Глава III. Программирование разветвляющихся и циклических алгоритмов

> … и если ничто уже не помогает — посмотри инструкцию для пользователя.
>
> *—Из завещания неизвестного программиста*

Линейные алгоритмы и соответствующие им программы редко встречаются в практике решения на компьютере реальных задач. Ведь неизменная последовательность действий, как правило, не может соответствовать изменяющимся обстоятельствам разнообразных жизненных или учебных ситуаций. Чаще всего порядок выполнения действий (ход алгоритмического процесса) зависит от определённых условий, поэтому возникает необходимость использовать операторы управления — оператор безусловной передачи управления, оператор условной передачи управления, оператор цикла.

Изучение этих операторов позволит ответить на вопрос: может ли *статический* объект текст программы — быть построен таким образом, чтобы дать ясное представление о динамической ситуации — её выполнении?

III.1. Оператор безусловной передачи управления GOTO

> …квалификация программистов является убывающей функцией от плотности предложений GOTO в создаваемых ими программах.
>
> *—Э.Дейкстра*

Выполнение операторов в программе осуществляется в порядке их естественного следования. Оператор GOTO («to go to» — «перейти») позволяет нарушить этот порядок.

Оператор безусловной передачи управления имеет следующую структуру:

```
GOTO n
```
,

где:

- GOTO («to go to» — «перейти») — служебное слово;
- n — любой существующий номер строки в программе, 0≤n≤65529.

Оператор GOTO передаёт управление программной строке с указанным номером n. Если в программе нет строки с этим номером, то выдаётся сообщение:

«Undefined line number»
(*«Номер строки не определён»*).

Управление передаётся заданной строке, даже если в ней нет выполняемого оператора; например, в строке может находиться оператор REM. Поскольку, однако, оператор REM не исполняется, лучше передавать управление следующему за ним исполняемому оператору. В этом случае программа будет выполняться правильно и после удаления программной строки, содержащей комментарии.

Выполнение оператора GOTO n в режиме прямого выполнения команд аналогично выполнению команды RUN n , но в отличие от этой команды оператор GOTO n начинает выполнение программы с первого оператора указанной строки, не присваивая начальных значений переменным и не меняя ранее присвоенных значений. С этой точки зрения оператор GOTO n в режиме прямого выполнения команд полезен при *отладке и тестировании*. В остальных случаях использовать его в этом режиме не рекомендуется, поскольку существует большая вероятность задать

неправильный номер строки или непреднамеренно изменить значение переменной так, что фактически изменится порядок работы программы, что приведёт к появлению многочисленных ошибок!

Оператор GOTO — одна из наиболее интуитивно ясных и в то же время одна из наиболее опасных конструкций языков программирования. Оператором GOTO следует пользоваться как можно *реже*, так как он делает программу трудно читаемой для программиста — приходится особо следить за тем, в каком порядке выполняются строки программы. Отметим, что в школьный алгоритмический язык эта конструкция не включена вообще!

Примеры:

- 1)

```
20 GOTO 20 'Зацикливание обеспечено! Стандартный приём при работе с графикой.
```

- 2)

```
NEW
Ok
5'Пример плохой,трудно читаемой программы!
10 INPUT X
20 GOTO 40
30 PRINT X+Y:GOTO 60
40 INPUT Y
50 GOTO 30
60 END
```

Нетрудно видеть, что приведённая программа эквивалентна программе, состоящей из единственной программной строки:

```
10 INPUT X,Y:PRINT X+Y:END
```

- 3)

```
  NEW
  Ok
  10 PRINT "2*2=4":GOTO 10
```

Эта программа никогда не закончит свою работу, т.к.после вывода текста «2*2=4» на экран оператор GOTO 10 возвращает к строке 10, которая опять выводит «2*2=4» и т.д. Пользователь, разумеется, может вмешаться и остановить такую «вечно» работающую программу(её называют *зациклившейся*) — для этого одновременно нажимают клавиши CTRL и STOP (напомним, что такое нажатие обозначается CTRL+STOP).

В случае, если оператор записывается двумя служебными словами GO TO n, то он выполняется аналогично — этот формат сохранен для совместимости с другими языками программирования.

Однако, использовать «двухсловную» форму GOTO не рекомендуется, так как она требует большого объёма памяти для хранения. Достигаемая экономия на первый взгляд несущественна, однако рано или поздно многие программисты попадают в ситуацию, когда «нескольких бит не хватает!»

В статье Бема и Якопини [40] было показано, что любой алгоритм может быть описан на языке программирования, использующем *только три* управляющие структуры: следование, цикл и ветвление. Возможность представления любых алгоритмов с помощью вложенных друг в друга структур следования, цикла и ветвления составляют основу метода структурного программирования.

В последующих разделах мы расскажем об операторах IF…THEN…ELSE и FOR…NEXT, которые являются простейшими эквивалентами управляющих конструкций Бема и Якопини.

III.2. Оператор условной передачи управления IF

ЕСЛИ нельзя, но очень хочется, ТО можно!

—Евг.Сазонов, программовед и языколюб

Сказочные герои, как известно, по дороге за тридевять земель в тридесятое царство встречали на перекрёстке

камень с надписью — альтернативой. У программиста при написании программы возникает потребность запрограммировать альтернативу, т.е. предусмотреть возможность выбора одного, либо другого действия в зависимости от некоторого условия. Для программирования таких разветвляющихся участков алгоритма используется оператор условной передачи управления IF.

Структура оператора:

```
IF условие THEN список операторов [ELSE список операторов]
```

Здесь:

- IF(«если»), THEN(«то»), ELSE(«иначе») — служебные слова;
- *условие* — арифметическое или логическое выражение;
- *список операторов* — несколько (или ни одного) операторов MSX BASIC, разделённых двоеточиями, или номер существующей программной строки.

Напомним, что квадратные скобки — обозначение, указывающее, что синтаксическая конструкция внутри них не является обязательной!

Кроме того, вместо служебного слова THEN может быть использовано служебное слово GOTO, при этом за GOTO должен следовать номер существующей программной строки. Более того, в случае, когда *список операторов* состоит лишь из оператора GOTO n, служебное слово GOTO может опускаться.

Таким образом, если мы обозначим:

- n1 и n2 — номера программных строк,
- O1 и O2 — последовательности операторов MSX BASIC (операторы внутри каждой последовательности разделяются символом «:»), то можно записать следующие формы оператора IF:

```
IF  условие  THEN  O1
IF  условие THEN  n1
IF  условие  GOTO  n1
IF  условие  THEN  O1  ELSE  O2
IF  условие  THEN  O1  ELSE  n2
IF  условие THEN  n1  ELSE  O2
IF  условие THEN  n1  ELSE  n2
IF  условие  GOTO  n1  ELSE  O2
IF  условие  GOTO  n1  ELSE  n2
IF  условие  GOTO  O1  ELSE  O2  (обратите внимание!)
```

Отметим,что в некоторых версиях BASIC (например, для ПЭВМ«ДВК–2М») реализованы только «усечённые» конструкции оператора IF — без ветви «иначе» (т.е. без служебного слова ELSE):

```
IF условие THEN n1

IF условие THEN O1
```

.

Теперь рассмотрим несколько примеров:

- 1)

  ```
  300 DEFDBL A,B:IF A=B GOTO 700
  ```

  Внимание! Может оказаться, что два числа, которые должны быть *равными*, всё–таки *немного* отличаются из–за ошибок округления! Переход на строку с номером 700 может не осуществиться!

  Правильной будет следующая запись строки 300:

  ```
  300 DEFDBL A,B:IF ABS(A-B)<=EPS GOTO 700
  ```

  где EPS — допустимая «точность несовпадения» чисел A и B.

  Напомним, что основное различие между целыми и действительными (с фиксированной или плавающей точкой)

константами в том, что целые константы представлены в компьютере *точно*, а действительные — лишь *приближённо*. Выполняя операции с действительными числами, мы *обязательно* должны считаться с погрешностью округления, которая может очень сильно исказить результат! Взгляните:

```
10 INPUT A
20 ? (A/3)^2*9
run
? 4
 15.999999999999
Ok

10 INPUT A!
20 PRINT A!/3*3
run
? 1
 .99999999999999
Ok
```

- 2)

```
10 INPUT A,A!
20 IF A!=A THEN PRINT "Верно!":END ELSE PRINT "Посмотрите комментарий к предыдущему примеру!":END
run
? 0.000001000002,0.000001000002
Посмотрите комментарий к предыдущему примеру!
Ok
```

- 3)

```
10 IF A<B THEN Y=X^2:END ELSE Y=X^3:END
20 'Использование оператора END в операторе IF
```

- 4)

```
100 IF X/N=FIX(X/N) THEN PRINT"X делится на N!"
```

или

```
100 IF X MOD N=0 THEN PRINT"X делится на N!"
```

или

```
100 IF X/N=INT(X/N) THEN PRINT"X делится на N!"
```

Однако обратите внимание на следующий фрагмент:

```
10 X=LOG(EXP(1)):N=1
20 IF X/N=INT(X/N) THEN ?"Ура!":END ELSE ?"О,ужас!":
END
run
О,ужас!
Ok
```

- 5)

```
NEW
Ok
10 INPUT A,B
20 IF A>B THEN DIM C(4) ELSE DIM C(7)
30 C(6)=1:PRINT C(6)
run
? 1,3
 1
Ok

run
? 3,1
Subscript out of range in 30
Ok
```

Сообщение об ошибке «Subscript out of range» переводится как *«Выход за границы массива»*, то есть обращение к

элементу массива со значением индекса, выходящим за пределы границ измерения массива.

Таким образом, в некоторых случаях вместо использования «стирающего» оператора ERASE (см. раздел I.5.) для переопределения размерности массива в программе можно применять указанный приём.

- 6)

```
Ok
?11*(1/11)
 1
Ok
```

Однако естественно возникновение вопроса: почему компьютер выводит число 1 в качестве результата вычисления выражения 11*(1/11)?

Объяснение этого факта следует искать в особенностях работы процедуры вывода, инициализируемой оператором PRINT, а именно: эта процедура самостоятельно выполняет определённые операции округления выводимых числовых данных. Однако результаты подобных округлений, как видно из приведённых примеров, труднопредсказуемы, что в свою очередь может приводить к непредвиденным последствиям.

Замечание.
Будьте осторожны при записи условий, ибо посмотрите:

```
10 IF Z%=&B11 THEN PRINT "1" ELSE PRINT "0"
run
Sintax error in 10
```

Исправить ошибку можно, заключив &B11 в круглые скобки!

Напомним, что *семантика* (от греч.«semantikos» — «обозначающий») — значения единиц языка, а *синтаксис* (от греч.«syntaxis» — «построение,порядок») — способы соединения слов (и их форм) в словосочетания, предложения и текст.

Опишем семантику конструкции IF…THEN…ELSE… . Вначале вычисляется значение условия. Если в качестве условия записано строковое выражение, то выдаётся сообщение об ошибке:

«Type mismatch»
(*«Несоответствие типов»*).

Если условие истинно (TRUE), то управление передаётся фразе THEN; если условие ложно (FALSE) и есть фраза ELSE, то управление передаётся фразе ELSE; если условие ложно (FALSE) и нет фразы ELSE, то управление передаётся первому оператору *следующей* программной строки (то есть условно можно считать, что фраза ELSE есть, но за ней стоит «пустой» оператор, не производящий никаких действий).

Если непосредственно за THEN или ELSE указан номер программной строки и управление передаётся этой фразе, то фактически выполняется оператор

```
GOTO Номер строки
```

В качестве примера рассмотрим программу, вычисляющую и показывающую на экране наибольший общий делитель (НОД) двух натуральных чисел M и N, введённых пользователем. Программа использует *алгоритм Евклида*, который основывается на том, что НОД пары различных натуральных чисел совпадает с НОД пары, полученной из предыдущей заменой большего числа разностью большего и меньшего чисел.

```
NEW
Ok
20 DEFINT M,N:INPUT"Введите M,N";M,N
50 IF M>N THEN M=M-N
60 IF M<N THEN N=N-M
70 IF M<>N THEN GOTO 50
80 PRINT "НОД...";M:END

run
Введите M,N? 34,51
```

```
НОД... 17
Ok

run
Введите M,N? 15,625
НОД... 5
Ok
run
Введите M,N? 11,121
НОД... 11
Ok
```

Проследим теперь за работой компьютера. При первом обращении к строке 50 переменная M имеет значение 34, а переменная N — значение 51. Условие M>N не соблюдено, поэтому следует переход к строке 60. Условие M<N соблюдено, N получает новое значение 51-34=17. Условие M<>N (M не равно N) тоже соблюдено и поэтому строка 70 передаёт управление назад, строке 50.

Теперь M имеет значение 34, а N — значение 17, и условие M>N соблюдено; поэтому переменной M присваивается значение 34-17=17. M=N=17. Условие строки 60 не соблюдено, условие M<>N строки 70 тоже не соблюдено, и компьютер, наконец, выполняет строку 80,выводящую на экран дисплея текст «НОД... 17». В строке 80 работа программы заканчивается. Эта короткая программа позволяет найти НОД для любой пары натуральных чисел — выполнение серии строк 50, 60, 70 всегда будет повторено необходимое число раз. Об этом «заботятся» операторы IF…THEN !

В качестве ещё одного примера приведём программу вычисления наименьшего общего кратного (НОК) двух натуральных чисел. В алгоритме используем тот факт, что произведение НОК и НОД двух чисел равно произведению самих чисел (в программе — переменная K).

```
NEW
Ok
10 DEFINT M,N,K:INPUT"Введите M,N";M,N:K=M*N
20 IF M>N THEN M=M-N
30 IF N>M THEN N=N-M
40 IF M<>N THEN 20
50 PRINT "НОК...";K/M:END
run
Введите M,N? 4,6
НОК... 12
Ok

run
Введите M,N? 12,45
НОК... 180
Ok

run
Введите M,N? 888,117
НОК... 103896
Ok
```

Важное замечание: оператор IF должен полностью размещаться в одной программной строке (программная строка, вообще говоря, отлична от дисплейной строки!) и быть последним в ней. В этом случае будьте внимательны: программная строка должна включать не более 255 символов (вместе с номером)!

Однако не всегда удаётся «втиснуть» в одну программную строку последовательности операторов O1 и O2. В таких случаях помогает оператор GOTO, который передаёт управление на строку, с которой начинается нужный фрагмент программы (если необходимо вернуться к строке программы, следующей за условным оператором, то в конце фрагмента опять приходится ставить оператор GOTO). Такая конструкция весьма громоздка, поэтому предпочтительнее использовать подпрограммы (см. раздел IV.4.).

Любое арифметическое выражение может быть использовано в качестве *условия*; при этом следует помнить, что если результат его вычисления отличен от нуля, то значением условия является TRUE, во всех остальных случаях — FALSE.

Пример.

```
IF X <> 0 THEN …'' идентично записи ''IF X THEN …
```

Отметим, что *список операторов* может содержать оператор IF…THEN…ELSE… (в этом случае говорят о *вложении* операторов IF друг в друга).

Например, алгоритм нахождения большего из трёх чисел A, B и C можно записать при помощи оператора:

```
10 IF A>B THEN IF A>C THEN Z=A ELSE Z=C ELSE IF B>C THEN Z=B ELSE Z=C
11 ' Не правда-ли, элегантно (!) и непонятно?
```

Сравните:

```
10 BIG=X
20 IF Y>BIG THEN BIG=Y
30 IF Z>BIG THEN BIG=Z
```

При вложении операторов IF возникают определённые логические трудности, связанные с проблемой «болтающегося ELSE». В самом деле, если написать оператор вида:

```
IF условие1 THEN IF условие2 THEN O1 ELSE O2
```

то сразу же возникает вопрос: к какому IF (первому или второму) принадлежит фраза ELSE O2 (как говорят, ELSE «болтается»!)?

Для устранения возможной двусмысленности учтите, что любой ELSE «связывается» с ближайшей *предшествующей* ему в программной строке конструкцией IF…THEN, которой ещё не поставлен в соответствие ELSE (сравните с расстановкой скобок в алгебраическом выражении!).

Поэтому можно сформулировать простое *правило*: добавляйте нужное количество «пустых» ELSE, начиная с конца программной строки, для удовлетворения всех вхождений IF…THEN (сравните с правилом «вложенного IF»на языке программирования PL/1: каждая из вложенных конструкций IF…THEN должна иметь соответствующий ELSE, возможно «пустой»).

Таким образом, конструкция

```
IF условие1 THEN IF условие2 THEN O1 ELSE O2
```

понимается как

```
IF условие1 THEN ( IF условие2 THEN O1 ELSE O2 )
```

Скобки поставлены *только* для наглядности. Упаси Вас бог употребить их в программе! Но если Вам всё–таки хочется выделить в программе группу операторов, то можете применить для этой цели последовательность двоеточий :: … :.

Например:

```
IF условие1 THEN:::IF условие2 THEN O1 ELSE O2:::
```

Смысл конструкции можно изменить, добавив «пустой» ELSE:

```
IF условие1 THEN:::IF условие2 THEN O1 ELSE:::ELSE O2
```

В какой–то мере последовательность двоеточий играет роль операторных скобок, присущих, например, языкам ALGOL, PL/1, Pascal, C !

Примеры:

- 1) 032-01.bas
  032-01.bas

  ```
  NEW
  ```

```
Ok
10 INPUT X
20 IF X>1 THEN IF X<3 THEN PRINT "1≤X<3":END ELSE PRINT "X>3":END
30 PRINT "X≤1"
run
? 0
X≤1
Ok

run
? 1
X≤1
Ok

run
? 2.5
1≤X<3
Ok

run
? 4
X>3
Ok
```

- 2) 032-02.bas
  032-02.bas

```
NEW
Ok
10 INPUT X
20 IF X>1 THEN IF X<3 THEN PRINT "1≤X<3":END ELSE ELSE PRINT "X≤1":END
30 PRINT "X≥3"
run
? 0
X≤1
Ok

run
? 1
X≤1
Ok

run
? 2.5
1≤X<3
Ok

run
? 4
X≥3
Ok
```

- 3) составить программу, определяющую, каким должен быть Ваш идеальный вес, по следующему алгоритму (формула Мэгони, уточнённая К. Купером). Мужчина берет свой рост в дюймах, умножает это число на 4 и вычитает 128. Женщине надо умножить свой рост в дюймах на 3.5 и затем вычесть 108. Эта формула рассчитана на мужчину со средней шириной кости и женщину среднего телосложения. Но Вы можете заметить: «У меня широкая кость. Годится ли мне эта формула?» Да, но тогда прибавьте 10% к окончательному результату. Что бы определить, насколько широка у Вас кость, используйте следующее правило. Измерьте окружность запястья доминирующей руки, т.е. руки, которой Вы пишите. Мужчина имеет широкую кость, если окружность его запястья больше 18 см, женщина — больше 16.5 см. Если размер окружности запястья меньше, то говорят о среднекостном или о тонкокостном типе сложения.

  Описанная выше формула предполагает измерение роста в футах и дюймах, а веса — в фунтах. Напомним, что

  - 1 фут = 0.3048 м,
  - 1 дюйм = 0.0254 м,
  - 1 фунт = 0.4536 кг.

  032-03.bas
  032-03.bas

```
NEW
Ok
10 INPUT"Если Вы мужчина, то нажмите клавишу  'SHIFT'+'1',а если Вы женщина, то нажмите клавишу 'клюшка'";X%
20 INPUT "Укажите Ваш рост в метрах";L
30 INPUT"Укажите обхват запястья сильнейшей руки в см";Y
35 PRINT"Ваш идеальный вес :"
40 IF X%=1 THEN IF Y>18 THEN PRINTUSING"###.###";(L/.0254*4-128)*.4536*1.1;ELSE PRINTUSING
"###.###";    (L/.0254*4-128)*.4536;ELSE    IF    Y>16.5    THEN    PRINTUSING
"###.###";L/.0254*3.5-108)*.4536*1.1;ELSE PRINTUSING "###.###";(L/.0254*3.5-108)*.4536;
50 PRINT " кг":END
run

Если Вы мужчина, то нажмите клавишу  'SHIFT'+'1',  а если Вы женщина, то нажмите клавишу 'клюшка'? 1
Укажите Ваш рост в метрах? 1.745
Укажите обхват запястья сильнейшей руки в см? 18.5
Ваш идеальный вес:
 73.249 кг
Ok
```

- 4) составить программу решения квадратного уравнения

```
a·x² + b·x + c = 0
```
.

032-04.bas
032-04.bas

```
NEW
Ok
100 INPUT"Задайте a,b,c";A,B,C
110 DISC=B*B-4*A*C:W=1/2/A:Z=-B/2/A
120 IF ABS(DISC)<1.E-14 THEN PRINT"Двойной корень,x=";Z:GOTO 100'Условие ABS(DISC)<1.E-14 взято вместо
условия DISC=0 с целью компенсации ошибок округления.
130 IF DISC<0 THEN 160
140 PRINT "Корни";Z+SQR(DISC)*W;
150 PRINT "и";Z-SQR(DISC)*W:GOTO 100
160 PRINT "Мнимые корни:";Z;"+/-";SQR(-DISC)*W;"i":GOTO100
run

Задайте a,b,c? 1,2,1
Двойной корень,x=-1
Задайте a,b,c? 1,1,1
Мнимые корни:-.5 +/- .8660254037844 i
Задайте a,b,c? 1,-5,6
Корни 3 и 2
Задайте a,b,c? 0,1,1
Division by zero in 110
Ok
```

Заметим, что указанная программа «не работает» в том случае, когда a=0. Поэтому добавим к программе строку:

```
105 IF A=0 THEN IF B=0 THEN IF C=0 THEN PRINT"x-любое":GOTO 100 ELSE PRINT"решений нет":GOTO 100 ELSE
PRINT"корень равен";-C/B:GOTO 100
```

Это приведёт к следующему результату:

```
run
Задайте a,b,c? 0,0,0
x-любое
Задайте a,b,c? 0,1,1
корень равен-1
Задайте a,b,c? 0,0,1
решений нет          и так далее...
```

Целые числа сотворил господь бог, всё остальное — дело рук человеческих.

—Л.Кронекер

- 5) написать программу, которая выводит на экран разложение введённого в компьютер натурального числа на простые сомножители, например,

```
24=2*2*2*3,  17=17
```
.

Напомним *основную теорему арифметики*.

Любое натуральное число n>2 может быть представлено в виде произведения простых множителей; с точностью до порядка множителей такое представление единственно.

032-05.bas
032-05.bas

```
10 PRINT "Введите натуральное N,N>1"
20 INPUT N:PRINT N;"=";:P=2
30 D=N/P
40 IF D=INT(D) THEN ?S$;P;:S$="*":N=D:GOTO 30 ELSE P=P+1
50 IF N>=P THEN 30
60 PRINT:END

run
Введите натуральное N,N>1
? 111
 111 = 3 * 37
Ok

run
Введите натуральное N,N>1
? 1111
 1111 = 11 * 101
Ok
```

Далее, легко получаются следующие результаты:

```
          11111 = 41 * 271
         111111 = 3 * 7 * 11 * 13 * 37
        1111111 = 239 * 4649
       11111111 = 11 * 73 * 101 * 137
                …           …
 11111111111111 = 11 * 239 * 4649 * 909091
```

Объектом исследования в данной программе являются числа, записанные цепочкой единиц, *репьюниты* (от английского «repeated unit» — «повторенная единица»): 1,11,111, и т.д. Этот термин придумал в 1964 году американец А. Бейлер. Первую же работу об этих удивительных числах написал двумя столетиями раньше Иоганн–III Бернулли в связи с исследованиями периодических десятичных дробей. В 1895 году француз Э.Люка в книге «Занимательная математика» публикует проверенную им таблицу всех простых делителей репьюнитов до 18–го репьюнита включительно.

В частности,

```
   111111111111111 = 3*37*31*41*271*290*6161
  1111111111111111 = 11*17*73*101*137*5882353
 11111111111111111 = 2071723*5363222357
111111111111111111 = 3*7*11*13*19*37*52579*333667 .
```

Сегодня таблица делителей репьюнитов достигла n=3000 (С. Ейтс, 1975), однако в ней ещё достаточно много пробелов, и даже не всё ясно в первой сотне репьюнитов. Практический интерес к репьюнитам существует в теории арифметических кодов, служащей основой помехоустойчивого кодирования в компьютерной технике, и в проблеме простых чисел: их ищут сейчас в основном среди так называемых *чисел Мерсенна*, имеющих вид

2^n-1 , n=1,2,3,…
,

и последним достижением на этом пути стало число

$2^{132049}-1$ (1984 год)
.

А ведь числа Мерсенна — это репьюниты в двоичной системе счисления! Проверьте этот факт!

Обязательно проведите аналогию между оператором условной передачи управления IF в языке программирования MSX BASIC и командой ветвления
если условие
│ то серия 1
│ иначе серия 2
всё
в школьном алгоритмическом языке! Кстати отметим, что аналог служебного слова всё в MSX BASIC не нужен — признаком конца оператора IF является конец программной строки, в которой он находится.

В школьном алгоритмическом языке введена ещё одна команда — команда *проверки контрольного условия*

утв *условие*

Если в процессе исполнения алгоритма условие нарушается, то компьютер прекращает исполнение алгоритма и сообщает, что возник отказ.

Аналогом данной команды в MSX BASIC может служить, например, конструкция IF…THEN…ELSE в следующем примере: 032-06.bas
032-06.bas

```
NEW
Ok
10 'Вычисление значения функции  y=√(x-2)
20 INPUT"Введите X";X
30 IF X>=2 THEN PRINT SQR(X-2) ELSE PRINT 1/0
run
Введите X? 4
 1.414213562373
Ok

run
Введите X?
Division by zero in 30
Ok
```

III.3. Оператор ON GOTO

Общая форма записи оператора–переключателя (оператора выбора по целому значению) следующая:

```
ON β GOTO N1, N2, …, Nk
```
,

где:

- ON («на»), GOTO («идти к») — служебные слова;
- β — арифметическое выражение, целая часть значения которого должна принадлежать отрезку [0,255];
- N1, N2, …, Nk — номера программных строк; оператор ON GOTO воспринимает столько номеров строк в списке, сколько Вы сможете записать в одной логической строке (помните, что она ограничена 255 символами!).

Выполнение оператора ON начинается с вычисления целой части значения арифметического выражения β, которое мы обозначим p.

Далее, если p∈{1, 2,… , k}, k — натуральное число, то управление переходит к строке Np.

Если p=0 или p>k, то выполняется оператор, следующий за ON.

Если же p<0, то последует сообщение об ошибке:

«Illegal function call»

(*«Неправильный вызов функции»*).

Примеры:

- 1) 033-01.bas
  033-01.bas

```
NEW
Ok
10 INPUT X
20 ON X GOTO 200,300,400:? X:GOTO 10
200 PRINT X+200:END
300 PRINT X+300:END
400 PRINT X+400:END

run
? 0
 0
? 3
 403
Ok

run
? -1
Illegal function call in 20
Ok

run
? 2
 302
Ok

run
? 1
201
Ok
```

Оператор `ON GOTO` удобен, когда выполнение одной или нескольких различных ветвей алгоритма определяется значением переменной или выражения (так называемый выбор *по значению*). Если эти ветви *короткие*, то смело используйте оператор `ON GOTO`. Если же ветви имеют достаточно сложную логику и могут быть оформлены как *подпрограммы*, то для реализации выбора по значению используйте оператор `ON GOSUB` (см. раздел IV.5.).

- 2) 033-02.bas
  033-02.bas

```
NEW
Ok
5 'Анализ символов, вводимых с клавиатуры
10 ON ASC(INPUT$(1))-26 GOTO 101,102,103
101 GOTO 10
102 PRINT "Нажата клавиша ⟶":END
103 PRINT "Нажата клавиша ⟵":END
run                                          ┌─────┐
Нажата клавиша ⟶       Нажмите клавишу │ ⟶ │ !
Ok                                           └─────┘
run                                          ┌─────┐
Нажата клавиша ⟵       Нажмите клавишу │ ⟵ │ !
Ok                                           └─────┘
run                                          ┌─────┐
Illegal function call in 10  Нажмите клавишу │ TAB │ !
Ok                                           └─────┘
```

О строковой функции `INPUT$()` см. в разделе VII.1.1.

- 3) Оператор

```
ON A GOTO 110,110,110
```

позволяет совершить переход к программной строке 110, если значение переменной A находится в полуинтервале [1,4).

- 4) Пусть в программе требуется выполнить три различные операции в зависимости от значения переменной K, которое может быть меньше, больше или равно нулю.

  Приведём фрагмент программы:

```
110 ON SGN(K)+1 GOTO 170,210
120 'Выполнение операции при  K<0
                …
170 'Выполнение операции при  K=0
                …
210 'Выполнение операции при  K>0
                …
```

- 5) Пусть в программе требуется совершить переход к различным строкам, если значение переменной A находится в следующих диапазонах:

```
   0≤A< 800 — переход к строке 900;
 800≤A<1600 — переход к строке 800;
1600≤A<2400 — переход к строке 700.
```

  Если же A≥2400, то должен выполняться следующий оператор. В этом случае программная строка с оператором `ON` будет иметь вид:

```
50 ON A/800+1 GOTO 900,800,700
```

Заметим, что оператор `ON GOTO` потенциально опасен, особенно для начинающих, так как его неаккуратное использование может привести к запутанной, трудной для понимания программе.

В заключение проведём аналогию между оператором ON GOTO и командой *выбор* в сокращённом варианте
выбор
│ при условие 1: серия 1
│ при условие 2: серия 2
│ …
│ при условие N: серия N
всё
в школьном алгоритмическом языке. Говорят, что команда выбора реализует «выбор *по условию*». Выбор по условию введён в школьный алгоритмический язык из языка программирования Рапира. В большинстве языков высокого уровня (в том числе и в MSX BASIC!) этой конструкции в чистом виде нет, но её можно моделировать следующей последовательностью вложенных команд ветвления:
если условие 1
│ то серия 1
│ иначе если условие 2
│ │ то серия 2
│ │ иначе …
│ │ если условие N
│ │ │ то серия N
│ │ всё
│ │ …
│ всё
всё

Зато в версии MSX BASIC есть оператор `ON GOTO` — другая форма выбора (по *значению* выражения).

Ясно, что, вообще говоря, механизмы работы выбора по условию и по значению совпадают. Отличие лишь в форме условия: в выборе по условию оно может быть любым, а в выборе по значению условие определяется значением арифметического выражения.

III.4. Программирование циклов

Три месяца каждую вторую среду он ходил смотреть

фильм «Девушка моей мечты», но на связь с ним никто не вышел.

—В.Кожевников

Понятие цикла является одним из основополагающих понятий информатики. Считается, что если бы в программах не было возможности организовывать циклы, то применять ЭВМ для решения многих задач не имело бы никакого смысла.

Цикл — это многократное повторение одной и той же группы операторов, возможно, с новыми значениями переменных, входящих в эту группу.

Типичный пример цикла мы видим в «Сказке о рыбаке и рыбке» А.С.Пушкина. Действия слабохарактерного старика повторялись *циклически*:

разговор с золотой рыбкой
путь от моря к старухе
получение нового приказания от старухи и снова
к морю,
к золотой рыбке.

Параметром цикла (в терминах языка BASIC) здесь выступает алчность старухи, которая увеличивается при каждом новом прохождении циклического участка.

Программы, содержащие циклы, называются *циклическими*.

Цикл, не содержащий внутри себя других циклов, называется *простым*. В противном случае его называют *кратным* или *вложенным*.

В школьном алгоритмическом языке для записи циклических алгоритмов применяются две конструкции: цикл «для» и цикл «пока».

Рассмотрим, как на языке программирования MSX BASIC программируется *цикл «для»*.

Для этого существуют два специальных оператора: оператор начала цикла (оператор FOR), оператор конца цикла (оператор NEXT). Операторы FOR и NEXT, как правило(!), используются совместно.

Оператор начала цикла (его называют ещё *заголовком* цикла) имеет вид:

```
FOR i = α TO β [ STEP γ ]
```
.

Здесь:

- FOR («для»), TO («до»), STEP («шаг»), NEXT («следующий») — служебные слова;
- i — имя числовой переменной (без индекса!);
- α, β, γ — арифметические выражения.

Вслед за оператором *начала* цикла располагают операторы программы, которые должны выполняться циклически (эти операторы образуют *тело* цикла). За последним оператором тела цикла размещается оператор *конца* цикла, который имеет следующий синтаксис:

```
NEXT [i]
```

Обратите внимание на то, что переменная, указываемая после NEXT, должна быть той же переменной, которая упоминается в операторе начала цикла после служебного слова FOR (эту переменную называют *параметром* цикла).

Значения арифметических переменных α и β задают соответственно начальное и конечное значения параметру цикла. Значение арифметического выражения γ — это *шаг*, на величину которого изменяется параметр цикла после каждого повторения тела цикла. Шаг может быть как положительным, так и отрицательным.

Если шаг цикла равен +1, указание STEP γ в операторе начала цикла может опускаться (говорят, что «по умолчанию шаг цикла равен 1»).

Итак, в программе цикл «для» записывается следующим образом:

```
FOR i = α TO β [ STEP γ ]
   тело цикла
NEXT i
```

Выполнение этого фрагмента программы происходит следующим образом: параметру цикла i присваивается его начальное значение (значение арифметического выражения α), и один раз выполняется тело цикла (т.е. группа операторов, размещённых между заголовком цикла и соответствующим оператором NEXT). Вслед за этим оператор NEXT изменяет значение параметра цикла i на величину шага (значение арифметического выражения γ) и проверяет, не вышло ли новое значение параметра цикла из рабочего интервала (если шаг положителен, то не стало ли оно больше конечного значения параметра цикла (значения арифметического выражения β); если шаг отрицателен — не стало ли меньше конечного значения параметра цикла).

Если этого не происходит, то осуществляется повторное выполнение тела цикла, в противном случае — *выход* из цикла, то есть переход к оператору, следующему за NEXT.

Заметим, что при *любых* значениях α, β, γ тело цикла будет выполнено *хотя* бы *один* раз (в отличие от школьного алгоритмического языка и других языков программирования!)!

Пример:
034-01.bas
034-01.bas

```
NEW
Ok
10 FOR A=1 TO 10 STEP-1
20 PRINT A
30 NEXT A
run
 1
Ok
```

Эта программа выведет на экран только начальное значение параметра цикла A=1. При первой же встрече с оператором NEXT A будет обнаружено, что конечное значение параметра цикла (A=10) уже «позади»!

Ещё раз обращаем Ваше внимание на то, что:

- проверка на окончание цикла производится *после* выполнения тела цикла;
- изменение параметра цикла происходит *перед* проверкой на окончание цикла.

Таким образом, следующие фрагменты эквивалентны:

```
 …
10 FOR K=1 TO 4 STEP 2
 …  тело цикла  …
100 NEXT K
 …
```

```
 …
10 K=1
20 'Пустой оператор
 …  тело цикла  …
100 K=K+2
110 IF K≤4 GOTO 20
 …
```

Отметим, что оператор цикла FOR…NEXT можно располагать на одной программной строке. Это целесообразно делать, если тело цикла состоит из одного–двух операторов. Например:
034-02.bas
034-02.bas

```
10 FOR L=1 TO 10:PRINT L;L^3:NEXT L
```

Выполнив эту строку, компьютер выведет на экран дисплея таблицу кубов чисел 1, 2, 3, …, 10.

Кстати, сравните конструкцию FOR…NEXT с циклом «для» в школьном алгоритмическом языке:
для x от x_{min} до x_{max} шаг $x_{шаг}$
│нц
│ серия команд
│кц

Напомним, что в том месте дисплейной строки, где надо обратить Ваше внимание на наличие символа пробел (« »), мы будем использовать в тексте символ «·».

Важное замечание!
Значения переменных α, β, γ можно изменять операторами, находящимися в теле цикла! Следующие две программы идентичны:

034-031.bas
034-031.bas

```
NEW
Ok
10 A=7:B=9:H=1
20 FOR I=A TO B STEP H
30 A=4:B=5:H=2
40 ? A;B;H:NEXT
run
·4··5··2
·4··5··2
·4··5··2
Ok
```

034-032.bas
034-032.bas

```
NEW
Ok
5 A=7:B=9:H=1:A1=A:B1=B:H1=H
10 FOR I=A1 TO B1 STEP H1
20 A=4:B=5:H=2
25 ? A;B;H:NEXT
run
·4··5··2
·4··5··2
·4··5··2
Ok
```

Таким образом, текущие значения переменных α, β, γ (в теле цикла) могут не совпадать с начальным, конечным значениями параметра цикла и значением шага соответственно!

В то же время, *значения* α, β и γ, указанные в заголовке цикла, «недоступны» для операторов тела цикла при условии, что в теле цикла нет обращения к заголовку цикла.

Пример:
034-04.bas
034-04.bas

```
NEW
Ok
10 INPUT A,B,H
20 FOR I=A TO B STEP H:?"I=";I
30 A=A+2:IF A<6 GOTO 20 ELSE ?A
40 NEXT:?I
run
? 2,3,1
I= 2
I= 4
 6
 5
Ok
```

Следовательно, изменить начальное и конечное значения параметра цикла, а также значение шага можно только(!) с помощью обращения к заголовку цикла.

Таким образом, операторы, находящиеся в теле цикла могут управлять числом повторений цикла!

Значение параметра цикла i также *можно* изменять операторами, находящимися в теле цикла!

По окончании работы цикла значение параметра цикла i равно первому его значению, для которого выполнилось неравенство

```
sign([β]-i)·[γ]<0
```
(∗)

,

где обозначено:

- [β] и [γ] — значения арифметических выражений β и γ соответственно;
- функция sign(x)(«signum» — «знак») определяется формулой:

```
    ⎧ 1, если  x>0
sgn(x)= ⎨ 0,  если  x=0
    ⎩-1, если  x<0
```
.

Условие (∗) принято вместо условия ([β]-i)·[γ]<0 потому, что при малых ([β]-i) и [γ] произведение ([β]-i)·[γ] может дать «машинный нуль», хотя будет выполнено неравенство (∗)!

Кстати, число повторений цикла «без фокусов» можно вычислить по формуле:

```
K=INT(ABS(([β]-[α])/[γ]))+1
```

Отметим, что во многих других языках программирования (ALGOL, Fortran) по окончании цикла значение параметра i становится *неопределённым*. В этом случае им можно «пользоваться» только после присвоения нового значения.

Примеры:

- 034-05.bas
  034-05.bas

```
NEW
Ok
10 INPUT A,B,H
20 FOR I=A TO B STEP H
40    ?I;:I=I+1:?I
50 NEXT
run
? 1,3,1
·1··2
·3··4
Ok
```

- 034-06.bas
  034-06.bas

```
NEW
Ok
10 INPUT A,B,H
20 FOR I=A TO B STEP H
30    H=H+1:I=I+H:?H;I
40 NEXT
run
? 1,4,1
·2··3
·3··7
Ok
```

- 034-07.bas
  034-07.bas

```
NEW
Ok
10 INPUT A,B,H
20 FOR I=A TO B STEP H
30 NEXT I:PRINT I
run
? 4,5,2
 6
Ok
```

- 034-08.bas
  034-08.bas

```
NEW
Ok
10 INPUT A,B,H
20 FOR I=A TO B STEP H
30 I=I^2:NEXT:?I
run
? 5,1,1
 26
Ok
```

Тело цикла по отношению к остальной части программы замкнуто, то есть *нельзя* «входить» в него, минуя заголовок цикла.

Возможен досрочный выход из цикла не через его окончание, а с использованием операторов `IF…THEN…ELSE…` и `GOTO` (в этом случае говорят, что цикл выполняется «n+1/2» раз!). При этом значение параметра цикла *определено* и равно тому значению, при котором произошёл выход из цикла.

Пример. Найти индекс первого положительного элемента одномерного числового массива A(N).
034-09.bas
034-09.bas

```
NEW
Ok
10 DEFINT N,I:INPUT "Введите N";N
20 DIM A(N)'Описание массива A
30 FOR I=1 TO N:INPUT A(I):NEXTI'Ввод массива  A
40 FOR I=1 TO N
50 IF A(I)>0 THEN ?"Искомый индекс:";I:GOTO 60 ELSE NEXTI
60 PRINT"We are out of the loop!" 'Мы вышли из цикла!
run
Введите N? 4
? -1
? -3
? 5
? -4
Искомый индекс: 3
We are out of the loop!
Ok
```

Отметим, что оператор `GOTO 60` в 50 строке можно заменить на последовательность операторов:

```
I=N:NEXT I
```

(мы пользуемся тем, что параметр цикла I можно менять внутри цикла!), которая обеспечит в случае A(I)>0 выход из цикла. Обратите внимание на то, что можно даже выйти из тела цикла в «окружающую» его часть программы, а затем вернуться снова в тело цикла, *минуя* заголовок!

Пример.
034-10.bas
034-10.bas

```
NEW
Ok
10 FOR I=1 TO 5
20 IF I≤2 THEN GOTO 100
30 M=I^2:?M
40 NEXT I:END
100 I=I+3:GOTO 30 'Возвращение в тело цикла!
run
 16
 25
```

```
Ok
```

Приведённый пример показывает, как, используя оператор GOTO, можно «расширить» тело цикла, включив в него произвольные части программы.

1. *Примеры простых циклов*

> 5% текста программы занимают 90% времени её выполнения.
>
> —*Аксиома программирования*

- 1)

  034-11.bas
  034-11.bas

  ```
  10 FOR I=1 TO 1000:NEXT 'Задержка ≈ 2.35 секунды
  ```

  Применение оператора цикла с «пустым» телом позволяет получить эффект *задержки* по *времени*. Запомните этот трюк!

- 2)

  034-12.bas
  034-12.bas

  ```
  Ok
  10 'Признание в любви!
  20 INPUT"Введите целое число N";N%
  30 FOR I%=1 TO N%
  40    PRINT"I love YOU!"
  50 NEXT I%
  ```

  Думаем, что результат работы этой N–«любвиобильной» программы Вас порадует. Заметим, что программа идентична алгоритму с использованием команды повторения n раз в школьном алгоритмическом языке [13]:
  нц число повторений раз
  серия команд
  | кц
  так как тело цикла не зависит от параметра цикла.

- 3)

  034-131.bas
  034-131.bas

  ```
  Ok
  10 DEFINT N,I:INPUT "Введите N";N
  20 DIM A(N) 'Массив A описан!
  30 FOR I=0 TO N:INPUT A(I):NEXT I
  40'Данный фрагмент позволяет осуществить ввод элементов одномерного массива A (в массиве N+1 элемент!)
  ```

  Приведём ещё один способ решения указанной проблемы:
  034-132.bas
  034-132.bas

  ```
  Ok
  10 DIM A(5) 'Массив A содержит 6 элементов!
  20 DATA 1.,2.,3.,4.,5.,6.
  30 FOR I=0 TO 5:READ A(I):NEXTI
  ```

- 4)

  инициализация массива (присвоение начальных значений) целыми псевдослучайными числами, принадлежащими отрезку [A,B]:
  034-14.bas
  034-14.bas

  ```
  NEW
  Ok
  10 DEFINT N,I,C:INPUT A,B,N:DIM C(N)
  20 X=RND(-TIME) 'начальная установка генератора псевдослучайных чисел
  30 FOR I=1 TO N
  40 C(I)=INT((B-A+1)*RND(1)+A):?C(I):NEXTI
  run
  ? -5,3,12
  -1··-3··0··-1··0··-1··-3··3··2··-5··0··1
  ```

```
Ok
```

Замечание. Как можно чаще применяйте указанный способ инициализации массива псевдослучайными числами при *тестировании* (процессе обнаружении ошибок) Ваших программ!

- 5) 034-15.bas
  034-15.bas

```
NEW
Ok
10 'Табулирование функции  y=exp(sin(x)) на [A,B] с шагом H.
100 INPUT A,B,H
110 FOR X=A TO B STEP H
120 ?USING"#.###  #.###";X;EXP(SIN(X))
130 NEXT
run
? .0,1.,.2
0.000  1.000
0.200  1.220
0.400  1.476
0.600  1.759
0.800  2.049
1.000  2.320
Ok
```

- 6)

составить программу табулирования функции y=sinx на отрезке [a,b], причём на первой половине отрезка — с шагом h, а на второй половине отрезка — с шагом h/2.

Приведённые ниже программы эквивалентны и дают полное представление о работе оператора `FOR…NEXT`.

  - 6.1) 034-161.bas
    034-161.bas

```
NEW
Ok
10 INPUT A,B,H:P=0
20 FOR X=A TO B STEP H
30 IF X>=(A+B)/2ANDP=0 THEN P=1:A=(A+B)/2:H=H/2:GOTO 20
35 PRINT X;SIN(X)
40 NEXT X
run
? 0,1,.1
 0  0
.1  .099833416646831
.2  .19866933079507
    …          …
.95  .81341550478941
 1  .84147098480792
Ok
```

  - 6.2) 034-162.bas
    034-162.bas

```
NEW
Ok
10 INPUT A,B,H:P=0'P-флажок!
20 A1=A:B1=B:H1=H
25 X=A1
30 IF X>=(A+B)/2ANDP=0 THEN P=1:A=(A+B)/2:H=H/2:GOTO 20
35 PRINT X;SIN(X)
40 X=X+H1:IF X≤B1 GOTO 30
```

  - 6.3) 034-163.bas
    034-163.bas

```
NEW
Ok
10 INPUT A,B,H:P=0:X=A
20 IF X>=(A+B)/2 THEN FOR X=(A+B)/2+H/2 TO B STEP H/2:P=1 ELSE FOR X=A TO (A+B)/2 STEP H
30 PRINT X;SIN(X)
40 NEXT:IF P=0 THEN GOTO 20
```

- 7)

пусть требуется вычислить значение функции $y=x_2$ для x=0, 0.1, 0.2,..., 1., кроме x=0.3. Предостережём от *неверного* (в принципе!) решения:
034-171.bas
034-171.bas

```
10 FOR X=0 TO 1 STEP 0.1
20 IF X><0.3 THEN ?X;X^2:NEXT X ELSE NEXT X
```

Значение шага цикла (0.1) есть десятичная константа с фиксированной точкой. Поэтому, вследствие накопления погрешности округления, мало надежды (хотя она и есть!) на то, что на четвёртом цикле значение X будет точно равно 0.3. Да и на одиннадцатом цикле значение X может оказаться «чуточку» большим (меньшим) единицы!

Поэтому верное решение будет, например, таким:
034-172.bas
034-172.bas

```
10 FOR Y%=0 TO 10
20 IF Y%><3 THEN X=Y%/10:?X;X^2:NEXT Y% ELSE NEXT Y%
```

- 8) 034-18.bas
  034-18.bas

```
NEW
Ok
5 'Вычисление значения многочлена  A(0)+A(1)·x+A(2)·x₂+···+A(n)·xⁿ  в точке  x=X0  по схеме Горнера
10 INPUT"Введите степень многочлена  n=";N:INPUT "X0=";X
20 FOR I=0 TO N:PRINT"A("I")=";:INPUT A(I):NEXTI
36 C=A(N)
40 FOR I=N-1 TO 0 STEP -1:C=C*X+A(I):NEXTI:PRINT "c=";C
```

- 9)

последовательность чисел Фибоначчи $F(1),F(2),\ldots,F(n),\ldots$ определяется по следующим рекуррентным формулам: $F(1)=1,F(2)=1,F(n+2)=F(n)+F(n+1),n=1,2,\ldots$.

Напишите программу, вычисляющую $F(N)$ по заданному N.
034-191.bas
034-191.bas

```
NEW
Ok
10 DEFINT N,I:INPUT N:A=1:B=1
20 IF N=1ORN=2 THEN C=1:? C:END
30 FOR I=1 TO N-2:C=A+B:A=B:B=C:NEXT
35 PRINT C:END
run
? 5
 5
Ok

run
? 10
 55
Ok

run
? 100
 3.5422484817927E+20
Ok
```

Известно, что для членов последовательности Фибоначчи имеет место красивая формула:

$$F(n)=\frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n-\left(\frac{1-\sqrt{5}}{2}\right)^n\right],n=1,2,\ldots$$

Используем её для написания второго варианта предыдущей программы: 034-192.bas
034-192.bas

```
NEW
Ok
10 FOR N=1 TO 63
20    A=FIX((((1+SQR(5))/2)^N-((1-SQR(5))/2)^N)/SQR(5)+0.5)
30    PRINT"n=";N;"A=";A
40 NEXT
run
n= 1  A= 1
   …
n= 63  A= 6557470319842
Ok
```

Как уже говорилось, тело цикла может содержать второй цикл — *внутренний* по отношению к первому, *внешнему* циклу

(«бумажные стаканчики»). Внутренний цикл должен целиком содержаться в теле внешнего цикла. Таким образом, оператор NEXT для внутреннего цикла должен стоять перед оператором NEXT внешнего цикла. Нарушение этого правила приводит к появлению на экране сообщения об ошибке:

«NEXT without FOR in ...»
(*«Оператор NEXT без оператора FOR в строке ...»*).

Пример.
034-20.bas
034-20.bas

```
NEW
Ok
10 FOR I=1 TO 3:FOR J=2 TO 4
20 IF J>1 THEN NEXT I
30 NEXT J
run
NEXT without FOR in 30
Ok
```

Тело внутреннего цикла в свою очередь может содержать ещё один цикл, внутренний по отношению к нему и так далее. Интерпретатор MSX BASIC устанавливает максимальную «глубину» (число) вложений циклов, однако она настолько велика, что превысить её практически невозможно!

Учтите, что каждый оператор FOR…NEXT занимает при работе 25 байтов *стека*, причём стек освобождается только при завершении цикла, после последнего выполнения оператора NEXT!

Тем не менее, если программа использует слишком много циклов одновременно, может возникнуть сообщение об ошибке:

«Out of memory»
(*«Выход за пределы памяти»*).

Если требуется, чтобы несколько вложенных операторов FOR заканчивались в одном месте, можно использовать один оператор NEXT, содержащий список параметров циклов, разделённых запятыми: первое имя в списке соответствует самому внутреннему, последнее — самому внешнему циклу.

2. *Примеры вложенных циклов*
   - 1) 034-21.bas
     034-21.bas

     ```
     Ok
     10 FOR X=1 TO 100
     20 FOR Y=1 TO  10
     30 ? "1000 раз"
     40 NEXT Y
     50 ? "Только 100 раз"
     60 NEXT X
     ```

   - 2) 034-22.bas
     034-22.bas

     ```
     Ok
     10 'Инициализация диагональной (единичной) матрицы  X(3,3)
     20 FOR I=1 TO 3:FOR J=1 TO 3
     30 X(I,J)=INT(I/J)*INT(J/I)
     40 NEXT J
     50 NEXT I
     ```

   - 3)

     034-23.bas
     034-23.bas

     ```
     10 'Ввод двумерного массива  MAS  по столбцам
     20 DIM MAS(30,20):FOR I=1 TO 20:FOR J=1 TO 30
     30 INPUT MAS(J,I):? MAS(J,I):NEXT J,I
     ```

     Оператор ? MAS(J,I) позволяет осуществить контроль вводимой информации. Вопрос к читателю: что необходимо изменить в приведённом фрагменте, если Вы пожелаете ввести массив MAS по строкам?

   - 4)

034-24.bas
034-24.bas

```
Ok
5'Проверка результатов логических операций NOT,AND,OR,XOR,EQV,IMP для операндов i, j∈-{-1,0}.
10 FOR I=-1 TO 0:FOR J=-1 TO 0
20 PRINT I;J;NOT(I);IANDJ;IORJ;IXORJ;IEQVJ;IIMPJ
40 NEXT:NEXT
run
-1 -1  0 -1 -1  0 -1 -1
-1  0  0  0 -1 -1  0  0
 0 -1 -1  0 -1 -1  0 -1
 0  0 -1  0  0  0 -1 -1
Ok
```

Сравните полученные результаты с таблицей, приведённой ранее в разделе I.7.2.

- 5) 034-25.bas
034-25.bas

```
NEW
Ok
5'"Удобный" вывод на экран двумерного массива.
10 FOR I=1 TO 3:FOR J=1 TO 4:READ A(I,J):NEXTJ,I
15 FOR I=1 TO 3:FOR J=1 TO 4
20 'Обратите внимание на символ ";" в строке 25!
25 PRINT A(I,J);:NEXT J:PRINT:NEXT I
30 DATA 1,2,3,4,3,4,5,6,0,4,5,3
run
 1  2  3  4
 3  4  5  6
 0  4  5  3
Ok
```

Операторы FOR и NEXT, как правило, должны идти в паре и быть согласованы друг с другом. Если оператор NEXT выполняется раньше оператора FOR с тем же параметром цикла, то при выполнении такого NEXT выдаётся сообщение об ошибке:

«NEXT without FOR»
(«Нет оператора FOR для данного NEXT»).

Оператор FOR…NEXT — единственный способ в MSX BASIC для изменения естественного порядка выполнения операторов без задания номеров программных строк, на которые нужно перейти.

Весьма поучительно сопоставить конструкцию FOR…NEXT с конструкцией цикла «пока» в школьном алгоритмическом языке:
пока условие
нц
│ серия команд
│кц

Отметим, что в некоторых версиях BASIC (например, для персональных компьютерах фирмы IBM: IBM PC, IBM PC/XT, IBM PCjr) существует конструкция, почти до словно повторяющая цикл «пока»; её структура([11], с.74):

```
WHILE  условие STEP 0
 тело цикла
WEND
```

здесь WHILE («while» — «пока»), WEND («конец цикла WHILE») — служебные слова.

Оператор цикла WHILE повторяет выполнение операторов тела цикла до тех пор, пока выполняется требуемое условие. Если условие не истинно или становится не истинным, операторы тела цикла не выполняются ни разу, и управление передаётся первому оператору, следующему за WEND.

Обратите внимание на то, что если условие оказывается ложным (не удовлетворяется при первой же проверке),то операторы тела цикла вообще не выполняются!

Как правило, условие в операторе цикла WHILE…WEND должно когда–нибудь стать ложным, иначе повторение будет

бесконечным. Впрочем, если Ваша программа вошла в бесконечный цикл или каким–нибудь другим образом вышла из–под Вашего контроля, *не паникуйте*! В MSX BASIC предусмотрена возможность *прерывания* программы — одновременным нажатием на две клавиши CTRL+STOP.

> Метод решения хорош, если с самого начала мы можем предвидеть — и далее подтвердить это, — что, следуя этому методу, мы достигнем цели.
>
> *—Г.В.Лейбниц*

> Истина всегда оказывается проще, чем можно было предположить.
>
> *—Р.Фейнман*

Попытаемся циклом FOR…NEXT(!) описать цикл «пока». Следующая весьма нетривиальная конструкция позволяет это сделать:

```
   …
10 IF NOT условие THEN 60'Если условие ложно,то в цикл не входим!
20 FOR I=1 TO 0 STEP 1 'Оригинальный заголовок цикла, не правда ли?
30  тело цикла
40    I = условие 'Потрясающе! Если условие ложно, то I=0, и цикл заканчивается; если условие истинно,то
I=-1, и мы продолжим выполнение тела цикла.
50 NEXT I
60 … 'Продолжение программы.
```

Если в операторе IF использовать указатель шага цикла STEP 0, то цикл будет бесконечным. *Иногда*(?) это бывает полезным!

Пример 1.
034-31.bas
034-31.bas

Вычислить сумму вида: $$S=\sum_{n=1}^{k} n$$

```
Ok
5  DEFINT K,N,I,S:INPUT K:S=0:N=1
20 IF N>K THEN 70
30 FOR I=1 TO 0 STEP 1
40 S=S+N:N=N+1 'Т е л о  ц и к л а
50 I=N≤K
60 NEXT I
70 PRINT S;"Тестовый пример:";K*(K+1)/2
run
? 100
5050 Тестовый пример: 5050
Ok
```

Ясно,что в приведённой программе можно применить оператор цикла FOR…NEXT, поэтому рассмотрим пример, где использование оператора цикла FOR…NEXT не помогает.

Пример 2.
034-32.bas
034-32.bas

Пусть для данного положительного числа S требуется найти наибольшее натуральное K, такое, что

$S \geqslant 1^2+2^2+\ldots+K$ ^ .

```
Ok
10 DEFINT I,J:INPUT S:J=0
20 IF S≤0 THEN PRINT"S неположительно!":END
30 FOR I=1 TO 0 STEP 1
40 J=J+1:S=S-J^2
50 I=S>=0
60 NEXT I
70 PRINT J-1:END
run
? 23
 3
Ok

run
? 300
9
Ok
```

Из сопоставления примеров 1 и 2 ясно, что цикл «для» используется, как правило, когда количество исполнения цикла известно заранее. Если же число исполнений определить нельзя, если цикл должен быть прерван не по достижению параметром цикла конечного значения,а по какому–то другому условию (в примере 2: $S \leqslant 1^2+2^2+\ldots+k^2$), необходим цикл «пока».

Прежде чем решать задачу, подумай, что делать с её решением.

—Р.Хемминг

В науке часто недостаточно решить какую–нибудь задачу или группу задач. После этого нужно присмотреться к этим задачам и заново осмыслить, какие же вы задачи решили. Нередко, решая одну задачу,мы автоматически находим ответ и на другой вопрос, о котором раньше вовсе не думали.

—Н.Винер

Пример 3.
034-33.bas
034-33.bas
Составить программу нахождения суммы возрастающей арифметической прогрессии.

```
NEW
Ok
5 DEFINT I:INPUT A1,AN,D'A1-первый член арифметической прогрессии;AN-последний ее член;D-знаменатель
(D>0!)
7 IF D<0 OR A1>AN THEN ?"Прогрессия должна быть возрастающей! Повторите ввод!":GOTO 5
10 S=0:N=A1
20 IF N>AN THEN GOTO 70
30 FOR I=1 TO 0 STEP 1
40 S=S+N:N=N+D 'Тело цикла
50 I=N<=AN
60 NEXT I
70 ? S;"Тестовый пример:";((AN-A1)/D+1)*(A1+D/2*(AN-A1)/D)
run
? 1,20,1
```

```
210 Тестовый пример: 210
Ok
run
? 20,2,1
Прогрессия должна быть возрастающей! Повторите ввод!
?  …
```

Другой способ программирования на языке программирования MSX BASIC цикла WHILE…WEND предполагает использование *«опасного»* оператора GOTO:

```
10 GOTO 100
20 операторы   тела цикла
100 IF  условие  THEN GOTO 20
```

Если количество операторов тела цикла невелико, то допустима следующая конструкция:

```
10 IF  условие  THEN операторы  тела цикла:
GOTO 10 ELSE продолжение программы …
```

Если же количество операторов цикла велико, то можно операторы тела цикла поместить в подпрограмму (см. раздел IV.4.), вызываемую оператором GOSUB:

```
10 IF условие THEN GOSUB 1000:GOTO 10 ELSE продолжение программы
                …
1000 Тело цикла
                …
1200 RETURN
```

Пример 4. Вычислить сумму S знакочередующегося числового ряда: $\sum_{n=1}^{\infty} \frac{(-1)^n}{n^2}$ с точностью E.

Приведём три варианта программы:

- 4.1) 034-341.bas
  034-341.bas

```
NEW
Ok
10 INPUT E
20 S=0:N=1:A=(-1)^N/N^2
30 GOTO 50
40 S=S+A:N=N+1:A=(-1)^N/N^2
50 IF ABS(A)>=E THEN GOTO 40
60 PRINT "Результат:";S
run
? .00001
Результат:-.82246204205923
Ok
```

- 4.2) 034-342.bas
  034-342.bas

```
NEW
Ok
10 DEFINT I:INPUT E
20 FOR I=1 TO SQR(1/E)
30 S=S+(-1)^I/I^2
40 NEXT I
50 PRINT S
60 END
run
? .00001
```

```
-.82246204205923
Ok
```

- 4.3) 034-343.bas
  034-343.bas

```
NEW
Ok
10 INPUT E
20 S=0:N=1:A=(-1)^N/N^2
30 IF ABS(A)>=E THEN S=S+A:N=N+1:A=(-1)^N/N^2:GOTO 30 ELSE PRINT "Результат:";S:PRINT"Контроль:";-(4*ATN(1))^2/12:PRINT"Количество повторений цикла:";N-1 'Напомним, что значение π совпадает с 4*ATN(1)
run
? .1
Результат:-.86111111111111
Контроль: -.82246703342412
Количество повторений цикла: 3
Ok

run
? .01
Результат:-.81796217561099
Контроль: -.82246703342412
Количество повторений цикла: 10
Ok

run
? .001
Результат:-.82297055860543
Контроль: -.82246703342412
Количество повторений цикла: 31
Ok
```

Для проверки работоспособности программы мы воспользовались *тестовым* примером на основе известного математического результата: $\sum_{n=1}^{\infty}\frac{(-1)^n}{n^2}=-\frac{\pi^2}{12}$.

Наконец, отметим, что в данном примере количество повторений цикла существенно зависит от значения переменной E (однако, заранее количество повторений цикла нам неизвестно!).

Пример 5.

Написать программу для вычисления $\sqrt[n]{A}$, используя итерационную формулу Ньютона:
X(0)=A; X(k+1)=X(k)+(A/X(k)^(n-1)-X(k))/n, k=1,2,...

$$X(0)=A;X(k+1)=X(k)+\frac{\left(\frac{A}{X}(k)^{(n-1)}-X(k)\right)}{n},k=1,2,\ldots$$

Вычисления продолжать до тех пор, пока $\left|X(k+1)-X(k)\right|<E$, где $E>0$ — заданное число (впрочем, возможен другой вариант «останова» вычислений по выполнению условия $\left|X(K)^N-A\right|<E$).
034-35.bas
034-35.bas

```
Ok
10 DEFINT N:INPUT N,A,E
20 IF A<0ANDNMOD2=0 THEN ?"Арифметический корень четной степени из отрицательного числа не существует":END
30 X=A:Y=X+(X/A^(N-1)-X)/N:GOTO 50
```

```
40 X=Y:Y=X+(A/X^(N-1)-X)/N
50 IF ABS(Y-X)>=E THEN GOTO 40 ELSE ? "Результат:";Y
run
? 2,12,0.001
Результат: 3.4641016533502
Ok
```

А теперь — проверка:

```
Ok
print SQR(12)
3.4641016151377
Ok
```

...Поверил
Я алгеброй гармонию.

—А.С.Пушкин. Моцарт и Сальери

Обратимся теперь к лирике. Вот стихотворение Л.Мартынова (речь в нем идёт о девушке):

Кто геометрическое среднее между атомом и Солнцем?
Ты, первое и последнее воплощение красоты.
Не имеющая представления о строении вещества,
слушающая в изумлении эти непонятные слова.
Неспособная принять их к сведению,будучи ужасно молодой.
Вот ведь, какова ты, нечто среднее между атомом и звездой.

Проверим, прав ли поэт. Масса атома водорода Ма=1.67E-27 кг, масса Солнца Мс=1.99E+30 кг.Их произведение найдём в режиме непосредственного выполнения команд:

```
print 1.67E-27*1.99E+30
3323.3
Ok
```

Поэт утверждает, что масса девушки (в кг) равна квадратному корню из числа 3323.3 . А тогда, используя ранее написанную программу, получим:

```
run
? 2,3323.3,.001
            ▲
            |
 с точностью до грамма!
Результат: 57.648070219218
Ок
```

Результат вполне правдоподобен: вес девушки ≈ 58 кг.

III.5. Примеры

Бросая в воду камешки, смотри на круги, ими образуемые: иначе твоё занятие будет простой забавой.

—Козьма Прутков

Если я и открыл некоторые новые истины в науках, то я могу утверждать, что все они либо являются прямыми следствиями пяти или шести главных задач, которые мне удалось решить, либо зависят от них; я рассматриваю их как такое же число сражений, в которых военное счастье было на моей стороне.

—Рене Декарт

В заключение рассмотрим несколько, на наш взгляд, интересных примеров. Разумеется, некоторые из приведённых ниже программ не являются оптимальными по времени выполнения и по занимаемой памяти. Однако, есть ещё одна величина, которую надо принимать в расчёт при оптимизации программы. Это — *время* программиста. Программа, требующая много времени на написание и отладку,может оказаться очень дорогой, даже если она расходует меньше машинного времени и занимает меньше места в памяти, чем какая–то другая, в целом эквивалентная ей программа. По этой причине распространена практика написания простых и понятных, но относительно медленных программ (для них имеется жаргонное выражение «quick and dirty programs» – «скорые, но грязные программы»).

Если программу предлагается выполнить всего один–два раза, или если её придётся часто модифицировать, метод «скорых, но грязных программ» почти всегда даёт наилучшие результаты!

Пример 1. Найти наибольший элемент в одномерном массиве A(N), не пользуясь оператором `IF`(!).
03-01.bas
03-01.bas

```
Ok
10 DEFINT I,J,N:INPUT N:DIM A(N):X=RND(-TIME):FOR I=1 TO N:A(I)=INT(40*RND(1)):PRINT
A(I);:NEXT:M=1:PRINT
20 FOR I=1 TO N-1:FOR J=I+1 TO N
30 Y=-((A(I)>A(J))*A(I))-(A(I)<=A(J))*A(J):M=-(Y>M)*Y-(Y<=M)*M:NEXT J:
NEXT I:PRINT"Наибольший элемент:";M
run
? 6
 28  33  7  32  28  20
Наибольший элемент: 33
Ok
```

Пример 2. Написать программу, которая для введённой пользователем суммы печатает способ выдачи этой суммы наименьшим числом монет (имеются монеты 1,2,3,5,10,15 и 20 копеек).
03-02.bas
03-02.bas

```
NEW
Ok
20 DATA 20,15,10,5,3,2,1
30 PRINT "Введите сумму в копейках"
40 DEFINT S,I:INPUT S
50 IF S<>INT(S) OR S<1 THEN 20 ELSE PRINT"Разменяю-"
70 FOR I=1 TO 7:READ M:K=INT(S/M)
100 IF K>0 THEN PRINT K;"раз";M;"коп.":S=S-K*M ELSE S=S-K*M
120 NEXT I:END
run
Введите сумму в копейках
? 105
 Разменяю-
 5 раз 20 коп.
 1 раз 5 коп.
Ok

run
Введите сумму в копейках
? 555
 Разменяю-
```

```
 27 раз 20 коп.
 1 раз 15 коп.
Ok
```

Каждая решённая мною задача становилась образцом, который служил впоследствии для решения других задач.

—Рене Декарт. Рассуждения о методе

...создаётся впечатление, что можно построить целый курс программирования, выбирая примеры только из задач сортировки.

—Н.Вирт

Пример 3. Напишите программу, располагающую массив слов на английском языке по алфавиту (выполняющую лексикографическое упорядочение) методом *обменной сортировки* (методом «пузырька»).
03-03.bas
03-03.bas

```
Ok
10 DEFINT N,I,J:INPUT "Количество слов";N
20 DIM A$(N)
30 FOR I=1 TO N:INPUT A$(I):NEXT
40 FOR I=1 TO N-1:FOR J=I+1 TO N
50 IF A$(I)<=A$(J) THEN 60 ELSE SWAP A$(I),A$(J)
60 NEXT:NEXT
70 FOR I=1 TO N:PRINT A$(I):NEXT
run
Количество слов? 5
? Sedov
? Filippov
? Sokolova
? Shvetski
? Sedova
Filippov
Sedov
Sedova
Shvetski
Sokolova
Ok

run
Количество слов? 5
? Седов
? Филиппов
? Соколова
? Швецкий
? Седова
Филиппов
Седов
Седова
Соколова
Швецкий
Ok
```

Заметим,что коды ASCII букв русского алфавита не упорядочены в соответствии с алфавитом (например, не соблюдено условие «ВОВА»<«ГЕНА»), так как в компьютере буква «В» следует за буквой «Г» — в соответствии с латинскими буквами W и G, поэтому вышеприведённая программа не работает для слов, записанных русскими

буквами!

Следует отметить, что новый стандарт на отечественные персональные компьютеры, принятый в конце 1986 года, кардинально решает проблему – в нем коды русских букв возрастают в строгом соответствии с алфавитом (лексикографически упорядочены), и сортировка строковых переменных ничем не отличается от аналогичной процедуры над числовыми переменными. Но компьютеров с подобным кодированием русского алфавита, к сожалению, пока (1990 г.) очень мало!

Пример 4.
03-04.bas
03-04.bas

```
NEW
Ok
10 'Сортировка с применением индекса.
11 'Вместо переупорядочения самих значений в процессе сортировки можно образовать вспомогательный массив индексов, в котором отмечаются правильные места значений в массиве.
12 'Во время сортировки значения остаются на исходных местах, а изменяется лишь массив индексов. По окончанию сортировки массив индексов используется для копирования сортируемых значений в новый массив или служит справочником для работы с исходным массивом.
20 INPUT N:DIM A(N),I(N)
30  FOR M=1 TO N:INPUT A(M):NEXT M
40  FOR L=1 TO N:I(L)=L:NEXT
1000 'Фрагмент пузырьковой сортировки.
1040 FOR M=N TO 2 STEP -1
1050    FOR J=1 TO M-1
1052       I1=I(J):I2=I(J+1)
1060       IF A(I1)>=A(I2) THEN 1100
1070       I(J)=I2:I(J+1)=I1
1100    NEXT J
1110 NEXT M
1120 FOR L=1 TO N:I1=I(L):PRINT A(I1);:NEXT L
run
? 4
? 56
? 34
? 87
? 6
 87  56  34  6
Ok
```

Пример 5. В массиве X(1),X(2),... ,X(N) каждый элемент равен 0, 1 или 2. Переставьте элементы массива так, чтобы сначала располагались все нули, затем все единицы, и, наконец, все двойки. Дополнительного массива не вводить!
03-05.bas
03-05.bas

```
10 DEFINT N,I,J:INPUT N:DIM X(N):Z=RND(-TIME)
30 FOR I=1 TO N:X(I)=INT(3*RND(1)):?X(I);:NEXTI
40 FOR I=1 TO N-1:FOR J=I+1 TO N
50 IF X(I)<X(J) THEN GOTO 60 ELSE SWAP X(I),X(J)
60 NEXT:NEXT:?
70 FOR I=1 TO N:PRINT X(I);:NEXT
run
? 10
·0··1··0··0··0··0··2··0··2··2
·0··0··0··0··0··0··1··2··2··2
Ok
```

Программные строки 10–30 позволяют описать и идентифицировать массив X. Операторы в программных строках 40–60 упорядочивают элементы по возрастанию (если предыдущий элемент больше последующего,то они меняются местами). Операторы 70–й строки выводят на экран дисплея упорядоченный массив X.

Пример 6. Дан одномерный массив X(1),X(2),...,X(N). Все его элементы не равные нулю, переписать в начало массива, сохраняя порядок, а нулевые элементы записать в конец массива. Новый массив не заводить!

03-06.bas
03-06.bas

```
NEW
Ok
10 DEFINTN,I,J:INPUT"Введите N";N:DIM X(N):C=RND(-TIME)
20 FOR I=1 TO N:X(I)=INT(3*RND(1))
30  PRINT  X(I);:NEXT  I:PRINT'Сформирован массив псевдослучайных чисел для облегчения процесса тестирования программы!
50 FOR J=1 TO N-1:FOR I=1 TO N-1
60 IF X(I)=0 THEN SWAP X(I),X(I+1)
70 NEXT I,J
80 FOR I=1 TO N:PRINT X(I);:NEXT
run
Введите N? 8
·0··0··2··2··0··1··2··0
·2··2··1··2··0··0··0··0
Ok
```

Идея алгоритма проста! Надо менять местами элемент, равный нулю, и следующий за ним элемент. Подумайте, с какой целью в программе используется цикл с параметром цикла J?

Пример 7. Найти наибольший общий делитель(НОД) одномерного массива натуральных чисел A(N).
03-07.bas
03-07.bas

```
NEW
Ok
5 'Идентификация массива  A(N)
10 DEFINTN,A,I:INPUT"Введите N";N:DIMA(N)
15 PRINT "Для каких чисел ищется НОД?"
20 FOR I=1 TO N:INPUT A(I):NEXT
30 FOR I=1 TO N-1
35 'Далее находится НОД элементов  A(I)  и A(I+1)  с использованием алгоритма Евклида.
40 IF A(I)>A(I+1) THEN A(I)=A(I)-A(I+1)
50 IF A(I)<A(I+1) THEN A(I+1)=A(I+1)-A(I)
60 IF A(I)<>A(I+1) THEN GOTO 40
70 NEXT
75 'Последний вычисленный НОД выводится на экран.
80 PRINT A(I)
run
Введите N? 4
Для каких чисел ищется НОД?
? 125
? 25
? 625
? 1225
 25
Ok

run
Введите N? 4
Для каких чисел ищется НОД?
? 12
? 144
? 16
? 20
 4
Ok
```

Пример 8. Даны 5 цифр: 0, 1, 2, 3, 4. Найти сумму всех трёхзначных чисел, кратных 3, которые можно записать с помощью этих цифр (цифры могут повторяться).
03-08.bas

03-08.bas

```
NEW
Ok
10 DEFINT I-K,S:FOR I=1 TO 4:FOR J=0 TO 4:FOR K=0 TO 4
20 IF (I+J+K)MOD3=0 THEN S=S+I*100+J*10+K
30 NEXT K,J,I:PRINT"Искомая сумма=";S
run
Искомая сумма= 8937
Ok
```

Пример 9. Найти количество тех трёхзначных чисел, которые равны сумме факториалов своих цифр (понятием *подпрограммы* не пользоваться!).
03-09.bas
03-09.bas

```
NEW
Ok
10 DEFINT I,J:FOR I=100 TO 999
15 'Ищем факториал старшей цифры!
20 P=1:FOR J=1 TO I\100:P=P*J:NEXT J
25 'Ищем факториал второй цифры!
30 Y=1:FOR J=1 TO (IMOD100)\10:Y=Y*J:NEXT J
40 Z=1:A=(IMOD100)MOD10
45 'Ищем факториал третьей цифры!
50 FOR J=1 TO A:Z=Z*J:NEXT J
60 IF P+Y+Z=I THEN K=K+1:PRINT "Искомые числа:";I
70 NEXT I:PRINT"Количество таких чисел:";K
run
Искомые числа: 145
Количество таких чисел: 1
Ok
```

Верочка же, получив письмо (она его ждала раньше, ещё в марте), не сразу, а как–то помедлив, словно боясь чего–то, встала и, взяв со стола старинный костяной нож с инкрустацией (один перламутровый цветок давно выпал, чернело лишь гнездо на месте, куда он был вставлен), и вскрыла конверт неловкой рукой.

—И.Н.Горелов

Пример 10. Найти наименьшее трёхзначное число, кратное N, такое, что его первая цифра 7 и все его цифры различны. Программу уместить в одну программную строку!
03-10.bas
03-10.bas

```
NEW
Ok
1 'Пример "абсолютно нечитабельной" программы!
5
DEFINTN,I:INPUTN:Y$="Нет":FORI=700TO799:IFI\100<>(IMOD100)\10THENIFI\100<>IMOD100MOD10THENIF(IMO
D100)\10<>IMOD100MOD10THENIFIMODN=0THENY$="Есть!":CLS:PRINTY$,I:ENDELSEIFI=798THENCLS:PRINTY$:EN
DELSENEXTELSENEXTELSENEXTELSENEXT
run
? 5
Есть! 705
Ok

run
```

```
? 70
Нет!
Ok

run
? 18
Есть! 702
Ok

run
? 56
Есть! 728
Ok
```

Пример 11. Задан одномерный массив A, элементы которого обозначим A(1), A(2),..., A(N). Найдите длину самой длинной подпоследовательности подряд идущих элементов массива, равных 0.
03-11.bas
03-11.bas

```
NEW
Ok
10 DEFINT N,I,M,K:INPUT N:DIM A(N):X=RND(-TIME)
20 FOR I=1 TO N:A(I)=INT(2*RND(1)):PRINT A(I);:NEXT
40 FOR I=1 TO N
50 IF A(I)=0 THEN K=K+1:IF K>M THEN M=K:NEXT I ELSE NEXT I ELSE K=0:NEXT I
60 PRINT:PRINT"Максимальная длина:";M
run
? 8
·0··1··0··0··1··1··1··1
Максимальная длина:·2
Ok
```

Программные строки 10–20 позволяют описать массив X и идентифицировать его псевдослучайными целыми числами (для контроля его элементы выводятся на экран в строке 20). Операторы в программных строках 40–60 подсчитывают максимальное количество подряд идущих нулей в массиве и выводят результат на экран.

Пример 12. Найдите наибольшую по длине неубывающую подпоследовательность подряд идущих элементов в данной последовательности чисел (в одномерном массиве A(N)).
03-12.bas
03-12.bas

```
NEW
Ok
5 'Идентификация массива  A(N)  псевдослучайными целыми числами (программные строки 10-20).
10 DEFINT N,I,J,K,M:INPUT N:K=1:DIM A(N):X=RND(-TIME)
20 FOR I=1 TO N:A(I)=INT(12*RND(1)):PRINT A(I);:NEXT
30 'Выделение неубывающей подпоследовательности, сравнение ее длины с длиной предыдущей выделенной подпоследовательности; наибольшая длина запоминается в переменную  M  (строки 40-70).
40 FOR I=1 TO N-1
50 IF A(I)<=A(I+1) THEN K=K+1:IF K>M THEN M=K:J=I+1:NEXT I ELSE NEXT I ELSE K=1:NEXT I
60 ?:?"Наибольшая длина:";M:PRINT"Подпоследовательность:";
70 FOR I=J-M+1 TO J:PRINT A(I);:NEXT
Ok
run
? 8
·0··9··2··10··1··6··3··5
Наибольшая длина·2
Подпоследовательность:·0··9
Ok
```

Пример 13. Определите, сколько различных элементов встречается в одномерном массиве A(N) более, чем по одному разу.
03-13.bas

03-13.bas

```
NEW
Ok
5 INPUT"Введите N";N:DIM A(N):X=RND(-TIME)
8 ? "Исходный массив:"
10 FOR I=1TO N:A(I)=INT(10*RND(1)):PRINT A(I);:NEXT
20 FOR I=1 TO N-1:FOR J=I+1 TO N
23 IF A(I)=<A(J) THEN GOTO 27 ELSE SWAP A(I),A(J)
27 NEXT:NEXT :PRINT
30 FOR I=1 TO N-1
40 IF A(I)<>A(I+1) THEN P=0:NEXT I ELSE IF P=0 THEN K=K+1:P=1:NEXT ELSE NEXT 'P-флажок!
50 PRINT:PRINT K;"элементов в массиве встречаются более одного раза"
run
Введите N? 13
Исходный массив:
 0  4  7  6  7  5  6  1  2  0  5  2  3
 5 элементов в массиве встречаются более одного раза
Ok
```

Пример 14. Определить количество повторяющихся элементов в одномерном массиве A(N).
03-14.bas
03-14.bas

```
NEW
Ok
5  INPUT N:DIM A(N)
6  X=RND(-TIME)
7  FOR I=1 TO N:A(I)=INT(10*RND(1)):PRINT A(I);:NEXT:PRINT
10    S=0
20    FOR I=1 TO N
30       P=0
40       FOR J=1 TO N
50          IF A(I)=A(J) THEN P=P+1
60       NEXT J
70 IF P>1 THEN S=S+1
80 NEXT I:PRINT S
run
? 10
 3  7  5  6  9  6  9  4  5  0
 6
Ok

run
? 7
 9  2  7  3  0  0  1
 2
Ok
```

Пример 15. Из заданного на плоскости множества точек выбрать три такие, не лежащие на одной прямой, которые будут вершинами треугольника:

- наибольшей площади;
- наименьшего периметра.

03-15.bas
03-15.bas

```
NEW
Ok
20 DEFINT H,I-K,M:INPUT"Число точек";H:D=RND(-TIME):M=2*H
30 DIM A(M),B(M),C(M):IF H<3 THEN RUN 20
40 FOR I=1 TO M:A(I)=INT(190*RND(1)):B(I)=A(I):C(I)=A(I):NEXT:S=0:P=900 'Генерируем три одинаковых
```

```
массива координат точек плоскости случайным образом.
50 FOR I=2 TO M STEP 2
60 FOR J=2 TO M STEP 2
70 FOR K=2 TO M STEP 2
80  U=SQR((A(I)-B(J))^2+(A(I-1)-B(J-1))^2):Q=SQR((A(I)-C(K))^2+(A(I-1)-C(K-1))^2):W=SQR((B(J)-C(K))^2+(B(J-1)-C(K-1))^2):G=U+Q+W  'Нашли периметр треугольника.
90 V=SQR((G/2)*(G/2-U)*(G/2-Q)*(G/2-W)) 'Нашли его площадь.
100 IF V<1.E-10 THEN 130'Этим оператором гарантируем, что взяты три различные точки, не лежащие на одной прямой.
110 IF G<P THEN P=G:A1=A(I):A2=A(I-1):B1=B(J):B2=B(J-1):C1=C(K):C2=C(K-1)'Ищем треугольник с наименьшим периметром и запоминаем координаты его вершин.
120 IF V>S THEN S=V:A3=A(I):A4=A(I-1):B3=B(J):B4=B(J-1):C3=C(K):C4=C(K-1)'Ищем треугольник наибольшей площади и запоминаем координаты его вершин
130 NEXT K,J,I
140 PRINT"Вершины треугольника наиб.площади:(";A3;",";A4;"),(";B3;",";B4;"),(";C3;",";C4;");"
150 PRINT"Площадь треугольника:";S
160 PRINT"Вершины треугольника наим.периметра:(";A1;",";A2;"),(";B1;",";B2;"),(";C1;",";C2;");"
170 PRINT"Периметр треугольника:";P
180 FOR I=1 TO 5000:NEXT 'Задержка по времени (≈13 сек)!
185 'Графическая иллюстрация полученного результата.
190 SCREEN 2
200 FOR I=2 TO M STEP 2:PSET(A(I),A(I-1)),2:NEXT
210 LINE(A1,A2)-(B1,B2),1:LINE-(C1,C2),1:LINE-(A1,A2),1
220 LINE(A3,A4)-(B3,B4),2:LINE-(C3,C4),2:LINE-(A3,A4),2
230 GOTO 230'На экране изображены искомые треугольники!
run
Число точек? 5
Вершины треугольника наиб.площади:( 87 , 175 ),( 160 , 85),( 165 , 7 );
Площадь треугольника: 2622.0000000032
Вершины треугольника наим.периметра:( 142 , 57 ),( 165 , 7 ),( 161 , 24 );
Периметр треугольника: 110.57946634916
Далее следует графическая иллюстрация полученного решения.
```

Пример 16. Конечное множество из N точек в трёхмерном пространстве задано своими координатами: (X(N),Y(N),Z(N)), N=1,2,...,n. Составьте программу подсчёта количества тех точек данного множества, которые принадлежат:

- шару $X^2+y^2+z^2\le 25$;
- полупространству $x+y+z\le 3$;
- пересечению шара и полупространства;
- объединению шара и полупространства.

03-16.bas
03-16.bas

```
NEW
Ok5 'Идентификация массивов X,Y,Z псевдослучайными целыми числами (программные строки 10–30).
10 DEFINT N,I:INPUT "Количество точек";N
20 DIM X(N),Y(N),Z(N):U=RND(-TIME)
25 PRINT " X  Y  Z "
30 FOR I=1 TO N:X(I)=INT(11*RND(1)-5):Y(I)=INT(11*RND(1)-5):Z(I)=INT(11*RND(1)-5):PRINTX(I)Y(I)Z(I):NEXT
35 'Подсчет количества точек,лежащих в шаре (строки 40-60).
40 FOR I=1 TO N
50 IF(X(I)^2)+(Y(I)^2)+(Z(I)^2)<=25THENK=K+1:NEXTELSE NEXT
60 PRINT "Количество точек,лежащих в шаре:";K
65 'Подсчет количества точек, принадлежащих полупространству (строки 70-90).
70 FOR I=1 TO N
80 IF X(I)+Y(I)+Z(I)<=3 THEN C=C+1:NEXTI ELSE NEXTI
90 PRINT "Количество точек,принадлежащих полупространству:";C
95 'Подсчет количества точек,лежащих в шаре и одновременно в полупространстве (строки 100-120).
100 FOR I=1 TO N
110 IF X(I)+Y(I)+Z(I)=<3 AND (X(I))^2+(Y(I))^2+(Z(I))^2<=25 THEN B=B+1NEXT ELSE NEXT
120 PRINT "Количество точек,принадлежащих пересечению шара и полупространства:";B
```

```
125 'Подсчет количества точек,принадлежащих объединению шара и полупространства (строки 130-140)
130 P=C-B+K
140 PRINT "Количество точек,принадлежащих объединению шара и полупространства:";P
run
Количество точек? 5

 X  Y  Z
 0 -3  3
-4 -4  4
 4  5  4
-5  5  1
-1  4  4

Количество точек,лежащих в шаре: 1
Количество точек,принадлежащих полупространству: 3
Количество точек,принадлежащих пересечению шара и полупространства: 1
Количество точек,принадлежащих объединению шара и полупространства: 3
Ok
```

Пример 17. В заданном двухмерном массиве A(N,M) найти минимальное из чисел, встречающееся там более одного раза.
03-17.bas
03-17.bas

```
NEW
Ok
5 'Описание массива A(N,M) и заполнение его псевдослучайными целыми числами (строки 10-20).
10 DEFINT I,J,N,M,Y,K,L:INPUT N,M:DIM A(N,M):X=RND(-TIME)
20 FOR I=1 TO N:FOR J=1 TO M:A(I,J)=INT(10*RND(1)):PRINT A(I,J);:NEXT J:PRINT:NEXTI:IF N+M=2
THEN RUN10
30 'Идентификация дополнительного массива B (строки 40-50).
40 K=N*M:DIM B(K)
50 FORI=1TO N:FOR J=1 TO M:Y=M*(I-1)+J:B(Y)=A(I,J):NEXTJ,I
60 'Этот блок (строки 70-110) позволяет найти минимальное число элементов, встречающихся в массиве более одного
раза.
70 FOR Y=1 TO K-1:FOR L=Y+1 TO K
80 IF B(Y)<B(L) THEN 90 ELSE SWAP B(Y),B(L)
90 NEXT:NEXT
100 FOR I=1 TO K-1
110 IF B(I)=B(I+1) THEN PRINT"Минимальное число:";B(I) ELSE NEXTI:PRINT"Такого элемента нет!"
run
? 3,4
·3··0··3··9
·9··0··3··2
·9··3··9··3
Минимальное число: 0
Ok

run
? 1,2
·7··1
Такого элемента нет!
Ок
```

Пример 18. Определите, сколько различных элементов в данном двухмерном массиве A(N,M)?
03-18.bas
03-18.bas

```
NEW
Ok
5 'Идентификация массива A(N,M) псевдослучайными целыми числами и вывод его на экран дисплея для контроля
(строки 10-20).
10 DEFINT N,M,I,J,Y,K,L:INPUT N,M:DIM A(N,M):X=RND(-TIME)
20 FOR I=1 TO N:FOR J=1 TO M:A(I,J)=INT(10*RND(1)):?A(I,J);:NEXT J:?:NEXTI:IF N+M=2 THEN ?"Один-
```

```
одинешенек!":END
30 'Идентификация дополнительного массива  B  (строки 40-50).
40 K=N*M:DIM B(K)
50 FOR I=1 TON:FOR J=1 TOM:Y=M*(I-1)+J:B(Y)=A(I,J):NEXTJ,I
65'Упорядочивание элементов в массиве  B(строки 70-90)
70 FOR Y=1 TO K-1:FOR L=Y+1 TO K
80 IF B(Y)<B(L) THEN 90 ELSE SWAP B(Y),B(L)
90 NEXT:NEXT
93 'Подсчет количества различных элементов в массиве  B  и вывод результата на экран дисплея (строки 95-120).
95 U=1
100 FOR I=1 TO K-1
110 IF B(I)<>B(I+1) THEN U=U+1:NEXT I ELSE NEXT I
120 PRINT"Количество разл.элементов:";U
run
? 3,3
·9··6··9
·3··9··4
·2··1··8
Количество разл.элементов:·7
Ok

run
? 3,4
·1··5··3··5
·8··9··7··5
·0··4··4··9
Количество разл.элементов:·8
Ok
```

Пример 19. Проверьте, есть ли в двухмерном массиве C(M,N) отрицательные элементы, если есть, найдите среди них тот, у которого сумма индексов будет наименьшей.
03-19.bas
03-19.bas

```
NEW
Ok
5 'Идентификация массива  A(M,N)  (строки 10-30).
10 DEFINT M,N,I,J,K:INPUT M,N:DIM A(M,N):K=N+M:X=RND(-TIME)
20 FOR I=1 TO M:FOR J=1 TO N
30 A(I,J)=INT(11*RND(1)-5):PRINT A(I,J);:NEXTJ:?:NEXT I
45 IF M=1 AND N=1 THEN IF A(1,1)<0 THEN ?"Искомый элемент:";A(1,1):END ELSE ?"Такого элемента нет!":END'Случай  M=N=1
47 'Нахождение отрицательного элемента массива  A,  у которого сумма индексов минимальна (строки 50-70).
50 FOR I=1 TO M:FOR J=1 TO N
60 IF A(I,J)<0 THEN IF I+J<K THEN K=I+J:C=A(I,J):NEXTJ:NEXTI ELSE NEXT J:NEXT I ELSE NEXT J:NEXT I
70 PRINT"Искомый отрицат.элемент:";C
run
? 2,5
·0··2··3·-3··3
-5··5·-4··3·-5
Искомый отрицат.элемент:-5
Ok

run
? 2,3
-2··1··5
·1··3··4
Искомый отрицат.элемент:-2
Ok
```

Пример 20. Операция сглаживания матрицы даёт новую матрицу того же размера, каждый элемент которой получается как среднее арифметическое имеющихся соседей соответствующего элемента исходной матрицы. Построить результат сглаживания заданной вещественной матрицы A размером N×N.

03-20.bas
03-20.bas

```
NEW
Ok
10 INPUT"FORMAT";A$
15  'FORMAT-строковое выражение, состоящее из символов "#" и десятичной точки (например, ##.###);
используется в операторе  ''PRINTUSING''  (строка 210) для вывода элементов массива с требуемой точностью.
16 'Точность определяется количеством символов "#", стоящих правее десятичной точки.
20 DEFINT N,I,J:INPUT"Число строк (столбцов)";N
30 DIM A(N,N),B(N,N):K=RND(-TIME)
35 'Формирование матрицы  A,  элементами которой являются случайные числа (для облегчения тестирования
программы!).
40 FOR I=1 TO N:FOR J=1 TO N
50 A(I,J)=INT(10*RND(1)):PRINT A(I,J);
60 NEXT J:PRINT:NEXT I:PRINT"Результат сглаживания"
70 IF N=2 THEN 160 ELSE IF N=1 THEN ?"Спорный вопрос!":END
75 'Программные строки 80-190 позволяют сгладить матрицу  A(N,N).Результат сглаживания - матрица  B(N,N).
80 FOR I=2 TO N-1:FOR J=2 TO N-1
90 B(I,J)=(A(I-1,J)+A(I,J-1)+A(I,J+1)+A(I+1,J))/4:NEXT J,I
100 FOR J=2 TO N-1
110 B(1,J)=(A(1,J-1)+A(2,J)+A(1,J+1))/3
120 B(N,J)=(A(N,J-1)+A(N-1,J)+A(N,J+1))/3:NEXT J
130 FOR I=2 TO N-1
140 B(I,1)=(A(I-1,1)+A(I,2)+A(I+1,1))/3
150 B(I,N)=(A(I-1,N)+A(I,N-1)+A(I+1,N))/3:NEXT I
160 B(1,1)=(A(1,2)+A(2,1))/2:B(1,N)=(A(1,N-1)+A(2,N))/2
180 B(N,1)=(A(N-1,1)+A(N,2))/2:B(N,N)=(A(N-1,N)+A(N,N-1))/2
200 FOR I=1 TO N:FOR J=1 TO N
210 PRINT USING A$;B(I,J);:NEXT J:?:NEXT I
run
FORMAT? ###.##
Число строк (столбцов)? 3
·2··0··7
·9··9··9
·1··7··1
Результат сглаживания
··4.50··6.00··4.50
··4.00··6.25··5.67
··8.00··3.67··8.00
Ok
```

Пример 21. Для заданной целочисленной матрицы найдите максимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы размером M×M.
03-21.bas
03-21.bas

```
NEW
Ok
5 'Идентификация массива  A(M,M)  (строки 10-30).
10 DEFINT M,I,J,K:INPUT M:IF M<=1 THEN 10
20 DIM A(M,M):X=RND(-TIME)
30 FOR I=1 TO M:FOR J=1 TO M:A(I,J)=INT(10*RND(1)):PRINT A(I,J);:NEXTJ:PRINT:NEXT I
35 'Нахождение максимальной среди сумм элементов диагоналей, расположенных выше главной (строки 40-50).
40 FOR K=1 TO M-1:FOR I=1 TO M:FOR J=1 TO M
50 IF J=I+K THEN B=B+A(I,J):NEXT J:NEXT I ELSE NEXTJ:NEXTI:IF B>C THEN C=B:B=0:NEXT K ELSE
B=0:NEXT K
55 'Нахождение максимальной среди сумм элементов диагоналей, расположенных ниже главной (строки 60-70).
60 FOR K=1 TO M-1:FOR I=1 TO M:FOR J=1 TO M
70 IF J=I-K THEN B=B+A(I,J):NEXT J:NEXTI ELSE NEXTJ:NEXTI:IF B>D THEN D=B:B=0:NEXT K ELSE
B=0:NEXT K
80 IF C>D THEN PRINT C ELSE PRINT D 'Сравнение этих максимумов и вывод большего на экран дисплея.
run
? 4
```

```
·5··8··4··8
·0··3··2··7
·6··6··7··3
·7··6··6··9
·13
Ok

run
? 3
·3··6··6
·8··0··2
·3··6··2
·14
Ok

run
? 5
·2··0··5··7··7
·9··6··7··2··9
·9··9··8··5··5
·6··9··1··1··7
·7··3··9··8··3
·27
Ok
```

Пример 22. Вычислить сумму тех элементов двухмерного целочисленного массива A(M,N), в десятичной записи которых используются только цифры 0, 1, 2, 3.
03-22.bas
03-22.bas

```
NEW
Ok
10 DEFINT M,N,I,J,K,A:INPUT"Введите M и N";M,N
Кто геометрическое среднее между атомом и Солнцем? \\
20 INPUT"Числовой отрезок[X,Y]";X,Y
30 Z=RND(-TIME):DIM B(M,N),A(M,N)
40 'Заполнение массива B(M,N) псевдослучайными целыми числами из отрезка[X,Y](строки 50-70)
50 FOR I=1 TO M:FOR J=1 TO N:B(I,J)=INT((Y-X+1)*RND(1)+X):A(I,J)=B(I,J):NEXT J,I
80 S=0:FOR I=1 TO M:FOR J=1 TO N
90 'Пока не просмотрим все цифры числа A(i,j),выполняем цикл "пока"(строки 100-160)
100 IF A(I,J)=0 THEN 180 ELSE FOR K=1 TO 0 STEP 1
120 B=ABS(A(I,J))-INT(ABS(A(I,J))/10)*10 'Выделяем последнюю цифру числа A(i,j)
130 IF B>=4 THEN 180 ELSE A(I,J)=(ABS(A(I,J))-B)/10
150 K=A(I,J) 'проверка условия: если A(i,j)=0,то из цикла "пока" выходим
160 NEXT K
170 IF A(I,J)=0THEN S=S+B(I,J)'Ищем сумму нужных элементов
180 NEXT J,I:FOR I=1 TO M:FOR J=1 TO N
210 PRINT B(I,J);:NEXT J:PRINT:NEXT I:?"Искомая сумма:";S
run
Введите M и N? 4,6
Числовой отрезок[X,Y]? 1,9
9  4  9  3  5  6
3  1  6  4  3  3
1  9  2  3  9  1
3  8  4  8  9  2
Искомая сумма: 25
Ok
```

Пример 23. Подсчитайте количество элементов двухмерного массива A(M,N) натуральных чисел, которые являются составными.
03-23.bas
03-23.bas

```
NEW
```

```
Ok
10 DEFINT M,N,I,J,A,K:INPUT"Введите M и N";M,N:X=RND(-TIME):DIM A(M,N):IF M<1 OR N<1 THEN RUN 10
30 FOR I=1 TO M:FOR J=1 TO N
40 A(I,J)=INT(30*RND(1)+1):IF A(I,J)=2 THEN PRINTUSING"\  \";STR$(A(I,J))+"*";:NEXT:PRINT:NEXT
50 K=0:FOR K=2 TO A(I,J)-1
60 IF A(I,J)/K<>FIX(A(I,J)/K) THEN NEXT K:PRINTUSING"\  \";STR$(A(I,J))+"*";:NEXT J:PRINT:NEXT I
ELSE Y=Y+1:PRINT USING"\  \";STR$(A(I,J))+" ";:NEXT J:PRINT:NEXT I
70 PRINT"Звездочками отмечены простые числа;составных чисел в Вашем массиве";Y
run
Введите M и N? 3,5
·27··20··25··0···2*
·26··17*·23*·2*··23*
·20··12··22··9···21
Звездочками отмечены простые числа;составных чисел в Вашем массиве·10
Ok
```

В этой программе во время идентификации массива A(M,N) случайными числами сразу же производится проверка каждого элемента массива: является ли он простым? Если «да»,то на экран этот элемент выводится со «звёздочкой» («∗») позади. Если «нет», то — без неё. Затем подсчитывается количество составных чисел и результат выводится на экран.

Пример 24. *Характеристикой* столбца целочисленной матрицы назовём сумму модулей её отрицательных чётных элементов. Переставляя столбцы заданной матрицы, расположить их в соответствии с ростом характеристик.
03-24.bas
03-24.bas

```
NEW
Ok
10 DEFINT M,N,I,J,W,K,B,D:INPUT"Укажите M,N";M,N
15 IF M<2 OR N<2 THEN PRINT "Слишком малое количество столбцов (строк)!":RUN 10
20 INPUT"Числовой отрезок[X,Y]";X,Y
30 Z=RND(-TIME):DIM A[M,N],Q[M,N],L[N],F[N]
40 'Заполняем массив A[M,N] случайными целыми числами из отрезка [X,Y] и копируем его в массив
Q[M,N](строки 50-70)
50 FOR I=1 TO M:FOR J=1 TO N
60 A[I,J]=INT((Y-X+1)*RND(1)+X):Q[I,J]=A[I,J]
70 NEXT J,I
80 'Находим характеристики столбцов(строки 90-110)
90 FOR J=1 TO N:FOR I=1 TO M
100 IF A[I,J]<0 AND ABS(A[I,J])MOD2=0 THEN L[J]=L[J]+ABS(A[I,J])
110 F[J]=L[J]:NEXT I,J
120 'Упорядочиваем столбцы по возрастанию характеристик(строки 130-180)
130 FOR W=1 TO N-1 'Удивительный цикл!
140 FOR B=1 TO N-1:FOR K=B+1 TO N
150 IF L[B]<=L[K] THEN 180 ELSE SWAP L[B],L[K]
170 FOR D=1 TO M:SWAP A[D,B],A[D,K]:NEXT D
180 NEXT K,B,W
190 'Вывод результатов
200 FOR I=1 TO M
210 FOR J=1 TO N:PRINT USING"####";Q[I,J];:NEXTJ:? SPC(4)
220 FOR J=1 TO N:PRINT USING"####";A[I,J];:NEXT J:PRINT
230 NEXT I
240 FOR H=0 TO 30:PRINT "-";:NEXT H:PRINT
250 'А теперь заполняем строки характеристик!
260 FOR J=1 TO N:PRINT USING"####";F[J];:NEXT:? SPC(4)
270 FOR J=1 TO N:PRINT USING"####";L[J];:NEXT
run
Укажите M,N? 4,3
Числовой отрезок[X,Y]? -40,10
 -17 -18  -6      -6 -17 -18
 -23 -16   1       1 -23 -16
   1 -38   7       7   1 -38
 -12  -5  -1      -1 -12  -5
------------------------------
```

```
 12 72  6    6 12 72   ◄──── строки характеристик
Ok
```

Подумайте, каким был бы результат тестового примера, если бы в программе отсутствовал цикл с параметром W (строки 130–180).

Пример 25. Вычисление определённого интеграла от функции, заданной на отрезке [-1,1], по квадратурной формуле Гаусса.
03-25.bas
03-25.bas

```
NEW
Ok
10 DEFINT N,I:DEF FNF(Z)=Z*SIN(Z)
20 INPUT"Порядок интегрирования=";N:DIM X[N],H[N]
40 IF N=2 THEN RESTORE 160
50 IF N=3 THEN RESTORE 170
60 IF N=4 THEN RESTORE 180
70 IF N=5 THEN RESTORE 190
80 FOR I=1 TO N:READ X[I]:NEXT
90 IF N=2 THEN RESTORE 200
100 IF N=3 THEN RESTORE 210
110 IF N=4 THEN RESTORE 220
120 IF N=5 THEN RESTORE 230
130 FOR I=1 TO N:READ H[I]:NEXT:J=0
140 FOR I=1 TO N:J=J+FNF(X[I])*H[I]
150 NEXT:PRINT "Интеграл=";J:END
155 'Коэффициенты метода Гаусса.
160 DATA-.57735026919, .57735026919
170 DATA-.774596669241,0 , .774596669241
180 DATA-.861136311594,-.339981043585,.339981043585,.861136311594
190 DATA-.906179845939,-.538469310106,0,.538469310106,.906179845939
200 DATA 1,1
210 DATA .55555555555556,.88888888888889,.55555555555556
220 DATA .347854845137,.652145154863,.652145154863,.347854845137
230 DATA .236926885056,.478628670499,.56888888888889,.478628670499,.236926885056
run
Порядок интегрирования=? 2
Интеграл= .6302420371144
Ok

run
Порядок интегрирования=? 4
Интеграл= .6023395913427
Ok
```

Диск с примерами

Загрузить образ диска

Открыть диск в WebMSX
