ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

Институт межкультурной коммуникации и международных отношений

Кафедра английской филологии и межкультурной коммуникации

Разработка виртуального словаря специальной лексики с использованием баз данных

Выпускная квалификационная работа
обучающегося по направлению подготовки
45.03.04 Интеллектуальные системы в гуманитарной сфере
очной формы обучения,
группы 04001320
Бойко Анатолия Игоревича

Научный руководитель
профессор, доктор
филологических наук
Куприева И.А.

БЕЛГОРОД 2017

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 3
Глава 1. Изучение теоретических аспектов, связанных с решением задачи 6
1.1 Изучение теоретического базиса в области лингвистики 6
1.2 Диаграммы потока данных и модели проектирования программных продуктов 13
1.3 Язык программирования C++ и основы объектно-ориентированного программирования 20
1.4 Сравнение сред разработки программных продуктов для языка C++ 25
Вывод к Главе 1 28
Глава 2. Разработка и создание виртуального словаря на основе базы данных .. 30
2.1 Разработка алгоритма, создание диаграмм потоков данных и пользовательского интерфейса 30
2.2 Описание процесса отбора лексем для создания физической базы данных 38
2.3 Описание процесса установки связей между приложением и физической базой данных 40
2.4 Описание отдельных блоков программного кода 50
2.5 Апробация 55
Вывод к Главе 2 61
ЗАКЛЮЧЕНИЕ 62
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 63

ВВЕДЕНИЕ

В настоящее время наблюдается тенденция увеличения использования компьютерной обработки лексикографических данных вместо традиционных методов. Использование компьютера позволяет не только экономить ресурсы и время, но и благоприятно сказывается на качестве получаемой информации, а также отсеивать большую часть ошибок.

Процесс компьютеризации лексикографической деятельности заключается в первую очередь в создании баз данных, в которых материал сгруппирован в соответствии с определенными требованиями. Помимо этого, данный процесс входят такие составляющие как разработка программных средств, позволяющих обрабатывать полученные данные, а также отвечающих за визуальное представление информации в группированном виде.

Благодаря использованию компьютера значительно упрощается процесс поиска требуемой лексической единицы в базе данных, помимо этого, создание новых программных продуктов, реализующих поиск, отбор и отображение данных из баз данных, вместо использования уже готовых решений, позволяет увеличить качество отбора информации посредством узкой направленности каждого разрабатываемого модуля.

Таким образом, актуальность данной работы заключается в необходимости создания программного продукта, который позволит обрабатывать лексемы, собранные в базе данных и будет отвечать за представление этой информации для пользователя в наиболее удобном формате и в соответствии с заданными пользователем условиями отбора лексем.

Основной целью представленного исследования является освещение процесса разработки программных продуктов, отбора лексических единиц, а также непосредственно создание программного продукта, который, работая с

занесенными в базу данных лексемами, позволит пользователю в как можно более короткие сроки получить как можно более полную информации по каждой из лексем, с учетом его предпочтений, задаваемых при помощи параметров фильтрации. Таким образом будет создан виртуальный словарь на основе базы данных. Научная и практическая значимость данной работы заключается в разработке программного продукта для работы с базами данных, содержащими определенные лексемы.

Объектом исследования данной работы будут являться как сами лексемы, номинирующие процесс познания, так и методологии, устанавливающие требования к проектированию и разработке программных продуктов.

Предметом настоящего исследования стали средства проектирования и разработки программного продукта, взаимодействующего с базами данных, с целью фильтрации и вывода имеющейся в ней информации на основе запроса пользователя.

Данная работа включает традиционные и современные когнитивные методы исследования. Были использованы методы анализа литературы, сравнения, моделирования, абстрагирования, а также методы обобщения и синтеза.

Для достижения описанных выше целей необходимо решить следующие задачи:

- изучить теоретический материал по темам «Слово как единица языка», «Этапы создания программного продукта», «Язык программирования C++», «Среды разработки C++»;
- выполнить разработку программного обеспечения, включающую в себя разработку моделей потоков данных, алгоритма и интерфейса;
- создать приложение для работы с базой данных, включая создание пользовательского интерфейса;
- выполнить подключение предоставленной базы данных;

- провести апробацию полученного программного продукта и исправить выявленные недостатки;
- создать установочный файл программного продукта.

Выполнение вышеперечисленных задач приведет к созданию готового к использованию программного продукта, который позволит пользователю получать необходимую ему информацию и задавать параметры фильтрации по различным категориям. Что в свою очередь облегчит процесс обработки данных по лексическим единицам номинирующих процесс ментальной деятельности, позволяя тем самым ускорить проведение исследований в данной области.

Структура данной работы включает введение, две главы, заключение и список использованных источников.

Материалом исследования послужили языковые данные полученные из Интернет-ресурсов, толковых словарей и художественных произведений, а также научные работы, посвященные методологиям проектирования и разработки программных продуктов.

Глава 1. Изучение основ лексикологии и разработки и проектирования программных продуктов

1.1 Изучение ключевых понятий лексикологии связанных со словом и его значением

В рамках данной работы необходимо осветить ключевые понятия лексикологии, поскольку создание виртуального словаря невозможно без изучения основной терминологии предметной области.

Слово изучается в различных разделах языкознания, поскольку имеет различные характеристики и объединяет собой разные аспекты языка. С точки зрения лексикологии слово считается прежде всего основной структурно-семантической самостоятельной языковой единицей обладающей значением и служащей для именования предметов и их свойств, явлений, отношений к действительности. Слово обладает совокупностью семантических, фонетических и грамматических признаков, особых для каждого языка. Именно благодаря наличию и разнообразию признаков возможна группировка и изучение слов, грамматики и языка в целом.

Слово как единица языка имеет свои особенности, позволяющее отличать его от морфем.

Грамматическая оформленность слова заключается в том, что оно, в отличие от морфемы, является определенной частью речи с соответствующими ей грамматическими признаками (Стернин, 2015).

Фонетическая оформленность слова заключается в том, что оно – такой звуковой комплекс, внутрь которого не может быть вставлено другое слово, что отличает слово от словосочетания. Помимо этого, недопустима произвольная

смена расположения частей слова. Кроме этих свойств, слово может быть ограниченно паузами любой длины и иметь одно ударение (Стернин, 2015).

Фразеологичность значения слова заключается в том, что его значение не является суммой значения составляющих его морфем. Для производных слов характерно, что значение слова заключено не только в значении составляющих его морфем, но и в компонентах смысла не имеющих формального выражения. Для непроизводных слов под фразеологичностью значения может пониматься независимость значения слова от его внешней формы (Стернин, 2015).

Говоря о значении слова необходимо упомянуть существование двух видов значения, грамматического и лексического. В широком понимании термина «значение слова» имеется в виду его внутренне содержание, состоящее как из лексического, так и из грамматического значений. В свою очередь в узком смысле термин «значение слова» тождественен термину «лексическое значение», и является предметом изучения лексикологии, в то время как грамматическое значение рассматривается отдельно и является предметом изучения морфологии.

Грамматическое значение слова – отвлеченное и обобщенное языковое значение, которое характеризует слово как элемент определенного класса, части речи и выступает как добавочное к лексическому значению слова (Гируцкий, 2016). Такое значение обязательно для всех слов данного класса и имеет регулярное выражение. К обязательным грамматическим формам можно отнести, например, значения рода, числа, падежа, которые выражаются за счет соответствующих окончаний. Грамматическое значение слова выражает различные отношения, такие как отношения к другим словам в предложении, отношение к лицу, совершающему действие и так далее.

Лексическое значение слова – является соотнесённостью звуковой оболочки с соответствующими предметами или явлениями объективной действительности (Гируцкий, 2016). Лексическое значение включает в себя не всю совокупность признаков присущих какому-либо предмету, явлению или

действию, а только наиболее существенные из них, которые помогают отличить один предмет от другого. Лексическое значение слова раскрывает признаки, по которым определяются общие свойства для ряда предметов, явлений, действий, а также устанавливает различия выделяющие конкретный предмет, явление или действие. Кроме этого лексическое значение слова также можно назвать содержанием слова, которое отображает и закрепляет в сознании представление человека о предмете, свойстве, процессе или явлении. Слово может иметь одно или несколько значений. В зависимости от этого оно будет являться или однозначным, или многозначным соответственно.

Также значение слова может быть рассмотрено с различных аспектов:

- семиологический аспект;
- структурно-семантический аспект;
- функционально-стилевой аспект.

При семиологическом рассмотрении значение слова является отражение объективного мира сознанием человека в слове. В данном случае мы рассматриваем слово как знак – заместитель явления реальности в мыслительно речевой деятельности. Этот знак в свою очередь обладает планом содержания, иначе называемым значением, и планом выражения, иначе называемым формой. Связь между этими планами не естественна, а условна. Все знаки представляют собой определенную знаковую систему, в случае если значения этих знаков согласованны между собой. Функционирование слова как знака связанно с его тремя характеристиками: семантикой, синтактикой и прагматикой (Гируцкий, 2016).

Структурно-семантический аспект, в свою очередь, характеризует саму структуру значения слова, его внутреннюю семантическую производность и разного рода семантические включения, и смысловые возможности слова. Для структурно-семантического аспекта характерно наличие нескольких значений слова:

- обыденное понятие;
- научное понятие;
- коннотация;
- потенциальное значение;
- вероятностное значение;
- энциклопедическое значение.

Обыденным понятием называют общепринятую в языке форму знания, которая сложилась в процессе практической и культурной деятельности человека, и которая раскрывает его функциональные потребности.

Под научным понятием в свою очередь понимают форму знания характерную для определенной отрасли науки, и закрепившую в себе наиболее общие и существенные признаки специализированного осмысления данного слова.

Коннотация, или дополнительное значение, заключает в себе содержательные и стилистические компоненты значения слова, которые устойчиво связаны с его основным значением в сознании носителей данного языка.

Потенциальное значение представляет собой содержащиеся в слове смысловые возможности, известные носителям данного языка и проявляющиеся при контекстном употреблении в сочетательных связях.

Под вероятностным значением в свою очередь понимают прагматические ассоциативные смыслы, которые возникают из знания о мире и служат для означивания слова в данном контексте в синтагматических связях.

К энциклопедическому значению относят как обширную характеристику научного понятия, так и функциональное назначение объекта, и элементы его бытового описания.

Рассматривая значение слова с точки зрения его функционально-стилевого аспекта, анализируются социальные задачи коммуникации. Связи языка с

общественными сферами обуславливают его социальную стратегию и функционально-стилевое назначение.

В языках выделяют в основном три стиля: нейтральный, книжный и разговорный. Каждый из данных функциональных стилей обладает характерными фонетическими и лексико-фразеологическими особенностями, специфическими морфологическими формами и синтаксическими конструкциями.

К нейтральной лексике относят общеупотребительную лексику, которая именует обыденные понятия из повседневной жизни общества и лишена терминологии. Также нейтральная лексика не имеет экспрессивных, эмоционально-оценочных коннотаций. Данный пласт лексики представляет собой основу для функционально-стилевой дифференциации слов, поскольку принадлежность слова к тому или иному стилю определяется за счет сравнения его с нейтральным.

Для книжного стиля характерна специализация информации. Данный стиль включает в себя три подстиля, которые характеризуются специфичным составом лексики:

- научный стиль;
- официально-деловой стиль;
- публицистический.

Научный стиль, как следует из названия, преимущественно используется в сфере науки и характеризуется обезличенной информацией о природе, человеке и обществе. Отличительными чертами научной лексики являются:

- терминированность семантики слова;
- систематика научных понятий;
- отсутствие эмоционально-экспрессивных коннотаций;
- логизация значения научного понятия;
- отсутствие слов разговорного характера.

В то же время в научный стиль входят терминоситсемы, терминированные значения слов, абстрактные имена, освещающие отвлеченные понятия, модальные слова, которые выражают достоверность или же ложность высказывания, а также слова, определяющие порядок изложения мысли.

Официально-деловой стиль обслуживает сферу письменных официально деловых отношений и имеет следующие признаки:

- четко выражена социально-функциональная направленность;
- существует относительная стилевая закрытость;
- характерна стандартизация и унификация имён;
- присутствуют функциональные коннотации;
- распространены языковые штампы.

В состав официально-деловой лексики входит множество парадигм, зависящих от различия отраслей деловой жизни. Внутристилевые изменения в данном пласте лексики обусловлены сферой применения её в общественной жизни.

Для публицистического стиля характерны следующие отличительные черты:

- ориентированность сферы употребления слова;
- информативная точность слова;
- образность и броскость слова;
- использование слов с коннотативной семантикой, включая слова, несущие личностный компонент значения, и слова, с эмоционально-оценочными коннотациями;
- отсутствие стилевой замкнутости слова и его штампованность;
- языковые трафареты.

Наличие данных аспектов обусловлено тем, что публицистическая лексика преимущественно используется средствами массовой информации, для

освещения актуальных общественно-политических вопросов и формирования у читателя определенного отношения к существующей проблеме.

Разговорный стиль представляет собой сферу устного литературного языка, определяемую ситуацией неофициальности общения и характеризующуюся непринужденностью. Для разговорного стиля характерны такие маркеры как:

- устная форма речи;
- обусловленность ситуацией общения;
- неполнота выражения мысли;
- непринужденность общения;
- жесты и мимика как дополнительные средства передачи информации.

В разговорном стиле выделяют группу литературно-разговорной лексики и группу разговорно-бытовой лексики. Обеих групп характерно широкое использование указательных слов, использование гиперонимов вместо гипонимов, наличие слов «сниженного» значения с экспрессивными и эмоционально-оценочными коннотациями, распространенность отглагольных существительных, а также употребление составных номинаций вместо забытого слова.

Отдельно выделяют просторечную лексику, которая не является литературной разновидностью лексики. Слова из данного пласта речи обладают экспрессивно-стилистическими значениями, коннотативны. Также просторечные слова характеризуются следующими свойствами:

- сужением семантики слова;
- ложной этимологией;
- развитием просторечно-метафорических значений;
- эвфемизацией – заменой слова синонимом более высокого стиля;
- употреблением просторечных именных, глагольных и наречных именований;
- собственной сферой ласкательно-уменьшительных имен.

Как лингвосоциальное явление просторечие рассматривается как особый, нелитературный пласт речи.

После изучения предметной области, с которой предстоит работать, необходимо изучить основные этапы и стандарты проектирования и разработки непосредственно программного обеспечения, чтобы обезопасить себя от неудач, связанных с неправильным расчётом времени и необходимых средств для реализации требуемого программного продукта.

1.2 Диаграммы потока данных и модели проектирования программных продуктов

Разработка программного обеспечения подразумевает под собой определение внутренних свойств системы, детализацию её внешних свойств на основе требований, предъявляемых к проектируемому продукту. В процессе разработки для выражения характеристик программы принято использовать различные нотации, такие как блок-схемы, ER-диаграммы, а также DFD-диаграммы.

Начальным этапом в разработке программного обеспечения является анализ предметной области, установление границ решаемой задачи. Для этого разработчик прибегает к средствам системного анализа с целью выявления сущностей и последующего планирования необходимого функционала программы, её внешнего вида и структуры на основе полученной информации.

В данном контексте под сущностью понимают графическое представление логической группировки данных. Сущности могут являть собой как вещественный, реальный объект, так и концептуальную абстракцию. Сущности не предназначены для представления какого-либо единичного объекта, они

скорее представляют определенный набор экземпляров, которые содержат определенную информацию и представляют интерес в виду своей уникальности (Вендров, 1998).

Системный анализ – научно-практическая дисциплина, направленная на разработку методов исследования или проектирования сложных систем, а также путей и способов решения сложных проблем прикладного характера (Вендров, 1998).

В рамках проведения системного анализа перед разработчиком ставится задача выявления сущностей, их группировка, разбиение на соответствующее категории, для использования в дальнейшем.

Существуют зависимые и независимые сущности, в отличие от первых, таблицы представляющие второй тип не имеют внутри свои колонок ссылок на другие таблицы.

Выделяют следующие типы сущностей:

- стержневые сущности;
- кодовые сущности;
- ассоциативные сущности;
- характеристические сущности;
- структурные сущности.

Стержневая сущность может быть, как независимой, так и зависимой, однако она не является ни ассоциацией, ни обозначением, ни характеристикой. Также называется первичной или главной. Представляет собой основу для последующего моделирования и определения других сущностей.

Кодовая сущность – независимая сущность, определяющая область определения для значений атрибутов, которые принадлежат другим сущностям. Также называются ссылками, классификаторами или сущностями типов, в зависимости от используемой проектировщиком методологии (Вендров, 1998).

Ассоциативная сущность – зависимая сущность, содержащая первичные ключи двух или более других сущностей. Подобные сущности используются для моделирования отношения многие-ко-многим, в которых со множеством экземпляров одной сущности связывают множество экземпляров другой сущности. Благодаря ассоциативным сущностям обеспечивается возможность моделирования пересечения экземпляров двух сущностей с обеспечением уникальности каждого экземпляра ассоциации (Вендров, 1998).

Характеристическая сущность – зависимая сущность, используемая проектировщиком в случае, если появляется необходимость в хранении различных наборов атрибутов. Такие сущности имеют одну или более равноправных сущностей, связь которых с родительской сущностью может быть исключающей, либо включающей (Вендров, 1998).

Структурная сущность – служит для представления отношений между экземплярами одной и той же сущности (Вендров, 1998).

Первичный ключ – понятие, обозначающее атрибут или набор атрибутов, для идентификации уникального экземпляра сущности. В качестве основных требований, предъявляемых к первичному ключу, отмечают, что ключ должен быть статическим и неразрушаемым. Под этим подразумевается неизменяемость ключа на протяжении всего функционирования базы данных и достигается путем устранения зависимостей ключа от экземпляров сущности.

На основе выделенных сущностей, разработчиком строится визуальная модель, позволяющая выявить отношения и взаимосвязи между различными объектами. Для построения подобных моделей принято использовать различные инструменты анализа и моделирования, такие как AllFusion Process Modeler, Ramus Educational, Microsoft Visio.

Одним из примеров инструмента анализа и моделирования бизнеса является пакет BPwin, использующий технологии моделирования IDEF0, DFD и IDEF3. С помощью набора графических инструментов для отображения действий

и объектов BPwin позволяет легко построить схему процесса, на которой показаны исходные данные, результаты операций, ресурсы, необходимые для их выполнения, связи между отдельными процессами. Несмотря на критику в адрес данного программного продукта касаемо простоты его использования и удобности интерфейса, данная программа может быть использована для создания диаграмм потоков данных для различных проектов. Благодаря использованию подобных программ разработчик получает возможность точно смоделировать будущую систему, оценить необходимые затраты и риски и скорректировать сформированный план работ. Используя графическое отображение всех процессов появляется возможность детально рассмотреть все проблемные участки будущего программного продукта и учесть их при дальнейшем проектировании интерфейса программного продукта и написании его кода.

Диаграммы потоков данных (DFD) – основное средство графического моделирования функциональных требований к проектируемой системе. Благодаря использованию данных диаграмм, разработчик получает возможность графически представить структуру разрабатываемого программного продукта, связи между отдельными частями проекта, а также потоки данных и процессы, использующие их. Существующее вне системы объекты обозначаются как внешняя сущность, которая способна получать и передавать в функционирующую систему потоки данных.

Стандарт IDEF0 предназначен для создания функциональной модели, которая будет отражать структуру и функции системы, потоки информации и материальных объектов, связывающих данные функции. Описательная графическая модель демонстрирует что, как и кем выполняется в рамках изучаемой системы.

Стандарт IDEF3 предоставляет инструменты для наглядного исследования и моделирования сценариев – последовательности свойств объекта, в рамках рассматриваемого процесса.

Проектирование затрагивает планирование архитектуры программного обеспечения, устройства компонентов программного обеспечения, пользовательских интерфейсов. Благодаря правильному и грамотному проектированию возможно избежать множество неточностей и проблем с выполнением задания. Важно правильно оценить риски, выработать наиболее подходящую целям проекта стратегию, выбрать необходимую модель разработки программного продукта.

При выборе модели необходимо как можно точнее сформулировать требования к готовому программному продукту, и, на их основе, выбрать модель, которая будет наиболее точным образом отвечать всем предъявляемым к проекту требованиям. Для этого проект делится на несколько частей, каждая из которых отвечает своим требованиям. Выделяют следующие шаги в разработке программного обеспечения:

- анализ требований;
- проектирование программного обеспечения;
- программирование;
- тестирование;
- внедрение программного обеспечения;
- сопровождение программного обеспечения.

В зависимости от конкретной модели реализации, состав или порядок данных шагов может быть изменен в соответствии с требованиями, предъявляемыми к конкретной модели.

В рамках выполнения дипломной работы были рассмотрены следующие модели разработки программного обеспечения:

- каскадная модель;
- спиральная модель;
- итерационная модель.

Каскадная модель – модель процесса разработки программного обеспечения, в которой данный процесс выглядит как поток, в линейной последовательности проходящий через все фазы от анализа до тестирования. Переход к следующему этапу подразумевает под собой полное завершение настоящего, что в долгосрочной перспективе ставит под угрозу качество выполнения проекта, в случае его недостаточно точного планирования, поскольку возникают проблемы с возвращением к предыдущим этапам, вызванные особенностями работы данной модели (Вендров, 1998).

Для данной модели характерно четкое формирование всех требований и спецификаций на начальной стадии проекта, что ведет к трудностям в случае корректировки технического задания. Однако, среди преимуществ подобного метода разработки выделяют выпуск полного комплекта документации при завершении каждого этапа, а также, благодаря жесткому планированию сроков и целей, простоту определения сроков и затрат на реализацию проекта.

Спиральная модель – при использовании данной модели, создается несколько витков, соответствующих созданию какого-либо фрагмента или версии программного обеспечения. При каждой итерации осуществляется уточнение целей и характеристик проекта, оценивается качество полученных результатов и планируются работы следующей итерации. Большое внимание при использовании данной модели уделяется рискам, влияющим на организацию жизненного цикла. На каждом этапе возможно применение разных моделей процесса разработки программного обеспечения. В конечном итоге на выходе получается готовый программный продукт. Переход на следующий этап разработки возможен даже при неполном завершении предыдущего, поскольку существует возможность последующей доработки при следующих итерациях (Вендров, 1998).

Главная задача при использовании подобной модели разработки – как можно быстрее продемонстрировать работоспособный программный продукт,

для получения возможности уточнения требований и корректировки технического задания. В то же время, главной проблемой подобной модели является определение момента перехода на следующий этап, что решается путем установления четких сроков перехода на следующий этап. Данные сроки определяются исходя из плана, составляемого на основе данных полученных в процессе выполнения предыдущих проектов, а также на основе опыта разработчиков. Важно понимать, что спиральная модель не является альтернативой эволюционной, а является лишь специально проработанным вариантом.

Итерационная модель – подход в разработке программного обеспечения, при котором выполнение работ происходит параллельно с непрерывным анализом, полученных результатов и корректировкой предыдущих этапов работы. При этом проект каждый раз проходит повторяющийся цикл, что позволяет исправить ошибки на ранних стадиях проекта (Вендров, 1998).

Использование данной модели имеет множество преимуществ, поскольку позволяет разработчику сосредоточить свое внимание на более важных направлениях проекта, обеспечивает возможность постоянного согласования с заказчиком целей и задач программного продукта, непрерывное тестирование и обнаружение конфликтов между требованиями и реализацией проекта на более ранних этапах. Также, благодаря итерированию разработки возможно эффективное использование накопленного опыта, а также более равномерная загрузка участников проекта. Любые затраты на распределяются равномерно по всему проекту, а не группируются в завершающей стадии.

Также важным фактором в процессе разработки программного обеспечения является выбор непосредственно языка программирования, поскольку в том числе и от выбора языка зависит структура проекта, возможности, требования и способы реализации программного продукта.

1.3 Язык программирования C++ и основы объектно-ориентированного программирования

Язык программирования C++ является потомком языка C и изначально возник из ряда улучшений к нему, придуманных Бьёрном Страуструпом для собственных нужд. В последствие данная работа развилась в отдельный язык, который обладал огромной функциональностью и открывал перед программистом возможности объектно-ориентированного программирования.

C++ — универсальный язык программирования, задуманный так, чтобы сделать программирование более приятным для серьезного программиста. За исключением второстепенных деталей C++ является надмножеством языка программирования C. Помимо возможностей, которые дает C, C++ предоставляет гибкие и эффективные средства определения новых типов. Используя определения новых типов, точно отвечающих концепциям приложения, программист может разделять разрабатываемую программу на легко поддающиеся контролю части. Такой метод построения программ часто называют абстракцией данных. Информация о типах содержится в некоторых объектах типов, определенных пользователем. Такие объекты просты и надежны в использовании в тех ситуациях, когда их тип нельзя установить на стадии компиляции. Программирование с применением таких объектов часто называют объектно-ориентированным. Одним из ключевых понятий C++ является класс.

Класс — это тип, определяемый пользователем. Он обеспечивает скрытие данных, гарантированную инициализацию данных, неявное преобразование типов для типов, определенных пользователем, динамическое задание типа, контролируемое пользователем управление памятью и механизмы перегрузки операций.

Наряду с классом необходимо осветить такие понятия как инкапсуляция, полиморфизм и наследование, поскольку именно они определяют саму концепцию объектно-ориентированного программирования.

Инкапсуляция – представляет собой механизм, ограничивающий доступ одних компонентов программы к другим (Лафоре, 2013). Понятие инкапсуляции связано с принципом абстракции данных, характерным для объектно-ориентированного программирования. Благодаря инкапсуляции программист получает возможность выделять в теле программы основные функции и писать более надежные с точки зрения безопасности программы.

Полиморфизм – в программировании под полиморфизмом понимается способность функции обрабатывать различные типы данных (Лафоре, 2013). Выделяют два вида полиморфизма: специальный и параметрический, который является истинной формой полиморфизма. Благодаря специальному полиморфизму программист получает возможность реализовать выполнение потенциально разного кода для различных типов данных. Подобный эффект достигается путем перегрузки функций. В отличие от специального, параметрический полиморфизм обеспечивает возможность реализации одного и того же кода для разного набора типов. Это делает язык программирования более выразительным и существенно повышает коэффициент повторного использования кода, что в свою очередь является одной из основных целей программирования.

Наследование – одна из концепции объектно-ориентированного программирования, согласно которой некий абстрактный тип данных способен наследовать данные и функционал другого типа, тем самым способствуя повторному использованию компонентов программного обеспечения (Лафоре, 2013). Помимо передачи части свойств и данных наследование обеспечивает возможность реализации иерархии классов, которая позволяет программисту писать программы, обладающие более строгой и удобочитаемой структурой.

Абстракция данных как важная часть объектно-ориентированного программирования упоминается во всех книгах, посвященных C++. В контексте программирования есть придание объекту характеристик, которые чётко определяют границы его концепта, демонстрируя отличия между данным объектом и остальными объектами. Основная идея состоит в том, чтобы отделить способ использования составных объектов от деталей их реализации.

Как и любой другой язык программирования, C++ несомненно, имеет свои достоинства и недостатки, учитывать которые необходимо перед тем, как приступать к проектированию и написанию программы на C++.

Среди преимуществ C++ программисты использующие его склонны выделять следующие пункты:

- высокая совместимость с языком C, что означает, что написанный на C код может быть скомпилирован компилятором C++ подвергаясь минимальным изменениям. Также множество библиотек C возможны к использованию с языком C++, это повышает порог вхождения, делая C++ более доступным для широкой аудитории;
- вычислительная производительность в C++ также выделяется как преимущество, поскольку данный язык проектировался таким образом, чтобы обеспечить программисту максимальный контроль над всеми аспектами структуры и порядка исполнения программы;
- поддержка различных стилей программирования: традиционного, структурного, объектно-ориентированного, функционального, обобщенного и других;
- автоматический вызов деструкторов объектов в порядке обратном вызову конструкторов упрощает и повышает надёжность управления памятью и другими ресурсами, такими как, например, соединения с базами данных;
- перегрузка операторов, которая гарантирует краткую и емкую запись выражений над пользовательскими типами;

- существование шаблонов в C++ дает пользователю возможность построения обобщенных контейнеров и алгоритмов для разных типов данных, а также шаблоны предоставляют возможность производить вычисления на этапе компиляции;
- доступность C++ в следствие существования огромного количества учебной литературы, обеспечивающей высокий порог вхождения.

Однако, несмотря на существование огромного количества преимуществ, с точки зрения критиков, язык программирования C++ обладает и довольно внушительным количеством недостатков, делающих его использование в некоторых случаях нецелесообразным. В качестве подобных минусов данного языка приводятся следующие положения:

- отсутствие системы модулей, что приводит к использованию заголовочных файлов, влекущих за собой необходимость дважды записывать одну и ту же функцию, определение в файле с исходным кодом и объявление в заголовочном файле, а также увеличивает время компиляции;
- сложный синтаксис и сложная спецификация языка;
- отсутствие возможностей рефлексивного, порождающего метапрограммирования, функционального программирования;
- необходимость полного контроля поведения функций, которые, однако, не до конца возможны, в виду того, что при высокой сложности системы, программист не способен в полной мере осознать систему или её элемент, что будет приводить к написанию неэффективного кода или даже кода, результат выполнения которого будет отличаться от ожидаемого;
- поскольку в C++ инкапсуляция и сокрытие ошибочно отождествляются, контролируя совпадение типов на уровне идентификаторов, но не сигнатур, это ведет к невозможности подмены модулей, основываясь на совпадении функциональности их интерфейсов;

- средства макроподстановки из языка С являются опасным средством, однако все равно были сохранены в С++, несмотря на более строгие и специализированные средства, что увеличивает зависимость программного кода от человеческого фактора.

В ответ на перечисление подобных недостатков сторонники языка С++ утверждают, что программист не вынужден использовать «плохие» языковая средства, если у него имеются необходимые навыки и он способен использовать имеющиеся в языке для этой цели «хорошие» средства.

Таким образом, можно сделать вывод, что в данном языке большая часть успеха зависит от навыков программиста, поскольку именно он отвечает за выбор средств для реализации решения выбранной проблемы и именно на него ложится ответственность по контролю за правильным использованием всего функционала языка. Отсюда следует, что С++ предоставляет гораздо лучшие, чем в С, средства выражения модульности программы и проверки типов. В С++ сохранены возможности языка С по работе с основными объектами аппаратного обеспечения (биты, байты, слова, адреса и т.п.). Это позволяет весьма эффективно реализовывать типы, определяемые пользователем. Данный язык, как и его стандартные библиотеки спроектированы так, чтобы обеспечивать переносимость. Имеющаяся на текущий момент реализация языка будет работать в большинстве систем, поддерживающих С. Из С++ программ можно использовать С библиотеки, и с С++ можно использовать большую часть инструментальных средств, поддерживающих программирование на С.

После освещения основных положений С++ как языка объектно-ориентированного программирования возникает необходимость в изучении сред разработки для данного языка, которые будут соответствовать требованиям, который предъявляет разработчик.

1.4 Сравнение сред разработки программных продуктов для языка C++

В виду популярности C++ как языка программирования, существует множество сред разработки, обеспечивающих удобное использование всех возможностей C++.

Говоря о средах разработки обычно перечисляют следующие программные продукты:

- Microsoft Visual Studio;
- Borland Developer Studio;
- Eclipse CDT;
- NetBeans;
- JetBrains Clion.

Интегрированная среда разработки Microsoft Visual Studio представляет собой набор инструментов для создания программного обеспечения. Имеющийся функционал позволяет выполнять множество различных задач, начиная от планирования до разработки пользовательского интерфейса, написания кода, его тестирования и отладки, до анализа его качества и производительности. Все эти инструменты доступны в интегрированной среде разработки и предназначены для максимально эффективного взаимодействия друг с другом.

Данная среда разработки может быть использована для выполнения большого спектра задач, начиная от разработки небольших приложений для персональных компьютеров и мобильных телефонов и заканчивая проектированием и созданием больших и сложных систем, которые будут использоваться для обслуживания предприятий. Наличие большого количества документации и обратной связи с командой разработчиков позволяет эффективно и быстро решать встречающиеся при использовании этого программного

продукта проблемы. Также данная среда предоставляет конструктор для проектирования пользовательского интерфейса, позволяющий создавать его без написания кода.

Borland C++ Builder является средой быстрой разработки приложений. В основе систем быстрой разработки (RAD-систем, Rapid Application Development — среда быстрой разработки приложений) лежит технология визуального проектирования и событийного программирования, суть которой заключается в том, что среда разработки берет на себя большую часть работы по генерации кода программы, оставляя программисту работу по конструированию диалоговых окон и написанию функций обработки событий. Производительность программиста при использовании RAD систем существенно увеличивается (Послед, 2003).

Система объектно-ориентированного программирования Borland C++ Builder, предназначена для операционных систем Windows. Интегрированная среда C++ Builder обеспечивает скорость визуальной разработки, продуктивность повторно используемых компонентов в сочетании с мощью языковых средств C++, усовершенствованными инструментами и разномасштабными средствами доступа к базам данных. Поддерживая технологию Drag&Drop, данная интегрированная среда разработки позволяет создавать элементы пользовательского интерфейса программы путем их перетаскивания на форму. Изменение расположения и размеров элемента не требует от программиста никаких дополнительных манипуляций с кодом.

C++ Builder может быть использован везде, где требуется дополнить существующие приложения расширенным стандартом языка C++, повысить быстродействие и придать пользовательскому интерфейсу качества профессионального уровня.

Eclipse CDT является бесплатной программой с открытым исходным кодом. Реализована на языке программирования Java и своей основной целью ставит

повышение продуктивности процесса разработки программного обеспечения. Является одной из самых популярных бесплатных платформ наравне с NetBeans.

Среди её преимуществ выделяют следующее пункты:

- доступность;
- существование множества как коммерческих, так и бесплатных модулей, позволяющих расширить функционал и специфицировать данную среду разработки под требуемый набор задач;
- кроссплатформенность.

Однако, количество учебной литературы для данной интегрированной среды разработки, гораздо меньше, чем для предыдущих. Также стоит принимать во внимание возможные сложности, с которыми может столкнуться пользователь при поиске необходимых ему дополнений и установке отдельных частей, как, например, компиляторов, которые не поставляются вместе с данной средой.

NetBeans также, как и Eclipse является модульной динамической средой, и имеет схожие достоинства и недостатки. Однако, в отличие от предыдущей среды, NetBeans использует платформо-независимую библиотеку Swing.

JetBrains CLion является относительно новой интегрированной средой и предоставляет пользователю широкий функционал, включающий в себя динамические справки, подсветку синтаксиса, автодополнения, многочисленные механизмы поиска и анализ кода на потенциальные ошибки. Поставляемый программный продукт представляет собой полную интегрированную среду разработки, которая не требует загрузки дополнительных модулей или сторонних компиляторов.

Все перечисленные среды разработки имеют свои достоинства и недостатки, а выбор необходимой среды, в которой будет разрабатываться требуемый программный продукт, ложится на плечи разработчика, который принимает данное решение исходя из личных предпочтений.

Вывод к Главе 1

На основании рассмотренного теоретического материала по лексикологии, проектированию программных продуктов и языку программирования C++ было установлено следующее.

Слово, как языковая единица, имеет ряд свойств, благодаря которым становится возможно отличать его от морфем. Помимо этого, оно также может иметь как грамматическое, так и лексическое значение. Значения в свою очередь возможно рассматривать с различных аспектов, что несомненно дает большую область для проведения различного рода лингвистических исследований.

Использование диаграмм потоков данных при разработке программного продукта значительно облегчает задачу разработчика. В зависимости от задачи, которую необходимо решить, следует выбрать наиболее подходящий стандарт.

Существует множество моделей разработки программного обеспечения. Каждая из них обладает своим набором требований к процедуре создания программного продукта, что налагает свои ограничения на разработчика. При выборе модели разработки следует учитывать множество факторов, которые способны влиять на сроки выполнения поставленной задачи.

Помимо выбора модели разработки программного продукта, от разработчика также зависит выбор языка реализации. Как было установлено C++ является объектно-ориентированным языком и содержит в себе огромный функционал для решения разнообразных задач. Являясь мощным средством разработки язык C++ налагает на программиста ответственность за принятие различного рода решений, и предоставляя ему огромный потенциал для выполнения поставленной задачи, требует от программиста полного контроля производимых манипуляций. Не смотря на множество негативных комментариев,

C++ является одним из самых популярных языков программирования, что свидетельствует о его популярности среди разработчиков.

В связи с тем, что язык C++ является достаточно популярным, существует огромное множество сред разработки, которые облегчают работу программиста. Несмотря на то, что большинство сред разработки практически идентичны, было принято решение об использовании Borland C++ Builder для решения данной задачи. Являясь одной из самых старых сред разработки, данный программный продукт имеет множество документации. Помимо этого, благодаря возможности быстрого создания пользовательского интерфейса с использованием технологии Drag&Drop, Borland C++ Builder значительно ускоряет процесс разработки программного продукта.

Глава 2. Разработка и создание виртуального словаря на основе базы данных

2.1 Разработка алгоритма, создание диаграмм потоков данных и пользовательского интерфейса

При разработке программного продукта, первым этапом является создание модели данных в нотации IDEF0, с целью выработки необходимых решений путём представления функционала программы в виде схемы, в которой указываются все участники процесса и выполняемые ими действия.

Для выполнения данной задачи было использовано приложение для создания диаграмм и схем Microsoft Visio 2010. Данный программный продукт содержит большое количество шаблонов, упрощающих работу по визуальному представлению данных. Поставляемая компанией Microsoft, данная программа имеет выполненный в том же стиле что и вся линейка программ Microsoft Office интерфейс, также произведена замена формата меню на ленту, что также благоприятно сказывается на удобстве использования данной программы. Благодаря наличию большого количества справочной литературы использование программы становится простым даже для не знакомого с ней пользователя. Также нельзя не отметить быстродействие и интуитивно понятные интерфейс данного программного продукта.

В ходе выполнения работы была построена диаграмма IDEF0 и её декомпозиция.

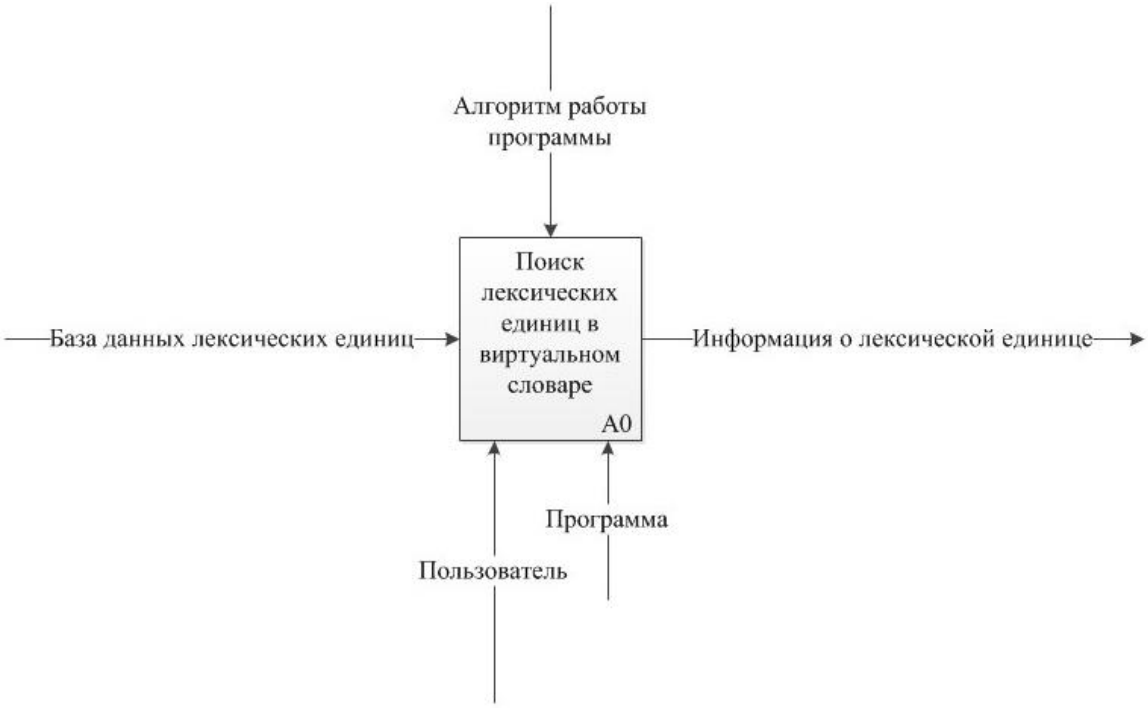

Рис. 2.1. Диаграмма IDEF0

Как видно на данной диаграмме (Рис. 2.1) представлен процесс «Поиск лексических единиц в виртуальном словаре», в качестве источника входного набора данных для дальнейшего использования выступает база данных лексических и фразеологических единиц русского и английского языков. В процессе поиска требуемой единицы принимают участие как пользователь данного программного продукта, так и сама программа. Отбор выводимых данных производится на основе разработанного алгоритма программы поиска лексических единиц. В результате обработки внутри процесса поступившего на вход набора данных, выводится информация об искомой лексической единице, на основе выбранных пользователем параметров фильтрации.

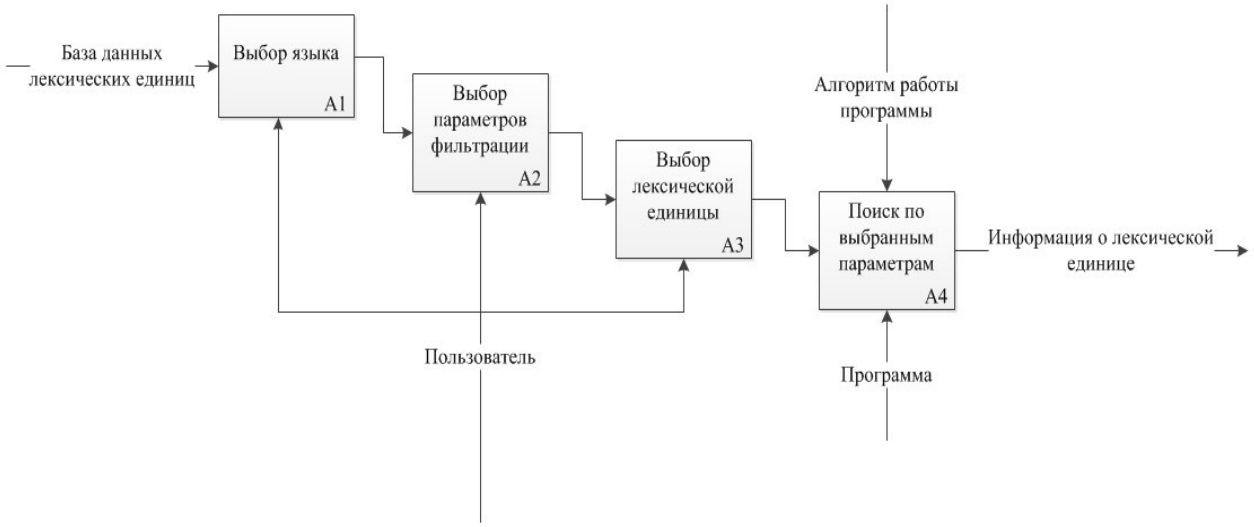

Рис. 2.2. Декомпозиция диаграммы IDEF0

Результатом декомпозиции диаграммы IDEF0 процесса «Поиск лексических единиц в виртуальном словаре» является пошаговое представление включенных в процесс этапов. Как видно из представленной схемы (Рис. 2.2), после того как из базы данных в программу загружается информация, пользователь выбирает необходимый язык, для отображения соответствующего набора лексических единиц, после чего определяет необходимые ему параметры фильтрации и выбирает требуемую лексическую единицу, после чего, на основе установленных пользователем параметров фильтрации, программа, используя разработанный для этого алгоритм выполняет фильтрацию, исключает не соответствующие параметрам поиска лексические единицы и выводит на экран запрошенную пользователем при помощи фильтров информацию.

Благодаря визуализации процессов, происходящих внутри программного продукта, разработчик получает более ясное представление стоящей перед ним задачи, что в свою очередь благоприятно сказывается на итоге разработки требуемой программы. После визуального представления процессов можно приступать к написанию программного кода.

В первой главе были перечислены среды разработки программных продуктов, выявлены их основные достоинства и недостатки. В качестве среды разработки для создания данного программного продукта была выбрана Borland C++ Builder, которая представляет собой совершенную интерактивную среду программирования на языке C++. Система обеспечивает высокую продуктивность и производительность, удовлетворяя современным требованиям в разработке приложений под Windows.
Следующим шагом после выбора языка программирования и среды разработки стала разработка алгоритма реализации виртуального словаря на основе баз данных. При разработке алгоритма необходимо не только описать функции, которые будет выполнять программа, но и рассчитать необходимое количество форм, используемых для отображения компонентов управления программой, и повышающих простоту её использования. Благодаря тому, что на предыдущем этапе была создана диаграмма потоков данных, стало значительно проще представить каким функционалом должна обладать разрабатываемая программа, и на основе этого выделить ряд требований, которые к ней предъявляются, для формирования алгоритма работы программного продукта.
Разрабатываемая программа должна отвечать следующим требованиям:

1. обеспечивать вывод необходимой информации по искомой лексеме;
2. обеспечивать пользователя возможностью настраивать параметры фильтрации;
3. гарантировать невозможность изменения пользователем записей в базе данных;
4. исключать отображение излишней информации, используемой исключительно для межтабличных связей и фильтрации данных;
5. иметь справочный раздел, содержащий информацию о работе с программным продуктом.

В связи с данными условиями, было принято решение в необходимости создания трех форм, каждая из которых будет отвечать определенным требованиям реализуя часть задуманного функционала.

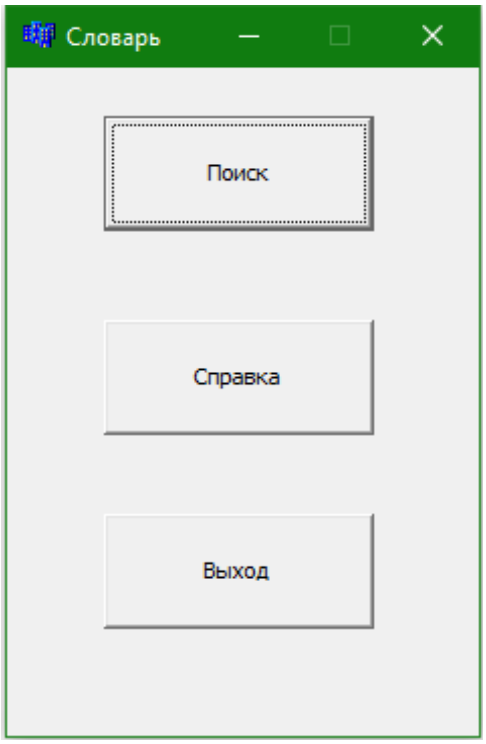

Рис. 2.3. Форма «Главное Меню»

На первой форме (Рис. 2.3) были размещены три компонента Button, которые в дальнейшем будут отвечать за переход на соответствующие формы поиска и справки или выход из приложения.

Для облегчения работы пользователя с программой разработчик дает каждому компоненту осмысленное название, отображающее суть действия, выполняемого этим компонентом. Так, например, для размещения на кнопке собственного текста достаточно поменять его значение в поле Caption, располагающемся на вкладке Properties, пример для компонента Button приведен на Рис. 2.4.

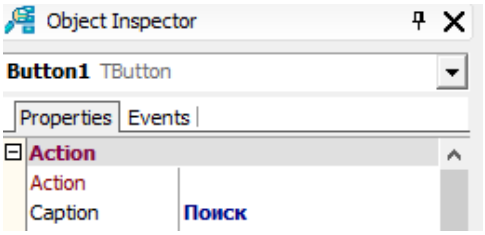

Рис. 2.4 Свойство Caption

Для того, чтобы при определенном событии выполнялось какое-либо действие, необходимо выбрать вкладку Events и, выбрав из списка необходимое условие задать при помощи программного кода, действия, которые будут выполнятся при выполнении данного условия. Пример использования вкладки Events и выбора условия OnClick, выполняющего определенное действие при щелчке по данному элементу левой кнопкой мыши приведен ниже на Рис. 2.5.

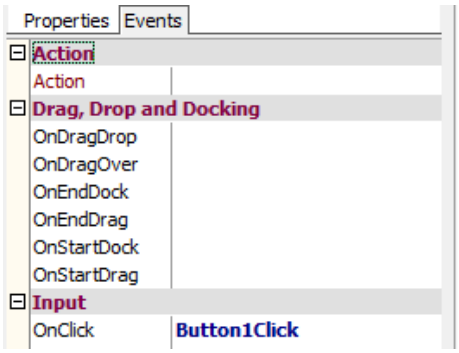

Рис. 2.5. Событие OnClick

На второй форме (Рис. 2.6) были размещены следующие компоненты:

- DBGrid – таблица, отображающая инофрмацию содержащуюся в определенной таблице в базе данных;
- два компонента Label предназначенных для вывода на форму текста, который выполняет роль подписи для блоков переключателей;
- два компонента RadioButton обеспечивающих выбор одной опции из определенного набора;

- четыре компонента CheckBox предназначенных для задания пользователем параметров фильтрации выводимой программой информации;
- компонент Edit, который используется для ввода пользователем какого-либо текста с клавиатуры;
- компонент RichEdit, выводящий на экран текст, который содержит информацию по искомой лексеме на основе установленных пользователем параметров поиска.

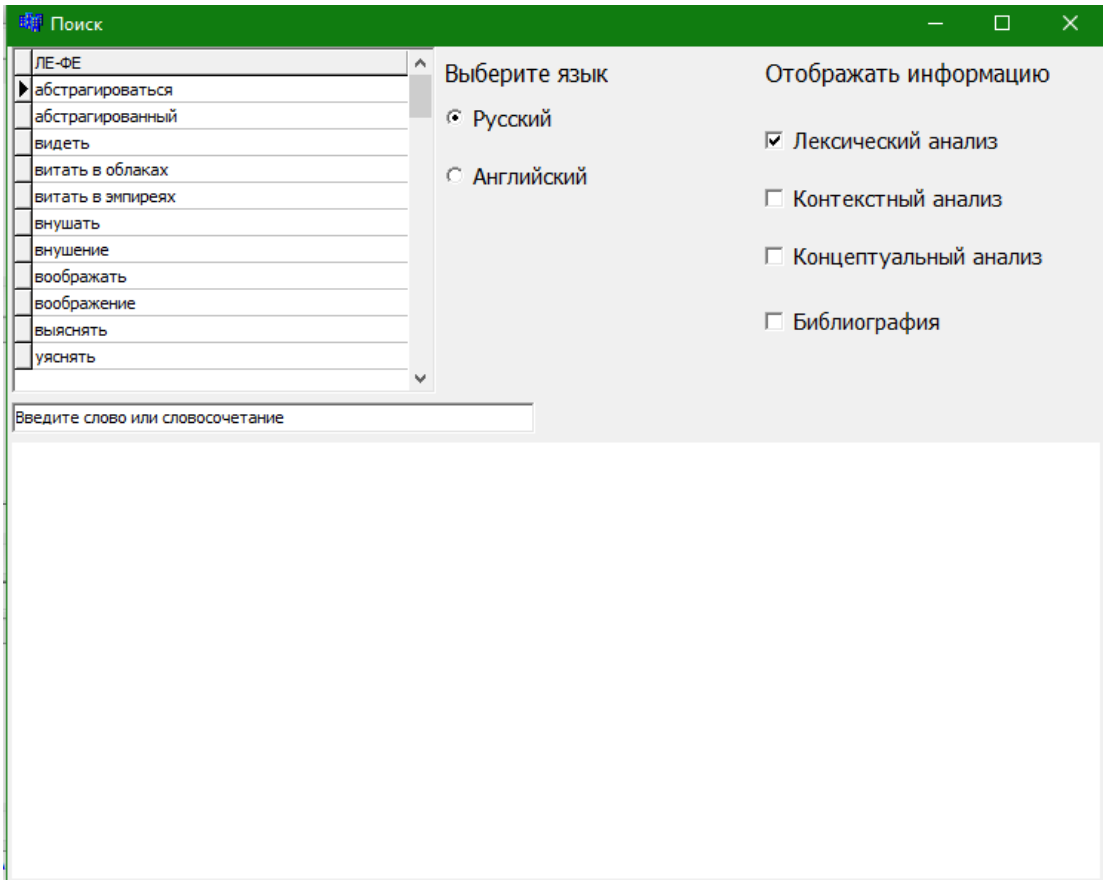

Рис. 2.6. Форма «Поиск»

Переход на данную форму осуществляется при нажатии соответствующей кнопки в главном меню программы. Как и в случае с первой формой, для того чтобы разместить необходимый текст, разработчик меняет свойство Caption у интересующего его компонента. Такие компоненты как CheckBox и RadioButton

имеют на вкладке Properties свойство Checked, благодаря которому разработчик может установить в каком из двух состояний будет находиться компонент при инициализации приложения. Несмотря на то, что технически, разработчик может оставить все кнопки не выбранными, многие руководства, в том числе и Apple HIG рекомендуют не оставлять группы без выбранных кнопок.

На следующей форме (Рис. 2.7) размещен один компонент – RichEdit – отображающий текстовый файл, содержащий в себе пояснения касаемо данного программного продукта в целом. Данный компонент позволяет выводить текст, учитывая его особенности форматирования.

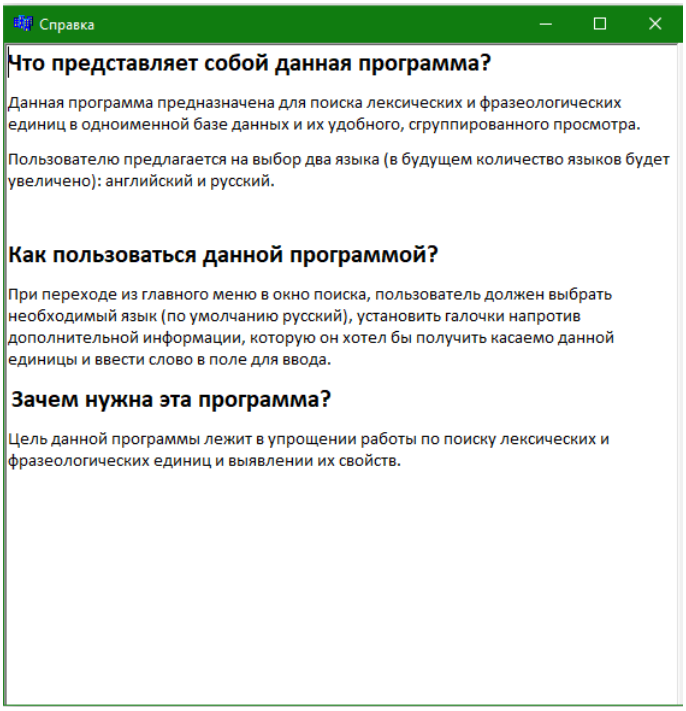

Рис. 2.7. Форма «Справка»

В отличие от компонентов Edit и Label, за изменение текста, отображающегося в компоненте RichEdit при инициализации программы, отвечает свойство Lines, находящееся на вкладке Properties, при выборе которого появляется окно с редактором текста (Рис. 2.8).

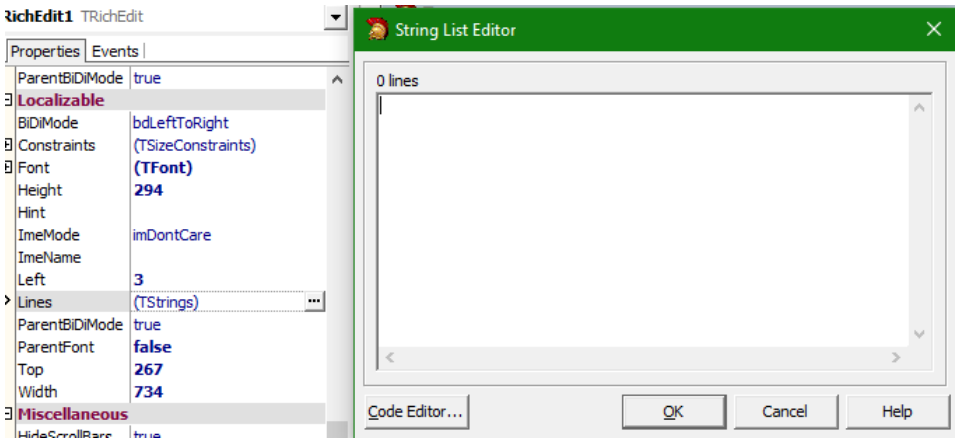

Рис. 2.8. Свойство Lines и редактор текста

После создания требуемых форм и размещения на них необходимых компонентов перейдем к отбору лингвистического материала, на основе которого и будет строиться база данных.

2.2 Описание процесса отбора лексем для создания физической базы данных

Для создания базы данных, прежде всего, необходимо произвести отбор лингвистического материала, который в последующем будет внесен в базу данных. Ниже описан процесс отбора лексических единиц.

На первом этапе производилась сплошная выборка из аутентичных источников фактических данных, учитывая способность отбираемых слов номинировать когницию. Однако формирование исследовательского тезауруса на следующем этапе идет по принципу уточнения его содержательного показателя. С этой целью, то есть с целью верификации полученных данных, проводится лексикографический анализ на предмет способности лексемы

системно описывать рассматриваемый референт. Для этого выявляются все возможные дефиниции единицы, и отслеживается прямая способность номинации ситуации когниции. Иными словами, лексема проверяется на соответствие критерию «способность номинировать когницию на системном уровне». Дальнейшая верификация происходит в корреляции с выявленным ранее концептуальным содержанием искомой ментальной структуры. Это дословно означает, что все с учетом вышеперечисленных параметров та или иная лексема должна на системном и/или функциональном уровне актуализировать такие концептуальные составляющие как «внимание», «перцепция», «ментальная деятельность», «память», «воображение».

После того как произведен отбор по критериям номинации когниции и частотности, отобранные слова исследуются с точки зрения соответствия таким концептуальным составляющим как «внимание», «перцепция» и так далее. Учитывая принципиальные составляющие гештальта – фигуру и фон, а также экстралингвистическое соотношение понятий «синкретизм : : дискреность» мы также говорим о включенности компонентов и их соотношении с фоном, а также наличии собственных специфических черт, которые предопределяют и предопределяются дискретностью компонентов ментальной модели. Так, например, компонент «Перцепция» является синкретичным фоном для включения в него компонента «Восприятие», а компонент «Ментальная деятельность» выступает на таких же основаниях мотивирующим началом для компонентов «Мышление», «Воображение», «Память». Маргинальное положение по отношению к компонентам «Перцепция» и «Ментальная деятельность» занимает компонент «Внимание» на том основании, что экстралингвистический референт рассматриваемого компонента сопровождает и восприятие, и ментальную деятельность, не существуя при этом «в чистом виде». На более высоком уровне иерархии гештальта также наблюдается соотношение

«целое-часть» в области компонентов «Результат познания» и «Процесс познания», соответственно.

В итоге при отборе лексических единиц для составления базы данных и последующего её использования необходимо провести глубокий анализ учитывая взаимосвязанность компонентов.

2.3 Описание процесса установки связей между приложением и физической базой данных

После того как процесс отбора лексем и создания базы данных завершен, необходимо выполнить подключение базы данных к приложению. Данная операция выполняется в среде разработки Borland C++ Builder. Для размещения компонентов работы с базой данных удобно использовать отдельный компонент-контейнер DataModule, который, условно, можно считать разновидностью формы. Его основное отличие заключается в том, что он не виден пользователю во время работы программы и его удобно использовать для размещения компонентов, отвечающих за бизнес-логику программного продукта и доступ программы к базе данных, при этом не затрудняя проектирование интерфейса, в отличие от компонента Form, на котором принято размещать видимые элементы, для взаимодействия пользователя с программой и размещение на котором большого количества невидимых компонентов приводит к затруднению проектирования пользовательского интерфейса. Для того чтобы добавить в проект компонент DataModule необходимо выбрать последовательно меню File, New и DataModule, пример представлен на Рис. 2.9.

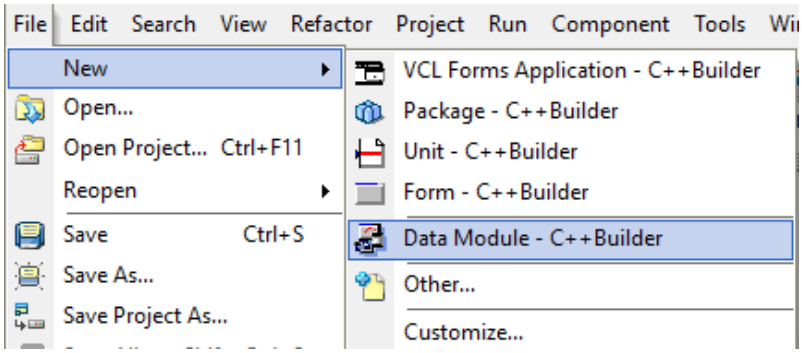

Рис. 2.9 выбор компонента DataModule

Для взаимодействия с информацией, содержащейся в базе данных, в Borland C++ Builder содержится большое количество компонентов, различающихся по выполняемым ими функциям. После создания компонента DataModule разработчик размещает на неё компонент ADOConnection содержащийся на вкладке компонентов dbGo (Рис. 2.10) и отвечающий за связь разрабатываемого приложения с физической базой данных.

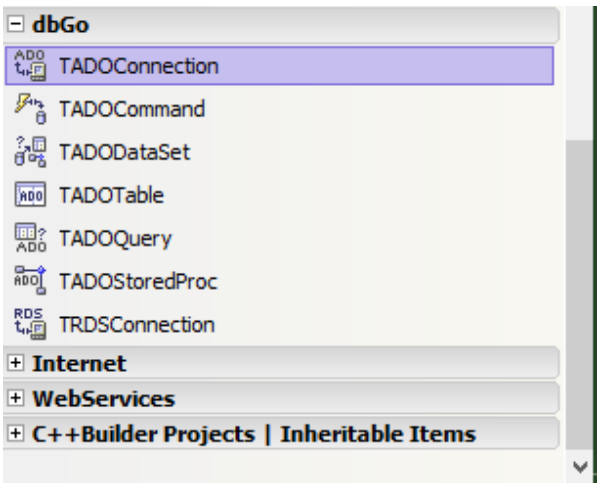

Рис. 2.10. Компонент ADOConnection

После добавления данного компонента на форму, необходимо настроить подключение к физической базе данных для того, чтобы получить возможность оперировать данными используя разрабатываемое приложение. Первым шагом в

настройке является указание пути к необходимой базе данных. Открыть настройку данного параметра можно двумя способами: двойным щелчком по свойству ConnectionString находящемуся в разделе Database вкладки Properties (Рис. 2.11), либо двойным щелчком по добавленному на ворму компоненту ADOConnection.

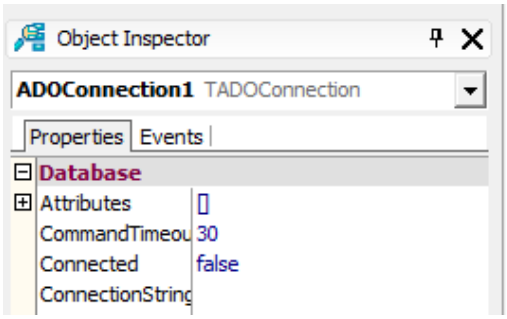

Рис. 2.11. Свойство ConnectionString

После выполнения данного действия любым из двух способов перед разработчиком откроется окно настройки свойств канала передачи данных (Рис. 2.12), в котором первым шагом будет предложено выбрать поставщика данных. Поскольку в данной программе используется база данных стандарта mdb от Microsoft Access, в качестве подключаемых данных необходимо выбрать Microsoft Jet 4.0 OLE DB Provider.

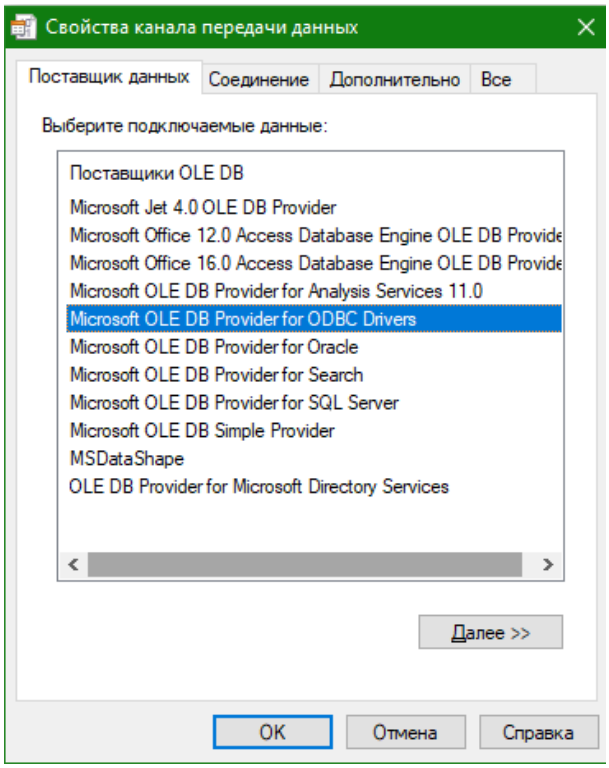

Рис. 2.12. Настройка свойств канала передачи данных. Вкладка «Поставщик данных»

После выбора необходимого драйвера разработчик, нажимая кнопку далее, попадает на следующую вкладку, в которой ему необходимо указать сведения для подключения к данным Access (Рис. 2.13). Полный путь можно как записать вручную, так и выбрать, воспользовавшись стандартным проводником Windows, который открывается на кнопку с многоточием (на рисунке выделена синей рамкой). Кроме того, существует возможность задать дополнительные свойства, на соответствующей вкладке, а также установить пароль, для запуска базы данных. В случае если использование пароля не требуется, то следует установить в графе «Пустой пароль» галочку. При выполнении этого действия пароль при работе с базой данных запрашиваться не будет.

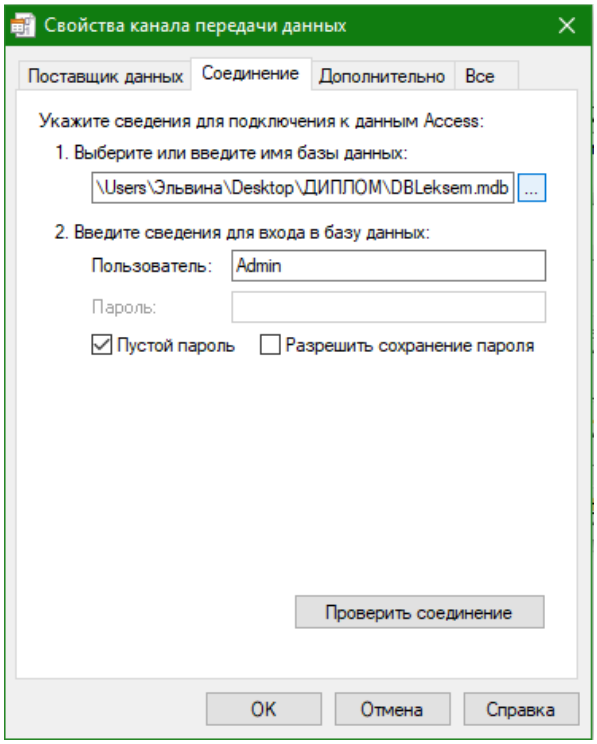

Рис. 2.13. Настройка свойств канала передачи данных. Вкладка «Соединение»

Для того чтобы проверить корректно ли указан путь и установлено ли соединение между приложением и физической базой данных, разработчик может воспользоваться кнопкой «Проверить соединение». В случае если путь указан правильно, программа выведет соответствующее окно, подтверждающее установление соединения (Рис. 2.14).

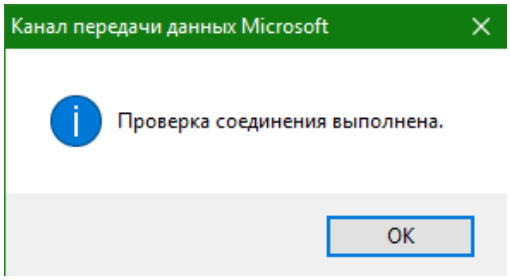

Рис. 2.14. Окно, подтверждающее успешное установление связей между базой данных и приложением

Если, как в нашем случае, отсутствует необходимость запрашивать при инициализации программы имя пользователя и пароль для выполнения входа в базу данных, разработчику следует изменить свойство LoginPrompt в группе Database вкладки Properties, на false (Рис. 2.15), в противном случае приложение, перед тем как выполнить выгрузку записей из базы данных, будет запрашивать имя пользователя и пароль.

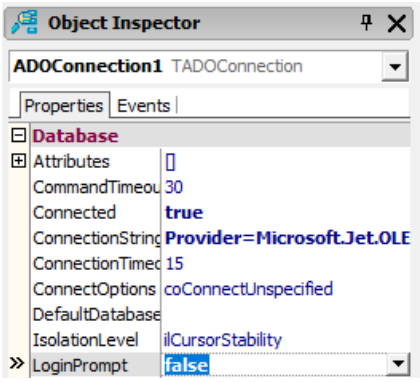

Рис. 2.15. Изменения свойства LoginPrompt

После того как все описанные выше действия выполнены и связь с базой данных установлена, можно приступать к размещению компонентов ADOTable и DataSource. Первый компонент находится на той же вкладке что и ADOConnection (Рис. 2.11) и отвечает за соединение с конкретной таблицей в физической базе данных. Во вкладке Properties данного компонента (Рис. 2.16) имеются несколько полей, использование которых необходимо в данной программе. Так поле Name во вкладке Miscellaneous аналогично свойству Caption для Label и отвечает за то, какое название будет отображаться у данного компонента, поскольку работа с данной базой данных подразумевает размещение большого количество похожих компонентов, необходимо давать им осмысленные названия, отражающее суть содержимого. Далее параметр Active в той же вкладке, определяет будет ли данная таблица активна во время инициализации программы, данному полю можно присвоить значение true, в

случае если выбрано имя отображаемой таблицы TableName из вкладки Database. Кроме этого, данный компонент имеет свойство Filtered, отвечающее за фильтрацию данных на основе параметров, задаваемых в поле Filter, работа с которым будет подробнее рассмотрена позже.

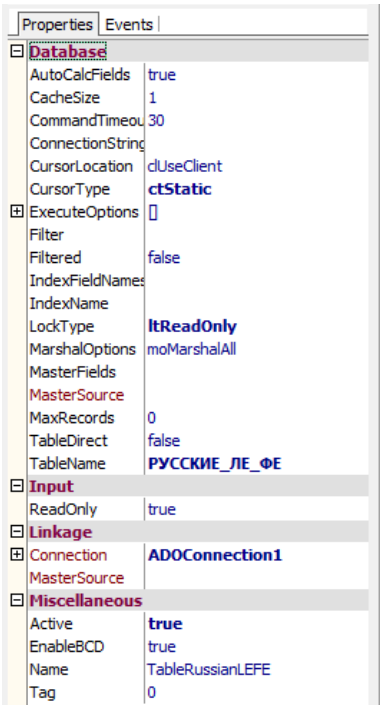

Рис. 2.16. Вкладка Properties компонента ADOTable

После того, как мы определили, с какой из таблиц физической базы данных будет связан данный компонент, необходимо выбрать поля, к которым он будет иметь доступ. Для этого необходимо дважды щелкнуть левой кнопкой мыши по компоненту расположенному на форме, после чего появится окно, отображающее уже подключенные поля (Рис. 2.17). Для открытия формы добавления полей можно воспользоваться сочетанием клавиш Ctrl+A либо щелкнуть правой кнопкой мыши и выбрать пункт Add fields.

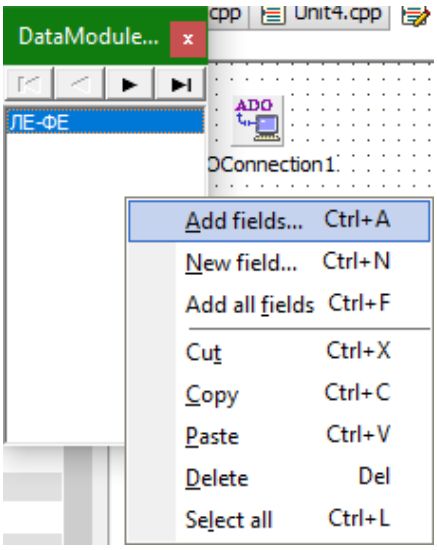

Рис. 2.17. Форма отображения подключенных полей

После выполнения предыдущих действия откроется форма (Рис. 2.18), содержащая список всех доступных полей, добавление которых в компонент ADOTable производится двойным щелчком левой кнопки мыши по требуемому полю, любо путем выбора на предыдущей форме пункта Add all fields, в случае если необходимо добавить все поля, включенные в данную таблицу.

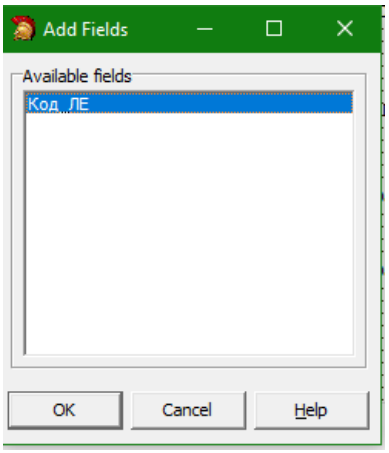

Рис. 2.18 Форма добавления полей в компонент ADOTable

Поскольку некоторые поля, имеющиеся в физической базе данных, не представляют для конечного пользователя никакого интереса и используются

только в качестве аргументов при выполнении фильтрации, будет логично скрыть их от пользователя, выбрав интересующее нас поле во вкладке Structure (Рис. 2.19) и затем установив значение свойства Visible во вкладке Object Inspector как false. Данная операция позволит программе использовать информацию из данного поля, при этом не отображая для пользователя не относящуюся к его запросу информацию.

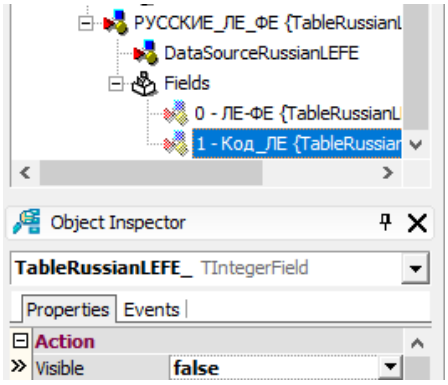

Рис. 2.19. Установка свойств поля

Наконец, для того, чтобы связать компонент ADOTable, расположенный на форме DataModule с компонентом, отображающим информацию из базы данных для пользователя DBGrid, описанным ранее, необходимо добавить на форму компонент DataSource, находящийся во вкладке Data Access (Рис. 2.20) и отвечающий за установление связей между компонентами работы с информацией, поступающей из физической базы данных.

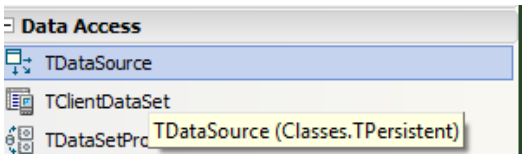

Рис. 2.20 Выбор компонента DataSource

После добавления необходимо указать в свойствах данного компонента из какой именно таблицы ему следует отображать информацию использовав свойство DataSet из вкладки Database (Рис. 2.21).

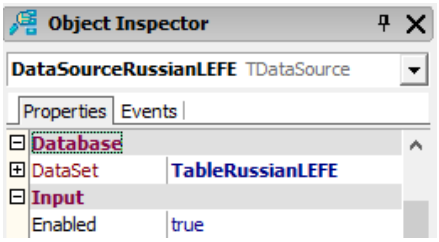

Рис. 2.21 Установка связи с отображаемой таблицей

Поскольку связь с физической базой данных настроена, опишем работу с компонентом DBGrid, расположенным на форме «Поиск» (Рис. 2.6). Для того чтобы пользователь могу увидеть данные, выводимые из подключенной базы данных необходимо установить связь, между данным компонентом и компонентом DataSource, который отвечает за связь между компонентами работы с базой данных внутри программы. Как видно на приложенном изображении (Рис. 2.22), во вкладке Linkage расположено поле DataSource, в котором и необходимо выбрать требуемый компонент. Также, исходя из того, что данная программа предполагает только поиск, без возможности изменения информации, находящейся в базе данных и скрытой от пользователя, необходимо установить свойство поля ReadOnly вкладки Input в true, для того чтобы предотвратить изменение записей внутри базы данных.

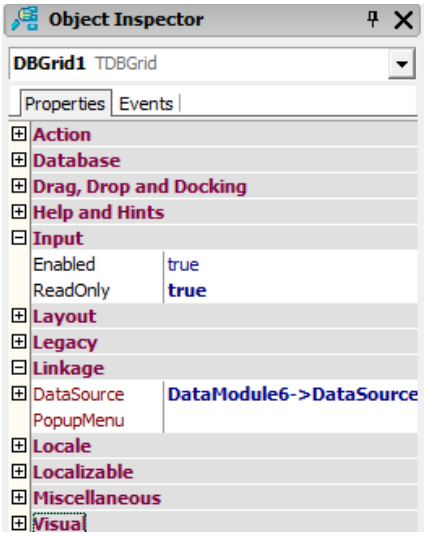

Рис. 2.22. Свойства компонента DBGrid

После того как все связи между компонентами приложения, а также между приложением и физической базой данных были установлены, можно приступать к написанию программного кода.

2.4 Описание отдельных блоков программного кода

После размещения на форме необходимых компонентов необходимо создать для них события, которые, при срабатывании определенного действия, будет выполнять описанные в программном коде действия. Для создания события необходимо выбрать определенное событие из вкладки Events, в случае с кнопками главного меню, было выбрано событие OnClick, вызывающее действие при одиночном щелчке левой клавишей мыши. Для обеспечения перехода с формы главного меню на остальные воспользуемся методом Show, который задает свойству visible значение true, делая невидимую ранее форму видимой, ту форму с которой осуществляется переход можно скрыть при помощи

метода Hide, выполняющего действие обратное методу Show, в случае с формами, это позволит сконцентрировать внимание только на требуемой форме. Для обеспечения возможности взаимодействовать с другими формами, необходимо в файле, содержащем программный код, с помощью директивы #include указать заголовочные файлы тех формы, с которыми планируется взаимодействие.

Для облегчения взаимодействия пользователя с программой, в свойстве Caption компонента Edit, был введен текст, который уточняет для пользователя формат вводимой информации. Также для данного компонента был создан обработчик события OnClick, в теле которого использован метод Clear, таким образом, при нажатии левой кнопкой мыши в поле Edit введенный ранее текст удалится, и пользователь сможет сразу приступить к поиску требуемой лексемы. Для упрощения работы пользователя с большим объемом данных, содержащихся в таблице, было решено использовать динамическую фильтрацию. За реализацию данной функции отвечает блок программного кода, приведенный ниже (Рис. 2.23). Данный код вызывается при выполнении события OnChange, что соответствует концепции динамической фильтрации. Создается строковая переменная text в которую далее поступает введенный в поле Edit набор символов, извлекаемый из него при помощи свойства Text. Условный оператор if проверяет истинность утверждения RadioButtonRu->Checked, и используется для того, чтобы использовать подходящую таблицу. В случае если данное утверждение окажется ложным, то будет проверяться истинность утверждения RadioButtonEn->Checked, которое имеет схожий с приведенным ниже программный код и отличается только целевой для фильтрации таблицей. В случае истинности первого утверждения, выполняется проверка соответствия переменной text пустой строке, в случае если значение данной переменной является пустым, свойству Filtered компонента TableRussianLEFE присваивается значение false, что отключает любую фильтрацию данной таблицы и обеспечивает вывод полной таблицы. В случае если в поле Edit был введен символ, свойству Filtered

присваивается значение true, после чего свойству Filter, присваивается строка содержащая имя колонки, в данном случае ЛЕ-ФЕ, оператор like, реализующий поиск по заданному шаблону, и переменная text помещенная в функцию QuotedStr, отвечающую за правильное взаимодействие с базами данных и помещающее значение переменной text в одинарные кавычки. Данная функция необходима в виду определенных требований к оформлению запросов, поступающих в свойство Filter. Астериск в данном случае используется для описания поискового запроса, допускающего любое количество символов после введенного пользователем. Благодаря этому у пользователя появляется возможность не вводить искомую лексему полностью, а выбрать её из списка при достаточном сужении области поиска.

```
void __fastcall TForm2::Edit1Change(TObject *Sender)
{

        String text = "";
   text = Edit1->Text;

if (RadioButtonRu->Checked) {

if (text == "") {
    DataModule6->TableRussianLEFE->Filtered = false;
}
else{
DataModule6->TableRussianLEFE->Filtered = true;
DataModule6->TableRussianLEFE->Filter = "(ЛЕ-ФЕ like " + QuotedStr(text+'*') + ")";
}

}
```

Рис. 2.23 Программный код, отвечающий за функцию динамической фильтрации лексем

Наличие только одной формы поиска накладывает на разработчика необходимость предусмотреть смену целевой таблицы, а значит и смену соответствующего свойства компонента DBGrid. Реализация данного функционала рассмотрена на приведенном ниже программном коде (Рис. 2.24).

```
void __fastcall TForm2::RadioButtonRuClick(TObject *Sender)
{
DBGrid1->DataSource=DataModule6->DataSourceRussianLEFE;
DataModule6->DataSourceRussianLEFE->DataSet->Active=true;
}
//
```

Рис. 2.24. Реализация переключения целевой таблицы

Как видно выше, при срабатывании события OnClick, на компоненте RadioButtonRu, свойству DataSource компонента DBGrid присваивается название необходимой таблицы, после чего её свойство DataSet меняется на Active, что обеспечивает вывод информации из данной таблицы в требуемый компонент. В связи с тем, что реализация данной функции для таблицы, содержащей лексические единицы английского языка аналогична данной за исключением названия таблицы, программный код, описывающий данный алгоритм не приводится.

Далее рассмотрим алгоритм реализующий фильтрацию данных на основе указанных пользователем параметров и их последующий вывод в поле RichEdit.

```
void __fastcall TForm2::DBGrid1CellClick(TColumn *Column)
{

RichEdit1->Clear();



if (RadioButtonRu->Checked) {

//Запоминаем код лексемы
LECode = DataModule6->TableRussianLEFE_->AsString;
```

Рис. 2.25. Получение кода лексемы

Как видно из приведенного выше фрагмента программного кода (Рис. 2.25), процесс вывода на экран требуемой лексемы происходит при условии

выполнения события OnCellClick, компонента DBGrid. Использование метода Clear, делает просмотр информации по нескольким лексемам более простым, поскольку очищает компонент RichEdit от результатов, полученных при выборе предыдущей лексемы. После чего выполняется проверка истинности высказывания RadioButtonRu->Checked, от результатов которой зависит набор таблиц используемых для поиска информации. Если утверждение истинно, то объявленной в заголовочном файле переменной LECode присваивается значение поля, скрытого от пользователя и содержащего код лексемы. Далее, как видно ниже (Рис. 2.26), условным оператором if выполняется проверка истинности утверждения LekBox->Checked, по умолчанию имеющего значение true. После чего выполняется фильтрация таблицы, содержащей результаты лексического анализа, в соответствии с полученным ранее кодом лексемы, выполняющей в таблице роль ключевого поля. Сам процесс фильтрации, как и программный код отвечающий за его реализацию был описан ранее и является модификацией алгоритма реализующего динамическую фильтрацию лексем при их вводе.

```
if (LekBox->Checked) {
  DataModule6->TableRussianLEK->Filtered = true;
  DataModule6->TableRussianLEK->Filter = "(Код_ЛЕ like " + QuotedStr(LECode) + ")";
  RichEdit1->SelAttributes->Color = clHotLight;
  RichEdit1->SelAttributes->Size = 16;
  RichEdit1->Lines->Add("Лексический анализ");
while (!DataModule6->TableRussianLEK->Eof)
    {
      RichEdit1->Lines->Add
      (DataModule6->TableRussianLEK->Fields->FieldByName("Результат")->AsString);
      DataModule6->TableRussianLEK->Next();
    }
}
```

Рис. 2.26. Фильтрация и вывод результата лексического анализа

При помощи свойства SelAttributes и его свойств Color и Size Выводимая программой информация приобрела более удобочитаемый формат, поскольку различные виды анализов отделены от остального текста путем увеличения

размера шрифта и изменения цвета текста. В связи с тем, что некоторые лексические единицы имеют несколько значений, возникла необходимость в использовании цикла while и свойства Eof компонента ADOTable. Данное свойство возвращает значение true когда достигает конца списка, описанный внутри цикла алгорит отвечает за вывод результатов лексического анализа. При помощи функции Add компонента RichEdit, на него выводятся значения полей из колонки «Результат», в формате AnsiString, после чего, при помощи функции Next осуществляется переход к следующей записи в таблице. Благодаря использованию функции FieldByName, и указанием в ней имени колонки, выводятся только значения поля «Результат».

Поскольку программный код, отвечающий за фильтрацию и вывод результатов других видов анализа, аналогичен данному за исключением названия таблиц, участвующих в выборке, его детальный разбор не приводится. После завершения процесса разработки можно приступе к апробации готового программного продукта.

2.5 Апробация

Тестирование программы – это этап, на котором проверяется, как ведет себя программа на как можно большем количестве входных наборов данных, в том числе и заведомо неверных.

К основным принципам организации тестирования можно отнести:

- точное и полное описание всех ожидаемых результатов работы программы, позволяет максимально быстро и точно выяснить наличие или отсутствие в ней всевозможных ошибок и недочетов;

- желательно избегать тестирования программы ее же автором, в силу того, что обнаружение недостатков в собственной деятельности расходится с основами человеческой психологии (несмотря на это, отладка программы эффективнее всего выполняется именно автором программы, так как он в большей степени ознакомлен с ее внутренним устройством и принципами работы);
- доскональное изучение результатов каждого теста, позволяет не пропустить малозаметную, но в то же время критически важную, на поверхностный взгляд ошибку в программе;
- необходимо тщательно подбирать и тестировать не только всевозможные правильные (предусмотренные) входные данные, но и неправильные (непредусмотренные);

Учитывая все вышеперечисленные принципы, можно приступить к тестированию программы. На приведенных ниже изображениях (Рис. 2.27 а, б, в) показаны примеры использования программы.

Рис. 2.27 а. Главное меню

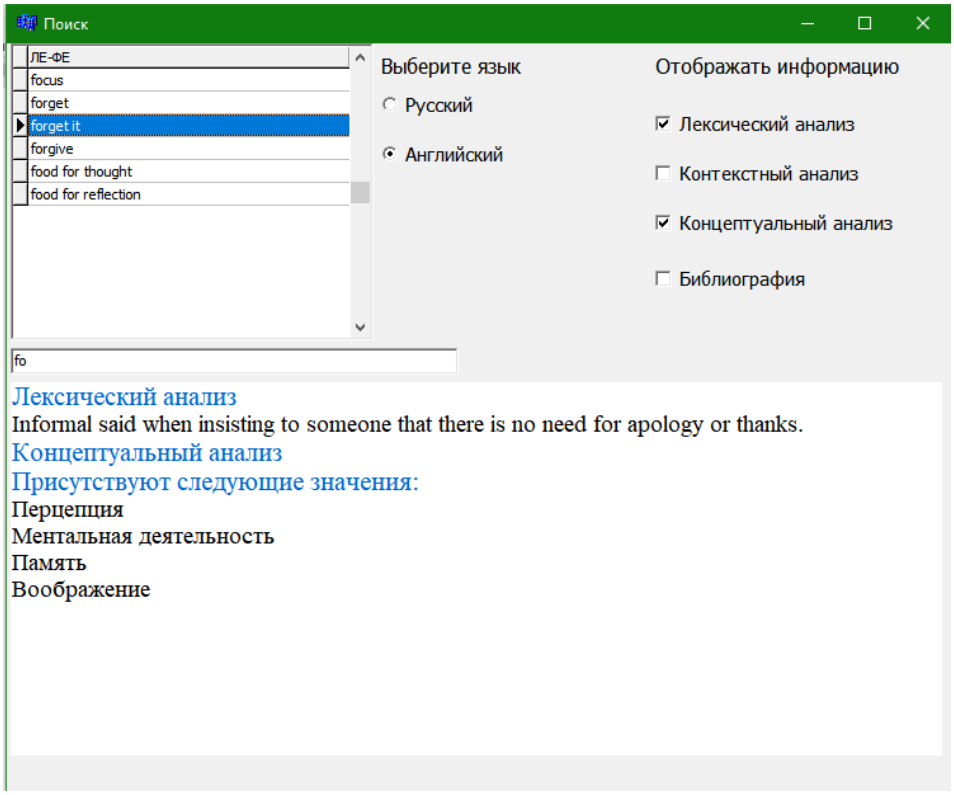

Рис. 2.27 б. Окно поиска лексем

На приведенном выше изображении видно правильное функционирование фильтров, форматирования текста и динамического поиска.

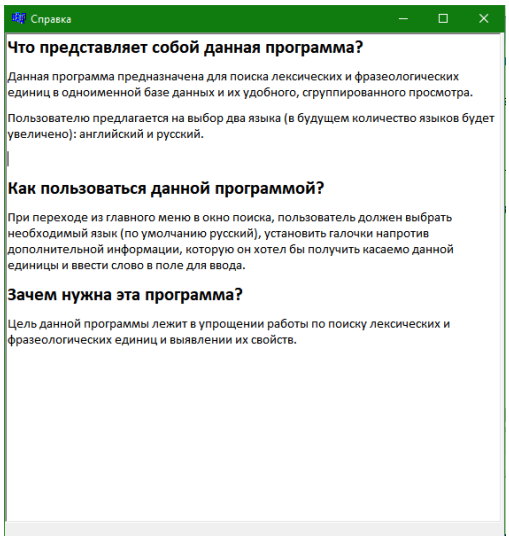

Рис. 2.27 в. Окно «Справка»

Справочная информация касаемо разработанной программы выводится верно.

В ходе тестирования программного продукта был выделен такой недостаток как невозможность переключения между формами, исходя из этого было принято решение добавить компонент MainMenu с помощью которого создается закрепленное в верху формы меню. Доступ к форме добавления кнопок открывается при двойном щелчке по компоненту (Рис. 2.28), а изменение названия добавленной кнопки и действия для её активации устанавливается при помощи описанных ранее свойств Caption и событий OnClick.

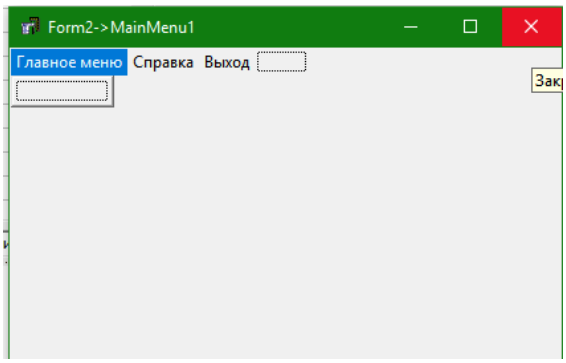

Рис. 2.28. Форма компонента MainMenu

В результате подобных действий, данный недочет, как видно ниже (Рис. 2.29), был исправлен, что повысило удобство использования разработанного приложения.

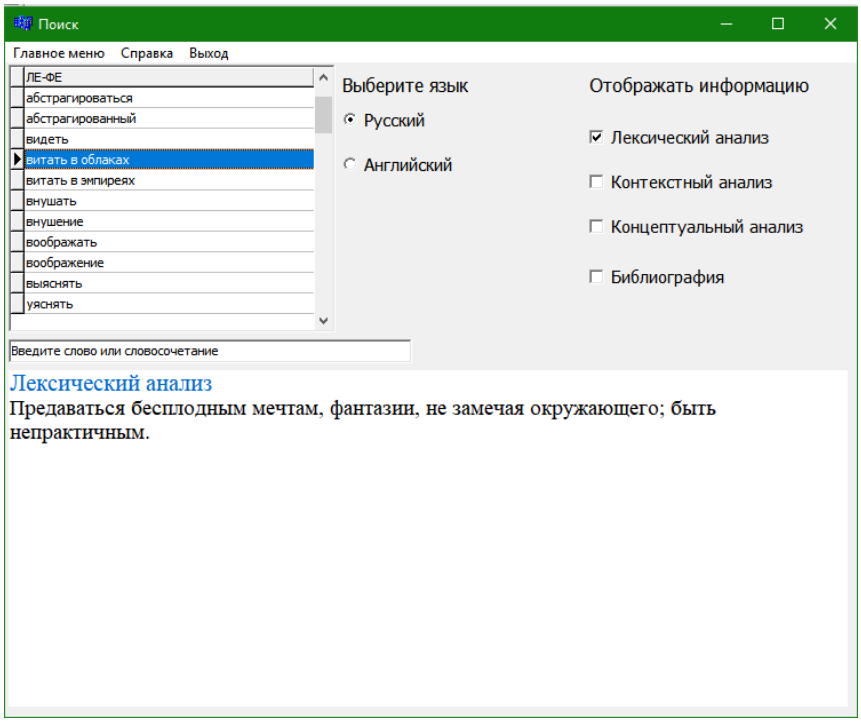

Рис. 2.29. Форма с добавленным на неё компонентом MainMenu

Также, использовав свойство Icon стандартный значок приложения был заменен на созданный заранее в графическом редакторе. При изменении свойства Icon появляется форма (Рис. 2.30) в которую загружается предварительно созданное изображение.

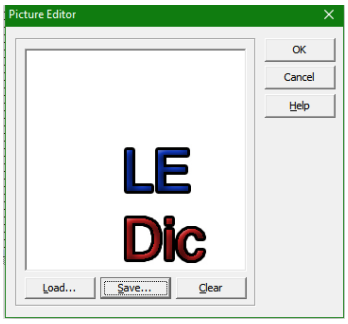

Рис. 2.30. Форма добавления изображения

После внесения всех изменений, с помощью бесплатного программного продукта InnoSetup, разработанное приложение было упаковано в установочный

файл, для повышения удобства его установки. Поскольку описание процесса создания установочного файла не представляет значимости для дипломной работы, его было решено опустить.

Вывод к Главе 2

Работа с выбранной в качестве объекта данного исследования физической базой данных лексических единиц, номинирующих процесс познания в современном английском и русском языках, позволила получить новые сведения как о семантике и особенностях функционирования слов в предложении, так и о процессе разработки программных продуктов.

В результате рассмотрения лексических единиц, было выявлено, что для формирования необходимого для создания базы данных тезауруса, требуется глубокий анализ отбираемых лексем. И если при первой итерации и создании первичной базы данных достаточно лишь отбора по критериям номинации когниции, то при последующем отборе необходимо учитывать соответствие отобранных лексических единиц таким концептуальным составляющим как «внимание», «перцепция» и так далее.

Благодаря использованию графического редактора Microsoft Visio 2010 стало возможным создание диаграмм потоков данных, облегчающих задачу по созданию программного продукта.

Использование Borland C++ Builder в свою очередь обеспечило широкий выбор компонентов для создания пользовательского интерфейса, а благодаря технологии Drag&Drop процесс разработки был значительно ускорен. Созданная в итоге данной работы программа полностью отвечает предъявленным ей в начале процесса разработки требованиям, однако, в то же время, очевидна возможность улучшения программы, путем как увеличения набора лексем для различных языков, так и ввода дополнительных возможностей, позволяющих пользователю данного программного продукта получить больше информации по искомой лексеме.

ЗАКЛЮЧЕНИЕ

Реализация данного исследования представляет собой попытку создания удобного в использовании программного продукта, позволяющего пользователю получать необходимую информацию о лексеме, не тратя при этом время на создание запросов и сопоставление значений в различных таблицах, используя такие СУБД как Microsoft Access.

Полученные в ходе проведения исследования результаты позволили выявить основные методы и этапы в проектировке и создании программного продукта. Полученные результаты способствовали созданию виртуального словаря на основе баз данных.

Главная цель работы заключается в разработке и создании программного продукта, при помощи которого пользователь будет иметь возможность получить интересующую его информацию по определенной лексической или фразеологической единице в соответствии с заданными параметрами фильтрации, не задумываясь при этом о необходимости формирования запроса для выборки информации из отдельных таблиц. Разрабатываемое приложение берет на себя всю работу по сопоставлению данных и их выводу, оставляя пользователю возможность формировать объем выводимой информации путем использования соответствующих фильтров. Достижение поставленной цели реализуется путем выполнения ряда действий: анализа предметной области, создания диаграмм потоков данных, разработки алгоритма, пользовательского интерфейса и непосредственно написания программного кода, реализующего задуманный функционал программы.

Таким образом, можно заключить, что задачи, поставленные в данной работе, решены в полном объеме, что говорит о достижении цели.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Apple iOS H. I. G. Apple iOS Human Interface Guidelines. – 2012.
2. Bowden J. R., Rusnock C. F. Evaluation of human machine interface design factors on situation awareness and task performance //Proceedings of the Human Factors and Ergonomics Society Annual Meeting. – Sage CA: Los Angeles, CA : Sage Publications, 2015. – Т. 59. – №. 1. – С. 1361-1365.
3. Malik D. S. C++ Programming: Program Design Including Data Structures. – Nelson Education, 2014.
4. Telles M. A., Dimtemann J. Borland C++ Builder. – International Thomson Publishing, 1997.
5. Березин Б. И., Березин С. Б. Начальный курс С и С++. – Directmedia, 2013.
6. Вальвачев А. Н. и др. Объектно-ориентированное программирование на языках Delphi и C++: учебное пособие для студентов [Электронный ресурс]. – 2016.
7. Вендров А. М. CASE-технологии. Современные методы и средства проектирования информационных систем //М.: Финансы и статистика,-1998.-176 с. – 1998.
8. Гамма Э. и др. Приемы объектно-ориентированного проектирования. – " Издательский дом"" Питер""", 2013.
9. Гируцкий, А.А. Введение в языкознание: учеб. пособие/ А.А. Гируцкий – Минск: Вышэйшая школа, 2016. – 238 с. : ил.
10. Джозеф М. Самоучитель Microsoft Visual Studio 2010. – БХВ-Петербург, 2011.
11. Дубичинский В.В. Лексикография русского языка: учеб. пособие / Дубичинский В.В. – М.: Наука: Флинта, 2008. – 432 с.

12. Жвакина А. В. Разработка Windows-приложений в среде визуального программирования: пособие. – 2016.
13. Илюшечкин В. Основы использования и проектирования баз данных. – Litres, 2017.
14. Карпов В. В., Карпов А. В. Особенности применения современных методов разработки программного обеспечения защищенных автоматизированных систем //Программные продукты и системы. – 2016. – №. 1 (113).
15. Лафоре Р. Объектно-ориентированное программирование в C++:[пер. с англ.]. – Издательский дом" Питер", 2013.
16. Липпман С., Лажойе Ж. ЯЗЫК ПРОГРАММИРОВАНИЯ C++. ПОЛНОЕ РУКОВОДСТВО. УЧЕБНОЕ ПОСОБИЕ. – ДМК Пресс ББК: 32.973 УДК: 681.3, 2009.
17. Маслобоев А. В., Быстров В. В., Ломов П. А. Моделирование бизнес-процессов. – 2014.
18. Мельникова Р. В. Проектирование пользовательского интерфейса //Восточно-Европейский журнал передовых технологий. – 2010. – Т. 6. – №. 8 (48).
19. Николаев Д. М., Николаев В. С. Моделирование бизнес-процессов. – 2012.
20. Орлов С. А., Цилькер Б. Я. Технологии разработки программного обеспечения. Учебник для вузов. 4-е издание. Стандарт третьего поколения. – Издательский дом" Питер", 2012.
21. Послед Б. С. Borland C++ Builder 6. Разработка приложений баз данных. – DiaSoft, 2003.
22. Репин В., Елиферов В. Процессный подход к управлению. Моделирование бизнес-процессов. – Манн, Иванов и Фербер, 2004.
23. Солоницын Ю. А. Microsoft Visio 2007. Создание деловой графики. – Издательский дом" Питер", 2008.

24. Стернин И. А. Лексическое значение слова в речи. – Directmedia, 2015.
25. Стернин И. А. Проблемы анализа структуры значения слова. – Directmedia, 2015.
26. Страуструп Б. Язык программирования C++. Специальное издание //М.: Бином-Пресс. – 2008.
27. Телия В. Русская фразеология. Семантический, прагматический и лингвокультурологический аспекты. – Litres, 2017.
28. Федоренко Ю. Алгоритмы и программы на C++ Builder. – Litres, 2017.
29. Фирсова Н. В. Инструментальные средства моделирования бизнес-процессов и оценка их применения для целей реинжиниринга //Вестник Санкт-Петербургского университета. Серия 8. Менеджмент. – 2005. – №.4.
30. Шерегов Н. А., Полушин А. А. Моделирование бизнес-процессов.