Основы встроенного языка

ПРАВИЛА НАПИСАНИЯ КОДА

ФОРМАТ ОПЕРАТОРОВ

Операторы разделяются «;»

```
ТекущаяТема = "Основы встроенного языка";
Сообщить("Сейчас изучаем: " + ТекущаяТема);
```

В одной строке может быть несколько операторов

```
А=1; Б=2; В=3; Сообщить("Сумма: " + (А+Б+В));
```

Двуязычное написание

```
Если Дата1 > Дата2 Then
      // ...
EndIf
```

КОММЕНТАРИИ

Признак начала комментария

```
Продолжительность = 86400; // Количество секунд в сутках
```

Многострочные комментарии

```
////////////////////////////////////////////////////////////
// Чем больше комментариев,
// тем понятнее!
// …
```

Специального синтаксиса для комментирования больших фрагментов кода нет. «Блочные» комментарии оформляются так же, как и «однострочные».

ИМЕНА ПЕРЕМЕННЫХ, ПРОЦЕДУР И ФУНКЦИЙ

Могут содержать буквы, цифры и знак подчеркивания

```
НормальнаяПеременная_1    = 0;
Совсем плохая^переменная = 0;
```

Не могут начинаться с цифры

```
1ЕщеОднаПлохаяПеременная = 0;
```

Нечувствительны к регистру символов

```
НазваниеОрганизации = "ООО 'Торговый дом'";
СООБЩИТЬ(нАзВаНиЕоРгАнИзАцИи);
```

При выборе имен переменных, процедур и функций следует стремиться не к краткости, а к понятности.

ТИПЫ ДАННЫХ

ТИП «БУЛЕВО»

Истина и Ложь

```
ЗанятиеНачалосьНедавно = Истина;
СкороПерерыв           = Ложь;
```

Использование в логических выражениях

```
Если ЗанятиеНачалосьНедавно = Истина Тогда … // Можно и так…
Если ЗанятиеНачалосьНедавно Тогда …          // Но так короче
```

Булевы операции

```
СкороПерерыв = НЕ ЗанятиеНачалосьНедавно;
Если КолТоваров > 0 И (МинимальнаяЦена > 100 ИЛИ НЕ ПроверятьЦены) Тогда …
```

Приоритет булевых операций: НЕ → И → ИЛИ. Операции, заключенные в круглые скобки, имеют наивысший приоритет.

ТИП «ЧИСЛО»

В 1С:Предприятии есть только один тип данных для работы с числами – «Число». Разновидности (int, float, double, …) отсутствуют.

Основные операции

```
Результат1 = (120-20) / 10 + 2*5;
```

Остаток от деления

```
Результат2 = Результат1 % 8; // Чему равен Результат2?
```

Встроенные функции для работы с числами

```
Результат = 10 / 3;

Примерно   = Окр(Результат, 2);
ЦелаяЧасть = Цел(Результат);

// Подробное описание всех функции см. здесь:
//  Конфигуратор → Справка → Синтакс-помощник → Общее описание встроенного языка → Встроенные функции → …
```

ТИП «СТРОКА»

Определение строки

```
НазваниеОрганизации = "ООО Торговый дом ";

// Если в строку входят кавычки, то так:
НазваниеОрганизации = "ООО ""Торговый дом""; // ООО "Торговый дом"
```

Значение типа «Строка» может содержать любые символы Unicode (£, Ω, ⅞, ≈, ☼, …).

Сложение (конкатенация) строк

```
ПолноеНазвание = НазваниеОрганизации + " (г. Санкт-Петербург)";
```

«Многострочные» строки

```
// Вариант 1
Предупреждение = "Документ будет удален." + Символы.ПС + "Продолжить?";
Сообщить(Предупреждение);

// Вариант 2
Предупреждение = "Документ будет удален.
                 |Продолжить?";
Сообщить(Предупреждение);
```

Встроенные функции для работы со строками

```
ФИО = "Иванов Иван Иванович";

Фамилия  = Лев (ФИО, 6);
Имя      = Сред(ФИО, 8, 4);
Отчество = Прав(ФИО, 8);

// Подробное описание всех функции см. здесь:
// Конфигуратор → Справка → Синтакс-помощник → Общее описание встроенного языка → Встроенные функции → …
```

ТИП «ДАТА»

Определение даты

```
// Вариант 1
НачалоКурса = '20121017183000'; // 17 октября 2012 года, 18 ч. 30 м. 0 сек.
НачалоДня   = '20121017';       // 17 октября 2012 года, 0 ч. 0 м. 0 сек.

// Вариант 2
НачалоКурса = Дата(2012, 10, 17, 18, 30, 0);
НачалоДня   = Дата(2012, 10, 17);
```

Значение типа «Дата» всегда содержит в себе не только дату, но и время с точностью до секунды.

Операции с датами

```
КонецЗанятия     = НачалоКурса + 10800;       // Количество секунд
СледующееЗанятие = НачалоКурса + 2*24*60*60; // Послезавтра

ДлительностьЗанятия = КонецЗанятия – НачалоКурса; // = 10800 секунд
```

Значение по умолчанию

```
ПустаяДата = '00010101'; // То же самое, что ноль для типа "Число"

// Предположим, что в переменную "ВыбраннаяДата" попала дата,
// введенная пользователем на форме. Проверяем эту дату:
Если ВыбраннаяДата = '00010101' Тогда … // Значит дата не введена
```

Встроенные функции

```
Сейчас = ТекущаяДата();
Отбой  = КонецДня(Сейчас); // Время 23:59:59

Если ДеньНедели(Сейчас) = 5 Тогда … // Пятница! :)

// Подробное описание всех функции см. здесь:
// Конфигуратор → Справка → Синтакс-помощник → Общее описание встроенного языка → Встроенные функции → …
```

ОСОБЕННОСТИ РАБОТЫ С ТИПАМИ ДАННЫХ

«Мягкая» типизация

```
// Тип переменной может изменяться:
МояПеременная = 777;
МояПеременная = "А теперь в переменной строка";
МояПеременная = ТекущаяДата();
```

Тип переменной определяется типом того значения, которое она в данный момент содержит. В процессе выполнения программы тип переменной может изменяться.

Преобразование типов

```
// Вариант 1: "НЕЯВНОЕ" преобразование (не всегда возможно)
МояПеременная = "АБВ" + 777;       // "АБВ777"
МояПеременная = 777 + "АБВ";       // Ошибка!
МояПеременная = "" + 777 + "АБВ"; // "777АБВ"

// Вариант 2: "ЯВНОЕ" преобразование (надежнее)
МояПеременная = Строка(777) + "АБВ";   // "777АБВ"
МояПеременная = Число("23,5000") + 10; // 33.5

// Подробное описание всех функции преобразования см. здесь:
// Конфигуратор → Справка → Синтакс-помощник → Общее описание встроенного языка → Встроенные функции → …
```

Проверка типа значения

```
// Значение какого типа содержится сейчас в переменной "Нечто"?

Если ТипЗнч(Нечто) = Тип("Число") Тогда
      // …

ИначеЕсли ТипЗнч(Нечто) = Тип("Строка") Тогда
      // …

ИначеЕсли ТипЗнч(Нечто) = Тип("Дата") Тогда
      // …
```

ОСНОВНЫЕ ОПЕРАТОРЫ

ПЕРЕМЕННЫЕ

Явное объявление

```
Перем МаксимальныйКредит;

Перем глТекущийПользователь Экспорт; // Область видимости увеличена
```

Неявное объявление

```
ЕдиницаПоУмолчанию = "шт";

НовыйГод = КонецГода(ТекущаяДата());
```

Локальные переменные (внутри процедуры/функции) обычно объявляют неявно. Переменные с более широкой областью видимости (например, глобальные) можно объявить только явно.

ПРОЦЕДУРЫ И ФУНКЦИИ

Функция: объявление и вызов

```
Функция ПолучитьСумму(Первое, Второе)

      Сумма = Первое + Второе;

      Возврат Сумма; // Функция должна возвращать результат своей работы

КонецФункции

// Примеры вызова функции:
Результат1 = ПолучитьСумму(10, 5);

Результат2 = ПолучитьСумму(36, 6);
```

Процедура: объявление и вызов

```
Процедура Разделить(Первое, Второе)

      Если Второе = 0 Тогда

            Сообщить("Деление на ноль запрещено!");

            Возврат; // Процедура не может ничего возвращать!
                     // "Возврат" просто прерывает выполнение процедуры.
      КонецЕсли;

      РезультатДеления = Первое / Второе;
      Сообщить("Результат деления чисел: " + РезультатДеления);

КонецПроцедуры

// Примеры вызова процедуры:
Разделить(35, 0);
Разделить(10, 5);
```

Передача параметров «по ссылке» и «по значению»

```
Процедура Сложить(Первое, Знач Второе)

      Первое = Первое + 10; // Параметр "Первое" передан "ПО ССЫЛКЕ"!
      Второе = Второе + 10; // Параметр "Второе" передан "ПО ЗНАЧЕНИЮ"!

КонецПроцедуры

А=40; Б=40;
Сложить(А, Б);
// Что содержится в переменных "А" и "Б" после вызова процедуры?
```

По умолчанию все параметры процедур и функций передаются «по ссылке», а не «по значению»!

Параметр со значением по умолчанию

```
Процедура Сложить(Первое, Второе=7)

      Первое = Первое + Второе;

КонецПроцедуры

А=40; Б=40;
Сложить(А);     // Чему равно "А"?
Сложить(А, Б); // А теперь?
```

ВЕТВЛЕНИЕ

Оператор «Если»

```
Цена = ПолучитьАктуальнуюЦену("Яблоки");

Если Цена < 100 Тогда
      Скидка = 10;

ИначеЕсли Цена >= 100 И Цена < 200 Тогда
      Скидка = 15;

Иначе
      Скидка = Цена * 0.2;

КонецЕсли;
```

Оператор «?»

```
КатегорияТовара = ?(Цена > 10000, "Дорогой", "Дешевый");

Скидка = ?(Цена < 100, 10, ?(Цена < 200, 15, Цена*0.2));
```

В 1С:Предприятии отсутствует оператор многозначного выбора (switch – case).

ЦИКЛЫ

Цикл со счетчиком

```
Для Счетчик = 1 По 100 Цикл

      Если Счетчик < 20 Тогда
            Продолжить; // На следующий виток
      ИначеЕсли Счетчик = 50 Тогда
            Прервать;   // Досрочное прерывание цикла
      КонецЕсли;

      Сообщить("Виток №" + Счетчик); // Сколько раз сработает?

КонецЦикла;
```

В операторе «Для … Цикл» величина приращения счетчика всегда равна +1. Переопределить «шаг цикла» невозможно!

Цикл с предусловием

```
ВспомДата = НачалоМесяца(ТекущаяДата());

Пока ВспомДата <= ТекущаяДата() Цикл

      Сообщить(ВспомДата); // Сколько раз сработает?

      ВспомДата = ВспомДата + 1;

КонецЦикла;
```

Цикл для обхода коллекций значений

```
Для каждого ТекущийЭлемент Из МассивДанных Цикл

      Сообщить(ТекущийЭлемент); // Сколько раз сработает?

КонецЦикла;
```

Цикл «Для каждого» завершает свою работу автоматически – после обработки последнего элемента коллекции.

ЭЛЕМЕНТЫ ООП

РАБОТА С ПРОГРАММНЫМИ ОБЪЕКТАМИ (СИНТАКСИС)

Создание нового объекта

```
Генератор = Новый ГенераторСлучайныхЧисел(); // Есть и другие
Письмо    = Новый ТекстовыйДокумент();       // способы создания
Почтальон = Новый ИнтернетПочта();           // программных объектов

// См. список некоторых полезных объектов:
// Конфигуратор → Справка → Синтакс-помощник → Общие объекты → …
```

В 1С:Предприятии отсутствует возможность программного описания собственных классов объектов. Разработчик может создавать экземпляры объектов только тех классов, которые уже встроены в систему.

Свойства объекта

```
ПолезныйОбъект = Новый ИзвлечениеТекста();

// Изменяем значение свойства "ИмяФайла":
ПолезныйОбъект.ИмяФайла = "C:\Договор.PDF";

// Узнаём значение свойства "ИмяФайла":
Сообщить("Текущий файл: " + ПолезныйОбъект.ИмяФайла);
```

Методы объекта

```
// Вызываем метод "ПолучитьТекст" (без параметров):
СтрокаСТекстомДоговора = ПолезныйОбъект.ПолучитьТекст();

// Вызываем метод "Записать" (с параметром):
ПолезныйОбъект.Записать("C:\Текст договора.TXT");
```

УНИВЕРСАЛЬНЫЕ КОЛЛЕКЦИИ ЗНАЧЕНИЙ

ОБЩИЕ СВЕДЕНИЯ ОБ УНИВЕРСАЛЬНЫХ КОЛЛЕКЦИЯХ

Виды универсальных коллекций

Чаще всего на практике применяют следующие виды коллекций: «Массив», «Структура» и «ТаблицаЗначений». Но есть и другие. Подробное описание всех видов коллекций и всех их возможностей см. здесь: *Конфигуратор → Справка → Синтакс-помощник → Универсальные коллекции значений → …*

Зачем нужны коллекции?

Универсальные коллекции предназначены для хранения и обработки временных наборов данных в течение сеанса работы пользователя. Значения хранятся в оперативной памяти (как обычные переменные) и не сохраняются в базе данных.

Чем коллекции отличаются от «обычных» переменных?

В обычной переменной может содержаться только одно значение (одно число, одна строка, одна ссылка на элемент справочника или т.п.).

Универсальные коллекции значений позволяют более эффективно работать с информацией. Например, в переменной типа «Массив» можно сохранить сразу несколько значений, причем разного типа. А в переменной типа «ТаблицаЗначений» – целую таблицу данных, с произвольным количеством строк и колонок.

МАССИВ

Что такое «массив»?

Массив – это набор значений, хранящихся в оперативной памяти. Упрощенно массив можно воспринимать как простую таблицу, в которой каждому элементу массива соответствует одна строка, например:

Индекс элемента	Значение элемента
0	«Иванов И.И.»
1	«Петров П.П.»
2	«Сидоров С.С.»

Массив с тремя элементами

Индексы элементов массива

Каждому элементу массива назначен свой «индекс» (целое положительное число, по сути «внутренний номер» элемента). Индексация начинается с нуля, т.е. первый элемент любого массива всегда имеет индекс 0.

Индексы позволяют обращаться к отдельным элементам массива для чтения/изменения значений.

Создание массива

```
// Изначально пустой массив (потом можно будет добавить сколько угодно элементов):
ВспомогательныйМассив = Новый Массив;

// Массив из 7 элементов (потом можно будет добавить сколько угодно элементов):
ДниНедели = Новый Массив(7);
```

Заполнение массива

```
// В одном массиве можно хранить значения самых разных типов:
ВспомогательныйМассив.Добавить("Самый первый элемент");
ВспомогательныйМассив.Добавить(22222);
ВспомогательныйМассив.Добавить(Истина);
ВспомогательныйМассив.Добавить(ТекущаяДата());
```

Доступ к элементам массива по индексам

```
// Чтение:
ПервыйЭлемент = ВспомогательныйМассив[0];
ВторойЭлемент = ВспомогательныйМассив[1];

// Изменение:
ДниНедели[0] = "Понедельник";
ДниНедели[1] = "Вторник";
…
ДниНедели[6] = "Воскресенье";
```

Обход всех элементов массива

```
Для каждого ЭлементМассива Из ДниНедели Цикл

	// Чтение:
	Сообщить(ЭлементМассива);

	// Изменение:
	ЭлементМассива = ВРег(ЭлементМассива); // "Понедельник" => "ПОНЕДЕЛЬНИК"

КонецЦикла;
```

Прочие возможности массивов

Синтакс-помощник → Универсальные коллекции значений → Массив → Методы → …

СТРУКТУРА

Что такое «структура»?

Структура – это набор именованных значений, хранящихся в оперативной памяти. В отличие от массива, к элементам структуры обращаются не по индексам, а по строковым ключам.
Упрощенно структуру можно воспринимать как простую таблицу, в которой каждому элементу структуры соответствует одна строка, например:

Ключ элемента	Значение элемента
ДатаОтчета	'07.05.2013 16:57:00'
ВыводитьНомераСтраниц	Истина
ЗаголовокОтчета	«Анализ продаж»
КоличествоКопий	2

Структура с четыремя элементами

«Ключ» элемента – всегда строка. Причем, без пробелов, без специальных символов и т.п. (для ключей действуют те же самые правила, что и для имен переменных).

«Значение» элемента может быть произвольного типа.

Создание и заполнение структуры

```
ПараметрыОтчета = Новый Структура;

ПараметрыОтчета.Вставить("ДатаОтчета",             ТекущаяДата());
ПараметрыОтчета.Вставить("ВыводитьНомераСтраниц", Истина);
ПараметрыОтчета.Вставить("ЗаголовокОтчета",        "Анализ продаж");
ПараметрыОтчета.Вставить("КоличествоКопий",        2);
```

Доступ к значениям структуры по ключам

```
// Чтение:
Заголовок      = ПараметрыОтчета.ЗаголовокОтчета;
КолЭкземпляров = ПараметрыОтчета.КоличествоКопий;

// Изменение:
ПараметрыОтчета.ДатаОтчета             = НачалоГода(ТекущаяДата());
ПараметрыОтчета.ВыводитьНомераСтраниц = Ложь;
```

Обход всех элементов структуры

```
Для каждого ЭлементСтруктуры Из ПараметрыОтчета Цикл

      // При таком способе обхода возможно ТОЛЬКО ЧТЕНИЕ:
      Сообщить(ЭлементСтруктуры.Ключ + " = " + ЭлементСтруктуры.Значение);

КонецЦикла;
```

Прочие возможности структур

Синтакс-помощник → Универсальные коллекции значений → Структура → Методы → …

ТАБЛИЦА ЗНАЧЕНИЙ

Что такое «таблица значений»?

Таблица значений представляет собой таблицу данных (в оперативной памяти) с произвольным количеством строк и колонок. В ячейках таблицы можно хранить значения любых типов, например:

Индекс строки	Товар	Количество	Сумма
0	«Яблоки»	50	5000
1	«Груши»	20	3600
2	«Бананы»	40	5000

Таблица значений с тремя строками

Как и в случае с массивом, к любой строке таблицы значений можно обратиться по индексу. Индексация начинается с нуля, т.е. первая строка любой таблицы всегда имеет индекс 0.

В отличие от массивов и структур, таблицы значений можно использовать только «на стороне сервера»! Попытка создания таблицы значений в «клиентской» процедуре или функции приведет к ошибке.

Создание таблицы значений

```
ТЗ = Новый ТаблицаЗначений; // Только на сервере!

ТЗ.Колонки.Добавить("Товар");
ТЗ.Колонки.Добавить("Количество");
ТЗ.Колонки.Добавить("Сумма");
```

Заполнение таблицы значений

```
НоваяСтрока = ТЗ.Добавить();

НоваяСтрока.Товар      = "Яблоки";
НоваяСтрока.Количество = 50;
НоваяСтрока.Сумма      = 5000;
```

Доступ к строкам таблицы по индексам

```
ПерваяСтрока = ТЗ[0];

Сообщить(ПерваяСтрока.Количество); // Чтение
ПерваяСтрока.Сумма = 7000;         // Изменение
```

Обход всех строк таблицы значений

```
Для каждого СтрокаТЗ Из ТЗ Цикл
      СтрокаТЗ.Сумма = СтрокаТЗ.Сумма * 2;
КонецЦикла;
```

Упорядочивание строк таблицы

```
// По колонке "Количество" по убыванию:
ТЗ.Сортировать("Количество Убыв");
```

Прочие возможности таблиц значений

Синтакс-помощник → Универсальные коллекции значений → Таблица значений → …