Обзор системы 1С Предприятие

Учебная карта дисциплины

№	Виды контрольных мероприятий	Баллы	Рубежный контроль *(при наличии)*
1	Лабораторная работа 1	10	
2	Лабораторная работа 2	10	
3	Лабораторная работа 3	10	
6	Лабораторная работа 4	10	
7	Лабораторная работа 5	10	
8	Самостоятельная работа	10	
4	Письменный контрольный опрос по теории		40
	Бонусные баллы	**10**	Бонусные баллы начисляются за проявление академической активности на теоретических занятиях
	Промежуточная аттестация в форме зачета		

Тематический сайт по предмету:
https://poks-mop.nethouse.ru
Литература – любые книги Радченко и Ко по программированию 1С

Самостоятельная работа

Изучение интернет-курса

«Управление проектами средствами Microsoft Project»

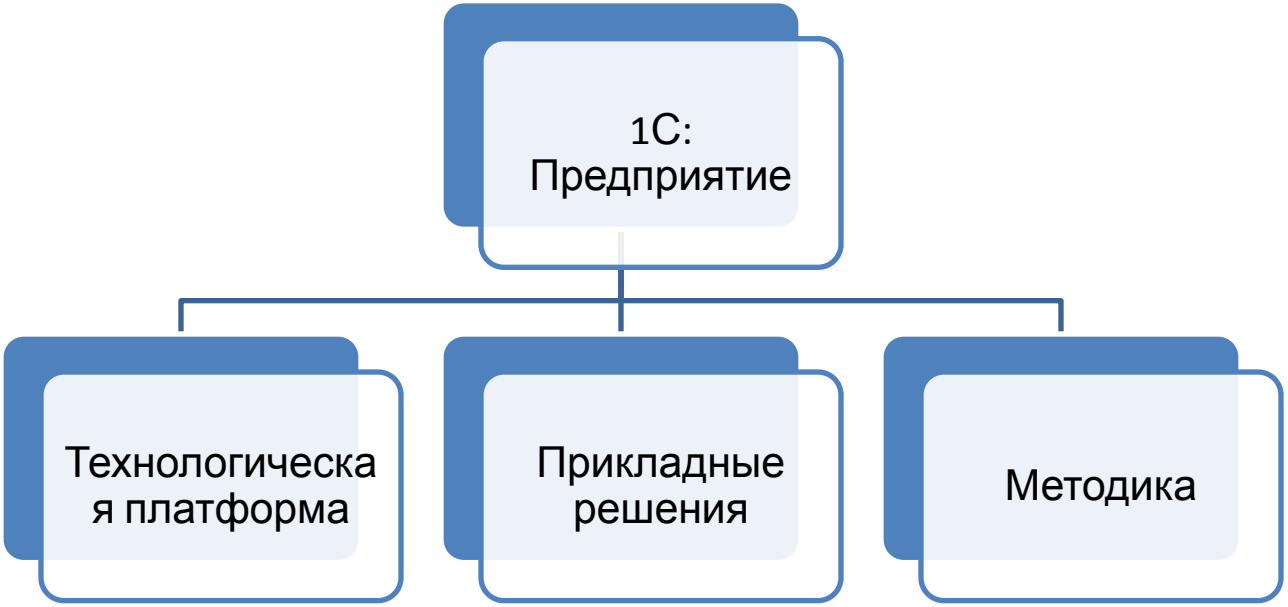
http://www.intuit.ru/studies/courses/496/352/info

и сдача интернет-экзамена по этому курсу

Идеология 1С:Предприятие

Система «1С:Предприятие» представляет собой совокупность трех составляющих:

- технологической платформы;
- прикладных решений различного масштаба и различной направленности, созданных на основе технологической платформы;
- методики создания прикладных решений.



Технологическая платформа

Платформа состоит из двух составляющих:
- среда исполнения,
- среда разработки.

Среда разработки.
- Называется **конфигуратором**.
- Процесс разработки называется **конфигурированием**.
- Прикладное решение называется **конфигурацией**.
- Используется технология метаданных.
- Метаданные - иерархическая структура объектов, полностью описывающая все прикладное решение. Эта структура называется «дерево объектов конфигурации».

Среда исполнения.
- Является интерпретатором.
- Режим исполнения принято называть "**режим Предприятия**".
- «исполняет» метаданные, аналогично тому, как операционная система исполняет код обычной программы.

Прикладное решение

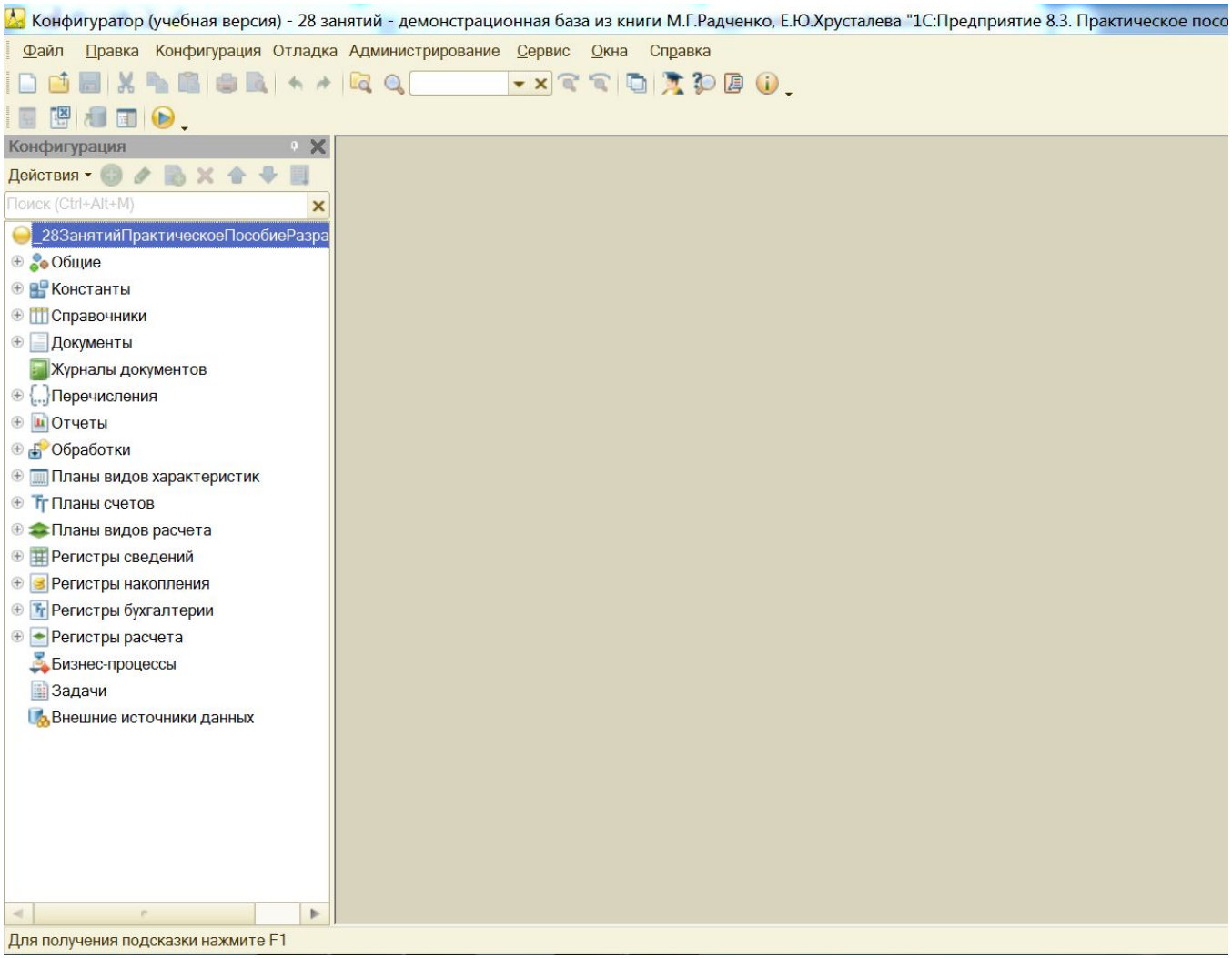
- Является самостоятельной сущностью.
- Может выступать в качестве отдельного программного продукта.
- Создание, модификация и собственно функционирование прикладного решения невозможны без использования технологической платформы.
- Как следствие, платформа поставляется с каждым прикладным решением.

Примеры прикладных решений:

- 1С:Бухгалтерия,
- 1С:Университет,
- и др.

Методика создания прикладных решений

- Все прикладное решение представляется в виде иерархической структуры объектов конфигурации.



Методика создания прикладных решений

- Разработчик использует встроенный язык для того, чтобы описать алгоритмы поведения объектов конфигурации в различные моменты исполнения прикладного решения.
- Для описания структуры прикладного решения разработчик использует не произвольные, а строго определенные объекты конфигурации.
- Платформа содержит ограниченный набор прототипов (шаблонов) объектов конфигурации. Например, шаблон справочника, документа, регистра накопления, бизнес-процесса и т. д.
- Каждый такой шаблон (прототип) содержит определенную базовую реализацию объекта конфигурации.
- Когда разработчик добавляет в дерево объектов конфигурации новый объект конфигурации, этот объект наследует базовую реализацию прототипа.
- Благодаря этому разработчик, не производя никаких дополнительных действий, тут же может запустить прикладное решение и работать с только что добавленным объектом. Базовая реализация объекта, унаследованная от прототипа (шаблона), обеспечит выполнение всех необходимых типовых действий.

Объекты конфигурации

Константы

- Используются для работы с постоянной и условно постоянной.
- Информация, хранящаяся и константах, редко изменяется, но, как правило, часто используется в работе.
- Примеры констант: наименование предприятия, адрес, телефон, ИНН предприятия, фамилии директора и главного бухгалтера и т.п.

Справочники

- Используются для работы с постоянной и условно постоянной информацией, представленной некоторым множеством значений.
- Пользователь может изменять (добавлять, удалять, редактировать) элементы множества значений.
- Используются для создания базы данных об объектах операционной деятельности.
- Примеры справочников: списки материалов, товаров, организации, валют, сотрудников и др.

Объекты конфигурации

Перечисления

- Используются для описания постоянных наборов значений, не изменяемых в процессе работы конфигурации.
- Значения перечислений задаются на этапе конфигурирования и не могут быть изменены на этапе исполнения.
- Пример перечисления: пол – мужской, женский.

Документы

- Предназначены для отражения хозяйственных событии предприятия, которые имеют отношение к автоматизируемой предметной области.
- При конфигурировании создается произвольное количество видов документов.
- Каждый вид документа предназначен для отражения своею вида событий. Это событие определяет структуру, свойства и функциональность документа.
- Примеры видов документов: *Платежное поручение*, *Счет*, *Приходная накладная*, *Расходная накладная*, *Накладная на внутреннее перемещение*, *Приходный кассовый ордер* и др.

Объекты конфигурации

Журналы документов.

- Предназначены для регистрации и просмотра документов разных видов.
- Каждый вид документа может быть показан в нескольких журналах.
- Журнал документов не добавляет новые данные в систему, а является средством для отображения в едином списке зарегистрированных в нем документов нескольких видов.
- Например, журнал *Складские документы* отображает все приходные и расходные накладные и накладные на внутреннее перемещение.

Отчеты и обработки

- Отчеты предназначены для формирования печатных или экранных форм, отображающих накопленную в информационной базе информацию.
- Обработки реализуют вспомогательные алгоритмы обработки данных информационной базы.
- Система поддерживает возможность разработки внешних отчетов и обработок, хранящихся не в самой конфигурации, а в отдельных файлах.

Объекты конфигурации

Регистры

- Предназначены для хранения и обработки различной информации, отражающей хозяйственную или организационную деятельность предприятия.
- В регистрах хранится информация об изменении состояний объектов операционной деятельности. Например, наличие материалов, задлженность перед контрагентом и т.п.
- В регистрах может храниться информация, не отражающая непосредственно объекты операционной деятельности. Например, курсы валют, данные о приходе и расходе товаров.

Существует 4 вида регистров:

- регистры сведений,
- регистры накопления,
- регистры расчетов,
- регистры бухгалтерии.

Технологические средства. Командный интерфейс

Цель командного интерфейса - обеспечить структурированный доступ пользователей к той информации, которая необходима им в соответствии с их обязанностями.

- Это основное средство навигации пользователя по функциональности конфигурации.
- Строится на основе подсистем.
- Разработчик включает прикладные объекты в соответствующие подсистемы.
- На основе структуры подсистем и привязки объектов к подсистемам система автоматически строит командный интерфейс для пользователя.
- Пользователю отображается структура прикладного решения (иерархия подсистем) и предоставляются стандартные команды доступа к функциональности прикладных объектов (вызов списков справочников, документов, открытие отчетов, обработок и т. д.).
- Разработчик может отредактировать предлагаемое системой построение командного интерфейса (изменить порядок, видимость команд). Для этого предназначен редактор командного интерфейса.
- Сами команды, включаемые в командный интерфейс (открытие списков, ввод новых объектов, открытие отчетов и т. д.). создаются системой автоматически.
- Разработчик может создать свои команды, которые будут включаться в

Технологические средства. Форма

Форма – это совокупность экранного диалога, модуля, реквизитов и команд.

- Большинство объектов конфигурации могут иметь визуальную форму.
- В общем случае форма состоит из следующих частей:
  - экранный диалог, используемый для ввода и редактирования информации;
  - модуль формы – программа на встроенном языке системы. Выполняет обработку вводимой в диалоге информации для целей контроля данных, выполнения расчетов и т. д.;
  - список реквизитов;
  - команды, используемые в форме.
- С помощью формы реализуется интерактивное взаимодействие прикладного объекта с пользователем. Характер такого взаимодействия задается разработчиком конфигурации.
- Для разработки форм применяется редактор форм, позволяющий редактировать все компоненты формы во взаимосвязи.

Технологические средства. Модуль

Модулем называется программа на встроенном языке системы «1С: Предприятие».

- Модули располагаются в заданных точках структуры конфигурации и вызываются для выполнения в заранее известные моменты работы системы, называемые событиями.
- Разработчик использует модули для описания сложных алгоритмов взаимодействия объектов конфигурации, для которых недостаточно имеющихся в конфигураторе визуальных средств.
- В конфигурации существует несколько видов модулей:
  - модуль управляемого приложения,
  - модуль внешнего соединения,
  - модуль сеанса,
  - общие модули,
  - модули форм,
  - модули объектов конфигурации.

Технологические средства. Файловый вариант

1С:Предпрпятие поддерживает два варианта работы:

- Файловый,
- Клиент-серверный.

- Файловый вариант рассчитан на персональную работу одного пользователя или работу небольшого количества пользователей в локальной сети (5-7).
- Все данные информационной базы (конфигурация, база данных, административная информация) располагаются в одном файле.
- Обеспечивает легкость установки и эксплуатации автоматизированной системы.
- Для работы с информационной базой не требуются дополнительные программные средства, достаточно иметь операционную систему и 1С: Предприятие.

Технологические средства. Клиент-серверный вариант

Клиент-серверный вариант предназначен для использования в рабочих группах или в масштабе предприятия.

- Реализован на основе трехуровневой архитектуры «клиент-сервер».
- Клиентское приложение взаимодействует с кластером серверов 1С: Предприятие.
- Кластер взаимодействует с сервером баз данных (Microsoft SQL Server. PostgreSQL. IBM DB2 иди Oracle Database).
- Физически кластер серверов 1С:Предприятие и сервер баз данных могут располагаться как на одном компьютере, так и на разных. Это позволяет администратору при необходимости распределять нагрузку между серверами.

Существует 3 вида клиентских приложений:

- толстый клиент;
- тонкий клиент;
- веб-клиент.

Технологические средства. Виды клиентских приложений

Толстый клиент

- Позволяет реализовывать полные возможности 1С:Предприятия как в плане разработки, администрирования, так и в плане исполнения прикладного кода.
- Не поддерживает работу с информационными базами через интернет.
- Требует предварительной установки на компьютер пользователя и имеет максимальный объем дистрибутива.

Тонкий клиент

- Не позволяет разрабатывать и администрировать прикладные решения,
- Может работать с информационными базами через интернет.
- Требует предварительной установки на компьютер пользователя, но имеет значительно меньший размер дистрибутива, чем толстый клиент.

Веб-клиент

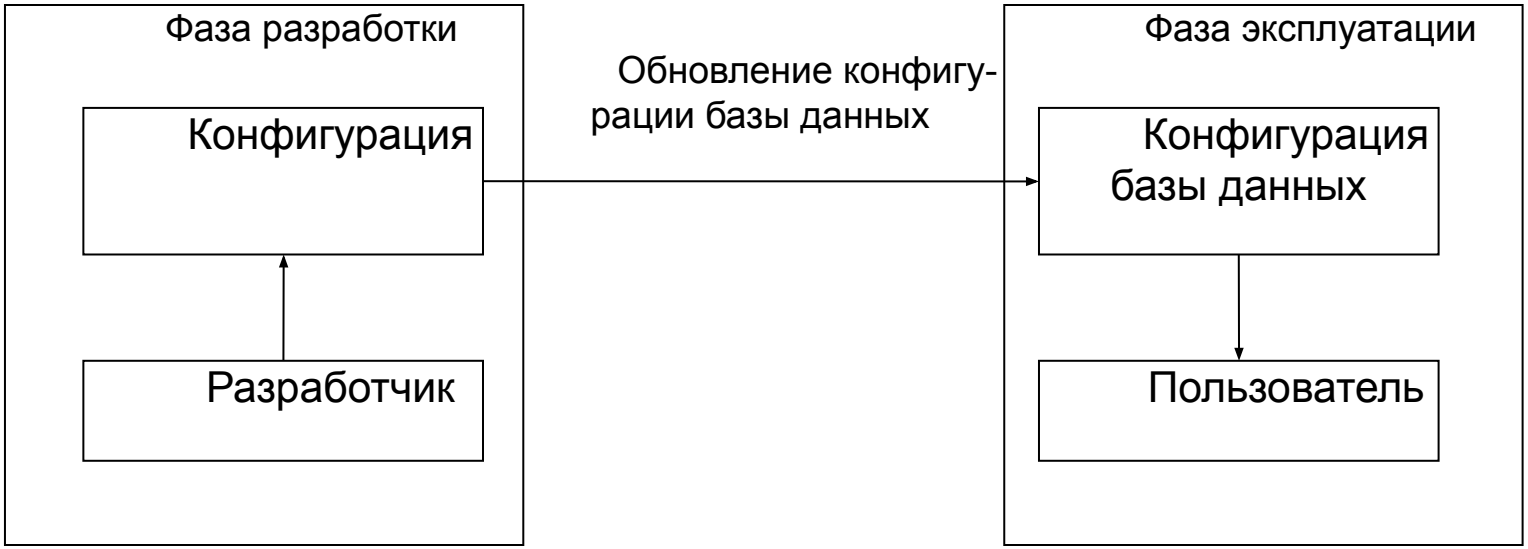
- Не требует какой-либо предварительной установки на компьютер.
- В отличие от толстого и тонкого клиентов, исполняется не в среде операционной системы компьютера, а в среде интернет-браузера.

Виды конфигураций

В системе используется два вида конфигураций:

- **конфигурация-** это инструмент разработчика, при помощи которого он создает прикладную систему,
- **конфигурация базы данных-** это инструмент пользователя, в котором он работает и реализует различные виды учета.

Цель разделения конфигураций - обеспечение возможности одновременной разработки конфигурации и ее эксплуатации.

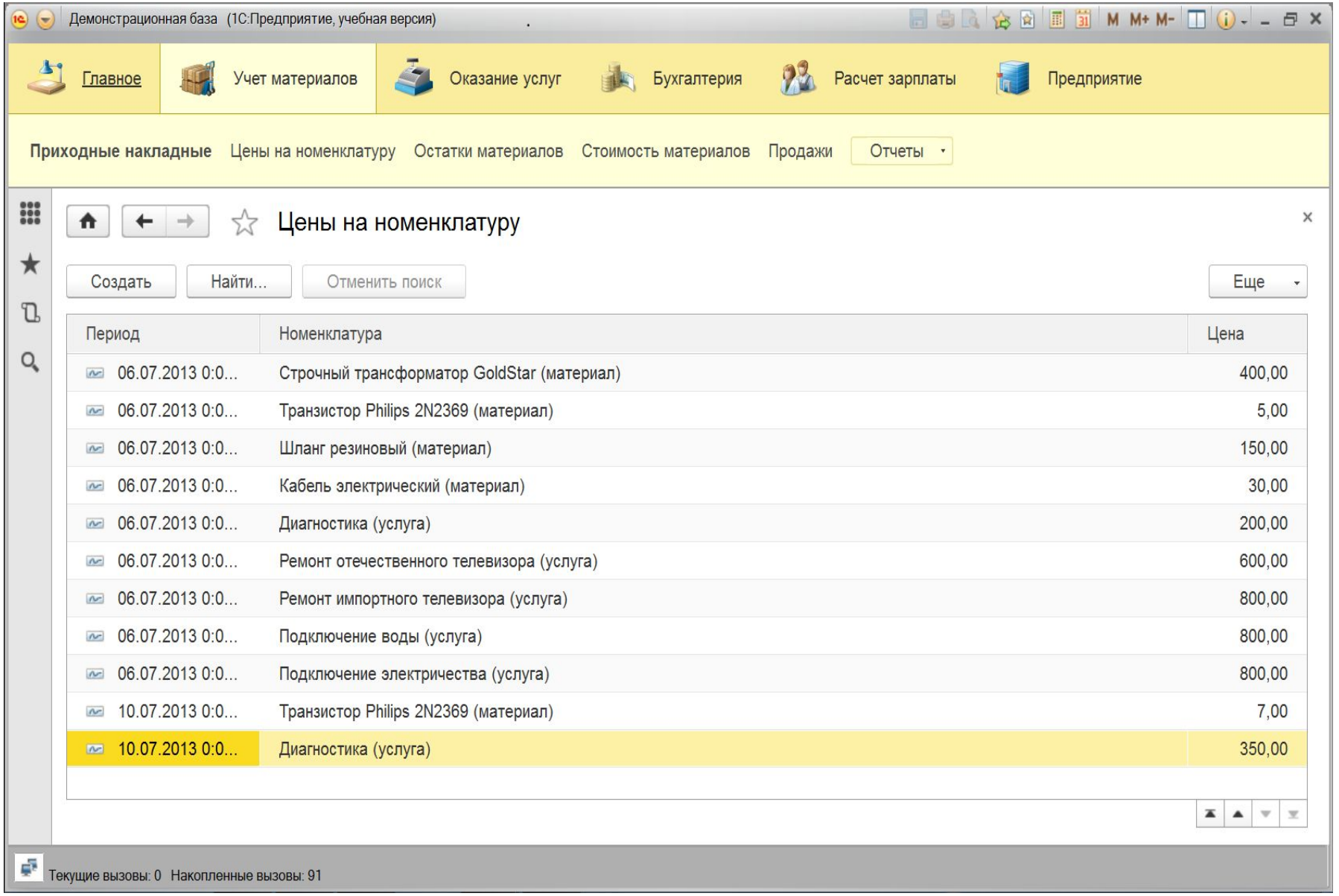


Взаимосвязь этих двух конфигураций реализуется при помощи операции ***Обновление конфигурации базы данных***, после которой доработанная программистом система загружается пользователю.

Интерфейс приложения

- Приложение представляется пользователю в виде иерархии, которая формируется подсистемами и входящими в них объектами конфигурации.
- Функционал приложения реализуется при помощи 2 видов окон:
    - Основное окно;
    - Вспомогательное окно.
- **Основное окно** предназначено для навигации но конфигурации и вызова различных команд.
- **Вспомогательное окно** открывается для выполнения какого-либо действия, а не для навигации по всему приложению в целом. В таких окнах открываются формы документов или элементов справочников, отчетов и обработок.

Основное окно приложения



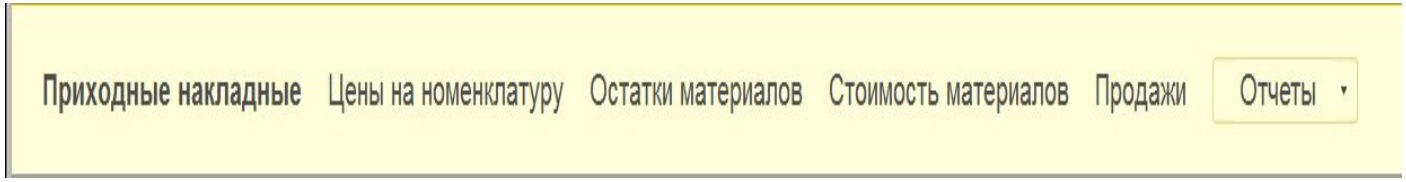
Элементы основного окна. Панель системных команд

- Содержит набор служебных команд (вызов меню, назад, вперед, калькулятор, календарь, панель быстрого доступа и т.п.).
- Не может быть удалена в настройках интерфейса.
- Особое место в ней занимает кнопка вызова главного меню приложения, при помощи которой доступны все функции прикладного решения, в том числе и те, которые не включены в набор реализованных окон.

Элементы основного окна. Панель разделов

- Показывает список подсистем верхнего уровня и позволяет быстро переключаться между ними.
- Первым разделом всегда является Главное, на который обычно помещаются самые необходимые и часто используемые инструменты приложения.
- Список подсистем может быть отображен в виде картинок, текста или картинок вместе с текстом.

Элементы основного окна. Панель функций текущего раздела



- Отображает структуру конфигурации, соответствующую подсистеме, выбранной в панели разделов.
- Если у подсистемы есть подчиненные подсистемы, они будут изображаться как группы.
- В этой панели существует три предопределенные группы: Важное, Обычное, См.также. Программист может создать свои группы, которые будут располагаться перед См.также.
- Выбор какой-либо гиперссылки этой панели приводит к открытию некоторой формы (например, журнал документов, справочник и т.п.).
- Данная панель может отображаться либо сокращенно в виде панели, либо развернуто по всей рабочей области.

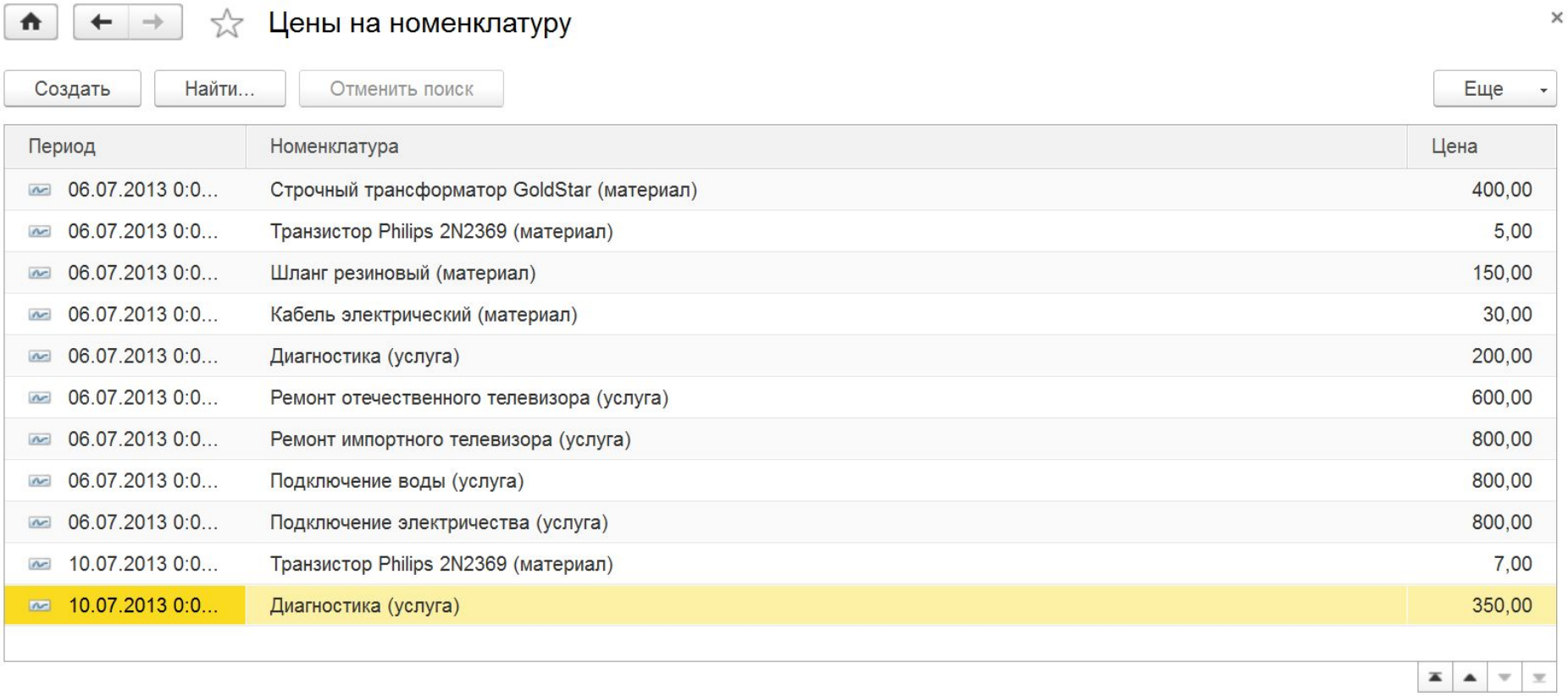
Элементы основного окна. Панель инструментов

Содержит кнопки для открытия других панелей, не отображенных на экране в текущий момент:

- меню функций – открыть развернутый вид панели функций текущего раздела;
- избранное – открыть панель избранного;
- история – открыть панель истории;
- поиск – открыть панель поиска.

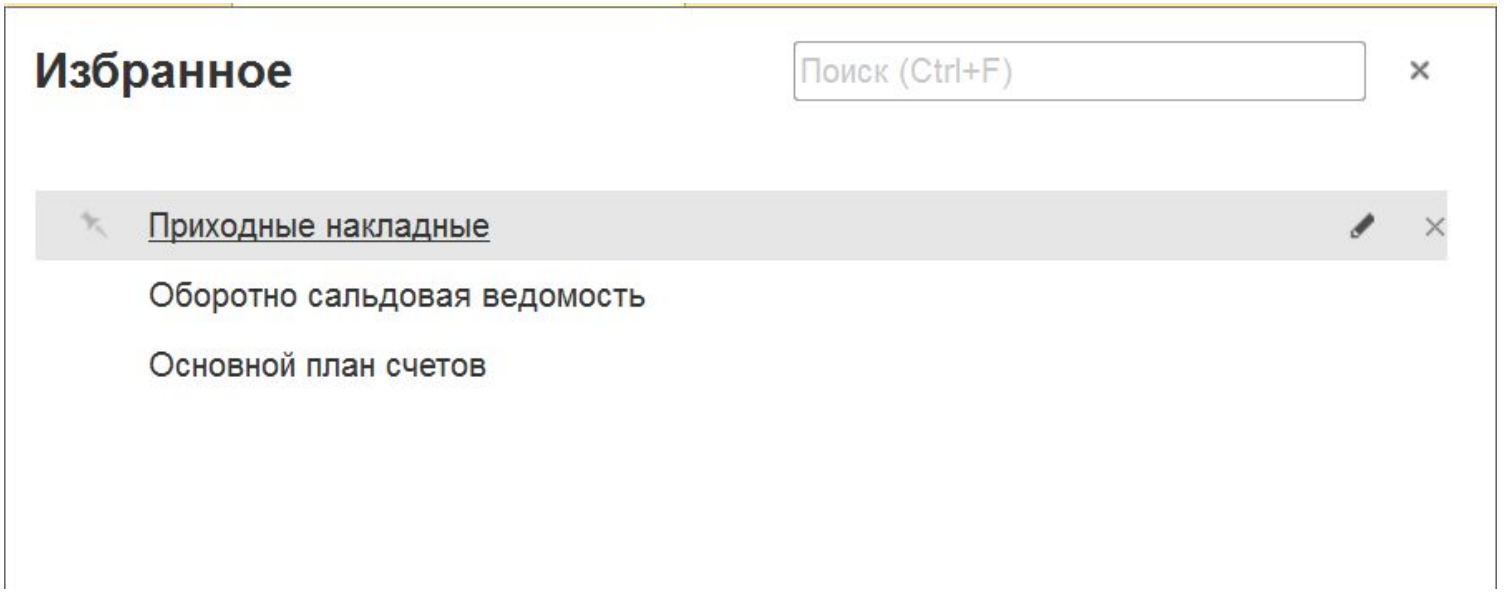
Элементы основного окна. Рабочая область

- Занимает большую часть экрана и содержит открытое в ней вспомогательное окно, предназначенное для работы с каким-либо объектом системы.



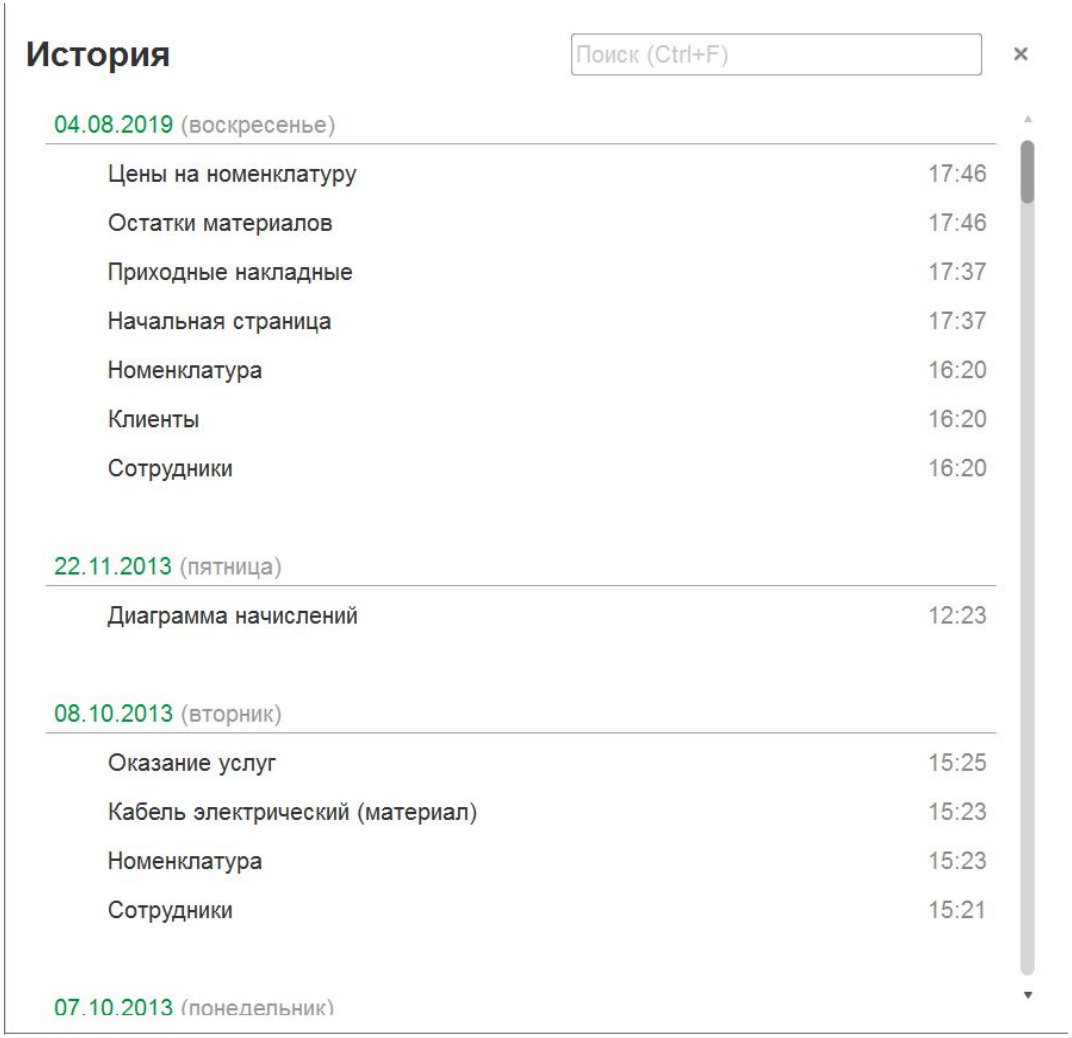
Элементы основного окна. Панель избранного

- Содержит названия окон, включенных в избранное, для быстрого перехода к нужному окну. Для включения окна в избранное, нужно открыть окно и нажать звездочку рядом с его названием.



Элементы основного окна. Панель истории

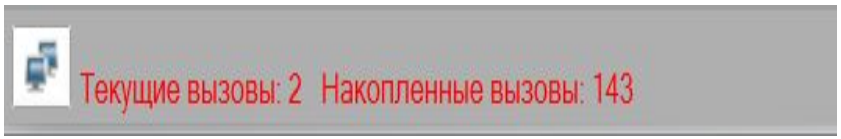
- Содержит историю открытия окон и позволяет быстро вернуться к ранее открытому окну.



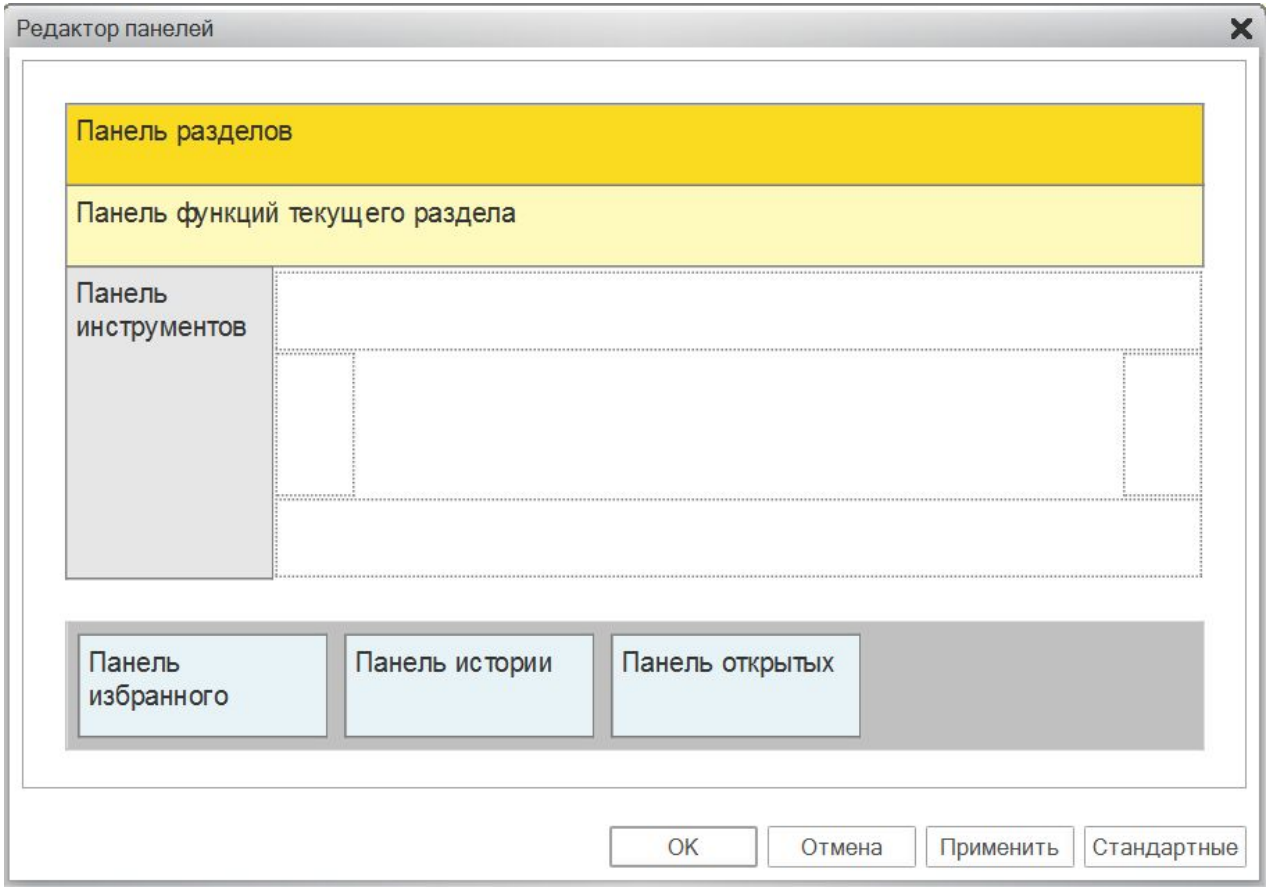
Элементы основного окна. Прочие панели

- Панель открытых содержит вкладки ранее открытых окон, которые не были закрыты и не видны в текущий момент. Эта панель позволяет быстро переместиться в одно из этих окон простым кликом по соответствующей вкладке.
- Панель поиска предназначена для реализации поиска по прикладному решению.

- Информационная панель расположена в самом низу основного окна и используется для вывода уведомлений



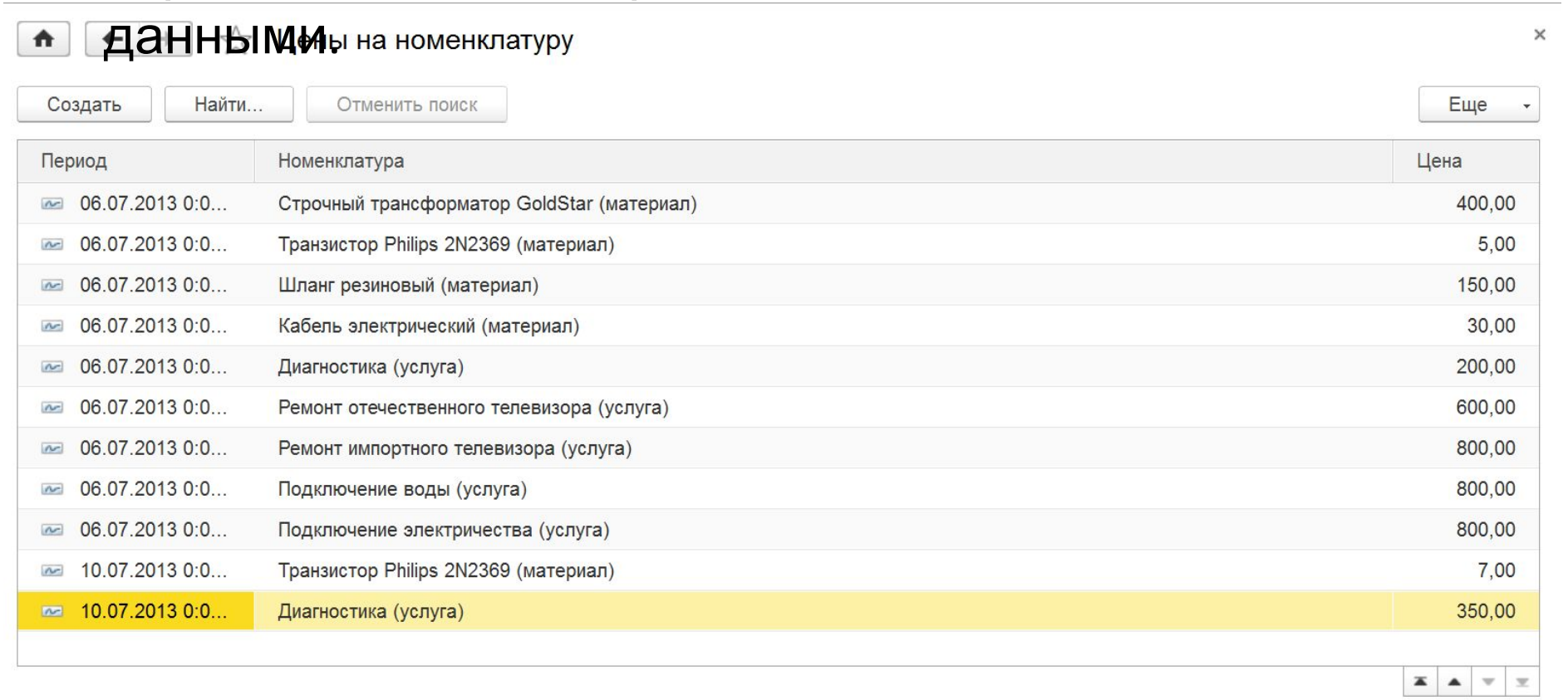
Редактор панелей



- Настройка панелей, соответствующая рисунку позволяет отобразить панели разделов, функций текущего раздела, инструментов. При этом панели избранного, истории и открытых будут скрыты. А панель поиска скрыта всегда при любой настройке редактора панелей.

Вспомогательное окно

- Отображается в рабочей области основного окна и предназначено для отображения и обработки некоторого объекта конфигурации.
- Состоит собственно из данных открытого объекта и перечня команд, которые можно выполнить над этими данными.



Встроенный язык 1С:Предприятие

Структура программного модуля

- Программный модуль состоит из 3 последовательных разделов;
- Каждый раздел может быть опущен;
- Последовательность разделов должна быть неизменна в любых случаях:
    - Раздел описания переменных.
    - Раздел описания подпрограмм.
    - Раздел основной программы.

Раздел описания переменных

ПЕРЕМ идентификатор1 [Экспорт], инетификатор2 …;

- Экспорт указывается для переменных, которые нужно экспортировать в другие модули, если текущий модуль позволяет экспортировать переменные.
- Тип переменной не задается при её описании. Он устанавливается по типу первого присваиваемого значения оператором присваивания.

Пример

ПЕРЕМ Сум, Глоб Экспорт; // Комментарий

По области видимости можно выделить несколько видов переменных:

- **Глобальные переменные** – описываются в модуле приложения с ключевым словом Экспорт. К ним можно будет обращаться из любых мест программы.
- **Глобальные переменные объекта** – описываются в модуле объекта с ключевым словом Экспорт. Доступ к этим переменным выполняется через конкретный объект.
- **Локальные переменные модуля** – переменные, описанные в разделе описания переменных этого модуля. Область видимости – модуль, в котором они определены.
- **Локальные переменные подпрограмм** – они видны только в своей

Раздел описания подпрограмм

Содержит описания процедур и функций.

```
ПРОЦЕДУРА Имя ([список параметров]) [Экспорт]
    // Описание локальных переменных
    // Операторы
    // Возврат;
    // Операторы
КОНЕЦПРОЦЕДУРЫ
```

- **Экспорт** – задает глобальную процедуру, которую можно вызывать из других модулей, если текущий модуль допускает описание таких процедур.
- Параметры в списке формальных параметров могут быть 2 видов:
  - передаваемые по значению;
  - передаваемые по ссылке (используется по умолчанию).
- ВОЗВРАТ – оператор, прерывающий выполнение подпрограммы.

- Описание параметров:

[ЗНАЧ] Имя_параметра [=ЗНАЧЕНИЕ],…

ЗНАЧ указывает на то, что параметр будет передаваться по значению.

=ЗНАЧЕНИЕ устанавливает значение параметра по умолчанию, присваивается в случае, если при вызове подпрограммы этот параметр

Раздел описания подпрограмм

```
ФУНКЦИЯ Имя_Функции ([список параметров]) [Экспорт]
    // Описание переменных
    // Операторы
    // Возврат значение;
    // Операторы
КОНЕЦФУНКЦИИ
```

Возврат значения функции выполняется оператором ВОЗВРАТ с указанием возвращаемого значения:

```
ВОЗВРАТ значение;
```

Раздел основной программы

Последовательность операторов, которая будет выполнена при активизации объекта, с которым связан модуль или при запуске системы 1С:Предприятие для модулей, относящихся к приложению в целом (модули приложения, сеанса, внешнего соединения).

Виды данных информационной базы

Все данные, которые хранятся в информационной базе, можно разделить на две категории:

- объектные,
- необъектные.

Объектные данные: справочники, документы, планы видов характеристик, планы счетов, планы видов расчета, бизнес-процессы, задачи, планы обмена.

- Состоят из отдельных объектов.
- Каждый из объектов обладает внутренним уникальным идентификатором, благодаря которому к объекту можно обращаться как к единому целому.
- Каждый объект имеет определенную значимость сам по себе: удаление объекта и создание точно такого же приводит к появлению другого объекта с другим идентификатором.

Необъектные данные: все виды регистров, константы, последовательности.

- Не имеют собственной ценности и полностью описываются значениями своих полей.
- Представляют собой записи, для которых не поддерживаются внутренние уникальные идентификаторы.
- Удалив некоторую запись и создав новую с точно такими же значениями полей, мы получим то же самое состояние базы данных, которое было до

Объектные данные

Для работы с объектными данными существуют два основных типа: ссылка и объект.

Ссылка

- Используется везде, где требуется однозначно идентифицировать объект базы данных.
- Фактически является значением внутреннего идентификатора.
- Ссылки можно сравнивать между собой. Две ссылки считаются равными, если они содержат один и тот же идентификатор одного и того же вида объекта.
- Может быть использована для доступа к данным объекта только на чтение.

Объект

- Предназначен для модификации (чтения и изменения) данных, содержащихся в объекте базы данных. Остальные объекты встроенного языка позволяют только читать информацию базы данных.
- Используется при создании новых объектов, для редактирования и удаления существующих объектов, а также для отображения и редактирования всех данных объекта в форме объекта.

Необъектные данные

- Представляют собой некоторый набор записей, которые хранятся в таблице.
- Каждая из этих записей полностью описывается значениями своих полей. Запись можно удалить, а затем создать новую, с такими же значениями полей.

Для чтения, модификации и удаления необъектных данных используется тип данных ***Набор записей***.

- Набор записей представляет собой коллекцию отдельных записей, принадлежащих некоторому объекту конфигурации.
- Для наборов записей не существует понятия удаления. Набор записей можно только записать, причем запись может быть выполнена либо с замещением существующих записей либо с добавлением новых записей к существующим.
- Для удаления всех записей нужно записать новый пустой набор записей.

Типы данных встроенного языка

Все типы данных системы можно разделить на 3 категории:

- **примитивные типы-** простые типы, для которых можно задать константу;
- **типы, образуемые в прикладном решении-** соответствуют объектам, созданным разработчиком в дереве конфигурации, предназначенные для работы с этими объектами средствами встроенного языка;
- **типы, предопределенные во встроенном языке-** объекты сложной структуры, создаваемые и обрабатываемые исключительно средствами встроенного языка (список значений, таблица значений, массив, структура).

Примитивные типы

Null - пустая ссылка на объект базы данных. Константа – Null.
Булевский - имеет 2 значения: *Истина* и *Ложь*, которые и являются его константами. Операции – сравнение (= <> < <= > >=), логические И, ИЛИ, НЕ. Порядок операций – НЕ, И, ИЛИ.
Числовой – число, в общем случае с дробной частью. Записывается в виде целого, либо в виде десятичного с фиксированной точкой: 50 или 50.5. Чисел с плавающей точкой в системе нет.
Операции: +, -, *, /, % (остаток от деления), сравнение (<, <=, >, >=, =, <>).
Строковый – строка символов в формате Unicode. Константа этого типа заключается в двойные кавычки, например “Пример константы строки”.
Операции: + (конкатенация), сравнение (<, <=, >, >=, =, <>).
Дата – значение даты от рождества Христова и время с точностью до секунды, в формате: ‘ГГГГММДДччммсс’. Константа этого типа указывается в одинарных кавычках с разделителями или без них: '25.10.08 10:15:25' или '25102008101525'. В значении даты время можно опустить – оно будет равно 00:00:00.
Операции: сложение даты и числа, разность даты и числа, разность дат = количество секунд между этими датами, операции сложения дат НЕТ!, сравнение дат.
Неопределено - пустое значение, не принадлежащее ни одному типу. Имеет единственное значение Неопределено.

Типы, образуемые в прикладном решении

- Создаются в конкретном прикладном решении в результате добавления в конфигурацию какого-либо объекта метаданных.
- Создаются платформой автоматически и позволяют работать с данными, хранящимися в тех структурах, которые описываются данным объектом конфигурации.

Объект	Тип данных						
	Менеджер	Выборка	Ссылка	Объект	Набор значений	Запись	Ключ записи
Константа	+						
Журнал документов	+	+					
Перечисление	+		+				
Справочник	+	+	+	+			
Документ	+	+	+	+			
Бизнес-процесс	+	+	+	+			
Задача	+	+	+	+			
Отчет	+			+			
Обработка	+			+			
Последовательность	+				+	+	
Регистр сведений	+	+			+	+	+
Регистр накопления	+	+			+	+	+

Типы, образуемые в прикладном решении

Пример

При добавлении в дереве конфигурации справочника Номенклатура, в приложении будут созданы типы данных:

- СправочникМенеджер.Номенклатура
- СправочникСсылка.Номенклатура
- СправочникОбъект.Номенклатура
- СправочникВыборка.Номенклатура.

Типы данных встроенного языка. Массив

- Является простейшей коллекцией значений.
- Можно описать многомерные массивы, массивы с фиксированной длиной и динамические массивы.

Работа со статическим массивом:

```
Массив = Новый Массив(99);
Для сч = 0 По Массив.Количество() – 1 Цикл
Сообщить(Массив[сч]);
КонецЦикла;
```

Работа с динамическим массивом:

```
Массив = Новый Массив();
 Массив.Добавить(1);
 Массив.Добавить(2);
Массив.Добавить(3);
…
Для сч = 0 По Массив.Количество() – 1 Цикл
Сообщить(Массив[сч]);
КонецЦикла;
```

Описание многомерного массива:

```
МногомерныйМассив = Новый Массив (2, 4, 10);
```

Типы данных встроенного языка. Список значений

Это коллекция значений, представляющая таблицу с фиксированным набором колонок (полей):

- значение – обязательно для заполнения, собственно значение в списке,
- представление – текстовое представление значения для отображения в интерфейсе,
- пометка – значение булевского типа,
- картинка – значение типа картинка, изображающее элемент.

Все поля, кроме значения, необязательны.

Пример. Составить список всех имеющихся в конфигурации документов и вывести пользователю их имена и синонимы.

```
СЗ = Новый СписокЗначений;
//На всякий случай сразу ее очищаем
СЗ.Очистить();
//Начинаем перебирать все имеющиеся документы
Для Каждого Документ Из Метаданные.Документы Цикл
//Добавляем информацию об очередном документе в нашу переменную
 СЗ.Добавить(Документ.Имя, Документ.Синоним);
КонецЦикла;
//Теперь выводим все содержимое нашего списка значений
Для сч = 0 по СЗ.Количество() - 1 Цикл
Сообщить("Представление: " + СЗ[сч].Представление + "; Значение: " + СЗ[сч].
Значение);
КонецЦикла;
```

Типы данных встроенного языка. Таблица значений

Этот тип обладает следующими возможностями:

- Колонки – можно создать сколько угодно колонок и использовать их в своих целях. Каждая колонка может быть как определенного типа, так и произвольного.
- Итоги – для числовых колонок имеется возможность подсчета итогов по колонке.
- Сортировка строк.
- Поиск строк.
- Группировка по значению (Свертывание).

Пример работы с таблицей значений.

```
//Создаем таблицу значений
ТЗ = Новый ТаблицаЗначений;
//Определяем колонки
 ТЗ.Колонки.Добавить("Товар");
 ТЗ.Колонки.Добавить("Количество");
 ТЗ.Колонки.Добавить("Сумма");
```

Типы данных встроенного языка. Таблица значений

```
//Заполняем таблицу значений данными
НоваяСтрока = ТЗ.Добавить();
//Записываем данные в каждую колонку
НоваяСтрока.Товар = "Огурец";
НоваяСтрока.Количество = 5;
НоваяСтрока.Сумма = 25;
НоваяСтрока = ТЗ.Добавить();
НоваяСтрока.Товар = "Морковь";
НоваяСтрока.Количество = 10;
НоваяСтрока.Сумма = 15.5;
НоваяСтрока = ТЗ.Добавить();
НоваяСтрока.Товар = "Огурец";
НоваяСтрока.Количество = 9;
НоваяСтрока.Сумма = 45;
//Перебираем строки ТЗ и выводим содержимое в окно служебных сообщений
Для Каждого СтрокаТЗ Из ТЗ Цикл
Сообщить("Строка таблицы значений: ");
Сообщить("");
Сообщить("Товар: " + СтрокаТЗ.Товар);
Сообщить("Количество: " + СтрокаТЗ.Количество);
Сообщить("Сумма: " + СтрокаТЗ.Сумма);
Сообщить("------------------");
КонецЦикла;
```

Типы данных встроенного языка. Таблица значений

```
//Посчитаем итоговую сумму по всем строкам
Сообщить("Всего товаров на сумму: " + ТЗ.Итог("Сумма"));
Сообщить("Общее количество: " + ТЗ.Итог("Количество"));

//Теперь посчитаем количество и суму за огорцы
ТЗ.Свернуть("Товар", "Количество, Сумма");

//Теперь после сворачивания строка с огурцами будет одна, а не две
```

Типы данных встроенного языка. Структура

Это коллекция, состоящая из пар: ключ – значение.

- **Ключ** может быть только строковым и должен удовлетворять требованиям, предъявляемым к именованию переменных встроенного языка.
- **Значение** – значение любого типа.
- Используется для хранения небольшого количества значений, каждое из которых имеет свое имя.
- Используется для задания **отбора** (фильтра) по заданным значениям полей.

```
Струк = Новый Структура:
Струк.Добавить(«Товар», ССылкаНаТовар);
Струк.Добавить(«Склад», ССылкаНаСклад);
Для сч = 0 Струк.Количество() - 1 Цикл
      Сообщить(«Товар: » + Струк.Товар + « Склад: » + Струк.Склад)
КонецЦикла;
```

Оператор присваивания

Назначение = Источник

- Назначение – переменная или свойство объекта встроенного языка, которое допускает запись.
- Источник – выражение, значение которого необходимо присвоить.
- Первое присваивание значения некоторой переменной выполняет роль ее объявления в программе и задает ее тип по типу присвоенного значения.
- В правой части оператора присваивания можно использовать условное выражение:

?(ЛогическоеВыражение, Выражение1,Выражение2)

ЛогическоеВыражение – любое выражение, результат которого имеет булевский тип, например, сравнение.
Выражение1 – вычисляется, если ЛогическоеВыражение истинно,
Выражение2 – вычисляется, если ЛогическоеВыражение ложно.

Пример.

```
Статус = ?(ПолучитьСкидку() > 10, “Особый клиент”, “Обычный клиент”);
```

Условный оператор

```
Если ЛогическоеВыражение1 Тогда
    // Операторы
[ИначеЕсли ЛогическоеВыражение2 Тогда]
    // Операторы
[Иначе]
    // Операторы
КонецЕсли;
```

- ЛогическоеВыражение – любое выражение булевского типа.
- Операторы, следующие за *Тогда* выполняются, если *ЛогВыражение1* истинно.
- *ЛогическоеВыражение2* вычисляется, если *ЛогическоеВыражение1* оказалось ложным.
- При истинности *ЛогическогоВыражения2* выполняются операторы, следующие за соответствующей ему *Тогда*-частью.
- Конструкций *ИначеЕсли* может быть неограниченное количество или они могут отсутствовать.
- Операторы, следующие за *Иначе*, выполняются, когда все предшествующие *ЛогическиеВыражения* имели значение *Ложь*.
- Иначе-часть может быть только одна или вообще отсутствовать.

Условный оператор. Пример

```
Если ДеньНедели(ТекущаяДата()) = 6 Тогда
    Сообщить(“Сегодня суббота”);
ИначеЕсли ДеньНедели(ТекущаяДата()) = 7 Тогда
    Сообщить(“Сегодня воскресенье”);
Иначе
    Сообщить(“Сегодня рабочий день”);
КонецЕсли;
```

Оператор перехода

Перейти Метка;

- Передача управления на оператор, помеченный меткой.
- Метка– идентификатор.
- Пометка оператора имеет вид:
  ~Метка: оператор;
- Передача управления может выполняться только внутри программного блока (процедуры или функции).
- Переход не может быть выполнен на операторы, расположенные внутри синтаксических конструкций: условный оператор, циклы, Попытка.

Пример.

```
Перейти Метка1;
. . .
~Метка1: Сообщить(“Выполнен переход по метке1”);
```

Цикл с параметром

```
Для Переменная=НачЗначение По КонЗначение Цикл
    // Операторы
    Прервать;
    // Операторы
    Продолжить;
    // Операторы
КонецЦикла;
```

- Циклическое выполнение действий для всех значений переменной-счетчика цикла от начального до конечного значения с шагом 1.
- Тело цикла выполняется пока переменная меньше либо равна конечному значению. Это условие проверяется перед очередным выполнением тела цикла.
- *Прервать* –прерывает выполнение цикла в любой точке тела цикла. Управление передается оператору, следующему за *КонецЦикла*.
- *Продолжить* –переход к следующей итерации цикла со следующим значением счетчика цикла.

Цикл с параметром. Пример

```
// Перебор дней текущего месяца
ПоследнийДеньМесяца = День(КонецМесяца(ТекущаяДата()));
Для ТекДень = 1 по ПоследнийДеньМесяца Цикл
    Состояние("Обрабатывается день:"+ТекДень);
    // Операторы обработки очередного дня
КонецЦикла;
```

Цикл по коллекции значений

```
Для каждого ИмяПеременной Из ИмяКоллекции Цикл
// Операторы
    Прервать;
    // Операторы
    Продолжить;
    //Операторы
КонецЦикла;
```

- Выполняет циклический обход коллекции значений (например, табличная часть документа).
- При каждой итерации цикла переменной присваивается значение очередного элемента коллекции.
- Обход выполняется, пока не будут выбраны все элементы коллекции, или цикл не будет прерван оператором *Прервать*.

Цикл по коллекции значений. Примеры

Пример 1.

```
// Перебор табличной части документа
Документ = Документы.Лекция.НайтиПоКоду(”12345”);
// Найден ли документ
Если Не Документ.Пустая() Тогда
    Для каждого Студ Из Документ.СписокСтудентов Цикл
        Если Студ.Присутствие Тогда
            // Обработка присутствующего
        Иначе
            // Обработка отсутствующего
        КонецЕсли;
    КонецЦикла;
КонецЕсли;
```

Пример2.

```
// Перебор табличной части текущего документа
Для каждого ТекСтрокаСписокСтудентов Из СписокСтудентов Цикл
    // Обработка текущей строки табл. части
     ТекСтрокаСписокСтудентов.Баллы = . . .
КонецЦикла;
```

Цикл с предусловием

```
Пока ЛогическоеВыражение Цикл
    // Операторы
    Прервать;
    // Операторы
    Продолжить;
    // Операторы
КонецЦикла;
```

- Циклическое выполнение операторов тела цикла, пока ЛогическоеВыражение является истинным.
- Проверка выражения на истинность выполняется перед выполнением тела цикла.

Пример1

```
// Перебор всех документов одного и того же вида.
Выборка = Документы.Лекция.Выбрать();
Пока Выборка.Следующий() Цикл
    // Информация для панели состояния
    Состояние("Обрабатывается документ №"+Выборка.Номер);
    // Обработка документа
КонецЦикла;
```

Цикл с предусловием. Пример

Пример2

```
// Перебор элементов справочника
Студ=Справочники.Студенты;
Выборка = Студ.Выбрать();
Пока Выборка.Следующий() Цикл
    Очередной=Выборка.ПолучитьОбъект();
    // Обработка очередной записи справочника
КонецЦикла;
```

Защищенный блок

```
Попытка
    // Операторы попытки
Исключение
    // Операторы обработки исключения
    [Вызвать Исключение;]
    // Операторы обработки исключения
КонецПопытки;
```

- Управляет выполнением программы, основываясь на возникающих при выполнении модуля ошибочных (исключительных) ситуациях и определяет обработку этих ситуаций.
- Не предусмотрено определенных пользователем исключений.
- Если при выполнении операторов попытки произошла ошибка, то управление передается на первый оператор обработки исключения. Управление будет передано даже если ошибку вызвал оператор, находящийся в процедуре иди функции, вызванной из операторов попытки. В этом случае выполнение подпрограммы
- После выполнения операторов обработки исключения управление передается на следующий за КонецПопытки оператор.
- Если операторы попытки выполнились без ошибок, то операторы обработки исключения пропускаются и управление передается за

Вложенность защищенных блоков

- Операторы Попытка могут быть вложенными.
- При возникновении исключения управление передается на обработчик, в попытке которого произошла ошибка.
- Если в последовательности операторов обработки исключения этого обработчика выполняется оператор ВызватьИсключение, управление передается вышестоящему обработчику исключения и так далее.
- Если вышестоящего обработчика нет, исключительная ситуация обрабатывается системно с прекращением выполнения программного модуля.

ВызватьИсключение выражение;

- Оператор допустим только внутри операторных скобок Исключение - КонецПопытки.
- Вызывает исключение, когда нужно передать управление вышестоящему обработчику исключения.
- Если есть вышестоящий обработчик, управление передается на его первый оператор. Если нет, исключительная ситуация обрабатывается системно, выдаемся сообщение о первоначально возникшей ошибке, а выполнение модуля прекращается.
- Выражение – текстовое выражение, выступающее в роли описания ошибки.

Защищенный блок. Пример

```
Процедура СформироватьВExcel()
    Попытка
        // Пытаемся обратиться к Excel
        Табл = Новый ComObject(“Excel.Application”);
    Исключение
        Предупреждение(ОписаниеОшибки());
        Возврат;
    КонецПопытки;
    // Операторы формирования отчета
КонецПроцедуры
```

Модули конфигурации

Контекст исполнения модуля

- Контекст исполнения модуля определяет «место» исполнения этого модуля.
- Для клиент-серверного варианта работы «1С:Предприятия» таких контекс[illegible]ента и конт

- В контексте клиента (на клиенте) и в контексте сервера (на сервере) доступны разные свойства, методы и объекты встроенного языка.
- Все действия, связанные с доступом к данным (их чтение и запись), возможны только на сервере.
- Отображение этих данных пользователю и другие интерактивные действия возможны только на клиенте.

Общие модули

- В конфигурации может быть определено произвольное количество общих модулей, в том числе и ни одного.
- Общий модуль может содержать только определения процедур и функций и выступает в качестве библиотеки подпрограмм.
- Общий модуль может быть глобальным. В этом случае все его экспортируемые подпрограммы доступны напрямую по имени из других модулей конфигурации.
- Если модуль не глобальный – при обращении требуется указать имя модуля (позволяет дублировать имена подпрограмм в разных модулях).

У общего модуля имеются следующие свойства (флажки), регламентирующие доступность экспортируемых подпрограмм:

- **Сервер** (по умолчанию) – все подпрограммы модуля доступны только на сервере.
- **Вызов сервера** – все серверные подпрограммы модуля становятся доступны и на клиенте.
- **Клиент** (управляемое приложение) – подпрограммы модуля доступны на клиенте.
- **Внешнее соединение** – подпрограммы доступны во внешнем соединении.

Модуль управляемого приложения

- Выполняется при запуске системы «1С:Предприятие» в управляемом режиме (свойство конфигурации *Основной режим запуска* установлено в значение *Управляемое приложение*, т.е. в режимах тонкого клиента, веб-клиента и толстого клиента в режиме управляемого приложения) или при обращении к приложению как к com-объекту.
- Этот модуль предназначен для отработки действий, связанных с сеансом работы конечного пользователя (клиента).
- Может содержать описание процедур- обработчиков событий, связанных с началом и окончанием сеанса работы пользователя:
  ПередНачаломРаботыСистемы– возникает при запуске системы до открытия главного окна (здесь можно отказаться от запуска системы);
  ПриНачалеРаботыСистемы– возникает при запуске системы после открытая главного окна (здесь уже нельзя отказаться от запуска системы(;
  ПередЗавершениемРаботыСистемы возникает при завершении работы системы до закрытия главного окна (здесь можно отказаться от завершения работы);
  ПриЗавершенииРаботыСистемы возникает при завершении работы системы после закрытия главного окна (здесь уже нельзя отказаться от завершения работы).

Модуль сеанса

- Автоматически выполняется при старте системы «1С:Предприятие» в момент загрузки конфигурации.
- Исполнение модуля сеанса происходит в привилегированном режиме сервера «1С:Предприятия» до начала исполнения модуля управляемого приложения или модуля внешнего соединения.
- Может содержать только определения неэкспортируемых процедур и функций и может использовать процедуры из общих модулей. конфигурации.
- Предназначен для инициализации параметров сеанса и отработки действий, связанных с сеансом работы.
- Для установки параметров сеанса предназначен обработчик события **УстановкаПараметровСеанса**.

Модуль внешнего соединения

- Выполняется при обращении к приложению как к COM-серверу (в режиме внешнего соединения).
- Может содержать экспортируемые переменные, процедуры и функции.
- В режиме внешнего соединения запускается «облегченный» вариант приложения, в котором недоступен интерфейс пользователя и все функции, так или иначе связанные с организацией интерфейса.
- Модуль реализует два обработчика событий: **ПриНачалеРаботыСистемы** и **ПриЗавершенииРаботыСистемы**. В них реализуются действия, выполняемые при инициализации и завершении соединения соответственно.

Модуль объекта

- Каждый прикладной объект конфигурации, данные которого могут быть модифицированы в режиме 1С:Предприятие, имеет свой модуль.
- Этот модуль исполняется при создании некоего объекта встроенного языка, который позволяет модифицировать данные объекта конфигурации.
- Соответствующий объект встроенного языка создается, например, при вводе нового объекта, при копировании, при получении данных существующего объекта и т. д.
- Для различных объектов конфигурации этот модуль имеет разное название.

Объект конфигурации	***Название модуля***
Константа	Модуль менеджера значения
Справочник, Документ, Отчет, Обработка, Бизнес-процесс, Задача	Модуль объекта
Регистр сведений Регистр накопления Последовательность	Модуль набора записей

Модуль объекта. Предопределенные процедуры

- Модуль объекта может содержать описание процедур-обработчиков событий, связанных с объектом конфигурации.
- Состав этих событий различен для разных объектов, однако есть три события, которые вызываются для всех объектов:

ОбработкаПроверкиЗаполнения – вызывается перед записью данных объекта, до начала транзакции записи. Здесь разработчик может реализовать собственные алгоритмы проверки заполнения реквизитов объекта и отказаться от записи объекта.

ПередЗаписью – вызывается перед записью данных, после начала транзакции записи, но до начала непосредственной записи данных в базу данных. Здесь разработчик имеет возможность отказаться от записи данных.

ПриЗаписи – вызывается после того, как была выполнена запись данных в базу данных, но до окончания транзакции записи. Эдесь выполняются действия над данными, неразрывно связанными с данными объекта, которые не могут быть изменены отдельно от основных данных. Здесь также разработчик может отказаться от записи данных, если, например, в результате записи этих данных в базу нарушаются какие-либо условия.

Модуль менеджера объекта

- Для каждого прикладного объекта существует менеджер, предназначенный для управления этим объектом как объектом конфигурации.
- С помощью менеджера можно создавать объекты, работать с формами и макетами.
- Модуль менеджера позволяет описать методы для объекта конфигурации (например, справочника), которые относятся не к конкретному экземпляру объекта базы данных, а к самому объекту конфигурации.
- Модуль менеджера исполняется только на сервере и может содержать только процедуры и функции.
- Если в модуле объявлены экспортируемые подпрограммы, к ним можно будет получить доступ через менеджер объекта.

Модуль управляемой формы

- Каждая форма имеет свой собственный модуль.
- Модуль управляемой формы исполняется при создании объекта УправляемаяФорма встроенного языка, а также при открытии формы прикладного объекта через интерфейс пользователя.
- Основная особенность формы как программного объекта заключается в том, что она существует одновременно и на клиенте, и на сервере.
- Таким образом, процедуры, содержащиеся в этом модуле, исполняются или в контексте клиента, или в контексте сервера.
- Поэтому все процедуры и функции, создаваемые разработчиком в модуле формы, должны иметь явное указание на то, в каком контексте они будут исполняться.
- В целом исполнение модуля формы на клиенте и на сервере характеризуется тем, что из клиентских процедур модуля формы можно вызывать серверные, тем самым передавая выполнение с клиента на сервер. После выполнения серверных процедур исполнение кода возвращается на клиент.
- Принудительно передать исполнение кода в обратную сторону, с сервера на клиент, нельзя.

Модуль управляемой формы. Директивы задания контекста исполнения

Директивы компиляции размещаются перед описанием подпрограмм:
&НаКлиенте – подпрограмма выполняется на стороне клиента в контексте формы (т.е. доступны все данные формы).
&НаСервере – подпрограмма выполняется на стороне сервера в контексте формы. Эта директива используется для обработчиков всех серверных событий формы. Разработчики используют эту директиву, чтобы передать исполнение с клиента на сервер.
&НаСервереБезКонтекста – определяет, что подпрограмма будет выполняться на стороне сервера, но контекст формы (реквизиты, элементы, параметры) будет в ней недоступен, что ускоряет обработку.
&НаКлиентеНаСервереБезКонтекста–определяет, что подпрограмма будет выполняться как на стороне сервера, так и на стороне клиента, смотря откуда она вызвана. Контекст формы будет в ней недоступен. Эту директиву разработчики обычно используют тогда, когда нужно выполнить одинаковые действия при создании формы на сервере и в процессе ее функционирования на клиенте.
!! Если никакая директива компиляции перед процедурой не указана, то платформа использует директиву &НаСервере.

Модуль управляемой формы. Обработчики событий

ПриЧтенииНаСервере – выполняется на сервере. Вызывается только для существующих в информационной базе объектов. В этой процедуре доступен прикладной объект, с которым работает форма.

ПриСозданииНаСервере– выполняется на сервере. Вызывается всегда при открытии форм и новых, и существующих объектов. Здесь уже недоступен прикладной объект, отображаемый в форме и можно отказаться от открытия формы.

ПриОткрытии – выполняется на клиенте. Событие возникает до показа открываемой формы пользователю. Здесь можно отказаться от открытия формы или выполнить некоторые интерактивные действия, которые невозможны на сервере: выдать предупреждение пользователю или открыть связанную форму и т.п..

ПередЗакрытием – выполняется на клиенте. Возникает при закрытии формы до закрытия окна формы. Здесь разработчик может отказаться от закрытия формы.

ПриЗакрытии – выполняется на клиенте. Возникает при закрытии формы после закрытия окна формы. Здесь задаются действия, которые должны быть выполнены только в случае, когда форма будет наверняка закрыта.

Модуль команды

- Для прикладных объектов конфигурации существуют подчиненные объекты **Команды**.
- Существуют общие объекты конфигурации — объекты **Общая команда**.
- У каждой команды есть модуль команды, в котором можно написать предопределенную процедуру **ОбработкаКоманды**, в которой на встроенном языке реализуются действия, выполняемые при вызове этой команды.
- Модуль команды, так же как и модуль управляемой формы, существует и на сервере, и на клиенте. В модуле команды используются следующие директивы компиляции:
  &НаКлиенте – выполняется на стороне клиента.
  &НаСервере – выполняется на стороне сервера.
  &НаКлиентеНаСервере – выполняется и на клиенте и на сервере.
- Процедура ОбработкаКоманды обязательно должна предваряться директивой &НаКлиенте, так как выполнение команды происходит в клиентском приложении.
- Из клиентских процедур модуля команды можно вызывать серверные. Но из серверных процедур/функций вызывать клиентские нельзя.
- Модуль команды должен содержать только описание процедур и

Справочники

Назначение справочника

Справочник – это ограниченный список значений некоторого реквизита (документа, формы, регистра и т.п.), имеющий средства для создания, удаления и редактирования списка пользователем прикладной системы.

Например. Реквизит *Студент* – ограниченный список всех студентов. Реквизит *Место рождения* – ограниченный список населенных пунктов. Реквизит *Сотрудник* – ограниченный список сотрудников.

Справочники предназначены для достижения 2 **целей**.

- Избежать возможности неоднозначного ввода информации. Выбор элемента из заранее заданного списка гарантирует однозначное представление одного и того же материального объекта всеми сотрудниками и во всех формах учета.
- Для реализации базы данных о всех объектах материального мира, задействованных в процессе функционирования и ведения учета на предприятии.

При помощи справочников реализуются различные классификаторы, которые установлены законодательством для каких-либо параметров или создаются самими разработчиками. Например, классификатор видов деятельности, классификатор адресов.

Одноуровневый справочник

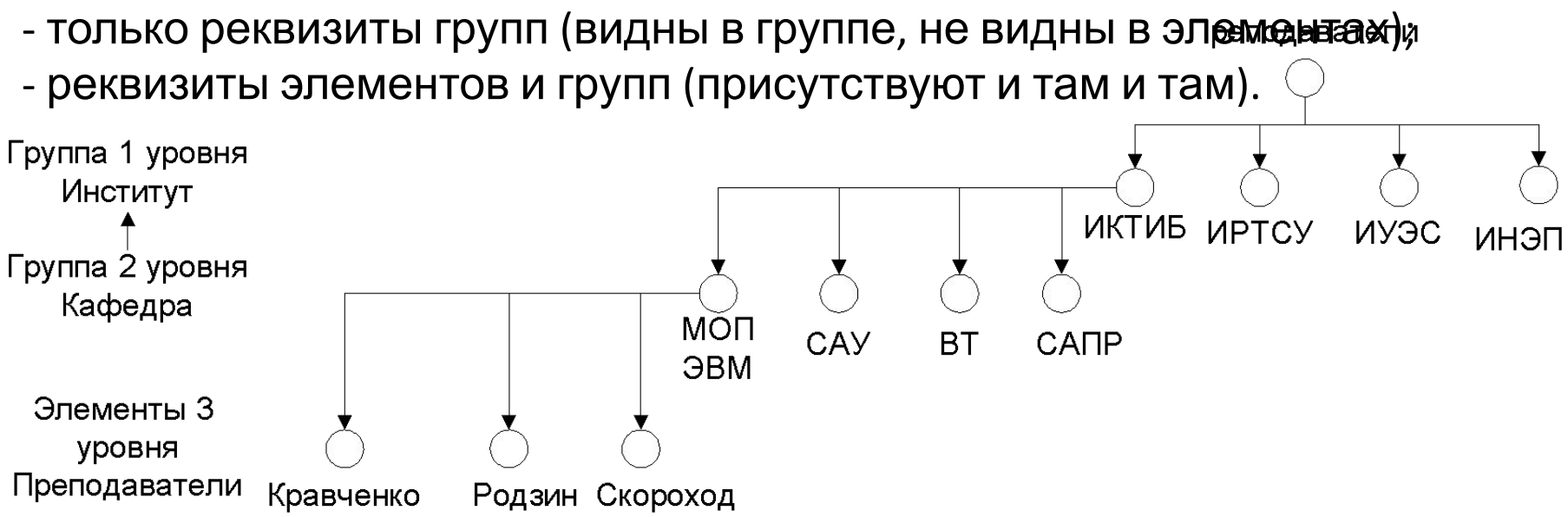
В этом режиме справочник является обычной реляционной таблицей.

- Столбцы таблицы называются «Реквизиты справочника»,
- Строки таблицы – «Элементы справочника»,
- Элементы хранят значения реквизитов.

Всегда существует, по крайней мере, два предопределенных реквизита: **Код** и **Наименование**.

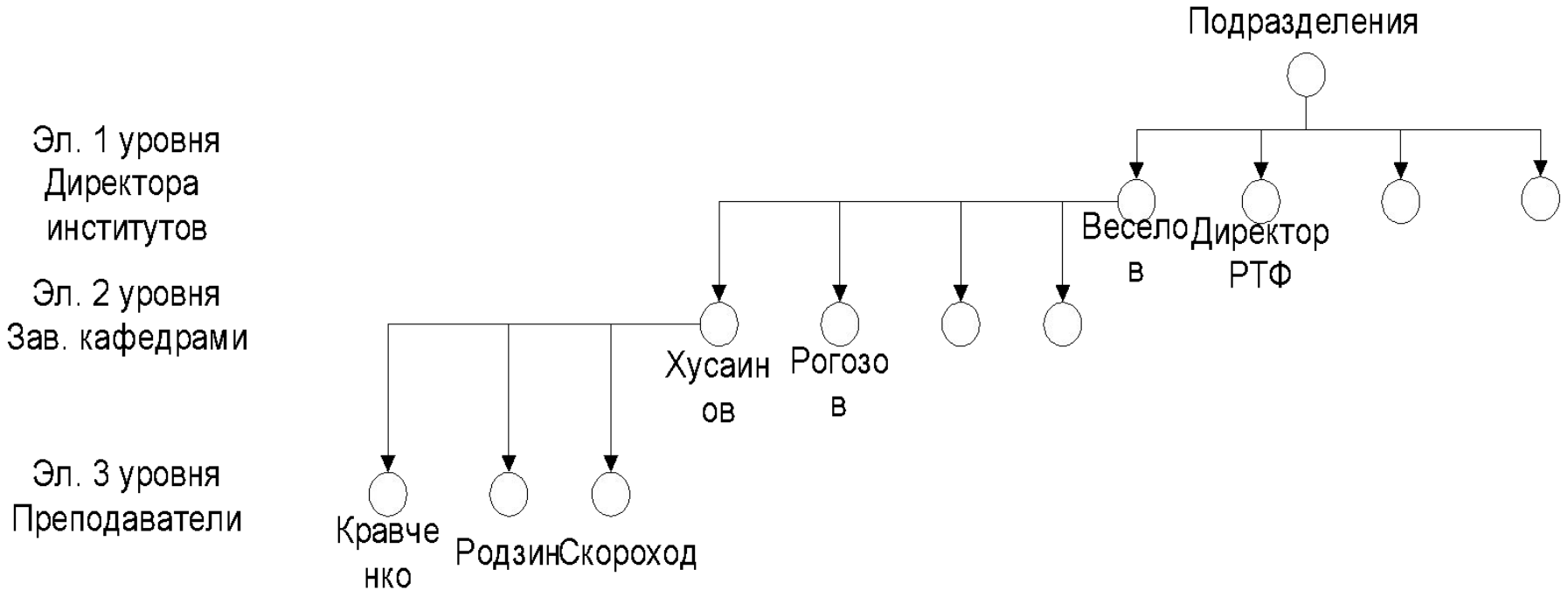
Иерархия групп и элементов

- Такой справочник состоит из элементов и групп.
- Группы предназначены для создания уровней группировки данных справочника. Группа может содержать другие группы и обычные элементы.
- Количество уровней групп можно ограничить соответствующим параметром в окне свойств справочника, но может быть и неограниченным.
- Можно отдельно задавать реквизиты для элементов и для групп справочника. Реквизиты бывают трех видов:
  - только реквизиты элементов (используется по умолчанию);
  - только реквизиты групп (видны в группе, не видны в элементах);
  - реквизиты элементов и групп (присутствуют и там и там).



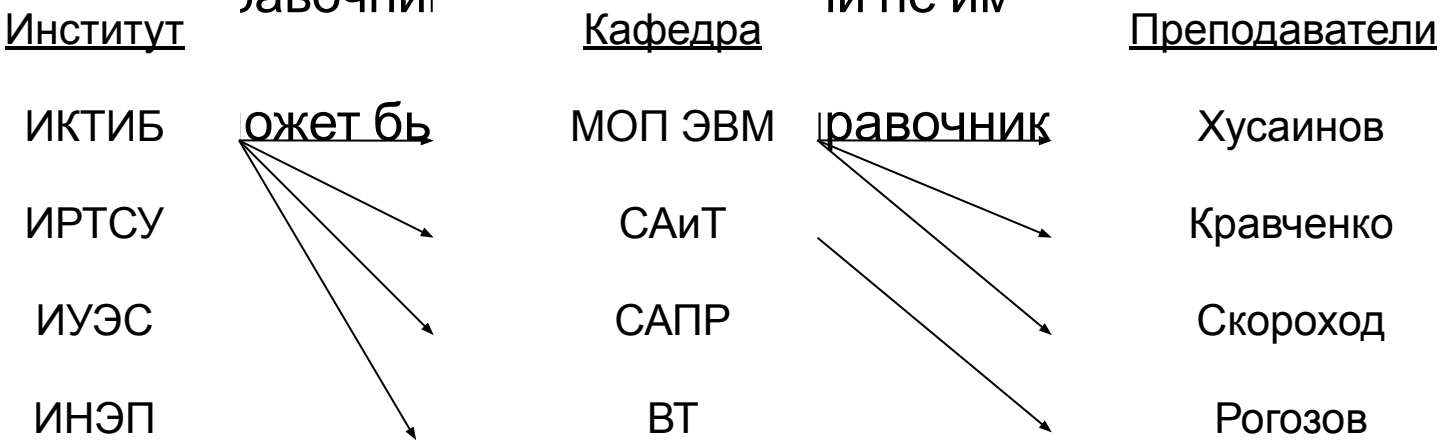
Иерархия элементов

- Такой справочник состоит только из элементов.
- Некоторые из элементов могут выступать в качестве групп, в которые сгруппированы другие элементы.



Подчиненные справочники

- Справочник может быть подчинен другому справочнику, при этом между ними устанавливается отношение (связь) «один ко многим» (1: М).
- Подчиненный справочник задается свойством *Владелец* в окне его свойств.
- Справочник, которому подчинен данный справочник, называется владельцем. Одному элементу владельца может соответствовать несколько элементов подчиненного справочника.
- Один элемент подчиненного справочника имеет только одного владельца из справочника-владельца или не имеет его, если он на
- У ожет быть равочник



Реквизиты справочника

Справочники имеют два обязательных реквизита:

- **Код** – используется системой для доступа к элементу справочника (может быть числовым или текстовым, длина кода ограничивает количество элементов справочника).
- **Наименование** - используется для отображения элементов справочника системой, имеет текстовый тип.

Помимо обязательных существует ряд предопределенных реквизитов, которые используются средствами встроенного языка.

- **ЭтоГруппа** – Истина или Ложь, в зависимости от того, является ли текущий элемент группой или нет.
- **Родитель** – ссылка на группу справочника, к которой принадлежит текущий элемент;
- **Владелец** – ссылка на элемент справочника-владельца, которому подчинен текущий элемент.
- **ПометкаУдаления** – Истина, если текущий элемент помечен на удаление и Ложь в противном случае.
- **Предопределенный** – Истина, если текущий элемент предопределен в конфигураторе, Ложь в противном случае.

Разработчик может создавать произвольное количество реквизитов, предназначенных для хранения данных.

Примеры работы со справочниками

1. Ссылка на справочник

Для работы со справочником из какого-либо модуля требуется сначала создать ссылку на этот справочник.

```
СпрСотрудники= Справочники.Сотрудники;
```

2. Создание и запись нового элемента справочника

```
НовЭл= Справочники.Сотрудники.СоздатьЭлемент();
НовЭл.Наименование= "Петров Петр Петрович";
НовЭл.Оклад= 25000;
НовЭл.Записать();// именно в этот момент происходит запись в базу данных
```

3. Создание и запись новой группы справочника (для иерархического справочника)

```
Нов = Справочники.Сотрудники.СоздатьГруппу();
Нов.Наименование= "Работающие";
Нов.Записать();
```

Примеры работы со справочниками

4. Поиск элемента справочника

```
СпрСотр = Справочники.Сотрудники;
НайденныйСотр= СпрСотр.НайтиПоКоду(123);   //ищем по коду
НайденныйСотр= СпрСотр.НайтиПоНаименованию("Иванов Иван Иванович");//по наименованию
НайденныйСотр= СпрСотр.НайтиПоРеквизиту("Оклад", 5000);    //по реквизиту
//далее нужно проверить найденное значение
ЕслиНайденныйСотр= Неопределено Тогда
   //элемент не найден
КонецЕсли;
```

5. Удаление элемента справочника

```
 СпрСотр = Справочники.Сотрудники;
СпрСотр.Удалить(); //непосредственное удаление текущего эл. справочника
СпрСотр.УстановитьПометкуУдаления(Истина); //пометка на удаление
СпрСотр.УстановитьПометкуУдаления(Ложь);//снять пометку на удаление
//можно проверить, помечен ли элемент на удаление
Пометка = СпрСотр.ПометкаУдаления;
```

Примеры работы со справочниками

6. Перебор элементов справочника

```
Выборка = Справочники.Сотрудники.Выбрать();
Пока Выборка.Следующий() = 1 Цикл // начало перебора элементов справочника в цикле
//действия с очередным элементом ...
Сообщить("Сотрудник " + Выборка.Наименование);
КонецЦикла;
```

7. Родитель. Перебор элементов внутри группы.

Группа в терминах 1С - это "родитель".

```
СпрСотр = Справочники.Сотрудники;
ГруппаРаботающие= СпрСотр.НайтиПоНаименованию("Работающие");
Выборка = СпрСотр.Выбрать(ГруппаРаботающие);
Пока Выборка.Следующий() = 1 Цикл
    //действия с очередным элементом
    Сообщить("Сотрудник " + Выборка.Наименование);
КонецЦикла;
```

Примеры работы со справочниками

8. Владелец. Перебор элементов справочника, принадлежащих элементу другого справочника.

Пусть справочник НалоговыеЛьготы подчинен справочнику Сотрудники.

```
Выборка = Справочники.НалоговыеЛьготы.Выбрать( ,Сотрудник);//тут
//отрудник - ссылка на элемент справочника сотрудники
Пока Выборка.Следующий() = 1 Цикл
    //действия с очередным элементом
    Сообщить("льгота " +Выборка.Наименование);
КонецЦикла;
```

9. Общий вид метода Выбрать

Спр.Выбрать(Родитель, Владелец, Отбор, Порядок).

- *Родитель* – отбор по родителю, т.е. в выборку будут включены элементы только заданной группы.

  Владелец – отбор по владельцу, т.е. в выборку будут включены только элементы, подчиненные заданному элементу справочника-владельца.
- *Отбор* – структура, задающая правила отбора по значениям реквизитов справочника. В качестве реквизитов отбора можно использовать Код, Наименование и любой реквизит, для которого задано индексирование.

Примеры работы со справочниками

9. Транзакция

- При создании множества элементов справочника для ускорения работы можно заключить цикл в транзакцию. При этом реальное создание всех элементов произойдет только по команде ЗафиксироватьТранзакцию().
- При использовании транзакций действие будет либо выполнено целиком (созданы все элементы справочника), либо не выполнено совсем (в случае какого-либо сбоя не будет создано ни одного нового элемента, все останется также, как до запуска транзакции).
- Транзакции широко применяются в банковской сфере. Ведь никто не хочет, чтобы при отправке денег, они благополучно списались с вашего счета, но из-за какого-либо сбоя не дошли до адресата.

```
СпрСотр = Справочники.Сотрудники;
НачатьТранзакцию();
Для Ном = 1 По 100 Цикл
   Нов = СпрСотр.СоздатьЭлемент();
   Нов.Наименование= "Новый " + Строка(Ном);
   Нов.Записать();
КонецЦикла;
ЗафиксироватьТранзакцию();
```

Документы

Назначение документа

Документ – объект конфигурации, предназначенный для ввода в систему данных о выполнении предприятием хозяйственных, организационных и т.п. операций. В большинстве случаев, документы конфигурации имеют некоторый бумажный аналог, но могут существовать и документы без бумажной копии.

- Все действия предприятия должны фиксироваться в виде документов и накапливаться в БД документов.
- В качестве БД выступают журналы документов.
- Документ может регистрироваться в одном или нескольких журналов.
- Один и тот же журнал может использоваться для регистрации одного или нескольких видов документов.

Для работы с документом требуется:

- Разработчику создать документ в конфигураторе и наполнить его функциональностью.
- Пользователю создать экземпляр документа в процессе работы прикладной системы.

Один единственный документ, созданный в конфигураторе, порождает множество документов в информационной базе.

Реквизиты документа

Документ может содержать реквизиты двух видов.

- **Реквизит** - присутствует в форме документа в единичном экземпляре и создает одиночное поле для ввода или отображения данных.
- **Реквизит табличной части -** задает столбец таблицы документа.

Документ может содержать несколько табличных частей со своим набором реквизитов.

У каждого документа есть набор предопределенных реквизитов.

- Интерфейсные (отображаются в форме документа):
  - Номер – содержит номер документа;
  - Дата – содержит дату документа;
- Не интерфейсные (доступны только средствами встроенного языка):
  - ПометкаУдаления – Истина, если документ помечен на удаление Ложь в противном случае.
  - Проведен – Истина, если документ проведён и Ложь в противном случае.

- Порядок формирования номеров документов устанавливается в окне свойств (ручная, автоматическая нумерация).
- Для того чтобы несколько документов разных видов имели сквозную нумерацию, используются нумераторы, которые создаются в

Проведение документа

Проведение документа – процесс занесения описываемых им действий в информационную базу, после которого изменяется состояние ее учетных механизмов. К учетным механизмам относятся регистры.

Документ может находиться в одном из трех состояний:

1. **Записан, но не проведен** - рабочий документ, который является незавершенным, находится в состоянии разработки. Этот документ никак не влияет на учетные механизмы системы.
2. **Проведен** - действия, описываемые документом, занесены в учетные механизмы. Такой документ отработан, и изменить его уже нельзя. Для изменения нужно отменить проведение, изменить, опять провести.
3. **Помечен на удаление** – документ не проведен и подготовлен к удалению при помощи пункта меню «Удаление помеченных объектов».

Обработка и удаление проведения

- Для обработки действий, выполняемых при проведении предназначен модуль документа. Для этого в модуле документа существуют две предопределенные процедуры: ОбработкаПроведения и ОбработкаУдаленияПроведения.
- Проведение документа фактически означает выполнение процедуры ОбработкаПроведения(), которая и должна выполнять запись в учетные механизмы.
- В случае отмены проведения необходимо откатить обратно все действия, которые выполнила процедура ОбработкаПроведения() и ликвидировать записи в учетных механизмах.
- Возможны два варианта обработки отмены проведения (режим настраивается в свойствах документа):
    - автоматический (используется по умолчанию);
    - вручную (используется, если автоматическое удаление движений не подходит). Для задания механизма обработки отмены проведения в системе и используется процедура ОбработкаУдаленияПроведения().

Режимы проведения документов

Существует 2 режима проведения:

- оперативное - документ проводится в реальном времени;
- неоперативное - проведение документа отражает уже свершившийся факт.

- Оперативное проведение в реальном времени необходимо, когда ввод и проведение документа не просто фиксируют в системе произошедшее событие, а участвуют в его формировании, помогая оператору правильно ввести информацию. Это имеет смысл только в тот момент, когда данное событие происходит в реальной жизни. Пример, продажа товара со склада, когда нужно проверить наличие достаточного количества товара и запретить продажу одной и той же единицы товара разными продавцами.
- Неоперативное проведение - в момент проведения известно, что событие уже произошло, документ проводится без каких-либо проверок.

Возможны следующие варианты проведения:

- Неоперативно всегда;
- Оперативно всегда;
- Автоматически – если дата документа раньше текущей, выполняется

Последовательности документов

Проблемы проведения

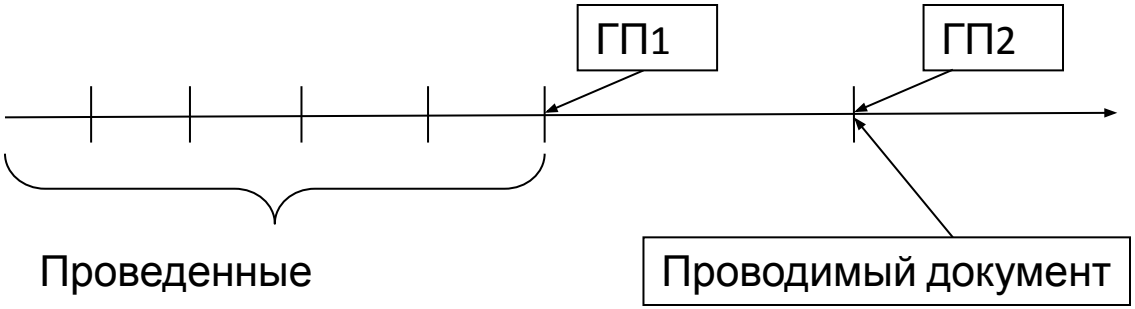
- Проведение некоторых документов зависит от остатков в регистрах на момент проведения документа!!!
- Если некоторый документ проводится задним числом, это ведет к изменению остатков и делает неверным проведение ранее правильно проведенных документов, которым будет требоваться перепроведение.

Последовательности

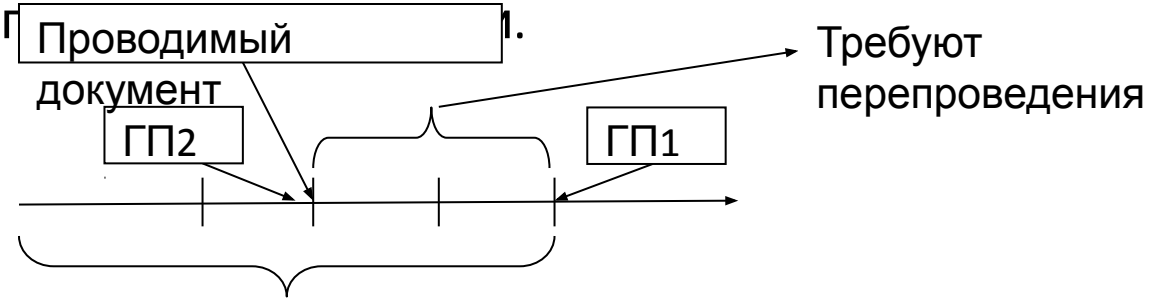
- Объект Последовательность предназначен для отслеживания таких ситуаций.
- Отслеживает актуальную границу последовательности, т.е. момент времени, до которого все документы последовательности проведены верно, после которого – требуют перепроведения (если такие есть).
- Содержит два перечня объектов.
- Объекты, входящие в последовательность – перечень документов, для которых отслеживается граница последовательности.
- Объекты, влияющие на границу последовательности. Их изменение в некоторый момент времени приводит к изменению границы и

Изменение границы последовательности

Вариант 1 (нормальная ситуация).
Граница последовательности совпадает с позицией последнего проведенного документа.

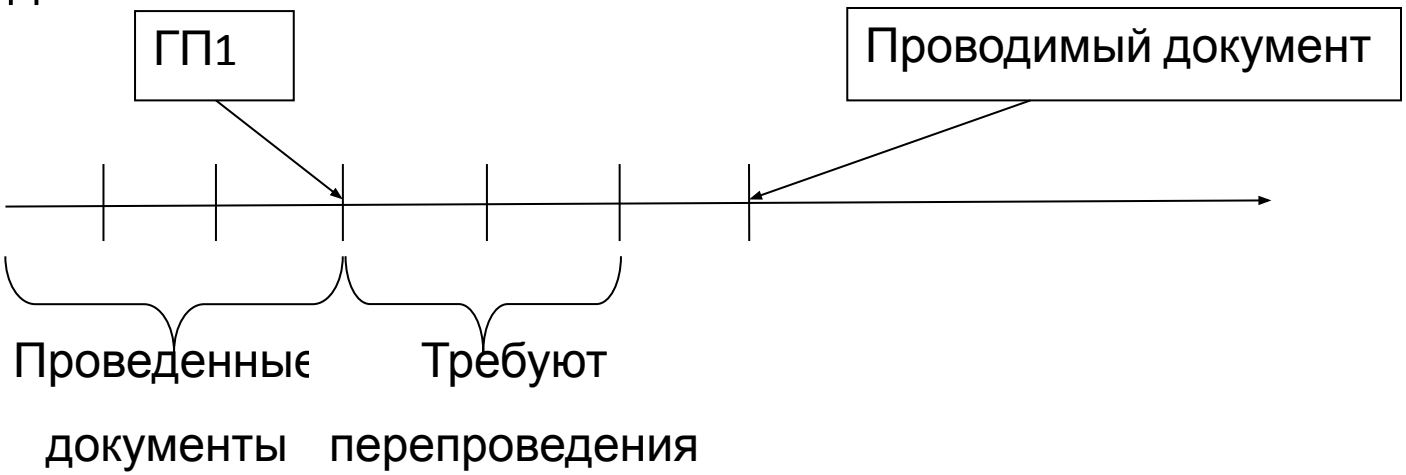


Вариант 2 (аномальная ситуация).
Возникает в случае проведения некоторого документа «задним числом», что приводит в итоге к смещению граница последовательности назад. Для исправления этой ситуации потребуется перепроведение всех документов, проведенных после границы ГП1.



Изменение границы последовательности

Вариант 3 (развитие аномального варианта).
Вытекает из второго варианта, если перепроведение документов не было выполнено. «Правильная» ГП должна располагаться на оси в том месте, куда указывает стрелка проводимого документа, однако, т.к. перепроведение выполнено не было, ГП будет располагаться на оси там же, где и ГП1.



Ввод на основании

Ввод на основании означает заполнение вновь создаваемого документа данными из ранее уже созданного документа.
Пример. Счет на оплату компьютера содержит перечень устройств, входящий в компьютер. Все последующие документы содержат тот же самый перечень устройств.

Для реализации ввода на основании следует выполнить 2 действия.
1. В окне свойств документа на вкладке *Ввод на основании* задать перечень документов, на основании которых может вводиться данный документ.
2. В модуле документа создается предопределенная процедура *ОбработкаЗаполнения*, которая вызывается при вводе документа на основании. Ее задача – проверить, на основании какого документа выполняется ввод и выполнить заполнение реквизитов документа значениями.
Заголовок процедуры имеет вид:
Процедура ОбработкаЗаполнения(ДанныеЗаполнения, СтандартнОбработка)

- ДанныеЗаполнения – ссылка на объект, на основании которого выполнен ввод. Через нее получаем доступ к данным уже введенного документа.

Ввод на основании. Пример

```
Процедура ОбработкаЗаполнения(ДЗ,СО)
    Если ТипЗнач(ДЗ)=Тип(ДокументыСсылка.Лекция) Тогда
        ЗапНаОснЛекции(ДЗ)
    КонецЕсли
КонецПроцедуры
Процедура ЗапНаОснЛекции(Лек)
    //Собственно заполнение через ссылку Лек
    Предмет=Лек.Предмет;
    . . .
КонецПроцедуры
```

Примеры работы с документами

1. Создание документа

```
&НаКлиенте
Процедура СоздатьНовыйДокумент()
    Если СоздатьНовыйДокументНаСервере() = 0 Тогда
        Сообщить("Не удалось создать новый документ");
    КонецЕсли;
КонецПроцедуры
&НаСервере
Функция СоздатьНовыйДокументНаСервере()
    НоваяРН=Документы.РасходнаяНакладная.СоздатьДокумент();
    //Заполняем реквизиты шапки
    НоваяРН.Дата=  ТекущаяДата();
    НоваяРН.Фирма=Справочники.Фирмы.ОсновнаяФирма;
    НоваяРН.Контрагент=Справочники.Контрагенты.НайтиПоКоду("000000001");
    НоваяРН.Склад=Справочники.Склады.ОсновнойСклад;
    //Заполняем табличную часть
    СтрокаТЧ=НоваяРН.ТЧТовары.Добавить();
    СтрокаТЧ.Товар=Справочники.Номенклатура.НайтиПоКоду("000000002");
    СтрокаТЧ.Цена=СтрокаТЧ.Товар.РозничнаяЦена;
    СтрокаТЧ.Количество= 2;
    СтрокаТЧ.Сумма=СтрокаТЧ.Цена*СтрокаТЧ.Количество;
```

Примеры работы с документами

1. Создание документа (продолжение)

```
    Попытка
        НоваяРН.Записать();
        Возврат 1;
    Исключение
        Возврат 0;
    КонецПопытки;
КонецФункции
```

Примеры работы с документами

2. Удаление документа.

```
&НаКлиенте
// Режим = 1 – пометка удаления, 0 – непосредственное удаление
Процедура УдалитьДокумент(Режим)
    УдалитьДокументНаСервере(Режим);
КонецПроцедуры

&НаСервере
Процедура УдалитьДокументНаСервере(ПометкаУдаления)
    УдаляемыйДокумент=Документы.РасходнаяНакладная.
НайтиПоНомеру("000000004");
    Если ПометкаУдаления= 1 Тогда
        УдаляемыйДокумент.ПолучитьОбъект().
УстановитьПометкуУдаления(Истина);
    Иначе
        УдаляемыйДокумент.ПолучитьОбъект().Удалить();
    КонецЕсли;
КонецПроцедуры
```

Примеры работы с документами

3. Перебор табличной части документа

```
&НаКлиенте
Процедура ПеребратьТабличнуюЧасть()

    Для Каждого СтрокаТЧ ИЗ Объект.ТЧТовары Цикл
        Сообщить(СтрокаТЧ.Номенклатура);
    КонецЦикла;

КонецПроцедуры
```

Примеры работы с документами

4. Поиск в табличной части документа

Для поиска необходимого значения воспользуемся функцией НайтиСтроки(). Ее можно использовать не только в табличной части документа, но и в таблице значений:

```
&НаСервере
ПроцедураНайтиТовар(СправочникНоменклатураСсылка)
	Отбор=НовыйСтруктура();
	Отбор.Вставить
("Номенклатура", СправочникНоменклатураСсылка);
	НайденноеЗначение=ТЧНоменклатура.НайтиСтроки(Отбор);
	Для Сч=1 По НайденноеЗначение.Количество() Цикл
		Сообщить(НайденноеЗначение[Сч].Наименование);
	КонецЦикла;
КонецПроцедуры
```

Регистры накопления

Назначение регистров накопления

- Регистры накопления используются для накопления учетных данных о наличии и движении каких-либо средств – материальных, денежных или других (сколько товаров на складе, какая задолженность по расчетам и т.п.).
- Вся информация о хозяйственных операциях, которая вводится при помощи документов, должна быть учтена в регистрах. Тогда ее можно будет извлечь, проанализировать и представить пользователю в виде отчетов.
- Основная задача регистра накопления – хранение данных о наличии материальных средств или сумме выполненных операций таким образом, чтобы их можно было обработать с максимальной скоростью в режиме реального времени (например, в момент продажи товара).
- Регистр накопления представляет некую внутреннюю таблицу, доступ к которой осуществляется средствами встроенного языка.
- Запись в регистр разрешается только модулю документа.
- Документ записывает в регистр изменения таблицы, называемые движениями регистра.
- Через интерфейс системы можно просмотреть только движения регистра.

Описание регистра

Регистр задается тремя составляющими:

- **Измерения** – задают разрезы данных, которые будут храниться в регистре. Обычно в качестве измерений выступают справочники.
- **Ресурсы** – задают состав хранящихся данных, это числовые данные и для каждого набора значений измерений регистр хранит набор определенных в нем ресурсов.
- **Реквизиты** – не имеют функционального назначения, их значения используются как доп. информация, заносимая в движения с целью выполнения отбора по значению реквизитов.

Пример регистра накопления

Регистр – Наличие товаров. Содержит данные о запасах некоторых видов товаров на складах некоторой фирмы.
Измерения – Товары, Склады (справочники).
Ресурсы – Количество, Сумма (суммарная стоимость).

	Склад 1 Р.Поле	Склад 2 Центр	Склад 3 Н.Вокзал
Память	100 50 000	50 25 000	200 100 000
Процессор	10 40 000	20 80 000	50 200 000
HDD	20 40 000	10 20 000	50 100 000
Материнская плата	0 0	0 0	0 0

Виды регистров накопления

В зависимости от характера информации регистры разделяются на два вида:
- регистр остатков;
- оборотный регистр.

Регистр остатков хранит информацию о наличии чего-либо. С ним можно выполнять 2 действия:
- приход (добавление средств);
- расход (потребление - уменьшение) средств.

Остаток, по своей сути, не должен быть отрицательным, хотя система позволяет хранить отрицательные остатки.
Предыдущая таблица – регистр остатков.

Оборотный регистр хранит информацию о сумме выполненных операций за некоторый период времени.
С оборотным регистром возможна только одна операция – движение регистра. Эта операция заключается в увеличении значения ресурса регистра.

Пример оборотного регистра

Регистр - Закупки товаров. Содержит данные о закупках (количество, сумма) товаров.
Измерения – товары, контрагенты.
Ресурсы – количество, сумма.

	ТТИ ЮФУ	ТГПИ	ТИУиЭ
Плазменные панели	200 40 000 000	0 0	0 0
Компьютеры	500 5 000 000	10 100 000	20 200 000
Принтеры	0 0	10 50 000	5 25 000

Структура регистра накопления

Регистр представляет собой базу данных, состоящую из трех частей:

- **Начальные таблицы**.
- Содержит данные об остатках или оборотах на начало всех предыдущих периодов, в которых использовался этот регистр.
- Оперативная работа с регистром ведется в текущем периоде, который открывается в пункте меню «Управление оперативными итогам».
- **Записи обо всех изменениях ресурсов регистра** – движения регистра.
- Движения регистра и начальная таблица позволяют восстановить значение регистра на любой заданный момент времени.
- Движения могут быть просмотрены через стандартный интерфейс системы.
- **Итоговая таблица** со значениями ресурсов на текущий момент времени.

Атрибуты регистров

Атрибуты используются для обращения к регистру средствами встроенного языка.

<**Измерение**> – идентификатор измерения регистра;

<**Ресурс**> – идентификатор ресурса регистра;

<**Реквизит**> – идентификатор реквизита регистра;

Для движений регистров:

ВидДвижения – содержит признак вида движения регистра остатков:

- ВидДвиженияНакопления.Приход или
- ВидДвиженияНакопления.Расход;

НомерСтроки – номер строки документа-регистратора, по которой выполняется движение;

Период – дата/время в которое записано движение;

Регистратор – ссылка на документ-регистратор, создавший движение.

- Каждый регистр должен иметь как минимум 1 регистратор – документ, выполняющий запись движений в этот регистр.
- Перечень регистраторов регистра задается в окне его свойств на вкладке *Регистраторы*.

Примеры работы с регистрами накопления

1. **Получение доступа к менеджеру конкретного регистра.**

```
РегОстатки = РегистрыНакопления. ОстаткиТоваров;
РегПродажи = РегистрыНакопления.Продажи;
```

2. Выборка движений регистра.

```
УчетНоменклатуры=РегистрыНакопления.УчетНоменклатуры;
ОтборПоТовару=Новый Структура();
ОтборПоТовару.Добавить("Номенклатура",ВыбТовар);
НачДата=НачалоГода(ТекущаяДата());
КонДата=ТекущаяДата();
Выборка=УчетНоменклатуры.Выбрать(НачДата,КонДата,
ОтборПоТовару);
Расход=0;
Пока Выборка.Следующий() Цикл
    Если Выборка.ВидДвижения=ВидДвиженияНакопления.Расход
Тогда
        Расход=Расход+Выборка.Количество;
    КонецЕсли;
```

Примеры работы с регистрами накопления

3. Запись движений в регистр остатков

Может выполняться только в процедуре **ОбработкаПроведения**

```
Движения.ИмяРегистра.Записывать = Истина;
Для каждого ТекСтр Из ТабЧастьДок Цикл
    Движ=Движения.ИмяРегистра.Добавить();
    Движение.Период=Дата;
    Движение.Измерение1=ТекСтр.ЗначениеИзм1;
    Движение.Измерение2=ТекСтр.ЗначениеИзм2;
    Если ТекСтр.ПризнакПрихода Тогда
        Движение.ВидДвижения= ВидДвиженияНакопления.Приход;
        Движение.Ресурс=ТекСтр.СуммаПрихода
    Иначе
        Движение.ВидДвижения= ВидДвиженияНакопления.Расход;
        Движение.Ресурс=ТекСтр.СуммаРасхода
    КонецЕсли
КонецЦикла;
```

Примеры работы с регистрами накопления

4. Запись движений в оборотный регистр

Может выполняться только в процедуре **ОбработкаПроведения**

```
Движения.ИмяРегистра.Записывать = Истина;
Для каждого ТекСтр Из ТабЧастьДок Цикл
    Движ=Движения.ИмяРегистра.Добавить();
    Движение.Период=Дата;
    Движение.Измерение1=ТекСтр.ЗначениеИзм1;
    Движение.Измерение2=ТекСтр.ЗначениеИзм2;
    Движение.Ресурс=ТекСтр.СуммаОборота
КонецЦикла;
```

Примеры работы с регистрами накопления

5. Получение итогов из регистра-остатков

Выполняется методом регистра накопления

Остатки(МоментВремени,Отбор,Измерения,Ресурсы)

МоментВремени – значение типа дата/время, на момент которого будут выбраны остатки.

Отбор – структура, содержащая пары ключ-значение, по которым будет выполняться отбор.

Измерения – строка со списком имен измерений, разделенных запятыми, для которых нужно получить остатки.

Ресурсы – строка со списком имен ресурсов, разделенных запятыми, по которым требуется получить остатки.

Результатом метода является таблица значений, имеющая следующую структуру.

ИмяИзм1	ИмяИзм2	…	ИмяРесурса1	…
ЗначИзм1	ЗначИзм2	…	ОстатокПоРес1	…

Примеры работы с регистрами накопления

5. Получение сводного итога из регистра остатков

- Если при вызова метода *Остатки* состав измерений в структуре отбора и строке **Измерения** совпадает, то получим сводный итог и в таблице значений будет только одна строка.
- Доступ к значениям полей таблица выполняется так: ТабЗнач[0].ИмяПоля.

```
Фильтр=Новый Структура;
Фильтр.Вставить(“Студент”, ССылкаНаСправСтудентов);
Фильтр.Вставить(“Предмет”,ССылкаНаСправПредметов);
Фильтр.Вставить(“ВидЗанятия”, ССылккаНаВидЗанятия);
Таб=РегистрыНакопления.БаллыСтудентов.Остатки(Дата,Фильтр,
“Студент,Предмет,ВидЗанятия”,”Балл”);
Сообщить(Таб[0].Балл);
```

Примеры работы с регистрами накопления

5. Получение развернутого итога из регистра остатков

- Развернутый итог – итог по нескольким значениям одного или нескольких измерений.
- Вычисляется тем же методом Остатки, что и в предыдущем случае, но для развернутого итога перечень фильтруемых измерений в параметре *Отбор* должен быть меньше перечня измерений, по которым получаем остатки – строка *Измерения*.
- Результатом метода является таблица значений из нескольких строк.

ИмяИзм1	ИмяИзм2	…	ИмяРесурса1	…
Знач1Изм1	Знач1Изм2	…	Остаток1ПоРес1	…
Знач2Изм1	Знач2Изм2	…	Остаток2ПоРес1	…
…				

Примеры работы с регистрами накопления

5. Получение развернутого итога из регистра остатков (продолжение)

- Для выбора данных из такой таблицы применяется цикл.

```
Кол=Таб.Количество(); //Кол-во строк таблицы
Для Ном=0 По Кол-1 Цикл
    Таб[Ном].ИмяСтолбца – обращение к значению столбца строки Ном
    …
КонецЦикла;
```

- В качестве примера получим баллы всех студентов по некоторому предмету и выведем их в качестве сообщения.

```
Фильтр=Новый Структура;
Фильтр.Вставить(“Предмет”,ССылкаНаСправПредметов);
Таб=РегистрыНакопления.БаллыСтудентов.Остатки(Дата,Фильтр,
    “Студент,Предмет ”,”Балл”);
Кол=Таб.Количество();
Для Н=0 По Кол Цикл
    Сообщить(Таб[Н].Студент);
    Сообщить(Таб[Н].Балл);
КонецЦикла;
```

Примеры работы с регистрами накопления

6. Получение развернутого итога из регистра оборотов

- Выполняется методом регистров накопления Обороты(НачПериода,КонПериода,Отбор,Измерения,Ресурсы)
  НачПериода – дата или момент времени, задающий с какого времени нужно вычислять обороты. Если не задан – с самого начала.
  КонПериода – дата или момент времени, задающий по какое время вычислять обороты, если не задано – по текущий момент.
  Отбор, Измерения, Реквизиты – имеют точно такой же смысл как и при вычислении остатков.
- Метод, аналогично остаткам, возвращает таблицу значений и используется аналогично, за исключением задания начального и конечного моментов времени.

Примеры работы с регистрами накопления

7. Перебор набора записей (движений) документа

Для выборки движений документа устанавливается отбор по регистратору с указанием ссылки на этот документ.

```
Набор=РегистрыНакопления.Продажи.СоздатьНаборЗаписей();
Набор.Отбор.Регистратор.Установить(ВыбДок);
Набор.Прочитать();
Для Каждого Движ Из Набор Цикл
    Сообщить(Движ.Сумма);
КонецЦикла;
```

8. Очистка набора записей (движений) документа.

```
Рег=РегистрыНакопления.ЗаказыПоставщикам;
НаборЗаказыПоставщикам=Рег.СоздатьНаборЗаписей();
НаборЗаказыПоставщикам.Отбор.Регистратор.Установить(Ссылка);
НаборЗаказыПоставщикам.Прочитать();
НаборЗаказыПоставщикам.Очистить();
НаборЗаказыПоставщикам.Записать();
```

Примеры работы с регистрами накопления

9. Добавление или замещение записей (движений) документа

```
Набор=РегистрыНакопления.Продажи.СоздатьНаборЗаписей(); Набор.
Отбор.Регистратор.Установить(ВыбДок);
Движ=Набор.Добавить();          // Эту последовательность операторов
Движ.Регистратор= ВыбДок;     //
Движ.Номенклатура = ВыбТовар;    //
Движ. Контрагент = ВыбКонтрагент; //
Движ.Период = РабочаяДата;    //
Движ.Количество = Количество;      //
Движ.Сумма = Сумма;         // Можно выполнять в цикле
Набор.Записать(Ложь); //добавить к набору записей по документу
или
Набор.Записать(Истина); //заменить все старые записи по документу на
новые
```

Регистры сведений

Назначение регистров сведений

Регистр сведений – это многомерная таблица данных, задачей которой является хранение и выборка информации, состав которой развернут по определенной комбинации значений.

Пример. Регистр сведений *ЦеныТоваров* для хранения информации о ценах в разрезе товаров и поставщиков.

	Вист-Дон	Санрайз	DNS
Процессор	7000	6500	6800
Память	2000	2100	2200
Мат. плата	3500	3400	3300

Периодичность регистра сведений

Регистры сведений могут быть двух видов:

- периодические,
- непериодические.

Непериодический - данные в регистре хранятся без привязки ко времени.

Например, актуальные цены товаров у поставщиков – данные на текущий момент.

Периодический - данные в регистре хранятся с привязкой ко времени, регистр хранит всю историю изменения своих данных.

Например, регистр с данными о курсах валют – для корректности расчетов должен «знать» курс на любую заданную дату.

Для периодического регистра имеется свойство *Периодичность*:

- задает интервал времени, в течение которого значения ресурсов являются постоянными и не могут быть изменены.
- Возможные варианты: в пределах секунды, дня, месяца, квартала, года.
- ***Внимание!*** При попытке записать повторное значение в пределах установленной периодичности система выдает сообщение об ошибке

Описание регистра сведений

Регистр задается двумя составляющими:

- **Измерения** – задают разрезы данных, которые будут храниться в регистре. Обычно в качестве измерений выступают справочники.
- **Ресурсы** – задают состав хранящихся данных, это числовые данные и для каждого набора значений измерений регистр хранит набор определенных в нем ресурсов. Для периодического регистра – историю изменения ресурса.

Необязательная составляющая описания регистра:

- **Реквизиты** – не имеют функционального назначения, их значения используются как доп. информация, заносимая в движения (записи) регистра с целью выполнения отбора по значению реквизитов.

Измерения регистра могут быть назначены как ***Ведущие***:

- Строка/столбец регистра с некоторым значением такого измерения имеет смысл, пока существует связанный с нею объект информационной базы.
- Если же объект удален, то должна быть удалена и соответствующая строка/столбец регистра сведений.

Например, измерение *Поставщики* является ведущим и соответствует справочнику поставщиков. Если из справочника удалить DNS, то из

Источники ввода данных в регистры

В зависимости от возможных источников ввода данных регистры разделяются на 2 категории:

- независимые регистры,
- регистры, подчиненные регистратору.

Независимые регистры - запись может осуществляться либо вручную, либо средствами встроенного языка.

Подчиненных регистратору :

- Запись может выполняться только документом при проведении документа.
- Все записи жестко привязаны к своим документам.
- Если документ удален или сделан непроведенным, все связанные с ним записи удаляются из регистра.

Атрибуты регистров

Атрибуты используются для обращения к регистру средствами встроенного языка.

<**Измерение**> – идентификатор измерения регистра, заданный в конфигураторе;

<**Ресурс**> – идентификатор ресурса регистра , заданный в конфигураторе;

<**Реквизит**> – идентификатор реквизита регистра , заданный в конфигураторе;

Период – дата/время изменения значения ресурса периодического регистра;

Примеры работы с регистрами сведений

1. **Запись в периодический регистр.**

```
Движения.ЦеныТоваров.Записывать = Истина;
Движение = Движения. ЦеныТоваров.Добавить();
Движение.Товар =ССылкаНаСправТоваров;
Движение.Поставщик = ССылкаНаСправПоставщиков;
Движение.Цена = УстанавливаемаяЦена;
Движение.Период=ДатаЦены
```

2. Запись в непериодический регистр.

```
Движения.ЦеныТоваров.Записывать = Истина;
Движение = Движения. ЦеныТоваров.Добавить();
Движение.Товар =ССылкаНаСправТоваров;
Движение.Поставщик = ССылкаНаСправПоставщиков;
Движение.Цена = УстанавливаемаяЦена;
```

Примеры работы с регистрами накопления

3. Извлечение значений ресурсов периодического регистра

Используется метод

Получить (Период, Отбор)

Период– момент времени, на который требуется получить значения ресурсов.

Отбор – структура для отбора по значениям измерений.

Возвращаемое значение – структура, ключами которой являются имена ресурсов.

```
Отбор = Новый Структура;
Отбор.Вставить(“Товары”,ССылкаНаСправТоваров);
Отбор.Вставить(“Поставщики”,ССылкаНаСправПоставщиков);
ЗначРесурсов=Регистры.ЦеныТоваров.Получить(Дата,Отбор);
ЦенаТовара=ЗначРесурсов.Цена;
```

Примеры работы с регистрами сведений

5. Извлечение значений ресурсов непериодического регистра

Отбор = Новый Структура;
Отбор.Вставить(“Товары”,ССылкаНаСправТоваров);
Отбор.Вставить(“Поставщики”,ССылкаНаСправПоставщиков);
ЗначРесурсов=Регистры.ЦеныТоваров.***Получить(Отбор)***;
ЦенаТовара=ЗначРесурсов.Цена;

6. Получение последних значений ресурсов периодического регистра

Выполняется методом
ПолучитьПоследнее(КонецПериода,Отбор)
КонецПериода– момент времени, на который нужно получить последние значения. Если он не задан, выдаются самые последние значения.
Отбор– структура, задающая отбор по значениям измерений.
Возвращаемое значение – структура, ключами которой являются имена ресурсов.

Отбор = Новый Структура;
Отбор.Вставить(“Товары”,ССылкаНаСправТоваров);
Отбор.Вставить(“Поставщики”,ССылкаНаСправПоставщиков);

Примеры работы с регистрами сведений

7. Получение первых значений ресурсов периодического регистра

Выполняется методом
ПолучитьПервое(НачалоПериода,Отбор)
НачалоПериода– момент времени, на который нужно получить последние значения. Если он не задан, выдаются самые первые значения.
Отбор– структура, задающая отбор по значениям измерений.
Возвращаемое значение – структура, ключами которой являются имена ресурсов.

```
Отбор = Новый Структура;
Отбор.Вставить(“Товары”,ССылкаНаСправТоваров);
Отбор.Вставить(“Поставщики”,ССылкаНаСправПоставщиков);
ЗначРесурсов=Регистры.ЦеныТоваров.ПолучитьПервое(Отбор);
ЦенаТовара=ЗначРесурсов.Цена;
```

Примеры работы с регистрами накопления

8. Выборка движений регистра по регистратору

```
ВыбДок = Документы.УстановкаЦен.Выбрать();   ЦеныТоваров
Пока ВыбДок.Следующий() Тогда
    ВыбДвиж= РегистрыСведений. ЦеныТоваров.
    ВыбратьПоРегистратору(                    ВыбДок );
    Пока ВыбДвиж.Следующий() Цикл Сообщить(ВыбДвиж.
    Номенклатура.Наименование + " " +          ВыбДвиж.
    Цена);
    КонецЦикла;
 КонецЦикла;
```

Примеры работы с регистрами сведений

9. Выборка движений периодического регистра

Выполняется методом:

Выбрать(НачалоИнтервала, КонецИнтервала, Отбор, Порядок)

НачалоИнтервала – задает начальную дату/время.

КонецИнтервала – задает конечную дату/время.

Отбор – структура для отбора движений (необязательно).

Порядок – правило упорядочивания движений (необязательно).

Пример.

```
Курсы = РегистрыСведений.КурсыВалют;
ОтборВалют = Новый Структура("Валюта");
ОтборВалют.Валюта = ВыбВалюта;
ВыборкаКурсовВалют = Курсы.Выбрать(Дата1,Дата2,ОтборВалют);
Пока ВыборкаКурсовВалют.Следующий() Цикл
      Сообщить("Дата " + ВыборкаКурсовВалют.Период +
            " Валюта "+СокрЛП(ВыборкаКурсовВалют.Валюта) +
            ", Курс "+ ВыборкаКурсовВалют.Курс );
КонецЦикла;
```

Примеры работы с регистрами сведений

10. Добавление новой записи периодического независимого регистра сведений

Так можно выполнять запись в регистр, не подчиненный регистратору, т. е. не в модуле документа.

```
НаборЗаписей = РегистрыСведений.КурсыВалют.
СоздатьНаборЗаписей(); НаборЗаписей.Отбор.Валюта.Установить
(ТекущаяВалюта);
НовЗапись = НаборЗаписей.Добавить();
НовЗапись.Валюта = ТекущаяВалюта;
НовЗапись.Период = ТекущаяДата;
НовЗапись.Курс = ТекущийКурс;
НовЗапись.Кратность = ТекущаяКратность;
НаборЗаписей.Записать(Истина);
```

Примеры работы с регистрами сведений

11. Чтение/изменение существующей записи периодического независимого регистра

```
НаборЗаписей = РегистрыСведений.Валюты.СоздатьНаборЗаписей();
НаборЗаписей.Отбор.Период.Установить(ДатаЗаписи); НаборЗаписей.
Прочитать();
Для Каждого Запись Из НаборЗаписей Цикл
    /
    / Чтение и вывод данных полей записи.
    Сообщить(Строка(Запись.Период) + " " + Строка(Запись.Валюта) + "
" +  Строка(Запись.Курс));

    // Изменение данных полей записи.
    Запись.Курс = 0;
КонецЦикла;
НаборЗаписей.Записать();
```

12. Удаление записи независимого регистра

```
НаборЗаписей = РегистрыСведений.КурсыВалют.
СоздатьНаборЗаписей(); НаборЗаписей.Записать();
```

Примеры работы с регистрами сведений

13. Добавление записей в непериодический независимый регистр

```
НаборЗаписей = РегистрыСведений.ОстаткиТоваров.
СоздатьНаборЗаписей(); НаборЗаписей.Отбор.Товар.Установить
(ССылкаНаТовар); НаборЗаписей.Отбор.Склад.Установить
(ССылкаНаСклад);
НоваяЗапись = НаборЗаписей.Добавить();
НоваяЗапись. Товар = ССылкаНаТовар;
НоваяЗапись.Склад = ССылкаНаСклад;
НоваяЗапись.Значение = ТекущееЗначение;
НаборЗаписей.Записать();
```

14. Чтение записей непериодического независимого регистра

```
НаборЗаписей = РегистрыСведений.ОстаткиТоваров.
СоздатьНаборЗаписей();
// НаборЗаписей.Отбор.Товар.Установить(ССылкаНаТовар);
НаборЗаписей.Прочитать();
// Перебрать записи в цикле…
Для Каждого Запись из НаборЗаписей Цикл
    Склад = Запись.Склад;
    Количество = Запись.Количество;
```

Примеры работы с регистрами сведений

15. Изменение записей в непериодическом независимом регистре

```
НаборЗаписей = РегистрыСведений.ОстаткиТоваров.
СоздатьНаборЗаписей(); НаборЗаписей.Отбор.Товар.Установить
(ССылкаНаТовар); НаборЗаписей.Отбор.Склад.Установить
(ССылкаНаСклад);
НаборЗаписей.Записать();
НоваяЗапись = НаборЗаписей.Добавить();
НоваяЗапись. Товар = ССылкаНаТовар;
НоваяЗапись.Склад = ССылкаНаСклад;
НоваяЗапись.Значение = НовоеЗначение;
НаборЗаписей.Записать();
```