

**МИКРОКАЛЬКУЛЯТОР**

**"ЭЛЕКТРОНИКА МК 90"**

**МОДЕРНИЗИРОВАННОЕ  
РУКОВОДСТВО ПО ЭКСПЛУАТАЦИИ**

**1988**

## Содержание

### ЛИСТ

1. Общие указания .....	4
2. Технические данные .....	4
3. Комплект поставки .....	6
4. Требования по технике безопасности .....	6
5. Подготовка к работе .....	6
6. Порядок работы .....	9
6.1. Режимы работы клавиатуры и ЖКИ .....	9
6.1.1. Режим загрузки .....	9
6.1.2. Специальные клавиши .....	11
6.1.3. Работа клавиатуры в режиме верхнего и нижнего регистров .....	12
6.1.4. Работа клавиатуры в режимах русского/латинского регистров .....	14
6.1.5. Работа клавиатуры в функциональном режиме ...	15
6.1.6. Режим работы ЖКИ .....	16
6.2. Функции вычисления .....	17
6.2.1. Константы и точность вычисления .....	17
6.2.2. Арифметические функции БЕЙСИКа микро-калькулятора .....	17
6.2.3. Переменные .....	19
6.3. Команды БЕЙСИКа микрокалькулятора .....	20
6.3.1. Команды непосредственного режима .....	20
6.3.1.1. Команда AUTO .....	20
6.3.1.2. Команда DELETE .....	21
6.3.1.3. Команда EDIT .....	21
6.3.1.4. Команда HELP .....	22
6.3.1.5. Команда LIST .....	22
6.3.1.6. Команда MEM .....	23
6.3.1.7. Команда INIT .....	23
6.3.1.8. Команда SAVE .....	24
6.3.1.9. Команда NAME/AS .....	24
6.3.1.10. Команда LOAD .....	25
6.3.1.11. Команда KILL .....	25
6.3.1.12. Команда DEV .....	25
6.3.1.13. Команда FILES .....	26
6.3.1.14. Команда RUN .....	27
6.3.2. Команды программного режима .....	27
6.3.2.1. Оператор DEF FN .....	27
6.3.2.2. Оператор DIM .....	30
6.3.2.3. Оператор END .....	33
6.3.2.4. Оператор FOR-TO-STEP/NEXT .....	33
6.3.2.5. Операторы GOSUB, RETURN .....	34
6.3.2.6. Оператор GOTO .....	35
6.3.2.7. Оператор IF/THEN .....	37
6.3.2.8. Оператор INPUT .....	39
6.3.2.9. Оператор LET .....	39
6.3.2.10. Оператор PRINT .....	40

6.3.2.11.	Операторы READ, DATA, RESTORE .....	42
6.3.2.12.	Оператор REM (REMARK) .....	45
6.3.2.13.	Оператор RANDOMIZE .....	46
6.3.2.14.	Оператор CLS .....	47
6.3.2.15.	Оператор DRAW .....	47
6.3.2.16.	Оператор LOCATE .....	53
6.3.2.17.	Оператор STOP .....	54
6.3.2.18.	Оператор DIS .....	54
6.3.2.19.	Оператор WAIT .....	54
6.3.2.20.	Оператор PLAY .....	55
6.3.3.	Встроенные функции .....	56
6.3.3.1.	Функции SIN, COS .....	56
6.3.3.2.	Функция ATN .....	56
6.3.3.3.	Функция LOG .....	57
6.3.3.4.	Функция EXP .....	59
6.3.3.5.	Функция ABS .....	59
6.3.3.6.	Функция INT .....	60
6.3.3.7.	Функция INC .....	61
6.3.3.8.	Функция SGN .....	61
6.3.3.9.	Функция SQR .....	61
6.3.3.10.	Функция RND .....	62
6.3.3.11.	Функция PI .....	62
7.	Правила хранения .....	63
Приложение 1.	Схема клавиатуры для работы в функцио- нальном режиме .....	64
Приложение 2.	Команды БЕЙСИКа микрокалькулятора .....	65
Приложение 3.	Сообщения об ошибках .....	76
Приложение 4.	Примеры программ .....	79
Приложение 5.	Таблицы кодов символов .....	88
Приложение 6.	Рекомендации по работе с СМП .....	91

## 1. Общие указания

- 1.1. Перед началом эксплуатации микрокалькулятора "Электроника МК90" внимательно ознакомьтесь с настоящим руководством по эксплуатации, соблюдение требований которого обеспечит надежную работу изделия на протяжении длительного времени.
- 1.2. Для обеспечения длительной и надежной работы микрокалькулятора избегайте эксплуатации или хранения изделия в местах, подверженных резкому изменению температуры, повышенной влажности, не допускайте нагревания корпуса микрокалькулятора от попадания прямых солнечных лучей, берегите от механических повреждений.
- 1.3. Не снимайте заглушку с диагностического разъема микрокалькулятора в целях предохранения изделия от статического электричества.
- 1.4. Используйте для протирки корпуса микрокалькулятора только мягкую и чистую ткань.
- 1.5. Не подвергайте микрокалькулятор механическим ударам, т.к. экран жидкокристаллического индикатора (ЖКИ) изготовлен из стекла.
- 1.6. Не прилагайте больших усилий к регулятору контрастности ЖКИ во избежание его поломки.
- 1.7. Не допускайте хранения сменного модуля памяти (СМП) в микрокалькуляторе без установленных элементов питания. СМП следует хранить в футляре, предназначенном для его хранения и входящем в комплект поставки.
- 1.8. Не прикасайтесь к контактам разъема СМП в целях предохранения СМП от статического электричества.
- 1.9. После транспортировки микрокалькулятора в зимних условиях перед эксплуатацией выдержите его при комнатной температуре в течение двух часов.

## 2. Технические данные

- 2.1. Микрокалькулятор "Электроника МК90" предназначен для выполнения научных, инженерных, статистических расчетов; для оперативной обработки, хранения и отображения различной текстовой, символьной и графической информации.
- 2.2. Микрокалькулятор предоставляет пользователю следующие возможности:
  - Работа в качестве информационного устройства с использованием персонального набора программ, формируемого пользователем и хранящегося в СМП;
  - Ввод, редактирование, отладка и выполнение программ на языке программирования БЕЙСИК;
  - Получение справочной информации о назначении и формате более 50 команд и операторов языка БЕЙСИК;
  - Построение различных графиков, таблиц, диаграмм и схем на экране ЖКИ;
  - Вывод алфавитно-цифровой информации на экран ЖКИ в различном масштабе и направлении;
  - Редактирование текста;
  - Управление звуковым сигналом.

- 2.3. Микрокалькулятор обеспечивает работу с языком программирования БЕЙСИК.
- 2.4. Система счисления при вводе и выводе информации – десятичная.
- 2.5. Количество разрядов мантииссы числа – семь.
- 2.6. Количество разрядов порядка числа – три.
- 2.7. Диапазон представленных чисел  $\pm 10^{-980}$  до  $\pm 10^{979} - 1$ .
- 2.8. Число регистров памяти – 286.
- 2.9. Емкость памяти ОЗУ в распоряжении пользователя – 11824 байта.
- 2.10. Микрокалькулятор обеспечивает работу с двумя СМП.
- 2.11. Емкость памяти энергозависимого ОЗУ каждого СМП – 10 КБайт, в том числе в распоряжении пользователя – 8 Кбайт (16 блоков).
- 2.12. СМП обеспечивает хранение информации в отключенном от микрокалькулятора положении в течение 1,5 года со дня установки элемента питания в СМП и при:
  - Температуре окружающего воздуха  $25 \pm 10^{\circ}\text{C}$ ;
  - Относительной влажности воздуха не более 80%;
  - Атмосферном давлении 630–800 мм рт.ст.
- 2.13. Ввод информации осуществляется:
  - 1) с клавиатуры
  - 2) с двух СМП.
- 2.14. Вывод информации осуществляется:
  - 1) на графический ЖКИ (120 столбцов на 64 строки, для символьного представления информации – 8 строк по 20 символов);
  - 2) на два СМП.
- 2.15. Питание микрокалькулятора осуществляется от четырех аккумуляторов НКГЦ-0,45 ПС ПБЦ3.579.000 ТУ. Допускается вместо аккумуляторов типа НКГЦ-0,45 использовать источники питания типа А-316 "КВАНТ" или "ПРИМА". Питание микрокалькулятора может осуществляться от внешнего блока питания с выходным напряжением 4,5–6,0 В и с током нагрузки не менее 0,150 А.
- 2.16. Полностью заряженные аккумуляторы типа НКГЦ-0,45 обеспечивают:
  - 1) в режиме выполнения программ пользователя не менее 5–6 Ч непрерывной работы микрокалькулятора;
  - 2) в режиме набора и редактирования текста не менее 15–20 Ч работы.Ток потребления в состоянии "выключено" – не более 0,7 мА.
- 2.17. Микрокалькулятор предназначен для работы при температурах от  $+1$  до  $50^{\circ}\text{C}$ , относительной влажности воздуха до 80% при  $25^{\circ}\text{C}$ .
- 2.18. Габаритные размеры микрокалькулятора не более:
  - длина 255 мм;
  - ширина 100 мм;
  - высота 33,5 мм;
- 2.19. Масса не более 0,55 КГ.
- 2.20. Максимальная электрическая мощность, потребляемая микрокалькулятором не более 0,55 Вт.
- 2.21. Содержание драгоценных материалов:
  - золото – 0,6252 г;
  - серебро – 0,6846 г;
  - платина – 0,07510 г;
  - палладий – 0,3873 г;
  - рутений – 0,0002967 г.
- 2.22. Суммарная масса цветных металлов и их сплавов:
  - алюминий и алюминиевые сплавы – 3,3468 г;
  - медь и сплавы на медной основе – 21,06 г.

### 3. Комплект поставки

3.1. В комплект поставки микрокалькулятора входит:

— Микрокалькулятор "Электроника МК90" БК0.310.095 ТУ .....	1 шт.
— Кожух 8.634.022 .....	1 шт.
— Чехол 4.166.005 .....	1 шт.
— Модуль памяти 3.065.003 .....	2 шт.
— Футляр 4.161.016 .....	2 шт.
— Коробка 8.865.405 .....	2 шт.
— Аккумулятор НКГЦ-0,45 II С ПБЦ3.579.000 ТУ .....	4 шт.
— Зарядное устройство "Электроника ЗУ-1" в упаковке с руководством по эксплуатации 12.МО.081.159 ТУ .....	1 шт.
— Руководство по эксплуатации 3.055.003 РЭ .....	1 шт.
— Коробка упаковочная 4.180.267 .....	1 шт.

### 4. Требования по технике безопасности

4.1. Запрещается вскрывать микрокалькулятор и производить ремонтные и наладочные работы при подключенном блоке питания.

4.2. По окончании работ, а также в случае появления неисправностей выключить микрокалькулятор и отсоединить блок питания сначала от сети переменного тока с напряжением 220 В, а затем от микрокалькулятора (при работе от блока питания).

### 5. Подготовка к работе

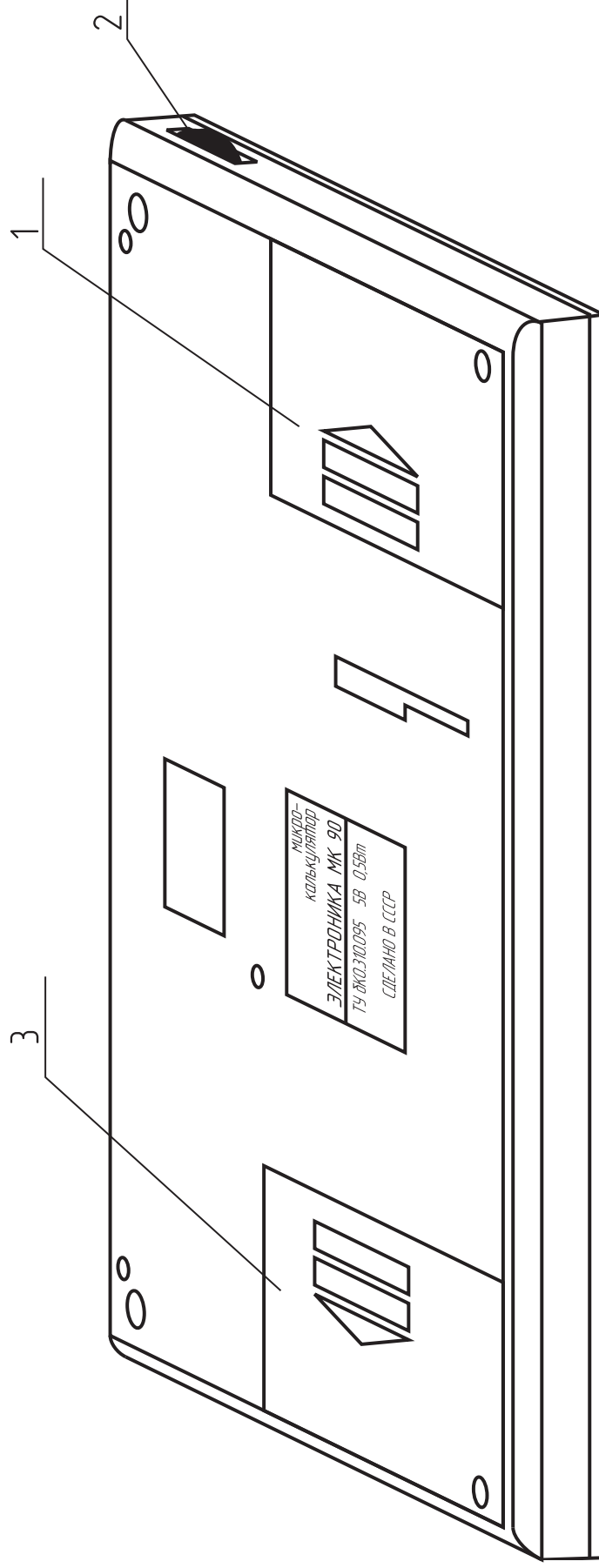
5.1. Микрокалькулятор поставляется с комплектом незаряженных аккумуляторов НКГЦ-0,45.

Зарядку аккумуляторов производить с помощью зарядного устройства "Электроника ЗУ-1" 12.МО.081.159 ТУ, входящего в комплект поставки.

5.2. Установить элементы питания в микрокалькулятор, для чего:

- 1) снять крышку 1 (см. рис. 5.1) батарейного отсека (слегка нажать на рифление крышки и сдвинуть ее в направлении стрелки, обозначенной на крышке);
- 2) установить элементы литания, соблюдая полярность, указанную в батарейном отсеке;
- 3) установить на место крышку батарейного отсека.

## Задняя панель микрокомпьютера



-7-

Рис. 5.1

- 1 — крышка батарейного отсека,
- 2 — регулятор контрастности,
- 3 — крышка отсека СМП.

## Лицевая панель микрокомпьютера

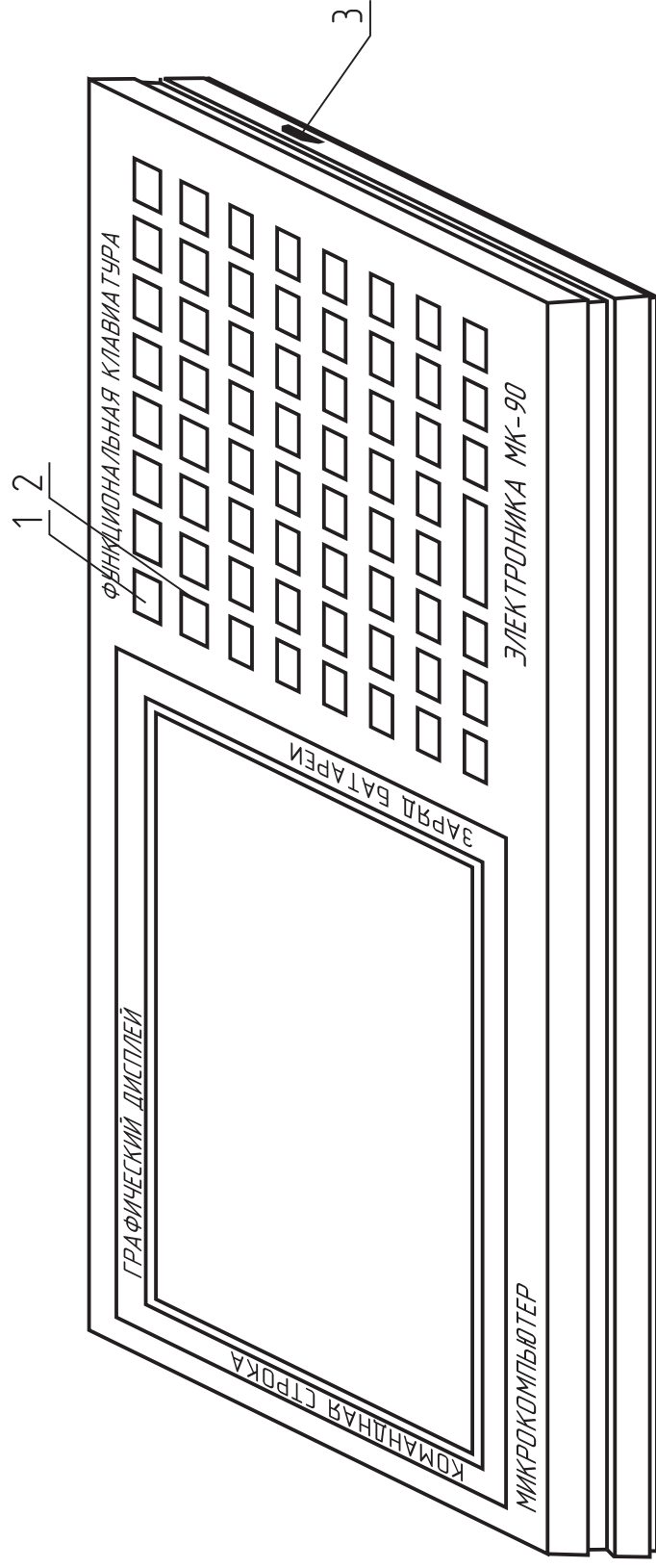
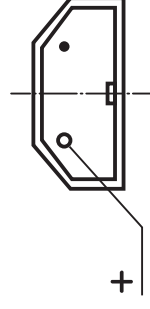


Рис. 5.2



- 1 — клавиша выключения,
- 2 — клавиша включения,
- 3 — место подключения внешнего блока питания.



5.3. Нажать последовательно клавиши выключения 1 и включения 2 (см. рис. 5.2). На экране ЖКИ должна появиться информация начального диалога (см. п. 6.1.1.), свидетельствующая о готовности микрокалькулятора к работе.

5.4. С помощью регулятора контрастности 2 (см. рис 5.1) установить оптимальный контраст экрана ЖКИ.

5.5. В микрокалькуляторе имеется место 3 подключения внешнего блока питания с техническими данными, указанными в п. 2.15.

5.6. Для подключения СМП к микрокалькулятору необходимо:

- 1) Снять крышку 3, закрывающую доступ к отсеку СМП, (слегка нажать на рифление крышки и сдвинуть ее в направлении стрелки, обозначенной на крышке).
- 2) Вставить СМП в гнездо, обозначенное цифрой 0 или 1;
- 3) Аккуратно подвинуть корпус СМП до полного входа вилки СМП в разъем микрокалькулятора;
- 4) Установить на место крышку отсека СМП.

5.7. Допускается подключать (отключать) СМП к (от) микрокалькулятору только в выключенном состоянии.

5.8. Микрокалькулятор выдает информацию о разряде элементов питания микрокалькулятора, признаком чего является отображение двух дополнительных точек в левом верхнем и правом нижнем углах ЖКИ.

5.9. При невыполнении или неправильном выполнении тестов или программ при питании микрокалькулятора от сети переменного тока через внешний источник питания рекомендуется перейти на автономный источник питания.

Если после выполнения п. 5.3 микрокалькулятор не включается, рекомендуется после нажатия клавиши "Вкл" вынуть один элемент питания (аккумулятор) или при их отсутствии вынуть из гнезда разъем источника питания и через 5–10 сек. установить обратно, после чего повторить операцию по п. 5.3.

СМП при этом следует вынуть из гнезда до включения микрокалькулятора и установить их после подачи питания на микрокалькулятор, проверки его включения и последующим нажатием клавиши "Выкл".

## 6. Порядок работы


### 6.1. Режимы работы клавиатуры и ЖКИ

#### 6.1.1. Режим загрузки

Используемые функциональные клавиши при загрузке программного обеспечения (ПО) микрокалькулятора.

 — передвинуть рамку вправо;

 — передвинуть рамку влево;

 — ввод (при нажатии данной клавиши происходит загрузка с выбранного устройства).

При включении питания микрокалькулятора на экране высвечивается следующая информация начального диалога:

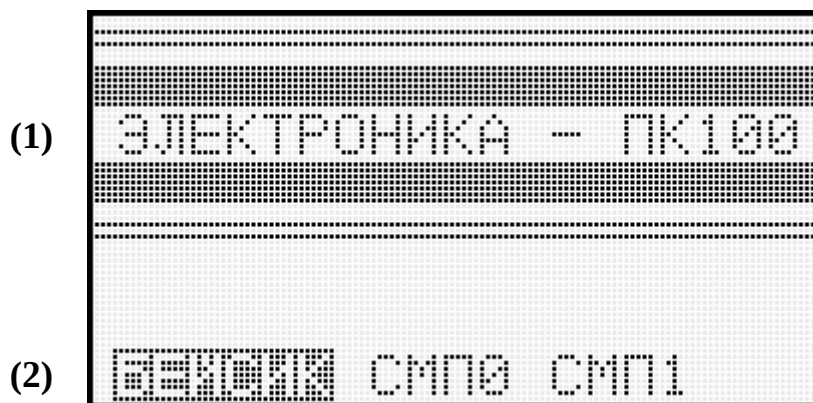



Рис. 6.1

Информация строки (2) экрана указывает на возможные устройства загрузки ПО (программного обеспечения).

С помощью установленной рамки, передвигая ее вправо или влево описанными выше клавишами, выбирается устройство, с которого будет загружено ПО.

Режим загрузки с СМП0 (СМП1) предусмотрен для дальнейшего расширения программного обеспечения микрокалькулятора.

Первоначально рамка устанавливается на текст БЕЙСИК

и нажатием клавиши  осуществляется запуск интерпретатора БЕЙСИК из внутреннего постоянного запоминающего устройства.

После запуска интерпретатора БЕЙСИК на экране появляется следующая информация:

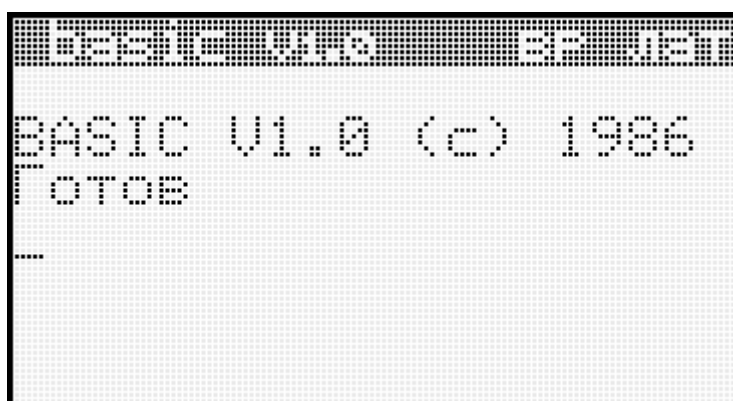





Рис. 6.2




### 6.1.2. Клавиши редактирования и специальные клавиши



Описанные ниже клавиши выполняют специальные функции и не зависят от нажатия других клавиш.


 — ввод пробела во всех режимах работы клавиатуры.  
(пробел)


 — удаляется последний введенный символ или знак и осуществляется перемещение курсора на позицию влево.  
(забой)


 — выполняет ввод программ и команд.  
(возврат  
каретки)


   — при последовательном вводе этих клавиш прекращается выполнение программы, кроме команд PLAY, LOAD, SAVE, NAME, FILES, KILL, INIT.

  — при последовательном вводе этих клавиш устанавливается/отключается режим звукового сопровождения при нажатии клавиш.

 — используется для перевода клавиатуры в функциональный режим.

При последовательном нажатии клавиши  и какой-либо другой вводится оператор (функция) БЕЙСИКа, который можно вводить и посимвольным способом.


Для установки клавиатуры в режим верхнего или нижнего регистра имеется клавиша . При включении микрокалькулятора всегда установлен режим верхнего регистра.


Для установки клавиатуры в режим латинского или русского регистра имеется специальная клавиша . При включении микрокалькулятора всегда устанавливался режим латинского регистра.


Режимы работы клавиатуры отображены в верхней служебной строке ЖКИ при работе с интерпретатором БЕЙСИКа.




### 6.1.3. Работа клавиатуры в режиме верхнего и нижнего регистров


При вводе с клавиатуры в режиме верхнего (нижнего) регистра осуществляется ввод символа, обозначенного на клавише (над клавишей) для следующей группы клавиш, которые не зависят от режима русского/латинского регистра.


\* — знак умножения;  
 — двоеточие;


+ — знак сложения;  
 — точка с запятой;

? — знак вопроса;  
 — наклонная черта;

! ... ) — специальные символы;  
 ...  ...  — цифры;

= — знак равенства;  
 — знак минус;

< — знак меньше;  
 — запятая;

> — знак больше;  
 — точка;

Группа клавиш верхнего (нижнего) регистра, вводимые символы которых зависят от установленного режима русского/латинского регистра:



Заглавные (строчные) буквы русского алфавита при установленном режиме русского регистра или заглавные (строчные) буквы латинского алфавита при установленном режиме латинского регистра.



" Ч " — заглавная (строчная) в режиме русского регистра  
" ˆ " — в режиме латинского регистра для верхнего регистра



" Ш " — заглавная (строчная) в режиме русского регистра  
" [ " — в режиме латинского регистра для верхнего регистра



" Щ " — заглавная (строчная) в режиме русского регистра  
" ] " — в режиме латинского регистра для верхнего регистра



" Ъ " — в режиме русского нижнего регистра  
" \_ " — в режиме латинского регистра для верхнего регистра



" Э " — заглавная (строчная) в режиме русского регистра  
" \ " — в режиме латинского регистра для верхнего регистра



" Ю " — заглавная (строчная) в режиме русского регистра  
" @ " — в режиме латинского регистра для верхнего регистра



" ˆ " — в режиме латинского регистра для нижнего регистра



" { " — в режиме латинского регистра для нижнего регистра



" } " — в режиме латинского регистра для нижнего регистра



" ■ " — символ "забой" - в режиме латинского нижнего и русского верхнего регистров




" | " — в режиме латинского регистра для нижнего регистра



" ‘ " — в режиме латинского регистра для нижнего регистра

#### 6.1.4. Работа клавиатуры в режимах русского/латинского регистров

Режим русского/латинского регистра устанавливается нажатием специальной клавиши  и действует только на группу клавиш символов, которые зависят от установки режима русско-латинского регистров.

При установленном режиме русского регистра осуществляется ввод символа, обозначенного на панели над клавишей. Этот символ при установленном режиме верхнего регистра является заглавной русской буквой, а при установленном режиме нижнего регистра - строчной русской буквой.

При установленном режиме латинского регистра осуществляется ввод символа, обозначенного непосредственно на клавише. Этот символ при установленном режиме верхнего регистра является заглавной латинской буквой, а при установленном режиме нижнего регистра – латинской строчной буквой.

Ввод русского текста должен начинаться с перевода клавиатуры в режим русского регистра, а заканчиваться переводом клавиатуры в режим латинского регистра. Данная последовательность ввода должна строго соблюдаться. При ее нарушении возможны ошибки. Ниже приведены примеры правильного и неправильного ввода русского текста.

Правильно	Неправильно
PRINT "Р/Л Русский текст Р/Л"	PRINT Р/Л "Русский текст" Р/Л PRINT "Р/Л Русский текст" Р/Л
REM Р/Л Комментарий Р/Л	REM Р/Л Комментарий

Повсюду в тексте угловые скобки (<>) означают обязательность присутствия информации, а квадратные ([]) – ее возможность или необязательность.

### 6.1.5. Работа клавиатуры в функциональном режиме

Для облегчения работы с клавиатурой введена функциональная клавиша **ФК**, то есть ввод директивы или функции может осуществляться последовательным нажатием только двух клавиш: **ФК** и какой-либо другой из таблицы соответствия директив и клавиш. Выводимые на экран директивы и функции имеют соответственно следующий вид:

<пробел> директива <пробел>  
или  
функция (

Если клавиша **ФК** была нажата случайно, то отмена режима осуществляется повторным нажатием этой клавиш.

**Таблица соответствия директив и клавиш.**

Директивы		Функции
AUTO — <b>A</b>	REM — <b>X</b>	ABS — <b>3</b>
CLS — <b>C</b>	RESTORE — <b>]</b>	ATN — <b>9</b>
DATA — <b>D</b>	RETURN — <b>[</b>	COS — <b>6</b>
DEF FN — <b>,</b>	RUN — <b>R</b>	EXP — <b>2</b>
DIS — <b>↑</b>	SAVE — <b>T</b>	INT — <b>5</b>
DIM — <b>.</b>	STEP — <b>S</b>	LOG — <b>7</b>
DEV — <b>←</b>	STOP — <b>U</b>	RND — <b>0</b>
DELETE — <b>→</b>	THEN — <b>Z</b>	SGN — <b>8</b>
DRAW — <b>B</b>	WAIT — <b>W</b>	SIN — <b>1</b>
EDIT — <b>V</b>	PLAY — <b>↵</b>	SQR — <b>4</b>
END — <b>E</b>	PRINT — <b>P</b>	
FILES — <b>\</b>	RANDOMIZE — <b>_</b>	
FOR — <b>F</b>	READ — <b>Y</b>	
GOSUB — <b>↓</b>		
GOTO — <b>G</b>		
INIT — <b>J</b>		
INPUT — <b>I</b>		
HELP — <b>H</b>		
KILL — <b>K</b>		
LET — <b>L</b>		
LIST — <b>M</b>		
LOAD — <b>@</b>		
LOCATE — <b>Q</b>		
NAME — <b>O</b>		
NEXT — <b>N</b>		

Схема клавиатуры для работы в функциональном режиме приведена в **Приложении 1**. Надписи под некоторыми клавишами (LLIST, LPRINT, LFILES) зарезервированы под расширение функций. При нажатии клавиши **БК** после ввода команд LLIST, LPRINT, LFILES микрокалькулятор переходит в режим "зависания", выход из которого осуществляется повторным включением микрокалькулятора.

#### 6.1.6. Режим работы ЖКИ

На экран ЖКИ информация может выводиться в режиме отображения символов и в графическом режиме.

##### 6.1.6.1. Режим отображения символов

На экране ЖКИ размещается 8 строк по 20 символов в каждой строке. После заполнения строки 20 символами, двадцать первый символ автоматически переходит на следующую строку, а после заполнения всех строк происходит сдвиг всего изображения на одну строку вверх, при этом верхняя строка стирается, а последняя строка очищается.

##### 6.1.6.2. Графический режим

На экране ЖКИ можно высвечивать точки с заданными координатами. Расположение точек задается оператором DRAW (см п. 6.3.2.15), который позволяет вычерчивать прямые линии и окружности, стирать прямые линии состоящие из непрерывно расположенных точек. Графические координаты состоят из 7680 точек: 120 точек в направлении X и 64 точки в направлении Y (см рис. 6.3).

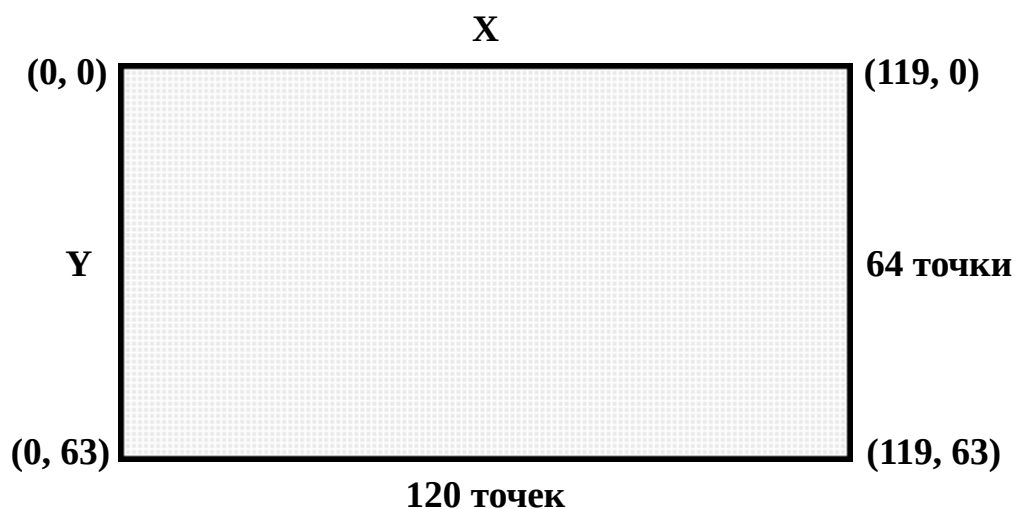


Рис. 6.3



## 6.2. Функции вычисления

### 6.2.1. Константы и точности вычисления

В языке БЕЙСИК определены целые и вещественные константы. Вещественные константы записываются в основной и экспоненциальной форме.

Например, число 2340,0 на языке БЕЙСИК может быть записано как целое число 2340 или вещественное 2340.0 (основная форма), 23.4E2 или 2.34E3 (экспоненциальная форма).

Для использования чисел, имеющих экспоненциальное представление, существуют следующие правила:

— Порядок может не иметь знака, если он положителен (1234.56E3).

Отрицательный порядок должен иметь знак (1234.56E-3);

— Порядок должен находиться в интервале от минус 980 до +980;

Результаты вычислений выводятся на экран ЖКИ в виде целых или десятичных чисел, если для представления результата достаточно девяти позиций, т.е. семь цифр, знак (пробел в случае положительного результата) и десятичная точка. В противном случае результат выводится на экран ЖКИ в экспоненциальном представлении и занимает максимум пятнадцать позиций, т.е. знак (если результат положительный – пробел), десятичная точка, семь значащих цифр мантииссы, символ E, знак порядка (минус или пробел) и порядок, имеющий до четырех цифр.

Ниже приведены некоторые вводимые пользователем значения с эквивалентами, печатаемыми БЕЙСИКом.

Вводимое значение	Значение, печатаемое БЕЙСИКом
.01	.01
9.90000E-3	.0099
999999	999999
1.000000E6	1000000
2.718281828459045	2.718282
0.00000000036218	.36218E-9

БЕЙСИК точен по крайней мере до семи цифр с точностью до единицы в седьмом разряде. Поэтому число 4.0000000225 является эквивалентом 4.0.

### 6.2.2. Арифметические функции БЕЙСИКа микрокалькулятора

⌈ возведение в степень  
+, – сложение, вычитание  
\*, / умножение, деление

Ограничение области вычислений:

- 1) Деление на 0 приводит к ошибке;
- 2) При переполнении, когда результат не укладывается в диапазон вычислений, возникает ошибка.

Допустимые степени:

$$X^Y$$

Y – любое действительное число, ошибка при  $X \leq 0$

Примеры:

1.	$0.5 \neg 0$	1
2.	$-0.5 \neg 0$	-1
3.	$(-0.5) \neg 0$	ОШИБКА
4.	$0.5 \neg 2$	0.25
5.	$0.5 \neg (-2)$	4
6.	$(-0.5) \neg (-2)$	ОШИБКА
7.	$0.5 \neg 0.5$	0.7071068
8.	$(-0.5) \neg 0.5$	ОШИБКА
9.	$2 \neg (-0.5)$	0.7071068
10.	$(2) \neg -0.5$	ОШИБКА

Приоритет последовательности вычислений:

- 1) Элементы в фигурных скобках.
- 2) Функции (встроенные функции (см. п. 6.3.3.) и функции пользователя (см. п. 6.3.2.1.))
- 3) Возведение в степень ( $\neg$ )
- 4) Умножение (\*) и деление (/)
- 5) Сложение (+), вычитание (-)
- 6) Операторы сравнения  $\leq$ ,  $\geq$ ,  $<$ ,  $>$ ,  $><$ ,  $=$  и т.д.

## Операторы сравнения

Операторы сравнения могут быть использованы только с применением оператора IF (см. п. 6.3.2.7.).

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

Пример:  $A+B<>0$ . Результат сложения A и B не равен нулю.

## 6.2.3. Переменные

Переменная есть величина, значение которой может изменяться в процессе выполнения программы. Переменная обозначается одной буквой или одной буквой (заглавной латинской) с последующей за ней цифрой.


Например:

I, B3, C8

Так же как и константы, переменные могут быть целыми и вещественными.

### 6.3. Команды БЕЙСИКа микрокалькулятора

#### 6.3.1. Команды непосредственного режима

Некоторые команды БЕЙСИКа могут вводиться в строке без номера. В этом случае выполнение команды осуществляется после нажатия клавиши  в конце строки. Результат выполнения хранится для последующего использования, но команда теряется. Этот режим называется непосредственным.

Непосредственный режим необходим при редактировании и отладке программ, записанных в программном режиме.





В приводимых форматах команд квадратные скобки [ ] означают возможность или необязательность присутствия информации.

Краткое описание команд БЕЙСИКа микрокалькулятора приведено в **Приложении 2**. Сообщения об ошибках при вводе команд приведены в **Приложении 3**.

##### 6.3.1.1. Команда **AUTO**

Функция: Переводит БЕЙСИК в режим автоматической нумерации строк во время ввода программы.

Формат: `AUTO [<номер строки>] [, [<шаг>]]`

Выполнение этой команды приводит к генерации номеров строк после каждого нажатия клавиши  в конце очередной программной строки. Нумерация начинается с <номера строки>, а последующие номера строк отличаются на величину, заданную <шагом>. Если <номер строки> или <шаг> не заданы, то значением по умолчанию принимается 10. Прекращение автонумерации осуществляется по нажатию клавиш   .

Пример:

```
AUTO 100,50
```

Генерация номеров строк с шагом 50 и начальным номером 100 (100, 150, 200, ...).

Минимальный номер строки и минимальный шаг равны 1. Максимальный номер строки равен 8191. Размер строки не должен превышать 80 символов (4 строки экрана ЖКИ).

### 6.3.1.2. Команда **DELETE**

Функция: Стирает строку с указанным номером

Формат: DELETE <номер строки N1>[, <номер строки N2>]

N1 – номер первой строки, которую необходимо удалить

N2 – номер последней строки

Команда DELETE должна иметь аргумент.

Если в команде DELETE задается только один номер строки, стирается соответствующая строка. Если даны два номера, разделенные запятой, стираются соответствующие две строки и все строки между ними.

Пример:

DELETE 54

Стирается строка 54

DELETE 56, 876

Стираются строки с 56 по 876 включительно.

Если в команде DELETE начальный номер строки N1 больше конечного N2 — БЕЙСИК отвечает "Готов", но ни одна строка не стирается.

DELETE 1, 8191

Стираются все строки программы.

### 6.3.1.3. Команда **EDIT**

Функция: Переводит БЕЙСИК в режим редактирования программы, расположенной в буфере команд

Формат: EDIT <номер строки>

Команда EDIT используется в непосредственном режиме для перевода БЕЙСИКа в режим редактирования строки. Редактирование русского текста не допускается. Для редактирования используются следующие клавиши.



— перемещение маркера на позицию влево;



— перемещение маркера на позицию вправо;



— перевод редактора в режим вставки-замены;



— смыкание строки влево к позиции маркера;



— установка маркера в начало редактируемой строки;



— переход к предыдущему слову строки;



— удаление символов строки, начиная с текущей позиции маркера до конца строки;



— переход к следующему слову строки;



— установка маркера за последним символом строки;




— запись отредактированной строки в буфер команд и переход в командный режим.

#### 6.3.1.4. Команда **HELP**

Функция: Выдает на экран ЖКИ информацию о составе команд языка БЕЙСИК, их назначении, формате записи и привязке к функциональной клавиатуре

Формат: `HELP *`  
`HELP <тема>`

Информация по `HELP` печатается построчно по нажатию клавиши .

Если нужна информация об определенной команде или функции, набирают `HELP <тема>`.

Информацию по всем командам и функциям можно получить по команде `HELP *`.

Примеры:

```
HELP *
HELP DRAW/A
HELP HELP
```


#### 6.3.1.5. Команда **LIST**

Функция: Печатает строки программы на экране

Формат: `LIST [<номер строки N1>[, <номер строки N2>]]`

Для получения текста программы на экране нужно задать команду `LIST`. По окончании печати БЕЙСИК выйдет на "Готов".

Для печати одной строки нужно в команде `LIST` указать номер этой строки. Для печати ряда строк в команде `LIST` следует указать номера начальной и конечной строк; в этом случае, кроме указанных строк, будут печататься все строки между ними.

Если между указанными номерами находится более 7 строк программы, то на экран выводится сразу 7 строк, а далее по одной строке после нажатия клавиши .

Примечание: при вызове командой `LIST` несуществующего номера строки (строк) на экран выводится ближайшая к ней (ним) существующая строка (строки).

Пример:

```
LIST 37
```

Печатается строка 37

```
LIST 37,126
```

Печатаются строки с 37 по 126 включительно

```
LIST
```

Печатается вся программа начиная со строки с меньшим номером.

#### 6.3.1.6. Команда **MEM**

Функция: Выдает распределение памяти в текущий момент

Формат: MEM

Пример:

MEM

Выдает распределение памяти в следующем виде:



Эта команда дает информацию об объеме набранного текста программы, при этом необходимо учитывать, что объем памяти, занимаемый переменной или элементом массива (см. п. 6.3.2.2.) занимает 6 байт.

#### 6.3.1.7. Команда **INIT**

Функция: Стирание всех файлов с СМП и его разметка

Формат: INIT ["<имя устройства>:"]

Если <имя устройства> не указано, то стирается и размечается рабочий СМП (см. п. 6.3.1.12.). Вся информация с СМП пропадает.

Примеры:

```
INIT "SM0 : "  
INIT
```

Примечание: командой INIT, как и всеми последующими командами, предназначенными для работы с СМП, необходимо пользоваться только при подключенном СМП.

#### 6.3.1.8. Команда **SAVE**

Функция: Сохраняет файлы на СМП

Формат: `SAVE "<спецификация файла>"`

Эта команда сохраняет БЕЙСИК-программу на СМП. Для этого указывается имя устройства, имя и тип файла. Если имя устройство опущено, то им считается рабочий СМП (см. п. 6.3.1.12.). А если отсутствует тип файла, то записывает тип "BAS".

Пример 1:

```
SAVE "ADDR"
```

(записывается файл ADDR.BAS на рабочий СМП)

Пример 2:

```
SAVE "SM1:PROG.ABC"
```

(записывается файл PROG.ABC на СМП1)

Имя файла должно содержать не более 6 символов, тип файла – не более 3 символов.

#### 6.3.1.9. Команда **NAME/AS**

Функция: Изменяет имена файлов на СМП

Формат: `NAME "<спецификация старого файла>" AS "<спецификация нового файла>"`

Как <спецификация старого файла>, так и <спецификация нового файла> задаются именем устройства, именем и типом файла. Имя устройства допускается опускать, если это рабочий СМП.

Имя файла, указанное в <спецификации старого файла>, должно быть именем существующего файла. Если имя нового файла уже существует, то файл с таким именем будет удален с СМП, а его имя присвоится файлу со спецификацией старого файла. Эта команда изменяет только имя указанного файла, но не переписывает его в другую область СМП.

Пример:

```
FILES "SM1: "  
PROG1 .BAS      2      4  
PROG3 .BAS      5      6  
Готов
```

```
NAME "SM1:PROG1.BAS" AS "SM1:PROG2.BAS"  
Готов
```

```
FILES "SM1: "  
PROG2 .BAS      2      4  
PROG3 .BAS      5      6  
Готов
```

Примечание: набранную или исправленную программу необходимо сохранить на СМП прежде, чем запустить выполнение командой RUN. Это обеспечивает более компактное и надежное хранение программы.



#### 6.3.1.10. Команда **LOAD**

Функция: Загружает в память программу, записанную в СМП

Формат: `LOAD "<спецификация файла>" [, R]`

В спецификации файла необходимо указать имя устройства, имя и тип файла. Если имя устройства опущено, подразумевается рабочий СМП; если не указан тип файла, то подразумевается ".BAS".

Если команда **LOAD** выполняется без задания "R", то после загрузки программы БЕЙСИК переходит в командный режим. Однако, если "R" задано, то как только загрузка программы завершится, сразу же начинается ее выполнение.

Необходимо помнить, что в результате работы команды **LOAD** все переменные очищаются, а любая программа, находившаяся до этого в буфере команд, уничтожается.

Пример:

```
LOAD "SM1 : PROG1 . BAS "
```

#### 6.3.1.11. Команда **KILL**

Функция: Удаляет файлы с СМП

Формат: `KILL "<спецификация файла>"`

Команда **KILL** может использоваться для удаления файлов любых типов с СМП. Для удаления файла обязательно указывается полная спецификация файла, если удаляемый файл расположен на устройстве, отличном от рабочего. В противном случае достаточно указать имя и тип файла.

Пример:

```
KILL "SM1 : GRAPH . BAS "
```

Удаляется файл GRAPH.BAS с устройства SM1.

#### 6.3.1.12. Команда **DEV**

Функция: Задает <имя устройства> в качестве рабочего СМП

Формат: `DEV [ "<имя устройства>: " ]`

Если <имя устройства> не задано, то в качестве рабочего выбирается СМП0. Использование команды **DEV** позволяет не указывать <имя устройства> в <спецификации файла> во всех командах, работающих с СМП.

Пример 1:

```
DEV "SM1 : "
```

Устанавливает в качестве рабочего СМП1.

Пример 2:

```
DEV
```

Устанавливает в качестве рабочего СМП0.

### 6.3.1.13. Команда **FILES**


Функция: Распечатывает имена файлов на <имя устройства>

Формат: FILES ["<имя устройства>:"]

Если <имя устройства> не указано, по этой команде печатается список всех файлов, расположенных на СМП, выбранном в качестве рабочего.

Если <имя устройства> присутствует, то в этом случае команда FILES распечатывает список всех файлов на указанном СМП.

Кроме имен файлов, во втором и третьем столбцах списка печатаются соответственно: длина файла в блоках по 512 байт и начальный адрес этого файла на СМП.

Список файлов разбивается и выводится на экран по страницам после нажатия клавиши .

Пример 1:

```
FILES "SM1 : "
```

Распечатывает список всех файлов, расположенных на СМП1.

Пример 2:

```
FILES
```

Распечатывает список всех файлов, расположенных на рабочем СМП.

При наборе командной строки могут быть ошибки – отсутствие кавычек, двоеточия или неправильно задано символическое имя устройства, а также набрано больше шести символов в имени файла.

Во всех этих случаях выдается одно сообщение об ошибке в командной строке.

Примечания:

1. Перед работой с СМП (команды: INIT, LOAD, SAVE, KILL, FILES) убедиться, что требуемые СМП вставлены в нужные разъемы микрокалькулятора.

2. Смену СМП производить только при выключенном микрокалькуляторе, в противном случае может произойти "зависание" микрокалькулятора либо неверное выполнение команд работы с СМП.

#### 6.3.1.14. Команда **RUN**

Функция: Осуществляет запуск программы

Формат: RUN

По команде RUN запускается программа в программном режиме.

При использовании команды RUN для повторного прохождения программы все значения переменных стираются из памяти. Если в программу включена функция RND(X), то ее начальное значение восстанавливается.

Прекратить выполнение программы можно последовательным нажатием клавиш



#### 6.3.2. Команды программного режима

Чаще всего программы на языке БЕЙСИК записываются в программном режиме. Где каждая строка начинается с номера, указывающего на последовательность выполнения. В программном режиме команды называются операторами. В этом режиме выполнение операторов задерживается до подачи специальной команды (например, RUN или GOTO).

Примеры программ приведены в **Приложении 4**.

Программа, загруженная из СМП или набранная пользователем на клавиатуре, удаляется при выключении калькулятора, либо в случае загрузки из СМП другой программы. При вводе строки с уже имеющимся номером последняя записывается на место строки с этим номером. В одной строке можно записать несколько операторов, которые разделяются двоеточием и выполняются последовательно.

##### 6.3.2.1. Оператор **DEF FN**

Функция: Определение функций пользователя

Формат: DEF FN<буква> (<переменная>) =<выражение>

В некоторых программах возникает необходимость вычисления одного и того же математического выражения, часто с различными данными. В БЕЙСИКе имеется возможность определять функции или выражения и вызывать их так же, как и математические функции, т.е. синус, косинус, квадратный корень и т.д.

Определяемая пользователем функция идентифицируется именем, состоящим из трех букв, и аргументом в скобках. Первые две буквы имени функции – FN; третья буква может быть любой буквой латинского алфавита. Таким образом можно определить до 26 функций.

Такая функция должна быть определена прежде, чем она будет использована, определяется она с помощью оператора определения функции DEF:

DEF FNA (параметр) = выражение,

Где A – буква латинского алфавита.

Например, посредством оператора

```
10 DEF FNX (S) = S-2+4
```

оператор

```
20 LET R=FNX (4)
```

будет вычисляться как

```
LET R=4-2+4,
```

а оператор

```
20 LET R=FNX (2)
```

будет вычисляться как

```
LET R=2-2+4.
```

Параметры, указанные в определяемой пользователем функции, являются формальными параметрами. Параметры указанные при вызове данной функции, являются фактическими параметрами.

В нашем примере S – формальный параметр, 4 и 2 – фактические параметры.

Аргумент в скобках может быть любым допустимым выражением; значение выражения подставляется вместо аргумента функции.

В следующем примере функция FNX в строке 10 будет возводить в квадрат число, подставляемое вместо аргумента функции. В строке 30 выражение 2+A дает 4; затем 4 возводится в квадрат и печатается значение функции FNZ(X).

Пример:

```
10 DEF FNZ (X) = X^2
```

```
20 LET A=2
```

```
30 PRINT FNZ (2+A)
```

```
ГОТОВ
```

```
RUN
```

```
16
```

```
ОСТ В СТРОКЕ 30
```

```
ГОТОВ
```

Другой пример:

```
10 DEF FNA (Z) =Z+A+B
20 DATA 2,4
30 READ A,B
40 LET F=FNA (9)+1
50 PRINT F
Готов

RUN
16
ОСТ В СТРОКЕ 50
Готов
```

Функция может использоваться рекурсивно, как показано в следующем примере. Выражение в строке 30 вычисляется как  $(2+(2*4))$  прежде чем оно будет возведено в квадрат.

Пример:

```
10 DEF FNA (X) =X^2
20 LET A=2
30 PRINT FNA (2+A*FNA (2) )
Готов
RUN

100
ОСТ В СТРОКЕ 30
Готов
```

Если одна и та же функция определяется несколько раз, используется первое определение, а последующие определения игнорируются.

Пример:

```
10 DEF FNX (X) =X^2
20 DEF FNX (X) =X+X
30 PRINT FNX (6)
40 END
Готов

RUN
36
ОСТ В СТРОКЕ 40
Готов
```

Формальный параметр функции может не использоваться в выражении функции. В следующем примере подстановка фактического параметра вместо формального не оказывает никакого влияния на значение функции.

Пример:

```
10 DEF FNA (X) =4+2
20 LET R=FNA (10)+1
30 PRINT R
ГОТОВ

RUN
7
ОСТ В СТРОКЕ 30
ГОТОВ
```

#### 6.3.2.2. Оператор **DIM**

Функция: Объявляет массив

Формат: DIM <идентификатор массива1> (<индекс1>[, <индекс2>])  
[, <идентификатор массива2> (<индекс1>[, <индекс2>])...]

DIM резервирует место в памяти для массива данных. Максимальное значение списка индексов – 255. Максимальное количество всех массивов программы не более 1960.

Идентификатором массива может быть любая переменная.

Максимальное значение размерности – 2, то есть допускается работа с одномерными и двумерными массивами. В операторе должны содержаться максимально возможные индексы:

```
DIM A1 (4, 4) , B (10)
```

Так как индексы в БЕЙСИКе начинаются с 0, приведенный выше оператор будет резервировать место для массива A1, способного вместить 25 элементов (5\*5) и массива B — 11 элементов. Задавать размерность ранее использованной переменной недопустимо.

Например:

```
10 LET A3=C-C+B/3
.
.
.
40 DIM A3 (2, 5)
```

Вызовет сообщение об ошибке.

```
ОШ      6  СТР  40
```

Применение.

Введем следующую программу:

```
10 DIM A(19)
20 FOR I=0 TO 19
30 LET A(I)=0
40 NEXT I
60 FOR I=0 TO 4
70 INPUT A(I)
85 NEXT I
```

Выполним ее:

RUN 

?5  
?6  
?7  
?8  
?9

Ввод A(I)

ОСТ В СТРОКЕ 85  
ГОТОВ

Останов программы и переход в режим команд

Посмотрим результат выполнения программы:

```
PRINT A
5
PRINT A(1)
6
PRINT A(2)
7
PRINT A(3)
8
PRINT A(4)
9
PRINT A(5)
0
PRINT A(6)
0
PRINT A(29)
```

ОШ 6 СТР 40  
ГОТОВ

После ввода пяти констант выполнение программы останавливается. затем используется оператор PRINT в непосредственном режиме для проверки начальных элементов массива. Попытка напечатать содержимое ячейки массива, находящейся за пределами, установленными оператором DIM, вызывает сообщение об ошибке.

Ниже приводится продолжение рассмотренной программы, демонстрирующей автоматический вывод.

```
10 DIM A(19)
20 FOR I=0 TO 19
30 LET A(I)=0
40 NEXT I
60 FOR I=0 TO 4
70 INPUT A(I)
90 NEXT I
95 REM - СТРОКИ 100-130 ПЕЧАТАЮТ СПИСОК ПОСЛЕДОВАТЕЛЬНО
100 FOR I=0 TO 6
110 PRINT "A("I")", A(I)
120 NEXT I
130 STOP
135 REM - СТРОКИ 140-160 ПЕЧАТАЮТ ЧАСТЬ СПИСКА
136 REM - В ОБРАТНОМ ПОРЯДКЕ
140 FOR I=6 TO 0 STEP -1
145 PRINT "A("I")", A(I)
160 NEXT I
Готов
```

```
RUN
? 1
? 2
? 3
? 4
? 5
A( 0 )    1
A( 1 )    2
A( 2 )    3
A( 3 )    4
A( 4 )    5
A( 5 )    0
A( 6 )    0
```

```
ОСТ В СТРОКЕ    130
Готов
```

```
ГOTO 140
A( 6 )    0
A( 5 )    0
A( 4 )    5
A( 3 )    4
A( 2 )    3
A( 1 )    2
A( 0 )    1
```

```
ОСТ В СТРОКЕ    160
Готов
```



#### 6.3.2.3. Оператор **END**

Функция: Последний оператор программы

Формат: END

Оператор END следует включать в программы, предназначенные для неоднократного использования. Он должен помещаться в конце программы, т.е. в строке, являющейся логическим концом программы.

#### 6.3.2.4. Оператор **FOR-TO-STEP/NEXT**

Функция: FOR-TO-STEP используется для обозначения начальной точки цикла программы

Формат: FOR <переменная>=E1 TO E2 [STEP E3],  
где E1, E2, E3 – выражения

Функция: NEXT используется для обозначения конечной точки цикла программы

Формат: NEXT <переменная>

Все операторы, стоящие между FOR и соответствующим ему оператором NEXT, циклически выполняются в соответствии с условиями, заданными в операторе FOR. Переменная, следующая за оператором FOR, рассматривается как управляющая переменная. Та же переменная должна присутствовать в операторе NEXT, определяющем конец цикла.

Минимальное и максимальное значения, представленные выражениями слева и справа от слова TO в операторе FOR, являются областью действия управляющей переменной. Выражение, стоящее после слова STEP, указывает на какую величину изменится управляющая переменная после каждого прохода цикла. Если E3 не задано, шаг цикла равен 1.

Пример:

```
10 LET A=5
20 FOR B=1 TO 5
30 PRINT A+B
40 NEXT B
Готов
```

RUN

```
6
7
8
9
10
```

```
ОСТ В СТРОКЕ 40
Готов
```

В строке 20 переменной "В" присваиваются значения 1, 2, 3, 4, 5. Так как она не может иметь все эти значения одновременно, создается цикл, начинавшийся с оператора FOR в строке 20 и заканчивавшийся оператором NEXT в строке 40. Операторы внутри цикла выполняются пять раз, причем каждый раз переменная "В" получает новое значение с шагом единица и печатается значение A+B. оператор NEXT обуславливает переход на строку 20 до тех пор, пока "В" не достигнет своего конечного значения 5.

Каждый оператор FOR внутри программы должен сопровождаться оператором NEXT; NEXT не может использоваться без предшествующего оператора FOR.

#### 6.3.2.5. Операторы **GOSUB, RETURN**

Функция: Позволяет организовать переход к подпрограмме

Формат: GOSUB N

где N – номер строки, которой передается управление

Функция: Последний оператор подпрограммы. Передает управление оператору, следующему за соответствующим оператором GOSUB

Формат: RETURN

Часто в программе необходимо записать группу операторов (подпрограмму) только один раз, а затем неоднократно обращаться к подпрограмме из различных точек программы. Оператор GOSUB используется для перехода к первому оператору программы. Заканчивается подпрограмма оператором RETURN.

Оператор RETURN должен использоваться с GOSUB. однако, одного оператора RETURN достаточно для случая, когда несколько GOSUB используется для перехода к одной программе.

Например\*:

```
5 PRINT " X"; " X^2"; " X^3"; " X^4"
10 FOR A=1 TO 3
15 LET X=A
20 GOSUB 50
25 NEXT A
28 FOR A=1 TO 3
30 LET X=A^2
40 GOSUB 50
45 NEXT A
48 STOP
50 FOR J=1 TO 4
51 PRINT X^J;
52 NEXT J
53 PRINT
54 RETURN
55 END
Готов
```

RUN

X	X^2	X^3	X^4
1	1	1	1
2	4	8	16
3	9	27	81
1	1	1	1
4	16	64	256
9	81	729	6561

ОСТ В СТРОКЕ 48  
Готов

Вход в подпрограмму (строки 50-54) осуществляется операторами GOSUB в строках 20 и 40, а выход из нее – оператором RETURN в строке 54.

#### 6.3.2.6. Оператор **GOTO**

Функция: Передает управление строке N и продолжает выполнение программы с этой строки

Формат: GOTO N

Оператор GOTO осуществляет безусловный переход к строке с указанным номером. Выполнение программы продолжается последовательно, начиная с оператора, к которому был произведен переход.

---

**\*Прим. редактора:** авторами оригинальной инструкции был выбран очень неудачный пример. БЕЙСИК в МК90 не позволяет адекватно построить таблицу, вызывая видимые в примере расхождения столбцов. К тому же значение  $X^5$  в строку не умещается. Исходная программа содержит много ошибок, в связи с чем ее пришлось ощутимо скорректировать.

Пример:

```
10 DATA 1, 2, 3, 4, 5      (см. п. 6.3.2.11. – описание операторов DATA и READ)
```

```
15 READ X
```

```
20 PRINT X+1
```

```
40 GOTO 15
```

```
Готов
```

```
RUN
```

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

```
ОШ      20      СТР      15
```

```
Готов
```

Операторы 15, 20 и 40 представляют цикл в приведенном примере. Эти три оператора непрерывно выполняются с новым значением X при каждом выполнении строки, пока не будет использовано последнее значение константы оператора DATA.

Выполнение программы заканчивается сообщением об ошибке, показывающим, что использованы все значения констант.

В следующем примере оператор GOTO осуществляет переход к оператору INPUT так, что программа закичивается.

```
10 INPUT A                  (см. п. 6.3.2.8. – описание оператора INPUT)
```

```
20 PRINT A
```

```
30 LET A=A^A
```

```
40 PRINT A
```

```
50 PRINT
```

```
60 GOTO 10
```

```
Готов
```

```
RUN
```

```
?2
```

```
2
```

```
4
```

```
?3
```

```
3
```

```
27
```

```
?4
```

```
4
```

```
256
```

```
?5
```

```
5
```

```
3125
```

```
?^P
```



```
Готов
```

При записи в строке нескольких операторов, разделяемых символом ":", оператор GOTO должен быть последним оператором в строке.

### 6.3.2.7. Оператор **IF/THEN**

Функция: Обеспечивает условный переход к программе

Формат: IF <выражение> THEN <оператор>  
          THEN <номер строки>  
          GOTO <номер строки>

За словом THEN может следовать любой оператор, включая второй оператор IF. Когда за THEN следует номер строки, оператор IF-THEN действует такие, как и IF-GOTO.

Примечание: буква "T" не должна использоваться как наименование переменной в операторе IF.

Когда БЕЙСИК встречает оператор IF, он вычисляет выражения и сравнивает их в соответствии с требованиями логической операции согласно Табл. 6.1.

**Таблица 6.1.**

Логическая операция языка БЕЙСИК	Пример	Значение
=	A = B	A равно B
<	A < B	A меньше B
<=	A <= B	A меньше или равно B
>	A > B	A больше B
>=	A >= B	A больше или равно B
<>	A <> B	A не равно B

Если условие истинно, выполняется остальная часть оператора, начинающаяся с THEN или GOTO. Если условие ложно, выполняется следующий оператор после оператора IF.

Пример:

Используется IF GOTO <номер строки>

```
10 FOR A=1 TO 10
20 LET X=INT(A^2)
25 IF X>20 GOTO 40
30 PRINT X
35 NEXT A
40 PRINT "Значение A=" A
55 STOP
Готов
```

```
RUN
1
4
9
16
Значение A= 5
```

```
ОСТ В СТРОКЕ 55
Готов
```

Если заменить строку 25 на  $X > 9$  GOTO 40, то цикл заканчивается, когда будет  $A=4$ .

```
RUN
1
4
9
16
Значение A= 4
```

```
ОСТ В СТРОКЕ 55
Готов
```

В следующем примере в строках 40, 50 используется IF-THEN <оператор>:

```
10 REM - ПРОГРАММА ОЦЕНКИ УСПЕВАЕМОСТИ
25 INPUT A,B,C,D
30 LET X=(A+B+C+D)/4
40 IF X>=70 THEN PRINT X; "Удв. оценка"
50 IF X<70 THEN PRINT "Неусп. на курсе"
55 GOTO 25
Готов
```

```
RUN
?55,67,78,89
72.25 Удв. оценка
```

```
?55,44,32,21
Неусп. на курсе
```

```
?~P   
Готов
```


Оператор IF может записываться в любом месте в строке с несколькими операторами за исключением случая, когда за THEN следует оператор GOSUB или GOTO. В этом случае оператор IF должен быть последним оператором в строке. Оператор IF-THEN-GOSUB обеспечивает переход к подпрограмме и должен быть последним оператором в строке.

#### 6.3.2.8. Оператор **INPUT**

Функция: Осуществляет ввод данных с клавиатуры во время выполнения программы.

Формат: INPUT [V1, V2, . . . , VN]

БЕЙСИК во время выполнения оператора INPUT печатает знак вопроса и ждет, когда пользователь введет от единицы до N числовых значений.

При выполнении оператора INPUT числовые значения должны разделяться запятыми и заканчиваться . Для каждого оператора INPUT печатается только один знак вопроса.




Если введено недостаточное количество данных, БЕЙСИК печатает:

ОШ 121 СТР NNN

Если введено слишком много данных, БЕЙСИК печатает:

ОШ 122 СТР NNN




Где NNN – номер строки оператора INPUT.

В обоих случаях БЕЙСИК печатает второй знак вопроса и ждет выполнения повторного ввода. Строка ввода должна быть перепечатана. Выход в командный режим осуществляется по нажатию    в ответ на последний параметр.

Оператор INPUT может записываться в любом месте программы, а также может включаться в строку с несколькими операторами.

Пример:

```
10 INPUT A,B,C
20 GOTO 10
Готов
```

```
RUN
?1,2,3
?4,5,6
?7,8,9
?¬P   
Готов
```

#### 6.3.2.9. Оператор **LET**

Функция: Присваивает значение выражения переменной

Формат: LET <переменная>=<выражение>

Следующий пример иллюстрирует использование оператора LET:

```
10 LET A=1
20 LET B=2
30 LET C=A+1
40 LET X=A+B+C+1
```

После выполнения этих операторов значение X будет равно 6. Оператор LET может записываться в любом месте программы и в строке с несколькими операторами.

#### 6.3.2.10. Оператор **PRINT**

Функция: Осуществляется вывод данных на ЖКИ

Формат: PRINT [функция] [<список>]

Список может содержать выражение и (или) строку текста. Элементы списка разделяются с помощью ", или ";", которые задают формат печати.

Оператор PRINT без списка используется с целью вывода пустой строки.

PRINT может использоваться для вычислений: производится вычисление выражения, содержащегося в списке, и вывод на печать его значения. Полученное значение не сохраняется для последующего использования.

```
10 LET A=1
15 LET B=5
20 PRINT A+B
Готов
```

```
RUN
6
```

```
ОСТ В СТРОКЕ    20
Готов
```

Значения выражений, разделенных в операторе PRINT символом ",", печатаются через четыре позиции справа от предыдущего. Если при печати требуется более компактное расположение значения, используется символ ";". В этом случае каждое значение печатается через две позиции справа от предыдущего, при условии, что эта строка еще не закончена. Ниже приведен пример задания форматов печати.

```
10 DATA 1,2,3
20 READ A,B,C
30 PRINT A;B;C;
40 PRINT A;B;C
50 PRINT A,B,C
60 PRINT A,;B,C
Готов
```

```
RUN
1  2  3  1  2  3
1   2   3
1   2   3
```

```
ОСТ В СТРОКЕ    60
Готов
```

Для печати сообщений, комментариев или любой строки знаков оператор PRINT имеет функции, приведенные в Табл. 6.2.



Таблица 6.2.

<b>S</b>	Изменение размеров символов и знаков
<b>Q</b>	Задание ориентации символов и знаков
<b>Y</b>	Задание направления печати
<b>Z</b>	Задание расстояния между символами и между строками
<b>N</b>	Вывод информации в негативе
<b>P</b>	Вывод информации в позитиве

Список функций располагается за оператором PRINT и заключается в угловые скобки <>, между собой функции разделяются символом ";". Текст, который должен быть напечатан заключается в кавычки.

Функции оператора PRINT имеют следующие форматы:

1) Функция: S

Формат: S<размер символа по X, размер символа по Y>

Параметр размера символа означает во сколько раз размеры данного символа превосходят стандартные размеры (5\*7). По умолчанию параметр равен "1".

Например, символ "М" можно печатать с использованием размеров от 1 до 3 следующим образом:

```
10 CLS
20 DRAW O60,32
30 FOR I=1 TO 3
40 PRINT <SI,I>"М";
50 NEXT I
```

2) Функция: Q

Формат: Q<тип ориентации>

Параметр типа ориентации принимает значения 0, 1, 2, 3. Приведем пример печати буквы "F" в разных ориентациях:

```
10 FOR I=0 TO 3
20 PRINT <QI>"F";:PRINT I
30 NEXT I
```

3) Функция: Y

Формат: Y<направление печати>

Параметр направления печати принимает значения 0, 1, 2, 3.

0 – печать символов слева направо

1 – печать символов сверху вниз

2 – печать символов справа налево

3 – печать символов снизу вверх

Функцию Y рекомендуется использовать с LOCATE, т.к. символы, выходящие за пределы экрана, не отображаются.

Пример:

```
PRINT <Y1>"ABW"
```

4) Функция: Z

Формат: Z<расстояние между символами, расстояние между строками>

Вывод информации с заданным расстоянием между символами и строками. Действует до конца оператора PRINT.

5) Функция: N

Формат: N

Вывод информации в негативе. Действует до функции <P> или до конца оператора PRINT.

6) Функция: P

Формат: P

Вывод информации в позитиве. Действует до функции <N> или до конца оператора PRINT.

#### 6.3.2.11. Операторы **READ, DATA, RESTORE**

Функция: Переменным от V1 до VN включительно присваиваются значения соответствующих констант оператора DATA

Формат: READ V1, V2, . . . , VN

Функция: Определение значений констант N1,...,NN для оператора READ

Формат: DATA N1, . . . , NN

Функция: Устанавливает указатель в начало строки констант DATA

Формат: RESTORE

Для присвоения значений одиночным переменным в предыдущих примерах использовался оператор LET; для большего числа переменных использовалось и большее количество операторов LET.

Однако в программах со многими переменными для их определения следует использовать READ и DATA.

Оператор DATA вводит в программу числовую константу или группу констант. Оператор READ последовательно связывает имена переменных со значением констант, которые задаются операторами DATA.

Операторы READ и DATA должны сопровождать друг друга в программах пользователей, но они не обязательно должны быть парными. Если в одном или более операторах READ содержится 9 переменных, то в одном или более операторах DATA должно содержаться по крайней мере девять констант (исключение составляет использование оператора RESTORE).

В следующем примере все константы задаются одним оператором DATA. Они используются операторами READ в разных строках.

```
10 DATA 1, 5, 3, 7, 9
20 READ A, B
30 LET X=A+B
40 PRINT A, B, X
50 READ U, Q, R
60 LET X=X+U+Q+R
70 PRINT X, U, Q, R
ГОТОВ

RUN
  1      5      6
 25      3      7      9

ОСТ В СТРОКЕ   70
ГОТОВ
```

При выполнении программы БЕЙСИК игнорирует оператор DATA до тех пор, пока не встретится оператор READ. Затем он осуществляет поиск оператора DATA и находит его в строке 10 т.е. в первой строке программы.

БЕЙСИК последовательно берет значения констант и связывает их с переменными в операторе READ, которые также выбираются последовательно: переменной A присваивается значение 1, B – значение 5. Установив указатель на следующем элементе данных 3, он возвращается на строку 30, т.е. к следующему невыполненному оператору.

В строке 50 встречается другой оператор READ, который содержит три новых переменных. Теперь БЕЙСИК обращается к указателю для получения следующей неиспользованной константы: переменным U, Q, R присваивается значение констант 3, 7, 9.

С операторами DATA и READ связано несколько сообщений об ошибках:

- 19 – неправильная переменная в списке операторов READ.
- 20 – данные в списке оператора READ исчерпаны.
- 21 – неправильный формат оператора DATA.
- 123 – несуществующая переменная.

Относительно кода 123 следует помнить, что каждая переменная программы должна определяться или оператором READ, или оператором LET или FOR прежде, чем она будет использована в выражении или в списке оператора PRINT.

Оператор RESTORE дает возможность осуществить повторное использование констант операторов DATA, начиная с операторов DATA с самым низким номером строки в программе.

Пример:

```
40 DATA 1,2
50 READ A,B
60 PRINT A,B
70 RESTORE
80 READ C,D
90 PRINT C+D,C,D
ГОТОВ
```

RUN

```
1      2
3      1      2
```

Если бы не было оператора RESTORE, в строке 80 появилось бы сообщение об ошибке, указывающее на отсутствие констант для оператора READ.

Пример:

```
10 DATA 1,2
20 READ A,B
30 PRINT A,B
40 RESTORE
50 DATA 3,4
60 READ C,D
70 PRINT C+D,C,D
ГОТОВ
```

RUN

```
1      2
3      1      2
```

В данном примере под действием оператора RESTORE второй оператор READ в строке 60 получает значение констант первого оператора DATA в строке 10, а не второго, находящегося в строке 50.

Пример:

```
10 DATA 1,2
20 READ A,B
30 PRINT A,B
40 DATA 3,4
50 READ C,D
60 PRINT C+D,C,D
ГОТОВ
```

RUN

```
1      2
7      3      4
```

Оператор DATA нельзя включать в строку с несколькими операторами; он должен быть единственным на нумерованной строке.

Оператор READ может быть записан в любом месте в строке с несколькими операторами.

Оператор RESTORE может использоваться в любом месте программы; он может включаться в строку с несколькими операторами.

Оператор RESTORE не имеет действия в программах без операторов DATA и READ.

#### 6.3.2.12. Оператор **REM (REMARK)**

Функция: Позволяет напечатать комментарии к программе

Формат: REM – КОММЕНТАРИЙ

REMARK – КОММЕНТАРИЙ

Комментарии могут состоять из любых знаков.

Операторы REM не оказывают действия на выполнение программ, однако они занимают часть памяти.

При использовании в строке с несколькими операторами, REM должен записываться последним, т.к. БЕЙСИК игнорирует все, что идет в строке после REM.

Оператор REM может быть заменен открывающей кавычкой < ' > (клавиша В/Н и 7).

### 6.3.2.13. Оператор **RANDOMIZE**

Функция: Позволяет получать ряд случайных чисел в программе, используя функцию RND

Формат: RANDOMIZE

При выполнении оператора RANDOMIZE функция RND выбирает начальное значение для получения случайного числа.

Пример:

```
10 REM - СЛУЧАЙНЫЕ ЧИСЛА С ИСПОЛЬЗОВАНИЕМ RANDOMIZE
15 RANDOMIZE
25 PRINT "Случайные числа:"
30 FOR I=1 TO 4
40 PRINT RND (0);
50 NEXT I
60 END
Готов
```

```
RUN
Случайные числа:
.5157776 .5863953
.8763733 .980682
```

```
ОСТ В СТРОКЕ 60
Готов
```

```
RUN
Случайные числа:
.837677 .9383435
.2109681 .6407166
```

```
ОСТ В СТРОКЕ 60
Готов
```

```
RUN
Случайные числа:
.2499695 .7420959
.2028503 .5382385
```

```
ОСТ В СТРОКЕ 60
Готов
```

При удалении оператора RANDOMIZE получаем следующее:

```
RUN
Случайные числа:
.1002502 .9648132
.8866272 .6364441
```

```
ОСТ В СТРОКЕ    60
Готов
```

```
RUN
Случайные числа:
.1002502 .9648132
.8866272 .6364441
```

```
ОСТ В СТРОКЕ    60
Готов
```

```
RUN
Случайные числа:
.1002502 .9648132
.8866272 .6364441
```

```
ОСТ В СТРОКЕ    60
Готов
```

#### 6.3.2.14. Оператор **CLS**

Функция: Очищает экран ЖКИ и переводит курсор в верхнюю левую позицию  
Формат: CLS

Используется для очистки экрана ЖКИ по мере необходимости.

После выполнения оператора курсор переходит в позицию (0, 0), при этом изображение маркера отсутствует.

#### 6.3.2.15. Оператор **DRAW**

Оператор DRAW используется для высвечивания или гашения точки (линии) на экране.

1) Оператор **DRAW/O**

Функция: Задает текущую точку экрана

Формат: DRAW O<координата X>, <координата Y>

Координаты точки X, Y ограничены размерами экрана:  $0 \leq X \leq 119$ ,  $0 \leq Y \leq 63$ .

2) Оператор **DRAW/H**

Функция: Высвечивает точку в заданной позиции экрана

Формат: DRAW H<координата X>, <координата Y>

Координаты точки X, Y ограничены размерами экрана:  $0 \leq X \leq 119$ ,  $0 \leq Y \leq 63$ .

3) Оператор **DRAW/D**

Функция: Вычерчивает отрезки прямых между последовательностью заданных точек

Формат: DRAW D<начальная координата X>,  
          <начальная координата Y>,  
          <координата X>, <координата Y>

Используется для высвечивания на экране точки, прямой линии, ломанной линии, заданных абсолютными координатами X, Y.

Координаты точек X, Y ограничены размерами экрана:  $0 \leq X \leq 119$ ,  $0 \leq Y \leq 63$ .

Если соседние пары координат X, Y одинаковы, то команда вычерчивает в данной позиции точку.

Пример:

```
5  CLS
10 DRAW D50,30,50,30
20 DRAW D70,5,70,5
25  DIS
```



#### 4) Оператор **DRAW/E**

Функция: Гасит точку (линию) в заданном положении

Формат: DRAW E<начальная координата X>,  
          <начальная координата Y>,  
          <координата X>,<координата Y>

Используется для гашения на экране точки, линии, заданных абсолютными координатами X, Y.

Координаты точек X, Y ограничены размерами экрана:  $0 \leq X \leq 119$ ,  $0 \leq Y \leq 63$ .

Если соседние пары координат X, Y одинаковы, то команда гасит в данной позиции точку.

Пример:

```
5 CLS
10 FOR I=1 TO 3
20 DRAW H10,I
25 REM - ЗАДЕРЖКА
30 FOR J=1 TO 200: NEXT J
40 DRAW E10,I,10,I
50 NEXT I
```

#### 5) Оператор **DRAW/I**

Функция: Вычерчивает прямую линию, координаты которой заданы относительно

Формат: DRAW I<приращение по X>,<приращение по Y>

Используется для высвечивания отрезков прямых, координаты конечных точек которых заданы приращениями относительно текущей точки экрана.

Пример:

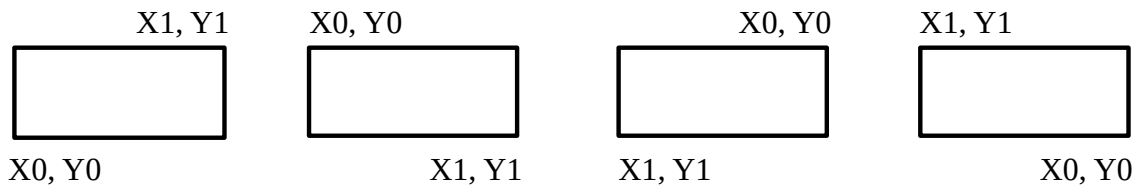
```
5 CLS
10 DRAW O20,32
20 DRAW I0,0,20,10,-5,-30,-10,0
30 REM - ЗАДЕРЖКА
40 FOR J=1 TO 200: NEXT J
50 REM - УСТАНОВИТЬ СЛУЖЕБНУЮ СТРОКУ, МАРКЕР
60 DIS
```

6) Оператор **DRAW/A**

Функция: Вычерчивает прямоугольник, заданный координатами его диагонали

Формат: DRAW A<начальная координата X>,  
<начальная координата Y>,  
<диагональная координата X>,  
<диагональная координата Y>

Используется для вычерчивания прямоугольника, координаты которого задаются следующим образом:



Координаты точек X, Y ограничены размерами экрана:  $0 \leq X \leq 119$ ,  $0 \leq Y \leq 63$ .

Пример:

```
5 CLS
10 DRAW O60,32
20 FOR I=30 TO 5 STEP -5
30 DRAW A60-I,32+I,60+I,32-I
40 NEXT I
```

Результат выполнения приведен на Рис. 6.4.

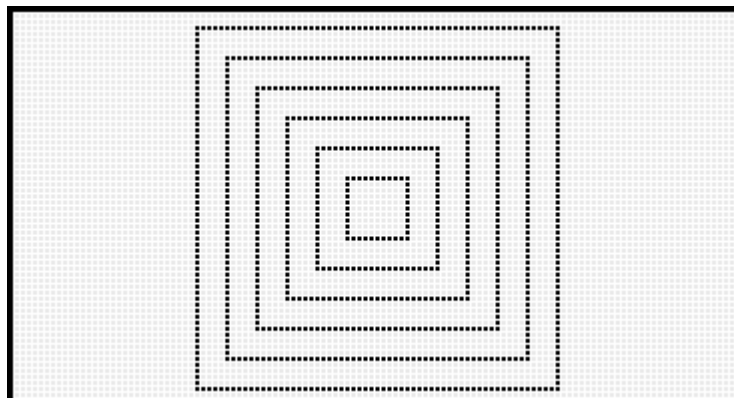


Рис. 6.4

### 7) Оператор **DRAW/X**

Функция: Вычерчивает оси координат

Формат: `DRAW X<направление оси координат>,  
<размер деления оси>,<количество делений>`

Используется для высвечивания осей координат для линейных графиков и гистограмм. Параметр направления оси координат принимает значения 0, 1, 2 или 3.

0 – вычерчивается ось Y из начала координат вверх (+Y)

1 – вычерчивается ось X из начала координат вправо (+X)

2 – вычерчивается ось Y из начала координат вниз (-Y)

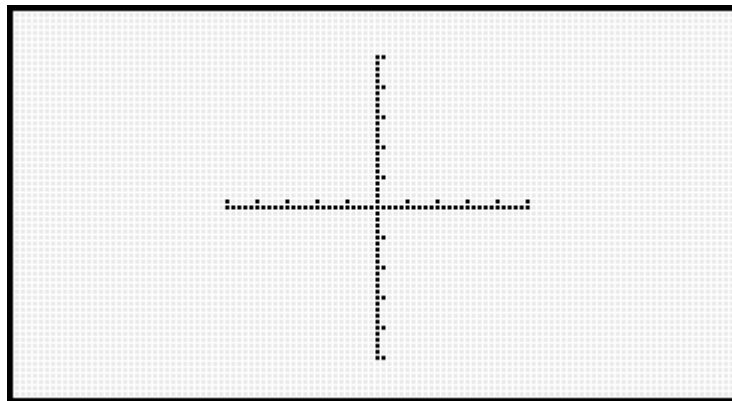
3 – вычерчивается ось X из начала координат влево (-X)

Следующие параметры – целые положительные числа, задают расстояние между метками шкалы в точках и количество меток шкалы соответственно.

Пример:

```
5 CLS
10 DRAW O60,32
20 FOR I=0 TO 3
30 DRAW XI,5,5
40 NEXT I
```

Графический пример представлен на Рис. 6.5.



**Рис. 6.5**

### 8) Оператор **DRAW/G**

Функция: Высвечивание вертикальных или горизонтальных полос заданной ширины

Формат: `DRAW G<направление штриховки>,  
<протяженность по оси X>,  
<протяженность по оси Y>  
[, <расстояние между линиями>]`

Используется для штриховки в заданном направлении прямоугольной области, начальной точкой для которой служит текущая точка.

Параметр направления штриховки принимает значения 0, 1, 2

0 – без штриховки

1 – горизонтальные полосы

2 – вертикальные полосы

Параметр <расстояние между линиями> может быть опущен.

Пример:

```
5 CLS
10 DRAW OO,0
20 FOR I=0 TO 2
30 DRAW AI*30,20,I*30+20,0
35 LOCATE 30,I
40 DRAW GI,20,20,2
50 NEXT I
```

## 9) Оператор **DRAW/C**

Функция: Вычерчивает окружность по заданным координатам центра и радиусу

Формат: DRAW C<координата X центра>,<координата Y центра>,<радиус>

Используется для вычерчивания окружности заданного радиуса вокруг центра, заданного абсолютными координатами (X, Y). Координаты центра (X, Y) ограничены размерами экрана:  $0 \leq X \leq 119$ ,  $0 \leq Y \leq 63$  и  $\langle X \rangle$  не равно  $\langle Y \rangle^*$ .

Пример:

```
10 CLS
20 FOR I=1 TO 3
30 DRAW C60,32,I*10
40 NEXT I
```

## 10) Оператор **DRAW/M**

Функция: Высвечивает маску, заданную шестнадцатеричным числом

Формат: DRAW M<маска>

Используется для заполнения прямоугольных областей заданной <маской>. Числа из заданной маски выбираются попарно и формируются в точек экрана по горизонтали. Заполнение областей осуществляется сверху вниз от текущего положения маркера или позиции, заданной командой LOCATE. Позволяет выводить нерегулярные фигуры с произвольной конфигурацией.

---

**\*Прим. редактора:** обусловлено ошибкой в программе отрисовки окружности. При  $X=Y$  не отрисовывается вторая координатная четверть окружности.

Пример:

```
10 CLS
20 FOR I=1 TO 120
30 DRAW MF0F0F0F00F0F0F0F
40 NEXT I
```

Высвечивание точек выполняется следующим образом:

Маска	Двоичный эквивалент	
F0F0	11110000	11110000
AAAA	10101010	10101010

Высвеченные точки

Если текущее положение маркера или позиции равно точке с координатами (X, Y), то при работе оператора DRAW/M для каждой из пар шестнадцатеричных чисел происходит автоматическое изменение координаты Y, т.е.  $Y=Y+1$ . При  $Y=64$  автоматически полагается  $Y=0$ , а  $X=X+8$ .

При  $X>119$  и  $Y>63$  маски не отображаются.

#### 6.3.2.16. Оператор LOCATE

Функция: Задает позицию курсора

Формат: LOCATE <X>, <Y>

Используется для изменения положения курсора. Координаты X, Y ограничены размерами экрана:  $0 \leq X \leq 119$ ,  $0 \leq Y \leq 63$ .

Пример:

```
1) 5 CLS
    10 LET X=1
    20 LET X=X+1
    40 PRINT "X="; X
    50 GOTO 20
```

```
2) 5 CLS
    10 LET X=1
    20 LET X=X+1
    30 LOCATE 20,20
    40 PRINT "X="; X
    50 GOTO 20
```

В случае 1) печать результата начинается с текущей позиции курсора.

В случае 2), в связи с добавлением строки 30, печать результата начинается с позиции (20, 20).

#### 6.3.2.17. Оператор **STOP**

Функция: Останов выполнения программы. БЕЙСИК печатает:

ОСТ В СТРОКЕ     XXX

ГОТОВ

Формат: STOP

Оператор STOP используется для логического окончания программы, может помещаться в любом месте программы.

Программа, предназначенная для временного использования, может быть без оператора STOP; ее выполнение обычно заканчивается на строке с самым большим номером.

Оператор STOP помогает при отладке программ пользователя, он может помещаться в критических точках программы, выполнение программы останавливается в этих точках. При останове для исследования значения конкретных переменных можно использовать оператор PRINT в непосредственном режиме.

Если в этой точке выполнение программы оказывается правильным, она может быть продолжена оператором GOTO в непосредственном режиме, который обращается в этом случае к оператору, следующему за оператором STOP, либо оператор STOP может быть вычеркнут, в этом случае выполнение программы возобновляется с помощью команды RUN.

#### 6.3.2.18. Оператор **DIS**

Функция: Восстанавливает служебную строку

Формат: DIS

Оператор DIS используется для восстановления служебной строки после использования графических команд, которые размещают информацию на месте, выделенном под служебную строку.

#### 6.3.2.19. Оператор **WAIT**

Функция: Приостанавливает выполнение программы

Формат: WAIT <A>

Размер аргумента – от 1 до 32536. Минимальная задержка при A=1, максимальная — при A=32536. Максимальная задержка – 0.1 мс.

### 6.3.2.20. Оператор **PLAY**

Функция: Генерирует тон заданной частоты и длительности

Формат: **PLAY** <тон>, <длительность>

Параметр <тон> задает номер тона от 0 до 40.

Параметр <длительность> определяет продолжительность звучания генерируемой ноты и задается числом от 1 до 32767. Длительность целой ноты определяется экспериментальным путем.

Пример: программа воспроизведения мелодии "Подмосковные вечера"

```

10 FOR J=0 TO 2
20 READ K
30 FOR I=1 TO K
40 READ A,B,C
50 PLAY A,B*15
60 WAIT C
70 NEXT I
80 RESTORE
90 NEXT J
100 END
110 DATA 54: 'КОЛИЧЕСТВО НОТ
120 DATA 20,2,1,23,2,1,27,2,1,23,2,1,25,4,1,23,2,1
130 DATA 22,2,1,27,4,1,25,4,1,20,8,1,23,2,1,27,2,1
140 DATA 30,2,1,30,2,1,32,4,1,30,2,1,28,2,1,27,8,1
150 DATA 29,4,1,31,4,1,34,2,1,32,2,1,27,6,1,22,4,1
160 DATA 20,2,1,27,3,1,25,1,1,28,6,1,30,2,1,28,2,1
170 DATA 27,4,1,25,2,1,23,2,1,27,4,1,25,4,1,32,8,1
180 DATA 29,4,1,31,4,1,34,2,1,32,2,1,27,6,1,22,4,1
190 DATA 20,2,1,27,3,1,25,1,1,28,8,1,30,2,1,28,2,1
200 DATA 27,4,1,25,2,1,23,2,1,27,4,1,25,4,1,20,12,1

```

Таблица приблизительных частот каждой ноты (Гц):

Нота	Част.	Нота	Част.	Нота	Част.	Нота	Част.	Нота	Част.	Нота	Част.
<b>0</b>	700	<b>7</b>	1050	<b>14</b>	1570	<b>21</b>	2350	<b>28</b>	3515	<b>35</b>	5280
<b>1</b>	740	<b>8</b>	1110	<b>15</b>	1665	<b>22</b>	2495	<b>29</b>	3730	<b>36</b>	5535
<b>2</b>	785	<b>9</b>	1175	<b>16</b>	1760	<b>23</b>	2640	<b>30</b>	3970	<b>37</b>	5925
<b>3</b>	830	<b>10</b>	1245	<b>17</b>	1865	<b>24</b>	2790	<b>31</b>	4190	<b>38</b>	6275
<b>4</b>	880	<b>11</b>	1320	<b>18</b>	1975	<b>25</b>	2960	<b>32</b>	4430	<b>39</b>	6640
<b>5</b>	930	<b>12</b>	1400	<b>19</b>	2095	<b>26</b>	3140	<b>33</b>	4690	<b>40</b>	7050
<b>6</b>	990	<b>13</b>	1480	<b>20</b>	2215	<b>27</b>	3320	<b>34</b>	4970		

Формула для вычисления приблизительной длительности каждой ноты:

$$t = \frac{\text{длительность} \times 16}{\text{частота ноты}} \quad [\text{сек}]$$

### 6.3.3. Встроенные функции

БЕЙСИК содержит десять функций для выполнения вычислительных операций. Эти функции обозначаются именем, состоящим из трех букв, за которым следует аргумент в круглых скобках. Функции могут использоваться в любом месте программы в качестве выражений или элементов выражения везде, где выражения являются допустимыми.

#### 6.3.3.1. Функции SIN(X), COS(X)

В качестве аргумента задается угол, выраженный в радианах. Величина аргумента ограничивается значением  $\pm 51470$ . Если угол выражен в градусах, нужно перевести градусы в радианы с помощью следующей формулы:

$$\text{радианы} = \text{градусы} \times \frac{\pi}{180}$$

В примере\* в качестве PI используется число 3,14159265. В строке 40 приведенное значение используется для перевода в радианы.

```
10 REM - ПЕРЕВЕСТИ УГОЛ (X) В РАДИАНЫ
11 REM - ВЫЧИСЛИТЬ SIN И COS
30 INPUT X
40 LET Y=X*PI/180
60 PRINT "Градусы:" X
70 PRINT "Радианы:" Y
80 PRINT "Синус:" SIN(Y)
90 PRINT "Косинус:" COS(Y)
92 PRINT
95 GOTO 30
Готов
```

```
RUN
?0          ?10          ?45
Градусы: 0   Градусы: 10   Градусы: 45
Радианы: 0   Радианы: .1745329 Радианы: .7853982
Синус: 0     Синус: .1736482 Синус: .7071068
Косинус: 1   Косинус: .9848078 Косинус: .7071068
```

```
?90          ?ПР
Градусы: 90
Радианы: 1.570796   Готов
Синус: 1
Косинус: .1462918E-8
```



#### 6.3.3.2. Функция ATN(X)

Функция арктангенс вычисляет во заданному значению аргумента X значение угла в радианах в интервале от  $+\frac{\pi}{2}$  до  $-\frac{\pi}{2}$ .

В приведенной ниже программе\* вводимое число представляется в градусах, в строке 40 оно переводится в радианы. В строке 50 значение угла в радианах используется для получения тангенса с помощью уравнения:

$$\tan(X) = \frac{\sin(X)}{\cos(X)}$$



В строке 70 значение тангенса (Z) задается в качестве аргумента функции ATN для получения значения, которое печатается с заголовком ATAN(Z). В строке 70 значение функции арктангенса в радианах снова переводится в градусы и при распечатке помещается в пятой линии, оно служит для проверки значения, указанного в первой линии.

```
20 PRINT "Задать угол в грд."
30 INPUT X
40 LET Y=X*PI/180
50 LET Z=SIN(Y)/COS(Y)
60 PRINT "Градусы:" X
70 PRINT "Радианы:" Y
75 PRINT "TAN(X) :" Z
80 PRINT "ATAN(Z) :" ATN(Z)
85 PRINT "ATAN градусы:" ATN(Z)*180/PI
86 PRINT
90 GOTO 30
Готов
```

```
RUN
Задать угол в грд.
?0
Градусы: 0
Радианы: 0
TAN(X) : 0
ATAN(Z) : 0
ATAN градусы: 0
```

```
?45
Градусы: 45
Радианы: .7853982
TAN(X) : 1
ATAN(Z) : .7853982
ATAN градусы: 45
```

```
?90
Градусы: 90
Радианы: 1.570796
TAN(X) : .6835653E 9
ATAN(Z) : 1.570796
ATAN градусы: 90
```

?~P



Готов

### 6.3.3.3. Функция LOG(X)

Функция LOG определяет натуральный логарифм заданного аргумента.

---

**\*Прим. редактора:** помеченные \* программы были переадаптированы из табличного отображения в строковое для удобства просмотра на ЖКИ устройства.

Пример:

```
10 INPUT X
20 PRINT LOG (X)
25 PRINT
30 GOTO 10
Готов
```

```
RUN
?54.59815
4
```

```
?22026.47
10
```

```
?100
4.60517
```

```
? .720049E11
25
```

```
?¬Р   
```

Готов

Натуральные логарифмы легко перевести в логарифмы с любым другим основанием, используя следующую формулу:

$$\log_A(N) = \frac{\log(N)}{\log(A)}$$

Где А – нужное основание. Следующая программа иллюстрирует перевод в десятичные логарифмы.

Пример\*:

```
1 REM - ПЕРЕВОД НАТУРАЛЬНОГО ЛОГАРИФМА В ДЕСЯТИЧНЫЙ
15 INPUT X
17 PRINT "Значение:" X
20 PRINT "LN:" LOG (X)
40 PRINT "LOG:" LOG (X) / LOG (10)
45 PRINT
50 GOTO 15
60 END
Готов
```

RUN		
?4	?250	?60
Значение: 4	Значение: 250	Значение: 60
LN: 1.386294	LN: 5.521461	LN: 4.094345
LOG: .60206	LOG: 2.39794	LOG: 1.778151

```
?¬Р   
```

Готов

#### 6.3.3.4. Функция **EXP(X)**

Показательная функция возводит число "Е" в степень X. EXP является функцией, обратной функции LOG, т.е.

$$\log(\exp(X)) = X$$

Пример:

```
10 INPUT X
20 PRINT EXP (X)
25 PRINT
30 GOTO 10
Готов
```

RUN

?4

54.59815

?9.421006

12344.99

?25

.790049E 11

?¬P



Готов

#### 6.3.3.5. Функция **ABS(X)**

Абсолютная функция ABS вычисляет абсолютное значение любого аргумента.

Пример:

```
10 INPUT X
20 LET X=ABS (X)
30 PRINT X
35 PRINT
40 GOTO 10
Готов
```

RUN

?-35.7

35.7

?2

2

?105555567

.1055556E 9

?-44.555566668899

44.55557

?¬P



Готов

### 6.3.3.6. Функция INT(X)

Целочисленная функция вычисляет значение наибольшего целого числа, не превышающего значения аргумента.

Например:

```
PRINT INT(34.67)
34
```

```
PRINT INT(-5.1)
-6
```

Функция может использоваться для округления чисел до ближайшего целого числа с помощью `INT (X+.5)`.

Например:

```
PRINT INT(34.67+.5)
34
```

```
PRINT INT(-5.1+.5)
-5
```

INT(X) может использоваться для округления до любого десятичного разряда с помощью следующего выражения в качестве аргумента:

$$\frac{\text{int}(X \times 10^D + 0.5)}{10^D}$$

Где D – целое число, задаваемое пользователем.

Пример:

```
20 PRINT "Округление:"
25 INPUT A
40 PRINT "Число дес. разрядов:"
45 INPUT D
60 LET B=INT (A*10^-D+.5)/10^-D
70 PRINT "А после окр. =" B
75 PRINT
80 GOTO 20
90 END
Готов
```

```
RUN
Округление:
?55.65842
Число дес. разрядов:
?2
А после окр. = 55.66
```

```
Округление:
?78.375
Число дес. разрядов:
?-2
А после окр. = 100
```

```
?~P
```



Готов

### 6.3.3.7. Функция INC

Вводит десятичный код клавиши, нажатой в текущий момент. Нулевое значение в качестве результата функции означает, что либо не нажата клавиша, либо нажата одна из клавиш:



Пример:

```
10 LET A=INC
20 IF A=0 THEN GOTO 10
30 PRINT A
40 GOTO 10
```

Таблицы кодов символов приведены в **Приложении 5**.

### 6.3.3.8. Функция SGN(X)

Функция знака SGN(X) дает значение:

1, если  $X > 0$   
0, если  $X = 0$   
-1, если  $X < 0$

Пример:

```
PRINT SGN (3.42)
1
PRINT SGN (-42)
-1
PRINT SGN (23-23)
0
```

### 6.3.3.9. Функция SQR(X)

Функция квадратного корня извлекает квадратный корень из любого положительного значения аргумента (X).

Пример:

```
10 INPUT X
20 LET X=SQR(X)
30 PRINT X
40 GOTO 10
Готов
```

RUN

?16

4

?1000

31.62278

?123456789

11111.11

?25E2

50

?¬P

Готов



#### 6.3.3.10. Функция **RND(X)**

Функция случайных чисел генерирует случайное число или совокупность случайных чисел в интервале между 0 и 1. Аргумент (X) не используется и может быть любым числом.

Пример:

```
10 PRINT "Случайные числа"
30 FOR I=1 TO 15
40 PRINT RND(0)
50 NEXT I
60 END
Готов

RUN
Случайные числа
.1092502 .9648132 .8866272 .6364441 .8390198 .3061218 .285553
.9582214 .1793518 .4521179 .9854126E-1 .5221863 .2462463
.7778015 .450592
ОСТ В СТРОКЕ 60
Готов
```

Можно генерировать случайные числа в заданном интервале. Если нужен интервал (A, B), используется выражение:

$$(B - A) \times rnd(0) + A$$

Следующая программа генерирует множество случайных чисел в интервале (4, 6).

Пример:

```
10 REM - ПОСЛЕДОВАТЕЛЬНОСТЬ СЛУЧАЙНЫХ ЧИСЕЛ ИЗ (4, 6)
20 FOR B=1 TO 15
30 LET A=(6-4)*RND(0)+4
40 PRINT A
50 NEXT B
60 END
Готов

RUN
4.2005 5.929626 5.773254 5.272888 5.67804 4.612244 4.571106
5.916443 4.358704 4.904236 4.197083 5.044373 4.492493
5.555603 4.901184

ОСТ В СТРОКЕ 60
Готов
```

#### 6.3.3.11. Функция **PI**

Функция, определяет число с точностью до седьмого знака: 3,141696.

Используется при вычислении тригонометрических функций. О применении функции PI смотрите в разделе применения тригонометрических функций.

Пример:

```
LET A=SIN(PI)
```

## **7. Правила хранения**

7.1. Микрокалькулятор необходимо хранить в сухом отапливаемом помещении при отсутствии в воздухе кислотных, щелочных и других агрессивных примесей при температуре от +1 до +50°C и при относительной влажности не более 85%.

Схема клавиатуры для работы в функциональном режиме





## Команды микрокалькулятора

### 1. Команды непосредственного режима

#### HELP

---

Функция: Выдает на экран ЖКИ информацию о составе команд языка БЕЙСИК, их назначении, формате записи и привязке к функциональной клавиатуре

Формат: HELP \*  
HELP <тема>

Примеры: HELP DRAW/A  
HELP HELP

#### DELETE

---

Функция: Удаляет строки из программы

Формат: DELETE [начальный номер строки] [, конечный номер строки]

Примеры: DELETE 50  
DELETE 70,120  
DELETE 1,8191

#### LIST

---

Функция: Печатает текст программы

Формат: LIST [начальный номер строки] [, конечный номер строки]

Примеры: LIST  
LIST 20  
LIST 30,50

#### RUN

---

Функция: Выполняет программу

Формат: RUN

Пример: RUN

## SAVE

---

Функция: Сохраняет программы на СМП

Формат: SAVE "[SM\*:]имя файла[.тип]"

Примеры: SAVE "SM0:TEXT.BAS"  
SAVE "SM1:BASIC"

## LOAD

---

Функция: Загружает в память программу из СМП

Формат: LOAD "[SM\*:]имя вводимого файла[.тип]"

Примеры: LOAD "SM0:TEXT.BAS"  
LOAD "SM1:BASIC"

## INIT

---

Функция: Инициализирует СМП

Формат: INIT ["SM\*:"]

Примеры: INIT "SM1:"  
INIT

## KILL

---

Функция: Удаляет из СМП указанный файл

Формат: KILL "[SM\*:]имя файла.тип"

Пример: KILL "SM0:TEXT.BAS"

## NAME (AS)

---

Функция: Переименовывает файл на СМП

Формат: NAME "<старое имя>" AS "<новое имя>"

Пример: NAME "SM1:OLD.BAS" AS "NEW.BAS"

## FILES

---

Функция: Распечатывает справочник СМП

Формат: FILES ["SM\*:"]

Примеры: FILES  
FILES "SM1:"

## DEV

---

Функция: Задает имя устройства в качестве рабочего СМП

Формат: DEV ["SM\* : "]

Примеры: DEV

DEV "SM1 : "

## AUTO

---

Функция: Автоматически нумерует строки (выход из режима по СУ/P)

Формат: AUTO [[начальный номер][, шаг]]

Примеры: AUTO

AUTO , 20

AUTO 5, 10

## MEM

---

Функция: Выдает распределение памяти

Формат: MEM

Пример: MEM

## EDIT

---

Функция: Редактирует строки программы

Формат: EDIT <номер строки>

Пример: EDIT 150

### Подфункции EDIT

СУ/A — перемещение маркера в начало строки

СУ/B — перемещение маркера в начало предыдущего слова

СУ/E — стирание символа от маркера до конца строки

СУ/F — перемещение маркера в начало следующего слова

СУ/X — перемещение маркера в конец строки



— режим вставки



— сдвиг строки до позиции маркера



— перемещение маркера на одну позицию вправо



— перемещение маркера на одну позицию влево



— выход из режима редактирования

## 2. Команды программного режима

### DIM

---

Функция: Резервирует память для формирования массива

Формат: DIM имя массива (числовое значение [, числовое значение])

Примеры: DIM A (10)  
          DIM B (5, 2)

### DEF (FN)

---

Функция: Определяет функции и выражения как отдельную функцию

Формат: DEF FN<буква> (параметр) =<выражение>

Примеры: DEF FNX (S) = S<sup>2</sup> + 4  
          DEF FNZ (X) = X<sup>2</sup>  
          DEF FNA (X) = 4 + 2

### END

---

Функция: Последний оператор программы

Формат: END

Пример: END

### FOR-TO-STEP/NEXT

---

Функция: Операторы FOR и NEXT используются для обозначения начальной и конечной точек программы.

Все операторы, стоящие между оператором FOR и соответствующим ему оператором NEXT, будут циклически выполняться в соответствии с условиями, заданными в операторе FOR.

Формат: FOR <имя переменной> = <числовое выражение>  
          TO <числовое выражение> [STEP <числовое выражение>]  
          NEXT <имя переменной>

Примеры: FOR I = 0 TO 9  
          NEXT I

          FOR A = 1 TO 9 STEP 2  
          PRINT A  
          NEXT A

## GOSUB/RETURN

---

Функция: Осуществляет вход в подпрограмму и возвращение из подпрограммы

Формат: GOSUB <номер строки>  
RETURN

Примеры: GOSUB 200  
RETURN

## GOTO

---

Функция: Осуществляет безусловный переход к строке с указанным номером

Формат: GOTO <номер строки>

Пример: GOTO 210

## IF (THEN)

---

Функция: Обеспечивает условный переход в программе

Формат: IF <условное выражение> THEN <оператор>  
THEN <номер строки>  
GOTO <номер строки>

Примеры: IF A>B THEN 10  
IF A>4 GOTO 50  
IF x>=70 THEN GOSUB 100

## INPUT

---

Функция: Вводит данные с клавиатуры во время выполнения программы

Формат: INPUT <имя переменной> [, <имя переменной>, ...]

Примеры: INPUT X  
INPUT A, B, C

## LET

---

Функция: Присваивает значение переменной

Формат: LET <имя переменной> = <числовое выражение>

Примеры: LET A=2  
LET X=A+B  
LET X=Y\*2

## PRINT

---

Функция: Выводит (печатает) данные на экран

Формат: PRINT  
PRINT <выражение> [, <выражение>, ...]  
PRINT "строка символов" [, "строка символов", ...]

Примеры: PRINT A, B  
PRINT A+B  
PRINT "Результат"

## DATA

---

Функция: Вводит в программу числовые константы, с которыми оператор READ последовательно связывает имена переменных

Формат: DATA <число> [, <число>, ...]

Пример: DATA 5, 6, 7, 8

## READ

---

Функция: Читает данные, которые были определены в DATA и последовательно связывает их со своими переменными

Формат: READ <имя переменной> [, <имя переменной>, ...]

Пример: READ S, T

## RESTORE

---

Функция: Осуществляет повторное использование констант оператора DATA, который имеет самый низкий номер строки в программе

Формат: RESTORE

Пример: RESTORE

## REM (REMARK)

---

Функция: Определяет комментарии в программе пользователя

Формат: REM – комментарий  
REMARK – комментарий

Пример: REM – ВЫЧИСЛЕНИЕ ФАКТОРИАЛА X

## WAIT

---

Функция: Приостанавливает выполнение программы

Формат: WAIT A  
Аргумент A определяет величину задержки.

Пример: WAIT 3530

## **RANDOMIZE**

---

Функция: Позволяет получить ряд случайных чисел в программе,  
используя функцию RND

Формат: RANDOMIZE

Пример: RANDOMIZE

## **STOP**

---

Функция: Вызывает прекращение выполнения программы

Формат: STOP

Пример: STOP

## **CLS**

---

Функция: Очищает экран ЖКИ и переводит курсор в крайнюю левую позицию

Формат: CLS

Пример: CLS

## **DIS**

---

Функция: Восстанавливает служебную строку

Формат: DIS

Пример: DIS

## **DRAW/O**

---

Функция: Определяет текущую точку экрана

Формат: DRAW O<координата X>, <координата Y>

Примеры: DRAW OX, Y  
DRAW O60, 32

## **DRAW/H**

---

Функция: Высвечивает точку в заданной позиции

Формат: DRAW H<координата X>, <координата Y>

Пример: DRAW HX, Y

## DRAW/D

---

Функция: Выводит отрезки прямых между последовательностью заданных точек

Формат: DRAW D<начальная координата X>,<начальная координата Y>,  
<координата X>,<координата Y>

Пример: DRAW D34,34,67,50

## DRAW/E

---

Функция: Удаляет точку (линию) в заданном положении

Формат: DRAW E<начальная координата X>,<начальная координата Y>,  
<координата X>,<координата Y>

Пример: DRAW E34,34,67,50

## DRAW/I

---

Функция: Выводит прямую линию, координаты которой заданы относительно

Формат: DRAW I<приращение по X>,<приращение по Y>[,...]

Примеры: DRAW I20,10,-5,-30  
DRAW I-10,0

## DRAW/A

---

Функция: Выводит прямоугольник, заданный координатами его диагонали

Формат: DRAW A<начальная координата X>,<начальная координата Y>,  
<диагональная координата X>,  
<диагональная координата Y>

Примеры: DRAW A0,0,30,60  
DRAW A-I,I,I,-I

## DRAW/C

---

Функция: Выводит окружность по заданным координатам центра и радиусу

Формат: DRAW C<координата X центра>,<координата Y центра>,  
<радиус>

Пример: DRAW C60,32,10



## **DRAW/X**

---

Функция: Выводит оси координат

Формат: DRAW X<направление оси координат>, <размер деления оси>,  
<количество делений>

Пример: DRAW XI, 5, 5

## **DRAW/G**

---

Функция: Вычерчивание вертикальных или горизонтальных полос заданной ширины

Формат: DRAW G<направление штриховки>, <протяженность по оси X>,  
<протяженность по оси Y>  
[, <расстояние между линиями>]

Пример: DRAW GI, 20, 20, 2

## **DRAW/M**

---

Функция: Выводит маски, заданные шестнадцатеричным числом

Формат: DRAW M<маска>

Пример: DRAW MF0F0F0F0F0F0F0F0F

## **LOCATE**

---

Функция: Определяет позицию курсора

Формат: LOCATE <координата X>, <координата Y>

Пример: LOCATE 60, 32

## **PLAY**

---

Функция: Подает звуковые сигналы

Формат: PLAY <тон>, <длительность>

Пример: FOR I=1 TO 40  
PLAY 1, 20  
NEXT I

### 3. Вычислительные функции

#### SIN

---

Функция: Тригонометрическая функция синуса –  $\text{SIN}(X)$

Формат:  $\text{SIN}(\text{числовое выражение})$

Пример:  $\text{SIN}(A/B)$

#### COS

---

Функция: Тригонометрическая функция косинуса –  $\text{COS}(X)$

Формат:  $\text{COS}(\text{числовое выражение})$

Пример:  $\text{COS}(A*10)$

#### ATN

---

Функция: Обратная тригонометрическая функция от  $\text{TAN}(X)$

Формат:  $\text{ATN}(\text{числовое выражение})$

Пример:  $\text{ATN}(A/100)$

#### SQR

---

Функция: Квадратный корень

Формат:  $\text{SQR}(\text{числовое выражение})$

Пример:  $\text{SQR}(30)$

#### EXP

---

Функция: Показательная функция

Формат:  $\text{EXP}(\text{числовое выражение})$

Пример:  $\text{EXP}(1)$

#### LOG

---

Функция: Натуральный логарифм

Формат:  $\text{LOG}(\text{числовое выражение})$

Пример:  $\text{LOG}(2.71828)$

## ABS

---

Функция: Абсолютное значение

Формат: ABS (числовое выражение)

Пример: ABS (-10.5)

## INT

---

Функция: Целочисленная функция. Вычисляет значение наибольшего числа, не превышающего значения аргумента

Формат: INT (числовое выражение)

Пример: INT (3.14)

## SGN

---

Функция: Знак числового выражения

Формат: SGN (числовое выражение)

Пример: SGN (-1)

## RND

---

Функция: Генерирует случайное число или совокупность случайных чисел в интервале [0, 1]

Формат: RND (любое число)

Пример: RND (0)

## INC

---

Функция: Выдача кода нажатой клавиши

Формат: INC

Пример: 10 LET A=INC: IF A=0 GOTO 10  
20 PRINT A

## PI

---

Функция: Определяет число пи с точностью до седьмого знака

Формат: PI

Пример: LET A=SIN(PI)

## Сообщения об ошибках

### 1. Формат сообщений

БЕЙСИК осуществляет проверку операторов программы и вводимых данных и для каждой обнаруженной ошибки печатает соответствующее сообщение об ошибке.

Сообщения об ошибках имеют следующий формат:

ОШ    XXX    СТР    YYY

Где XXX – код ошибки, YYY – номер строки, в которой произошла ошибка.

Если YYY = 0, это означает, что ошибка произошла в режиме команд, т.е. в последней вводимой строке.

### 2. Ошибки

Коды ошибок с 0 по 64 указывают на неустраняемые ошибки, выполнение программы прекращается после печати сообщения об ошибке.

Неустраняемые ошибки приведены в Табл. 1.

Коды ошибок с 89 по 127 указывают на устранимые ошибки; после печати сообщения об ошибке выполнение программы продолжается.

Устранимые ошибки приведены в Табл. 2.

Таблица 1

#### Неустраняемые ошибки

Код ошибки	Содержание ошибки
0	Переполнение памяти, отведенной пользователю
1	Нераспознаваемый оператор
2	Недопустимый оператор GOTO или GOSUB
3	Недопустимый знак, ограничивающий оператор (обычно в случае неправильно сформированного оператора, который вызывает преждевременное окончание действий оператора)
4	Оператор RETURN без соответствующего оператора GOSUB
5	Неправильно сформирован индекс
6	Индекс не попадает в интервал от 0 до 255 или превышает максимум, установленный программой
7	Несоответствие скобок в операторе

Продолжение Табл. 1

Код ошибки	Содержание ошибки
8	Недопустимый оператор LET
9	Недопустимый знак операции отношения в операторе IF
10	Недопустимый оператор IF
11	Недопустимый оператор PRINT
12	Слишком длинная вводимая строка (превышает 80 знаков)
13	Неправильная размерность в операторе DIM
14	В памяти недостаточно места для массива
15	Неправильно сформирован оператор DEF
16	Недопустимый номер строки или значение размерности
17	Оператор DIM для ранее описанного или использованного элемента
18	Неправильная переменная в списке оператора INPUT
19	Неправильная переменная в списке оператора READ
20	Данные в списке оператора READ исчерпаны
21	Неправильный формат оператора DATA
22	Недопустимый оператор FOR
23	FOR не сопровождается соответствующим оператором NEXT
24	Оператор NEXT без оператора FOR
25	Несоответствие кавычек в операторе
27	Неправильно сформировано выражение (опущен порядок числа в формате E)
29	Неверная функция оператора DRAW
30	Не задан параметр
31	Неверная нота в операторе PLAY
59	Нет места в СМП
60	Не найден переименовываемый файл
61	Неверный формат оператора NAME
62	Файл не найден
63	Синтаксическая ошибка в имени файла

Таблица 2

Устранимые ошибки

Код ошибки	Содержание ошибки
120	Недопустимые знаки при вводе
121	Введено недостаточно данных по оператору INPUT
122	Введено слишком много данных по оператору INPUT
123	Несуществующая переменная
124	Число слишком велико для фиксации (вероятно, комбинация индексов превышает диапазон)
125	Переполнение или заем при делении/умножении
126	Квадратный корень из отрицательного числа
127	Логарифм отрицательного числа или нуля; Переполнение при вычислении EXP

## Примеры программ

### 1. Построение кривых

#### 1.1. Построение окружности

При использовании оператора DRAW/C может быть вычерчена окружность. При этом достаточно определить координаты ее центра и радиус.

Программа, с помощью которой производится вычерчивание окружности радиусом 25 и с центром в координатах (50, 32):

```
10 DRAW C50,32,25
```

Теперь вычертим концентрические окружности радиусами 10, 20 и 30 с центром в координатах (50, 32).

```
5 CLS
10 FOR R=1 TO 3
20 DRAW C50,32,R*10
30 NEXT R
```

#### 1.2. Построение синусоиды

В программе построения синусоиды используется построение осей координат, нанесение делений шкалы и вычерчивание синусоиды с помощью оператора DRAW/X.

Значения шкал могут быть расположены в вертикальном направлении с помощью оператора PRINT.

```
10 CLS
20 FOR I=0 TO 2 STEP 2
30 DRAW O20,32
40 DRAW XI,5,4: NEXT I
50 DRAW O20,32
60 DRAW X1,10,8
70 GOSUB 200
80 GOSUB 400
90 GOSUB 600
100 END
200 LET A=28
205 FOR I=90 TO 360 STEP 90
210 DRAW OA,25
220 PRINT <Q3; Y3> I;
225 LET A=A+25
226 IF I=90 THEN LET A=A-6
230 NEXT I
240 RETURN
400 FOR I=-1 TO 1 STEP 1
```

```
410 DRAW O2, 30-I*20
420 PRINT I
430 NEXT I
440 RETURN
600 FOR I=0 TO 360 STEP 4.5
610 LET X=20+I*(2/9)
620 LET Y=32+20*SIN(PI/180*I)
630 DRAW HX, Y
640 NEXT I
650 RETURN
```

## 2. Построение линейных графиков

Поскольку для построения графика требуется задание координат, то их нужно определить прежде всего.

Начало координат может быть помещено в положение (X, Y) с помощью оператора:

```
DRAW OX, Y
```

Вычерчивание осей координат производится с помощью операторов:

```
DRAW X0, 5, 10
DRAW X1, 5, 10
DRAW X2, 5, 10
DRAW X3, 5, 10
```

Числа, следующие после X, указывают на направление осей.

- 0: вычерчивается ось Y из начала координат вверх (+Y)
- 1: вычерчивается ось X из начала координат вправо (+X)
- 2: вычерчивается ось Y из начала координат вниз (-Y)
- 3: вычерчивается ось X из начала координат влево (-X)

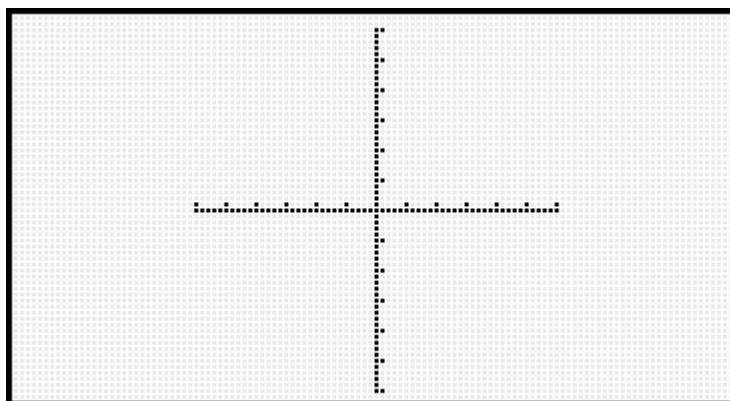
Следующим числовым значением (5) определяется расстояние между метками шкалы (в точках), число 10 определяет количество меток шкалы.

Эти операторы используются в качестве примера в следующей программе

```
5 CLS
10 FOR I=0 TO 3
20 DRAW O60, 32
30 DRAW XI, 5, 6
40 NEXT I
```

При выполнении этой программы вычерчиваются оси координат с началом в точке (60, 32), показанные на Рис. 1.





**Рис. 1**

Линейные графики часто используются при исследовании процессов, протекающих во времени. Рассмотрим программу, с помощью которой строится линейный график средней температуры за месяц.

Предположим, что средняя температура за месяц в определенном городе изменялась в соответствии с Табл. 1. Эти данные вводятся в программу с помощью оператора DATA. Затем, с помощью программы производится построение графика средней температуры за месяц. Считывание данных производится с помощью оператора READ.

**Таблица 1**

Месяц	Температура	Месяц	Температура	Месяц	Температура
Январь	11,5	Май	19,8	Сентябрь	22,3
Февраль	9,8	Июнь	23,4	Октябрь	18,5
Март	13,7	Июль	26,6	Ноябрь	15,9
Апрель	18,3	Август	28,2	Декабрь	14,7

```

10 CLS
20 DRAW O30,50
30 DRAW X9,10,4
35 DRAW O30,50
40 DRAW X1,5,12
50 FOR I=1 TO 3
60 LOCATE 15,I*10
70 PRINT 40-I*10
80 NEXT I
90 FOR I=1 TO 12
100 LET X=30+(I-1)*5
110 READ A
120 LET Y=INT(50-A)
130 IF I=1 THEN 150
140 DRAW DP,Q,X,Y
150 LET P=X: LET Q=Y
160 NEXT I
170 DATA 11.5,9.8,13.7,18.3,19.8,23.4
175 DATA 26.6,28.2,22.3,18.5,15.9,14.7

```

### 3. Построение гистограмм

#### 3.1. Построение гистограмм с помощью символов

Принцип построения гистограммы заключается в том, что на экране дисплея высвечивается рисунок в виде столбиков, длина которых пропорциональна значению N. Поскольку на экране дисплея вмещается только 20 цифр, то для того, чтобы высвечивать линейные величины, превышающие число 20, необходимо использовать масштабирование.

Если вы выполните следующую программу и присвоите N значения от 1 до 20, то на экране высветится линия длиной в N символов. Этот столбик будет непрерывным.

```
10 CLS
20 INPUT N
25 LOCATE 0,32
30 FOR I=1 TO N
40 PRINT "-";
50 NEXT I
60 END
```

#### 3.2. Построение гистограмм с помощью графиков

Приведенную выше программу можно переделать таким образом, чтобы вместо символов использовались точки. Следующая программа является таким примером.

С помощью этой программы по горизонтали может быть высвечено 120 точек, при этом каждая точка по длине в шесть раз короче символа.

```
10 CLS
20 INPUT N
30 FOR I=0 TO N-1
40 DRAW HI,10
50 NEXT I
60 END
```

Получаемые с помощью этой простой программы гистограммы имеют вид прямых линий. Можно написать программу в которой вместо горизонтальных линий будут высвечиваться столбики.

```
10 CLS
20 INPUT N
30 DRAW A0,32,N-1,26
40 END
```

Получаемые с помощью этой простой программы столбики светятся монотонно. Можно написать программу, в которой столбик будет выглядеть в виде определенного узора. Такие узоры можно реализовать путем внесения изменения и добавлений в выше приведенную программу.

1) Горизонтальная штриховка столбика:

```
35 DRAW O0,26
36 DRAW G1,N-1,6,2
```

2) Вертикальная штриховка столбика:

```
35 DRAW O0,26
36 DRAW G2,N-1,6,2
```

### 3.3. Построение круговых диаграмм

Для построения окружности используется команда DRAW/C. Однако, для построения круговой диаграммы необходимо провести линии, определяющие сектора, соответствующие определенным величинам.

Ниже в качестве примера показана программа, которая позволяет разделить окружность на 5 частей.

```
10 CLS
20 DRAW C50,32,30
30 LET S=0: LET T=0
40 FOR I=1 TO 5: READ A: LET T=T+A: NEXT I
50 RESTORE: LET A=0
60 FOR I=1 TO 5: LET A1=A
70 READ A: LET S=S+A
80 LET X=50+INT(30*COS(2*PI*S/T))
90 LET Y=32-INT(30*SIN(2*PI*S/T))
100 DRAW D50,32,X,Y
110 LET A2=(A1+A)/2: GOSUB 200
115 NEXT I
120 END
130 DATA 100,200,300,400,500
200 LET A3=COS(2*PI*S/(T+A2))
205 LET X=84
206 IF A3<0 THEN LET X=5
210 LET Y=25-INT(22*SIN(2*PI*S/(T+A2)))
220 DRAW OX,Y
230 PRINT I*100
240 RETURN
```

### 3.4. Два примера построения гистограмм

Гистограммы часто используются для наглядного представления соотношения некоторых данных.

Однако необходимо, чтобы наряду с сопоставительной наглядностью столбики гистограммы правильно отражали количественно отображаемую величину. Поэтому здесь важно правильно выбрать масштаб, чтобы гистограмма хорошо уложилась на экране.

Существует много путей построения графиков в зависимости от их применения. В основном, используются такие гистограммы, в которых одним столбиком выражается количественное значение одной величины (например, когда иллюстрируется количество различных изделий). Бывают гистограммы, в которых одним столбиком иллюстрируется как общее количество, так и какие-нибудь детали (например, соотношение общего количества и конкретного изделия).

Напишем программы, с помощью которых с использованием следующего примера производится построение таких гистограмм двух типов.

Пример данных: количество автомобилей двух типов А и В, изготавливаемых на одном производстве, приведено в Табл. 2.

Таблица 2

	1980	1981	1982
<b>Автомобили А</b>	48200	57200	67200
<b>Автомобили В</b>	39200	31100	27500

Программа построения гистограммы, в которой одним столбиком отображается количественное значение только одной величины.

Столбик гистограммы, отображающий производство автомобилей типа А – светлый, столбик, отображающий производство автомобилей типа В – темный.

```
10 CLS
20 FOR I=0 TO 2
30 LOCATE 12,10+I*16
40 PRINT 80+I
50 FOR J=1 TO 2
60 READ A
70 LET Y=I*16+(J-1)*8+10
80 DRAW A30,Y+4,30+INT(A/1000),Y
90 IF J=1 THEN GOTO 120
100 DRAW O30,Y
110 DRAW G1,INT(A/1000),4
120 NEXT J: NEXT I
130 DATA 48200,39260,57200,31100,67200,27509
140 END
```

Программа построения гистограммы, в которой одним столбиком одновременно отображается общее количество и детали.


```
10 CLS
20 FOR I=0 TO 2
30 LOCATE 12,10+I*16
40 PRINT 80+I
50 READ A,B
60 LET Y=I*16+10
70 DRAW A30,Y+8,30+INT(A/1000),Y
75 DRAW O30,Y
80 DRAW A30,Y+8,30+INT(B/1000),Y
85 DRAW O30,Y
90 DRAW G1,INT(B/1000),8
100 NEXT I
110 DATA 48200,39200,57200,31100,67200,27500
120 END
```

## 4. Изображения движущихся объектов

### 4.1. Создание движущихся объектов

Если вы выполните следующую программу, метка '\*' будет передвигаться слева направо и справа налево.

```
10 CLS
20 FOR I=0 TO 14
30 LOCATE I*6,8
40 PRINT <S2,2>"*"
50 NEXT I
60 FOR I=14 TO 0 STEP -1
70 LOCATE I*6,8
80 PRINT <S2,2>"* "
90 NEXT I
100 GOTO 20
```



Как показано в программе, при изменении управляющей переменной I в строке 20 от 0 до 17, с помощью оператора строки 30 положение курсора изменяется от (0, 8) до (114, 8). Метка '\*' высвечивается на экране путем вычерчивания " \*", в то время, как метка '\*', выветившаяся перед этим, стирается пробелом.

При повторении этой процедуры создается впечатление, что она перемещается слева направо. Операторы, расположенные в строках 60-90, аналогичным образом перемещает метку справа налево.

Вертикальное положение теоретически выполняется аналогичным образом. Однако здесь не может происходить одновременное высвечивание и стирание, и программа приобретает следующий вид:

```
10 CLS
20 FOR I=0 TO 7
30 LOCATE 12,I*8: PRINT "*";
40 IF I=0 THEN 60
50 LOCATE 12,(I-1)*8: PRINT " ";
60 NEXT I
70 FOR I=7 TO 0 STEP -1
80 LOCATE 12,I*8: PRINT "*"
90 IF I=7 THEN 110
100 LOCATE 12,(I+1)*8: PRINT " ";
110 NEXT I
120 GOTO 20
```

#### 4.2. Изменение скорости движения

Предыдущая программа обеспечивает довольно быстрое перемещение метки '\*'. Скорость ее движения управляется оператором FOR/NEXT.

Выполним программу, полученную путем добавления к предыдущей программе следующей строки:

```
45 FOR J=1 TO 50: NEXT J
```

При использовании этого оператора скорость перемещения слева направо становится несколько меньше. Скорость увеличивается путем уменьшения конечного значения оператора FOR/NEXT, а уменьшается путем увеличения этого значения. Управление скоростью производится путем добавления этого оператора в необходимую часть программы.

### 4.3. Перемещение точки по прямой

Для перемещения точки по прямой необходимо выполнить следующую программу:

10 CLS		
20 FOR I=0 TO 119		
30 DRAW HI, 32		
40 IF I=0 THEN 60		
50 DRAW EI-1, 32, I-1, 32		
60 NEXT I		
70 FOR I=119 TO 0 STEP -1		
80 DRAW HI, 32		
90 IF I=119 THEN 110		
100 DRAW EI+1, 32, I+1, 32		
110 NEXT I		
120 CLS		
130 FOR I=0 TO 63		
140 DRAW H60, I		
150 IF I=0 THEN 170		
160 DRAW E60, I-1, 60, I-1		
170 NEXT I		
190 FOR I=63 TO 0 STEP -1		
200 DRAW H60, I		
210 IF I=63 THEN 230		
220 DRAW E60, I+1, 60, I+1		
230 NEXT I		
240 GOTO 10		

движение слева направо

движение справа налево

движение сверху вниз

движение снизу вверх

### 4.4. Перемещение точки по кривой траектории

Следующая программа обеспечивает перемещение точки по траектории косинусоиды вперед и назад.

```
10 CLS
20 LET P=0: LET Q=0
30 FOR I=0 TO 360 STEP 4.5
40 LET X=20+I*(2/9)
50 LET Y=32+20*COS(PI/180*I)
60 DRAW HX, Y
70 DRAW EP, Q, P, Q
80 LET P=X: LET Q=Y
90 NEXT I
100 FOR I=360 TO 0 STEP -4.5
110 LET X=20+I*(2/9)
120 LET Y=32+20*COS(PI/180*I)
130 DRAW HX, Y
140 DRAW EP, Q, P, Q
150 LET P=X: LET Q=Y
160 NEXT I
170 GOTO 30
```

Таблицы кодов символов

Таблица 1

Специальные символы

Дес. код	Символ	Дес. код	Символ	Дес. код	Символ	Дес. код	Символ
0		16		32	<i>пробел</i>	48	0
1		17		33	!	49	1
2		18		34	"	50	2
3		19		35	#	51	3
4		20		36	\$	52	4
5		21		37	%	53	5
6		22		38	&	54	6
7		23		39	'	55	7
8		24		40	(	56	8
9		25		41	)	57	9
10	<i>ПС</i>	26		42	*	58	:
11		27		43	+	59	;
12		28		44	,	60	<
13	<i>ВК</i>	29		45	-	61	=
14	<i>РУС</i>	30		46	.	62	>
15	<i>ЛАТ</i>	31		47	/	63	?



Таблица 2

Символы латинского регистра

Дес. код	Символ	Дес. код	Символ	Дес. код	Символ	Дес. код	Символ
64	@	80	P	96	‘	112	p
65	A	81	Q	97	a	113	q
66	B	82	R	98	b	114	r
67	C	83	S	99	c	115	s
68	D	84	T	100	d	116	t
69	E	85	U	101	e	117	u
70	F	86	V	102	f	118	v
71	G	87	W	103	g	119	w
72	H	88	X	104	h	120	x
73	I	89	Y	105	i	121	y
74	J	90	Z	106	j	122	z
75	K	91	[	107	k	123	{
76	L	92	\	108	l	124	
77	M	93	]	109	m	125	}
78	N	94	¬	110	n	126	–
79	O	95	–	111	o	127	ЗБ

Таблица 3

Символы русского языка

Дес. код	Символ	Дес. код	Символ	Дес. код	Символ	Дес. код	Символ
64	ю	80	п	96	Ю	112	П
65	а	81	я	97	А	113	Я
66	б	82	р	98	Б	114	Р
67	ц	83	с	99	Ц	115	С
68	д	84	т	100	Д	116	Т
69	е	85	у	101	Е	117	У
70	ф	86	ж	102	Ф	118	Ж
71	г	87	в	103	Г	119	В
72	х	88	ь	104	Х	120	Ь
73	и	89	ы	105	И	121	Ы
74	й	90	з	106	Й	122	З
75	к	91	ш	107	К	123	Ш
76	л	92	э	108	Л	124	Э
77	м	93	щ	109	М	125	Щ
78	н	94	ч	110	Н	126	Ч
79	о	95	ъ	111	О	127	ЗБ

## Рекомендации по работе с СМП

**ВНИМАНИЕ!!!** Для обеспечения достоверности хранимой на СМП информации, перед записью программы пользователя из ОЗУ микрокомпьютера в СМП после ее отладки (отладка предполагает по меньшей мере однократное выполнение программы посредством команды RUN) необходимо придерживаться следующих правил:

1. Обратить внимание, что любая программа должна заканчиваться оператором END, имеющим следующий формат:  
`<номер строки> <пробел> END.`
2. Ввести в начале программы первой строкой оператор STOP.  
 Запустить программу командой RUN. После ее выполнения удалить введенную строку с оператором STOP и, используя команду SAVE, записать программу в СМП.
3. Командой FILES вызвать справочник файлов, расположенных на СМП, определив объем, занимаемый программой (файлом) в блоках.
4. Используя команду KILL удалить на СМП записанную программу.
5. Изменить формат последней строки программы, записав следующим образом:  
`<номер строки> END,`  
 удалив пробел между номером строки и оператором END.
6. Выполнить действия, аналогичные п. 3, записав в СМП новую программу.
7. Используя команду FILES определить объем занимаемой новой программой памяти в блоках.
8. Если полученная программа будет занимать в СМП такой же объем памяти, как и предыдущая, то данный вариант программы оставляется в СМП для последующего использования ее в качестве рабочей.
9. Если полученная программа занимает в СМП объем памяти меньше, чем предыдущая, то данную программу необходимо удалить из СМП, восстановить последнюю строку программы аналогично предыдущей с сохранением пробела между номером строки и оператором END. Данный вариант программы записать в СМП и использовать в качестве рабочей.
10. После удаления какого-либо файла из СМП необходимо следить, чтобы на СМП была только одна свободная область, расположенная после последнего файла. Если хотя бы одна свободная область расположена между двумя файлами, ее необходимо "вытеснить" за последний файл. Для этого необходимо перекопировать каждый файл, начиная с первого файла, расположенного после первой свободной области, с помощью последовательности команд: LOAD, KILL, SAVE.

Пример:

### Справочник SM0

A.BAS	4	4	
<свободно>	3	7	← первая свободная область
B.BAS	5	12	
<свободно>	4	16	← вторая свободная область

Чтобы "вытеснить" первую свободную область за последний файл (B.BAS)

необходимо выполнить последовательно следующие команды:

LOAD "B"                      KILL "B.BAS"                      SAVE "B"

### Справочник SM0

A.BAS	4	4	
B.BAS	5	9	
<свободно>	7	16	← свободная область СМП

после "вытеснения" свободной области