

Государственное общеобразовательное учреждение
средняя общеобразовательная школа № 463

Пособие

по языку программирования Бейсик
для обучающихся
(часть I)

Составила учитель информатики:
Григорьева Светлана Сергеевна

г.Санкт-Петербург 2011г.

Оглавление

Предисловие	3
Понятие «программа для ЭВМ».....	5
Трансляция программы.....	6
Элементы языка Basic... ..	7
Арифметические операции на языке Basic	9
Структура программы на языке Бейсик	11
Оператор ввода данных с клавиатуры INPUT	13
Оператор вывода данных на экран PRINT	14
Оператор безусловного перехода GOTO (иди к)	19
Операторы DATA, READ, RESTORE.....	20
Оператор условного перехода IF-THEN	22
Оператор условного перехода IF-THEN-ELSE.....	25
Оператор ON _GOTO	27
Циклы в алгоритмах и программах.....	28
Оператор FOR – NEXT	29
Оператор цикла WHILE-WEND.....	33
Массивы.....	34
Встроенные функции.....	40
Несколько примеров готовых программ	42

Предисловие

Трудно назвать другой алгоритмический язык, который бы имел такое распространение, как Бейсик, и в то же время имеющий, как много ревностных его почитателей, так и не менее воинствующих приверженцев иных языков. "Можно смело сказать, что именно с Бейсика часто начинается влюбленность в ЭВМ и "туземец" (т.е. непрофессионал) превращается в маститого "аборигена", бесстрашно решающего сложнейшие задачи с помощью ЭВМ" /Растрьгин Л.А. "Вычислительные машины, системы, сети..."/.

Существует предположение, что своим наименованием Бейсик обязан довольно популярному среди эмигрантов упрощенному английскому языку "Basic English", насчитывающему порядка 300-800 слов. Первоначальная версия машинного Бейсика обходилась всего 14-ю служебными словами, почти вдвое уступая словарному запасу знаменитой Элочки-людоедки.

При желании в термине "basic" можно усмотреть и некоторую претенциозность (basic - основа, база), а возможно его название образовано от сокращения английских слов - Beginner's All Symbolic Instruction Code. В переводе это значит "Многоцелевой код (язык) символических команд для начинающих.

Язык Бейсик и соответствующий транслятор для ЭВМ были разработаны сотрудниками Дармутского колледжа (США) в 1964 году.

В нашей стране интерес к Бейсику проявили независимо друг от друга несколько организаций. Первая Бейсик - система для ЭВМ была разработана сотрудниками Горьковского университета в 1969-70 г. В 1973 году на ее основе был введен в строй первый учебный терминал-класс.

Вообще, Бейсик был создан не только для развлечений и игр. Так, например, фирма IBM заключила контракт стоимостью 1.7 млрд. долларов на создание программного обеспечения боевой системы SABACS управления подводными лодками

ВМС США. А в качестве языка программирования этой системы утвержден Бейсик (электроника - 1985г.).

На ЭВМ язык программирования позволяет составлять самые различные программы: для проведения математических расчетов и построения графиков, организации игр и различных экспериментов, диалоговых справочников, обучающие, контролирующие, консультирующие и многие другие программы.

Работа на ЭВМ с языком Бейсик проходит в режиме диалога. В ходе этого диалога человек вводит свою программу, запускает ее на выполнение, редактирует, исправляет ошибки и вновь запускает программу на выполнение.

Проще всего с языком Бейсик познакомиться непосредственно работая на ЭВМ.

В настоящее время интерес к Бейсику не ослабевает. Современный "Бейсик" по своей структуре мало отличается от языков программирования высокого уровня.

Вообще, любой язык программирования представляет собой совокупность слов, специальных знаков и команд, которые способен "ПОНЯТЬ" компьютер. А поэтому прежде всего необходимо правильно писать вводимые в компьютер команды.

Понятие «программа для ЭВМ»

Известно, что ЭВМ способна быстро решать сложные задачи. Однако не все знают, что она решает задачу в строгом соответствии с заданной программой.

Программа четко указывает, какие операции и в каком порядке должна выполнить ЭВМ для решения задачи. Так, что такое «программа»? Мы уже знаем, что процесс решения задачи на ЭВМ предполагает выполнение следующих основных этапов: формулировка задачи, выбор метода решения задачи, составление алгоритма, составление программы, решение задачи на ЭВМ по заданной программе.

Исходя из этого, основное понятие указанного процесса: метод – алгоритм -программа.

Значит, выбрав метод, мы составляем последовательность действий, в результате которых произойдет решение задачи. В алгоритме мы должны предусмотреть те операции, которые ЭВМ способна выполнить. Причем сначала следует описать процесс решения на родном для программиста языке, например в виде схемы алгоритма, «отшлифовать» его и только после этого перевести на язык ЭВМ.

Средства написания алгоритма на языке ЭВМ называются языками программирования, а алгоритм, написанный на соответствующем языке - программой.

Таким образом:

Программа – описание последовательности операций, адаптированное к определенной среде аппаратных и программных средств.

Программа для ЭВМ – это последовательность команд, которые должна выполнить машина. Иными словами, программа для ЭВМ – это алгоритм, записанный на языке, понятной для ЭВМ.

Языков программирования очень много. Самый распространенный в мире алгоритмический язык - BASIC.

Программа, как правило, имеет массовый характер, т.е. рассчитана на обработку не одного, а многих наборов данных.

Прошло время, когда ЭВМ могла лишь “молча” решать задачи и выдавать пользователю колонки “сухих” цифр.

Сейчас ЭВМ может не только вычислять, но и выдавать различные сведения пользователю, советовать, обучать его, играть с ним и т.д. Тем самым ЭВМ стала собеседником человека, сотрудником или даже соперником (в игре).

Это стало возможным опять-таки лишь благодаря использованию программ, составленных человеком. Ни одна отрасль не обходится без программ.

Трансляция программы.

Программа, описанная на языке программирования имеет два представления: текстовое (при написании и изучении программы) и кодовое (при исполнении программы). Аналогично значения данных программы имеют два представления: внешнее (в тексте программы, при их вводе и выводе) и внутреннее (при исполнении программы).

При исполнении программы в оперативной памяти компьютера отводится определенное место коду этой программы. Описанные в программе данные также размещаются в оперативной памяти, при этом действуют определенные правила из размещения, которые соответствуют структуре программы и используемым языковым средствам управления размещением данных.

Программа, написанная на языке программирования называется исходной программой.

Отметим, что современные системы программирования типа ТВ, ТР позволяют очень просто для любой исходной программы (файлы типа BAS, PAS) получить загрузочную программу (файл типа EXE), для работы с которой достаточно MS DOS. Правда, при этом лишаемся возможности вносить в программу какие-либо изменения.

Транслятор – это программа, обеспечивающая перевод исходной программы пользователя на внутренний язык ЭВМ.

На практике используются трансляторы с разными принципами работы.

Первый вид транслятора обеспечивает поочередный перевод каждого оператора исходной программы на машинный язык и немедленного выполнения его. Такого типа трансляторы называются интерпретаторы.

Второй вид транслятора – компилятор. В отличие от интерпретатора он сначала осуществляет перевод всей исходной программы на машинный язык, одновременно проверяя ее правильность, результатом является загрузочная программа. Загрузочную программу можно записать на диск и использовать многократно для решения задачи, но при этом трансляция программы уже не требуется, и задача решается намного быстрее.

Интерпретаторы возможностью создания загрузочных программ не обладают.

В настоящее время для набора программ, отладки, запуска на выполнение и трансляции используется среда программирования, которая включает транслятор, экранный редактор для ввода и редактирования текстов программы, отладчик программ и многое другое. Короче, среда программирования включает весь комплекс средств, необходимых для изготовления программы и работы с ней.

Элементы языка BASIC.

Программа на языке TURBO BASIC формируется с помощью конечного набора знаков, образующих алфавит языка, и состоит из букв, десятичных и шестнадцатеричных цифр, специальных символов.

В качестве букв используются прописные и строчные буквы латинского алфавита.

Алфавит:

Цифры: 0,1,2,3,4,5,6,7,8,9

Шестнадцатеричные цифры: цифры 0...9 и буквы A...F (a...f)

Специальные символы:

+ - * / = < > [] . , () : ; ^ @ { } \$ #

Комбинации специальных символов образуют составные символы: <= >= := ..

Стандартные функции и операции.

Запись выражений на языке программирования.

Функция	Математическое определение
Sqr(x)	\sqrt{x} - корень квадратный из x
Exp(x)	e – экспоненциальная функция e
Log(x)	lnx – натуральный логарифм от x
Abs(x)	x – абсолютная величина x
Sin(x)	sinx – синус от x (x в радианах)
Cos(x)	cosx – косинус от x (x в радианах)
Tan(x)	tgx – тангенс от x (x в радианах)
Atan(x)	arctgx – арктангенс x (x в радианах)
Int(x)	Наибольшее целое, не превосходящее x (целая часть от x)
Rnd(x)	Датчик случайных чисел (случайное число в диапазоне от 0 до 1)
Pi	π - число = 3,1415926
Fix(x)	Отбрасывание дробной части от x

Выражения в программировании, в отличие от арифметического вида, записываются в одну строку.

Например:

$B^2 + 3,14C$ в программировании выглядит так, $(B^2) + (3.14 * C)$

$y = \sin a \frac{c+8}{2-d}$ в программировании, $y = \sin(a) * ((c+8)/(2-d))$.

В Бейсике все операции делятся на арифметические операции, операции отношения и логические операции

Арифметические операции на языке Basic.

Операция	Обозначение	Пример	Результат
Сложение	+	2+5	7
Вычитание	-	10-8	2
Умножение	*	3*4	12
Деление	/	15/3 15/4	5 3.75
Целочисленное деление	\	15\4	3
Возведение в степень	^	2^3	8
Остаток от деления	MOD	13 MOD 5	3

Операции отношения

Знаки операций	Операции
>	Больше
<	Меньше
=	Равно
<>	Не равно
>=	Больше или равно
<=	Меньше или равно

Логические операции

NOT	Отрицание («НЕ»)
AND	Логическое умножение («И»)
OR	Логическое сложение («ИЛИ»)

Константы и переменные. Типы величин.

Константы не изменяют своего значения до завершения работы программы

В Бейсике существуют два типа констант – числовые и символьные (литерные).

Числовые константы (или просто числа). В Бейсике используются целые и действительные числа. Знак «+» перед положительным числом можно не ставить. А для отделения целой части при записи десятичных дробей используется точка.

Отличительным признаком целой константы является знак %, который не имеет к исчислению процентов

Действительные числа могут представляться в экспоненциальной форме. Так, например, одно и тоже число 0,0007435 может быть представлено несколькими способами: .7435E-3, или .000007435E2, или 74.35E-5. здесь латинская буква «Е» имеет смысл «возвести 10 в степень». Для положительного порядка степени знак «+» может отсутствовать, для отрицательного знак «-» обязателен.

Целые	Действительные	Десятичные в форме E
5%	.05	5E-02
100%	-9.083478	-.9083478E+1
-834%	37.	3.7E1
1275%	+340.033	0.340033E3

Символьные (литерные) константы – последовательность любых отображаемых символов, заключенные в кавычки. Например “Информатика”, “Temperature 10.03.1994”. в цепочке символов могут быть любые символы языка (знаки препинания, буквы разных алфавитов и т.п.), кроме кавычек. Допускается случай, когда литерная константа не содержит ни одного символа; соответствующий текст (пусто) обозначается двумя кавычками “”.

Переменные – это величины, значения которых могут изменяться во время выполнения действий.

Переменная имеет имя (его называют идентификатор).

В Бейсике различают два вида переменных: числовые и символьные. Числовые переменные обозначаются латинской буквой или набором латинских символов, включая цифры. Символьные переменные латинской буквой и символом \$ (знаком доллара) или набором латинских букв и символом \$ (знаком доллара) Например: **MIN** – числовая переменная, **C\$** - символьная переменная. Числовая переменная – это ячейка, куда записывается только число, а символьная переменная – это ячейка, куда можно записать любой набор символов.

Структура программы на языке Бейсик.

Программа на языке программирования представляет собой текст, построенный по определенным правилам, иначе можно представить как набор команд, либо (в программировании как называют) операторов, которые записываются по определенным правилам и выполняют определенные действия.

Оператор – это конкретное указание машине, оформленное в виде математической формулы либо в виде обозначения (ключевого слова) какого-то действия, для которого требуется выполнить целый ряд машинных операций.

Правильно организованная последовательность этих операторов, т.е. действий выполнения программы и даёт необходимый результат. Каждый оператор имеет соответствующий номер, который желательно записывать кратным 10 или 5, для того чтобы при необходимости можно было вставить пропущенные операторы, которые выявляются в процессе отладки программы.

В современной версии языка программирования TURBO BASIC нумерация операторов необязательна, если только нет ссылок на операторы, но и то, вместо номера оператора можно указывать метку.

Метка – это любой набор символов, оканчивающий двоеточием. Например: **МЕТКА1:**

Одна строка программы может содержать один или несколько операторов. Операторы в строке программы отделяются друг от друга двоеточием. При вводе программы с клавиатуры вы должны заканчивать каждую строку нажатием клавиши ENTER

Приведем пример простой программы, состоящей из трех строк:

```
10 PRINT "Умножить 17 на 3"  
20 PRINT 17*3  
30 END
```

Строки 10 и 20 этой программы содержат по одному оператору PRINT и является отдельной инструкцией компьютеру. Строка 30 с оператором END указывает компьютеру, что в этой

строке программа заканчивается. Номера строк 10, 20 и 30 указывают, в каком порядке выполнять эти инструкции.

Для ввода комментария в программу используется оператор **REM**.

Например: 10 REM Начало программы

Все символы находящиеся после оператора REM компьютером не воспринимаются.

Каждый оператор записывается строго определённым образом. Как правило, он содержит имя и данные (в том или ином виде) и указывает, какую операцию и над какими величинами ЭВМ должна выполнить.

Возможны операторы, которые содержат только имя, их будем называть операторами без аргументов, например: **STOP**, **RESTORE**, **END**.

Оператор ввода данных с клавиатуры **INPUT**.

Он служит для ввода в ЭВМ с клавиатуры значений исходных величин в процессе выполнения программы и размещения их в ячейках памяти, выделенных для их величин.

Встретив команду **INPUT**, машина останавливает выполнение программы и ждёт ввода данных с клавиатуры. Набор данных заканчивается нажатием клавиши "ENTER". Оператор **INPUT** "не умеет" считать, поэтому нельзя набирать в ответ на "?" арифметическое выражение, т.е. нельзя ввести число 3/7.

Общий вид оператора:

INPUT <Подсказка> {; /,} X_1, X_2, \dots, X_n

(список вводимых величин)

где: **INPUT** – имя оператора (переводится "вести");

<Подсказка> - текстовая константа, служит для пояснения: какие величины нужно ввести и в каком порядке, записывается в кавычках. Может быть опущена;

X_i – имя переменной или элемента массива; в фигурных скобках:
; - означает, что при выполнении оператора INPUT знак вопроса выводится на экран;
, - означает, что знак вопроса не выводится на экран.

Пример:

```
10 INPUT "Введите число"; A  
20 INPUT "Введите слово"; S$
```

При запуске этого фрагмента выводится сообщение:

Введите число ?

Компьютер находится в моменте ожидания, необходимо набрать любое число и нажать клавишу **"ENTER"**.

Далее выводится сообщение: **Введите слово ?**

Компьютер находится в моменте ожидания, необходимо набрать любое слово и нажать клавишу **"ENTER"**.

Таким образом в ячейку по имени **A** запишется введенное Вами число, а в ячейку **S\$** запишется слово Вами введенное.

Оператор вывода данных на экран PRINT.

Он служит для вывода значений величин на экран дисплея в процессе выполнения программы.

Общий вид оператора:

№ строки PRINT X₁; X₂;; X_i

или

№ строки PRINT X₁, X₂, , X_i

(список вводимых величин)

где: **PRINT** - имя оператора (переводится «печать»);

X_i - элемент списка выводимых величин, может являться символьной константой, переменной или элементом массива.

Расположение выводимых значений в строке определяется разделителем элементов списка:

а) разделитель « ; »

В этом случае числовые значения дополняются в конце одним пробелом, значения текстовых величин выводятся без каких-либо дополнений.

Пример:

10 X=30

20 PRINT 100; 20; X; "Фара"; "он"; X-80; "BCE"

в результате выполнения операторов на экране получится текст:

100 20 30 Фараон-50 BCE

б) разделитель – « , ».

В этом случае экран разбивается на пять колонок по 14 позиций в каждой. Значение каждой величины печатается с начала очередной колонки.

Пример:

10 B= -10

20 D= 13564

30 PRINT B, B^2, -3, -4, D, «КОНЕЦ»

На экране:

-10	100	-3	-4	13564	КОНЕЦ
1	15	29	43	57	71

Здесь числа 1, 15, 29, 43, 57, 71 означают номера позиций строки экрана (на экран они не выводятся).

Особенности записи и работы оператора PRINT:

1. Оператор PRINT может записываться без списка величин. В этом случае он выводит на экран пустую строку (строку пробелов).
2. В одном операторе PRINT можно использовать различные разделители («;», «,»).
3. Если в конце значений оператора PRINT стоит разделитель «;» или «,», то выполнение следующего оператора PRINT, как продолжение предыдущего оператора PRINT. Если в конце значений оператора PRINT разделитель отсутствует, то выполнение следующего оператора PRINT начинается с новой строки.

Пример:

PRINT «Вас»,

PRINT «приветствует»,

PRINT «ЭВМ!»

PRINT «Здравствуйте!»

На экран получим 2 строки:

Вас приветствует ЭВМ!

Здравствуйте!

Форматированный вывод чисел на экран: PRINT USING

Знаки # указывают количество позиций, которое может занять число при печати.

Пример:

10 CLS

20 PRINT USING «#####»; -6; 387

30 PRINT USING «##.###»; 13.7833; 1.342; .34956


```

40 PRINT USING «####.##»; -175.368
50 PRINT USING «+####.##»; -68.95; 2.4; 55.6; -.9
60 PRINT USING «###»; 56342
70 PRINT USING «#. #»; 12.34

```

На экране:

Номер строки

Экран

20	-6	387			
30	13.78	1.34	0.35		
40	-175.37				
50	-68.95	+2.40	+55.60	-0.90	
60	%56342				
70	%12.3				

Объяснение программы

Число при печати выравнивается по правому краю отведённого места

30,40 числа при выводе округляются

50,60 «+» - число печатается со знаком

70,80 формат задан с ошибкой, отведённое место мало для размещения числа, знак % указывает, что число не поместилось на отведённое ему место

Вывод данных на экран с указанной позиции можно организовать с использованием оператора LOCATE

Общий вид оператора:

LOCATE НОМЕР СТРОКИ, НОМЕР СТОЛБЦА

Пример:

```
10 CLS
```

```
20 PRINT 2.135
```

```
30 PRINT «ЭВМ»
```

```

40 a= -12.34
50 b$= «БЕЙСИК»
60 PRINT a
70 PRINT b$
80 PRINT a*2-2
90 PRINT «A+5=»;a+5
100 PRINT a;a*(-1)
110 PRINT
120 PRINT a; a-8
130 PRINT b$
140 PRINT a, a+1, a+2
150 PRINT TAB(18); «РАБОТА»
160 LOCATE 13,10
170 PRINT «КОМПЬЮТЕР»
180PRINT«*****»

```

На экране:

Номер строки	Экран
20	2.135
30	ЭВМ
60	-12.34
70	БЕЙСИК
80	-26.68
90	A+5=-7.34
100	-12.34 12.34
110	
120,130	-12.34 -20.34 БЕЙСИК
140	-12.34 -11.34 -10.3
150	РАБОТА
160,170	КОМПЬЮТЕР
180	*****

Отметим роль операторов INPUT и PRINT в языке Бейсик – они позволяют организовать диалог пользователя с ЭВМ (программой).

Примеры программ с использованием операторов ввода-вывода (линейной структуры)

Пример: Вычислите площадь прямоугольника по его сторонам.

```
REM Площадь прямоугольника
INPUT "Введите сторону a", a
INPUT "Введите сторону b", b
s = a * b
PRINT "Площадь равна: ", s
END
```

Оператор безусловного перехода GOTO (иди к)

Общий вид оператора:

№ строки GOTO № строки либо (метка)

указывают на какую строку произвести переход

Пример:

```
10 X=1.5
20 IF X>10.5 THEN 70
30 Y=X^5*SIN(X)
40 PRINT X,Y
50 X=X+1
60 GOTO 20
70 END
```

Объяснение программы:

- 10 в ячейку под именем X запишется 1.5
- 20 проверка условия: если яч. X содержит значение >10.5 , то происходит переход на строку №70, если же в ячейке X значение меньше 10.5, то выполняется строка №30
- 30 при соответствующем X вычисляется выражение и результат записывается в ячейку Y
- 40 выводится на экран содержимое ячейки X и Y (вывод при зводится по зонам, т.к. разделитель между ними запятая)
- 50 содержимое ячейки X увеличивается на 1
- 60 безусловный переход на строку №20
- 70 конец программы.

Операторы DATA, READ, RESTORE.

Они служат для организации наборов данных внутри программ. Необходимость в этом возникает при использовании редко изменяющихся данных, например, справочных.

Общий вид операторов:

№ строки DATA $C_1, C_2, \dots, C_i, \dots, C_n$ (список значений)

№ строки READ $X_1, X_2, \dots, X_i, \dots, X_n$ (список переменных)

где: **DATA, READ** – имена операторов (переводятся «данные» и «читать»);

C_i - константа (числовая или текстовая), где $i=1,2,3,\dots,n$

X_i – имя переменной или элемента массива. где $i=1,2,3,\dots,n$

Работа операторов : последовательно каждой величине X_i оператора READ присваивается значение C_i из оператора DATA, т.е. действие операторов равносильно выполнению следующих операций: $X_1 = C_1$; $X_2 = C_2$;; $X_n = C_n$

Оператор RESTORE.

Это оператор без аргументов. Он служит для повторного чтения одних и тех данных операторами READ. После выполнения оператора RESTORE оператор READ начинает чтение данных с первого элемента оператора DATA

Пример 1:

```
10 DATA 5,17,30,50
20 READ A,B
30 RESTORE
40 READ C,D
```

После выполнения операторов переменные A,B,C,D получают 5, 17, 5, 17

Пример2:

```
10 DATA 100,350,"ЭВМ",56.5
20 READ X,Y,A$,Z
30 C=(X^2*Y^3)/Z^2
40 PRINT A$
50 PRINT "C=";C
60 RESTORE
70 READ L,K
80 N=L^5*K^5
90 PRINT "N=";N
100 END
```

Объяснение программы:

10 организуется блок данных
20 соответствующие данные считываются в ячейки под именем указанным в операторе
30 вычисляется выражение, результат которого записывается в ячейку С
40 выводится на экран содержимое ячейки А\$
50 выводится на экран С = и содержимое яч. С
60 оператор, который восстанавливает данные т.е. данные можно снова считывать сначала в той же последовательности в другие ячейки.
70 считываются данные в ячейку L и K, т.е. L – значение 100, K – значение 350
80 вычисляется выражение, результат которого запишется в ячейку N
90 выводится на экран N = и содержимое яч. N
100 конец программы

При программировании разветвления на языке Бейсик

Оператор условного перехода IF-THEN

Общий вид операторов

IF <УСЛОВИЕ> THEN <ОПЕРАТОР>

или

**IF <УСЛОВИЕ> THEN
<ОПЕРАТОР 1>**

...

<ОПЕРАТОР N>

END IF

Если условие справедливо, то программа выполняет тот оператор, который стоит после ключевого слова THEN (или серию операторов от ключевого слова THEN до END IF), и дальше руководствуется обычным порядком действий.

Если условие не справедливо, то оператор, стоящий после THEN (или серия операторов от THEN до END IF) не выполняется, и программа сразу переходит к обычному порядку действий.

Конструкция IF...THEN позволяет в зависимости от справедливости условия либо выполнить оператор, либо пропустить этот оператор. Конструкция IF...THEN...END IF позволяет в зависимости от справедливости условия либо выполнить группу операторов, либо пропустить эту группу операторов.

Следует иметь в виду, что по схеме: отдельные ветви разветвления расположены параллельно, а в программе они должны следовать друг за другом. Причем, если одна ветвь выполнена, то другую выполнять не нужно. Этот обход второй ветви (действия 2) осуществляется оператором GOTO N (в современных версиях языка БЕЙСИК оператор GOTO можно упускать, достаточно указать номер оператора, куда необходимо перейти), который обеспечивает переход к общей части программы после выполнения первой ветви (действие 1).

Сразу за оператором IF-THEN должны располагаться операторы действия 2. Эта часть программы начинает выполняться, если условие в операторе IF имеет значение нет, т.е. структура разветвления имеет следующий программный эквивалент:

- 1 IF условие истина THEN иди на 4
- 2 Оператор действия 2
- 3 GOTO 5
- 4 Операторы действия 1
- 5 Операторы общей части программы

Условия - еще один тип логических выражений. В них используются следующие операторы сравнения:

=	равно
<>	не равно
>	больше
<	меньше
>=	больше или равно
<=	меньше или равно

Справа и слева от знака сравнения должны стоять величины, относящиеся к одному типу. В результате сравнения получается логическая величина, имеющее значение ИСТИНА (TRUE) или ЛОЖЬ (FALSE).

Пример1:

10 F=0

20 PRINT 2+F,F+F

30 F=F+2

40 IF F<=16 THEN 20

50 STOP

Объяснение программы:

10 строка - в ячейку по имени F запишется 0

20 строка - вывод на экран значений 2 и 0, выводятся по зонам

30 строка – в яч. F предыдущее значение увеличивается на 2

40 строка – если значение в ячейке F меньше, либо равно 16, то переход происходит на строку 20, иначе остановка, т.е. в программе будет выполняться до тех пор пока F не станет больше 16. В процессе выполнения программы будут выводиться на экран два значения и значение ячейки F будет изменяться на 2.

Пример2: Проверить, равно ли введенное число некоторому значению, и в случае равенства выдать на экран сообщение о равенстве чисел.


```
REM сравнить число с каким-то значением
INPUT "Введите а", а
IF а=7 THEN PRINT "Числа равны"
END
```

После запуска программы проверяется равно ли введенное значение семи или нет. Если равно, то на экран выводится сообщение **Числа равны.**

Оператор условного перехода IF-THEN-ELSE

(**IF** - если, **THEN** - тогда, **ELSE** – иначе)

Предыдущие конструкции позволяли обойти или выполнить серию оператор в зависимости от справедливости условия. Это еще не было ветвлением. Чтобы вычисления могли разветвляться по нескольким направлениям, служит конструкция :

IF...THEN...ELSE... или IF...THEN...ELSE...END IF

Общий вид оператора

IF <УСЛОВИЕ> THEN <ОПЕРАТОР 1>

ELSE<ОПЕРАТОР 2>

или

IF <УСЛОВИЕ> THEN

<ОПЕРАТОРЫ 1>

ELSE

<ОПЕРАТОРЫ 2>

END IF

Если условие справедливо (ИСТИНА), то выполняются <операторы 1> (стоящие между THEN и ELSE), а <операторы 2> (стоящие между ELSE и END IF) будут пропущены.

Если условие не справедливо (ЛОЖЬ), то <операторы 1> игнорируются и выполняются <операторы 2>.

Пример: Определить большее из двух чисел, вывести его на экран, затем - увеличить его в двое и вывести результат на экран.

REM определить большее из двух чисел...

INPUT "Введите a", a

INPUT "Введите b", b

IF a>b THEN

PRINT "Большее число: ", a

c=2*a

ELSE

PRINT "Большее число: ", b

c=2*b

END IF

PRINT "результат: ", c

END

Сначала программа запрашивает оба числа, затем проверяет условие $a > b$. Если условие верно, то на экран выводится число a, затем это число удваивается, иначе на экран выводится число b, затем число b удваивается. В завершении на экран выводится удвоенное значение большего числа.

Обратите внимание: программа имеет один недостаток - не учитывается тот случай, когда введенные числа равны. Исправим это, использовав вложение одного условия в другое.

REM определить большее из двух чисел...

INPUT "Введите a", a

INPUT "Введите b", b

IF a=b THEN

PRINT "Числа равны"

c=2*a

ELSE

IF a>b THEN

PRINT "Большее число: ", a

c=2*a

```

ELSE
  PRINT "Большее число: ", b
  c=2*b
END IF
END IF
PRINT "результат: ", c
END

```

В этой программе два условных оператора, первым проверяется условие равенства чисел и, в случае его выполнения, будет выдано сообщение о равенстве чисел, если числа не равны, то проверяется второе условие...

Оператор ON _GOTO

(переключение на указанные ветви)

Общий вид оператора:

ON переменная GOTO N1,N2,N3,....


УКАЗАТЕЛЬ

При выполнении этого оператора предварительно вводится числовая переменная, если ее значение равно 1, осуществляется переход к строке с номером N1; если 2 - то к строке с номером N2 и т.д. Если значение переменной меньше 1 или больше числа указанных номеров строк, то выдается ошибка.

Пример:

Составить программу для вычисления площади одной из трех фигур - квадрата, круга или равностороннего треугольника – по значению X, интерпретируемому как сторона квадрата, радиус окружности или сторона треугольника.

Пояснение: Площадь треугольника $= x^2 \bullet \sqrt{3} / 4$

```

5 REM Программа вычисления площади квадрата, круга или
равностороннего треугольника
10 INPUT "Введите число X";X
20 INPUT "Введите указатель от 1 до 3";A
30 IF A<1 OR A>3 THEN 45
40 ON A GOTO 50,60,70
45 PRINT "Указатель введен неверно":goto 20
50 PRINT "Площадь квадрата = ";X^2:GOTO 80
60 PRINT "Площадь круга = ";3.14*X^2:GOTO 80
70 PRINT "Площадь треугольника = ";X^2*SQR(3)/4
80 END

```

Циклы в алгоритмах и программах.

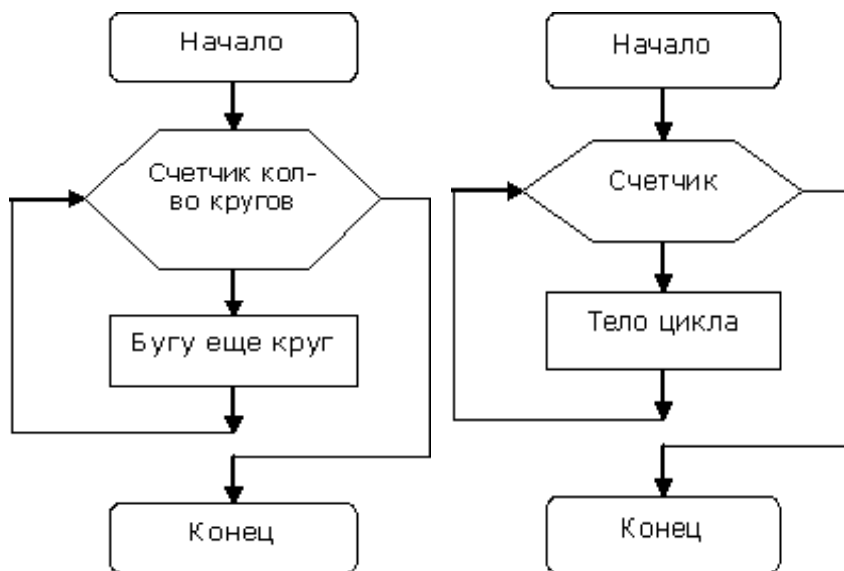
Лучшее качества компьютеров проявляются не тогда, когда они рассчитывают значения сложных выражений, а когда многократно, с незначительными изменениями, повторяют сравнительно простые операции. Даже очень простые расчеты могут поставить человека в тупик, если их надо повторить тысячи раз, а повторять операции миллионы раз человек совершенно не способен.

С необходимостью повторяющихся вычислений программисты сталкиваются постоянно. Например, если надо подсчитать, сколько раз буква "о" встречается в тексте необходимо перебрать все буквы. При всей простоте этой программы исполнить ее человеку очень трудно, а для компьютера это задача на несколько секунд.

Циклический алгоритм - описание действий, которые должны повторяться указанное число раз или пока не выполнено заданное условие.

Перечень повторяющихся действий называют телом цикла.

Например, на уроке физкультуры вы должны пробежать некоторое количество кругов вокруг стадиона.



Такие циклы называются - циклы со счетчиком.

Оператор FOR – NEXT

Общий вид:

FOR пер=Кнач TO Ккон STEP ΔK

P

NEXT пер

- заголовок цикла.

- тело цикла.

- конец цикла.

FOR – имя оператора (для)

пер – переменная цикла

Кнач, Ккон, ΔK – начальное, конечное значения и шаг изменения величины K, могут быть представлены в виде арифметического выражения.

TO, STEP – операторы языка Бейсик (ОТ, ШАГ)

P – оператор или группа операторов

NEXT – оператор языка Бейсик (СЛЕДУЮЩИЙ).

Пример:

```
10 FOR X=1 TO 11 STEP 2.5
20 Y= SIN(X)
30 PRINT X,Y
40 NEXT X
50 END
```

Объяснение программы

10 Оператор цикла, где указано, что X в процессе работы программы будет изменяться от 1 до 11 с шагом 2.5, т.е. первое значение X=1, второе X=3.5 и т.д.

20 Арифметическое выражение, где в ячейку Y запишется результат вычисления, где X=1.

30 Вывод на экран содержимого ячейки X и Y, вывод производится по зонам, т.к. разделить между переменными запятая

40 Оператор, который производит переход на следующий шаг выполнения цикла, т.е. X увеличиться на 2.5

Таким образом: в результате работы программы на экран будет выводиться:

1	0.84
3,5	-0.35
6	-0.27
8,5	0.79
11	-0.99

Пример1: Вывести на экран все числа от 1 до 100. Для этого можно было бы написать следующую программу:

```
REM Вывод чисел от 1 до 100
PRINT 1
PRINT 2
PRINT 3
PRINT 4
PRINT 5
...
PRINT 98
PRINT 99
PRINT 100
END
```

Всего каких-то 102 строчки. Хотя эту же программу можно написать намного короче:

```
REM Вывод чисел от 1 до 100
FOR I=1 TO 100
PRINT I
NEXT
END
```

Немного исправив программу можно сделать, чтобы она выводила все числа от *a* до *b*.

```
REM Вывод чисел от a до b
a=55
b=107
FOR I=a TO b
PRINT I
NEXT
END
```

В этом случае счетчик при первом прохождении цикла принимает значение переменной **a**, после чего выполняются операторы до ключевого слова **NEXT**. После этого счетчик увеличивается на единицу и сравнивается со значение переменной **b**, если счетчик меньше, то цикл выполняется еще.

Легко сделать чтобы программа выводила числа в обратном порядке. Для этого шаг цикла должен быть равен -1 (минус один). В этом случае значение счетчика каждый раз после прохождения цикла будет уменьшено на единицу.

```
REM Вывод чисел от b до a
a=55
b=107
FOR I=b TO a STEP -1
PRINT I
NEXT
END
```

Пример2: Вычислить сумму двухзначных натуральных чисел.

```
REM Вычислить сумму двухзначных натуральных чисел
FOR I=10 TO 99
S=S+I
NEXT
PRINT "Результат = ",S
END
```

Программа перебирает числа от 10 до 99 каждый раз выполняя действия $s=s+I$. С точки зрения математики это совершенно бессмысленная запись, но рассмотрим её внимательней.

Процесс решения вычислительной задачи - это процесс последовательного изменения значений переменных. В итоге - в определенных переменных получается результат. Переменная получает определенное значение в результате присваивания. Вы помните, что присваивание - это занесение в ячейку, отведенную под переменную, определенного значения в результате выполнения команды.

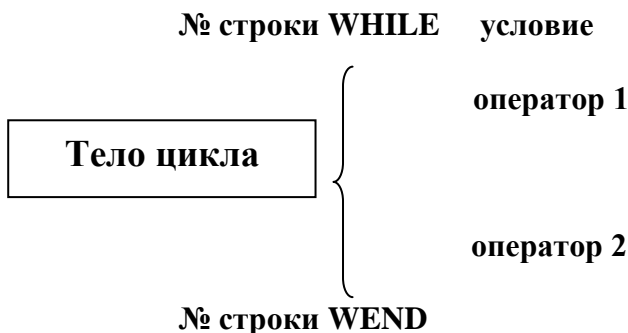
В результате операции, **a=5** переменная a получает значение 5. В результате операции **c=a+b** переменная c получает значение равное сумме значений переменной a и b. В результате

операции $S=S+I$ переменная **S** получает значение равное сумме предыдущего значения переменной **S** и значения переменной, т.е., если до операции присваивания значение **S** было равно **5**, а переменной **I** равно **3**, то после операции значение переменной **S** будет равно **8** ($5+3$, старое значение **S** + значение **I**).

Значит после выполнения нашей программы в переменной **S** будет храниться сумма всех двузначных чисел от **10** до **99**.

Оператор цикла WHILE-WEND

Общий вид оператора:



WHILE – перевод на русский **пока**

WEND – конец работы оператора **WHILE**

Объяснение работы оператора:

Пока условие истина, выполняются операторы находящиеся в теле цикла, как только условие неистинно выполнение программы передаётся оператору следующему после **WEND**.

Пример:

Используя операторы WHILE-WEND, составить программу на языке BASIC, которая вычисляет сумму числового ряда вида: $S=1+1/2+1/4...1/2^n$

Вычисление суммы производить до тех пор, пока значение члена ряда больше 0,01. Результат вывести на экран в виде: Сумма числового ряда $S = \dots$

```
10 n=0: S=0
20 WHILE 1/2^n>0.01
30 S=S+1/2^n
40 n=n+1
50 WEND
60 PRINT "СУММА РЯДА =" ; S
70 END
```

Массивы.

Переменные, которые мы использовали при составлении программ до сих пор, называются простыми переменными. Каждая простая переменная имеет имя и под каждую такую переменную выделяется ячейка памяти, обращение к которой осуществляется по этому имени. Использование только простых переменных затрудняет или делает вообще невозможным решение многих важных задач. Такую возможность представляет использование массивов.

Массивом называется упорядоченная последовательность величин, обозначаемая одним именем.

Упорядоченность заключается в том, что элементы массива располагаются в последовательных ячейках памяти.

При работе с большим числом данных одного типа очень удобно использовать массивы.

Итак, что же такое массивы...

Массив, это разновидность переменной. Он дает возможность хранить сколько угодно значений под одним и тем же именем. К каждому конкретному значению массива, необходимо обращаться через числовой индекс.

Массив - это набор переменных, имеющих одинаковое имя (идентификатор), но различающихся порядковыми номерами (индексами).

Массив, вспоминая аналогию с ящиками, можно представить себе, как несколько одинаковых ящиков, скрепленных вместе. На конструкцию, как целое, повешен один ярлык с именем массива (например А). Все ящики в одной такой конструкции перенумерованы, начиная с 1. Ящики - это элементы массива. Номер ящика - значение индекса элемента массива. Чтобы получить доступ к нужному ящику, нужно указать имя массива и его индекс. Индекс записывается в круглых скобках после имени массива. Если нужно присвоить, например, значение 5 третьему элементу массива А, нужно написать $A(3)=5$. При выполнении этого оператора будет найден массив А, отсчитан третий ящик (ячейка памяти) с начала и в него положено значение 5. Также можно поступить со всеми ящиками (яч. памяти) изменяя значение индекса. Имя массива образуется также, как имя переменной (буква или буква и цифра).

В Бейсике массивы могут быть одномерные (вектор) и двумерные (матрица). Хотя возможны и многомерные массивы (до 8 измерений), но мы их рассматривать не будем, так как они не имеют готового аналога в реальном мире.

Одномерные массивы.

Обычно массивы применяют для группировки переменных, имеющих много общих свойств. Например, если в классе 30 учеников, то имя каждого ученика можно было бы сохранить в отдельной строковой переменной: name1, name2, ... Но вводить

30 новых переменных крайне неудобно. Можно сделать проще: объявить один массив name(), имеющий 30 элементов. В скобках проставляется индекс, когда надо обратиться к какому-то конкретному элементу.

Отсчет элементов массива во многих языках начинается с нуля. Поэтому имя первого (по классному журналу) ученика будет храниться в переменной name(0), второго - в переменной name(1), а последнего (тридцатого) - в переменной name(29).

Для того чтобы использовать массив его надо сначала объявить в программе. Для этого используют оператор DIM. По умолчанию (если нет оператора DIM в программе) считается заданным массив из 10 элементов.

Пример:

DIM mas1(10)

mas1

5	2	23	111	65	87	65	333	7	21
0	1	2	3	4	5	6	7	8	9

Обращение к элементам массива:

mas1(1)=5, mas1(2)=2, mas1(3)=23 и т.д.

DIM mas2 (10)

mas2

3	66	34	76	2	99	345	2	90	4
1	2	3	4	5	6	7	8	9	10

Двумерные массивы.

Двумерные массивы можно представить себе как таблицы, в ячейках которых хранятся значения элементов массива, а индексы элементов массива являются номерами строк и столбцов.

Объявляются двумерные массивы так же, как и одномерные массивы. Например, целочисленный числовой массив, со-

держащий 3 строки и 4 столбца объявляется следующим образом:

DIM tabl(3,4)

tabl

	0	1	2	3
0	2	7	8	3
1	22	1	3	34
2	5	56	9	777

С помощью двумерного массива 9х9 и двух вложенных циклов можно легко составить программу, реализующую таблицу умножения. Сомножителями будут значения индексов строк и столбцов, а их произведения будут значениями элементов массива.

Пример:

REM Таблица умножения

DIM tabum(9,9)

REM Заполнение массива - создание таблицы умножения

FOR I=1 TO 9

FOR J=1 TO 9

tabum(I, J)=I*J

NEXT J

NEXT I

REM Вывод массива на экран в виде таблицы

FOR I=1 TO 9

FOR J=1 TO 9

PRINT tabum(I,J);

NEXT J

PRINT

NEXT I

END

В Бейсике не определены операции с массивами, поэтому любая обработка массивов, а также ввод-вывод массивов осуществляется поэлементно.

Пример ввода одномерного массива:

```
10 DIM A(10)
20 FOR I=1 TO 10
30 INPUT A(I)
40 NEXT I
```

Пример вывода одномерного массива:

```
10 DIM A(10)
20 FOR I=1 TO 10
30 PRINT A(I);
40 NEXT I
```

Пример ввода двумерного массива:

```
10 DIM B(4,5)
20 FOR I=1 TO 4
30 FOR J=1 TO 5
40 INPUT B(I,J)
50 NEXT J
60 NEXT I
```

Пример вывода двумерного массива:

```
10 DIM B(4,5)
150 FOR I=1 TO 4
160 FOR J=1 TO 5
170 PRINT B( I, J);
180 NEXT J
190 PRINT
200 NEXT I
```

Пример:

```
DIM tabl(3, 4)
REM Заполнение массива
FOR I=1 TO 3
FOR J=1 TO 4
```

```

INPUT "Введите элемент массива:"; tabl(I, J)
NEXT J
NEXT I
REM Вывод массива на экран в виде таблицы
CLS
FOR I=1 TO 3
FOR J=1 TO 4
PRINT tabl(I,J);
NEXT J
PRINT
NEXT I

```

Объяснение программы:

На экран 12 раз выведутся сообщения **Введите элемент массива: ?**, на каждое сообщение необходимо ввести любое число, при нажатии на ввод каждое введенное число запишется в соответствующую ячейку **tabl(I, J)**, где I и J имеют значения, которые соответствуют номеру строки и столбца.

Например:

Начало цикла для I=1 и J=1 выводится сообщение

Введите элемент массива: ? мы набираем с клавиатуры **45** после клавиши ENTER в ячейку **tabl(1, 1)** запишется **45**, затем для I=1 и J=2 и т.д.

выглядеть на экране будет так:

Введите элемент массива: ?45

Введите элемент массива: ?3

Введите элемент массива: ?5

Введите элемент массива: ?14

Введите элемент массива: ?67

Введите элемент массива: ?1

Введите элемент массива: ?77

Введите элемент массива: ?6

Введите элемент массива: ?9
 Введите элемент массива: ?22
 Введите элемент массива: ?21
 Введите элемент массива: ?8

45 3 5 14
67 1 77 6
9 22 21 8

Встроенные функции

LEN(s\$)	Вычисляет длину строки (количество символов).
MID\$(s\$,n,k)	Выделяет из строки s\$ k символов начиная с n-го символа.
VAL(s\$)	Преобразует числовую часть начала строки в число.
STR\$(x)	Преобразует число в символьную форму.
ASC(s\$)	Вычисляет десятичный код символа.
CHR\$(x)	Преобразует код в символ.

Встроенные» функции для работы с символьными данными
Пример:

```
10 CLS
20 a$ = «текстовый редактор»
30 b$ = «блок-схема» : c$ = «1256»
40 PRINT LEN(a$), LEN(«текст»)
50 PRINT LEFT$(a$,5)+ «новый»,
60 PRINT LEFT$(b$,LEN(b$)-6)
70 PRINT RIGHT$(a$,8) + « новый»,
80 PRINT RIGHT$(b$ LEN(b$)-5)
90 PRINT MID$(c$,1,1), MID$(c$,2,3),
100 PRINT MID$(c$,3,1)
```



```

110 PRINT c$ + «22», VAL(c$) + 22
120 PRINT STR$(45) + «12»,STR$(45 + 12)
130 PRINT STR$(45) + c$,45 + VAL(c$)
140 PRINT VAL(MID$(c$,4,1)) + VAL(MID$(c$,2,1))
150 END

```

Объяснения

20 Ячейке a\$ присвоили текст -текстовый редактор
 30 Ячейке b\$ - блок-схема, ячейке c\$ - символы 1256
 40 Функция LEN определяет длину цепочки литер
 50 Функция LEFT\$(a\$,5) извлекает 5 левых литер цепочки a\$
 60 Функция LEFT\$(b\$, LEN(b\$)-6) уничтожает 6 правых литер цепочки b\$
 70 Функция RIGHT\$(a\$,8) извлекает 8 правых литер цепочки a\$
 80 Функция RIGHT\$(b\$ LEN(b\$)-5) уничтожает 5 левых литер цепочки b\$
 90,100 Функция MID\$(c\$,k,n) позволяет извлекать n литер цепочки c\$, начиная с k-ой
 110 Функция VAL(c\$) преобразует цепочку цифровых литер в число
 120 Функция STR\$(x) преобразует число в цепочку цифровых литер
 130 Первая операция конкатенация, вторая – сложение
 140 Подсчитана сумма четвёртой и второй цифр числа 1256

Номер строки

Экран

40	18	5	
50,60	текст новый	блок	
70,80	редактор новый		схема
90,100	1	256	5
110	125622	1278	
120	4512	57	
130	451256	1301	
140	8		

Несколько примеров готовых программ.

$$c = \frac{\sqrt{2ab}}{a+b}$$

1. Вычислить выражение

```
REM Вычисление выражения
INPUT "Введите a", a
INPUT "Введите b", b
c = SQR(2*a*b)/(a+b)
PRINT "Площадь равна: ", c
END
```

2. Вычислите длину окружности и площадь круга по данному радиусу.

```
REM Вычисление длины окружности и площади круга
INPUT "Введите радиус ", r
PI = 3.14
L = 2 * PI * r
S = PI * r * r
PRINT "Длина окружности равна: ", L
PRINT "Площадь равна: ", S
END
```

3. В продаже книг в книжном магазине принимает участие ЭВМ. Составить программу, которая спрашивает стоимость книг, сумму денег, внесенную покупателем, а далее определяет причитающую сдачу (если денег внесено больше), печатает "спасибо", если сдачи не требуется, или выдает сообщение о недостаточности внесения суммы. Исходные данные задать самостоятельно.

```
10 INPUT "ЗАДАЙТЕ СТОИМОСТЬ КНИГИ";ST
20 INPUT "КАКУЮ СУММУ ВЫ ПЛАТИТЕ?";SUM
```

```

30 IF SUM=ST THEN PRINT "СПАСИБО ЗА ПОКУПКУ!"
40 IF SUM>ST THEN 70
50 PRINT "У ВАС НЕ ХВАТИЛО НА ПОКУПКУ";ST-SUM
60 GOTO 80
70 PRINT "ПОЛУЧИТЕ СДАЧУ";SUM-ST;
80 PRINT "СПАСИБО ЗА ПОКУПКУ!"
90 END

```

4. Решение квадратного уравнения.

Решение квадратного уравнения зависит от значения дискриминанта.

```

REM Решение квадратного уравнения
INPUT "Введите коэффициент a: ", a
INPUT "Введите коэффициент b: ", b
INPUT "Введите коэффициент c: ", c
d=b*b-4*a*c
IF d<0 THEN
PRINT "Корней нет"
ELSE
  IF d=0 THEN
    x=-b/(2*a)
    PRINT "корень уравнения: ", x
  ELSE
    x1=(-b-SQR(d))/(2*a)
    x2=(-b+SQR(d))/(2*a)
    PRINT "корни уравнения: ", x1, x2
  END IF
END IF

```

5. В простую переменную последовательно вводятся N вещественных чисел. Вычислить максимальное значение.

```

10 REM Определение максимального значения
20 INPUT "Сколько чисел в последовательности?"; N%

```

```

30 INPUT "Первое число ="; MAX
40 FOR I = 2 TO N%
50 INPUT "Очередное число"; X
60 IF X >= MAX THEN MAX = X
70 NEXT I
80 PRINT "MAX ="; MAX
90 END

```

6. Ввести 20 чисел и подсчитать количество положительных, отрицательных и равных нулю.

```

5  K1=0: K2=0 : K3=0
10 FOR I=1 TO 20
20 PRINT «ВВЕДИТЕ»;I «-ОЕ ЧИСЛО»;
30 INPUT A(I)
40 IF A(I)>0 THEN K1=K1+1
50 IF A(I)<0 THEN K2=K2+1 ELSE K3=K3+1
60 NEXT I
70 PRINT «КОЛ-ВО ПОЛОЖИТЕЛЬНЫХ ЧИСЕЛ =»;K1
80 PRINT «КОЛ-ВО ОТРИЦАТЕЛЬНЫХ ЧИСЕЛ =»; K2
90 PRINT «КОЛ-ВО РАВНЫХ НУЛЮ=»; K3
100 END

```

7. Заменить все буквы "а" в предложении на буквы "о".

```

REM замена букв
ss$=""
INPUT "Введите предложение", s$
FOR i=1 TO LEN(s$)
IF MID$(s$,i,1)="а" THEN ss$=ss$+"о" ELSE
ss$=ss$+MID$(s$,i,1)
NEXT
PRINT "Исправленная строка: ", ss$
END

```

8. С клавиатуры вводится положительное двузначное число N. Определить, кратно ли оно 12. Предусмотреть проверку вводимых значений.

```
10 REM Проверка кратности
20 CLS
30 INPUT "N ="; N
40 IF N < 10 OR N > 99 OR N <> FIX(N) THEN GOTO 30
50 IF N MOD 12 = 0 THEN PRINT N; "кратно 12"
60 IF N MOD 12 <> 0 THEN PRINT N; "не кратно 12"
70 END
```

9. В простую переменную с клавиатуры последовательно ввести N чисел и посчитать их сумму.

```
10 REM Определение суммы чисел последовательности
20 CLS
30 INPUT "Сколько чисел в последовательности"; N%
40 S = 0
50 FOR I = 1 TO N%
60 INPUT "Очередное число ="; X
70 S = S + X
80 NEXT I
90 PRINT "Сумма равна"; S
100 END
```

10. Определить в двумерном массиве максимальный элемент и его координаты, считая, что он единственный.

```
10 INPUT "Количество строк ="; M%; IF M% < 1 THEN 10
20 INPUT "Количество столбцов ="; N%; IF N% < 1 THEN 20
30 DIM AR(M%, N%)
40 FOR I=1 TO M%
50 FOR J=1 TO N%
60 PRINT "Элемент "; I; "-й строки, "; J; "-го столбца = ";
70 INPUT AR(I, J)
```

```
80 NEXT J
90 NEXT I
100 MAX = AR(1, 1): IMAX = 1: JMAX = 1
110 FOR I=1 TO M%
120 FOR J=1 TO N%
130 IF AR(I, J) >= MAX THEN MAX = AR(I, J): IMAX = I: JMAX = J
140 NEXT J
150 NEXT I
160 PRINT "Значение максимального элемента"; MAX
170 PRINT "Он расположен в строке №"; IMAX; ", столбце №"; JMAX
180 END
```