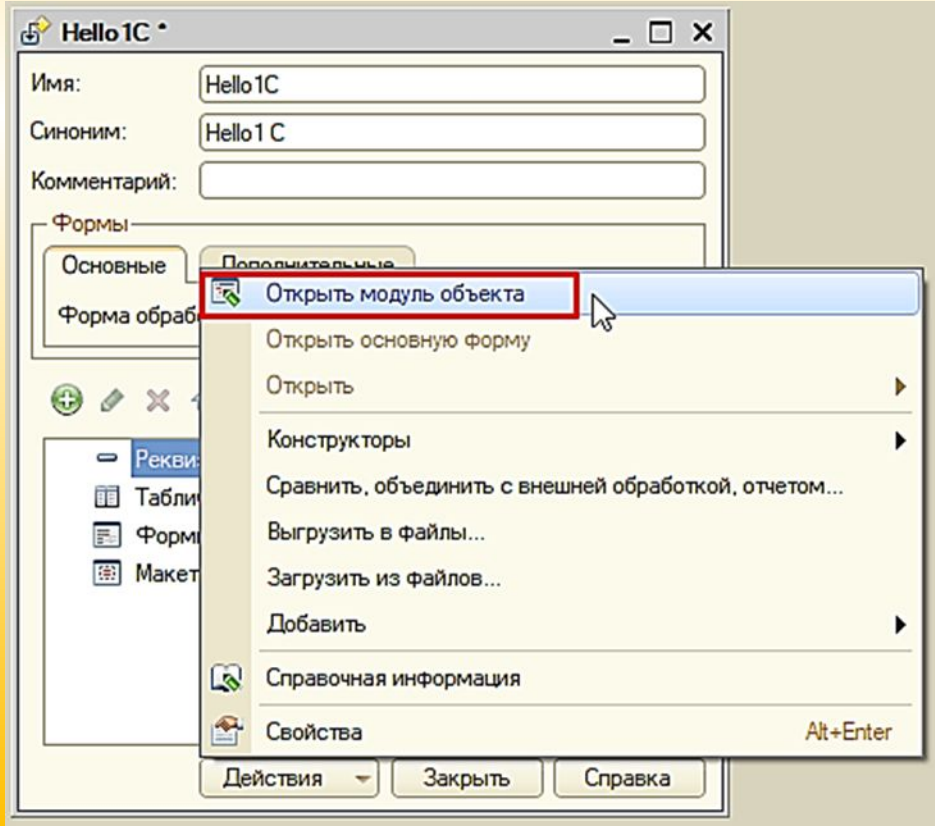
ВСТРОЕННЫЙ ЯЗЫК 1С



Встроенный язык системы «1С: Предприятие»

Встроенный язык системы «1С:Предприятие» предназначен для описания (на стадии разработки конфигурации) **алгоритмов функционирования прикладной задачи**.

Встроенный язык представляет собой **предметно-ориентированный язык программирования**, специально разработанный с учетом его применения для задач автоматизации деятельности различных организаций.

Язык ориентирован на его использование **профессиональными программистами и квалифицированными пользователями** системы 1С:Предприятие.

Особенности встроенного языка

- **отсутствие программного описания объектов конфигурации:** разработчик может использовать либо ***встроенные в платформу объекты, либо объекты, созданные системой в результате визуального конструирования прикладного решения.***
- **событийная ориентированность встроенного языка:** вызов процедур и функций встроенного языка реализован как реакция на конкретные события (например, при нажатии кнопки в диалоговом окне).

Синтаксис встроенного языка 1С:Предприятие

Язык написания программных модулей

Встроенный язык системы «1С:Предприятие» является ***двуязычным***. Почти все зарезервированные слова, имена типов значений, свойств, методов, событий имеют два имени: русское и английское. Исключение составляют слова, не имеющие аналогов в русском языке. В тексте программных модулей эти имена можно свободно смешивать, используя то русские, то английские имена без каких-либо ограничений.

Регистр букв (строчные или заглавные) при написании имен переменных, свойств, методов, процедур, функций, а также функций встроенного языка ***не имеет значения***.

Язык написания программных модулей

- Исходный текст программного модуля может состоять из ***операторов и комментариев.***

```
&НаКлиенте
Процедура ПеречеьНоменклатурыНоменклатураПриИзменении(Элемент)

    // Получить текущую строку табличной части
    СтрокаТабличнойЧасти = Элементы.ПеречеьНоменклатуры.ТекущиеДанные;

    // Установить цену
    СтрокаТабличнойЧасти.Цена = РаботаСоСправочниками.РозничнаяЦена(Объект.Дата, СтрокаТабличнойЧасти.Номенкла

    // Пересчитать сумму строки
    РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);

        // Установить скидку
    СтрокаТабличнойЧасти.СкидкаНаМатериалы = СтрокаТабличнойЧасти.Сумма / 20;

    // Установить Итоговую сумму
    СтрокаТабличнойЧасти.ИтоговаяСумма = СтрокаТабличнойЧасти.Сумма - СтрокаТабличнойЧасти.СкидкаНаМатериалы;



КонецПроцедуры
```

Формат операторов

Между собой операторы встроенного языка обязательно следует **разделять символом точка с запятой**.

Конец строки не является признаком конца оператора, т. е. операторы могут свободно переходить через строки и продолжаться на другой строке. ***Можно располагать произвольное число операторов в одной строке***, разделяя их символом точка с запятой.

```
&НаКлиенте
Процедура ПеречеьНоменклатурыНоменклатураПриИзменении(Элемент)

    // Получить текущую строку табличной части
    СтрокаТабличнойЧасти = Элементы.ПеречеьНоменклатуры.ТекущиеДанные;

    // Установить цену
    СтрокаТабличнойЧасти.Цена = РаботаСоСправочниками.РозничнаяЦена(Объект.Дата, СтрокаТабличнойЧасти.Номенкла

    // Пересчитать сумму строки
    РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти);

        // Установить скидку
    СтрокаТабличнойЧасти.СкидкаНаМатериалы = СтрокаТабличнойЧасти.Сумма / 20;

    // Установить Итоговую сумму
    СтрокаТабличнойЧасти.ИтоговаяСумма = СтрокаТабличнойЧасти.Сумма - СтрокаТабличнойЧасти.СкидкаНаМатериалы;



КонецПроцедуры
```

Комментарии

Комментарий используется ***для размещения в исходном тексте программного модуля всякого рода пояснений к работе модуля.***

В тексте программного модуля комментарий ***начинается парой символов «//» и заканчивается концом строки. Комментарий можно начинать с начала строки или записывать его после оператора на той же строке.*** После начала комментария нельзя писать оператор на той же строке, необходимо закончить комментарий концом строки.

Пример:

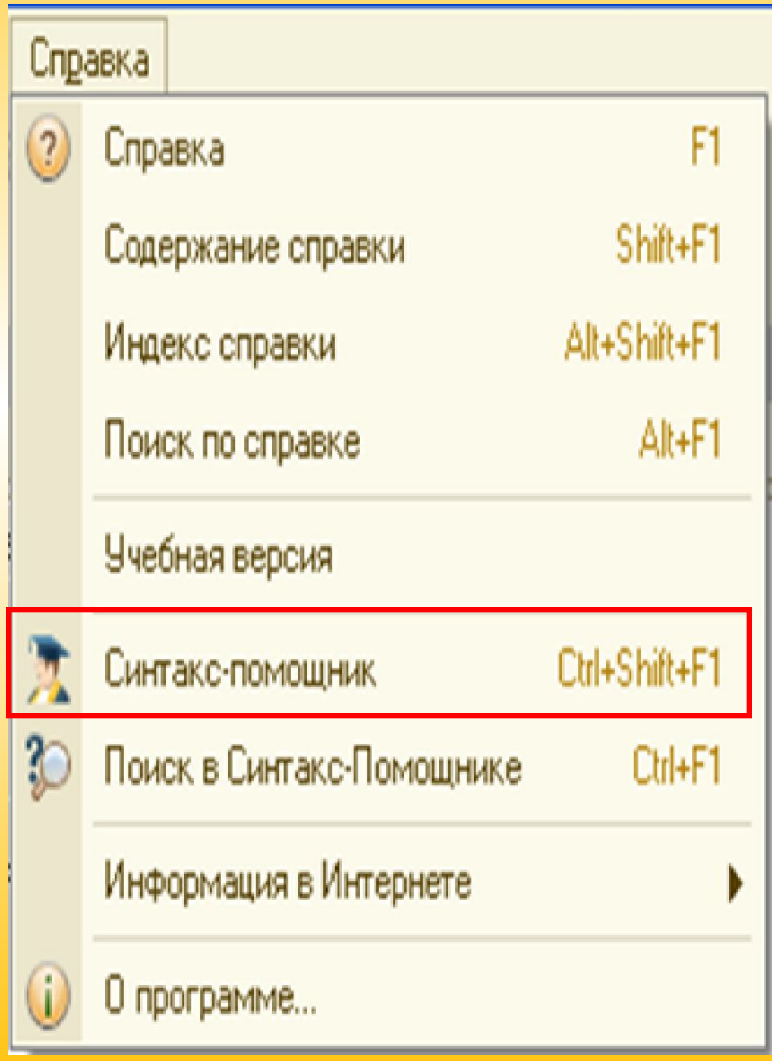
A=B; // Это - комментарий

// это тоже комментарий

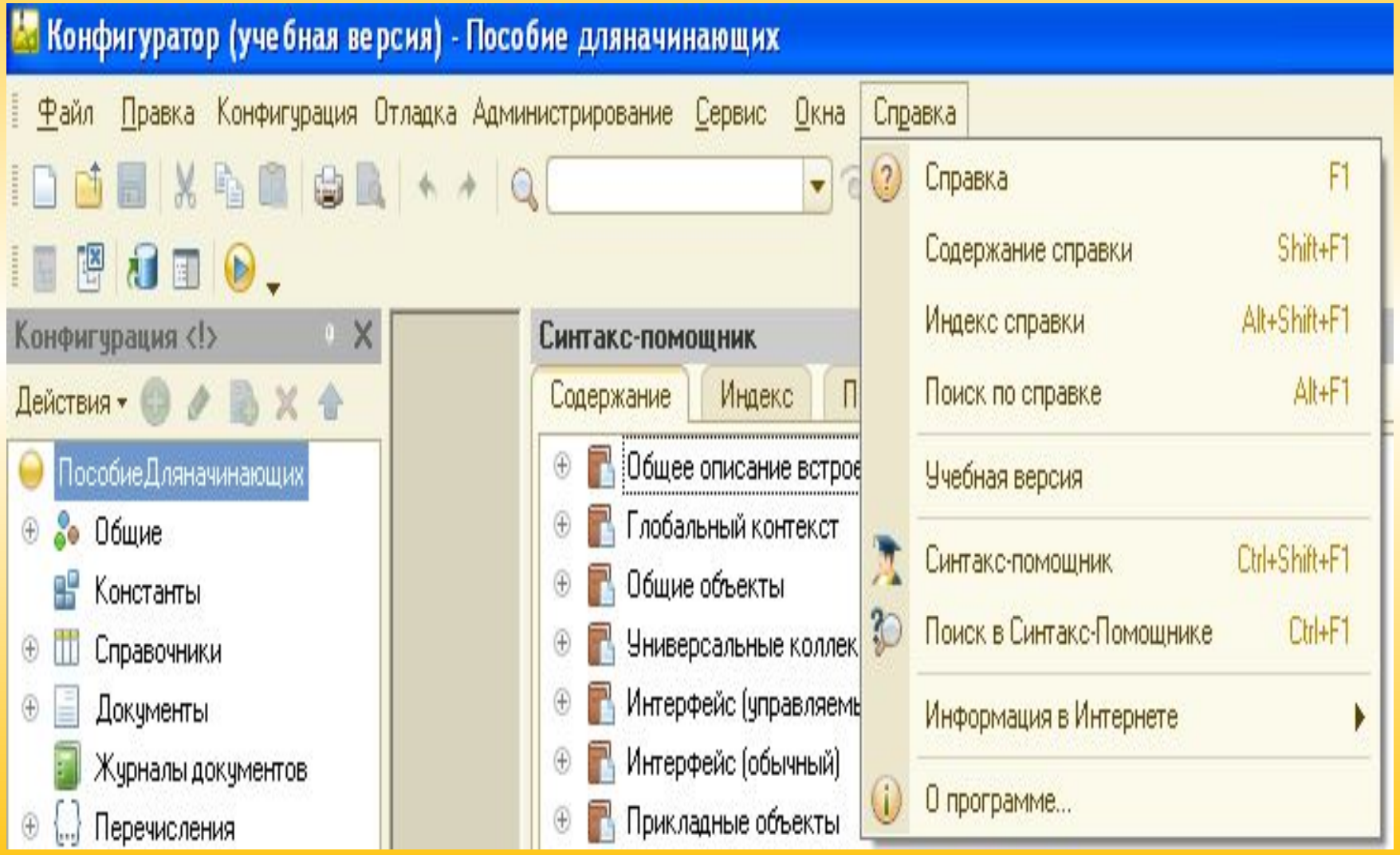
Синтакс-помощник

Синтакс-помощник

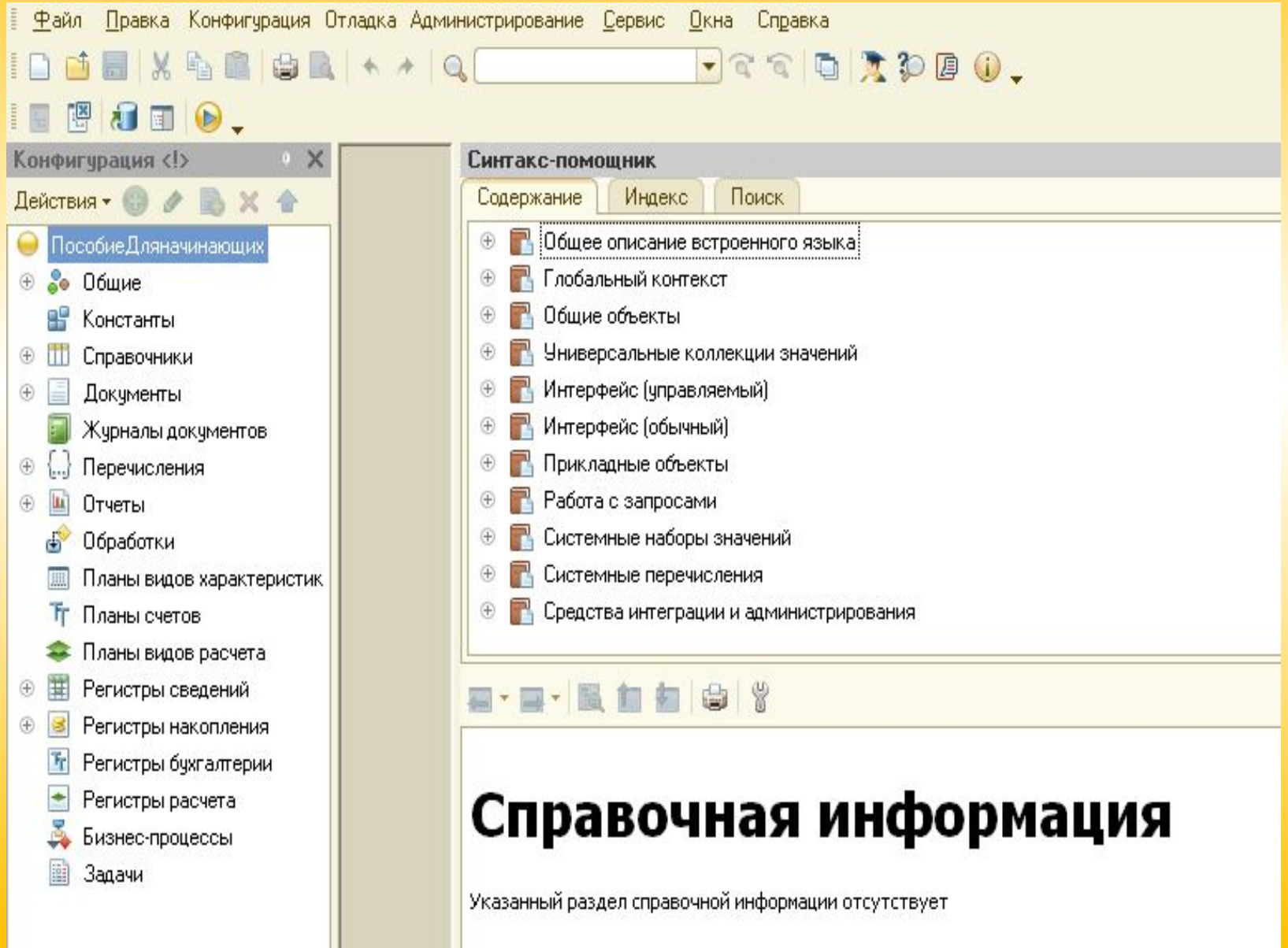
- В режиме Конфигуратора из пункта Справка главного меню можно вызвать Синтакс-помощник.
- Синтакс-помощник – инструмент, созданный для помощи разработчику, содержащий описание всех программных объектов, которые использует система, их методов, свойств, событий и т.д.
- Содержание Синтакс-помощника полностью дублирует описание встроенного языка в семи томах, входящее в стандартный комплект поставки 1С:Предприятие



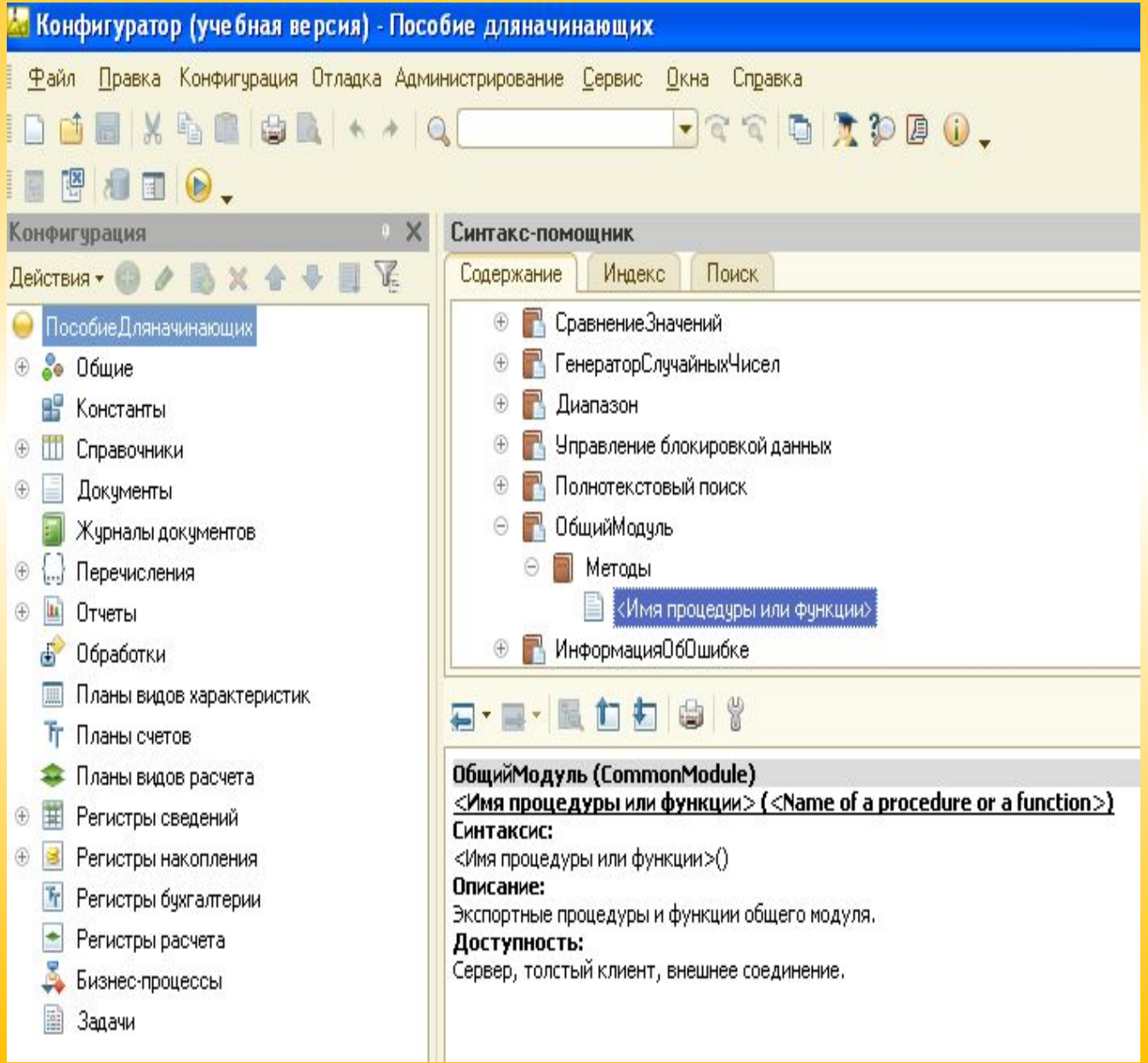
Синтакс-помощник



Синтакс-помощник

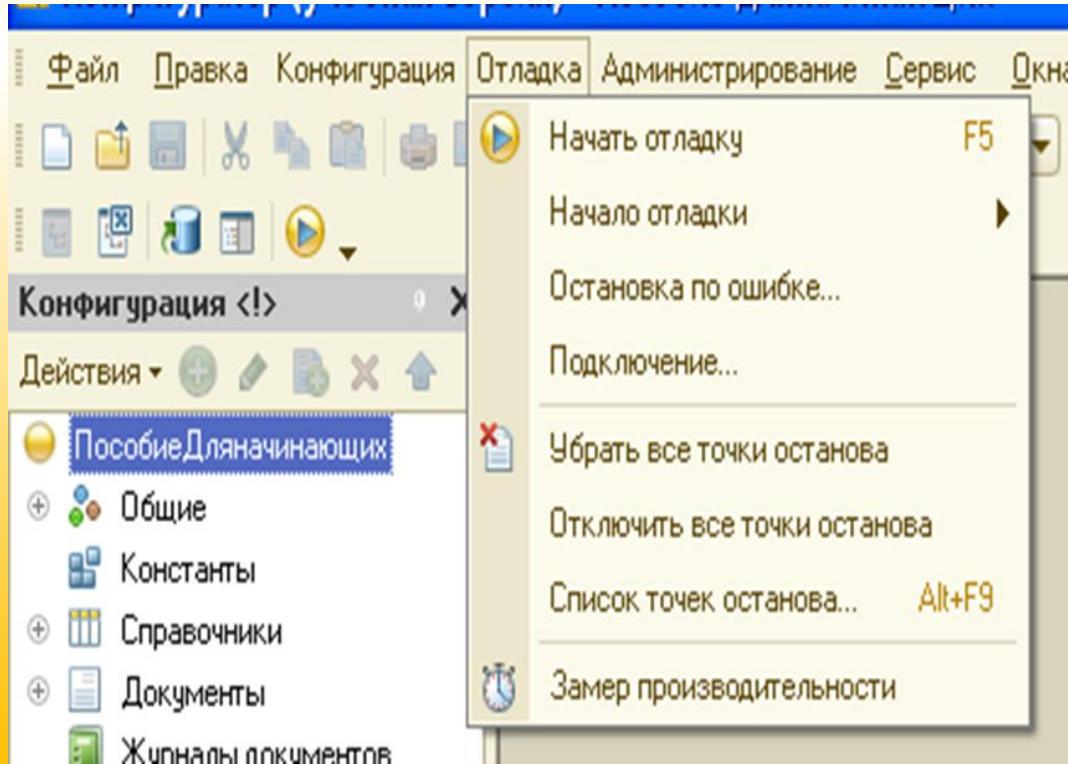


Синтакс-помощник

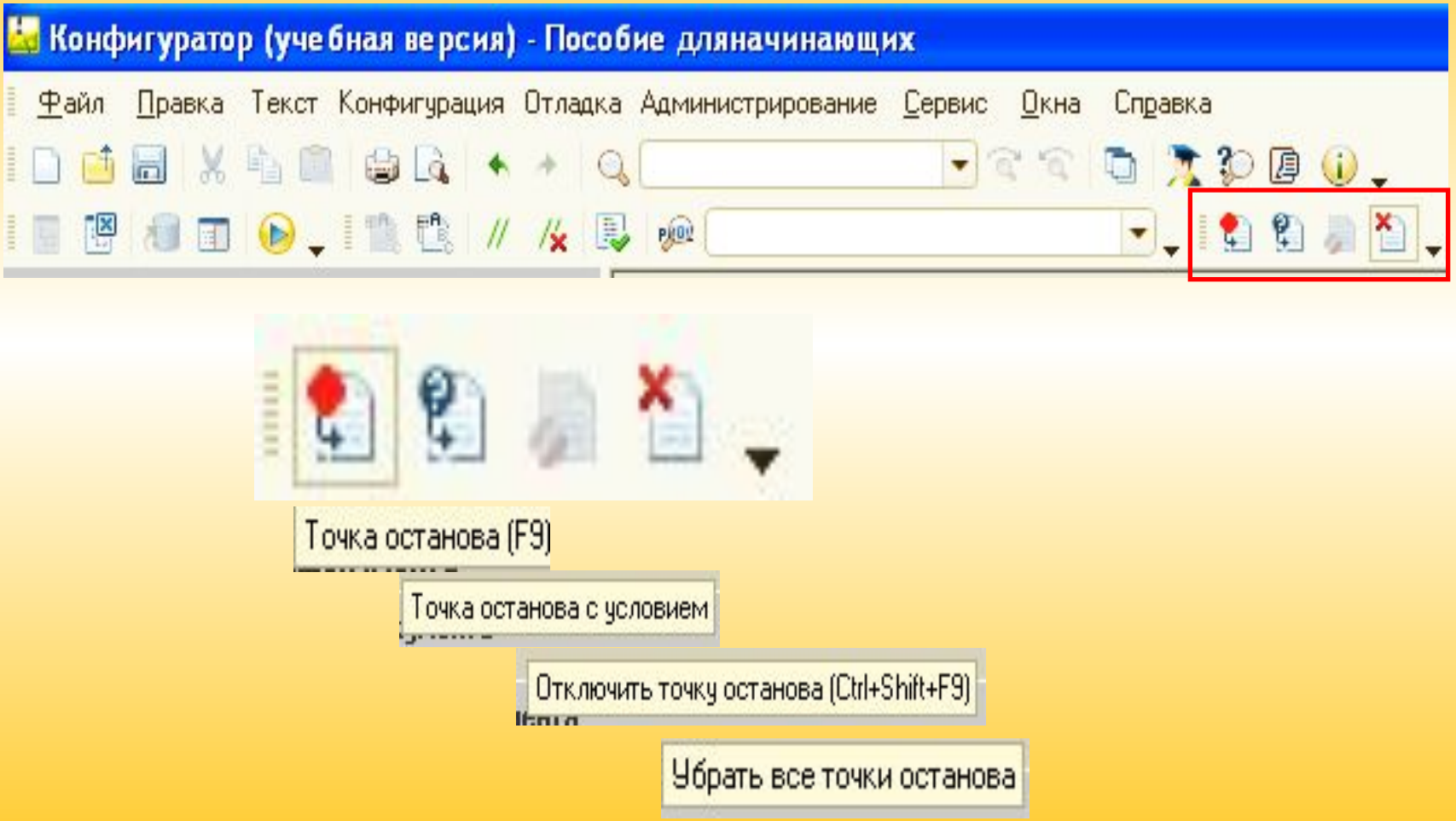


Отладчик

- Отладчик – вспомогательный инструмент технологической платформы, облегчающий разработку и отладку программных модулей системы 1С: Предприятие.



Расстановка точек останова



Типы данных

Примитивные (базовые) типы данных

- В компьютерных системах существуют различные способы работы с данными и их представления.
- В системе 1С:Предприятие в памяти компьютера хранятся не только данные, но и информация о том, как с ними работать. Информация о способе ввода и представлении вводимых данных называется типом данных:

Данные	Тип
1	Число
«Пример»	Строка
Истина	Булево
21.12.2020 0:00:00	Дата

ПРИМИТИВНЫЕ ТИПЫ ДАННЫХ

Во встроенном языке системы «1С:Предприятие» поддерживается набор ***примитивных типов данных***:

- СТРОКА
- ЧИСЛО
- ДАТА
- БУЛЕВО
- НЕОПРЕДЕЛЕНО
- NULL
- ТИП

ПРИМИТИВНЫЕ ТИПЫ ДАННЫХ

Для большинства примитивных типов данных предусмотрена возможность использования в тексте модуля литералов, т.е. значений соответствующих типов.

// Пример использования литерала типа Строка

А = "Моя строка"; //строка заключается в двойные кавычки

// Пример использования литерала типа Булево

Б = истина;

// Пример использования литерала типа Число

В = 12345.6789; //числа имеют знак плюс и минус

// Пример использования литерала типа Дата

С=‘19571004’ //Дата всегда заключается в одинарные кавычки

Тип значения Строка (String)

Значения данного типа содержат строку произвольной длины в формате Unicode.

Литералы:

Литералы строкового типа представляют собой набор символов, заключенных в двойные кавычки.

// Пример строки

моястрока = "Это правильная строка";

Для указания кавычки в строковом значении необходимо задавать двойную кавычку (**" "**)

Пример:

A= " Город проживания " "Москва " " ";

Тип значения Строка (String)

Многострочные строковые константы задаются с использованием символа **|** (вертикальная черта).

```
// Пример многострочной строки
МояМногострочнаястрока = "Это
| правильная
| многострочная
| строка";
```

Тип значения Число (Number)

Числовым типом может быть представлено любое ***десятичное число***.

ВНИМАНИЕ!

Максимально допустимая разрядность числа - ***32 знака***.

Литералы: Набор цифр, написанных непосредственно в тексте модуля вида:

[+|-]{0|1|2|3|4|5|6|7|8|9}[.{0|1|2|3|4|5|6|7|8|9}

В качестве ***разделителя целой и дробной части*** используется ***точка***.

Пример:

А =15;

Б = -968.612;

Для определения отрицательного числа перед ним ставится знак – (минус).

Тип значения Дата и время (Date)

Значения данного типа содержат ***дату от Рождества Христова (с 01 января 0001 года) и время с точностью до секунды***.

Литералы:

Строка цифр, заключенная в одинарные кавычки вида 'ГГГГММДДччммсс', где:

- ГГГГ - четыре цифры года (включая тысячелетие и век);
- ММ - две цифры месяца;
- ДД - две цифры даты;
- чч - две цифры часа (в 24-часовом формате);
- мм - две цифры минут;
- сс - две цифры секунд.

Во встроенном языке в литерале типа Дата *обязательно должно задаваться значение года, месяца и дня*. Для задания даты, соответствующей началу отсчета, достаточно указать '00010101'. Допускается при указании литералов типа Дата опускать последние символы (секунды, минуты, часы). Это означает, что данные параметры будут равны нулю.

В литерале даты допускается использование различных разделителей: «/», «\», «-», «:», « »

Пример:

Дата('2008.03.23 10:45:23') = "23.03.2008 10:45:23";

Тип значения Булево (Boolean)

Значения данного типа имеют два значения - **Истина и Ложь**, задаваемые соответствующими литералами. *Значения данного типа возвращаются в качестве результата вычисления логических выражений.*

Литералы:

истина (True), ложь (False).

A=истина;

Тип значения Неопределено *(Undefined)*

Значение данного типа применяется, когда необходимо использовать **пустое значение**, ***не принадлежащее ни к одному другому типу.*** Например, такое значение изначально имеют реквизиты с составным типом значения.

Существует одно-единственное значение данного типа, задаваемое литералом.

Литералы:

неопределено (*Undefined*)

А = неопределено;

Тип значения NULL

Значения данного типа означает **пустое значение с незаданным типом в базе данных**. Используются исключительно *для определения отсутствующего значения при работе с базой данных*, например, при соединении таблиц.

Литералы:

Null

Тип значения ТИП

- Значение данного типа используются **для идентификации типов значений.** Это необходимо для определения и сравнения типов.
- *Данный тип не имеет литералов* и возвращается функциями встроенного языка «типЗнч» и «Тип».

Выражения и операторы

Выражения

Строковое представление формулы, предназначенной для выполнения действия над данными, называется **Выражением**. В зависимости от операций и значений выражения можно разделить на 3 группы:

- **Арифметические операции**
- **Операции конкатенации**
- **Логические операции**

ВЫРАЖЕНИЯ ЯЗЫКА
1. Арифметические операции

Название	Выражение
Сложение	**(Оп1 + Оп2)**
Вычитание	**(Оп1 -Оп2)**
Умножение	**(Оп1 * Оп2)**
Деление	**(Оп1 /Оп2)**
Остаток от деления	**(Оп1 % Оп2)**
Унарный минус	**(- Оп1)**

Операция конкатенации

Операция конкатенации («+») используется для того, чтобы присоединить одну строку к другой. Длина результирующей строки равна сумме длин соединяемых строк. В случае несовпадения типа данных второго или последующих операндов со строковым типом их значение преобразуется к строковому типу в соответствии с правилами преобразования типов.

ФИО = Фамилия + " " + Имя + " " + Отчество;

Логические операции
Операции сравнения

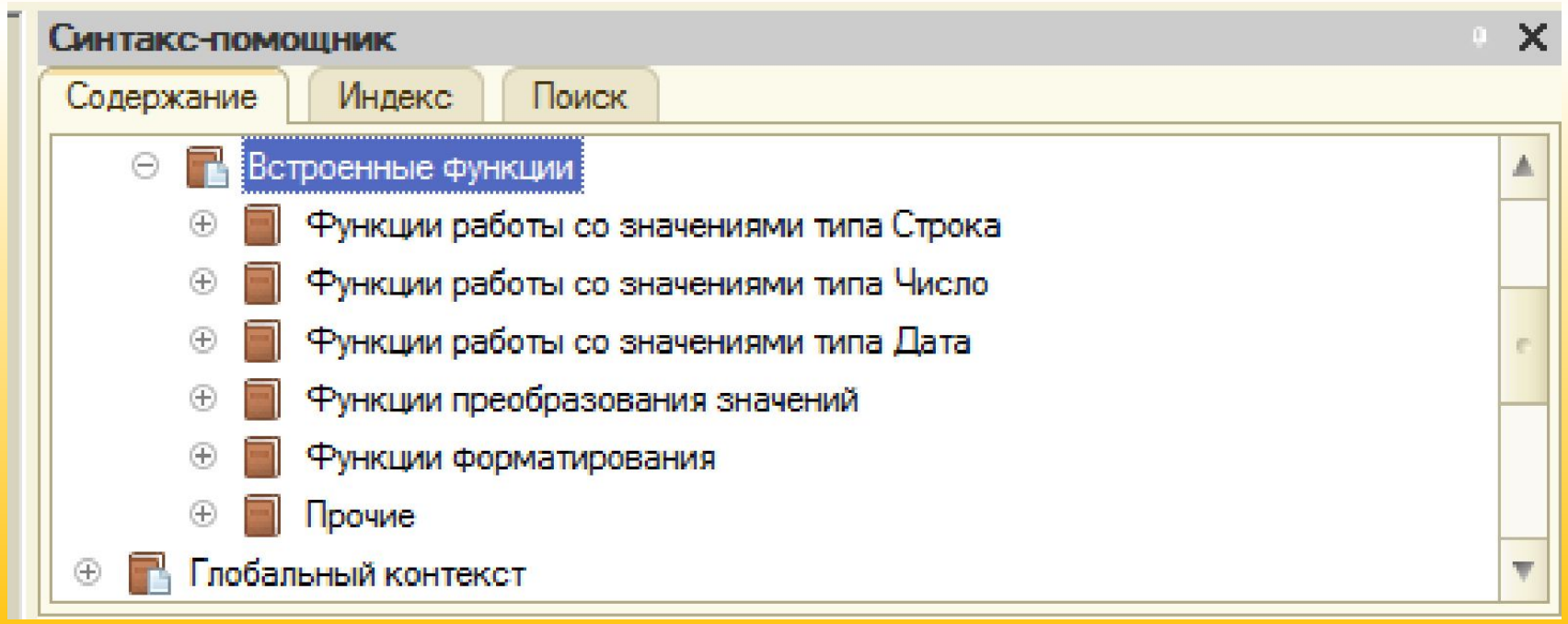
Операция	**Выражение операции**
Больше	**Оп1 >Оп2**
Больше или равно	**Оп1 >= Оп2**
Равно	**Оп1 = Оп2**
Не равно	**Оп1 <> Оп2**
Меньше	**Оп1 < Оп2**
Меньше или равно	**Оп1 <=Оп2**

Булевы операции

И (AND)	**конъюнкция (булево И)**
ИЛИ (OR)	**дизъюнкция (булево ИЛИ)**
НЕ (NOT)	**логическое отрицание (булево отрицание НЕ)**

Встроенные функции работы со значениями

- Встроенные функции встроены в технологическую платформу 1С: Предприятие 8. Описание правил использования и набор таких функций можно найти в синтакс-помощнике в режиме Конфигуратор:

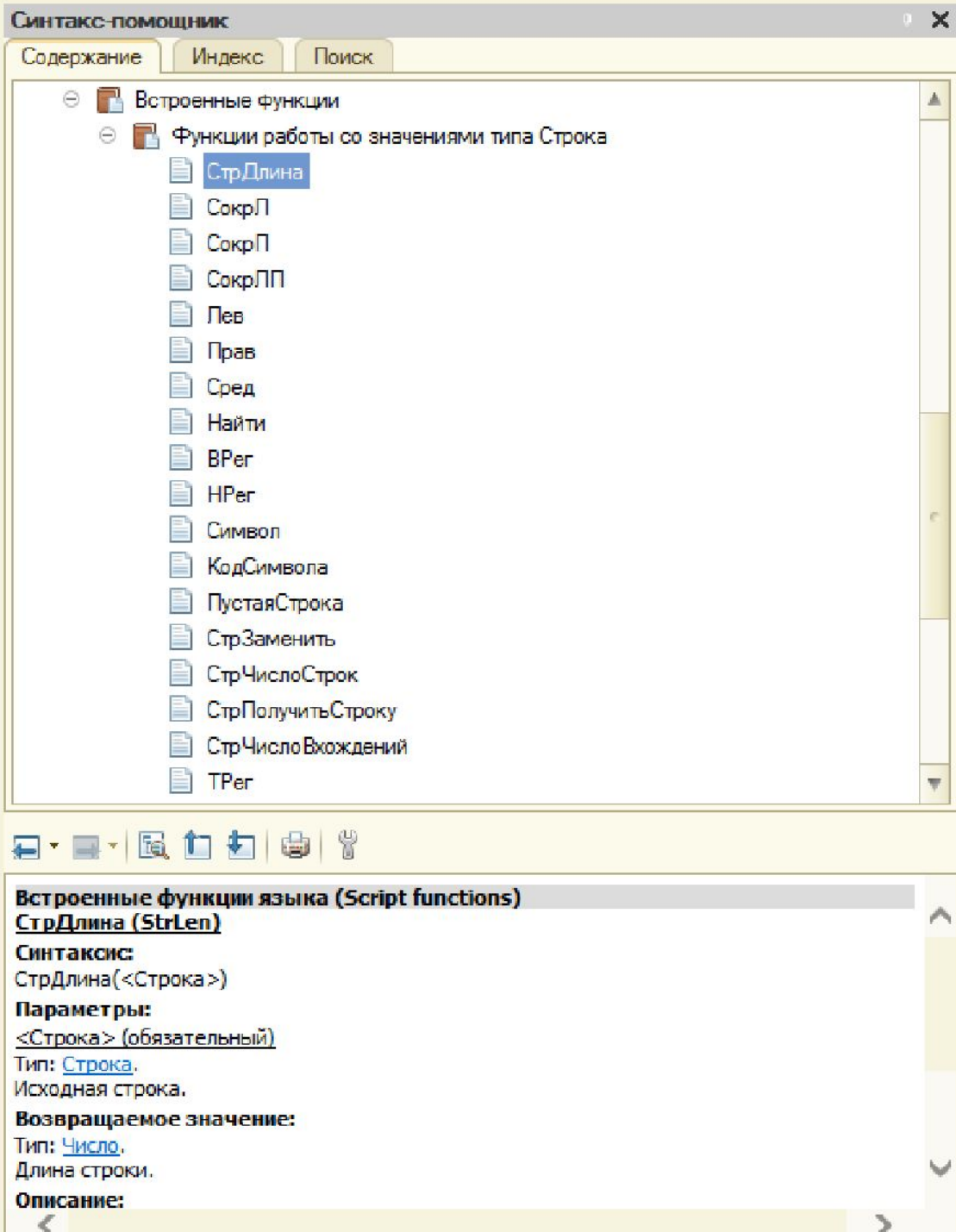


Встроенные функции работы со значениями

- ***Для строковых значений***
  - Найти слово в предложении
  - Определить длину строки
  - Определить количество строк в тексте и т.д.
- ***Для числовых значений***
  - Произвести округление числа
  - Возвести в степень
  - Получить целую часть числа и т.д.
- ***Для значений типа дата***
  - Получить составную часть даты (год, месяц и т.д.)
  - Получить значение системной (текущей) даты и т.д.

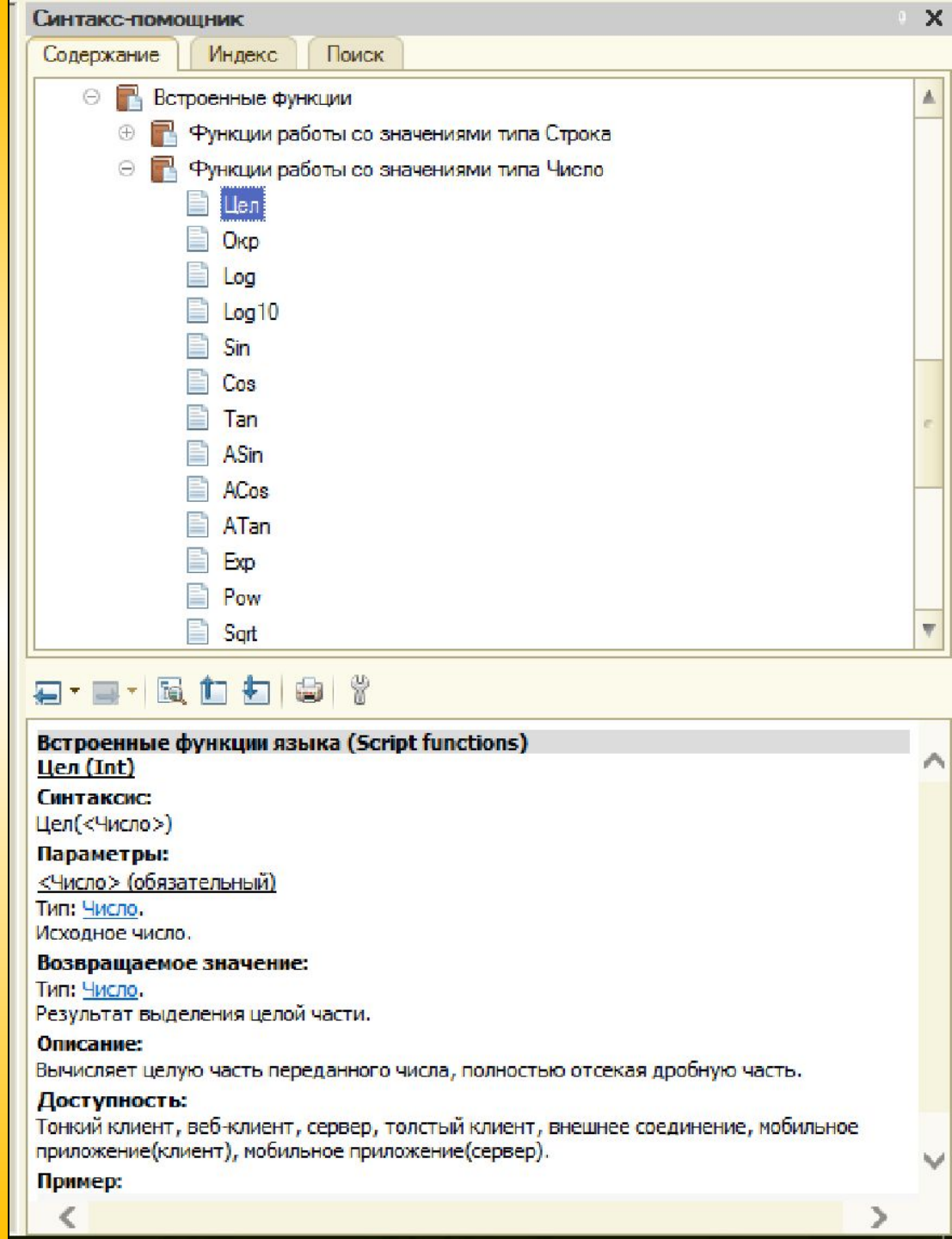
Функции работы со значениями типа Строка

```
А="Университет";
СтрДлина(А);
//вернет 11
```



Функции работы со значениями типа Число

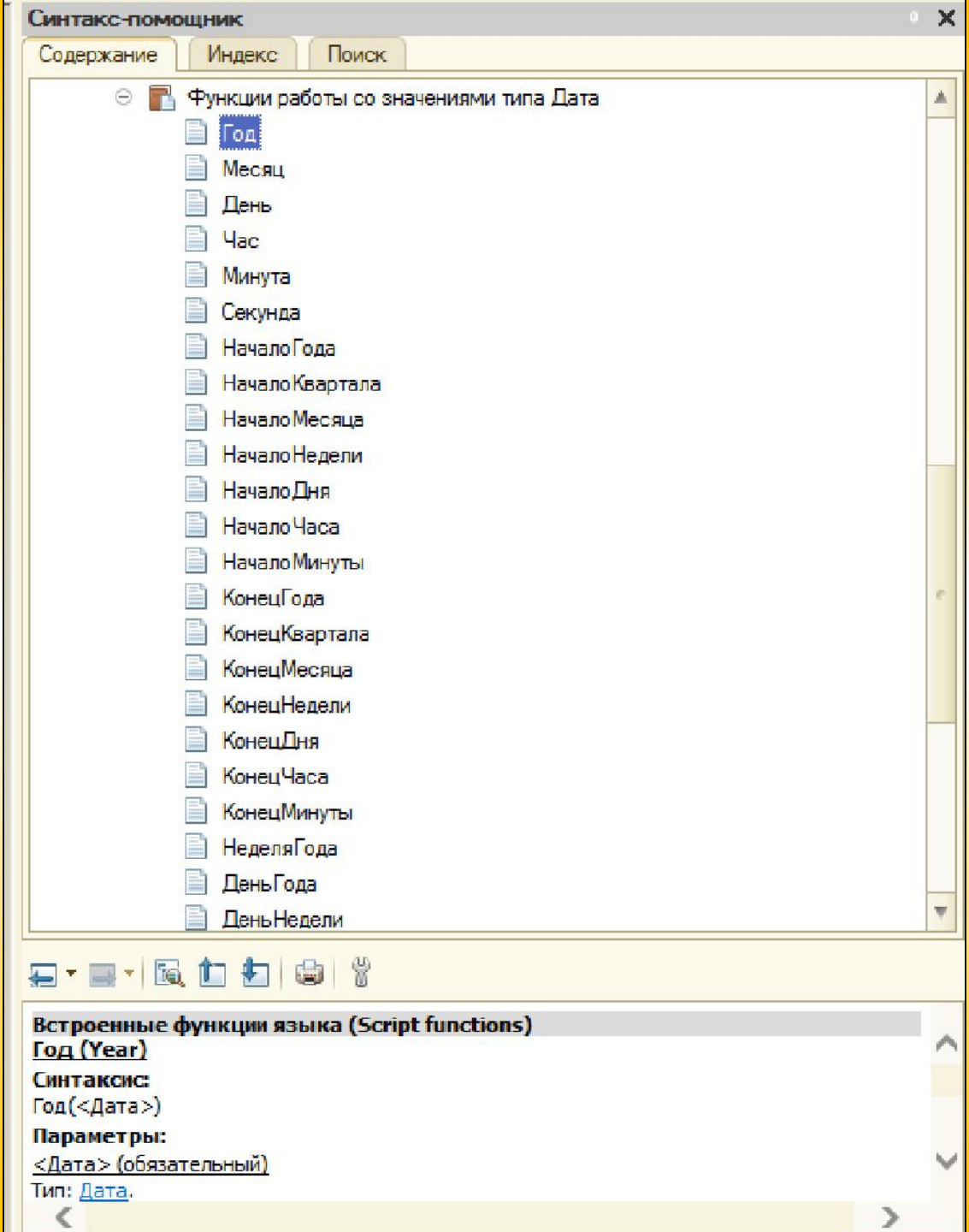
Пример:
Цена=100.45;
ОкругленнаяЦена=Окр(Цена,1);
// вернет 100.5
По умолчанию функция Окр
Делает округление в большую
Сторону
ОкругленнаяЦена=Окр(Цена,1,0);
// вернет 100.4
Цел(Цена);
// вернет 100



Функции работы со значениями типа Дата

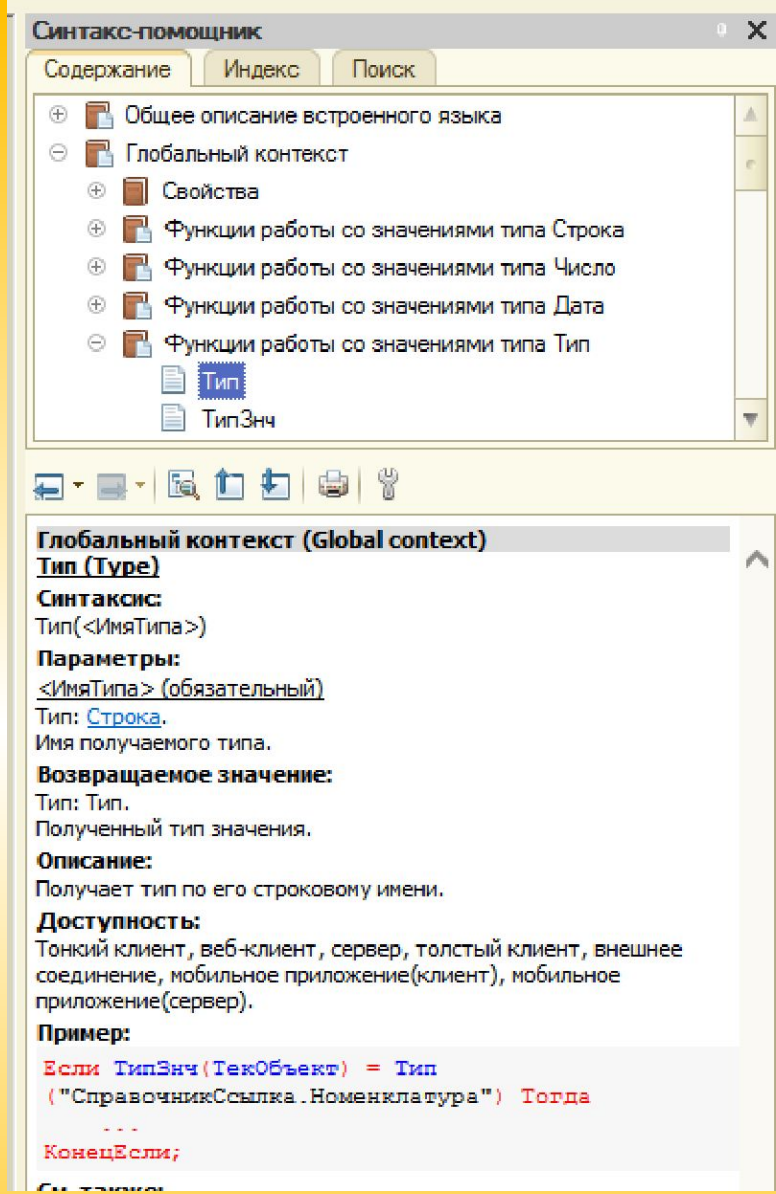
Пример:

```
Сегодня=ТекущаяДата();
Сообщить(Год(Сегодня));
Сообщить(Месяц(Сегодня));
Сообщить(День(Сегодня));
```



Функции работы со значениями типа Тип

- Функция Тип возвращает тип значения по его строковому имени
- Функция ТипЗнч возвращает тип значения

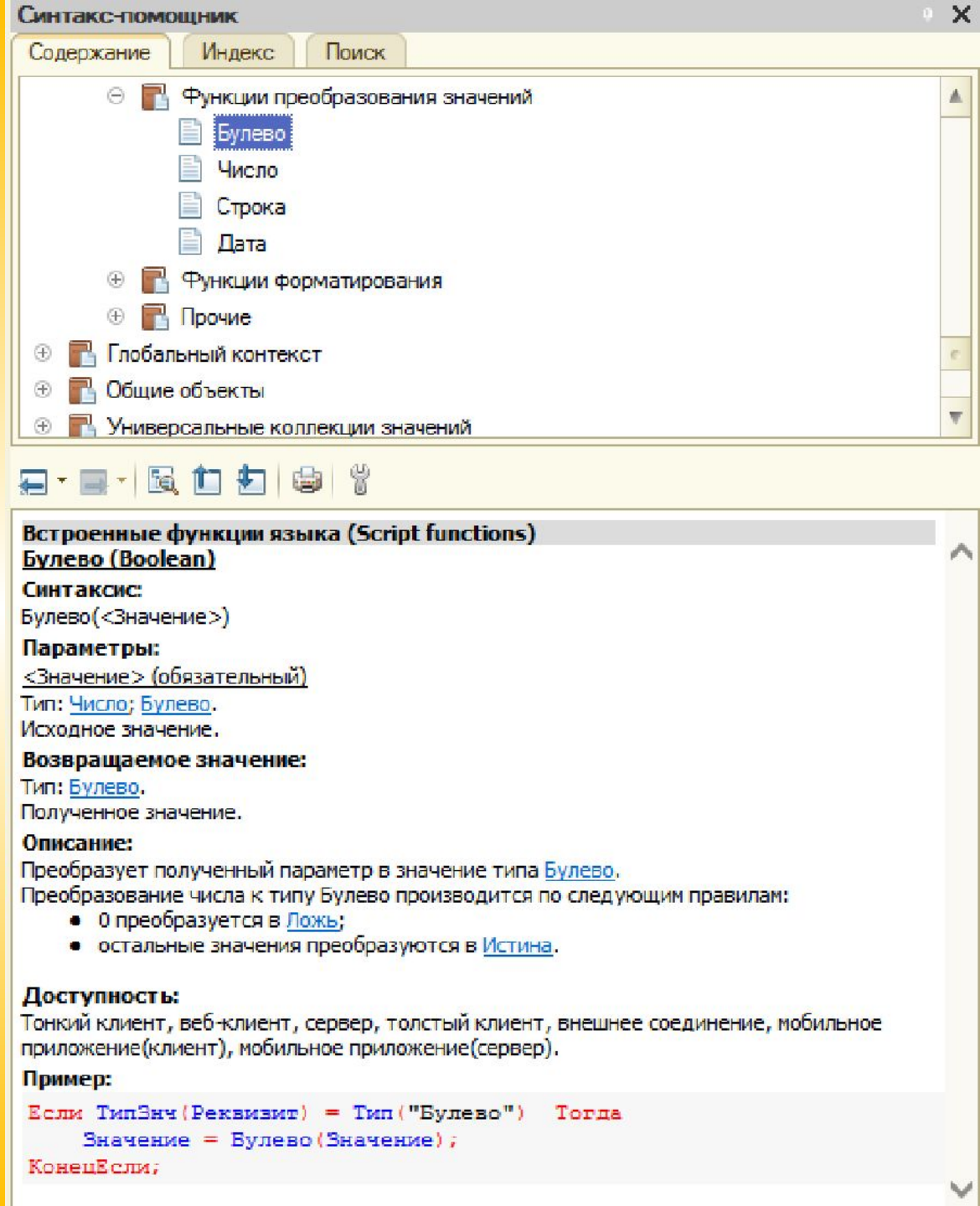


Пример:

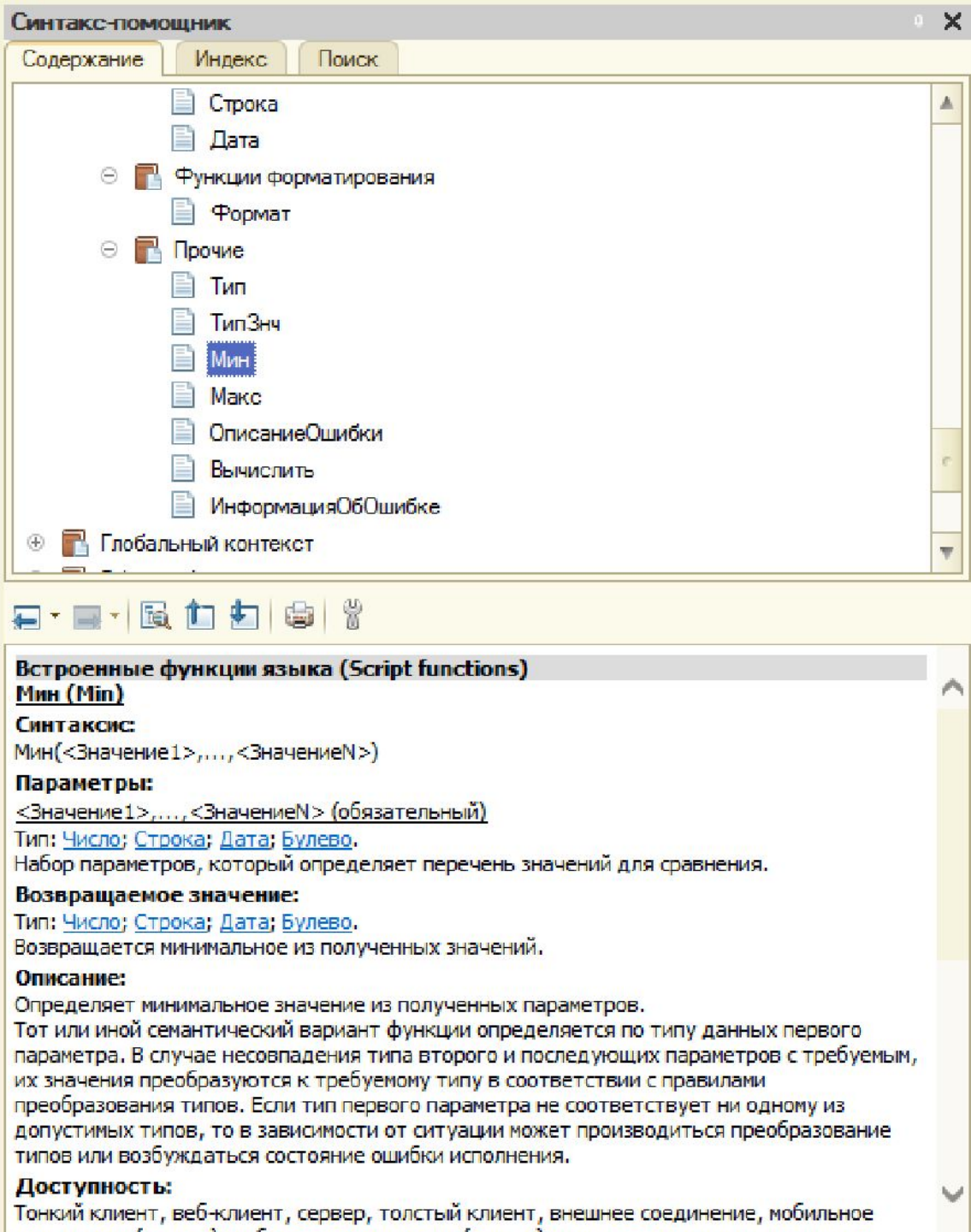
```
Если ТипЗнч(ТекОбъект) = Тип("СправочникСсылка.
Номенклатура") Тогда
    ...
КонецЕсли;
```

Функции преобразования значений

```
А=0;
Булево(А);
//вернет Ложь
А=5;
Булево(А);
//вернет Истина
```



Функции форматирования и Прочие



Работа с переменными

- Для построения универсальных алгоритмов, которые позволяют обрабатывать различные значения входных данных, в выражениях и операторах необходимо использовать **переменные**. Для обращения к ним используются **имена переменных.**

Например:

B=A+5;

//A, B - это переменные

Имена переменных, процедур и функций

Именем переменной, объявленной процедуры или функции может быть ***любая последовательность букв, цифр и знаков подчеркивания, начинающаяся с буквы или знака подчеркивания.***

Вновь создаваемые ***имена не должны совпадать с зарезервированными словами языка или именами свойств***, непосредственно доступных в текущем контексте.

Распознавание имен переменных, процедур и функций ведется ***без учета регистра букв***.

Являются ли допустимыми следующие имена переменных 1С?

1. ПервыйКосмонавт
2. 7ЦветовРадуги
3. СемьЦветовРадуги
4. Цветов Радуги7
5. Гора”Эверест”
6. Цветов_радуги_7
7. _7ЦветовРадуги
8. Северный<полюс>

ОПЕРАТОРЫ И СИНТАКСИЧЕСКИЕ КОНСТРУКЦИИ

Формат операторов

Операторы языка в программном модуле можно подразделить на две категории:

- *операторы объявления переменных,*
- *исполняемые операторы.*

***Операторы объявления переменных создают имена переменных, которыми манипулируют исполняемые операторы*.**

Перем (Var)

Описание:

Позволяет в явном виде объявить переменную.

Синтаксис:

Перем <Имя переменной 1> [Экспорт] [, <Имя переменной 2>, ...];

Параметры:

<Имя переменной 1>[, <Имя переменной 2>, ...]

Задается имя или имена объявляемых переменных

Экспорт

Необязательное ключевое слово. Указывает, что данная переменная доступна при обращении из других модулей. Данное ключевое слово необходимо указывать для каждой объявляемой переменной отдельно. Не имеет смысла при объявлении переменных отдельных процедур или функций.

Перем (Var)

Пример:

// Пример объявления одной переменной

Перем А Экспорт;

Перем Б;

//пример объявления нескольких переменных одним оператором

Перем А, Б Экспорт;

Неявное объявление переменных:

В языке необязательно объявлять переменные в явном виде. Неявным определением переменной является **первое ее появление в левой части оператора присваивания**. Тип переменной определяется типом присвоенного ей значения.

Не допускается использование в выражениях переменных, не объявленных ранее в явном или неявном виде.

ОПЕРАТОР ПРИСВАИВАНИЯ

Оператор присваивания (=)

Описание:

Оператор присваивания (символ «=») означает присваивание значения <Источник> переменной, обозначенной как <Назначение>.

Синтаксис:

<Назначение> =<Источник>;

Пример:

```
А = в;
Стр1 = "777";
ДатаДокумента = '20020717';
```

Формат операторов

Любой исполняемый оператор может иметь **метку**, используемую в качестве точки перехода в операторе Перейти.

В общем случае формат оператора языка следующий:

~метка:оператор[(параметры)] [добключевоеслово];

В качестве меток используются специальные идентификаторы, начинающиеся с символа тильда и состоящие из последовательности букв, цифр и символов подчеркивание. ***Чтобы пометить оператор, нужно поместить перед ним метку и следующий за ней символ двоеточие.***

~метка:А=в;

Новый (New)

Описание:

Оператор позволяет создать ***значение указанного типа***.

Синтаксис:

Новый <Имя типа>[(<Парам 1>, ..., <Парам N>)]

Параметры:

Имя типа

Указывается имя типа, значение которого создается.

<Парам 1>, ..., <Парам N>

После имени типа в скобках могут указываться параметры, если они определены в конструкторах для данного типа. Допустимое количество параметров и их назначение указываются в описании конструкторов объекта.

Пример:

// Пример создания массива из трех элементов.

Массив = Новый Массив(3);

Синтаксические конструкции

Самые простые алгоритмы описывают линейный процесс.

Например:

1. A=3;
2. B=7;
3. C=A+B;

Синтаксические конструкции

В реальности могут возникнуть более сложные алгоритмы:

1. **С условиями** – выполнение действий зависит от каких-либо условий.

   Например алгоритм перехода улицы пешеходом: Если цвет светофора= «зеленый», то «идем», иначе «стоим».

   2. С циклами – применяется, если последовательность действий может повторяться некоторое количество раз.

   Например алгоритм работы светофора включает циклически повторяющиеся действия: горит зеленый свет, горит желтый свет, горит красный свет.

   3. С переходами – применяется в алгоритмах для обхода некоторой последовательности действий.

   Для реализации этих нелинейных алгоритмов во встроенном языке 1С:Предприятие предусмотрены исполняемые операторы.

Работа с условиями

Система 1С:Предприятие поддерживает два вида таких операторов:

1. ? (вычислить выражение по условию)
2. Если (if)

? (вычислить выражение по условию)

Описание:

Позволяет вычислить одно из двух заданных выражений в зависимости от результата вычисления логического выражения.

Синтаксис:

?(<логическое выражение>, <Выражение 1>, <Выражение 2>)

Параметры:

<логическое выражение>

Логическое выражение, результат вычисления которого определяет одно из результирующих выражений, которые будут вычислены. Если результат его вычисления Истина, то будет вычисляться **<Выражение 1>**. Если результат Ложь, то **<Выражение 2>.**

Пример:

```
Статус = ?(ПолучитьСкидку() > 10,  "Особый клиент",  "Обычный клиент")
```

Если (if)

Описание:

Оператор *Если* управляет выполнением программы, основываясь на результаты одного или более логических выражений. Оператор может содержать любое количество групп операторов, возглавляемых конструкциями *ИначеЕсли - Тогда.*

Синтаксис:

Если <Логическое выражение> Тогда
// Операторы
[ИначеЕсли <Логическое выражение> Тогда]
// Операторы
[Иначе]
// Операторы
КонецЕсли;

Параметры:

<Логическое выражение>

Логическое выражение.

Тогда

Операторы, следующие за Тогда, выполняются, если результатом логического выражения является значение истина.

// Операторы

Исполняемый оператор или последовательность таких операторов.

ИначеЕсли

Логическое выражение, следующее за ключевым словом ИначеЕсли, вычисляется только тогда, когда условия в Если и всех предшествующих ИначеЕсли оказались равны Ложь. Операторы, следующие за конструкцией ИначеЕсли - Тогда, выполняются, если результат логического выражения в данном ИначеЕсли равен истина.

Иначе

Операторы, следующие за ключевым словом Иначе, выполняются, если результаты логических выражений в конструкции Если и всех предшествующих конструкциях ИначеЕсли оказались равны Ложь.

КонецЕсли

Ключевое слово, которое завершает структуру оператора условного выполнения.

Пример:

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

Работа с циклами

Система 1С:Предприятие поддерживает три вида таких операторов:

1. Для (For)
2. Пока (While)
3. Для каждого (For each)

Для (For)

Описание:

Оператор цикла Для предназначен для циклического повторения операторов, находящихся внутри конструкции Цикл - КонецЦикла. Перед началом выполнения цикла значение *<Выражение 1>* присваивается переменной *<Имя переменной>* Значение <Имя переменной> автоматически увеличивается при каждом проходе цикла. Величина приращения счетчика при каждом выполнении цикла равна 1. Цикл выполняется, пока значение переменной *<Имя переменной>* меньше или равно значению *<Выражение 2>*. Условие выполнения цикла всегда проверяется вначале, перед выполнением цикла.

Синтаксис:

Для <Имя переменной> = <Выражение 1> По <Выражение 2>
Цикл
// Операторы
[Прервать;]
// Операторы
[Продолжить;]
// Операторы
КонецЦикла;

Для (For)

Параметры:

<Имя переменной>

Счетчик цикла.

<Выражение 1>

Числовое выражение, которое задает начальное значение, присваиваемое счетчику цикла при первом проходе цикла.

<Выражение 2>

Максимальное значение счетчика цикла. Когда переменная <Имя переменной> становится больше чем <Выражение 2>, выполнение оператора цикла Для прекращается.

Цикл

Операторы, следующие за ключевым словом Цикл, выполняются, пока значение переменной <Имя переменной> меньше или равно значению <Выражение 2>.

// Операторы

Исполняемый оператор или последовательность таких операторов.

Прервать

Позволяет прервать выполнение цикла в любой точке. После выполнения этого оператора управление передается оператору, следующему за ключевым словом КонецЦикла.

Продолжить

Немедленно передает управление в начало цикла, где производится вычисление и проверка условий выполнения цикла. Операторы, следующие в теле цикла за ним, на данной итерации обхода не выполняются.

КонецЦикла

Ключевое слово, которое завершает структуру оператора цикла

Пример:

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

Пока (While)

Описание:

Оператор цикла Пока предназначен для циклического повторения операторов, находящихся внутри конструкции Цикл - КонецЦикла. Цикл выполняется, пока логическое выражение равно Истина. Условие выполнения цикла всегда проверяется вначале, перед выполнением цикла.

Синтаксис:

Пока <Логическое выражение> Цикл
// Операторы
[Прервать;]
// Операторы
[Продолжить;]
// Операторы
КонецЦикла;

Пример:

```
ВыборкаДок = Документы. РасходнаяНакладная.
    Выбрать ();
// Цикл по всем документам
Пока ВыборкаДок.Следующий() Цикл
// Отобразим документ в панели состояния
Состояние("Обрабатывается документ №" +
    ВыборкаДок.Номер);
// Операторы выполнения действий над
    документом
. . .
КонецЦикла;
```

Для каждого (For each)

Описание:

Оператор цикла Для каждого предназначен для циклического обхода коллекций значений. При каждой итерации цикла возвращается новый элемент коллекции. Обход осуществляется до тех пор, пока не будут перебраны все элементы коллекции, или может быть завершен досрочно при выполнении оператора Прервать.

Синтаксис:

Для каждого <Имя переменной 1> Из <Имя переменной 2> Цикл
// Операторы
[Прервать;]
// Операторы
[Продолжить;]
// Операторы
КонецЦикла

Параметры:

<Имя переменной 1>

Переменная, которой при каждом повторении цикла присваивается значение очередного элемента коллекции.

<Имя переменной 2>

Переменная или выражение, предоставляющее коллекцию. Элементы этой коллекции будут присваиваться параметру <Имя переменной 1>.

Цикл

Операторы, следующие за ключевым словом Цикл, выполняются до тех пор, пока не будут перебраны все элементы коллекции.

// **Операторы**

Исполняемый оператор или последовательность таких операторов.

Прервать

Позволяет прервать выполнение цикла в любой точке. После выполнения этого оператора управление передается оператору, следующему за ключевым словом КонецЦикла.

Продолжить

Немедленно передает управление в начало цикла, где производится вычисление и проверка условий выполнения цикла. Операторы, следующие в теле цикла за ним, на данной итерации обхода не выполняются.

КонецЦикла

Ключевое слово, которое завершает структуру оператора цикла.

Пример:

```
// Перебор строк табличной части документа.
Документ = Документы. РасходнаяНакладная.
    НайтиПоКоду (12345);
// Проверим, найден нужный нам документ или нет
Если Не Документ.Пустая() Тогда
Для каждого СтрокаСостава Из Документ.Состав Цикл
Состояние("Строка: " + Документ.Состав.Индекс
    (СтрокаСостава)+1);
// Операторы обработки очередной строки табличной
    части
КонецЦикла;
КонецЕсли;
```

Работа с переходами

Система 1С:Предприятие поддерживает три вида таких операторов:

Перейти (Goto) – безусловный переход

Прервать – переход в конструкциях цикла к оператору, следующему за концом цикла, позволяет прервать цикл.

Продолжить - переход в конструкциях цикла к началу цикла, позволяет прервать только итерацию цикла.

Перейти (Goto)

Описание:

Безусловная передача управления на другой оператор программы. Передает управление от одного оператора к другому.

Область действия оператора ограничивается программным модулем, процедурой или функцией; он не может передать управление за пределы программного модуля, процедуры или функции.

ПРИМЕЧАНИЕ 1

Метка в этом операторе не должна быть меткой перехода на оператор Процедура или Функция.

ПРИМЕЧАНИЕ 2

Оператор безусловного перехода не может быть использован для передачи управления на операторы, находящиеся внутри конструкций: Пока - Конеццикла, Для - Конеццикла, Для каждого - Конеццикла, Если - КонецЕсли, Попытка - Исключение - Конецпопытки извне этих конструкций.

Синтаксис:

Перейти <метка>;

Пример:

```
Перейти  ~метка1;
…
~ метка1: Сообщить ("Осуществлен переход по метке.");
```

Процедура (*Procedure)*

Описание:

Процедура <ИмяПроцедуры> (< Список параметров >)

…

КонецПроцедуры

Ключевое слово Процедура начинает ***секцию исходного текста, выполнение которого можно инициировать из любой точки программного модуля, просто указав ИмяПроцедуры() со списком параметров*** (круглые скобки обязательны даже если параметры не передаются).

Переменные, объявленные в теле процедуры в разделе Объявления локальных переменных, являются локальными переменными данной процедуры, поэтому доступны только в этой процедуре (за исключением случая передачи их как параметров при вызове других процедур, функций или методов).

ПРИМЕЧАНИЕ

Ключевые слова **Процедура, КонецПроцедуры** являются не операторами, а операторными скобками, поэтому не должны заканчиваться точкой с запятой

Синтаксис:

Процедура<ИмяПроцедуры>([[Знач] <Парам1>
[=<ДефЗнач>],…,[Знач]
<ПарамN> [=<ДефЗнач>]])[Экспорт] //Объявления локальных переменных; //Операторы;
. . .
[Возврат;]
// Операторы;
. . .
КонецПроцедуры

Если в теле описания процедуры использовано ключевое слово **Экспорт**, то это означает, что данная процедура является доступной из всех других программных модулей конфигурации.

При выполнении оператора **Возврат** процедура заканчивается и возвращает управление в точку вызова. Если в тексте процедуры не встретился оператор Возврат, то после выполнения последнего исполняемого оператора происходит выполнение неявного оператора Возврат. Конец программной секции процедуры определяется по оператору КонецПроцедуры.

Параметры:

<ИмяПроцедуры> назначает имя процедуры.

Знач

Необязательное ключевое слово, которое указывает на то, что следующий за ним **параметр передается по значению**, т. е. изменение значения формального параметра при выполнении процедуры никак не повлияет на фактический параметр, переданный при вызове процедуры. Если это ключевое слово не указано, то параметр процедуры передается по ссылке, то есть изменение внутри процедуры значения формального параметра приведет к изменению значения соответствующего фактического параметра.

<парам1>, ..., <парамN>

Необязательный **список формальных параметров, разделяемых запятыми**. Значения формальных параметров должны соответствовать значениям передаваемых при вызове процедуры фактических параметров. В этом списке определяются имена каждого из параметров так, как они используются в тексте процедуры. Список формальных параметров может быть пуст.

=<ДефЗнач>

Необязательная **установка значения параметра по умолчанию**. Параметры с установленными значениями по умолчанию можно располагать в любом месте списка формальных параметров

Экспорт

Необязательное ключевое слово, которое указывает на то, что данная процедура является доступной из других программных модулей.

- **// Объявления локальных переменных**
- Объявляются локальные переменные, на которые можно ссылаться только в рамках этой процедуры
- **// Операторы**
- Исполняемые операторы процедуры.
- **Возврат**
- Необязательное ключевое слово, которое завершает выполнение процедуры и осуществляет возврат в точку программы, из которой было обращение к процедуре. Использование данного оператора в процедуре необязательно.
- **КонецПроцедуры**
- Обязательное ключевое слово, обозначающее конец исходного текста процедуры, завершение выполнения процедуры. Возврат в точку, из которой было обращение к процедуре.

Пример

```
Перем Глоб;
// Описание процедуры
Процедура мояПроцедура(Пар1, Пар2,
    Пар3) Экспорт
Глоб =  Глоб + Пар1 + Пар2 + Пар3;
Возврат;
КонецПроцедуры

Глоб =123;
МояПроцедура(5, 6, 7); // Вызов процедуры
```

Функция (Function)

Описание:

Функция <имяФункции> (< Список параметров >)

…

КонецФункции

Ключевое слово **Функция** начинает секцию исходного текста функции, выполнение которой можно инициировать из любой точки программного модуля, просто указав ИмяФункции со списком параметров (если параметры не передаются, то круглые скобки, тем не менее, обязательны). Если в модуле приложения или общем программном модуле в теле описания функции использовано ключевое слово **Экспорт**, то это означает, что данная функция является доступной из всех других программных модулей конфигурации.

Выполнение функции заканчивается оператором **Возврат**. Функции отличаются от процедур только тем, что возвращают **ВозвращаемоеЗначение**. Конец программной секции функции определяется по оператору КонецФункции.

ПРИМЕЧАНИЕ

Ключевые слова Функция, КонецФункции являются не операторами, а операторными скобками, поэтому не должны заканчиваться точкой с запятой (это может приводить к ошибкам выполнения модуля).

Функция (Function)

Синтаксис:

Функция <имяФункции>([[Знач] <Парам1>[=<ДефЗнач>], ... ,[знач] <парам N> [=<ДефЗнач>]]) [Экспорт]

// Объявления локальных переменных;

//Операторы;

. . .

Возврат <Возвращаемое значение>;

// операторы;

. . .

КонецФункции

Пример:

```
Перем Глоб;

// Описание функции
Функция МояФункция(Пар1, Пар2, Пар3) Экспорт

Глоб = Глоб + Пар1 + Пар2 + Пар3;
Возврат Глоб;

КонецФункции

Глоб = 123;
Рез = МояФункция(5, 6,  7);  // Вызов функции
```

Выполнить (Execute)

Описание:

Позволяет выполнить фрагмент кода, который передается ему в качестве строкового значения.

Синтаксис:

Выполнить(<Строка>)

Параметры:

<Строка>

Строка, содержащая текст исполняемого кода.

ВНИМАНИЕ!

Не рекомендуется реализовывать с помощью этого метода существенную часть функциональности прикладных решений.

Пример:

```
// Выводит в окно сообщений текущую дату Выполнить ("Сообщить(текущаядата())");
```

ВызватьИсключение (Raise)

Описание:

При использовании данной формы оператора вызывается новое исключение.

Синтаксис:

ВызватьИсключение <Выражение>

Параметры:

<Выражение>

Результат вычисления выражения преобразуется к строке, и данная строка используется в качестве описания исключения.

Пример:

ВызватьИсключение "Документ не может быть проведен";

Попытка (Тгу)

Описание:

Оператор Попытка ***управляет выполнением программы, основываясь на возникающих при выполнении модуля ошибочных (исключительных) ситуациях, и определяет обработку этих ситуаций.***

В качестве ошибочных (исключительных) ситуаций воспринимаются ошибки ***времени выполнения модуля***. Не предусмотрено определяемых пользователем исключений.

Если при выполнении последовательности операторов попытки произошла ошибка времени выполнения, то ***выполнение оператора, вызвавшего ошибку, прерывается и управление передается на первый оператор последовательности операторов исключения.***

Синтаксис:

Попытка
// Операторы попытки
Исключение
// Операторы исключения
[ВызватьИсключение;]
// Операторы исключения
КонецПопытки;

Параметры:

// Операторы попытки

Исполняемый оператор или последовательность таких операторов.

Исключение

Операторы, следующие за ключевым словом Исключение, выполняются, если при выполнении последовательности операторов произошла ошибка времени выполнения.

// Операторы исключения

Исполняемый оператор или последовательность операторов, которые обрабатывают исключительную ситуацию.

ВызватьИсключение

Оператор позволяет вызвать исключение в тех случаях, когда, несмотря на отработку исключительной ситуации, необходимо прервать выполнение модуля с ошибкой времени выполнения. Оператор допустим только внутри операторных скобок ***Исключение - КонецПопытки***.

КонецПопытки

Ключевое слово, которое завершает структуру оператора обработки исключительных ситуаций.

Пример:

```
Процедура СформироватьВExceL()
Попытка
// Пытаемся обратиться к программе MS Excel
Табл = Новый ComObject("Excel.Application");
Исключение
Предупреждение(ОписаниеОшибки());
Возврат;
КонецПопытки;
// Операторы формирования отчета
. . .
КонецПроцедуры
```

ДобавитьОбработчик (AddHandler)

Описание:

Добавляет обработчик события.

При добавлении обработчика события производится проверка соответствия числа параметров события числу параметров метода, назначаемого в качестве обработчика.

Синтаксис:

ДобавитьОбработчик <Событие>, <ОбработчикСобытия>;

Параметры:

<Событие>

Событие, которому добавляется обработчик.

Событие задается в форме **<Выражение>.<ИмяСобытия>,** где:

<Выражение> - произвольное выражение на встроенном языке. Его результатом должен быть объект, к событию которого добавляется обработчик.

<ИмяСобытия> - идентификатор (имя) события.

<ОбработчикСобытия>

Процедура/функция-обработчик события.

ДобавитьОбработчик
Пример

```
Обработка = Обработки.Контрольдокумента. Создать ();
Накладная = Документы.Накладная.Создатьдокумент();
ДобавитьОбработчик Накладная.ПриЗаписи,
Обработка.ПриЗаписидокумента;

msword = Новый COMОбъект(“word.Application");
ДобавитьОбработчик msword.DocumentChange,
Приизмененииидокумента;

Процедура ПриИзмененииидокументаО
Сообщить("Документ изменен");
КонецПроцедуры
```

УдалитьОбработчик (RemoveHandler)

Описание:

Удаляет обработчик события.

При удалении обработчика события производится проверка соответствия числа параметров события числу параметров метода, назначенного в качестве обработчика.

Синтаксис:

УдалитьОбработчик <Событие>, <ОбработчикСобытия>;

УдалитьОбработчик (RemoveHandler)

Параметры:

<Событие>

Событие, обработчик которого удаляется. Событие задается в форме **<Выражение>.<ИмяСобытия>**, где:

<Выражение> - произвольное выражение на встроенном языке. Его результатом должен быть объект, обработчик события которого удаляется.

<ИмяСобытия> - идентификатор (имя) события.

<ОбработчикСобытия>

Процедура/функция-обработчик события.

Пример:

УдалитьОбработчик Накладная.ПриЗалиси,
Обработка.ПриЗаписиДокумента;