маркеру**, она в выпадающем меню панели **Указатель**.

определено, что ищется целое слово. Теоретически это не ошибка, а особенность инструмента grep, он считает скобки знаками пунктуации.



(Картинка выше — это экранная ссылка на русскоязычный документ по grep-кодам)

Но в указатель такие случаи попадать не должны.

Если в индексе номера страниц не нужны

На третьей странице есть пример двухуровневого указателя. Там **Планеты, Созвездия, Звёзды с именами** — это первый уровень, просто название, для них номера страниц искать не надо. А вот названия астрономических объектов должны иметь номера страниц. Указателем, что номера надо искать, служит буква **№** во второй ячейке строки. При создании таблицы она есть везде, и уже пользователь должен убрать её из тех строк, для которых номера страниц искать не надо.

Сбой в размещении индексного маркера

Редко, но к сожалению, бывает иногда так, что скрипт не может разместить маркер. По этой причине во второй версии оставлена функция ведения лог-файла обработки, хотя скрипт **ShowFoundByGrep.jsx** сделал её не нужной. Вот как это выглядит в лог-файле

```
>> Рычков Н. М., нарком юстиции СССР = (?i)\bРычко[вауеоым]+\b
        Рычковым [449]
        Рычковым [421]
Запись 'Рычков Н. М., нарком юстиции СССР' не удалось поместить
в индексный указатель.  ←
        Рычков [421]
Запись 'Рычков Н. М., нарком юстиции СССР' не удалось поместить
в индексный указатель.  ←
        Рычкова [420]
        Рычков [419]
        Рычкова [417]
```

Видите, дважды Запись 'Рычков Н. М., нарком юстиции СССР' не удалось поместить в индексный указатель. Эта проблема не новая, она зарегистрирована ещё в версии CS3, но похоже, эту часть кода так и не поправят.

Я обсуждал с Питером Карелом этот вопрос, он считает, что это имеет место только в файлах, где есть таблицы. И предложил решение, я его включил в программу, но оно не всегда срабатывает. Если интересно, то можете тут почитать об этой проблеме:

https://community.adobe.com/t5/indesign-discussions/index-marker-is-not-placed-in-the-right-location-when-added-via-a-script/m-p/10967488#M178124

Вот описание появления этой ошибки: а) данного сбоя нет, когда нет таблицы; б) может возникнуть сбой, когда таблица помещается в текстовый поток; в) не возникает сбоя, когда эта таблица в отдельном фрейме, не важно, стоит он на полосе сам по себе, или это прикреплённый фрейм.

Итак, я об этой ситуации знаю, и фиксирую в файле результатов. И если она возникла, вы это узнаете из лог-файла, достаточно выполнить поиск текста не удалось поместить в индексный указатель.

Что тут можно сделать? Пока только одно — добавить вручную потерянный термин, если это нужно. Когда на одной странице фамилия несколько раз, то добавлять её ещё смысла, конечно, нет. Но если термин вообще потерян, то это надо исправить.

Обновление номеров в панели указателя

Если вы откроете idml-файл с указателем, то вместо номеров страниц там будут буквы PN. Это не ошибка, просто требуется обновить номера в новом файле. Для этого в панели **Указатель** в выпадающем меню есть команда **Обновить экранную версию**. В английском варианта индизайна эта команда называется **Update Preview**.

Есть ещё ситуация, когда требуется выполнить эту команду: если добавить несколько страниц в начало файла, то после этой операции номера страниц в индексе останутся прежними. И выполнение этой команды обновит номера страниц.

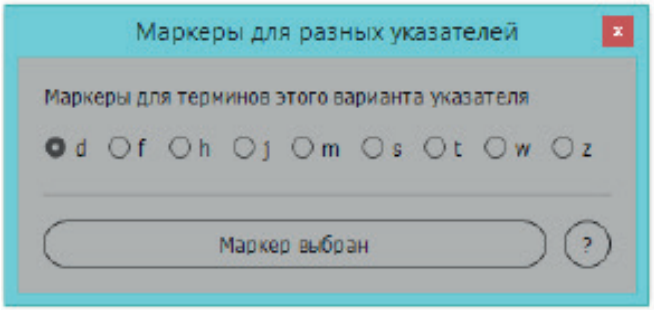
Несколько указателей в одном документе

На сайте https://helpx.adobe.com/ru/indesign/using/creating-index.html русским по-белому сказано, что «В документе или книге можно создать только один указатель.»

Но иногда их надо иметь несколько. Например, в одном издании могут требоваться именной и географический указатели.

Упомянутое ограничение индизайна можно обойти, добавляя маркеры к терминам, относящимся к разным указателям. Решено ставить перед терминами строчные латинские буквы и отделять от термина дефисом. Буквы выбраны такие, чтобы они заметно отличались от русских: d f h j m s t w z .

И если в книге предполагается иметь несколько указателей, надо сперва скриптом **UseReadyTable.jsx** подготовить таблицы обрабатываемых терминов, а потом для каждого IndexList-nnn.indd файла запустить скрипт **AddMarker.jsx** (окно в средней колонке), в котором выбрать маркеры для каждого указателя.



Когда файл IndexList-nnn.indd открыт и запущен скрипт **AddMarker.jsx**, он поместит выбранный маркер и дефис в начало каждой строки. Имя файла тоже изменится: перед номером вместо дефиса будет выбранный маркер в квадратных скобках. Если открыт файл IndexList-001.indd и выбран маркер d, то имя этого файла станет IndexList[d]001.indd.

При обработке документа программой **ProcStoryOrDoc.jsx** будет создан цвет для окрашивания найденных терминов текущего указателя. Название цвета будет начинаться с IndexColor, и в нём будут маркер и номер файла, например, IndexColor-@001.

GREP-запросы для терминов готовит **GetInfoForSearch.jsx**, он работает, когда активен файл, имя которого начинается с IndexList. Эта программа проверяет наличие маркера в названии файла, и если он есть, то при генерации grep-запроса убирает из термина первые два знака, предполагая, что это маркер и дефис.

Индизайн на основе имеющихся терминов соберёт свой единственный указатель, но поскольку термины разных указателей начинаются с неодинаковых знаков, то это обусловит, что даже в формально одном указателе термины с совпадающими маркерами будут собраны вместе.

Для удаления маркеров можно использовать grep-запрос со строкой поиска ^[dfhjmstwz]-, в поле замены ничего нет. И потом понадобится уже определённая работа по оформлению этих указателей.

После обработки файлов словников всех указателей в документе будет очень много цветовой разметки, но ей можно управлять: сделайте все IndexColor-цвета чёрными, кроме того цвета, что относится к указателю, с которым сейчас будет работа.

Одинаковые термины в разных указателях

Это может быть хлопотной проблемой. Для работы с одним указателем есть поиск совпадений терминов в соседних строках. Совпавшие термины можно отметить разными символьными стилями.

Но если случится так, что в разных указателях окажутся одинаковые термины, это может сильно затормозить работу. А это ситуация не надуманная: Вашингтон, один из президентов США и Вашингтон, столица США, город Вологда и река Вологда, и другие совпадения, которые неожиданно могут оказаться в разных предметных указателях.

И чтобы перед началом работы с несколькими указателями была уверенность, что в файлах с терминами совпадений нет, предусмотрен скрипт FindSameItems.jsx.

Он в открытых IndexList-файлах проверяет — нет ли термина из текущего файла в других файлах. Случаи совпадения терминов в текущем файле пропускаются, для этой ситуации есть другие решения, описанные ранее.

Поиск ведётся без учёта регистра. Для программы неважно, отмечены там термины уже маркерами, или пока в названии файлов между IndexList и номером стоит дефис. Сведения обо всех совпадениях собираются в текстовом файле @SameTermsInfo.txt, он в той же папке, где IndexList-файлы.

Поиск одинаковых терминов в файле. Есть одно условие, которому надо следовать, чтобы этот поиск работал верно: если в IndexList-файле есть одинаковые термины, то их надо правильно отметить. Сначала надо такие одинаковые термины найти, и для их поиска тоже есть свой инструмент — программа ColorSimilarLines.jsx, она отметит зелёным цветом все строки с одинаковыми терминами. Теперь надо после каждого такого термина поставить разделитель и порядковый номер. Знак разделителя не должен встречаться в терминах в таком виде, что в конце текста этот разделитель и число. По умолчанию в качестве разделителя используется знак подчёркивания (переменная usedSeparator в начале текста программы FindSameItems.jsx). Пробелов до и после разделителя быть не должно. Скрипт FindSameItems.jsx проверяет последние знаки терминов в поиске номеров именно в таком формате. Будут найдены варианты _1, _2, и пр., и для сравнения терминов будет текст без этих служебных номеров.

Данные номера — служебная информация, если надо изменить grep-запрос, то в нём таких номеров быть не должно.

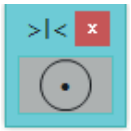
* * *

Очевидно, что скриптом ColorSimilarLines.jsx можно искать одинаковые строки при работе с одним указателем, но о ней рассказывается тут, поскольку она была создана при решении проблемы оформления нескольких указателей в одном документе.

Удаление терминов и случайных ссылок

В процессе работы может быть ситуация, что какой-то термин оказался лишним, и его надо удалить. В панели **Указатель** (**Index**) можно выбрать этот термин и отправить в корзину. Не если скрипт этот термин уже обрабатывал, то в тексте останется цветное выделение этих терминов. Поэтому в этом скрипте есть радиокнопка **Удалить из индекса выбранную запись**, это в окне скрипта ProcStoryOrDoc.jsx (см. с. 12), определяющая, что надо не только удалить выбранный термин из указателя, но и снять цветное выделение найденных терминов. За один раз можно удалить только одну запись из индекса. Именно *из индекса* — если этот термин присутствует на 30 страницах, маркер индекса перед всеми этими словами исчезнет, и они станут чёрного цвета.

Другая ситуация — отмечены слова, которых в указателе быть не должно. Например, в словнике был термин царица У́ма, и этот термин надо обработать с учётом регистра. Но этот флажок поставлен не был, и в результате оказались окрашенными слова ума, уме, умы, и в указателе они относятся к этой царице. Можно удалить их индекса эту запись, термин исчезнет в индексе, но в IndexList-файле останется, и затем заново собрать термины в документе. А можно по-другому: выделяя случайный текст и удаляя его скриптом DeleteUnnecessarySign.jsx, справа его окно — только кнопка обработки выбранного текста, и красный крестик в шапке, чтобы запомнить координаты окна и закончить работу.



Работает это так: выделяешь область с ненужными маркерами индекса, чтобы весь красный цвет попал в выборку, и надо чуть прихватить черных букв до и после красных букв. После нажатия на кнопку скрипт удалит все маркеры, попавшие в выборку и вернёт всему выбранному тексту чёрный цвет. Окно можно поместить в угол экрана, чтобы запускать его, когда надо убрать лишние маркеры.

При нажатии кнопки удаления запоминаются текущие установки окна grep-поиска, и после обработки выделенного текста они восстанавливаются. Это сделано вот для чего: может стоять задача найти все такие случаи, и тогда для поиска очередного случайного слова не надо снова вводить grep-запрос и искомый цвет. Они там будут восстановлены, останется только нажать кнопку **Найти далее**.

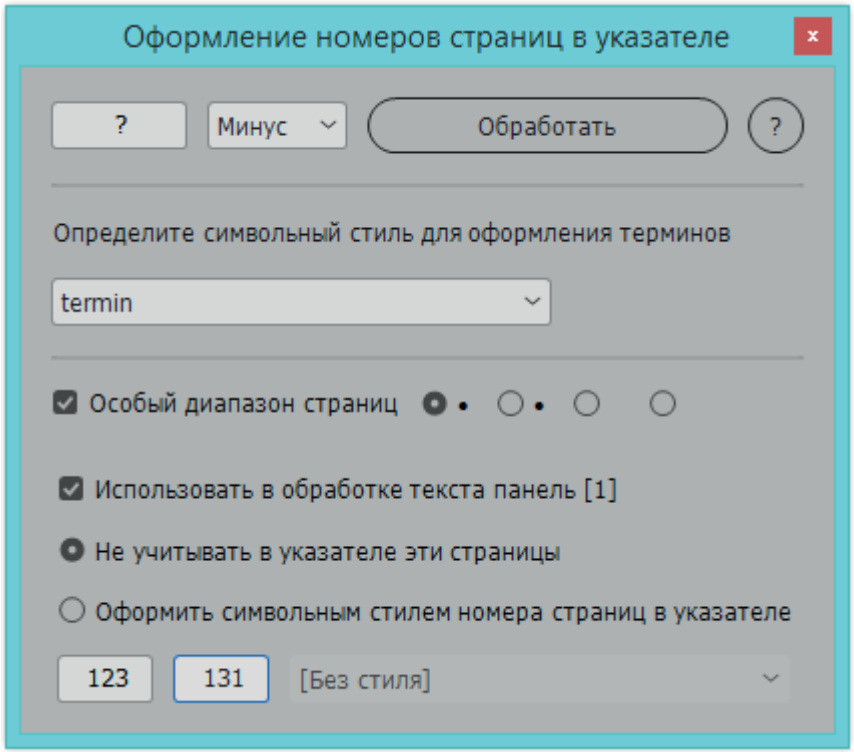
Упорядочение номеров в указателе

Особенность подготовленного индизайном указателя — это повторяющиеся одинаковые номера, и последовательности номеров идущих одна за другой страниц.

17, 17, 17, 27, 28, 29, 35, 39, 40, 41 надо преобразовать в 17, 27–29, 35, 39–41.

Это выполняет скрипт ProcNumberLines.jsx, рабочее окне показано на следующей странице.

Изменение номеров. Во время приведения строк указателя в порядок можно определить, на какое число изменить все номера страниц. Для этого надо ввести в верхнее левое текстовое поле положительное или от-



рицательное целое число. При введении отрицательного числа сперва вводится число, а потом знак. Эта опция полезна и для случая, когда указатель уже собран, оформлен, но что-то изменилось, и вся вёрстка сдвинулась на несколько страниц. Чем переоформлять указатель, проще ввести поправку на этот сдвиг.

Разделитель номеров. Выпадающий список слева от кнопки **Обработать** — выбор варианта разделителя номеров: дефис минус тире. По умолчанию используется минус.

Собрать диапазоны номеров — это безусловно полезная функция, реализуемая этой программой. И указатель будет удобнее в работе, если при его подготовке выполнить ещё несколько действий.

Стилевое оформление терминов. Поле **Определите символьный стиль для оформления терминов** предназначено для выбора символьного стиля оформления терминов предметного указателя. Сделайте привычкой сразу оформлять термины символьным стилем. Это даст больше возможностей на завершающем этапе оформления указателя.

Особые диапазоны страниц. Бывают случаи, когда в одной последовательности номеров страниц есть и номера появления термина в основном тексте, и номера страниц с этим термином в комментариях к основному тексту. При этом требуется, чтобы номера из таких разных частей книги как-то визуально различались, например, номера страниц в комментариях оформить символьным стилем, чтобы числа стали курсивными.

Можно задать четыре особых диапазона страниц. *Особость* заключается в следующем: номера страниц из этого диапазона или не учитываются в указателе (это для терминов с титула или оглавления), или их номера оформляются выбранным символьным стилем. Чтобы окно выбора стало активным, надо установить флажок **Использовать в обработке текста панель**, справа от названия этого флажка выводится номер панели. Если панель используется, то справа от радиокнопки её выбора будет чёрная точка. На картинке слева видно, что задействованы две панели из четырёх.

Обрабатываемый диапазон указывается внизу окна.

После нажатия на кнопку **Обработать** её название меняется на **Оформление**, и это название будет до тех пор, пока не исчезнет прогрессбар.

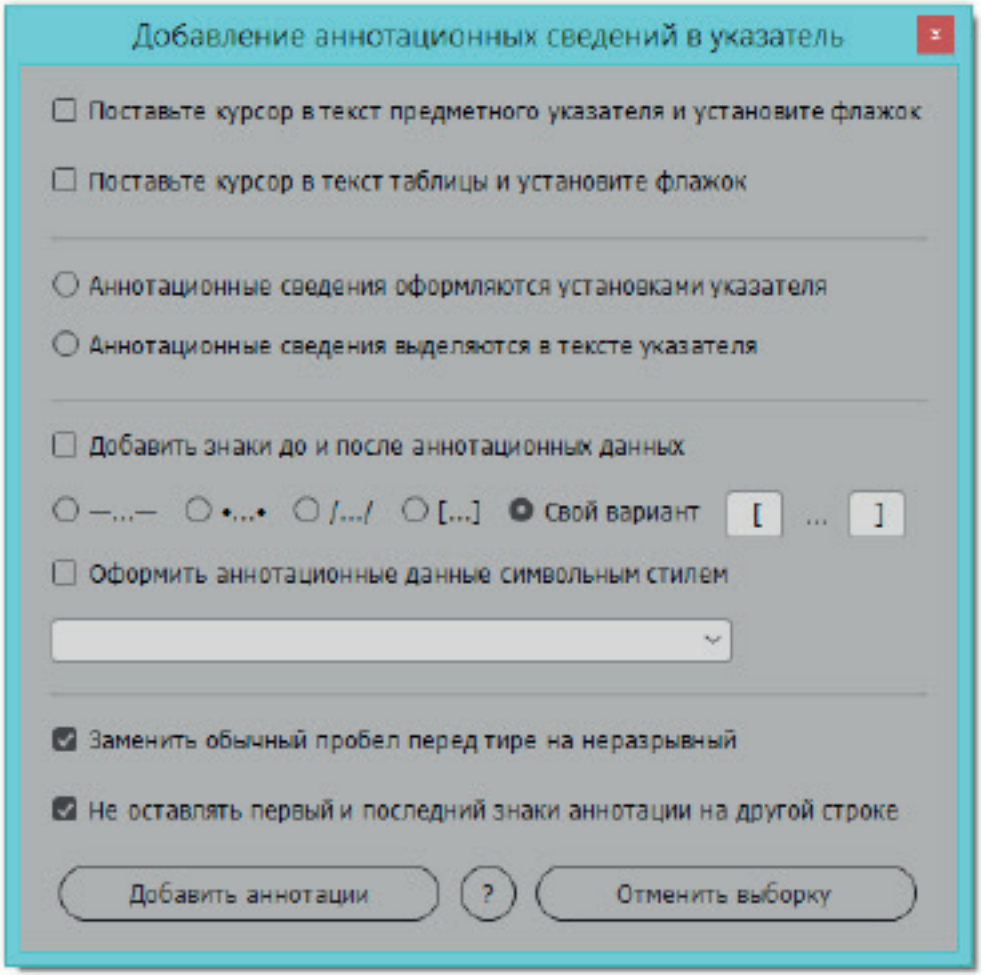
Преобразование обычного указателя в аннотированный

Вариант указателя, в котором между термином и последовательностью номеров страниц размещается краткая информация о термине, называется аннотированным указателем.

Скрипт **AddAnnotationData.jsx** позволяет добавлять в оформленный скриптом **ProcNumberLines.jsx** указатель аннотационные данные. Для его работы должны быть открыты два файла — подготовленный указатель и двухколонная таблица, в которой слева термины, а справа аннотационные данные к ним.

Вы можете оформить аннотационные сведения теми же установками, что и текст указателя, но можете сделать его более удобным для глаза пользователя. То, что я видел в этом плане, подавляющее большинство указателей было оформлено очень аскетично, тем же шрифтом и начертанием, что указатель. Очень редко было выделение курсивом терминов. Я объясняю это лишь сильной усталостью от кропотливой работы над ним. А наборщику всё равно в этом плане, как сказали, так и сделает.

Но хорошо бы как-то обособить аннотационные данные. Поставить до и после тире — это не всегда подходящее решение, поскольку в таких данных очень часто уже есть два-три таких знака. В скрипте есть предустановленные варианты, и есть поле ввода своих данных. Можно попробовать применить символьный стиль. Он, кстати, обязателен, если вы оформили курсивом или полужирным термин — без указания тут символьного



стиля аннотационные данные будут того же начертания, что есть у термина.

И даже если ничего не нравится из предложенного, то полукруглые шпации до и после аннотационных данных чуть-чуть выделят их общего потока текста. В общем — инструмент есть, а как вы им распорядитесь, зависит от вас и редактора, только надо, чтобы он знал о таких возможностях.

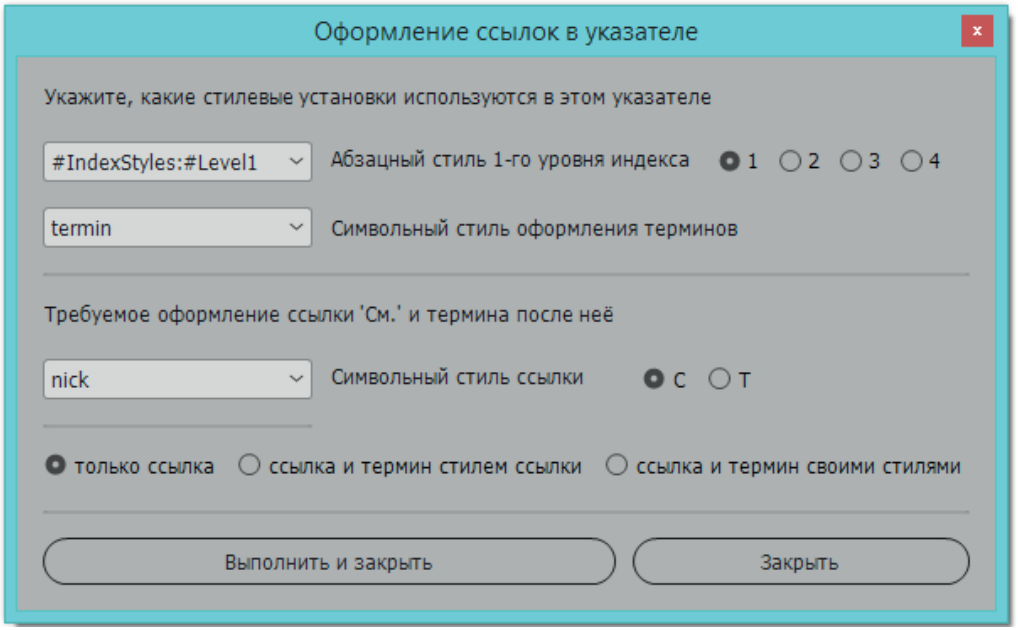
Ссылка 'См.' для терминов нижних уровней

В опциях индизайна для оформления указателя есть разные варианты ссылок на другие термины: 'См.', 'См. также', 'См. здесь', 'См. также здесь', 'См. [также]'. Это переведённый на русский язык американский опыт работы с указателями, возможно, там есть обоснованные причины иметь и 'См. также', и 'См. [также]'.

У нас действует **ГОСТ 7.78-99 Издания. Вспомогательные указатели.** Там подобная адресация определена в п. 3.1.6:

3.1.6. **ссылка «см.», и ссылки «см. также», «сравни» и т.п.**: Записи в указателе, отсылающие от одного заголовка (или подзаголовка) рубрики к другому (альтернативному или добавочному.

Сделана программа AddSeeTopic.jsx, справа её рабочее окно, она расширяет возможности подготовки предметных указателей на русском языке, тут автоматизируется работа только с вариантом ссылки 'См.', и для неё перекрёстная ссылка не формируется, есть только текстовый вид. Но по тем возможностям, что в ней есть, она превосходит штатный инструмент.



В указателе могут быть ссылки 'См.' на иное название какого-то термина, это всегда записи первого уровня. Например, ссылка в книге «Общая химия»

Адденды см. Лиганды

но то, о чём говорится в данной главе — это работа с терминами второго и следующих уровней: их названия надо поднять на верхний уровень, чтобы читателю можно было легко их найти.

Какой будет строка со ссылкой 'См.'

Допустим, у нас есть двухуровневый указатель:

Вяжущие вещества 1
Воздушные вяжущие 1, 2
Гидравлические вяжущие 1, 2
Минеральные вяжущие 1
Портландцемент 1, 2, 4-8
Клинкер 4- 8, 10, 11

и требуется в указателе для терминов второго уровня в алфавитном перечне *терминов первого уровня* иметь

такие записи: Воздушные вяжущие *См.* Вяжущие вещества, Клинкер *См.* Портландцемент и пр.

Просмотр в сети сканов страниц предметных указателей навёл на мысль, что не всех, кто берётся за создание указателей, устраивает пришедшее из горячего набора невыразительное оформление указателей. Есть и изменение стиля ссылки 'См.', видел и стилевое выделение терминов, на которые указывает ссылка.

С помощью данной программы это легко реализуется. В верхнем блоке окна надо указать стили, используемые в обрабатываемом указателе. Абзацные стили для первого и второго уровней должны быть обязательно. Если в указателе третьего и четвёртого уровней нет, то надо выбрать [Основной абзац], для стилей уровней указателя этот служебный стиль использоваться не должен.

Символьный стиль оформления терминов должен быть обязательно, он определяет, какая часть строки будет в строке со словом-ссылкой. На с. 23 термины окрашены синим цветом — это приложен символьный стиль. Если там будет служебный стиль (в названии есть квадратная скобка), программа работать не будет.

Радиокнопки **С** и **Т** — для выбора символьных стилей оформления ссылки и термина, на который ссылка. Ниже три радиокнопки выбора оформления.

После нажатия кнопки **Выполнить и закрыть** программа найдёт все термины второго и следующих уровней, определит, в гнезде какого верхнего уровня они находятся. Затем будут сделаны строки со ссылками. Эти строки помещаются в алфавитном порядке в предметный указатель, вся строка оформляется абзацным стилем первого уровня, и в конце приводится в порядок слово ссылки.

Приведённый ранее пример после обработки будет выглядеть примерно так:

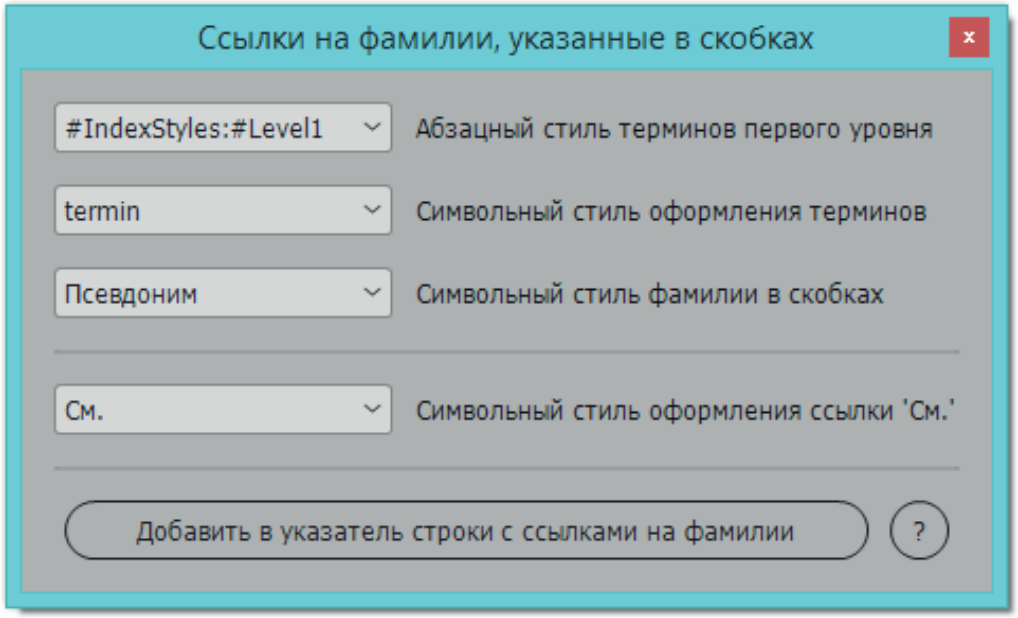
Воздушные вяжущие *См.* Вяжущие вещества
Вяжущие вещества 1
Воздушные вяжущие 1, 2
Гидравлические вяжущие 1, 2
Минеральные вяжущие 1
Гидравлические вяжущие *См.* Вяжущие вещества
Клинкер *См.* Портландцемент
Минеральные вяжущие *См.* Вяжущие вещества
Портландцемент 1, 2, 4-8
Клинкер 4- 8, 10, 11

Фамилии в скобках

Другая отнимающая много времена задача — извлечь взятые в скобки фамилии и оформить ссылки на них. Эта задача легко решается скриптом **NameAndNick.jsx**.

Ищутся строки с фамилиями в круглых скобках. Создаётся новая строка в указателе: найденная фамилия со ссылкой *См.* на тот термин, в котором она обнаружена. Эта строка помещается на первый уровень указателя, не нарушая алфавитный порядок следования терминов.

В большинстве случаев можно обойтись и без уточнения символьного стиля фамилий, выполняя поиск текста в скобках, а символьный стиль для фамилии



указать тот, что у терминов указателя. Но если в указателе есть термины со скобками, то для идентификации фамилии символьный стиль надо указывать обязательно, и скобки тоже должны быть оформлены этим символьным стилем.

Поиск выполняется только в строках первого уровня индекса. В итоге, если в указателе была строка

Ленин (Ульянов) Владимир Ильич 12, 35-39, 54

то после завершения работы скрипта число строк в указателе увеличится: среди терминов, начинающихся с буквы У, появится строка со ссылкой на эту строку:

Ульянов *См.* Ленин Владимир Ильич

и ссылка *См.* будет оформлена выбранным символьным стилем.

Оформление длинных строк, в которых первые слова совпадают

Вот скан фрагмента указателя книги К. К. Пономарева «Специальный курс высшей математики», 1974, страница 372: прочерки на месте слов, которые есть в предыдущей строке указателя.

Метод вариации произвольных постоянных для линейных неоднородных дифференциальных уравнений с переменными коэффициентами 45, 46, 152—155
— — — — — — — — — — постоянными коэффициентами 136, 137, 142—144
— — — — — — — — систем дифференциальных уравнений 195, 196
— вырожденных ядер 345—347
— интегрируемых комбинаций 186, 187
— исключения 181
— конечных разностей 298—301
— матричной факторизации 316—322, 324—327
— неопределенных коэффициентов 129, 158
— последовательного интегрирования 177
— последовательных приближений для уравнения Вольтерра 342, 343
— — — — — Фредгольма 336—339
— разностной факторизации 312—314

Утверждение «которые есть в предыдущей строке указателя» верно до того момента, пока редактор, глубоко вздохнув, не стал как-то отмечать слова, которые есть в предыдущей строке, чтобы наборщик заменил их на знак тире. На предыдущей странице вместо совпавших слов уже прочерки.

Такой подход к оформлению указателей можно найти в книгах XIX века, можно сказать, что это полиграфическая традиция, помогающая делать указатели более компактными. Традиции, безусловно, надо беречь, но уже, похоже, нет желающих тратить время на такое оформление.

И всё-таки *давайте сохраним эту традицию и при компьютерном наборе*. Для проверки соседних строк и замены во второй сроке совпавших слов на знак тире есть программа **DashInsteadWord.jsx**, она берёт на себя просмотр всех выделенных строк и замену совпадающих слов на прочерки: первая строка выделения становится эталоном, содержимое следующей строки запоминается, а сама она слово за словом сравнивается со словами в эталонной строке. При совпадении слова заменяются на прочерки. После завершения сравнения запомненная строка становится эталоном, и дальше работа со следующей строкой.

Кропотливый труд редакторов делает программа, текст обрабатывается в соответствии с такими условиями: а) однобуквенные предлоги тоже считаются словами; б) составные слова считаются одним словом; в) при переходе на другой уровень индексного указателя первая строка, оформленная иным абзацным стилем, становится эталоном для начала сравнения.

По умолчанию прочерк — это знак тире, но можно заменить его на минус или дефис: это определяется в переменной dash в начале кода программы.

Если совпадение только в первом слове, то прочерки всё равно нужны, чтобы отличать двухуровневые указатели от одноуровневых. В двухуровневом указателе иерархия обозначается увеличенным левым отступом. Сочетание Ctrl+Z вернёт выделенный текст к состоянию до запуска этой программы.

Сортировка терминов в указателе

Казалось бы, что разумно указывать все термины с прописной буквы. Но просмотр книг показывает, что по каким-то причинам они иногда начинаются со строчной буквы.

И тут возникает вопрос: как их сортировать в указателе. По умолчанию стандартная сортировка — это сперва все слова в алфавитном порядке, начинающиеся с прописных букв, потом слова в алфавитном порядке, начинающиеся со строчных букв.

Но в индизайне слова в указателе сортируются в предположении, что все буквы строчные. Поэтому в скриптах **AddSeeTopic.jsx** и **NameAndNick.jsx** включена такая же сортировка. Слова, начинающиеся с букв с акцентами (ä, ś, ī и др.), помещаются в конец списка.

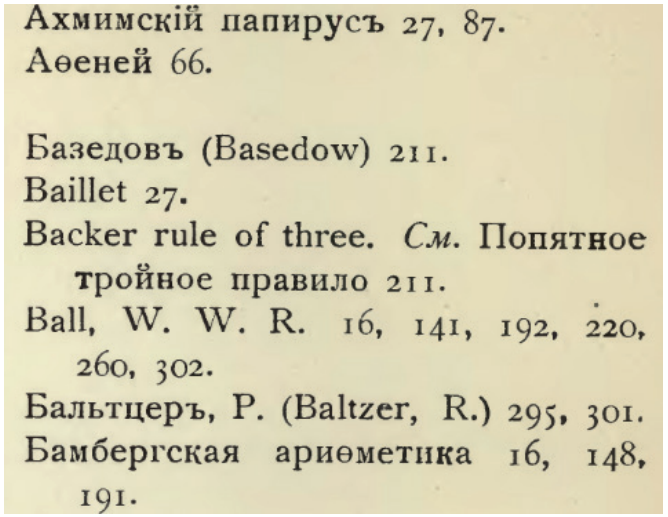
Преобразование указателя в алфавитный

В указателе термины отсортированы по алфавиту. Если выполняется скрипты AddSeeTopic.jsx или NameAndNick.jsx, то добавленные строки помещаются тоже с сохранением алфавитной очерёдности. И после того, как всё сделано, этот упорядоченный по алфавиту перечень абзацев надо превратить в полноценный алфавитный указатель. И тут надо сказать, что есть различие в подходе к оформлению первой буквы начала нового блока терминов.

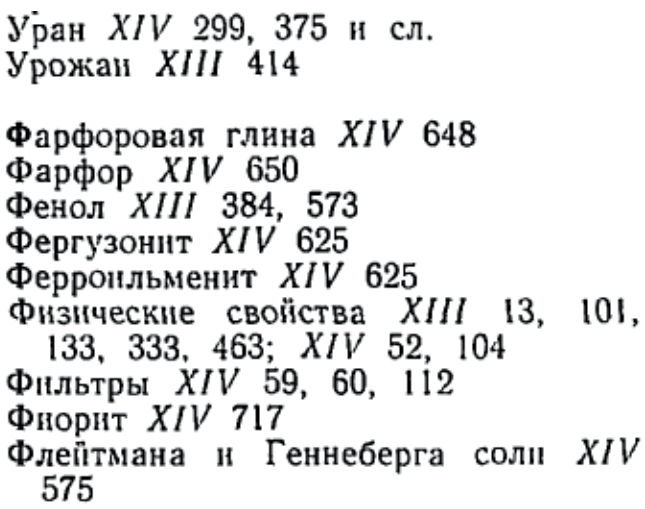
Скан из книги Ф. Кэджори «История элементарной математики, 1910 — внизу слева. Вверху страница XIV тома сочинений Д. И. Менделеева, 1949. На обеих картинках видно, что блоки разных букв отделены пустой строкой, кегль и начертание буквы не изменены. Есть примеры указателей, сделанных ручным набором, где используется литера того же кегля, делающая на бумаге полужирную букву.

Внизу справа скан фрагментов указателя из собрания сочинений Г. В. Плеханова, 1956 год. Тут кроме отбивки заметно увеличен кегль первой буквы первого слова в блоке фамилий, начинающихся на одну и ту же букву.

И есть вообще отказ от такого оформления — выделения буквы.


Ахмимскій папирусъ 27, 87.
Аѳеней 66.

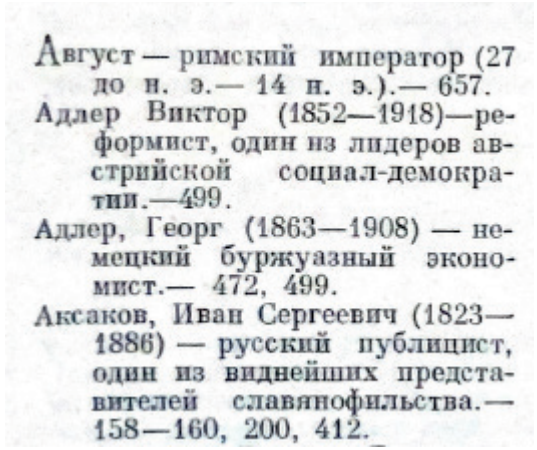
Базедовъ (Basedow) 211.
Baillet 27.
Backer rule of three. *См.* Попятное тройное правило 211.
Ball, W. W. R. 16, 141, 192, 220, 260, 302.
Бальтцеръ, P. (Baltzer, R.) 295, 301.
Бамбергская ариѳметика 16, 148, 191.

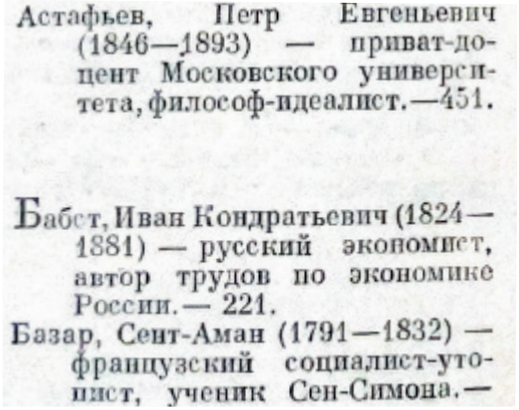

Уран *XIV* 299, 375 и сл.
Урожаи *XIII* 414

Фарфоровая глина *XIV* 648
Фарфор *XIV* 650
Фенол *XIII* 384, 573
Фергузонит *XIV* 625
Ферроильменит *XIV* 625
Физические свойства *XIII* 13, 101, 133, 333, 463; *XIV* 52, 104
Фильтры *XIV* 59, 60, 112
Фиорит *XIV* 717
Флейтмана и Геннеберга соли *XIV* 575

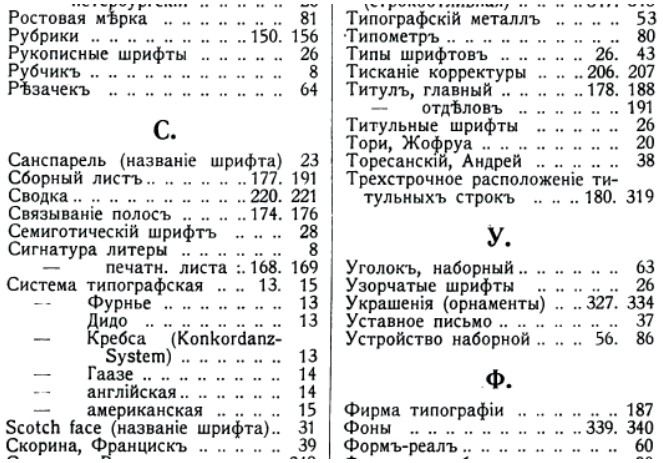
— марганец *XIII* 748
— мышьяк *XIV* 607
— фосфор *XIV* 591
— хром *XIV* 345
Фтороборная кислота *XIV* 644
Фумарола *XIV* 634

Халцедон *XIV* 713, 716
Хлопчатобумажный порох = пироксилин
Хлор *XIII* 653, 693
Хлорангидрид азотистой кислоты *XIII* 706
— борной кислоты *XIV* 647


Август — римский император (27 до н. э.— 14 н. э.).— 657.
Адлер Виктор (1852—1918)—реформист, один из лидеров австрийской социал-демократии.—499.
Адлер, Георг (1863—1908) — немецкий буржуазный экономист.— 472, 499.
Аксаков, Иван Сергеевич (1823—1886) — русский публицист, один из виднейших представителей славянофильства.— 158—160, 200, 412.


Астафьев, Петр Евгеньевич (1846—1893) — приват-доцент Московского университета, философ-идеалист.—451.

Бабст, Иван Кондратьевич (1824—1881) — русский экономист, автор трудов по экономике России.— 221.
Базар, Сент-Аман (1791—1832) — французский социалист-утопист, ученик Сен-Симона.—



Ростовая мѣрка .. 81
Рубрики .. 150. 156
Рукописные шрифты .. 26
Рубчикъ .. 8
Рѣзачекъ .. 64

С.

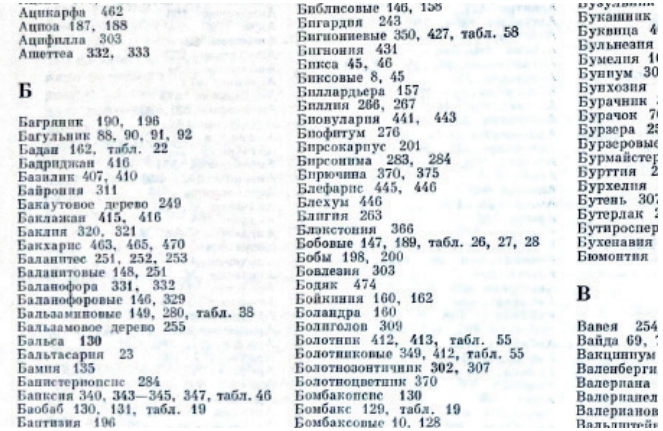
Санспарель (названіе шрифта) 23
Сборный листъ .. 177. 191
Сводка .. 220. 221
Связываніе полосъ .. 174. 176
Семиготическій шрифтъ .. 28
Сигнатура литеры .. 8
— печатн. листа .. 168. 169
Система типографская .. 13. 15
— Фурнье .. 13
— Дидо .. 13
— Кребса (Konkordanz-System) .. 13
— Гаазе .. 14
— англійская .. 14
— американская .. 15
Scotch face (названіе шрифта) .. 31
Скорина, Францискъ .. 39
Типографскій металлъ .. 53
Типометръ .. 80
Типы шрифтовъ .. 26. 43
Тисканіе корректуры .. 206. 207
Титулъ, главный .. 178. 188
— отдѣловъ .. 191
Титульные шрифты .. 26
Тори, Жофруа .. 20
Торесанскій, Андрей .. 38
Трехстрочное расположеніе титульныхъ строкъ .. 180. 319

У.

Уголокъ, наборный .. 63
Узорчатые шрифты .. 26
Украшенія (орнаменты) .. 327. 334
Уставное письмо .. 37
Устройство наборной .. 56. 86

Ф.

Фирма типографіи .. 187
Фоны .. 339. 340
Формъ-реалъ .. 60



Ацинарфа 462
Ацноа 187, 188
Ацифилла 303
Ашеттеа 332, 333

Б

Багрянник 190, 196
Багульник 88, 90, 91, 92
Бадан 162, табл. 22
Бадриджан 416
Базилик 407, 410
Байронна 311
Бакаутовое дерево 249
Баклажан 415, 416
Баклия 320, 321
Бакхарис 463, 465, 470
Баланитес 251, 252, 253
Баланитовые 148, 251
Баланофора 331, 332
Баланофоровые 146, 329
Бальзаминовые 149, 280, табл. 38
Бальзамовое дерево 255
Бальса 130
Бальтасария 23
Бамия 135
Банистериопсис 284
Банксия 340, 343—345, 347, табл. 46
Баобаб 130, 131, табл. 19
Бантизия 196
Библисовые 146, 158
Бигардия 243
Бигнониевые 350, 427, табл. 58
Бигнония 431
Бикса 45, 46
Биксовые 8, 45
Биллардьера 157
Биллия 266, 267
Биовулария 441, 443
Биофитум 276
Бирсокарпус 201
Бирсонима 283, 284
Бирючина 370, 375
Блефарис 445, 446
Блехум 446
Блигия 263
Блэкстония 366
Бобовые 147, 189, табл. 26, 27, 28
Бобы 198, 200
Бовлеания 303
Бодяк 474
Бойкиния 160, 162
Боландра 160
Болиголов 309
Болотник 412, 413, табл. 55
Болотниковые 349, 412, табл. 55
Болотноазотничник 302, 307
Болотноцветник 370
Бомбаконсис 130
Бомбакс 129, табл. 19
Бомбаксовые 10, 128
Букашник
Буквица
Бульнезия
Бумелия
Буннум
Бухозия
Бурачник
Бурачок
Бурзера
Бурзеровые
Бурмайстер
Буртгия
Бурхеллия
Бутень
Бутердак
Бутироспер
Бухенавия
Бюмонтия

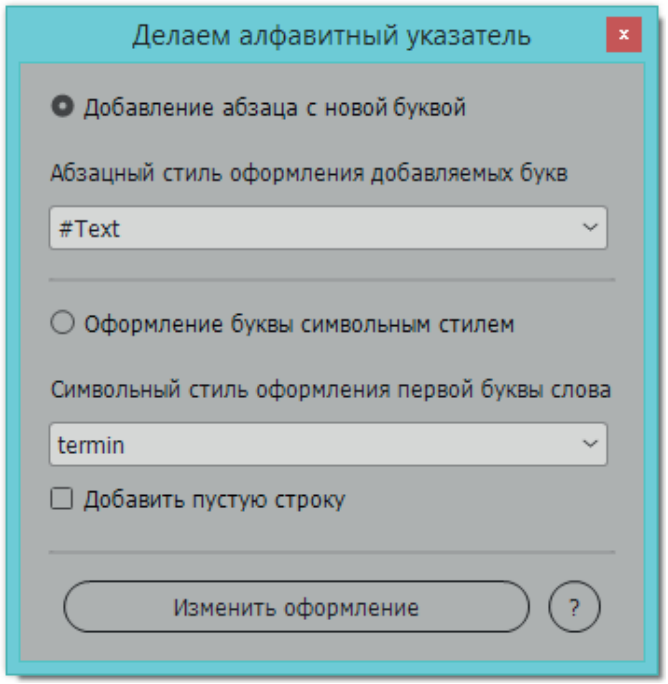
В

Вавея 254
Вайда 69,
Вакцинум
Валенберги
Валериана
Валерианел
Валерианов
Вальцштейн

Вместо этого в отдельной строке над каждым блоком терминов, начинающихся с одной и той же буквы, ставится та буква, с которой эти слова начинаются.

Вот выше два примера. Первый из книги Фридриха Бауэра «Руководство для наборщиков», 1911 год, второй — из энциклопедии «Жизнь растений», 1974 год.

Итак, два варианта оформления алфавитного указателя, и скрипт **AddLetter.jsx**, позволяет использовать любой из них. Он работает для выбранной области строк.



Такое решение — для выбранной области, а не для всего указателя — обусловлено тем, что в работе может быть несколько разных указателей, поэтому важно определять область действия скрипта.