Общий синтаксис

Любая инструкция на этом языке записывается в виде очень простого синтаксиса, за каждым названием инструкции, после двоеточия следует список параметров, отделяемых запятой. Инструкция заканчивается точкой с запятой.

Пример инструкции:

***ИНСТРУКЦИЯ: параметр1, параметр2, … , параметр N;***

Инструкция и её параметры считываются до точки с запятой. Это значит, что в одной строчке можно как записать несколько инструкций, например:

***VAR: $FIRST, DBL3.14; VAR: $SECOND, NTG1;***

Так же можно и форматировать инструкцию для удобства на несколько строк, например:

***PRINT: ‘строка1’,***

***NTG5,***

***‘строка2’;***

Инструкции можно писать так же и в нижнем регистре, например:

***var: $FIRST, DBL3.14; var: $SECOND, NTG1;***

**ВНИМАНИЕ:** Инструкции допускается писать либо в верхнем либо в нижнем регистре, смешение символов верхнего и нижнего регистра в одной инструкции недопустимо!

**Типы данных**

В языке существуют следующие типы данных:

|  |  |  |
| --- | --- | --- |
| **Наименование** | **Описание типа** | **Диапазон** |
| UNTG | Беззнаковый целочисленный | 0 … 18 446 744 073 709 551 |
| NTG | Целочисленный | −9 223 372 036 854 775 808 … 9 223 372 036 854 775 807 |
| DBL | Вещественное число двойной точности с плавающей точкой | 1.7E - 308 … 1.7E +308 |
| CHR | Символ ASCII, байт | -128 … 127 |
| UCHR | Беззнаковый байт | 0 … 255 |
| BLN | Логический | TRUE / FALSE |
| STR | Строка из широких, 16 — ти разрядных символов | - |
| NIL | Аналог NULL | NULL |
| ARR | Массив | Массив из любых значений, в том числе других массивов и ассоциативных массивов |
| MAP | Ассоциативный массив с доступом по ключу | Ассоциативный массив из любых значений, в том числе других массивов и ассоциативных массивов |
| UNKNOWN | Внутренний тип, обозначающий ошибку работы с данными интерпретатора. Не должен появляться | UNKNOWN |

**Литералы**

Для понимания типа литерала интерпретатору необходимо сообщить тип данных перед значением. Например, чтобы создать переменную с плавающей точкой и значением 6.5, необходимо перед числом поставить метку DBL, например:

***VAR: $VAR, DBL6.5;***

Разрешено использовать экспоненциальную запись. Возможно использование как малого так и большого символа Е:

***VAR: $VAR, DBL-1.85633e+11***

А если, к примеру, необходимо создать переменную с целочисленным значением 2, то необходимо поставить метку NTG:

***VAR: $VAR, NTG2;***

Всего существуют следующие метки для литералов:

***NTG*** – целочисленный литерал

***UNTG*** – беззнаковый целочисленный литерал

***DBL*** – число двойной точности с плавающей запятой

***CHR*** – символ, значения от -128 до +127

***UCHR*** – беззнаковый символ от 0 до 255

***TRUE*** – булево значение true, истина

***FALSE*** – булево значение false, ложь

***NIL*** – значение null, ничто

***ARR*** – массив (особая метка без значения)

Для литералов типа: NTG, UNTG, DBL, CHR и UCHR возможно использование знака + для явного указания положительного значения, например:

***VAR: $VAR, NTG+13;***

**ВАЖНО:** Для удобства можно не писать тип литерала, если это число, то числа без точки интерпретируются в NTG автоматически. То же происходит, если написано число с десятичной дробью после точки. Такие литералы автоматически считаются как DBL. Кроме того, не обязательно писать идентификатор типа данных вплотную к самим данным.

***VAR: $VAR, +13; #интерпретируется как NTG***

***VAR: $VAR, 6.5; #интерпретируется как DBL***

Это может сильно облегчить читабельность кода.

Так же допустимо указывать тип литерала в нижнем регистре, например:

***var: $var, dbl 6.5;***

***var: $var, dbl -1.85633e+11***

**ВНИМАНИЕ:** Тип литерала допускается писать либо в верхнем либо в нижнем регистре, смешение символов верхнего и нижнего регистра в одном типе литерала недопустимо!

**Строки**

Строки в языке начинаются с одинарной кавычки и заканчиваются ею же. Например, вот строковый литерал:

***‘Это строка а это одинарная кавычка \‘ вот так!‘***

Чтобы использовать символы форматирования, такие, как перевод на новую строку, табуляция, необходимо использовать управляющие последовательности непосредственно в месте, где требуется символ.

Например, данная строка содержит в себе одинарную кавычку, табуляцию и перевод на новую строку

***‘Это одинарная кавычка \‘ а это перевод на новую строку \n а это табуляция \t пример.‘***

Распечатанная строка будет выглядеть так:

*Это одинарная кавычка ‘ а это перевод на новую строку*

*а это табуляция пример.*

В языке существуют следующие управляющие последовательности:

***\’*** – Одинарная кавычка

***\q*** – Одинарная кавычка

***\t*** – Табуляция

***\n*** – Перевод на новую строку

***\a*** – Звуковой сигнал

***\b*** - Backspace

***\f*** – Подача страницы

***\r*** – Возврат каретки

***\v*** – Вертикальная табуляция

***\”*** - Двойная кавычка

***\?*** - Вопросительный знак

***\\*** - Обратная косая черта

**Комментарии**

В языке имеется возможность добавления комментариев. Комментарии вырезаются из исходного когда до разбора кода на составляющие части.

**Однострочный комментарий:**

Однострочный комментарий начинается с символа #. Все, что идет после этого знака, игнорируется до конца строки. Комментарий так же, может занимать всю строку.

Примеры:

***NOP:; # Это однострочный комментарий***

***NOP:; # И это однострочный комментарий***

***# Это тоже однострочный комментарий***

**Многострочный комментарий:**

Многострочный комментарий начинается с двух символов ## подряд без пробелов между ними. Заканчивается многострочный комментарий точно так же, двумя символами ## без пробелов между ними

Примеры:

***##***

***Это***

***Многострочный***

***Комментарий***

***##***

***## И это многострочный комментарий, но в одной строке ##***

**ВНИМАНИЕ:**

Необходимо избегать внутри многострочного комментария последовательности ##, потому что в таком случае это интерпретируется как закрытие многострочного комментарий

Примеры:

***## Это неправильный многострочный комментарий с использованием ## внутри, потому что этот текст уже не комментарий, а после весь код будет закомментирован ##***

***## Закомментирован кусочек кода, это приведет к ошибке***

***PRINT: ‘## Ой!‘;***

***##***

**Список инструкций**

**Тестовые**

***NOP*** – инструкция, ничего не делающая.

Пример использования:

***NOP:;***

**Прочие**

***END*** *–* инструкция, завершающая работу основной программы или программный блок (подпрограммы). Принимает один параметр в виде имени переменной или литерала, который возвращается в виде результата выполнения подпрограммы.

Пример использования:

***END: $VAR;***

***END: NTG5****;*

***PAUSE*** – инструкция, приостанавливающая работу программы до нажатия любой клавиши на клавиатуре. Не принимает параметров

Пример использования:

***PAUSE:;***

***SLEEP*** – инструкция, приостанавливающая работу программы на определенное количество миллисекунд. После истечения указанного времени программа продолжает свою работу автоматически. Принимает один параметр. Параметр может быть именем переменной или литералом любого типа, который может быть приведен к типу UNTG.

Пример использования:

***SLEEP: NTG500;***

***SLEEP: $VARSLEEP;***

**Работа с переменными**

**Наименование переменных:**

Все названия переменных должны начинаться со знака $. Имена переменных — регистрозависимые, т. е. переменные $VAR, $Var и $var – разные переменные. Переменные начинаются со знака $ и заканчиваются оператором (,) разделяющим параметры инструкции. Поэтому внутри переменных допустимы абсолютно любые комбинации символов, спецсимволов, символов, не входящих в таблицу ASCII и даже пробелы. Единственный нюанс - замыкающие пробелы перед оператором (,) стираются. Давайте ясные и читаемые имена своим переменным.

***VAR*** – инструкция, объявляющая переменную и выделяющая место под нее в динамической памяти. Переменная может быть любого допустимого в языке типа. Принимает два параметра. Первый — имя будущей переменной. Второй — значение переменной. Имя переменной должно начинаться со знака $. Само имя переменной может содержать пробелы, спецсимволы, начинаться с цифры. Если имя переменной уже занято, это приводит к ошибке. Перезаписи значения не происходит. Для перезаписи значения существует инструкция CHANGE.

Пример использования:

***VAR: $PI, DBL3.14;***

***VAR: $FIRST, NTG1665;***

***FREE –*** инструкция, высвобождающая динамическую память от указанной переменной и удаляющая ее в конечном итоге. Принимает произвольное количество параметров. Это должны быть имена переменных. Необходимо тщательно следить за объявленными переменными и освобождать от них память, если они будут более не нужны, во избежание утечки памяти.

Пример использования:

***FREE: $FIRST, $SECOND, $THIRD;***

***CHANGE –*** инструкция, позволяющая изменить значение уже созданной переменной. Так как инструкция VAR имеет главное предназначение в создании переменных, инструкция CHNG – единственный способ присвоить уже существующей переменной другое значение. Принимает два параметра: Первый параметр — имя существующей переменной, значение которой будет изменено. Второй параметр — значение, на которое будет изменено значение первой переменной. Второй параметр может быть как литералом, так и именем другой переменной.

Пример использования:

***CHANGE: $SECOND, $FIRST;***

***NEWTEMP*** – создает блок из временных переменных. Служит для того, чтобы не утруждать себя созданием временных переменных через команду VAR. Все созданные переменные изначально заполняются значением NIL. Принимает 1 параметр. Параметр нужен для указания количества выделяемых временных переменных. Все временные переменные выглядят как $0, $1, $2 и т.д. Поэтому не стоит объявлять переменные, названные таким способом через инструкцию VAR.

Пример использования:

***NEWTEMP: 10;***

***FORGET*** – очищает память от временных переменных, удаляя их. Принимает от 0 до 1 параметра. В случае, если передано 0 параметров, команда удаляет из памяти все выделенные переменные. В случае, если принимает 1 параметр, удаляет то количество временных переменных с конца, какое указано в параметре.

Пример использования:

***FORGET:;***

***FORGET: 10;***

***TCOUNT*** – инструкция, которая кладет в переменную значение количества временных переменных

Пример использования:

***NEWTEMP: 10;***

***FORGET: 6;***

***TCOUNT: $0;***

***PRINT: $0;***

выведет 4.

***ISSET*** – инструкция, проверяющая существование переменной в памяти. Принимает 2 параметра. 1 параметр используется для хранения результата команды, 2 параметр — имя переменной для проверки.

Пример использования:

***var: $result, nil;***

***isset: $result, $var;***

***print: $result;***

выведет FALSE

***var: $var, nil;***

***var: $result, nil;***

***isset: $result, $var;***

***print: $result;***

выведет TRUE

***TYPEOF*** – инструкция, определяющая тип переменной. Принимает 2 параметра. Первый параметр — всегда переменная, в которую кладется результат выполнения инструкции. Второй параметр — литерал или переменная, по которому определяется тип данных.

Возможные типы данных: NTG, UNTG, DBL, CHR, UCHR, BLN, STR, ARR, MAP.

Пример использования:

***NEWTEMP: ntg3;***

***CHANGE: $0, '3.1415';***

***TYPEOF: $1, $0;***

***PRINT: $1, '\n';***

*выведет STR.*

***SWAP –*** инструкция, меняющая значение двух переменных местами. Принимает только 2 параметра:

1 параметр (обязательно) имя перемнной 1;

2 параметр (обязательно) имя переменной 2;

Пример использования:

***VAR: $1, 1;***

***VAR: $2, 2;***

***PRINT: $1, ‘ ‘, $2, ‘\n’;***

***SWAP: $1, $2;***

***PRINT: $1, ‘ ‘, $2, ‘\n’;***

выведет:

1 2

2 1

**Условные и безусловные переходы, метки**

***LABEL* –** инструкция, устанавливающая в код метку, на которую после можно ссылаться инструкциями перехода, что переводит исполнение кода на инструкцию после метки. Принимает один особый параметр — имя метки. Имя метки всегда начинается с символа &. (*Техническая информация*. Все метки обрабатываются до начала исполнения кода и заносятся в специальную память для меток и содержат номер строки с инструкцией. После перехода на метку указатель на исполняемую инструкцию становится на следующую инструкцию сразу после метки. При выполнении инструкции метки ничего не происходит, как при инструкции NOP).

Пример использования:

***LABEL: &LABEL1;***

***DLABEL –*** инструкция, позволяющая создавать метки в динамике и перезаписывать их по названию прямо во время работы кода. На такую метку можно будет перескочить после того, как она была создана этой командой (это значит, что команды JUMP, JIF, JIFNOT могут перейти на метку, созданную этой инструкцией только если инструкция будет расположена выше по коду, чем указанные команды.) После перехода на динамическую метку,указатель на исполняемую инструкцию становится на инструкцию, следующую за этой меткой.

Пример использования:

***DLABEL: &dynamic;***

***JUMP***– инструкция, выполняющая безусловный переход на указанную метку (указатель на исполняемую инструкцию „прыгает“ на эту метку). Принимает один параметр — имя метки, на которую нужно сделать прыжок.

Пример использования:

***JUMP: &LABEL1;***

***JIF*** – инструкция, которая выполняет условный переход по метке на основании 1 параметра. Принимает 2 параметра:

1 параметр — Основание на котором делается или не делается переход.

2 параметр — метка, на которую будет сделан переход. Переход состоится, если первый параметр будет равен true или приведение типов 1 параметра к BLN приведет значение параметра к true. Подробности в таблицах приведения типов.

Пример использования:

***JIF: $condition, &label;***

***JIFNOT*** – практически такая же инструкция, как и предыдущая, но в данном случае переход делается, если первый параметр равен false

**Ввод/вывод**

***INPUT –*** инструкция, выполняющая запрос на ввод информации с клавиатуры пользователя в консоль. Принимает один параметр, обязательно являющийся именем существующей переменной, в которое кладется введенное значение. Важное примечание: Инструкция INPUT отдает в переменную строковое значение. Если необходим другой тип данных, например, численный, после ввода необходимо сделать приведение типов до нужного типа.

Пример использования:

***INPUT: $MY\_VAR;***

***PRINT*** – инструкция, выводящая в консоль (на экран) данные в строковом представлении. Принимает произвольное количество параметров, отделенных запятой (но обязательно не меньше одного). Не может отобразить содержимое обычных массивов и ассоциативных массивов напрямую.

Пример использования:

***PRINT: 'переменная #1 равна: ', $FIRST,***

***'переменная #2 равна: ', $SECOND,***

***'\n';***

**Приведение типов**

Приведение типов производится инструкцией TO.

***TO*** – инструкция, приводящая иной тип переменной к указанному типу. Принимает 2 либо 3 параметра.

1 параметр: Строковая переменная или строковый литерал с типом данных, к которому приводится значение. Допустимые типы данных: NTG, UNTG, DBL, CHR, UCHR, BLN, STR, ARR. Возможно указывать тип данных в нижнем регистре, например: ntg, bln, str и т. д. Смешанный регистр названия типа данных недопустим (например, Ntg, BlN)!

2 параметр: Всегда является именем переменной. В данную переменную заносится результат приведения типов.

3 параметр (опционально): Переменная или литерал, из которых берется значение для приведения.

В случае, если используется 2 параметра, берется значение 2 параметра и приводится к типу, указанному в 1 параметре. Результат кладется в переменную 2 параметра. В случае, если используется 3 параметра, берется значение 3 параметра, приводится к типу, указанному в 1 параметре. Результат кладется в переменную 2 параметра

Пример использования:

c двумя параметрами, тип данных указан в переменной:

***VAR: $CAST, 'NTG';***

***VAR: $VAR, DBL3.14;***

***TO: $CAST, $VAR;***

*c двумя параметрами, тип данных указан в инструкции:*

***VAR: $VAR, DBL3.14;***

***TO: 'NTG', $VAR;***

*с тремя параметрами, тип данных указан в переменной:*

***VAR: $CAST, 'NTG';***

***VAR: $SOURCE, DBL3.14;***

***VAR: $RESULT, NIL;***

***TO: $CAST, $RESULT, $SOURCE;***

*с тремя параметрами, тип данных указан в инструкции:*

***VAR: $SOURCE, DBL3.14;***

***VAR: $RESULT, NIL;***

***TO: 'NTG', $RESULT, $SOURCE;***

Таблицы приведения типов:

**К NTG:**

|  |  |
| --- | --- |
| **Приводимый тип** | **Описание особенностей:** |
| NTG | Приводится к типу NTG без потери данных |
| UNTG | При выходе значения UNTG за пределы NTG, NTG получает отрицательное значение |
| DBL | Отбрасывается дробная часть. При выходе за пределы NTG результат недетерминирован |
| CHR | Приводится к типу NTG без потери данных |
| UCHR | Приводится к типу NTG без потери данных |
| BLN | FALSE приводится к 0  TRUE приводится к 1 |
| STR | Происходит попытка преобразовать строку в число, если строка состоит из цифр. Если же это невозможно, выбрасывается исключение. |
| NIL | Приводится к 0 |
| ARR | Выбрасывается исключение |
| MAP | Выбрасывается исключение |
| UNKNOWN | Приводится к 0 |

**К UNTG:**

|  |  |
| --- | --- |
| **Приводимый тип** | **Описание особенностей:** |
| NTG | При выходе за пределы UNTG (если значение отрицательное) получает положительное значение |
| UNTG | Приводится к типу UNTG без потери данных |
| DBL | Отбрасывается дробная часть. При выходе за пределы UNTG результат недетерминирован |
| CHR | При выходе за пределы UNTG (если значение отрицательное) получает положительное значение |
| UCHR | Приводится к типу UNTG без потери данных |
| BLN | FALSE приводится к 0  TRUE приводится к 1 |
| STR | Происходит попытка преобразовать строку в число, если строка состоит из цифр. Если же это невозможно, выбрасывается исключение. |
| NIL | Приводится к 0 |
| ARR | Выбрасывается исключение |
| MAP | Выбрасывается исключение |
| UNKNOWN | Приводится к 0 |

**К DBL:**

|  |  |
| --- | --- |
| **Приводимый тип** | **Описание особенностей:** |
| NTG | При больших значениях часть мантиссы теряется. |
| UNTG | При больших значениях часть мантиссы теряется. |
| DBL | Приводится к типу DBL без потери данных |
| CHR | Приводится к типу DBL без потери данных |
| UCHR | Приводится к типу DBL без потери данных |
| BLN | FALSE приводится к 0.0  TRUE приводится к 1.0 |
| STR | Происходит попытка преобразовать строку в число. Если же это невозможно, выбрасывается исключение. |
| NIL | Приводится к 0.0 |
| ARR | Выбрасывается исключение |
| MAP | Выбрасывается исключение |
| UNKNOWN | Приводится к 0.0 |

**К CHR:**

|  |  |
| --- | --- |
| **Приводимый тип** | **Описание особенностей:** |
| NTG | При значениях, выходящих за пределы CHR происходит переполнение с потерей данных |
| UNTG | При значениях, выходящих за пределы CHR происходит переполнение с потерей данных |
| DBL | Отбрасывается дробная часть. При значениях, выходящих за пределы CHR происходит переполнение с потерей данных |
| CHR | Приводится к типу CHR без потери данных |
| UCHR | При значениях, выходящих за пределы CHR происходит переполнение. |
| BLN | FALSE приводится к 0  TRUE приводится к 1 |
| STR | Происходит попытка преобразовать строку в число и преобразовать это число к типу CHR. Если же это невозможно, выбрасывается исключение. |
| NIL | Приводится к 0 |
| ARR | Выбрасывается исключение |
| MAP | Выбрасывается исключение |
| UNKNOWN | Приводится к 0 |

**К UCHR:**

|  |  |
| --- | --- |
| **Приводимый тип** | **Описание особенностей:** |
| NTG | При значениях, выходящих за пределы UCHR происходит переполнение с потерей данных |
| UNTG | При значениях, выходящих за пределы UCHR происходит переполнение с потерей данных |
| DBL | Отбрасывается дробная часть. При значениях, выходящих за пределы UCHR происходит переполнение с потерей данных |
| CHR | При значениях, выходящих за пределы UCHR происходит переполнение. |
| UCHR | Приводится к типу UCHR без потери данных |
| BLN | FALSE приводится к 0  TRUE приводится к 1 |
| STR | Происходит попытка преобразовать строку в число и преобразовать это число к типу UCHR. Если же это невозможно, выбрасывается исключение. |
| NIL | Приводится к 0 |
| ARR | Выбрасывается исключение |
| MAP | Выбрасывается исключение |
| UNKNOWN | Приводится к 0 |

**К BLN:**

|  |  |
| --- | --- |
| **Приводимый тип** | **Описание особенностей:** |
| NTG | Любое значение, кроме 0 приводится к TRUE, значение 0 приводится к FALSE |
| UNTG | Любое значение, кроме 0 приводится к TRUE, значение 0 приводится к FALSE |
| DBL | Любое значение, кроме 0 приводится к TRUE, значение 0 приводится к FALSE |
| CHR | Любое значение, кроме 0 приводится к TRUE, значение 0 приводится к FALSE |
| UCHR | Любое значение, кроме 0 приводится к TRUE, значение 0 приводится к FALSE |
| BLN | Приводится к типу BLN без потери данных |
| STR | Пустая строка приводится к значению FALSE. Любая другая к значению TRUE. |
| NIL | Приводится к FALSE |
| ARR | Пустой массив приводится к значению FALSE. Любой другой к значению TRUE. |
| MAP | Пустой ассоциативный массив приводится к значению FALSE. Любой другой к значению TRUE. |
| UNKNOWN | Приводится к FALSE |

**К STR:**

|  |  |
| --- | --- |
| **Приводимый тип** | **Описание особенностей:** |
| NTG | Приводится к типу STR без потери данных |
| UNTG | Приводится к типу STR без потери данных |
| DBL | Приводится к типу STR с возможным сокращением слишком длинной дробной части |
| CHR | Приводится к числу, число приводится к типу STR без потери данных |
| UCHR | Приводится к строковому виду без потери данных |
| BLN | TRUE приводится к строковому значению TRUE. FALSE к FALSE |
| STR | Приводится к типу STR без потери данных |
| NIL | Приводится к пустой строке |
| ARR | Выбрасывается исключение |
| MAP | Выбрасывается исключение |
| UNKNOWN | Приводится к пустой строке |

**К ARR:**

|  |  |
| --- | --- |
| **Приводимый тип** | **Описание особенностей:** |
| NTG | Приводится к массиву из 1 элемента, значение которого равно значению приводимого типа |
| UNTG | Приводится к массиву из 1 элемента, значение которого равно значению приводимого типа |
| DBL | Приводится к массиву из 1 элемента, значение которого равно значению приводимого типа |
| CHR | Приводится к массиву из 1 элемента, значение которого равно значению приводимого типа |
| UCHR | Приводится к массиву из 1 элемента, значение которого равно значению приводимого типа |
| BLN | Приводится к массиву из 1 элемента, значение которого равно значению приводимого типа |
| STR | Приводится к массиву из 1 элемента, значение которого равно значению приводимого типа |
| NIL | Приводится к массиву из 1 элемента, значение которого равно значению приводимого типа |
| ARR | Остается прежним с прежними значениями |
| MAP | Выбрасывается исключение |
| UNKNOWN | Приводится к массиву из 1 элемента, значение которого равно значению приводимого типа |

**Математические операции:**

Приведение типов производится инструкцией CALC.

***CALC*** – инструкция, совершающая указанную математическую операцию. Может иметь от 2 до 4 параметров.

1 параметр: Строковая переменная или строковый литерал с типом выполняемой математической операции. Допустимые математические операции: +, -, \*, /, % (деление по модулю), ^ (степень), ROOT (корень) INC (инкремент), DEC (декремент), LOG (логарифм), LN (натуральный логарифм), FACT(факториал). Допустимо указывать названия математических операций в нижнем регистре, например: root, inc, dec, log и т. д. Но не в смешанном регистре!

2 параметр: Всегда является именем переменной. В данную переменную заносится результат математической операции.

3 параметр (опционально): Переменная или литерал, из которых берется значение для произведения математической операции. Если передается только 2 параметра, значение берется из 2 параметра и в него же записывается

4 параметр (опционально): Переменная или литерал, из которых берется значение для произведения математической операции.Если передано только 3 параметра, то берется число из 3 параметра, результат заносится во 2 параметр.

Примеры использования:

Инструкции с двумя параметрами:

***VAR: $val, ntg 5;***

***CALC: ‘inc’, $val;***

*Результат: $val = 6*

***VAR: $val, ntg 5;***

***CALC: ‘dec’, $val;***

*Результат: $val = 4*

***VAR: $val, ntg 5;***

***CALC: ‘fact’, $val;***

*Результат: $val = 120*

*Инструкции с 3 параметрами:*

*С 3 параметрами могут быть так же и математические операторы inc, dec, fact, ln. Остальные математические операторы имеют от 3 до 4 параметров. Операторы, принимающие 2-3 параметра не могут быть использованы в инструкции с 4 параметрами.*

***VAR: $result, ntg 5;***

***VAR: $operand, ntg 7;***

***CALC: ‘+’, $result, $operand;***

*Результат: $result = 12*

*Инструкции с 4 параметрами:*

***VAR: $result, ntg 5;***

***VAR: $operand1, ntg 7;***

***VAR: $operand2, ntg 6;***

***CALC: ‘-’, $result, $operand1, $operand2;***

*Результат: $result = 1*

*Использование операций ROOT, LOG:*

*Квадратный корень из 100:*

***VAR: $result, NIL;***

***VAR: $operand1, ntg100;***

***CALC: ‘root’, $result, $operand1, ntg 2;***

*Результат: $result = 1*

*Кубический корень из 8:*

***VAR: $result, NIL;***

***VAR: $operand1, ntg 8;***

***VAR: $operand2, ntg 3;***

***CALC: ‘root’, $result, $operand1, $operand2;***

*Результат: $result = 2*

*Логарифм 25 по основанию 5:*

***VAR: $result, NIL;***

***VAR: $operand1, ntg 25;***

***VAR: $operand2, ntg 5;***

***CALC: ‘log’, $result, $operand1, $operand2;***

*Результат: $result = 2*

*Логарифм 8 по основанию 2:*

***VAR: $result, NIL;***

***VAR: $operand1, ntg 8;***

***VAR: $operand2, ntg 2;***

***CALC: ‘log’, $result, $operand1, $operand2;***

*Результат: $result = 3*

***Таблица неявного приведения типов:***

*При арифметических операциях с разными типами данных, производится неявное приведение типов. Оно указано ниже:*

|  |  |
| --- | --- |
| **типы операндов** | **тип результата** |
| dbl / dbl | dbl |
| dbl / ntg | dbl |
| ntg / dbl | dbl |
| ntg / ntg | ntg |

Любые математические действия с NIL приведут к результату NIL.

Любые математические действия с ARR и MAP приведут к выбросу исключения на попытке расчета.

Строки всегда пытаются преобразоваться в тот же тип, что и другой нестроковый параметр. Если у нас складываются две строки, происходит конкатенация а не математическая сумма.

Остальные правила неявного преобразования типов такие же, как и у С/С++

***COMP –*** инструкция, делающая математические сравнения. Всегда имеет 4 параметра.

1 параметр — тип математического сравнения (==, !=, >, <, >=, <=) передаваемый в строковом литерале или в переменной.

2 параметр — всегда имя существующей переменной. В нее записывается результат в виде булева значения (TRUE/FLASE).

3 параметр — переменная или литерал, первое сравниваемое значение.

4 параметр — переменная или литерал, второе сравниваемое значение.

Примеры использования:

***VAR: $one, ntg 5;***

***VAR: $two, dbl 7.0;***

***VAR: $result, nil;***

***COMP: ‘>’, $result, $one, $two;***

Результат: FALSE

***ВНИМАНИЕ:***

При разных типах сравниваемых данных происходит неявное приведение типов. Правила неявного преобразования типов такие же, как и у С/С++

***LOGIC*** – инструкция, предоставляющая набор логических вентилей, таких, как И, ИЛИ, НЕ и т.д. Принимает 3 параметра для инструкции NOT и 4 для всех остальных вентилей. Перед тем, как произвести логическую операцию, все типы данных приводятся к типу BLN (смотреть подробно таблицы приведений в инструкции NOT)

1 параметр: название операции, литералом или переменной строкового типа. Поддерживает следующие операции: NOT, AND, OR, NAND, NOR, XOR, XNOR. Они же доступны в маленьком регистре: not, and, or, nand, nor, xor, xnor.

2 параметр: переменная, в которую будет выведен результат

3 параметр: первая переменная или литерал, использующийся для произведения логического действия.

4 параметр (не обязательно): вторая переменная или литерал, использующийся для проведения логического действия.

Примеры использования:

***var: $result, nil;***

***logic: ’OR’, $result, true, false;***

вернет true;

***var: $one, 'xnor';***

***var: $two, dbl0;***

***var: $three, false;***

***var: $result, nil;***

***logic: $one, $result, $two, $three;***

вернет true

**Работа с массивами словарями и строками:**

***VAR –*** инструкция, способная создавать кроме обычных переменных пустые одномерные массивы. Принимает 2 параметра:

1 параметр (обязательно) — имя для нового массива.

2 параметр (обязательно) — указание массивного типа данных.

Пример использования:

***VAR: $my\_array, ARR;***

***ARRAY*** – инструкция, создающая многомерный массив, мерности не ограничены. Созданные многомерные массивы изначально заполнены значением null. Принимает не меньше 2 параметров.

1 параметр (обязательно) — имя нового массива

2 параметр (обязательно) — первая мерность массива

3+ параметры (опционально) — остальные мерности массива

Пример использования:

***ARRAY: $my\_******parallelepiped, 5, 5, 7;***

*создает трехмерный массив 5х5х7*

***VTOARR* –** инструкция, преобразующая ряд перечисленных параметров в массив, содержащий их. Принимает минимум 2 параметра.

1 параметр (обязательно) — имя нового массива

2 параметр (обязательно) — первый элемент массива

3+ параметры (опционально) — остальные элементы нового массива

Пример использования:

**var: $arr, arr;**

**var: $dbl, 3.14;**

**vtoarr: $new\_arr, $arr, 123, $dbl, 'string';**

**print: $new\_arr, '\n';**

*выведет: [[], 123, 3.140000, string]*

***PUSHB*** – инструкция, заталкивающая в конец массива указанное значение. Всегда принимает 2 обязательных параметра.

1 параметр (обязательно) — имя массива, в который заталкивается значение

2 параметр (обязательно) — имя переменной или литерал, указывающий значение, помещаемое в массив

Пример использования:

***vtoarr: $arr, 1, 2, 3;***

***print: $arr, '\n';***

***pushb: $arr, 4;***

***print: $arr, '\n';***

*выведет:*

*[1, 2, 3]*

*[1, 2, 3, 4]*

***POPB*** – инструкция, выталкивающая из конца массива значение. Принимает от 1 до 2 параметров.

1 параметр (обязательно) — имя массива, из которого выталкивается значение или имя переменной, в которое запишется выталкиваемое значение, если используется второй параметр.

2 параметр (опционально) — имя массива, из которого выталкивается значение

Пример использования:

***var: $popped, nil;***

***vtoarr: $arr, 1, 2, 3;***

***print: $arr, '\n';***

***popb: $arr;***

***print: $arr, '\n';***

***popb: $popped, $arr;***

***print: $arr, '\n';***

***print: $popped, '\n';***

*выведет:*

*[1, 2, 3]*

*[1, 2]*

*[1]*

*2*

***PUSHF*** – инструкция, заталкивающая в начало массива указанное значение. Всегда принимает 2 обязательных параметра.

1 параметр (обязательно) — имя массива, в который заталкивается значение

2 параметр (обязательно) — имя переменной или литерал, указывающий значение, помещаемое в массив

**ВНИМАНИЕ:** Если имеется возможность заменить PUSHF на PUSHB, стоит заменять. PUSHB работает быстрее, а PUSHF после заталкивания значения в начало, сдвигает остальные значения в конец, перезаписывая их.

Пример использования:

***vtoarr: $arr, 1, 2, 3;***

***print: $arr, '\n';***

***pushf: $arr, 4;***

***print: $arr, '\n';***

*выведет:*

*[1, 2, 3]*

*[4, 1, 2, 3]*

***POPF*** – инструкция, выталкивающая из начала массива значение. Принимает от 1 до 2 параметров.

1 параметр (обязательно) — имя массива, из которого выталкивается значение или имя переменной, в которое запишется выталкиваемое значение, если используется второй параметр.

2 параметр (опционально) — имя массива, из которого выталкивается значение

**ВНИМАНИЕ:** Если имеется возможность заменить POPF на POPB, стоит заменять. POPB работает быстрее, а POPF после удаления значения с начала массива, сдвигает остальные значения, перезаписывая их.

Пример использования:

***var: $popped, nil;***

***vtoarr: $arr, 1, 2, 3;***

***print: $arr, '\n';***

***popf: $arr;***

***print: $arr, '\n';***

***popf: $popped, $arr;***

***print: $arr, '\n';***

***print: $popped, '\n';***

*выведет:*

*[1, 2, 3]*

*[2, 3]*

*[3]*

*2*

***ERASE –*** инструкция, стирающая значение по его индексу и сдвигающая последующие элементы на место удаленного элемента. Принимает от 2 до 3 параметров.

Если 2 параметра:

1 параметр — имя массива, из которого стирается значение.

2 параметр — индекс элемента массива, который стирается.

Если 3 параметра:

1 параметр — имя переменной, которой присвоится стираемое значение.

2 параметр — имя массива, из которого стирается значение.

3 параметр — индекс элемента массива, который стирается.

Пример использования:

***var: $erased, nil;***

***vtoarr: $arr, 1, 2, 3, 4, 5;***

***print: $arr, '\n';***

***erase: $arr, 1;***

***print: $arr, '\n';***

***erase: $erased, $arr, 2;***

***print: $arr, '\n';***

***print: $erased, '\n';***

*выведет:*

*[1, 2, 3, 4, 5]*

*[1, 3, 4, 5]*

*[1, 3, 5]*

*4*

***INSERT*** *– инструкция, вставляющая элемент в массив по индексу, сдвигает элементы после вставленного на позицию вперед. Принимает 3 обязательных параметра.*

1 параметр — имя массива, в который вставляется значение.

2 параметр — имя переменной или литерал со значением, которое вставляется массив.

3 параметр — индекс элемента массива, на место которого вставляется значение.

Пример использования:

***var: $inserted, 'inserted';***

***vtoarr: $arr, 1, 2, 3, 4, 5;***

***print: $arr, '\n';***

***insert: $arr, $inserted, 2;***

***print: $arr, '\n';***

*выведет:*

*[1, 2, 3, 4, 5]*

*[1, 2, inserted, 3, 4, 5]*

***CLEAR –*** *инструкция, стирающая все элементы массива. Принимает 1 обязательный параметр.*

*1 параметр (обязательно) — имя стираемого массива.*

Пример использования:

***vtoarr: $arr, 1, 2, 3, 4, 5;***

***print: $arr, '\n';***

***clear: $arr;***

***print: $arr, '\n';***

*выведет:*

*[1, 2, 3, 4, 5]*

*[]*

***SIZE*** *–* инструкция, возвращающая число элементов массива. Принимает 2 обязательных параметра.

1 параметр (обязательно) — имя переменной, которой присваивается длина массива.

2 параметр (обязательно) — имя массива, по которому определяется размер.

Пример использования:

***var: $sz, nil;***

***vtoarr: $arr, 1, 2, 3, 4, 5;***

***size: $sz, $arr;***

***print: $sz, '\n';***

*выведет: 5*