Руководство по эксплуатации учебного компьютера

SONIC REC-9388

ВВЕДЕНИЕ

Русско-английский учебный компьютер REC-9388 объединяет в себе учебный компьютер и игровую приставку. Центральный процессор этого учебного компьютера основан на ЦПУ Motorola 6502. Его адресное пространство 64 кбайт. Учебный компьютер REC-9388 имеет широкие возможности:

- язык программирования BASIC, который может использоваться не только для обучения начинающих изучать языки программирования, но и для сложных математических расчетов;
- язык программирования LOGO;
- язык программирования F-BASIC, позволяющий легко составлять игровые программы с элементами мультипликации и музыкальным сопровождением;
- графический редактор, который позволяет из широкого набора картинок, подобно мозаике, составлять сложные рисунки;
- музыкальный редактор, превращающий компьютер в многоголосый музыкальный инструмент;
- различные обучающие программы;
- позволяет сохранять написанные программы на магнитофоне;
- к REC-9388 можно подсоединить принтер для получения распечаток;
- имеет мощную систему аудио-видео обучения, использующую специально разработанные магнитофонные кассеты с курсом обучения по различным предметам средней школы, а также с курсами для начинающих изучать иностранные языки.

Также компьютер REC-9388 имеет отличные характеристики, такие как многоцветная графика с хорошим разрешением, 5 звуковых и музыкальных каналов, стандартная клавиатура IBM, три разъема расширения. Такой компьютер - наилучшее средство для детей при обучении их работе на компьютере и для учителей начальной и средней школы при применении ими компьютеров в учебных программах.

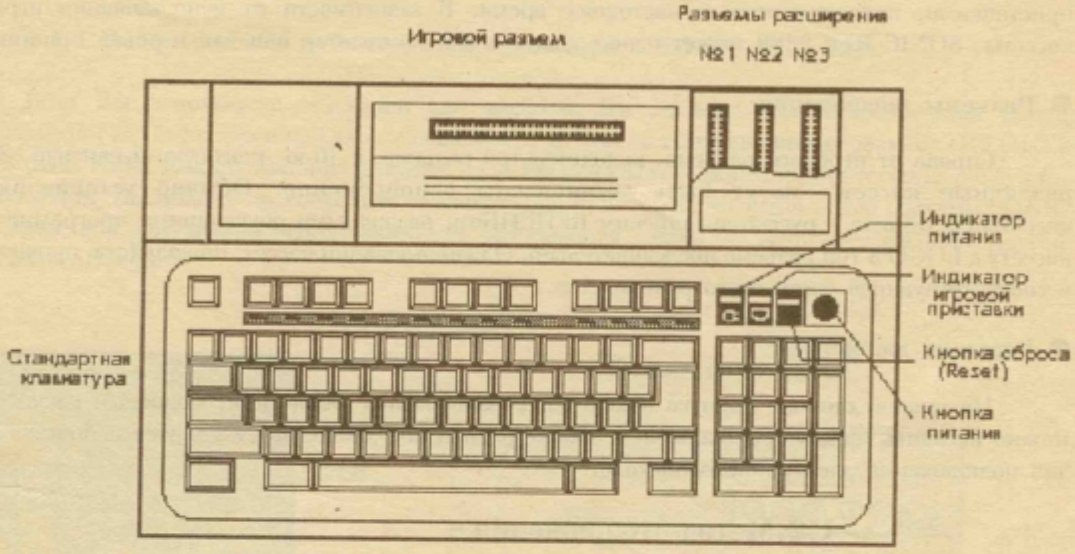
SONIC REC-9388 можно использовать как игровую приставку, которая полностью совместимая с другими игровыми приставками, продающимися в настоящее время. Игровые кассеты (картриджы) могут использоваться в учебном компьютере REC-9388 независимо от их емкости. Компьютер SONIC в качестве игровой приставки принесет Вам и Вашим детям огромное удовольствие.

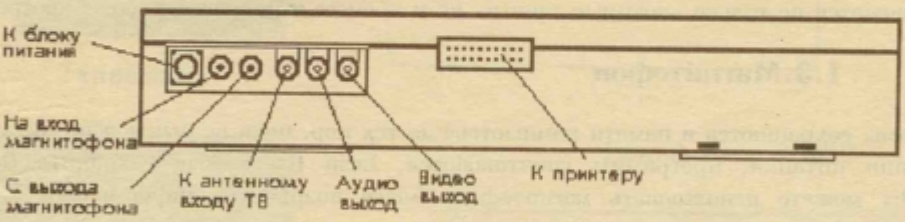
Содержание.

Часть 1. Как использовать SONIC REC-9388 - русско-английский учебный компьютер

Глава 1. Краткое введение о SONIC REC-9388 русско-английском учебном компьютере

SONIC REC-9388 - русско-английский учебный компьютер, который может использоваться как компьютер и как игровая приставка. Его главные составляющие: главное устройство и клавиатура. Главное устройство установлено в корпус с клавиатурой, как показано на рис.1.





1.1.Главное устройство

❶ Главное управляющее устройство

ЦПУ - *центральное процессорное устройство, которое использует 8-битную шину данных и 16-битную адресную шину, его адресное пространство 64 кБайт.*
ВУ - *видеоустройство, которое преобразует цифровые сигналы в видеосигнал.*
ASIC - *три специальные микросхемы на главном устройстве, которые используются для некоторых специальных функций.*

❷ Функциональные разъемы для кассет

На главном устройстве всего находится 4 разъема для кассет (один на 60 контактов и три на 36 контактов каждый, которые именуются №1, №2, №3 соответственно).Разъем на 60 контактов имеет преимущество перед другими разъемами, если Вы вставили одновременно функциональные и игровые кассеты. Главное устройство позволяет запускать программы с любого из трех разъемов (конечно, если там были вставлены кассеты). Если кассета не была вставлена в разъем с 60 контактами,то при включении питания начнет работать системная кассета в разъеме №1. При этом главное устройство входит в режим компьютера и Вы можете сделать выбор из показанного меню. Чтобы использовать кассету в разъеме №2 или №3 нажмите цифровую клавишу 2 или 3 соответственно.

❸ Клавиатура

Используется стандартная клавиатура с 88 клавишами. Главное устройство воспринимает соответствующие инструкции исходя из типа клавиши на клавиатуре. Расположение этих клавиш точно такое же, как и у компьютеров IBM PC. Все клавиши имеют расположение, как у стандартной пишущей машинки, и это позволяет использовать данную клавиатуру для практики работы на пишущей машинке. Все клавиши на клавиатуре поддерживают коды ASCII.

❹ Разъем для игровых кассет

В средней части главного устройства есть разъем с 60-ю контактами, который может быть использован для установки игровых кассет. Этот разъем полностью совместим с другими игровыми приставками, продающимися в настоящее время. В зависимости от использования игровой кассеты, SONIC REC-9388 может использоваться как компьютер или как игровая приставка.

❺ Разъемы расширения

Справа от игрового разъема, находятся три разъема, с 36-ю контактами каждый. Здесь различные кассеты могут быть установлены одновременно. Обычно устанавливают кассеты:системную с русско-английским БЕЙСИКом, различными обучающими программами и кассету с LOGO в три разъема последовательно. О использовании кассет, пожалуйста, прочитайте в соответствующей части этого руководства.

❻ Разъемы джойстика

На правой стороне корпуса находятся 2 разъема под джойстики, в каждый из которых можно вставить "вилку" от джойстика. Также разъем под джойстик №2 может использоваться для подключения электронного пистолета.

1.2.Метод отображения

Любой телевизор или монитор может быть использован как дисплей для SONIC REC-9388. На нем отображаются не только вводимые данные, но и процесс и результат работы программ.

1.3.Магнитофон

Программы сохраняются в памяти компьютера до тех пор, пока не выключено питание. При отключении питания, программы уничтожаются. Если Вы хотите сохранить Ваши программы, Вы можете использовать магнитофон. Более подробную информацию можно получить из главы 4.

1.4.Принтер

Интерфейс SONIC REC-9388 позволяет подсоединять принтер с 9 или 24 иголками. Более подробную информацию можно получить в соответствующей части этого описания.

1.5.AV конвектор (Аудио-видео конвектор)

Этот конвектор используется вместе с магнитофоном и кассетами аудио-видео обучения. С ними SONIC REC-9388 может стать прекрасной обучающей аудио-видео системой и незаменимым помощником учителю.

1.6.Джойстики

К SONIC REC-9388 прилагается пара джойстиков, которые используются в режиме игровой приставки.

1.7.Радиокабель

Радиокабель используется для подсоединения компьютера с телевизором или монитором.

1.8.Питание

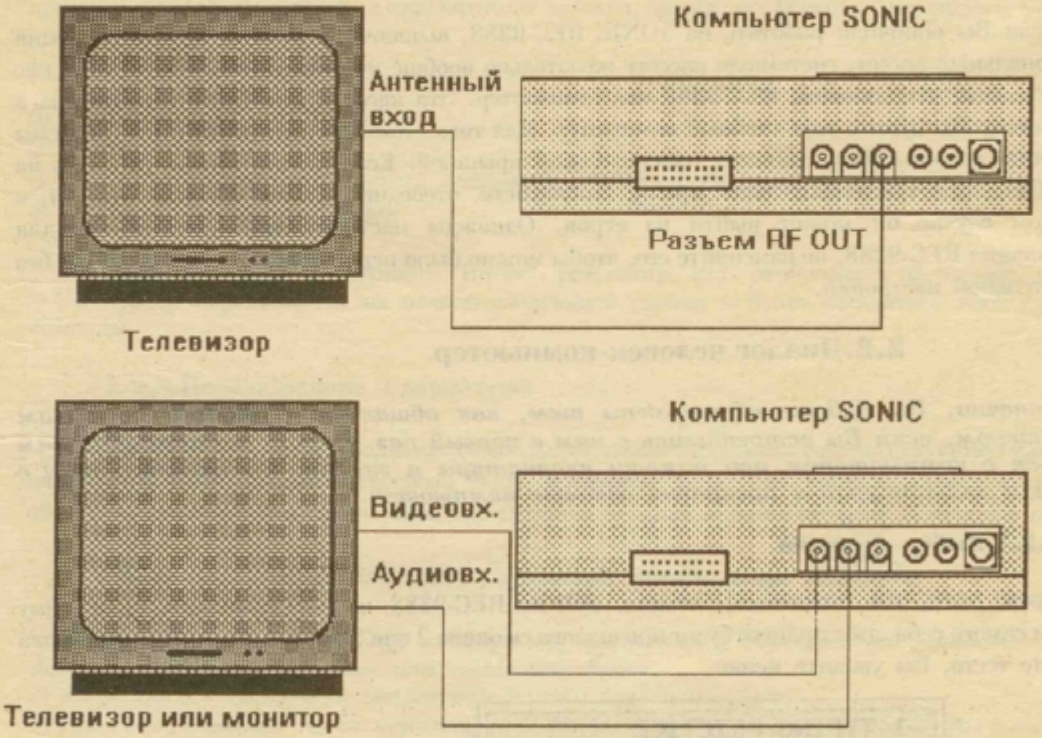
К SONIC REC-9388 прилагается блок питания. Входное напряжение блока питания 220В 50 Гц и выходное 9 В . Этот блок питания не может использоваться в других игровых приставках.

Глава 2. Инструкции по эксплуатации

2.1.Подсоединения и включение питания

2.1.1.Подсоединение телевизора или монитора к SONIC REC-9388

Если Вы используете телевизор как дисплей, Вы должны подсоединить один конец радиокабеля (75 Ом) в антенный вход телевизора, а другой - к телевизионному разъему (RF OUT) SONIC REC-9388. Как показано на рисунке, если Вы используете монитор или телевизор с аудио-видео входом, Вы можете подсоединить телевизор или монитор через аудио-видео разъемы SONIC REC-9388, для чего нужно использовать два радиокабеля (один для звукового сигнала, другой для видеосигнала).



2.1.2.Источник питания в SONIC REC-9388

Вставьте шнур блока питания в разъем (DC IN) компьютера, затем вставьте блок питания в розетку сети. Причем напряжение сети должно быть 220 В 50 Гц, иначе компьютер может выйти из строя.

2.1.3.Подсоединение джойстиков

Джойстики могут быть подсоединены в разъемы для джойстиков, если Вы хотите поиграть в игры.

2.1.4.Использование функциональных кассет

Если Вы желаете использовать SONIC REC-9388 как компьютер, сначала Вы должны вставить системную кассету в разъем расширения №1, затем вставьте две другие кассеты в разъемы расширения №2 и №3 соответственно (эти два разъема могут не использоваться) или, если Вы желаете использовать SONIC REC-9388 как игровую приставку, Вы должны просто вставить игровую кассету в игровой разъем.

2.1.5.Включение питания и выбор канала

После того, как Вы произвели все предыдущие операции, Вы можете включить питание. Помните, что вставлять и вынимать кассеты из разъемов при включенном питании нельзя, так как это может привести к выходу из строя кассет и компьютера.

Если Вы используете телевизор как дисплей, для SONIC REC-9388 нужно выбрать рабочий канал (обычно не канал ТВ передач) с помощью соответствующих клавиш телевизора. При этом нужно достичь правильной картинки без шумов и искажений.

2.1.6.Замена функциональных кассет

Выключите питание, затем отсоедините или вставьте кассету. Далее Вы можете включить питание.

2.1.7.Операции после окончания работы на SONIC REC-9388

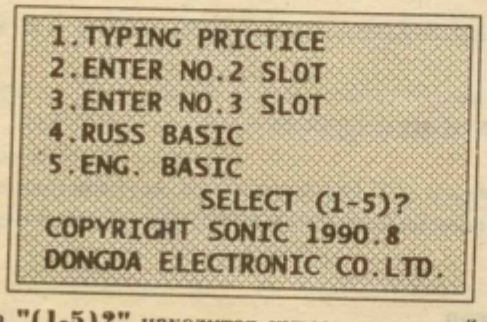
Если Вы окончили работать на SONIC REC-9388, выключите питание до отсоединения функциональных кассет, системную кассету желательно вообще не отсоединять, так как без нее Вы не сможете использовать REC-9388 как компьютер. Эта кассета уже будет установлена в разъем, когда Вы купите этот учебный компьютер. Для того, чтобы не поступала пыль, разъемы расширения должны быть закрыты пластмассовой крышкой. Если Вы не будете работать на SONIC REC-9388 продолжительное время, пожалуйста, отсоедините блок питания от сети, в противном случае он может выйти из строя. Однажды настроив канал телевизора для использования REC-9388, не изменяйте его, чтобы можно было использовать его в другой раз без дополнительной настройки.

2.2.Диалог человек-компьютер

Конечно, Вы будете обеспокоены тем, как общаться с Вашим домашним компьютером, если Вы встретились с ним в первый раз. Сейчас мы поможем Вам общаться с компьютером при помощи клавиатуры и экрана. Вы ознакомитесь с экраном и клавиатурой и прекрасно освоите компьютер.

2.2.1.Выбор функции

После того, как включено питание, SONIC REC-9388 начнет выполнять программу проверки самого себя, этот процесс будет продолжаться около 2 или 3 секунд. Терпеливо подождав окончание теста, Вы увидите меню:



Следом за **"(1-5)?"** находится курсор, используйте цифровые клавиши от **1** до **5** для выбора соответствующей функции, которая Вам нужна. Когда Вы введете **"1"**, экран очистится и курсор переместится в левый верхний угол экрана. Это значит, что компьютер в режиме ввода текстов. Переключая латинские и русские буквы функциональной клавишей **F10**, Вы можете практиковаться машинописи на компьютере.

Когда Вы нажали цифровую клавишу **4** или **5**, экран очистится и подсказка **">"** отобразится в верхнем левом утлу экрана. Подсказка говорит, что Вы находитесь в системе программирования Бейсик.

В бейсике Вы можете вводить и русские буквы, после нажатия функциональной клавиши **F10**, обратный эффект можно получить, нажав **F10** еще раз.

Если Вы желаете запустить программы из функциональных кассет, установите их в разъемы расширения №2 или №3 и нажмите цифровую клавишу **2** или **3**. Система начнет выполнять программы из функциональных кассет в разъемах расширения **2** или **3**. Сейчас Вы хотите остановить выполнение программ из функциональной кассеты в разъеме расширения 2 или 3 и возратиться в экран выбора функций, для этого нажмите три клавиши **(Ctrl+Alt+Del)** одновременно. Вы также можете для этого выключить и включить питание, но это не рекомендуется, так как может сократить ресурс выключателя питания.

Если Вы хотите запустить программы с игровой кассеты, то выключите питание и затем вставьте игровую кассету в игровой разъем, и после этого включите питание. Система начнет выполнять игровые программы (игровой разъем имеет наибольший уровень приоритета). Подобно этому, когда Вы хотите выполнять программы с функциональных кассет расширения, выключите питание,отсоедините игровую кассету от игрового разъема, включите питание, и система готова для выполнения программ с функциональных кассет. Помните, что не нужно отсоединять системную кассету, пусть она постоянно будет в разъеме.

2.2.2.Ввод команд в SONIC REC-9388

Прочитав описание к различным функциональным кассетам, мы узнаем, что SONIC REC-9388 имеет различные возможности в зависимости от определенных программных продуктов. Для примера, мы введем состояние стандартного бейсика, нажав "5". И сейчас напечатаем следующее:

```
> PRINT  "123"
```

① *Введите команду PRINT*
② *Нажмите на пробел*
③ *Введите в кавычках текст, который нужно отобразить (в нашем примере "123")*
④ *Нажмите клавишу* ***Enter***

Когда Вы нажмете клавишу **"Enter"**, результат "123" отобразится на экране, подсказка ">" и курсор передвинутся на начало следующей строки и будет ожидаться ввод следующей команды.

2.2.3.Использование клавиатуры

Вы можете увидеть вводимые символы на экране, нажимая соответствующие клавиши на клавиатуре. Сейчас напечатаем слово "book" и нажмем "Enter". Вы увидите на экране сообщение об ошибке, потому что слово "book" не команда бейсика и компьютер не может его опознать.

Объясним, как использовать следующие специальные клавиши:

1) **Shift** - *Нажав клавишу Shift и любую другую клавишу одновременно, можно вводить символы в верхнем регистре или маленькие буквы.*
2) **Caps Lock** - *Включает-выключает режим заглавных букв.*
3) **Esc** - *Эта клавиша часто используется для выхода из работы, которую Вы выполняете, и возращению к более ранней.*
4) **Ctrl** - *Эта клавиша работает сходно клавише Shift и изменяет одну какую-нибудь клавишу.*
5) **Tab** - *Клавиша табуляции.*
6) **Enter** - *Клавиша возврата работает также, как клавиша возврата каретки, то есть передвигает курсор в начало следующей строки.*
7) **Backspace** - *Эта клавиша удаляет предыдущий символ и передвигает курсор на одну позицию назад.*
8) **←→↑↓** - *Эти четыре клавиши управления курсором используются, чтобы легко и просто редактировать ваши программы.*
9) **Num Lock** - *Работает сходно с клавишей Shift для цифровой клавиатуры.*
10) **Ctrl+Alt+Del** - *В некоторых случаях одновременное нажатие этих трех клавиш приведет к тому, что система начнет выполнять тестирующую программу.*

11) **Reset** - *Клавиша сброса.*
12) **F1 - F12** - *Двенадцать функциональных клавиш используются как многоцелевые клавиши, каждой из которых определена какая-либо задача. Детальную информацию можно получить из таблицы Б.*
13) **Home** - *Клавиша очищает экран и передвигает курсор в его начало.*
14) **Ins** - *Клавиша вставки будет объяснена на примере.*
15) **Space** - *Клавиша пробела расположена в нижней части клавиатуры. Она будет печатать пробелы в текущей позиции курсора, если в текущей позиции символ, то он будет заменен на пробел.*
16) **Клавиши арифметических операций:** *Используя их вместе с оператором PRINT, Вы можете выполнять арифметические операции непосредственно на экране.*

Эти клавиши:

+ : плюс	- : минус
* : умножить	/ : делить
^ : возведение в степень	= : равно

Примеры:

❶ `> PRINT 4+5-2`
`7` - результат вычислений
❷ `> PRINT (4*5+1)/6-7+23`
`19.5` - результат вычислений

2.2.4.Пример программы

Пример: Наберем следующую программу:

```
>10 PRINT "ПРИМЕР ПРОГРАММЫ"
>20 PRINT "      ДЛЯ"
>30 PRINT "   # SONIC #"
>40 END
>RUN
ПРИМЕР ПРОГРАММЫ
    ДЛЯ
 # SONIC #
```

Как видно из примера, компьютер будет выполнять программу после того, как Вы ее напечатали и ввели команду RUN. При этом результат выполнения будет отображен на экране.

Если при выполнении программы в ней обнаружены ошибки, то компьютер выдаст о них сообщения. Для примера, если Вы ввели программу:

```
>10 PRINY "ПРИМЕР ПРОГРАММЫ"
>20 PRINT "      ДЛЯ"
>30 PRINT "   # SONIC #"
>40 END
>RUN
SYNTAX ERROR IN 10
```

"SYNTAX ERROR IN 10" говорит о том, что в операторе на строке десять обнаружена синтаксическая ошибка. Чтобы исправить, Вы должны заменить неправильную букву "Y" на "T" в операторе PRINT.

Сейчас мы рассмотрим два метода исправления:

Метод 1: Заново ввести неправильный оператор:

① Введите команду LIST, компьютер покажет на экране текст программы.

```
> LIST
 10 PRINY "ПРИМЕР ПРОГРАММЫ"
20 PRINT "      ДЛЯ"
30 PRINT "   # SONIC #"
40 END
>
```

② Правильно введите оператор.
>10 PRINT "KEYBOARD TEST"
③ Введите команду RUN для выполнения программы.
>RUN

Метод 2: Использовать для исправления ошибки команду EDIT.

>EDIT 10 - Это значит, что Вы хотите исправить оператор на строке 10.
Сейчас на экране отобразится:
10 PRINY "ПРИМЕР ПРОГРАММЫ" и курсор в первой позиции строки.

Используя клавишу движения курсора вправо, поставьте курсор на букву "Y", затем напечатайте "T" и нажмите клавишу **Enter**. Вы можете посмотреть исправленную программу командой LIST. Теперь запустите программу командой RUN и Вы получите правильный результат!

Глава 3. Как писать программы

Существует много компьютерных языков программирования. Каждый из них имеет свой синтаксис. Любая программа составляется исходя из синтаксических правил написания программ, так как возможности компьютера ограничены. Среди этих языков **бейсик** - язык программирования высокого уровня. Название BASIC есть сокращение от слов "Beginners All purpose Symbolic Instruction Code", что в переводе означает "Многоцелевой язык символических команд для начинающих". Он специально предназначен для тех, кто начал изучать языки программирования. Так называемые бейсик-программы есть некоторое количество определенных шагов, состоящих из операторов **бейсика**, которые производят вычисления или решают задачи управления. Рассмотрим следующий пример:

Пример: Вы хотите найти два корня квадратного уравнения: $AX^2+BX+C=0$

Из алгебры мы знаем: $X_{1,2}= \frac{-B \pm \sqrt{D}}{2A}$, где $D = B^2 - 4AC$

D - дискриминант этого уравнения.

Возможны три случая:

D > 0 - уравнение имеет два различных действительных корня
D = 0 - уравнение имеет два равных корня
D < 0 - не имеет действительных корней

3.1. Алгоритм

Для решения вышеприведенной задачи, вначале составим алгоритм:

1. Вводим коэффициенты A,B,C с клавиатуры
2. Вычисляем дискриминант $D=B^2-4AC$
3. Если D>0, то два различных действительных корня, которые вычисляются по формуле: $X_{1,2}=(-B\pm\sqrt{D})/2A$, далее выполняем п.6
4. Если D=0, то один действительный корень $X_1=X_2=-B/2A$, далее выполняем п.6
5. Если D<0, то печатаем, что нет действительных корней и переходим на п.7
6. Печатаем на экране значения корней уравнения X_1 и X_2
7. Конец алгоритма

3.2. Написание программы в бейсике

```
10 INPUT A,B,C                          вводим A,B,C
20 D=B*B-4*A*C                          вычисляем D
30 IF D>0 THEN GOTO 80                      если D больше нуля, переходим на 80
40 IF D=0 THEN GOTO 110                     если D равен нулю, переходим на 110
60 PRINT "НЕТ ДЕЙСТВИТЕЛЬНЫХ КОРНЕЙ"   печатаем, что нет действительных корней
70 GOTO 140                             переходим на 140
80 X1=(-B+SQR(D))/(2*A)                 вычисляем два действительных корня
90 X2=(-B-SQR(D))/(2*A)
95 PRINT "ДВА РАЗЛИЧНЫХ КОРНЯ"          печатаем, что два различных корня
```

```
100 GOTO 130
110 X1=-B/(2*A)                    находим два равных корня
120 X2=X1
125 PRINT "ДВА РАВНЫХ КОРНЯ"
130 PRINT "X1=";X1,"X2=";X2        печатаем значения корней
140 END                            конец программы
```

3.3. Выполнение программы

```
>RUN              - Введем команду RUN
? 1,-2,1          - Введем три коэффициента
ДВА РАВНЫХ КОРНЯ
X1=1  X2=1        - Результат вычислений отобразится на экране
```

3.4. Исправление программы

Если есть какие-нибудь ошибки в операторах программы, компьютер остановит выполнение программы и укажет номер строки, где находится неправильный оператор. Вы можете использовать команду LIST, чтобы проверить операторы, и затем исправить ошибки, используя один из методов, приведенных в предыдущей главе.

Чтобы получить более подробную информацию о стандартном бейсике, пожалуйста, прочитайте часть 2 этого описания.

Глава 4. Сохранение программ на магнитной ленте

Учебный компьютер SONIC REC-9388 имеет разъем для подключения магнитофона. С этого разъема обычный кассетный магнитофон может использоваться как внешняя память. Информация, записанная в памяти компьютера, может быть сохранена на магнитофонной кассете и затем может быть загружена обратно в память.

Чтобы подключить магнитофон, проделайте следующие шаги:

❶ Приготовьте два магнитофонных шнура. Один из них подсоедините к разъему TAPE IN компьютера и к линейному выходу или выходу на наушники магнитофона.
❷ Используйте другой шнур для подключения к разъему TAPE OUT компьютера и к линейному входу или микрофонному входу магнитофона.
❸ Установите регуляторы громкости и тембра в среднее положение.
❹ Рекомендуется пометить шнуры для входа и выхода, для избежания путаницы.
❺ Вставьте сетевой шнур магнитофона в сеть.
❻ Вставьте кассету в магнитофон. Для записи данных из компьютера предпочтительно использовать малошумящие кассеты.

Сейчас напишем программу и посмотрим, как записать ее на магнитофон:

```
>NEW
>AUTO
>10 PRINT"ПРОВЕРКА МАГНИТОФОНА"
>20 FOR I=1 TO 10
>30 PRINT I
>40 NEXT
>50 END
>60
>SAVE"TEST"
```

Команда SAVE показывает компьютеру, что нужно записать программу на магнитную ленту. Имя программы требуется, чтобы можно было вызвать программу в следующий раз. Имя состоит из букв, цифр и пробелов. Имя должно начинаться на букву и его длина не должна превышать 7 символов. Вы должны знать, что имя определяет загрузку программы в последующем.

Процесс записи следующий:

❶ Наберите **SAVE"TEST"**
❷ Нажмите на магнитофоне клавиши **Запись** и **Воспроизведение** одновременно.

❸ Нажмите клавишу **Enter**, затем компьютер начнет передавать программу на магнитофон.
❹ По окончании передачи нажмите клавишу **Стоп** магнитофона.
❺ Вы не должны беспокоиться, что допустили ошибку, просто подождите, пока не появится подсказка ">".

Для загрузки программы с магнитофона проделайте следующие шаги:

❶ Используя клавиши перемотки магнитофона, найдите начало программы. Отсоедините шнур от линейного входа магнитофона, затем нажмите клавишу **Воспроизведение** и прослушайте сигнал. Если Вы слышите непрерывный сигнал, то перемотайте ленту в его начало.
❷ Введите **LOAD"TEST"**.
❸ Нажмите клавишу **Воспроизведение** магнитофона, и компьютер будет искать на магнитной ленте программу с именем **"TEST"**.
❹ Программа будет полностью загружена в память компьютера, когда на экране появится подсказка ">". Если Вы не указали имя программы, то будет загружена первая встретившаяся программа.
☹ Если компьютер не захватывает сигнал с ленты, то подрегулируйте громкость и тембр.

В процессе загрузки программы могут возникнуть две причины возникновения ошибок:

Первая: Недостаточный размер памяти.

Сообщение TP ERROR появится на экране до того, как слышится звук. Это значит, что программа, которая загружается, слишком большая, или в памяти уже находятся другие программы.

Вторая: Ошибки, возникающие в процессе загрузки программы.

Сообщение TP ERROR появится на экране после звукового сигнала.

Если вы желаете соединить две программы, то старший номер строки первой программы должен быть меньше, чем младший номер второй программы. Пример:

```
>LOAD"PROG1"
>LIST
10 PRINT "ABC"
20 PRINT "DEF"
30 PRINT "GHI"
>LOAD"PROG2"
>LIST
10 PRINT "ABC"   }
20 PRINT "DEF"   }      Программа №1
30 PRINT "GHI"   }
40 PRINT "JKL"   }
50 PRINT "MNO"   }      Программа №2
60 PRINT "PQR"   }
```

Глава 5. Как использовать кассету с LOGO

5.1.О языке LOGO

LOGO - название языка программирования, который позволяет реализовать обучение на домашнем компьютере. Он показывает, как можно связать компьютер и обучение наукам, компьютер и искусственный интеллект. Он повышает способность людей к обучению и развивает интеллект.

Язык LOGO разработан в МИТ в 1967 году группой по созданию искусственного интеллекта, возглавляемой профессором Сеймром Папертом. Он легок в изучении и освоении. Он помогает людям преодолеть такое препятствие, как "математический талант" в изучении идей программирования и написания программ. Язык программирования LOGO широко используется в различных областях. В нашей стране он является необходимым компонентом к компьютеру для реализации программ обучения в начальной и средней школе, и дети с удовольствием его изучают.

При достаточно простой структуре, LOGO имеет много функций и методов вычислений.

Отличительные особенности LOGO:

❶ LOGO - процедурный язык. Программы компануются из групп команд или процедур.

❷ LOGO - интерактивный язык. Программы и процедуры просто вводятся с клавиатуры.

❸ Данные языка программирования LOGO включают в себя числа, слова и списки. Списки - это вид формата, который формируется из упорядоченных чисел и слов (строк символов). Процедура может быть сама записана и выполнена в форме списка.

❹ Наиболее важная особенность языка LOGO в том, что процедура может вызывать сама себя в форме возврата. Это очень эффективно при написании программ высокого уровня структурирования.

❺ Замечательная особенность LOGO, что "turtle" ("черепаха", далее мы будем называть ее "указателем") может использоваться для рисования на экране. Управляя указателем, можно рисовать различные картинки.

В этой главе Вы ознакомитесь с некоторыми общими вопросами использования языка и техникой программирования. Для получения более детальной информации, пожалуйста ознакомьтесь с описанием MIT LOGO.

5.2.Прежде чем использовать LOGO

1.Типы экранов

❶ Текстовый экран.

Он используется для отображения команд и результатов вычислений, поддерживает диалог человек-компьютер. Его размеры: **28** шагов указателя в ширину и **25** шагов в высоту.

❷ Графический и текстовый смешанный экран.

Этот экран используется для рисования. В нижней части экрана расположены четыре строки для отображения команд и текстовых результатов. В верхней части расположен графический экран с размерами **256** шагов указателя в ширину и **192** - в высоту. Этот графический экран используется для отображения нарисованного. Заметим, что указатель можно передвинуть за нижний край графического экрана, но в этой части картинка не будет видна.

Оба этих типа экрана легко переключаются между собой нажатием клавиш F1 и F2. F1 для текстового экрана и F2 для графическо-текстового.

2.Рабочее состояние LOGO

❶ Командное состояние также называют состояние выполнения. В нем программы выполняются шаг за шагом согласно командам, введенным для выполнения. Оно может быть поделено на текстовое командное состояние и командное состояние для рисования. "?" используется как подсказка в командном состоянии.

❷ Состояние определения также называется состоянием редактирования. Оно используется для написания LOGO процедур. В этом состоянии программы вводятся для записи в память и не выполняются. Однако, Вы можете выполнить их в командном состоянии.

5.3.Рисование с помощью указателя

Экран автоматически перейдет в графический режим, когда Вы начнете рисовать картинку. При рисовании указатель - ручка, а экран - лист бумаги. Все картинки в LOGO рисуются на экране с помощью движения указателя, его опускания и поднятия. Начальная позиция указателя в центре (координаты:(0,0)) графического экрана.

1.LOGO команды и операционные строки

Команды **LOGO** пишутся английскими буквами. Они могут использоваться вместе с определяющими параметрами. Пробел должен использоваться для разделения команды и параметров. На одной строке допускается написание более одной команды, в этом случае они разделяются между собой пробелами. Строка с командами называется **операционной строкой.** Вместе с символами возврата каретки, количество символов в операционной строке не должно превышать 255 (включая пробелы).

2.Команды рисования

1)Вращение и движение указателя

DRAW : Эта команда переведет экран в графический режим, очистит экран и поставит указатель в начальную позицию.

FD X : Эта команда будет двигать указатель вперед на X шагов.
BK X : Движение указателя на X шагов назад.
RT X : Вращение указателя на X градусов по часовой стрелке.
LT X : Вращение указателя на X градусов против часовой стрелки.
HOME : Движение указателя в начальную позицию.

Движение указателя измеряется в его шагах, вращение в градусах. При движении указателя за ним будет оставаться след, с помощью которого рисуются картинки.

2)Поднятие и опускание указателя, скрытие и установка указателя

PU : Эта команда поднимет указатель, то есть при его движении за ним не будет оставаться следа.
PD : Опустит указатель, при движении за ним будет след.
HT : Сделает указатель невидимым.
ST : Сделает указатель видимым.
PC N X : Выбор цвета X для рисунка, где N - целое число от 1 до 3, X - целое число от 0 до 15.

3)Команда REPEAT

Формат: REPEAT A [группа команд]

Эта команда выполняет команды в квадратных скобках A раз.

Пример: Нарисуем квадрат со стороной 50.

```
FD 50 RT 90
FD 50 RT 90
FD 50 RT 90
FD 50 RT 90
```

Однако, эти команды проще написать, используя команду REPEAT:

```
REPEAT 4 [FD 50 RT 90]
```

4) Рисование кривых

Так как основные команды в LOGO позволяют рисовать только прямые линии, то для рисования кривых Вы можете использовать короткие линии. Кривую можно представить как множество коротких прямых линий. На пример: если Вы хотите нарисовать окружность, то можно использовать 360 одинаковых линий, описывающих ее. Когда будет рисоваться окружность, то для каждой линии направление движения указателя должно быть изменено на 1 градус. Для рисования окружности используем команду **REPEAT**:

```
REPEAT 360 [FD 1 RT 1]
```

Конечно, быстрее использовать 36 сегментов, описывающих окружность, но в этом случае она будет не такой правильной.

```
REPEAT 36 [FD 10 RT 10]
```

3.Управление экраном

Переключение между двумя типами экранов:

F1 - переключение в полный текстовой экран,
F2 - переключение в смешанный графическо-текстовой экран.

Функциональные клавиши **F1** и **F2** используются для переключения режимов экрана, при нахождении компьютера в командном состоянии.

❷ Управление границей

Состояние **WRAP** устанавливается по умолчанию, при включении питания. Если указатель передвинется за границу экрана, то он появится с противоположной стороны экрана и будет продолжать движение.

В состоянии **NOWRAP** при пересечении указателем границы экрана, компьютер выдаст сообщение об ошибке.

❸ Рисование в координатах

Каждая точка графического экрана имеет свои координаты. Точка отсчета координат находится в центре экрана. Вертикальная координата может принимать значения от **-199** до **+120**, горизонтальная от **-128** до **+127**.

Команды:

SETXY X Y	- Двигать указатель в позицию (X,Y)
SETX X	- Двигать указатель горизонтально в позицию (X,...)
SETY Y	- Двигать указатель вертикально в позицию (...,Y)
SETH D	- Повернуть указатель на D градусов по часовой стрелке
XCOR	- Выдать значение координаты X текущего положения указателя
YCOR	- Выдать значение координаты Y текущего положения указателя
HEADING	- Выдать текущее направление указателя (в градусах)

Пример: Двигать указатель в новую позицию, которая на 20 шагов левее и на 30 шагов выше текущей позиции: SETXY XCOR -20 YCOR +30

5.4.Операции в LOGO

1.Числа, используемые в LOGO

❶ Целые числа.

Числа, не имеющие десятичной точки, называются целыми. LOGO может воспринимать целые числа в пределах от **-214745367** до **+214745367**. Целое число будет изменено в действительное, если оно выходит за эти пределы. Результат сложения, вычитания и умножения, будет целое число, если оно не выходит за пределы, воспринимаемые LOGO. Но если в операции участвует действительное число, то результат будет действительное число.

❷ Действительные числа.

Числа с десятичной точкой называются действительными. LOGO может обрабатывать действительные числа в пределах от -10^{38} до $+10^{38}$. Если число выходит за эти пределы, то возникает ошибка переполнения.

2.Операции в LOGO

❶ Арифметические операции

В LOGO для операций сложения, вычитания, умножения и деления используются символы + , - , * , / соответственно. Круглые скобки могут использоваться для изменения порядка выполнения операций.

Пример: **(3+4):2-6** *математическое выражение*
(3+4)/4-6 *выражение LOGO*

❷ Переменные в LOGO

Переменная, используемая в LOGO, - это имя, начинающееся на английскую букву и следующие за ней буквы или цифры (или без них). Значение переменной может изменяться в процессе выполнения. Значение определяется оператором присваивания.

***Формат*: MAKE имя переменной значение**

***Примечание*:** При использовании переменной, перед ее именем должно быть двоеточие.

***Примеры*:** FD :A - передвинуть указатель вперед на количество шагов, определяемых значением переменной А.

MAKE "A :A+1 - значение переменной А увеличится на единицу

❸ Функции

Функции - это подпрограммы, которые возвращают определенное значение по их вызову. Часто используемые функции: тригонометрические функции, возведение в степень, вычисление корня

и т.д. Подробную информацию смотрите в таблице. Соответствие некоторых функции LOGO и математических следующее:

математика	LOGO
sin 45	SIN 45
$\sqrt{X}$	SQRT :X
arctg(4/5)	ATAN 4 5

5.5.Процедуры LOGO

Соединенные команды в одну выполняемую группу могут решать определенную задачу или нарисовать картинку. Процедуры LOGO позволяют сохранять группы команд в памяти. В командном состоянии наберите: **TO имя процедуры**

Сейчас Вы вызвали состояние определения процедур. Имя процедуры состоит из нескольких букв и цифр. Как сказано выше, состояние определения процедур называется состоянием редактирования.

❶ Редактирование команд процедур

В состоянии редактирования, Вы можете вводить команды строка за строкой. Если команда полностью не помещается на строке, поставьте в конце строки знак "!" и переведите курсор на новую строку. Вы можете продолжать ввод команды, не считаясь со знаком "!".

Для редактирования используются следующие команды:

Ctrl-D	*- Удаляет символ под курсором.*
Ctrl-K	*- Удаляет все символы справа от курсора.*
←→↑↓	*- Четыре клавиши управления движением курсора.*
Ctrl-A	*- Передвигает курсор в начало строки.*
Ctrl-E	*- Передвигает курсор в конец строки.*
Ctrl-B	*- Передвигает курсор на предыдущий экран.*
Ctrl-L	*- Пролистывает текст, так что строка, содержащая курсор будет вверху экрана.*
Enter	*- Разделяет строку на две в месте, где находится курсор.*
Ctrl-C	*- Выход из состояния редактирования с определением процедуры.*
Ctrl-G	*- Выход из состояния редактирования с отменой определяемой процедуры.*

❷ Вызов процедур

Запустить процедуру, Вы можете просто напечатав имя этой процедуры. Процедура может вызывать другие процедуры (включая саму себя).

Пример: Если Вы определили процедуру с названием ZFX, которая используется для рисования квадрата. Сейчас Вы можете вызвать ее, введя ZFX.

Другой пример: Вы хотите определить процедуру с названием QE, которая вызывает процедуру ZFX:

```
TO QZ
FD 100
ZFX
```

Эта процедура будет использоваться для рисования линии и квадрата.

❸ Параметры процедур

Многие процедуры могут быть сделаны для широкого использования. Например, если Вы хотите рисовать квадрат различных размеров, Вы можете подсоединить вызываемые параметры к процедуре ZFX. Когда Вы вызовете процедуру ZFX рисовать квадрат, то параметру будет присвоено определенное значение. Когда определяется такого вида процедура, то первый параметр должен следовать за именем процедуры и отделяться от нее пробелом. Параметры должны начинаться с английской буквы и далее содержать несколько букв или цифр. Если Вы используете более одного параметра, то они должны отделяться друг от друга пробелами.

Пример:
```
TO ZFX L
REPEAT 4 [FD :L RT 90]
END
```

Теперь, если Вы желаете нарисовать квадрат со стороной 46, наберите следующее: ZFX 46

❹ Оператор сравнения

Формат оператора сравнения: **IF выражение сравнения THEN команда 1 ELSE команда 2**

Команда 2 может быть опущена, если в этом есть нужда. Этот оператор означает, что если выражение сравнения истинно, то выполняется команда 1, в противном случае выполняется команда 2.

Ниже приведены некоторые простые выражения сравнения:

LOGO	математика
:A = :B	A = B
:A > :B	A > B
:A < :B	A < B
NUMBER? :N	N число ?
WORD? :N	N слово ?
LIST? :N	N список ?

В выражениях сравнения можно использовать следующие операции:

ANYOFF сравнение 1 сравнение 2 - если одно из сравнений истинно, то результат истина.
ALLOFF сравнение 1 сравнение 2 - если все сравнения истинны, то результат истина.
NOT сравнение - если сравнение ложно, то результат истина.

Следующие команды обычно используются с оператором сравнения:

STOP - Остановить выполнение процедуры.
OP :X - Прекратить выполнение и возвратить X вызывавшей.
TOPLEVEL - Прекратить выполнение процедуры и возвратиться в состояние выполнения.
GO "M - Оператор перехода на строку с меткой M.

Другая команда, которая использует результат сравнения. Формат:

TEST сравнение - результат сравнения
:
IFT команда 1 - если сравнение истинно, то выполняется команда 1
:
IFF команда 2 - если сравнение ложно, то выполняется команда 2

Операторы IFT и IFF могут оба присутствовать или оба отсутствовать за оператором TEST. В процедуре может использоваться более одного оператора TEST. И более, чем один оператор IFT или IFF также могут следовать за каждым оператором TEST. Операторы IFT и IFF используют результат последнего встретившегося оператора TEST. Допускается вставлять более, чем один оператор между этими операторами.

❺ Повторяющиеся процедуры

Повторяющиеся процедуры это процедуры, которые вызывают сами себя. Условие повторения процедур может быть определено оператором сравнения.

Пример: Рассмотрим следующую простую повторяющуюся процедуру:

```
TO AA
FD 20 RT 21
AA
END
```

Результатом процедуры AA будет движение указателя на 20 шагов вперед и поворот его на 21 градус по часовой стрелке.

Другой пример: Указатель должен передвинуться сто раз, а затем остановиться.

```
TO AA I
IF I=0 THEN STOP
FD 20 RT 21
AA :I-1
END
```

Результат будет получен при присвоении параметру I значения 100.

5.6.Слова и списки

❶ Слова

Слова - это строка символов, начинающаяся с кавычек " и заканчивающаяся пробелом. Эти символы могут быть буквами, цифрами и другими символами за исключением кавычек. Каждый символ строки называется элементом строки.

Примечание: ① Слово "5*3 не есть число 15.

② Кавычки могут быть опущены, если слово состоит только из цифр.

Команды, которые используются для обработки слов:

FIRST "слово - первый символ старого слова образует новое слово.
LAST "слово - последний символ старого образует новое.
BF "слово - все символы старого слова за исключением первого образуют новое слово.
BL "слово - все символы старого слова за исключением последнего образуют новое слово.
WORD "слово 1 "слово 2 - комбинация двух слов образует новое слово.

❷ Списки

Список - это последовательность слов, заключенных в квадратные скобки. Все кавычки в словах должны быть опущены. Слово в списке называется элементом списка. Количество слов в списке называется длиной списка. Список, не имеющий ни одного слова, называется пустым списком. Пустой список имеет только один символ.

Операции со списками следующие:

FIRST [список] - выделяет первый элемент списка
LAST [список] - выделяет последний элемент списка
BF [список] - удаляет первый элемент списка
BL [список] - удаляет последний элемент списка
FPUT элемент список - подсоединяет элемент в начало списка
LIST элемент 1 элемент 2 - формирует список, используя два элемента
LPUT элемент список - подсоединяет элемент в конец списка

Глава 6. Эксплуатация SONIC REC-9388

❶ SONIC REC-9388 должен быть расположен подальше от мест с высокими или низкими или быстро меняющимися температурами. Оберегайте компьютер от попадания прямых солнечных лучей и храните его в свободном от пыли месте.

❷ Не пытайтесь открыть нижнюю крышку компьютера, не заменяйте электронные компоненты и не включайте какие-нибудь новые на плате компьютера. Если Вы столкнулись с какой-нибудь трудностью при использовании SONIC REC-9388, пожалуйста, обратитесь к Вашему местному дистрибьютеру.

❸ Оберегайте компьютер от ударов и вибраций.

❹ Будьте осторожны, не разбрызгивайте кофе, соки, чай или другие жидкости на поверхности компьютера. Не используйте органические растворители для чистки пластмассовых деталей.

❺ Если в вашем компьютере неисправность, то обратитесь в Ваш местный центр по техническому обслуживанию.

❻ Не включайте питание до установки Вашего компьютера. Не вынимайте и не вставляйте кассеты при включенном питании.

❼ Не используйте блоки питания от других устройств вместо прилагаемого блока питания.

Не оставляйте этот компьютер долгое время без работы, работайте на нем время от времени.

Часть 2. Составление программ на BASIC

Глава 1. Основные понятия BASIC (Бейсик)

BASIC - это сокращение от слов "Beginner's All Purpose Instruction Code" или "Многоцелевой язык символических команд для начинающих". Это очень популярный язык программирования. Это язык программирования высокого уровня, который соединяет в себе простоту освоения и практическое его использование. Легко изучить любой другой язык программирования высокого уровня, после того как изучил BASIC.

1.1.Особенности BASIC

BASIC (бейсик) можно разделить на стандартный бейсик и расширения к нему. Любая версия бейсика содержит в себе все 17 операторов стандартного бейсика. Действие, использование и форматы всех 17 операторов почти такие же во всех версиях бейсика. Это делает программы на бейсике более обобщенными. Во многих версиях языка программирования бейсик, также используются другие операторы совместно с 17 основными операторами. Все эти операторы формируют расширение бейсика. Причем одни расширения бейсик могут отличаться от других.

Языки программирования бейсик могут подразделяться на два типа по способу представления в них чисел, а именно на бейсик с числами с плавающей точкой и бейсик с целыми числами. Бейсик, который использует числа с плавающей точкой, может производить алгебраические функции, такие как синус, логарифм, извлечение корня и другие. Целочисленный бейсик может производить только операции над целыми числами и алгебраические функции не имеет.

Кассета с системой языка программирования бейсик, разработанная для учебного компьютера SONIC REC-9388, содержит бейсик с плавающей точкой, который может использоваться для сложных инженерных и математических расчетов. Язык программирования F-BASIC, записанный на РВ-кассете, есть целочисленный бейсик и может использоваться для операций над целыми числами. Но он содержит много операторов для рисования, имеет хорошие графические возможности. Это дает возможность рисовать картинки и писать игровые программы. (Подробности смотрите в соответствующей части описания).

Особенности языка программирования бейсик следующие:

❶ **Легкость изучения**
Стандартный бейсик содержит 17 операторов, которые происходят из английских слов. Символы операций и выражений в стандартном бейсике также очень похожи на используемые в математике. Это облегчает их понимание и запоминание. Кроме того структура бейсик программ очень проста и синтаксические правила изучать не сложно. Действительно, бейсик очень прост для обучения начинающих.

❷ Язык программирования бейсик один из самых удобных в реализации диалога человек-компьютер. Компьютер автоматически проверяет, есть ли какие-то синтаксические ошибки или ошибки, произошедшие при выполнении программы, которая была введена. Все сообщения об ошибках отображаются на экране для того, чтобы можно было исправить ошибки.
❸ Язык программирования бейсик предлагает путь обработки программ немедленно после ввода. Использование этого пути, дает возможность производить какие-нибудь простые вычисления или выполнять какие-нибудь операторы. Также допускаются специально предназначенные операторы, которые позволяют программе сделать паузу в течение выполнения, для того чтобы узнать как изменились значения некоторых переменных или изменить значения опеределенных переменных. Это очень удобно для отладки программ.
❹ **Бейсик** - язык высокого уровня и для общего применения. Он может производить математические операции не только со значениями, но также и без них.

1.2.Операторы и функции бейсика

1.Стандартные операторы бейсика
❶ **Следующие 17 операторов используются всеми версиями бейсика.** (*В скобках дается перевод английских слов, от которых образованы названия операторов*)

1.LET	(Пускай)	**2.INPUT**	(Ввести)
3.READ	(Читать)	**4.DATA**	(Данные)
5.RESTORE	(Восстановить)	**6.PRINT**	(Печатать)

7.GOTO (Идти на) **8.IF .... THEN**(Если .... то)
9.STOP (Остановить) **10.REM** (Примечания)
11.FOR .... NEXT(Для ...следующий) **12.GOSUB** (Перейти к подпрограмме)
13.RETURN (Возврат) **14.DEF** (Определить)
15.FN (Функция) **16.DIM** (Размер)
17.END (Конец)

❷ **Операторы расширения бейсик для домашнего компьютера SONIC REC-9388**

1.HOME (Домой) **2.FRE** (Свободно)
3.CALL (Вызвать) **4.POKE**
5.TRACE (Проследить) **6.UNTRACE** (Прекратить прослеживание)

❸ **Стандартные функции, используемые в бейсике для SONIC REC-9388**

Название	Функция	Примечания
SIN(X)	вычисляет значение sin(x)	x в радианах
COS(X)	вычисляет значение cos(x)	x в радианах
TAN(X)	вычисляет значение tg(x)	x в радианах
ATN(X)	вычисляет значение arctg(x)	x $[-\pi/2,\pi/2]$
LOG(X)	вычисляет значение ln(x)	x > 0
EXP(X)	вычисляет значение e^x	e = 2.71828
ABS(X)	вычисляет абсолютное значение x	
SQR(X)	вычисляет квадратный корень x	x > 0
INT(X)	вычисляет целую часть от x	INT(-8.6)=-9,INT(8.6)=8
SGN(X)	определяет знак числа x	1 (x > 0) 0 (x = 0) -1 (x < 0)
RND(X)	выдает случайное число меньше x	x -действительное число

❹ **Символьные функции**

LEN(строка символов) - выдает длину данной строки.
STR$(арифметическое выражение) - выдает соответствующую строку символов для данного арифметического выражения.
VAL(строка символов) - выдает соответствующее арифметическое значение для данной строки символов.
CHR$(число код ASCII) - выдает соответствующий символ для данного значения кода ASCII.

❺ **Другие функции**

TAB - эта функция для перехода к данной позиции строки при печати.
PEEK- функция значения указанного байта из памяти.

1.3.Создание программ на бейсике

Для начала рассмотрим следующий простой пример:

Задача: Дано: В кармане "A" имеется 15 долларов,в кармане "B" - 20 долларов и 50 центов. Сколько всего денег имеется в наличии ?

Решим эту задачу на бейсике, для этого напишем следующую программу:

```
10 LET A=15
20 LET B=20.5
30 LET C=A+B
40 PRINT C
50 END
```

В то время, пока программа будет выполняться, переменной A присвоится 15, затем переменной B присвоится 20.5 и сумма A и B присвоится переменной C (Выполнение

операторов LET), затем значение переменной С печатаем на экран (Выполнение оператора PRINT), далее выполнение программы остановится (Оператор END).

Правила построения программ на бейсике следующие:

1.Программы на бейсике состоят из нескольких строк. В нашем примере мы видим, что программа из 5 строк.

2.Все строки нумеруются и имеют операторы.

❶ **Строки:** В вышеприведенной программе 10,20 и т.д.,все строки пронумерованы. Номера строк могут быть десятичными целыми числами. Они определяют порядок выполнения операторов. Компьютер будет выполнять операторы последовательно от наименьшего номера до наибольшего. Пределы номеров строк определяются используемым компьютером. Номера строк для учебного компьютера REC-9388 могут изменяться от 0 до 9999. Нумерация строк в программах должна иметь разрывы, для того чтобы было удобно вставлять новые операторы. В нашей задаче, если Вы хотите узнать разницу денег между А и В и ее напечатать, Вам только нужно вставить новую строку: 45 PRINT B-A

❷ **Операторы:** В строках программы, в правой части от номера строки находится оператор. Оператор состоит из названия оператора и описания к нему.

а)*Названия операторов*: В нашей программе: LET, PRINT и END - названия операторов. Они используются для определения, какое действие выполняет оператор. Они определяются в зависимости от языка программирования, где они используются и они могут распознаваться при переводе программы компьютером. Название операторов это зарезервированные слова, которые не могут быть изменены или написаны свободным образом.

б)*Описание к оператору в нашем примере*, А=15, В=20.5 и С=А+В - это все описания к оператору. Они являются объектами обработки компьютером. Также могут быть операторы, не имеющие описаний, например оператор END.

3.Обычно говорится, что на каждой строке программы может помещаться один оператор, но многие версии бейсика позволяют располагать на одной строке более, чем один оператор. В этом случае операторы отделяются друг от друга двоеточием, т.е. знаком ":".

4.Оператор должен помещаться на одной строке программы. Не разрешается написание оператора на двух строках. Как из нашего примера, нельзя написать оператор на строке 30 в следующем виде:

```
30 LET C=
35 A+B
```

В случае, если оператор длинный и написать его на одной строке невозможно, то можно использовать несколько операторов, заменяющих его. Для примера:

```
10 LET A=12345+6789+654-768-567+(X-Y)-23456
```

Перепишем этот оператор в следующем виде:

```
10 LET B=12345+6789+654-768
20 LET A=B-567+(X+Y)-23456
```

5.Оператор END обычно последний оператор в программе и показывает конец программы.

1.4.Константы и переменные

1.Константы

Значение константы не может изменяться в течение выполнения программы.

Константы подразделяются на три вида, такие как:

❶ **Целочисленные.** Целые числа используются в REC-9388 в пределах от **-999999999** до **+999999999**, целые числа вне этих пределов не могут обрабатываться и воспринимаются как реальные (действительные) числа. Предел изменения целых чисел, используемый в REC-9388, в некоторой степени небольшой, но зато они могут быть быстрее обработаны.

❷ **Реальные (действительные) числа.** Предел изменения реальных чисел в учебном компьютере REC-9388 от -10^{38} до $+10^{38}$, если абсолютное значение реального числа меньше чем 10^{-39},то компьютер считает его как нуль. Если абсолютное значение реального числа больше чем 10^{39}, то компьютер выдает на экран сообщение, что произошла ошибка переполнения. Число знаков в реальном числе равно 9 . Если число знаков больше чем 9, то они будут изменены в соответствии с правилами преобразования.

❸ **Строковые константы.** Последовательность символов, заключенных в кавычки, называется строковой константой. Число символов в ней не должно превышать 104. До тех пор пока кавычки не определили символы как строку, символы не будут восприниматься как строка. Тем не менее апостроф может использоваться вместо него. Строка, не имеющая в себе символов, называется пустой строкой.

2.Переменные

Значение переменной может быть изменено в процессе работы программы.

Имя переменной должно соответствовать следующим правилам:

① Число символов в имени переменной не должно превышать 104, но только два первых символа эффективны.

② Первым символом имени переменной должна быть заглавная английская буква, другие символы могут быть как буквы, так и цифры.

③ Зарезервированные слова не могут использоваться в качестве имен переменных или как часть их.

④ В соответствии с видами значений переменных, переменные подразделяются на целочисленные, реальные и строковые.Символ "$" в конце имени переменной говорит о том, что переменная строковая. Переменные арифметического типа несовместимы с переменными символьного типа.

1.5. Символы операций и выражения

Если несколько констант, переменных и функций одного типа используются вместе, то между ними должен находиться символ операции, для того чтобы сформировать выражение. При этом выражение будет иметь свое собственное значение.

1.Арифметические символы и арифметические выражения

Арифметические символы операций:
+ (плюс), - (минус), * (умножить), / (делить), ^ (возведение в степень).

Одно арифметическое число или два арифметических числа, соединенных арифметическим символом, называются **арифметическим выражением**, которое имеет значение арифметического типа.

2.Символы сравнения и выражения сравнения

Эти символы следующие:

=	эквивалентно "равно" в математике
>=	эквивалентно "больше или равно" в математике
>	эквивалентно "больше" в математике
<	эквивалентно "меньше" в математике
<=	эквивалентно "меньше или равно" в математике
<>	эквивалентно "не равно" в математике

Два выражения с одинаковым (или совместимым) типом представления чисел, соединенные между собой символом сравнения, образуют выражение сравнения. Выражение сравнения имеет логическое значение, т.е. **TRUE** (Истина) или **FALSE** (Ложь), TRUE и FALSE могут быть представлены как 1 и 0 соответственно.

3.Логические символы и логические выражения

Логические символы :

NOT	-	логическое "не"
AND	-	логическое "и"
OR	-	логическое "или"

Выражение, которое использует логические символы, называется **логическим выражением**. Значение логического выражения может быть 1 или 0. Предположим, что A и B образуют логическое выражение, тогда правила логических операций следующие:

A	B	NOT A	A AND B	A OR B
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

4.Приоритеты выполнения символов операций

В выражении могут использоваться различные символы операций, приоритеты при их выполнения следующие:

() - *скобки*
функции
+,-,NOT - *положительное, отрицательное, символ отрицания*
^ - *возведение в степень*
***,/** - *умножить, разделить*
+,- - *плюс, минус*
<,<=,>,>=,<> - *символы сравнения*
AND - *логическое "и"*
OR - *логическое "или"*

5.Символы операций над строками

Есть только один символ операции для строк - "+" (подсоединить), который используется для соединения двух строк символов.

Например : Пусть нам необходимо соединить две строки "ABCD" и "123". Соединяем их при помощи символа подсоединения : "ABCD"+"123" . В результате получаем : "ABCD123"

1.6.О работе на SONIC REC-9388

В этом разделе, мы рассмотрим процесс работы и некоторые наиболее часто используемые команды.

Методы работы различаются в зависимости от используемого компьютера. В нашем случае метод работы ограничивается системой бейсик учебного компьютера **SONIC REC-9388.**

1.Установка системы

Пожалуйста, ознакомьтесь с главой 2 первой части руководства.

2.Схема клавиатуры

См.рис. на стр.38 "Руководства по эксплуатации" на английском языке.

1) Включение питания.
2) Кнопка сброса.
3) Индикатор состояния. (**Не светится** - состояние компьютера, **Светится** - состояние игровой приставки)
4) Индикатор питания (**Светится** - компьютер включен).
5) Специальные функциональные клавиши **F1-F12**.
6) Клавиша ключа (**ESC**).
7) Клавиши цифр и символов.
8) Клавиша табуляции.
9) Заглавные буквы.
10) Клавиша - верхний регистр.
11) Управляющая клавиша.
12) Клавиши букв алфавита.
13) Клавиша пробела.
14) Клавиша включения/выключения цифр на правой клавиатуре.
15) Клавиша "домой".
16) Клавиша ввода.
17) Клавиша вставки.
18) Клавиша удаления.
19) ←→↑↓ - клавиши управления курсором.

3.Ввод программы

Лучше всего использовать две команды после установки системы *бейсик*.

1.**NEW** - эта команда очищает все содержимое памяти.
2.**HOME** - эта команда очищает экран и помещает курсор в верхний левый угол экрана.

Внимание: Пожалуйста, нажимайте клавишу **Enter** после ввода каждой команды, чтобы они могли быть восприняты REC-9388.

После выполнения этих двух команд, Вы можете вводить Вашу программу. При вводе программы, Вы должны ввести все символы, включая номер строки и следующие за ним операторы. Лучше всего использовать пробелы для отделения номера строки, операторов и описаний к ним, так как это облегчит чтение программы. Если Вы допустили ошибки при наборе, то используйте клавишу курсора "влево" для перемещения курсора к позиции, где допущена ошибка и далее исправьте ее. Помните, что нужно нажать клавишу **Enter** в конце строки программы для ввода новой.

Для облегчения набора номеров строк используйте команду **AUTO**.

Формат AUTO:

***AUTO* начальный номер строки , интервал в нумерации**

По умолчанию интервал в нумерации равен 10. Команда не может использоваться без указания начального номера.

4.Отображение программы на экране

После ввода программы на бейсике, можно использовать команду ***LIST***, для просмотра текста программы на экране.

Форматы *LIST*:

***LIST* номер строки**	*- вывод определенной программной строки*
***LIST* номер строки 1-номер строки 2**	*- вывод текста программы от номера строки 1 до номера строки 2*
***LIST* - номер строки**	*- вывод текста программы от начала до номера строки*
LIST	*- вывод на экран всего текста программы.*

5.Изменение программы

EDIT - команда, обычно используемая для редактирования строки программы.

Формат *EDIT*:

***EDIT* номер строки**

В режиме редактирования Вы можете исправить ошибки, передвигая курсор в позицию, где они находятся. После выполнения изменений нажмите клавишу **Enter** для завершения режима редактирования и сохранения сделанных изменений.

6.Выполнение программ

Есть два вида выполнения: с задержкой и немедленное выполнение.

Используя **вид выполнения с задержкой,** программа будет выполняться только после ввода команды ***RUN*** и будет оставаться в памяти после ее выполнения.

Используя **немедленное выполнение,** оператор или операторы (без номера строки, разделенные знаком ":") будут выполняться немедленно после нажатия клавиши **Enter.** При этом исполняемые операторы не будут сохранены в памяти.

7.Загрузка и вызов программ

Пожалуйста, ознакомьтесь с главой 4 части I.

Глава 2. Ввод / вывод данных

Некоторые программы могут нуждаться в вводе данных. Также результаты выполнения программ (включая и промежуточные результаты) нуждаются в отображении на экране или печати на бумаге. Поэтому процессы ввода/вывода данных необходимы.

В этой главе, мы рассмотрим операторы ввода/вывода, такие как: ***LET, INPUT, READ, DATA, RESTORE*** и ***PRINT***.

Для написания форматов операторов будем использовать следующие символы (которые не принадлежат самому оператору):

❶ **[]** - *выделенный контекст может быть выбран.*
❷ **{}** - *выделенный контекст может неоднократно повторяться.*
❸ **|** - *нужно выбрать часть справа или слева.*

2.1.Операторы ввода - вывода

1.*PRINT* - главный оператор вывода данных в языке программирования бейсик.

Формат *PRINT:*
[номер строки] *PRINT* [выражение]{,|;[выражение]}

Функция этого оператора - печать на экране значений выражений следующих за PRINT.

Примечание:

❶ Результат вывода есть значения выражений, а не сами выражения.
Пример: PRINT 2+3
Число "5" отобразится вместо "2+3".

При печати строковых переменных будет отображаться содержание, заключенное в кавычки.
Пример: PRINT "ПРИВЕТ!!!"
Содержание ПРИВЕТ!!! будет отображено вместо "ПРИВЕТ!!!".

❷ Если в конце описания к оператору ***PRINT*** нет символов "," и ";", то следующая печать будет продолжаться с начала новой строки.
Пример:

```
10 PRINT 15
20 PRINT 30
30 END
```

После ввода RUN отобразится:

```
15
30
```

Если в операторе ***PRINT*** отсутствует описание, то будет пропускаться одна строка.

❸ Если в конце описания к оператору ***PRINT*** находится символ ";", то следующая печать будет осуществляться с позиции сразу после результатов вывода предыдущего ***PRINT***.
Пример:

```
10 PRINT 3+5;
20 PRINT "-ВОСЕМЬ"
30 END
RUN

 8-ВОСЕМЬ
```

❹ Если в конце описания к оператору ***PRINT*** находится символ ",", то следующая печать будет осуществляться со следующей части экрана.

Экран в русско-английском учебном компьютере SONIC REC-9388 имеет в каждой строке по 30 символов, которые разделены на 3 части: с 1 по 7, с 8 по 15, с 16 по 30.

Пример:

```
10 PRINT 3,
20 PRINT "ПЛЮС",
30 PRINT 2,
40 PRINT "ПЯТЬ"
50 END

3       ПЛЮС    2
ПЯТЬ
```

❺ Оператор ***PRINT*** может выводить более одного выражения, которые разделяются между собой знаками "," или ";".

❻ При вводе ***PRINT*** можно заменить знаком "**?**".

2.*LET*

*Оператор **LET** используется для присваивания значений переменным.*

Формат *LET*:

[номер строки] [*LET*] имя переменной = выражение

Функция этого оператора состоит в том, чтобы присвоить значение данного выражения переменной.

Пример:

LET A=10 - Это значит, что A присвоено значение 10

Примечания:

❶ При вводе название оператора "LET" можно опустить.
❷ Знак присваивания "=" отличен от математического "равно".
❸ Переменная и выражение должны быть совместимы по типу представления данных.
❹ Если переменной не было присвоено значение, то по умолчанию оно равно нулю или пустой строке.
❺ Оператор не может быть продолжен.
Пример: 10 A=B=30 - неправильный оператор,

однако 10 A=30
20 B=A - правильно.

3.*INPUT*

Формат *INPUT*:

[номер строки] *INPUT* ["строка для подсказки"] имя переменной [{,имя переменной}]

Функция оператора ***INPUT*** - ввод значений переменных с клавиатуры.

Если вводится более чем одна переменная, то Вы можете ввести каждое значение переменной в отдельности или все значения вместе (при этом вводимые значения должны отделяться друг от друга запятой).

Пример:

```
10 INPUT A,B,C
20 PRINT A,B,C
30 END
```

Можно ввести значения переменных A,B,C тремя различными способами:

① RUN
? 12

```
? 34
? 567.8
12    34    567.8
```

②
```
RUN
? 12,34
? 567.8
12    34    567.8
```

③
```
RUN
? 12,34,567.8
12    34    567.8
```

Примечания:

❶ Если используется строка символов для подсказки, то она будет отображена перед знаком вопроса.

❷ При ответе на оператор *INPUT* могут использоваться только константы. Имена переменных, выражения и функции восприниматься не будут.

Если в процессе ввода данных Вы сделали ошибку, то на экране появится сообщение "?REENTER" (введите еще раз) и Вам нужно ввести данные еще раз.

Если Вы хотите ввести данные в строковую переменную, то данные должны быть строковыми константами, причем константы не нужно заключать в кавычки.

❸ Константы будут вводиться в том количестве, сколько переменных находится после оператора *INPUT*.

4. *READ* и *DATA*

Вы можете присвоить значения переменным, используя операторы *READ* и *DATA*.

Форматы операторов *READ* и *DATA*:

[номер строки] *READ* имя переменной [{,имя переменной}]
номер строки *DATA* [константа [{,константа}]

Значения констант в операторе *DATA* могут быть присвоены переменным в операторе *READ*.

Пример:

```
10 READ A,B,C,D
20 DATA 75,92,80,63
30 PRINT A,B,C,D
RUN

75    92    80
63
```

Примечания:

❶ Оператор *DATA*, который используется для представления значений, не выполняется и может быть вставлен в любое место программы. Один оператор *DATA* может быть поделен на несколько операторов *DATA*, но порядок следования значений должен быть сохранен. Оператор *DATA* в нашем примере может быть переписан в следующем виде:

```
20 DATA 75,92
25 DATA 80,63
```

Кроме того, значения в операторе *DATA* не могут быть переменными, выражениями или функциями.

❷ Число значений в операторах *DATA* не должно быть меньше, чем переменных в операторах *READ*. Позволительно для программ иметь операторы *DATA* и не иметь операторы *READ*, но обратное не позволяется.

❸ Тип представления данных констант должен быть совместим соответствующим переменным в операторе *READ*.
❹ Если в операторе *DATA* присутствуют строковые константы, то кавычки можно опустить. Но кавычки не могут быть опущены, если в строковой константе присутствуют пробелы и запятые.

5.*RESTORE*

Этот оператор позволяет переопределить значения в операторе *DATA*.

Формат оператора *RESTORE*:
[номер строки] *RESTORE* [номер строки в которой оператор DATA]

Если в операторе *RESTORE* не используется описание, то восстанавливаются значения всех операторов *DATA*. Иначе восстанавливается обозначенный оператор *DATA*.

Примеры:

Пример 1:
```
10 READ A,B,C,D
20 DATA 10,20
30 DATA 30,40
40 RESTORE
50 READ E,F
60 PRINT A,B,C,D
70 PRINT E,F
80 END
RUN

10      20      30
40
10      20
```

Пример 2:
```
10 READ A,B,C,D
20 DATA 10,20
30 DATA 30,40
40 RESTORE 30
50 READ E,F
60 PRINT A,B,C,D
70 PRINT E,F
80 END
RUN

10      20      30
40
30      40
```

2.2.Сравнение трех операторов присвоения значения переменным

LET, *INPUT* и *DATA/READ* - операторы, присваивающие значения переменным. Каждый из них имеет свои особенности.

LET присваивает переменным значения выражений. Таким образом, используя оператор *LET*, можно сохранять множество промежуточных результатов. Эта его главная особенность. Но оператор LET неудобно использовать для присвоения переменным большого числа значений. Недостаток оператора *LET* и то, что программа не сможет изменить сама себя, если исходные значения нужно поменять.

Операторы *READ* и *DATA* могут быть использованы для присвоения групп данных. Программа также не сможет изменить исходные значения данных, если это необходимо сделать.

Оператор *INPUT* может присваивать значения переменным с клавиатуры в процессе работы программы. Не нужно менять текст программы, когда нужно изменить исходные значения переменных.

Используйте один из этих операторов в соответствии с необходимостью.

2.3.Использование математических функций

Математические функции часто используются вместе с операторами присваивания печати. Формат этих функции такой:

Имя функции (арифметическое выражение)

Выражение в этом формате точно такое же, как у независимой переменной X, которая была обсуждена в главе 1.

1.*EXP(X)*

Используется для вычисления значения e в степени x, причем e=2.17828183
Если Вы хотите вычислить значение e в степени 10, Вы можете использовать следующую программу:

```
10 PRINT EXP(10)
20 END
RUN

22026.4658
```

2.*LOG(X)*

Используется для вычисления натурального логарифма от x.
Если Вы желаете вычислить значение ln(100), Вы можете использовать следующую программу:

```
10 PRINT LOG(100)
20 END
RUN

4.60517019
```

3.*SIN(X),COS(X),TAN(X),ATN(X)*

В ***SIN(X),COS(X)*** и ***TAN(X)*** значение X должно быть представлено в радианах. Результат ***ATN(X)*** также представляется в радианах.
Чтобы перевести градусы в радианы нужно использовать коэффициент $\pi/180$.

Пример:

```
10 PRINT "ВВЕДИТЕ УГОЛ В ГРАДУСАХ"
20 INPUT A
30  AC=COS(A*3.14159/180)
40 PRINT "КОСИНУС=";AC
50 END
```

4.*SQR(X)*

Значение X должно быть положительным.

Пример: Проверим равенство $\sin^2\alpha+\cos^2\alpha=1$, для чего составим следующую программу:

```
10 PRINT "ВВЕДИТЕ УГОЛ"
20 INPUT K
30  C=3.1416/180
30 PRINT SQR((SIN(K*C))^2+(COS(K*C))^2)
40 END
RUN

1
```

5.*INT(X)*

Эту функцию можно объяснить на следующей программе:

```
10  A=89.6:B=45.6754:C=1235.87
20  D=5437.98776:E=659.56
30 R=A+B+C+D+E
40 S=INT(R)
50 T=INT(R+0.5)
60  U=INT(R*100+0.5)/100
70 PRINT R,S,T,U
80 END
RUN
7468.69316          7468
7469       7468.69
```

В программной строке 30 вычисляется сумма A,B,C и D - 7468.69316. В строке 40 вычисляется целая часть числа - 7468. В программной строке 50 вычисляется число, округленное до целого. В строке 60 вычисляется число, округленное до второго знака после запятой.

В следующей программе вычислим частное и остаток от деления 59 на 16:

```
10 A=INT(59/16)
20 B=59-A*16
30 PRINT "59/16=";A;"...";B
40 END
RUN

59/16=3...11
```

Другие функции, такие как ***SGN(X)***, ***ABS(X)*** и ***RND(X)*** просты в использовании и не нуждаются в детальном пояснении.

Глава 3. Переходы в программе

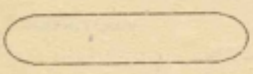
Программы, которые мы рассматривали в предыдущих главах, выполнялись строчка за строчкой. Однако, на практике во многих программах требуется нарушать это правило. В этой главе, рассмотрим два оператора ***GOTO*** и ***IF...THEN***, которые используются для реализации функции перехода. Также в этой главе мы рассмотрим использование операторов ***STOP***, ***CONT*** и ***REM*** и методы отладки программ.

3.1.Блок-схемы

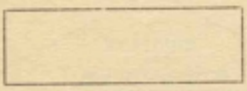
Несложно написать программу для решения простых математических задач. Но как справиться со сложными задачами, если Вы должны предусмотреть в программе все логические сравнения и различные условия, определяющие ход выполнения программы. В этом случае очень удобно писать программу с помощью блок-схем.

Преимущество блок-схем в простоте и в удобстве проверки и обмене идеями.

В блок-схемах в основном используются следующие составляющие:

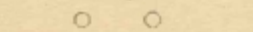


- овальный прямоугольник, который используется, чтобы показывать начало и конец программы.



- прямоугольник, используется для рабочих операций.

- ромб, используется для операций ветвления программы.



- соединительные кружки, в блок-схеме показывают места соединения.

3.2.Оператор перехода

1.Оператор безусловного перехода *GOTO*

Формат *GOTO*:
[номер строки] *GOTO* номер строки

Функция этого оператора в том, что выполнение программы безусловно переходи строку с указанным номером.

Пример: Для трех чисел сделаем так, чтобы производилась операция накоплен результат выводился на экран. Для этого мы напишем следующую программу:

```
10 S=0
20 INPUT X
30 S=S+X
40 PRINT S
50 INPUT X
60 S=S+X
70 PRINT S
80 INPUT X
90 S=S+X
100 PRINT S
110 END
```

Одиннадцать строк включает в себя эта программа, если же количество чисел использу больше, то это приведет и к росту количества строк программы.

Внимательно рассмотрев эту программу, можно обнаружить, что операторы в стро 20,30 и 40 постоянно повторяются (в нашем случае 3 раза). Можно предположить, что если сделаем программу, в которой осуществляется переход от строки с номером 40 на строку с номер 20, то эти три оператора будут выполняться бесконечно (в действительности пока не произой ошибки переполнения переменной S).

Оператор ***GOTO*** может реализовать такое предположение, для этого перепишем програм в следующем виде:

```
10 S=0
20 INPUT X
30 S=S+X
40 PRINT S
50 GOTO 20
60 END
```

При таком простом использовании оператора ***GOTO*** для остановки выполнения програм можно использовать одновременное нажание клавиш **Ctrl + C** .

2.Оператор условного перехода *IF ... THEN*

Для начала рассмотрим следующую программу:

```
10 S=0
20 INPUT X
30 IF X=0 THEN END
40 S=S+X
50 PRINT S
60 GOTO 20
```

Теперь Вы можете остановить выполнение программы, введя 0, дополнительно к нажати **Ctrl+C**.

[номер строки] *IF* условие *THEN* оператор [{: оператор}]

Причем часть, следующую за ***THEN***, будем называть подоператорами ***THEN***.

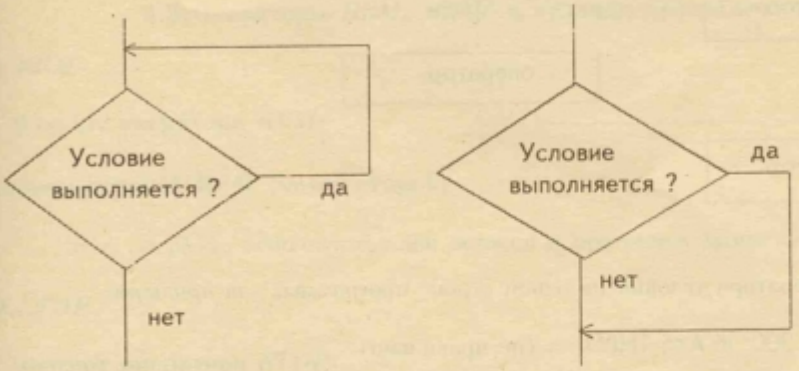
Условное выражение может быть выражением сравнения или логическим выражением. Подоператоры ***THEN*** выполняются, когда условие выполняется, в противном случае выполняются операторы следующей строки.

Благодаря различным формам представления подоператоров ***THEN***, приблизительно разделим форматы ***IF...THEN*** как следующие:

❶ **[номер строки] *IF* условие *THEN GOTO* номер строки**

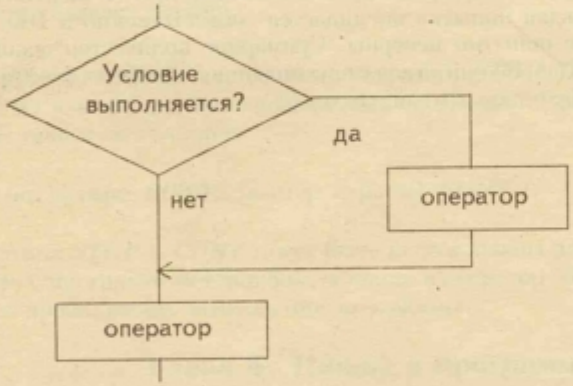
Программа будет переходить на указанную строку программы, если условие выполняется, в противном случае будет выполняться следующая программная строка.

Блок-схема



❷ **[номер строки] *IF* условие *THEN* оператор**

Программа пропускает подоператор ***THEN***, если условие не выполняется, или будет выполняется подоператор ***THEN***.

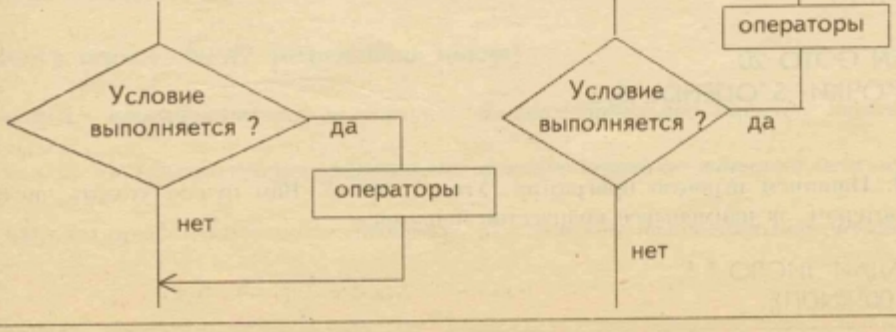


❸ **[номер строки] *IF* условие *THEN* оператор[{:оператор}]: *GOTO* номер строки**

Причем **оператор,оператор,...** не операторы ***GOTO***.

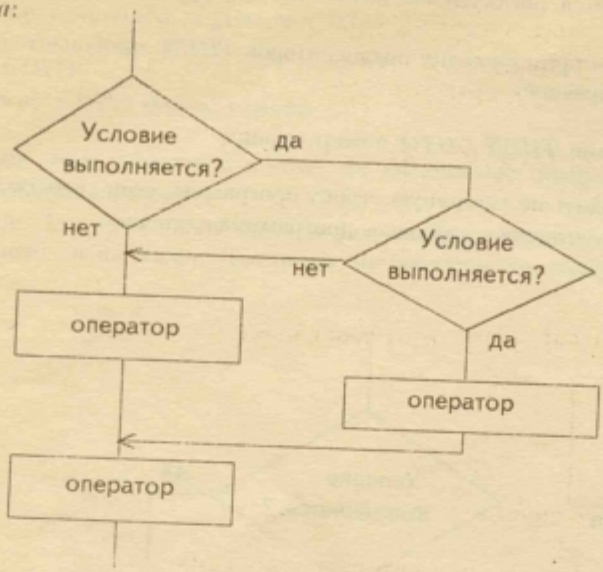
Программа будет выполнять следующую строку программы, если условие не выполняется, или будут выполняться подоператоры ***THEN***.

Блок-схемы:



❹ **[номер строки] *IF* условие *THEN IF* условие *THEN* [{*IF*...*THEN*}] оператор,[{:..**

Блокехема:



❺ Нельзя иметь два оператора условия на одной строке программы, *для примера*:

100 IF A=5 THEN GOTO 300: IF A<5 THEN .... (не правильно)

3. Примеры составления программ

Пример 1: Составим программу, которая может использоваться в начальной школе для обучения операции сложения чисел до 100. Предоставляется пять задач, причем на решение каждой задачи дается четыре попытки. Каждая попытка оценивается соответственно в 100, 75, 50 и 25 очков. Ноль очков дается, если все попытки неверны. Суммарное количество очков и оценка дается после выполнения всех пяти задач. Все числа для сложения не должны быть фиксированными, для этого мы будем использовать функцию случайных чисел.

Программа следующая:

```
10 S=0:M=1
20 N=1
30 A=INT(RND(1)*100)
40 B=INT(RND(1)*100)
50 HOME:PRINT:PRINT
60 PRINT "НОМЕР ЗАДАЧИ ";M
70 PRINT A;"+";B;"=";
80 INPUT C
90 PRINT
100 IF C=A+B THEN GOTO 150
110 N=N+1
120 IF N>4 THEN GOTO 160
130 PRINT "ОТВЕТ НЕВЕРЕН"
140 GOTO 50
150 PRINT "ПРАВИЛЬНО !!!"
160 S=S+100-(N-1)*25
170 M=M+1
180 IF M<=5 THEN GOTO 20
190 PRINT:PRINT"ОЧКИ:";S,"ОЦЕНКА:";S/5
200 END
```

Пример 2: Напишем игровую программу "Угадай число". Вам нужно угадать число, загаданное компьютером, за наименьшее количество попыток.

```
10 PRINT "* * УГАДАЙ ЧИСЛО * *"
20 N=1: A=INT(100*RND(1))
```

```
30 PRINT: PRINT "ПОПЫТКА -";N
40 INPUT "ВАШЕ ЧИСЛО";B
50 IF ABS(B-A)>15 THEN PRINT "ХОЛОДНО": GOTO 100
60 IF ABS(B-A)>5 THEN PRINT "ТЕПЛО": GOTO 100
70 IF ABS(B-A)>0 THEN PRINT "ГОРЯЧО": GOTO 100
80 PRINT "ВЫ УГАДАЛИ!!!"
90 PRINT "ВАШИ ОЧКИ=";(10-N)*5: GOTO 120
100 N=N+1
110 GOTO 30
120 END
```

3.3.Операторы *REM*, *CONT* и отладка программы

1.*REM*

Формат оператора *REM*:

[номер строки] ***REM*** **{комментарий}**

Этот оператор используется для вставки в программу комментариев.

2.*STOP*

Формат оператора *STOP*:

[номер строки] ***STOP***

Функция этого оператора - делать паузу в процессе выполнения программы и отображать номер строки, где произошел останов.

Нажатие клавиши **Ctrl+C** эквивалентно оператору ***STOP***. В обоих случаях можно использовать команду ***CONT*** для продолжения выполнения программы. Причем программа начнет выполняться с места, где произошел останов.

Формат оператора *CONT*: [номер строки] ***CONT***

Два оператора ***STOP*** и ***CONT*** могут быть использованы для отладки программы. Когда Вы не уверены, что в программе нет ошибок, вставьте в оператор ***STOP*** в надлежащее место программы и проверьте правильность выполнения программы.

Глава 4. Циклы в программах

Оператор цикла используется для многократного повторения части программы.

4.1.Основные понятия оператора цикла

1.Формат оператора цикла.

[номер строки] ***FOR*** **переменная цикла = выр.1** ***TO*** **выр.2 [*STEP* выр.3]**

тело цикла

[номер строки] ***NEXT*** **[переменная цикла]**

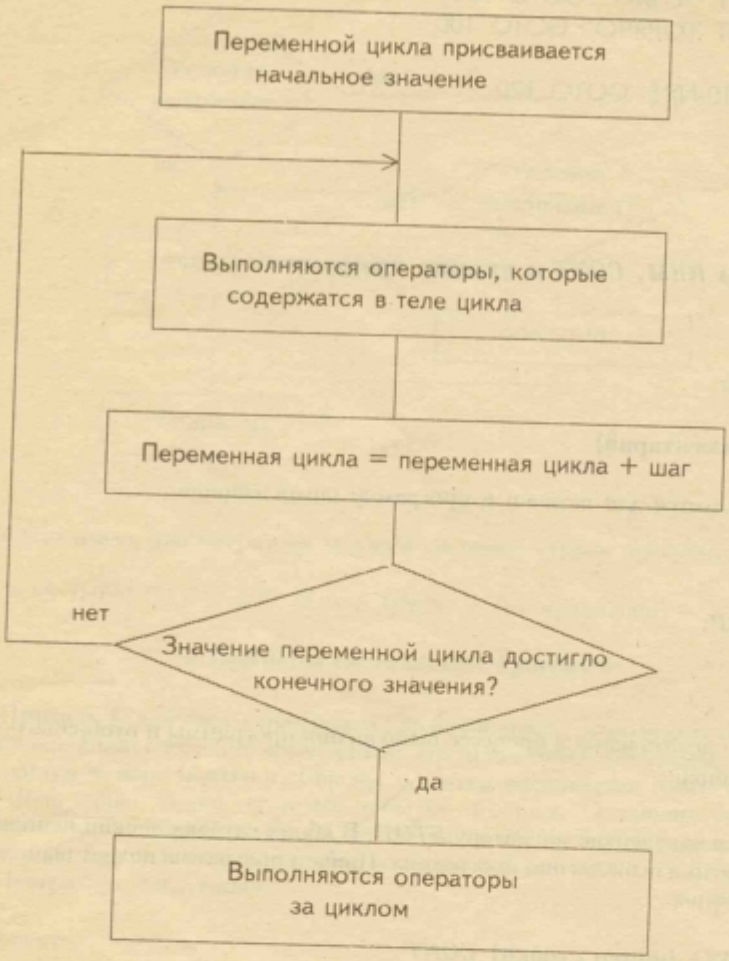
выр.1 - *арифметическое выражение, дающее начальное значение переменной цикла,*

выр.2 - *арифметическое выражение, дающее конечное значение переменной цикла,*

выр.3 - *арифметическое выражение, дающее шаг изменения переменной цикла.*

2.Процесс выполнения оператора цикла.

Этот процесс можно показать на следующей блок-схеме:



1.Переменной цикла присваивается начальное значение и считываются значения конца и шага цикла.

2.Выполняются операторы тела цикла.

3.Переменной цикла присваивается новое значение.

4.Если новое значение переменной цикла не достигло конечного значения, то выполнение цикла повторяется, в противном случае выполняются операторы за циклом.

3.Другие пояснения к циклу

❶ По умолчанию значение шага равно 1.
❷ Значение шага цикла может быть отрицательным.
❸ Значения начала, конца и шага цикла не обязательно константы или целые числа.
❹ Значение переменной цикла не должно изменяться в теле цикла.
❺ В теле цикла значение конца и шага цикла не должно изменяться.
❻ Допускается переход за пределы цикла, но возврат допускается не во всех случаях.
❼ Все виды операторов могут использоваться в теле цикла, или в теле цикла операторов может не быть.

4.Пример программы

Дана последовательность чисел 0,1,1,2,3,5,8, ... , первые два числа равны 0 и 1, все последующие образуются как сумма двух предыдущих чисел.

Напишем программу, которая напечатает эту последовательность чисел, причем количество чисел не должно превышать 100 или последнее число не должно быть больше чем 10^{10}.

```
10 A=0:B=1
20 PRINT A,B,
30 FOR I=3 TO 100
40 C=A+B
50 IF C>1E10 THEN GOTO 90
60 PRINT C,
70 A=B:B=C
80 NEXT I
90 END
```

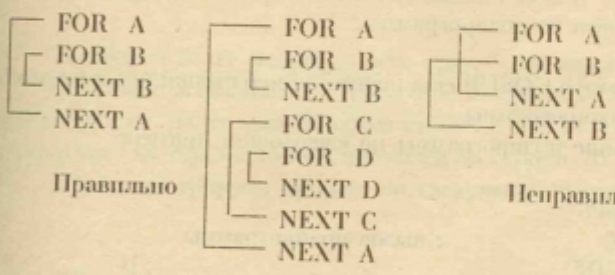
4.2.Циклы в циклах

Во многих практических задачах требуется использование операторов цикла в других операторах цикла. Причем большие циклы могут содержать меньшие, которые могут содержать еще более меньшие и т.д. Максимальное количество циклов, вложенных друг в друга, называется глубиной вложений. Глубина вложений для учебного компьютера SONIC равна 5.

Пример: Вычислим остатки от деления всех чисел от 3 до 100.

```
10 FOR I=3 TO 100
20 FOR J=2 TO I-1
30 PRINT I-INT(I/J)*J
40 NEXT J
50 NEXT I
60 END
```

Операторы цикла не должны пересекаться:



Примечания:

❶ Вложенные циклы не должны пересекаться.

❷ Вложенные циклы не должны иметь одинаковые имена переменных цикла на различных уровнях.

❸ Допускается переход за пределы цикла, но возврат не допускается.

❹ Оператор ***NEXT*** может присутствовать в подоператорах ***THEN***, если там присутствует соответствующий оператор ***FOR***.

4.3.Использование команды *TRACE*

Проблема "мертвых циклов" может возникнуть при выполнении операторов цикла, когда программа выполняется бесконечно долго. Не так легко решить эту проблему. Однако, команда ***TRACE*** может значительно облегчить поиск таких ошибок.

Формат команды *TRACE*:
[номер строки] *TRACE*

Перед началом работы программы введите команду ***TRACE*** для того, чтобы система перешла в состояние трассировки. Затем командой ***RUN*** запустите выполнение программы. Сейчас Вы сможете увидеть на экране все номера строк операторов, которые будут выполняться. Это даст Вам возможность выявить "мертвые циклы", проверяя значения их переменных.

Команда *UNTRACE* используется для выхода из состояния трассировки.

Формат команды *UNTRACE*:
[номер строки] *UNTRACE*

Примечание: Символ "#" используется с номером строки, чтобы их отделять от значен переменных.

Глава 5. Программные модули

5.1.Подпрограммы

Подпрограмма - это независимая часть программы, которая выполняет некотору определенную работу. Используя подпрограммы, программы можно сделать более коротким которые используют меньше памяти. Также это хорошо для упрощения ввода программ и модификации.

Два оператора ***GOSUB*** и ***RETURN*** используются для подпрограмм в языке програ мирования бейсик.

Формат оператора *GOSUB*:
[номер строки] *GOSUB* номер строки

Этот оператор используется для вызова подпрограммы.

Формат оператора *RETURN*:
[номер строки] *RETURN*

Этот оператор используется для возврата из подпрограммы.

Примечание, номер строки после оператора GOSUB есть первый номер строки подпрограмм Оператор RETURN требуется в конце подпрограммы.

Сейчас мы рассмотрим использование подпрограммы на следующем примере:

```
Основная программа    10 PRINT "12345"
                      20 GOSUB 100          ; вызов подпрограммы
                      30 PRINT "ABCDE"
                      40 GOSUB 100
                      50 PRINT 5*5
                      60 GOSUB 100
                      70 END

Подпрограмма          100 PRINT "***************"
                      110 PRINT "???????????????"
                      120 PRINT "^^^^^^^^^^^^^^^"
                      130 PRINT "==============="
                      140 RETURN            ; возвратиться в основную программу
```

Примечание: Эта программа не имеет практического применения, она приведена тольк для того, чтобы показать процесс выполнения подпрограммы.

```
>RUN
12345
***************
???????????????
^^^^^^^^^^^^^^^

===============
ABCDE
***************
???????????????
^^^^^^^^^^^^^^^
```

```
==============
25
**************
??????????????
^^^^^^^^^^^^^^
==============
```

А теперь запустим выполнение программы в режиме трассировки.

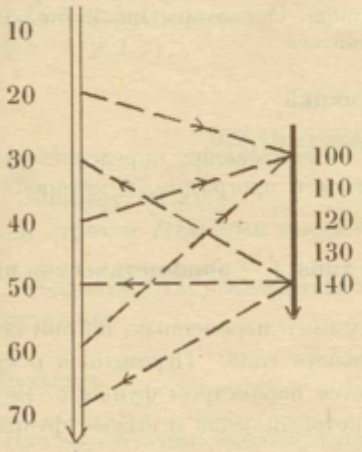
```
>TRACE
>RUN

#10 12345 #20 #100 **************
#110 ??????????????
#120 ^^^^^^^^^^^^^^
#130 ==============
#140 #30 ABCDE #40 #100 **************
#110 ??????????????
#120 ^^^^^^^^^^^^^^
#130 ==============
#140 #50 25 #60 #100 **************
#110 ??????????????
#120 ^^^^^^^^^^^^^^
#130 ==============
#140 #70
```

Символ "#" перед числом показывает номер выполняемой строки.

Разберем более подробно весь процесс выполнения:

В начале выполняется строка под номером 10, затем на строке 20 переход к выполнению подпрограммы, далее выполняются строки с 100 по 130, возвращение к выполнению основной программы на строке 140, и так далее до строки 70.

Проиллюстрируем процесс на следующей схеме:

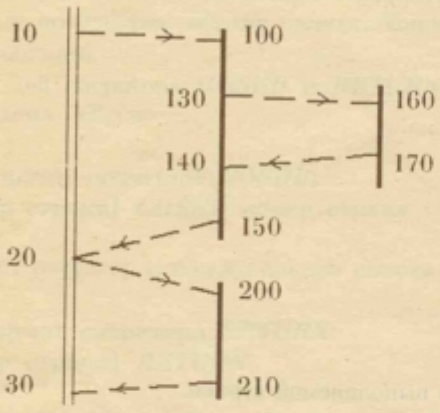


Подпрограммы могут содержать вызов других подпрограмм. Этот случай рассмотрим на следующем примере:

```
10 GOSUB 100
20 GOSUB 200
30 END
100 N=0:S=0
110 INPUT A
```

```
120 IF A=0 THEN 150
130 GOSUB 160
140 GOTO 110
150 RETURN
160 S=S+A:N=N+1
170 RETURN
200 PRINT S,S/N
210 RETURN
```

Эта программа вычисляет и печатает в одной строке сумму и среднее арифметическое всех чисел, введенных с клавиатуры. Процесс выполнения этой программы проиллюстрируем на следующей схеме:



Примечания:

❶ Подпрограммы могут содержать вызов других подпрограмм, причем глубина вложений не должна превышать 10.

❷ Подпрограмма не может прямо или косвенно содержать вызов самой себя.

❸ Не допускается вызов подпрограммы без использования оператора ***GOSUB***. И оператор ***RETURN*** обязательно должен использоваться для выхода из подпрограммы.

❹ Главное сказать, что операторы ***GOSUB*** и ***RETURN*** должны использоваться парами.

❺ Операторы ***GOSUB*** может использоваться в теле цикла. Операторы цикла также могут использоваться в подпрограммах. Но они не могут пересекаться.

5.2.Использование определяемых функций

В языке программирования **бейсик** допускается использование определяемых Вами функций. Эти определяемые функции могут использоваться в программе, в которой Вы их определили.

Формат *DEF FN*:

[номер строки] *DEF FN* имя (переменная реального типа) = арифметическое выраж.

Правила написания имени функции такое же, как у имен переменных, но они не могут совпадать. Значение определяемых функций - число реального типа. Переменная в круглых скобках, следующая за "FN" и именем функции, называется параметром функции. Во время вызова функции параметр получает значение аргумента, который задан в вызове функции.

Пример:

```
10 DEF FN AB(X)=100*X*X+5*X-200
20 PRINT FN AB(5)
30 PRINT FN AB(8)
40 PRINT FN AB(11)
50 END
RUN
2325
6240
11955
```

Примечания:

❶ Использовать определяемую функцию можно только после ее определения.
❷ Только один параметр можно использовать в функции.
❸ Параметр обычно используется в арифметическом выражении, как "X" выше.
❹ Если в арифметическом выражении сделаны синтаксические ошибки, то они обнаружатся при первом вызове функции.

Глава 6. Массивы

В предыдущих главах мы рассмотрели особенности использования переменных.

Эти переменные могут иметь в любой момент выполнения программы только одно значение. Этот вид переменных, как мы знаем, называется простыми переменными.

В этой главе мы рассмотрим другие виды переменных, такие как массивы.

6.1.Основные понятия о массивах

Массивы это один из видов переменных, которые содержат параметры, целые числа, разделенные между собой запятыми.

Параметры не могут быть отрицательными или строкой символов.

Все переменные с параметрами и одинаковым именем формируют массив, причем каждая переменная является элементом массива.

Все переменные в массиве имеют некоторое количество параметров. Это число называется размерностью массива. Максимальная размерность массивов для SONIC REC-9388 равна 3.

Пример:

```
A(1),A(2),.....      - одномерный массив

A(1,1),A(1,2),...
A(2,1),A(2,2),...    - двухмерный массив
A(3,1),A(3,2),...

A(1,1,1),A(1,1,2),...
A(1,2,1),A(1,2,2),...
.....                - трехмерный массив
A(2,1,1),A(2,1,2),...
A(2,2,1),A(2,2,2),...
```

6.2.Оператор размерности массивов

Формат *DIM*:
[номер строки] *DIM* имя массива (целое число1 [,целое число2 [,целое число3]])

Оператор ***DIM*** используется для определения размерности массивов и максимального числа его элементов.

Примечания:

❶ Для определения массива используйте оператор DIM только один раз.
❷ В одном операторе DIM можно определить более чем один массив.
❸ Оператор DIM должен определять массив до его использования, если нет, то по умолчанию максимальное количество элементов массива равно 10.

6.3.Примеры составления программы

Напишем программу, которая моделирует проведение тиража Спортлото 5 из 36.

```
10 REM "ТИРАЖ СПОРТЛОТО"
20 PRINT "ТИРАЖ СПОРТЛОТО 5 ИЗ 36"
```

```
30 DIM K(5)
40 FOR I=1 TO 5
50  K(I)=INT(RND(1)*36)+1
60 IF I=1 THEN GOTO 100
70 FOR J=1 TO I-1
80 IF K(I)=K(J) THEN GOTO 50
90 NEXT J
100 PRINT K(I)
110 NEXT I
120 END
RUN

ТИРАЖ СПОРТЛОТО 5 ИЗ 36
28
4
35
10
20
```

Глава 7. Специальные функции и операторы

В этой главе мы рассмотрим некоторые операторы и специальные функции, также рассмотрим функции для операций над строками. Эти операторы и функции не содержатся в стандартном бейсике.

7.1.Специальные операторы

1.*HOME*

Формат *HOME*:
[номер строки] ***HOME***

Этот оператор очищает экран и ставит курсор в левый верхний угол экрана. Но программа в памяти сохраняется.

Оператор ***HOME*** используется в программах для подготовки экрана для вывода данных.

2.*FRE*

Формат *FRE*:
[номер строки] ***FRE***

Этот оператор покажет, какое количество свободной памяти можно использовать.

3.*POKE*

Формат *POKE*:
[номер строки] ***POKE*** **арифм.выр.1 , выр.2**

Арифм.выр.1 определяет адрес памяти, куда будет записано **выр.2**. Значение **выр.2** должно быть положительным целым числом, не большим чем 255. Функция этого оператора в записи числа непосредственно в память по данному адресу.

Программа, написанная на машинном языке, может быть записана в определенную область памяти, используя оператор ***POKE*** вместе с операторами ***FOR...NEXT,READ*** и ***DATA***.

4.*CALL*

Формат *CALL*:
[номер строки] ***CALL*** **адрес памяти**

Этот оператор запускает программу на машинном языке с данного адреса памяти.

7.2.Специальные функции

1.*TAB*

Формат *TAB*: *TAB*(арифметическое выражение)

Значение арифметического выражения должно принимать значения от 0 до 255. Эта функция используется с оператором ***PRINT*** для передвижения курсора от левой позиции в указанную правую. Значение арифметического выражения не может быть меньше нуля.

Для примера:

```
10 FOR I=5 TO 10
20 PRINT TAB(I);"******"
30 NEXT I
40 END
RUN
    ******
     ******
      ******
       ******
        ******
         ******
          ******
```

2.*PEEK*

Формат *PEEK*: *PEEK*(арифметическое выражение)

Значение арифметического выражения должно быть положительным, так как оно показывает адрес памяти. Значение этой функции есть данные, считанные из памяти по указанному адресу. Действие ***PEEK*** обратно ***POKE*** .

7.3.Функции для операций над строками

1.*LEN*

Формат *LEN*: *LEN*(строковое выражение)

Эта функция выдает длину данной строки символов (т.е.количество символов)

Пример:
```
10 A$="ABC"
20 B$="1234"
30 C$=A$+B$
40 PRINT LEN(A$),
50 PRINT LEN("ABCDEF"),
60 PRINT LEN(C$)
70 PRINT LEN(A$+"D")
80 END
RUN
3     6     7
4
```

2.*STR$*

Формат *STR$*: *STR$*(арифметическое выражение)

Эта функция представляет значение арифметического выражения в виде строки символов.

Пример:
```
10 A=2345
20 B=46
```

```
30 PRINT STR$(A)
40 PRINT STR$(A+B)
50 PRINT STR$(A)+STR$(B)
60 PRINT STR$(10000000000)
70 END
RUN
2345
2391
234546
1E+10
```

3.*VAL*

Формат *VAL*: *VAL*(строковое выражение)

Эта функция переводит строку символов в арифметическое значение.

Пример:

```
10 A$="123"
20 PRINT VAL(A$)
30 PRINT VAL(A$+"55")
40 PRINT VAL("ABC12")
50 PRINT VAL("12AB")
60 PRINT VAL(A$)+VAL("55")
70 PRINT VAL("2E-8")
80 END
RUN
123
12355
0
12
178
2E-08
```

Основное действие этой функции - перевод арифметических констант, заключенных скобки, в число. Если константа, заключенная в скобки, не число, то перевод не буде осуществляться. Функция принимает значение нуля, если первый символ в строке не цифра Однако функция будет просматривать строку слева направо до момента появления символа который не используется для представления числа, и строка символов, которые могут быт восприняты как арифметическая константа, будет переведена в число.(примечание: буква допускается в обозначении чисел)

4.*CHR$*

Формат *CHR$*: *CHR$*(арифметическое выражение)

Результат этой функции - символ, код ASCII которого определяет выражение.

Пример:

```
10 FOR I=0 TO 255
20 PRINT CHR$(I);
30 NEXT
40 END
```

В процессе выполнения этой программы Вы услышите звук, который образуется при печати CHR$(7). Эта программа выводит все символы ASCII (которые можно увидеть).

Все эти символы могут использоваться в программах. Для примера, если вы желаете использовать в строковых данных двойные кавычки, то используйте эту функцию, зная, что у кавычек код ASCII равен 34.

Глава 8. Русские буквы в бейсике

В бейсике русские буквы могут использоваться для ввода и вывода. Русские слова в программах используются главным образом для пояснений и облегчения чтения и обучения. Но русские буквы не допускается использовать в других операциях.

Русские буквы могут использоваться в операторах ***REM***, ***INPUT*** и ***PRINT*** и должны быть заключены в кавычки.

При наборе программ для переключения с русских букв на английские и обратно используйте клавишу **F10**.

Глава 9. Бейсик программы и печать результатов

Учебный компьютер REC-9388 позволяет печатать на принтере программы на бейсике и результаты их выполнения.

Для работы с принтером произведите следующие операции:

❶ Выключите принтер и компьютер.
❷ Подсоедините один конец широкого шнура с разъемом Centronix к принтеру, а другой к компьютеру SONIC в разъем "К принтеру" (см.рис.1)
❸ Включите компьютер и принтер.
❹ Войдите к систему бейсик

Для работы с принтером в бейсике используются следующие команды:

1.***PRON***

Команда ***PRON*** используется для связи компьютера с принтером. При этом все вводимые строки, а также результаты выполнения команд и программ будут одновременно отображаться на экране и печататься на принтере.

2.***PROFF***

Комадна ***PROFF*** отменяет связь компьютера с принтером. При этом вывод будет осуществляться только на экран.

Пример: Если Вы хотите распечатать текст программы
❶ Введите программу
❷ Введите команду ***PRON***
❸ Введите команду ***LIST***
❹ После окончания печати введите команду ***PROFF***

Часть 3. Работа с РВ - кассетой

Описание функций

Эта кассета была разработана и предназначена для учебного компьютера SONIC REC-9388. Оформление ее было специально сделано для учебы и игры детей. РВ-кассета поможет им учиться и стремиться получать знания, а также разовьет их интеллект и способность писать и сочинять музыку. Активно играя и одновременно обучаясь, у ребенка развиваются всевозможные способности.

Эта кассета имеет следующие функции:

1.Сочинение и исполнение музыки:
Эта функция позволяет исполнять различные мелодии, используя клавиатуру компьютера как клавиши музыкального инструмента, или просто введя ноты мелодии. При этом исполняемая музыка имеет хорошую имитацию электронного органа.

2.Тренировка печати на английской печатающей машинке:
Клавиатура у этого учебного компьютера стандартна. Расположение клавиш совершенно такое же, как и у английской печатающей машинки. Когда Вы печатаете, то, что Вы напечатали, отображается на экране, поэтому легко можно контролировать правильность печати. Это хороший инструмент для Вас при обучении машинописи.

3.Арифметическая тренировка

Она дает возможность выполнять арифметические операции и проверять ответы. Это помощь младшим школьникам в освоении арифметических операций.

4.Рисование картинок:

Эта функция есть расширение F-BASIC на PB-кассете. Из 104 небольших рисунков Вы сложить, подобно мазанке, серию различных картин и реализовать всю игру воображения.

5.Программирование в Бейсике:

Эта функция представляет вам возможность работы с F-BASIC. Вы можете научиться и создавать различные прикладные и игровые программы, использующие мультиплика

6.Обучение со звуком и рисунками:

Эта функция кассеты также имеет эффект обучения. Картинки и звуковое сопровож загружаемые с учебных магнитофонных кассет позволяют реализовать многие про обучения.

Более подробно с возможностями PB-кассеты мы познакомимся ниже.

Глава 1.Установка системы "диалог человек - компьютер"

PB-кассета должна быть установлена с системной кассетой. После подсоединения сист кассеты в разъем № 1 и PB-кассеты в разъем № 2 или № 3, включите питание. На экра увидите меню, предлагающее вам выбор. Затем вы вводите номер разъема, где находит кассета, и вы в PB-системе. Вся процедура выглядит в следующем виде:

```
1. TYPING PRACTICE
2. Enter No.2 slot
3. Enter No.3 slot
4. Russ BASIC
5. English BASIC
select: (1-5) ?
DONGDA Electronic CO.LTD.
```

Затем нажмите клавишу 2 для выхода в PB-систему.(Проверьте, чтобы PB-кассета вставлена в разъем № 2)

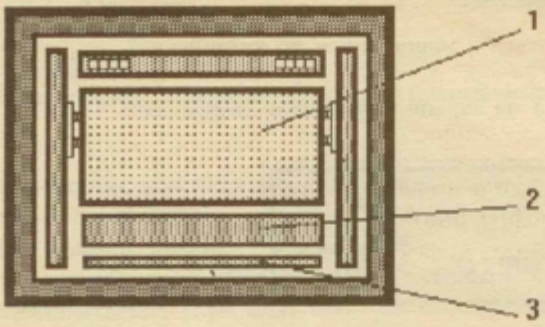
```
SONIC

PB-card


DONGDA Electronic CO.LTD.
```

Нажмите клавишу **RESET**, и на экране отобразится меню:

```
1.Music and Drawing
2.Education with tape
3.Music No.1
4.Music No.2
5.Music No.3
```

Указатель для выбора находится слева от пункта.Вы можете войти в определени функцию, используя клавиши курсора ↑↓ для движения курсора, пока он не будет показыв пункт, который Вы желаете выбрать, и затем нажмите клавишу **C** . Пункты под номерами 4, 5 выполняют три мелодии. Они предложены для вашей оценки. Сейчас выбираем номер раздел обучения музыки и рисования.

На экране появляется рисунок диалога:
Экран разбит на 3 части:



Часть 1 - Отображает содержание компьютерного разговора
Часть 2 - Отображает содержание вашего разговора
Часть 3 - Показывает управляющие клавиши

Нажмите клавишу **Enter** , и система войдет в состояние диалога.

Диалог выглядит следующим образом:

SONIC REC-9388	Соник Рек-9388
COMPUTER SYSTEM	Компьютерная система
NOW WE START	Сейчас мы начнем

Это значит что мы можем начать диалог.

WHO ARE YOU	Кто вы?
INPUT NAME	Введите имя

Кто вы? Пожалуйста, введите ваше имя.

Когда курсор находится в части 2, Вы можете ввести ваше имя по английски. Допустим, что ваше имя "JINDA", пожалуйста, введите "JINDA" и затем нажмите клавишу **Enter**. Разговор отобразится на экране в следующем виде:

```
OK !
JINDA
IS YOU
NOW WE START
```

Если Вы ничего не введете, то Вы должны также нажать клавишу Enter. Далее у Вас на экране появится следующее:

```
"F-BASIC"
NEED ?
```

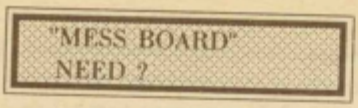
Вы хотите работать с языком программирования F-BASIC ?

```
"CALCULATOR"
NEED ?
```

Вы хотите работать на калькуляторе ?

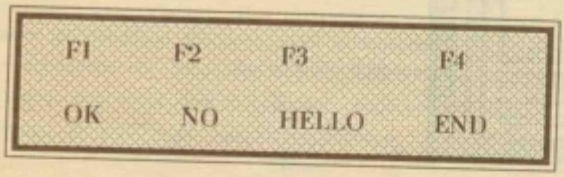
```
"MUSIC BOARD"
NEED ?
```

Вы хотите исполнять музыку ?



Вы хотите потренироваться в машинописи по английски ?

В тоже время в части 3 на экране появляется следующее:



Вы можете нажать клавишу "F1" или ввести "OK", если функция, о которой Вас спрашивают, вам нужна, и нажать клавишу Enter для ее вызова.

Если Вы нажмете клавишу "F2" или введете "NO" и нажмете **Enter**, то система спросит Вас о следующей функции.

Если Вы нажмете "F3" или введете "HELLO" и затем нажмете Enter, то компьютер будет Вас приветствовать на экране.

Если Вы нажмете "F4" или введете "END" и нажмете Enter, то система возвратится на начальную картинку диалога.

Если Вы желаете вызвать необходимую Вам функцию как можно скорее, то введите следующие команды:

"BASIC" - для программирования в Бейсике,
"CAL" - для работы на калькуляторе,
"MUS" - для работы с музыкой,
"MES" - для тренировки набора английских текстов, и нажмите клавишу **Enter** .

Глава 2. Музыкальный редактор: создание и исполнение мелодий

Эта функция может быть использована для обучения и исполения мелодий. При этом мелодию можно исполнять непосредственно, используя клавиатуру компьютера, либо введя нотный текст мелодии на экран компьютера.

Экран для написания мелодий составлен из четырех линий. Причем каждая линия разделена на первый, второй и третий параграф. На каждую линию можно ввести 24 тона (включая паузы). В нижней части экрана размещается описание управляющих клавиш.

Клавиши и ноты

❶ Для исполнения мелодий используйте клавиши **F1-F7**, которые заменяются на экране на c,d,e,f,g,a и b, и соответствуют гамме из 7 нот: до,ре,ми,фа,соль,ля,си.

Клавиши C, D, E, F, G, A и B соответствуют 7 нотам, повышенным на полтона.

❷ Первому параграфу соответствуют высокие октавы, второму - средние и третьему низкие октавы.

❸ Страницы переключаются между собой клавишами **PgUp** и **PgDn**. **PgUp** используется для пролистывания к началу, а **PgDn** - к концу произведения. Всего можно ввести 16 страниц с номерами от 00 до 15.

Когда вы заполнили полностью страницу, то для продолжения ввода текста музыки нажмите клавишу **PgDn** и продолжайте ввод на новой странице.

❹ Чтобы ввести ноту в нужную позицию, указатель можно перемещать клавишами курсора ←→↑↓.

❺ Если Вы желаете, чтобы музыкальное произведение продолжалось на следующей странице, но при этом предыдущая страница заполнена не полностью, то введите знак "#" в конце текста музыки. Для этого поставьте указатель в конце текста музыки на данной странице, причем не важно, в каком параграфе, и нажмите клавиши **Shift** и **3** одновременно.

❻ Символ "$" служит для повторения мелодии. Если Вы желаете, чтобы происходило повторение музыкального произведения, введите в его конце символ "$", для чего нажмите клавиши **Shift** и **4** одновременно.

❼ Для исправлений и улучшений музыки используйте клавиши курсора ←→↑↓, для перемещения указателя на ноту, которую надо исправить. Также используйте клавиши **PgUp** и

PgDn для перелистывания страниц к началу и концу произведения. Если Вы хотите перевести указатель в левый верхний угол, то нажмите клавишу **Home**.

Нажимая клавиши **Del** и **←Backspace** , Вы можете удалить ноту слева от указателя.

Нажав клавишу **Ins** , Вы можете вставить пробел слева от указателя. Нажав один раз, вставится один пробел.

Нажав клавишу **End** , Вы полностью удалите текст на странице.

Для того, чтобы начать исполнение музыкального произведения, которое Вы ввели, нажмите клавишу **F8** .

Музыкальный редактор позволяет исполнять музыкальные произведения с 6-ю различными скоростями исполнения и 8-ю видами ударного сопровождения.

Сопровождение музыки ударными инструментами:

В верхнем левом углу экрана вы можете увидеть символы D0 - D8, которые соответствуют различным видам сопровождения (при этом D0 - нет сопровождения). Нажав одну из клавиш **F9 - F12** или комбинацию клавиши **Shift + F9 - F12** (для этого необходимо нажать клавишу Shift и одну из клавиш F9 - F12 одновременно), Вы можете выбрать необходимый Вам ударный аккомпанемент.

Нажав клавишу **Num Lock**, вы сможете отключить аккомпанемент, то есть установить режим D0.

Изменение скорости исполнения:

В нижней части экрана установлен указатель скорости исполнения,

FAST <------■------> SLOW

что означает **Быстро <--> Тише** . Используя клавиши ←→ во время исполнения музыкального произведения, Вы можете установить необходимую скорость исполнения.

Нажав клавишу **Space** (пробел), Вы сможете остановить исполнение произведения. При этом указатель будет показывать ноту, на которой произошла остановка.

Нажав клавишу **Esc** , вы сможете выйти из музыкального редактора и возвратиться в режим выбора функций системы "диалог человек-компьютер".

Упражнение:

МАЛЕНЬКАЯ ЗВЕЗДА

```
c   c   g   g   a   a   g       f   f   e   e
c g e g c g e g c a f a c g e g   f d f   e c e
c   g   e   g   c   a   g       g   d   g   c

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
d   d   c       g   g   f   f   e   e   d
  d   d c       c g e g   f d f   e c e   d   d
g   b   c g a b       g   b   g   c       g   b

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
g   g   f   f   e   e   d       c   c   g   g
c g e g   f d f   e c e   d   d c g e g c g e g
g   c   g   b   g   c   g   b   c   g   e   g

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
a   a   g       f   f   e   e   d   d   c $
c a f a c g e g   f d f   e c e   d   d c
c   a   g       g   d   g   c   g   b   c

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

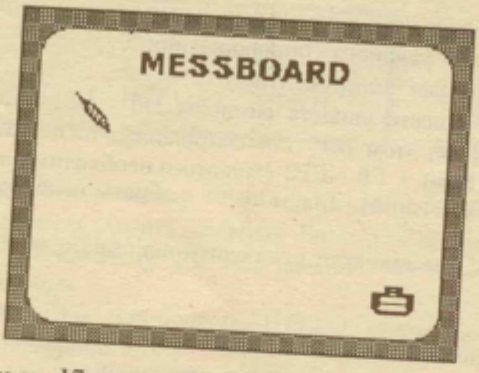
Пожалуйста, введите текст музыкального произведения в музыкальный редактор и по окончании ввода нажмите клавишу F8. Тогда Вы сможете услышать всемирно известное музыкальное произведение.

Глава 3. Тренировка машинописи

*** Функции:**

❶ Тренироваться печатанью на машинке.
❷ Использовать экран для сообщений.

После выбора этой функции, на вашем экране появится следующее:



Экран поделен на 17 строк. В каждую строку можно ввести по 24 символа. Когда экра будет наполнен текстом, указатель остановится в нижнем правом углу экрана и не будет двигаться

Клавишей **Caps Lock** переключаются заглавные и прописные буквы.

Нажмите клавишу **Enter**, чтобы указатель перешел на следующую строку.

Для очистки экрана, нажмите клавишу **End**.

*** Исправления.**

❶ Передвигая указатель клавишами ←→↑↓ , установите его в позицию, где требуется исправление, затем введите правильный символ.
❷ Нажмите клавиши Del или Backspace , чтобы символ слева от указателя был удален.
❸ Нажмите клавишу Ins , чтобы пробел был вставлен слева от указателя.

*** Выход.**

Нажмите клавишу **Esc** , чтобы система вышла из состояния тренировки печати и возвратилась в состояние выбора функции.

Глава 4. Калькулятор

Эта функция может применяться в школах для практического обучения арифметическим операциям.

*** Функции и описание.**

❶ Могут производится операции: "+","-","x",":" или их комбинации.
❷ Операции производятся над числами из 8 знаков в пределах от **0.0000001** до **99999999**.
❸ Если введено число, содержащее более 8 знаков, то система выдаст сообщение, что произошла ошибка: "WRONG".
❹ Если результат вычислений целое число и содержит более 8 знаков, то система выдаст сообщение, что произошло переполнение: "OVERFLOW".
❺ Если результат содержит более 8 знаков, но не целое число, то система будет игнорировать младшие разряды числа.
❻ Десять скобок могут использоваться в арифметическом выражении, но они не могут вкладываться друг в друга. Для примера:

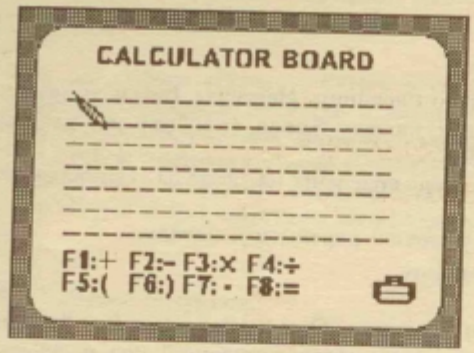
(5+16):(72-59)x(54+8-34)= - правильно

100-(50x(30-15))= - неправильно, скобки вложены друг в друга

❼ Когда система производит вычисления, то она будет следовать следующей очередности выполнения арифметических операций: сначала "()", затем "x" и ":", и операции "+" и "-" в последнюю очередь.
❽ Для получения результата введите в конце выражения знак "=".

*** Управление.**

❶ Вы можете установить систему в состояние калькулятора из состояния выбора функций "диалог человек-компьютер". Затем на экране отобразится следующее:



❷ Результат вычислений может использоваться для дальнейших вычислений.
❸ Нельзя использовать в выражении буквы и другие не предусмотренные символы, иначе на экране отобразится сообщение "WRONG".
❹ Отрицательные числа после символа операции должны быть заключены в скобки.
Для примера: - 3 + - 5 = - неправильно, - 3 + (- 5) = - правильно.
❺ Нажмите **Enter** , чтобы указатель перешел на следующую строку.
❻ Нажмите **Home** , чтобы перевести указатель в верхний левый угол экрана.
❼ Нажмите **End** , чтобы очистить экран.

*** Выход.**

Нажмите клавишу **Esc** , чтобы система вышла из состояния калькулятора и возвратилась в состояние "диалог человек-компьютер".

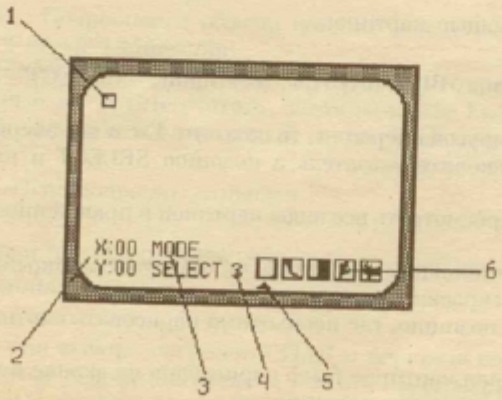
Глава 5. BG-GRAPHIC: Рисование с помощью вспомогательных картинок

BG-GRAPHIC - рисование с помощью картинок значит, что вы можете составлять, подобно мозаике, рисунки с помощью 104 картинок, которые заложены в систему. Эти картинки могут иметь четыре различных вида раскраски. Методика рисования очень проста, и полученные рисунки позволят полностью проявить ваш юмор и воображение.

*** Загрузка системы рисования:**

Когда Вы загрузите систему "диалог человек-компьютер" и выберите F-BASIC на экране отобразится меню.

Нажмите клавишу "2", и система перейдет в состояние BG-GRAPHIC, а на экране отобразится следующее:



⓪ Курсор: Он определяет текущую позицию для рисования. Вы можете передвигать его по экрану клавишами ←→↑↓.

② Значение координат позиции курсора:

Позиция курсора на экране отслеживается автоматически.

Пределы:

абсцисса - X : 00 ~ 27

ордината - Y : 00 ~ 20

③ Отображается состояние системы.

④ Указатель выбора картинки знак "⮝":

Он показывает на текущую выбранную картинку. Нажмите **Ins**, и указатель переместится влево, если нажать **Del**, то вправо. Если нажать **Space** (пробел), то выбранная картинка отобразится на экране в позиции курсора.

⑤ Есть 104 различных вида картинок. Они поделены на разделы. Одновременно для выбора может отображаться 8 картинок.

Нажмите **Home** для переключения разделов картинок к началу.

Нажмите **End** для переключения к концу.

⑥ Код раскраски.

Система допускает 4 вида раскраски картинок. Они имеют коды 0 ~ 3. Нажмите клавишу **Enter** для изменения вида раскраски, при этом соответствующий код раскраски появится на экране.

Выбор операций для рисования

Нажмите клавишу **Esc**, и в левой верхней части экрана появятся следующие операции:

Поставьте указатель с помощью клавиш ↑↓ на нужную операцию. Затем нажмите клавишу **Space** для перевода системы в состояние выбранной вами операции. Текущее состояние системы будет отображено в части 3 экрана.

Функции операций:

* SELECT : Рисование с помощью картинок.
* COPY : Копирование рисунков.
* MOVE : Передвинуть рисунок в определенную позицию.
* CLEAR : Очистить экран.
* FILE : Записать или считать рисунок на магнитофоне.
* CHAR : Использовать символы.

Операции в BG-GRAPHIC

1.**SELECT** - рисование с помощью картинок.

❶ Когда вы входите в функцию BG-GRAPHIC, состояние "SELECT" устанавливается автоматически.

❷ Если вы находитесь в состоянии другой операции, то нажмите Esc и все операции отобразятся в левой верхней части экрана. Установите указатель в позицию SELECT и нажмите клавишу **Enter**.

❸ Нажимая **Home** и **End** , можно просмотреть все виды картинок в правой нижней части экрана для дальнейшего их выбора.

❹ Нажимая **Ins** и **Del** , можно установить указатель "⮝" на нужную картинку.

❺ Нажмите **Enter** для выбора раскраски картинки.

❻ Вы можете установить курсор в позицию, где необходимо нарисовать картинку, при помощи клавиш ←→↑↓.

❼ Нажмите **Space** , чтобы выбранная картинка была нарисована на экране в позиции курсора, при этом курсор передвинется на следующую позицию.

❽ Нажмите клавишу "D", если Вы хотите удалить картинку в позиции курсора.

(Примечание: Удалять картинки с помощью клавиши "D" можно только в состоянии операции SELECT).

2.**COPY** - копирование рисунков.

Операция COPY позволяет копировать существующую уже на экране картинку в другую позицию экрана, при этом картинка в прежней позиции сохраняется. При использовании этой операции процесс рисования рисунков проще и быстрее.

❶ Нажмите **Esc** для выбора операции **COPY** и установки системы в состояние копирования.
❷ Поставьте курсор клавишами ←→↑↓ в позицию, которую необходимо скопировать.
❸ Нажмите **Ins** для подтверждения картинки, которую нужно скопировать.
❹ Клавишами ←→↑↓ поставьте курсор в позицию, в которую нужно скопировать картинку.
❺ Нажмите **Del** для осуществления копирования.
❻ Клавишами ←→↑↓ и **Del**, картинка может сколько угодно раз копироваться в различных позициях экрана.

3.**MOVE** - перенос рисунков.

Операция MOVE позволяет переносить картинки в другую позицию экрана, при этом на прежнем месте картинка не сохраняется.

❶ Нажмите **Esc** для выбора операции MOVE и установки системы в состояние передвижения картинок.
❷ Поставьте курсор клавишами ←→↑↓ в позицию, которую необходимо передвинуть.
❸ Нажмите **Ins** для подтверждения картинки, которую нужно передвинуть.
❹ Клавишами ←→↑↓ поставьте курсор в позицию, в которую нужно передвинуть картинку.
❺ Нажмите **Del** для того, чтобы картинка сохранилась на выбранном месте.
❻ Если вы желаете удалить картинку в состоянии MOVE, нажмите клавишу **Ins** дважды.

4.**CLEAR** - очистить рисунок с экрана.

Операция CLEAR используется для очистки всех картинок, нарисованных на экране. Ее используют для подготовки экрана к рисованию новых рисунков.

Нажмите **Esc**, выберите операцию CLEAR, нажмите **Space** и экран будет очищен, затем система автоматически перейдет в состояние SELECT.

5.**FILE** - сохранение и загрузка рисунков.

Операция FILE позволяет записывать нарисованные картинки на магнитную ленту или считывать записанные картинки с магнитной ленты.

❶ Нажмите клавишу **Esc** для выбора операции FILE и состояния системы записи - считывания рисунков.
❷ На экране появится: SAVE(S), LOAD(L)?
Если Вы желаете записать, введите "**S**" и затем введите имя файла позади приглашения "FILE NAME", далее нажмите клавишу "Запись" на магнитофоне, и после этого нажмите клавишу **Enter** для записи рисунка на магнитофоне.
Примечание: Подробности обмена данных между компьютером и магнитофоном можно узнать из главы 4 части 1 описания.
❸ Если Вы желаете считать картинку, которую вы сохранили ранее, нажмите "**L**" и введите имя файла, который вы хотите считать, затем нажмите Enter и клавишу "Воспроизведение" на магнитофоне, и информация с магнитной ленты будет загружена.

6.**CHAR** - Использование символов.

После вызова операции CHAR, вы сможете вводить все виды символов с клавиатуры в необходимых позициях рисунка.

❶ Нажмите Esc для выбора операции CHAR и перехода системы в состояние ввода символов.
❷ Поставьте курсор в позицию экрана, где необходимо напечатать символ.
❸ Введите нужный символ, как с обычной клавиатуры.
(Примечание: Если символ печатается в позиции, где нарисована картинка, то картинка будет уничтожена).

Глава 6. Программирование в F-BASIC

F-BASIC (*Домашний бейсик*) расширен многими функциями, необходимыми для рисования, в отличии от обычного бейсика, но в тоже время лишен многих математических функций.

Более подробно об обычном бейсике вы можете прочитать во 2 части этого описания. Здесь мы рассмотрим особенности F-BASIC и их употребление.

6.1.Управляющие символы

Управляющие символы подразделяются на три группы, т.е. символы арифметических операций, символы операций сравнения и логические управляющие символы.

Символы арифметических операций:

Управляющий символ	*Значение*	*Пример*	*Выражение*
+	плюс	A + B	A + B
-	минус	A - B	A - B
*	умножить	A * B	A × B
/	деление	A / B	A : B
MOD	остаток от деления	A MOD B	остаток от A : B

Символы операций сравнения:

Управляющий символ	*Значение*	*Пример*	*Выражение*
=	равно	A = B	A = B
<>	неравно	A <> B	A = B
>	больше чем	A > B	A > B
<	меньше чем	A < B	A < B
=>	больше или равно	A => B	A ≥ B
<=	меньше или равно	A <= B	A ≤ B

Логические управляющие символы

Логические управляющие символы используются в логических операциях, для операций в двоичной системе и для операций сравнения.

Управляющий символ	*Значение*	*Пример*	*Выражение*
NOT	не	NOT A	$\overline{A}$
AND	и	A AND B	$A \cap B$
OR	или	A OR B	$A \cup B$
XOR	исключающее или	A XOR B	$(A \cap \overline{B}) \cup (\overline{A} \cap B)$

Приоритет выполнения управляющих символов для трех видов, начиная с наибольшего:

❶ Заключенное в скобки ( )
❷ Функции
❸ * , /
❹ MOD
❺ + , -
❻ = , <> , > , < , => , <=
❼ NOT
❽ AND
❾ OR
❿ XOR

6.2.Вводимые команды

*** CLEAR [CLE.]** - Установить максимальный размер области для размещения программы
Формат: **CLEAR & Hnnnn**

Hnnnn - числовое выражение в шестнадцатиричной системе, значение которого задает максимальный объем памяти для размещения программы
Описание: Для предотвращения конфликта между внутренней памятью и программой, так как в случае длинной программы максимальный объем памяти может быть определен оператором CLEAR. Если в программе оператор CLEAR используется без параметра, то во внутренней памяти очищаются все переменные.
Пример: CLEAR & 7600

Таким образом, мы определили, что внутренняя память до 7600 байт может использоваться только программой. Число "7600" - шестнадцатиричное.

*** NEW** - Очистка всех программ и данных, которые находились во внутренней памяти.
Формат: **NEW**

*** LIST [L.]** - Вывод текста программы на экран
Формат: **LIST {m-n}**
m - начальный номер строки программы
n - конечный номер строки программы
Примечание: Знаки {} показывают, что оператор может использоваться без параметров ***m-n***, но при написании программы их следует опускать.
Описание: Это можно использовать для проверки программы.

LIST m - просмотр строки программы под номером m
LIST -n - просмотр программы от начала до строки n
LIST m- - просмотр программы от строки m до конца
LIST m-n (m<n) - просмотр программы от строки m до строки n
LIST - просмотр программы от начала до конца

*** RUN [R.]** - Запуск программы
Формат: **RUN {n}**

n - номер строки с которой происходит запуск программы;если параметра нет, то с начала программы

*** CONT [C.]** - Продолжить выполнение остановленной программы
Формат: **CONT**
Описание: Программа, прерванная оператором ***STOP***, может быть продолжена при помощи оператора ***CONT***. Когда программа прерывается из-за появления ошибки в процессе работы программы, то она также может быть продолжена при помощи оператора ***CONT***.

*** SAVE [SA.]** - Запись текста программы на магнитную ленту.
Формат: **SAVE {"*Имя файла*"}**
Описание: Так как программы после выключения питания в памяти компьютера не сохраняются, то для их сохранения, необходима запись на магнитную ленту. Каждая программа должна иметь при записи название - "***имя файла***",это поможет Вам при загрузке найти нужную программу.

*** LOAD [LO.]** - Загрузка программы с магнитной ленты в память компьютера.
Формат: **LOAD {"*Имя файла*"}**
Описание: "***Имя файла***" - наименование программы, которое использовалось оператором ***SAVE*** для записи программы на магнитную ленту.Старые программа и данные будут уничтожены после выполнения оператора ***LOAD***.

*** LOAD? [LO.?]** - Проверка программы, которая хранится записанной на магнитной ленте.
Формат: **LOAD? {"*Имя файла*"}**
Описание: Если "***Имя файла***" используется, то проверка будет производится только для записи программы с наименованием "***Имя файла***", другие программы будут пропускаться. Если "***Имя файла***" опущено, то проверяться будет первая встретившаяся запись программы.

*** SYSTEM [S.]** - Выход из системы программирования F-BASIC.
Формат: **SYSTEM**

6.3.Операторы ввода/вывода данных

① = - Присвоение
Формат: ***Имя переменной = присваемое выражение***

Примечание: "***Имя переменной***" может состоять только из английских букв.
Описание: Присвоение означает, что выражение или переменная справа от "=" присваиваются переменной слева от "=". Все переменные справа от "=" должны быть заранее определены. Когда слева от "=" находится числовая переменная, то и справа должно находиться числовое выражение. Если слева символьная переменная, то справа должно быть символьное выражение.

② **PRINT [P.]** - Вывод на экран результатов вычислений программы и выходных данных.
Формат: **PRINT {*Константы, переменные, выражение*}**
Описание: Используя этот оператор можно вывести результаты операций и работы программы, текст, содержащие переменных. Несколько переменных и выражений при выводе на экран необходимо разделять между собой знаками "," или ";".

③ **INPUT [I.]** - Ввод данных с клавиатуры в процессе работы программы.
Формат: **INPUT {" *Подсказка* "} {":" *или* ","} *переменная* {,*переменная, ...*}**
Описание: Когда программа будет выполнять оператор *INPUT* она остановится и будет ждать ввода данных, после ввода она продолжит работу. "***Подсказка***" отображается на экране перед вводимыми данными.

Если переменные отделены знаком ":", то ввод данных будет ожидаться после знака "?" на экране. Если знаком ",", то непосредственно после "***Подсказки***". Когда Вы хотите вводить несколько переменных, то их необходимо разделить знаком "," , при этом вводимые данные необходимо разделять также.

④ **LINPUT [LIN.]** - Ввод строки символов с клавиатуры (включая символы управления курсором).
Формат: **LINPUT {"*Подсказка*"}{":" *или* ","} *символьная переменная***
Описание: Общее между операторами **LINPUT** и **INPUT** в том, что в обоих случаях ввод информации осуществляется через клавиатуру. Различие в том, что "," и "***Подсказка***" в *LINPUT* могут быть использованы как вводимые данные, но при этом длина вводимых данных не должна превышать 31 символа.

⑤ **READ [REA.]** - Оператор ввода данных.
Формат: **READ *переменная* {, *переменная, переменная, ...*}**
Описание: Оператор *DATA* должен находиться в соответствии с *READ*, и количество данных описываемых *DATA* не должно быть меньше, чем количество переменных *READ*. Может произойти ошибка, если данные в *DATA* не будут востановлены оператором *RESTORE*.
READ присваивает данные определенные оператором *DATA* переменным следующим за *READ*.

⑥ **DATA [D.]** - Определение данных для *READ*.
Формат: **DATA *данное* {, *данное, данное, ...*}**
Описание: Соответствие определенных данных с операторами *READ*.

⑦ **RESTORE [RES.]** - Восстановление данных *DATA* для дальнейшего их чтения оператором *READ*.
Формат: **RESTORE {*Номер строки*}**
Описание: Если в операторе RESTORE не задан номер строки, следующий оператор *READ* будет обращаться к первому элементу первого оператора *DATA*. Если в операторе *RESTORE* задан номер строки, то следующий оператор *READ* будет обращаться к первому элементу оператора DATA на указанной строке.

⑧ **SWAP [SW.]** - Оператор обмена содержимого переменных.
Формат: ***SWAP переменная 1, переменная 2***
Описание: Оператор *SWAP* обменивает содержимое двух переменных. Если два оператора разных типов, то обмен не может быть произведен.

⑨ **INKEY$ [INK.]** - Ввод символов с клавиатуры.
Формат: **INKEY$ {n}**
Описание: ***n*** - показывает какое количество символов необходимо ввести. После их ввода программа продолжит работу. В случае, когда параметр ***n*** не используется, *INKEY$* не вызывает ожидания ввода и в случае, когда не нажата никакая клавиша, возвращает пустой символ. (CHR$(0))

Пример 1.
```
10 A=1234        ; присвоим значение 1234 переменной A
20 A$="ABCD"     ; присвоим значение ABCD символьной переменной A$
30 PRINT A       ; напечатаем на экране значение переменной A
40 PRINT A$      ; напечатаем на экране значение переменной A$
50 END           ; конец программы
RUN              ; введите RUN и нажмите клавишу ENTER для запуска программы
```

Результат:

```
1234
ABCD
```

Пример 2.

```
10 INPUT "A=";A            ; Ждем ввода данных . В " " - подсказка.
20 INPUT "B=",B            :Знак ";" значит, что при вводе на экране появится
                           "подсказка" и "?", и будет ожидаться ввод данных.
30 LINPUT "STRING=";A$     ; Знак ";" значит, что когда "подсказка" отобразится на
                           экране, будет ожидаться ввод данных
40 C=A+B
50 D=A-B
60 E=A*B
70 PRINT "A+B=";C
80 PRINT "A-B=";D
90 PRINT "A*B=";E
100 PRINT A$
110 END
RUN
Результат:
A=?  8                     ; Введите значение переменной A и нажмите ENTER
B=  4                      ; "?" может появляться после "подсказки" или нет в
                           зависимости от ";" и ",".
STRING=? 23.5":/ MT        ; Введите строку символов и нажмите ENTER.
A+B=12
A-B=4
A*B=32
STRING=23.5":/MT
```

Пример 3.

```
10 INPUT A,B,C$
20 READ E,D,F,G            ; читаем данные из строки 100 оператора DATA
30 PRINT A                 ; с каждым новым оператором PRINT переменные будут
                           печататься на экране с новой строчки
40 PRINT B
50 PRINT C$
60 PRINT E;D;F;G           ; Знак ";" значит, что значения переменных будут печататься
                           на экране плотно.
70 RESTORE 100             ; Восстанавливаем данные в операторе DATA на 100 строке
80 READ H,I,J,K
90 PRINT H,I,J,K           : "," значит, что значения этих переменных будут печататься
                           раздельно в разных частях..
                           Примечание: Всего на экране 28 колонок. Они делятся на 4
                                       части, а именно 2-8, 9-15,16-22, 23-28
100 DATA 50,60,70,80
110 END
RUN
Результат:
? 5,6,DD                   ; Когда вводится несколько величин, то они разделяются знаком ","
5                          ; Печать идет с новой строки
6
DD
50 60 70 80                ; Печатается плотно
50   60   70   80          ; Печатается раздельно, каждое число в отдельной части
```

Пример 4.

```
10 A$="XXX"
20 B$="YYY"
30 PRINT "A$=";A$,"B$=";B$
40 SWAP A$,B$:PRINT "AFTER SWAP"  : Когда два оператора пишутся в одной строке,
                                    то они разделяются знаком ":"
50 PRINT "A$=";A$,"B$= ";B$
60 END
RUN
```

Результат:

```
A$=XXX     B$=YYY
AFTER SWAP
A$=YYY     B$=XXX
```

6.4.Функции

❶.Числовые функции

* ***ABS [AB.]*** - Абсолютное значение числа.
Формат: ***ABS (X)***
Описание: Результат функции абсолютное значение числа X. X может меняться в пределах от -32768 до +32767.

* ***SNG [X.]*** - Для определения знака числа.
Формат: ***SNG (X)***
Описание: X может меняться в пределах от -32768 до +32767.
Если X > 0, то SNG(X)=1;
если X = 0, то SNG(X)=0;
если X < 0, то SNG(X)=-1.

* ***RND [RN.]*** - Функция случайных чисел.
Формат: ***RND(X)***
Описание: Функция выдает случайное число от 0 до X. X может меняться в пределах от 1 до 32767.

❷Символьные функции

* ***ASC [AS.]*** - Переводит символ в ASCII код.
Формат: ***ASC(символ)***
Описание: Результат функции число - код ASCII для символа.

* ***CHR$ [CH.]*** - Переводит ASCII код в символ
Формат: ***CHR$(X)***
Описание: X - код ASCII символа, его значение от 0 до 255.Функция ASC производит обратное действие к CHR$.

* ***VAL [VA.]*** - Переводит шестнадцатеричное число в десятичное.
Формат: ***VAL(&Hxxxx)***
Описание: Область определения X : &H0000 - &H7FFF

* **STR$ [STR.]** - Переводит десятичное число в шестнадцатеричное.
Формат: ***STR$(X)***
Описание: Область определения X : -32768 - +32767

* **HEX$ [H.]** - Переводит десятичное число в шестнадцатеричный вид.
Формат: ***HEX$(X)***
Описание: Область определения X : -32768 - +32767

* ***LEFT$ [LEF.]*** - Выделяет определенное количество символов слева символьной строки.
Формат: ***LEFT$(строка,n)***
Описание: ***"Строка"*** может содержать не более 31 символа. ***n*** - число выделяемых символов.

* ***RIGHT$ [RI.]*** - Выделяет определенное количество символов справа символьной строки.
Формат: ***RIGHT$(строка,n)***
Описание: ***"Строка"*** может содержать не более 31 символа. ***n*** - число выделяемых символов.

*** MID$ [MI.]**
Функция: Выделяет определенное количество символов с определенного места в символьной строке.
Написание: **MID$(строка,m,n)**
Описание: "Строка" может содержать не более 31 символа.
m - порядковый номер символа, с которого начать выделение.
n - число выделяемых символов.

*** LEN [LE.]**
Функция: Вычисляет длину строки.
Написание: **LEN(строка)**
Описание: Возвращает число - длину "строки" (включая пробелы).

3.Другие функции

*** POS**
Функция: Возвращает номер колонки экрана, где находится курсор.
Написание: **POS(0)**
Описание: Возвращаемое число от 0 до 27.

*** CSRLIN [CSR.]**
Функция: Возвращает число номер строки экрана, где находится курсор.
Написание: **CSRLIN**
Описание: Возвращаемое число от 0 до 23.

*** FRE [FR.]**
Функция: Возвращает число - количество свободной памяти.
Написание: **FRE**
Описание: Знать количество свободной памяти необходимо в особенности в длинных программах, так как может не хватить памяти для размещения данных.

Примеры программ

Пример 1.

```
10 A=50:B=0:C=-50
20  PRINT"ABS(";A;")=";ABS(A)        : печатать абсолютное значение от A
30  PRINT"ABS(";C;")=";ABS(C)        : печатать абсолютное значение от C
40  PRINT"SGN(";A;")=";SGN(A)        : печатать полученные значения от A,B,C
50  PRINT"SGN(";B;")=";SGN(B)
60  PRINT"SGN(";C;")=";SGN(C)
70 INPUT"RND  EXERCISE";R            : введем некоторое значение R
80  PRINT"RND(";R;")=";RND(R)        : вычислим случайное число меньшее R
90 INPUT"ASC  EXERCISE";B$           : введем символ
100  PRINT"ASC(";B$;)=";ASC(B$)      : выведем код ASCII этого символа
110 INPUT"CHR$  EXERCISE";H          : введем число - код ASCII (0-255)
120  PRINT"CHR$(";H;")=";CHR$(H)     : выведем символ, соответствующий этому коду
130 END
RUN
```

Результат:

```
ABS(50)=50
ABS(-50)=50
SGN(50)=1
SGN(0)=0
SGN(-50)=-1
RND EXERCISE? 100          ; введем число - 100
RND(100)=45                ; получили случайное число
ASC EXERCISE? F            ; введем символ "F"
ASC(F)=70                  ; код ASCII для "F"
CHR$ EXERCISE? 80          ; введем код ASCII - 80
CHR$(80)=P                 ; символ "P" соответствует коду 80
```

Пример 2.

```
10 A$="8901"
20 A=VAL(A$)               ; &H8901 - 8901
30 B$=STR$(8901)           ; 8901 - &H8901
40 C$=HEX$(8901)           ; перевод числа 8901 в шестнадцатиричный вид
50 D$=HEX$(A)
60 PRINT A:PRINT B$
70 PRINT C$:PRINT D$
80 END
RUN
```

Результат:
```
8901
8901
22C5
22C5
```

Пример 3.
```
10 A$="PYROMID COMPUTER"
20 PRINT LEFT$(A$,5)          ; печатаем первые 5 символов строки A$
30 PRINT RIGHT$(A$,5)         ; печатаем последние пять символов строки A$
40 PRINT MID$(A$,5,10)        ; печатаем 10 символов, начиная с 5-го
50 PRINT "A$ LONG=";LEN(A$)   ; печатаем длину строки A$
60 END
```

Результат:
```
PYROM
PUTER
MID COMPUT
A$ LONG=16
```

Пример 4.
```
10 CLS
20 FOR I=0 TO 20
30 LOCATE I,I                 ; установим курсор в позицию с координатами I,I
40 PRINT POS(0);",";CSRLIN    ; выведем координаты курсора
50 PAUSE 50
60 NEXT I
70 END
```

6.5.Операторы перехода

*** GOTO [G.]**
Функция: Переход выполнения программы к заданной строке.
Написание: **GOTO N**
Описание: N - номер строки программы, к которой будет переход. После перехода выполнение программы будет осуществляться со строки под номером N.

*** GOSUB [GOS.]**
Функция: Оператор вызова подпрограммы.
Написание: **GOSUB N**
Описание: N - номер строки программы, с которой начинается подпрограмма. Если часть программы часто используется, нет нужды переписывать эту часть несколько раз и лучше оформить ее как подпрограмму. Для возвращения из подпрограммы в основную программу используется оператор RETURN.

*** RETURN [RE.]**
Функция: Возвращение из подпрограммы в основную программу.
Написание: **RETURN {N}**
Описание: N - номер строки программы, куда будет осуществлен возврат. Если номер строки N не используется, то возврат из подпрограммы осуществляется на строку программы следующей за строкой GOSUB.

*** IF ... THEN ... [IF ... T. ...]**
Функция: Оператор сравнения. Выполнение программы разделяется в зависимости от условия.
Написание: **IF выражение THEN номер строки**
Описание: Выражение: логическое выражение истинности.

Номер строки: Номер строки программы, с которой начнется выполнение программы в случае, если логическое выражение истинно.

В случае, если логическое выражение ложно, то выполняться будет следующая за IF THEN строка программы.

IF и THEN должны использоваться одновременно.

*** ON [O.]**
Функция: Передача управления в зависимости от значения переменной.
Написание: **ON переменная {GOTO,GOSUB,RETURN} N1,N2,...**
Описание: Если переменная = 1, то передача управления произойдет на строку N1, если переменная = 2, то на N2 и т.д.
N1,N2,... - номера строк программы, на которые перейдет управление для различных операторов (GOTO,GOSUB,RETURN)

*** STOP [STO.]**
Функция: Оператор останова.
Написание: **STOP**
Описание: Оператор STOP используется для останова выполнения программы. Если Вы желаете продолжить выполнение программы, введите команду CONT.

*** END [E.]**
Функция: Оператор конца программы.
Написание: **END**
Описание: END показывает, что программа закончилась и выполнение ее должно быть прекращено.

Пример 1.

```
10 PRINT "PUSH Y!"
20 A$=INKEY$(0)
30 IF A$<>"Y" THEN BEEP: GOTO 20      ; введем с клавиатуры символ
                                       ; если нажата клавиша не "Y", то
                                       компьютер издаст сигнал и выполнение
                                       программы перейдет на строку 20
40 PRINT:PRINT "PRESS 'Y'"             ; если клавиша-буква "Y", то будет напечатано
                                       "PRESS 'Y'"
50 PRINT "KEY AGAIN!"
60 GOTO 20
70 END
```

Примечание: Остановить выполнение программы можно комбинацией клавиш **Ctrl+Alt+C** .

Пример 2.

```
10 CLS
20 INPUT "KEYIN 1-6"; N
30 ON N GOSUB 100,200,300,400,500,600      ; в зависимости от значения переменной N
                                           вызывается соответствующая подпрограмма
40 IF N<1 OR N>6 THEN 20
50 PRINT N;"=";X$
60 GOTO 20
100 X$="HELLO1":RETURN
200 X$="HELLO2":RETURN
300 X$="HELLO3":RETURN
400 X$="HELLO4":RETURN
500 X$="HELLO5":RETURN
600 X$="HELLO6":RETURN
```

6.6.Оператор цикла

*** FOR ... TO ... STEP ... NEXT [F ... TO ... ST.S.N.]**
Функция: Неоднократное выполнение операторов между FOR и NEXT
Написание: **FOR переменная = m TO n {STEP s} ..... NEXT**
Описание:

❶ m - начальное значение переменной цикла;
n - конечное значение переменной цикла;
s - шаг цикла.

❷ Если STEP опущен, то шаг цикла равен 1.

❸ Каждому FOR должен соответствовать свой NEXT, в противном случае произойдет ошибка с сообщением "NF ERROR".

❹ Циклы могут быть вложенными друг в друга. Если циклы вложены, то оператор NEXT, заканчивающий внутренний цикл, должен появляться раньше, чем оператор NEXT , заканчивающий внешний цикл.

❺ Количество вложений цикла в цикл ограничено лишь размерами внутренней памяти компьютера.

Описание:

❶ N - номер функциональной клавиши, которую необходимо определить.

❷ "Строка" - значение присваемое функциональной клавише. Может содержать до 15 символов.

❸ Функциональные клавиши уже установленные могут быть заново определены.
Эти клавиши следующие:

F1 : LOAD	**F5** : SPRITE
F2 : PRINT	**F6** : CONT
F3 : GOTO	**F7** : LIST
F4 : CHR$(	**F8** : RUN

*** KEYLIST [K.L]**

Функция: Вывод на экран установки функциональных клавиш.
Написание: **KEYLIST**

*** PAUSE [PA.]**

Функция: Оператор паузы. Используется для остановки программы на некоторое время.
Написание: **PAUSE {n}**
Описание:

❶ n : 0 ~ 32767 - время остановки программы.

❷ Если параметр отсутствует, то программа останавливается до первого нажатия клавиши.

6.8.Операторы управления музыкой

*** BEEP [B]**

Функция: Подает звуковой сигнал.
Написание: **BEEP**

*** PLAY [PL.]**

Функция: Исполнение музыки.
Написание: **PLAY "музыкальная информация"**
Примечание: Последовательность символов - музыкальная информация
Описание:

❶ Скорость исполнения определяются T1 ~ T8. T1(быстро) - T8(медленно).

❷ Эффекты звука определяются Y0 ~ Y3. Y0: 12.5%, Y1: 25%, Y2: 75%

❸ Тональность:
M0: определяет полноту V0 ~ V15, V0(малая) - V15(большая).
M1: определяет длительность V0 ~ V15, V0(короткая) - V15(длинная).

❹ Октава - определяется O0 ~ O5, O0(низкая) - O5(высокая).

❺ Ноты - определяются C,D,E,F,G,A и B как показано ниже:

Нота	ДО	ДО#	РЕ	РЕ#	МИ	ФА	ФА#	СОЛЬ	СОЛЬ#	ЛЯ	ЛЯ#	СИ
Символ	C	C#	D	D#	E	F	F#	G	G#	A	A#	B

❻ Если нота соединяется с ":" , то будет выполняться удвоение или утроение тона.

❼ Пауза - длина паузы определяется R0 ~ R9.

❽ Длина - число от 0 до 9 позади ноты, паузы используется для определения длительности ноты, паузы.

Длина	Соответствующее число
1/8 (16-я нота)	0
1/4 (8-я нота)	1
3/8 (3/16 нота)	2
1/2 (четвертная)	3
3/4 (3/8 нота)	4
1 (половинная)	5
3/2 (три четверти)	6
2 (целая)	7
3 (полуторная)	8
4 (двойная)	9

Длина тона соответствует половинной ноте - 1. (C5R1). Если число длительности не указано, то длина тона будет такая же, как и у предыдущей.

Для примера:

```
1) PLAY"CDEFGAB"
2) PLAY"T4Y2M0V15O3C5R5D5R5E5"
3) PLAY"C:E:G:"
```

Примечание:

T-скорость, Y-звуковые эффекты, M-тональность, V-длина и полнота тональности, O-октава, R-пауза.

Для примера исполним мелодию "Для Алисы".

```
00 REM "FOR ALICE"
10  PLAY"M1V10Y2T2:M1V10Y2T8"
20  PLAY"O4E1#DE#DEO3BO4DC:R7"
30  PLAY"O3A3R1CEA:O2R1EAR1R3"
40  PLAY"B3R1E#GB:R1EBR1R3"
50  PLAY"O4C3R1O3EO4E#D:R1EAR1R3"
60  PLAY"E#DEO3BO4DC:R5R3"
70  PLAY"O3A3R1O3CEA:R1EAR1R3"
80  PLAY"B3R1EO4CO3B:R1EBR1R3"
90  PLAY"A3R1R1O4E#D:R1EAR1R3"
100  PLAY"E#DEO3BO4DC:R5R3"
110  PLAY"O3A3R1CEA:R1EAR1R3"
120  PLAY"B3R1E#GB:R1EBR1R3"
130  PLAY"O4C3R1O3EO4E#D:R1EAR1R3"
140  PLAY"E#DEO3BO4DC:R5R3"
150  PLAY"O3A3R1CEA:R1EAR1R3"
160  PLAY"B3R1EO4CO3B:R1EBR1R3"
170  PLAY"A3R1R1R3:R1EABO3CD"
180  PLAY"E1O4EER1R3:R1R3O2G1O3FE"
190  PLAY"O3D1O4FFR1R3:R1R3O2F1O3ED"
200  PLAY"O3C1OREER1R3:R1R3O2E1O3DC"
210  PLAY"O2B1O3EO4ER1O3EE:O2B3R1O2E4"
220  PLAY"O1ER1O3E#GEO4#D:R1ER5"
230  PLAY"EO3#DEO4#DE#D:R3R5"
240  PLAY"E#DEO3BO4DC:R3R5"
250  PLAY"O3A3R1CEA:R1EAR1R3"
260  PLAY"B3R1E#GB:R1EBR1R3"
270  PLAY"O4C3R1O3EO4E#D:R1EAR1R3"
280  PLAY"E#DEO3BO4DC:R3R5"
290  PLAY"O3A3R1CEA:R1EAR1R3"
300  PLAY"B3R1EO4CO3B:R1EBR1R3"
305 IF X=1 THEN 330
310  PLAY"A3R1R1R3:R1EABO3CD"
320 X=1:GOTO 180
330 PLAY"A8"
340 END
```

6.9.Экранный редактор

Для редактирования программ F-BASIC имеет экранный редактор.

❶ Вызовите текст программы, который необходимо исправить, на экран с помощью команды LIST.

❷ Поставьте курсор в позицию, которую необходимо исправить, с помощью клавиш ←→↑↓. Курсор можно передвинуть в любую позицию экрана.

❸ Вставка клавиша **Ins** , удаление - **Del** .

❹ После исправлений в строке нажмите клавишу **Enter** , для того, чтобы исправления были сохранены.
❺ Для удаления строки программы введите номер строки и нажмите **Enter** .
❻ После окончания исправлений поставьте курсор в позицию "OK", затем вы можете вводить другие команды.
❼ Все команды и программы можно вводить только заглавными буквами.

Глава 7. Рисование мультфильмов

Рисование мультфильмов - программа с операторами рисования, содержащихся в F-BASIC. Когда выполняется программа, все виды движущихся картинок могут отображаться на экране и создавать эффект мультфильма.

7.1. Введение

1. Операторы управления отображения

*** LOCATE [LOC.]**
Функция: Устанавливает курсор в определенную позицию экрана.
Написание: **LOCATE X,Y**
Описание:
❶ X - значение абсциссы, может принимать значения от 0 до 27.
Y - значение ординаты, может принимать значения от 0 до 23.
❷ **LOCATE X,Y** устанавливает курсор в позицию X,Y.
❸ Разрешение экрана 28x24.
Пример программы:

```
10 CLS
20 FOR I=0 TO 20
30 LOCATE I,I: PRINT "*"
40 NEXT
50 LOCATE 0,10
60 END
```

*** COLOR [COL.]**
Функция: Установка цвета данной позиции экрана.
Написание: **COLOR X,Y,N**
Описание:
❶ X - абсцисса, значение 0 ~ 27,
Y - ордината, значение 0 ~ 23,
N - код цвета раскраски 0 ~ 3.
❷ Четыре позиции вокруг позиции X,Y приобретут такой же цвет. Также изменить раскраску можно с помощью BG-GRAPHIC и CGSET.
Для примера:

```
10 CLS
20 FOR I=0 TO 447
30 PRINT CHR$(195)
40 NEXT
50 FOR J=0 TO 3
60 COLOR 5+J*3,5+J*2,J
70 NEXT
80 LOCATE 0,20
90 END
```

*** CGEN [CGE.]**
Функция: Установка символов фона и спрайта.
Написание: **CGEN(N)**
Описание:
❶ N - в пределах от 0 до 3
❷ Точно определяет символы, которые отображаются на поверхности фона и которые отображается на поверхности SPRITE.

Пример 1:

```
10 FOR I=1 TO 9
20 FOR J=1 TO 9
30 PRINT I;"*";J;"=";I*J
40 NEXT:NEXT
50 END
```

Пример 2:

```
10 FOR I=1 TO 20
20 FOR J=1 TO I
30 FOR K=1 TO J
40 PRINT "X";
50 NEXT
60 PRINT
70 NEXT
80 PRINT
90 NEXT
100 END
```

6.7.Другие операторы

*** DIM [DI.]**

Функция: Оператор определения массивов.

Написание: **DIM имя массива (m_1{,m_2}), имя массива .....**

Описание:

❶ Несколько матричных переменных могут быть определены в массив оператором DIM.

❷ Число одномерных матричных переменных в массиве m_1+1, т.к. $0 - m_1$.
Двухмерных - $(m_1 + 1)*(m_2 + 1)$, т.к. $m_1 - 0$, $m_2 - 0$.

Пример программы:

```
10 REM THISS IS A F-BASIC PROGRAM
20 DIM A(3),B(3,3)
30 FOR I=0 TO 3
40 A(I)=I
50 PRINT A(I);" ";
60 NEXT
70 PRINT:PRINT
80 FOR I=0 TO 3
90 FOR J=0 TO 3
100 B(I,J)=I*10+J
110 PRINT B(I,J);" ";
120 NEXT:NEXT
130 END
RUN
```

Результат:

```
0 1 2 3
0 1 2 3 10 11 12 13 20 21 22 23 30 31 32 33
```

*** REM**

Функция: Оператор комментариев. Используется для вставки в программу комментариев и пояснений.

Написание: **REM {комментарий}**

Описание:

❶ Желательно использовать комментарии и пояснения к программе, так как это поможет чтению и ознакомлению с программой. REM неисполняемый оператор.

❷ Число символов в комментарии не должно превышать 255.
Пример программы показан выше.

*** KEY [K.]**

Функция: Определение функциональных клавиш.

Написание: **KEY N,"строка"**

❶ Таблицы символов и их установка от N:

Таблица сиволов A	Таблица символов B
Спрайты	Английские символы и картинки

n	Фон	SPRITE
0	A	A
1	A	B
2	B	A
3	B	B

❹ Начальная установка CGEN(2)

Для примера:

```
10 CLS:SPRITE ON:CGSET ,1
20 FOR I=32 TO 255
30 PRINT CHR$(I)
40 NEXT
50 DEF SPRITE 0,(0,1,0,0)
= CHR$(64)+CHR$(65)+CHR$(66)
+CHR$(67)
60 SPRITE 0,100,150
70 PAUSE 100:BEEP
80 CGEN 0
90 PAUSE 100:BEEP
100 CGEN 1
110 PAUSE 100:BEEP
120 CGEN 2
130 PAUSE 100:BEEP
140 CGEN 3
150 END
```

*** CLS [CL.]**

Функция: Очистка экрана

Написание: **CLS**

Описание:

❶ CLS очищает экран и устанавливает курсор в левый верхний угол экрана, но программа в памяти сохраняется.

❷ Если Вы хотите скопировать рисунок из BG-GRAPHIC, то выполните сначала CLS, а затем VIEW.

*** CGSET [CG.]**

Функция: Устанавливает композицию цветов для поверхности фона или спрайтов.

Написание: **CGSET {m},{n}**

m - код раскраски для фона

n - код раскраски для спрайтов

Описание:

❶ Применяют, если цвета фона или спрайта из графической таблицы нужно устанавливать вручную.

❷ 2 вида групп раскрасок приготовлены для фона и 3 вида для спрайтов, причем каждая раскраска в группе нумеруется от 0 до 3.

❸ Спрайты, раскраска которых устанавливается CGSET, определяются операторами DEF SPRITE и DEF MOVE. Код раскраски из группы выбирается кодом (0-3).

❹ Если нарисованная картинка в BG-GRAPHIC копируется на поверхность фона, то m=1.

*** PALET**

Функция: Установка цветов для раскраски.

Написание: **PALET {B,S} n,c1,c2,c3,c4**

Описание:

❶ B - поверхность фона
S - поверхность спрайта
N - код раскраски
c1 - код цвета для фона
c2,c3,c4 - определяют цвет нарисованного

❷ Всего можно получить 52 разновидности раскраски фона и спрайта.

<< 52 вида кода цветов >>

Цвет	шестн.	дес.	шестн.	дес.	шестн.	дес.	шестн.	дес.
	00	0	10	16	20	32	30	48
Синий	01	1	11	17	21	33	31	49
	02	2	12	18	22	34	32	50
	03	3	13	19	23	35	33	51
	04	4	14	20	24	36	34	52
Красный	05	5	15	21	25	37	35	53
	06	6	16	22	26	38	36	54
	07	7	17	23	27	39	37	55
	08	8	18	24	28	40	38	56
Зеленый	09	9	19	25	29	41	39	57
	0A	10	1A	26	2A	42	3A	58
	0B	11	1B	27	2B	43	3B	59
	0C	12	1C	28	2C	44	3C	60
	0E	13	1E	29	2E	45		
	0D	14	1D	30	2D	46		
	0F	15	1F	31	2F	47		

2. Серия операторов движения

*** DEF MOVE [DE.M.]**

Функция: Определение движения спрайта (движущейся картинки).

Написание: **DEF MOVE(N)=SPRITE(A,B,C,D,E,F)**

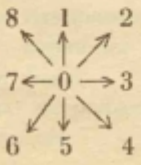
Описание:

❶ N - номер спрайта

❷ A - вид спрайта (0-15)

0:брат Мари	1:Лиза	2:полет	3:летающая рыба
4:пингвин	5:огненный шар	6:машина	7:космическая база
8:спутник-убийца	9:космический корабль	10:взрыв	11:фея
12:лазер	13:черепаха	14:краб	15:птица

❸ B - определяет направление движения (0-8)



❹ C - скорость движения (1-255)

1(быстро) - 255 (медленно), показывает скорость смещения объекта движения на две точки за кадр (приблизительно 1/30 секунды).

Скорость C=60 - 1 точка/сек,а если C=1, то скорость 60 точек/сек

❺ D - количество движений 1-255

Этот параметр определяет дистанцию движения спрайта. Движение спрайта измеряется двойными точками, поэтому общий пройденный путь равен 2*D точек. Если движение по диагонали, то на 2*D и по X и по Y.

❻ E - приоритет отображения (0-1)

0: спрайта

1: фона

❼ F - код раскраски (0-3)

❽ Всего могут быть выбраны для движения 8 видов спрайтов из 16.

*** MOVE [M.]**

Функция: Движение спрайта

Написание: **MOVE N0{,N1,N2,N3,N4,N5,N6,N7}**

Описание:

❶ N0-N7 - номера движущихся спрайтов.

❷ Движущиеся спрайты должны быть определены DEF MOVE.

❸ 8 видов спрайтов могут двигаться одновременно.

❹ Когда выполняется оператор MOVE, спрайты будут контролироваться бейсиком, пока их движение определенное в DEF MOVE не закончится.

*** CUT [CU.]**

Функция: Когда спрайты двигаются при помощи MOVE, их можно остановить в процессе движения.

Написание: **CUT N0{,N1,N2,N3,N4,N5,N6,N7}**

Описание:

❶ N0-N7 - номера движения, определенные оператором DEF MOVE.

❷ Остановит действие MOVE и неподвижный спрайт будет ожидать следующего оператора MOVE.

❸ Одновременно можно определить 8 спрайтов.

❹ В отличии от ERA спрайты не будут исчезать.

*** ERA [ER.]**

Функция: Остановить движение спрайта и сделать, чтобы он исчез.

Написание: **ERA N0{,N1,N2,N3,N4,N5,N6,N7}**

Описание:

❶ N0-N7 - номера движения, определенные оператором DEF MOVE.

❷ Скроет спрайты, начавшие движение оператором MOVE и остановленные оператором CUT.

❸ Одновременно можно определить 8 спрайтов.

*** POSITION [POS.]**

Функция: Установит координаты для движения спрайта.

Написание: **POSITION N,X,Y**

Описание:

❶ N - номер движения (0-7)

X - значении абсциссы (0-255)

Y - значение ординаты (0-255)

(Примечание: Для SPRITE эффективно: X: 0-240, Y: 0-220)

❷ До начала движения спрайта MOVE, нужно задать значения координат начальной позиции с помощью POSITION. Значения координат отсчитываются от верхнего левого угла экрана.

❸ Если не использовать оператор POSITION, то система автоматически установит координаты X=120 Y=120.

Для примера:

```
10 CLS:SPRITE ON
20 DEF MOVE(0)=SPRITE(11,3,2,10,1,2)
30 X=RND(256):Y=RND(240)
40 PRINT "X,Y=";X,Y
50 POSITION 0,X,Y
60 MOVE 0
70 PAUSE 80
80 GOTO 10
90 END
```

*** XPOS [XP.]**

Функция: Вычисляет значение абсциссы до движения спрайта.

Написание: **XPOS(N)**

Описание:

❶ N - номер движения спрайта, определенный DEF MOVE.

❷ Выдает значение абсциссы спрайта для определенного номера движения.

❸ Когда изменяется движение спрайта, то эта функция может использоваться совместно с оператором POSITION.

*** YPOS [YP.]**

Функция: Вычисляет значение ординаты до движения спрайта.

Написание: **YPOS(N)**

Описание:

❶ N - номер движения спрайта, определенный DEF MOVE.

❷ Выдает значение ординаты спрайта для определенного номера движения.

❸ Когда изменяется движение спрайта, то эта функция может использоваться совместно с оператором POSITION.

Для примера:

```
10 CLS:SPRITE ON
20 DEF MOVE(0)=SPRITE(0,2,3,10,0,0)
30 MOVE 0
40 LOCATE 8,20:PRINT " "
50 LOCATE 8,21:PRINT " "
60 LOCATE 0,20:PRINT "XPOS(0)=";XPOS(0)
70 LOCATE 0,21:PRINT "YPOS(0)=";YPOS(0)
80 PAUSE 50
90 GOTO 30
100 END
```

*** MOVE(N) [M.(N)]**

Функция: Определить, закончилось движение или нет.

Написание: **MOVE(N)**

Описание:

❶ N - номер движения.

❷ Если движение не закончилось, то значение MOVE равно -1.

Для примера:

```
10 SPRITE ON:CGSET 1,0
20 DEF MOVE(0)=SPRITE(0,3,1,150,0,0)
30 MOVE(0)
40 IF MOVE(0)=-1 THEN PRINT "MOVE(0)=";
MOVE(0):GOTO 40
50 PRINT "MOVE(0)=";MOVE(0)
60 END
```

3. 3-D команды

*** DEF SPRITE [DE.SP]**

Функция: Определяет отображаемую картинку.

Написание: **DEF SPRITE N,(A,B,C,D,E)=символьное выражение**

Описание:

❶ Символьное выражение может определяться CHR$, символами заключенными в скобки "ABC" или символьными переменными.

❷ N - номер отображаемой картинки (0-7).

❸ A - код раскраски (0-3).

❹ B - тип составления (0-1):

0 - 8x8 точек (эквивалентно одному символу)

1 - 16x16 точек (эквивалентно четырем символам)

❺ C - приоритет отображения (0-1).

0 - приоритет картинки

1 - приоритет фона

❻ D - индикатор направления по X - перевернуто или нет.

0 - не перевернуто

1 - перевернуто

❼ E - индикатор направления по Y - перевернуто или нет.

0 - не перевернуто

1 - перевернуто

Примечания:

❶ DEF SPRITE и DEF MOVE не могут использоваться независимо.

❷ Если используется CGEN, графические и простые символы будут взяты как для SPRITE.

❸ и для функции CHR$(n) в символьной строке берутся из таблицы графических символов.

❹ DEF SPRITE 0,(0,1,0,0,0)=CHR$(68)+CHR$(69)+CHR$(70)+CHR$(71)
можно переписать, как
DEF SPRITE 0,(0,1,0,0,0)="DEFG"

*** SPRITE [SP.]**

Функция: Отображает или удаляет определенные DEF SPRITE картинки в некоторой позиции.

Написание: **SPRITE N,(X,Y)**

Описание:

❶ N - номер картинки (0-7)
X - значение абсциссы (0-255)
Y - значение ординаты (0-255)

❷ Этот оператор отображает картинку в указанной позиции экрана.

❸ Если X и Y опущены, то картинка удаляется с экрана.

❹ Если в некоторой позиции отображаются несколько картинок, то при удалении становится видимой с наименьшим номером.

❺ В одно и тоже время в горизонтальном направлении можно отобразить не более четырех картинок (8 символов).

❻ Отсчет координат ведется от левого верхнего утла экрана.

❼ Соответствие координат BG-GRAPHIC и SPRITE:
Выражения для перевода координат:
X=(x*8)+16
Y=(y*8)+24

где
X,Y - значения координат для SPRITE,
x,y - значения координат для BG-GRAPHIC.

*** SPRITE ON [SP.O.]**

Функция: Значит, что картинки и спрайты будут отображаться на экране.

Написание: **SPRITE ON**

Описание:

❶ Отображение может происходить неоднократно.

❷ SPRITE ON может быть после операторов DEF SPRITE и DEF MOVE.

*** SPRITE OFF [SP.OF.]**

Функция: Остановить отображение спрайтов.

Написание: **SPRITE OFF**

Описание: Оператор уничтожит все картинки и спрайты, отображенные на экране и запретит их дальнейшее отображение.

4.Операторы работы с джойстиком

*** STICK [STI.]**

Функция: Выдает состояние джойстика вверх,вниз,влево,вправо.

Написание: **STICK(N)**

Описание: 1) N - 0 или 1. Значит, джойстик I или II.
2) Коды направления: 8 - вверх, 4 - вниз, 2 - влево, 1 - вправо.

Для примера:

```
10 S=STICK(0)
20 IF S=1 THEN PRINT "RIGHT"
30 IF S=2 THEN PRINT "LEFT"
40 IF S=4 THEN PRINT "DOWN"
50 IF S=8 THEN PRINT "UP"
60 GOTO 10
70 END
```

*** STRIG [STR.]**

Функция: Выдает нажатие клавиш A, B, START, SELECT.

Написание: **STRIG(N)**

Описание:

❶ N - 0 или 1, значит джойстик I или II.

❷ Коды: START - 1,SELECT - 2, A - 4, B - 8.

Пример:

```
10 T=STRIG(0)
20 IF T=1 THEN PRINT "START"
30 IF T=2 THEN PRINT "SELECT"
40 IF T=4 THEN PRINT "A"
50 IF T=8 THEN PRINT "B"
60 GOTO 10
```

5.Другие

*** SCR$ [SC.]**

Функция: Выдает код символа или графики BG-GRAPHIC, отображенных на экране.

Написание: **SCR$(X,Y,S)**

Описание:

❶ X - значение абсциссы (0-27)
Y - значение ординаты (0-23)
S - вычисляем код раскраски или нет (0 или 1, 0 может быть опущен)

❷ При S=1 будет вычислен код раскраски (0-3).

Для примера:

```
10 CLS
20 LOCATE 0,10
30 PRINT "FAMILY-COMPUTER"
40 PRINT "................"
50 LOCATE 10,15
60 PRINT SCR$(0,10)
70 A$=SCR$(1,10)
80 PRINT A$
90 C$=SCR$(1,10,1)
100 PRINT "COLOR=";ASC(C$)
110 END
```

*** VIEW [V.]**

Функция: Вызов BG GRAPHIC.

Написание: **VIEW**

Описание: Когда BG GRAPHIC вызывается при помощи VIEW в бейсике, рисунок нельзя изменить нажатием клавиш или выполнением программы. Эта специальная функция для разрешения рисунка в 3-D в F-BASIC.

7.2.Комбинация между рисованием на BASIC и рисунками BG-GRAPHIC.

*** Метод комбинирования спрайтов с рисунками BG-GRAPHIC.**

❶ Установите систему в состояние BG-GRAPHIC и нарисуйте рисунок.
❷ Нажмите Esc и затем Ctrl+C и установите систему в состояние F-BASIC.
❸ Введите программу и вставьте оператор CLS в строку с наименьшим номером. Затем запустите программу командой RUN и проверьте правильность ее выполнения.
❹ Если программа не допускает ошибок, то выведите текст программы на экран с помощью LIST и поменяйте CLS на VIEW.
❺ Запустите программу командой RUN, и спрайты бейсика будут отображены совместно с рисунком.

*** Предостережение о изменении и исправлении программы.**

❶ Изменять и исправлять бейсик программу можно только после уничтожения BG- GRAPHIC картинки. В противном случае возникает ошибка.
❷ Нажмите End для очистки BG-GRAPHIC картинки.
❸ Затем распечатайте текст программы командой LIST и сделайте необходимые изменения и исправления.

7.3.Пример игровой программы.

```
10 CLS
20 FOR X=1 TO 26
30 LOCATE X,0
40 PRINT CHR$(189)
50 COLOR X,0,2
60 LOCATE X,19
70 PRINT CHR$(188)
80 COLOR X,19,2
90 NEXT
100 FOR Y=0 TO 19
110 FOR X=0 TO 27 STEP 27
120 LOCATE X,Y
130 PRINT CHR$(186)
140 COLOR X,Y,0
150 NEXT:NEXT
160 SPRITE ON
170 DEF SPRITE 1,(0,1,0,1,0)=CHR$(1)+CHR$(3)+CHR$(2)
180 DEF SPRITE 0,(2,1,0,0,0)=CHR$(32)+CHR$(33)+CHR$(34)+CHR$(35)
190 SPRITE 1,84,50:SPRITE 0,160,50
200 FOR X=11 TO 17
210 LOCATE X,4:PRINT CHR$(237)
220 COLOR X,4,2
230 NEXT
240 LOCATE 2,7
250 PRINT"HERO SAVES BEAUTIFUL GIRL"
260 Y=12
270 FOR I=1 TO 3
280 LOCATE 10,Y
290 PRINT"LEVEL:";I
300 Y=Y+1
310 NEXT
320 LOCATE 8,12
330 PRINT CHR$(236):COLOR 8,12,2
340 LE=0:Y=12
350 SG=STRIG(0):PAUSE 5
360 IF SG<>1 AND SG<>2 THEN 350
370 IF SG=2 THEN 380
375 GOTO 440
380 LE=LE+1:LE=LE-3*(LE/3)
390 LOCATE 8,Y:PRINT CHR$(255)
400 IF LE=0 THEN Y=12:GOTO 420
410 Y=Y+1
420 LOCATE 8,Y,2
425 PRINT CHR$(236)
430 COLOR 8,Y,2
440 IF SG<>1 THEN 350
450 CE=3-LE:ED=LE+4
460 LOCATE 8,Y:PRINT CHR$(255)
470 Y=14
480 FOR I=1 TO 3
490 LOCATE 10,Y
500 PRINT "         "          - в кавычках 9 пробелов
510 Y=Y-1
520 NEXT
530 LOCATE 2,7
```

```
540 PRINT "                         "  - в кавычках 25 пробелов
550 FOR X=11 TO 17
560 LOCATE X,4:PRINT CHR$(255)
570 NEXT
580 SPRITE 0:SPRITE 1
590 LOCATE 8,10
600 PRINT "GOOD LUCK!"
610 PAUSE 100
620 LOCATE 8,10
630 PRINT "           "  - в кавычках 11 пробелов
640 FOR Y=4 TO 16 STEP 4
650 FOR X=1 TO 26
660 LOCATE X,Y
670 PRINT CHR$(187):COLOR X,Y,2
680 NEXT:NEXT
690 LOCATE 0,21
700 PRINT "ENEMY"
710 EN=10
720 LOCATE 6,21: PRINT EN
730 FOR I=1 TO 3
740 DEF SPRITE I,(0,1,0,0)=CHR$(0)+CHR$(1)+CHR$(2)+CHR$(3)
750 NEXT
760 SPRITE 3,193,188
770 SPRITE 2,207,188
780 SPRITE 1,221,188
790 SPRITE 0,216,38
800 I=1 TO 3
810 LOCATE 24,I
820 PRINT CHR$(244):COLOR 24,I,1
830 NEXT
840 MM=3:TY=19
850 DIM AE(5),HI(5),FE(5)
860 AE(1)=13:AE(2)=14:AE(3)=3
870 AE(4)=11:AE(5)=5
880 FE(1)=3:FE(2)=2:FE(3)=0
890 FE(4)=2:FE(5)=0
900 HI(1)=164:HI(2)=135:HI(3)=103
910 HI(4)=71:HI(5)=40
920 ML=-1:BM=7:N=1
930 SK=STICK(0):SG=STRIG(0)
940 IF MM=0 THEN 1480
950 IF N=5 AND EN=0 THEN 1200
960 IF ML=-1 THEN GOSUB 2000
970 IF EN>0 AND MOVE(6)=0 THEN GOSUB 2300
980 IF EN=0 AND T=-1 AND N<5 THEN GOSUB 2600
990 IF SK=8 AND EN=-1 AND MOVE(MM)=0 THEN GOSUB 2700
1000 IF SK=1 AND MOVE(MM)=0 THEN BM=3:GOSUB 2100
1010 IF SK=2 AND MOVE(MM)=0 THEN BM=7:GOSUB 2100
1020 IF SG=4 AND MOVE(4)=0 THEN GOSUB 2200
1030 GOSUB 2400
1040 GOTO 930
1200 MX=XPOS(MM)
1205 DEF MOVE(6)=SPRITE(10,0,1,1,0,3)
1210 FOR I=1 TO 4
1220 POSITION 6,204,38
1230 MOVE 6:PAUSE 60
1240 NEXT
```

```
1250 ERA 6
1260 FOR I=1 TO 3
1270 LOCATE 23,I
1280 PRINT CHR$(255)
1290 NEXT
1300 PAUSE 80
1310 LOCATE 20,1:PRINT"THANK"
1320 LOCATE 20,2:PRINT"YOU!!"
1330 PAUSE 100
1340 LOCATE 20,1"PRINT"     "   - в кавычках 5 пробелов
1350 LOCATE 20,2"PRINT"     "   - в кавычках 5 пробелов
1360 SPRITE 0
1370 LS=(216-MX)/4-2
1380 DEF MOVE(0)=SPRITE(1,7,3,LS,0,2)
1390 DEF MOVE(MM)=SPRITE(0,3,3,LS,0,0)
1400 POSITION 0,216,38
1410 POSITION ,MM,MX,HH(N)
1420 MOVE 0,MM
1430 LOCATE 4,9
1440 PRINT"YOU HAVE SAVED LISA"
1450 LOCATE 4,11
1460 PRINT"YOU ARE A TRUE HERO"
1470 GOTO 1500
1480 LOCATE 9,10
1490 PRINT"GAME OVER"
1500 PAUSE 600
1510 SPRITE OFF:CLS
1520 END
2000 MX=120:PAUSE 15
2010 SPRITE MM
2020 DEF MOVE(MM)=SPRITE(0,7,1,3,0,0)
2030 POSITION MM,MX,HH(N)
2040 MOVE MM
2050 ML=1
2060 RETURN
2100 MX=XPOS(MM)
2110 IF MX>218 THEN MX=MX-6
2120 IF MX<23 THEN MX+6
2130 DEF MOVE(MM)=SPRITE(0,BM,1,3,0,0)
2140 POSITION MM,MX,HH(N)
2150 MOVE MM
2160 RETURN
2200 DEF MOVE(4)=SPRITE(12,BM,1,30,0,2)
2210 POSITION 4,XPOS(MM),HH(N)+2
2220 MOVE(4)
2230 RETURN
2300 EX=RND(220)
2310 IF EX<25 OR (EX>100 AND EX<140) THEN 2300
2320 IF EX-XPOS(MM) THEN BE=7:LL=(EX-24)/2:GOTO 2340
2330 BE=3:LL=(219-EX)/2
2340 DEF MOVE(6)=SPRITE(AE(N),BE,CE,LL,0,FE(N))
2350 POSITION 6,EX,HH(N)
2360 MOVE 6
2370 RETURN
2400 IF XPOS(4)>217 OR XPOS(4)<24 OR MOVE(4)=0 THEN ERA 4:
POSITION 4,0,0
2410 DX=XPOS(6)
2420 IF ABS(XPOS(4)-DX)>ED THEN 2495
```

```
2430 ERA 4:POSITION 4,0,0
2440 DEF MOVE(6)=SPRITE(10,0,1,1,0,FE(N))
2450 POSITION 6,DX,HI(N)
2460 MOVE 6:PAUSE 40
2470 ERA 6:PAUSE 40
2480 EN=EN-1
2485 IF EN=0 THEN T=-1
2490 LOCATE 6,21:PRINT " ";EN
2495 XM=XPOS(MM)
2500 IF ABS(XPOS(6)-XM)>3 THEN 2550
2510 DEF MOVE(MM)=SPRITE(0,0,1,1,0,0)
2520 POSITION MM,XM,HI(N)
2530 MOVE MM:PAUSE 80:ERA MM
2540 ML=-1:MM=MM-1:BM=7
2550 RETURN
2600 TX=RND(20)
2610 IF TX<4 THEN 2600
2620 FOR I=1 TO 4
2630 LOCATE TX,TY
2640 PRINT CHR$(210)
2650 TY=TY-1
2660 NEXT
2665 T=1:EN=-1
2670 TD(TX*8)+16
2680 RETURN
2700 XT=XPOS(MM)
2710 IF ABS(TD-XT)>9 THEN 2770
2720 DEF MOVE(MM)=SPRITE(0,1,1,15,0,0)
2730 POSITION MM,XT,HI(N)
2740 MOVE MM:BM=3
2750 EN=10:N=N+1
2760 LOCATE 6,21:PRINT EN
2770 RETURN
```

Примерный сюжет игры: Лиза стала жертвой зла, и брат Мари хочет освободить ее. После того, как он уничтожит демонов, он в конце концов спасет ее. В игре управлять Мари влево или вправо и стрелять нужно с помощью джойстика. Клавиша влево будет двигать Мари влево, клавиша вправо - вправо. Клавиша В - стрельба. После уничтожения первых десяти демонов, появляется следующая ступень лестницы. Когда лестница поднимется, нажмите клавишу вверх, чтобы освободить Лизу.

Глава 8. AV обучение (аудио-видео обучение)

Доказано, что наилучший эффект обучение достигает при помощи картинок и звука. По просьбам обеспечить школьников хорошей помощью в обучении, наша компания "вывела в свет" систему **AV обучения** впервые из всех учебных компьютеров. Истратив значительно меньше средств, Вы получите такой же эффект обучения, как от видеопленок.

После выбора функции аудио-видео обучения, заставка AV обучения появится на экране. Вставьте один конец AV конвектора в компьютер, а другой вставьте в разъем для наушников магнитофона. Вставьте AV кассету в магнитофон. Затем нажмите клавишу ВОСПРОИЗВЕДЕНИЕ, и Вы погрузитесь в океан знаний.

Если Вы хотите использовать AV конвектор, обратите внимание на следующее:

❶ Стереомагнитофон должен хорошо воспроизводить высокие звуки.
❷ Когда слышатся высокие отрывистые звуки, установите переключатель на конвекторе в другую позицию.
❸ Настройте громкость, чтобы индикатор на конвекторе замигал, в то же время установите регулятор тембра в максимальное положение.

❹ Регулятор баланса должен быть установлен в среднее положение, при этом должна быть хорошая картинка и чистый звук.

❺ Если Ваш магнитофон не загружает картинки, то добавьте громкость и подрегулируйте тембр. Если и в этом случае картинка не загружается, то замените магнитофон на более качественный. Когда магнитофонная кассета AV обучения используется продолжительное время, то ее качество ухудшается, и ее необходимо заменить на новую.

Таблица A1. Коды ASCII

Дес.	Шест.	Клавиша	Дес.	Шест.	Клавиша
0	00		32	20	Пробел
1	01	Ctrl+A	33	21	!
2	02	Ctrl+B	34	22	"
3	03	Ctrl+C	35	23	#
4	04	Ctrl+D	36	24	Y
5	05	Ctrl+E	37	25	%
6	06	Ctrl+F	38	26	&
7	07	BEL	39	27	'
8	08	BS	40	28	(
9	09	TAB	41	29	)
10	0A	Ctrl+J	42	2A	*
11	0B	Ctrl+K	43	2B	+
12	0C	Ctrl+L	44	2C	,
13	0D	CR	45	2D	-
14	0E	Ctrl+N	46	2E	.
15	0F	Ctrl+O	47	2F	/
16	10	Ctrl+P	48	30	0
17	11	Ctrl+Q	49	31	1
18	12	Ctrl+R	50	32	2
19	13	Ctrl+S	51	33	3
20	14	Ctrl+T	52	34	4
21	15	Ctrl+U	53	35	5
22	16	Ctrl+V	54	36	6
23	17	Ctrl+W	55	37	7
24	18	Ctrl+X	56	38	8
25	19	Ctrl+Y	57	39	9
26	1A	Ctrl+Z	58	3A	:
27	1B	ESC	59	3B	;
28	1C	→	60	3C	<
29	1D	←	61	3D	=
30	1E	↑	62	3E	>
31	1F	↓	63	3F	?

Таблица А2. Коды ASCII

Дес.	Шест.	Клавиша	Дес.	Шест.	Клавиша
64	40	@	96	60	
65	41	A	97	61	a
66	42	B	98	62	b
67	43	C	99	63	c
68	44	D	100	64	d
69	45	E	101	65	e
70	46	F	102	66	f
71	47	G	103	67	g
72	48	H	104	68	h
73	49	I	105	69	i
74	4A	J	106	6A	j
75	4B	K	107	6B	k
76	4C	L	108	6C	l
77	4D	M	109	6D	m
78	4E	N	110	6E	n
79	4F	O	111	6F	o
80	50	P	112	70	p
81	51	Q	113	71	q
82	52	R	114	72	r
83	53	S	115	73	s
84	54	T	116	74	t
85	55	U	117	75	u
86	56	V	118	76	v
87	57	W	119	77	w
88	58	X	120	78	x
89	59	Y	121	79	y
90	5A	Z	122	7A	z
91	5B	[	123	7B	{
92	5C	\	124	7C	|
93	5D	]	125	7D	}
94	5E	^	126	7E	
95	5F		127	7F	Del

Таблица Б . Функциональные клавиши

Клавиша	Функция
F1	Отображать символы зелеными
F2	Отображать символы красными
F3	Отображать символы синими
F4	Отображать символы розовыми
F5	Курсор зеленый
F6	Курсор красный
F7	Курсор синий
F8	Курсор розовый
F9	Установить символы и курсор в белый цвет
F10	Русский/Английский
F11	Не определена
F12	Не определена
Caps Lock	Переключение между большими и маленькими буквами
Del	Клавиша удаления
Esc	Изменить функцию
Home	Установить курсор в начало экрана
Ins	Клавиша вставки
Ctrl	Управляющая клавиша
Num Lock	Переключение между цифрами и символами

Таблица В . Таблица зарезервированных слов

ABS	EXP	LET	READ	STOP
AND	FN	LIST	REM	STR$
ATN	FOR	LOAD	RESTORE	TAB
CALL	FRE	LOG	RETURN	TAN
CHR$	GOSUB	NEW	RND	THEN
CONT	GOTO	NEXT	RUN	TO
COS	HOME	NOT	SAVE	TRACE
DATA	IF	OR	SGN	UNTRACE
DEF	INPUT	PEEK	SIN	VAL
DIM	INT	POKE	SQR	
END	LEN	PRINT	STEP	

Таблица Г. Стандарт и описание F-BASIC

Пределы чисел	Десятичные: -32768 +32767 Шестнадцатиричные: 0000H - FFFFH Длина строки: 0-31
Длина переменных	В переменной определяющими являются только первые два символа. Первый символ должен быть буквой. Максимальная длина 255 символов.
Пределы нумерации строк	0-65534
Написание более одного оператора на строке	Допускается. Операторы разделяются знаком ":".
Подпрограммы и циклы	Ограничиваются только свободной памятью.
Функции редактирования	Экранный редактор
Состояния экрана	BG GRAPHIC, спрайты, картинки
Разрешение	BG GRAPHIC:28x24, спрайты: 256x240, символы:8x8.
Цвета	Можно сгенерировать до 52 видов раскраски
Управляющие входы	Для ввода направления, выбор/пауза и стрельбы используются джойстики I и II.
Сохранение информации	Информация может записываться и считываться с магнитофона. Скорость передачи 1200 бод.
Создание мультфильмов	Можно выбрать из 16 видов движущихся картинок.

Приложение Д. Кодовая таблица символов. (Используется только для SPRITE)

Код		Картинка	Код		Картинка	Код		Картинка
128	80	Машина	160	A0	Спутник-	192	C0	Краб 1
129	81	влево-	161	A1	убийца	193	C1	
130	82	вверх 1	162	A2	вверх	194	C2	
131	83		163	A3		195	C3	
132	84	Машина	164	A4	Косм.	196	C4	Краб 2
133	85	влево-	165	A5	корабль	197	C5	
134	86	вверх 2	166	A6	влево	198	C6	
135	87		167	A7		199	C7	
136	88	Машина	168	A8	Косм.	200	C8	Птица 1
137	89	вверх 1	169	A9	корабль	201	C9	
138	8A		170	AA	влево-	202	CA	
139	8B		171	AB	вверх	203	CB	
140	8C	Машина	172	AC	Косм.	204	CC	Птица 2
141	8D	вверх 2	173	AD	корабль	205	CD	
142	8E		174	AE	вверх	206	CE	
143	8F		175	AF		207	CF	
144	90	Косм.	176	B0	Взрыв	208	D0	Лазер 1
145	91	станция	177	B1	начало	209	D1	1,2
146	92	1	178	B2		210	D2	Лазер 2
147	93		179	B3		211	D3	1,2
						212	D4	Лазер 3
148	94	Косм.	180	B4	Взрыв	213	D5	1,2
149	95	станция	181	B5	конец			
150	96	2	182	B6		214	D6	Код рас-
151	97		183	B7		215	D7	краски
						216	D8	
152	98	Спутник-	184	B8	Черепаха			
153	99	убийца	185	B9	1	217	D9	Три интер-
154	9A	влево	186	BA		218	DA	вала в
155	9B		187	BB		219	DB	муз.редак
156	9C	Спутник-	188	BC	Черепаха	220	DC1	цель
157	9D	убийца	189	BD	2	221	DD2	цель
158	9E	влево-	190	BE		222	DE1	индикт.
159	9F	вверх	191	BF		223	DF2	индикт.

Приложение Е. Характеристики учебного компьютера SONIC REC-9388

ЦПУ	- 6557 система секам.
	- 6527P система пал.
ОЗУ	- 20 кБ
Объем расширения	- 3 МБ (минимум, если используются три разъема расширения)
Текстовой экран	- 30x32
Разрешение	- 240x256
Цвета	- 64 разновидностей
Яркость	- 8 градаций
Клавиатура	- 88 клавиш (стандартная, русифицированная)
Радио выход на ТВ	- 1 В, сопр.75 Ом
Видеовыход	- макс.3 В
Выход на магнитофон	- 1.5 В
Вход с магнитофона	- 3 В
Игровой интерфейс	- один разъем 60 контактов
Интерфейс расширения	- три разъема по 36 контактов
Управление	- Два джойстика
Питание	- 9 В, 1000 mA
Внешние размеры	- 398x268x68 мм
Масса	- 2 кг
Рабочая температура	- 0 -40 С

Приложение Ж. Таблица сообщений об ошибках в F-BASIC

Если в процессе выполнения программы в F-BASIC произошла ошибка, то на экране отобразится сообщение об ошибке, выполнение программы остановится и компьютер будет ожидать ваших действий. Если вы ввели неправильную команду, то сообщение будет выглядеть в следующем виде:

Пример: ? SN ERROR

Если в процессе выполнения программы:

Пример: ? SN ERROR IN 100

где SN - код ошибки

100 - строка в которой ошибка.

Код ошибки	Сообщение об ошибке
NF	NEXT без FOR
SN	Синтаксическая ошибка
RG	RETURN без GOSUB
IL	Неправильный вызов функции
OV	Переполнение
OM	Нет свободной памяти
UL	Неопределенный номер строки
SO	Большая длина строки
DD	Двойное определение
DZ	Деление на нуль
TM	Несовместимый тип
ST	Большая длина строки
FT	Ошибка формулы
CC	Нельзя продолжить
MO	Неправильный операнд
TP	Ошибка чтения с магнитофона

Приложение 3. Сообщения об ошибках

	0	Сброс
	1	Синтаксическая ошибка
	2	Нет места в памяти
	3	Нельзя продолжить
	4	Буфер строки переполнен
FC	5	Неправильный вызов функции
OV	6	Переполнение
UN	7	Недополнение
IQ	8	Неправильное количество
EC	9	Выражение комплексное
UF	10	Функция неопределена
RD	11	Массив переопределен
/O	12	Деление на нуль
TM	13	Недопустимый тип
LS	14	Слишком длинная строка
ST	15	Оператор слишком длинен
NF	16	NEXT без FOR
FN	17	FOR без NEXT
RG	18	RETURN без GOSUB
OD	19	Кончились данные
BL	20	Плохой номер строки
BS	21	Плохое написание
IS	22	Неправильный оператор
ID	23	Неправильное направление
TP	24	Ошибка чтения/записи на магнитофон

Команды LOGO

DRAW	;Очистка и перевод экрана в режим рисование, установка указателя в начальное положение
FD :X	;Двигать указатель на X шагов вперед
BK :X	;Двигать указатель на X шагов назад
RT :X	;Повернуть указатель на X градусов по часовой стрелке
LT :X	;Повернуть указатель на X градусов против часовой стрелки
HOME	;Переместить указатель в начальное положение
PU	;Указатель оставляет след
PD	;Указатель не оставляет след
HT	;Указатель невидимый
ST	;Указатель видимый
TS	;Определить положение указателя
PICKPEN :N	;Выбрать карандаш как текущий (0-3)
PC :N :X	;Установить цвет
REPEAT	;Повтор команд, заключенных в скобках N раз
FULLSCREEN	;Полный графический экран
TEXTSCREEN	;Полный текстовой экран
SPITSCREEN	;Смешанный графический и текстовой экран
WRAP	;Разрешение движения указателя за пределы экрана
NOWRAP	;Запрещение движения указателя за пределы экрана
ASPECT :X	;Установка осей экрана
CS	;Очистить экран, но курсор не двигать
CLEARTEXT	;Очистить текстовой экран
ND	;Выйти из графики и очистить картинку
BG	;Установить цвет фона
BRIGHT :N :X	;Установить яркость фона и рисунка
SETXY :X :Y	;Передвинуть указатель в точку (X,Y)
SETX :X	;Двигать указатель горизонтально в точку (X,)

```
SETY :Y                 ;Двигать указатель вертикально в точку ( ,Y)
SETH :D                 ;Установить направление указателя
XCOR                    ;Выдать текущую позицию X указателя
YCOR                    ;Выдать текущую позицию Y указателя
HEADING                 ;Выдать текущее направление указателя
TO имя процедуры        ;Режим определения процедуры
END                     ;Конец процедуры
POTS                    ;Отобразить имена процедур
PO имя процедуры        ;Отобразить определенную процедуру
PO                      ;Отобразить последнюю процедуру
POPROCEDURES            ;Отобразить все процедуры
PONAME                  ;Отобразить все переменные
POALL                   ;Отобразить все процедуры и переменные
ER имя процедуры        ;Стереть определенную процедуру
ERNAME "имя переменной  ;Стереть определенную переменную
ERPROCEDURES            ;Стереть все процедуры
ERNAMES                 ;Стереть все переменные
ERALL                   ;Стереть все переменные и процедуры
GOODBY                  ;Стереть все и возвратиться в обычный режим LOGO
IF-THEN-ELSE-           ;Оператор сравнения
NUMBER? :N              ;N число?
WORD? :N                ;N слово?
LIST? :N                ;N список?
ANYOF условие1 условие2   ;Если одно из двух условий истина, то результат истина
ALLOF условие1 условие2   ;Если оба условия истина, то результат истина
NOT условие             ;Если условие ложно, то результат истина
STOP                    ;Остановить выполнение процедуры
OP :X                   ;Прекратить выполнение процедуры и возвратить X вызывавшей
TOP LEVEL       ;Прекратить выполнение процедуры и возвратиться в режим выполнения
GO "M                   ;Оператор перехода на строку M
TEST /IFT /IFF          ;Операторы сравнения
ROUND :X                ;Округление числа X
INTEGER :X              ;Целая часть числа X
SQRT :X                 ;Квадратный корень числа X
QUOTIENT :X :Y          ;Целая часть X/Y
REMAINDER :X :Y         ;Если a и b целые и a*y+b=x, и y больше чем b, то b есть значение этой
                        функции
RANDOM :N               ;Случайное число меньшее N
ASCII "символ           ;код символа
CHAR :N                 ;перевод числа-кода символа в символ
SIN :X                  ;синус X
COS :X                  ;косинус X
ATAN :X                 ;арктангенс X
FIRST "слово            ;первый символ слова
LAST "слово             ;последний символ слова
BF "слово               ;все символы слова кроме первого
BL "слово               ;все символы слова кроме последнего
WORD "слово1 "слово2    ;комбинация двух слов образуют новое слово
FIRST [список]          ;Вывести первое слово списка
LAST [список]           ;Вывести последнее слово списка
BF [список]             ;Стереть первое слово списка
BL [список]             ;Стереть последнее слово списка
SE элемент1 элемент2    ;Формирование списка из комбинации элементов
FPUT                    ;Добавить элемент в начало списка
LIST Элемент1 элемент2        ;формируют список, используя два элемента
LPUT                          ;Добавления элемента в конец списка
PR выр                        ;Печать в разных строках
```

RC?	;Упакованная печать
PRINTI	;Вывести процедуру
CURSOR :X :Y выр	;Установить курсор
THING "A	;Вывести A
THING? "A	;A переменная?
RC	;Ожидание для клавиши в слове
RQ	;Ожидание для клавиши в строке
SAVE "имя файла	;Записать файл
READ "имя файла	;Читать файл
OUTPRN	;Печатать LOGO экран
TRACE	;Трассировка
NOTRACE	;Остановить трассировку

Цвета LOGO:

0: Белый	1:Голубой	2:Синий	3:Бордовый
4: Темно-красный	5:Розовый	6:Красный	7:Оранжевый
8: Желтый	9:Желто-зеленый	10:Зеленый	11:Светло-зеленый
12:Светло-голубой	10:Серый	14:Черный	15:Нет цвета

Подписано в печать 26. 04. 94 г. Печать офсетная. Формат 60x84 1/8.
Объем 10,1 п. л. Тираж 12000. Заказ 107.

Ревдинская типография Управления печати и массовой информации
623270, г. Ревда Свердловской области,
Комсомольская, 51.