**1. Ввод данных, циклы, разветвления, массивы, матрицы, процедуры, функции и программы в языке Basic и операции с ними**

1. BASIC - семейство высокоуровневых [языков программирования](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Был разработан в [1964 году](http://ru.wikipedia.org/wiki/1964_%D0%B3%D0%BE%D0%B4) профессорами [Дартмутского колледжа](http://ru.wikipedia.org/wiki/%D0%94%D0%B0%D1%80%D1%82%D0%BC%D1%83%D1%82%D1%81%D0%BA%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BB%D0%BB%D0%B5%D0%B4%D0%B6) [Томасом Курцем](http://ru.wikipedia.org/wiki/%D0%9A%D1%83%D1%80%D1%86,_%D0%A2%D0%BE%D0%BC%D0%B0%D1%81) и [Джоном Кемени](http://ru.wikipedia.org/wiki/%D0%9A%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8,_%D0%94%D0%B6%D0%BE%D0%BD_%D0%94%D0%B6%D0%BE%D1%80%D0%B4%D0%B6).

Алфавит языка BASIC включает:

* все латинские прописные и строчные буквы;
* арабские цифры 0-9;
* знаки арифметических операций: ^; \*; /; +; -; \; MOD.
* знаки операций отношения: =; >; <; >=; <=; <>.
* разделители и прочие символы:
* . – точка;
* , - запятая;
* ; - точка с запятой;
* : - двоеточие;
* \_ - пробел;
* ! – признак вещественной величины;
* # - признак вещественной величины двойной точности;
* % - признак целой величины;
* & - признак длинной целой величины
* $ - признак текстовой величины;
* () – круглые скобки;
* “ – кавычки;
* ‘ – апостроф.

Используются также буквы русского алфавита, но в текстовых константах или комментариях.

Представим данные в языке Basic на следующей схеме:



Переменная – это величина, значение которой может изменяться в процессе выполнения программы, обозначается именем (идентификатором). Имя переменной – это произвольный набор символов, который может содержать о 1 до 40 символов. Первый символ должен быть латинской буквой, а остальные – латинские буквы или цифры. Регистр букв не имеет значения.

Например: A; dN; GodRozd; k1%; dlina!

Константы (числовые или строковые) не меняются в ходе исполнения программы. Значения констант записываются явно, числом или строкой символов.

Например: CONST F2=60, n%=12

Для ввода данных существует оператор INPUT.

Формат: INPUT “Сообщение”, список переменных.

При выполнении команды INPUT, вычисления приостанавливаются, и на экран дисплея выводится поясняющее сообщение, если оно указано. В списке переменных через запятую указываются имена переменных, которые принимают вводимые данные.

1. Конструкцию разветвления можно записать в одном из двух форматов - блочном (в несколько строк) или линейном (в одной строке).

Линейный формат:

IF логическое выражение THEN <блок команд 1> [ELSE <блок команд 2>]

Блочный формат:

IF логическое выражение THEN

<блок команд 1>

[ELSE

<блок команд 2>]

END IF

Если логическое выражение принимает значение «Истина», выполняется блок команд 1, если логическое выражение – «Ложь», выполняется блок команд 2. Если ELSE отсутствует, то выполняется команда, стоящая за END IF.

1. В языке BASIC предусмотрено два основных способа организации циклов:

* повторение блока команд заданное количество (число) раз (цикл со счетчиком). Этот оператор используется, когда некоторый блок команд необходимо выполнить заданное число раз.

Например:

FOR счетчик=E1 TO E2 [STEP E3]

Блок команд

NEXT счетчик

Где, счетчик (параметр)- числовая переменная;

E1 - начальное значение счетчика;

E2 - конечное значение счетчика;

E3 - шаг изменения счетчика. По умолчанию этот шаг равен 1;

Блок команд - набор операторов, предназначенных для повторения.

* циклическое повторение блока команд, пока выполняется (или не выполняется) некоторое условие.

Например:

WHILE логическое выражение

Блок команд

WEND

Данный оператор позволяет выполнять блок команд до тех пор, пока значение логического выражения «Истина». После завершения цикла управление передается команде, следующей за WEND.

1. Массив - набор однотипных данных, хранящихся вместе и имеющих общее имя.

Формат объявления массива:

DIM имя\_массива (максимальный\_индекс)

Матрица, является двумерным массивом.

Формат объявления матрицы:

DIM имя\_матрицы (максимальный\_индекс\_строк, максимальный\_индекс\_столбцов)

1. Подпрограммой называется участок программы, оформленный определенным образом, к которому можно обращаться из разных точек программы любое число раз. При этом подпрограмма может решать каждый раз одну и ту же задачу с разными значениями исходных данных. Использование подпрограмм позволяет существенно уменьшить объем рабочей программы. Деление программы на подпрограммы делает программу более понятной.

Обращение к подпрограмме осуществляется с помощью команды:

GOSUB номер строки или метка

По этой команде управление передается подпрограмме - блоку команд, первая из которых помечена номером строки или меткой. Последней выполняемой командой подпрограммы является RETURN. По этой команде управление передается в основную программу на оператор, стоящий за оператором GOSUB.

Подпрограммы располагаются в конце основной программы и отделяются от основной программы оператором END.

1. Иногда при решении задач возникает необходимость вычисления одного и того же выражения при различных значениях величин, входящих в это выражение. Вот тогда и применяются функции, определяемые пользователем.

Формат:

DEF FN <имя функции>(<список формальных параметров>)=<выражение>

Обращение к функции пользователя осуществляется

FN <имя функции>(<список фактических параметров>)

Имя определяемой функции должно быть уникально: оно не может повторять имя другой процедуры, переменной или функции.

**2. Программирование на языке Basic с применением цикла по параметру**

Цикл по параметру или цикл со счётчиком используется, когда некоторый блок команд необходимо выполнить заданное число раз.

Формат

FOR счетчик=E1 TO E2 [STEP E3]

Блок команд

NEXT счетчик

Где счетчик (параметр)- числовая переменная;

E1 - начальное значение счетчика;

E2 - конечное значение счетчика;

E3 - шаг изменения счетчика. По умолчанию этот шаг равен 1;

Блок команд - набор операторов, предназначенных для повторения.

Выполнение. Если E3 >0, то цикл продолжается все время, пока счетчик < или = E2. Если E3 <0, то цикл продолжается все время, пока счетчик > или = E2.

Рассмотрим примеры.

Пример №1.

FOR I=1 TO 5

PRINT I;

NEXT I

В результате имеем: 1 2 3 4 5.

Тело цикла будет выполняться 5 раз. Шаг изменения счетчика по умолчанию равен 1.

Пример №2.

FOR I=1 TO 5 STEP 2

PRINT I;

NEXT I

В результате имеем: 1 3 5

Для цикла FOR предусмотрена возможность безусловного выхода из цикла (до его завершения). Управление передается команде, следующей за NEXT.

Формат команды выхода:

EXIT FOR

Рассмотрим пример:

FOR I=1 TO 10

INPUT F

IF F=100 THEN EXIT FOR

S=S+F

NEXT I

PRINT “Значения”

**3. Программирование на языке Basic с применением цикла с ограничением по условию и с завершением в случае выполнения заданного условия.**

В языке Basic применяются циклы с ограничением по условию и с завершением в случае выполнения заданного условия. Такие циклы существуют в двух вариантах: с предусловием, и с постусловием.

* Цикл с предусловием:

Формат:

DO UNTIL условие

Блок команд

LOOP

Выполнение. Пока условие ложно, будут выполняться команды, что находятся между DO UNTIL и LOOP.

* Цикл с постусловием:

Формат:

DO

Блок команд

LOOP UNTIL условие

Выполнение. Пока не будет выполнено условие, будут выполняться команды, что находятся между DO и LOOP UNTIL. Необходимо отметить, что блок команд между DO и LOOP UNTIL, будет выполнен как минимум один раз, поскольку условие находится в конце цикла.

Рассмотрим примеры.

Пример №1.

C = 1

DO UNTIL С >10

PRINT C;

C = C+2

LOOP

В результате имеем: 1 3 5 7 9

Пример №2.

C = 20

DO

PRINT C;

C = C+2

LOOP UNTIL С >10

В результате имеем: 20

**4. Программирование на языке Basic с применением цикла с ограничением по условию и с пропуском в случае невыполнения заданного условия**

В языке Basic применяется цикл с применением цикла с ограничением по условию и с пропуском в случае невыполнения заданного условия.

Формат:

WHILE логическое выражение

Блок команд

WEND

Данный оператор позволяет выполнять блок команд до тех пор, пока значение логического выражения «Истина». После завершения цикла управление передается команде, следующей за WEND.

Рассмотрим пример:

WHILE PAS$ <> “Петуния”

INPUT “Введите пароль”;PAS$

WEND

PRINT “Добро пожаловать”

Этот цикл будет работать до тех пор, пока не будет введен правильный пароль.

Отметим, что в блоке команд должен присутствовать оператор, влияющий на значение логического выражения, иначе будет происходить зацикливание.

Например:

I=0

WHILE I<=10

I=I+1 'оператор, который влияет на значение логического выражения

WEND

PRINT I

Представим пример зацикливания:

I=0

WHILE I<=10

INPUT A

S=S+A

WEND

PRINT S

В блоке команд нет оператора, который бы изменял значение I.

Отметим, что если при первой проверке условия окажется, что значение логического выражения – «Ложь», то блок команд не будет выполнен ни разу.

**5. Программирование на языке Basic вложенных циклов**

Можно организовать весьма сложные повторения, используя циклы внутри циклов, так называемые вложенные циклы.

Например:

FOR

…

FOR ---------

тело внутреннего цикла

NEXT --------

…

NEXT

Рассмотрим примеры.

Пример №1. Напечатать таблицу умножения до 12 в виде:

1\*2=2

2\*2=4 и т.д.

FOR I=2 TO 12

FOR J=1 TO 12

PRINT J;”\*”;I;”=”;J\*I

NEXT J

NEXT I

Пример №2. Напечатать таблицу умножения до 5 в виде таблицы:

1 2 3 4 5

2 4 6 8 10 и т.д.

FOR I=2 TO 12

FOR J=1 TO 12

PRINT J\*I;

NEXT J

PRINT

NEXT I

Пример №3. Напечатать все четырехзначные натуральные числа в десятичной записи которых нет двух одинаковых цифр.

FOR T=1 TO 9

FOR S=0 TO 9

FOR D=0 TO 9

FOR E=0 TO 9

IF T<>S AND T<>D AND T<>E AND S<>D AND S<>E AND D<>E THEN

M=T\*1000+S\*100+D\*10+E

PRINT M,

ENDIF

NEXT E

NEXT D

NEXT S

NEXT T

**6. Программирование на языке Basic простого разветвления одной строкой**

В языке Basic используют простое разветвление одной строкой. Конструкция такого разветвления представлена ниже:

IF логическое выражение THEN <блок команд 1> [ELSE <блок команд 2>]

Выполнение. Если логическое выражение принимает значение «Истина», выполняется блок команд 1, если логическое выражение – «Ложь», выполняется блок команд 2.

Рассмотрим примеры.

Пример №1:

INPUT “Сколько будет 2x2”,X

IF X=4 THEN PRINT “Правильно” ELSE PRINT “Не правильно”

END

Пример №2:

Вычислить значение функции Y:



INPUT “Введите значение аргумента Х”,X

IF X<=1 THEN Y=(X-1)/(2\*X^2+3) ELSE Y=1.05\*(X-1)^2

PRINT “При значении Х=”;X;“функция Y=”;Y

END

**7. Программирование на языке Basic простого разветвления из нескольких строк**

В языке Basic используют простое разветвление из нескольких строк. Конструкция такого разветвления представлена ниже:

IF логическое выражение THEN

<блок команд 1>

[ELSE

<блок команд 2>]

END IF

Выполнение. Если логическое выражение принимает значение «Истина», выполняется блок команд 1, если логическое выражение – «Ложь», выполняется блок команд 2. Если ELSE отсутствует, то выполняется команда, стоящая за END IF.

Рассмотрим пример:

Вычислить значение функции Y:



INPUT "Введите значение аргумента Х ",X

IF X<=1 THEN

Y=(X-1)/(2\*X^2+3)

ELSE

Y=1.05\*(X-1)^2

END IF

PRINT "При значении Х=";X;" функция Y=";Y

END

**8. Программирование на языке Basic множественного разветвления**

В языке Basic возможно использование множественного разветвления.

Формат:

SELECT CASE арифм\_выражение или симв\_выражение

CASE условие 1

<блок команд 1>

CASE условие 2

<блок команд 2>

……………….

[CASE ELSE

<блок команд n>]

END SELECT

Условие оператора CASE можно указать в одном из трех форматов:

1) CASE константа 1, константа 2, …

2) CASE IS знак\_отношения константа

3) CASE константа 1 TO константа 2

Константы в условии должны быть того же типа, что и выражение в SELECT CASE.

Алгоритм множественного выбора заключается в следующем:

* вычисляется значение выражения, записанного в SELECT CASE.
* проверяется, удовлетворяет ли это значение одному из указанных в CASE условий
* если значение удовлетворяет какому-то условию, выполняется блок команд, следующий за данным CASE, иначе выполняются операторы из CASE ELSE.

Рассмотрим пример:

INPUT "Введите значение ",A

SELECT CASE A

CASE 1,5

PRINT "А равно 1 или 5"

CASE IS >5

PRINT "А больше 5"

CASE -8 TO 2.5

PRINT "А не меньше -8, но не больше 2.5"

CASE else

PRINT "Ни одно условие не выполняется"

END SELECT

После выполнения того или иного блока команд управление передается команде, следующей за END SELECT.

**9. Составление программ на языке Basic с использованием массивов данных**

Массив - набор однотипных данных, хранящихся вместе и имеющих общее имя. Возможность объединения групп элементов в массив позволяет, с одной стороны, облегчить массовую обработку данных, а с другой - упростить идентификацию элементов массива.

Каждому элементу массива может быть присвоено одно числовое или символьное значение, поэтому различаются массивы числовые и символьные. Кроме того, массивы могут быть одномерными и многомерными.

Формат объявления массива:

DIM имя\_массива (максимальный\_индекс)

Где «Имя\_массива» обычно выбирается по тем же правилам, что и имена простых переменных.

«Максимальный\_индекс» указывает максимально допустимый в программе индекс и должен быть положительным.

Примеры описания массивов:

DIM X(50) ' объявление одномерного числового массива X для 50 чисел;

DIM V$(12) 'объявление одномерного массива V для 12 символьных элементов.

Рассмотрим пример.

Задан массив К(45). Определить минимальный элемент массива и его индекс.

DIM K(45)

FOR I=1 TO 45 ' Заполнение массива

INPUT K(I)

NEXT I

MIN=A(1) ' Переменной MIN присваивается первое значение массива

FOR I=2 TO 45

IF A(I) < MIN THEN MIN=A(I):K=I

NEXT I

PRINT "Минимальный элемент=";MIN, "его индекс=";K

END

**10. Объявление и создание массивов данных на языке Basic**

Массив - набор однотипных данных, хранящихся вместе и имеющих общее имя. Возможность объединения групп элементов в массив позволяет, с одной стороны, облегчить массовую обработку данных, а с другой - упростить идентификацию элементов массива.

Каждому элементу массива может быть присвоено одно числовое или символьное значение, поэтому различаются массивы числовые и символьные. Кроме того, массивы могут быть одномерными и многомерными.

Формат объявления массива:

DIM имя\_массива (максимальный\_индекс)

Где «Имя\_массива» обычно выбирается по тем же правилам, что и имена простых переменных.

«Максимальный\_индекс» указывает максимально допустимый в программе индекс и должен быть положительным.

Представим объявление массива с переменным размером.

INPUT K

DIM A(K)

........

Рассмотрим распространённые виды ошибок:

* Если указать в программе элемент массива с индексом большим, чем значение размерности, объявленное в операторе DIM или принятое по умолчанию, то выдается сообщение об ошибке 9: Subscript out of range (выход за пределы массива);
* Если оператор DIM указан после использования имени массива или массив повторно объявлен, то появляется сообщение 10: Redimensioned array (повторное задание размерности массива).

**11. Объявление и создание многомерных массивов данных на языке Basic**

Массив представляет собой набор значений, логически связанных друг с другом, например, число учеников в каждом классе в школе.

Массив позволяет ссылаться на эти связанные значения по одному имени и используя номер, называемый индексом или подиндексом, чтобы отличить элементы друг от друга. Отдельные значения, называются элементами массива. Они представляются непрерывной последовательностью от индекса 0 до наибольшего значения индекса.

Массив может быть одномерным или многомерным. Если он имеет более одного измерения, то он называется многомерным.

Формат объявления многомерного массива:

DIM имя\_массива (максимальный\_индекс1, максимальный\_индекс2, …, максимальный\_индексn)

Где «Имя\_массива» обычно выбирается по тем же правилам, что и имена простых переменных.

«Максимальный\_индексi» указывает максимально допустимый в программе индекс i-го измерения и должен быть положительным.

Рассмотрим пример:

Задана матрица U(7,3). Отрицательные элементы матрицы заменить на 0. Измененный массив вывести на экран.

DIM U(7,3)

FOR I=1 TO 7 ' Заполнение массива отрицательными и

FOR J=1 TO 3 ' положительными числами

U(I,J)=INT(RND\*20-10)

PRINT U(I,J);

NEXT J

PRINT

NEXT I: PRINT

FOR I=1 TO 7

FOR J=1 TO 3

IF U(I,J)<0 THEN U(I,J)=0

PRINT U(I,J);

NEXT J

PRINT

NEXT I

END

**12. Действия с элементами массивов данных на языке Basic**

Под массивом понимают объединение переменных одного типа в единую структуру. Образующие массив переменные называются элементами массива. Каждый элемент массива обозначается именем массива с индексом, заключенным в круглые скобки.

A(1), A(2), A(3), ..., A(n).

Индекс определяет положение элемента массива данных относительно его начала.

Представим основные операции и порядок работы с массивами:

* объявление массива (задание максимального размера массива);
* заполнение массива;
* обработка элементов массива;
* вывод результата обработки.

Представим формат объявления одномерного массива:

DIM имя\_массива (максимальный\_индекс)

Где «Имя\_массива» обычно выбирается по тем же правилам, что и имена простых переменных.

«Максимальный\_индекс» указывает максимально допустимый в программе индекс и должен быть положительным.

Существует два способа присваивания значений элементам массива:

* cтатический, с использованием операторов DATA, READ и оператора присваивания;
* динамический, с использованием оператора INPUT и функции RND.

При работе с массивами очень удобно пользоваться оператором цикла FOR...NEXT. В качестве индекса массива используют параметр цикла.

Представим пример работы с элементами массивов.

Дан массив М(30). Элементы массива - произвольные числа. Выдать на экран значения каждого пятого элемента. Указанные элементы выдать в строку.

DIM M(30)

FOR I=1 TO 30

M(I)=INT(RND(1)\*15)

PRINT M(I);

NEXT I

PRINT

FOR I=5 TO 30 STEP 5

PRINT M(I); ' Вывод каждого пятого элемента массива

NEXT I

**13. Программирование математических операций с матрицами на языке Basic**

**14. Форматы данных и типы переменных, используемые в языке Basic**

Алфавит языка BASIC включает:

* все латинские прописные и строчные буквы;
* арабские цифры 0-9;
* знаки арифметических операций: ^; \*; /; +; -; \; MOD.
* знаки операций отношения: =; >; <; >=; <=; <>.
* разделители и прочие символы:
* . – точка;
* , - запятая;
* ; - точка с запятой;
* : - двоеточие;
* \_ - пробел;
* ! – признак вещественной величины;
* # - признак вещественной величины двойной точности;
* % - признак целой величины;
* & - признак длинной целой величины
* $ - признак текстовой величины;
* () – круглые скобки;
* “ – кавычки;
* ‘ – апостроф.

Используются также буквы русского алфавита, но в текстовых константах или комментариях.

Представим данные в языке Basic на следующей схеме:



Переменная – это величина, значение которой может изменяться в процессе выполнения программы, обозначается именем (идентификатором). Имя переменной – это произвольный набор символов, который может содержать о 1 до 40 символов. Первый символ должен быть латинской буквой, а остальные – латинские буквы или цифры. Регистр букв не имеет значения.

Имеется 4 типа переменных, которые определяются по последнему символу в ее имени:

% - целые (А1%;МАХ1%);

! - действительные (А!;NM!);

# - действительные двойной точности (DR#, X5#);

$ - символьные (BS$;P$).

При любом другом последнем символе переменная является действительной. Переменные различных типов могут иметь одинаковые имена, например, X и X$.

Константы (числовые или строковые) не меняются в ходе исполнения программы. Значения констант записываются явно, числом или строкой символов.

**15. Объявление применения и выполнение процедур в языке Basic**

Процедурой называется участок программы, оформленный определенным образом, к которому можно обращаться из разных точек программы любое число раз. При этом процедура может решать каждый раз одну и ту же задачу с разными значениями исходных данных. Использование процедур позволяет существенно уменьшить объем рабочей программы. Деление программы на процедуры делает программу более понятной.

Обращение к процедуре осуществляется с помощью команды

GOSUB номер строки или метка

По этой команде управление передается процедуре - блоку команд, первая из которых помечена номером строки или меткой. Последней выполняемой командой процедуры является RETURN. По этой команде управление передается в основную программу на оператор, стоящий за оператором GOSUB.

Если нарушено согласование операторов RETURN и GOSUB, то выводится сообщение об ошибке “RETURN without GOSUB”.

Подпрограммы располагаются в конце основной программы и отделяются от основной программы оператором END.

Рассмотрим пример.

Вычислить функцию: C = m!n!/(m-n)!

1 INPUT “Введите m>n”;m,n

IF M<=N THEN 1 ‘проверка входных данных

R=M

GOSUB A ‘вызов подпрограммы для вычисления m!

R1=F : R=N

GOSUB A ‘вызов подпрограммы для вычисления n!

R2=F : R=M-N

GOSUB A ‘вызов подпрограммы для вычисления (m-n)!

C=(R1+R2)/F : PRINT “C=”;C

END

A: F=1 ‘подпрограмма вычисления факториала

FOR I=2 TO R

F=F\*I

NEXT I

RETURN ‘оператор возврата в основную программу

**16. Объявление применения и вычисление функций в языке Basic**

Помимо стандартных функций в программе можно определить и далее использовать другие (нестандартные) функции. Они особенно полезны, если повторяющиеся действия можно выразить в виде одного оператора, результатом исполнения которого является единственное значение. Такую однострочную функцию можно определить оператором DEF FN. Общий вид оператора

DEF FNимя\_функции(список\_формальных\_параметров) = арифметическое\_выражение

где имя\_функции — заданное пользователем имя функции;

список\_формальных\_параметров – последовательность идентификаторов, разделяемых запятыми. Значения формальных параметров передаются функции при ее вызове.

арифметическое\_выражение — формула, по которой вычисляется функция.

Оператор DEF FN должен располагаться в программе до первого использования определяемой им функции.

Вычисление функции, описанной оператором DEF FN, осуществляется при обращении к ней при помощи записи указателя функции:

FNимя\_функции(список\_фактических\_параметров)

где список\_фактических\_параметров — арифметические выражения, заменяющие формальные параметры в арифметическом выражении правой части оператора DEF FN перед вычислением.

Использование функции, заданной оператором DEF FN, аналогично использованию стандартных функций. Обращение к нестандартной функции (указатель функции) можно записать в арифметическом выражении, в списке вывода оператора PRINT и т. д., вообще везде, где требуется значение этой функции.

Представим пример:

DEF FNT(X)= X\*X+A

A=5

………………..

P=FNT(3)

**17. Применение фактических и формальных параметров при использовании функций в языке Basic**

Помимо стандартных числовых функций (sin, cos и т.д.) или символьных функций пользователь может определить и свои собственные. Иногда при решении задач возникает необходимость вычисления одного и того же выражения при различных значениях величин, входящих в это выражение. Вот тогда и применяются функции, определяемые пользователем.

Формат:

DEF FN <имя функции>(<список формальных параметров>)=<выражение>

Обращение к функции пользователя осуществляется

FN <имя функции>(<список фактических параметров>)

Имя определяемой функции должно быть уникально: оно не может повторять имя другой процедуры, переменной или функции.

Формальные параметры резервируют место в памяти для фактических параметров, которые должны быть определены к моменту выполнения функции. Параметры могут быть любого типа. Представим примеры определения функции:

DEF FNKT(X)=COS(X)/SIN(X)

DEF FNP(A,B)=A/B\*100

X,A,B – формальные параметры

Между формальными и фактическими параметрами и их типами существует однозначное позиционное соответствие слева направо. Примеры обращения:

FNKT(X)

FNP(C,D)

X,C,D - фактические параметры

Имена формальных и фактических параметров могут не совпадать.

С помощью оператора DEF FN можно описать более сложную функцию, занимающую несколько строк

Формат:

DEF FN <имя функции>(<список аргументов>)

<блок команд>

FN<имя функции>=<выражение>

END DEF

В строке FN<имя функции>=записывается выражение, значение которого функция возвращает.

Рассмотрим пример.

Отрезки заданы координатами концов. Сравнить эти отрезки.

DEF FNL(x1,x2,y1,y2)=SQR((x1-x2)^2+(y1-y2)^2) ‘функция нахождения длины отрезка

INPUT “введите координаты первого отрезка”,x1,x2,y1,y2

L1=FNL(x1,x2,y1,y2)

INPUT “введите координаты второго отрезка”,x1,x2,y1,y2

L2=FNL(x1,x2,y1,y2)

IF L1>L2 THEN

? “длина первого отрезка больше”

ELSEIF L2>L1 THEN

? “длина второго отрезка больше

ELSE

? “отрезки равны”

END IF

**18. Объявление и применение локальных переменных при использовании процедур в языке Basic**

Различают локальные и глобальные переменные. Как правило, локальная переменная в процедуре прекращает свое существование сразу же после окончания выполнения процедуры.

Переменная – это величина, значение которой может изменяться в процессе выполнения программы, обозначается именем (идентификатором). Имя переменной – это произвольный набор символов, который может содержать о 1 до 40 символов. Первый символ должен быть латинской буквой, а остальные – латинские буквы или цифры. Регистр букв не имеет значения.

Имеется 4 типа переменных, которые определяются по последнему символу в ее имени:

% - целые (А1%;МАХ1%);

! - действительные (А!;NM!);

# - действительные двойной точности (DR#, X5#);

$ - символьные (BS$;P$).

При любом другом последнем символе переменная является действительной. Переменные различных типов могут иметь одинаковые имена, например, X и X$.

Статические переменные продолжают существовать, сохраняя самое последнее значение. При следующем вызове процедуры переменная не инициализируется повторно, а хранит последнее значение, которое ей было присвоено при последнем вызове процедуры. Статические переменные продолжают существовать в течение времени существования класса или модуля, в которых она определена.

STATIC <имя переменной>

**19. Способы ввода данных и значений переменных, а также вывода на печать результатов вычислений в языке Basic**

1. Переменной в Basic можно задать значение с помощью оператора присвоения.

Формат:

Имя переменной = выражение

Эта команда выполняется в два шага:

* вычисляется значение арифметического или символьного выражения, указанного в правой части команды;
* это значение присваивается переменной, записанной в левой части команды.

Например:

A = Y+X/Z

1. Также в Basic есть оператор ввода данных.

Формат:

INPUT "Сообщение", список переменных

При выполнении команды INPUT вычисления приостанавливаются и на экран дисплея выводится поясняющее сообщение. В списке переменных через запятую указываются имена переменных, которые принимают вводимые данные.

Например:

Команда: INPUT"Введите коэффициент и код режима",B,KR$

1. Также в Basic есть оператор вывода данных.

Формат:

PRINT список\_выражений

В поле операндов через запятую или через точку с запятой перечисляются выражения, значения которых надо вывести. Значения данных выводятся с текущей позиции курсора.

В случае если список\_выражений отсутствует, печатается пустая строка. Если список\_выражений задан, то на дисплее распечатываются значения выражений. Выражения в списке могут быть числовыми и (или) строковыми. Строки должны быть заключены в кавычки. Позиция каждого печатаемого элемента определяется пунктуацией, используемой для разделения элементов в списке. Компилятор Basic разделяет строку на зоны печати, каждая из которых состоит из 14 позиций. Задание запятой в списке выражений приводит к тому, что следующее значение будет напечатано в начале следующей зоны. Если же задана точка с запятой, то следующее значение выражения будет распечатано непосредственно после предыдущего значения. Задание одного или нескольких пробелов между выражениями приводит к такому же результату, что и задание точки с запятой.

Если запятая или точка с запятой стоят в конце списка выражений, то следующий оператор PRINT будет осуществлять печать значений выражений в той же строке, что и предыдущий оператор PRINT. В случае если список выражений не заканчивается запятой или точкой с запятой, то в конце каждой строки на печать будет выдаваться код возврата каретки. Если печатаемая строка длиннее, чем длина строки дисплея, то осуществляется переход на следующую физическую строку и печать продолжается.

**20. Способы включения в программу комментариев, пояснений и объявлений в языке Basic**

Любую команду в программе можно снабдить поясняющим текстом - комментарием. Комментарии облегчают понимание текста программы. Комментарии можно записать двумя способами:

* REM произвольный текст;
* ' произвольный текст;

На ход вычислений команда REM никакого влияния не оказывает.

Например:

REM вычисление функции

' вычисление функции

**21. Правила записи, сохранения, загрузки и запуска файлов программ в языке Basic**

BASIC является приложением MS-DOS, которое нормально работает под Windows. Чтобы начать работу с BASIC, надо запустить программу BASIC.EXE.

Для управления средой BASIC можно использовать:

* Меню команд и клавиши перемещения курсора, а также клавишу Enter.
* Клавиши быстрого доступа (каждой команде меню соответствует определенная комбинация клавиш).
* Мышку (установить курсор мыши на выполняемую команду и нажать левую клавишу мышки).

Опишем основные команды меню File:

* New Program - перейти к набору новой программы. Этот пункт следует выполнить, если перед тем мы работали с другой программой и окно редактора не пусто. Basic автоматически выполняет этот пункт при обычном запуске;
* Open Program... - загрузить готовую программу. Basic запросит имя программы в специальном появившемся окне. Кроме имени программы возможно придется указать диск и каталог, в котором она находится.
* Save As... - записать программу на диск с тем же или новым именем.
* Print... - напечатать текст программы на принтере.
* Exit - выйти из среды программирования Basic.

Запуск программы на выполнение осуществляется с помощью меню Run:

* Start -запуск программы с самого начала;
* Continue-продолжить выполнение прерванной программы.

**22. Порядок составления и сохранения программ на языке Basic**

Алгоритм решения практически любой задачи можно трансформировать в компьютерную программу. Такую программу в общем случае можно рассматривать как последовательность некоторых слов и специальных символов. Слова и символы образуют более сложные конструкции, которые должны формироваться по определенным правилам. В отличие от естественных, в языках программирования эти правила должны соблюдаться более строго: даже забытая запятая может привести к тому, что программа либо будет работать не корректно, либо вовсе не сможет выполняться. Набор правил для написания программ называется синтаксисом языка программирования.

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.

Включенные в программу комментарии BASIC просто игнорирует. Использование комментариев позволяет включать в программу любой текст, не нарушая ее семантики. Комментарии используются для объяснения тех или иных действий в программе. Комментарии можно записать двумя способами:

* REM произвольный текст;
* ' произвольный текст;

Алфавит языка BASIC включает:

* все латинские прописные и строчные буквы;
* арабские цифры 0-9;
* знаки арифметических операций: ^; \*; /; +; -; \; MOD.
* знаки операций отношения: =; >; <; >=; <=; <>.
* разделители и прочие символы:
* . – точка;
* , - запятая;
* ; - точка с запятой;
* : - двоеточие;
* \_ - пробел;
* ! – признак вещественной величины;
* # - признак вещественной величины двойной точности;
* % - признак целой величины;
* & - признак длинной целой величины
* $ - признак текстовой величины;
* () – круглые скобки;
* “ – кавычки;
* ‘ – апостроф.

Используются также буквы русского алфавита, но в текстовых константах или комментариях.

Программа, написанная на любом языке программирования, представляет собой набор инструкций, описывающий процесс выполнения алгоритма задачи. Инструкции записываются по определенным правилам, предусмотренным требованиями языка. Инструкции исходной программы называются командами или операторами.

Очень часто, в зависимости от некоторых условий, приходится изменять последовательность выполнения команд действия.

Команды управления ходом выполнения программы делятся на три группы:

* Команды безусловного перехода;
* Команды условного перехода;
* Команды организации циклов.

**23. Способы составления и описания алгоритмов для разработки программ инженерных расчетов.**

Алгоритм – это конечная последовательность точно определённых действий, приводящих к решению поставленной задачи.

При составлении алгоритмов следует учитывать ряд требований, выполнение которых приводит к формированию необходимых свойств:

* алгоритм должен быть однозначным, исключающим произвольность толкования любого из предписаний и заданного порядка исполнения. Это свойство алгоритма называется определённостью;
* любой алгоритм должен иметь только одно начало (один вход) и одно окончание (один выход);
* реализация процесса, предусмотренного алгоритмом, должна выдать результаты или сообщение о невозможности решения задачи, Это свойство алгоритма называется результативностью;
* способность алгоритма обеспечить решение однотипных задач с различными исходными данными. Это свойство называется массовостью;
* расчленение процесса, предусмотренного алгоритмом, на отдельные этапы, элементарные операции. Это свойство называется дискретностью.

Для строгого задания различных структур данных и алгоритмов, их обработки, требуется иметь такую систему формальных обозначений и правил, чтобы смысл всякого используемого предписания трактовался точно и однозначно. Для выполнения этого условия или требования существуют следующие способы описания алгоритмов:

* словесное описание (запись на естественном языке);
* графическое описание (блок схема);
* программное описание (тексты программ на алгоритмическом языке).

Различают следующие виды алгоритмов:

* линейный – список команд (указаний), выполняемых последовательно друг за другом;
* разветвляющийся – алгоритм, содержащий хотя бы одну проверку условия, в результате которой обеспечивается переход на один из возможных вариантов решения;
* циклический – алгоритм, предусматривающий многократное повторение одной и той же последовательности действий. Количество повторений обусловливается исходными данными или условием задачи.

Любая алгоритмическая конструкция может содержать в себе другую конструкцию того же или иного вида, т. е. алгоритмические конструкции могут быть вложенными.