Глава IV. Функции и подпрограммы

> Любая по-настоящему полезная классификация содержит от трёх до шести категорий.
>
> *—Энон*

В MSX BASIC различают следующие основные типы функций:

1. встроенные функции:
    1. встроенные числовые функции;
    2. встроенные функции преобразования;
    3. встроенные функции для работы со строками;
2. функции пользователя.

Встроенные функции называют иначе *стандартными* функциями, вычисление значений которых реализуется специальными программами, постоянно находящимися в памяти компьютера. Каждая такая программа имеет уникальное *имя*, по которому и происходит обращение к ней (`SIN`, `LOG`, `VAL`, `MID$` и т.д.).

К функциям *пользователя* относятся те функции, которые пользователь определяет в программе сам.

Заметим, что встроенные *числовые* функции мы уже изучили (см. раздел I.7.4.)!

IV.1 Встроенные функции преобразования

Напомним, что множество значений строковых переменных — это множество упорядоченных наборов символов. Упорядоченность означает, что строковые значения различаются не только набором, но и последовательностью символов, например, «краб» и «брак» — это разные значения. Количество символов в значении строковой переменной меняется в пределах от 0 до 255.

В дальнейшем, *строкой* `A$` или *словом* `A$` будем называть значение строковой переменной `A$`.

Обратим особое внимание на то, что строка может не содержать символов. Такая строка называется *пустой* и обозначается "" (кавычки идут подряд). Длина этой строки равна 0. Не следует путать пустую строку со строкой, содержащей *пустой* символ (пробел). Строка, содержащая пробел, обозначается " " (между кавычками есть пробел) и имеет длину 1. Отметим, что символ «пробел» — равноправный с остальными символ!

IV.1.1. LEN–функция

Её общий вид:

```
LEN(α)
```
,

где:

- `LEN` («LENgth» — «длина») — служебное слово;
- α — строковое выражение.

Результатом функции LEN(α) является длина значения строкового выражения α (или что то же самое, количество символов в строке α), LEN(α)∈[0,255].

Напомним, что в том месте дисплейной строки, где надо обратить Ваше внимание на наличие символа пробел (« »), мы будем использовать символ «·»[1)].

Пример 1.
0411-01.bas
0411-01.bas

```
NEW
Ok
10 X$="парк": Y$="детский парк":Z$=" ау! ":u$="":w$=" "
20 PRINT LEN(X$);LEN(Y$);LEN(Z$);len(u$);len(w$)
run
·4··12··5··0··1
Ok
```

Ещё раз подчеркнём, что *пробел* — такой же символ, как, например «а»!

Эта функция, пожалуй, чаще всего используется при работе со строковыми переменными. С её помощью, например, можно определить, не является ли строка пустой до выполнения действий, которые не могут выполняться над пустой строкой ("").

Пример 2.
0411-02.bas
0411-02.bas

```
10 IF LEN(A$)>0 THEN PRINT "Все в порядке!" ELSE PRINT "Строка пустая!" 'Проверка, является ли строка пустой, т.е. не содержащей ни одного символа.
```

Пример 3.
0411-03.bas
0411-03.bas

```
NEW
Ok
10 'Выравнивание слов по правому краю.
20 FOR I=1 TO 3:READ A$
30 ?TAB(20-LEN(A$));A$:NEXTI
40 DATA "Хакер-","фанатичный","программист"
run
··············Хакер-
···········фанатичный
·········программист
Ok
```

Сравните функцию `LEN()` с функцией длин(A), где длин(A) — длина текста A, в школьном алгоритмическом языке [11, с.53]!

IV.1.2. INSTR–функция

Общий вид функции:

```
INSTR([n,]α,β)
```
,

где: `INSTR`(«IN STRing» — «в строке») — служебное слово;

- n — арифметическое выражение, целая часть значения которого должна принадлежать [1,255] (по умолчанию n=1);
- α — строковое выражение;
- β — строковое выражение.

Эта функция исследует значение строкового выражения α слева направо, начиная с символа, номер которого задан величиной целой части значения параметра n или с его первого символа (по умолчанию). Значением функции является номер позиции в значении α, с которой значение β первый раз встречается в α. Если значение β не найдено, то результатом функции `INSTR` является 0 (как говорят, «функция `INSTR` возвращает значение 0»). Значением выражения α или β может быть пустая строка.

При работе с функцией `INSTR` возможны следующие ошибки. Сообщение:

«Illegal function call»
(*«Неправильный вызов функции»*)

возникает, если значение арифметического выражения n неположительно, а сообщение:

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

если α или β нестрокового типа. Это сообщение выдаётся очень часто при работе с функциями преобразования и с функциями для работы со строками; как правило, причиной этого служит пропущенный знак «$» после имени строковой переменной или имени строковой функции.

Пример.

```
NEW
Ok
10 INPUT X$,Y$,N
20 PRINT INSTR(N,X$,Y$)
run
? ХАФАНАНА,НА,3
 5
Ok

run
? ХАФАНАНА,НА,100
 0
Ok

run
? "","",5
 0
Ok

run
? KEN,"",2
 2
Ok

run
? Кузьма Кузьмич,Кузьм,1.5
 1
Ok

run
? Кузьма Кузмич,Кузьм,0
Illegal function call in 20
Ok
```

IV.1.3. VAL–функция

Цифровая строка — это строка, содержащая любое количество расположенных в любом месте строки пробелов, знак «плюс» или «минус», десятичные цифры и десятичную точку.

Цифровая строка также может содержать:

1. запись числа в экспоненциальном представлении (в форме с плавающей точкой);
2. запись числа в двоичном, восьмеричном и шестнадцатеричном представлениях; в этих случаях цифровая строка называется *двоичной, восьмеричной и шестнадцатеричной* цифровой (символьной) строкой соответственно.

В ряде случаев удобнее работать с числами, а не с их представлениями в виде цифровых строк. Цифровую строку можно преобразовать в числовое значение с помощью функции преобразования VAL, общий вид которой:

```
VAL(α)
```
,

где: VAL(«VALue» — «значение») — служебное слово;

- α — строковое выражение.

Преобразование начинается с крайнего левого символа значения строкового выражения α и заканчивается, когда преобразован последний символ значения α, либо когда встретился нецифровой символ.

Примеры:

```
Ok
print VAL("12.5E-3")
.0125
Ok

Ok
print val("&b1");val("&o76");val("&hF")
·1··62··15
Ok
```

Если строка является пустой или начинается с нецифрового символа, то функция VAL возвращает значение 0.

Примеры:

- 1)

```
 Ok
? VAL("a12");VAL("23A12:");VAL("")
·0··23··0
Ok
```

- 2)

```
 Ok
? VAL("-1.24")
-1.24
Ok
```

Если одним из символов цифровой строки является восклицательный знак, то в зависимости от его расположения возможны следующие случаи.

Примеры:

- 1)

```
Ok
? VAL("1234.56789!")
·1234.57
Ok
```

  результат преобразования: число одинарной точности (шесть значащих цифр!)

- 2)

```
Ok
   ? VAL("34!5.1")
   ·34
   Ok
```

- 3)

```
Ok
```

```
? VAL("3456789!.7")
·3456790
Ok
```

- 4)

```
Ok
? VAL("!1")
·0
Ok
```

И наконец, парочка патологических случаев:

- 5)

```
Ok
? VAL("&")
Syntax error
Ok
```

- 6)

```
 Ok
? VAL("&H")
 0
Ok
```

Таким образом, функция VAL позволяет выделить цифры, входящие в значение строкового выражения, и образовать из них число для последующей математической обработки.

IV.1.4. STR$–функция

Часто бывает необходимо осуществить преобразование числа в цифровую строку, например, число 1234.56 преобразовать в цифровую строку "1234.56" .

Это преобразование осуществляет функция STR$, общий вид которой:

```
STR$(α)
```
,

где:

- STR («convert to STRing» — «преобразовать в строку») — служебное слово;
- α — арифметическое выражение.

Данная функция преобразует значение арифметического выражения α в цифровую строку, что позволяет выделять и обрабатывать каждый символ (цифру) полученной строки с помощью *строковых* (!) функций.

Примеры

- 1)

```
Ok
print STR$(-15)
-15
Ok
```

- 2)

```
Ok
print STR$(1/3)
·.33333333333333
Ok
```

- 3)

```
Ok
? STR$(1.E-3)
·1E-03
Ok
```

- 4)

```
Ok
? STR$(&b111);STR$(&o23);STR$(&HF1)
·7·19·241
Ok
```

Обратите *особое* внимание на следующие примеры (первый символ в получаемых цифровых строках зарезервирован для указания знака числа):

- 5)

```
Ok
? LEN(STR$(20))
3
Ok
```

- 6)

```
Ok
? LEN(STR$(-56.20))
5
Ok
```

- 7)

```
Ok
? LEN(STR$(-5620))
5
Ok
```

- 8)

```
Ok
? LEN(STR$(1.2E-27)
8
Ok
```

Ещё один пример использования функции STR$: оператор

```
PRINT n;"X"
```

выводит на экран дисплея пробел между последней цифрой значения арифметического выражения n и символом «X». Для исключения этого пробела воспользуйтесь оператором

```
PRINT STR$(n);"X"
```

- 9)

```
NEW
Ok
10 INPUT N
20 PRINT N;"i"
30 ? STR$(N);"i"
run
? 10
10 i
10i
Ok
```

- 10)

```
NEW
Ok
```

```
10 INPUT N
20 PRINT "i";n
30 ?"i";MID$(STR$(N),2)
run
? 10
i 10
i10
Ok
```

(о строковой функции `MID$()` см. в разделе IV.2.1.).

Отметим, что обычно в программах функции `VAL()` и `STR$()` используются совместно.

- 11)

```
Ok
input N$:print STR$(VAL(N$))
? 23123.45
·23123.45
Ok

? -4.7
-4.7
Ok

? &HFF
·255
Ok

однако...
? 12ABBA
·12
Ok
```

- 12)

```
Ok
input N:print VAL(STR$(N))
? 12
·12
Ok

? -14.6E-2
-.146
Ok

? &b111
·7

Ok

? &o21
·17
Ok

? &h1E
·30
Ok
```

- 13) найдите натуральные числа, не превосходящие заданного N и равные сумме кубов своих цифр.
0414-13.bas
 0414-13.bas

```
NEW
Ok
10 INPUT N
20 FOR I=1 TO N
30 A$=STR$(I):S=0
40 FOR J=2 TO LEN(A$)
50 S=S+VAL(MID$(A$,J,1))^3
```

```
60 NEXT J
70 IF S=I THEN PRINTI;:NEXT I ELSE NEXT I
run
? 400
·1··153··370··371
Ok
```

- 14) определите наибольшую из цифр, используемых в десятичной записи натурального числа M.
  0414-14.bas
  0414-14.bas

```
NEW
Ok
10 INPUT I
20 A$=STR$(I)
30 M=VAL(MID$(A$,2,1))
40 FOR J=2 TO LEN(A$)-1
50 IFM<VAL(MID$(A$,J+1,1))THEN M=VAL(MID$(A$,J+1,1)):NEXTJ ELSE NEXTJ
60 PRINT M
run
? 789103
 9
Ok

run
? 7777777777777777
 8         ▲
Ok         │
    16 символов
```

- 15) найти наибольшую из цифр, встречающуюся в десятичной записи данного натурального числа M более одного раза.
  0414-15.bas
  0414-15.bas

```
NEW
Ok
10 INPUT M
20 N$=STR$(M)
30 FOR I=2 TO LEN(N$)-1
40 FOR J=I+1 TO LEN(N$)
50 IF MID$(N$,I,1)=MID$(N$,J,1) THEN IF VAL(MID$(N$,I,1))>A THEN A=VAL(MID$(N$,I,1)):NEXTI ELSE ELSE
NEXTJ:NEXTI
60 PRINT A
run
? 1234245
 4
Ok

run
? 1243.67E34
 7
Ok

Последний результат вызывает удивление, не правда ли?
Однако...
print n$
 1.24367E+37
Ok
и все становится ясным!
```

IV.1.5. ASC–функция

Напомним, что при вводе в ЭВМ символы преобразуются в соответствии с кодом ASCII (см. раздел I.7.3.). Функция

```
ASC(α)
```
,

где:

- ASC(«ASC» — «American Standard Code») — служебное слово;
- α — строковое выражение,

даёт возможность установить десятичный код ASCII первого символа значения строкового выражения α. Результатом функции ASC является целое число из отрезка [0,255] (говорят, что функция ASC *возвращает* целое число из отрезка [0,255]).

Пример.

```
ASC
NEW
Ok
10 INPUT X$:Y=ASC(X$):PRINT X$;Y
run
? π
π 163
Ok

run
? αδβ
αδβ 160
Ok

run
? ;
; 59
Ok

run
? ","
, 44
Ok

run
? ♪ ◄ символ,
♪ 1   имеющий
Ok      двухбайтовый код
```

Из приведённого примера ясно, что символ «π», например, имеет десятичный код 163, а символ «α»(первый символ значения строковой константы«αδβ») — десятичный код 160.

Далее, если значением строкового выражения α является пустая строка (""), то фиксируется ошибка

«Illegal function call».

Используя предыдущий пример, попытаемся получить код символа " (кавычка).

```
run
? "
Illegal function call in 10
Ok
Неудача!... Попытаемся ещё разок...
run
? """
? Redo from start
?▉  и т.д.
```

Снова неудача! Не волнуйтесь, код символа «кавычка» можно получить косвенным путём с помощью функции

преобразования CHR$.

IV.1.6. CHR$-функция

Общий вид функции:

```
CHR$(α)
```
,

где:

- CHR(«CHaRacter» — «символ») — служебное слово;
- α — арифметическое выражение.

Вначале компьютер вычисляет *код* — целую часть значения выражения α. Код должен принадлежать отрезку [0,255], иначе на экране появится сообщение об ошибке:

«Illegal function call».

Результатом функции CHR$ является однобайтовая строка, содержащая символ, соответствующий полученному коду (говорят, что функция CHR$ *возвращает* однобайтовую строку, содержащую символ, соответствующий полученному коду).

Примеры:

- 1)

```
INPUT X:Y$=CHR$(X):PRINT X;Y$
? 34
  34 "
Ok

? 187
  187 √
Ok

? 163
 163 π
Ok
```

- 2) переменной A$ присвоить значение «A$="BANSAJ!"»

```
A$="A$="+CHR$(34)+"BANSAJ!"+CHR$(34)
А теперь...
print A$
A$="BANSAJ!"
Ok
```

- 3) вывести на экран дисплея текст: «Нажмите клавишу "SHIFT"+"1"».

```
Ok
PRINT "Нажмите клавишу ";CHR$(34);"SHIFT";CHR$(34);"+";CHR$(34);"1";CHR$(34)
Нажмите клавишу "SHIFT"+"1"
Ok
```

- 4) инициализация строкового массива B$(N) «псевдослучайными» словами
0416-04.bas
0416-04.bas

```
Ok
20 Z=RND(-TIME)
30 INPUT "Введите наибольшее количество символов слова";G
40 INPUT"Введите количество слов";N:DIM B$(N)
```

```
60 FOR K=1 TO N:A=INT((G+1)*RND(1))
80 FOR I=1 TO A:B$(K)=B$(K)+CHR$(INT(255*RND(1)))
100 NEXT I,K
110 FOR K=1 TO N:PRINT K;"слово: ";B$(K):NEXT
run

Введите наибольшее количество символов слова? 4
Введите количество слов? 2
1 слово: ▲щГ
2 слово: X∞p
Ok
```

Функции ASC() и CHR$() являются взаимообратными, то есть X=ASC(CHR$(X)) и Y$=CHR$(ASC(Y$)), если только значение арифметического выражения X находится в допустимых пределах (напомним, что 0≤X≤255!), а LEN(Y$)=1.

- 5) присвоить строковой переменной Y$ значение «yes», если X≥1 и «no» — если X<1. Оператор условного перехода IF…THEN…ELSE… не применять!
  0416-05.bas
  0416-05.bas

```
NEW
Ok
10 INPUT X
20         Y$=CHR$(-(X>=1)*ASC("y"))+CHR$(-(X>=1)*ASC("e"))+CHR$(-(X>=1)*ASC("s"))+CHR$(-
(X<1)*ASC("n"))+CHR$(-(X<1)*ASC("o")):print  Y$
run
? 5
yes
Ok

run
? -1
no
Ok
```

Наконец заметим, что функцию CHR$ удобно использовать для работы с непечатаемыми «символами» (BS, SELECT, Ввод ⏎, клавиши управления курсором).

IV.1.7. BIN$-функция

BIN$-функция применяется для преобразования целого числа в двоичную символьную строку. Её общий вид:

```
BIN$(α)
```
,

где:

- BIN («BINary» — «двоичный») — служебное слово;
- α — арифметическое выражение.

Вначале вычисляется значение арифметического выражения α; результат преобразуется в целое число (возможно со знаком); если это число не попадает на отрезок [-32768,65535], то компьютер фиксирует ошибку переполнения:

«Overflow».

Полученное число записывается в двоичной системе счисления, а затем преобразуется в символьную строку, соответствующую его двоичному коду.

Таким образом,

```
Ok
```

```
? BIN$(3),BIN$(0)
11···········0
Ok

Ok
? BIN$(32767)
111111111111111
Ok

Ok
? BIN$(65536)
Overflow
Ok
```

Максимальная длина строки результата функции `BIN$()` — 16.

Обозначим целую часть значения выражения α буквой N.

Заметим, что для отрицательных N компьютер вычисляет значение функции `BIN$(N)` по рекуррентной формуле:

```
BIN$(N)=BIN$(65536+N)
```
.

Примеры:

- 1)

```
BIN$(-1)     возвращает "1111111111111111",
BIN$(-32768) возвращает "1000000000000000",
BIN$(32768)  возвращает "1000000000000000",
BIN$(65535)  возвращает "1111111111111111".
```

- 2)

```
print VAL("&B"+BIN$(15))
15
Ok
```

Наконец, при помощи функции `BIN$` можно «научить» компьютер двоичной арифметике. Посмотрите, например, как работает следующая программа:

- 3)
0417-03.bas
0417-03.bas

```
Ok
10 PRINT"Могу найти сумму двух двоичных чисел!"
20 INPUT "Первое число, второе число";N1$,N2$
40 ANS=VAL("&B"+N1$)+VAL("&B"+N2$)
50 BN$=RIGHT$(STRING$(16,"0")+BIN$(ANS),16)'См.аналогию в п. 1.9!
60 PRINT"Ответом является:";BN$;" или";ANS;"десятичное"
run
Могу найти сумму двух двоичных чисел!
Первое число, второе число? 1111111111111111,1
Ответом является:0000000000000000 или 0 десятичное
Ok
```

Неизвестные Вам пока строковые функции `RIGHT$` и `STRING$` рассмотрены ниже в разделах IV.2.3. и IV.2.4.

- 4) среди простых чисел, не превосходящих N, найти такое, в двоичной записи которого максимальное число единиц.
0417-04.bas
0417-04.bas

```
NEW
```

```
Ok
10 INPUT N
20 FOR I=1 TO N:FOR J=2 TO INT(SQR(I))
40 IF IMODJ=0 GOTO 100 ELSE NEXT J
50 M$=BIN$(I):PRINT I;M$:L=0
60 FOR K=1 TO LEN(M$):IF MID$(M$,K,1)="1" THEN L=L+1
80 NEXTK
90 IF L>R THEN R=L:R1=I
100 NEXT I
110 PRINT R;R1
run
? 18
·1·1
·3·11
·5·101
·7·111
·11·1011
·13·1101
·17·10001
·3··7
Ok
```

- 5) выделить старший и младший байты двоичного числа `A%`
  0417-05.bas
  0417-05.bas

```
NEW
Ok
10 INPUT"Введите число  A% (не более 16 двоичных цифр)";A%
20 IF A%<0 THEN H%=A%\256-1 ELSE H%=A%\256
25 L%=A%MOD256
30 PRINT RIGHT$("00000000"+BIN$(H%),8);" ";RIGHT$("00000000"+BIN$(L%),8)
run
Введите число A% (не более 16 двоичных цифр)? &B1011110001010011
10111100 01010011                                  ▲
Ok                                                 │ число отрицательное!

run
Введите число A% (не более 16 двоичных цифр)? &B0011110001010000
00111100 01010000                                  ▲
Ok                                                 │ число положительное!
```

IV.1.8. OCT$-функция

`OCT$`-функция служит для преобразования числа в восьмеричную символьную строку. Её общий вид:

```
OCT$(α)
```
,

где:

- `OCT` («OCTal» — «восьмеричный») — служебное слово;
- α — арифметическое выражение.

Эта функция вычисляет значение арифметического выражения α, преобразует результат в целое (возможно со знаком); если результат не принадлежит отрезку [-32768,65535], то фиксируется ошибка:

«Overflow»
(*«Переполнение»*).

В противном случае результат преобразуется в символьную строку, представляющую его значение в восьмеричной системе счисления.

Максимальная длина строки–результата OCT$() — 6 символов (6 байт).

Пример.

```
Ok
print OCT$(&HFFFF)
177777
Ok
```

Обозначим буквой N целую часть значения выражения α.

Если N принадлежит диапазону [0,65535], то значение функции OCT$() находится компьютером без обращения к дополнительному коду, например:

```
print OCT$(0)
0
Ok

print OCT$(65535)
177777
Ok
```

Если же N принадлежит диапазону [-32768,-1], то значение функции OCT$() находится ЭВМ с использованием дополнительного кода, т.е. по рекуррентной формуле:

```
OCT$(N)=OCT$(65536+N)
```
.

Например:

- 1)

```
Ok                              0k
print OCT$(-1)   сравните с   print OCT$(65535)
177777                          177777
Ok                              Ok
```

- 2)

```
Ok                              Ok
? OCT$(-32768)   сравните с   print OCT$(32768)
100000                          100000
Ok                              Ok
```

Задача. Попытайтесь научить компьютер «восьмеричной» арифметике. Используйте для этой цели идею следующего фрагмента:

```
print VAL("&O"+OCT$(100))
 100
Ok
```
И, наконец...

IV.1.9. HEX$–функция

HEX$–функция служит для преобразования целого числа в шестнадцатеричную символьную строку. Её общий вид:

```
HEX$(α)
```
,

где:

- `HEX` («HEXadecimal» — «шестнадцатеричный») — служебное слово;
- α — арифметическое выражение.

Вначале вычисляется N — целая часть значения арифметического выражения α. Если N не попадает на отрезок [-32768,65535], то компьютер сообщает об ошибке:

«Overflow»
(«*Переполнение*»).

«Правильное» N преобразуется в символьную строку, соответствующую его шестнадцатеричному коду. Максимальная длина результата вычисления значения функции равна *четырём* байтам (четырём символам).

Если целое число N находится на отрезке [0,65535], то преобразование элементарно:

- `HEX$(0)` возвращает «0»,
- `HEX$(65535)` возвращает «FFFF».

Однако, значение функции при отрицательных N, принадлежащих [-32768,-1], вычисляется компьютером с использованием дополнительного кода, т.е. с применением рекуррентной формулы:

```
     HEX$(N)=HEX$(65536+N);
    HEX$(-1)=HEX$(65535) возвращает "FFFF";
HEX$(-32768)=HEX$(32768) возвращает "8000".
```

Функция `VAL()` в комбинации с функцией `HEX$()` позволяют получить обратное преобразование:

```
N=VAL("&h"+HEX$(N))
```
.

Примеры:

```
Ok
? VAL("&h"+HEX$(3))
 3
Ok

Ok
? VAL("&H"+HEX$(-10))
-10
Ok
```

Существование функции `HEX$()` делает возможным «обучение» компьютера шестнадцатеричной арифметике. Единственная трудность состоит в добавлении к числу нулей, стоящих в старших разрядах (если, конечно, это необходимо!).

Подумайте!... Придумали? Если нет, то возьмите на вооружение следующий красивый приём:

```
X$=RIGHT$("0000"+HEX$(N),4)
```

или

```
X$=RIGHT$(STRING$(4,"0")+HEX$(N),4)
```
.

Пример.

- 1)
  0419-01.bas
  0419-01.bas

```
NEW
Ok
10 PRINT"Введите два шестнадцатеричных числа"
20 INPUT A$,B$
30 SM=VAL("&H"+A$)+VAL("&H"+B$)
40 DF=VAL("&H"+A$)-VAL("&H"+B$)
50 PRINT "Сумма, разность:";RIGHT$("0000"+HEX$(SM),4);SPC(1);RIGHT$("0000"+HEX$(DF),4)
run
Введите два числа
? 1,1
Сумма, разность:0002 0000
Ok

run
Введите два числа
? F000,FFF1
Сумма, разность:EFF1 F00F
Ok

run
Введите два числа
? FFFF,FFFF
Сумма, разность:FFFE 0000
Ok

run
Введите два числа
? FFFFF,E
Overflow in 30
Ok
```

- 2) выделить старший и младший байты шестнадцатеричного числа A%.
  0419-02.bas
  0419-02.bas

```
NEW
Ok
10 INPUT "Введите число A% (не более 4 шестнадцатеричных цифр)";A%
20 IF A%<0 THEN H%=A%\256-1 ELSE H%=A%\256
25 L%=A%MOD256
30 PRINT RIGHT$("00"+HEX$(H%),2);" ";RIGHT$("00"+HEX$(L%),2)
run
Введите число A% (не более 4 шестнадцатеричных цифр)? &H67EA
67 EA                                                  ──▲──
Ok                                        положительное число┘
run
Введите число A% (не более 4 шестнадцатеричных цифр)? &HE2A9
E2 A9                                                  ──▲──
Ok                                        отрицательное число┘
```

Теперь настало время подумать, почему встроенные функции,изученные нами в разделах IV.1.1. — IV.1.9. IV.1.9, называются «функциями преобразования».

IV.2. Встроенные строковые функции

Пока слепо плыл сон по разбитым надеждам,
Космос с болью сочился над разбитой любовью,
Был из скрытных людей свет твой медленно изгнан,
И небо не спало.

—Избранные стихотворения компьютера RCA-301, поэма 929

В этом разделе мы рассмотрим следующие встроенные функции: MID$, LEFT$, RIGHT$, STRING$, SPACE$.

IV.2.1. MID$-функция

Общий вид `MID$` — функции следующий:

```
MID$(α,m[,n])
```
,

где:

- `MID` («MIDdle» — «середина») — служебное слово;
- α — строковое выражение;
- m, n — арифметические выражения, целые части значений которых должны принадлежать отрезку [1,255].

Напомним, что квадратные скобки, встречающиеся при описании синтаксиса, — обозначение, указывающее, что элемент синтаксической конструкции внутри них не является обязательным.

Условимся целые части значений арифметических выражений m и n обозначать соответственно M и N.

По умолчанию N равно количеству символов значения строкового выражения α от позиции M и до конца строки включительно.

Рассмотрим два возможных случая.

1. Функция `MID$` *слева* от знака равенства в операторе присваивания `LET`.

   ```
   [LET] MID$(α,m[,n]) = β
   ```

   Здесь β — строковое выражение.

   Пусть L — длина значения строкового выражения β. При выполнении оператора `LET` символы значения строкового выражения α, начиная с позиции M, последовательно замещаются на первые `min(L,N)` символов строкового выражения β. Остальные символы значения строкового выражения α остаются без изменений (см. рис.1).

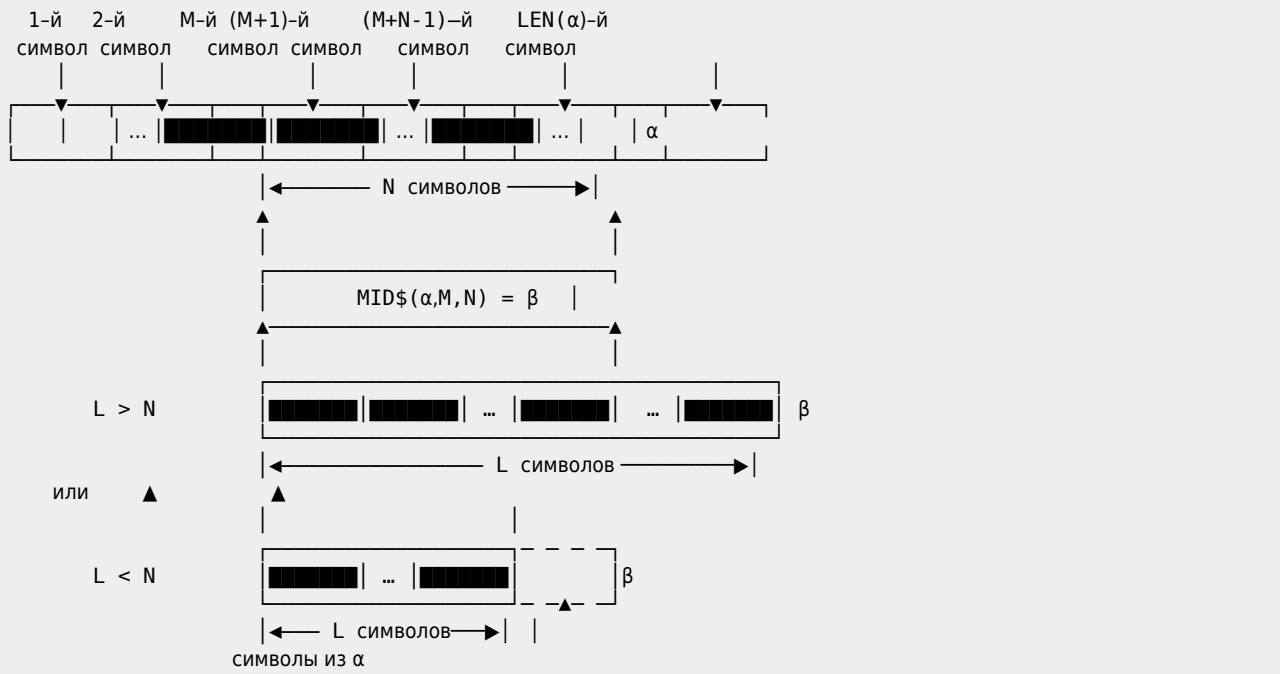


Рис. 1

Пример.
0421-01.bas
0421-01.bas

```
10 X$="12345678910":INPUT Y$,M,N:MID$(X$,M,N)=Y$:PRINT X$
```

```
run                 ┌──────────────────┐
? стол,8,4          │X$="12345678910"  │
1234567стол         │      └──┘        │
Ok                  │        Y$="стол" │
                    └──────────────────┘

run                 ┌──────────────────┐
? сто,8,3           │X$="12345678910"  │
1234567сто0         │      └─┘         │
Ok                  │        Y$="сто"  │
                    └──────────────────┘


run                 ┌──────────────────┐
? стул,1,4          │X$="12345678910"  │
стул5678910         │  └──┘            │
Ok                  │Y$="стул"         │
                    └──────────────────┘

run                 ┌──────────────────┐
? казак,8,5         │X$="12345678910"  │
1234567каза         │      └──┘        │
Ok                  │        Y$="каза" │
                    └──────────────────┘
```

И наконец, вместо оператора присваивания вида:

```
A$ = строковое выражение
```
,

используйте оператор: `MID$(A$,1) = строковое выражение`, чтобы избежать использования дополнительной памяти. Но учтите, что строковая переменная A$ уже должна существовать и `LEN(A$)≥LEN (строковое выражение)`!

2. Функция `MID$` *справа* от знака равенства в операторе присваивания `LET`.

```
[LET] γ = MID$(α,m[,n])
```

Здесь γ — строковая переменная.

При выполнении указанной конструкции N последовательных символов значения α, начиная с позиции M, становятся значением строковой переменной γ. Предварительное определение γ не обязательно (см. рис.2).

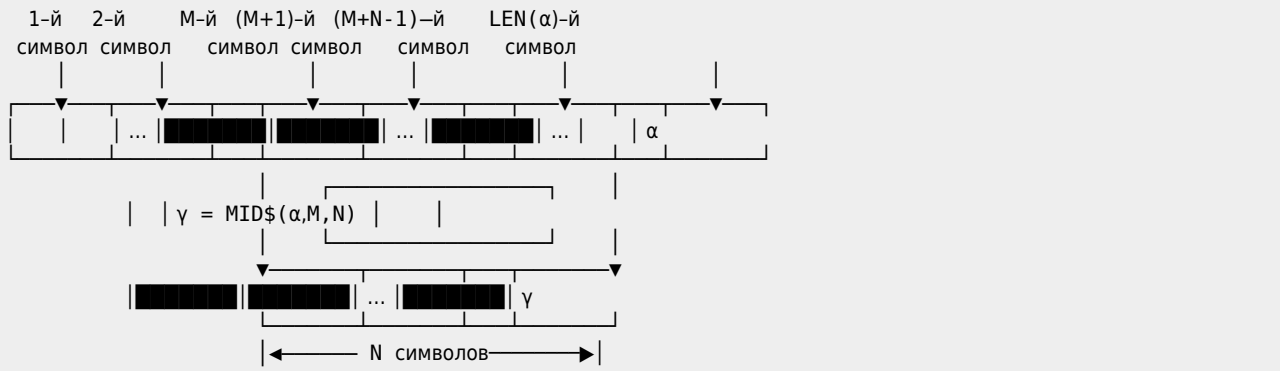


Рис. 2

Примеры:

- 1)

```
Ok
? MID$("КЛАВИАТУРА",3,4)
АВИА   └──┘
Ok
```

- 2)

```
Ok
? MID$("капрал"+"литр",4,5)
ралли  └────┘
```

```
Ok
```

- 3)

```
Ok
? MID$("молоко"+"брат",5,5)
кобра    └──────┘
Ok
```

- 4)

```
Ok
? MID$("СОР У НОР",3,5)
Р У Н   └────┘
Ok
```

- 5)

```
Ok
10 input I,J:? MID$("234",I,J)
run    run     run     run
? 1,1  ? 2,90  ? 10,1  ? 1,0
2      34
Ok     Ok      Ok      Ok

run       или     run
? -1,1             ? 0,0
Illegal function call in 10
Ok

run       или     run
? 2,-1             ? 0,1
Illegal function call in 10
Ok
```

- 6)

0421-06.bas
 0421-06.bas

```
NEW
Ok
1 Z$="123"
2 INPUT X$,M
3 Z$=MID$(X$,M)
4 Y$=MID$(X$,M+4)
5 PRINT Z$
6 PRINT Y$
run
? Баба-Яга-костяная нога,6
Яга-костяная нога
костяная нога
Ok
```

- 7) определить третий символ в слове X$.

```
Ok
input x$:z$=mid$(x$,3,1):print z$
? интеграл
т
Ok
```

Если требуется определить K-й символ в слове X$, то программа будет иметь следующий вид (разумеется, 1≤K≤LEN(X$)):
0421-07.bas
0421-07.bas

```
NEW
Ok
10 INPUT X$,K
20 IF K<1 OR K>LEN(X$) THEN PRINT"Буквы с указанным номером в слове нет":END ELSE Z$=MID$(X$,K,1):PRINT
Z$:END
run
? информатика,3
```

```
ф
Ok

run
? информатика,12
Буквы с указанным номером в слове нет
Ok
```

Пример 8. Для получения числа, состоящего из двух младших цифр целого числа N, воспользуйтесь строкой:

```
10 X%=VAL(MID$(STR$(N),LEN(STR$(N))-1,2))
```

Заметим, что Вы можете выполнить то же самое, используя функцию `INT()`:

```
10 INPUT N:PRINT N-INT(N/100)*100
```

Пример 9. Написать программу, определяющую, сколько раз слово Z$ встречается в слове X$.
0421-09.bas
0421-09.bas

```
NEW
Ok
10 INPUT X$,Z$:J=0
20 FOR I=1 TO LEN(X$)-LEN(Z$)+1
30 IF Z$<>MID$(X$,I,LEN(Z$)) THEN NEXTI ELSE J=J+1:NEXTI
40 PRINT J:END
run
? БАОБАБ,БА
 2
Ok
```

Пример 10. Написать программу, заменяющую в слове X$ слово Z$ на слово Y$ той же длины.
0421-10.bas
0421-10.bas

```
NEW
Ok
10 INPUT X$,Z$,Y$:IF LEN(Z$)><LEN(Y$) THEN 10
20 FOR J=1 TO LEN(X$)-LEN(Z$)+1
30 IF Z$<>MID$(X$,J,LEN(Z$)) THEN 50
40 MID$(X$,J,LEN(Y$))=Y$
50 NEXT J:PRINT X$:END
run
? ДОМ КИРПИЧНЫЙ
?? КИРПИЧНЫЙ
?? БАНАНОВЫЙ
ДОМ БАНАНОВЫЙ
Ok
```

Косо лети же, житель осок.

—В.Хлебников

Я разуму уму заря,
Я иду с мечом судия.

—Г.Р.Державин

Мечтатель! Летать чем?

—С.Кирсанов

Пример 11. Слова и фразы, переходящие в себя при «акустическом отображении», получили название *палиндромов*. Это слово греческого происхождения и означает «движущийся обратно», «обратимый». Найти палиндром практически невозможно, но его нетрудно построить. В качестве «строительных блоков» простейших палиндромов следует выбирать слова, не изменяющиеся при чтении от конца к началу. Выдерживают отражение в «акустическом зеркале» такие слова, как боб, дед, кок, мим, иди, топот, потоп, колок, ротор, кабак, моном и выражения, такие, например,как «Лёша на полке клопа нашёл».
0421-11.bas
 0421-11.bas

```
Ok
10 INPUT X$:Y$=""
20 FOR I=1 TO LEN(X$):Y$=MID$(X$,I,1)+Y$:NEXT
30 IF X$=Y$ THEN PRINT"Палиндром!" ELSE PRINT"Не палиндром!"
run
? косолетижежителосок
Палиндром!
Ok

run
? мечтателлетатчем
Палиндром!
Ok

run
? я разуму уму заря
Не палиндром!
Ok
```

Сравните результат действия функции `MID$(A$,I,J)`, стоящей справа от знака равенства в операторе присваивания, с результатом операции *вырезки*:

```
A[i:i+j-1]
```

,

где:

- A — текст;
- i — номер первого символа фрагмента, вырезаемого из текста A;
- j — количество символов в фрагменте, вырезаемом из текста A,

в школьном алгоритмическом языке.

Рассмотрите интересный часто встречающийся частный случай `j=1`, т.е. проведите аналогию между `MID$(A$,I,1)` и `A[i:i]`.

И, наконец, сопоставьте результат действия функции `MID$`, стоящей слева от знака равенства в операторе присваивания, с результатом работы команды частичного изменения значения литерной величины (команды присваивания вырезке)

```
A[i:j]=B
```

,

позволяющей заменить часть слова A, начинающуюся с i–той буквы и оканчивающуюся j–той буквой, на слово B, в школьном алгоритмическом языке.

IV.2.2. LEFT$–функция

Общий вид функции:

`LEFT$(α,n)`,

где:

- LEFT («left» — «левый») — служебное слово;
- α — строковое выражение;
- n — арифметическое выражение, целая часть значения которого должна принадлежать отрезку [0,255].

Функция LEFT$ позволяет выделить самые *левые* n символов значения строкового выражения α.

LEFT$ — функция является «частным случаем» MID$-функции и через неё может быть определена так:

`LEFT$(α,n) = MID$(α,1,n)`.

Примеры:

- 1)
  0422-01.bas
  0422-01.bas

```
Ok
10 INPUT X$,N
20 U$=MID$(X$,1,N)
30 V$=LEFT$(X$,N)
40 PRINT U$;V$
run
? 1234567890,6
123456123456
Ok
```

- 2)

```
Ok
? LEFT$("деньги",0)
Illegal function call
Ok
```

- 3)

```
Ok
? LEFT$("деньги",4)
день   ┗━┛
Ok
```

- 4)

```
Ok
? LEFT$("abc",2);LEFT$(LEFT$("def",2),1)
abd
Ok
```

- 5)

```
Ok
? LEFT$("wap",-1)
Illegal function call
Ok
```

IV.2.3. RIGHT$-функция

Общий вид функции:

```
RIGHT$(α,n)
```
,

где:

- RIGHT («right» — «правый») — служебное слово;
- α — строковое выражение;
- n — арифметическое выражение, целая часть значения которого должна принадлежать отрезку [0,255].

Функция RIGHT$ позволяет выделить самые *правые* n символов значения строкового выражения α (в этом смысле функция RIGHT$() симметрична функции LEFT$()!).

RIGHT$ — функция является частным случаем MID$-функции и через неё может быть определена так:

```
RIGHT$(α,n) = MID$(α,L-n+1,n)
```
,

где L — длина значения строкового выражения α.

Примеры:

- 1)

```
Ok
? RIGHT$("abcdgoldfish",8)
goldfish      └──────┘
Ok
```

- 2)

```
Ok
? RIGHT$("гамбит",3)
бит     └─┘
Ok
```

- 3)
0423-03.bas
 0423-03.bas

```
Ok
10 INPUT X$,N
20 U$=MID$(X$,LEN(X$)-N+1)
30 V$=RIGHT$(X$,N):? U$;V$
run
? 1234567890,4
78907890
Ok
```

- 4)

```
Ok
? RIGHT$(LEFT$("рубайте!",6),4)
байт
Ok
```

- 5)

```
Ok
? MID$("рубайте!",3,4)
байт   └──┘
Ok
```

Заметим, что хотя функции LEFT$ и RIGHT$ являются «частными случаями» функции MID$, их нельзя использовать *слева* от символа присваивания в операторе LET!

Посмотрите...

```
NEW
Ok
1 A$="Барсик":MID$(A$,1,4)="Корт":PRINT A$
2 B$="Комик":LEFT$(B$,3)="Том":PRINT B$
run
Кортик
Syntax error in 2
Ok
```

Пример 6. Образовать слово Y$, состоящее из первых N и последних K символов данного слова X$.
0423-061.bas
0423-061.bas Первоначальный вариант программы выглядит так:

```
NEW
Ok
10 INPUT X$,N,K
20 Y$=LEFT$(X$,N)+RIGHT$(X$,K)
30 PRINT "Результат:";Y$
40 END
run
? аллигатор,3,6
Результат:аллигатор
Ok

run
? аллигатор,0,6
Результат:игатор
Ok

run
? аллигатор,0,0
Результат:
Ok

однако...

run
? аллигатор,-4,3
Illegal function call in 20
Ok
```

Поэтому дополним программу строкой:
0423-062.bas
0423-062.bas

```
15 IF N<0 OR N>LEN(X$) OR K<0 OR K>LEN(X$) THEN PRINT"Не балуйтесь! Повторите!":GOTO 10
```

Выполним полученную программу:

```
run
? аллигатор,9,10
Не балуйтесь! Повторите!
? аллигатор,6,0
Результат:аллига
Ok

run
? аллигатор,-1,5
Не балуйтесь! Повторите!
? аллигатор,9,10
Не балуйтесь! Повторите!
?
```

Строка 15 обеспечивает повторный запрос на ввод исходных данных в том случае, когда K<0 или K> длины слова X$, или N<0, или N>длины слова X$ (своеобразная «защита» программы от непредусмотренной исходной информации).

Пример 7. Написать программу перевода чисел, записанных римскими цифрами, в числа, записанные арабскими цифрами.
0423-07.bas
0423-07.bas

```
NEW
Ok
1 DATA 1000,M,900,CM,500,D,400,CD,100,C,90,XC,50,L,40,XL,10,X,9,IX,5,V,4,IV,1,I:INPUT"N$";N$
2 READ A,A$
3 IF A$=LEFT$(N$,LEN(A$)) THEN N$=RIGHT$(N$,LEN(N$)-LEN(A$)):N=N+A:GOTO 3
4 IF N$>" "GOTO 2
5 PRINT N
run
N$? IV
 4
Ok

run
N$? CMIV
 904
Ok

run
N$? MMMDXCLLI
 3691
Ok

run
N$? MCMXVII
 1917
Ok

run
N$? MDCCCXII
 1812
Ok
```

А как насчёт программы обратного перевода? Пожалуйста!

Пример 8. Написать программу перевода чисел, записанных арабскими цифрами, в числа, записанные римскими цифрами.
0423-08.bas
0423-08.bas

```
NEW
Ok
1 DATA 1000,M,900,CM,500,D,400,CD,100,C,90,XC,50,L,40,XL,10,X,9,IX,5,V,4,IV,1,I:INPUT"N";N
2 READ A,A$
3 IF A<=N THEN PRINT A$;:N=N-A:GOTO 3
4 IF N>0 GOTO 2
run
N? 654
DCLIV
Ok

run
N? 25
XXV
Ok

run
```

```
N? 11111
MMMMMMMMMMCXI
Ok

run
N?3691
MMMDCXCI
Ok
```

IV.2.4. STRING$–функция

Общий вид STRING$ — функции следующий:

`STRING$(n,α)`,

или

`STRING$(n,m)`,

где:

- `STRING` («string» — «строка») — служебное слово;
- α — строковое выражение;
- n,m — арифметические выражения, целые части значений которых должны принадлежать отрезку [0,255].

Обозначим N—значение арифметического выражения n. Тогда, в случае формата `STRING$(n,α)` функция возвращает строку, состоящую из N одинаковых символов, равных первому символу значения строкового выражения α. А в случае формата `STRING$(n,m)` значением функции является последовательность N одинаковых символов, код ASCII которых одинаков (ведь символы одинаковы!) и равен целой части значения арифметического выражения m.

Поэтому, если Вы хотите инициализировать строковую переменную A$, N *пробелами* (наиболее часто встречающаяся операция!), то два приведённых ниже оператора выполняют идентичное действие:

`A$=STRING$(N,32)`,

`A$=STRING$(N," ")` (ведь `CHR$(32)=" "` !).

Таким образом, с помощью `STRING$`–функции можно назначать длину и проводить инициализацию строковых переменных пробелами. Приведём несколько тривиальных примеров:

- 1)

```
Ok
Y$=STRING$(10,"*"):? LEN(Y$):? "#"+Y$+"FILE"
 10
#**********FILE
Ok
```

- 2)

```
Ok
print STRING$(10,52);STRING$(10,CHR$(52))
4444444444444444444
Ok
```

- 3)

```
Ok
A$=STRING$(5," "):MID$(A$,3,3)="Все!":? A$
```

```
··Все
Ok
```

Если n=0, то функция STRING$ возвращает пустую строку (""), независимо от того, какие α или m Вы задали.

В тех случаях, когда целые части значений арифметических выражений n и m находятся вне отрезка [0,255], на экране дисплея появляется сообщение об ошибке:

«Illegal function call»
(*«Недопустимый вызов функции»*).

IV.2.5. SPACE$-функция

Общий вид функции:

```
SPACE$(n)
```
,

где:

- SPACE («space» — «пространство») — служебное слово;
- n — арифметическое выражение, целая часть значения которого должна принадлежать отрезку [0,255].

С помощью SPACE$-функции назначается длина и проводится начальная инициализация пробелами строковых переменных. Например, если Вы хотите инициализировать новую строковую переменную X$ N пробелами, то используйте оператор присваивания X$=SPACE$(N)

SPACE$-функция является частным случаем STRING$-функции и через неё может быть определена так:

```
SPACE$(n)=STRING$(n," ")
```

IV.2.6. Примеры

- 1) выделить из текста, являющегося значением строковой переменной T$, отдельные слова (они отделены друг от друга одним пробелом) и записать их в строковый массив с именем W$. (Психологи утверждают, что из минуты, затраченной на чтение, мы 58 секунд считываем промежутки между символами!)
  0426-01.bas
  0426-01.bas

```
NEW
Ok
10 LINEINPUT T$:T$=T$+" ":K=0'K-количество слов в  T$
20 FOR I=1 TO LEN(T$):IF MID$(T$,I,1)=" " THEN K=K+1
35 NEXT I 'Количество слов в T$ найдено!
40 DIM W$(K):DIM P(K)'Вот теперь можно описать массивы!
45 FOR J=1 TO K:W$(J)=SPACE$(5):NEXTJ 'Начальная инициализация пробела ми элементов массива  W$(K)
47 N=1:FOR J=1 TO K:FOR I=N TO LEN(T$)
70 IF MID$(T$,I,1)=" " GOTO 85
80 NEXT I
85 P(J)=I:W$(J-1)=MID$(T$,N,P(J)-P(J-1)-1):N=P(J)+1:PRINT W$(J-1);SPC(5)
90 NEXT J
run
123 4 567
123·····4·····567
Ok
```

Елечвок енмяет ослог.

—А.Вознесенский. Человек

- 2) даны два слова X$ и Y$. Проверить, можно ли из символов, входящих в слово X$, составить слово Y$. Символы можно переставлять, и каждый символ можно использовать несколько раз!
  0426-02.bas
  0426-02.bas

```
NEW
Ok
10 INPUT X$,Y$
30 FOR I=1 TO LEN(Y$):FOR J=1 TO LEN(X$)
50 IF MID$(Y$,I,1)=MID$(X$,J,1) THEN A$="да":NEXT I ELSE A$="нет":NEXT J
80 PRINT A$:END
run                    run
? школаа,алокша       ? AB,BB
да          да
Ok                     Ok
```

- 3) даны два натуральных числа M и N. Проверьте,можно ли из цифр числа M составить число N. Цифры можно переставлять и использовать:
  - a) более одного раза
    0426-031.bas
    0426-031.bas

```
NEW
Ok
10 INPUT M,N:M$=STR$(M):N$=STR$(N)
30 FOR I=2 TO LEN(N$):FOR J=2 TO LEN(M$)
40 IF MID$(N$,I,1)<>MID$(M$,J,1) THEN NEXT J:?"Нельзя!"ELSE NEXTI:?"Можно!"
run              run              run
? 6785,58758     ? 6785,58358     ? 6785,87
Можно!      Нельзя!     Можно!
Ok               Ok               Ok
```

  - b) не более одного раза
    0426-032.bas
    0426-032.bas

```
b)
NEW
Ok
10 INPUT M,N:X$=" ":M$=STR$(M):N$=STR$(N)
40 FOR I=2 TO LEN(N$):FOR J=2 TO LEN(M$)
50  IF  MID$(N$,I,1)=MID$(M$,J,1)  THEN  MID$(N$,I,1)=X$:MID$(M$,J,1)=X$:NEXTI:?"Можно!"  ELSE
NEXTJ:?"Нельзя!"
run                    run
? 12345432,1236        ? 1234554320,1022
Нельзя!        Можно!
Ok                     Ok
```

- 4) число 41 обладает следующим свойством: $41^2=1681$,$\sqrt{16}=4$ и $\sqrt{81}=9$, т.е. числа 16 и 81 — точные квадраты; найти все натуральные числа, не превосходящие N, и такие, что первые две и последние две цифры квадрата числа являются точными квадратами.
  0426-04.bas
  0426-04.bas

```
NEW
Ok
10 INPUT"Сколько чисел необходимо проверить";N:PRINT"Интересующие Вас числа:";
20 FOR I=10 TO N:B=I^2
40 B$=MID$(STR$(B),2):IF VAL(RIGHT$(B$,2))<>0 THEN 50 ELSE NEXT I: END
50       IF       SQR(VAL(LEFT$(B$,2)))=FIX(SQR(VAL(LEFT$(B$,2))))       AND
SQR(VAL(RIGHT$(B$,2)))=FIX(SQR(VAL(RIGHT$(B$,2))))THENPRINT  I;:NEXTI  ELSE  NEXTI
run
Сколько чисел необходимо проверить? 100
Интересующие Вас числа: 41
Ok
```

- 5) написать программу, которая заменяет часть слова Z$ текста T$ на текст Y$ с сохранением одного пробела между словами.
  0426-05.bas
  0426-05.bas

```
NEW
Ok
5 PRINTSPC(8);"Вводите текст !":LINEINPUT T$:T$=T$+" ":K=0
15 INPUT "Какое слово менять";Z$
16 INPUT "На какое слово менять";Y$
20 FOR I=1 TO LEN(T$):IF MID$(T$,I,1)=" " THEN K=K+1:NEXTI ELSE NEXTI 'K-количество слов в тексте T$
40 DIM W$(K):DIM P(K)'Вот теперь можно описать массивы!
45 FOR J=1 TO K:W$(J)=SPACE$(5):NEXT J
47 N=1'N-номер первой буквы текущего слова в тексте
50 FOR J=1 TO K: FOR I=N TO LEN(T$)
70 IF MID$(T$,I,1)=" " GOTO 85
80 NEXT I
84 'P()-массив,содержащий номера позиций пробелов в тексте
85 P(J)=I:W$(J-1)=MID$(T$,N,P(J)-P(J-1)-1):N=P(J)+1
90 NEXT J
95 A$="Образец не найден! Ваш текст:"
100 FOR J=0 TO K-1
105 IF W$(J)=Z$ THEN W$(J)=Y$:A$="":GOTO 120
106 FOR I=1 TO LEN(W$(J))-LEN(Z$)+1
107 IF Z$<>MID$(W$(J),I,LEN(Z$))THEN115
108 W$(J)=LEFT$(W$(J),I-1)+Y$+RIGHT$(W$(J),LEN(W$(J))-I-LEN(Z$)+1):A$=""
115 NEXTI
120 NEXT J
130 FOR J=0 TO K-1:B$=B$+W$(J)+" ":NEXT
145 PRINT A$
150 PRINT B$:END
run
·····Вводите текст !
Мама мыла Эмму
Какое слово менять? мы
На какое слово менять? очень люби
.............................
Мама очень любила Эмму
Ok
```

- 6) составьте программу, в результате работы которой выяснялось бы, можно ли переставить цифры десятичной записи числа K так, чтобы они образовывали арифметическую или геометрическую прогрессию.
  0426-06.bas
  0426-06.bas

```
NEW
Ok
10 CLS:DIM K,M%,D%,Q,F$:INPUT"Введите нат. число";K
30 F$=STR$(K):M%=LEN(F$):DIM A(M%)
40 FOR I=1 TO M%:A(I)=VAL(MID$(F$,I,1)):NEXT
50 FOR I=2 TO M%-1:FOR J=I+1 TO M%
60 IF A(I)>A(J)THEN SWAP A(I),A(J)
70 NEXT J,I
80  D%=A(3)-A(2)
90 IF M%<3 THEN PRINT"Число должно содержать не менее 2 цифр":GOTO 20
100 IF M%=3 THEN PRINT"Арифм.прогрессия есть!":GOTO 150
110 FOR I=4TO M%
120 IF A(I)-A(I-1)<>D%THEN ?"Арифм.прогрессии нет":GOTO160
130 NEXT I
140 PRINT"Арифм.прогрессия есть!"
150 IF M%=3 AND A(3)=0 THEN PRINT"Геом.прогрессии нет":END
160 FOR I=2 TO M%:IF A(I)=0 THEN?"Геом.прогрессии нет":END
170 NEXT I
180 Q=A(3)/A(2)
190 IF M%=3 THEN ?"Геом.прогрессия есть!":END
200 FOR I=4 TO M%
210 IF A(I)/A(I-1)<>Q THEN PRINT"Геом.прогрессии нет":END
220 NEXT I
230 PRINT"Геом.прогрессия есть!":END
run                                run
```

```
Введите нат. число? 7132546    Введите нат. число? 42
Арифм.прогрессия есть!         Арифм.прогрессия есть!
Геом.прогрессии нет            Геом.прогрессия есть!
Ok                                        Ok
```

- 7) составить программу, которая m–кратно повторяет каждый символ в слове X$.
  0426-07.bas
  0426-07.bas

```
NEW
Ok
10 INPUT"Введите слово";X$
20 INPUT"Укажите число повторений символов";M%
30 IF M%=0 OR M%*LEN(X$)>255 THEN ?"Повторите ввод":GOTO10
40 N$="":FOR I=1 TO LEN(X$):FOR J=1 TO M%
70 N$=N$+MID$(X$,I,1):NEXTJ,I
90 PRINT "Вы этого хотели?  ";N$
run
Введите слово? керпкпр
Укажите число повторений символов? 2
Вы этого хотели?  ккееррппккппрр
Ok
```

- 8) составить программу вывода всех натуральных чисел, меньших N, сумма квадратов цифр которых равна M.
  0426-08.bas
  0426-08.bas

```
NEW
Ok
5 DEFINT I,J,N,M,Y:INPUT N,M
10 FOR I=1 TO N:Y=0:A$=STR$(I)
20 FOR J=2 TO LEN(A$):Y=Y+VAL(MID$(A$,J,1))^2:NEXTJ
30 IF Y=M THEN PRINTI;:NEXTI ELSE NEXTI
40 END
run
? 123,65
·18··47··74··81··108
Ok
```

- 9) по данному натуральному N найдите наименьшее из чисел, имеющих столько же и таких же цифр, что и N, если известно, что каждую цифру числа N можно использовать при записи один раз.
  0426-09.bas
  0426-09.bas

```
NEW
Ok
10 DEFINT I,J: DEFSTR A
20 INPUT "В каком числе хотите перетрясти цифры";N
25 IF LEN(STR$(N))>14 THEN PRINT "Не озоруй!":GOTO 20
30 FOR I=0 TO9:FOR J=2 TO LEN(STR$(N))
40 IF I=VAL(MID$(STR$(N),J,1))THEN A=A+STR$(I)
50 NEXT J,I
60 PRINT VAL(A):END
run
В каком числе хотите перетрясти цифры? 87655543
·34555678
Ok
run
В каком числе хотите перетрясти цифры? 112235799435213
Не озоруй!
В каком числе хотите перетрясти цифры? ...
```

- 10) написать программу, устанавливающую, какие из натуральных чисел, небольшие заданного натурального M, делятся на каждую свою цифру.
  0426-10.bas
  0426-10.bas

```
10 DEFINT I,J,M:DEFSTR A:INPUT M
```

```
30 FOR I=1 TO M:A=STR$(I):FOR J=2 TO LEN(A)
55  IF  VAL(MID$(A,J,1))=0  THEN  NEXT  J:NEXT  I:END  ELSE::::IF  I  MOD  VAL(MID$(A,J,1))=0  THEN
NEXTJ:PRINTI;:NEXTI ELSE NEXTI
run
? 40
·1··2··3··4··5··6··7··8··9··11··12··15··22··24··33··36
Ok
```

- 11) определите, сколько цифр используется в записи натурального числа X только по одному разу.
0426-111.bas
0426-111.bas
Первый способ.

```
10 INPUT X:X$=STR$(X)
20 K=0
30 FOR J=2 TO LEN(X$)
40 P=0
50 FOR I=2 TO LEN(X$)
60 IF MID$(X$,J,1)=MID$(X$,I,1) THEN P=P+1
70 NEXTI
80 IF P=1 THEN K=K+1
90 NEXTJ
100 PRINT "K=";K
run
? 45687345976
K=3
Ok
```

0426-112.bas
0426-112.bas
Второй способ

```
10 INPUT"Введите число";X$
20 L=LEN(X$):P=L
30 FOR I=1 TO L:FOR J=1TO L
40 IFMID$(X$,I,1)=MID$(X$,J,1)ANDI<>J THEN P=P-1:NEXTI ELSE NEXT J,I
50 PRINT"Цифр,используемых 1 раз:";P
run
Введите число? 12341524162
Цифр,используемых 1 раз: 3
Ok
```

- 12) из каждого слова массива C$(N) слов вычеркнуть те символы, которые употребляются при написании каждого из слов массива.
0426-12.bas
0426-12.bas

```
NEW
Ok
10 CLEAR 1500
20 INPUT"Введите количество слов";N:PRINT"Вводите слова!"
27 'В строках 30-60 идентификация исходного массива  C$(N).
30 DIM C$(N):FOR I=1 TO N:INPUTC$(I):NEXT:K$=""
75 'В строках 80-130 находятся "общие" символы. Значение переменной K$ состоит из "общих" символов.
80 FOR I=1 TO N:FOR J=1 TO LEN(C$(I)):FOR R=1 TO N
110 FOR K=1 TO LEN(C$(R))
120 IF MID$(C$(I),J,1)=MID$(C$(R),K,1) THEN NEXT R:K$=K$+MID$(C$(I),J,1):NEXT J ELSE NEXT K,J
130 NEXT I
135 'В строках 140-180 происходит"вычеркивание"из слов массива "общих" символов.
140 FOR I=1 TO N:FOR J=1 TO LEN(C$(I)):FOR K=1 TO LEN(K$)
170 IF MID$(C$(I),J,1)=MID$(K$,K,1)THENMID$(C$(I),J,1)=" "
180 NEXT K,J,I
185 'Строки 190-230 выполняют "сжатие" слов массива  C$(N),т.к.при "вычеркивании" "общих" символов образуются
пробелы.
190 DIM N$(N)
200 FOR I=1 TO N:FOR J=1 TO LEN(C$(I))
220 IF MID$(C$(I),J,1)<>" "THENN$(I)=N$(I)+MID$(C$(I),J,1)
230 NEXT J,I
```

```
240 FOR I=1TO N:PRINT"Вот";I;"слово:" ;N$(I);"    -  не правда ли?":NEXT
run
Введите количество слов? 4
Вводите слова!
? апро
? ажип
? сонп
? портфель
Вот 1 слово:апро   - не правда ли?
Вот 2 слово:ажи   - не правда ли?
Вот 3 слово:сон   - не правда ли?
Вот 4 слово:ортфель   - не правда ли?
Ok
```

- 13) определить количество *различных* букв, встречающихся в слове A$ более одного раза.
  0426-13.bas
  0426-13.bas

```
NEW
Ok
10 INPUT A$:S=0
20 FOR I=1 TO LEN(A$)
30    P=0
40    FOR J=1 TO LEN(A$)
50       IF MID$(A$,I,1)=MID$(A$,J,1) THEN P=P+1
60    NEXT J
70    IF P>1 THEN S=S+1/P
80 NEXT I
90 PRINT FIX(S+.5):END
run
 aaaaaaaa
 1
Ok

run
? бегемот
 1
Ok

run
? Кинг-Конг
 3
Ok

run
? мяучело
 0
Ok
```

IV.3. Функции пользователя. Оператор DEF FN

Язык MSX BASIC предоставляет возможность программисту (пользователю) *определять* в составляемой им программе одну или несколько «собственных» функций с помощью специального оператора `DEF FN`, имеющего следующий синтаксис:

```
DEF FNα[(аргумент [, аргумент] ...)] = β
```
,

где:

- `DEF FN` («FuNction DEFinition» — «определение функции») — служебные слова;
- α — имя переменной, для которой выделяется память;
- аргумент — имя переменной, для которой память не выделяется (функции пользователя могут иметь не более 9 аргументов);
- β — выражение, имеющее тот же тип, что и α.

Итак, оператор, определяющий функцию пользователя, начинается со служебного слова DEF. В идентификаторе определяемой функции два первых символа (буквы FN) являются обязательными, а остальные (α) выбираются автором программы по его желанию. *Аргументы* перечисляются через запятую, и вся совокупность *аргументов* заключается в круглые скобки. Отметим, что функция пользователя может вообще не иметь аргументов.

Каждый *аргумент* является *формальным* параметром: это означает, что он на самом деле «не существует» (память для него не выделяется!) и поэтому не совпадает с переменными, имеющими такое же имя в программе.

Далее, после символа присваивания «=» в записи оператора, определяющего функцию пользователя, следует выражение β, задающее алгоритм вычисления значения функции пользователя. Это есть «образец», по которому программа каждый раз вычисляет значение функции, если, конечно, это необходимо. В выражении допустим вызов любой встроенной функции и вызовы других функций пользователя.

Приведём примеры *описания* функций пользователя:

- 1)

```
    Формальные параметры
           | |      |     |
           ▼ ▼      ▼     ▼
10 DEF FNC(X,Y)=SQR(X+SIN(Y))
```

- 2)

```
 Параметры (аргументы), не являющиеся формальными
                 |   |
                 ▼   ▼
30 DEF FNBUS(Z)=A*Z+B*LOG(ABS(Z))+5467.453*FNC(4,0.7)
             ▲    ▲           ▲             ▲
             |    |           |             |
   Формальный параметр  Функция пользователя
```

- 3)

```
  Параметр (аргумент), не являющийся формальным
                                |
                                ▼
100 DEF FNQU$(QU,ST)=MID$(OL$(QU),ST)
              ▲   ▲            ▲   ▲
              |   |            |   |
       Формальные параметры
```

Отметим, что *рекурсия запрещена* (и прямая, и косвенная!). Кстати, о понятии *рекурсии* см. в разделе IV.4.

Посмотрите...

```
NEW
Ok
10 DEF FNY(X)=(X<1)*FNY(X)
20 PRINT FNY(3)
run
Out of memory in 20
Ok
```

Напомним, что *исполняемым* оператором мы называем оператор программы, определяющий конкретные действия, которые должны быть выполнены.

Оператор DEF FN может стоять в любом месте программной строки. Однако, он, в отличие от оператора DATA, — *исполняемый* оператор, т.е. функция пользователя должна быть определена до обращения к ней. Если же обращение к функции пользователя произведено раньше её определения оператором DEF FN, то на экране дисплея появляется сообщение об ошибке:

«Undefined user function in ...»
(«Функция пользователя не определена в строке ...»).

Поэтому, как правило, в начале программы для операторов DEFFN специально резервируется несколько программных строк.

Если в ходе выполнения программы, необходимо использовать значение функции пользователя, определённой в этой программе, при заданных значениях её аргументов, то необходимо *вызвать* её, т.е. записать идентификатор функции в соответствующее выражение, заменив формальные параметры нужными значениями аргументов — *фактическими* параметрами (константами или выражениями).

Приведём примеры *вызова* функции пользователя:

- a)

```
50 R=FNC(3.07,5.)
          ▲   ▲
          |   |
    фактические параметры 3.07 и 5.
```

- b)

```
90 A=10:B=2:D=3.3:C=FNBUS(5*D-7)
                                ▲
                                |
           фактический параметр 5*D-7
```

Если выражение β содержит программные объекты (имена переменных, элементов массивов, функции), описанные в качестве формальных параметров в операторе DEF FN, то их значения определяются из списка фактических параметров, указываемых при вызове функции пользователя.

Если выражение β содержит программные объекты, не входящие в список формальных параметров, то берутся их «текущие» значения из программы (на момент вызова функции пользователя).

В каждом из таких случаев интерпретатор обратится к описанию (определению) соответствующей функции пользователя, «подставит» вместо *формальных* параметров необходимые значения *фактических* параметров и произведёт требуемые вычисления, после чего результат вычисления значения функции возвращается в точку вызова функции.

Другими словами, компьютер выполнит над значениями фактических аргументов (параметров) все те действия, которые оператор DEF FNα выполняет над своими формальными аргументами.

Примеры:

- 1)
043-01.bas
 043-01.bas

```
Ok
10 DEF FNP(X)=X^2+X+1:S=0
           ▲  ▲   ▲
           |  |   |
   Формальный параметр X
20 FOR K=1 TO 7
30 S=S+FNP(K)
           ▲
           |
  Фактический параметр K
40 NEXT K
50 PRINT S:END
run
175
Ok
```

- 2)
043-02.bas
 043-02.bas

```
   Формальный параметр X
```

```
NEW          |  |
Ok           ▼  ▼
10 DEF FNA(X)=X*7
20 INPUT"Сколько лет Вашей собаке";D
30 PRINT"Если бы собака была человеком,ей было бы";FNA(D);"лет!"
                                                              ▲
                                                              |
                      Фактический параметр  D
run
Сколько лет Вашей собаке? 4
Если бы собака была человеком,ей было бы 28 лет!
Ok
```

- 3)
  043-03.bas
  043-03.bas

```
Аргумент, не являющийся формальным параметром
NEW                        |
Ok                         ▼
10 DEF FNZ(A,B)=A^2+B^2+C
           ▲ ▲  ▲   ▲
           | |  |   |
    Формальные параметры  A  и B
20 INPUT X,Y,C
30 PRINT FNZ(X,Y):END
             ▲ ▲
             | |
    Фактические параметры  X  и Y
run
? 2,-3,5
 18
Ok
```

- 4)
  043-04.bas
  043-04.bas

```
NEW
Ok
10 DEF FNS$(X$,K)=MID$(X$,K,1)
15 DEF FNQ$(X$)=FNS$(X$,K)+"$"
20 INPUT X$,K:PRINT FNQ$(X$):END
run
? APR,2
P$
Ok

run
? betta,4
t$
Ok
```

- 5)
  043-05.bas
  043-05.bas

```
NEW
Ok
10 INPUT X:IF X>1 THEN DEF FNY(X)=X^2 ELSE DEF FNY(X)=X^3
20 ? FNY(X):END
run
? 2
4
Ok

run
? -12
-1728
```

```
Ok
```

- 6)
  043-06.bas
  043-06.bas

```
10 'Функция FNODD$(N) помогает проверить,является ли целое число N нечетным:если значением функции является -1, то число N — нечетное, если же значением функции является 0, то число N — четное.
20 DEF FNODD(N)=RIGHT$(BIN$(N),1)="1"
```

Отметим, что можно определить 4 различные функции пользователя, имеющие одинаковое имя, но различающиеся по типам.

- 7)
  043-07.bas
  043-07.bas

```
NEW
Ok
10 DEF FNQ%=1:DEF FNQ!=2:DEF FNQ=3:DEF FNQ$="4"
20 DEFINTQ:PRINT FNQ;
30 DEFSNGQ:PRINT FNQ;
40 DEFDBLQ:PRINT FNQ:PRINT
50 PRINT FNQ%;FNQ!;FNQ;FNQ$
run
·1··2··3
·1··2··3·4
Ok
```

Этот факт приводит к удивительным последствиям: например, программа может модифицировать сама себя с помощью динамического изменения вызова функции:

несоответствие типов формальных и фактических параметров при обращении к функции пользователя
ошибка
обработка прерывания по ошибке ("ловушка" ошибки)
изменение типа функции пользователя

- 8)
  043-08.bas
  043-08.bas

```
NEW
Ok
5 ON ERROR GOTO 100
10 DEF FNY(X)=X^3
15 DEF FNY$(X$)=X$+"(Козьма Прутков)"
18 DEFSTR X
20 LINEINPUT X:PRINT FNY(X):END
100 IFERR=13 THEN DEFSTR Y:RESUME0'Код 13 имеет ошибка"Type mismatch"!
run
Смотри в корень!
Смотри в корень!(Козьма Прутков)
Ok
```

(см. раздел VIII.3.4.).

Поскольку выражение β, содержащее алгоритм вычисления значения функции пользователя, хранится в памяти компьютера как *часть программы*, то DEFFN — один из нескольких операторов, которые нельзя использовать в режиме прямого выполнения команд (это пример *оператора*, который не является *командой* !).

- 9)

```
Ok
DEF FNR(X)=X^5:PRINT FNR(2)
Illegal direct
Ok
```

Сообщение об ошибке «Illegal direct» означает, что в качестве команды непосредственного выполнения встречается оператор, недопустимый в этом режиме.

Учтите, что если Вы допустили ошибку во время определения функции, эта ошибка будет обнаружена только тогда, когда функция встретится в выражении. Компьютер будет указывать как на источник ошибки на ту строку текста, где использована неверно заданная функция, а не на ту строку, где эта функция была определена!

IV.4. Подпрограммы

Прежде всего отметим, что *подпрограммы* — это специальным образом оформленные группы программных строк. Подпрограммы бывают двух типов: *стандартные* и *нестандартные*.

Первые из них входят в состав математического обеспечения компьютера. Нам часто приходится неявным образом обращаться к ним, когда, например, вводим данные в память, печатаем результаты счета, вычисляем значения встроенных функций и т.п.

Нестандартные подпрограммы составляются самим пользователем и являются фактически фрагментами его программ.

В виде подпрограмм целесообразно оформлять логически завершённые части алгоритма, имеющие самостоятельную ценность; кроме того, использование подпрограмм позволяет экономить время и место в том случае, когда нужно в одной и той же программе несколько раз выполнить какую–либо последовательность операторов.Вместо того, чтобы многократно переписывать эту последовательность, напишите единственный программный *модуль*, к которому Вы сможете обращаться всякий раз при необходимости.

Подпрограмма, следовательно, играет в какой–то степени ту же роль, что в математике «теорема» или «лемма»: когда нужно доказать результат в некоторой специальной области, например, в дифференциальной геометрии, вовсе не обращаются каждый раз к базовым аксиомам теории множеств, а опираются на теоремы, которые косвенно через несколько уровней абстракции основываются на этих аксиомах.

Общая форма записи оператора обращения к подпрограмме (оператора вызова подпрограммы) следующая:

```
GOSUB n
```
,

где:

- `GOSUB` («GO to SUBroutine» — «идти к подпрограмме») — служебное слово;
- n — номер первой строки подпрограммы, 0≤n≤65529.

Разумеется, нежелательно, чтобы строка с номером n находилась внутри цикла!

Использование несуществующего номера программной строки n вызывает по явление на экране дисплея сообщения об ошибке вида:

«Undefined line number»
(*«Не определён номер строки»*).

Подпрограмма, как правило (!), завершается оператором

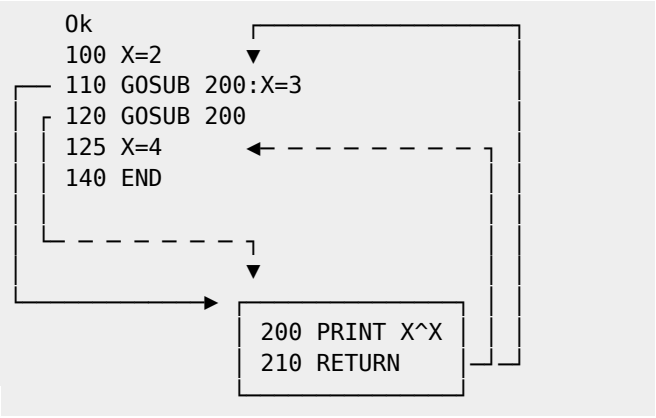
```
RETURN
```
,

где `RETURN` («return» — «возврат») — служебное слово.

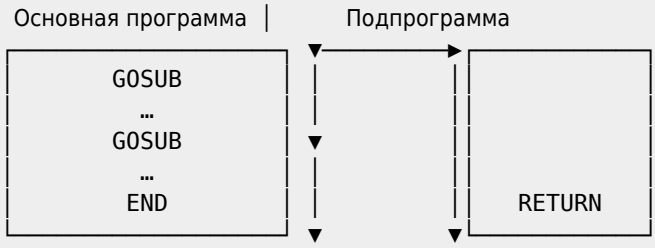
Оператор `GOSUB n` используется для вызова подпрограммы — группы операторов, начинающихся программной строкой с номером n и заканчивающихся оператором `RETURN`. Подпрограмма может начинаться с комментария и находиться в любом месте Вашей программы.

Главной особенностью оператора `GOSUB` является то, что оператор `RETURN` возвращает управление оператору, стоящему за последним выполненным оператором `GOSUB`. Заметим, что оператор, которому возвращается управление, может находиться не на следующей, а на той же программной строке! Приведём пример: 044-01.bas
044-01.bas

```
Ok
100 X=2
110 GOSUB 200:X=3
120 GOSUB 200
125 X=4
140 END
200 PRINT X^X
210 RETURN
run
 4
 27
Ok
```



Оператор `GOSUB n` может показаться очень похожим на оператор `GOTO n`. Существенная разница между ними в том, что оператор `GOSUB` как бы «посылает в командировку» — после выполнения операторов подпрограммы происходит возвращение назад.



Все переменные в подпрограмме являются *глобальными*. Это означает, что «доступ» к их значениям возможен из *любого* места Вашей основной программы.

Глобальная переменная — переменная, областью существования которой является вся программа. Локальная переменная — переменная с ограниченной областью существования в программе.

Образно говоря, переменные, имеющие одно и то же имя в основной (вызывающей) программе и подпрограмме не «однофамильцы», а одно и то же лицо!

При вызове подпрограммы значения переменных в основной программе не «замораживается», а могут быть изменены операторами подпрограммы! Без «замораживания» то и дело происходят пренеприятнейшие вещи.

Представьте себе, что Вы подарили своему товарищу очень ценную для него подпрограмму и объяснили, как ею пользоваться. Разумеется, Вашему товарищу нет никакого дела до того, какие имена переменных Вы использовали в Вашей подпрограмме. Поэтому может случиться, что и в его основной программе будут использоваться переменные с теми же именами, что и в подаренной Вами подпрограмме. Ясно, что эти переменные «неразличимы» для интерпретатора. Дальнейшее ужасно!

Как говорят программисты, в MSX BASIC постоянно присутствует «побочный эффект подпрограммы»!

Следовательно, важно выделить конкретные имена переменных, используемых *только* в данной подпрограмме и применить их для передачи значений аргументов из основной программы и возвращения результатов работы подпрограммы в основную программу.

Впрочем, оператор `RESTORE` (см. раздел II.4.5.) предоставляет возможность каждой подпрограмме иметь свои собственные (*локальные*) переменные. Необходимо просто при *каждом* обращении к ней выполнять оператор `RESTORE` для установки указателя на начало данных подпрограммы.

Пример.
044-02.bas
044-02.bas

```
NEW
Ok
10 DATA 5,2:'Данные основной программы
20 DATA 7,.5:'Данные подпрограммы
30 READ A,B:?A+B;:GOSUB 40:?A+B:END
40 RESTORE 20:READ A,B:?A+B;:RESTORE 10:READA,B:RETURN'A и B - локальные переменные подпрограммы!
run
·7··7.5··7
Ok
```

Наиболее распространённой ошибкой при использовании подпрограмм является их неполное отделение от основной программы. Основными способами защиты от указанной ошибки являются:

1. использование оператора `END`; например, если *пакет* подпрограмм (совокупность нескольких подпрограмм) начинается с программной строки 100, то строка основной программы

   ```
   99 END
   ```

   предотвратит вход в подпрограмму, начинающуюся со строки 100;

2. использование в основной программе перед первой строкой подпрограммы оператора `GOTO k`, где k — номер программной строки, расположенной за строкой, в которой расположен оператор `RETURN` данной подпрограммы (см. пример из раздела IV.4.1.).

Поэтому сформулируем важный практический вывод.

Помещайте подпрограммы или в *самом конце* или в *самом начале* основной программы !

При случайном(!) попадании в подпрограмму без использования оператора `GOSUB` она выполнится нормально, но компьютер выдаст сообщение об ошибке

«RETURN without GOSUB»
(«RETURN *без* GOSUB»).

Однако, если Вы случайно забыли поставить оператор `RETURN`, то интерпретатор может иногда не заметить ошибку! Он будет тщетно искать «забытый» оператор `RETURN` до конца программы, и, не найдя, прекратит все вычисления (т.к. программа уже будет выполнена полностью). Сообщения об ошибке не будет! Берегитесь!

Учтите, что операторы, стоящие за оператором `RETURN` в той же программной строке, *никогда* не выполняются, поэтому за оператором `RETURN` целесообразно помещать только комментарии.

Например,

```
50 RETURN '⟶
```

или

```
50 RETURN ⟶
```

.

Поговорим теперь о совместном использовании операторов `FOR…NEXT` и `GOSUB n`.

Внутри цикла использовать оператор `GOSUB`, *можно*, но оператор `NEXT` должен быть выполнен только *после* выполнения оператора `RETURN`.

В противном случае последует сообщение:

«NEXT without FOR»
(«NEXT *без* FOR»).

Пример.
044-03.bas
044-03.bas

```
NEW
Ok
5 INPUT K,N:FOR I=1 TO N:GOSUB 100:PRINT I;:NEXTI:END
100 IF K>I THEN NEXTI:RETURN ELSE RETURN
run
? 2,6
NEXT without FOR in 100
Ok

run
? 0,6
·1··2··3··4··5··6
Ok
```

С другой стороны, если цикл FOR…NEXT используется в подпрограмме, то оператор RETURN, находящийся в цикле FOR…NEXT, позволяет выйти из цикла и вернуться в основную программу. Это наводит на мысль, что наиболее эффективно размещать процедуру поиска элемента массива, обладающего требуемыми свойствами, *внутри* подпрограммы. Как только искомый элемент будет найден, выполнится выход из подпрограммы.

Однако учтите, что в момент выполнения оператора RETURN любой незаконченный цикл FOR…NEXT в подпрограмме заканчивается и часть стекового пространства, которое он (цикл) занимал, освобождается!

Пример.
044-04.bas
044-04.bas Вывести на экран дисплея первое нечётное число, встретившееся в целочисленном массиве A(K).

```
NEW
Ok
10 DEFINT A:INPUT K:DIM A(K):DATA 2,1,4,7,8
30 FOR I=1 TO K:READ A(I):NEXTI:GOSUB 100:END
100 FOR J=1 TO K:IF VAL(RIGHT$(STR$(A(J)),1))MOD2=1THEN?A(J):RETURN ELSE NEXTJ
120 PRINT "нечетных элементов в массиве нет":RETURN
run
? 3
 1
Ok

run
? 5
1
Ok
```

При многократном выполнении выхода из подпрограмм с помощью операторов GOTO, ON GOTO или IF…THEN…ELSE (вместо оператора RETURN) в конце концов на экране может появиться сообщение об ошибке:

«Out of memory»
(«*Не хватает памяти*»).

Причина возникающей ошибки довольно своеобразна: дело в том, что интерпретатор отводит сравнительно небольшой участок динамической памяти для хранения списка адресов возврата из подпрограмм. Такой список организуется в виде стека и называется *рабочим стеком*. Каждый раз при вызове подпрограммы в стек заносится соответствующий этому вызову *адрес возврата*, занимающий 7 байт. При выполнении оператора RETURN из стека извлекается адрес возврата, записанный последним, и происходит передача управления по этому адресу, после чего он удаляется из стека. Операторы, подобные GOTO или IF…THEN…ELSE, осуществляют выход из подпрограммы, не

затрагивая стека! Если такой выход из подпрограмм происходит часто, то адреса из стека не удаляются, а лишь добавляются при каждом новом вызове подпрограммы, так что в результате размер стека может превысить величину отведённого ему участка памяти. В итоге программа прекращает работу, и выдаётся сообщение об ошибке.

Пример.
044-05.bas
044-05.bas

```
10 GOSUB 20:END
20 FOR I=1 TO 1:K=K+1:GOTO 10:NEXTI:RETURN
run
Out of memory in 20
Ok

а теперь… print K
·894
Ok
```

В программе может быть несколько операторов RETURN, относящихся к одному оператору GOSUB n. Отметим, что наличие оператора RETURN в некотором месте подпрограммы ещё не означает фактического окончания подпрограммы в данном месте.

Пример. Написать программу, вычисляющую значение функции $y=|x+x^2|$ в точке x=A (не применяя функции ABS()).
044-06.bas
044-06.bas

```
NEW
Ok
10 INPUT A:U=A+A^2:GOSUB 100:PRINT Z:END
100 'Подпрограмма вычисления |x|: аргумент U, результат Z
110 IF U>=0 THEN Z=U:RETURN ELSE Z=-U:RETURN
run
? -4
 12
Ok
```

Подпрограмма может, в свою очередь, вызывать другую подпрограмму, та — другую и так далее. Глубина вложения (степень вложения) подпрограмм ограничивается лишь размерами стека.

Наглядно это можно представить себе следующим образом. Пусть Вас из Куйбышева послали в Москву на повышение квалификации, а оттуда ещё в Ленинград на курсы. По окончании этих курсов Вы возвращаетесь в Москву, заканчиваете учёбу там и только после этого возвращаетесь домой.

Вложенные подпрограммы являются довольно мощным средством разработки и отладки больших программ. Используя принцип вложений, можно не только разбивать сложную программу на отдельные модули и для каждого из них писать свою подпрограмму, но и *разделять* на *модули* любые подпрограммы. Чем меньше будет каждый выделенный программный модуль, тем легче будет его программировать и отлаживать и тем больше вероятность его многократного использования в других программах!

«Трудности, обусловленные бессистемным написанием (без использования модульной структуры) большой и сложной программы, можно сравнить с трудностями, возникающими при попытке съесть сразу *целиком* весь батон колбасы; в то же время, если разрезать колбасу на ломтики, то съесть её не представит никакого труда» (Л.Пул).

Приведём пример, иллюстрирующий вложение подпрограмм:
044-07.bas
044-07.bas

```
NEW
Ok
1' Программа вычисления суммы вида
2'              k    k         k
3'         S = x  + x  +...+ x  ,
4'              1    2         n
5' где  x  -  корни линейного алгебраического уравнения  n-ой степени
```

```
6'      i
7'Алгоритм решения поставленной задачи основан на формулах Ньютона, приведенных в книге: А.П.Мишина,
И.В.Проскуряков "Высшая алгебра". М.:ГИФМЛ, 1962, гл.III, 3, с.245.
11 INPUT "Укажите степень многочлена";N:INPUT"Укажите k";K
15 DIM A(N) 'Описан массив коэффициентов уравнения!
18 PRINT"Вводите коэффициенты уравнения"
20 FOR I=0 TO N:INPUT A(I):NEXT 'Итак, массив A(N) введен!
40 DIM SIG(N):FOR I=1 TO N:SIG(I)=(-1)^I*A(I)/A(0):NEXTI
80 IF N>=K THEN DIM S(N):A1=N:A2=K:GOSUB 200 ELSE DIM S(K):A1=K:A2=N:GOSUB 300
110 PRINT S(K):END
200 '¤¤¤¤¤ Начало подпрограммы 1 ¤¤¤¤¤
205 S(1)=SIG(1):FOR J=2 TO A2:S(J)=(-1)^(J+1)*J*SIG(J)
230 FOR I=1 TO J-1:S(J)=S(J)+(-1)^(J-I+1)*S(I)*SIG(J-I)
250 NEXTI:NEXTJ:RETURN '⟶
300 '¤¤¤¤¤ Начало подпрограммы 2 ¤¤¤¤¤
305 GOSUB 200  'Вот оно, в л о ж е н и е  подпрограмм!
360 FOR J=N+1 TO K:S(J)=0
380 FOR I=1 TO N:S(J)=S(J)-(-1)^S(J-I)*SIG(I)
400 NEXT I,J:RETURN '⟶
run
Укажите степень многочлена? 2
Укажите k? 2
Введите коэффициенты уравнения
? 1
? 0
? -1
 2
Ok
```

IV.4.1. Примеры

Пример 1. Написать программу, вычисления числа сочетаний из m по n по формуле:

$$C_m^n = \frac{m!}{n!(m-n)!}, (m \geqslant n)$$

.

Целесообразно разработать подпрограмму вычисления факториала и затем трижды (почему?) обратиться к ней из основной программы. Итак,
0441-01.bas
0441-01.bas

```
NEW
Ok
40 INPUT "Введите значения m,n";M,N:GOTO 100' ──┐
50 'Подпрограмма вычисления факториала F!      │
60 F=1:FOR J=1 TO K:F=F*J:NEXTJ'                │
90 RETURN 'Конец подпрограммы!                  │
100 K=M:GOSUB 60:M1=F'     ◄────────────────────┘
110 K=N:GOSUB 60:N1=F
120 K=M-N:GOSUB 60:R1=F
130 CMN=M1/N1/R1
140 ? "Число сочетаний из M=";M;"по N=";N;"равно";CMN:END
run
Введите значения m,n? 6,3
Число сочетаний из M= 6 по N= 3 равно 20
Ok
```

Заметим, что оператор GOSUB может оказаться полезным в качестве команды прямого режима. Если Ваша программа состоит из подпрограмм, которые Вы хотите проверить, то, используя команду GOSUB в прямом режиме, можно войти в любую подпрограмму. После выполнения оператора RETURN система MSX BASIC снова вернётся в режим прямого выполнения команд.

В примере 1, например, можно проверить работоспособность подпрограммы следующим образом (находясь в прямом режиме!):

```
F=1:k=4:GOSUB 60
Ok
print F
 24
Ok
```

> Ибо это недостойно совершенства человеческого, подобно рабам тратить часы на вычисления.
>
> *—Г.В.Лейбниц*

Пример 2. Составить программу, которая позволяет методом половинного деления (дихотомии) на отрезке [A,B] с точностью E определить какой–либо (!) вещественный корень уравнения F(x)=0, где F(x) непрерывна на [A,B] и на концах отрезка принимает значения разных знаков (F(A)·F(B)<0). Пусть, например, $F\left(x\right)=\cos e^{x}-\sin \pi^{x}, A=1, B=3, E=0.00001$.

0441-02.bas

0441-02.bas

```
NEW
Ok
10 INPUT A,B,E
20 X=A:GOSUB 200:F1=Y
30 X=(A+B)/2
40 GOSUB 200:F2=Y:IF Y=0 THEN 90
50 IFSGN(F1)*SGN(F2)>0 THEN 70
60 B=X:GOTO 80
70 A=X:F1=F2
80 IF B-A>E THEN 30
90 PRINTUSING "Корень####.##### точность##^^^^";X,E
100 END
200 Y=COS(EXP(X))-SIN(EXP(X*LOG(3.14159)))
210 RETURN
run
? 1,3,.00001
Корень···1.27163·точность·1E-05
Ok
```

Пример 3. Если у Вас есть знакомый художник, попросите его разделить произвольный отрезок на две неравные части. Возможно он сделает это в золотом отношении, если его чувство меры воспитано на классических образцах.

Такое чувство меры целесообразно «внушить» компьютеру, которому часто приходится отыскивать единственную точку минимума некоторой функции y=f(x) на отрезке [a,b]. Основа алгоритма — последовательное приближение к минимуму до достижения заданной точности E.

0441-03.bas

0441-03.bas

```
NEW
Ok
10 'Поиск минимума функции  y=f(x)  на отрезке  [a,b]  методом золотого сечения
20 DEFFN Y(X)=-X^2+5: 'Вид функции
30 INPUT"A,B,точность";A,B,E
40 GOSUB 110:GOSUB120
50 IF ABS(B-A)<E THEN 90
60 IF Y1>Y2 THEN 70
65 B=X2:X2=X1:Y2=Y1:GOSUB 110:GOTO 80
70 A=X1:X1=X2:Y1=Y2:GOSUB120 '⟶
80 GOTO 50
```

```
90 X=(A+B)/2:PRINT"Y мин=";FNY(X);"при x=";X
100 END
110 X1=.618*A+.382*B:Y1=FNY(X1):RETURN '⟶
120 X2=.382*A+.618*B:Y2=FNY(X2):RETURN '⟶
run
A,B,точность? -1,4
?? 0.00001
Y мин=-10.999971821564 при x= 3.999996477694
Ok
```

Пример 4. Вычислить

$$\frac{1}{\pi} \times \int_0^{\pi} \left[\frac{\sin\left(10 \times x\right)}{\sin\left(x\right)} \right]^6 dx$$

по формуле парабол (формуле Симпсона). Внимание! Интеграл несобственный!

0441-04.bas
0441-04.bas

```
NEW
Ok
20 INPUT "Нижний предел";A
30 INPUT "Верхний предел";B
40 INPUT "Количество точек";M
70 S=0:H=(B-A)/M
80 S1=0:FOR X=A+H/2 TO B-H/2 STEP H
90 GOSUB 180:S1=S1+Y:NEXT:S1=S1*4
100 S2=0:FOR X=A+H TO B-H STEP H
110 GOSUB 180:S2=S2+Y:NEXT:S2=S2*2
120 X=A:GOSUB 180:S=S+Y
130 X=B:GOSUB 180:S=S+Y
140 S=(S+S1+S2)*H/6
160 PRINT"Интеграл = ";S:END
180 Y=(SIN(10*X)/SIN(X))^6/(4*ATN(1)):RETURN ⟶
run
Нижний предел? 0.00000001
Верхний предел? 3.14159265
Количество точек? 250
Интеграл = 55251.99567425
Ok

run
Нижний предел? 0.00000001
Верхний предел? 3.14159265
Количество точек? 500
Интеграл = 55251.995674258
Ok
```

Пример 5. Написать программу, осуществляющую лексикографическое упорядочение (расположение в алфавитном порядке) массива фамилий на русском языке (после фамилий можно указывать инициалы, например: Бобров А.В.).
0441-05.bas
0441-05.bas

```
NEW
Ok
10                                        DATA                                                  "
",".","А","Б","В","Г","Д","Е","Ж","З","И","Й","К","Л","М","Н","О","П","Р","С","Т","У","Ф","Х","Ц","Ч","Ш","Щ","Ы","Ь","Э","Ю",
"Я"
15                              DATA                                "                              ","."," -
","а","б","в","г","д","е","ж","з","и","й","к","л","м","н","о","п","р","с","т","у","ф","х","ц","ч","ш","щ","ы","ь","э","ю","я"
20 INPUT"Введите количество фамилий";S:DIM C$(S)
30 FOR I=1TOS:INPUT"Введите очередную фамилию";C$(I):NEXTI
```

```
40 GOSUB 190
50 DIM N(L1,S)
56 FOR I=1 TO S:RESTORE 10:FOR J=1 TO 33:READB$
57 IF B$=MID$(C$(I),1,1) THEN N(1,I)=J
58 NEXTJ
70 FOR K=2 TO LEN(C$(I))-4
71 IF MID$(C$(I),K-1,1)="-"THEN RESTORE 10 ELSE RESTORE 15
80 FOR J=1 TO 34
90 READ B$:IF B$=MID$(C$(I),K,1) THEN N(K,I)=J
100 NEXT J,K
101 FOR K=LEN(C$(I))-3 TO L1:RESTORE 10
103 FOR J=1 TO 33
105 READ B$:IF B$=MID$(C$(I),K,1) THEN N(K,I)=J
107 NEXT J,K
108 NEXT I
110 FOR K=1 TO L1
120 FOR I=1 TO S-1:FORJ=I+1 TO S
130    IF    MID$(C$(I),1,K)=MID$(C$(J),1,K)    THEN    GOTO    140    ELSE
IF(N(K,I)>N(K,J))AND(MID$(C$(I),1,K-1)=MID$(C$(J),1,K-1)) THEN SWAP C$(I),C$(J):FOR L=1 TO
L1:SWAP N(L,I),N(L,J):NEXT L
140 NEXT J,I,K
150 CLS 'Очистим экран дисплея!
160 PRINT SPC(10);"Итоговый список"
170 FOR I=1 TO S:PRINT SPC(10);STR$(I);". ";C$(I):NEXTI
180 END
190 L1=0:FOR P=1 TO S:IF LEN(C$(P))>L1THEN L1=LEN(C$(P))
200 NEXT:RETURN
run
Введите количество фамилий? 4
Введите очередную фамилию? Ухин И.Н.
Введите очередную фамилию? Ми-Ка М.У.
Введите очередную фамилию? Ми-Ша А.А.
Введите очередную фамилию? Лим А.
Итоговый список
1. Лим А.
2. Ми-Ка М.У.
3. Ми-Ша А.А.
4. Ухин И.Н.
Ok
```

Пример 6. Абсолютно простым числом назовём натуральное число, обладающее следующим свойством: при любой перестановке цифр данного числа образуется также простое число. Найдите все абсолютно простые числа, принадлежащие [C,E].

0441-06.bas

0441-06.bas

```
NEW
Ok
1 CLS 'Вначале очистим экран!
2 INPUT"Введите два натуральных числа через запятую, причём второе число должно быть больше первого (эти числа задают диапазон, в котором ищутся абсолютно простые числа)";C,E
3 PRINT"Вы хотите узнать всю правду об абсолютно простых числах, лежащих на отрезке [";C;",";E;"] ? Пожалуйста..."
4  FOR  A=C  TO  E:IF  A<10AND(A=1ORA=2ORA=3ORA=5ORA=7)THEN  PRINTA;:NEXTA  ELSE  IF
A<10AND(A=4ORA=6ORA=8ORA=9)THEN NEXTA ELSE A$=MID$(STR$(A),2)
5 N=LEN(A$):DIM A(N)
10 FOR L=1TO N:A(L)=VAL(MID$(A$,L,1)):NEXT L:GOSUB 200
20 I=N
30 GOTO 110
40 GOSUB 300:J=N-1
50 IF A(J)>=A(J+1)ANDJ>0 THEN J=J-1:GOTO 50
60 I=J
70 IF I>0 THEN J=N:GOTO 80 ELSE GOTO 110
80 IF A(I)>=A(J) THEN J=J-1:GOTO 80
```

```
90 SWAP A(I),A(J):P1=I+1:P2=N
100 IF P1<P2 THEN SWAP A(P1),A(P2):P1=P1+1:P2=P2-1:GOTO100
110 IF I>0 THEN GOTO 40
120 ERASEA:IF JJ<>1 THEN PRINT A;:NEXT AELSE JJ=0:NEXTA
125 PRINT:PRINT"Надеюсь, Вы остались довольны?!"
130 END
190 'Подпрограмма, производящая сортировку массива A(N), состоящего из цифр числа A.
200 FOR M=1 TO N-1:FOR S=M+1 TO N
210 IFA(M)<=A(S)THEN NEXT:NEXT ELSE SWAP A(M),A(S):NEXT:NEXT
220 RETURN
290 'Подпрограмма, позволяющая определить, является ли число V, составленное из цифр числа A, простым.
300 V=0:FOR P=1TON:V=V+A(P)*10^(N-P):NEXT
310 FOR Q=2 TO INT(SQR(V))
320 IF VMODQ=0 THEN JJ=1:RETURN ELSE NEXT Q:RETURN
run

Введите два натуральных числа через запятую, причём второе число должно быть больше первого (эти числа задают диапазон, в котором ищутся абсолютно простые числа)? 1,100

Вы хотите узнать всю правду об абсолютно простых числах,лежащих на отрезке [1, 100] ? Пожалуйста...
·1··2··3··5··7··11··13··17··31··37··71··73··79··97
Надеюсь, Вы остались довольны?
Ok
```

Пример 7.
0441-07.bas
0441-07.bas

```
10'*****************************************************
30'*        Программа   LONGMULT                    *
50'*  позволяет найти точное значение произведения  *
70'*       не более чем 210-значных чисел       *
81'*        Max. время работы - 1.5 мин.      *
83'*****************************************************
90 CLEAR 1000:DIMA(30),C(31,61),D(62),D$(62),B(30),S(62):CC=10000000!
92 DEFFND$(X)=RIGHT$("0000000"+RIGHT$(STR$(X),LEN(STR$(X))-1),7):CLS
100 PRINT"Введите 1-й множитель":PRINT"a=";:GOSUB 150
110 FOR I=1TO N:A(I)=S(1):NEXT:NA=N
120 ?:PRINT"Введите 2-й множитель":PRINT"b=";:GOSUB 150
130 FOR I=1 TO N:B(I)=S(I):NEXT:NB=N:GOSUB 190
135 IF D(1)=0 THEN S=2 ELSE S=1
140 PRINT:PRINT"a*b=";:FOR I=S TO NA+NB:PRINT FND$(D(I));:NEXT
145 PRINT :PRINT:END
150 VV$=""
160 S$=INKEY$:IF S$=""THEN160 ELSE IF S$=CHR$(13) THEN PRINT:GOTO170 ELSE IF ASC (S$)<48 OR
ASC(S$)>57 THEN160 ELSE PRINT S$;:VV$=VV$+S$:GOTO160
170 N=LEN(VV$):IFNMOD7=0THEN180 ELSE VV$="0"+VV$: GOTO170
180 N=N/7:FOR I=1 TO N:S(I)=VAL(MID$(VV$,1+(I-1)*7,7)):NEXT:RETURN
190 '*** Собственно длинное умножение ***
200 FOR I=NB TO 1 STEP-1:S=0
210 FOR J=NA TO 0 STEP-1
220 C(I,J+I)=S+B(I)*A(J)-INT((S+B(I)*A(J))/CC)*CC:S=INT((S+B(I)*A(J))/CC):NEXT:NEXT
230                                                                    S=0:FORJ=NA+NBTO1
STEP-1:FORI=1TONB:D(J)=D(J)+C(I,J):NEXT:D(J)=D(J)+S:S=INT((D(J)+S)/CC):D(J)=D(J)-
INT(D(J)/CC)*CC:NEXT:RETURN
run
Введите 1-й множитель
a=1111111111111111111111111111111111111111111111111
Введите 2-й множитель
b=1000000000000000000000000000000000000000000000001
a*b=11110001111000111100011110001111000
 111100011110001112111000111100011110001
 11100011110001111000111100011111
```

```
Ok
```

У попа была собака, он её любил.
Она съела кусок мяса — он её убил!
Убил и закопал, и надпись написал:
«У попа была собака, он её любил.
Она съела кусок мяса — ...»

—Из русского народного фольклора

Допустимо использование рекурсивных подпрограмм.

Рекурсией называется обращение подпрограммы (функции) к самой себе непосредственно или посредством других подпрограмм. Такие программы (функции) называются *рекурсивными*. Само слово «рекурсия» означает «возвращение» (лат. «recursio»).

В программировании различают два типа рекурсии — рекурсивное определение или *прямую* рекурсию и рекурсивное использование или *косвенную* рекурсию.

Рекурсивное использование — это способ описания процесса через подпроцессы, идентичные основному процессу, а рекурсивное определение — способ описания процесса через самого себя.

Рекурсия получается в программах, реализующих алгоритмы, которые содержат *рекуррентные* формулы, то есть формулы, в которых значение функции от некоторого аргумента X выражается через значение этой же функции, но от другого аргумента Y, который в каком–то смысле «предшествует» X.

Рекуррентной является, например,формула вычисления факториала:

```
   ⎧    1, если k=1,
F(k)= ⎨
      ⎩  F(k-1)·k, если k>1.
```

Пример 1. Составим программу вычисления факториала натурального числа K, содержащую рекурсивную подпрограмму.
0441-11.bas
0441-11.bas

```
NEW
Ok
10 INPUT K:F=1:GOSUB 100 '⟶
30 PRINT F:END
90 'Внимание! Рекурсивная подпрограмма!
100 IF K>1 THEN F=F*K:K=K-1:GOSUB 100 '⟶
110 RETURN '⟶
run
? 4
 24
Ok

run
? 6
720
Ok

run
? 48
1.2413915592536E+61
Ok

run
? 49
Overflow in 100
```

```
Ok
```

Конечно, любая рекурсивная подпрограмма может быть написана как циклическая программа. Например, последнюю программу можно переписать следующим образом:

```
10 INPUT K:F=1
20 IF K>1 THEN F=F*K:K=K-1:GOTO 20
30 PRINT F:END
```

Но многим программистам больше нравится использование рекурсивных подпрограмм, несмотря на то, что коварство рекурсии проявляется в том, что бывает довольно трудно заметить дефект программы или алгоритма, а в циклическом процессе все на виду!

Пример 2. Пользуясь рекуррентной формулой

$$C_n^0 = 1; C_n^m = \frac{n-m+1}{m} \times C_n^{m-1}, (*)$$

вычислить число сочетаний из n по m при $m, n = 1, k, k \geqslant 1, m \leqslant n$.

Приведём два варианта программы.

- a) 0441-121.bas
  0441-121.bas

```
10 INPUT K
20 FOR N=0 TO K:C=1:FOR M=1 TO N:IF M>N THEN NEXTN ELSE C=(N-M+1)/M*C:PRINT N;M;FIX(C+.5):NEXT:NEXT
run
? 3
·1··1··1
·2··1··2
·2··2··1
·3··1··3
·3··2··3
·3··3··1
Ok
```

- b) 0441-122.bas
  0441-122.bas

```
NEW
Ok
5 'Программа находит число сочетаний из  N  по  M  по формуле (*) с применением рекурсивной подпрограммы.
10 INPUT N,M:C=N:GOSUB100:PRINT FIX(C+.5):END
100 IF M>1THEN C=C*(N-M+1)/M:M=M-1:GOSUB100:RETURN ELSE RETURN
run
? 232,231
  232
Ok

run
? 233,232
 0   ←— ?!
Ok

run
? 4080,4079
Out of memory in 100
Ok

далее...
Ok
print m
 2
Ok
```

Пример 3 [6].
0441-131.bas

0441-131.bas

```
NEW
Ok
100 INPUT "Строка для сжатия:";FS$
109 'Удаление пробелов из FS$.
110 S$=FS$:RMV$=" ":GOSUB 200
130 PRINT S$:END
140 '**** Рекурсивная подпрограмма сжатия строк. ****
150 'B RMV$ - удаляемые символы
160 'B S$ - исходная строка и результат
200  IF  INSTR(1,S$,RMV$)<>0  THEN  S$=LEFT$(S$,INSTR(1,S$,RMV$)-1)+RIGHT$(S$,LEN(S$)-
INSTR(1,S$,RMV$)):GOSUB 200
210 RETURN
```

Сравните:
0441-132.bas
0441-132.bas

```
NEW
Ok
10 INPUT"Строка для сжатия:";A$
20 FOR I=1 TO LEN(A$):C$=MID$(A$,I,1)
30 IF C$<>" " THEN B$=B$+C$
40 NEXT:PRINT B$
```

Существует ограничение на число обращений подпрограммы к себе самой. Оно зависит от нескольких факторов, в частности от количества других вложенных одна в другую подпрограмм, выполняемых в этот же момент времени, и от размеров рабочего стека.

Отметим, что понятие рекурсии охватывает также и весьма интересную ситуацию, при которой первая подпрограмма вызывает вторую, а та в свою очередь вызывает первую!

IV.5. Оператор ON GOSUB

Кроме оператора перехода на подпрограмму в MSX BASIC есть оператор выбора подпрограмм (оператор—«переключатель»).

Его общий вид:

```
ON β GOSUB n1,n2,…nk
```
,

где:

- ON(«на»), GOSUB(«to GO to SUBroutine» — «идти к подпрограмме») — служебные слова;
- β — арифметическое выражение;
- n1,n2,...,nk — номера начальных строк подпрограмм.

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

Далее, если p ∈ {1,2,...,k}, где k — целое положительное число, то управление переходит к строке программы.

Если p=0 или p>k, то выполняется оператор, следующий за ON. После выполнения соответствующей подпрограммы управление передаётся оператору, расположенному за ON GOSUB .

Если p<0, то компьютер выдаёт сообщение об ошибке:

«Illegal function call».

Пример 1.
045-01.bas
045-01.bas

```
NEW
Ok
20 INPUT "Ваш выбор (1-3)";CH
27 ON CH GOSUB 31,32,33:END
31 PRINT "1":RETURN' Попадаем сюда, если CH=1
32 PRINT "2"'RETURN' Попадаем сюда, если CH=2
33 PRINT "3"'RETURN' Попадаем сюда, если CH=3
run
Ваш выбор (1-3)? 2
2
Ok
```

Пример 2.
045-02.bas
045-02.bas

```
NEW
Ok
10 PRINT "Draw,Print,Change,Quit:";:I$=INPUT$(1)
20 ON INSTR(2," DdPpCcQq",I$)/2 GOSUB 100,200,300,400
30 PRINT:GOTO 10
100 PRINT "D":RETURN '⟶
200 PRINT "P":RETURN '⟶
300 PRINT "C":RETURN '⟶
400 PRINT "Q":RETURN '⟶
run
Draw,Print,Change,Quit:
Draw,Print,Change,Quit:      ⟵ нажата клавиша "a"
Draw,Print,Change,Quit:Q     ⟵ нажата клавиша  "q"
Draw,Print,Change,Quit:P     ⟵ нажата клавиша  "p"
…
```

Обратите внимание на примеры из раздела III.3.!

Выводы [8]. При применении подпрограмм BASIC придерживайтесь следующих правил:

1. Чётко обозначайте начало и конец каждой подпрограммы. Если это возможно, выделяйте их путём отступа операторов вправо.
2. Никогда не пользуйтесь оператором GOTO для входа в подпрограмму и для выхода из неё. Каждую подпрограмму надо рассматривать как независимый логически завершённый *модуль*.
3. Имейте под рукой список глобальных входных переменных и выходных результатов каждой подпрограммы. Лучше всего оформить его комментариями к программе с помощью операторов REM.
4. Обязательно убедитесь, что в подпрограмме не изменяются значения таких внешних переменных, как счётчики циклов. Если это возможно, в каждой подпрограмме вводите свой принцип именования внутренних переменных.
5. В отличии от функций, подпрограммы не появляются в программе до того, как ими будут пользоваться, поэтому полезно группировать подпрограммы вместе после оператора END, указанного в конце основной программы.

IV.6. Дополнение 1 [77]

[77]

Программа позволяет умножить два числа, причём каждое из них может иметь «*любое*» количество цифр при следующих ограничениях: в произведении должно содержаться не более 765 цифр и длина каждого из чисел должна быть ограничена возможной длиной строки.

В заключение отметим, что приведённая программа работает неправильно либо когда одно из введённых чисел

меньше 10, либо когда их произведение меньше 1000. Безусловно имеет смысл сделать программу «защищённой» в этом отношении!
046-01.bas
046-01.bas

```
5  CLEAR 1000
10 DIM A(255),B(255),C(255)
20 T=1000:Z$="000"
30 PRINT"Точное произведение целых чисел"
40 INPUT"Введите первое число: ";A$
50 LA=LEN(A$):NA=INT((LA-1)/3)
60 FOR I=0 TO NA-1
70    A(I)=VAL(MID$(A$,LA-2-3*I,3))
80 NEXT I
90 A(NA)=VAL(LEFT$(A$,LA-3*NA))
100 INPUT"Введите второе число: ";B$
110 LB=LEN(B$):NB=INT((LB-1)/3)
120 FOR I=0 TO NB-1
130    B(I)=VAL(MID$(B$,LB-2-3*I,3))
140 NEXT I
150 B(NB)=VAL(LEFT$(B$,LB-3*NB))
160 NC=NA+NB+1
170 FOR I=0 TO NC:C(I)=0:NEXT I
180 FOR I=0 TO NA
190    FOR J=0 TO NB
200       K=I+J:X=A(I)*B(J)+C(K)
210       Y=INT(X/T):C(K)=X-T*Y
220       IF Y>0 THEN K=K+1:X=Y+C(K):GOTO 210
230    NEXT J
240 NEXT I
250 PRINT "Произведение = ";
260 IF C(NC)=0 THEN NC=NC-1
270 PRINT C(NC)
280 FOR I=NC-1 TO 0 STEP -1
290    PRINT",";RIGHT$(Z$+STR$(C(I)),3);
300 NEXT I
320 END
run
Точное произведение целых чисел
Введите первое число: 12345678
Введите второе число: 987654321
Произведение =  12,193,262,222,374,638
Ok
print 12345678*987654321
 1.2193262222375E+16
Ok
run
Точное произведение целых чисел
Введите первое число: 1111111111111111111
Введите второе число: 666666666666666666666666666666666666666666
Произведение =  74, 74, 74, 74, 74, 74, 73,333,333,333,333,333,333,325,925,925,925,925,925,926
Ok
print 1111111111111111111*666666666666666666666666666666666666666666
 7.4074074074074E+58
Ok
```

IV.7. Дополнение 2 [90]

[90]

Нахождение факториалов «огромных» чисел.
047-01.bas

047-01.bas

```
10 DIM A(100):PRINT"Введите число N":INPUT N:A(1)=1:J=1
60 FOR I=2 TO N
80    P=0
90    FOR K=1 TO J
100      C=A(K)*I+P:P=INT(C/10):A(K)=C-P*10
130   NEXT K
140   IF P=0 GOTO 200
150   J=J+1
160   IF J>100 THEN PRINT"Не хватает места в массиве":STOP
170   A(J)=P-INT(P/10)*10:P=INT(P/10)
190   GOTO 140
200 NEXT I
210 PRINT STR$(N);"!=";
220 FOR K=J TO 1 STEP -1:PRINT RIGHT$(STR$(A(K)),1);:NEXT K
run
Введите число N
? 23
 23!=25852016738884976640000
Ok
run
Введите число N
? 50
 50!=30414093201713378043612608166064768844377641568960512000000000000
Ok
```

Диск с примерами

Загрузить образ диска

Открыть диск в WebMSX

---

1)

Примечание редактора: Это символ называется «точка по центру» или «знак умножения», подробнее здесь

http://sysadminmosaic.ru/msx/basic_programming_guide/04?rev=1583570504

2020-03-07 11:41