Краткая выдержка-конспект руководства к интерпретатору команд «Quick Basic» для учащихся 9-го класса.

Оператор (команда) — наименьшая автономная часть языка программирования, выполняющая определенные действия. Не возвращает результат своих действий, является только исполнителем.

Функция — наименьшая автономная часть языка программирования, выполняющая определенные действия и возвращающая результат выполнения этих действий.

> Оператор можно сравнить с подчиненным, который при поручении какого либо дела выполняет его, но не сообщает о результате выполнения этого действия. Функция же, в любом случае, «отчитывается» о результате проделанных действий.
>
> В Quick Basic операторы, как правило, записываются без скобок. В синтаксис же функций, для которых необходима передача аргументов, скобки включены в любом случае.

Переменная — поименованная и (или) адресуемая область оперативной памяти компьютера, используемая программой.

> В любых языках программирования переменные имеют уникальные имена. В Quick Basic имена переменных начинаются с букв латинского алфавита и могут содержать цифры. Пробелы, знаки кавычек, минусы, плюсы и другие спецсимволы в имени переменной недопустимы. Помимо этого, имена переменных и операторов являются *регистронезависимыми*, т.е. вне зависимости от того заглавными буквами написано имя или строчными, оно воспринимается интерпретатором команд одинаково.
> Пример указания имени переменных:
>
> ```
> myVar
> name15
> ```

Вспомогательные символы и символьные операторы:

‘	*(апостроф, верхняя запятая)* — позволяет закомментировать строку (любой код в строке после этого символа воспринимается как пояснение или комментарий и не участвует в процессе выполнения программного кода);
=	оператор присвоения и эквивалентности (в зависимости от условий использования);
+	оператор сложения (применим как арифметическая операция, так и оператор слияния строк);
–	оператор вычитания;
*	оператор умножения;
/	*(прямая косая черта, прямой слеш)*, оператор деления;
\	*(обратная косая черта, backslash, обратный слеш)*, оператор целочисленного деления;
^	*(циркумфлекс, «крышка»)*, оператор возведения в степень;
,	*(запятая)*, разделитель параметров в некоторых операторах;
;	*(точка с запятой)*, разделитель параметров в некоторых операторах;
:	*(двоеточие)*, разделитель нескольких записей в одной строке, может использоваться в качестве разделителя строк;
.	*(точка)*, разделитель целой и дробной части десятичной дроби;
<	(открывающая угловая скобка, знак «меньше»), сравнение «меньше»;
>	(закрывающая угловая скобка, знак «больше»), сравнение «больше»;
< =	сравнение «меньше или равно»;
> =	сравнение «больше или равно»;
< >	сравнение «не равно»;
%	указатель на тип переменной INTEGER;
&	*(амперсанд)*, указатель на тип переменной LONG;
!	указатель на тип переменной SINGLE;
#	*(октоторп, «решётка», хэш, диез)*, указатель на тип переменной DOUBLE;
$	указатель на тип переменной STRING;
“	*(двойная кавычка)* ограничивает значение строковых переменных.

Типы переменных Quick Basic:

Название	Описание типа	Размер в памяти	Диапазон значений
INTEGER	Целочисленный	2 байта	–32 768 … 32 767
LONG	Целочисленный длинный	4 байта	–2 147 483 648 … 2 147 483 647
SINGLE	С плавающей запятой одинарной точности	4 байта	$-3{,}4\cdot10^{38}$ … $-1{,}4\cdot10^{-45}$— $1{,}4\cdot10^{-45}$ … $3{,}4\cdot10^{38}$
DOUBLE	С плавающей запятой двойной точности	8 байт	$-1{,}79\cdot10^{308}$…$-4{,}94\cdot10^{-324}$ — $4{,}94\cdot10^{-324}$…$1{,}79\cdot10^{308}$
STRING	Строковая переменная	10 байт и более	0…~2 млрд символов

Основные математические операторы и функции Quick Basic:

ABS ( )
название от англ. ***abs****olute* — абсолютный;
возвращает модуль числа (аргумента).

Примеры:

```
myVar = ABS (-5.8)        вернет 5.8
myVar = ABS (6.4)         вернет 6.4
```

FIX ( )
название от англ. *fix* — исправление;
возвращает целую часть числа (аргумента).

Примеры:

```
myVar = FIX (-5.8)        вернет -5
myVar = FIX (-5.3)        вернет -5
myVar = FIX (6.1)         вернет 6
myVar = FIX (6.9)         вернет 6
```

INT ( )
название от англ. ***int****eger* — целый;
округляет аргумент до ближайшего меньшего целого.

Примеры:

```
myVar = INT (-5.8)        вернет -6
myVar = INT (-5.3)        вернет -6
myVar = INT (6.4)         вернет 6
myVar = INT (6.9)         вернет 6
```

RND ( )
название от англ. ***rand****om* — произвольный;
возвращает псевдослучайное число типа SINGLE от 0 до 1 включительно

Пример:

```
myVar = RND()             вернет 0.7055475
```

SGN ( )
название от англ. ***sign*** — знак, признак;
возвращает знак числа, единицу с таким же знаком, какой имеет аргумент функции

Примеры:

```
myVar = SGN (-5.8)        вернет -1
myVar = SGN (0)           вернет 0
myVar = SGN (6.9)         вернет 1
myVar = SGN ("пять")      завершится ошибкой
```

SQR ( )
название от англ. ***sq****uare* ***r****oot* — квадратный корень;
возвращает квадратный корень из аргумента функции

Примеры:

```
myVar = SQR (2)           вернет 1.414214
myVar = SQR (0)           вернет 0
myVar = SQR (-4.9)        завершится ошибкой
```

DIV
оператор, название от англ. ***div****ide* — деление;
возвращает целую часть числа, получившегося в результате деления чисел
синтаксис следующий:

x = a ***div*** *b*
x — результат операции
a — делимое
b — делитель
Примеры:

```
myVar = 4 div 3                                  вернет 1
myVar = -5 div 2                                 вернет –2

dim a as Long
dim b, x as Long
a = 15
b = 6
x = a mod b                                      вернет 2
```

Во многих случаях, можно заменять оператор div операцией целочисленного деления:

```
x = 15 / 6                                       вернет 2,5
x = 15 \ 6                                       вернет 2
```

MOD
оператор, название от англ. ***mod****ulo* — деление по модулю;
возвращает остаток от деления, получившегося в результате деления чисел
синтаксис следующий:

x = a ***mod*** *b*
x — результат операции
a — делимое
b — делитель
Примеры:

```
myVar = 4 mod 3                                  вернет 1
myVar = -5 mod 2                                 вернет –1

dim a as Long
dim b, x as Long
a = 15
b = 6
x = a mod b                                      вернет 3
```

Основные структуры и операторы Quick Basic:

DIM
название от англ. ***dim****ension* — измерение;
инициализирует переменную или массив переменных.

Примеры:

```
DIM myVar1
```
инициализирует переменную myVar1

```
DIM myVar2 AS INTEGER
```
инициализирует переменную с именем myVar2 имеющую целочисленный тип INTEGER

```
DIM myVar3 (-5 TO 11) AS SINGLE
```
инициализирует линейный массив переменных размерностью 17, с именем myVar3 с индексами от -5 до 11 включительно, имеющий тип SINGLE

```
DIM myVar4 (-6 TO 9, 3 TO 21) AS LONG
```
инициализирует двухмерный массив переменных размерностью 16×19 с именем myVar4 с индексами от –6 до 9 включительно по «одной стороне» (по одному измерению), и с индексами от 3 до 21 включительно по «другой стороне» (по второму измерению) и имеющий тип LONG

INPUT
название от англ. *input* — вводить, вводной;
присваивает значение переменной, путем ввода с клавиатуры.

Примеры:

```
INPUT myVar
```
присваивает переменной myVar значение введенное с клавиатуры

```
INPUT "Введите значение", myVar2
```
выведет на экран текстовое приглашение Введите значение и присваивает переменной myVar2 значение введенное с клавиатуры

```
INPUT "Введите значения (3)", myVar, herVar, ourVar
```
выведет на экран текстовое приглашение Введите значения (3) и присвоит переменным myVar, herVar, ourVar значения введенное с клавиатуры через запятую.

PRINT
название от англ. *print* — печатать, издавать;
выводит на экран значения указанных переменных

Примеры:

```
myVar = 5
PRINT myVar
```
‘выведет на экран число 5.

```
myVar = "Здравствуй, Солнце!"
PRINT myVar
```
выведет на экран строку:
Здравствуй, Солнце!

```
DIM ARR(1 TO 3) AS STRING
ARR(1) = "Люблю"
ARR(2) = "Тебя"
ARR(3) = "Жизнь"
PRINT ARR(1)
PRINT ARR(1), ARR(2), ARR(3)
PRINT ARR(1); ARR(2); ARR(3)
PRINT "Я " + ARR(1); ARR(2); ARR(3)
PRINT "Я " + ARR(1) + " " + ARR(2) + " " + ARR(3)
PRINT ARR(3) + ", Я " + ARR(1) + " " + ARR(2) + "!"
```

выведет на экран следующее:

```
Люблю
Люблю      Тебя              Жизнь
ЛюблюТебяЖизнь
Я ЛюблюТебяЖизнь
Я Люблю Тебя Жизнь
Жизнь, Я Люблю Тебя!
```

END
название от англ. *end* — заканчивать;
завершает выполнение программы

Пример:

```
PRINT "начало"
PRINT "середина"
END
PRINT "конец"
```

выведет на экран:

```
начало
середина
```

RANDOMIZE
название от англ. *randomize* — перемешивать;
устанавливает основу для выбора случайного (произвольного) числа

Примеры:

Пример 1:

```
RANDOMIZE
PRINT RND
```

выведет на экран запрос:

```
Случайное число (-32768 до 32767)?
```

а затем, после ввода, например, числа 8 выведет на экран случайное число:

```
0.8887751
```

Пример 2:

```
RANDOMIZE 13
PRINT RND
```

выведет на экран:

```
0.1684687
```

Пример 3:

```
RANDOMIZE TIMER
PRINT RND
```

выведет на экран случайное число, на основе возврата функции `TIMER`, которая в свою очередь возвращает текущее значение метки времени:

```
0.1684687
```

таким образом, `RANDOMIZE TIMER` позволяет выводить случайные числа в основе которых лежит зависимость от момента времени, в который производится обращение к функции RANDOMIZE.

IF…THEN…ELSE

IF…THEN…ELSE … END IF

название от англ. *if* — если, при; *then* — тогда, затем; *else* — ещё, иначе, другой;

выполняет часть программного кода в случае выполнения или невыполнения заданных условий.

Примеры:

Пример 1:

```
myVar = 5
IF myVar = 5 THEN PRINT ”Значение равно пяти”
```

выведет на экран фразу:

```
Значение равно пяти
```

Пример 2:

```
myVar = 7
IF myVar = 5 THEN PRINT ”Значение равно пять” ELSE PRINT ”Значение не равно
пяти”
```

выведет на экран фразу:

```
Значение не равно пяти
```

Пример 3:

```
myVar = 11
IF myVar >= 10 THEN myVar = 10
PRINT ”Потолок ”, myVar
```

выведет на экран фразу:

```
Потолок          10
```

Пример 4:

```
myVar = 32
IF myVar >= 0 THEN
    myVar = myVar - 1
    PRINT ”Знач.: ”; myVar
END IF
```

выведет на экран:

```
Знач.: 31
```

Пример 5:

```
myVar = ”Апельсин”
IF myVar = ”Яблоко” THEN
    myVar = ”Зеленое ” + myVar
ELSE
    myVar = myVar + ” - не яблоко!”
END IF
PRINT myVar
```

выведет на экран фразу:

```
Апельсин - не яблоко!
```

Пример 6:

```
myVar = 7
IF myVar > 5 THEN
    IF myVar < 10 THEN
          PRINT ”Меньше десяти, но больше пяти”
    END IF
ELSE
    Print ”Меньше или равно пяти”
END IF
PRINT myVar
```

выведет на экран фразу:

```
Меньше десяти, но больше пяти
```

Пояснения: условный оператор обязательно содержит ключевые слова IF (если) и THEN (тогда) между которыми указывается логическое выражение. После слова THEN располагается блок, который будет выполнен в том случае, если условие выполняется. Так, например, условие `X>=5` будет выполнено только в том случае, если переменная `X` будет больше или равна 5. Блок кода после ключевого слова ELSE (иначе), будет выполнен только том в случае, если логическое выражение, стоящее между IF и THEN, является ложным (не истинным).

Стоит отметить, что использование ELSE не является обязательным. В случае если ELSE не указано, то *ложное* выражение внутри IF…THEN будет проигнорировано.
Весьма важным является так же и то, что существует два способа использования IF …THEN …ELSE

Если код позволяет указать условный блок операторов в одной строке, он будет выглядеть так:

IF *условие* **THEN** *исполнители*

или так, если производится обработка ложного выражения (условия):

IF *условие* **THEN** *исполнитель* **ELSE** *исполнитель*

Если же число строк кода требует бóльшего пространства, то имеет смысл записывать код так:

IF *условие* **THEN**
исполнитель
исполнитель
…
исполнитель
END IF

или так, если производится обработка ложного выражения (условия):

IF *условие* **THEN**
исполнитель
исполнитель
…
исполнитель
ELSE
исполнитель
исполнитель
…
исполнитель
END IF

Использование ключевого выражения *END IF* в данном случае являет обязательным, поскольку оно является ограничителем исполнительного блока. Оно не указывается в однострочной структуре, потому что окончание строки само по себе является ограничителем для однострочных команд.

Не путайте END и END IF.
Ключ **END** завершает выполнение программы, а ключевое выражение **END IF** лишь указывает на окончание структуры блока IF … THEN … ELSE.

FOR…TO…STEP…NEXT
название от англ. *for* — для; *to* — до; *next* — следующий, другой; *step* — шаг;
выполняет часть программного кода некоторое количество раз, до тех пор, пока значение счетчика, увеличивающееся каждый раз на значение шага, не превысит указанное значение.
Примеры:

Пример 1:
```
FOR I = 3 TO 7 STEP 1
    PRINT I
NEXT I
```
выведет на экран:
```
3
4
5
6
7
```

Пример 2:
```
FOR I = -4 TO 7 STEP 2
    PRINT I
NEXT I
```
выведет на экран:
```
-4
-2
0
2
4
6
```

Пример 3:
```
FOR I = -1 TO 3
    PRINT I
NEXT I
PRINT "после выполнения ", I
```
выведет на экран:
```
-1
0
1
2
3
после выполнения        4
```

Пример 4:
```
FOR I = 11 TO 3
    PRINT I
NEXT I
```
не выведет на экран ничего, потому что начальное значение `I` уже больше предельного, равного `3`, цикл закончится, не успев начаться.

Пример 5:
```
FOR I = 5 TO 3 STEP -1
    PRINT I
NEXT I
```
выведет на экран:
```
5
4
3
```

Пояснения: циклическая структура **FOR … NEXT** работает на основе увеличения переменной-счетчика на заданное значение шага цикла, которое указывается после ключевого слова **STEP**. Указание шага не является обязательным. По умолчанию (в случае если оно не задано) значение шага принимается равным единице. Крайне не рекомендуется присваивать переменной-счетчику значения внутри цикла. Это может привести к ситуации, в которой цикл не закончится.

DO … LOOP … UNTIL … WHILE
название от англ. *do* — делать; *loop* — петля; *until* — до…; *while* — в то время как…;
выполняет часть программного кода некоторое количество раз, до тех пор, пока условие указанное после ключевого слова WHILE выполняется или до тех пор, пока не будет выполнено условие, указанное после ключевого слова UNTIL.
Граничные условия UNTIL и WHILE могут располагаться как в начале таки в конце цикла.
Примеры:

Пример 1:

```
I = 3
DO WHILE I < 10
    PRINT I
    I = I + 1
LOOP
```

выведет на экран:

```
3
4
5
6
7
8
9
```

После выполнения кода, переменная будет иметь значение 10. Условия этого цикла проверяются в начале. Выполнение цикла будет продолжаться *до тех пор, пока* значение переменной *i* будет меньше 10 (до тех пор, пока условие выполняется).

Пример 2:

```
I = 3
DO
    PRINT I
    I = I + 1
LOOP WHILE I < 10
```

выведет на экран:

```
3
4
5
6
7
8
9
```

После выполнения кода, переменная будет иметь значение 10. Условия этого цикла проверяются в конце. Выполнение цикла будет продолжаться *до тех пор, пока* значение переменной *i* будет меньше 10 (до тех пор, пока условие выполняется).

Пример 3:

```
I = 3
DO UNTIL I >= 10
    PRINT I
    I = I + 1
LOOP
```

выведет на экран:

```
3
4
5
6
7
8
9
```

После выполнения кода, переменная будет иметь значение 10. Условия этого цикла проверяются в начале. Выполнение цикла будет продолжаться до того момента, пока значение переменной *i* не станет равно или превысит 10 (выполнение цикла будет прервано, когда условие выполнится).

Пример 4:

```
I = 3
DO
    PRINT I
    I = I + 1
LOOP UNTIL I >= 10
```

выведет на экран:

```
3
4
5
6
7
8
9
```

После выполнения кода, переменная будет иметь значение 10. Условия этого цикла проверяются в конце. Выполнение цикла будет продолжаться до того момента, пока значение переменной *i* не станет равно или превысит 10 (выполнение цикла будет прервано, когда условие выполнится).

Пояснения: два момента, которые следует уяснить:

— ограничения, накладываемые на цикл, могут находиться в начале цикла или в конце (циклы с предусловием или постусловием);

— условия после ключевого слова WHILE и после слова UNTIL отличаются тем, что в первом случае цикл продолжается пока условие истинно, а во втором случае цикл будет остановлен, как только условие станет истинным.

Очень полезным является операторы **EXIT DO** и **EXIT FOR**. В случае их выполнения происходит выход программы из тела цикла в этом месте. Важно, что для блока DO…LOOP используется EXIT DO, а для FOR NEXT используется EXIT FOR.

Пример 5:

```
I = 3
DO WHILE I < 10
    PRINT I
    I = I + 1
    IF I > 5 THEN EXIT DO
LOOP
```

выведет на экран:

```
3
4
5
```

Пример 6:

```
I = 3
FOR I = 3 TO 9
    PRINT I
    IF I > 5 THEN EXIT FOR
NEXT I
```

выведет на экран:

```
3
4
5
6
```

Основные графические операторы Quick Basic:

SCREEN *S*
название от англ. *screen* —экран; где *S* номер режима (0 — текстовый, 12 — графический 16 цветов 640×480 пикселей, 13 — 256 цветов, 320×240 пикселей);
устанавливает режим экрана в текстовый или графический режим.

Примеры:

Пример 1:

```
SCREEN 13
```

переведет экран в графический режим 256 цветов, 320×240 пикселей

Пример 2:

```
SCREEN 0
```

переведет экран в текстовый режим

CLS
название от англ. ***cl****ear* ***s****creen* — очищать экран;
очищает содержимое экрана и возвращает курсор в начальное состояние.

Пример:

```
CLS
```

PSET (*X* **,** *Y* **),** *C*
название от англ. ***p****oint* ***set*** — устанавливать точку; где *X* и *Y* координаты по горизонтали и по вертикали соответственно, *C* – индекс цвета;
закрашивает точку заданным цветом с заданными координатами и; работает только в графическом режиме. Параметр *C* является необязательным.

Примеры:

Пример 1:

```
PSET (0, 0), 1
```

нарисует точку зеленого цвета в левом верхнем углу экрана.

Пример 2:

```
X = 110
Y = 76
PSET (X, Y / 2 + 10), 15
```

нарисует точку белого цвета в координате 110 по горизонтали и 48 по вертикали.

LINE (X_1 **,** Y_1 **) — (**X_2 **,** Y_2 **),** *C*

LINE (X_1 **,** Y_1 **) — (**X_2 **,** Y_2 **),** *C* **, B**

LINE (X_1 **,** Y_1 **) — (**X_2 **,** Y_2 **),** *C* **, BF**
название от англ. *line* — линия; где X_1,Y_1,X_2,Y_2 координаты по горизонтали и по вертикали соответственно точки одного и другого конца отрезка, *C* – индекс цвета;
строит отрезок между точками с указанными координатами и указанным цветом; работает только в графическом режиме.
В случае указания параметра **B** строит прямоугольник с горизонтальными и вертикальными сторонами, ограниченными указанными координатами (параметр b от англ. ***b****lock*).
В случае указания параметра **BF** строит закрашенный прямоугольник с горизонтальными и вертикальными сторонами, ограниченными указанными координатами (параметр bf от англ. ***b****lock* ***f****ill*).
Параметр *C*, а так же ключи *B* и *BF* являются необязательными.

Примеры:

Пример 1:

```
SCREEN 12
LINE (2, 8)-(120, 14), 15
```

Переведет экран в графический режим и нарисует линию белого цвета.

Пример 2:

```
SCREEN 12
LINE (182, 140)-(13, 104), 1, B
```

Переведет экран в графический режим и нарисует прямоугольник зеленого цвета.

Пример 3:

```
SCREEN 12
LINE (182+1, 140)-(13-2, 104), 14, BF
```

Переведет экран в графический режим и нарисует заполненный прямоугольник жёлтого цвета.

Пример 4:

```
SCREEN 12
myX1 = -12
myY1 = -41
myX2 = 240
myY2 = 320
FOR I = -40 TO 40
    IF I < 0 THEN Col = 8 ELSE Col = 9
    LINE (myX1 + I*10, myY1)-(myX2, myY2 + I*20), Col
NEXT I
```

Переведет экран в графический режим и нарисует серию из 81 пересекающейся линии серого и синего цветов.

CIRCLE (*X* , *Y*), *R, C, Start, End, Aspect*
название от англ. *circle* — круг; где *X* и *Y* координаты центра окружности по горизонтали и по вертикали соответственно, *R* — радиус окружности, *C* — индекс цвета, *Start* — начало рисования дуги окружности, *End* — конец рисование дуги окружности, Aspect — отношение длины оси Y эллипса к длине оси X; параметры *C*, *Start*, *End*, *Aspect* не являются обязательными.

строит окружность или эллипс с центром в заданной точке, указанным радиусом и цветом с заданным индексом; работает только в графическом режиме.

Примеры:
Пример 1:

```
SCREEN 12
CIRCLE (182, 140), 100, 15
```

Переведет экран в графический режим и нарисует окружность белого цвета с радиусом 100 пикселей.

Пример 2:

```
SCREEN 12
FOR Asp = 0 TO 1 STEP 0.1
    CIRCLE (240, 320), 100, 6, , ,Asp
NEXT Asp
```

Переведет экран в графический режим и нарисует серию из 11 эллипсов с осью X длинной 100 и осью Y длинной от 0 до 100 пикселей.

Пояснения: в графическом режиме, так же как и в текстовом, начало координат находится в левом верхнем углу экрана, при этом положительное направление оси X вправо, а оси Y — вниз. Построение графических примитивов допустимо вне области экрана. Следует отметить, что чрезмерное смещение объектов за область экрана может привести к неправильному построению или появлению ошибок. Поэтому рекомендуется проверять аргументы графических операторов на допустимые значения и при необходимости корректировать их.

Дополнительный синтаксис:

Как уже говорилось выше, в интерпретаторе команд QuickBasic можно использовать замену стандартного синтаксиса упрощенными аналогами.

Так, например:

```
DIM a&
```

равносильно:

```
DIM a AS LONG
```

или:

```
DIM MyVar!
```

равносильно:

```
DIM MyVar AS SINGLE
```

Таким образом, ключевые сочетания AS INTEGER, AS LONG, AS SINGLE, AS DOUBLE и AS STRING можно заменять на символы `% & ! # $` соответственно.

Так же, несколько странной может показаться запись нескольких действий (операторов или функций) в одной строке.

Например:

```
DIM a AS INTEGER : DIM b AS LONG : a = 14 : b = 15
```

которое равносильно записи на нескольких строках:

```
DIM a AS INTEGER
DIM b AS LONG
a = 14
b = 15
```

Такие методы записи используются не всегда, но их можно рекомендовать для пользователей, которые освоились с основами стандартного синтаксиса.