КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

Институт физики

Кафедра радиофизики

П. А. Корчагин, И. П. Корчагин

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Часть 3. Методические указания к практическим занятиям по языку программирования 1С: Предприятие 8.3

Казань – 2026

УДК 004.43
ББК 32.973.26-018.1
К70

Принято на заседании УМК ИФ

Протокол № 5 от 22 января 2026 года

Рецензент

кандидат технических наук,

доцент кафедры радиоастрономии КФУ

Е. Ю. Зыков

Корчагин П. А., Корчагин И. П.

К70 Языки программирования. Часть 3. Методические указания к практическим занятиям по языку программирования 1С:Предприятие 8.3 / П. А. Корчагин, И. П. Корчагин. – Казань: Казан. ун-т, 2026. – 76 с.

Методическое пособие представляет собой систематизированный курс по основам программирования в среде «1С:Предприятие 8.3». В пособии подробно рассматриваются базовые конструкции встроенного языка, работа с примитивными и ссылочными типами данных, универсальными коллекциями, а также специфика разработки прикладных решений. Особое внимание уделено практическим аспектам: работе со справочниками, документами, регистрами и запросами. Пособие содержит большое количество примеров кода, детальные пояснения и рекомендации по написанию чистого и эффективного кода. Для каждого раздела разработан набор вариативных практических заданий разного уровня сложности, что позволяет организовать эффективный учебный процесс. Предназначено для студентов направлений подготовки «Информационная безопасность», «Прикладная информатика», а также для начинающих разработчиков 1С.

Оглавление

ВВЕДЕНИЕ 6
2. ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ 7
2.1. Введение в платформу 1С:Предприятие 8.3 7
Пример 1. Простейшая процедура и диагностическое сообщение 8
2.2. Режимы работы и организация разработки. Отладка и журнал 9
регистрации 9
Пример 2. Диагностические сообщения и фиксация контекста ошибки 10
2.3. Синтаксис встроенного языка и структура модулей 10
Пример 3. Процедура и функция. Документирование ожиданий 12
2.4. Типы данных, значения и преобразования 12
Пример 4. Проверка заполненности и пустых ссылок 13
2.5. Коллекции и табличные структуры данных 14
Пример 5. Индексация таблицы значений и быстрый поиск строк 16
2.6. Операторы, выражения и форматирование 17
Пример 6. Форматирование даты и суммы и нормализация строк 18
2.7. Условные конструкции, проверки и исключения 18
Пример 7. Проверка заполненности в событии ПередЗаписью 19
2.8. Циклы, обход выборок и типовые алгоритмы 20
Пример 8. Агрегирование и раннее завершение поиска 21
2.9. Процедуры, функции, общие модули 22
Пример 9. Разделение: клиентская команда и серверная выборка 23
2.10. Метаданные и объектная модель: справочники и документы 24
Пример 10. Программное создание документа и заполнение табличной части 25
2.11. Регистры и проведение: движения, наборы записей, итоги 26
Пример 11. Формирование приходных движений по регистру 28
накопления 28
2.12. Запросы 29

Пример 12. Левое соединение номенклатуры и остатков 30
2.13. Отчёты и СКД 31
Пример 13. Концептуальная схема отчёта «Остатки на дату» 32
2.14. Управляемые формы 32
Пример 14. Обработчик изменения реквизита и серверный перерасчёт 33
2.15. Права доступа, роли и безопасная разработка 34
Пример 15. Проверка роли перед выполнением операции 35
3. ПРАКТИКУМ: СОЗДАНИЕ УЧЕБНОЙ КОНФИГУРАЦИИ «УЧЁТ СКЛАДА» 36
3.1. Создание информационной базы и подготовка конфигурации 36
3.2. Проектирование справочников 37
3.3. Проектирование документов и табличных частей 37
3.4. Проектирование регистра накопления и аналитических разрезов 38
3.5. Проведение документов и формирование движений 38
3.6. Отчёт «Остатки товаров на дату» 39
3.7. Роли и тестирование под разными пользователями 39
ПРАКТИЧЕСКИЕ ЗАДАНИЯ 40
Задание 1: Переменные, типы данных и коллекции 40
Задание 2: Операторы, выражения и форматирование 41
Задание 3: Условные конструкции и исключения 42
Задание 4: Циклы и обработка наборов данных 43
Задание 5: Процедуры и функции 44
Задание 6: Работа со справочниками (объектный интерфейс) 44
Задание 7: Документы и табличные части 45
Задание 8: Регистры и проведение документов 46
Задание 9: Запросы 47
Задание 10: Формы, команды и клиент-серверное взаимодействие 48
ТИПОВЫЕ ПРИМЕРЫ КОДА 49
Пример А. Создание элемента справочника с проверкой уникальности реквизита 49

Пример Б. Автозаполнение табличной части в форме документа 50
Пример В. Контроль остатков при списании (запрос к виртуальной таблице) ... 51
Пример Г. Универсальная проверка заполненности документа 53
РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЗАДАНИЙ 55
СПИСОК ЛИТЕРАТУРЫ 56
ПРИЛОЖЕНИЕ 1. ГЛОССАРИЙ ОСНОВНЫХ ТЕРМИНОВ 58
ПРИЛОЖЕНИЕ 2. ШПАРГАЛКА ПО СИНТАКСИСУ И ТИПОВЫМ ПАТТЕРНАМ 61
Шаблон 1. Попытка–Исключение 61
Шаблон 2. Если–ИначеЕсли–Иначе 61
Шаблон 3. Для каждого по таблице значений 61
Шаблон 4. Параметризованный запрос 62
Шаблон 5. Проверка заполненности 62
ПРИЛОЖЕНИЕ 3. ТИПОВЫЕ ОШИБКИ И ПРИЁМЫ ОТЛАДКИ 63
ПРИЛОЖЕНИЕ 4. КРАТКИЙ СПРАВОЧНИК ВСТРОЕННЫХ ФУНКЦИЙ И ПРИЁМОВ 65
ПРИЛОЖЕНИЕ 5. КОНТРОЛЬНЫЕ ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ 68
ПРИЛОЖЕНИЕ 6. ОБРАЗЕЦ ВЫПОЛНЕНИЯ ЗАДАНИЯ: ДОКУМЕНТ «ПОСТУПЛЕНИЕ ТОВАРА» 70
А. Структура метаданных 70
Б. Модуль объекта документа: проверки перед записью 71
В. Модуль объекта документа: проведение и движения 73
Г. Модуль формы 73
Д. Проверка результата через отчёт по остаткам 74

ВВЕДЕНИЕ

Платформа «1С:Предприятие 8» является распространённой технологической основой для построения информационных систем учёта, управления и аналитики в организациях. В типовых и отраслевых конфигурациях платформа предоставляет готовые модели предметных областей, а прикладной разработчик адаптирует эти модели под конкретные регламенты, бизнес-процессы и интеграционные требования. В результате компетенции по программированию на встроенном языке и по проектированию метаданных становятся важной частью профессиональной подготовки специалиста по корпоративным информационным системам.

Актуальность изучения платформы обусловлена тем, что прикладная разработка в 1С находится на стыке программирования и моделирования данных. Разработчик не только пишет алгоритмы, но и проектирует структуру данных, определяет правила жизненного цикла объектов, строит пользовательский интерфейс, настраивает права доступа и обеспечивает корректность вычислений на больших объёмах данных. Практика показывает, что успешное освоение 1С требует не «заучивания приёмов», а понимания принципов платформы и инженерной дисциплины: проверки данных, управления контекстом выполнения, декомпозиции логики и тестирования.

Настоящее пособие ориентировано на практические занятия и построено по принципу последовательного усложнения. Сначала рассматриваются основные элементы встроенного языка и работа с данными, затем – объектная модель платформы (справочники, документы, регистры), язык запросов и формы управляемого приложения. После теоретической части приведён практикум по созданию учебной конфигурации, а также набор вариативных заданий для самостоятельной работы.

В качестве учебного стенда предполагается наличие установленной платформы «1С:Предприятие 8.3» и создание учебной файловой информационной базы. Все примеры формулируются в нейтральной предметной области (склад и

движение товаров), что позволяет сосредоточиться на механике разработки. При выполнении заданий допускается расширение конфигурации: добавление реквизитов, документов, регистров и форм, если это улучшает решение и соблюдает исходные требования задачи.

2. ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

2.1. Введение в платформу 1С:Предприятие 8.3

Платформа «1С:Предприятие 8.3» разделяет технологическую и прикладную части. Технологическая часть представляет собой исполняемую среду, обеспечивающую хранение данных, выполнение кода, построение интерфейса, работу с транзакциями, кэширование, механизм блокировок и интеграционные функции. Прикладная часть задаётся конфигурацией, в которой описываются метаданные: структура данных, формы, команды, роли и алгоритмы.

В терминах платформы различают «метаданные» и «данные». Метаданные определяют, какие прикладные объекты существуют и как они устроены: какие реквизиты у справочника, какие табличные части у документа, какие измерения и ресурсы у регистра. Данные – это конкретные записи в информационной базе: элементы справочников, документы, движения регистров. Платформа интерпретирует метаданные и обеспечивает работу с данными так, чтобы разработчик мог использовать единую объектную модель, не погружаясь в детали хранения.

Архитектурно платформа поддерживает файловый и клиент-серверный варианты. В файловом варианте данные хранятся в одном файле базы данных, а клиент и «сервер» фактически совпадают. В клиент-серверном варианте данные хранятся в СУБД, а сервер 1С выполняет бизнес-логику, обслуживая множество клиентов. В учебных целях чаще используется файловая база, однако многие принципы (клиент-серверный контекст, запреты на прямой доступ к данным с клиента,

транзакции) актуальны и в файловом варианте, поскольку логика управляемого приложения остаётся клиент-серверной.

Существенной особенностью разработки в 1С является событийный характер выполнения кода. Значительная часть процедур выполняется в ответ на события интерфейса (изменение реквизита формы, выполнение команды) или события жизненного цикла объекта (перед записью, при проведении, при удалении). Поэтому разработчик должен знать точки расширения: где ставить проверки, где рассчитывать итоги, где формировать движения регистров, и какие последствия это имеет для пользователей и отчётности.

Следует также различать «объектный» и «запросный» способы работы с данными. Объектный способ удобен для создания и изменения объектов: создать документ, заполнить, записать, провести. Запросный способ необходим для выборок, агрегирования и аналитики: быстро выбрать остатки, обороты, сгруппировать по складам и номенклатуре. На практике оба подхода сочетаются в рамках одной задачи.

Пример 1. Простейшая процедура и диагностическое сообщение

Для первичной проверки рабочей среды достаточно создать экспортную процедуру в общем модуле и вывести сообщение в окно сообщений режима «Предприятие».

```
Процедура Приветствие() Экспорт
    Сообщить("Среда «1С:Предприятие 8.3» готова. Можно начинать практику.");
КонецПроцедуры
```

2.2. Режимы работы и организация разработки. Отладка и журнал регистрации

Платформа предоставляет два основных режима: «Конфигуратор» и «Предприятие». В режиме «Конфигуратор» выполняются проектирование метаданных, создание и изменение объектов конфигурации, настройка форм, командного интерфейса, прав доступа, а также написание и отладка программного кода. В режиме «Предприятие» происходит эксплуатация решения: пользователи вводят данные, проводят документы, формируют отчёты и выполняют регламентные операции.

Организация разработки в учебных задачах строится вокруг цикла «изменил – обновил – протестировал». Изменения в конфигурации применяются к информационной базе через обновление. После обновления рекомендуется выполнять короткий регрессионный тест: открыть ключевые формы, создать тестовый элемент справочника, записать тестовый документ, выполнить одну контрольную выборку. Подобный режим работы дисциплинирует и предотвращает накопление ошибок.

Для анализа ошибок платформа предоставляет средства пошаговой отладки, точки останова, просмотр значений переменных и выражений, а также инструменты профилирования. На начальном этапе особенно полезны точки останова в ключевых событиях: ПередЗаписью, ПриЗаписи, ОбработкаПроведения, ПриОткрытии. При отладке управляемых форм следует учитывать, что часть кода выполняется на клиенте, а часть – на сервере, поэтому важно понимать, в каком контексте сработала точка останова.

Журнал регистрации (а также сообщения диагностики) служит средством фиксировать значимые события выполнения: ошибки, предупреждения, вызовы интеграции, изменения данных. В учебной практике рекомендуется развивать привычку формулировать сообщения так, чтобы они отвечали на вопросы «что произошло», «над каким объектом» и «какое значение параметров было критично». Такая

привычка облегчает дальнейшее сопровождение решений в промышленной эксплуатации.

Типовой набор ошибок начинающего разработчика включает: обращение к незаполненному реквизиту; попытку выполнять запросы из клиентской процедуры без серверного вызова; неверное понимание пустой ссылки; смешивание кода формы и кода объекта; попытку изменять данные в обработчиках, где это запрещено логикой жизненного цикла. Эти ошибки не являются «частными», они отражают необходимость понять архитектуру платформы и дисциплину разработки.

Пример 2. Диагностические сообщения и фиксация контекста ошибки

В примере показано, как в блоке исключения вывести краткое сообщение пользователю и одновременно сформировать более детальное диагностическое сообщение, пригодное для анализа в ходе отладки.

```
Процедура ВыполнитьОперацию() Экспорт
    Попытка
        // Здесь размещается логика, потенциально ведущая к ошибке
        ВызватьИсключение "Имитация ошибки для демонстрации";
    Исключение
        Сообщить("Операция не выполнена: " + ОписаниеОшибки());
        // При необходимости здесь может быть запись в журнал регистрации
    КонецПопытки;
КонецПроцедуры
```

2.3. Синтаксис встроенного языка и структура модулей

Код в 1С размещается в модулях. Модуль представляет собой текстовый объект, содержащий процедуры, функции, переменные модуля и директивы компиляции. Модули бывают разных типов: модуль объекта (логика конкретного элемента справочника или документа), модуль менеджера (логика «класса», управляющего

объектами), модуль формы (логика интерфейса), общий модуль (общая библиотека процедур и функций), модуль сеанса и др. Выбор места размещения кода – инженерное решение, влияющее на повторное использование и корректность контекста выполнения.

Процедура объявляется ключевым словом «Процедура», функция – «Функция». Если требуется доступ из других модулей, применяется ключевое слово «Экспорт». Имена процедур и функций, как правило, отражают действие или результат: «РассчитатьСумму», «ПроверитьЗаполнение», «ПолучитьОстатки». Встроенный язык не требует явного объявления типов, однако в сложных алгоритмах целесообразно документировать ожидания по типам в комментариях и проверках.

Блоки кода завершаются явными конструкциями: «КонецЕсли», «КонецЦикла», «КонецПопытки». Это повышает читаемость и снижает вероятность ошибок при вложенности. Комментарии однострочные, начинаются с «//». Для многострочных пояснений обычно применяют несколько однострочных комментариев. В промышленном коде комментарии должны объяснять не очевидный синтаксис, а мотив: почему выбран тот или иной способ вычисления, какие ограничения данных учитываются.

Отдельное место занимает понятие «контекста выполнения». В управляемом приложении выделяют клиентский и серверный контекст. Директивы «&НаКлиенте», «&НаСервере», «&НаСервереБезКонтекста» задают, где компилируется и выполняется метод. Это принципиально влияет на доступность объектов: например, обращение к базе данных и выполнение запросов относится к серверной части, а взаимодействие с элементами формы – к клиентской.

Для стандартизации кода рекомендуется придерживаться единых соглашений по отступам, именованию, размещению проверок и обработке ошибок. Даже в учебных работах это повышает читаемость и облегчает проверку решений.

Пример 3. Процедура и функция

Пример демонстрирует разницу между процедурой и функцией и иллюстрирует простой приём документирования ожиданий по параметрам через проверки и сообщения об ошибках.

```
Функция РассчитатьНДС(Сумма, Ставка) Экспорт
    // Ожидается, что Сумма и Ставка – числа, Ставка задана в процентах (например, 20)
    Если Сумма = Неопределено Или Ставка = Неопределено Тогда
        ВызватьИсключение "Не заданы параметры расчёта НДС";
    КонецЕсли;

    Возврат Окр(Сумма * Ставка / 100, 2);
КонецФункции

Процедура ПоказатьРасчётНДС(Сумма, Ставка) Экспорт
    НДС = РассчитатьНДС(Сумма, Ставка);
    Сообщить("НДС: " + Формат(НДС, "ЧГ=15.2"));
КонецПроцедуры
```

2.4. Типы данных, значения и преобразования

Встроенный язык является динамически типизированным: переменная хранит значение, а тип определяется самим значением. Это упрощает многие операции, однако требует внимательности при работе с данными, которые поступают из пользовательского ввода, из реквизитов формы и из внешних источников.

К базовым типам относятся Число, Строка, Дата и Булево. Тип Число может представлять целые и дробные значения, при этом точность зависит от контекста и настроек форматов. Тип Дата хранит дату и время. Булево представлено

значениями Истина и Ложь. Строка – последовательность символов, используемая для текстовых данных и часто для кодов.

Значение «Неопределено» означает отсутствие значения. Оно используется и в переменных, и в реквизитах формы, и как результат некоторых операций. При работе с прикладными объектами существует также концепция пустой ссылки: это корректное значение ссылочного типа, которое обозначает «не выбран объект». Пустая ссылка отличается от Неопределено тем, что имеет конкретный тип, но не указывает на существующий объект.

Преобразования типов выполняются встроенными функциями: Число(), Строка(), Дата(). На практике преобразование из строки в число является источником ошибок из-за форматов и разделителей. Поэтому предпочтительно ограничивать преобразования пользовательским вводом через поля формы с типизацией и использовать Формат при выводе.

Определение типа значения выполняется функцией ТипЗнч, а проверка заполненности часто выполняется функцией ЗначениеЗаполнено. Для ссылок на объекты полезны методы получения пустой ссылки: например, Справочники.Номенклатура.ПустаяСсылка(). Для строк применимы функции СокрЛП и ПустаяСтрока. Для дат важно учитывать часовые пояса и представление времени в отчётах и обменах.

В промышленной практике значимым является также понятие сериализации: при передаче значения между клиентом и сервером оно должно быть сериализуемым. В качестве универсального контейнера обычно используют таблицы значений и структуры, поскольку они корректно передаются и сохраняют свою структуру.

Пример 4. Проверка заполненности и пустых ссылок

Пример показывает различение Неопределено и пустой ссылки, а также применение ЗначениеЗаполнено при валидации входных параметров.

```
Процедура ПроверитьПараметры(КонтрагентСсылка, Комментарий) Экспорт

    Если КонтрагентСсылка = Неопределено Тогда
        ВызватьИсключение "Ссылка не передана (Неопределено)";
    КонецЕсли;

    Если КонтрагентСсылка = Справочники.Контрагенты.ПустаяСсылка() Тогда
        ВызватьИсключение "Контрагент не выбран (пустая ссылка)";
    КонецЕсли;

    Если Не ЗначениеЗаполнено(Комментарий) Тогда
        Сообщить("Комментарий не заполнен, будет использовано значение по умолчанию");
    КонецЕсли;

КонецПроцедуры
```

2.5. Коллекции и табличные структуры данных

Универсальные коллекции являются основным инструментом промежуточной обработки данных в 1С. Они используются при подготовке данных для форм и отчётов, при формировании пакетов обмена, при агрегации результатов запросов и при построении алгоритмов проведения. На практике разработчик постоянно «перекладывает» данные из объектов прикладного решения в универсальные коллекции и обратно.

Массив представляет собой упорядоченную последовательность значений, индексируемую с нуля. Он удобен для простых списков, когда требуется сохранять порядок. Для добавления и удаления используются методы Добавить, Вставить, Удалить. Массив может содержать значения разных типов, однако в инженерной практике предпочтительно поддерживать однородность ради прогнозируемости.

Структура хранит пары «имя–значение» и часто применяется для передачи параметров между процедурами и для формирования «контекстов» вычисления. В отличие от массива, доступ к элементам структуры выполняется по имени, что повышает читаемость. Соответствие также хранит пары, но ключом может быть не только строка, а любое хешируемое значение, включая ссылки на объекты. Соответствие удобно для быстрых «поисков по ключу», например для кэширования найденных ссылок по ИНН.

Таблица значений (ТаблицаЗначений) является наиболее мощной универсальной структурой. Она содержит коллекцию колонок и коллекцию строк. Колонки имеют имена и типы, строки содержат значения по колонкам. Таблица значений поддерживает сортировку, поиск, итоги, фильтрацию, а также может иметь индексы. Индексы используются для ускорения поиска строк по значениям колонок и особенно полезны при обработке больших объёмов данных на сервере.

При использовании таблицы значений важно различать операции «в памяти» и операции, требующие обращения к базе данных. Таблица значений живёт в памяти, поэтому подходит для промежуточных расчётов и передачи на клиент, но не заменяет запросы при необходимости выбрать большой объём данных из базы. Часто используется паттерн: запрос выгружается в таблицу значений, затем на сервере выполняется дополнительная обработка, после чего результат передаётся на клиент для отображения.

Для обеспечения переносимости между клиентом и сервером рекомендуется в качестве аргументов экспортных процедур использовать структуры и таблицы

значений, а также избегать передачи «несериализуемых» объектов, привязанных к контексту формы.

Пример 5. Индексация таблицы значений и быстрый поиск строк

Пример демонстрирует создание индекса по колонке Артикул и поиск строки по значению. В учебных задачах это полезно при проверках уникальности и при сопоставлении данных обмена.

```
Процедура ТаблицаСИндексом() Экспорт

    Товары = Новый ТаблицаЗначений;
    Товары.Колонки.Добавить("Артикул");
    Товары.Колонки.Добавить("Цена");

    Стр = Товары.Добавить();
    Стр.Артикул = "А-001";
    Стр.Цена = 100;

    Стр = Товары.Добавить();
    Стр.Артикул = "А-002";
    Стр.Цена = 150;

    // Индекс по артикулу ускоряет поиск
    Товары.Индексы.Добавить("Артикул");

    НайденнаяСтрока = Товары.Найти("А-002", "Артикул");
    Если НайденнаяСтрока <> Неопределено Тогда
```

```
        Сообщить("Цена по А-002: " + НайденнаяСтрока.Цена);
    КонецЕсли;

КонецПроцедуры
```

2.6. Операторы, выражения и форматирование

Операторы и выражения определяют вычислительную часть алгоритма. Для чисел доступны операции сложения, вычитания, умножения и деления, а также функции округления и математические функции (Окр, Цел, Макс, Мин). Для строк широко применяются конкатенация, подстроки и функции обработки текста (Лев, Прав, Сред, Найти, СтрЗаменить, СокрЛП). Для дат используются функции добавления интервалов, извлечения частей даты, вычисления разницы.

В типовых конфигурациях особое внимание уделяется форматированию. Пользователь ожидает видеть суммы в денежном представлении, даты в принятом формате и числа с нужной точностью. Функция Формат позволяет задавать строку формата, в которой описываются правила отображения числа или даты. Для чисел задаётся количество знаков после запятой, разделитель дробной части и группировка разрядов. Для дат задаётся шаблон представления (короткая дата, полная дата, дата и время).

Для компактных выражений часто применяется функция ?(). Она удобна в местах, где нужно вычислить значение на основе условия без развёрнутого «Если». Однако инженерно важно не злоупотреблять компактностью, если она ухудшает читаемость.

Операции сравнения и логические выражения используются при проверках заполненности и корректности данных. Следует помнить, что сравнение строк чувствительно к пробелам, поэтому для кодов и идентификаторов рекомендуется нормализация (СокрЛП). Сравнение ссылочных значений сравнивает идентификаторы

объектов, а не реквизиты. Для проверки отсутствия выбранного объекта применяется сравнение с пустой ссылкой соответствующего типа.

При построении алгоритмов расчётов рекомендуется явно фиксировать денежные и количественные показатели в установленной точности, чтобы избежать накопления ошибок округления. В документах и регистрах обычно применяются типы с заданной длиной и точностью.

Пример 6. Форматирование даты и суммы и нормализация строк

Пример показывает типовой приём: сокращение пробелов в строке, вывод даты в коротком формате и вывод суммы с двумя знаками после запятой.

```
Процедура ФорматИСтроки() Экспорт
    Код = "  А-001  ";
    Код = СокрЛП(Код);

    ДатаДок = ТекущаяДата();
    Сумма = 123456.7;

    Сообщить("Код: " + Код);
    Сообщить("Дата: " + Формат(ДатаДок, "ДФ=dd.ММ.уууу"));
    Сообщить("Сумма: " + Формат(Сумма, "ЧГ=15.2; ЧРД=.; ЧРГ= ;"));
КонецПроцедуры
```

2.7. Условные конструкции, проверки и исключения

Условные конструкции используются для ветвления логики и реализации правил предметной области. Базовый инструмент – «Если … Тогда … Иначе … КонецЕсли». Для нескольких альтернатив применяется «ИначеЕсли». Конструкция

«Выбор … Когда … Тогда … Иначе … КонецВыбора» удобна при обработке фиксированных наборов значений, например перечислений.

Проверки корректности данных в 1С размещаются в разных точках исполнения. В модуле формы проверки часто выполняются «по месту» при изменении реквизита, чтобы пользователь получил быстрый отклик. В модуле объекта проверки выполняются перед записью и перед проведением, чтобы обеспечить целостность данных независимо от интерфейса. При проведении документа особое значение имеет возможность установить флаг Отказ, который прекращает проведение.

Исключения применяются как механизм аварийного завершения алгоритма с сообщением о причине. Важно различать ситуации, когда целесообразно исключение, и когда следует использовать более мягкий механизм. Исключение удобно, когда продолжение выполнения бессмысленно (например, отсутствует обязательный реквизит) или когда требуется остановить цепочку вызовов. В обработчиках событий записи и проведения часто используется флаг Отказ вместе с сообщением, поскольку это встроенный протокол взаимодействия с платформой.

Для пользовательских сообщений в управляемом приложении применяются Сообщить, Предупреждение и СообщениеПользователю. При этом следует учитывать контекст: пользовательские диалоги и предупреждения обычно выполняются на клиенте. Если проверка выполняется на сервере, то результат проверки следует вернуть на клиент в виде сообщения или структуры ошибок.

В инженерной практике рекомендуется формулировать сообщения об ошибках так, чтобы они были проверяемыми: сообщение должно указывать, какой реквизит неверен и какое значение ожидается. Это облегчает исправление данных и уменьшает нагрузку на поддержку.

Пример 7. Проверка заполненности в событии ПередЗаписью

Фрагмент демонстрирует типовой паттерн в модуле объекта: проверка обязательных реквизитов и отказ от записи с понятным сообщением.

```
Процедура ПередЗаписью(Отказ, РежимЗаписи)

    Если Поставщик = Справочники.Контрагенты.ПустаяСсылка() Тогда
        Отказ = Истина;
        Сообщить("Не указан поставщик. Запись документа отменена.");
        Возврат;
    КонецЕсли;

    Если Товары.Количество() = 0 Тогда
        Отказ = Истина;
        Сообщить("Документ должен содержать хотя бы одну строку табличной части.");
        Возврат;
    КонецЕсли;

КонецПроцедуры
```

2.8. Циклы, обход выборок и типовые алгоритмы

Циклы во встроенном языке позволяют повторять операции и обрабатывать наборы данных. Конструкция «Для … По …» применяется при необходимости явного индекса. Конструкция «Для каждого … Из …» используется при обходе коллекций, таблиц значений, наборов записей и выборок. Конструкция «Пока» применяется, когда число итераций заранее неизвестно и определяется условием.

Типовые алгоритмы обработки данных в 1С часто сводятся к трём операциям: поиск, агрегирование и преобразование. Поиск включает нахождение элемента, удовлетворяющего условию, с возможностью раннего завершения цикла через Прервать. Агрегирование включает накопление суммы, подсчёт количества,

вычисление минимума и максимума. Преобразование включает формирование новой структуры данных из исходной: например, группировка строк по ключу в соответствие или построение итоговой таблицы значений.

При обработке данных из базы данных рекомендуется отдавать предпочтение запросам для фильтрации и агрегирования. Циклы на языке 1С следует использовать для дополнительной логики, которую неудобно выразить запросом, или для обработки уже выбранного небольшого набора. В противном случае можно получить низкую производительность из-за обработки больших объёмов «в цикле» вместо обработки на стороне СУБД.

При обходе выборок запросов применяется метод Следующий(). Для повышения удобства часто выгружают результат в таблицу значений и затем применяют к ней методы сортировки, поиска и агрегации. Однако выгрузка также требует памяти, поэтому для больших выборок следует выбирать построчный обход.

Отдельной темой являются циклы по табличным частям документов и по наборам записей регистров. Здесь важно помнить, что изменения набора записей должны завершаться записью набора, а в проведении документов записи обычно формируются через коллекцию Движения, которая будет записана платформой автоматически.

Пример 8. Агрегирование и раннее завершение поиска

Пример показывает вычисление суммарной суммы по таблице значений и поиск первой строки, удовлетворяющей условию.

```
Процедура ПоискИАгрегирование(Таблица) Экспорт


    Итог = 0;
    Найдено = Ложь;
```

```
    Для каждого Стр Из Таблица Цикл
        Итог = Итог + Стр.Сумма;

        Если (Не Найдено) И Стр.Сумма > 10000 Тогда
            Найдено = Истина;
            Сообщить("Первая строка с суммой > 10000: " + Стр.Сумма);
            // При необходимости можно Прервать, если дальнейший обход не
нужен
        КонецЕсли;
    КонецЦикла;

    Сообщить("Итоговая сумма: " + Итог);

КонецПроцедуры
```

2.9. Процедуры, функции, общие модули

Декомпозиция задач на процедуры и функции является базовой инженерной практикой. Процедуры и функции позволяют выделять повторяемые фрагменты логики, упрощать тестирование и повышать читаемость. В 1С принята практика выделять «проверки» в отдельные процедуры, «расчёты» – в функции, а «операции с данными» – в серверные процедуры, работающие с информационной базой.

Общие модули используются как библиотека функциональности. В них размещают общие функции форматирования, проверки, сервисные операции, а также бизнес-логику, не привязанную к конкретному объекту. Общие модули настраиваются по контексту: можно разрешить их использование на клиенте, на сервере или в обоих контекстах. При разработке управляемых приложений рекомендуется максимально переносить вычисления, связанные с данными, на сервер.

Клиент-серверный контекст – ключевой концепт 1С 8.3. Клиент отвечает за интерфейс и взаимодействие с пользователем. Сервер отвечает за доступ к данным и выполнение операций, требующих транзакций и блокировок. Директивы компиляции определяют, где выполняется конкретный метод. «&НаСервереБезКонтекста» полезна для чистых серверных функций, не использующих контекст формы, что улучшает производительность и переносимость.

При передаче данных между клиентом и сервером используются сериализуемые структуры: таблицы значений, структуры, массивы. При этом следует помнить, что передача больших таблиц значений может быть дорогой, поэтому рекомендуется передавать на клиент только объёмы, необходимые для отображения, а детальные вычисления выполнять на сервере.

В задачах, связанных с длительными операциями, следует учитывать, что пользовательский интерфейс не должен блокироваться слишком надолго. В промышленной разработке применяются фоновое выполнение и регламентные задания, однако в базовом курсе достаточно усвоить принцип: тяжёлые выборки и массовые изменения выполнять на сервере, а пользователю показывать прогресс и результат.

Пример 9. Разделение: клиентская команда и серверная выборка

Пример демонстрирует типовой подход: клиент вызывает серверную функцию, сервер возвращает таблицу значений, клиент отображает её в табличном поле формы.

```
&НаСервере
Функция ПолучитьСписокТоваров() Экспорт
    Запрос = Новый Запрос;
    Запрос.Текст =
    "ВЫБРАТЬ
```

```
    |   Номенклатура.Наименование КАК Наименование,
    |   Номенклатура.Артикул КАК Артикул,
    |   Номенклатура.Цена КАК Цена
    |ИЗ
    |   Справочник.Номенклатура КАК Номенклатура";
    Возврат Запрос.Выполнить().Выгрузить();
КонецФункции

&НаКлиенте
Процедура КомандаОбновить(Команда)
    Таб = ПолучитьСписокТоваров();
    ЭлементыФормы.Товары.Значение = Таб;
КонецПроцедуры
```

2.10. Метаданные и объектная модель: справочники и документы

Метаданные прикладного решения определяют состав бизнес-сущностей и их свойства. Справочник предназначен для хранения нормативно-справочной информации: номенклатура, контрагенты, сотрудники, склады. Документ фиксирует факт хозяйственной операции и является основой для формирования движений по регистрам и для построения аналитики.

Справочник включает реквизиты, табличные части (если требуется), формы и модули. Для справочника важно определить уникальные признаки (код, артикул, ИНН) и правила поиска. На уровне платформы доступны методы НайтиПоКоду, НайтиПоНаименованию и выборки. В промышленной разработке часто создают индексы на уровне СУБД и применяют запросы для поиска по произвольным реквизитам.

Документ имеет дату, номер (как правило, формируется автоматически), реквизиты шапки и табличные части. Документ проходит жизненный цикл: создание, заполнение, запись, проведение, отмена проведения. В событиях ПередЗаписью и ПриЗаписи размещают проверки и расчёты, в событии ОбработкаПроведения – формирование движений по регистрам.

Объектная модель платформы предоставляет менеджеры объектов (например, Справочники.Номенклатура и Документы.ПоступлениеТовара) и объекты данных (элемент справочника, документ). Менеджер предоставляет методы создания и выборки, объект – реквизиты и методы Записать, Провести, ПолучитьОбъект. Разделение напоминает разделение «класс – экземпляр» в объектно-ориентированных языках.

Для табличных частей важно понимать, что строка табличной части является объектом с реквизитами, а сама табличная часть поддерживает методы Добавить, Удалить, Очистить, Количество. При расчёте итогов документа типично пересчитывать сумму строки по количеству и цене и затем пересчитывать общую сумму.

В инженерной дисциплине разработки рекомендуется минимизировать дублирование логики между формой документа и модулем объекта. Проверки, критичные для целостности данных, следует размещать в модуле объекта, чтобы они выполнялись независимо от интерфейса и внешних обработок.

Пример 10. Программное создание документа и заполнение табличной части

Пример демонстрирует создание документа, заполнение реквизитов шапки и табличной части и запись. Имена объектов следует согласовать с учебной конфигурацией.

```
Процедура СоздатьДокументПоступления() Экспорт
```

```
    Док = Документы.ПоступлениеТовара.СоздатьДокумент();
    Док.Дата = ТекущаяДата();
    Док.Склад = Справочники.Склады.НайтиПоНаименованию("Основной");

    Стр = Док.Товары.Добавить();
    Стр.Номенклатура = Справочники.Номенклатура.НайтиПоНаименованию("Товар 1");
    Стр.Количество = 5;
    Стр.Цена = 200;

    Док.Записать();

    Сообщить("Документ записан: " + Док.Ссылка);

КонецПроцедуры
```

2.11. Регистры и проведение: движения, наборы записей, итоги

Регистры являются инструментом хранения итогов и аналитических данных, получаемых из документов и других источников. Наиболее распространены регистры накопления и регистры сведений. Регистр накопления используется для учёта остатков и оборотов ресурсов (количество, сумма) в разрезе измерений (номенклатура, склад, серия) и обычно заполняется движениями документов. Регистр сведений применяется для хранения параметров и состояний: цены, курсы валют, статусы, настройки.

Движение регистра накопления является записью, которая содержит измерения, ресурсы, период и вид движения (приход или расход). При проведении документа формируется набор движений, который платформа записывает в регистр в

рамках транзакции проведения. В управляемых приложениях проведение является ключевой точкой обеспечения целостности: именно здесь контролируется наличие остатков, корректность сумм и соответствие бизнес-правилам.

В модуле объекта документа проведение реализуется процедурой ОбработкаПроведения. Внутри процедуры доступна коллекция Движения, в которую добавляются записи по нужным регистрам. Для регистров сведений обычно создают набор записей и записывают его вручную, либо используют механизм движений, если регистр связан с документом как регистр, заполняемый при проведении.

Для получения итогов по регистру накопления применяются виртуальные таблицы, такие как Остатки и Обороты. Виртуальные таблицы позволяют получать остатки на дату и обороты за период без ручного суммирования движений. В запросах виртуальные таблицы дают эффективный доступ к аналитике, поскольку расчёты выполняются на стороне СУБД.

При проектировании регистра важен выбор измерений и ресурсов. Измерения определяют аналитические разрезы, ресурсы – то, что учитывается. Чем больше измерений, тем больше детализация и тем больше объём итогов; это влияет на производительность. В учебных конфигурациях полезно сравнить варианты: учёт остатков только по номенклатуре и учёт остатков по номенклатуре и складу.

Контроль отрицательных остатков является типовой задачей. Он может быть реализован либо запретом проведения документов, приводящих к отрицательному остатку, либо разрешением отрицательных остатков с фиксацией отклонений. В учебном курсе рекомендуется реализовать строгий запрет, чтобы закрепить навыки получения остатков виртуальными таблицами и формулирования исключений.

Пример 11. Формирование приходных движений по регистру накопления

Фрагмент предназначен для модуля объекта документа. Он демонстрирует формирование приходных движений по каждой строке табличной части.

```
Процедура ОбработкаПроведения(Отказ, РежимПроведения)

    Движения.ОстаткиТоваров.Очистить();

    Для каждого Стр Из Товары Цикл

        Если Стр.Количество <= 0 Тогда
            Отказ = Истина;
            Сообщить("Количество должно быть больше нуля.");
            Возврат;
        КонецЕсли;

        Движение = Движения.ОстаткиТоваров.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
        Движение.Номенклатура = Стр.Номенклатура;
        Движение.Склад = Склад;
        Движение.Количество = Стр.Количество;

    КонецЦикла;

КонецПроцедуры
```

2.12. Запросы

Язык запросов 1С предназначен для выборки и обработки данных. Он близок к SQL по структуре, однако работает с объектами метаданных и предоставляет специальные конструкции для платформенных объектов: табличных частей документов, виртуальных таблиц регистров, представлений и специальных функций.

Запрос формируется как текст и выполняется объектом Запрос. Параметры следует передавать через УстановитьПараметр. Это делает запрос устойчивым к ошибкам конкатенации строк, улучшает читаемость и позволяет СУБД оптимизировать выполнение. Типовая структура запроса включает раздел ВЫБРАТЬ, источник ИЗ, условия ГДЕ, а также УПОРЯДОЧИТЬ ПО и СГРУППИРОВАТЬ ПО при необходимости.

Соединения применяются, когда нужно совместить данные из нескольких источников: например, связать номенклатуру и остатки, документы и строки табличных частей. В языке запросов доступны внутренние и левосторонние соединения. Левое соединение позволяет получить все строки из левого источника и дополнить их данными правого, что полезно при формировании отчёта, включающего элементы без движений.

Агрегирование выполняется с помощью функций СУММА, МАКСИМУМ, МИНИМУМ, КОЛИЧЕСТВО и группировки. При построении отчётов по регистрам накопления часто применяют агрегирование по номенклатуре и складу. Важно помнить, что агрегирование выполняется на стороне СУБД и обычно эффективнее, чем суммирование в цикле языка 1С.

Для регистров накопления важны виртуальные таблицы. Остатки(&Дата) возвращают остатки на дату, Обороты(&Нач, &Кон) – обороты за период. В запросах виртуальные таблицы позволяют строить отчёты «остатки на дату», «движение за период», «анализ оборотов» без ручного пересчёта движений.

При оптимизации запросов следует соблюдать базовые правила: выбирать только нужные поля; использовать отборы по индексируемым полям; избегать лишних соединений; применять параметры. В учебных работах достаточно научиться писать корректные запросы и понимать, как по тексту запроса определить источник данных и форму результата.

Пример 12. Левое соединение номенклатуры и остатков

Пример демонстрирует получение списка всех товаров с их остатками, включая товары без движений, для которых остаток следует считать равным 0.

```
Процедура ОтчётОстатков(ДатаСреза) Экспорт

    Запрос = Новый Запрос;
    Запрос.Текст =
    "ВЫБРАТЬ
    |   Ном.Ссылка КАК Номенклатура,
    |   Ном.Наименование КАК Наименование,
    |   ЕСТЬNULL(Ост.КоличествоОстаток, 0) КАК Остаток
    |ИЗ
    |   Справочник.Номенклатура КАК Ном
    |    ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ОстаткиТоваров.Остатки(&Дата)
КАК Ост
    |   ПО Ост.Номенклатура = Ном.Ссылка
    |УПОРЯДОЧИТЬ ПО
    |   Наименование";

    Запрос.УстановитьПараметр("Дата", ДатаСреза);
```

```
Таб = Запрос.Выполнить().Выгрузить();
// Далее таблицу можно вывести в табличное поле формы или обработать на сервере

КонецПроцедуры
```

2.13. Отчёты и СКД

Отчёт в 1С – это средство представления данных пользователю в удобной форме. В управляемых приложениях наиболее распространённым инструментом построения отчётов является система компоновки данных (СКД). СКД позволяет описывать отчёт декларативно: задаётся набор данных (обычно запрос), ресурсы (суммируемые показатели), группировки, отборы, параметры и варианты. Пользователь может менять отборы и группировки без изменения кода.

Базовые элементы СКД включают: схему компоновки данных, наборы данных (обычно запросы), параметры, ресурсы, поля, настройки компоновки. Разработчик задаёт «что выбрать» и «как сгруппировать», а платформа обеспечивает выполнение и формирование табличного документа. В учебных работах достаточно освоить концепцию разделения: запрос отвечает за получение набора данных, а СКД – за представление и интерактивные настройки.

Типовая архитектура отчёта строится так: на сервере формируется объект КомпоновщикМакетаКомпоновкиДанных или используется стандартный механизм отчёта, затем выполняется компоновка и формируется табличный документ, который отображается на клиенте. При этом параметры отчёта (период, склад, номенклатура) могут задаваться пользователем через форму отчёта.

СКД особенно удобна для отчётов по регистрам накопления: остатки, обороты, анализ продаж и закупок. Однако даже при использовании СКД понимание

запросов остаётся ключевым: качество и производительность отчёта определяются тем, насколько корректно выбран источник данных и насколько эффективно сформулирован запрос.

В учебной конфигурации рекомендуется реализовать хотя бы один простой отчёт: «Остатки товаров на дату» или «Обороты за период». Это позволяет связать воедино проектирование регистра, проведение документов и аналитическую выборку.

Пример 13. Концептуальная схема отчёта «Остатки на дату»

В учебных целях полезно зафиксировать архитектурный принцип: данные формируются запросом к виртуальной таблице остатков, а вывод настраивается в СКД. Конкретная реализация в конфигураторе выполняется средствами платформы и зависит от состава метаданных.

2.14. Управляемые формы

Управляемые формы являются основой пользовательского интерфейса в современных конфигурациях 1С. Форма содержит реквизиты формы (данные, с которыми работает форма), элементы формы (визуальные компоненты) и команды.

Код формы размещается в модуле формы и реагирует на события: ПриОткрытии, ПриИзменении, ПередЗаписью, ПриНажатии и т. п.

Реквизиты формы могут быть привязаны к реквизитам объекта (например, документа) или быть независимыми (параметры отбора, служебные переменные). Элементы формы отображают реквизиты и обеспечивают ввод.

Команды используются для действий пользователя и могут быть привязаны к кнопкам, меню и командному интерфейсу. Команда обычно вызывает процедуру в модуле формы.

При разработке форм важно разделять интерфейсную логику и бизнес-логику. Форма должна обеспечивать удобство ввода и отображения, а бизнес-правила

– находиться в модуле объекта или в общих модулях. Это обеспечивает корректность данных независимо от способа ввода (форма, обмен, внешняя обработка) и уменьшает дублирование.

Динамические списки позволяют отображать список объектов на основе запроса с отбором и сортировкой. Это мощный инструмент для списков справочников и документов. В учебной практике полезно освоить добавление быстрых отборов и обновление списка по команде.

События формы часто используются для мгновенной валидации ввода. Однако следует избегать тяжёлых операций в обработчиках изменения реквизитов, чтобы не ухудшать отзывчивость интерфейса.

Если требуется сложный расчёт, его следует выполнять на сервере и возвращать результат на клиент.

Пример 14. Обработчик изменения реквизита и серверный перерасчёт

Фрагмент демонстрирует клиентский обработчик изменения и вызов серверной функции для пересчёта итогов. Такой подход сохраняет корректность и не перегружает клиент тяжёлой логикой.

```
&НаСервере
Функция РассчитатьИтог(Количество, Цена) Экспорт
    Если Количество < 0 Или Цена < 0 Тогда
        ВызватьИсключение "Количество и цена не могут быть отрицательными";
    КонецЕсли;
    Возврат Окр(Количество * Цена, 2);
КонецФункции

&НаКлиенте
```

```
Процедура КоличествоПриИзменении(Элемент)
    Итог = РассчитатьИтог(Количество, Цена);
КонецПроцедуры
```

2.15. Права доступа, роли и безопасная разработка

Права доступа определяют, какие пользователи и в каком объёме могут работать с данными и функциональностью. В 1С права обычно задаются через роли.

Роль содержит набор разрешений на чтение, запись, проведение, удаление объектов, а также доступ к отчётам и обработкам.

В учебных конфигурациях роли помогают закрепить принцип: разработчик должен мыслить не только алгоритмами, но и моделью безопасности.

На уровне платформы существуют проверки прав при выполнении операций.

Например, попытка записать документ без права записи приведёт к ошибке. В коде иногда требуется явная проверка прав перед выполнением операции, особенно если операция сложная и лучше заранее сообщить пользователю о невозможности выполнения. Для этого используются функции проверки прав и механизмы ролей.

Безопасная разработка включает отказ от конкатенации строк в запросах и применение параметров, минимизацию прав для пользователей, аккуратную обработку ошибок без раскрытия внутренних деталей, а также соблюдение принципа «проверка на сервере». Если проверка выполняется только в форме, то внешняя обработка или обмен могут обойти эту проверку, поэтому критичные проверки должны быть в модуле объекта или в серверном коде.

В учебных работах достаточно реализовать две роли: «Администратор» с полными правами и «Пользователь» с ограниченными правами (например, без удаления и без изменения справочников). Это позволит на практике увидеть, как

поведение системы зависит от прав, и закрепить привычку тестировать решения под разными ролями.

Пример 15. Проверка роли перед выполнением операции

Пример иллюстрирует концептуальный подход: если роль отсутствует, выполнение операции запрещается.

Конкретная реализация проверки может зависеть от состава ролей и используемых функций платформы.

```
Процедура ОперацияТолькоДляАдминистратора() Экспорт
    // Демонстрационный подход: в реальной конфигурации проверка прав задаётся
    //через роли и права доступа
    Если Не РольДоступна("Администратор") Тогда
        ВызватьИсключение "Операция доступна только пользователю с ролью «Администратор»";
    КонецЕсли;
КонецПроцедуры
```

3. ПРАКТИКУМ: СОЗДАНИЕ УЧЕБНОЙ КОНФИГУРАЦИИ «УЧЁТ СКЛАДА»

Практикум предназначен для поэтапного построения небольшой учебной конфигурации, демонстрирующей полный цикл: проектирование метаданных, разработка форм, запись и проведение документов, формирование движений по регистру и получение отчёта. Предметная область намеренно упрощена: учитывается движение товаров по складу без сложных расчётов, налогов и расширенной аналитики. При этом в ходе выполнения практикума закрепляются ключевые механизмы платформы.

Результатом практикума является конфигурация, содержащая справочники «Номенклатура», «Склады» и «Контрагенты», документы «ПоступлениеТовара» и «СписаниеТовара», регистр накопления «ОстаткиТоваров», а также отчёт «ОстаткиТоваровНаДату». Дополнительно вводятся базовые проверки заполненности и контроль отрицательных остатков при списании.

Практикум рекомендуется выполнять последовательно. После каждого этапа следует запускать режим «Предприятие» и проверять работоспособность: создание элементов справочников, запись документа, проведение и отражение движений. Такой подход предотвращает накопление ошибок и позволяет осмысленно связывать теорию с практикой.

3.1. Создание информационной базы и подготовка конфигурации

Создайте новую файловую информационную базу для учебных целей. В режиме «Конфигуратор» выполните первичную настройку: укажите имя конфигурации, включите использование управляемого приложения и убедитесь, что доступны формы списка и формы объекта для основных метаданных. При необходимости настройте основную форму приложения и командный интерфейс, чтобы справочники и документы были доступны пользователю в режиме «Предприятие».

Рекомендуется сразу установить единый стиль именования объектов: использовать понятные русские имена объектов и реквизитов, а также следить за единообразием. Например, для реквизитов документов использовать имена «Склад», «Контрагент», «Комментарий». Для табличных частей использовать имена, отражающие смысл, например «Товары».

3.2. Проектирование справочников

Создайте справочник «Номенклатура». В качестве реквизитов добавьте «Артикул» (строка фиксированной длины) и «Цена» (число с точностью до двух знаков). Обсудите, является ли цена постоянной характеристикой товара или должна храниться в регистре сведений. В учебной конфигурации допустимо хранить цену в справочнике, чтобы упростить задачи.

Создайте справочник «Склады». Для учебной конфигурации достаточно реквизита «Адрес» (строка), однако можно оставить справочник без дополнительных реквизитов и использовать только наименование.

Создайте справочник «Контрагенты». Добавьте реквизит «ИНН» и, при необходимости, «Телефон». На этом этапе полезно реализовать проверку уникальности ИНН при записи. Проверку можно выполнить в модуле объекта справочника: перед записью выполнить запрос или поиск по реквизиту ИНН и запретить запись при обнаружении другого элемента с тем же ИНН.

3.3. Проектирование документов и табличных частей

Создайте документ «ПоступлениеТовара». В шапке документа добавьте реквизиты «Склад» (ссылка на справочник «Склады») и «Поставщик» (ссылка на справочник «Контрагенты»). Создайте табличную часть «Товары» с реквизитами строки: «Номенклатура» (ссылка на справочник), «Количество» (число), «Цена» (число) и «Сумма» (число). Реализуйте пересчёт суммы строки как произведение количества и цены, а также пересчёт итоговой суммы документа.

Создайте документ «СписаниеТовара» с аналогичной табличной частью. В шапке можно оставить только реквизит «Склад». Для списания важно реализовать контроль отрицательных остатков при проведении. Это позволит закрепить навыки работы с виртуальной таблицей остатков в запросе.

3.4. Проектирование регистра накопления и аналитических разрезов

Создайте регистр накопления «ОстаткиТоваров» с видом «Остатки». В качестве измерений добавьте «Номенклатура» и «Склад». В качестве ресурса добавьте «Количество». Такой состав обеспечивает получение остатков по товарам и складам. При желании можно добавить ресурс «Сумма», чтобы вести денежный учёт, однако для базового практикума достаточно количества.

Изменение измерений напрямую влияет на аналитику. Например:

При исключении измерения «Склад» остатки будут агрегированы по всем складам. Это упрощает модель, но исключает возможность анализа данных в разрезе отдельных складов.

При добавлении измерения «Серия» модель становится более детализированной, однако возрастает её сложность. Таким образом, выбор измерений определяет баланс между уровнем детализации данных и ресурсозатратностью вычислений.

3.5. Проведение документов и формирование движений

В модуле объекта документа «ПоступлениеТовара» реализуйте процедуру ОбработкаПроведения. В ней создайте приходные движения по регистру «ОстаткиТоваров» для каждой строки табличной части. Проверьте, что количество положительное, а номенклатура заполнена. При ошибке установите Отказ и выведите понятное сообщение.

В модуле объекта документа «СписаниеТовара» реализуйте расходные движения. Перед добавлением расходного движения выполните запрос к виртуальной

таблице остатков регистра на дату документа и склад, чтобы получить доступный остаток по номенклатуре. Если доступного остатка меньше списываемого количества, запретите проведение с сообщением. Такой контроль иллюстрирует правильный подход: проверка должна выполняться на сервере и быть независимой от формы.

После реализации проведения протестируйте в режиме «Предприятие». Создайте номенклатуру, склад и контрагента. Проведите поступление, затем выполните списание в пределах остатка и убедитесь, что движения и остатки корректны. Затем попробуйте списать больше, чем доступно, и убедитесь, что проведение запрещается.

3.6. Отчёт «Остатки товаров на дату»

Создайте отчёт, выводящий остатки товаров на выбранную дату и, при необходимости, по выбранному складу. Для получения данных используйте запрос к виртуальной таблице Остатки регистра накопления. В запросе включите номенклатуру, склад и количество остатка. Далее используйте СКД для группировки по складу и номенклатуре и для вывода ресурса «КоличествоОстаток».

В форме отчёта добавьте параметры: дата среза и склад. Параметры должны передаваться в запрос. Настройте вариант отчёта по умолчанию и проверьте, что пользователь может менять отбор по складу. Это закрепляет связку «параметр формы – параметр запроса – настройка СКД».

3.7. Роли и тестирование под разными пользователями

Создайте роли «Администратор» и «Пользователь». Для роли пользователя ограничьте возможности удаления и изменения справочников, оставив возможность ввода и проведения документов. Создайте тестовых пользователей и назначьте им роли. В режиме «Предприятие» войдите под пользователем и проверьте, что ограничения работают: пользователь не может удалять элементы справочников и не может выполнять недоступные операции.

Отдельно протестируйте сценарии ошибок: попытка записать документ без обязательного реквизита, попытка провести списание при недостатке остатков, ввод отрицательных значений. Убедитесь, что сообщения об ошибках понятны и что данные не оказываются в частично записанном состоянии. Этот этап формирует инженерную привычку проверять решение на целостность и удобство пользователя.

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Задания предназначены для закрепления теоретического материала и формирования устойчивых навыков разработки. Каждое задание включает пять вариантов, что позволяет организовать индивидуальную работу студентов. При выполнении вариантов рекомендуется вести рабочий журнал: фиксировать состав созданных метаданных, размещение кода и результаты тестирования.

Общие требования к выполнению: программа должна корректно обрабатывать незаполненные значения и некорректный ввод; сообщения об ошибках должны быть понятными пользователю; код должен быть структурирован на процедуры и функции; при работе с данными информационной базы следует соблюдать клиент-серверный контекст. В заданиях, связанных с документами и регистрами, обязательны проверки заполненности шапки и табличной части, а также контроль значений количества и цены.

Задание 1: Переменные, типы данных и коллекции

Цель: освоить создание переменных примитивных и ссылочных типов, работу с массивами, структурами и таблицами значений, базовые проверки заполненности.

Методические указания: Перед началом определите ожидаемые типы значений. Для ссылок используйте проверку на пустую ссылку, для строк – СокрЛП и

проверку на пустоту. При работе с таблицей значений заранее добавьте колонки и, при необходимости, индекс для ускорения поиска.

Вариант 1. Создайте структуру «Товар» с полями Наименование, Артикул, Цена, ВНаличии. Сформируйте строку с представлением товара и выведите её. Если цена не задана, подставьте 0 и отразите это в сообщении.

Вариант 2. Создайте таблицу значений «СкладОстатки» с колонками Номенклатура, Количество, Склад. Заполните пять строк тестовыми значениями и вычислите суммарное количество, игнорируя строки с незаполненной номенклатурой.

Вариант 3. Сформируйте массив дат на ближайшие семь дней от текущей даты и для каждой даты выведите сообщение с датой в коротком формате и номером дня недели.

Вариант 4. Создайте соответствие «КурсВалют», где ключ – код валюты, значение – курс. Реализуйте процедуру получения курса по коду с обработкой ситуации отсутствия ключа.

Вариант 5. Реализуйте процедуру, принимающую произвольное значение и выводящую его тип (ТипЗнч) и признак заполненности (ЗначениеЗаполнено). Проверьте работу на числе, строке, дате, пустой ссылке справочника и значении Неопределено.

Задание 2: Операторы, выражения и форматирование

Цель: закрепить арифметические и строковые операции, сравнения, логические выражения, применение функции ?() и функции Формат.

Методические указания: Старайтесь отделять расчёты от вывода. Форматирование выполняйте на этапе вывода пользователю. При расчёте денежных сумм используйте округление. Для строк применяйте нормализацию пробелов.

Вариант 1. Рассчитайте сумму с НДС: пользователь вводит сумму и ставку. Выведите НДС и итоговую сумму, отформатировав денежные значения с двумя знаками после запятой.

Вариант 2. Сформируйте строку вида «Фамилия И.О.» из трёх входных строк. Учтите, что отчество может быть пустым, а имя и фамилия могут содержать лишние пробелы.

Вариант 3. Создайте функцию, возвращающую текст «Критично» или «Нормально» по числу ошибок: если ошибок больше пяти, то «Критично», иначе «Нормально». Используйте ?().

Вариант 4. Сравните две ссылки на элементы справочника «Контрагенты» и выведите результат. Дополнительно покажите проверку ссылки на пустую ссылку и поясните смысл этой проверки сообщением.

Вариант 5. Даны две даты. Вычислите количество дней между датами, выведите результат и определите логическим выражением, превышает ли период 30 дней.

Задание 3: Условные конструкции и исключения

Цель: научиться строить ветвления с Если/ИначеЕсли/Иначе и Выбор, применять Попытка/Исключение для обработки ошибок и контролировать корректность данных.

Методические указания: Проверки выполняйте в начале процедур (принцип «fail-fast»). Для некорректных данных используйте исключение или отказ, но обеспечьте понятное сообщение. В блоке Попытка/Исключение фиксируйте Описание-Ошибки для диагностики.

Вариант 1. По введённому баллу (0–100) определите буквенную оценку A–F. Некорректный ввод обработайте исключением с пояснением.

Вариант 2. По номеру месяца (1–12) верните название месяца и сезон. Реализуйте через конструкцию Выбор.

Вариант 3. Проверьте ИНН как строку: длина 10 или 12 и только цифры. При ошибке сформируйте исключение, иначе выведите сообщение о успешной проверке.

Вариант 4. Реализуйте проверку доступа: если текущий пользователь не имеет роль «Администратор», запретите выполнение операции через исключение; вывод сообщения выполните в обработчике исключения.

Вариант 5. Организуйте ввод цены в цикле: при нечисловом вводе или цене меньше либо равной нулю повторяйте ввод до получения корректного значения. Ошибки преобразования обработайте через Попытка/Исключение.

Задание 4: Циклы и обработка наборов данных

Цель: освоить циклы Для, Для каждого и Пока, применять Прервать и Продолжить, выполнять расчёты по коллекциям и выборкам.

Методические указания: Отличайте обработку данных «в памяти» от выборок из базы. Применяйте Прервать для раннего завершения поиска. Для сумм и количеств задавайте стартовое значение и избегайте повторного вычисления в теле цикла.

Вариант 1. Сформируйте массив из 20 чисел и выведите только чётные значения. Для нечётных используйте Продолжить.

Вариант 2. По таблице значений с колонками Цена и Количество вычислите общую сумму. Нулевые количества пропускайте. При отрицательном количестве прервите цикл и сообщите об ошибке.

Вариант 3. Реализуйте поиск первого элемента массива, удовлетворяющего условию «больше порога». При нахождении прекратите цикл и выведите значение и его индекс.

Вариант 4. Используя цикл Пока, организуйте ввод строк в таблицу значений до команды «Стоп». После завершения выведите число введённых строк и суммарное количество.

Вариант 5. Смоделируйте выборку: создайте таблицу значений с колонками ФИО и Оклад. Выведите сотрудников с окладом выше порога.

Задание 5: Процедуры и функции

Цель: отработать декомпозицию задач на процедуры и функции, передачу параметров, возврат значений и повторное использование кода.

Методические указания: Выделяйте повторяемые расчёты в функции. Структуры используйте для возврата нескольких результатов. Экспортные функции размещайте в общем модуле, чтобы их можно было вызывать из форм и объектов.

Вариант 1. Напишите функцию расчёта НДС, принимающую сумму и ставку. Функция должна возвращать структуру с полями НДС и Итого.

Вариант 2. Реализуйте функцию подсчёта слов в строке. Учтите лишние пробелы в начале, конце и внутри строки.

Вариант 3. Создайте процедуру, которая принимает таблицу значений и имя колонки и выводит значения этой колонки построчно.

Вариант 4. Реализуйте функцию, определяющую, является ли дата рабочим днём. В качестве результата вернуть Истина для понедельника–пятницы.

Вариант 5. Создайте экспортную функцию в общем модуле, формирующую номер по шаблону «PRE-000001» по префиксу и порядковому номеру.

Задание 6: Работа со справочниками (объектный интерфейс)

Цель: закрепить создание, чтение и изменение элементов справочников, работу с менеджерами объектов, поиск и отбор.

Методические указания: Создание элемента выполняйте через СоздатьЭлемент, заполнение реквизитов – через свойства объекта, запись – методом Записать. Для массовых операций используйте выборку или запрос, но учитывайте права доступа и необходимость обработки ошибок записи.

Вариант 1. Создайте справочник «Номенклатура» с реквизитами Артикул и Цена. Напишите процедуру, создающую пять элементов справочника программно и записывающую их.

Вариант 2. Напишите обработку поиска контрагента по ИНН. Если найден – выведите наименование, иначе создайте новый элемент с заданным ИНН и запишите его.

Вариант 3. Сформируйте выборку всех сотрудников и выведите ФИО и оклад. Сортировку по окладу реализуйте запросом или предварительной выгрузкой в таблицу значений.

Вариант 4. Реализуйте массовое исправление: для всех товаров с ценой 0 установить цену 1. Подсчитайте количество изменённых элементов и выведите итог.

Вариант 5. Создайте форму списка справочника и команду «Показать статистику», выводящую количество элементов и диапазон цен.

Задание 7: Документы и табличные части

Цель: освоить структуру документа, заполнение реквизитов и табличных частей, запись и проведение, базовые проверки перед записью.

Методические указания: Проверки количества и цены выполняйте как в форме (для удобства), так и в модуле объекта (для целостности). Пересчёт суммы строки делайте в обработчиках изменения и дублируйте в модуле объекта перед записью.

Вариант 1. Создайте документ «ПоступлениеТовара» с реквизитами Склад и Поставщик и табличной частью Товары. Реализуйте проверки положительности количества и цены.

Вариант 2. Добавьте команду автозаполнения табличной части тестовыми строками. Команда должна очищать табличную часть и добавлять три строки.

Вариант 3. Реализуйте пересчёт сумм: сумма строки и общая сумма документа должны пересчитываться при изменении количества или цены.

Вариант 4. При записи документа автоматически заполните реквизит «Ответственный» текущим пользователем, если он не заполнен.

Вариант 5. Запретите запись документа без строк в табличной части, сформировав исключение с понятным текстом.

Задание 8: Регистры и проведение документов

Цель: изучить механизм движений, формирование записей регистров при проведении документа и получение итогов по регистрам.

Методические указания: Движения формируйте в ОбработкаПроведения. Для контроля остатков используйте запрос к виртуальной таблице Остатки на дату документа. При недостатке остатка устанавливайте Отказ и формируйте сообщение с указанием номенклатуры и доступного остатка.

Вариант 1. Создайте регистр накопления «ОстаткиТоваров» и при проведении документа «ПоступлениеТовара» формируйте приходные движения по количеству.

Вариант 2. Добавьте документ «СписаниеТовара» и реализуйте расходные движения. Перед проведением проверьте наличие остатка и запретите проведение при недостатке.

Вариант 3. Сформируйте обработку, выводящую остатки на выбранную дату и склад. Используйте виртуальную таблицу остатков в запросе.

Вариант 4. Добавьте измерение «Серия» к регистру и измените проведение так, чтобы серия учитывалась. Заполните серию в табличной части документа.

Вариант 5. Реализуйте контроль отрицательных остатков: при попытке ухода в минус сформируйте исключение и зарегистрируйте информацию о попытке в журнале регистрации.

Задание 9: Запросы

Цель: закрепить написание запросов к справочникам, документам и регистрам, использование параметров, группировок и обход выборки.

Методические указания: Всегда используйте параметры запроса вместо конкатенации строк. Для соединений внимательно сопоставляйте поля ссылок. Для итогов применяйте СГРУППИРОВАТЬ ПО и агрегатные функции. Результат для формы обычно удобнее выгрузить в таблицу значений.

Вариант 1. Напишите параметризованный запрос к справочнику «Номенклатура», выбирающий товары с ценой выше заданной и сортирующий по цене.

Вариант 2. Сформируйте запрос к документу «ПоступлениеТовара» и его табличной части, выводящий строки поступления за период по выбранному складу.

Вариант 3. Напишите запрос к регистру «ОстаткиТоваров», группирующий данные по номенклатуре и вычисляющий суммарный остаток.

Вариант 4. Реализуйте запрос с левым соединением, показывающий все товары и их остатки, включая товары без движений, где остаток следует считать равным 0.

Вариант 5. Сформируйте краткий отчёт по периоду: количество документов, суммарная сумма поступлений, топ-5 товаров по количеству.

Задание 10: Формы, команды и клиент-серверное взаимодействие

Цель: сформировать навыки разработки управляемых форм, обработки событий и правильного разделения клиентского и серверного кода.

Методические указания: События формы не должны выполнять тяжёлые выборки. Организуйте серверные функции для получения данных и клиентские процедуры для отображения. Для команд используйте понятные имена и учитывайте, что пользователю важен быстрый отклик.

Вариант 1. Создайте управляемую форму обработки с параметрами и кнопкой «Рассчитать». Расчёт выполните на сервере, результат покажите на клиенте.

Вариант 2. В форме документа реализуйте команду «Проверить заполнение», которая проверяет обязательные реквизиты и выводит найденные проблемы.

Вариант 3. Создайте форму списка справочника с быстрым отбором по цене и командой обновления списка. Отбор должен применяться без закрытия формы.

Вариант 4. Реализуйте обработку, получающую на сервере данные запросом, возвращающую их на клиент в виде таблицы значений и отображающую в табличном поле формы.

Вариант 5. Добавьте в форму обработчик перед записью: при некорректных значениях отмените запись и покажите предупреждение пользователю.

ТИПОВЫЕ ПРИМЕРЫ КОДА

Раздел содержит типовые фрагменты, иллюстрирующие распространённые операции: создание и изменение объектов, выполнение запросов, формирование движений, передача данных между клиентом и сервером. Примеры следует адаптировать под состав метаданных учебной конфигурации. При переносе в реальную конфигурацию необходимо учитывать права доступа, ограничения уникальности, транзакции и производительность.

Пример А. Создание элемента справочника с проверкой уникальности реквизита

Подход заключается в том, чтобы перед записью выполнить поиск по уникальному реквизиту (например, ИНН) и при обнаружении другого элемента запретить запись. В промышленной разработке уникальность можно обеспечивать также индексами и ограничениями, однако учебный пример полезен для закрепления навыков запросов и проверок.

```
Процедура ПередЗаписью(Отказ, РежимЗаписи)

    Если Не ЗначениеЗаполнено(ИНН) Тогда
        Отказ = Истина;
        Сообщить("ИНН обязателен для заполнения.");
        Возврат;
    КонецЕсли;

    Запрос = Новый Запрос;
    Запрос.Текст =
    "ВЫБРАТЬ ПЕРВЫЕ 1
    |   Контр.Ссылка
```

```
    |ИЗ
    |   Справочник.Контрагенты КАК Контр
    |ГДЕ
    |   Контр.ИНН = &ИНН
    |   И Контр.Ссылка <> &ТекущаяСсылка";

    Запрос.УстановитьПараметр("ИНН", ИНН);
    Запрос.УстановитьПараметр("ТекущаяСсылка", Ссылка);

    Если Не Запрос.Выполнить().Пустой() Тогда
        Отказ = Истина;
        Сообщить("Контрагент с таким ИНН уже существует.");
        Возврат;
    КонецЕсли;

КонецПроцедуры
```

Пример Б. Автозаполнение табличной части в форме документа

Автозаполнение удобно для тестирования и для ускорения ввода. В управляемой форме команда очищает табличную часть и добавляет несколько строк. В реальной задаче автозаполнение может получать данные из запроса или из внешнего источника.

```
&НаКлиенте
Процедура КомандаАвтозаполнить(Команда)
    Товары.Очистить();
```

```
    ДобавитьСтроку("Товар 1", 1, 100);
    ДобавитьСтроку("Товар 2", 2, 50);
    ДобавитьСтроку("Товар 3", 3, 10);

КонецПроцедуры

&НаКлиенте
Процедура ДобавитьСтроку(НаименованиеТовара, Количество, Цена)
    Стр = Товары.Добавить();
    Стр.Номенклатура = Справочники.Номенклатура.НайтиПоНаименованию(НаименованиеТовара);
    Стр.Количество = Количество;
    Стр.Цена = Цена;
    Стр.Сумма = Окр(Количество * Цена, 2);
КонецПроцедуры
```

Пример В. Контроль остатков при списании (запрос к виртуальной таблице)

Контроль отрицательных остатков выполняется на сервере в момент проведения. Пример показывает получение остатка по номенклатуре и складу на дату документа и сравнение с требуемым количеством. В учебном варианте контроль выполняется по каждой строке отдельно.

```
Процедура ОбработкаПроведения(Отказ, РежимПроведения)

    Для каждого Стр Из Товары Цикл
```

```
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|   ЕСТЬNULL(Ост.КоличествоОстаток, 0) КАК Остаток
|ИЗ
|   РегистрНакопления.ОстаткиТоваров.Остатки(&Дата) КАК Ост
|ГДЕ
|   Ост.Номенклатура = &Номенклатура
|   И Ост.Склад = &Склад";

Запрос.УстановитьПараметр("Дата", Дата);
Запрос.УстановитьПараметр("Номенклатура", Стр.Номенклатура);
Запрос.УстановитьПараметр("Склад", Склад);

Выборка = Запрос.Выполнить().Выбрать();
Остаток = 0;
Если Выборка.Следующий() Тогда
    Остаток = Выборка.Остаток;
КонецЕсли;

Если Остаток < Стр.Количество Тогда
    Отказ = Истина;
    Сообщить("Недостаточно остатка по " + Стр.Номенклатура + ". Доступно: " + Остаток);
    Возврат;
```

```
        КонецЕсли;

        Движение = Движения.ОстаткиТоваров.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
        Движение.Номенклатура = Стр.Номенклатура;
        Движение.Склад = Склад;
        Движение.Количество = Стр.Количество;

    КонецЦикла;

КонецПроцедуры
```

Пример Г. Универсальная проверка заполненности документа

Пример иллюстрирует подход «собрать ошибки – показать пользователю». Это удобно в форме документа: вместо остановки на первой ошибке пользователь получает полный список проблем, которые нужно исправить.

```
&НаСервере
Функция ПроверитьДокумент(ДокументОбъект) Экспорт
    Ошибки = Новый Массив;

    Если ДокументОбъект.Склад = Справочники.Склады.ПустаяСсылка()
Тогда
        Ошибки.Добавить("Не заполнен склад.");
    КонецЕсли;

    Если ДокументОбъект.Товары.Количество() = 0 Тогда
```

```
        Ошибки.Добавить("Не заполнена табличная часть «Товары».");
    КонецЕсли;

    Для каждого Стр Из ДокументОбъект.Товары Цикл
        Если Стр.Номенклатура = Справочники.Номенклатура.ПустаяСсылка() Тогда
            Ошибки.Добавить("В табличной части есть строка без номенклатуры.");
        КонецЕсли;
        Если Стр.Количество <= 0 Тогда
            Ошибки.Добавить("Количество должно быть больше нуля.");
        КонецЕсли;
    КонецЦикла;

    Возврат Ошибки;
КонецФункции
```

РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЗАДАНИЙ

Эффективное освоение 1С требует системной дисциплины работы с конфигурацией. Рекомендуется начинать каждое практическое задание с описания мини-проекта: какие объекты метаданных нужны, какие реквизиты обязательны, какие проверки должны выполняться и где именно они должны быть реализованы. Полезно заранее определить архитектуру: бизнес-логика в модуле объекта и общих модулях, интерфейсная логика в форме, операции доступа к данным – на сервере.

В ходе разработки придерживайтесь принципа малых шагов. После каждого изменения метаданных выполняйте обновление информационной базы и короткий тест. Если ошибка обнаружена, локализуйте её сразу: поставьте точку останова, проверьте значения переменных, убедитесь в ожидаемом типе значений и контексте выполнения. Практика показывает, что большинство ошибок начинающих связано не с «сложностью языка», а с непониманием момента выполнения кода и состояния данных.

Особое внимание следует уделять различению состояний: неопределено, пустая ссылка, пустая строка и нулевое число являются разными ситуациями. Привычка явно проверять заполненность реквизитов и корректность числовых значений существенно повышает надёжность решений. Сообщения об ошибках должны быть сформулированы так, чтобы пользователь мог исправить данные без обращения к разработчику.

При работе с запросами следует использовать параметры и избегать конкатенации строк. Для отчётов по регистрам применяйте виртуальные таблицы. Для обработки больших объёмов данных старайтесь переносить фильтрацию и агрегирование в запрос, а не выполнять их в циклах. Такой подход не только повышает производительность, но и формирует правильное инженерное мышление: СУБД предназначена для выборок и группировок, язык 1С – для бизнес-логики.

В заданиях, связанных с формами, следует избегать тяжёлых операций в обработчиках изменения реквизитов. Если требуется перерасчёт или выборка, организуйте серверную функцию и вызывайте её из клиентской процедуры. Это улучшает отзывчивость интерфейса и соответствует архитектуре управляемых приложений.

Для самоконтроля полезно после завершения задания выполнить тестовый сценарий: создать минимальный набор данных; выполнить типовую операцию (создание элемента, запись документа, проведение); сформировать отчёт или выборку; проверить корректность итогов; повторить сценарий под пользователем с ограниченными правами.

СПИСОК ЛИТЕРАТУРЫ

1. Фирма «1С». Руководство разработчика. «1С:Предприятие 8.3» : электрон. документация. – URL: **https://its.1c.ru/db/v8327doc** (дата обращения: 17.12.2025).
2. Фирма «1С». 1С:ИТС (информационно-технологическое сопровождение) : официальный ресурс. – URL: **https://its.1c.ru/** (дата обращения: 17.12.2025).
3. Радченко М. Г., Хрусталева Е. Ю. 1С:Предприятие 8.3. Практическое пособие разработчика : примеры и типовые приемы. Изд. 3-е : электрон. книга. – М. : Фирма «1С», 2022. – URL: **https://its.1c.ru/db/pubdevguide83** (дата обращения: 17.12.2025).
4. Гилёв А. И. 1С:Предприятие 8.3. Разработка управляемых приложений: проектирование, формы, запросы, регистры : методические материалы/публикации автора (gilev.ru). – URL: **http://www.gilev.info/** (дата обращения: 17.12.2025).
5. Фирма «1С». Материалы сертификаций «1С:Профессионал» и «1С:Специалист» по платформе «1С:Предприятие 8.3» : электрон. ресурсы (разделы

подготовки/экзаменов на ИТС). – URL: **https://its.1c.ru/** (дата обращения: 17.12.2025).

6. Фирма «1С». Архитектура и производительность прикладных решений «1С:Предприятие 8.3» : методические рекомендации (разделы по оптимизации, запросам, массовым операциям на ИТС/в документации). – URL: **https://its.1c.ru/db/v8327doc** (дата обращения: 17.12.2025)

ПРИЛОЖЕНИЕ 1. ГЛОССАРИЙ ОСНОВНЫХ ТЕРМИНОВ

Информационная база. Совокупность данных и настроек, с которыми работает прикладное решение 1С. Может быть файловой или клиент-серверной.

Платформа. Технологическая часть «1С:Предприятие», обеспечивающая выполнение кода, интерфейс, хранение и целостность данных.

Конфигурация. Описание прикладного решения: метаданные, формы, модули, роли, отчёты и обработки.

Метаданные. Структурное описание объектов конфигурации (справочники, документы, регистры и т. п.) и их свойств.

Справочник. Объект метаданных для хранения нормативно-справочной информации (товары, контрагенты, склады).

Документ. Объект метаданных для фиксации фактов хозяйственной деятельности; часто формирует движения по регистрам при проведении.

Регистр накопления. Объект для учёта ресурсов (количество, сумма) по измерениям и получения остатков и оборотов.

Регистр сведений. Объект для хранения наборов записей с параметрами и состояниями (цены, статусы, настройки).

Измерение. Аналитический разрез регистра (например, номенклатура, склад), по которому ведётся детализация.

Ресурс. Учитываемый показатель регистра (например, количество, сумма).

Движение. Запись в регистр, формируемая при проведении документа; имеет вид движения (приход/расход) и период.

Виртуальная таблица. Специальное представление данных регистра (остатки, обороты), формируемое платформой для запросов.

Запрос. Механизм выборки и обработки данных, выполняемый на стороне СУБД по тексту запроса и параметрам.

Параметр запроса. Значение, передаваемое в запрос методом УстановитьПараметр; повышает безопасность и производительность.

СКД. Система компоновки данных: механизм построения отчётов с интерактивными настройками группировок и отборов.

Управляемая форма. Интерфейсный объект управляемого приложения, содержащий реквизиты, элементы и команды.

Реквизит формы. Данные формы, которые могут быть связаны с реквизитами объекта или служебными параметрами.

Команда. Действие пользователя, запускающее обработчик в модуле формы или вызывающее серверную логику.

Модуль объекта. Модуль, содержащий логику конкретного объекта (элемента справочника или документа).

Общий модуль. Модуль библиотеки, содержащий общие процедуры и функции для разных объектов и форм.

Контекст выполнения. Среда выполнения кода: клиентская или серверная; определяет доступность данных и объектов.

Директива компиляции. Аннотация вида &НаКлиенте или &НаСервере, задающая место компиляции и выполнения метода.

Пустая ссылка. Специальное значение ссылочного типа, обозначающее отсутствие выбранного объекта.

Неопределено. Значение, обозначающее отсутствие значения вообще; отличается от пустой ссылки.

Табличная часть. Табличный набор строк внутри документа (или справочника), содержащий реквизиты строк.

Отказ. Флаг, используемый в событиях записи и проведения для отмены операции платформой.

Проведение. Операция, при которой документ формирует движения по регистрами и влияет на итоги.

Журнал регистрации. Системный журнал событий, фиксирующий ошибки, предупреждения и значимые действия.

Роль. Набор прав доступа, назначаемый пользователям для ограничения операций с объектами и данными.

Транзакция. Атомарная единица изменений данных; при ошибке изменения откатываются.

Блокировка. Механизм предотвращения конфликтов при одновременной работе пользователей с одними данными.

Индекс. Структура, ускоряющая поиск по полям в базе данных или по колонкам в таблице значений.

Таблица значений. Универсальный табличный контейнер в памяти, используемый для передачи данных и промежуточных расчётов.

Структура. Универсальный контейнер пар «имя–значение», часто используемый для параметров и возвращаемых результатов.

Соответствие. Контейнер пар «ключ–значение» с быстрым доступом по ключу; ключом может быть ссылка.

Выборка. Итератор результата запроса или выборки объектов, обход которого выполняется методом Следующий().

ПРИЛОЖЕНИЕ 2. ШПАРГАЛКА ПО СИНТАКСИСУ И ТИПОВЫМ ПАТТЕРНАМ

Приложение содержит компактные шаблоны, которые удобно использовать при выполнении практических заданий. Шаблоны не заменяют понимание механики, но помогают быстро вспомнить синтаксис и стандартные конструкции.

Шаблон 1. Попытка–Исключение

```
Попытка
    // потенциально опасная операция
Исключение
    Сообщить("Ошибка: " + ОписаниеОшибки());
КонецПопытки;
```

Шаблон 2. Если–ИначеЕсли–Иначе

```
Если Условие1 Тогда
    // ветка 1
ИначеЕсли Условие2 Тогда
    // ветка 2
Иначе
    // иначе
КонецЕсли;
```

Шаблон 3. Для каждого по таблице значений

```
Для каждого Стр Из Таблица Цикл
    // Стр.Колонка1, Стр.Колонка2 ...
КонецЦикла;
```

Шаблон 4. Параметризованный запрос

```
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|   Поле1,
|   Поле2
|ИЗ
|   Источник КАК Т
|ГДЕ
|   Т.Поле = &Параметр";

Запрос.УстановитьПараметр("Параметр", ЗначениеПараметра);
Таблица = Запрос.Выполнить().Выгрузить();
```

Шаблон 5. Проверка заполненности

```
Если Не ЗначениеЗаполнено(Параметр) Тогда
   ВызватьИсключение "Параметр не заполнен";
КонецЕсли;
```

ПРИЛОЖЕНИЕ 3. ТИПОВЫЕ ОШИБКИ И ПРИЁМЫ ОТЛАДКИ

Разработка в 1С, как и в любой платформе, сопровождается типовыми ошибками. В учебной практике важно не только исправлять ошибку, но и понимать её причину, чтобы не повторять в дальнейшем. Ниже приведены наиболее распространённые ситуации и подходы к их диагностике.

Ошибки заполненности и пустых значений. Частая причина аварий – обращение к свойству или методу у значения Неопределено или у пустой ссылки. Для профилактики следует: в начале процедур проверять обязательные параметры; использовать ЗначениеЗаполнено для строк и чисел; для ссылок сравнивать с ПустаяСсылка соответствующего типа; при работе с формой помнить, что реквизит может быть не заполнен даже если пользователь «намеревался» его заполнить.

Ошибки контекста клиент–сервер. В управляемых приложениях ошибки возникают при попытке выполнить серверную операцию на клиенте: запросы к базе, запись объектов, массовые операции. Диагностика начинается с определения места выполнения: проверьте директивы &НаКлиенте и &НаСервере, а также стек вызовов. Инженерное решение – вынести работу с данными на сервер и оставить на клиенте только вызов и отображение результата.

Ошибки проведения и движений. При неправильном проведении возникают расхождения в остатках. Диагностика включает просмотр движений документа, проверку корректности измерений и ресурсов, анализ даты документа и периода движения. Следует убедиться, что движения очищаются перед формированием новых, что вид движения установлен корректно, а количество имеет правильный знак. Для контроля остатков лучше использовать виртуальные таблицы Остатки и Обороты в запросах.

Ошибки запросов. Типовые проблемы: неверные соединения, забытые параметры, несоответствие типов в условиях, выбор лишних полей. При диагностике полезно: временно упростить запрос до минимального работающего, затем

постепенно усложнять; выводить параметры, передаваемые в запрос; проверять имена объектов метаданных и полей. Для больших запросов важно помнить, что лишние поля увеличивают объём передаваемых данных.

Приёмы отладки. На практических работах достаточно освоить: установку точек останова; пошаговое выполнение; просмотр значений переменных; использование Сообщить для фиксации ключевых значений; анализ ОписаниеОшибки при исключениях. При систематическом подходе большинство ошибок устраняется быстро и становится источником понимания, а не фрустрации.

ПРИЛОЖЕНИЕ 4. КРАТКИЙ СПРАВОЧНИК ВСТРОЕННЫХ ФУНКЦИЙ И ПРИЁМОВ

Ниже приведены наиболее употребимые функции и приёмы, которые регулярно встречаются в учебных и практических задачах. Справочник не является заменой официальной документации, но помогает быстро сориентироваться при выполнении работ.

ЗначениеЗаполнено(Значение). Возвращает Истина, если значение считается заполненным (не пустая строка, не 0 для чисел и т. п.).

ТипЗнч(Значение). Возвращает объект типа значения, позволяющий сравнивать типы и выполнять проверки.

СокрЛП(Строка). Удаляет пробелы слева и справа; полезно для нормализации ввода.

ПустаяСтрока(Строка). Проверяет, является ли строка пустой (после учёта пробелов).

Лев(Строка, N). Возвращает N первых символов строки.

Прав(Строка, N). Возвращает N последних символов строки.

Сред(Строка, Начало, Длина). Возвращает подстроку; часто используется при разборе кодов.

Найти(Строка, Подстрока). Возвращает позицию подстроки; 0 означает «не найдено».

СтрЗаменить(Строка, Что, НаЧто). Заменяет подстроки; полезно для очистки ввода.

ВРег/НРег. Преобразует строку к верхнему/нижнему регистру.

Окр(Число, Точность). Округляет число до указанной точности.

Макс/Мин. Возвращает максимум или минимум из двух значений.

ТекущаяДата(). Возвращает текущую дату и время.

НачалоДня/КонецДня. Возвращают границы дня для формирования периодов.

ДобавитьМесяц/ДобавитьДень. Добавляют интервал к дате; удобны при расчёте периодов.

Формат(Значение, СтрокаФормата). Форматирует число или дату для вывода пользователю.

Сообщить(Текст). Выводит диагностическое сообщение; удобно в учебной отладке.

Предупреждение(Текст). Показывает предупреждение пользователю на клиенте.

ВызватьИсключение(Текст). Генерирует исключение и прерывает выполнение текущей ветки.

ОписаниеОшибки(). Возвращает текст последней ошибки в блоке Исключение.

Новый <Тип>. Создаёт новый объект (Структура, Массив, ТаблицаЗначений, Запрос и т. п.).

ТаблицаЗначений.Выгрузить(). Стандартный способ получить таблицу значений из результата запроса.

ТаблицаЗначений.Найти(). Поиск строки по значению колонки; ускоряется индексами.

ТаблицаЗначений.Сортировать(). Сортировка по указанным колонкам.

ТаблицаЗначений.Итоги. Механизм вычисления итогов по колонкам таблицы значений.

Запрос.УстановитьПараметр(). Установка параметра запроса, предпочтительнее конкатенации строк.

Выборка.Следующий(). Переход к следующей строке выборки.

Справочники.<Имя>.СоздатьЭлемент(). Создание элемента справочника объектным способом.

Документы.<Имя>.СоздатьДокумент(). Создание документа объектным способом.

Объект.Записать(). Запись объекта в информационную базу.

Документ.Провести(). Проведение документа, формирующее движения.

РегистрНакопления.<Имя>.Остатки(). Виртуальная таблица остатков в запросах.

ЕСТЬNULL(Поле, Значение). Возвращает значение по умолчанию при NULL; полезно при левом соединении.

ВЫБРАТЬ ПЕРВЫЕ N. Ограничение количества строк, полезно при проверках уникальности.

СГРУППИРОВАТЬ ПО. Группировка в запросе для агрегирования.

УПОРЯДОЧИТЬ ПО. Сортировка результата запроса.

ЛЕВОЕ СОЕДИНЕНИЕ. Получение всех строк левого источника с возможным отсутствием правых.

Отказ. Флаг отмены записи/проведения в соответствующих событиях.

Движения.<Регистр>. Коллекция движений, формируемых при проведении документа.

&НаКлиенте / &НаСервере. Директивы, определяющие контекст выполнения метода.

ПРИЛОЖЕНИЕ 5. КОНТРОЛЬНЫЕ ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

Контрольные вопросы предназначены для самостоятельной проверки понимания материала. Ответы рекомендуется формулировать письменно, опираясь на собственную учебную конфигурацию и выполненные задания. Вопросы сгруппированы по темам и предполагают не только воспроизведение определений, но и объяснение причин инженерных решений.

Поясните различие между платформой и конфигурацией. Какие задачи решаются на уровне платформы, а какие – на уровне конфигурации. Что такое метаданные и чем они отличаются от данных.

Опишите жизненный цикл документа: создание, заполнение, запись, проведение, отмена проведения. В каких событиях целесообразно выполнять проверки заполненности, а в каких – формировать движения регистров.

Что означает клиент-серверный контекст в управляемом приложении. Почему запросы и запись объектов следует выполнять на сервере. Какие типы данных удобно передавать между клиентом и сервером и почему.

Чем отличается значение Неопределено от пустой ссылки. Приведите примеры ситуаций, когда может возникать каждое из этих значений. Как корректно проверять заполненность ссылочного реквизита.

Опишите назначение справочников, документов, регистров накопления и регистров сведений. Почему регистр накопления удобен для получения остатков и оборотов. Что такое измерения и ресурсы и как их выбор влияет на аналитику.

Какие преимущества даёт параметризация запросов. Почему не рекомендуется формировать запросы конкатенацией строк. Что такое левое соединение и в каких отчётах оно особенно полезно.

Поясните роль системы компоновки данных. Что такое набор данных, ресурс и группировка в СКД. Почему даже при использовании СКД важно понимать, как устроен запрос.

Опишите типовые проверки в табличной части документа: заполненность номенклатуры, положительность количества и цены, пересчёт суммы. Где должна находиться логика проверок, чтобы её нельзя было обойти.

Поясните, как реализовать контроль отрицательных остатков при списании. Какой запрос следует выполнить, какие параметры передать и где разместить проверку. Какие сообщения об ошибке будут понятны пользователю.

Какие средства отладки предоставляет платформа. Как вы используете точки останова и просмотр переменных. Почему полезно фиксировать диагностические сообщения и какие данные в них следует включать.

Как организовать права доступа в учебной конфигурации. Какие права должны быть у роли «Пользователь» и у роли «Администратор». Какие проверки следует выполнять в коде, а какие достаточно настроить на уровне ролей.

Объясните, почему обработка больших объёмов данных в цикле языка 1С может приводить к низкой производительности. Какие операции лучше переносить в запрос. Какую роль играют индексы и отборы.

Сформулируйте требования к качеству учебного решения: корректность данных, понятность сообщений, структурированность кода, соответствие клиент-серверной архитектуре. Приведите примеры типичных анти-паттернов и объясните, чем они опасны.

ПРИЛОЖЕНИЕ 6. ОБРАЗЕЦ ВЫПОЛНЕНИЯ ЗАДАНИЯ: ДОКУМЕНТ «ПОСТУПЛЕНИЕ ТОВАРА»

Приложение демонстрирует целостный пример реализации одного учебного сценария. Цель примера – показать согласованность решений между метаданными, формой и модулем объекта. Пример не претендует на полноту промышленной конфигурации, однако отражает инженерно корректный подход: проверки и движения находятся в модуле объекта, форма выполняет интерфейсные действия и вызывает серверные расчёты при необходимости, запросы параметризованы, сообщения об ошибках ориентированы на пользователя.

Сценарий включает следующие элементы: справочники «Номенклатура» и «Склады»; документ «ПоступлениеТовара» с табличной частью «Товары»; регистр накопления «ОстаткиТоваров». Документ при проведении формирует приходные движения по количеству. В форме документа реализованы пересчёты суммы строки и общей суммы.

А. Структура метаданных

Документ «ПоступлениеТовара» содержит реквизиты шапки: «Склад» (ссылка на «Склады»), «Поставщик» (ссылка на «Контрагенты», если используется), «Комментарий» (строка). Табличная часть «Товары» содержит «Номенклатура» (ссылка), «Количество» (число), «Цена» (число), «Сумма» (число). Регистр накопления «ОстаткиТоваров» имеет измерения «Номенклатура» и «Склад» и ресурс «Количество».

Ключевой принцип: документ должен быть корректен сам по себе, то есть его запись и проведение должны быть невозможны при некорректных данных независимо от формы ввода. Поэтому проверки обязательных реквизитов и значений размещаются в модуле объекта.

Б. Модуль объекта документа: проверки перед записью

```
Процедура ПередЗаписью(Отказ, РежимЗаписи)

    Если Склад = Справочники.Склады.ПустаяСсылка() Тогда
        Отказ = Истина;
        Сообщить("Не заполнен склад.");
        Возврат;
    КонецЕсли;

    Если Товары.Количество() = 0 Тогда
        Отказ = Истина;
        Сообщить("Добавьте хотя бы одну строку в табличную часть «Товары».");
        Возврат;
    КонецЕсли;

    Итог = 0;

    Для каждого Стр Из Товары Цикл

        Если Стр.Номенклатура = Справочники.Номенклатура.ПустаяСсылка() Тогда
            Отказ = Истина;
            Сообщить("В табличной части есть строка без номенклатуры.");
            Возврат;
```

```
        КонецЕсли;

        Если Стр.Количество <= 0 Тогда
            Отказ = Истина;
            Сообщить("Количество должно быть больше нуля.");
            Возврат;
        КонецЕсли;

        Если Стр.Цена < 0 Тогда
            Отказ = Истина;
            Сообщить("Цена не может быть отрицательной.");
            Возврат;
        КонецЕсли;

        // Пересчёт суммы строки на сервере как гарантия корректности
        Стр.Сумма = Окр(Стр.Количество * Стр.Цена, 2);
        Итог = Итог + Стр.Сумма;

    КонецЦикла;

    // Общая сумма документа (если реквизит предусмотрен)
    СуммаДокумента = Окр(Итог, 2);

КонецПроцедуры
```

В. Модуль объекта документа: проведение и движения

```
Процедура ОбработкаПроведения(Отказ, РежимПроведения)

    Движения.ОстаткиТоваров.Очистить();

    Для каждого Стр Из Товары Цикл

        Движение = Движения.ОстаткиТоваров.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
        Движение.Номенклатура = Стр.Номенклатура;
        Движение.Склад = Склад;
        Движение.Количество = Стр.Количество;

    КонецЦикла;

КонецПроцедуры
```

Г. Модуль формы

Пересчёт на клиенте нужен для удобства пользователя, чтобы итоговые значения обновлялись сразу при вводе. Однако такой пересчёт не является гарантией корректности данных, поэтому аналогичный пересчёт выполняется в модуле объекта перед записью. В форме пересчёт можно реализовать в обработчиках изменения количества и цены.

```
&НаКлиенте
Процедура ТоварыКоличествоПриИзменении(Элемент)
    ПересчитатьСтрокуИтоги(Элемент.ТекущиеДанные);
```

```
КонецПроцедуры

&НаКлиенте
Процедура ТоварыЦенаПриИзменении(Элемент)
    ПересчитатьСтрокуИтоги(Элемент.ТекущиеДанные);
КонецПроцедуры

&НаКлиенте
Процедура ПересчитатьСтрокуИтоги(СтрокаТЧ)
    СтрокаТЧ.Сумма = Окр(СтрокаТЧ.Количество * СтрокаТЧ.Цена, 2);

    Итог = 0;
    Для каждого Стр Из Товары Цикл
        Итог = Итог + Стр.Сумма;
    КонецЦикла;

    СуммаДокумента = Окр(Итог, 2);
КонецПроцедуры
```

Д. Проверка результата через отчёт по остаткам

После записи и проведения документа рекомендуется проверить результат через отчёт «Остатки товаров на дату». Это обеспечивает замкнутый цикл: документ формирует движения, движения изменяют итоги регистра, отчёт получает итоги через виртуальную таблицу остатков. Если остатки не изменились, значит проведение не сформировало движений либо сформировало их с неверными измерениями или ресурсами.

```
Процедура КонтрольОстатков(ДатаСреза) Экспорт

    Запрос = Новый Запрос;
    Запрос.Текст =
    "ВЫБРАТЬ
    |   Ост.Номенклатура КАК Номенклатура,
    |   Ост.Склад КАК Склад,
    |   Ост.КоличествоОстаток КАК Остаток
    |ИЗ
    |   РегистрНакопления.ОстаткиТоваров.Остатки(&Дата) КАК Ост";

    Запрос.УстановитьПараметр("Дата", ДатаСреза);

    Таб = Запрос.Выполнить().Выгрузить();
    Для каждого Стр Из Таб Цикл
      Сообщить(Стр.Номенклатура + " / " + Стр.Склад + ": " + Стр.Остаток);
    КонецЦикла;

КонецПроцедуры
```

Учебное издание

Корчагин Павел Анатольевич, Корчагин Илья Павлович

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Часть 3. Методические указания к практическим занятиям по языку программирования 1С:Предприятие 8.3