# Встроенный язык

```
////////////////////////////////////////////////////////////////////////////////
//// ФУНКЦИИ ВЗАИМОДЕЙСТВИЯ С ТОРГОВЫМ ОБОРУДОВАНИЕМ (СКАНЕР ШТРИХКОДА)

// Функция осуществляет обработку считывания штрихкода номенклатуры //...
Функция СШКНоменклатура(Номенклатура, Характеристика, Серия, Качество,
                         Единица, Количество, СШК) Экспорт

    Результат = Истина;

    СтруктураПоиска = Новый Структура();
    СтруктураПоиска.Вставить("Качество",                 Качество);
    СтруктураПоиска.Вставить("Номенклатура",             Номенклатура);
    СтруктураПоиска.Вставить("СерияНоменклатуры",        Серия);
    СтруктураПоиска.Вставить("ХарактеристикаНоменклатуры", Характеристика);
    СтруктураПоиска.Вставить("ЕдиницаИзмерения",         Единица);

    Владельцы = Товары.НайтиСтроки(СтруктураПоиска);
    Если Владельцы.Количество() = 0 Тогда
        Ответ = Вопрос("Номенклатура """
                       + СокрЛП(Номенклатура)
                       + """ отсутствует в списке проверки.
                         |Добавить позицию в документ?",
                       РежимДиалогаВопрос.ДаНет);
        Если Ответ = КодВозвратаДиалога.Да Тогда
            ДобавитьПозицию(Номенклатура,
                            Единица,
                            Характеристика,
                            Серия,
                            Качество,
```

# Введение

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

# Введение

Встроенный язык представляет собой предметно-ориентированный язык программирования, специально разработанный с учетом возможности его применения не только профессиональными программистами.

В частности, все операторы языка имеют как русское, так и англоязычное написание, которое можно использовать одновременно в одном исходном тексте.

# Введение

Программные модули в конфигурации системы «1С:Предприятие» не являются самостоятельными программами в общепринятом понимании этого слова, поскольку они являются только частью всей конфигурации.

# Введение

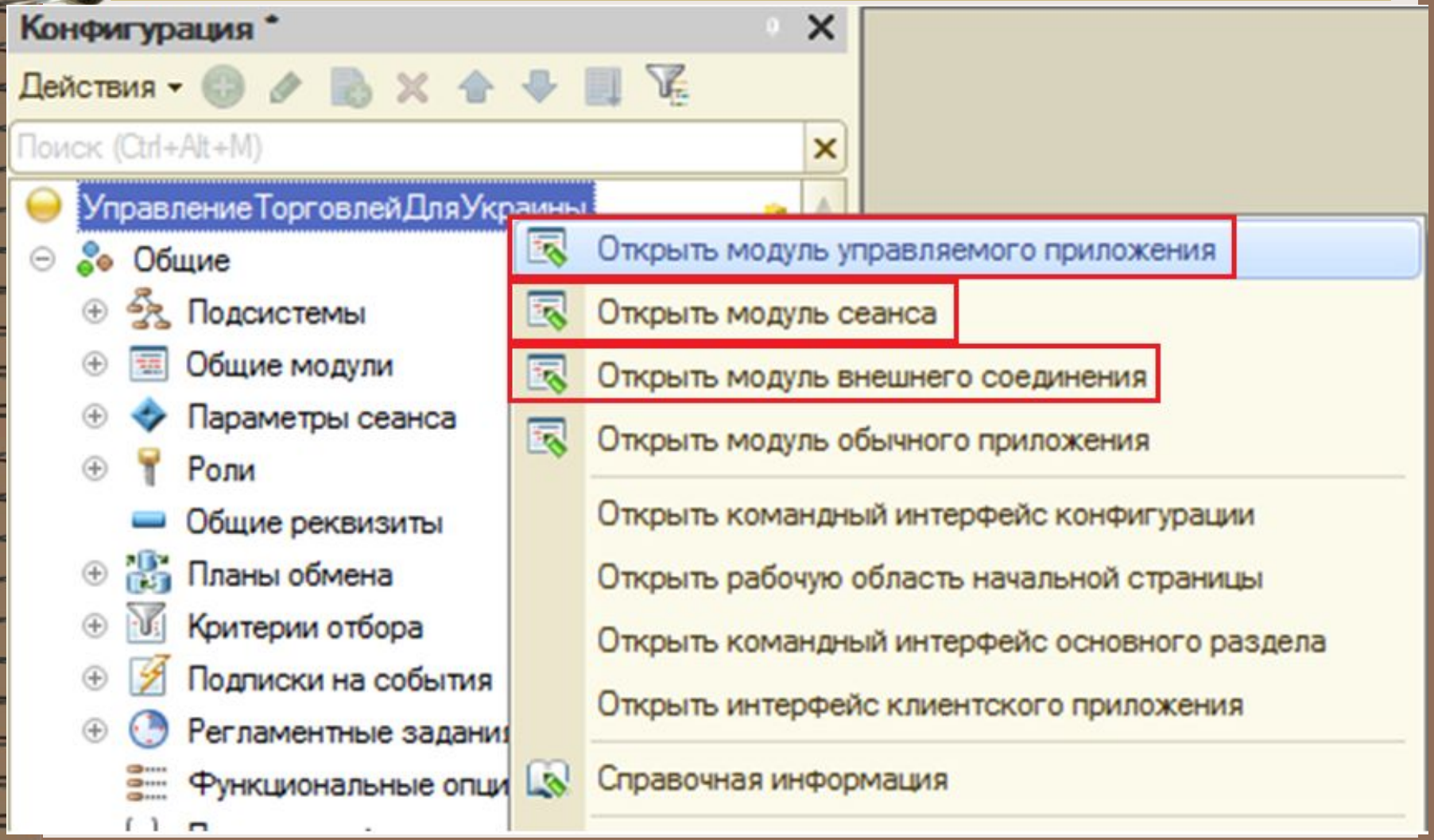
Программный модуль – это текст на встроенном языке, в котором размещены тексты процедур и функций с необходимыми алгоритмами, вызываемые системой в определенные моменты работы.

# Виды программных модулей

В системе «1С:Предприятие» существуют несколько видов программных модулей. Они различаются по месту размещения и доступному контексту.

- Модуль управляемого приложения
- Модуль внешнего соединения
- Модуль сеанса
- Общие модули
- Модули прикладных объектов
- Модули форм

# Виды программных модулей



# Виды программных модулей

Модулем управляемого приложения называется модуль, который автоматически выполняется в момент загрузки конфигурации, при старте системы «1С:Предприятие».

Модуль управляемого приложения предназначен для отработки действий, связанных с сеансом работы конечного пользователя (прежде всего обработки начала и окончания сеанса работы).

# Виды программных модулей

Модуль внешнего соединения расположен, как и модуль приложения, в корневом разделе конфигурации. В нем располагаются процедуры-обработчики событий, которые инициализируются при старте и окончании работы системы в режиме внешнего соединения (COM-соединения)

# Виды программных модулей

Модулем сеанса называется модуль, который автоматически выполняется при старте системы «1С:Предприятие» в момент загрузки конфигурации.

Модуль сеанса предназначен для инициализации параметров сеанса и отработки действий, связанных с сеансом работы.

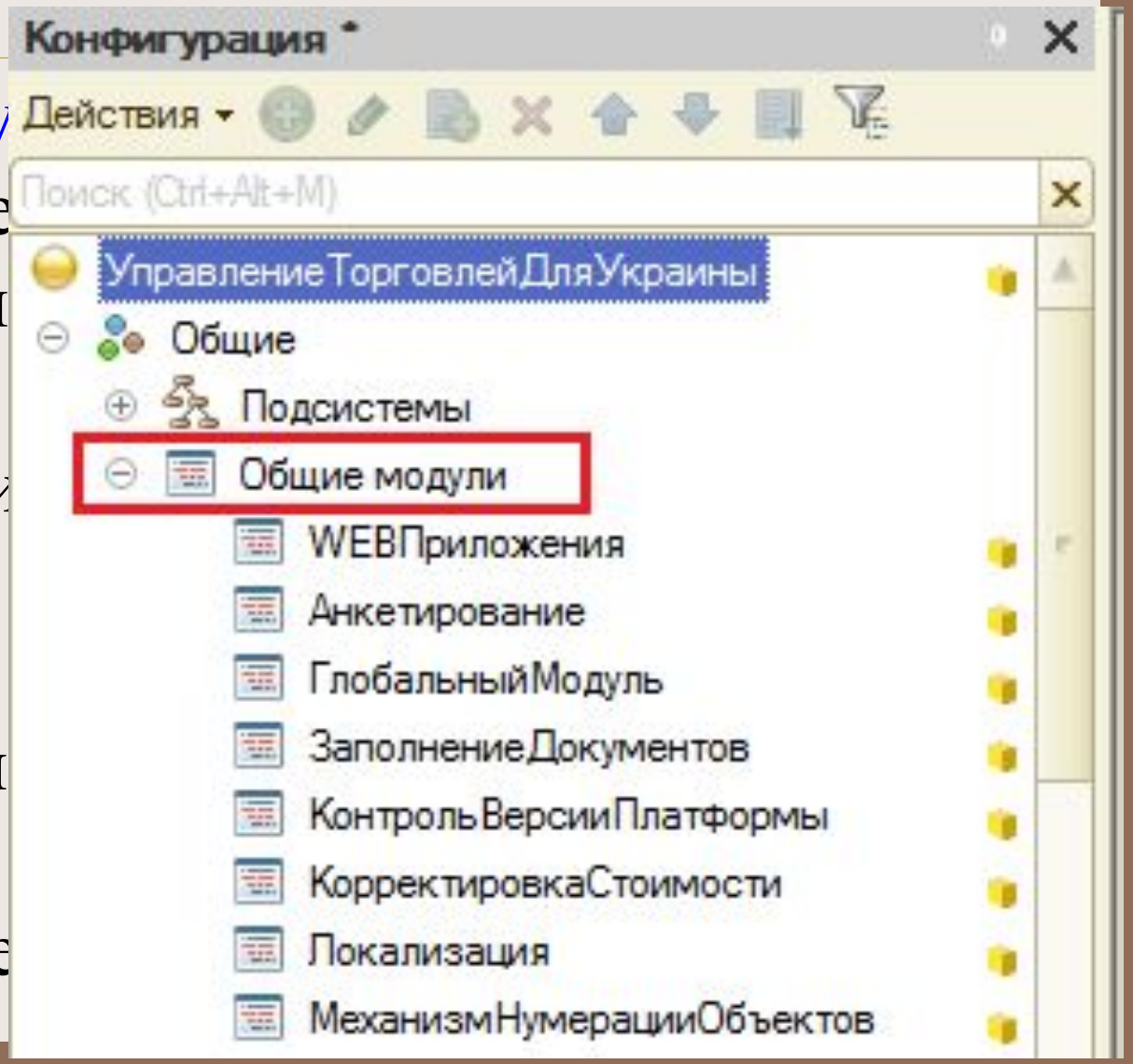
# Виды программных модулей

Установка параметров сеанса выполняется в обработчике события УстановкаПараметровСеанса().

Исполнение модуля сеанса происходит до начала исполнения модуля управляемого приложения (модуля внешнего соединения).

# Виды программных модулей

Общие моду
ветке дере
назначением
содержание
конфигураци
модулей.
В общих
определения
программы,
раздел проце



# Виды программных модулей

Набор прикладных объектов имеет собственные модули. К таким объектам относятся:

- Менеджеры значения константы,
- Справочники,
- Документы,
- Отчеты,
- Обработки,
- Планы видов характеристик,

# Виды программных модулей

- Планы счетов,
- Планы видов расчетов,
- Планы обмена,
- Бизнес-процессы,
- Задачи,
- Регистры.

# Виды программных модулей

Модули располагаются в ветках конфигурации, в которых содержатся сами объекты, и являются свойствами объектов. Каждый объект имеет свой индивидуальный модуль.

В этих модулях возможно объявление переменных, процедур и функций, которые будут доступны при работе с объектом извне во встроенном языке, дополняя контекст объекта.

# Виды программных модулей

Модули форм содержатся в формах конфигурации. Каждая форма имеет свой индивидуальный модуль.

В этих модулях возможно объявление переменных, процедур и функций, которые будут доступны при работе с формой извне во встроенном языке, дополняя контекст формы.

# Формат программного модуля

Структуру программного модуля можно подразделить на следующие разделы:

- раздел определения переменных,
- раздел процедур и функций,
- раздел основной программы.

В конкретном программном модуле любой из разделов может отсутствовать.

# Формат программного модуля

Раздел определения переменных размещается от начала текста модуля до первого оператора Процедура, или оператора Функция, или любого исполняемого оператора.

В этом разделе могут находиться только операторы объявления переменных Перем.

# Формат программного модуля

Раздел основной программы размещается от первого исполняемого оператора вне тела последней процедуры или функции до конца модуля. В этом разделе могут находиться только исполняемые операторы.

Раздел основной программы исполняется в момент инициализации модуля.

# Формат программного модуля

Обычно в разделе основной программы имеет смысл размещать операторы инициализации переменных какими-либо конкретными значениями, которые необходимо провести до первого вызова любой из процедур или функций модуля.

Исходный текст программного модуля может состоять из операторов и комментариев.

# Формат программного модуля

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

В режиме исполнения программы комментарии пропускаются. В тексте

```
//  Общие функции

// Функция возвращает список договоров для контрагента //...
Функция ПолучитьСписокДоговоров(Контрагент) Экспорт ...

// Функция возвращает список валют, доступных для выбора //...
Функция ПолучитьСписокВалют(ДоговорКонтрагента) Экспорт ...
```

# Формат программного модуля

Операторы имеют вид стандартного обращения к процедуре, за исключением оператора присваивания (A = B;) и синтаксических конструкций встроенного языка (например, таких как Для, Пока, Если).

Между собой операторы обязательно следует разделять символом точка с запятой «;».

# Формат программного модуля

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.

# Формат программного модуля

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.

# Формат программного модуля

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

Встроенный язык системы «1С: Предприятие» является двуязычным. Почти все зарезервированные слова, имена типов значений, свойств, методов, событий имеют два имени: русское и английское.

# Формат программного модуля

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

# Формат программного модуля

Встроенный язык системы «1С: Предприятие» является двуязычным. Почти все зарезервированные слова, имена типов значений, свойств, методов, событий имеют два имени: русское и английское.

| Русское имя | Английское имя |
|---|---|
| Если | If |
| Тогда | Then |
| ИначеЕсли | ElsIf |
| Иначе | Else |
| КонецЕсли | EndIf |
| Для | For |
| Каждого | Each |
| Из | In |
| По | To |
| Пока | While |
| Цикл | Do |
| КонецЦикла | EndDo |
| Процедура | Procedure |
| функция | Function |
| КонецПроцедуры | EndProcedure |
| КонецФункции | EndFunction |
| Перем | Var |
| Перейти | Goto |

# Формат программного модуля

| | |
|---|---|
| Возврат | Return |
| Продолжить | Continue |
| Прервать | Break |
| И | And |
| Или | Or |
| Не | Not |
| Попытка | Try |
| Исключение | Except |
| ВызватьИсключение | Raise |
| КонецПопытки | EndTry |
| Новый | New |
| Выполнить | Execute |

# Примитивные типы данных

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

Для большинства примитивных типов данных предусмотрена возможность использования в тексте модуля литералов, то есть указание значения соответствующего типа непосредственно в модуле.

# Примитивные типы данных

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

NULL - значения данного типа используются исключительно для определения отсутствующего значения при работе с базой данных.

# Примитивные типы данных

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

Например:

```
Если МояПеременная = Истина Тогда
КонецЕсли;
```

```
Если МояПеременная Тогда
КонецЕсли;
```

# Примитивные типы данных

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

Обозначается как строка цифр, заключенная в одинарные кавычки вида ‘ГГГГММДДччммсс’, где:

- ГГГГ – четыре цифры года
- ММ – две цифры месяца

# Примитивные типы данных

- ДД – две цифры даты
- чч – две цифры часа (в 24-часовом формате)
- мм – две цифры минут
- сс – две цифры секунд.

# Примитивные типы данных

Допускается при указании литералов типа Дата опускать последние символы (секунды, минуты, часы и т.д.). Это означает, что данные параметры будут равны нулю (для времени).

Например:

```
Дата('2008.03.23 10:45:23')
```

# Примитивные типы данных

Число - числовым типом может быть представлено любое десятичное число. Определены основные арифметические операции над данными числового типа: сложение, вычитание, умножение и деление.

Пример:

```
А = 15;
Б = -968.612;
```

# Примитивные типы данных

Строка - значения данного типа содержат строку произвольной длины в формате Unicode.

Литералы строкового типа представляют собой набор символов, заключенных в кавычки "".
Для задания в строке символа " (кавычка) необходимо записать две кавычки подряд ("").

# Примитивные типы данных

Примеры:

```
// Пример строки
МояСтрока = "Это правильная строка";
// Пример 1 многострочной строки
МояМногострочнаяСтрока = "Это
      |правильная
      |многострочная
      |строка";
// Пример 2 многострочной строки
МояМногострочнаяСтрока = "Это тоже" //Это комментарий
      "правильная"
      "многострочная"
      "строка";
// Пример 3 строки с кавычками
НазваниеФирмы = "ООО ""Василек""";
```

# Примитивные типы данных

Неопределено - значение данного типа применяется, когда необходимо использовать пустое значение, не принадлежащее ни к одному другому типу.

# Оператор присваивания

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

<Назначение> = <Источник>;

# Операции

Арифметические операции имеют один или два операнда, в зависимости от типа которых операция имеет ту или иную семантику.

Сложение:

- Число + Число
- Дата + Число (к дате прибавляется число секунд)

# Операции

Вычитание:

- Число – Число
- Дата – Число (от даты отнимается число секунд)
- Дата – Дата (результатом является разница между двумя датами, измеренная в секундах)

# Операции

Умножение:

- Число * Число

Деление:

- Число / Число

Остаток от деления:

- Число % Число

# Операции

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;
```

# Операции

В языке определены следующие виды операций сравнения:

> >= = <> < <=

Операции сравнения определены для следующих типов: Число, Строка, Дата.

# Операторы

Вычислить выражение по условию:
?(<Логическое выражение>, <Выражение 1>, <Выражение 2>)

Если результат вычисления <Логического выражения> Истина, то будет вычисляться <Выражение 1>. Если результат Ложь, то <Выражение 2>.

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

# Операторы

Если (If) - оператор управляет выполнением программы, основываясь на результаты одного или более логических выражений.

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

# Операторы

Для (For) - оператор цикла предназначен для циклического повторения операторов, находящихся внутри конструкции Цикл – КонецЦикла.

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

# Операторы

Для каждого (For each) - оператор цикла предназначен для циклического обхода коллекций значений. При каждой итерации цикла возвращается новый элемент коллекции.

```
Для каждого <Имя переменной 1> Из <Имя переменной 2> Цикл

// Операторы

[Прервать;]

// Операторы

[Продолжить;]

// Операторы

КонецЦикла
```

# Операторы

Пока (While) - оператор цикла предназначен для циклического повторения операторов, находящихся внутри конструкции Цикл – КонецЦикла

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

# Операторы

Новый (New) - оператор позволяет создать значение указанного типа. Допустим только для тех типов, для которых разрешено создание новых значений.

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

Например:

```
// Пример создания массива из трех элементов.
Массив = Новый Массив(3);
```

# Операторы

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

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

# Операторы

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. После выполнения последовательности операторов исключения управление передается на следующий за ключевым словом КонецПопытки оператор.

# Операторы

Если же последовательность операторов попытки выполнилась без ошибок, то последовательность операторов исключения будет пропущена и управление также будет продолжено с оператора, следующего за ключевым словом КонецПопытки.

# Операторы

Например:

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

# Операторы

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

# Операторы

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

# Операторы

## Синтаксис:

```
Процедура <ИмяПроцедуры>([[Знач] <Парам 1> [=<ДефЗнач>],...,[Знач] <Парам N> [=<ДефЗнач>]])[Экспорт]
// Объявления локальных переменных;
// Операторы;
...
[Возврат;]
// Операторы;
...
КонецПроцедуры
```

# Операторы

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.

# Операторы

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.

# Операторы

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.

# Операторы

Например:

```
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
```

# Операторы

Например:

```
Процедура МояПроцедура(Параметр1, Параметр2 = "по умолчанию")...
КонецПроцедуры
// При выполнении следующего вызова в процедуре "МояПроцедура",
// значение параметра Параметр1 будет равно 1,
// а значение параметра Параметр2 - "по умолчанию"
МояПроцедура(1);
```

# Операторы

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).

# Операторы

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.

# Операторы

Выполнение функции заканчивается оператором Возврат. Функции отличаются от процедур только тем, что возвращают ВозвращаемоеЗначение.

Конец программной секции функции определяется по оператору КонецФункции.

# Операторы

## Синтаксис:

```
Функция <Имяфункции>([[Знач] <Парам 1>[=<ДефЗнач>],...,[Знач] <Парам N>[=<ДефЗнач>]])[Экспорт]

// Объявления локальных переменных;

// Операторы;

...

Возврат <Возвращаемое значение>;

// Операторы;

...

КонецФункции
```

# Операторы

Например:

```
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
```

# Работа со строками

Строка — примитивный тип данных в языке программирования 1С. Помимо 1С, этот тип данных используется во всех известных языках программирования, обычно он называется «string».

Рассмотрим основные функции работы со строками встроенного языка программирования 1С.

# Работа со строками

Функция Строка() позволяет получить текстовое представление переменных других типов.

```
Строка(ТекущаяДата()) //"23.02.2015 21:31:24"

Строка(Истина)   // Да

Строка(1058)   // "1 058"
```

# Работа со строками

СтрДлина() - функция позволяет получить количество символов в строке 1С, включая пробелы и незначащие символы.

```
СтрДлина("Тестовая строка") // 15
```

# Работа со строками

СокрЛП(), СокрЛ(), СокрП() - функции для удаления незначащих символов в строковом значении.

СокрЛ обрабатывает символы, стоящие слева от последнего значащего символа.

СокрП — справа.

СокрЛП — справа и слева

```
СокрЛП(" Слева и справа есть незначащие символы - пробелы ") // получим строку без пробелов слева и справа
```

# Работа со строками

Помимо пробелов, функция 1С удаляет такие символы, как перенос строки, неразрывный пробел и т.д

# Работа со строками

Лев(), Прав(), Сред() - с помощью данных функций очень просто получить первые (Лев) или последние (Прав) символы в строке, а также произвольное количество символов (Сред).

```
Лев("Тестовая строка", 4) // "Тест"

Сред("Тестовая строка", 5, 6) // "овая с"
```

# Работа со строками

Найти - функция для поиска подстроки внутри другой подстроки. Синтаксис Найти(<Строка для поиска>, <Подстрока поиска>). Функция возвращает число — номер позиции символа (или символов) в исходной подстроке. Если подстроки не найдено, вернётся число 0. Если в искомой

```
Найти("Тестовая строка", "строка") // 10

Найти("Тестовая строка", "123") // вернется 0

Найти("Тестовая строка", "я") // 8
```

# Работа со строками

ПустаяСтрока() - функция, позволяющая сравнить исходную строку с пустой строкой. Проверяется наличие незначащих символов.

```
ПустаяСтрока("Тест") //ложь

ПустаяСтрока(" ") // истина
```

# Работа со строками

Врег(), Нрег(), Трег() - функции для управления регистром строки.

ВРег — преобразует строку в верхний регистр.

НРег — в нижний.

ТРег — у каждого нового слова первая буква преобразуется в верхний регистр, остал

```
ВРег("Тестовая строка")  // "ТЕСТОВАЯ СТРОКА"
НРег("Тестовая строка")  // "тестовая строка"
ТРег("Тестовая строка") // "Тестовая Строка"
```

# Работа со строками

СтрЗаменить() - функция позволяет массово заменить искомое значение на нужное нам.

```
СтрЗаменить("Произвольная Строка", " " , "") //"ПроизвольнаяСтрока"
```

# Работа со строками

СтрЧислоСтрок() - позволяет получить количество строк в многострочном тексте. Функция считает количество переносов строк (Символы.ПС).

```
Для Н=1 по СтрЧислоСтрок(ТекстФайла) Цикл

.....

КонецЕсли;
```

# Работа со строками

СтрПолучитьСтроку() - с помощью данной функции можно получить произвольную строку из многострочного текста.

Синтаксис:

СтрПолучитьСтроку(<Строка>, <Номер строки>).

```
СтрокаНомер5 =  СтрПолучитьСтроку(ИсходнаяСтрока, 5)
```

# Работа со строками

СтрЧислоВхождений() - функция для подсчета числа вхождений произвольного фрагмента в строку.

```
СтрЧислоВхождений ("Тестовая строка", "т")  // 2
```

# Работа со строками

Символ() и КодСимвола() - две обратные функции. С помощью КодСимвола можно получить код символа в кодировке Unicode. Функцией Символ можно получить символ, зная Unicode кодификатор.

```
КодСимвола("Т")   // 1 058

Символ(1058) // "Т"
```

# Работа со строками

Конкатенация, или объединение строк в 1С делается оператором «+».

```
ИсходнаяСтрока = "Строка №1" + Символы.ПС + "Строка №2";
СтрокаНомер2 =  СтрПолучитьСтроку(ИсходнаяСтрока, 2);
Сообщить(СтрокаНомер2);
```

# Работа со строками

Набор специальных символов:

- Символы.ВК - возврат каретки;
- Символы.Втаб - символ табуляции (вертикальной).
- Символы.НПП - неразрывный пробел.
- Символы.ПС - перевод строки.
- Символы.ПФ - перевод формы (страницы).
- Символы.Таб - символ табуляции (горизонтальной).

# Работа со строками

СтрСравнить() - выполняет сравнение строк без учета регистра.

Возвращает:

- -1 - первая строка меньше второй.
- 1 - первая строка больше второй.
- 0 - первая строка равна второй.

# Работа со строками

СтрНачинаетсяС(Строка, СтрокаПоиска) - определяет, что строка начинается с указанной подстроки. Определение выполняется с учетом регистра.

СтрЗаканчиваетсяНа(Строка, Подстрока) - определяет, заканчивается ли строка указанной подстрокой.

# Функции работы со значениями типа Число

Цел(<Число>) - вычисляет целую часть переданного числа, полностью отсекая дробную часть.

```
МожноКупить = Цел(Наличность/Цена);
```

# Функции работы со значениями типа Число

Окр(<Число>, <Разрядность>, <РежимОкругления>) - округляет исходное число до нужной разрядности в соответствии с заданным режимом округления.

Режим округления:

- 0 - если при округлении 1.5 = 1;
- 1 - если при округлении 1.5 = 2.

# Функции работы со значениями типа Число

<Разрядность> - определяет число знаков дробной части, до которых производится округление. Если параметр отрицательный, то число округляется до соответствующего разряда в целой части, начиная с младших разрядов

# Функции работы со значениями типа Число

Например:

```
Цена = 123.98765;

// Округлим цену до сотен гривен
ОкругленнаяЦена1 = Окр(Цена, -2);
Сообщить(ОкругленнаяЦена1);

// Округлим цену до копеек
ОкругленнаяЦена2 = Окр(Цена, 2);
Сообщить(ОкругленнаяЦена2);
```

Служебные сообщения

```
100
123,99
```

# Функции работы со значениями типа Число

Log(<Число>) - вычисляет натуральный логарифм параметра <Число>.

Log10(<Число>) - вычисляет десятичный логарифм параметра <Число>.

Sin(<Угол>) - вычисляет синус от аргумента <Угол>, заданного в радианах.

Cos(<Угол>) - вычисляет косинус от аргумента <Угол>, заданного в радианах.

Tan(<Угол>) - вычисляет тангенс от аргумента <Угол>, заданного в радианах

# Функции работы со значениями типа Число

ASin(<Число>) - вычисляет арксинус от аргумента <Число>.

ACos(<Число>) - вычисляет арккосинус от аргумента <Число>.

ATan(<Число>) - вычисляет арктангенс от аргумента <Число>.

Exp(<Число>) - вычисляет результат возведения основания натурального логарифма (числа e) в степень <Число>.

# Функции работы со значениями типа Число

Pow (<X>, <Y>) - возводит число <X> в степень <Y>.

Sqrt(<Число>) - вычисляет квадратный корень параметра <Число>.

# Функции работы со значениями типа Дата

Во встроенном языке в литерале типа Дата обязательно должно задаваться значение года, месяца и дня.

Инициализация датой:

```
Дата = '20130724'; // 24.07.2013
Дата = Дата("20130724"); // 24.07.2013
Дата = Дата(2013, 07, 24); // 24.07.2013
Дата = '20130724132506'; // 24 июля 2013 г. 13 ч. 25 мин. 6 сек.
```

# Функции работы со значениями типа Дата

Год(<Дата>) - определяет год в указанной дате.

Месяц(<Дата>) - определяет месяц в указанной дате.

День(<Дата>) - определяет день в указанной дате.

Час(<Дата>), Минута(<Дата>) и Секунда (<Дата>) - определяет час, минуты и секунды в указанной дате.

# Функции работы со значениями типа Дата

Например:

```
Дата = '20130724132506';

Г =  Год(Дата); // 2013
М =  Месяц(Дата); // 7
Д =  День(Дата); // 24
Ч =  Час(Дата); // 13
Ми = Минута(Дата); // 25
С =  Секунда(Дата); // 6
```

# Функции работы со значениями типа Дата

ТекущаяДата() - определяет текущую (системную) дату на компьютере.

ДобавитьМесяц(<Дата>, <ЧислоМесяцев>) - добавляет (или вычитает) к указанной дате заданное число месяцев. Если <ЧислоМесяцев> принимает отрицательное значение, то число месяцев вычитается.

# Функции работы со значениями типа Дата

Например:

```
Сообщить(ТекущаяДата());
Сообщить(ТекущаяДата() + 1); // прибавили секунду

Сообщить(ДобавитьМесяц(ТекущаяДата(), 1)); // прибавили месяц
Сообщить(ДобавитьМесяц(ТекущаяДата(), -1)); // отняли месяц
```

```
Дата = '20130110125905'; // 10 января 2013 года 12:59:05

Сообщить(ДеньГода(Дата)); // 10

Сообщить(ДеньНедели(Дата)); // 4 т.е. четверг (нумерация с понедельника)

Сообщить(НеделяГода(Дата)); // 2

Сообщить(НачалоГода(Дата)); // 01.01.2013 0:00:00
Сообщить(КонецГода(Дата)); // 31.12.2013 23:59:59

Сообщить(НачалоКвартала(Дата)); // 01.01.2013 0:00:00
Сообщить(КонецКвартала(Дата)); // 31.03.2013 23:59:59

Сообщить(НачалоМесяца(Дата)); // 01.01.2013 0:00:00
Сообщить(КонецМесяца(Дата)); // 31.01.2013 23:59:59

Сообщить(НачалоНедели(Дата)); // 07.01.2013 0:00:00
Сообщить(КонецНедели(Дата)); // 13.01.2013 23:59:59

Сообщить(НачалоДня(Дата)); // 10.01.2013 0:00:00
Сообщить(КонецДня(Дата)); // 10.01.2013 23:59:59

Сообщить(НачалоЧаса(Дата)); // 10.01.2013 12:00:00
Сообщить(КонецЧаса(Дата)); // 10.01.2013 12:59:59

Сообщить(НачалоМинуты(Дата)); // 10.01.2013 12:59:00
Сообщить(КонецМинуты(Дата)); // 10.01.2013 12:59:59
```

# Диалоги

Например:

```
Сообщить("Какой вы молодец!");
```

```
Предупреждение(
    "Будьте осторожны.", // предупреждение
    0, // (необ.) таймаут в секундах
    "Это предупреждение." // (необ.) заголовок
);
```

```
Если
    ВвестиДату(Дата, "Введите дату рождения", ЧастиДаты.Дата) = Истина
Тогда
    Сообщить("Вы родились " + Дата);
КонецЕсли;
```

# Диалоги

Например:

```
Если ВвестиЗначение(Ч, "Введите значение", "Число") Тогда
    Сообщить(Ч);
КонецЕсли;
```

```
Если ВвестиСтроку(
    Стр,
    "Введите строку",
    0, // (необ.) длина
    Истина // (необ.) многострочность
)
Тогда
    Сообщить(Стр);
КонецЕсли;
```

# Диалоги

Например:

```
Если ВвестиЧисло(
     Ч,
     "Введите число",
     3, // длина числа, включая дробную часть без разделителя
     1 // длина дробной части
)
Тогда
     Сообщить(Ч);
КонецЕсли;
```