Элементы языка QBasic

Элементы языка QBasic

1. Ключевые слова

Некоторые слова, используемые в программах, имеют особый смысл и поэтому называются ключевыми (или зарезервированными) . Интерпретация или компиляция ключевого слова всегда вызывает вполне определенное действие компьютера. Например, ключевое слово PRINT задает вывод информации на экран.

2. Синтаксические соглашения

Имеются различные способы описания синтаксиса языковых конструкций. Для описания элементов языка QBASIC будет использоваться та же форма описания синтаксиса, что и в Help-системе QBASIC.

Элементы языка QBasic

Элемент	Значение
Ключевое слово	Ключевые слова QBASIC всегда записываются большими (заглавными) буквами
Метка-заполнитель	Выражение
[ ]	Необязательный параметр
{A1|A2}	Выбор одного из альтернативных значений
...	Три точки, следующие одна за другой, обозначают, что предшествующий элемент может повторяться
Ключевое слово1 . . . Ключевое слово2	Три точки одна под другой обозначают несущественный в данном контексте фрагмент программы.

Элементы языка QBasic

3. Комментарии

Включенные в программу комментарии QBASIC игнорирует. Использование комментариев позволяет включать в программу любой текст, не нарушая ее семантики. В QBASIC начало комментария обозначается знаком ' (одинарная кавычка). Остаток строки нельзя переносить в начало следующей строки, содержащей оператор.

Пример:

```
'Это комментарий
'PRINT без дополнений выводит на экран пустую строку
PRINT
PRINT 'За каждым оператором следует комментарий
PRINT
'Вывод пустой строки
PRINT
```

Элементы языка QBasic

4. Арифметические выражения.

Арифметическое выражение — это комбинация чисел, арифметических констант и некоторых функций, которые связаны между собой знаками арифметических операций. Числа — простейшая форма арифметических выражений. Число состоит из цифр "0...9" и специальных знаков "- + . D".

Вместо запятой ",", отделяющей целую часть от дробной, используется точка ".".

4.23 вместо 4,23

9.23D+78 вместо $9,23*10^{78}$

-2.77 вместо -2,77

-1.34D-23 вместо $-1,34*10^{-23}$

Элементы языка QBasic

4. Арифметические выражения

Сложные арифметические выражения можно получить, если связать в программе числа друг с другом, например, для вычислений. Символы, с помощью которых числа вступают друг с другом в некоторые отношения, называют знаками операций.

Примечание: В QBASIC операции "*", "/", "^" имеют больший приоритет, чем операции "+","-". Впрочем на последовательность вычислений можно влиять, применяя в нужных местах круглые скобки.

Знак	Значение
*	умножение
/	деление
+	сложение
-	вычитание
(	открывающая скобка
)	закрывающая скобка
^	возведение в степень

Элементы языка QBasic

Оператор **PRINT** дает возможность работать с арифметическими выражениями.

PRINT [Выражение]

Выражение - Арифметическое выражение. Значение выражения выводится на экран. Так как в операторе PRINT выражение — необязательный параметр, то с его помощью можно выводить на экран пустую строку.

Пример:

```
'Пример программы  применения оператора PRINT
'Для начала выведем простые числа
PRINT 12
PRINT 23.34
PRINT-345.454
PRINT 'Вывод пустых строк
PRINT
PRINT
'Вывод арифметических выражений
PRINT 12-4
PRINT+6+56-6
PRINT 3*4-4* (4+1)
PRINT 3+2^3
'Конец программы
```

Элементы языка QBasic

5. Строковые выражения

Строковое выражение — это комбинация текстов, текстовых констант, текстовых переменных и определенных функций, которые связаны операцией "+". Простейшая строка — это текст, заключенный в кавычки ("), состоящий из произвольных комбинаций букв, цифр и специальных знаков. Тексты используются, например, чтобы выводить на экран заглавия, комментарии или примечания.

Примеры:

"Задайте высоту"

"2345,34"

"Программа окончена? (Y/N)"

"Эта программа вычисляет объем"

Элементы языка QBasic

Оператор **PRINT** позволяет работать со строковыми выражениями.

PRINT [Выражение]

Выражение - Числовое или строковое выражение. Значение выражения выводится на экран.

Пример:

```
'Вывод выражений
PRINT "Вывод чисел:"
PRINT 23.4
PRINT-10.2
PRINT
PRINT
PRINT "Вычислим (10+4) - 4*(2-3'^2)"
PRINT (10 + 4)-4* (2-3^2)
PRINT
PRINT "В заключение объединим отдельные"
PRINT
PRINT "слова в текст:"
PRINT "Сегодня" + " " + "хорошая" + " погода"
'Конец программы
```

Элементы языка QBasic

6. Типы данных

В простейшем случае различают два типа данных: числовые и строковые. В QBASIC разные типы данных должны иметь четкие различия и признаки.

Стандартные типы данных.

Тип	Символ	Содержание
INTEGER	%	целые числа в интервале от -32 768 до +32 768
LONG	&	целые числа в интервале от -2 147483648 до +2 147483648
SINGLE	!	числа с плавающей запятой в интервале от ±8,43*10-37 до ± 3,37*1038
DOUBLE	#	числа с плавающей запятой в интервале от ±4,19*10-307 до ±1,67*10308
STRING	$	любые тексты с максимальной длиной 32 767 знаков

Примечание: Обработка данных типа INTEGER производится значительно быстрее, чем данных типа DOUBLE.

Константы и переменные

Константы

Константы определяют в начале программы. После определения они могут использоваться во всей программе.

CONST Имя_константы = Выражение [, Имя_константы "Выражение]...

Имя_константы - Имя константы содержит максимум 40 символов и должно начинаться с буквы. Допустимые символы: "A...Z","0...9" и точка".". Причем несущественно, строчные или заглавные буквы составляют имя константы. Последним должен быть один из символов, идентифицирующих тот или иной стандартный тип (! # % $). По умолчанию QBASIC сам присвоит тип SINGLE. Следует следить за тем, чтобы не присвоить константе ключевое слово в качестве имени. Выражение - Значение, которое должно храниться в константе. Это значение может быть просто числом, другой константой, текстом или арифметическим выражением (не применять операцию"^").

Пример:

```
'Работа с константами
CONST p1!-3.14
CONST mwst% - 14
```

Константы и переменные

Переменные
Так же, как и константы, переменные определяют в самом начале программы. В отличие от констант, значения переменных могут меняться в процессе выполнения программы.

Объявление переменных выглядит так:

DIM [SHARED] Имя_переменной [, Имя_переменной]...

SHARED - Указание на то, что переменную можно использовать во всех функциях и процедурах программы.
Имя_переменной - Его максимальная длина 40 символов. Имя переменной подчиняется тем же правилам, что и имя константы.

Примечание: переменные не обязательно объявлять в начале программы. Если во время работы программы необходима переменная,ее можно просто применять.

Примеры:

```
DIM SHARED height!
DIM breadth!
DIM height!, length!, okl$, telefon$
```

Константы и переменные

Присвоение значений переменным

Присвоить некоторое значение переменной можно следующим образом:

Имя_переменной = {Строковое_выражение | Арифметическое_выражение}

Пример:

LET X=“СТРОКОВАЯ КОНСТАНТА“
sim=F+A/2.345

Ввод/вывод

Вывод на экран
Оператор **PRINT** позволяет выводить данные на экран.

PRINT [Выражение] [{;|,} Выражение]...[{;|,}]

Выражение - Арифметическое или строковое выражение. Значение выражения выводится на экран.
{;|,} - Устанавливает начало последующего вывода:
;: — значение очередного выражения следует непосредственно за значением предыдущего,
, — значение очередного выражения размещается в начале следующей области вывода. Область вывода — блок из 14 символов.
Если в конце **PRINT**-выражения нет ни знака ";", ни ",", то выполняется перевод строки, и вывод следующего значения начинается с новой строки.

Примечание: Оператор **LPRINT** выводит данные на печатающее устройство (принтер). В остальном синтаксис **LPRINT** идентичен синтаксису оператора **PRINT**.

Ввод/вывод

Очистка экрана.

Оператор ***CLS*** очищает экран и используется без параметров. Следующий оператор вывода начинает размещение данных в левом верхнем углу экрана (первая строка, первый столбец).

Пример:

```
'В данной программе демонстрируется очистка экрана
CLS
'Экран очищен
PRINT "Вывод в левый верхний угол, ";
PRINT "остальное поле экрана свободно"
'Конец программы
```

Ввод/вывод

Ввод с клавиатуры

Описанный далее оператор ***INPUT*** дает возможность вводить данные во время выполнения программы, после чего эти данные обрабатываются программой. Появление на экране вопросительного знака означает запрос ввода. Ввод данных завершается нажатием клавиши , и введенное значение сохраняется в заданной переменной.

INPUT [Текст{;|,}] Имя_переменной [, Имя_переменной].

Текст- Текст, который будет выведен на экран перед тем, как пользователь осуществит ввод.
{;|,} - Точка с запятой обозначает, что после текста выводится знак вопроса для обозначения необходимости ввода данных. Если вместо точки с запятой стоит запятая, знак вопроса после текста не появится.
Имя_переменной - Переменные, в которых сохраняются вводимые данные. Объявление переменных должно быть выполнено в начале программы.

Ввод/вывод

Ввод с клавиатуры

```
'Пример 1
DIM height!, breadth!, length!
CLS
INPUT height!
INPUT breadth!
INPUT length!
PRINT
PRINT "Результат "; height! * breadth! * length!
'Конец программы
```

Ввод/вывод

Ввод с клавиатуры

```
'Пример 2
DIM height!, breadth!, length!
CLS
INPUT "Введите, пожалуйста, высоту :"; height!
INPUT "Введите, пожалуйста, ширину :"; breadth!
INPUT "Введите, пожалуйста, длину :"; length!
PRINT
PRINT "Результат V="; height! * breadth! * length!
'Конец программы
```

Применение точки с запятой приводит к тому, что после вывода текста на экране появляется знак вопроса, вывод которого можно отменить, если точку с запятой заменить запятой. Предложение INPUT будет выглядеть так:

```
INPUT "Введите, пожалуйста, высоту :", height!
```

Ввод/вывод

Форматированный вывод на экран

PRINT USING Шаблон Выражение [;]

Шаблон- Определяет вид, в котором выводятся выражения.
Выражение- Арифметическое или строковое выражение.

Значение выражения появляется на экране, преобразованное шаблоном.

Как и в операторе PRINT, точка с запятой подавляет перевод строки. Шаблон в операторе PRINT USING всегда заключен в двойные кавычки (""). Внутри кавычек находятся символы, определяющие формат вывода.

Ввод/вывод

Форматированный вывод на экран

#	Вывод одного цифрового разряда. Если выводимое значение короче, чем количество цифровых знаков в шаблоне, то число выравнивается по правому краю и лишние (свободные) позиции слева заполняется пробелами.
+	Вывод знака. Выводит в явном виде знак "+". Вывод пробела, если выводимое значение положительно.
-	Вывод пробела, если выводимое значение положительно, и знака "-", если значение отрицательно.
**	Ведущие пробелы заменяются звездочками
$$	Ведущие пробелы заменяются знаком доллара
	Символы шаблона для вывода строк:
&	Указание на вывод всего строкового выражения
!	Указание на вывод лишь первого символа строкового выражения
\\	Указывает на вывод строки определенной длины. Количество выведенных символов определяется интервалом между двумя знаками "косая черта". Причем, считаются как ограничители (знак "косая черта"), так и пробелы между ними.
	Специальные символы внутри шаблона:
Текст	Реализуется возможность вводить текст в шаблон
_	Вывод следующего символа в непосредственном виде

Ввод/вывод

Форматированный вывод на экран

```
'Пример вывода с помощью оператора PRINT USING
CLS
PRINT "Сначала несколько арифметических шаблонов
PRINT
PRINT "Вывод с шаблоном:"
PRINT USING "#####" 112
PRINT USING "#####" 10
PRINT USING "#####" 1123
PRINT "Вывод без шаблона:" 'Для сравнения
PRINT 34
PRINT 123
PRINT 1123
PRINT
PRINT "Вывод с шаблоном:"
PRINT USING "###.##"; 34
PRINT USING "#*#.*#"; 45.2
PRINT USING "###.*#"; 123.23
PRINT "Вывод без шаблона:" 'Для сравнения
PRINT 23 "Для сравнения
PRINT 345.34
```

Ввод/вывод

Позиционирование курсора

Для перемещения курсора в требуемую позицию экрана, как правило, применяют оператор *LOCATE*. Это очень удобно, например, для создания сложных форм вывода на экран.

LOCATE [Строка] [, [Столбец][,[Курсор] [,Старт[,[Стоп]]]]

Строка - Строка экрана (1-25), на которую должен переместиться курсор.
Столбец - Столбец экрана (1-80), к которому должен переместиться курсор.
Курсор - Указывает, в каком состоянии находится курсор при выводе. 0— выключен, 1 — включен.
Старт, Стоп - Числа от 1 до 31, которые задают высоту курсора.

Ввод/вывод

Позиционирование курсора

Пример :
После очистки первый вывод на экран должен быть в десятой строке.

```
'Вывод в десятую строку после CLS
CLS
LOCATE 10
PRINT "Вывод появляется в десятой строке экрана"
'Конец программы
```

Ввод/вывод

Вывод пробелов
Функция ***SPC*** в сочетании с оператором **PRINT** может выводить на экран пробелы. Это полезно, например, при оформлении вывода информации на экран.

SPC (Значение)

Значение - Арифметическое выражение типа INTEGER в диапазоне значений от 0 до 32767.

Пример:

```
CLS
PRINT "Text 1“; SPC(10); "Text 2"
```

Ввод/вывод

Функция табуляции

Функция ***TAB*** так же, как функция **SPC**, применяется в операторе **PRINT**. Позволяет начать вывод данных текущей строки с определенной позиции.

TAB (Столбец)

Столбец - Переменная типа **INTEGER**, задает номер столбца вывода данных текущей строки.

Ввод/вывод

Остановка программы

В основном любой алгоритм должен как-то заканчиваться, поэтому отсутствие необходимости использовать оператор **END** в последних версиях **QBASIC**, считается отрицательным его качеством. Но он всё же существует, и вставлять его в конец программы всё же стоит (потом пригодится).

Также если хотите прервать вашу программу в любом её месте то надо использовать оператор ***STOP.***
Он останавливает выполнение программы на том месте, где стоит.

Управляющие операторы

Оператор безусловного перехода

Оператор **GOTO** позволяет изменить последовательность выполнения шагов программы,

GOTO Метка_перехода

Метка_перехода Объявленная в программе метка. Метка объявляется произвольным именем, заканчивающимся двоеточием.

Пример 1:

```
'Оператор GOTO
CLS
PRINT "Хелло, "
GOTO weiter
PRINT "Здесь проявляется GOTO";
Welter: ‘Метка перехода
PRINT "Долли!"
'Конец программы
```

Управляющие операторы

Оператор безусловного перехода

Пример 2:

```
'Зациклившаяся программа (прервать нажатием клавиш "CTRL" + "Pause")
anfang:
PRINT 234.23
GOTO anfang 'Недосягаемый конец программы
```

Управляющие операторы

Условия

Результат сравнения может принимать только два значения: **"истина**", когда высказывание справедливо, или **"ложь"** — в противном случае. Говорят также, что истинность таких выражений либо **"true" (t)(-1),** либо **"false" (f)(0).** В качестве сравниваемых значений (сравниваемых операндов) могут участвовать строки, числа, константы, переменные, арифметические и строковые выражения. Используют следующие **операции сравнения:**

Операция	Значение
>	больше
<	меньше
>=	больше или равно
<=	меньше или равно
=	равно
<>	Не равно

Управляющие операторы

Условия

Схематически условие можно представить следующим образом:

Выражение_1 Операция_сравнения Выражение_2

Значение переменной будет зависеть от истинности или ложности определяющего выражения. Если значение выражения ложно, то переменной будет присвоено значение 0, а если истинно, то -1.

Пример:

```
'Использование логических выражений
A=2>5
B=6<=6
C=8<>9
PRINT A
PRINT B
PRINT C
END
'Конец программы
```

В результате на экране вы увидите:

```
0
-1
-1
```

Управляющие операторы

Условия

В логических выражениях можно использовать **логические операции AND**(и), **OR**(или), **NOT**(не), **XOR**(исключающее или, либо).

Таблица истинности и приоритет операций:

Выражение A	Выражение B	A AND B	A OR B	NOT A	A XOR B
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	0	0

Управляющие операторы

Условия

Логические операции. Логическое И (AND)

AND (И) коньюктивно объединяет логические условия:

Условие_1 AND Условие_2 [AND Условие_3] …

Пример:

Составить программу, ы которой входные данные должны вводится только тогда, когда zahll % больше нуля, и одновременно zahl1% меньше нуля. Рассмотрим способ, позволяющий объединить оба условия.

```
DIM zahl1%, zahl2%
CLS
DO
INPUT "Введите число 1> 0", zahl1%
INPUT "Введите число 2 < 0 ", zahl2%
LOOP UNTIL zahll %>0 AND zahl2% < 0
PRINT "Спасибо за корректный ввод"
'Конец программы
```

Только тогда, когда zahl1% > 0 и zahl2%< 0, значение всего логического выражения станет истинно, а значит, выполнение цикла закончится.

Управляющие операторы

Условия

Логические операции. Логическое ИЛИ (OR)
OR (ИЛИ) дизъюнктивно объединяет логические условия:

Условие_1 OR Условие_2 [OR Условие_3...]

Результат такого объединения только тогда ложен, когда ложны все составляющие.

Пример:

```
DIM zahl1%, zahl2%
CLS
DO
PRINT "Число 1 или число 2 должно быть больше нуля"
INPUT "Введите число 1", zahl1%
INPUT "Введите число 2", zahl2%
LOOP UNTIL zahl1% >0 OR zahl2% >0
PRINT "Спасибо за корректный ввод"
'Конец программы
```

Только тогда, когда zahl1% > 0 и zahl2%< 0, значение всего логического выражения станет истинно, а значит, выполнение цикла закончится.

Управляющие операторы

Условия

Логические операции. Логическое ИЛИ (OR)
OR (ИЛИ) дизъюнктивно объединяет логические условия:
Условие_1 OR Условие_2 [OR Условие_3...]
Результат такого объединения только тогда ложен, когда ложны все составляющие.
Пример:

```
DIM zahl1%, zahl2%
CLS
DO
PRINT "Число 1 или число 2 должно быть больше нуля"
INPUT "Введите число 1", zahl1%
INPUT "Введите число 2", zahl2%
LOOP UNTIL zahl1% >0 OR zahl2% >0
PRINT "Спасибо за корректный ввод"
'Конец программы
```

В этом примере выход из LOOP-цикла не произойдет только в том случае, если значение каждого из вводимых чисел будет меньше или равно нулю.

Управляющие операторы

Условия

Логические операции. Логическое отрицание (NOT)
NOT инвертирует (меняет на противоположное) значение логического выражения, т.е. то, что было "ложь", становится "истиной" и наоборот.

Пример:

```
DIM zahl%
CLS
INPUT "Введите число " zahl%
IF NOTzahl% > 0 THEN
PRINT "Число не больше нуля"
ELSE
PRINT "Число больше нуля"
END IF
'Конец программы
```

Управляющие операторы

Условия

Логические операции. Приоритет

В первую очередь вычисляется значение функции под операцией **NOT**, потом **AND**, и в конце **OR**, **XOR**.

Пример:

```
'Использование в логических выражениях логических операций
A=((2>5) AND (6<=6)) OR (NOT(8<>9))
PRINT A
END
'Конец программы
```

В результате на экране вы увидите:

0

Управляющие операторы

Условия

Оператор условного перехода в программе
Этот оператор позволяет изменять порядок выполнения операторов в программе в зависимости от определенных условий:

IF Условие THEN
[Оператор 1-1]
[Оператор 1-n]
[ELSE
[Оператор 2-1]
[Оператор 2-m]
END IF

Если отсутствует **ELSE**-ветвь и условие в операторе **IF** ложно, то работа программы всегда продолжается с оператора, следующего за **END IF**. В отличие от **ELSE** ключевое слово **THEN** пропускать нельзя.

Управляющие операторы

Циклические структуры

Цикл WHILE

С помощью конструкции **WHILE** … **WEND** можно реализовать выполнение ряда операторов до тех пор, пока выполняется определенное условие. Последовательность операторов, выполнение которых повторяется циклически, называется циклом.

WHILE Условие
[Оператор_1]
.
.
.
[Оператор_n]
WEND

До тех пор пока соблюдается условие, последовательно выполняются операторы от 1 до n. Ключевое слово WEND закрывает конструкцию по аналогии с командой **END IF**. Если условие цикла больше не соблюдается, то выполнение программы продолжается, начиная с оператора, следующего за **WEND**.Если условие цикла **WHILE** не выполняется с самого начала, то управление сразу же передается оператору, расположенному за **WEND**.

Управляющие операторы

Циклические структуры

Цикл DO

Конструкция **DO**...**LOOP** очень похожа на **WHILE**...**WEND**. Здесь также имеется последовательность операторов, повторное выполнение которых зависит от некоторых условий.

Вариант 1:

```
DO
[Оператор_1]
.
.
.
[Оператор_n]
[EXIT DO]
LOOP [{WHILE I UNTIL}
Условие ]
```

Вариант 2:

```
DO [{WHILE I UNTIL}
Условие]
[Оператор_1]
.
.
.
[Оператор_n]
[EXIT DO]
LOOP
```

Управляющие операторы

Циклические структуры

Цикл DO

{WHILE I UNTIL} - Ключевыми словами **WHILE** или **UNTIL** определяется способ проверки условий. При использовании **WHILE** цикл выполняется до тех пор, пока соблюдается условие (значение логического выражения истинно). И, наоборот, при использовании **UNTIL** цикл выполняется только тогда, когда условие не соблюдается (значение логического выражения ложно).
EXIT DO - Оператор **EXIT DO** преждевременно прерывает выполнение цикла.
DO...LOOP- Работа программы продолжается с оператора, следующего за **LOOP**.
В первом варианте цикл выполняется по крайней мере один раз, так как проверка условия находится в конце цикла. А во втором варианте цикл может вообще не выполняться, если соответствующее условие с самого начала не позволяет входить в него. Этот вариант очень похож на цикл WHILE...WEND. Как видно из синтаксического описания, DO...LOOP может работать без проверки условий. В этом случае из бесконечного цикла можно выйти с помощью оператора **EXIT DO.**

Управляющие операторы

Циклические структуры

Цикл FOR...NEXT

Используя оператор FOR...NEXT, можно программировать циклы, количество прохождений которых зависит от значения счетчика.

FOR Счетчик = Нач_значение TO Кон_значение [STEP Шаг]
[Оператор_1]
.
.
.
[Оператор_n]
[EXIT FOR]
NEXT Счетчик

Управляющие операторы

Циклические структуры

Цикл FOR...NEXT

Счетчик - Арифметическая переменная, которая изменяется при повторении цикла. Ее часто называют управляющей переменной цикла.
Нач.значение- Арифметическое выражение, задающее начальное значение счетчика.
Кон.значение - Арифметическое выражение, задающее конечное значение счетчика.
Шаг - Арифметическое выражение, задающее приращение счетчика при каждом прохождении цикла. Если эта опция пропущена, значение шага по умолчанию принимается равным +1
EXIT FOR- Прерывает выполнение цикла. Программа продолжает работу с оператора,
следующего за **NEXT FOR.**

Управляющие операторы

4. Оператор выбора SELECT CASE

SELECT CASE предназначен для выполнения одного из альтернативных действий, перечисленных в нем. Выбор определяется значением управляющей переменной. Начнем с представления синтаксиса:

SELECT CASE Переменная
CASE Сравнение_1
[Операторы_1]
[CASE Сравнение_2
[Операторы_2]]...
[CASE ELSE
[Операторы]]
END SELECT

Сначала выполняется **Сравнение_1.** Если результат истинен, выполняются **Операторы_1**, после чего выполнение программы продолжается с оператора, следующего за **END SELECT**. Если результат **Сравнения_1** ложен, то проверяется условие следующей ветви **CASE**. В итоге выполняются операторы той **CASE**-ветви, для которой выполняется условие сравнения. Если же ни для одной ветви результатом сравнения не является истина, то выполняются операторы ветви **CASE ELSE**.

Управляющие операторы

4. Оператор выбора SELECT CASE

CASE-сравнение в простейшем случае состоит только из одного выражения (например, из чисел или из переменных). Однако можно включать списки выражений (**выражение_1, выражение_2**,.„) или даже целые области (**выражение_1 TO выражение_2**). Далее значение переменной можно оценивать с помощью операторов сравнения, как в случае с условиями. Для этого после **CASE** применяется ключевое слово **IS**, за которым следует операция отношения и выражение.

Управляющие операторы

4. Оператор выбора SELECT CASE

Пример 1:

В этом примере вводимое число проверяется на принадлежность к определенному интервалу.

```
‘ SELECT ... CASE
CONST zehn%= 10
DIM zahl%
CLS
INPUT "Задать число "; zahl%
SELECT CASE zahl%
CASE 1, 2 'список значений
PRINT "Число 1 или 2"
CASE 3 TO 10 'область значений
PRINT "Число в диапазоне от 3 до 10"
CASE IS = 11 'сравнение с IS и оператор
PRINT "Число 11"
'Разумеется, выражение для сравнения может быть и посложнее
CASE IS < zehn% + 10
PRINT "Число меньше 20"
CASE ELSE
PRINT "Это все, что я знаю о числе "
END SELECT
PRINT "Конец"
```

Управляющие операторы

4. Оператор выбора SELECT CASE

Пример:

В этом примере вводимое число проверяется на принадлежность к определенному интервалу.

```
‘ SELECT ... CASE
CONST zehn%= 10
DIM zahl%
CLS
INPUT "Задать число "; zahl%
SELECT CASE zahl%
CASE 1, 2 'список значений
PRINT "Число 1 или 2"
CASE 3 TO 10 'область значений
PRINT "Число в диапазоне от 3 до 10"
CASE IS = 11 'сравнение с IS и оператор
PRINT "Число 11"
'Разумеется, выражение для сравнения может быть и посложнее
CASE IS < zehn% + 10
PRINT "Число меньше 20"
CASE ELSE
PRINT "Это все, что я знаю о числе "
END SELECT
PRINT "Конец"
```