COMMODORE

64

Содержание СТР.

Введение

Руководство по использованию микрокомпьютера "Коммодор-64" содержит всю информацию, необходимую для установки аппаратуры, ознакомления с работой микрокомпьютера, и представляет собой начальный курс обучения составлению собственных программ.

Для тех, кто не желает обременять себя учебой в отношении программирования, мы представили всю информацию, необходимую для использования программ фирмы "Коммодор" или других заранее пакетированных программ и/или игровых кассет (программное обеспечение со стороны), практически с самого начала настоящей брошюры. Следовательно, для того чтобы начать работу, Вам нет необходимости искать нужную информацию по всей брошюре.

Теперь рассмотрим некоторые интересные возможности, которые предоставляет Вам микрокомпьютер "Коммодор-64". В отношении графики "Коммодор-64" является наиболее развитым микрокомпьютером. Графические изображения микрокомпьютера мы называем "спрайт-графикой" (SPRITE GRAPHICS). Он позволяет рисовать нужные изображения в четырех различных цветах, аналогично изображениям, которые Вы наблюдаете в видеоиграх аркадного типа. Кроме того, "спрайт-редактор" дает возможность "оживлять" до восьми различных уровней изображения одновременно. Вы можете перемещать свои изображения по всему экрану и даже ставить одно изображение перед другим или за ним. "Коммодор-64" даже обеспечивает автоматическое определение касаний изображений, что дает возможность запрограммировать компьютер на какое-либо действие, когда изображения "ударяются" друг о друга.

Помимо этого, "Коммодор-64" имеет встроенное устройство обеспечения музыкальных и световых эффектов, которое не уступит многим хорошо известным синтезаторам музыки. Эта часть компьютера обеспечивает три независимых голоса, типа фортепиано, в пределах полного девятиоктавного диапазона. Кроме того, компьютер обеспечивает четыре различных формы сигнала (пилообразную, треугольную, импульсную (с изменяющимися параметрами) и шум), программируемый генератор ADSR (нарастание, затухание, выдерживание и отпускание (release), генератор пачки импульсов, программируемые фильтры верхних и нижних частот, полосовые фильтры с регулируемыми резонансом и громкостью звука. Если Вы хотите воспроизвести свою музыку с помощью профессиональных звуковоспроизводящих устройств, "Коммодор-64" дает возможность подключать звуковой выходной сигнал практически к любой усилительной системе высокого класса.

Что касается подключения компьютера "Коммодор-64" к другим устройствам, то по мере роста Ваших потребностей Ваша система может быть расширена с помощью дополнительных устройств, которые называются внешними устройствами. К внешним устройствам относятся такие устройства, как кассетный магнитофон типа C-2N или запоминающее устройство на магнитных дисках типа VIC-1541 (их может быть до пяти), предназначенные для хранения программ, которые вы изготовляете и/или которыми пользуетесь. Если у Вас уже имеется запоминающее устройство на магнитном диске типа

VIC-1540, Ваш поставщик может приспособить его для работы с микрокомпьютером "Коммодор-64". Вы также можете подключить к микрокомпьютеру матрично-точечный принтер фирмы VIC для получения отпечатанных копий Ваших программ, писем, накладных и т.п. Если Вы хотите подключиться к более мощным компьютерам и их обширным базам данных, Вам достаточно вставить кассету VICMODEM, и тогда Вы получите сервис, над которым работали сотни специалистов, а также будете иметь доступ к большому количеству информационных сетей через Ваш домашний или служебный телефон. Наконец, если Вы интересуетесь широким и разнообразным применением программного обеспечения, имеющегося в CP/M*), компьютер "Коммодор-64" может быть снабжен микропроцессором Z-80, подключающимся с помощью разъема.

Представляемое в Ваше распоряжение аппаратное обеспечение совместно с настоящим руководством поможет Вам развить свои знания о компьютерах. Это руководство не даст Вам исчерпывающих знаний о компьютерах, однако познакомит Вас с обширной библиографией в случае необходимости более глубокого изучения указанных тем, фирма "Коммодор" надеется, что Вы получите большое удовлетворение от работы с Вашим новым микрокомпьютером "Коммодор-64". Для того чтобы получить такое удовлетворение, необходимо помнить, что программирование - это не такая вещь, которую можно освоить за один день. Наберитесь терпения и внимательно прочитайте настоящее руководство. Перед тем как запустить "Коммодор-64" в работу, заполните и отошлите регистрационную карточку владельца, бланк которой приложен к Вашему компьютеру. Этим Вы обеспечите надлежащую регистрацию Вашего компьютера в соответствующем бюро фирмы "Коммодор", а также возможность получения самой современной информации относительно будущих усовершенствований Вашего компьютера. Добро пожаловать в увлекательный мир, который откроет Вам Ваш компьютер!

Примечание. В то время как составлялось настоящее руководство, многие программы находились в разработке. Просим Вас связаться с местным представителем фирмы "Коммодор", а также с клубами и журналами пользователей нашей продукции, с тем чтобы Вы могли идти в ногу с развитием прикладных программ, которые пишутся во всем мире для микрокомпьютера "Коммодор-64".

---

*) CP/M - торговая марка фирмы "Диджитал рисерч". Спецификации могут изменяться.

Глава 1. Сборка

Распаковка и подключение микрокомпьютера

Ниже указывается, как следует подключать микрокомпьютер "Коммодор-64" к телевизору или телевизионному монитору и обеспечить нормальную работу всех подключаемых блоков.

Перед подключением к микрокомпьютеру какого-либо устройства проверьте содержимое контейнера, в котором поставляется микрокомпьютер "Коммодор-64". Помимо настоящего руководства, в контейнере должны находиться:

1. Микрокомпьютер "Коммодор-64"
2. Блок питания
3. Видеокабель

Если какой-либо из вышеуказанных приборов отсутствует, немедленно свяжитесь с Вашим поставщиком на предмет замены поставки.

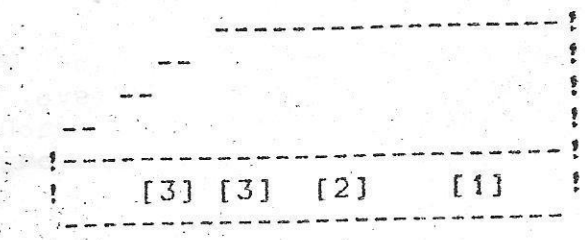
Теперь рассмотрим размещение различных разъемов для подключения компьютера и функции, которые они выполняют.

Разъемы на боковой панели

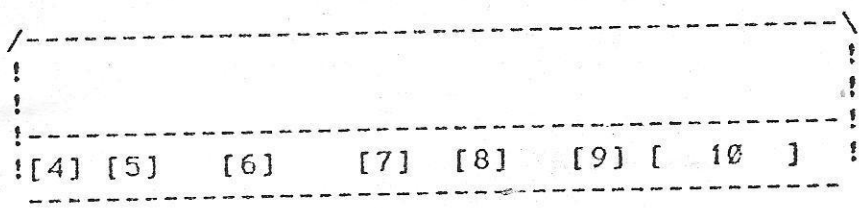
1. РАЗ'ЕМ ПИТАНИЯ (POWER SOCKET). К этому разъему подключается свободный конец кабеля от девятивольтового выхода источника питания, по которому на микрокомпьютер подается питание.
2. ТУМБЛЕР ПИТАНИЯ (POWER SWITCH). Подключает питание к микрокомпьютеру.
3. ПОРТЫ ДЛЯ ИГР (GAME PORTS). В каждый разъем для игр может подключаться координатная ручка управления, игровой рычаг или световое перо. Они подключаются именно сюда.

Разъемы на задней панели

4. ГНЕЗДО ДЛЯ КАССЕТЫ (CARTRIDGE SLOT). Прямоугольное гнездо слева принимает кассеты с программами или игровые кассеты.
5. СЕЛЕКТОР КАНАЛОВ (CHANNEL SELECTOR). Этот переключатель используется для выбора канала, по которому будут выводиться на дисплей изображения, выдаваемые микрокомпьютером.
6. ТЕЛЕВИЗИОННЫЙ РАЗ'ЕМ (TV CONNECTOR). Этот разъем обеспечивает подачу на Ваш телевизор как звука, так и изображения.
7. ВЫХОД ЗВУКА И ИЗОБРАЖЕНИЯ (AUDIO & VIDEO OUTPUT). Этот разъем обеспечивает звуковой сигнал, который может быть подан на высококачественную систему воспроизведения звука, а также "составной" видеосигнал, который может быть подан на телевизионный монитор.
8. ПОСЛЕДОВАТЕЛЬНЫЙ ПОРТ (SERIAL PORT). Через этот разъем Вы можете подключить к микрокомпьютеру "Коммодор-64" печатающее устройство или дисковод.



3 GAME PORTS
2 POWER SWITCH
1 POWER SOCKET



4 CARTRIDGE SLOT
5 CHANNEL SELECTOR
6 TV CONNECTOR
7 AUDIO/VIDEO CONNECTOR
8 SERIAL PORT
9 CASSETTE INTERFACE
1Ø USER PORT

Рис. 1

1 - разъем питания; 2 - тумблер питания; 3 - порты для игр; 4 - гнездо для кассеты; 5 - селектор каналов; 6 - телевизионный разъем; 7 - выход звука и изображения; 8 - последовательный порт; 9 - кассетный интерфейс; 1Ø - порт пользователя; 11 - порт управления; 12 - выключено; 13 - включено

9. КАССЕТНЫЙ ИНТЕРФЕЙС (CASSETTE INTERFACE). К компьютеру может быть подключен кассетный блок типа C2N, что обеспечивает запись на кассету данных и программ для их последующего использования.

1Ø. ПОРТ ПОЛЬЗОВАТЕЛЯ (USER PORT). К порту пользователя можно подключить различные интерфейсные кассеты, такие как VICMODEM или интерфейсную кассету RS-232.

Установка

Подключение компьютера к Вашему телевизору

Подключите компьютер к Вашему телевизору или монитору, как показано на рис.2, выполняя следующие действия.

1. Подключите один конец видеокабеля ко входному разъему телевизионного сигнала, находящемуся на задней панели компью-

тера (на рис.1 этот разъем обозначен цифрой 6). Этот разъем соответствует только одному из штеккеров видеокабеля.

2. Подключите второй штеккер видеокабеля к антенному разъему Вашего телевизора. Штеккер следует только вставить в разъем.

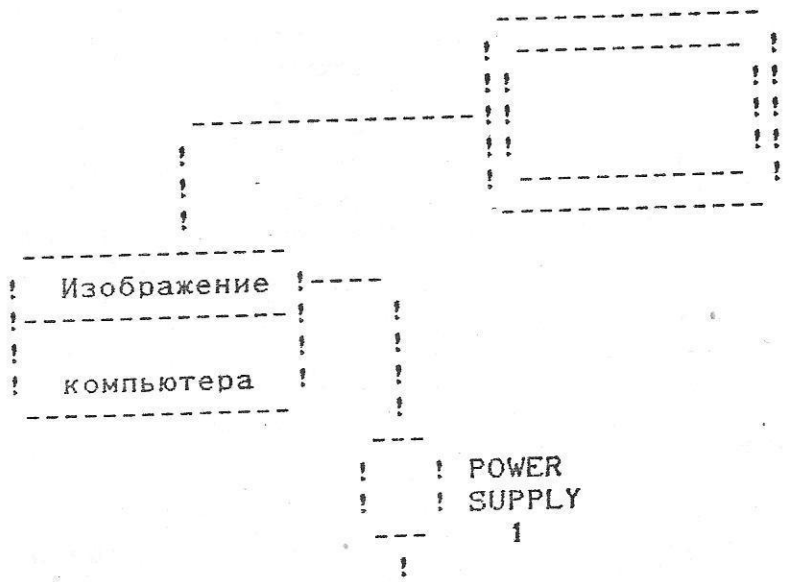


Рис.2

1 - блок питания

3. Теперь подключите блок питания к сети 220 вольт 50 Гц.

4. Убедитесь, что компьютер выключен, и вставьте кабель блока питания в разъем питания на боковой панели компьютера.

5. Включите тумблер питания. При этом должна загореться красная лампочка в правом верхнем углу передней панели компьютера. Если она не загорается, обратитесь к таблице выявления неисправностей, помещенной в конце этого раздела, в которой указаны возможные причины некоторых неисправностей.

6. Включите телевизор и выберите канал, на котором Вы не принимаете никаких программ. (Большинство телевизоров имеет по крайней мере четыре кнопки выбора каналов). Почти наверняка Ваш телевизор не будет настроен на выходной сигнал компьютера. Эту настройку теперь Вы должны осуществить сами.

7. Ваш компьютер "Коммодор-64" "передает" на канале 36. Настраивайте Ваш телевизор, пока на экране Вы не увидите синий прямоугольник в голубой рамке, в верхней части которого будет иметься следующее сообщение:

```
**** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM 38911 BASIC BYTES FREE
```

Перевод: КОММОДОР-64 БЕЙСИК V2
СИСТЕМА ОЗУ НА 64К, 38911 БАЙТ БЕЙСИК СВОБОДНЫ.

Теперь Ваш компьютер подключен правильно и готов к работе.

Дополнительные подключения

Так как "Коммодор-64" обеспечивает звуковой сигнал высокого качества, Вы можете воспроизводить этот сигнал через высококачественный усилитель. Кроме того, компьютер обеспечивает стандартный "составной" видеосигнал, который может быть подан на видеомонитор.

Эти возможности реализуются с помощью разъема "звук/видеосигнал" на задней панели компьютера. Наиболее просто эти сигналы можно получить с помощью стандартного низкочастотного кабеля с пятиштырьковым разъемом (в комплект поставки не входит). Два из четырех штырьков разъема на противоположном конце кабеля несут звуковой и видео сигналы. Кроме того, Вы можете изготовить свой собственный кабель, руководствуясь схемой разводки, показанной в Приложении И.

Обычно ЧЕРНЫЙ (BLACK) разъем кабеля DIN обеспечивает подачу звукового сигнала (AUDIO). Этот штеккер может быть подключен ко входу AUXILIARY (ВСПОМОГАТЕЛЬНЫЙ) усилителя или же к разъему AUDIO IN (ВХОД ЗВУКОВОГО СИГНАЛА) монитора или другой видеосистемы, такой как кассетный видеомагнитофон.

БЕЛЫЙ (WHITE) или КРАСНЫЙ (RED) разъем обычно обеспечивает подачу непосредственно сигнала VIDEO (ВИДЕО). Этот штеккер подключается к разъему VIDEO IN (ВХОД ВИДЕОСИГНАЛА) монитора или ко входу видеосигнала какой-либо другой видеосистемы, например, ко входному разъему кассетного видеомагнитофона.

Цветная кодировка штеккеров кабеля DIN зависит от изготовителя. Применяйте распайку выводов в соответствии с Приложением И для согласования соответствующих соединений, в случае если Вы не можете получить звуковой или видео сигнал с помощью указанных соединений.

Если Вы приобрели внешние устройства, такие как дисковое VIC-1541 или печатающее устройство типа VIC, Вы можете теперь подключить эти устройства к компьютеру. Для правильного подключения этих устройств к компьютеру пользуйтесь прилагаемыми к ним инструкциями.

Работа

Использование компьютера "Коммодор 64"

1. Включите компьютер в сеть с помощью выключателя на левой боковой панели.

2. Через несколько секунд на телевизионном экране появится следующее сообщение:

```
!    **** COMMODORE 64 BASIC V2 ****                !
!64K RAM SYSTEM 38911 BASIC BYTEE FPEE               !
!          1                          2              !
!    READY                      CURSOR SIGNALS       !
!                               COMMODORE 64 IS      !
!                               WAITING FOR YOUR     !
!                               INPUT.               !
```

1 - готово; 2 - курсор сигнализирует о том, что компьютер ожидает входной сигнал

3. Если Ваш телевизор имеет ручку точной настройки, отрегулируйте телевизор на получение четкого изображения.

4. Вы можете отрегулировать цвета и оттенки на экране телевизора. Для достижения лучших результатов Вы можете применить процедуру регулировки цвета, описанную ниже. Когда Вы получаете изображение впервые, экран сначала должен быть темносиним с голубой рамкой и голубыми буквами, цифрами и знаками.

Если Вы не получаете желаемых результатов, проверьте кабели и соединения. Приводимая ниже таблица поможет Вам устранить неисправность.

Таблица 1

Поиск неисправностей

Внешний признак неисправности	Причина	Способ устранения
При включении не загорается индикаторная лампочка	Компьютер не включается	Убедитесь, что выключатель находится в положении ON (ВКЛЮЧЕНО)
	Силовой кабель не вставлен	Проверьте розетку питания - не ослабло ли соединение с вилкой кабеля питания
	Блок питания не подключен к сети	Проверьте правильность подключения к сети блока питания
	В компьютере неисправен плавкий предохранитель	Замените плавкий предохранитель с помощью уполномоченного представителя фирмы
Изображение отсутствует	Телевизор переключен на другой канал	Переключите телевизор на другой канал (3 или 4) для получения изображения на экране
	Неправильное подключение	Компьютер подключен к антенному СВЧ-входу
	Видеокабель не вставлен	Проверьте подключение телевизионного кабеля
	Компьютер установлен на другой канал	Установите компьютер на тот канал, на который установлен телевизор

Таблица 1 (продолжение)

Поиск неисправностей

Отсутствие связного изображения при установленной кассете	Кассета вставлена неправильно	Отключить питание и правильно вставить кассету
Отсутствие цвета в изображении	Плохая настройка телевизора	Правильно настроить телевизор
Плохой цвет изображения	Плохая цветовая регулировка телевизора	Отрегулировать в телевизоре цвет/оттенок/яркость
Изображение имеет чрезмерный шумовой фон	Слишком большое усиление звука в телевизоре	Отрегулировать уровень усиления звукового сигнала в телевизоре
Изображение хорошее, но звук отсутствует	Усиление звука в телевизоре недостаточно	Отрегулировать уровень усиления звукового сигнала в телевизоре
	Неправильное подключение к вспомогательному входу	Подключите штеккер со звуковым сигналом к вспомогательному входу усилителя и осуществите переключение на этот вспомогательный вход

Примечание. Компьютер "Коммодор-64" был разработан с тем расчетом, чтобы им мог пользоваться каждый. Однако, мы понимаем, что пользователи могут сталкиваться с трудностями. Чтобы ответить на возможные вопросы пользователей и дать им советы в отношении программирования, фирма "Коммодор" выпустила несколько публикаций. Пользователи могут также вступить в клуб пользователей компьютером "Коммодор", что поможет им вступать в контакт с другими пользователями компьютером "Коммодор-64" и обмениваться идеями и опытом.

Курсор

Курсором называется мигающий квадратик, находящийся рядом со словом READY (ГОТОВО). Курсор указывает место, где на экране

будет появляться то, что Вы набираете на клавиатуре. Когда Вы нажимаете на клавишу, курсор будет продвигаться вперед на один шаг, а на том месте, где курсор находился один шаг тому назад, появляется знак, набранный на клавиатуре. Попробуйте набирать знаки на клавиатуре и посмотрите, как знаки, которые Вы набираете на клавиатуре (печатаете) появляются на экране телевизора.

Регулировка цвета

Существует простой способ получения на экране телевизора стандартного набора цветов для регулировки телевизора. Даже если Вы сейчас не знакомы с принципами работы компьютера, следуйте нашим указаниям, и Вы увидите, как легко пользоваться компьютером "Коммодор-64".

Сначала посмотрите на левую часть клавиатуры и найдите клавишу, обозначенную буквами CTRL. Эти буквы обозначают слово CONTROL (УПРАВЛЕНИЕ). Эта клавиша используется в сочетании с другими клавишами для выдачи компьютеру команды на выполнение того или иного задания.

Чтобы реализовать функцию управления, нажмите клавишу CTRL и, удерживая ее, нажмите другую клавишу.

Попробуйте осуществить такое действие: нажмите клавишу CTRL, нажав также клавишу 9. Затем отпустите обе клавиши. Внешне ничего не произойдет, но если теперь Вы дотронетесь до любой клавиши, на экране появится соответствующий знак, изображенный не в нормальном, а в обратном виде, например, первоначальное сообщение или еще что-либо, напечатанное Вами ранее.

Нажмите и удерживайте в нажатом состоянии клавишу SPACEBAR (ПЛАНКА ПРОБЕЛА). Что при этом произойдет? Если Вы правильно выполнили вышеописанную процедуру, Вы увидите, как голубая полоска движется по экрану и затем перемещается вниз, к следующей строке, пока нажата клавиша SPACEBAR.

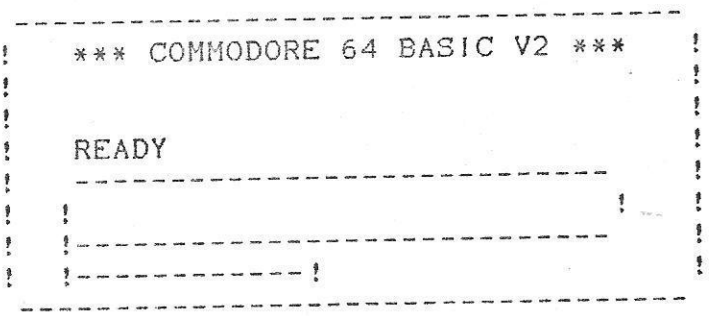


Рис.4

Теперь нажмите и удерживайте клавишу CTRL , нажимая любую из других цифровых клавиш. Каждая из этих клавиш имеет спереди цветную маркировку. С этого момента все, что будет выводиться на дисплей, будет изображаться в этом цвете. Например, нажав и удерживая клавишу CTRL , нажмите клавишу 8 , после чего отпустите обе клавиши. Теперь нажмите SPACEBAR (ПЛАНКА

ПРОБЕЛА).

Посмотрите на дисплей. Теперь на экране - полоска желтого цвета! Аналогичным образом Вы можете управлять цветом полоски, выбирая этот цвет из набора, имеющегося на цифровых клавишах, с помощью нажатия клавиши CTRL и соответствующей клавиши.

Еще несколько раз смените цвет полоски, а затем с помощью регуляторов цвета, оттенка на Вашем телевизоре отрегулируйте цвет изображения таким образом, чтобы он соответствовал тому цвету, который Вы выбираете.

На дисплее будет иметь место примерно такая картинка:

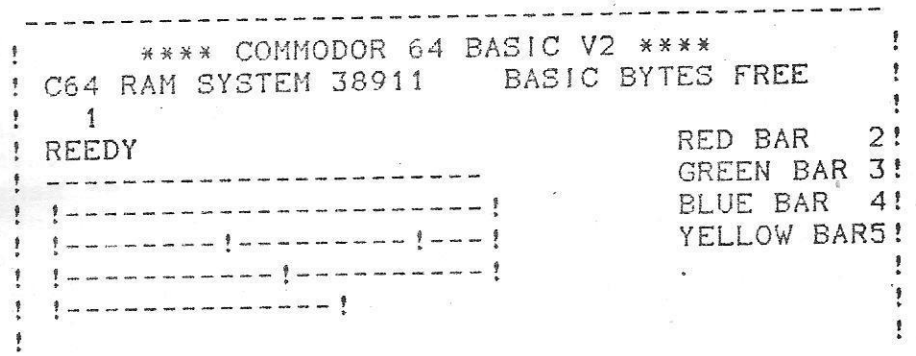


Рис.5

1 - готово; 2 - красная полоска; 3 - зеленая полоска; 4 - синяя полоска; 5 - желтая полоска

Теперь все нормально отрегулировано и работает правильно. В следующих главах настоящего руководства Вы познакомитесь с языком БЕЙСИК. Однако, Вы можете немедленно начать работу с компьютером, используя некоторые из многих заранее разработанных применений и игр, которые можно реализовать на компьютере "Коммодор-64", не имея никакого понятия о программировании компьютера.

Каждый из этих пакетов содержит подробную информацию о том, как следует использовать программу. Тем не менее мы рекомендуем Вам прочитать несколько первых глав настоящего руководства, для того чтобы познакомиться с основами работы Вашей новой системы.

Глава 2. Запуск

Клавиатура

Теперь, когда Ваша система установлена и отрегулирована, рекомендуем Вам затратить несколько минут на ознакомление с клавиатурой, так как клавиатура в компьютере "Коммодор-64" является самым важным средством коммуникации.

Вы увидите, что эта клавиатура во многом напоминает стан-

дартную клавиатуру пишущей машинки. Однако, в ней имеется ряд новых клавиш, с помощью которых осуществляется управление выполнением специальных функций. Ниже следует краткое описание различных клавиш и функций, которые они осуществляют. В последующих разделах подробно описывается работа каждой клавиши.

RETURN (ВОЗВРАТ)

Клавиша RETURN (ВОЗВРАТ) дает компьютеру команду просмотреть информацию, которую Вы набрали на клавиатуре, и вводит эту информацию в память компьютера.

SHIFT (СДВИГ)

Клавиша SHIFT (СДВИГ) работает так же, как и на стандартной пишущей машинке. Многие клавиши дают возможность выводить на дисплей две буквы или символа и два графических знака. В режиме "верхний/нижний регистр" клавиша SHIFT дает стандартные знаки верхнего регистра. В режиме "верхний регистр/графика" клавиша SHIFT выводит на дисплей графические знаки, показанные на правой стороне клавиши.

В регистре специальных функциональных клавиш клавиша SHIFT реализует функцию, обозначенную на нижней части клавиши.

Редактирование

Ничего совершенного нет, и "Коммодор-64" учитывает это. Ряд редактирующих клавиш позволяет корректировать ошибки печати и перемещать информацию по экрану.

CRSR (КУРСОР)

Имеются две клавиши с маркировкой CRSR (КУРСОР): одна со стрелками вверх/вниз CRSR , другая - со стрелками вправо/влево CRSR. Эти клавиши используются для перемещения курсора вверх и вниз или вправо и влево. В режиме "без сдвига" клавиши CRSR (КУРСОР) позволяют перемещать курсор вниз и вправо. При использовании клавиши SHIFT (СДВИГ) клавиши CRSR (КУРСОР) курсор будет перемещаться вверх и влево. Клавиши КУРСОР имеют специальную характеристику повтора: при нажатии клавиши курсор перемещается в автоматическом режиме до тех пор, пока клавиша не будет отпущена.

INST/DEL (ВСТАВЛЕНИЕ/СТИРАНИЕ ЗНАКОВ)

Если Вы нажмете клавишу INST/DEL (ВСТАВЛЕНИЕ/СТИРАНИЕ ЗНАКОВ), курсор переместится на один шаг назад, стирая предыдущий напечатанный знак. Если Вы находитесь в середине строки, предыдущий знак слева исчезает, а знаки справа автоматически сдвигаются влево, заполняя появившийся пробел.

Нажатие клавиши INST/DEL при нажатой клавише SHIFT (СДВИГ) позволяет вставить в строку дополнительную информацию. Например, если Вы заметили ошибку при наборе с клавиатуры в начале строки - допустим, Вы пропустили часть имени - Вы можете использовать клавишу <CRSR> для возвращения к ошибке и затем нажать клавишу INST/DEL для вставления пробела. Затем необходимо нажать на клавиатуре пропущенную букву.

CLR/HOME (ОЧИСТКА/ИСХОДНОЕ ПОЛОЖЕНИЕ)

Нажатие клавиши CLR/HOME (ОЧИСТКА/ИСХОДНОЕ ПОЛОЖЕНИЕ) устанавливает курсор в исходное положение (HOME) на экране, т.е. в левый верхний угол экрана. Нажатие этой же клавиши при нажатой клавише SHIFT (СДВИГ) очищает экран и помещает курсор в исходное положение.

RESTORE (ВОССТАНОВЛЕНИЕ)

Принцип работы клавиши RESTORE (ВОССТАНОВЛЕНИЕ) показан в ее названии. Нажатие этой клавиши возвращает компьютер в нормальное состояние, в котором он находился до того, как Вы изменили положение дел с помощью программы или какой-либо команды. Подробно об этом будет рассказано в последующих главах.

Функциональные клавиши

Четыре функциональные клавиши в правой части клавиатуры могут быть запрограммированы на осуществление большого количества функций. С их помощью многими способами можно выполнять повторные задачи.

CTRL (УПРАВЛЕНИЕ)

Клавиша CTRL (УПРАВЛЕНИЕ) позволяет устанавливать цвет и реализовывать другие специальные функции. Для этого Вы нажимаете и удерживаете клавишу CTRL, нажимая в то же время другую клавишу для реализации функции управления. Вы уже нажимали клавишу CTRL, когда Вы меняли цвет изображения, для получения разноцветных полос во время процесса установки компьютера.

RUN/STOP (ПРОГОН/ОСТАНОВ)

Обычно нажатие этой клавиши останавливает выполнение программы на языке БЕЙСИК. Нажатие этой клавиши дает компьютеру команду остановить (STOP) выполнение чего-либо. Нажатие клавиши RUN/STOP (ПРОГОН/ОСТАНОВ) в режиме с одновременным нажатием клавиши SHIFT (СДВИГ) позволяет осуществить автоматическую загрузку программы с ленты.

Клавиша C= фирмы "Коммодор"

Клавиша C= выполняет ряд функций. Во-первых, она позволяет осуществить переход от режима вывода текста к режиму вывода графики и обратно.

Когда компьютер включается впервые, то он находится в режиме "ВЕРХНИЙ РЕГИСТР/ГРАФИКА", т.е. при печати на экран будут выводиться знаки верхнего регистра. Как было указано ранее, нажатие клавиши SHIFT (СДВИГ) в этом режиме выводит на дисплей графику, указанную на правой стороне клавиши.

Если Вы нажали клавишу C= и клавишу SHIFT (СДВИГ), компьютер перейдет в режим вывода на дисплей знаков верхнего и нижнего регистров. Если теперь Вы нажмете клавишу C= и любую другую клавишу с графическим символом, на экране будет воспроизводиться графика, показанная на левой стороне клавиши.

Чтобы вернуться к режиму "ВЕРХНИЙ РЕГИСТР/ГРАФИКА", необходимо снова нажать клавишу C= и клавишу SHIFT.

Второй функцией клавиши C= является обеспечение доступа ко второму набору восьми текстовых цветов. При нажатии клавиши C= и любой цифровой клавиши любой напечатанный текст будет появляться в том цвете, который Вы выбрали нажатием соответствующей клавиши. Цвета, которые можно выбрать с помощью той или иной клавиши, описываются в Главе 5.

Возвращение к нормальному режиму

Теперь, когда Вы познакомились с клавиатурой, рассмотрим некоторые из многочисленных возможностей компьютера "Коммодор-64". Если на экране Вашего телевизора еще имеются полосы после регулировки телевизора, нажмите клавиши SHIFT (СДВИГ) и CLR/HOME (ОЧИСТКА/ИСХОДНОЕ ПОЛОЖЕНИЕ). Экран очистится, а курсор будет установлен в исходное положение (левый верхний угол экрана). Теперь одновременно нажмите клавиши C= и 7 . Это устанавливает первоначальный голубой цвет текста. Есть еще один шаг, который необходим для возвращения в нормальное состояние. Нажмите клавиши (УПРАВЛЕНИЕ) и (НУЛЬ, а не букву O). Это возвращает режим дисплея обратно в нормальное состояние. Как Вы помните, мы включили режим обратной (REVERSE) печати с помощью клавиш CTRL и 9 для создания цветных полос (на самом деле цветные полосы представляют собой "реверсированные" пробелы). Если бы во время цветового теста мы находились бы в режиме нормального (а не обратного) текста, то курсор перемещался бы, но оставлял вместо себя только пробелы.

Примечание. Теперь, когда Вы ознакомились с положением дел, укажем, что существует простой способ для возвращения системы к нормальному режиму дисплея. Одновременно нажмите клавиши RUN/STOP и RESTORE. Это очистит экран и возвратит всю систему к нормальному режиму. Если в компьютере имеется программа, она останется нетронутой. Эту полезную последовательность желательно запомнить, особенно если Вы много программируете.

Если Вы хотите сбросить машину, как если бы она была выключена и затем включена вновь, наберите на клавиатуре: SYS 64738 и нажмите клавишу RETURN (ВОЗВРАТ). Будьте осторожны с этой командой! Она стирает любую программу или информацию, находящуюся в машине.

Загрузка и хранение программ

Одной из наиболее важных характеристик компьютера "Коммодор-64" является возможность загружать программы с кассетной ленты или диска и выводить их на эти носители для хранения.

Эта характеристика позволяет хранить программы, которые Вы пишете, для их использования впоследствии или же закупать готовые программы для их использования на компьютере "Коммодор-64".

Убедитесь, что дисковое или кассетное запоминающее устройство правильно подключены к компьютеру.

Загрузка готовых пакетов программ

Тем, кто заинтересован только в использовании готовых пакетов программ на специальных кассетах, компакт-кассетах или дисках, необходимо помнить следующее:

1. Специальные кассеты (cartridges): Компьютер "Коммодор-64" имеет целую серию программ и игр на специальных кассетах. Эти программы предназначены для самых разнообразных деловых и личных применений, а игры представляют собой не имитации, а самые настоящие игры аркадного типа (arcade games). Для того чтобы загрузить эти игры, сначала включите свой телевизор. Затем ВЫКЛЮЧИТЕ свой компьютер "Коммодор-64". ПЕРЕД ВСТАВЛЕНИЕМ ИЛИ ИЗ'ЯТИЕМ СПЕЦИАЛЬНЫХ КАССЕТ ВЫ ДОЛЖНЫ ОБЯЗАТЕЛЬНО ВЫКЛЮЧИТЬ КОМПЬЮТЕР "КОММОДОР-64", В ПРОТИВНОМ СЛУЧАЕ СПЕЦИАЛЬНАЯ КАССЕТА БУДЕТ ИСПОРЧЕНА! После выключения компьютера вставьте специальную кассету. Затем включите компьютер. И, наконец, нажмите на клавиатуре соответствующую стартовую (START) клавишу согласно инструкции, прилагаемой к каждой игре.

2. Компакт-кассеты (cassettes): Используйте свой кассетный магнитофон типа C2N и обычные компакт-кассеты, которые составляют часть Вашего пакета программ. Убедитесь, что лента полностью перемотана на начало первой стороны. Затем наберите на клавиатуре слово LOAD (ЗАГРУЗКА). Компьютер даст Вам ответ: PRESS PLAY ON TAPE (НАЖМИТЕ КНОПКУ СЧИТЫВАНИЯ С ЛЕНТЫ), поэтому теперь Вы должны нажать кнопку считывания с ленты. Экран компьютера будет оставаться чистым до тех пор, пока не будет найдена программа. После нахождения программы на экране появятся слова FOUND (PROGRAM NAME) - НАЙДЕНА (ИМЯ ПРОГРАММЫ). Теперь нажмите клавишу C=. Это позволит реально загрузить программу в компьютер. Если Вы хотите остановить загрузку, нажмите клавишу RUN/STOP (ПРОГОН/ОСТАНОВ).

3. Диск: При использовании дискового запоминающего устройства аккуратно установите диск с готовой программой, так чтобы ярлычок на диске "смотрел" вверх и был обращен к Вам. Найдите на диске небольшую метку (она может быть закрыта небольшим кусочком ленты). Если Вы устанавливаете диск правильно, метка будет на левой стороне. Как только диск будет установлен на место, закройте защитную защелку, нажав на рычаг вниз. Теперь наберите на клавиатуре слова LOAD "PROGRAM NAME" (ЗАГРУЗКА "ИМЯ ПРОГРАММЫ"), 8 и нажмите на клавишу RETURN (ВОЗВРАТ). Диск начнет вращаться, а на экране появится следующая картинка:

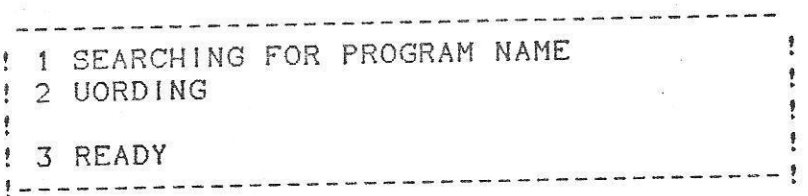


Рис. 6

1 - поиск - имя программы; 2 - загрузка; 3 - готово

Когда на дисплее появится слово READY (ГОТОВО) и курсор начнет мигать, наберите на клавиатуре слово RUN (ПРОГОН) и нажмите клавишу RETURN (ВОЗВРАТ) - Ваше пакетированное программное обеспечение готово к работе.

Загрузка программ с ленты

Загрузка программы с ленты или диска весьма проста. При работе с лентой перемотайте ленту к началу и наберите на клавиатуре слова LOAD "PROGRAM NAME" (ЗАГРУЗКА "ИМЯ ПРОГРАММЫ"). Если Вы не помните имени программы, наберите на клавиатуре только слово LOAD (ЗАГРУЗКА), и первая программа, имеющаяся на ленте, будет загружена в память.

После того как Вы нажмете клавишу RETURN (ВОЗВРАТ), компьютер выведет на экран дисплея ответ:
PRESS RLAY ON TAPE (НАЖМИТЕ КНОПКУ СЧИТЫВАНИЯ С ЛЕНТЫ). После того как Вы нажмете кнопку считывания с ленты, экран очистится и станет того же цвета, как и рамка экрана, в то время как компьютер будет осуществлять поиск программы. Когда программа будет найдена, на экране появятся слова FOUND PROGRAM NAME (НАЙДЕНО ИМЯ ПРОГРАММЫ).

Для реальной загрузки программы нажмите клавишу C= . Чтобы выйти из режима загрузки, нажмите клавишу RUN/STOP (ПРОГОН/ОСТАНОВ). Если Вы нажмете клавишу компьютера "Коммодор" C= , экран снова примет цвет рамки, пока программа загружается. После того как загрузка программы будет завершена, экран возвратится в нормальное состояние, и на экране вновь появится слово READY (ГОТОВО).

Чтобы перейти к следующей программе, нажмите любую клави-

шу, за исключением клавиши компьютера "Коммодор-64", C=, а затем клавишу RUN/STOP (ПРОГОН/ОСТАНОВ).

Загрузка программ с диска

Процедура загрузки программ с диска аналогична. Наберите на клавиатуре: LOAD "PROGRAM NAME", 8. После того как Вы нажмете клавишу RETURN, диск начнет вращаться, а на экране появится следующее сообщение:

```
 ---------------------------------------
 ! 1 SEBRCHING FOR PROGRAM NAME         !
 ! 2 LORDING                           !
 !                                     !
 ! 3 READY                             !
 ---------------------------------------
```

Рис. 7
1 - поиск - имя программы; 2 - загрузка; 3 - готово

Чтобы загрузить справочник, наберите на клавиатуре LOAD "$", 8. Затем наберите слово LIST (СПИСОК) и нажмите клавишу RETURN (ВОЗВРАТ).

Примечание. Когда Вы осуществляете загрузку в память компьютера новой программы, все команды, которые ранее находились в компьютере, будут стерты. Прежде чем загрузить новую программу, убедитесь, что Вы не теряете программу, с которой Вы работаете. Как только программа будет загружена, Вы можете прогнать ее (RUN), включить ее в список (LIST) или внести в нее изменения и записать в память новую версию.

Вывод программ на ленту

Если после ввода программы в компьютер Вы хотите сохранить ее на ленте, наберите на клавиатуре: SAVE "PROGRAM NAME" (СОХРАНИТЬ "ИМЯ ПРОГРАММЫ"). "ИМЯ ПРОГРАММЫ" ("PROGRAM NAME") может состоять из 16 знаков или менее. После того как Вы нажали клавишу RETURN, компьютер ответит сообщением: PRESS RLAY AND RECORD ON TAPE (НАЖМИТЕ КНОПКУ ЗАПИСЬ И ПУСТИТЕ ЛЕНТУ). Нажмите одновременно кнопки пуска и записи на кассетном магнитофоне. Экран очистится и примет цвет своей рамки.

После того как программа будет выведена на ленту, на экране вновь появится слово READY (ГОТОВО), что указывает на возможность работы с другой программой, или же Вы можете просто выключить компьютер до нового сеанса работы.

Вывод программ на диск

Вывод программ на диск осуществляется еще проще. Наберите

на клавиатуре: SAVE "PROGRAM NAME", 8. Цифра 8 представляет собой код диска, таким образом Вы даете компьютеру команду записать программу именно на диске.

После того как Вы нажали клавишу RETURN , диск начнет вращаться, и на дисплее появится следующее сообщение:

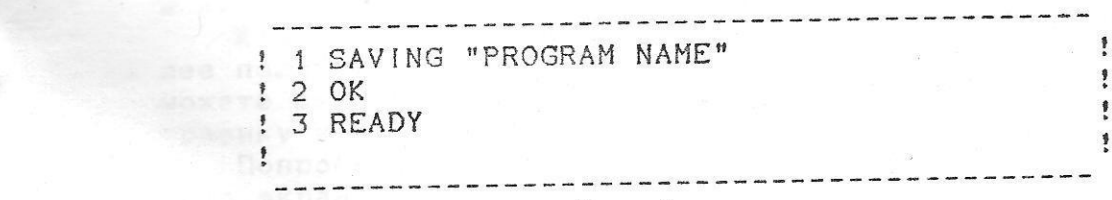


Рис.8
1 - вывод "имя программы"; 2 - выполнено; 3 - готово

Вывод на печать и вычисления

Теперь, когда Вы освоили более сложные операции, которые Вам необходимы для сохранения нужных Вам программ, составим несколько программ, которые необходимо сохранить.

Наберите на клавиатуре следующее:

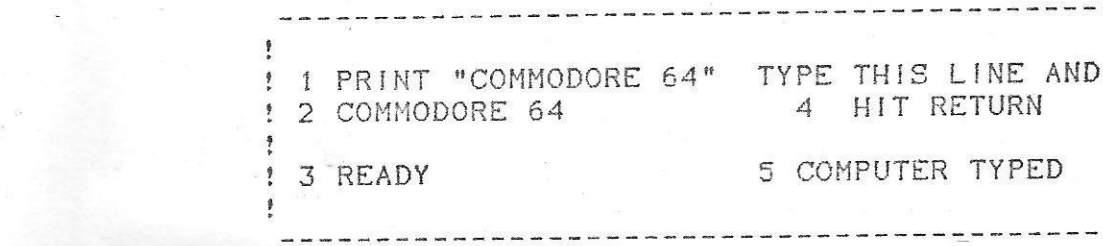


Рис. 9
1 - вывести на печать "COMMODORE 64"; 2 - Коммодор-64; 3 - готово; 4 - наберите на клавиатуре эту строку и нажмите клавишу RETURN; 5 - ответ компьютера

Если при наборе с клавиатуры Вы допустили ошибку, то для стирания знака, который стоит слева от курсора, используйте клавишу INST/DEL (ВСТАВЛЕНИЕ/СТИРАНИЕ). Вы можете стереть столько знаков, сколько необходимо.

Рассмотрим, что произошло в вышеприведенном примере. Во-первых, Вы дали компьютеру команду ВЫВЕСТИ НА ПЕЧАТЬ (PRINT) то, что заключалось между кавычками. Нажав на клавишу RETURN, Вы приказали компьютеру выполнить команду, и компьютер вывел на экран слова COMMODORE 64.

Когда Вы используете оператор PRINT (ВЫВЕСТИ НА ПЕЧАТЬ) в такой форме, то на печать (на экран) выводится все то, что стоит между кавычками.

Если компьютер ответил сообщением ?SYNTAX ERROR (СИНТАКСИЧЕСКАЯ ОШИБКА), проверьте, не сделали ли Вы ошибки при наборе с клавиатуры или не забыли ли Вы поставить кавычки.

Компьютер представляет собой точное устройство и выполняет только команды, подаваемые в специально установленной форме.

Здесь нет ничего особенного, просто необходимо помнить правила осуществления ввода информации в компьютер, как мы это показываем в примерах, и компьютер у Вас будет работать нормально.

Помните, что, набирая на клавиатуре любые знаки, Вы не сможете вывести компьютер из строя, и лучшим способом выучить язык БЕЙСИК является опробирование различных операций с клавиатуры и изучение того, что из этого получается.

Команда PRINT (ВЫВЕСТИ НА ПЕЧАТЬ) является одной из наиболее полезных команд в языке БЕЙСИК. С помощью этой команды Вы можете выводить на дисплей почти все, что Вы хотите, включая графику и результаты вычислений.

Попробуйте, например, выполнить следующую операцию. Очистите экран с помощью нажатия клавиши SHIFT (СДВИГ) и CLR/HOME (ОЧИСТКА/ИСХОДНОЕ ПОЛОЖЕНИЕ). Наберите на клавиатуре следующее (обратите внимание, что для набора цифры 1 (единица) необходимо нажимать клавишу цифры 1, а не клавишу латинской буквы I):

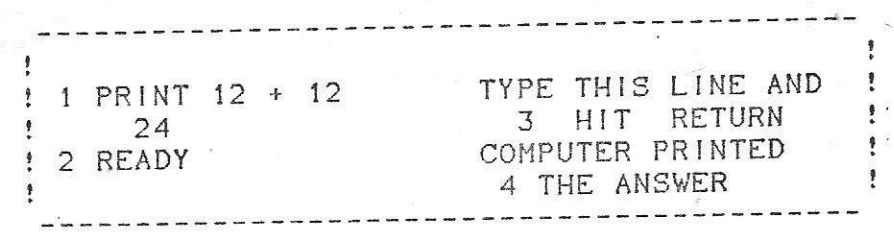


Рис. 1Ø

1 - вывести на печать 12 + 12; 2 - готово; 3 - напечатайте эту строку и нажмите клавишу RETURN; 4 - ответ, выведенный компьютером на дисплей

Вы видите, что компьютер "Коммодор-64" является по своей природе калькулятором. Результат "24" был вычислен компьютером и выведен на дисплей автоматически. Вы можете также осуществлять вычитание, умножение, деление, возведение в степень, а также вычисление сложных математических функций, таких как извлечение квадратного корня и т.п. Вы не ограничиваетесь единственно вычислением того, что набрано в строке, но можете осуществить большее (об этом будет рассказано далее).

Отметим, что в вышеприведенной форме оператор PRINT вел себя не так, как в первом примере. В этом случае на печать выводится результат вычисления, а не точное сообщение, которое Вы набрали на клавиатуре. Это произошло потому, что Вы не поставили кавычек.

Сложение

Знак плюс (+) означает сложение: мы даем компьютеру команду вывести на печать результат сложения 12 с 12. Другие арифметические операции имеют форму, аналогичную сложению. После набора на клавиатуре слова PRINT (ВЫВЕСТИ НА ПЕЧАТЬ) и осуществления вычислений не забывайте нажать клавишу RETURN.

Вычитание

Чтобы осуществить вычитание, используйте клавишу обычного знака минус (-). Наберите на клавиатуре:

```
 ----------------------------------------
!  1                              2       !
!  PRINT 12 - 9                    HIT RETURN !
!  3                                      !
 ----------------------------------------
```

Рис. 11

1 - вывести на печать 12 - 9; 2 - нажмите клавишу RETURN

Умножение

Если Вы хотите умножить 12 на 12, используйте клавишу "звездочка" (*) для обозначения операции умножения. Наберите на клавиатуре:

```
 ----------------------------------------
!  1                              2       !
!   PRINT 12 * 12                  HIT RETURN  !
!   144                                    !
 ----------------------------------------
```

Рис. 12

1 - вывести на печать 12 * 12; 2 - нажмите клавишу RETURN

Деление

Для обозначения операции деления используется клавиша со знаком "/". Например, для того чтобы разделить 144 на 12, наберите:

```
 ----------------------------------------
!  1                              2       !
!   PRINT 144/12                   HIT RETURN  !
!   12                                     !
 ----------------------------------------
```

Рис. 13

1 - вывести на печать 144 : 12; 2 - нажмите клавишу RETURN

Возведение в степень

Аналогичным образом Вы можете возвести число в степень (это эквивалентно умножению числа самого на себя определенное число раз). Для обозначения операции возведения в степень используется клавиша со знаком " ^ " .

```
! 1  PRINT 12 ^ 5                                    !
!    248832                                          !
```

Рис. 14

1 - вывести на печать 12^5

Это эквивалентно следующему:

```
!    PRINT 12 * 12 * 12 * 12 * 12                    !
!    248832                                          !
```

Рис. 15

1 - вывести на печать 12 x 12 x 12 x 12 x 12

Примечание. Язык БЕЙСИК имеет ряд способов укорочения выполнения операций. Одним из таких способов является сокращение написания команд (или клавишных обозначений) на языке БЕЙСИК. Например, вместо набора на клавиатуре слова PRINT (ВЫВЕСТИ НА ПЕЧАТЬ) можно использовать клавишу со знаком ?.

По мере того, как мы будем изучать компьютер, Вы познакомитесь с большим количеством команд. В Приложении Г приводятся аббревиатуры (сокращения) для каждой команды, а также показывается, что будет выводиться на дисплей, когда Вы набираете на клавиатуре сокращенную форму.

Последний пример имеет другую важную особенность: многие вычисления могут быть осуществлены в той же строке, причем это могут быть вычисления различных типов.

Вы можете решить такой пример:

```
!                                 1 THIS?           !
!  ? 3 + 5 - 7 + 2               REPLACES THE      !
!  3                             WORD PRINT        !
```

Рис. 16

1 - этот знак вопроса (?) заменяет здесь слово PRINT (ВЫВЕСТИ НА ПЕЧАТЬ)

До этого момента мы использовали только небольшие числа и легкие примеры, однако компьютер "Коммодор-64" способен осуществлять более сложные вычисления.

Например, Вы можете складывать ряд больших чисел. Попробуйте это осуществить, но не используйте запятых, так как это считается ошибкой.

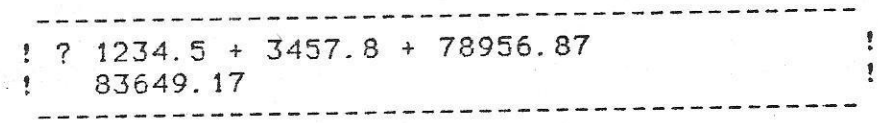


Рис. 17

Это выглядит правильно, однако теперь наберите таким образом:

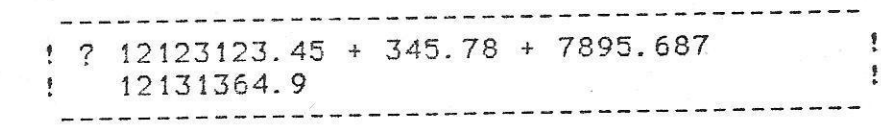


Рис. 18

Если Вы дадите себе труд осуществить эти вычисления вручную, то Вы получите другой результат.

Что же произошло? - Дело в том, что, хотя компьютер обладает большой вычислительной мощностью, существуют ограничения на числа, с которыми он может оперировать. Компьютер "Коммодор-64" может работать с десятичными числами. Однако при печати на дисплей выводится только девять цифр.

Поэтому в нашем примере результат был округлен, для того, чтобы можно было уложиться в отведенное число знаков. Компьютер "Коммодор-64" округляет вверх, когда следующая цифра - более 4, и вниз, когда следующая цифра 4 или менее.

Числа между 0,01 и 999999999 выводятся на печать с помощью обычной нотации.

Научная нотация представляет собой способ выражения очень больших или очень малых чисел как степени 10. Если Вы наберете на клавиатуре:

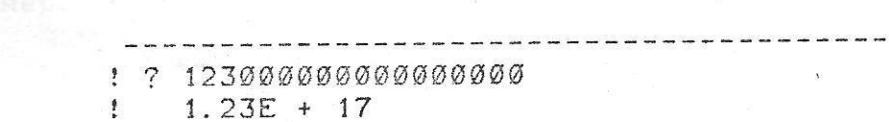


Рис. 19

то это то же самое, что $1,23 \times 10^{17}$. Последнее выражение используется для компактной записи результата. Однако, существуют пределы и для чисел в научной нотации, с которыми компьютер "Коммодор-64" может оперировать. Эти пределы таковы:

Наибольшее число: $+/- 1,70141183 \cdot 10^{38}$

Наименьшее число: $+/- 2,93873588 \cdot 10^{-39}$

Порядок выполнения операций

Если бы Вы попытались осуществить некоторые сложные смешанные вычисления, отличные от тех, которые мы ранее показали в примерах, то, возможно, Вы не получили бы тех результатов,

которые Вы ожидали. Причиной этого является тот факт, что компьютер выполняет вычисления в определенном порядке.

В примере 20 + 8/2 Вы не можете сказать, каков будет правильный результат - 14 или 24 - если Вы не знаете, в каком порядке следует выполнять вычисления. Если Вы сложите 20 и 8, которые Вы предварительно поделите на 2, (т.е. сложите 20 и 4), то результат будет 24. Однако, если Вы сложите 20 и 8, а результат сложения поделите на два, то ответ будет равен 14. Попробуйте решить этот пример на компьютере, каков будет результат.

Результат, который выдаст компьютер в этом случае, будет равен 24, так как компьютер "Коммодор-64" осуществляет вычисления слева направо в соответствии со следующими правилами.

В первую очередь осуществляется реализация знака минус (-), означающего, что число является отрицательным.

Во вторую очередь осуществляется операция возведения в степень (^), слева направо.

В третью очередь осуществляются операции умножения (*) и деления (/), слева направо.

В четвертую очередь выполняются операции сложения (+) и вычитания (-) слева направо.

Если Вы в вышеупомянутом примере будете применять указанный порядок операций, то сначала необходимо выполнить деление, а затем сложение, и результат, соответственно, будет равным 24.

Придумайте свои примеры и посмотрите, сумеете ли Вы следовать этим правилам и предсказать результаты, которые будет выдавать компьютер.

Имеется также удобный способ изменять порядок выполнения операций с помощью скобок, выделяющих операции, которые Вы хотите осуществить первыми.

Например, если Вы хотите разделить 35 на 5+2, наберите:

```
 ----------------------------------------
! ? 35 / 5 + 2                           !
! 9                                      !
 ----------------------------------------
```

Рис.20

Вы получите результат 9 (35 поделено на 5, и к частному прибавлено 2), что не соответствует Вашим желаниям. Чтобы получить действительно нужный Вам результат, необходимо сделать так:

```
 ----------------------------------------
! ? 35 / (5 + 2)                         !
! 5                                      !
 ----------------------------------------
```

Рис.21

В этом случае компьютер вначале выполняет то, что заключено в скобки. Если внутри одних скобок находятся другие, то сначала осуществляются действия, указанные во внутренних скобках.

Если имеет место ряд последовательных скобок, таких как в следующем примере:

```
? (12 + 9) * (5 + 2)
147
```

Рис.22

то в этом случае компьютер сначала осуществляет действия в скобках слева направо. Здесь 21 будет умножаться на 7, и результат будет равен 147.

Комбинирование

Хотя мы затратили много времени на изучение вопросов, которые Вам могут показаться и не очень важными, детали, изложенные в настоящем параграфе, будут весьма важны, когда Вы приступите к составлению программы.

Чтобы усвоить, как нужно оптимальным образом использовать компьютер, рассмотрим такой вопрос: как скомбинировать два типа операторов печати, которые мы рассмотрели ранее, для вывода на печать сообщений с максимальным количеством информации?

Нам известно, что, заключая информацию в кавычки, мы выводим ее на экран в том виде, в каком она набирается с клавиатуры, а используя математические операторы, мы заставляем компьютер осуществлять вычисления. Возникает мысль, а почему бы не скомбинировать эти два типа операторов PRINT следующим образом:

```
                1  SEMICOLON MEANS NO SPACE
? "5 * 9 =" ; 5 * 9
5 * 9 = 45
```

Рис.23

1 - точка с запятой означает отсутствие пробела

Хотя это, возможно, кажется несколько излишним, но то, что мы сделали - это всего лишь использование операторов печати обоих типов. Первая часть выражения - вывод на печать выражения "5 * 9=" в том виде, как оно набрано с клавиатуры. Вторая часть - это действительное выполнение операций, причем точка с запятой отделяет информационную часть оператора от действительного вычисления.

Для того, чтобы компьютер работал правильно, Вы всегда должны разделять части смешанного оператора печати каким-либо знаком пунктуации. Попробуйте использовать для этого запятую и посмотрите, что из этого получится.

Г Л А В А 4

НАПИСАНИЕ ПРОСТЫХ ПРОГРАММ НА БЕЙСИКЕ

Выше было показано, как выполнять простые действия на Вашем компьютере. Вы уже поэкспериментировали путем набора строк содержащих разные инструкции и получением результата нажатием клавиши RETURN. Этот простой способ называется режимом непосредственного исполнения или калькуляторным режимом.

Но вы наверное хотите, чтобы Ваш компьютер делал более сложную работу, которая требует использования более одного оператора. Когда вы скомбинируете несколько операторов в программу, вы сможете использовать Ваш "64" на полную мощь.

Для того, чтобы легко написать Вашу первую программу на "64", сделайте следующие шаги:

1. Очистите экран, нажав клавишу CLR/HOME, держа при этом нажатой SHIFT.

2. Наберите NEW м нажмите RETURN. Это очищает информацию, которая могла остаться в памяти во время Ваших экспериментов.

3. Наберите следующие две строки так, как показано ниже:

```
10 ?"COMMODORE 64"
20 GOTO 10
```

4. Незабудьте нажимать клавишу RETURN в конце каждой строки. После того, как Вы наберете первую строку и нажмете клавишу RETURN, Вы заметите, что команда PRINT не выполнится сразу, как это было раньше. Это потому, что Вы начали команду с номера строки (10). Когда Вы используете номера строк, компьютер знает, что Вы пишете программу, поэтому он ждет, пока вы не наберете всю программу, перед исполнением какой-нибудь инструкции.

5. Наберите RUN и нажмите RETURN. Команда RUN говорит компьютеру, что вы закончили набирать вашу программу, и готовы, чтобы компьютер ее выполнил. Вот что случится после того, как вы запустите вашу программу командой RUN.

```
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
.................
```

6. Остановите выполнение программы, нажав клавишу RUN/STOP. Компьютер будет выполнять Вашу программу, пока вы не прервете ее работу клавишей RUN/STOP. После этого экран будет выглядеть так:

```
! COMMODORE 64
! COMMODORE 64
! COMMODORE 64
! COMMODORE 64
! BREAK IN 10
! READY
```

Эта простая программа показывает несколько важных вещей, которые являются основой для всего программирования.

Нумерация строк

Мы уже упоминали раньше на 4 шаге, что нумерация строк показывает компьютеру, что вы пишете программу. Они также говорят компьютеру, в каком порядке вы хотите, чтобы операторы исполнялись. Без нумерации строк компьютер не будет знать, какой оператор выполнять первым, а какой вторым и т.д.

Чем сложнее и длиннее программа, тем важнее, чтобы компьютер знал не только ЧТО делать, но и КОГДА это делать. Одно хорошее свойство состоит в том, что Вы можете набирать строку с номером 10, после строку с номером 20, потому что компьютер различает последовательность строк по их номерам, а не по последовательности набора их.

Другое свойство номеров строк заключается в том, что вы можете ссылаться на эти строки из дугих строк. Когда вы хотите возвратиться и повторить выполнение оператора, Вы можете это сделать с помощью оператора GOTO, как это было показано в предыдущем примере.

Оператор GOTO

Когда вы запустите простую программу, которую мы описывали ниже, командой RUN, COMMODORE 64 напечатается несколько раз, а не один, т.к. в строке 20 есть оператор GOTO.

Оператор GOTO "говорит" компьютеру напрямую перейти к обозначенной строке. Потом компьютер выполнит инструкции в обозначенной строке и перейдет на выполнение следующей строки.

Вы можете использовать оператор GOTO для того, чтобы "сказать" компьютеру вернуться к строке, которая уже была выполнена. Или GOTO может "сказать" компьютеру перейти вперед, т.е. это значит, что несколько строк не должны выполняться.

В нашем примере программа печатает сообщение в строе 10 и перемещается к строке 20. Там оператор GOTO говорит компьютеру возвратиться к строке 10 и выполнить ее. Таким образом программа печатает сообщение в строке 10 опять, затем перемещается к строке 20, которая посылает компьютер назад к строке 10 и т.д.

Это повторение называется ЦИКЛ. Т.к. программа не дает компьютеру выход из цикла, цикл повторяется до бесконечности. Вы можете остановить цикл прерыванием программы с помощью клавиши RUN/STOP.

Было бы лучше включить оператор в вашу программу, который бы останавливал цикл и вам не нужно было бы использовать клавишу RUN/STOP. Далее в этой главе мы рассмотрим способы окончания циклов.

Использование команды LIST

Теперь, после того как вы прервали исполнение программы, наберите LIST и нажмите RETURN. Вы увидете вашу программу неизменной, т.к. она осталась в памяти компьютера, даже после того, как вы прервали ее работу. Единственное различие в том, что компьютер поменял знак вопроса "?" на слово PRINT. Но это не затрагивает вашей программы. Когда вы используете команду LIST, компьютер всегда показывает строки программы в правильном числовом порядке, даже если вы набирали их не попорядку.

Одна из главных особенностей между написанием программ и набором одиночных строк в калькуляторном режиме в том, что вы постоянно теряете одиночный оператор после того как вы выполните его м очистите экран. В режиме программирования вы можете всегда посмотреть свою программу набрав просто LIST.

Кроме того вы можете изменить вашу программу, записать ее (SAVE) или запустить ее (RUN) опять.

Советы по редактированию

Если вы сделали ошибку в строке во время набора, или вы просто хотите поменять строку, "64" предоставляет вам несколько возможностей по редактированию.

1. Вы можете перенабрать строку в любое время, и компьютер автоматически заменит новую строку на старую. Все что вы хотите сделать для замены строки делайте для одного номера строкк. Например:

```
!                                  !
! 10 ?"MY NAME IS SARAH            ! Меня зовут САРА
! 20 ?"I WAS BORN IN CALIFORNIA"   ! Я родилась в Калифорнии
! 20 ?"I LIVE IN PENNSYLVANIA"     ! Я живу в Пенсильвании
! RUN                              !
! MY NAME IS SARAH                 ! Меня зовут САРА
! I LIVE IN PENNSYLVANIA           ! Я живу в Пенсильвании
!                                  !
```

Как вы видите первая строка 20 никогда не исполняется, потому что она заменена второй строкой 20. Если вы теперь наберете команду LIST, вы увидите, что только вторая строка 20. осталась в программе.

2. Вы можете легко удалить строку, которая вам не нужна просто набрав номер строки и нажатием клавиши RETURN. Если вы сейчас наберете LIST, то увидете, что строка исчезла и номер строки тоже.

3. Вы можете легко редактировать сушествующую строку. Используйте клавиши управления курсором (CRSR) для перемещения курсора назад к строке, которую вы хотите отредактировать, а потом отредактируйте ее как хотите. Как только вы нажмете клавишу RETURN, редактируемая строка заменит старую строку. Не забудьте использовать клавишу INST/DEL для вставки или удаления символов.

Когда вы закончите редактирование, вы можете проверить вашу программу набрав LIST. Помните, что LIST всегда расставляет строки в числовом порядке, если вы набирали их не по-порядку. Попробуйте отредактировать нашу простую программу добавлением точки с запятой к концу строки и убранием "64". После изменений не забудьте переместить курсор за строку 2Ø перед тем как запустить программу. Программа теперь будет работать так:

```
LIST
1Ø PRINT "COMMODORE";
2Ø GOTO 1Ø
COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE
BREAK IN 1Ø
READY
```

Как использовать переменные

Переменные - это символ, который имеет значение. Иногда значение переменной неизвестно перед тем пока вы не запустите программу. Одно из назначений программы в присвоении одного или более значений переменной. Посмотрте на эту строку из программы:

```
-------------------------
!                       !
! 2Ø LET X = 28 + Y     !
!                       !
-------------------------
```

Знак = означает "станет" или "взять значение". Инструкция LET необязательна и может быть опушена.

В этом равенстве X и Y переменные. Пусть X имеет значение числа дней в месяце. Одна из замечательных особенностей переменной в том, что вы можете менять ее зннчение в программе, следовательно X имеет значение числа дней в месяце не только для одного месяца. Для этого используется Y. Все месяцы имеют 28 дней, следовательно Y имеет значение числа дней превышающих 28. Далее в этой главе есть программа, которая присваивает значения этим переменным.

Сейчас самая важная вешь - это понять как работают переменные, потому что переменные позволяют вам выполнять большие и сложные работы на вашем компьютере. Переменные также позволяют писать программы, которые можно использовать несколько раз для различных данных.

Представьте, что ваш компьютер имеет набор маленьких ячеек. Когда вы пишете программу, вы можете использовать несколько из этих ячеек по необходимости и во время работы программы "класть" некоторые значения в любые ячейки, используя их имена. Например, в приведенном выше равенстве мы использовали две ячейки с именами X и Y. В начале программы эти ячейки имеют имена, но они пустые. Вот что случается, когда вы "кладете" значение в ячейку X:

X	Y						
	3						

Теперь переменная Y имеет значение 3. Вы можете присвоить Y это значение просто написанием этого простого оператора:

```
10  Y = 3
```

Так как X равно 28 плюс Y, то когда вы запустите программу, ячейка X получит значения тоже:

X	Y						
31	3						

Вот как выглядит программа:

```
10  Y = 3
20  X = 28 + Y
30  ?"THE NUMBER OF DAYS IN MAY IS"; X
RUN
THE NUMBER OF DAYS IN MAY IS 31
(число дней в мае 31)
```

А вот другая программа, которая использует переменные:

```
10  X% = 15
20  X = 23.5
30  X$ = "TOTAL:"          (Итого)
40  Y = X% + X
50  ?X$; Y
```

Когда вы запустите программу, воображаемые ячейки будут выглядеть так после выполнения 30 строки:

X%	X	X$	Y			
15	23.5	TOTAL:				

После выполнения программы, Y получит значение 38,5. Приведенный пример использует 3 типа переменных

тип	символ	описание	примеры	примеры величин
целый	%	целые числа	X%, A1%	15, 1Ø2, 3
текстовая строка	¤	символы в кавычках	X¤, AB¤	"TOTAL:", "DAY 1"
с плавающей запятой		целые или вещественные числа	X, AB	23.5, 12, 1.3E + 2

Используйте правильные типы переменных в ваших программех. Если вы попробуете например присвоить целой переменной текстовую строку, ваша программа не будет работать.

А вот еще несколько правил, которые вы должны принимать во внимание когда вы выбираете имена для переменных:

- Имя переменной может иметь один или два символа не включая специальных символов для обозначениия целой и текстовой переменной.
- Вы можете использовать более двух символов для обозначения имени переменной, но компьютер распознает только первые два. Таким образом компьютер будет думать, что PA, PARTNO и PAGENO - это одна переменная и относится к одной "ячейке".
- Программа легкочитаема, когда вы используете длинные имена переменных, но когда вы используете более двух символов в имени, будьте уверены, что первые два символа для разных переменных разные.
- Вы можете использовать X, X% и X¤ в одной программе потому что специальные символы % и ¤ делают имена переменных различными. Это же верно и для A2, A2% и A2¤.
- Первый символ должен быть буквой (A-Z). Второй и последующие могут быть или буквами млм цифрами (Ø-9). Помните, что компьютер игнорирует все символы после второго, даже если в третьей позиции символы % или ¤.
- Имена переменных не могут содержать ключевые слова Бейсика, которые также называются зарезервированными словами. Это слова как например PRINT и RUN, которые являются частью языка Бейсик. В приложении D приведен список всех зарезервированных слов.

Вот еще одна программа, которая показывает вам как использовать переменные:

```
 NEW
1Ø  X=1.Ø5
2Ø  Y=3ØØ
3Ø  Z=X*Y
4Ø  PRINT "SEATS AVAILABLE:";Y
5Ø  PRINT "TICKETS AVAILABLE:";Z
6Ø  Y=Y+1
7Ø  PRINT "OVERBOOKING POINT:";Y
```

```
RUN
SEATS AVAILABLE: 300     возможные места
TICKETS AVAILABLE: 315   возможные билеты
OVERBOOKING POINT: 301   точка переполнения
```

Строки (10-30) присваивают имена переменным

Строки 40 и 50 печатают сообщение и текущее значение переменных Y и Z. Заметьте, что на строке 40, значение для Y - 300 строка 60 дает Y новое значение, и это новое значение печатается в строке 70. Строка 60 показывает, что переменное может иметь более одного значения в программе.

Строка 60 также показывает другие особенности переменных: вы можете делать переменную равной себе и другой величине. Это не допускается в обычной алгебре, но это тип оператора широко используется в программировании. Это означает: возьмите текущее значение переменной, скомбинируйте его с другой величиной и замените первое значение переменной на новое значение. Вы можете использовать операторы как например эти:

```
Y = Y - 1
Y = Y + X
Y = Y/2
Y = Y*(X+2)
```

Использование циклов FOR/NEXT

Мы обсуждали циклы выше в этой главе, когда рассказывали про оператор GOTO. Как вы помните, циклы повторяют выполнение одной или более строк в программе.

Оператор FOR/NEXT дает вам возможность создавать очень полезные циклы, который управляет выполнением части программы несколько раз. Оператор FOR устанавливает предельные значения числа выполнения цикла путем присваивания диапазона значений переменной.

Например:

```
FOR COINT = 1 TO 4
```

Оператор NEXT отмечает конец цикла FOR/NEXT. Когда программа достигает оператора NEXT, компьютер проверяет оператор FOR - не достигнут ли предел цикла. Если предельное значение не достигнуто, то цикл продолжается и значение переменной в операторе FOR увеличивается на 1. Например, если вы добавите цикл FOR/NEXT к программе в начале главы, то получите следующее:

```
10  FOR CT = 1 TO 4
20  ? "COMMODORE"
30  NEXT CT
RUN
COMMODORE
COMMODORE
COMMODORE
COMMODORE
```

Теперь когда вы добавили цикл FOR/NEXT, уже не нужно прерывать программу клавишей STOP. Этот цикл FOR/NEXT работает следующим образом:

Строка 1Ø дает переменной CT диапазон значений от 1 до 4 и говорит компьютеру выполнять следующие строки пока CT не равно 4. Строка 2Ø говорит компьютеру печатать COMMODORE 64.

Строка 3Ø "говорит" компьютеру прибавить 1 к текущему значению CT. Пока величина CT остается в пределах от 1 до 4, программа повторяется и COMMODORE печатается опять. Когда CT равна 4, строка 2Ø выполняется еще один раз. Когда строка 3Ø опять прибавляет 1 к CT, компьютер знает, что CT уже вышла за пределы. Поэтому компьютер прекращает выполнение цикла и программа заканчивает работу сама.

Чтобы быть уверенным, что вы поняли работу цикла FOR/NEXT, мы добавляем еще один оператор PRINT в строку 2Ø чтобы вы смогли проконтролировать значение CT.

```
2Ø  PRINT "COMMODORE"; "COINT ="; CT
3Ø  NEXT CT
RUN
COMMODORE COINT = 1
COMMODORE COINT = 2
COMMODORE COINT = 3
COMMODORE COINT = 4
```

Как вы можете заметить программа заканчивается автоматически когда CT выходит из пределов, установленных в операторе FOR. Вы можете увеличивать значение переменной в операторе FOR/NEXT величинами, отличающимися от 1. Все, что надо сделать - это добавить слово STEP (шаг) и значение шага, которое вы хотите использовать в конце оператора FOR. Например:

```
NEW
1Ø  FOR NB = 1 TO 1Ø STEP .5
2Ø  PRINT NB,       Эта замятая показывает компьютеру
3Ø  NEXT NB         печатать каждое значение величины
                    на первой позиции следующей 1Ø-ти
                    символьной зоны

RUN

1             1,5           2             2,5
3             3,5           4             4,5
5             5,5           6             6,5
7             7,5           8             8,5
9             9,5           1Ø

NEW

1Ø  FOR A = 2 TO 8   STEP 2
2Ø  PRINT A,
3Ø  NEXT A

RUN

2     4     6     8
```

Вы также можете использовать цикл FOR/NEXT для задания обратного счета. Для этого надо задать отрицательный шаг. Например, если вы измените сроку 10 на следующую:

```
10  FOR A = 8  TO 2  STEP  -2
```

Результат работы программы будет выглядеть так:

```
RUN

8     6     4     2
```

Использование оператора IF/THEN для управления программами

Оператор IF/THEN - это другой путь управления программой этот оператор "говорит" компьютеру проверить правильность выполнения условия. Если условие истинно, то выполняются команды после слова THEN. Если условие ложно, то программа переходит к выполнению следующей строки, не выполняя команды после слова THEN. Например:

```
10  X = 60
20  X = X+1
30  IF X = 64 THEN PRINT "GOT IT": END
40  GOTO 20
```

Вы можете использовать оператор IF для организациии цикла или для решения, какие части программы будут исполняться. Например:

```
10   A = 0
20   IF A<=8  THEN 40
30   END
40   ?"FRODO LIVES",A
50   A = A + 2
60   GOTO 20
RUN
FRODO LIVES 0
FRODO LIVES 2
FRODO LIVES 4
FRODO LIVES 6
FRODO LIVES 8
```

В этом примере оператор IF/THEN в строке 20 "говорит" компьютеру проверить текущее значение A. Если A меньше или равно 8, то программа пропускает строку 30 и продолжает со строки 40. Если A больше 8, или другими словами, если условие в строке 20 - ложно, компьютер игнорирует команду после слова THEN.

Если строка 20 - ложь, выполняется строка 30. Строка 40 печатает сообщение и текущее значение A. Строка 50 прибавляет 2 к значению A во время каждого прохода цикла. Как только A становится равной 10, строка 20 становится ложной,

выполняется строка 30 и программа заканчивает работу.

Вы можете использовать любые из этих операторов отношение в операторе IF/THEN:

символ	содержание
<	меньше
>	больше
=	равно
<>	не равно
>=	больше или равно
<=	меньше или равно

Г Л А В А 5

РАСШИРЕННЫЙ БЕЙСИК

Введение

Следующие несколько глав предназначены для людей, хорошо знающих язык Бейсик и дает все необходимое для написания расширенных программ.

Те из вас, кто только начал изучать программирование могут найти некоторую информацию "очень технической" для полного понимания. Но вы найдете несколько простых примеров, которые написаны для новых пользователей в двух главах, "спрайт-графика" и "созданиф звуковых эффектов". Это примеры подадут вам идею, как использовать замечательные графические и звуковые возможности, доступные на вашем 64.Если вы захотите узнать больше о написании программ на Бейсике, обратитесь к библиографии, приведенной в Приложении N.

Если вы уже хорошо знакомы с программированием на Бейсике, следующие главы помогут вам овладеть техникой расширенного программирования на Бейсике. Вы найдете дополнительную информацию о расширенном программировании в "Справочном руководстве программиста компьютера COMMODORE 64" (COMMODORE 64 PROGRAMMER'S RFFERFNCE GUUF), которое доступно через вашего поставщика.

Простое движение

Вы можете использовать некоторые из графических возможностей 64 используя то, что мы уже рассмотрели выше в этом руководстве с несколькими новыми понятиями.

Попробуйте ввести следующую программу для того, чтобы посмотреть, что вы можете делать с графикой. Заметьте, что вы можете включать команды управления курсором и экраном внутрь оператора PRINT. Когда вы увидете чтото типа <CRSR LEFT> в прогрpамме, нажмите клавишу <CRSR> держа нажатой при этом клавишу SHIFT. На экране вы увидете графическое представление кода "курсор влево", которое является негативным изображением двух вертикальных палочек. Графическое представление клавиши SHIFT и CLR/HOME - негативное изображение сердца.

```
          NEW
          10  REM BOUNCING BALL     (прыгающий мяч)
 ------>  20  PRINT "(CLR/HOME)"
!         25  FOR X=1 TO 10:    (индикатор новой команды)
!             PRINT "(CRSR/DOWN)";: NEXT
!  ---->  30  FOR BL=1 TO 40
! !       40  PRINT "    (SHIFT-Q)(CRSR LEFT)";
! ! !-->  50  FOR TM=1 TO 5
! ! !---  60  NEXT TM
! !------70  NEXT BL
!         75  REM MOVE BALL RIGHT TO LEFT (перемещение мяча
                                          справа налево)
!
!  ---->  80  FOR BL=40 TO 1 STEP -1
! !       90  PRINT "    (CRSR LEFT)(CRSR LEFT)(SHIFT-Q)
! !           (CRSR LEFT)";
! ! !-->  100 FOR TM=1 TO 5
! ! !---  110 NEXT TM
! !-----  120 NEXT BL
!-------  130 GOTO 20
```

Когда вы запустите программу, то увидите прыгающий мяч, движущийся по экрану слева направо и назад. Рассмотрим как это делается.

Строка 10 - это примечание, которое рассказывает, что делает программа. Оператор REM во время выполнения программы никаких действий не выполняет.

Строка 20 очищает экран.

Строка 25 печатает 10 команд "курсор вниз". Это просто позиция мяча в середине экрана. Без этой строки мяч двигался бы по верхней строке экрана.

Строка 30 задает цикл перемещения мяча на 40 колонок слева направо.

Строка 40 делает 3 вещи

1. Печатает пробел для удаления предыдущей позиции мяча
2. Печатает мяч
3. Выполняет команду курсор - влево для подготовки удаления текущей позиции.

Строки 50 и 60 задают цикл, который замедляет движение мяча. Без этого цикла мяч будет двигаться очень быстро и его будет видно недостаточно четко.

Строка 70 завершает задание цикла в строке 30 для печати мяча на экране. Каждый раз, когда выполняется цикл, мяч перемещается на следующее место направо. Как вы можете заметить на рисунке, программа содержит цикл внутри цикла. Вы можете включать до десяти циклов внутрь цикла. Единственная проблема может возникнуть когда циклы пересекают друг друга. Циклы могут быть вложены друг в друга. Другими словами, если вы начинаете цикл A и затем начинаете цикл B внутри цикла A вы должны завершить цикл B (внутренний цикл) первым. Таким образом могут быть вложены как минимум 9 циклов.

Когда вы пишете программу с циклами, желательно рисовать стрелки от начала цикла к его концу. Если ваши циклы пересекаются, компьютер не может определить, что вы хотите, и таким образом, не может выполнить вашу программу.

Строки с 80-ой по 120 просто реверсируют шаги в первой части программы и перемещают мяч справа налево. Строка 90 немного отличается от строки 40, потому что мяч двигается в

обратном направлении и вы должны удалить мяч справа и двигать влево.

Строка 13Ø возвращает программу назад к строке 2Ø для возобновления всего процесса.

Для модификации программы измените строку 4Ø следующим образом:

```
4Ø  PRINT "(SHIFT)(Q)"
```

Запустите программу и посмотрите, что сейчас произойдет. Так как вы опустили управление курсором, каждый мяч будет оставаться на экране пока он не будет стерт мячом, двигающимся справа налево во второй части программы.

Ввод

До сих пор все в программе задавалось перед ее запуском. Запустив программу, вы уже не можете ничего изменить или добавить. Оператор INPUT позволяет вам посылать информацию программе во время ее выполнения. Программа не только работает на основе этой информации, предоставленной вами, но она не будет продолжать работу пока вы не предоставите ее.

Чтобы понять, как работает INPUT (ввод), наберите NEW, нажмите RETURN и введите эту короткую программу.

```
1Ø  INPUT AД
2Ø  PRINT "YOU TYPED"; AД  ("Вы набрали")
3Ø  PRINT
4Ø  IF AД="STOP" THEN END
5Ø  GOTO 1Ø
RUN
? 6Ø   --------------------------  YOU TYPED
YOU TYPED 6Ø          !
? CONTINUE ----------!
YOU TYPED CONTINUE    !
? STOP --------------!
YOU TYPED STOP
```

В программе происходит следующее:

Строка 1Ø "говорит" компьютеру вывести на экран знак вопроса, предлагая вам ввести значение для AД, и ждать, пока вы не введете это значение.

Строка 2Ø печатает сообщение и введенную величину, а строка 3Ø печатает пустую строку.

Строка 4Ø "говорит" компьютеру закончить программу как только введенное вами значение для AД - STOP.

Строка 5Ø возвращает программу к строке 1Ø и, таким образом, вы можете ввести другую величину. Если условие в строке 4Ø является истинным, вследствие того, что последнее введенное вами значение для AД было STOP, то строка 5Ø не будет выполняться.

Вы можете вводить числовые или строковые переменные и с помощью оператора INPUT печатать сообщение со знаком вопроса для описания того, что надо вводить. Например, вот что происходит, когда вы добавляете приглашающее сообщение к строке 1Ø предыдущего примера:

```
10  INPUT  "KEEP GOING"; A¤        Приглашающее сообщение не
RUN                                может содержать свыше 38
KEEP GOING? 60                     символов
YOU TYPED 60

KEEP GOING? STOP
YOU TYPED STOP
```

Здесь представлен более сложный пример, который демонстрирует большинство из того, что бегло представлено выше, включая оператор INPUT.

```
NEW
1   REM TEMPERATURE CONVERSION (программа преобразования
    PROGRAM                    температуры)
5   PRINT "(SHIFT/CLR/HOME)"
10  PRINT "CONVERT FROM        (преобразовать из Фарен-
    FAHRENHEIT OR CELSIUS      гейта или Цельсия)
    (F/C)": INPUT A¤
20  IF A¤ ="  " THEN 10
30  IF A¤ ="F" THEN 100
40  IF A¤ <>"C" THEN END
50  INPUT "ENTER DEGREES CELSIUS:"; (введите градусы
    C                                Цельсия)
60  F=(C*9)/5+32
70  PRINT C; DEG.CELSIUS=";F;"DEG.FAHRENHEIT"
             (град.Цельсия)   (град.Фаренгейта)
80  PRINT
90  GOTO 10
100 INPUT "ENTER DEGREES FAHRENHEIT:";F (введите градусы
                                         Фаренгейта)
110 C=(F-32)*5/9
120 PRINT F; "DEG.FAHRENHEIT =";C;"DEG.CELSIUS"
           (град.Фаренгейта)      (град.Цельсия)
130 PRINT
140 GOTO 10
```

Строка 10 использует оператор INPUT для печати приглашающего сообщения и ожидание набора вами величины для A¤.

Строки 20,30 и 40 проверяют, что вы ввели и говорят компьютеру куда дальше идти. Строка 20 говорит компьютеру возвратиться назад к строке 10 и запросить опять ввод если ничего не было набрано (если просто была нажата клавиша RETURN). Строка 30 говорит компьютеру идти прямо к строке 100 и выполнить преобразования Фаренгейт-в-Цельсий, если величина, которую вы набрали для A¤ - F. Строка 40 проверяет, чтобы быть уверенным, что вы набрали только F или C. Если это не так, то строка 40 заканчивает программу. Если вы набрали C, компьютер автоматически перейдет к строке 50 для выполнения преобразования Цельсий-Фаренгейт.

Включение всех этих операторов IF может показаться очень подробным для проверки того, что вы ввели. Но это является хорошим стилем программирования, который избавит вас от многих неудач. Вы будете всегда стараться быть уверенным, что ваша программа предусматривает все возможности.

Возвращаясь к примеру: допустим программа знает, какого типа преобразование делать, вычисления сделаны. Затем программа печатает введенную вами температуру и преобразованную. Вычисление, выполняемое этой программой использует стандартную формулу преобразования температур. После того, как вычисления закончены и ответ напечатан, программа возвращается назад и возобновляет работу.

Образец выполнения этой программы:

```
CONVERT FROM FAHRENHEIT OR CELSIUS (F/C):?F
ENTER DEGREES FAHRENHEIT: 32
32 DEG.FAHRENHEIT = Ø DEG. CELSIUS

CONVERT FROM FAHRENHEIT OR CELSIUS (F/C):?
```

После этой программы вы можете захотеть записать ее на диск. Эта программа, как и другие в данном руководстве, может образовать часть вашей библиотеки программ.

Использование оператора GET для ввода данных

GET позволяет вводить с клавиатуры один символ без нажатия клавиши RETURN. Это во многих случаях реально ускоряет ввод данных.

После запуска программы, содержащей оператор GET, любая нажатая вами клавиша присваивается переменной, включенной вами в оператор GET.

Например:

```
1   PRINT "(SHIFT/CLR/HOME)"
1Ø  GET A$: IF A$=""THEN 1Ø   (Между кавычками пробела нет)
2Ø  PRINT A$
3Ø  GOTO 1Ø
```

Строка 1 очищает экран.

Строка 1Ø позволяет вам нажать любую клавишу на клавиатуре. Короче говоря, цикл в строке 1Ø "говорит" компьютеру ждать пока вы не нажмете клавишу перед тем, как перейти к строке 2Ø.

Строка 2Ø отображает на экране клавиши, нажатые вами.

Строка 3Ø отсылает программу назад для ввода следующего символа. Важно помнить, что символ, который вы ввели не будет отображен пока вы не выведите его на экран, как мы это сделали в строке 2Ø.

Оператор IF в строке 1Ø очень важен. GET непрерывно работает даже если вы не нажали никакую клавишу (не так, как INPUT, который ждет вашего ответа), таким образом вторая часть строки 1Ø постоянно проверяет, не нажали ли вы клавишу.

Постарайтесь опустить вторую часть строки 1Ø и посмотрите, что случилось. Для остановки этой программы нажмите клавиши RUN/STOP и RESTORE.

Вы можете легко переписать начало программы преобразования температуры, использовав GET вместо INPUT. Если вы записали эту программу, загрузите ее и измените строки 1Ø и 2Ø следующим образом:

```
10  PRINT "CONVERT FROM FAHRENHEIT OR CELSIUS (F/C)"
20  GET A$: IF A$ = "" THEN 20
```

Эти изменения делают работу программы более "плавной", так как пока вы не нажмете одну из двух возможных клавиш (F или C), которые определяют тип преобразования, ничего не произойдет. Если вы хотите сохранить программу, не забудьте записать ее опять.

Использование GET для программирования функциональных клавиш

Как вы помните, в предыдущей главе мы говорили, что клавиши на правой стороне клавиатуры (с f1 до f8) являются функциональными, которые вы можете запрограммировать для выполнения множества заданий. функциональная клавиша программируется следующим образом:

1. Используйте оператор GET для чтения клавиатуры.
2. Используйте операторы IF для сравнения нажатой вами клавиши с CHR$ кодом для функциональной клавиши, которую вы хотите использовать. Каждый символ на клавиатуре имеет уникальное число CHR$. Например, CHR$ - код клавиши f1-133. В приложении F представлены коды CHR$ для всех клавиш.
3. Используйте операторы THEN для указания компьютеру того, что должна сделать функциональная клавиша.

Когда вы запускаете программу, все ваши действия заключаются в нажатии функциональной клавиши, которую вы запрограммировали, а клавиша выполнит инструкции, заданнын в операторе THEN. Например:

```
10  GET A$: IF A$ ="" THEN 10
20  IF A$ = CHR$(137) THEN PRINT CHR$(14)
30  IF A$ = CHR$(134) THEN PRINT "YOURS TRULY"
```

Строка 10 присваивает переменной A$ значение нажатой клавиши. Как вы помните из предыдущего примера, цикла в строке 10 непрерывно проверяет клавиатуру на ввод.

Строка 20 программирует функциональную клавишу 2, CHR$ (137). Строка 20 "говорит" компьютеру сделать A$ равной CHR$ (14), если вы нажали функциональную клавишу 2. CHR$(14) - это переключатель с заглавных на строчные буквы на клавиатуре. Когда вы запустите программу, то увидите, что символы на экране мгновенно переключатся, если вы нажали f2.

Строка 30 программирует функциональную клавишу 3, CHR$ (134). Строка 30 "говорит" компьютеру сделать A$ равной строке символов "YOURS TRULY", и CHR$ (13), если вы нажали f3 во время выполнения программы. CHR$(13) - это код клавиши RETURN.

CHR$ - коды для функциональных клавиш следующие:

```
f1 = CHR$(133)        f2 = CHR$(137)
f3 = CHR$(134)        f4 = CHR$(138)
f5 = CHR$(135)        f6 = CHR$(139)
f7 = CHR$(136)        f8 = CHR$(140)
```

В справочном руководстве COMMODORE 64 имеется больше информации о программировании функциональных клавиш. Вы можете приобрести это расширенное руководство у вашего местного поставщика.

Случайные числа и другие функции

64 имеет также встроенные функции, которые вы можете использовать для выполнения специальных заданий. Функции аналогичны встроенным программам, включенным в BASIC. Большим преимуществом этих встроенных функций является то, что вам не надо набирать определенное количество операторов каждый раз, когда вы хотите выполнить специальные вычисления. Вместо этого вы набираете команду для необходимой вам функции и компьютер выполняет все остальное. Эти встроенные функции включают исчисление квадратного корня (SQR), определение содержимого ячейки памяти (PEEK), ннерция случайных чисел (RND), и т.д. В Приложении С представлены все имеющиеся в вашем компьютере функции.

Одной из наиболее интересных функций является функция случайных чисел, RND. Если вы хотите создать игру или учебную программу, вы часто будете нуждаться в генерации вашим компьютером случайных чисел. Например, вам будет необходимо имитировать игру в кости. Конечно, вы можете написать программу, которая будет генерировать эти случайные числа, но гораздо легче это сделать с помощью вызова функции случайных чисел (RND).

В целях ознакомления с тем, как работает RND, попробуйте эту короткую программу:

```
NEW
10  FOR X=1 TO 10     Если вы опустите запятую, появится
20  PRINT RND(1),     одна колонка чисел
30  NEXT
```

Когда вы запустите программу, экран отобразит:

```
.789280697          .664673958
.256373663          .0123442287
.682952381         3.90587279E-04
.402342724          .879300926
.158209063          .245596701
```

Ваши числа не похожи? Было бы неправдоподобно, если бы они были похожи, так как программа генерирует совершенно случайный список из десяти чисел.

Если вы запустите программу несколько раз, вы увидите, что результаты все время разные. Хотя числа различные, вы заметите некоторую закономерность в списке, который выводит программа.

Во-первых, результат всегда между 1 и 0, но никогда не равен ни 1, ни 0. Во-вторых, числа являются реальнымм (с десятичной точкой).

Итак, мы собирались имитировать игру в кости и результаты этой программы не совсем те, которых мы ждем. Мы добавим несколько деталей к этой программе, чтобы получить то, что нам нужно.

Сначала добавьте эту строку к программе, заменив строку 20, и запустите программу опять:

```
20  PRINT 6*RND(1),
RUN

3.60563664              4.52687513
5.48602315              1.09650123
3.10045018              4.39052168
3.91302584              5.06321506
2.32056144              4.10781302
```

Сейчас мы получили результаты больше 1, но все еще имеем реальные числа. Для решения этого мы используем другую функцию.

Функция INT преобразует реальные числа в целые. Итак, попробуйте заменить строку 20 снова:

```
20  PRINT INT (6*RND(1)),
RUN

2   3   1   0
2   4   5   5
0   1
```

Сейчас хотя мы уже ближе к нашей цели, но вы заметите, что числа колеблются от 0 до 5, а не от 1 до 6. Итак в качестве завершающего шага, мы заменим строку 20 опять:

```
20  PRINT INT (6*RND(1))+1
```

Сейчас, когда вы запустите программу, вы получите те результаты, которые хотите. Когда вы хотите сгенерировать ряд реальных чисел вместо целых, формула немного изменится, так как вам придется вычесть нижний предел ряда из верхнего. Например, вы можете сгенерировать случайные числа между 1 и 25 набрав:

```
20 PRINT RND(1)*(25-1)+1
```

Общая формула для генерации случайных чисел в определенном диапазоне следующая:

```
NUMBER = RND(1)*(UPPER LIMIT-LOWER LIMIT)+LOWER LIMIT

число  = RND(1)*(верхний    - нижний    )+ нижний
                 предел       предел       предел
```

Игра на угадывание

Ниже представлена игра, использующая случайные числа. Эта игра не только использует функцию RND, но также дополнительно демонстрирует теорию программирования.

Когда вы запустите программу, компьютер сгенерирует случайное число, NM, значение которого вы должны угадать за как можно меньшее число попыток.

```
NEW
                    (1)
1    REM NUMBER GUESSING GAME
2    PRINT "(CLR/HOME)"
                          (2)
5    INPUT"ENTER UPPER LIMIT FOR GUESS";LI
10   NM=INT(LI*RND(1))+1
15   CN=0
                         (3)
20   PRINT"I'VE GOT THE NUMBER.":PRINT
                   (4)
30   INPUT"WHAT IS YOUR GUESS";GU
                                        (5)
40   IF GU>NM THEN PRINT"MY.NUMBER IS LOWER":PRINT:GOTO 30
                                    (6)
50   IF GU<NM THEN PRINT"MY NUMBER IS HIGHER":PRINT:GOTO 30
                                           (7)
60   IF GU=NM THEN PRINT"GREAT!YOU GOT MY NUMBER"
                         (8)
65   PRINT"IN ONLY";CN;"GUESSES.":PRINT
                                          (9)
70   PRINT"DO YOU WANT TO TRY ANOTHER(YIN)"
80   GET AN$:IF AN$=""THEN 80
90   IF AN$="Y" THEN 2
100  IF AN$<>"N"THEN 70
110  END
```

Вы можете задать верхний предел чисел в начале программы. Определение конкретного значения числа зависит уже от вас.

Обазец работы пограммы с пояснениями следует ниже:

```
ENTER UPPER LIMIT FOR GUESS? 25
I'VE GET THE NUMBER

WHAT'S YOUR GUESS? 15
MY NUMBER IS HIGHER

WHAT'S YOUR GUESS? 20
MY NUMBER IS LOWER

WHAT'S YOUR GUESS? 19
GREAT:YOU GOT MY NUMBER
IN ONLY 3 GUESSES

DO YOU WANT TO TRY ANOTHER(YIN)?
```

(1) Игра на угадывание чисел
(2) Введите верхний предел
(3) Я загадал число
(4) Ваше предположение?
(5) Мое число меньше
(6) Мое число больше
(7) Здорово! Вы отгадали мое число
(8) Всего за CN попытки
(9) Хотите ли продолжить (Д/Н)?

Оператор IF/THEN (строки 40-60) сравнивают число, предполагаемое вами со случайным (NM), сгенерированным строкой 10. Если ваше предположение не верно, программа подскажет, больше или меньше предполагаемое вами число, чем NM.

Каждый раз, когда вы предлагаете число, строка 35 прибавляет 1 к CN. CN - это счетчик попыток, которые вы сделали для угадывания числа. Цель этой игры, несомненно, - угадать число за возможно меньшее число попыток.

Когда вы дойдете до правильного ответа, программа напечатает сообщение: Здорово! Вы отгадали мое число и скажет вам, сколько попыток вы сделали.

Не забудьте, что прогамма создает новое случайное число каждый раз, когда вы играете в игру.

Вы можете также добавить несколько строк в программу, которые определяют нижний предел чисел, генерируемых в игре.

Примечание к программе

В строках 40 и 50 двоеточие разделяет разные операторы, входящие в одну строку. Это не только уменьшает время набора программы, но и экономит память.

Заметьте также, что оператор IF/THEN в этих двух строках печатает сообщение перед ветвлением на другую строку.

Игра в кости

Следующая программа имитирует бросок двух костей. Вы можете играть в эту игру как в самостоятельную или использовать ее как часть другой программы.

```
                              (1)
5   PRINT"CARE TO TRY YOUR LUCK?"
                 (2)
10  PRINT"RED DICE=";INT(RND(1)*6)+1
                  (3)
20  PRINT"WHITE DICE=";INT(RND(1)*6)+1
                        (4)
30  PRINT"PRESS SPACE BAR FOR ANOTHER ROLL":PRINT
40  GET A$:IF A$=""THEN 40
50  IF A$=CHR$(32) THEN 10
```

На основе уже приобретенных вами знаний о Бейсике и случайных числах постарайтесь разобраться, как работает эта программа. Как вы поняли из параграфа о программировании случайных чисел, CHR$(32) является кодом символа пробел.

(1) Не хотите ли попытать счастье?
(2) Красная кость
(3) Белая кость
(4) Нажмите пробел для следующей игры.

Случайная графика

Как последнее замечание о случайных числах, а также как введение в графику, попробуйте набрать и запустить следующую программу:

```
10  PRINT"<CLR/HOME>"
20  PRINT CHR$(205,5+RND(1));
30  GOTO 20
```

Функция CHR$ (символьная строка) дает вам символ, основанный на числовом коде от 0 до 255. Каждый символ, печатающийся в "64", закодирован таким способом. В Приложении F имеется список CHR$ - кодов для всех клавиш.

Быстрый способ определения кода любого символа - использование функции ASC (для стандартного ASCII кода). Наберите:

```
PRINT ASC("X")
```

X - это символ, который вы проверяете. Вместо X может быть любой печатаемый символ, включая графические. Вы должны заключить символ в кавычки. Например:

```
PRINT ASC("G")
71
```

Функция CHR является обратной к ASC.

```
PRINT CHR$(71)
G
```

Если вы наберете:

```
PRINT CHR$(205);CHR$(206)
```

Компьютер напечатает два правосторонних графических символа на клавишах M и N, которые используются в маленькой программе, которую вы только что набирали.

Формула 205,5+RND(1) "говорит" компьютеру выдать случайное число между 205,5 и 206,5. С вероятностью 50 на 50 случайное число будет больше или меньше 206. CHR игнорирует дробные значения и, таким образом, половину времени работы программы печатается символ с кодом 205, а оставшееся время - символ с кодом 206.

Вы можете экспериментировать с этой программой, добавляя или отнимая десятые доли от 205,5. Это дает какому-либо символу возможность быть напечатанным с большей вероятностью.

Глава 5. Развитые команды управления цветом и графикой

Цвет и графика

До сих пор мы исследовали некоторые из весьма сложных вычислительных возможностей компьютера "Коммодор-64". Однако одной из наиболее замечательных особенностей этого компьютера является его поразительная способность воспроизводить цвет и графику.

Вы уже видели примеры реализации графики в программах "прыгающий шарик" и "лабиринт". Однако эти примеры только незначительная часть возможностей компьютера. В настоящем разделе для объяснения программирования графики и цвета и показа того, как Вы сами можете создавать свои собственные игры и сложное оживление, вводится ряд новых понятий.

Так как до сих пор мы концентрировали свое внимание на вычислительных возможностях компьютера, все дисплеи, которыми мы пользовались до этого, были одноцветными (голубой текст на темносинем фоне, с голубой рамкой).

В настоящей главе мы увидим, как можно добавлять цвет к программам и управлять всеми этими странными символами на клавиатуре.

Цвет при печати

Как Вы уже знаете, если Вы рассматривали тест на установку цвета в Главе 1, Вы можете менять цвет текста простым нажатием на клавишу CTRL и на одну из клавиш цвета. Это очень хорошо работает в немедленном режиме, но что случится, если Вы захотите ввести изменения цвета в свои программы?

Когда мы рассматривали программу "прыгающий шарик", Вы видели, как команды с клавиатуры, как например, движение курсора, могут быть встроены в оператор PRINT. При желании Вы можете добавлять к Вашим программам также и изменение цвета текста.

Вы можете работать с полным цветовым диапазоном из 16 цветов. Используя клавишу CTRL и цифровые клавиши, Вы имеете доступ к следующим цветам:

1	2	3	4	5	6	7	8
Черный	Белый	Красный	Бирюзовый	Пурпурный	Зеленый	Синий	Желтый

Если Вы нажмете клавишу ц C= одновременно с сооветствующей цифровой клавишей, Вы можете получить дополнительно восемь цветов:

1	2	3	4	5	6	7	8
Оранжевый	Коричневый	Розовый	Серый 1	Серый 2	Светло-зеленый	Голубой	Серый 3

Наберите на клавиатуре NEW и проэкспериментируйте следующим образом. Нажмите клавишу CTRL и одновременно нажмите клавишу 1 . Затем нажмите клавишу R , не нажимая клавиши CTRL.

Теперь, снова нажимая клавишу CTRL, одновременно нажмите клавишу 2 . Освободите клавишу CTRL и нажмите клавишу A. Ударяйте по клавишам цифровым, чередуя их с буквенными и наберите слово RAINBOW (РАДУГА) следующим образом:

```
1Ø PRINT"  R  A  I  N  B  O  W"
    ----  -  -  -  -  -  -  -
   !CTRL!!1!!2!!3!!4!!5!!6!!7!
    ----  -  -  -  -  -  -  -
```

Прогон "радуги"

Точно так же как органы управления курсором проявляются как графические знаки при заключении между кавычек в операторах вывода на печать, так и органы управления цветом представляются как графические знаки.

В предыдущем примере, когда Вы удерживали в нажатом состоянии CTRL и нажимали цифровую клавишу, на дисплее появлялся соответствующий псевдосимвол. Каждый орган управления цветом будет появляться на дисплее в виде своего уникального графического кода, если он будет использоваться таким образом. В нижеприведенной таблице показано графическое представление каждого органа управления цветом.

1 KEYBOARD	2 COLOR	3 DYSPLAY	1 KEYBOARD	2 COLOR	3 DYSPLAY
CTRL 1	BLACK 4	псев-	C= 1	ORANGE 12	псев-
CTRL 2	WHITE 5	до-	C= 2	BROWN 13	до-
CTRL 3	RED 6	сим-	C= 3	LT.RED 14	сим-
CTRL 4	CYAN	во-	C= 4	GRAY 1 15	во-
CTRL 5	PURPLE 8	лы	C= 5	GRAY 2 16	лы
CTRL 6	GREEN 9		C= 6	LT.GREEN 17	
CTRL 7	BLUE 1Ø		C= 7	LT.BLUE 18	
CTRL 8	YELLOW 11		C= 8	GRAY 3 19	

Рис.24

1 - клавиатура; 2 - цвет; 3 - дисплей; 4 - черный; 5 - белый; 6 - красный; 7 - бирюзовый; 8 - пурпурный; 9 - зеленый; 1Ø - синий; 11 - желтый; 12 - оранжевый; 13 - коричневый; 14 - светлокрасный; 15 - серый 1; 16 - серый 2; 17 - светлозеленый; 18 - голубой; 19 - серый 3

Хотя оператор PRINT (ВЫВЕСТИ НА ПЕЧАТЬ) может показаться на экране несколько странным, когда вы осуществляете ПРОГОН (RUN) программы, на дисплей будет выводиться только текст. Этот текст будет автоматически менять свой цвет в соответствии с теми мерами по управлению цветом, которые Вы поместили в оператор вывода на печать.

Поэкспериментируйте сами, смешивая любое количество цветов в рамках одного оператора PRINT. Не забывайте, что Вы можете

также использовать второй набор текстовых цветов, используя клавишу C= компьютера "Коммодор-64" и цифровые клавиши.

Примечание. После прогона программы с изменениями (реверсными) цвета или режима Вы заметите, что подсказка "READY" (ГОТОВО) и любой дополнительный текст, который Вы набираете с клавиатуры имеет тот же цвет или режим, что и последнее изменение цвета или режима. Чтобы вернуться к нормальному дисплею, нажмите RUN/STOP и RESTORE.

Коды CHR$ цветной функции

Бегло просмотрите Приложение E, затем снова возвращайтесь к настоящему разделу.

Возможно, Вы заметили, когда просматривали список кодов функции CHR$ в Приложении E, что каждый цвет (так же, как и большинство других клавиатурных органов управления, как например, перемещение курсора) имеет уникальный код. Эти коды могут быть непосредственно выведены на печать с получением тех же результатов, что и при нажатии клавиши CTRL и соответствующей клавиши в операторе PRINT.

В качестве примера рассмотрите следующее:

```
! NEW                                                    !
! 10 PRINT CHR$ (147):REM <CLR/HOME>                     !
! 20 PRINT CHR$ (28);"CHR$(28)CHANGES ME TO?             !
! RUN                                                    !
! CHR$ (28)!CHANGES ME TO?                               !
```

Рис. 25

Сейчас текст должен иметь зеленый цвет. Во многих случаях применение функции CHR$ оказывается гораздо более удобным, особенно если Вы хотите поэкспериментировать с изменением цвета. На следующей странице описан другой способ получения всех цветов радуги. Так как существует целый ряд строк, которые аналогичны друг другу (40-110), для экономии времени при наборе с клавиатуры используйте редактирующие клавиши. Для того, чтобы освежить память в отношении редактирующих процедур, прочитайте примечания, приведенные после листинга

```
    NEW
1   1  REM AUTOMATIC COLOR BARS
    5  PRINT CHR$(147) : REM CHR$(147)= CLR/HOME
2   10 PRINT CHR$(18); "       ";: REM REVERSE BAR
    20 CL = INT(8*RND(1)) + 1
    30 ON CL GOTO 40,50,60,70,80,90,100,110
    40 PRINT CHR$(5);:GOTO 10
    50 PRINT CHR$(28);:GOTO 10
```

```
60 PRINT CHR$(30);: GOTO 10
70 PRINT CHR$(31);: GOTO 10
80 PRINT CHR$(144);: GOTO 10
90 PRINT CHR$(156);: GOTO 10
100 PRINT CHR$(158);: GOTO 10
110 PRINT CHR$(159);: GOTO 10
```

Рис.26

1 - ... автоматические цветные полосы; 2 - ... обратная полоса

Напечатайте в нормальном режиме строки с 5 по 40. На Вашем дисплее будет иметь место следующее сообщение:

```
!1 REM AUTOMATIC COLOR BARS                    !
!5 PRINT CHR$(147):REM CHR$(147)=CLR/HOME      !
!10 PRINT CHR$(18);"    ";:REM REVERSE BARS    !
!20 CL = INT(8*RND(1))+1                       !
!30 ON CL GOTO 40,50,60,70,80,90,100,110       !
!40 PRINT CHR$(5);:GOTO 10                     !
```

Рис.27

Замечания относительно редактирования

Для помещения курсора на строку 40 используйте клавишу CRSR-up (КУРСОР ВВЕРХ). Затем напечатайте 5 вместо 4 в числе 40. Затем используйте клавишу CRSR-right (КУРСОР ВПРАВО) для перемещения к 5 в скобках CHR$. Нажмите клавиши SHIFT и INST/DEL чтобы получить пробел и впечатайте "28". Теперь нажмите клавишу RETURN , при этом курсор находится где-нибудь на строке.

На дисплей будет выведено следующее сообщение:

```
!1 REM AUTOMATIC COLOR BARS                    !
!5 PRINT CHR$(147):REM CHR$(147)CLR/HOME       !
!10 PRINT CHR$(18);"    ";:REM REVERSE BAR     !
!20 CL=INT(8*RND(1))+1                         !
!30 ON CL GOTO 40,50,60,70,80,90,100,110       !
!50 PRINT CHR$(28);:GOTO 10                    !
```

Рис.28

Не беспокойтесь. Строка 40 по-прежнему присутствует здесь. Выведите программу (LIST) и посмотрите. Используя ту же процедуру, продолжайте модифицировать последнюю строку новым номе-

ром строки и кодом CHR$, пока не будут введены все оставшиеся строки. Как мы и говорили ранее, редактирующие клавиши оказались очень полезными. В качестве окончательной проверки выведите листинг всей программы, чтобы убедиться, что все строки были введены правильно, перед тем как Вы осуществили ее прогон (RUN).

Ниже дается объяснение того, что имеет место.

Вероятно, Вы поняли большую часть программы цветных полос, за исключением несколько странного нового оператора в строке 30. Однако давайте быстро оценим, что на деле выполняет вся программа. Строка 5 печатает код CHR$ для CLR/HOME (ОЧИСТКА/ИСХОДНОЕ ПОЛОЖЕНИЕ).

Строка 10 включает реверсную (обратную) печать и печатает 5 пробелов, которые оказываются полосой, так как они реверсированы. Впервые во время реализации программы полоса будет голубого цвета - нормального текстового цвета.

Строка 20 использует уже известную нам случайную функцию для выбора случайного цвета в диапазоне между 1 и 8.

Строка 30 содержит вариант оператора IF ... THEN, который называется ON ... GOTO (ПРИ ... ПЕРЕХОДИТЕ К). Этот оператор позволяет программе осуществлять выбор из списка номеров строки, к которому следует переходить. Если переменная (в этом случае CL) имеет величину 1, выбирается номер первой строки (здесь 40). Если эта величина равна 2, то используется второй номер в списке и т.д.

Строки 40-110 преобразуют наши случайные цвета, полученные с помощью клавиш в соответствующий код CHR$ для этого цвета и возвращают программу к строке 10 для вывода на печать (PRINT) части полосы в этом цвете. Затем весь процесс начинается сначала.

Проверьте, знаете ли Вы, как получить 16 случайных чисел, расширить оператор ON ... GOTO для их обработки и добавить остальные коды CHR$ для вывода на дисплей остальных восьми цветов.

Операторы PEEK и POKE

Нет, мы не собираемся влезать в компьютер, но мы сможем "осмотреться" внутри компьютера, и посмотреть, как там обстоят дела.

Точно так же, как переменные могут быть представлены в виде содержимого "яшиков" в компьютере, куда Вы помещаете Вашу информацию, Вы также можете представить себе специальным образом определенные "ящики" в компьютере, которые представляют специальные ячейки памяти.

Компьютер "Коммодор-64" обращается к этим ячейкам памяти, чтобы определить, какого цвета должен быть фон экрана и рамка экрана, какие следует выводить на дисплей и где именно, а также решить большое количество других задач.

Помещая (POKE) различные величины в соответствующие ячейки памяти, мы можем менять цвет, определять и перемещать объекты и даже создавать музыку.

Эти ячейки памяти могут быть представлены следующим образом:

Рис.29
1 - цвет рамки экрана; 2 - цвет фона

Здесь показаны только четыре ячейки, две из которых управляют цветом экрана и фона. Напечатайте следующее:

```
POKE 53281,7 RETURN
```

Цвет фона экрана сменится на желтый, так как мы поместили величину "7" - соответствующую желтому цвету - в ячейку, которая управляет цветом фона экрана.

Попробуйте вставлять (POKE) разные величины в ячейку, управляющую цветом фона, и посмотрите, что из этого получится. Вы можете вставлять (POKE) в эту ячейку любую величину от Ø до 255, но рабочими величинами будут величины от Ø до 15.

Каждому цвету при работе оператора POKE соответствуют следующие величины:

Ø	Черный	8	Оранжевый
1	Белый	9	Коричневый
2	Красный	1Ø	Светлокрасный
3	Бирюзовый	11	Серый 1
4	Пурпурный	12	Серый 2
5	Зеленый	13	Светлозеленый
6	Синий	14	Голубой
7	Желтый	15	Серый 3

Подумайте, каким способом можно вывести на дисплей различные комбинации фона и рамки. Следующая программа может оказаться полезной в этом отношении.

```
NEW
1Ø FOR BO = Ø TO 15
2Ø FOR BA = Ø TO 15
3Ø POKE 53280,BA
4Ø POKE 53281,BO
5Ø FOR X = NEXT BO
6Ø NEXT BA : NEXT BO
RUN
```

Рис.3Ø

Для того, чтобы ВСТАВИТЬ (POKE) различные величины для изменения цвета фона и цвета рамки, было создано два простых цикла. Цикл DELAY (ЗАДЕРЖКА) в строке 50 просто несколько замедляет выполнение операций.

Для любознательных рекомендуем следующий оператор

```
? PEEK (53280) AND 15
```

Вы должны получить величину 15. Это последняя величина для рамки экрана (BORDER), и она имеет смысл, так как цвет рамки и цвет фона - СЕРЫЙ (GRAY), что соответствует величине 15, после того как осуществлен прогон программы.

Введением слова AND (И) 15 Вы вычеркиваете все остальные величины, кроме 1-15. Это происходит благодаря способу хранения в компьютере кодов цвета. Обычно мы рассчитываем найти ту же величину, которая была последней вставлена (POKE) в ячейку. В общем случае оператор PEEK позволяет нам исследовать специальную ячейку и посмотреть, какая величина находится там в данный момент. Можно ли сделать к программе добавление объемом в одну строку, которое выведет на дисплей величину фона (BACK) и рамки (BORDER) во время прогона программы? Рассмотрите следующее выражение:

```
25 PRINT CHR$(147); "BORDER = ";PEEK (53280) AND 15,
   "BACKGROUND ="; PEEK (53281) AND 15
```

Графика на экране

Во всех случаях вывода информации на печать, которые мы до сих пор имели, компьютер обычно обрабатывал информацию последовательно: один знак печатался после другого, начиная от текущего положения курсора (за исключением случаев, когда Вы запрашивали новую строку или использовали при форматировании печати оператор PRINT.

Чтобы ВЫВЕСТИ НА ПЕЧАТЬ (PRINT) данные в конкретном месте, Вы можете начать от известного места на экране и выводить на печать необходимое количество средств управления курсором для форматирования дисплея. Однако это требует команд в программе, а также времени.

Однако, аналогично тому что в памяти компьютера "Коммодор-64" имеются определенные места, предназначенные для управления цветом, так существуют и ячейки памяти, которые Вы можете использовать для непосредственного управления каждой ячейкой на экране.

Карта экранной памяти

Так как экран компьютера вмещает 1000 знаков (40 колонок по 25 строк), имеются 1000 ячеек памяти, которые отведены для обработки информации, выводимой на экран. В этом плане экран можно представить в виде сетки, каждый квадрат которой представляет ячейку памяти.

Так как каждая ячейка в памяти может содержать число от Ø до 255, для каждой ячейки памяти существуют 256 возможных величин. Эти величины представляют собой различные знаки, которые компьютер "Коммодор-64" может выводить на дисплей (см. Приложение Д). Вставляя (POKE) величину для знака в соответствующую экранную ячейку памяти, мы будем выводить этот знак на соответствующую позицию дисплея.

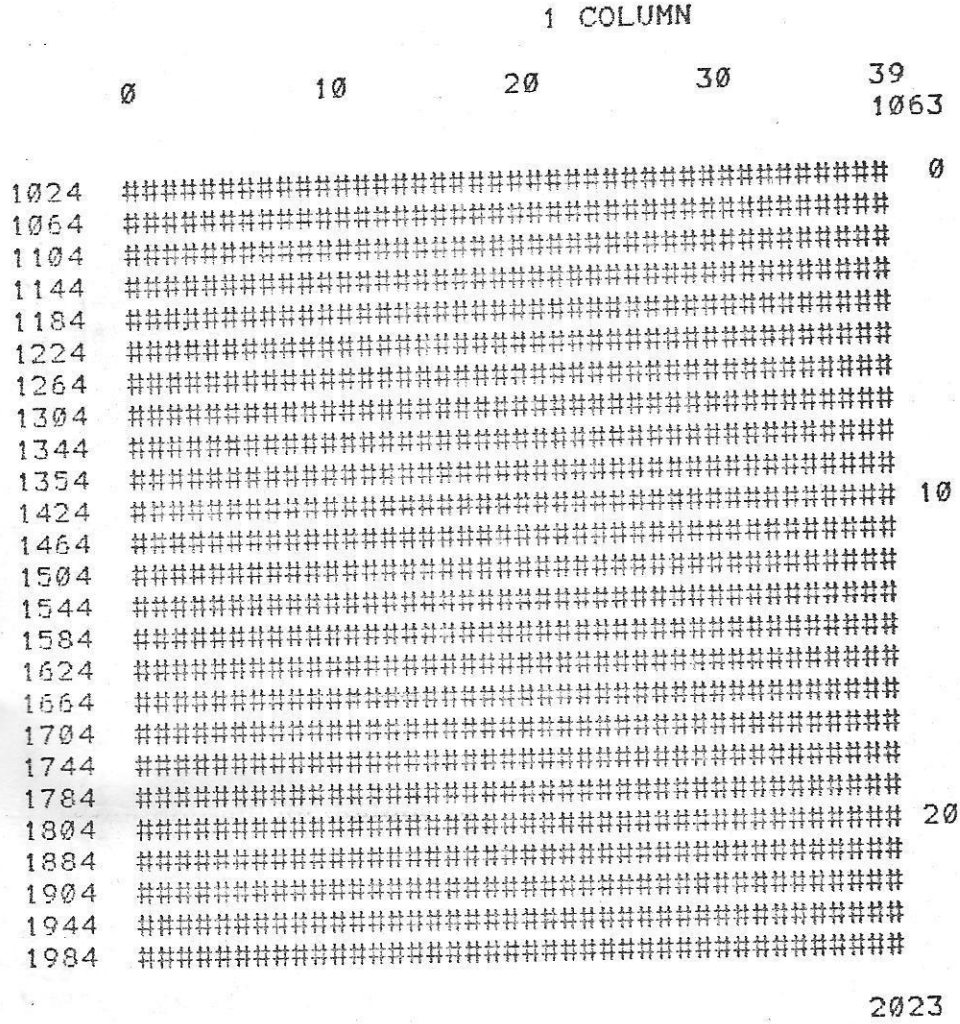


Рис.31
1 - колонка; 2 - ряд

Экранная память компьютера "Коммодор-64" начинается с ячейки памяти 1Ø24 и заканчивается ячейкой памяти 2Ø23. Ячейка 1Ø24 находится в левом верхнем углу экрана. Ячейка 1Ø25 занимает положение рядом справа от ячейки 1Ø24. Ячейка 1Ø63 занимает крайнее правое положение в первом ряду. Ячейка, следующая за последней ячейкой каждого ряда, занимает первое место (крайнее левое) в следующем ряду.

Допустим, что мы хотим проконтролировать движение прыга-

ющего шарика на экране. Шарик находится в центре экрана: колонка 20, ряд 12. Формула для вычисления ячейки памяти имеет вид:

POINT = 1024+X+40*Y,

где POINT - ячейка памяти, X - номер колонки, Y - номер ряда.

Следовательно, номер ячейки шарика вычисляется таким образом:

1024 + 20 + 480, т.е. 1524,

где 20 - номер колонки, 480 - произведение 40 и номера ряда (12).

Очистите экран с помощью клавиш SHIFT и CLR/HOME и напечатайте: POKE 1524;81, где 1524 - номер ячейки, 81 - код знака.

Карта памяти цвета

Шарик появляется в центре экрана! Вы поместили знак непосредственно в экранную память, не используя оператор PRINT. На экране - шарик белого цвета. Однако, можно изменять цвет объекта на экране. Это осуществляется изменением другой области памяти. Наберите на клавиатуре:

POKE 55796,2 Здесь 55796 - ячейка, 2 - код цвета

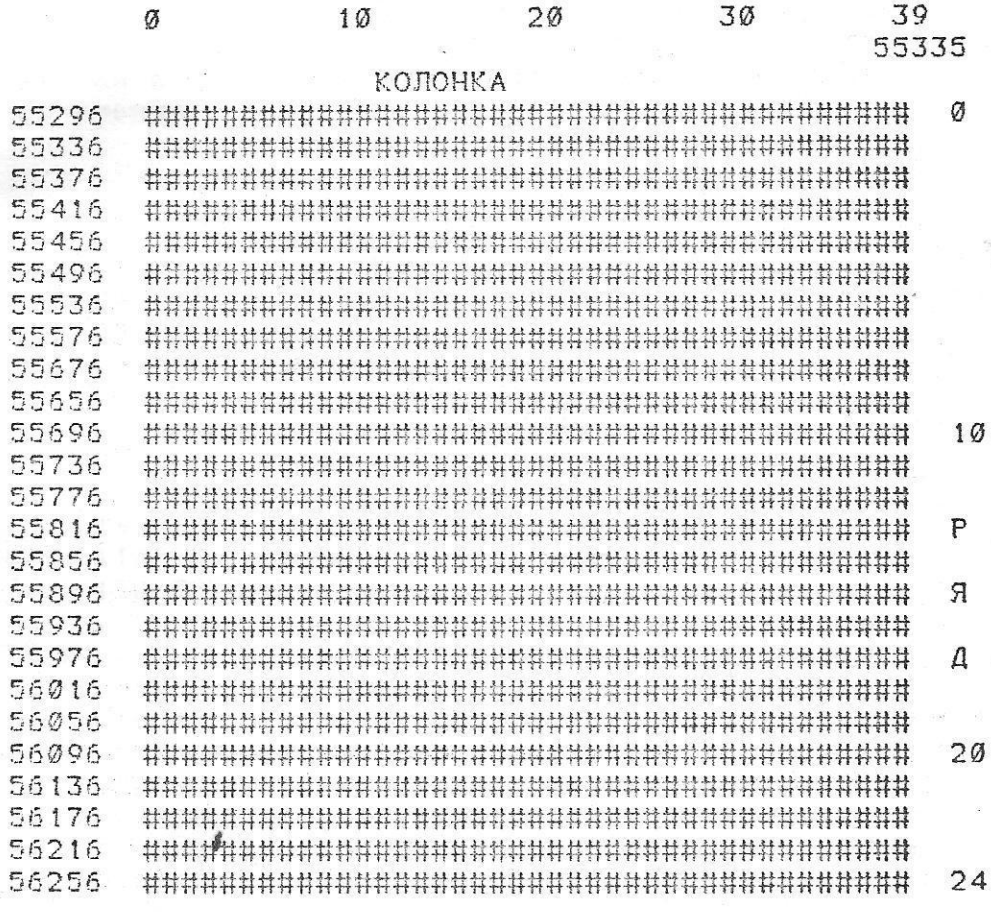


Рис. 32

Цвет шарика изменится на красный. Для каждого места на экране дисплея компьютера имеются две ячейки памяти: одна - для кода знака, другая - для кода цвета. Карта памяти цвета начинается с ячейки 55296 (левый верхний угол) и насчитывает 1000 ячеек. Коды цвета (0-15) те же, которые мы использовали для изменения цвета фона и рамки, и могут быть непосредственно использованы для изменения цвета знаков.

Формула для вычисления ячеек экранной памяти может быть модифицирована для получения номеров ячеек для вставления (POKE) кода цвета. Эта новая формула имеет вид

```
COLOR PRINT = 55296 + X + 40*Y,
```

что в данном случае соответствует прибавлению 54272 к соответствующему номеру ячейки экранной памяти.

Много прыгающих шариков

Ниже приведена пересмотренная программа прыгающего шарика, которая осуществляет вывод на дисплей непосредственно с помощью оператора POKE, а не с помощью управления курсором внутри оператора PRINT. Как Вы убедитесь после прогона этой программы, она является намного более гибкой, чем та программа, и дает нам возможности для программирования гораздо более сложного оживления.

```
NEW

10 POKE 53281,1:PRINT"<CTRL/WHITE><SHIFT CLR/OOME>"
20 POKE 53280,7 : POKE 53281,16
30 X = 1 : Y = 1
40 DX = 1 : DY = 1
50 POKE 1024 + X + 40*Y,81
60 FOR T = 1 TO 10 : NEXT
70 POKE 1024 + X + 40*Y,32
80 X = X + DX
90 IF X <= 0 OR X>= 39 THEN DX = -DX
100 Y = Y + DY
110 IF Y <= 0 OR Y >= 24 THEN DY = -DY
120 GOTO 50
```

Рис. 33

Строка 10 очищает экран, а строка 20 устанавливает желтый цвет фона и желтый цвет рамки.

Переменные X и Y в строке 30 следят за текущим положением ряда и колонки шарика. Переменные DX и DY в строке 40 представляют собой горизонтальное и вертикальное направления перемещения шарика. Когда к величине X добавляется +1, шарик движется вправо, когда добавляется -1, то шарик движется влево. Добавление +1 к Y перемещает шарик вниз на один ряд, добавление к Y -1 перемещает шарик вверх на один ряд.

Строка 50 помещает шарик на экране в текущее положение курсора. Строка 60 представляет собой уже знакомый нам цикл задержки, оставляя шарик на экране в течение достаточно долгого времени, чтобы его было удобно наблюдать.

Строка 70 стирает шарик, помещая его на то место, где только что был пробел (код 32).

Строка 80 добавляет координате X фактор направления. Строка 90 проверяет, достиг ли шарик одной из боковых "стенок", и в случае такого достижения реверсирует направление движения. Строки 100 и 110 осуществляют то же самое для вертикальной и горизонтальной "стенок".

Строка 120 отсылает программу обратно на дисплей, и шарик повторяет движение.

Изменяя код в строке 50 с величины 81 на какой-либо другой код знака, Вы можете сменить шарик на какой-либо другой знак. Если Вы замените DX или DY нулем, то шарик будет перемещаться вдоль краев экрана, а не по диагонали.

Мы можем несколько усложнить программу. До сих пор в отношении величин X и Y осуществлялся только контроль в отношении недопустимости их выхода за крайние пределы экрана. Добавьте в программу следующие строки:

```
                                          1
21 FOR L = 1 TO 10                        POKE CODE
25 POKE 1024 + INT<RND<1>*1000>, 160      (REVERSE SPACE)
27 NEXT L
115 IF PEEK <1024 + X + 40*Y> = 166 THEN DX = -DX:
    GOTO 80
```

Рис.34
1 - код POKE

Строки с 21 по 27 устанавливают на экране в произвольных местах десять блокировок. Строка 115 проверяет (PEEK), попадает ли на следующем шаге шарик в место блокировки, и если это так, то меняет направление движения шарика.

*)

Глава 6. Спрайт-графика

Введение в спрайты

В предыдущих главах, касавшихся графики, мы видели, что графические символы могут использоваться в операторах PRINT для создания оживления и добавления графикоподобных изображений на экране дисплея.

Был показан также способ вставления (POKE) кодов знаков в

---

*) Английское слово sprite означает "эльф". Здесь "спрайты" - это движущиеся (оживленные) изображения на экране дисплея (прим.пер.).

те или иные ячейки памяти. Тогда это приводило к непосредственному помещению соответствующих знаков в нужное место на экране.

Создание оживления в обоих этих случаях требовало большого количества работы, так как объекты должны были создаваться из существующих графических символов. Перемещение объектов требует ряда программных операторов для слежения за объектом и перемещения его в другое место. И вследствие того, что на применение графических символов накладываются определенные ограничения, форма и четкость объектов могут не удовлетворять предъявляемым требованиям.

Многие из этих проблем устраняются при использовании спрайтов в оживляющих последовательностях. Спрайт представляет собой программируемый объект с высоким разрешением, который может принимать практически любую форму - с помощью команд языка БЕЙСИК. Этот объект может быть легко перемещен по экрану с помощью простого указания компьютеру, в какое положение следует переместить спрайт. Остальное компьютер осуществляет сам.

Спрайты обладают гораздо большими возможностями, чем только что описанная. Их цвет может меняться; Вы можете сказать, столкнется ли один объект с другим; их можно заставить появляться друг перед другом или друг за другом, и, кроме того, их можно легко увеличивать в размерах, хотя бы для стартеров.

Затраты на все это минимальны. Однако, применение спрайтов требует знания некоторых дополнительных подробностей о работе компьютера "Коммодор-64" и о том, как числа обрабатываются внутри компьютера. Однако, это не так сложно, как кажется сначала. Вам предлагается только разобрать предлагаемые примеры, и Вы сможете создавать свои собственные спрайты и заставлять их выполнять удивительные действия, причем на это освоение не потребуется много времени.

Создание спрайтов

Спрайты управляются отдельной программой создания изображений в компьютере "Коммодор-64". Эта программа управляет видеодисплеем. Она выполняет всю сложную работу по созданию и организации траекторий знаков и графики, создает цвет и осуществляет перемещение объектов.

Цепь дисплея имеет 46 различных ячеек "ON/OFF" (ВКЛЮЧЕНО/ВЫКЛЮЧЕНО), которые действуют как внутренние ячейки памяти. Каждая из этих ячеек подразделяется на последовательность из восьми блоков. Каждый блок может находиться в состоянии "включено" или "выключено". Подробнее об этом будет сказано позже. Вставлением (POKE) соответствующей десятичной величины в нужную ячейку памяти Вы можете управлять созданием и перемещением своих спрайтов.

Кроме осуществления доступа ко многим ячейкам создания изображений, мы также будем использовать некоторые ячейки основной памяти компьютера для запоминания информации (данных), которая будет определять спрайты. Наконец, восемь ячеек памяти, стоящие непосредственно после экранной памяти, будут использоваться для указания компьютеру, из какой конкретно зо-

ны памяти каждый спрайт будет получать свою информацию.

Когда мы будем приводить примеры, этот процесс будет ясно показан, и Вы быстро его освоите.

Начнем с создания графики спрайтов. Спрайтовый объект состоит из 24 точек в ширину и 21 точки в длину. Одновременно можно управлять не более чем восемью спрайтами. Спрайты выводятся на дисплей в специальном режиме с высоким разрешением, в котором экран превращается в зону шириной в 320 точек и высотой в 200 точек.

Допустим, Вы хотите создать воздушный шар и пустить его плавать по небу. Воздушный шар может быть представлен в сетке размером 24 на 21 .

Следующим шагом является преобразование графического чертежа в данные, которые могут быть использованы компьютером. Возьмите листок бумаги в клеточку и начертите эталонную сетку размером 21 клеточка вниз и 24 клеточки в ширину. Наверху этого прямоугольника напишите числа 128, 64, 32, 16, 8, 4, 2, 1 три раза над каждой из 24 клеточек .
Слева пронумеруйте каждый ряд клеточек Вашей сетки от 1 до 21. В конце каждого ряда напишите слово DATA (ДАННЫЕ). Теперь заполните Вашу сетку любым изображением или же изображением воздушного шара, который мы предлагаем на рисунке. Лучше сначала обрисовать контуры объекта, а затем заполнять сетку.

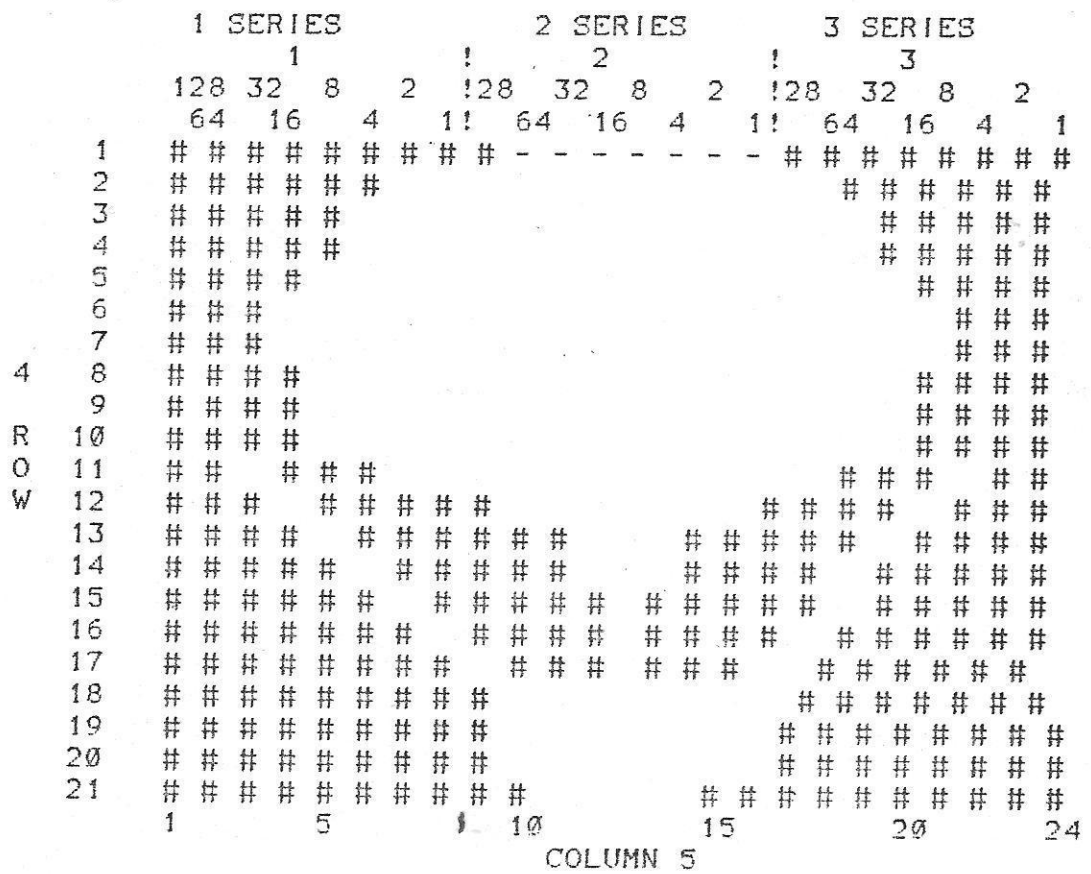


Рис. 35

1 - последовательность 1; 2 - последовательность 2; 3 - последовательность 3; 4 - ряд; 5 - колонка

Теперь все клетки, которые участвуют в изображении, считаются "включенными", и каждую из таких клеток замените единицей. Клетки, которые в изображении не участвуют, считаются "выключенными" и их следует заменить нулями.

Вам необходимо, начиная с первого ряда, преобразовать точки в три отдельных последовательности данных, которые компьютер может прочитать. Каждый набор из восьми клеток равен одной последовательности данных, называемой байтом. Если мы начнем слева, то мы видим, что первые восемь клеток не участвуют в создании изображения, являются пустыми, и поэтому им соответствует Ø, поэтому величина для этой последовательности равна нулю.

Средний (второй) байт первого ряда выглядит следующим образом (по-прежнему 1 соответствует точке, Ø соответствует пробелу):

```
  128   64    32    16     8     4     2     1
 -----------------------------------------------
!  Ø  !  1  !  1  !  1  !  1  !  1  !  1  !  1  !
 -----------------------------------------------

   Ø   + 64   + 32  + 16  + 8   + 4   + 2   + 1  = 127
```

Третья последовательность (третий байт) первого ряда также содержит одни нули и поэтому вся величина также равна нулю. Таким образом, данные для первой строки (ряда) имеют вид:

```
DATA Ø, 127, Ø
```

Последовательности, которые составляют второй ряд, вычисляются следующим образом:

```
                 --------------------------------
Series 1:       ! Ø ! Ø ! Ø ! Ø ! Ø ! Ø ! Ø ! 1 !
  1              --------------------------------
                                              1 = 1

                 --------------------------------
Series 2:       ! 1 ! 1 ! 1 ! 1 ! 1 ! 1 ! 1 ! 1 !
  2              --------------------------------

                 128  +64 +32 +16 +8  +4  +2  +1  = 255

                 --------------------------------
Series 3:       ! 1 ! 1 ! Ø ! Ø ! Ø ! Ø ! Ø ! Ø !
  3              --------------------------------

                 128  +64                        = 192
```

Рис.64:
1 - последовательность 1; 2 - последовательность 2;
3 - последовательность 3

Для ряда 2 данные будут иметь следующий вид:

```
DATA 1,255,192
```

Таким же образом содержимое трех последовательностей, составляющих каждый ряд, преобразуется в десятичную величину для каждого ряда.

Возьмите на себя труд сделать остальные преобразования для этого примера.

Теперь, когда Вы имеете данные для Вашего объекта, как Вы их можете использовать? Наберите с клавиатуры следующую программу и посмотрите, что получится.

```
1     1 REM UP, UP, AND AWAY:
      5 PRINT "(CLR/HOME)"
2    10 V = 53248  :  REM START OF DISPLAY CHIP
3    11 POKE V+21,4 : REM ENABLE SPRITE 2
4    12 POKE 2042,13  : REM SPRITE 2 DATA FROM BLOCK 13
     20 FOR N = 0 TO 62 : READ Q : POKE 832+N,Q: NEXT
     30 FOR X = 0 TO 200    !---> GETS ITS INFO FROM DATA  5
6    40 POKE V+4,X: REM UPOATE X COORDINATES
7    50 POKE V+5,X: REM UPOATE Y COORDINATES
     60 NEXT X
     70 GOTO 30
           !---->INFO, READ IN FROM Q    8
    200 DATA 0,127,0,1,255,192,3,255,224,3,231,224
    210 DATA 7,217,240,7,223,240,7,217,240,3,231,224
    220 DATA 3,255,224,3,255,224,2,255,160,1,127,64
    230 DATA 1,62,64,0,156,128,0,156,128,0,73,0,0,73,0
    230 DATA 1,62,0,0,62,0,0,62,0,0,28,0
9 FOR MORE DATA ON READ 6 DATA SEE CHAPTER 8.
```

Рис.36

1 - ...вверх, вверх и вдаль; 2- начало дисплейного кадра; 3 - ...запустить спрайт 2; 4 - ...данные спрайта 2 из 13-го блока; 5 - берет свою информацию из данных; 6 - ...модификация координаты X; 7 - ...модификация координаты ; 8 - ...информация, считанная из Q; 9 - ...более подробно о считывающих данных смотрите в Главе 8

Если Вы все набрали правильно, Ваш воздушный шар будет плавно лететь по небу .

Чтобы понять, что произошло, прежде всего Вы должны знать, каким образом ячейки, создающие изображение, управляют необходимыми Вам функциями. Эти ячейки, которые называются регистрами, иллюстрируются следующим образом:

Регистр(ы)	Описание
0	Координата X спрайта 0
1	Координата Y спрайта 0
2 - 15	Аналогичные пары координат для спрайтов 1-7
16	Наибольший значащий бит - координата X
21	Появление спрайта: 1 - появляется, 0 - исчезает
29	Расширение спрайта в направлении оси "X"
23	Расширение спрайта в направлении оси "Y"
39 - 46	Цвета спрайта от 0 до 7

Помимо этого, Вы должны также дать каждому спрайту указания, откуда ему брать свою информацию (данные).

Эти данные обрабатываются восемью ячейками, стоящими непосредственно после экранной памяти:

2040	41	42	43	44	45	46	2047
спрайт	0	1	2	3	4	5	6	7

Давайте теперь опишем точную процедуру приведения изображений в движение и после этого напишем соответствующую программу.

Для действительного создания объекта и перемещения его на экране необходимо знать всего несколько моментов.

1. Заставьте спрайт(ы) появиться на экране с помощью вставления (POKE) в ячейку 21, что включает спрайт.
2. Установите указатель спрайта (ячейки 2040-2047) на то, откуда следует считывать данные спрайта.
3. ВСТАВЬТЕ (POKE) реальные данные в память.
4. С помощью цикла модифицируйте координаты X и Y для перемещения спрайта.
5. Вы можете, если хотите, расширить объект, изменить цвета или осуществить ряд специальных функций. С помощью ячейки 29 Вы можете расширить Ваш спрайт в направлении оси "X", а с помошью ячейки 23 - в направлении "Y".

В программе имеется только несколько пунктов, которые, возможно еще не были упомянуты ранее.

В строке 10 выражение V=53248 устанавливает V в стартовую ячейку памяти видеомодуля. Таким образом мы просто увеличиваем величину V на число в памяти для получения действительного номера ячейки памяти. Регистровые числа приведены в регистровой карте.

В строке 11 выражение POKE V+21,4 заставляет появиться спрайт 2 с помошью помещения 4 в регистр 21, называемый регистром разрешения (enable register) для включения спрайта 2. Это иллюстрируется следующим:

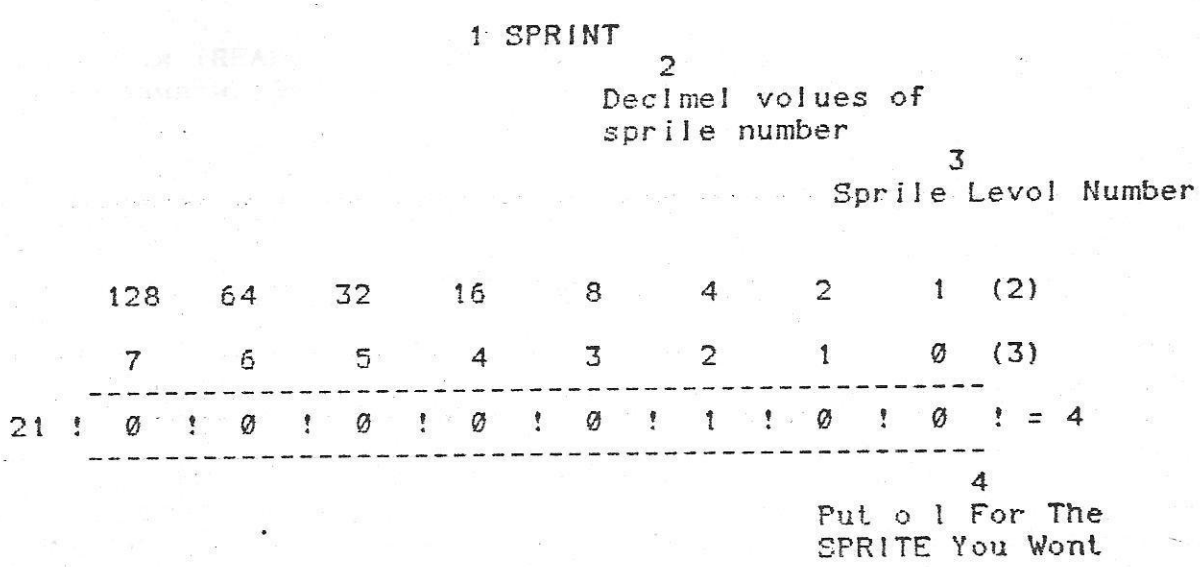


Рис.37

1 - спрайты; 2 - десятичные величины, соответствующие каждому номеру спрайта; 3 - номер уровня спрайта; 4 - поместите 1 в ячейку того номера спрайта, который вы хотите вызвать

В регистре 21 памяти спрайтов представлен уровень каждого спрайта, и уровень спрайта 2 оказывается равным 4. Если бы Вы использовали уровень 3, то Вам было бы необходимо поместить 1 в ячейку, соответствующую в регистре спрайту 3, что соответствовало бы десятичной величине 8. Если же Вы хотите использовать спрайты 2 и 3 одновременно, Вы должны поместить единицы в ячейки этих спрайтов, т.е. в ячейки с "весом" 4 и 8. Тогда Вы должны были бы сложить эти числа, точно так же, как Вы делали с данными (DATA) на Вашем листочке в клеточку. Поэтому одновременное включение спрайтов 2 и 3 было бы представлено выражением V+21,12.

В строке 12 выражение POKE 2Ø42,13 дает компьютеру указание взять данные для спрайта 2 (ячейка 2Ø42) из 13-й зоны памяти. Из процесса создания своего спрайта Вы знаете, что он забирает до 63 регистров памяти. Возможно, Вы этого еще не осознали, но те три числа, которые Вы надписали над Вашей сеткой, соответствуют тому, что называется тремя байтами компьютера. Другими словами, каждый набор чисел 128,64,32,16,8,4,2,1 соответствует одному байту компьютерной памяти. Поэтому если мы умножим 21 ряд Вашей сетки на 3 байта, имеющиеся в каждом ряду, то увидим, что каждый спрайт занимает 63 байта в памяти компьютера

1 целый байт

```
20 FOR N = Ø to 63: READ Q:POKE 832 +N,Q:NEXT
```

Эта строка реализует действительное создание спрайта. 63 байта данных, которые представляют созданный Вами спрайт, счи-

тываются (READ) в процессе цикла и вставляются (POKE) в 13-й блок памяти. Это начинается в ячейке 832 (13*64).

```
30 FOR X = 0 TO 200                                    1
                                    SPRITE 2s X COORDINATE
40 POKE V + 4,X                                        2
                                    SPRITE 2s Y COORDINATE
50 POKE V + 5,X
```

Рис. 38

1 - координата X спрайта 2; 2 - координата Y спрайта 2

Как Вы помните из школьной математики, координата X представляет горизонтальное перемещение на экране, в то время как координата Y представляет вертикальное перемещение спрайта по экрану. Поэтому, так как величины X меняются в строке 30 от 0 до 200 (одно число за каждый такт), то спрайт перемещается по экрану ВНИЗ и ВПРАВО на одну позицию для каждого числа. Числа считываются компьютером достаточно быстро, чтобы создать впечатление плавного движения (а не шагового). Если Вы желаете получить об этом более подробную информацию, посмотрите на регистровую карту в Приложении П.

Когда Вы будете заниматься несколькими движущимися объектами, Вы увидите, что одной секции (байта) памяти недостаточно для модификации ячеек всех восьми объектов. Поэтому каждый спрайт имеет свой собственный набор из двух секций памяти, позволяющих осуществлять его перемещение на экране.

Строка 70 запускает цикл заново, после одного прохода спрайта по экрану. Остальная часть программы представляет собой данные для воздушного шара. На экране он выглядит по-разному, не правда-ли?

Теперь добавьте к программе следующую строку:

```
25 POKE V + 23,4: POKE V + 29,4 : REM EXPAND SPRITE
```

и осуществите ПРОГОН (RUN) программы заново. Воздушный шар увеличился в два раза по сравнению с первоначальным изображением! Мы осуществили это увеличение весьма просто. С помощью вставления (POKE) 4 (опять для индикации спрайта 2) в секции памяти 23 и 29 спрайт 2 был расширен в направлении осей X и Y.

Важно отметить, что спрайт стартует в левом верхнем углу блока спрайта. При расширении объекта в любом направлении стартовая точка остается той же самой.

Чтобы наблюдать еще один эффект, осуществите следующие изменения:

```
11 POKE V + 21,12
12 POKE 2042,13 : POKE 2043,13
20 FOR X = 1 TO 190
45 POKE V + 6,X
55 POKE V + 7,190 - X
```

С помощью вставления 12 в ячейку памяти, которая заставляет спрайт появляться (V+21) был включен второй спрайт (номер 3). Число 12 включает спрайты 3 и 2 (ØØØØ1100 соответствует 12).

Добавленные строки 45 и 55 перемещают спрайт 3 по кругу с помощью ВСТАВЛЕНИЯ (POKE) величин в ячейки координат X и Y спрайта 3 (V+6 и V+7).

Если Вы хотите иметь в небе еще больше движения, попробуйте внести следующие изменения:

```
                              1  28=4+8+16 = SPRITES
                                   2,3 + 4 RESPECTINELY
    11 POKE V + 21,28
    12 POKE 2042,13:POKE 2043,13::POKE 2044,13
2   13 REM TELL EACH SPRITE WHERE TO GET DATA
3   25 POKE V + 23,12:POKE V + 29,12:REM EXPAND SPRITES 2 AND 2
    48 POKE V + 8,X
    58 POKE V + 9,100
```

Рис.39

1 - 28 = 4 + 8 + 16 = спрайты 2,3 + 4 соответственно;
2 - ...говорит каждому спрайту, где взять данные;
3 - ...расширение спрайта 2 и спрайта 3

Строка 12 указывает, что спрайт 4 будет получать свои данные из той же зоны памяти (13-я зона из 63 секций), как и другие спрайты (это осуществлено вставлением (POKE) 2044,13).

В строке 25 спрайты 2 и 3 расширены путем вставления (POKE) 12 (включены спрайты 2 и 3) в ячейки памяти расширения в направлении осей X и Y (Y+23 и Y+29).

Строка 48 перемещает спрайт 3 по оси X. Строка 58 устанавливает спрайт 3 на полпути вниз по экрану, в ячейке 100. Так как эта величина не меняется, как это имело место раньше, когда X менялся от Ø до 200, спрайт 3 перемещается только горизонтально.

Дополнительные замечания относительно спрайтов

Теперь, когда Вы поэкспериментировали со спрайтами, имеет смысл сказать еще несколько слов. Во-первых, Вы можете менять цвет спрайта на любой стандартный с помощью кодов цвета (Ø-15), которые были использованы для изменения цвета знаков. Подробнее об этом можно прочитать в Главе 5 или в Приложении Ж.

Например, для того чтобы сменить цвет спрайта 1 на светло-зеленый, наберите на клавиатуре : POKE V+40,13 (не забудьте установить V=53248).

Вы, возможно, заметили при работе с эталонными спрайтовыми программами, что объект никогда не перемещается к правой стороне экрана. Это имело место потому, что экран имеет в ширину 320 точек, а регистр направления X может вместить величину не более 255. Как же в таком случае заставить объект перемещаться по всему экрану?

На карте памяти имеется ячейка, о которой до сих пор мы не упоминали. Ячейка 16 (карты памяти) управляет наибольшим значащим битом ячейки направления X спрайта. На практике это позволяет осуществить перемещение спрайта в горизонтальные точки между 256 и 320.

Наибольший значащий бит регистра X работает следующим образом: после того, как спрайт переместился в ячейку 255 координаты X, поместите в ячейку памяти 16 величину, представляющую спрайт, который Вы хотите переместить. Например, чтобы переместить спрайт 2 в горизонтальные ячейки 256-320, ВСТАВЬТЕ (POKE) величину для спрайта 2 (это будет 4) в ячейку памяти 16:

```
POKE V+16,4
```

Теперь начинайте осуществлять перемещение в обычном регистре направления X для спрайта 2 (который находится в ячейке 4 карты), снова начиная с 1. Так как теперь Вы перемещаетесь еще только на 64 позиции, на этот раз перемещение по горизонтали будет осуществляться только на 64 ячейки (от Ø до 63).

Вся концепция наилучшим образом иллюстрируется версией первоначальной программы для спрайта 1:

```
1Ø V = 53248 : POKE V + 21,4 : POKE 2Ø42,13
2Ø FOR N = Ø TO 62 : READ Q : POKE 832 + N,Q : NEXT
25 POKE V+5, 1ØØ
3Ø FOR X = Ø TO 255
4Ø POKE V+4,X
5Ø NEXT
6Ø POKE V+16,4
7Ø FOR X = Ø TO 63
8Ø POKE V+4,X
9Ø NEXT
1ØØ POKE V+16,Ø
11Ø GOTO 3Ø
```

Рис. 4Ø

Строка 6Ø устанавливает наибольший значащий бит для спрайта 2. Строка 7Ø начинает стандартное движение в направлении оси X, перемещая спрайт 2 остальную часть пути по экрану.

Строка 1ØØ является весьма важной, так как она "выключает" наибольший значащий бит, так что спрайт теперь может вновь перемещаться от левого края экрана.

Двоичные арифметические операции

Вхождение в детали компьютерной обработки чисел выходит за рамки настоящего руководства. Однако мы дадим Вам хорошую основу для понимания этого процесса с тем, чтобы Вы могли осуществить сложное оживление.

Однако, сначала мы дадим определения нескольких терминов.

Бит - наименьшее количество информации, которое может храниться в памяти компьютера. Бит можно сравнить с выключателем, который имеет только два положения ("включено" и "выключено"). Когда бит соответствует положению "включено", он имеет величину 1, когда он соответствует положению "выключено", он имеет величину Ø.

Следующим понятием будет понятие байта.

Байт - определяется как последовательность бит. Так как байт состоит из 8 бит, в одном байте Вы можете иметь всего 255 различных наборов бит. Другими словами, Вы можете иметь все биты в положении "выключено", тогда Ваш байт будет выглядеть следующим образом:

128	64	32	16	8	4	2	1
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

и его величина будет равна нулю. Если все биты байта будут соответствовать положению "включено", то байт будет иметь вид

128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1

что равно 128+64+32+16+8+4+2+1 = 255

Следующее понятие - регистр.

Регистр - определяется как блок байтов, связанных воедино. Однако, в нашем случае каждый регистр имеет длину всего один байт. Последовательность регистров составляет регистровую карту (register map). Регистровые карты представляют собой таблицы, подобно той, на которую Вы смотрели при создании Вашего спрайта "воздушный шар". Каждый регистр управляет той или иной функцией, например, включение спрайта осуществляется регистром разрешения (enable register). Удлинение спрайта представляет собой расширение регистра X, в то время как увеличение спрайта в ширину представляет собой расширение регистра Y. Имейте в виду, что регистр представляет собой байт, который выполняет конкретную задачу.

Перейдем теперь к другим арифметическим двоичным операциям.

Перевод двоичных чисел в десятичные

Таблица 2

1 Decimal Value								
128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	1	2 ^0
0	0	0	0	0	0	1	0	2 ^1
0	0	0	0	0	1	0	0	2 ^2
0	0	0	0	1	0	0	0	2 ^3
0	0	0	1	0	0	0	0	2 ^4
0	0	1	0	0	0	0	0	2 ^5
0	1	0	0	0	0	0	0	2 ^6
1	0	0	0	0	0	0	0	2 ^7

1 - десятичная величина

Используя комбинации всех восьми бит, Вы можете получать любые десятичные величины от 0 до 255. Теперь Вы понимаете, почему, когда мы вставляли (POKE) знак или величину, характеризующую цвет в ячейки памяти, величины должны были находиться в диапазоне между 0 и 255? Каждая ячейка памяти может содержать один байт информации.

Любая возможная комбинация восьми нулей и единиц может быть преобразована в одну десятичную величину, лежащую в диапазоне от 0 до 255. Все восемь нулей в байте соответствуют нулевой величине, если же все биты байта равны единицам, то эта величина равна 255. Соответственно 00000011 равно 3 и так далее. Это является основой для создания данных, которые представляют спрайты, и дает возможность манипулировать с ними. Рассмотрим пример: если этот байт представляет собой часть спрайта (0 - это пробел, а 1 - цветная точка):

7	6	5	4	3	2	1	0
2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1

128+64 + 32+ 16+ 8 + 4 + 2 + 1 + = 255

тогда для представления этой части объекта мы ВСТАВИМ (POKE) 255 в соответствующую ячейку памяти.

Примечание. Для того чтобы избавить Вас от необходимости самому осуществлять преобразования двоичных чисел в десятичные (а этим придется заниматься очень много), мы предлагаем Вам программу, которая будет осуществлять для Вас такое преобразование. Рекомендуем ввести эту программу в компьютер и сохранить ее для использования в будущем.

```
1!  5 REM BINARY TO DECIMAL CONVERTER
2!  10 INPUT "ENTER 8-BIT BINARY NUMBER :"; A$
3!  12 IF'LEN (A$) <> S THEN PRINT "8 BITS PLEASE..":
 !     GOTO 10
 !  15 TL = 0 : C = 0
 !  20 FOR X = 8 TO 1 STEP  -1 : C = C + 1
 !  30 TL = TL + VAL(MID$(A$,C,1))*2^(X-1)
 !  40 NEXT X
4!  50 PRINT A$;" B^NARY ";" = ";TL;"DECIMAL"
 !  60 GOTO 10
```

Рис. 41
1 - преобразователь двоичной величины в десятичную; 2 - ...введите 8-разрядное двоичное число; 3 - 8 бит ...; 4 - двоичная ... десятичная

Эта программа берет Ваше двоичное число, которое было введено как цепочка, и рассматривает каждый знак цепочки слева направо (функция MID$). Переменная C указывает, над каким знаком необходимо работать, по мере того как программа проходит через цикл.

Функция VAL в строке 30 восстанавливает действительную величину знака. Так как мы имеем дело с числовыми знаками, то величина знака соответствует самому знаку. Например, если первым знаком A$ является 1, тогда величина также будет равна 1.

Последняя часть строки 30 умножает величину текущего знака на соответствующую степень 2. Так как первая величина находится на месте 2^7, например, TL будет равно 1 умножить на 2^7, или 128. Если бит равен нулю, то вся величина для этого места также будет равна нулю.

Этот процесс повторяется для всех восьми знаков, и в результате получается десятичная величина двоичного числа

Глава 7. Создание звуковых эффектов

Использование звука в случае если Вы не программист

Большинство программистов используют звук в компьютере для двух целей: создание музыки и создание звуковых эффектов. Перед тем как отправиться в лабиринты программирования звука, рассмотрим вкратце структуру типичной звуковой программы и покажем короткую звуковую программу, с которой Вы можете поэкспериментировать.

Структура звуковой программы

Начнем с того, что в компьютере "Коммодор-64" существует пять регулировок, которые Вам необходимо знать для генерирования звука на Вашем компьютере: ГРОМКОСТЬ (VOLUME), УПРАВЛЕНИЕ ФОРМОЙ СИГНАЛА (WAVEFORM CONTROL), НАРАСТАНИЕ/ЗАТУХАНИЕ (ATTACK/DECAY), ВЫДЕРЖИВАНИЕ/ОТПУСКАНИЕ (SUSTAIN/RELEASE) и ВЫСОКАЯ ЧАСТОТА/НИЗКАЯ ЧАСТОТА (HIGH FREQUENCY/LOW FREQUENCY). Первые четыре регулировки обычно устанавливаются один раз в начале Вашей программы. Регулировка по низкой и высокой частотам должна устанавливаться для КАЖДОЙ НОТЫ, которую Вы играете. Ниже приводится типичная структура звукомузыкальной программы.

Эталонная звуковая программа

Перед составлением программы Вы должны выбрать ГОЛОС. Существуют три голоса. Каждый голос требует различных чисел регулировки звука для формы сигнала и т.д. Вы можете осуществлять воспроизведение одного, двух, трех голосов вместе, но в нашей программе используется только ГОЛОС НОМЕР 1. Набирайте на клавиатуре эту программу строка за строкой, не забывайте нажать клавишу RETURN после каждой строки.

1. Установите VOLUME (ГРОМКОСТЬ) на самый высокий уровень:

```
10 POKE 54296,15
```

2. Установите уровень ATTACK/DECAY (НАРАСТАНИЕ/ЗАТУХАНИЕ) так, чтобы определить, как быстро нарастает звук ноты и затухает из своего максимального значения (0 до 255):

```
20 POKE 54277,190
```

3. Установите уровень SUSTAIN/RELEASE (ВЫДЕРЖИВАНИЕ/ОТПУСКАНИЕ) для выдерживания длительности ноты при определении громкости и ее отпускания

```
30 POKE 54278,248
```

4. Найдите ноту/тон, которую Вы хотите сыграть, в Таблице музыкальных нот в Приложении H и введите величины HIGH-FREQUENCY (ВЫСОКАЯ ЧАСТОТА) и LOW-FREQUENCY (НИЗКАЯ ЧАСТОТА) для этой ноты (каждая нота требует двух ВСТАВЛЕНИЙ (POKE).

```
50 POKE 54276,17
```

5. Установите регулировку WAVEFORM (ФОРМА СИГНАЛА) в одно из четырех стандартных положений (17,33,65 или 129):

```
60 FORT = 1TO250:NEXT
```

6. Введите временной цикл для установки ДЛИТЕЛЬНОСТИ (DURATION) ноты, которая воспроизводится (обычно четверть ноты равна приблизительно "250", однако, она может меняться, так как более длинная программа может влиять на временные соотношения).

7. Выключите регулировки WAVEFORM CONTROL (УПРАВЛЕНИЕ ФОРМОЙ СИГНАЛА), ATTACK/DECAY (НАРАСТАНИЕ/ЗАТУХАНИЕ) и SUSTAIN/RELEASE (ВЫДЕРЖИВАНИЕ/ОТПУСКАНИЕ)

```
70 POKE 54276,0:POKE 54277,0: POKE 54278,0
```

Чтобы прослушать ноту, которую Вы только что создали, наберите слово RUN (ПРОГОН) и затем нажмите на клавишу RETURN. Чтобы просмотреть или изменить программу наберите слово LIST (СПИСОК) и нажмите клавишу RETURN.

Создание музыки на компьютере "Коммодор-64"

Для того чтобы создавать музыку на компьютере "Коммодор-64" вовсе не обязательно быть музыкантом! Все, что Вам требуется знать - это несколько простых чисел, которые дают указания Вашему компьютеру, какой должна быть громкость звука, какие ноты воспроизводить, как долго играть их и т.п. Но сначала... рассмотрите программу, которая наглядно демонстрирует невероятные музыкальные способности компьютера "Коммодор-64" при использовании только ОДНОГО из трех голосов компьютера.

Наберите слово NEW (НОВАЯ) и нажмите клавишу RETURN для стирания Вашей предыдущей программы, затем введите эту программу, наберите на клавиатуре слово RUN (ПРОГОН) и нажмите клавишу RETURN.

```
1    5 REM MUSICAL SCALE <--------                                  2
     7 FORL=54272T054296:POKEL,0:NEXT clears SID chip
     10 POKE 54296,15 <-----------    sets volume                   3
     20 POKE 54277,7POKE54278,133     sets a/d/s/r                  4
     50 READ A <------------------    READs 1st number from
                                      line 110
     55 IF A= -1 THENEND <--------    ENDs loop                     5
     60 READ B  <-----------------    READs 2nd number              6
     80 POKE54273,A:POKE54272,B<--    POKEs 1st number              7
                                      from line 110as
                                      HI-FREQ and 2nd number
                                      as LOW-FREQ
     85 POKE54276,17 <------------    starts note                   8
     90 FORT=1TO250:NEXT:POKE54276,16 lets ote play, then           9
                                      stops it
     95 FORT=1TO50:NEXT <---------    sets time for RELEASE,
                                      time between notes
     100 GOTO20 <-----------------    restarts program
```

```
110 DATA 16,195,18,20,21,31,22,96      lists note valie

120 DATA 25,30,28,49,31,165,33,135     from chart in App.13
                                       M.Each part of
                                       numbers = one note
                                       (16 and 19 = 4th
                                       octave C)

999 DATA -1      <------------         ENDs program      14
                                       (see line 55)
```

Рис. 42

1 - комментарий "музыкальная шкала"; 2 - наименование программы; 3 - устанавливает громкость на максимальном уровне (15); 4 - устанавливает уровень ATTACK/DECAY (для каждой ноты); 5 - определяет форму сигнала (тип звучания); 6 - длительность звучания каждой ноты; 7 - считывает число в строке 110 DATA (ДАННЫЕ); 8 - считывает второе число в строке 110 DATA (ДАННЫЕ); 9 - программа выключает две регулировки VOICE (ГОЛОС) и заканчивается (END), когда происходит СЧИТЫВАНИЕ (READ) -1 в строке 900; 10 - осуществляется ВСТАВЛЕНИЕ (POKE) первого числа из данных (DATA) в строке 110 (A=17) в качестве регулировки HIGH FREQUENCY (ВЫСОКАЯ ЧАСТОТА) и второго числа (B=37) в качестве регулировки LOW FREQUENCY (НИЗКАЯ ЧАСТОТА); В следующий раз при осуществлении нового цикла произойдет считывание A=19 и B=63 и т.д., и эти числа будут вставляться (POKE) в ячейки, характеризующие HIGH FREQUENCY (ВЫСОКАЯ ЧАСТОТА) и LOW FREQUENCY (НИЗКАЯ ЧАСТОТА). Число 54273=HIGH FRECUENCY (ВЫСОКАЯ ЧАСТОТА) для VOICE 1 (ГОЛОС 1), а 54272=LOW FRECUENCY (НИЗКАЯ ЧАСТОТА) для VOICE 1 (ГОЛОС 1); 11 - выключает регулировку WAVEFORM CONTROL (УПРАВЛЕНИЕ ФОРМОЙ СИГНАЛА); 12 - зацикливается обратно для сброса CONTROL (УПРАВЛЕНИЕ) и воспроизведения следующей ноты; 13 - величины музыкальных нот из таблицы числовых величин для нот, приведенной в Приложении И. Каждая пара чисел соответствует одной ноте. 17 и 37 представляют ноту "до" четвертой октавы, 19 и 63 - ноту "ре" и т.д.; 14 - когда программа достигает -1, она выключает регулировки HIGH/LOW FREQUENCY (ВЫСОКАЯ/НИЗКАЯ ЧАСТОТА) и заканчивается (END), как указано в строке 70.

Чтобы сделать так, чтобы звучание было, как на клавикордах, измените строку 30 следующим образом:

POKE54276,33

и осуществите ПРОГОН (RUN) программы заново. (Чтобы сменить строку, нажмите клавишу RUN/STOP для останова программы, наберите на клавиатуре слово LIST (СПИСОК) и нажмите клавишу RETURN, затем заново перепечатайте программную строку, которую Вы хотите заменить, при этом новая строка автоматически заменит прежнюю). Здесь в действительности мы заменили форму сигнала с "треугольной" на "пилообразную". Изменение ФОРМЫ СИГНАЛА (WAVEFORM) может коренным образом изменить звучание на компьютере "Коммодор-64", однако ФОРМА СИГНАЛА является лишь одной из нескольких регулировок, которые Вы можете менять для создания различных музыкальных тонов и звуковых эффектов! Вы можете также менять скорость НАРАСТАНИЯ/ЗАТУХАНИЯ (ATTACK/DECAY) для каждой ноты... например, для изменения "клавикордного" звучания на звучание банджо попробуйте заменить строку 20 следующей строкой:

20 POKE54277,3

Как Вы только что видели, Вы можете заставить свой компьютер "Коммодор-64" звучать, как звучат различные музыкальные инструменты. Рассмотрим подробнее, как работает каждая звуковая регулировка.

Важные регулировки звука

1. ГРОМКОСТЬ (VOLUME) - Чтобы включить громкость и установить ее на максимальный уровень, наберите на клавиатуре: POKE 54296,15. Регулировка громкости лежит в диапазоне от 0 до 15, но в большинстве случаев Вы будете использовать значение 15. Для выключения регулировки ГРОМКОСТЬ наберите на клавиатуре: POKE 54296,0.

Вам необходимо установить громкость только ОДИН РАЗ в начале Вашей программы, так как одна и та же установка задействует все три голоса, которыми обладает компьютер "Коммодор-64". (Изменение громкости при воспроизведении музыкальной ноты или звукового эффекта дает интересные результаты, однако это лежит вне рамок настоящего вступления).

2. Регулировки ADSR (ATTACK/DECAY/SUSTAIN/RELEASE - НАРАСТАНИЕ/ЗАТУХАНИЕ/ВЫДЕРЖИВАНИЕ/ОТПУСКАНИЕ) и WAVEFORM CONTROL (УПРАВЛЕНИЕ ФОРМОЙ ИМПУЛЬСА) - Вы уже видели, как изменение формы сигнала может изменять звучание - от "ксилофонного" до "клавикордного". Каждый ГОЛОС (VOICE) имеет свою собственную регулировку формы сигнала (WAVEFORM CONTROL SETTING), которая позволяет Вам установить четыре различных типа формы сигнала: треугольную, пилообразную, импульсную (прямоугольную) и шум. УПРАВЛЕНИЕ (CONTROL) также задействует функции ADSR компьютера "Коммодор-64", и мы к этому скоро вернемся.

Стандартная установка формы сигнала имеет следующий вид:

POKE 54276,17

где первое число (54276) представляет регулировку (контрольную установку) для ГОЛОСА 1 (VOICE 1), а второе число (17) соответствует треугольной форме сигнала. Регулировки для каждого ГОЛОСА и ФОРМЫ СИГНАЛА показаны в Таблице 3.

Таблица 3

Регулировки ADSR и формы сигнала

	1	CONTROL SETTING	2 TRIANGLE	3 SAWTOOTH	4 PULSE	5 NOISE
6	VOICE 1	54276	17	33	65	129
7	VOICE 2	54283	17	33	65	129
8	VOICE 3	54290	17	33	65	129

1 - регулировки (контрольные установки); 2 - треугольная;
3 - пилообразная; 4 - импульсная (прямоугольная); 5 - шум;
6 - голос 1; 7 - голос 2; 8 - голос 3

Хотя контрольные установки для каждого голоса различны, установки для формы сигнала одинаковы для каждого типа формы сигнала. Чтобы разобраться в этом вопросе, посмотрите на строку 20 в программе музыкальной шкалы. В этой программе непосредственно за установкой ГРОМ-КОСТИ (VOLUME) в строке 10 мы устанавливаем контрольную установку (регулировку) (CONTROL SETTING) для ГОЛОСА 1 (VOICE 1) в строке 20 с помощью ВСТАВЛЕНИЯ (POKE) 54276,17. Это включает УПРАВЛЕНИЕ (CONTROL) для ГОЛОСА 1 (VOICE 1) и устанавливает ТРЕУГОЛЬНУЮ ФОРМУ СИГНАЛА (TRIANGLE WAVEFORM) (17). Впоследствии мы изменили регулировку формы сигнала с 17 на 33 для получения ПИЛООБРАЗНОЙ ФОРМЫ СИГНАЛА (SAWTOOTH WAVEFORM), и это придало звучанию "клавикордную" окраску. Как же взаимодействуют CONTROL SETTING (КОНТРОЛЬНАЯ УСТАНОВКА) И WAVEFORM (ФОРМА СИГНАЛА)? Установка (регулировка) формы сигнала аналогична установке (регулировке) громкости, за исключением того, что каждый голос имеет свою собственную регулировку и вместо ВСТАВЛЕНИЯ (POKE) уровней громкости мы определяем форму сигнала. Теперь рассмотрим другой аспект звука - характеристику ADSR.

3. Регулировка НАРАСТАНИЯ/ЗАТУХАНИЯ (ATTACK/DECAY) - Как мы отметили ранее, КОНТРОЛЬНАЯ УСТАНОВКА (CONTROL SETTING) не только определяет форму сигнала, но и задействует ADSR, или НАРАСТАНИЕ/ЗАТУХАНИЕ/ВЫДЕРЖИВАНИЕ/ОТПУСКАНИЕ (ATTACK/DECAY/SUSTAIN/RELEASE), в компьютере "Коммодор-64". Начнем рассмотрение с регулировки ATTACK/DECAY (НАРАСТАНИЕ/ЗАТУХАНИЕ). При-

водимая ниже таблица показывает различные уровни НАРАСТАНИЯ (ATTACK) и ЗАТУХАНИЕ (DECAY) для каждого голоса. Если Вы не знакомы с понятием нарастания и затухания звука, Вы можете представлять себе нарастание как скорость, с которой нота/звук достигают своей МАКСИМАЛЬНОЙ ГРОМКОСТИ (MAXIMUM VOLUME). ЗАТУХАНИЕ (DECAY) представляет собой скорость, с которой громкость ноты/звука падает от максимальной до нулевой. На приводимой ниже таблице показана установка НАРАСТАНИЯ/ЗАТУХАНИЯ (ATTACK/DECAY) для каждого голоса и числа для каждой установки (регулировки) нарастания и затухания. Обратите внимание, что ВЫ МОЖЕТЕ КОМБИНИРОВАТЬ УСТАНОВКИ ATTACK И DECAY ПУТЕМ СЛОЖЕНИЯ ИХ ВМЕСТЕ И ВВОДА ПОЛУЧЕННОЙ СУММЫ. Например, Вы можете установить ВЫСОКУЮ скорость НАРАСТАНИЯ (HIGH ATTACK) и НИЗКУЮ СКОРОСТЬ ЗАТУХАНИЯ (LOW DECAY) с помощью сложения числа высокого нарастания (64) и числа низкого затухания (1). Сумма (65) даст указания компьютеру установить высокую скорость нарастания и низкую скорость затухания. Вы можете также увеличить скорости нарастания сложением соответствующих чисел вместе (128 + 64 + 32 + 16 = МАКС. СКОРОСТЬ НАРАСТАНИЯ = 240).

Таблица 4

Установки скорости НАРАСТАНИЯ/ЗАТУХАНИЯ (ATTACK/DECAY)

	1 ATTACK	2 DECAY SETTING	3 HIGH ATTACK	4 MEDIUM ATTACK	5 LOW ATTACK	6 LOWEST ATTACK	7 HIGH DECAY	8 MED. DECAY	9 LOW DECAY	10 LOWEST DECAY
10	VOICE 1	54277	128	64	32	16	8	4	2	1
11	VOICE 2	54284	128	64	32	16	8	4	2	1
12	VOICE 3	54291	128	64	32	16	8	4	2	1

1 - установка НАРАСТАНИЕ/ЗАТУХАНИЕ; 2 - высокое нарастание; 3 - среднее нарастание; 4 - низкое нарастание; 5 - самое низкое нарастание; 6 - высокое затухание; 7 - среднее затухание; 8 - низкое затухание; 9 - самое низкое затухание; 10 - голос 1; 11 - голос 2; 12 - голос 3

Если Вы установите скорость нарастания, но не установите скорости затухания, то скорость затухания автоматически устанавливается нулевой и наоборот. Например, если Вы ВСТАВИТЕ (POKE) 54277,64, Вы установите среднюю скорость нарастания с нулевой скоростью затухания для ГОЛОСА 1. Если Вы ВСТАВИТЕ (POKE) 54277,66, Вы установите среднюю скорость нарастания и низкую скорость затухания (так как 66 = 64 + 2 и устанавливает ОБЕ установки). Вы можете также добавить несколько величин нарастания или же несколько величин затухания. Например, Вы можете добавить низкое нарастание (32) и среднее нарастание (64) для комбинированной скорости нарастания (96), затем добавить среднее затухание (4) и ... presto... ВСТАВИТ (POKE) 54277,100.

В этой точке лучше всего проиллюстрирует сказанное эталонная программа. Наберите на клавиатуре слово NEW (НОВАЯ), нажмите клавишу RETURN, наберите приведенную ниже программу и осуществите ее ПРОГОН (RUN).

```
9  10 PRINT "HIT ANY KEY"<------ Screen messoge                1
   20 GETK$:IFK$=""THEN60<------ Check ihe kevboard            2
   25 POKE54276,0:POKE54277,0:POKE54272,0<---Turn off          3

   30 POKE54296,15<------------- Set maximum volume             4
   40 POKE54277,64<------------- Set attack/decav               5
   50 POKE54276,17:FORT=1TO200:NEXT Start triangle waveform    6
   60 POKE54273,162:POKE54272,37 POKE one note in voice 1      7

   90 GOTO20<------------------- Repeat execution               8
```

Рис. 43

1 - сообщение на экране: 2 - проверка клавиатуры: 3 - выключение установок: 4 - установка громкости на максимальный уровень: 5 - установка ATTACK/DESAY (НАРАСТАНИЕ/ЗАТУХАНИЕ): 6 - установка управления формой сигнала (треугольная форма сигнала): 7 - вставление одной ноты в ГОЛОС 1 (VOICE 1): 8 - зацикливание программы и выполнения ее заново: 9 - ... нажмите любую клавишу

Здесь мы используем ГОЛОС 1 (VOICE 1) для создания одной ноты за один раз... с помощью СРЕДНЕЙ СКОРОСТИ НАРАСТАНИЯ (MEDIUM ATTACK RATE) и нулевого затухания (ZERO DECAY). Ключевой строкой является строка 40. ВСТАВЛЕНИЕ (POKE) установки ATTACK/DECAY с числом 64 задействует СРЕДНЮЮ скорость нарастания. В результате звук напоминает звук шарика, прыгающего в цистерне для нефти. Теперь займемся занимательной частью программы. Нажмите клавишу RUN/STOP для останова программы, затем наберите на клавиатуре слово LIST и нажмите клавишу RETURN. Теперь наберите на клавиатуре эту строку и нажмите клавишу RETURN (новая строка 40 автоматически заменит прежнюю строку 40):

```
40 POKE 54277,190
```

Наберите слово RUN (ПРОГОН) и нажмите клавишу RETURN, чтобы посмотреть, как звучит эта программа. Здесь мы скомбинировали несколько установок нарастания и затухания. Установки здесь таковы: ВЫСОКОЕ НАРАСТАНИЕ (HIGH ATTACK) (128) + НИЗКОЕ НАРАСТАНИЕ (LOW ATTACK) (32) + САМОЕ НИЗКОЕ НАРАСТАНИЕ (16) + ВЫСОКОЕ ЗАТУХАНИЕ (HIGH DECAY) (8) + СРЕДНЕЕ ЗАТУХАНИЕ (MEDIUM DECAY) (4) + НИЗКОЕ ЗАТУХАНИЕ (LOW DECAY) (2) = 190. Звучание носит характер игры на гобое или на подобном ему "тростниковом" инструменте. Если Вы хотите поэкспериментировать, попробуйте изменять форму сигнала и числа нарастания/затухания в примере с музыкальной шкалой, чтобы увидеть, как звучит "гобой". Таким образом Вы сможете увидеть, что изменение скоростей нарастания/затухания может быть использовано для создания различных типов звуковых эффектов.

4. Регулировка ВЫДЕРЖИВАНИЯ/ОТПУСКАНИЯ (SUSTAIN/RELEASE) - Как и регулировка нарастания, эта регулировка осуществляется с помощью управления функцией ADSR/WAVEFORM (ФОРМА СИГНАЛА). ВЫДЕРЖИВАНИЕ/ОТПУСКАНИЕ позволяет Вам "расширить" (SUSTAIN) часть того или иного звука, подобно "педали выдерживания" на пианино или органе, которая позволяет продлить ноту. Любая нота и любой звук могут выдерживаться на максимуме громкости... Вы можете даже установить уровень выдерживания на максимум (240) без отпускания, и нота будет звучать бесконечно долго. Регулировка ВЫДЕРЖИВАНИЯ/ОТПУСКАНИЯ (SUSTAIN/RELEASE) может использоваться с циклом FOR...NEXT (В ТЕЧЕНИЕ...ЗАТЕМ) для указания того, как долго нота будет выдерживаться при максимальной громкости перед отпусканием. В следующей таблице показаны числа, которые Вам должны ВСТАВИТЬ (POKE) для получения различных скоростей ВЫДЕРЖИВАНИЯ/ЗАТУХАНИЯ (SUSTAIN/RELEASE).

Таблица 5

Установка SUSTAIN/RELEASE (ВЫДЕРЖИВАНИЕ/ОТПУСКАНИЕ)

	1	2	3	4	5	6
	SUSTAIN CONTROL	RELEASE SETING	HIGH SUSTAIN	MEDIUM SUSTAIN	LOW SUSTAIN	LOWEST SUSTAIN	HIGH RELEASE
10	VOICE 1	54278	128	64	32	16	8
11	VOICE 2	54285	128	64	32	16	8
12	VOICE 3	54292	128	64	32	16	8

7	8	9
MED. RELEASE	LOW RELEASE	LOWEST RELEASE
4	2	1
4	2	1
4	2	1

1 - установка управления (регулировка) SUSTAIN/RELEASE (ВЫДЕРЖИВАНИЕ/ОТПУСКАНИЕ); 2 - высокое (долгое) выдерживание; 3 - среднее выдерживание; 4 - низкое (малое) выдерживание; 5 - самое малое выдерживание; 6 - высокое (долгое) отпускание; 7 - среднее отпускание; 8 - низкое (малое) отпускание; 9 - самое малое отпускание; 10 - голос 1; 11 - голос 2; 12 - голос 3

Если Вы используете VOICE 1 (ГОЛОС 1), то в качестве примера Вы можете установить ВЫСОКИЙ УРОВЕНЬ ВЫДЕРЖИВАНИЯ (HIGH SUSTAIN LEVEL) с помощью набора на клавиатуре: POKE 54278,128, или же Вы можете скомбинировать HIGH SUSTAIN LEVEL (ВЫСОКИЙ УРОВЕНЬ ВЫДЕРЖИВАНИЯ) С LOW RELEASE LEVEL (НИЗКИЙ УРОВЕНЬ ОТПУСКАНИЯ), складывая 128+2 и затем реализуя POKE 54278,130. Ниже приводится та же эталонная программа, которая была приведена для установок ATTACK/DECAY, но с добавлением регулировок SUSTAIN/RELEASE. Обратите внимание на разницу в звучании

```
1   10 PRINT"HIT ANY KEY"                                    2
    20 GETKS:IFKS=""THEN20                                   3
    25 POKE54276,0:POKE54277,0:POKE54278,0                   4
    30 POKE54296,15                                          5
    40 POKE54277,64                                          6
    45 POKE54278,128                                         7
    50 POKE54276,17                                          8
    60 POKE54273,17:POKE54272,37                             9
    90 GOTO20                                                10
```

Рис. 44

1 - ..."нажмите на любую клавишу"; 2 - сообщение на экране; 3 - проверка клавиатуры; 4 - выключение регулировок; 5 - установление громкости на максимальный уровень; 6 - установка ATTACK/DECAY (НАРАСТАНИЕ/ЗАТУХАНИЕ); 7 - установка SUSTAIN/RELEASE (ВЫДЕРЖИВАНИЕ/ОТПУСКАНИЕ); 8 - установка формы сигнала (треугольная); 9 - ВСТАВЬТЕ (POKE) одну ноту в VOICE 1 (ГОЛОС 1); 10 - зацикливание и выполнение программы заново

В строке 45 мы даем указание компьютеру ВЫДЕРЖАТЬ (SUSTAIN) ноту при ВЫСОКОМ ВРЕМЕНИ ВЫДЕРЖИВАНИЯ (HIGH SUSTAIN RATE) (128, как указано в Таблице 5)... после чего тон завершает свою обычную реализацию. Вы можете изменить часть ноты, которая ВЫДЕРЖИВАЕТСЯ с помощью изменения "счетных" чисел в строке 45, точно так же как Вы можете изменять длительность ноты изменением регулировки затухания в строке 40.

5. Выбор голосов и установка величин HIGH/LOW FREQUENCY (ВЫСОКАЯ/НИЗКАЯ ЧАСТОТА ЗВУКА) - Каждая отдельная нота при реализации на компьютере "Коммодор-64" требует ДВУХ РАЗДЕЛЬНЫХ КОМАНД POKE - одной для ВЫСОКОЙ ЧАСТОТЫ (HIGH FREQUENCY) и одной НИЗКОЙ ЧАСТОТЫ (LOW FREQUENCY). В таблице "Величины музыкальных нот" (MUSICAL NOTE VALUE) в Приложении H показаны со-

ответствующие ВСТАВЛЕНИЯ (POKE), которые Вам необходимы для воспроизведения любой ноты в восьмиоктавном диапазоне компьютера "Коммодор-64". Команды HIGH и LOW FPEQUENCY POKE различны для каждого голоса, который Вы используете - это позволяет Вам программировать все три голоса независимо для создания трехголосного музыкального произведения или экзотических звуковых эффектов.

Эти команды для каждого голоса показаны в Таблице 6, которая также содержит ВЕЛИЧИНЫ НОТ (NOTE VALUES) для средней (пятой) октавы.

Таблица 6

1 VOICE NUMBER FREQUENCY	2 POKE NUMBER	3 SAMPLE MUSICAL NOTES-FIFTH OCTAVE C	C#	D	D#	E	F	F#	G	G#	A	A#
4 VOICE1/HIGH	54273	34	36	38	40	43	45	48	51	54	57	61
5 VOICE1/LOW	54272	75	85	126	200	52	198	127	97	111	172	126
6 VOICE2/HIGH	54280	34	36	38	40	43	45	48	51	54	57	61
7 VOICE2/LOW	54279	75	85	126	200	52	196	127	97	111	172	126
8 VOICE3/HIGH	54287	34	36	38	40	43	45	48	51	54	57	61
9 VOICE3/LOW	54286	75	85	126	200	52	198	127	97	111	172	126

	B	C	C#
4	64	68	72
5	108	149	169
6	64	68	72
7	188	149	169
8	64	68	72
9	188	149	169

Рис. 45

1 - номер и частота голоса: 2 - число POKE: 3 - эталонные музыкальные ноты - пятая октава: 4- голос 1/высокая; 5 - голос 1/низкая: 6 - голос 2/высокая: 7 - голос 2/низкая; 8 - голос 2/высокая; 9 - голос 3/низкая

Как Вы можете видеть, для каждого голоса существуют две установки - установка HIGH FREQUENCY (ВЫСОКАЯ ЧАСТОТА) и установка LOW FREQUENCY (НИЗКАЯ ЧАСТОТА). Чтобы воспроизвести музыкальную ноту, Вы должны ВСТАВИТЬ (POKE) величину HIGH FREQU-

ENCY (ВЫСОКАЯ ЧАСТОТА) и ВСТАВИТЬ (POKE) другую величину в ячейку LOW FREQUENCY (НИЗКАЯ ЧАСТОТА). Выбирая соответствующие значения из Таблицы 6, мы представим Вам установку, которая воспроизводит ноту "до" пятой октавы (ГОЛОС 1):

```
POKE 54273,34:POKE 54272,75
```

Та же нота в ГОЛОСЕ 2 выглядит следующим образом:

```
POKE 54280,34:POKE 54279,75
```

В программе это будет выглядеть так:

```
10 V=54296:W=54276:A=54277:                                    1
   H=54273:L=54272
20 POKEV,15:POKEW,33:POKEA,190                                 2
30 FORT=1TO200:POKEH,34:POKEL,75:NEXT                          3

40 POKEH,0:POKEL,0:POKEW,0                                     4
```

Рис.76

1 - установить числа, соответствующие буквам; 2 - ВСТАВИТЬ (POKE) громкость, форму сигнала, нарастание/затухание; 3 - ДЛИТЕЛЬНОСТЬ, ВСТАВИТЬ (POKE) ВЫСОКУЮ/НИЗКУЮ ЧАСТОТЫ ДЛЯ НОТ; 4 - выключить ВЫСОКУЮ/НИЗКУЮ ЧАСТОТЫ И ФОРМУ СИГНАЛА

Воспроизведение песен на компьютере "Коммодор-64"

Приводимая ниже программа может быть использована для создания или воспроизведения песни (с использованием ГОЛОСА 1). В этой программе имеется два поучительных момента: во-первых, обратите внимание, на то как мы сокращаем все длинные уравляющие числа в первой строке программы, после этого мы можем использовать букву W для слова "waveform" (форма сигнала) вместо числа 54276.

Второй момент относится к способу использования ДАННЫХ (DATA). Эта программа дает Вам возможность ввести три числа для каждой ноты: ВЕЛИЧИНУ ВЫСОКОЙ ЧАСТОТЫ НОТЫ (HIGH FREQUENCY NOTE VALUE), ВЕЛИЧИНУ НИЗКОЙ ЧАСТОТЫ НОТЫ (LOW FREQUENCY NOTE VALUE) и ДЛИТЕЛЬНОСТЬ НОТЫ, КОТОРАЯ БУДЕТ ВОСПРОИЗВЕДЕНА (OURATION THE NOTE WILL BE PLAYED).

Для этой песни мы использовали "счет" длительности 125 для восьмой доли ноты, 250 - для четверти ноты, 375 - для трех восьмых ноты, 500 - для половины ноты, 1000 - для целой ноты. Эти числовые величины могут быть увеличены или уменьшены для согласования с конкретным темпом или с Вашим собственным музыкальным вкусом.

Чтобы понять, как осуществляется ввод песни, посмотрите на строку 100. В качестве установок HIGH и LOW FREQUENCY (ВЫСОКАЯ и НИЗКАЯ ЧАСТОТА) мы ввели величины 34 и 75 чтобы воспроизвести ноту "до" (из эталонной шкалы, которая была приведена ра-

нее) и затем число 250 для четверти ноты. Итак, первая нота нашей песни - это четверть ноты "до". Вторая нота - это также четверть ноты, но на этот раз ноты "ми" и так далее до конца мелодии. Таким способом Вы можете вводить почти любую песню, добавляя столько строк с оператором DATA (ДАННЫЕ), сколько необходимо. Вы можете продолжать числа нот и длительностей от одной строки к другой, но каждая строка должна начинаться со слова DATA (ДАННЫЕ). Последней строкой Вашей программы должно быть выражение DATA-1,-1,-1. Эта строка "заканчивает" песню.

Наберите на клавиатуре слово NEW (НОВАЯ), чтобы стереть предыдущую программу и наберите следующую программу, затем, для того, чтобы прослушать песню, наберите на клавиатуре слово RUN (ПРОГОН)

```
1    MICHAEL ROW THE BOAT ASHORE-1 MEASURE

5 V=54296:W=54276:A=54277:HF=54273:LF=54272:S=54278:PH
   =54275:PL=54274
10 POKEV,15:POKEW,65:POKEA,190:POKEPH,15:POKEPL,15
20 READM
30 READL
40 READD
50 IFM=-1THENEND
60 POKEHF,H:POKELF,L
70 FORX=D-50TOD-20:POKES,136:NEXT
80 FORT=1TOD:NEXT:POKEHF,0:POKELF,0:POKEW,0
90 GOTO10
100 DATA34,75,250,43,52,250,51,97,375,43,52,125,51,97
105 DATA250,57,172,250
110 DATA51,97,500,0,0,125,43,52,250,51,97,250,57,172
115 DATA1000,51,97,500
120 DATA-1,-1,-1
```

Рис.47

1 - "Майкл, греби к берегу" (название песни) - один такт

Создание звуковых эффектов

В отличие от музыки, звуковые эффекты чаще всего связаны с каким-нибудь действием при программировании, таким, как взрывоподобный звук, с которым космический истребитель преодолевает звуковой барьер в какой-либо космической игре, или же предупреждающий зуммер в деловой программе, который оповещает пользователя, что он по ошибке собирается затереть содержимое своего диска.

Если Вы хотите создавать звуковые эффекты, то компьютер "Коммодор-64" предлагает Вам широкий выбор. Ниже приведены 10 основных принципов программирования, которые помогут Вам при экспериментировании со звуковыми эффектами:

1. Изменяйте громкость при воспроизведении ноты, например, при создании "эхо-эффектов".

2. Быстро чередуйте две ноты для создания "тремоло".

3. Попробуйте различные установки формы сигнала для каждого голоса.

4. ATTACK/DECAY (НАРАСТАНИЕ/ЗАТУХАНИЕ) - меняйте скорость, с которой звук нарастает до максимума, и скорость, с которой он спадает от этого максимума.

5. SUSTAIN/RELEASE (ВЫДЕРЖИВАНИЕ/ОТПУСКАНИЕ) - попробуйте различные установки для увеличения громкости или внезапного прекращения звука или для комбинирования последовательности звуков для создания звукового эффекта.

6. Многоголосые звуковые эффекты - одновременное воспроизведение двух или трех голосов, с независимым управлением каждого голоса, или же с одним голосом, играющим дольше или меньше, чем другой, или же использование второго голоса как эха или другой реакции на первую ноту.

7. Изменение нот на музыкальной шкале или изменение октав с помощью величин в Таблице величин музыкальных нот.

8. Используйте импульсную (прямоугольную) форму сигнала и различные регулировки импульсной формы для создания звуковых эффектов.

9. Используйте шумовой сигнал для генерирования "белого шума" для акцентирования тональных звуковых эффектов или создания звуковой имитации взрывов, выстрелов или шагов. Для создания различных типов белого шума можно использовать те же музыкальные ноты с шумовой формой сигнала.

10. Скомбинируйте несколько частот HIGH/LOW (ВЫСОКАЯ/НИЗКАЯ) в быстрой последовательности на различных октавах.

11. Фильтр - попробуйте применить дополнительную установку POKE, как указано в Приложении H.

Примеры звуковых эффектов

Приведенная ниже программа может быть добавлена почти к любой программе на языке БЕЙСИК. Она приведена здесь, для того чтобы дать Вам представление о некоторых идеях программирования и продемонстрировать диапазон звуковых эффектов компьютера "Коммодор-64".

Обратите внимание на сокращения при программировании, которые мы применили в строке 10. Мы можем сокращать эти длинные и громоздкие числа для установки звука, заменяя их удобными в использовании буквами (числовыми переменными). Строка 10 просто означает, что эти удобные для запоминания БУКВЫ могут быть использованы вместо длинных чисел. Здесь V обозначает громкость (Volume), W - форму сигнала (Waveform), A - нарастание/затухание (Attack/decay), H - высокую частоту (High Frequency) (ГОЛОС 1) и L - низкую частоту (Low frequency) (ГОЛОС 1). После этого мы можем использовать эти буквы в нашей программе вместо чисел, что делает программу короче, сокращает время при

наборе программы с клавиатуры, а также облегчает запоминание и нахождение величин звуковых установок.

```
10 V=54296:W=54276:A=54277:H=54272
20 FORX=15TO0STEP-1:POKEV,X:POKEW,129:POKEA,
       15:POKEH,40:POKEL,200:NEXT
30 POKEW,0:POKEA,0
```

Рис. 48

Программа имитации звука выстрела с помощью ГОЛОСА 1 шумовой формы сигнала и затухающей громкости

Глава 8. Развитая обработка данных

Операторы READ (СЧИТЫВАНИЕ) и DATA (ДАННЫЕ)

Мы уже знаем, как осуществлять непосредственное присвоение величин переменных в программе (A = 2) и как присваивать различные величины во время прогона программы - через оператор INPUT.

Однако существует много случаев, когда ни один из этих способов не может быть применен при работе, которую Вы выполняете, особенно если она связана с большим количеством информации.

Рассмотрим следующую короткую программу:

```
10 READ X
20 PRINT "X IS NOW:";X
30 GOTO 10
40 DATA 1.34,10.5,16.234.56

RUN

X IS NOW:1
X IS NOW:34
X IS NOW:10.5
X IS NOW:16
X IS NOW:234.56

?OUT OF DATA ERROR IN 10
READY
```

Рис. 49

В строке 10 компьютер считывает (READ) одну величину из

оператора DATA (ДАННЫЕ) и присваивает эту величину переменной X. Каждый раз в течение цикла считывается следующая величина в операторе DATA (ДАННЫЕ), и эта величина присваивается переменной X и выводится на печать (PRINT). Указатель в самом компьютере следит за тем, какая величина должна использоваться в текущий момент:

```
                --- указатель
40 DATA 1, 34, 10.5, 16, 234.56
```

Когда использованы все величины и компьютер вновь осуществляет цикл, реализуя поиск другой величины, на дисплей выводится сообщение об ошибке OUT OF DATA (ДАННЫХ НЕТ), так как больше нет величин для считывания (READ).

Очень важно точно следовать формату ДАННЫХ:

```
40 DATA 1, 34, 10.5, 16, 234.56

   каждая величина                в конце запятой нет
   разделяется запятой
```

Операторы данных могут содержать целые числа, действительные числа (234.56) или числа в научной нотации. Однако Вы не можете выполнять считывание (READ) других переменных или осуществлять арифметические операции в строках DATA (ДАННЫЕ):

```
40 DATA A, 23/56, 2*5
```

Однако, Вы можете использовать цепочечную переменную в операторе READ и затем поместить цепочечную информацию в строку DATA (ДАННЫЕ). Так, следующая программа является приемлемой:

```
NEW

10 FOR X = 1 46 3
15 READ A$
20 PRINT "A$ IS NOW : ";A$
30 NEXT
40 DATA THIS; IS; FUN

RUN

A$ IS NOW : THIS
A$ IS NOW : IS
A$ IS NOW : FUN
READY
```

Рис.50

Заметим, что на этот раз оператор READ (СЧИТЫВАНИЕ) был помещен внутрь цикла FOR...NEXT (В ТЕЧЕНИЕ ... ЗАТЕМ). Затем был реализован этот цикл для согласования номера величин в данных оператора.

Во многих случаях Вы будете менять числа величин в операторе DATA (ДАННЫЕ) каждый раз, когда осуществляется прогон программы. Одним из способов избежать подсчета чисел величин и не получить сообщения OUT OF DATA ERROR (ОШИБКА В ДАННЫХ) является помещение файла ("FLAG") в качестве последней величины в строке DATA (ДАННЫЕ). Этот флаг должен быть величиной, которой никогда не могут быть Ваши данные, например, отрицательным числом, или очень маленьким, или очень большим числом. Когда осуществляется считывание этой величины (READ), программа переключается на следующую часть.

Существует способ повторного использования тех же самых ДАННЫХ (DATA) позже в программе. Это осуществляется ВОССТАНОВЛЕНИЕМ (RESTORE) указателя данных в начале списка данных. Добавьте строку 50 к предыдущей программе:

```
50 GOTO 10
```

Тем не менее вы получите сообщение ошибки OUT OF DATA (ОШИБКА В ДАННЫХ), так как когда программа переключается обратно к строке 10 для повторного считывания данных, указатель данных указывает все данные, которые были использованы. Теперь добавьте:

```
45 RESTORE
```

и осуществите ПРОГОН (RUN) программы еще раз. Указатель данных ВОССТАНОВЛЕН (RESTORE), и данные могут считываться непрерывно.

Средние величины

Практическое применение операторов READ (СЧИТЫВАНИЕ) и DATA (ДАННЫЕ) иллюстрируется следующей программой, которая считывает набор чисел и вычисляет их среднее

```
NEW
5 T = 0:CT = 0
10 READ X
20 IF X =-1THEN50:REM CHECK FOR FLAG
25 CT = CT + 1
30 T = T+X:REM UPDATE TOTAL
40 GOTO 10
50 PRINT"THERE WERE";CT;"VALUES READ"
60 PRINT "TOTAL = ";T
70 PRINT "AVERAGE=";T/GT
80 DATA 75,80,62,91,87,93,78,-1


RUN
THERE WERE 7 VALUES READ
TOTAL=566
AVERAGE=80.8571429
```

Рис.51

1 - ...проверка флага; 2 - обновление суммы; 3 - ...было...считано 8 величин; 4 - сумма; 5 - было считано 8 величин

Строка 5 устанавливает CT, счетчик, и T сумму, равными нулю. Строка 10 СЧИТЫВАЕТ (READ) величину и присваивает эту величину переменной X. Строка 20 проверяет, не является ли величина нашим флагом (в данном случае -1). Если считанная величина является частью считанных данных (DATA), то CT увеличивается на единицу и X складывается с суммой.

Когда осуществляется СЧИТЫВАНИЕ (READ) флага, программа переходит к строке 50, которая ВЫВОДИТ НА ПЕЧАТЬ (PRINT) число считанных величин. Строка 60 осуществляет ВЫВОД НА ПЕЧАТЬ (PRINT) сумму, а строка 70 делит сумму на количество величин для получения среднего арифметического.

Используя флаг в конце ДАННЫХ (DATA), Вы можете поместить любое количество величин в операторы DATA, которые могут занимать несколько строк - не заботясь о счете количества введенных величин.

Другой вариант оператора READ (СЧИТЫВАНИЕ) относится к присвоению информации из тех же строк DATA различным переменным. Эта информация может быть даже смесью цепочечных данных и числовых величин. Вы можете осуществлять все это в следующей программе, которая будет считывать имя, какой-либо счет, скажем в игре в боулинг, и осуществлять вывод на печать имени, счета и среднего счета:

```
    NEW

   10 READ N$,A,B,C
1  20 PRINT N$;""S SCORES WERE:";A;"";B;"";C
2  30 PRINT "AND THE AVERAGE IS:";(A+B+C)/3
   40 PRINT:GOTO 10
3  50 DATA MIKE,190,185,165,DICK,225,245,190
4  60 DATA JOHN.155,185,205,PAUL,160,179,187

    RUN

5 MIKE'S SCORES WERE:190 185 165
6 AND THE AVERAGE IS: 180

7 DICK'S SCORES WERE: 225 245 190
6 AND THE AVERAGE IS: 220
```

Рис.52

1 - "...счет был..."; 2 - "а среднее равно..."; 3 - ...Майк ...Дик...; 4 - ...Джон...Пол...; 5 - счет Майка был...; 6 - а среднее равно...; 7 - счет Дика был...

При прогоне программы операторы DATA были установлены в том же порядке, в котором оператор DATA ожидал информацию: имя (цепочка), затем три величины. Другими словами, N$ впервые получает ДАННЫЕ "MIKE", A в операторе READ соответствует 190 в операторе данных, "B" - 185 и "C" -165. Затем процесс повторяется в том же порядке для остальной информации (Дик и его счет, Джон и его счет и Пол и его счет).

Индексированные переменные

Ранее мы использовали только простые переменные языка БЕЙСИК, такие как A, A$ и NU для представления каких-либо величин. Это была буква, за которой следовала буква или одна цифра. Весьма сомнительно, чтобы в любой мыслимой программе нам потребовалось бы для обозначения имен переменных больше вариантов, чем представляют нам всевозможные сочетания допустимых букв или цифр. Однако существуют ограничения на способы применения переменных в программах.

Теперь введем понятие индексированной переменной.

```
A(1)
! !--------- индекс
!----------- переменная
```

Читается это так: A с индексом 1. Индексированная величина состоит из буквы, за которой следует индекс, заключенный в скобки. Отметьте различие между переменными A, A1, A(1). Каждая из этих переменных является уникальной. При этом только A(1) представляет собой индексированную переменную.

Индексированные переменные, как и простые переменные, обозначают ячейку памяти в компьютере. Представьте себе индексированные переменные как ящики для хранения информации, точно так же, как и простые переменные:

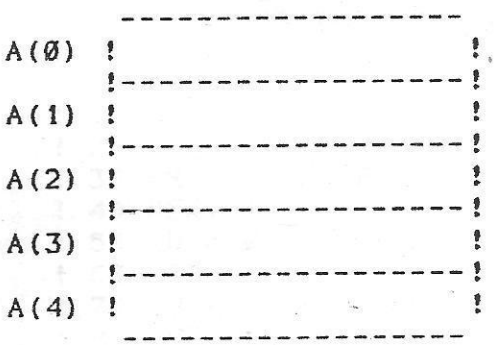


Рис.53

Если Вы напишете:

```
10 A(0) = 25: A(3) = 55: A(4) = -45.3
```

то память будет иметь следующий вид:

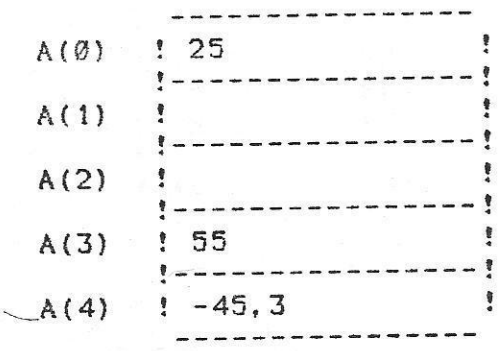


Рис.54

Эта группа индексированных переменных называется также массивом. В данном случае это одномерный массив. Позже мы введем понятие многомерных массивов

Индексы также могут быть более сложными и включать другие переменные или вычисления. Ниже приводятся действительные индексированные переменные:

```
A(X)   A(X+1)   A(2+1)   A(1*3)
```

Выражения в скобках оцениваются в соответствии с теми же правилами для арифметических операций, которые были описаны в Главе 2.

Теперь, когда Вам известны основные правила, рассмотрим, как можно использовать индексированные переменные. Одним способом их использования является запоминание списка чисел, введенных с операторами INPUT или READ.

Давайте используем индексированные переменные для вычисления средних величин другим способом.

```
5 PRINT CHR$(147)
10 INPUT "HOW MANY NUMBERS:";X
20 FOR A = 1 TO X
30 PRINT "ENTER VALUE#";A;:INPUT B(A)
40 NEXT
50 SU = Ø
60 FOR A = 1 TO X
70 SU = SU + B(A)
80 NEXT
90 PRINT:PRINT"AVERAGE = ";SU/X

RUN
```

```
! HOW MANY NUMBERS:? 5                    !
! ENTER VALUE#1 ? 125                     !
! ENTER VALUE#2 ? 167                     !
! ENTER VALUE#3 ? 189                     !
! ENTER VALUE#4 ? 167                     !
! ENTER VALUE#5 ? 158                     !
!                                         !
! AVERAGE = 161.2                         !
-------------------------------------------
```

Рис.55

Возможно, существует более легкий способ выполнения того, что осуществлено в вышеприведенной программе, однако эта программа показывает, как "действуют" индексированные переменные. Строка 1Ø запрашивает, сколько чисел будет введено. Эта переменная X действует как счетчик для цикла в пределах которого величины вводятся и присваиваются индексированной переменной B.

Каждый раз при цикле INPUT A увеличивается на единицу, и поэтому следующая введенная величина присваивается следующему элементу в массиве A. Например, первый раз в цикле A = 1, поэтому первая введенная величина присваивается B(1). Следующий раз A = 2, поэтому следующая величина присваивается B(2) и так далее пока все величины не будут введены.

Однако обратим внимание на одно большое различие. Когда все величины введены, они запоминаются в массиве, готовые к использованию различными способами. Ранее Вы имели текущую сумму каждый раз внутри цикла INPUT или READ, но не могли возвратиться к индивидуальным элементам данных без считывания информации заново.

В строках от 5Ø до 8Ø введен другой цикл для сложения различных элементов массива и затем вывода на дисплей средней величины. Эта отдельная часть программы показывает, что все величины запоминаются и к ним при необходимости имеется доступ.

Чтобы доказать, что в массиве все индивидуальные величины действительно запоминаются отдельно, наберите на клавиатуре сразу после прогона предыдущей программы:

```
FOR A = 1 TO 5 : ?B(A),: NEXT

B(1)     125
B(2)     167
B(3)     189
B(4)     167
B(5)     158
```

Когда содержимое массива будет выведено на печать, на дисплей будут выведены Ваши действительные величины.

Размерность массива

Если Вы попытаетесь в предыдущем примере ввести более 1Ø чисел, Вы получите сообщение DIMENSION ERROR (ОШИБКА В РАЗМЕР-

НОСТИ). Там, где необходимо, могут быть использованы массивы, содержашие до 11 элементов (индексы от Ø до 1Ø для одномерного массива), точно так же как простые переменные могут быть использованы где угодно в программе Массивы, имеющие более 11 элементов должны "заявляться" в операторе размерности.

Добавьте к программе следующую строку:

```
5 DIM B(1ØØ)
```

Это указывает компьютеру, что Вы будете иметь в массиве максимум 1ØØ элементов.

Оператор размерности может также быть использован с переменной, поэтому строка 5 может быть заменена следующей строкой (не забудьте стереть строку 5):

```
5 DIM B(X)
```

Это обеспечит размерность массива в соответствии с точным количеством величин, которые будут введены.

Однако, будьте внимательны. Если мы один раз определили размерность массива, мы не можем изменить ее в другой части программы. Однако, Вы можете иметь много массивов в программе и определить их размерность всех в одной строке, например:

```
1Ø DIM C(2Ø), D(5Ø), E(4Ø)
```

Моделирование игры в кости с массивами

По мере того как программы становятся более сложными, использование индексированных переменных сокращает количество требуемых операторов и облегчает составление программы.

Может быть использована и одна индексированная переменная, например, для слежения за количеством раз выпадания грани кости.

```
1    1 REM DICE SIMULATION : PRINT CHR$(147)
2    1Ø INPUT "HOW MANY ROLLS ";X
     2Ø FOR L = 1 TO X
     3Ø R = INT(6*RND<1>)+1
     4Ø F(R) = F(R) + 1
     5Ø NEXT L
3    6Ø PRINT "FACE", "NUMBER OF TIMES"
     7Ø FOR C = 1 TO 6 : PRINT C, F(C): NEXT
```

Рис.56

1 - моделирование игры в кости; 2 - сколько бросаний;
3 - ...грань

Массив F (для обозначения какой-либо одной грани кости - FACE) будет использован для подсчета того, сколько раз выпадает конкретная грань кости. Например, каждый раз, когда выпада-

ет 2, F(2) увеличивается на единицу. Используя тот же элемент массива для запоминания действительного числа на грани, которая выпала, мы устранили необходимость в пяти других переменных (одной на каждую грань) и многочисленных операторах для регистрации, какое число выпало.

Строка 10 запрашивает, сколько бросаний кости Вы хотите моделировать.

Строка 20 устанавливает цикл для реализации случайных бросаний и увеличения соответствующего элемента массива на единицу при каждом бросании кости.

После окончания всех бросаний строка 60 ВЫВОДИТ НА ПЕЧАТЬ (PRINT) заголовок, а строка 70 ВЫВОДИТ НА ПЕЧАТЬ (PRINT) количество раз выпадания той или иной грани.

Пробный прогон выглядит следующим образом:

```
 ----------------------------------------
! 1  HOW MANY ROLLS: ? 1000    3         !
!    FAGE  2            NUMBER OF TIMES  !
!      1                    148          !
!      2                    176          !
!      3                    178          !
!      4                    166          !
!      5                    163          !
!      6                    169          !
 ----------------------------------------
```

Рис.57

1 - сколько бросаний кости? 1000; 2 - грань; 3 - номер бросания

Однако, стандартной загрузки осуществлено не было!

Ниже для сравнения дается выражение, представляющее собой способ перезаписи той же программы, но без индексированных переменных. Не стоит набирать ее на клавиатуре, однако имеет смысл обратить внимание на дополнительные операторы, которые здесь необходимы.

```
1    10 INPUT "HOW MANY ROLLS ";X
     20 FOR L = 1 TO X
     30 R = INT(6*RND(1))+1
     40 IF R = 1 THEN F1 = F1 + 1:NEXT
     41 IF R = 2 THEN F2 = F2 + 1:NEXT
     42 IF R = 3 THEN F3 = F3 + 1:NEXT
     43 IF R = 4 THEN F4 = F4 + 1:NEXT
     44 IF R = 5 THEN F5 = F5 + 1:NEXT
     45 IF R = 6 THEN F6 = F6 + 1:NEXT
2    60 PRINT "FACE", "NUMBER OF TIMES"
     70 PRINT 1, F1
     71 PRINT 2, F2
     72 PRINT 3, F3
```

```
73 PRINT 4, F4
74 PRINT 5, F5
75 PRINT 6, F6
```

Рис. 58

1 - ... "сколько бросаний (кости)"; 2 - ..."грань", "количество раз"

Программа увеличилась в размере с 8 до 17 строк. В более длинных программах экономия места при использовании индексированных переменных будет еще более значительной.

Двумерные массивы

Ранее в этой главе Вы экспериментировали с одномерными массивами. Этот тип массива был представлен в виде группы последовательных ящиков в памяти, причем каждый ящик содержит элемент массива. А как, по Вашему мнению, выглядит двумерный массив?

Во-первых, двумерный массив обозначается так:

```
A(4,6)
! ! !
!  -------- индексы
!---------- имя массива
```

и может быть представлен в виде двумерной сетки в памяти:

```
     Ø     1     2     3     4     5     6
   ---------------------------------------------
Ø  !     !     !     !     !     !     !     !
   !-----------------------------------------!
1  !     !     !     !     !     !     !     !
   !-----------------------------------------!
2  !     !     !     !     !     !     !     !
   !-----------------------------------------!
3  !     !     !     !     !     !     !     !
   !-----------------------------------------!
4  !     !     !     !     !     !     !     !
   ---------------------------------------------
```

Можно считать, что индексы обозначают в таблице ряд и колонку, на пересечении которых находится клетка, в которую помешен конкретный элемент массива.

```
A(3,4) = 255
  ! !
  ! !----------- колонка
  !------ ряд
```

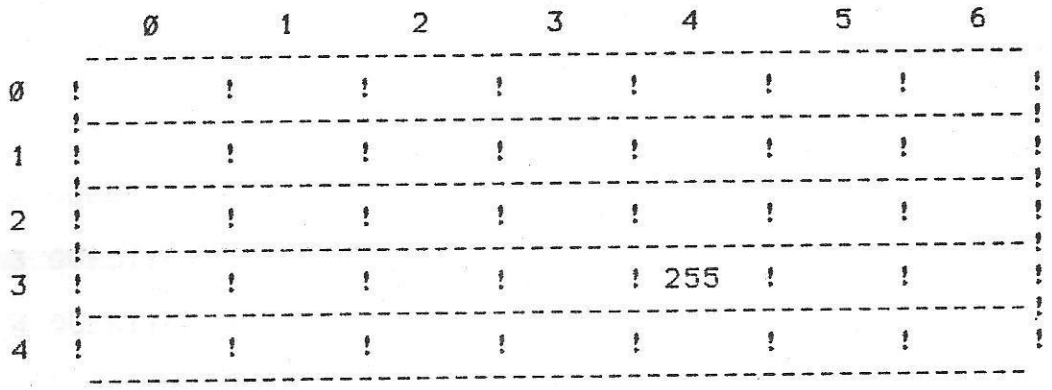


Рис.59

Если мы присвоили величину 255 A(3,4), то можно считать, что величина 255 помещена в клетку на пересечении четвертой колонки и третьего ряда таблицы.

С двумерными массивами можно обращаться в соответствии с теми же правилами, которые были установлены для одномерных массивов:

Должна быть определена их размерность:	DIM A(20,20)
Присвоение данных:	A(1,1) = 255
Присвоение величин другим переменным:	AB = A(1,1)
ВЫВОД НА ПЕЧАТЬ (PRINT) величин:	PRINT A(1,1)

Если двумерные массивы работают так же, как их одномерные аналоги, уместен вопрос, какими дополнительными возможностями обладают двумерные массивы?

Можно ли, например, с помощью двумерного массива табулировать результаты опросника для членов Вашего клуба, который содержит четыре вопроса, причем каждый вопрос может иметь до трех возможных ответов? Эта задача может быть представлена таким образом:

```
1     CLUB QUESTIONNAIRE

2  Q1: ARE YOU IN FAVOR OF RESOLUTION #1?

      ---              ---              ---
     !   ! 1-YES      !   ! 2-NO       !   ! 3-UNDECIDED
      ---              ---              ---
       3                4                5
```

Рис.60

1 - опросник для клуба; 2 - вопрос 1: высказываетесь ли Вы за резолюцию #1? 3 - да; 4 - нет; 5 - воздерживаюсь от ответа

Таблица массива для этой задачи может быть представлена так:

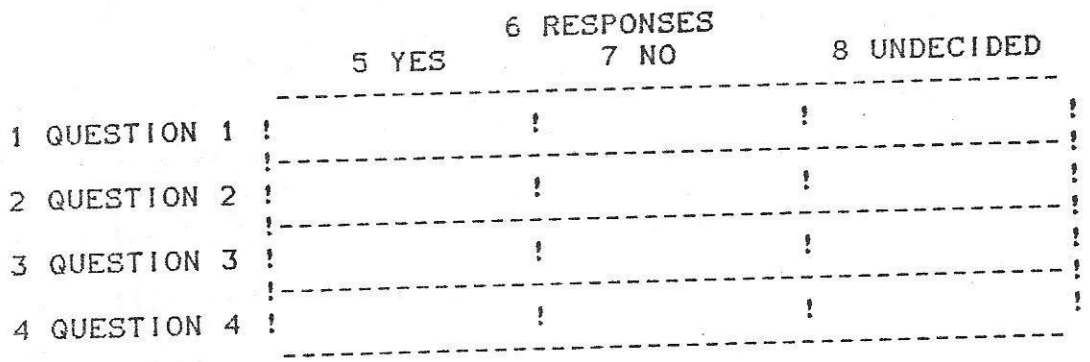


Рис.61

1 - вопрос1; 2 - вопрос 2; 3 - вопрос 3; 4 - вопрос 4;
5 - да; 6 - ответы; 7 - нет; 8 - воздерживаюсь от ответа

Программа для выполнения действительного табулирования использует многие из тех способов программирования, которые были показаны до сих пор. Даже если в настоящее время Вам эта программа не нужна, проверьте, сможете ли Вы понять, как она работает.

Ядром программы является двумерная сетка размером 4 на 3, A(4,3). Полные ответы для каждого возможного ответа на каждый вопрос помещаются в соответствующую клетку (элемент) массива. В целях упрощения мы не используем первые ряды и первую колонку (A(Ø,Ø) - A(Ø,4)). Помните, однако, что эти элементы всегда присутствуют в любом массиве, который Вы проектируете.

На практике, если на вопрос 1 дается ответ ДА (YES), то A(1,1) увеличивается на единицу - ряд 1 для вопроса 1 и колонки 1 для ответа ДА (YES). Остальные вопросы и ответы регистрируются по этому же образцу. Ответ НЕТ (NO) для вопроса 3 добавит единицу к элементу A(3,2) и так далее.

```
  1 !4Ø PRINT"QUESTION#:";R
  2 !5Ø PRINT "1-YES 2-NO 3-UNDECIDED"
    !6Ø PRINT"WHAT WAS THE RESPONSE:";
    !61 GET C:IFC<1orC>3 THEN61
  3 !65 PRINT C:PRINT
    !7Ø A(R,C)=A(R,C)+1:REM UPDATE ELEMENT
    !8Ø NEXT R
  4 !85 PRINT
  5 !9Ø PRINT"DO YOU WANT TO ENTER ANOTHER":PRINT
    !   "RESPONSE (Y/N)";
    !1ØØ GET A$:IF A$=""THEN 1ØØ
```

```
     !110 IF A$="Y"THEN 20                                    !
6    !120 IF A$<>"N"THEN 100                                  !
     !130 PRINT"(SHIFT/GLR/HOME)","THE TOTAL                  !
7    !    RESPONSES WERE:":PRINT                              !
8    !140 PRINT SPC(18);"RESPONSE"                            !
     !141 PRINT "QUESTION","YES","NO",                        !
     !    "UNDECIDED"                                         !
     !142 PRINT "--- ---------------"                         !
     !150 FOR R = 1TO4                                        !
     !160 PRINT R,A(R,1),A(R,2),A(R,3)                        !
     !170 NEXT R                                              !
     ! RUN                                                    !
     !                                                       !
9    ! QUESTION #:1                                          !
10   ! 1-YES 2-NO 3-UNDECIDED                                !
11   ! WHAT WAS THE RESPONSE : 1                             !
     !                                                       !
12   ! QUESTION #:2                                          !
10   ! 1-YES 2-NO 3-UNDECIDED                                !
11   ! WHAT WAS THE RESPONSE : 1                             !
     !                                                       !
13   ! And so on...                                          !
     !                                                       !
14   !                                                       !
     ! THE TOTAL RESPONSE WERE 18                            !
     !          16       17              19                 !
15   ! QUESTION   YES     NO          UNDECIDED              !
     ! --------   ---------------------------               !
     !    1         6       1             0                 !
     !    2         5       2             0                 !
     !    3         7       0             0                 !
     !    4         2       4             1                 !
      ----------------------------------------------------
```

Рис.62

1 - ..."1-да, 2-нет; 3-воздерживаюсь от ответа"; 2 - ... "каков был ответ..."; 3 - ...обновление элемента; 4 - ..."вы хотите ввести другой"; 5 - ..."ответ..."; 6 - ..."сумма ответов была..."; 7 - ..."ответ"; 8 - ..."вопрос", "да", "нет", "воздерживаюсь от ответа"; 9 - вопрос #1; 10 - 1-да, 2-нет, 3-воздерживаюсь от ответа; 11 - каков был ответ...; 12 - вопрос #2; 13 - и так далее; 14 - сумма ответов была; 15 - вопрос; 16 - да; 17 - нет; 18 - ответ; 19 - воздерживаюсь от ответа

Приложения

Введение

Теперь, когда Вы более подробно ознакомились с Вашим компьютером "Коммодор-64", мы информируем Вас о том, что наша помощь пользователю этим не ограничивается. Возможно, Вам не известно, что фирма "Коммодор" имеет 23-летний опыт работы. В 70-х годах мы представили первый автономный персональный компьютер (PET). С тех пор фирма "Коммодор" занимает ведущее место в области компьютеров во многих странах мира. Наши возможности разрабатывать и изготавливать наши собственные компьютерные микросхемы позволяют нам предлагать Вам новые компьютеры с улучшенными характеристиками по ценам, которые ниже тех, которые Вы можете ожидать для такого высокого технического уровня.

фирма "Коммодор" считает себя обязанной оказывать помощь не только Вам, конечному пользователю, но также и торговому агенту, у которого Вы приобрели Ваш компьютер, журналам, которые публикуют статьи, знакомящие Вас с новыми применениями или способами, и, что весьма важно, разработчикам программного обеспечения, которые изготавливают программы на кассете, диске и ленте для их применения в Вашем компьютере. Мы рекомендуем Вам учредить или вступить в существующий клуб пользователей компьютерами фирмы "Коммодор", где Вы можете ознакомиться с новыми способами, обменяться идеями и поделиться своими открытиями. Мы издаем два отдельных журнала, которые содержат советы по программированию, а также информацию о новых идеях и аппаратуре для компьютерного применения (см. Приложение O).

На североамериканском континенте фирма "Коммодор" обеспечивает "Информационную сеть фирмы "Коммодор" по информационному обслуживанию компьютерной техники. Для того, чтобы пользоваться услугами этой сети, необходимо иметь только компьютер "Коммодор-64", и нашу интерфейсную телефонную кассету VICMODEM (или другой совместимый модем).

Приводимые в настоящем руководстве Приложения содержат графики, таблицы и другую информацию, которая поможет Вам осуществить программирование для Вашего компьютера "Коммодор-64" быстрее и эффективнее. В них также содержится важная информация по широкому ассортименту аппаратуры, выпускаемой фирмой "Коммодор", которая может Вас заинтересовать, а также библиография из двадцати наименований (книги и журналы), которая может помочь Вам в повышении квалификации программиста и держать Вас в курсе самой последней информации относительно Вашего компьютера и внешних устройств.

Приложение А

Принадлежности и программное обеспечение для компьютера "Коммодор-64"

Принадлежности

К компьютеру "Коммодор-64" могут подключаться устройства памяти VIC2Ø и принадлежности фирмы "Коммодор" - кассетный магнитофон C2N, дисковое ЗУ, модем, принтер - так что Ваша система может быть расширена для удовлетворения новых требований.

- Кассетный магнитофон C2N - Это недорогое запоминающее устройство обеспечивает хранение программ и данных на магнитной ленте и их ввод в компьютер. Это устройство может также использоваться для ввода в компьютер предварительно записанных программ.

- Диск - В дисковом ЗУ с одним диском применяются стандартные гибкие диски диаметром 5 1/4 дюйма (133 мм), что приблизительно равно диаметру пластинки на 45 оборотов в минуту. На диске хранятся программы и данные. Диски обеспечивают более быстрый доступ к данным и содержат до 17ØØØØ знаков информации каждый. Дисковые ЗУ являются "разумными", т.е. они имеют свой собственный микропроцессор и память. Диски не требуют от компьютера никаких ресурсов, таких как использование части основной памяти.

- Модем - Недорогое связное устройство, модем типа VICMODEM позволяет осуществить доступ к другим компьютерам с помощью обычных телефонных линий связи. Пользователи при этом имеют доступ к большим базам данных, таким как "Источник" (The source), "Служба обслуживания компьютеров" (Compuserve) и "Служба передачи новостей об индексах Доу-Джонса" (только для Северной Америки).

- Принтер - Принтер типа VIC выдает печатные копии программ, данных или графиков. Этот точечно-матричный принтер, работающий со скоростью 3Ø знаков в секунду, использует обычную рулонную бумагу и другие недорогие расходные материалы. Принтер подключается непосредственно к компьютеру "Коммодор-64" без каких-либо дополнительных интерфейсов.

- Интерфейсные кассеты - Для компьютера "Коммодор-64" имеются специальные кассеты, которые обеспечивают возможность подключения к системе различных устройств, таких как модемы, принтеры, контроллеры и другие приборы.

С помощью специальной кассеты IEEE-488 (IEE-488 Cartridge) к компьютеру "Коммодор-64" может подключаться широкий диапазон внешних устройств фирмы CBM, включая дисковые запоминающие устройства и принтеры.

Специальная кассета Z8Ø позволит Вам пользоваться устройствами CP/Mx) на компьютере "Коммодор-64", что обеспечит Вам

---

x) CP/M - зарегистрированный торговый знак фирмы "Диджитал рисерч"

доступ к самой большой базе в мире по микрокомпьютерным применениям.

Программное обеспечение

Для компьютера "Коммодор-64" предлагается несколько категорий программного обеспечения, что обеспечивает большое разнообразие личных, игровых и образовательных применений компьютера.

Деловые программы

- Пакет программ "Электроник Спредшит" (на "электронных листках") позволит Вам планировать бюджеты и осуществлять анализ типа "что, если?" в деловой деятельности. Со специальной графической программой, поставляемой отдельно, можно создавать значащие графы на основе данных, зафиксированных на этих "электронных листках".
- С помощью пакета финансового планирования Вы можете легко осуществлять планирование финансовых операций, например, погашение займов.
- Ряд программ распределения производственного времени поможет Вам в управленческой деятельности и организации загрузки на рабочих местах.
- Удобные для применения программы с базой данных позволят Вам следить за информацией такого типа, как списки почтовых отправлений, номеров телефонов, инвентаря, а также организовать выдачу информации в удобной форме.
- Профессиональные программы обработки слов помогут Вам превратить "Коммодор-64" в процессор для обработки словарного текста с полным набором необходимых характеристик. При этом можно легко осуществлять набор текста и редактирование содержимого запоминающего устройства, а также писем и другого текстового материала.

Игровые применения

- С помощью вставных специальных кассет на компьютере "Коммодор-64" могут быть реализованы самые сложные игры. Вы можете иметь интересные игры, которыми Вы будете наслаждаться часами. Игровые программы используют графику с высоким разрешением и полный звуковой диапазон, которым обладает компьютер "Коммодор-64"
- Ваш компьютер "Коммодор-64" позволит Вам наслаждаться играми для ЭВМ MAX, так как эти два компьютера располагают совместимыми друг с другом специальными кассетами программного обеспечения.

Образовательные программы

Компьютер "Коммодор-64" - это учитель, который никогда не устает и который всегда уделяет Вам персональное внимание. Кроме доступа к обширному набору образовательных программ PET, дополнительные образовательные языки, доступные для компьютера "Коммодор-64", обеспечивают такие важные пакеты программ, как PILOT, LOGO и другие.

Приложение Б

Развитые операции при работе с кассетным ЗУ

Помимо записи Ваших программ на ленту, компьютер "Коммодор-64" может также записывать на ленту величины переменных и другие типы данных. Запись данных осуществляется группами, которые называются файлами. Это позволяет Вам хранить больше информации, чем может быть единовременно введено в основную память компьютера.

С данными используются следующие операторы: OPEN (ОТКРЫТЬ), CLOSE (ЗАКРЫТЬ), PRINT (ВЫВЕСТИ НА ПЕЧАТЬ), INPUT (ВВЕСТИ) и GET (ВЗЯТЬ). Системная переменная ST (статус) используется для проверки маркеров на ленте.

При записи данных на магнитную ленту используются те же принципы, что и при выводе информации на дисплей компьютера. Однако вместо ВЫВОДА информации (PRINT) на экран дисплея, информация выводится на магнитную ленту, что осуществляется применением разновидности команды PRINT (ВЫВЕСТИ НА ПЕЧАТЬ) - команды PRINT#.

Сказанное иллюстрируется следующей программой:

```
1    10 PRINT "WRITE-TO-TAPE-PROGRAM"
2    20 OPEN 1,1,1,"DATA FILE"
3    30 PRINT "TYPE DATA TO BE STORED OR TYPE STOP"
     50 PRINT
     60 INPUT "DATA";AД
     70 PRINT#1,AД
     80 IF,AД<>"STOP" THEN 50
     90 PRINT
4   100 PRINT "CLOSING FILE"
    110 CLOSE 1
```

Рис.63

1 - ..."программа записи на магнитную ленту"; 2 - ...1,1, 1, "файл данных"; 3 - ..."наберите на клавиатуре данные, подлежащие запоминанию или слово STOP"; 4 - ..."завершающий файл"

Первое, что Вы должны сделать, это ОТКРЫТЬ (OPEN) файл (в данном случае ФАЙЛ ДАННЫХ (DATA FILE). Это осуществляется в строке 10.

В строке 60 программа запрашивает, какие данные Вы хотите вывести на ленту. Строка 70 осуществляет вывод в память то, что Вы набрали на клавиатуре - содержимое A$ - осуществляется запись на ленту. Далее процесс продолжается.

Если Вы наберете на клавиатуре STOP (ОСТАНОВ), строка 110 ЗАКРЫВАЕТ (CLOSE) файл.

Чтобы считать записанную информацию, перемотайте ленту обратно и воспользуйтесь следующей программой:

```
1    10 PRINT "READ-TAPE-PROGRAM"
2    20 OPEN 1,1,0"DATA FILE"
3    30 PRINT "FILE OPEN"
     40 PRINT
     50 INPUT#1, A$
     60 PRINT A$
     70 IF A$ = "STOP" THEN END
     80 GOTO 40
```

Рис.64

1 - ..."программа считывания с ленты"; 2 - ...1,1,0, "файл данных"; 3 - ..."файл открыт"

Как и прежде, сначала необходимо ОТКРЫТЬ (OPEN) ФАЙЛ ДАННЫХ (DATA FILE). В строке 50 программа ВВОДИТ (INPUT) A$ с ленты и ВЫВОДИТ (PRINT) A$ на экран. Затем весь процесс повторяется заново, пока не будет найдено слово STOP (ОСТАНОВ), которое ЗАКАНЧИВАЕТ (END) программу.

Для считывания данных с ленты может быть также использована разновидность оператора GET (ВЗЯТЬ) - GET#. Поэтому Вы можете заменить строки 50-80 в вышеуказанной программе следующими строками:

```
50 GET#1,'A$
60 IF A$ = "" THEN END'
70 PRINT A$, ASC(A$)
80 GOTO 50
```

Приложение В

Язык БЕЙСИК в компьютере "Коммодор-64"

Данное руководство познакомило Вас с введением в язык БЕЙСИК. Этого было достаточно для того чтобы Вы почувствовали, что такое компьютерное программирование и познакомились с некоторыми элементами применяемого словаря. В Приложении В приводится полный список правил (синтаксис) языка БЕЙСИК, каким он применяется в компьютере "Коммодор-64", а также даются краткие определения. Мы рекомендуем Вам поэкспериментировать с этими командами. Еще раз напоминаем Вам, что Вы не можете вывести компьютер из строя, набирая с клавиатуры любые програм-

мы, и лучшим способом изучения работы компьютера является именно составление и набор на клавиатуре

Настоящее приложение подразделяется на секции в соответствии с различными типами операций в языке БЕЙСИК. Эти типы операций таковы:

1. Переменные и операторы: описывают различные типы переменных, разрешенные имена переменных, а также арифметические и логические операторы.
2. Команды: описывают команды, применяемые в программах для работы, редактирования, запоминания и стирания.
3. Операторы : (statements): описывают операторы программ в языке БЕЙСИК в нумерованных строках программ.
4. Функции: описывают цепочечные (string), числовые функции и функцию вывода на печать.

Переменные

Компьютер "Коммодор-64" использует три типа переменных в языке БЕЙСИК: действительные числовые, целочисленные и цепочечные (буквенно-цифровые) переменные.

Имена переменных могут состоять из одной буквы, буквы, за которой следует цифра или двух букв.

Целочисленная переменная обозначается знаком % после имени переменной. Цепочечные переменные для своего обозначения имеют знак % после своего имени.

Примеры

Имена действительных переменных: A, A5, BZ
Имена целочисленных переменных: A%, A5%, BZ%
Имена цепочечных переменных (последовательностей): A$, A5$, BZ$

Массивы представляют собой списки переменных с тем же именем, использующие дополнительные числа для определения элемента массива. Массивы определяются с помощью оператора DIM (РАЗМЕРНОСТЬ) и могут содержать переменные с плавающей запятой (точкой), целочисленные или цепочечные переменные. За именем переменной массива идут скобки ( ), в которых содержится количество переменных в списке.

```
A(7), BZ%(11), A$(50), PT(20,20)
```

Примечание. Существуют три имени переменных, которые выделяются для использования в компьютере "Коммодор-64" и не могут быть определены Вами. Эти переменные таковы: ST, TI и TI$. ST - это переменная статуса, которая относится к операциям входа/выхода. Величина переменной ST будет меняться, если имеется задача, в которой осуществляется загрузка с ленты или диска.

TI и TI$ - это переменные, которые относятся к часам реального времени, встроенным в компьютер. Переменная TI обновляется каждые одну шестидесятую секунды. Она начинается с нуля при включении компьютера и сбрасывается только изменением величины TI$.

Переменная TI$ представляет собой цепочку (последовательность), которая постоянно обновляется системой. Первые два знака представляют собой количество часов, третий и четвертый знаки- это минуты, а пятый и шестой знаки - это секунды. Эта переменная может быть задана любой числовой величиной и будет обновляться с этой точки.

TI$ = "101530" устанавливает часы в 10 час. 15 мин. 30 сек.

Эти часы сбрасываются, когда компьютер выключается, и стартуют с нуля, когда система включается снова.

Операторы

Арифметические операторы используют следующие знаки:

+ Сложение
- Вычитание
* Умножение
/ Деление
^ Возведение в степень

Если в строке содержится несколько операторов, то существует установленный порядок, в котором выполняются указанные операции. Если в одной и той же строке используется несколько операторов, то компьютер устанавливает для них следующий приоритет: сначала осуществляется возведение в степень, затем умножение и деление и, наконец, сложение и вычитание.

Вы можете менять порядок выполнения операций, заключая в скобки те вычисления, которые Вы хотите осуществить первыми. Операции, заключенные в скобки, выполняются ранее выполнения других операций.

Существуют также операции для равенств и неравенств:

= Равно
< Меньше чем
> Больше чем
<= Меньше или равно
>= Больше или равно
<> Не равно

И, наконец, существует три логических оператора:

AND (И)
OR (ИЛИ)
NOT (НЕ)

Чаще всего эти операторы используются для связи нескольких формул в операторах IF ... THEN (ЕСЛИ ... ТО). Например,

```
IF A = B AND C = D THEN100  (Требуется, чтобы обе части
                             были истинны)
IF A = B OR C = D THEN 100  (Требуется, чтобы хотя бы одна
                             часть была истинной)
```

Команды

CONT (ПРОДОЛЖИТЬ)

Эта команда используется для повторного старта выполнения программы, которая была остановлена либо нажатием клавиши STOP (ОСТАНОВ), либо оператором STOP (ОСТАНОВ), либо же оператором END (ЗАКОНЧИТЬ) в программе. Программа начнется с того места, где она была остановлена.

Оператор CONT не будет работать, если Вы изменили программу или добавили к ней строки, или же если программа остановилась из-за ошибки, или же если Вы вызвали состояние ошибки перед попыткой запустить программу снова. В этих случаях Вы получите сообщение об ошибке Can'T CONTINUE ERROR (Ошибка - продолжение невозможно).

LIST (СПИСОК)

Команда LIST позволяет Вам посмотреть на строки программы в языке БЕЙСИК, находящейся в памяти. Вы можете запросить всю программу на дисплей или же только строки с определенными номерами:

LIST	Показывает всю программу
LIST 10 -	Показывает только со строки 10 до конца
LIST 10	Показывает только строку 10
LIST-10	Показывает только строки от начала программы до строки 10
LIST 10-20	Показывает строки от 10 до 20 включительно

LOAD (ЗАГРУЗИТЬ)

Эта команда используется для передачи программы с ленты или диска в память компьютера для использования этой программы. Если Вы просто наберете на клавиатуре слово LOAD и нажмете клавишу RETURN, то в память будет помещена первая программа, найденная в установленной в накопитель кассете с магнитной лентой. За командой может следовать имя программы, заключенное в кавычки. Затем за именем может следовать запятая и число или числовая переменная, которые играют роль указателя устройства для индикации того, откуда взята эта программа.

Если никакого номера устройства не приводится, то компью-

тер "Коммодор-64" воспринимает это как указание на устройство # 1, которое является накопителем на кассетной магнитной ленте. Другим устройством, которое обычно применяется с командой LOAD, является накопитель на магнитном диске, который считывается устройством #8.

LOAD	Считывает с ленты следующую программу
LOAD "HELLO"	Осуществляется поиск на ленте программы с именем HELLO и, когда она найдена, осуществляется ее загрузка
LOAD A$	Осуществляется поиск программы, имя которой содержится в переменной A
LOAD "HELLO",8	Осуществляется поиск программы HELLO на дисковом накопителе
LOAD "*",8	Осуществляется поиск первой программы на дисковом накопителе

Если Вы хотите загрузить программу в машинном коде без перемещения в память, то Вы должны добавить вторичный адрес 1.

LOAD "M/C PROGRAM",1,1 Загружает с ленты машинный код без его перемещения

Эта команда стирает всю программу из памяти, а также стирает все переменные, которые, возможно, были использованы. Если не было дано команды на сохранение ее в памяти (SAVE), то эта программа теряется. ПРИ ИСПОЛЬЗОВАНИИ ЭТОЙ ПРОГРАММЫ БУДЬТЕ ВНИМАТЕЛЬНЫ.

Команда NEW (НОВАЯ) также может быть использована в качестве оператора программы на языке БЕЙСИК. Когда программа достигает этой строки, программа стирается. Это бывает удобным, когда Вы хотите все очистить после выполнения программы.

RUN (ПРОГОН)

Эта команда осуществляет выполнение программы, если программа загружена в память. Если вслед за словом RUN не идет номер строки, то компьютер начнет работу с наименьшего номера строки. Если номер строки обозначен, то программа будет выполняться с указанной строки.

RUN	Программа стартует со строки с наименьшим номером
RUN 100	Программа стартует со строки 100
RUN X	Вы получите сообщение UNDEFINED STATEMENT ERROR (ОШИБКА - ОПЕРАТОР НЕ ОПРЕДЕЛЕН). Вы всегда должны приводить действительный номер строки, а не наименование переменной.

SAVE (СОХРАНИТЬ)

Эта команда сразу же записывает программу в память накопителя на магнитной ленте или на диске. Если Вы наберете на клавиатуре слово SAVE и нажмете на клавишу RETURN, то программа будет записана на кассетную ленту. Компьютер не обращает внимания, записано ли уже что-либо на этой ленте, поэтому будьте в этом случае внимательны, так как при записи на ленту этой программы Вы можете непреднамеренно стереть другую нужную Вам программу.

Если Вы наберете на клавиатуре слово SAVE, за которым в кавычках идет имя или же идет цепочечная переменная, то компьютер присвоит программе это имя, поэтому в будущем эту программу будет легко отыскать и использовать. За именем может также следовать номер устройства.

После номера устройства может идти запятая или второй номер, 0 или 1. Если вторым номером является 1, компьютер "Коммодор-64" поставит после Вашей программы маркер END-OF-TAPE (КОНЕЦ ЛЕНТЫ). Это означает, что компьютер не будет осуществлять дальнейший поиск на этой ленте, если Вы дадите дополнительную команду LOAD. Если Вы попытаетесь ЗАГРУЗИТЬ (LOAD) программу, а компьютер найдет один из этих маркеров, то Вы получите сообщение FILE NOT FOUND ERROR (ОШИБКА - ФАЙЛ НЕ НАЙДЕН).

SAVE	Записывает программу на ленте без имени
SAVE "HELLO"	Записывает программу на ленте с именем HELLO
SAVE A$	Записывает программу на ленте с именем в A$
SAVE "HELLO",8	Записывает программу на диске с именем HELLO
SAVE "HELLO",1,1	Записывает программу на ленте с именем HELLO, после чего следует маркер END-OF-TAPE (КОНЕЦ ЛЕНТЫ)

VERIFY (ПРОВЕРИТЬ)

Эта команда приказывает компьютеру сверить программу на диске или ленте с командой, хранящейся в памяти компьютера. Это действие подтверждает, что программа действительно записана в накопителе, в случае если лента или диск вызывают опасения или же в процессе выполнения команды SAVE были возможны какие-либо сбои. Если после слова VERIFY не идет ничего, то компьютер сравнивает следующую команду на ленте, независимо от ее имени, с программой, хранящейся в памяти.

Если за словом VERIFY следует имя программы или цепочечная переменная, то будет осуществляться поиск указанной программы, а затем осуществляться проверка. В команду проверки может быть также включен номер устройства.

VERIFY	Проверяет следующую программу на ленте
VERIFY "HELLO"	Осуществляется поиск программы HELLO, которая затем сравнивается с программой в памяти
VERIFY "HELLO",8	Осуществляется поиск программы HELLO на диске, затем эта программа сравнивается с программой в памяти

Чтобы проверить, находится ли уже программа на ленте, просто наберите на клавиатуре VERIFY, и компьютер скажет, какую программу он отыскал (если он отыскал какую-либо программу).

Операторы

CLOSE

Эта команда завершает и закрывает все файлы, которые использовались с операторами OPEN (ОТКРЫТЬ). Число, следующее за оператором CLOSE, представляет собой номер файла, который необходимо закрыть.

CLOSE 2 Закрывается только файл 2

CLR

Эта команда стирает в памяти все переменные, но саму программу не затрагивает. Эта команда автоматически выполняется, когда дается команда RUN (ПРОГОН).

CMD

Эта команда посылает выход, который обычно отсылается на экран (например, операторы PRINT, LIST, но не POKE), на другое устройство. Этим устройством может быть принтер или же файл данных на ленте или диске. Это устройство или файл должны быть предварительно ОТКРЫТЫ (команда OPEN). Команда CMD должна иметь после себя число или числовую переменную, указывающие файл.

OPEN 1,4	ОТКРЫВАЕТ устройство #4 (этим устройством является принтер
CMD 1	Теперь весь выход идет на принтер
LIST	Листинг программы теперь выводится не на экран, а на принтер

DATA

За этим оператором следует список наименований, которые будут использоваться операторами READ. Наименования могут

представлять собой цепочки текста или числовые величины, и они должны отделяться друг от друга запятыми. Цепочечные наименования не следует заключать в кавычки, если только в них не содержатся пробел, двоеточие или запятая. Если между двумя запятыми ничего не содержится, то этот случай компьютером при считывании (READ) будет восприниматься как нуль для числа или пустая цепочка.

```
DATA 12, 14.5, "HELLO, MOM", 3.14, PART1
```

DEF FN

Эта команда позволяет Вам определить сложное вычисление как функцию с коротким именем. Если какая-либо длинная формула многократно используется в программе, это позволяет сэкономить место и время.

Имя функции будет обозначаться буквами FN и любым разрешенным именем переменной (одним-двумя знаками). Сначала Вы должны определить функцию с помощью оператора DEF, за которым следует имя функции. За именем функции следуют скобки, в которых стоит числовая переменная. Затем следует действительная формула, которую Вы хотите определить с именем переменной в нужном месте. Затем Вы можете "вызывать" формулу, заменяя указанную переменную любым нужным числом.

```
                  *
10 DEF FNA(X) = 12 (34.75 - X/.3)
20 PRINT FNA(7)              !
           !                 !
            ----------------
```

Во второй строке этой формулы переменная X заменена числом 7. В этом примере результат равен 137.

DIM

Когда Вы используете более одиннадцати элементов массива, Вы должны выполнить оператор DIM для массива. Имейте в виду, что целый массив занимает в памяти определенное место, поэтому не создавайте массив, много больший того, что Вам требуется в действительности. Чтобы представить себе количество переменных, создаваемых оператором DIM, умножьте друг на друга полное количество элементов по каждому размеру массива.

```
10 DIM AØ(40), B7(15), CC%(4,4,4)
         !       !         !
  41 элемент  16 элементов  125 элементов
```

В одном операторе DIM Вы можете определить размерность нескольких массивов, однако, следите, чтобы размерность каждого массива была определена только один раз.

Размерность массива может быть установлена заново. Для этого сначала надо использовать оператор CLR. Однако это приведет к стиранию всех созданных до этого переменных.

END

Когда программа встречает оператор END, программа останавливается. Вы можете для рестарта программы использовать оператор CONT.

FOR ... TO ... STEP

Этот оператор работает совместно с оператором NEXT для повторения части программы установленное количество раз. Формат имеет следующий вид:

FOR (ДЛЯ) (Имя переменной) = (Начало счета) TO (ДО) (Конец счета)

STEP (ШАГ) (Переменная цикла) = (Счет по)

Переменная цикла будет увеличиваться или уменьшаться на величину шага во время выполнения программы. Если ШАГ (STEP) не определен, принимается, что он равен 1. Стартовый счет и конечный счет являются пределами для величины переменной цикла.

```
10 FOR L = 1 TO 10 STEP.1
20 PRINT L
30 NEXT L
```

После окончания величины цикла может следовать слово STEP и другое число или переменная. В таком случае вместо единицы каждый раз добавляется величина, следующая за словом STEP. Это позволяет осуществлять отсчет назад или вести отсчет дробными величинами.

GET

Оператор GET позволяет Вам брать данные с клавиатуры, по одному знаку за один раз. Когда выполняется оператор GET, знак, который набирается на клавиатуре, присваивается переменной. Если после этого оператора не ставится никакого знака, тогда переменной присваивается нулевой (пустой) знак.

За оператором GET идет имя переменной, обычно цепочечная переменная. Если была использована числовая переменная, а нажата нецифровая клавиша, программа остановится и будет выдано сообщение об ошибке. Оператор GET может быть помещен в цикл для проверки любого нулевого (пустого) результата. Цикл будет продолжаться до тех пор, пока не будет нажата соответствующая клавиша.

```
10 GET A$: IF A$ = ""THEN 10
```

GET#

Оператор GET# используется с предварительно ОТКРЫТЫМ (OPEN) устройством или файлом для ввода одного знака за один раз от этого устройства или файла.

```
GET#1,A$
```

Это приведет ко вводу одного знака от файла данных.

GOSUB

Этот оператор аналогичен оператору GOTO, за исключением того, что компьютер помнит, какая строка программы была выполнена последней перед оператором GOSUB. Когда встречается строка с оператором RETURN, программа переходит обратно к оператору, который непосредственно следует за оператором GOSUB. Это бывает удобным, когда в Вашей программе имеется стандартная подпрограмма, которая встречается во многих частях программы. Вместо того, чтобы несколько раз набирать с клавиатуры стандартную подпрограмму, реализуйте оператор GOSUB каждый раз, когда требуется эта стандартная подпрограмма.

```
20 GOSUB 800
```

GOTO или GO TO

Когда встречается оператор с командой GOTO, следующей строкой, которая будет выполняться, будет строка, номер которой стоит непосредственно после оператора GOTO.

IF ... THEN

Этот оператор позволяет компьютеру проанализировать ситуацию и реализовать два возможных пути, в зависимости от обстоятельств. Если выражение является истинным, то осуществляется оператор, следующий за словом THEN. Это может быть любой оператор языка БЕЙСИК.

Если выражение ложно, то программа переходит к выполнению следующей строки.

Оцениваемое выражение может быть переменной или формулой, причем оно считается истинным, если оно не равно нулю, и ложным, если равно нулю. В большинстве случаев применяется выражение с операторами отношения (=, <, >, <=, >=, <>, AND (И), OR (ИЛИ), NOT (НЕ)).

```
10 IF X > 10 THEN END
```

INPUT

Оператор INPUT позволяет программе брать данные от пользо-

вателя, присваивая эти данные переменной. Программа останавливается, печатает вопросительный знак (?) на экране и ожидает, пока пользователь введет с клавиатуры ответ и нажмет клавишу RETURN.

За оператором INPUT следует имя переменной или список имен переменных, разделенных запятыми. Сообщение может быть заключено в кавычки, перед списком имен переменных, подлежащих ВВОДУ (INPUT). Если должно быть введено несколько переменных, при наборе с клавиатуры они должны разделяться запятыми.

```
1     10 INPUT "PLEASE ENTER YOUR FIRST NEME";A$
2     20 PRINT "ENTER YOUR CODE NUMBER";:INPUT B
```

Рис. 65

1 - ..."введите свое имя"...; 2 - ..."введите ваше кодовое число"...

INPUT #

Этот оператор аналогичен оператору INPUT, но он берет данные из предварительно открытых (OPEN) файла или устройства.

```
10 INPUT#1,A
```

LET

Этот оператор вряд ли когда-либо применяется в программах, так как он не относится к числу обязательных, однако этот оператор является ядром всех программ на языке БЕЙСИК. Имя переменной, которой должен присваиваться результат вычислений, находится с левой стороны от знака равенства, а формула - с правой.

```
10 LET A = 5
20 LET D$ = "HELLO"
```

Рис. 66

NEXT (ЗАТЕМ)

Оператор NEXT всегда употребляется в сочетании с оператором FOR. Когда программа достигает оператора NEXT, она проверяет оператор FOR на предмет того, не достигнут ли предел цикла. Если цикл не окончен, то переменная цикла увеличивается на величину, обусловленную оператором STEP (ШАГ). Если цикл окончен, то выполняется оператор, следующий за оператором NEXT.

За оператором NEXT может идти имя переменной или список имен переменных, отделенных друг от друга запятыми. Если в списке нет имен, то начавшийся цикл будет и последним. Если даются переменные, они берутся слева направо.

```
10 FOR X = 1 TO 100: NEXT
```

ON

Эта команда превращает команды GOTO и GOSUB в специфичные варианты оператора IF (ЕСЛИ). За оператором ON следует формула, которая подлежит оценке. Если результат вычисления равен 1, то выполняется первая строка в списке; если результат равен 2, выполняется вторая строка и так далее. Если результат равен нулю, отрицателен или больше чем список номеров, следующей строкой, подлежащей выполнению, будет оператор, следующий за оператором ON.

```
10 INPUT X
20 ON X GOTO 10,20,30,40,50
```

OPEN

Оператор OPEN обеспечивает компьютеру "Коммодор-64" доступ к таким устройствам, как накопители на магнитной ленте и диске, для работы с данными, к принтеру или даже к экрану. За оператором OPEN идет число (от 0 до 255), которое является числом, к которому будут относиться все последующие операторы. После первого числа обычно следует второе, которое является номером устройства.

Номера устройств таковы:

```
0 - экран
1 - накопитель на кассетной магнитной ленте
4 - принтер
8 - накопитель на магнитном диске
```

За номером устройства может следовать третье число, отделенное запятой. Это число означает вторичный адрес. В случае накопителя на магнитной ленте 0 относится к считыванию, 1 - к записи и 2 - к записи с маркером окончания ленты.

В случае работы с накопителем на диске это число относится к номеру буфера или канала. Для принтера вторичный адрес управляет такими функциями, как расширенная печать. Более подробные данные приведены в "Справочнике для программиста компьютера "Коммодор-64" (Sommodore 64 Programmer's Reference Manual).

```
10 OPEN 1,0       ОТКРЫВАЕТ ЭКРАН как устройство
20 OPEN 2,8,8,"D" ОТКРЫВАЕТ кассетный накопитель для считывания, отыскиваемый файл -D
30 OPEN 3,4       ОТКРЫВАЕТ принтер
40 OPEN 4,8,15    ОТКРЫВАЕТ канал данных на диске
```

Смотрите также: CLOSE, SMD, GET#, INPUT# и PRINT#, системную переменную ST и Приложение Б.

POKE

За оператором POKE всегда следуют два числа или формулы.

Первое число указывает на ячекйку памяти, второе число - это десятичное число от Ø до 255, которое будет помещено в указанную ячейку памяти, заменяя ранее хранившуюся в этой ячейке величину.

```
1Ø POKE 53281,Ø
20 S = 4Ø96 13
33 POKE S + 29,B
```

PRINT

Оператор PRINT является для большинства людей первым оператором, который они осваивают, однако имеется ряд случаев, о которых необходимо знать. Оператор PRINT может иметь после себя:

Текстовую цепочку с кавычками
Имена переменных
Функции
Знаки пунктуации

Знаки пунктуации используются для форматирования данных на экране. Запятая делит экран на четыре колонки, в то время как точка с запятой ликвидирует все пробелы. Любой знак пунктуации может быть в строке последним символом. Это приводит к тому, что следующая величина, выводимая на печать, имеет вид продолжения того же оператора PRINT.

```
1Ø PRINT "HELLO"
2Ø PRINT "HELLO",A
3Ø PRINT A+B

 4Ø PRINT J;
 6Ø PRINT A,B,C,D
```

Смотрите также функции POS, SPC и TAB.

PRINT#

Имеется несколько отличий этого оператора от оператора PRINT. За оператором PRINT# следует число, которое относится к устройству или файлу данных, которые были предварительно ОТКРЫТЫ (OPEN). За этим числом следует запятая и список, подлежащий выводу на печать. Запятая и точка с запятой имеют то же действие, как и при операторе PRINT. Следует отметить, что некоторые устройства не могут работать с функциями TAB и SPC.

```
1ØØ PRINT#1,"DATA VALUES"; A%, B1, C
```

READ

Оператор READ используется для присвоения переменным ин-

формации от операторов DATA, так что информация может использоваться . Следует проявлять осторожность и избегать случаев считывания цепочек там, где оператор READ ожидает числа, что привело бы к появлению сообщения TYPE MISMATCH ERROR (ОШИБКА ПРИ НАБОРЕ С КЛАВИАТУРЫ).

REM

Оператор REM является примечанием для читающего список (LIST) программы. Он может пояснять раздел программы или давать дополнительные команды. Операторы REM ни в коей мере не затрагивают выполнения программы, кроме того, что они несколько увеличивают ее длину. За оператором REM может следовать любой текст.

RESTORE

При выполнении программы указатель, указывающий, куда будет считываться сейчас информация в операторе DATA, устанавливается на первое наименование в списке. Это дает Вам возможность повторного прочтения информации. В строке оператор RESTORE стоит один.

RETURN

Этот оператор всегда используется в сочетании с оператором GOSUB. Когда в программе встречается оператор RETURN, она переходит к оператору, стоящему непосредственно после команды GOSUB. Если предварительно не был введен оператор GOSUB, то появляется сообщение RETURN WITHOUT GOSUB ERROR (ОШИБКА - ОПЕРАТОР RETURN БЕЗ ОПЕРАТОРА GOSUB).

STOP

Этот оператор останавливает выполнение программы. На экране появляется сообщение BREAK xxx (ВЫПОЛНЕНИЕ ПРЕРВАНО), причем xxx представляет собой номер строки, содержащей оператор STOP. Программа может быть запущена вновь с помощью команды CONT. Оператор STOP обычно применяется при отладке программы.

SYS

За оператором SYS идет десятичное число или числовая величина в диапазоне от Ø до 65535. Затем программа начинает выполнять программу на машинном языке, начиная с указанной этим числом ячейки памяти. Это аналогично функции USR, но не позволяет осуществить передачу параметра.

WAIT

Оператор WAIT используется для останова программы, пока содержимое ячейки памяти не изменится соответствующим образом.

За оператором WAIT идет ячейка памяти (X) и до двух переменных. Формат имеет вид:

WAIT X,Y,Z

Содержимое ячейки памяти сначала складываются по модулю 2 с третьим числом, если оно присутствует, и затем с результатом сложения и вторым числом осуществляется операция логического сложения. Если результат равен нулю, то программа возвращается к этой ячейке памяти и проверка осуществляется снова. Если результат не равен нулю, то программа продолжается со следующего оператора.

Числовые функции

ABS(X) (абсолютная величина)

ABS дает абсолютную величину числа без его знака (+ или -). Результат всегда положителен.

ATN(X) (арктангенс)

Результатом является угол в радианах, тангенс которого равен X.

COS(X) (косинус)

Результатом является величина косинуса угла X, где X - величина угла в радианах.

EXP(X)

Результатом является математическая константа e (2,71828183), возведенная в степень X.

FNxx(X)

Результатом является величина функции xx, определенной пользователем в операторе DEF FNxx(X).

INT(X)

Результатом является усеченная величина X, где все десятичные знаки справа от десятичной запятой (точки) отбрасываются. Результат всегда будет меньше или равен X. Таким образом, любые отрицательные числа с десятичными знаками после запятой станут целыми числами, меньшими своей действительной величины.

LOG(X) (логарифм)

Результатом будет натуральный логарифм X, т.е. логарифм по

основанию e (см. EXP(X)). Чтобы перевести эту величину в логарифм по основанию 10, необходимо разделить эту величину на LOG(10).

PEEK(X)

Используется для нахождения содержимого ячейки памяти X в диапазоне от 0 до 65535. Результат будет заключаться в диапазоне от 0 до 255. Функция PEEK часто используется в сочетании с оператором POKE.

RND(X) (случайное число)

Функция RND(X) выдает случайное число в диапазоне от 0 до 1. Первое случайное число должно вырабатываться с помощью формулы RND(-TI) для запуска каждый раз по-разному. После этого X должен быть единицей или любым положительным числом. Если X брать равным 0, то результатом будет то же случайное число, которое было выработано непосредственно перед этим.

Отрицательная величина X введет в генератор новое начальное число. Использование одного и того же отрицательного числа для X даст в результате ту же последовательность "случайных" чисел.

Формула для генерирования случайных чисел в диапазоне между X и Y имеет вид:

N = RND(1)*(Y-X)+X

где Y - верхний предел, X - нижний предел желаемых чисел.

SGN(X) (знак)

Функция выдает знак (положительный, отрицательный или нуль) числа X. Результат будет равен +1, если X больше нуля, 0, если X равен нулю и -1, если X меньше нуля.

SIN(X) (синус)

SIN(X) является тригонометрической функцией синус. Результатом будет синус X, где X - угол в радианах.

SQR(X) (квадратный корень)

Эта функция дает результатом квадратный корень из X, где X - положительное число или нуль. Если X - отрицательное число, возникает сообщение ILLECAL QUANTITY ERROR (ОШИБКА - НЕПРАВИЛЬНАЯ ВЕЛИЧИНА).

TAN(X) (тангенс)

Результатом будет тангенс X, где X - угол в радианах.

USR(X)

Когда используется эта функция, программа переходит к программе на машинном языке, начальная точка которой содержится в ячейках памяти. Параметр X передается в программу на машинном языке, которая возвращает в программу на языке БЕЙСИК другую величину. Более подробно об этой функции и программировании на машинном языке изложено в "Справочнике для программиста компьютера "Коммодор-64" (Commodore 64 Programmer's Reference Manual).

Цепочечные функции

ASC(X$) или ASC("A")

Эта функция указывает код ASCII первого знака последовательности (цепочки) X$.

CHR$(X)

Эта функция является обратной по отношению к функции ASC и выдает знак последовательности, код ASCII которого равен X.

LEFT$(X$,X)

Выдает цепочку, содержащую X знаков, стоящих слева в цепочке X$.

LEN(X$)

Результатом является количество знаков (включая пробелы и другие символы) в цепочке X.

MID$(X$,S,X)

Результатом является цепочка, содержащая X знаков, начиная с S-го знака в X.
Если X опущено, тогда результатом будут все знаки начиная с S-го знака до конца цепочки.

RIGHT$(X$,X)

Выдает цепочку, содержащую X знаков, стоящих справа в цепочке X$.

STR$(X)

Результатом будет цепочка, идентичная выведенному на печать варианту X.

VAL(XØ)

Эта функция преобразует XØ в число и является обратной операцией для STRØ. Цепочка проверяется от крайнего левого знака вправо на такое количество знаков, какое имеется в опознаваемом формате числа.

```
1Ø X = VAL("123.456")          X = 123.456
1Ø X = VAL("12A13B")           X = 12
1Ø X = VAL("RIUØ17")           X = Ø
1Ø X = VAL("-1.23.45.67")      X = -1.23
AS = "1234":X = VAL(AS):X = 1234
```

Другие функции

FRE(X)

Эта функция выдает количество неиспользуемых байтов, которые имеются в настоящий момент в памяти независимо от величины X.

POS(X)

Эта функция дает номер колонки (от Ø до 39), в которой следующий оператор PRINT появится на экране. X может иметь любую величину и не используется.

SPC(X)

Эта функция используется в операторе PRINT для перескока на X пробелов вперед.

TAB(X)

Функция TAB также используется в операторе PRINT; следующий элемент, подлежащий ВЫВОДУ НА ПЕЧАТЬ (PRINT), будет находиться в колонке X, если только текущее положение вывода на печать уже не прошло эту точку.

Приложение Г.

Сокращения ключевых слов в языке БЕЙСИК

Для экономии времени при наборе на клавиатуре программ и команд язык БЕЙСИК, применяющийся в компьютере "Коммодор-64", позволяет пользователю сокращать большинство ключевых слов. Так, сокращением для оператора PRINT (ВЫВЕСТИ НА ПЕЧАТЬ) служит знак вопроса. Сокращения других слов осуществляются набором на клавиатуре одной или двух первых букв слова, после чего нажимается клавиша SHIFT (СДВИГ) и набирается следующая

буква слова. Если сокращения используются в строке программы, ключевое слово будет внесено в СПИСОК (LIST) в полной форме. Обратите внимание, что некоторые ключевые слова, будучи сокращенными, включают в себя левую скобку.

1 Command	2 Abbrevlation	3 Looks like this on screen
ABS	A SHIFT B	A
AND	A SHIFT N	A
ASC	A SHIFT S	A
ATN	A SHIFT T	A
CHR$	C SHIFT H	C
CLOSE	CL SHIFT O	CL
CLR	C SHIFT L	C
CMD	C SHIFT M	C
CONT	C SHIFT O	C
COS	NONE	COS
DATA	D SHIFT A	D
DEF	D SHIFT E	D
DIM	D SHIFT I	D
END	E SHIFT N	E
EXP	E SHIFT X	E
FN	NONE	FN
FOR	F SHIFT O	F
FRE	F SHIFT R	F
GET	G SHIFT E	G
GET#	NONE	GET#
GOSUB	GO SHIFT S	GO
GOTO	G SHIFT O	G
IF	NONE	IF

1 Command	2 Abbrevlation	3 Looks like this on screen
INPUT	NONE	INPUT
INPUT#	I SHIFT N	I
INT	NONE	INT
LEFTS	LE SHIFT F	LE
LEN	NONE	LEN
LET	L SHIFT E	L
LIST	L SHIFT I	L
LOAD	L SHIFT O	L
LOG	NONE	LOG
MIDS	M SHIFT I	M
NEXT	N SHIFT E	N
NOT	N SHIFT O	N
ON	NONE	ON
OPEN	O SHIFT P	O
OR	NONE	OR
PEEK	P SHIFT E	P
POKE	P SHIFT O	P
POS	NONE	POS
PRINT	?	?
PRINT#	P SHIFT R	P
READ	R SHIFT E	R
REM	NONE	REM
RESTORE	RE SHIFT S	RE
RETURN	RE SHIFT T	RE
RIGHT¤	R SHIFT I	R

1 Command	2 Abbreviation	3 Locks like this on screen
RND	R SHIFT N	R
RUN	R SHIFT U	R
SAVE	S SHIFT A	S
SGN	S SHIFT G	S
SIN	S SHIFT I	S
SPC(	S SHIFT P	S
SQR	S SHIFT Q	S
STATUS	ST	ST
STEP	ST SHIFT E	ST
STOP	S SHIFT T	S
STR¤	ST SHIFT R	ST
SVS	S SHIFT Y	S
TAB(	T SHIFT A	T
TAN	NONE	TAN
THEN	T SHIFT H	T
TIME	TI	TI
TIME¤	TI¤	TI¤
USR	U SHIFT S	U
VAL	V SHIFT A	V
VERIFY	V SHIFT E	V
WAIT	W SHIFT A	W

Рис.67

1 - команда; 2 - сокращение; 3 - как это выглядит на экране

Приложение Д

Коды экранного дисплея

В приведенной ниже таблице показаны все знаки компьютера "Коммодор-64" и соответствующие им коды. В этой таблице показано, какое число следует ВСТАВИТЬ (POKE) в экранную память (ячейки 1Ø24-2Ø23) для того чтобы получить нужный знак. Показано также, какой знак соответствует числу, снятому (PEEK) с экрана.

Имеются два набора знаков, однако за один раз устанавливается только один набор. Это означает, что Вы не можете иметь на экране какой-либо знак одного набора, если на экране уже имеется знак другого набора. Наборы переключаются с помощью одновременного нажатия клавиши SHIFT и клавиши компьютера "Коммодор".

В языке БЕЙСИК команда POKE 53272,29 осуществит переключение на верхний регистр, а команда POKE 53272,31 - на нижний регистр.

Любое число в этой таблице может быть также выведено на дисплей в реверсированном, или обратном, (REVERSE) виде. Обратный код знака может быть получен прибавлением 128 к указанным в таблице величинам.

Если Вы хотите вывести на дисплей сплошной круг в ячейке 15Ø4, ВСТАВЬТЕ (POKE) код для круга (81) в ячейку 15Ø4: POKE

1504,81.

Для управления цветом каждого знака, выведенного на дисплей, существует соответствующая ячейка памяти (ячейки 55296-56295). Для изменения цвета круга на желтый (код цвета равен 7) Вы должны ВСТАВИТЬ в соответствующую ячейку памяти (55776) код цвета знака: POKE 55776,7.

Полные карты памяти экрана и цвета вместе с кодами цвета приведены в Приложении Ж.

Таблица 7

Экранные коды

1 SET 1	2 SET 2	3 POKE	1 SET 1	2 SET 2	3 POKE	1 SET 1	2 SET 2	3 POKE
		0	S	s	19	&		38
A	a	1	T	t	20	'		39
B	b	2	U	u	21	(		40
C	c	3	V	v	22	)		41
D	d	4	W	w	23	*		42
E	e	5	X	x	24	+		43
F	f	6	Y	y	25	,		44
G	g	7	Z	z	26	-		45
H	h	8			27	.		46
I	i	9			28	/		47
J	j	10			29	0		48
K	k	11			30	1		49
L	l	12	<-		31	2		50
M	m	13	SPACE		32	3		51
N	n	14	!		33	4		52
O	o	15	"		34	5		53
P	p	16	#		35	6		54
Q	q	17	¤		36	7		55
R	r	18	%		37	8		56

Таблица 7 (продолжение)

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
9		57		Q	81			105
:		58		R	82			106
;		59		S	83			107
<		60		T	84			108
=		61		U	85			109
>		62		V	86			110
?		63		W	87			111
		64		X	88			112
	A	65		Y	89			113
	B	66		Z	90			114
	C	67			91			115
	D	68			92			116
	E	69			93			117
	F	70			94			118
	G	71			95			119
	H	72	SPACE		96			120
	I	73			97			121
	J	74			98			122
	K	75			99			123
	L	76			100			124
	M	77			101			125
	N	78			102			126
	O	79			103			127
	P	80			104			

1 - набор 1; 2 - набор 2; 3 - код POKE
Коды с 128-255 являются обратными величинами кодов 0-127.

Приложение E

Коды ASCII и CHR$

В этом Приложении показано, какие знаки будут появляться, если Вы будете осуществлять реализацию оператора PRINT в отношении кодов CHR$(X) для всех возможных величин X. В Приложении также показано, какие величины Вы будете получать, набирая на клавиатуре PRINT ASC("x"), где x - любой знак клавиатуры. Это бывает полезным при оценке знака, полученного при реализации оператора GET, переходе с верхнего регистра на нижний и печатании команд, основанных на знаках (как, например, переключение регистров), которые не могут быть заключены в кавычки.

Таблица 8

1	2	1	2	1	2
PRINTS	CHR$	PRINTS	CHR$	PRINT	CHR$
	0		15	CRN	30
	1		16	BLU	31
	2	CRSH	17	SPACE	32
	3	RVS ON	18	!	33
	4	CLR HOME	19	"	34
WHT	5	INST DEL	20	#	35
	6		21	$	36
	7		22	%	37
DISABLE SHIFT C=	8		23	&	38
ENABLES SHIFT C=	9		24	.	39
	10		25	(	40
	11		26	)	41
	12		27	*	42
RETURN	13	RED	28	+	43
SWITCH TO LOWER CASE	14	CRSR -->	29	,	44

Таблица 8 (продолжение)

1 PRINTS	2 CHR¤	1 PRINT	2 CHR¤	1 PRINT	2 CHR¤
-	45	D	68	[	91
.	46	E	69		92
/	47	F	70	]	93
Ø	48	G	71		94
1	49	H	72	<--	95
2	50	I	73		96
3	51	J	74		97
4	52	K	75		98
5	53	L	76		99
6	54	M	77		100
7	55	N	78		101
8	56	O	79		102
9	57	P	80		103
:	58	Q	81		104
;	59	R	82		105
<	60	S	83		106
=	61	T	84		107
>	62	U	85		108
?	63	V	86		109
	64	W	87		110
A	65	X	88		111
B	66	Y	89		112
C	67	Z	90		113

Таблица 8 (продолжение)

1	2	1	2	1	2
PRINT	CHR¤	PRINT	CHR¤	PRINT	CHR¤
	114	14	138		162
	115	16	139		163
	116	18	140		164
	117	SHIFT RETURN	141		165
	118	SWITCH TO UPPER CASE	142		166
	119		143		167
	120	BI¤	144		168
	121	CRSR	145		169
	122	RVS OFF	146		170
	123	CLR HOME	147		171
	124	INST DEl	148		172
	125	Brown	149		173
	126	Lt.Red	150		174
	127	Grey 1	151		175
	128	Grey 2	152		176
Orange	129	Lt.Green	153		177
	130	Lt.Blue	154		178
	131	Grey 3	155		179
	132	PUR	156		181
11	133	<-- CRSR	157		182
13	134	YEL	158		183
15	135	CYN	159		184
17	136	SPACE	160		185
12	137		161		186

Таблица 8 (окончание)

1 PRINT	2 CHR$	1 PRINT	2 CHR$
	187		190
	188		191
	189		

CODES	192-223	SAME AS	96-127
CODES	224-254	SAME AS	160-190
CODE	255	SAME AS	126

1 - символы, выводимые на печать: 2 - код CHR$; 3 - коды 192-223 те же, что и 96-127: 4 - коды 224-254 те же, что и 160-190: 5 - код 255 тот же, что и 126

Приложение Ж

Карты экранной памяти и памяти цвета

В приводимых ниже таблицах показано, какие ячейки памяти управляют помещением знаков на экране и какие ячейки памяти используются для изменения индивидуальных цветов знаков. Кроме того, показаны коды цвета знаков.

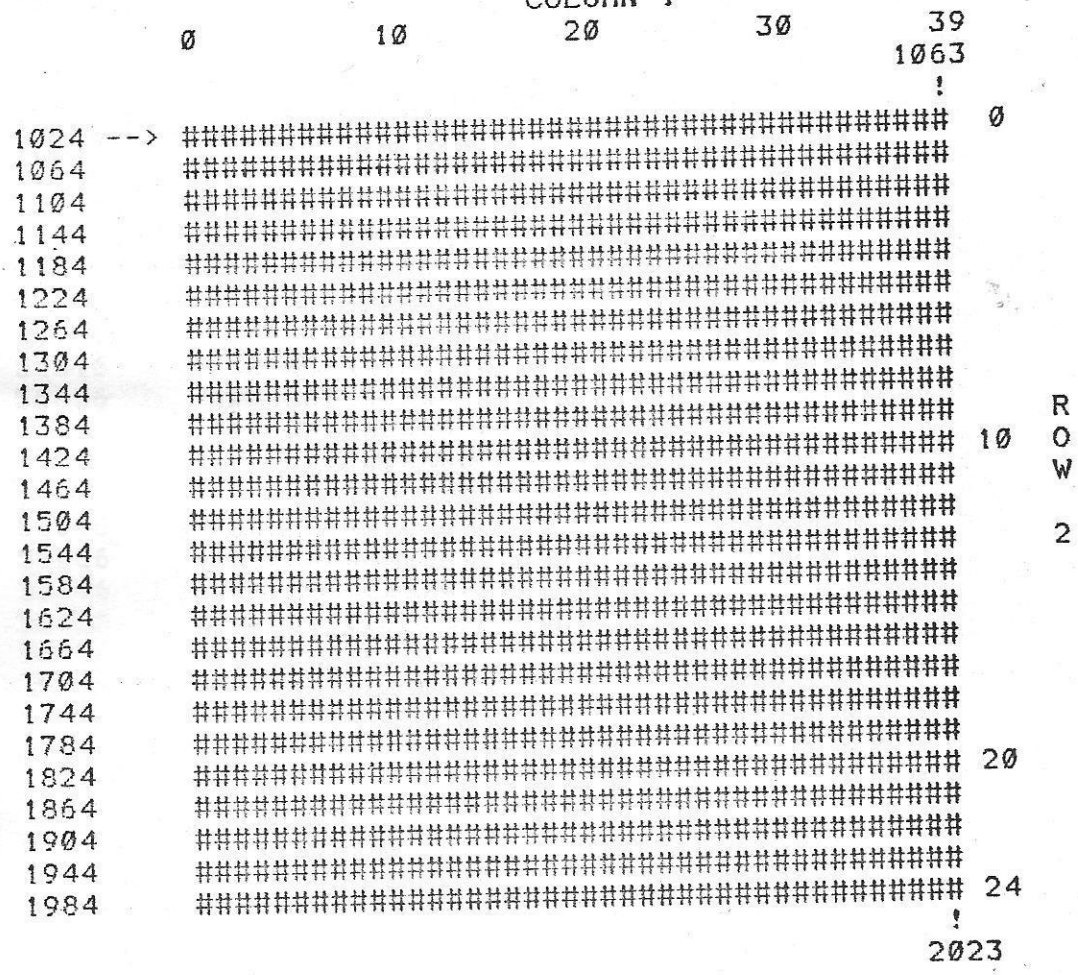


Рис. 68 Карта экранной памяти:

1 - колонка; 2 - ряд

Реальные величины, которые следует ВСТАВИТЬ (POKE) в ячейку памяти цвета для изменения цвета знака, таковы:

Ø - черный, 1 - белый, 2 - красный, 3 - сине-зеленый, 4 - пурпурный, 5 - зеленый, 6 - синий, 7 - желтый, 8 - оранжевый, 9 - коричневый, 1Ø - светлокрасный, 11 - серый 1, 12 - серый 2, 13 - светлозеленый, 14 - голубой, 15 - серый 3.

Например, для изменения цвета знака, находящегося в левом верхнем углу экрана на красный, необходимо набрать на клавиатуре: POKE 55296,2.

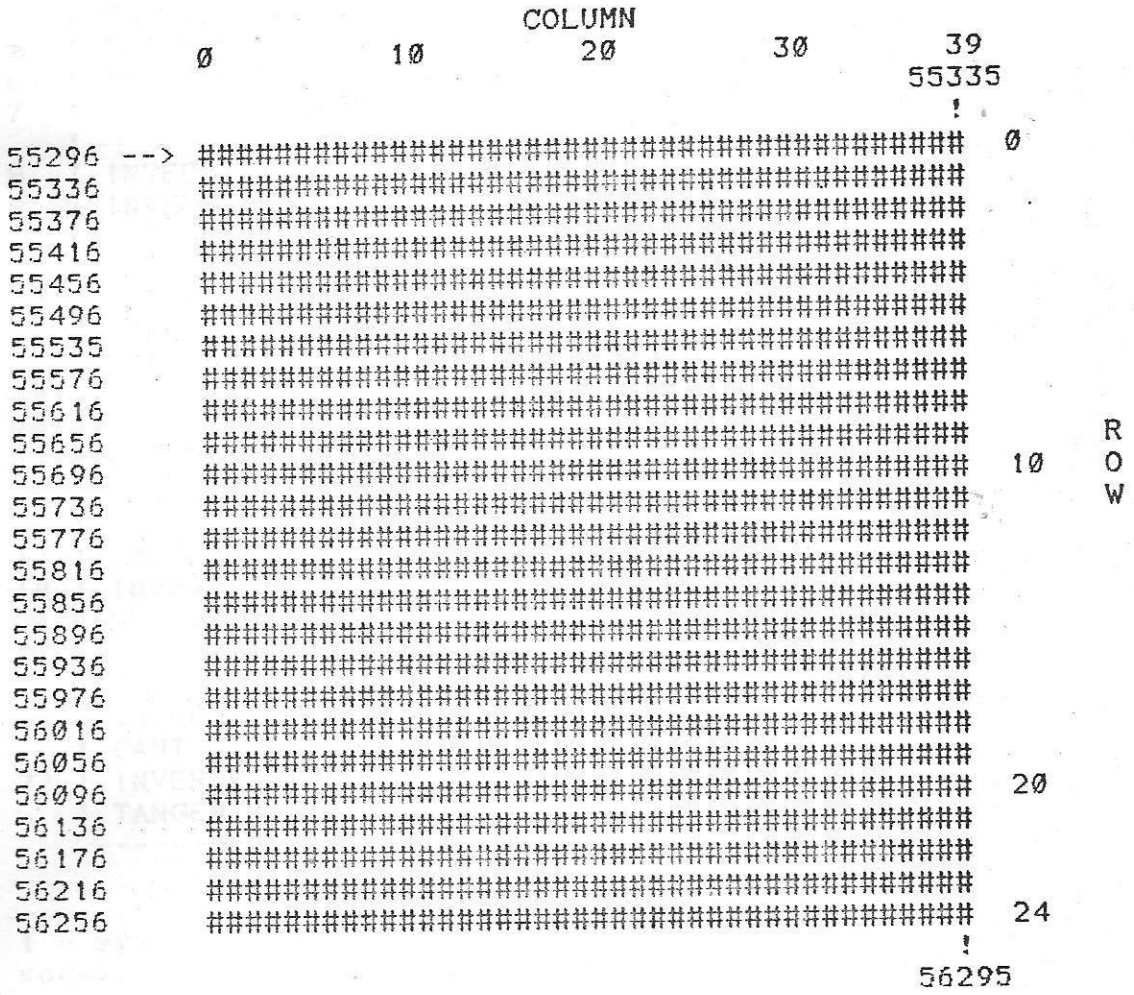


Рис.69: Карта памяти цвета:

1 - колонка; 2 - ряд

Приложение 3

Вычисление математической функции

Функции, реализация которых не заложена в компьютер "Коммодор-64", .могут быть вычислены следующим образом.

Таблица 9

	FUNCTION	BASIC EQUIVALENT
	SEC(X)=1/COS(X)	
3	SECANT	SEC(X)=1/COS(X)
4	COSECANT	CSC(X)=1/SIN(X)
5	COTANGENT	COT(X)=1/TAN(X)
6	INVERSE SINE	ARCSIN(X)=ATN(X/SQR(-X*X+1))
7	INVERSE COSINE	ARCCOS(X)=-ATN(X)SQR (-X*X+1))+ /2
8	INVERSE SECANT	ARCSEC(X)=ATN(X/SQR(X*X-1))
9	INVERSE COSECANT	ARCCSC(X)=ATN(X/SQR(X*X-1)) +(SGN(X)-1*/2
10	INVERSE COTANGENT	ARCOT(X)=ATN(X)+/2
11	HYPERBOLIC SINE	SINH(X)=(EXP(X)-EXP(-X))/2
12	HYPERBOLIC COSINE	COSH(X)=(EXP(X)+EXP(-X))/2
13	HYPERBOLIC TANGENT	TANH(X)=EXP(-X)/(EXP(x)+EXP (-X))*2+1
14	HYPERBOLIC SECANT	SECH(X)=2/(EXP(X)+EXP(-X))
15	HYPERBOLIC COSECANT	CSCH(X)=2/(EXP(X)-EXP(-X))
16	HYPERBOLIC COTANGENT	COTH(X)=EXP(-X)/EXP(X) -EXP(-X))*2+1
17	INVERSE HYPERBOLIC SINE	APCSINH(X)=LOG(X+SQR(X*X+1))
18	INVERSE HYPERBOLIC COSINE	ARCCOSH(X)=LOG(X+SQR(X*X-1))
19	INVERSE HYPERBOLIC TANGENT	ARCTANH(X)=LOG((1+X)/(1-X))/2
20	INVERSE HYPERBOLIC SECANT	ARCSECH(X)=LOG((SQR (-X*X+1)+1/X)
21	INVERSE HYPERBOLIC COSECANT	ARCCSCH(X)=LOG((SGN(X)*SQR (X*X+1/x)
22	INVERSE HYPERBOLIC COTANGENT	ARCCOTH(X)=LOG((X+1)/(X-1))/2

1 - функция; 2 - эквивалент в языке БЕЙСИК; 3 - секанс; 4 - косеканс; 5 - котангенс; 6 - арксинус; 7 - арккосинус; 8 - арксеканс; 9 - арккосеканс; 10 - арккотангенс; 11 - гиперболический синус; 12 - гиперболический косинус; 13 - гиперболический тангенс; 14 - гиперболический секанс; 15 - гиперболический косеканс; 16 - гиперболический котангенс; 17 - гиперболический арксинус; 18 - гиперболический арккосинус; 19 - гиперболический арктангенс; 20 - гиперболический арксеканс; 21 - гиперболический арккосеканс; 22 - гиперболический арккотангенс

Приложение И

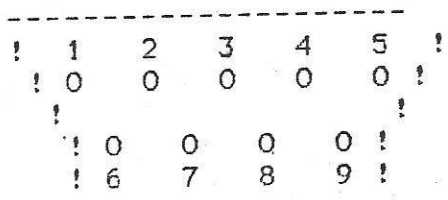
Разводка выводов для устройств входа/выхода

В этом Приложении показано, какие подключения могут быть сделаны к компьютеру "Коммодор-64".

1. Игровой вход/выход
2. Ниша для специальной кассеты
3. Звуковой/видео
4. Последовательный вход/выход (диск/принтер)
5. Выход модулятора
6. Кассетный вход/выход
7. Порт пользователя

1 Control Port 1

2 Pin	3 Type	4 Note
1	JOYA Ø	
2	JOYA 1	
3	JOYA 2	
4	JOYA 3	
5	POT AY 8	
6	BUTTON A/LP	
7	5 +5V	MAX. 1ØØmA
8	GND 7	6
9	9 POT AX	



13
Control Port 2

2 Pin	3 Type	4 Note
1	JOYBØ	
2	JOYB1	
3	JOYB2	
4	JOYB3	
5	POT BY 1Ø	
6	BUTTON B 11	
7	+5V	MAX. 1ØØmA
8	GND 7	6
9	POT BX 12	

Рис. 7Ø

1 - контрольный порт 1; 2 - вывод; 3 - тип; 4 - примечание; 5 - кнопка A/LP; 6 - максимум 1ØØ мА; 7 - земля; 8 - потенциал AY; 9 - потенциал AX; 1Ø - потенциал BY; 11 - кнопка B; 12 - потенциал BX; 13 - контрольный порт 2

2	3
Pin	Type
22	GND 1
21	CDØ
2Ø	CD1
19	CD2
18	CD3
17	CD4
16	CD5
15	CD6
14	CD7

13	DMA
12	BA

2	3
Pin	Type

11	ROML
1Ø	1/Ø2
9	-----
	EXROM

8	GAME 4
7	1/Ø1
6	Dot Clock 5

5	CR/W

4	IRQ
3	+5V
2	+5V
1	GND 1

Pin	Type
Z	GND
Y	CAØ
X	CA1
W	CA2
V	CA3
U	CA4
T	CA5
S	CA6
R	CA7
P	CA8
N	CA9

Pin	Type
M	CA1Ø
L	CA11
K	CA12
J	CA13
H	CA14
F	CA15
E	SO2

D	NMI

C	RESET 6

B	ROMN
A	GND 1

1	2	3	4	5	6	7	8	9	1Ø	11	12	13	14	15	16	17	18	19	2Ø	21	22
=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=
=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=
A	B	C	D	E	F	H	J	K	L	M	N	P	R	S	T	U	V	W	X	Y	Z

Рис.71. Разъем для подключения специального кассетного устройства:
1 - земля; 2 - вывод; 3 - тип; 4 - игра; 5 - тактовая частота изображения точки; 6 - сброс

Pin	Type		Note
1	LUMINANCE	4	
2	GND	5	
3	AUDIO OUT	6	
4	VIDEO OUT	7	
5	AUDIO IN	8	

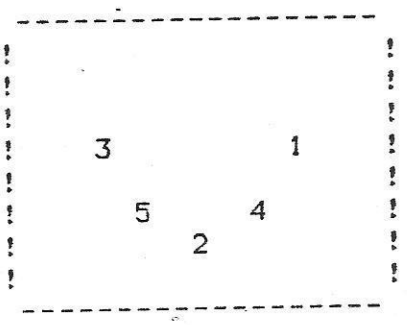


Рис.72. Разъем звукового/видеосигнала:
1 - вывод; 2 - тип; 3 - примечание; 4 - яркость; 5 - земля; 6 - выход звукового сигнала; 7 - выход видеосигнала; 8 - вход звукового сигнала

Pin	Type	
1	SERIAL SRQIN	3
2	GND	4
3	SERIAL ATN IN/OUT	5
4	SERIAL CLK IN/OUT	6
5	SERIAL DATA IN/OUT	7

6	RESET	

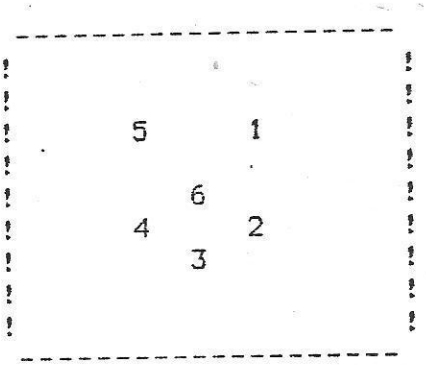


Рис.73. Последовательный вход/выход:
1 - вывод; 2 - тип; 3 - последовательный SRQIN; 4 - земля; 5 - вход/выход последовательного сигнала ATN; 6 - вход/выход последовательного сигнала тактовой частоты; 7 - вход-выход последовательного сигнала данных; 8 - СБРОС

Pin	Type	
A-1	GND	3
B-2	+5V	
C-3	CASSETTE MOTOR	4
D-4	CASSETTE READ	5
E-5	CASSETTE WRITE	6
F-6	CASSETTE SENSE	7

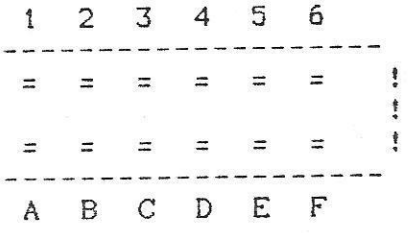


Рис.74. Разъем для подключения кассетного накопителя:
1 - вывод; 2 - тип; 3 - земля; 4 - двигатель кассетного накопителя; 5 - сигнал считывания с кассетного накопителя; 6 - сигнал записи на кассетный накопитель; 7 - опознавательный сигнал

1	2	3
Pin	Tipe	Note
1	GND 4	
2	+5V	MAX. 100 mA
	------	5
3	RESET 6	
4	CNT1 7	
5	SP1	
6	CNT2 8	
7	SP2	

8	PC2	
9	SER:ATN IN 9	5
10	9 VAC 11	MAX.100 mA
11	9 VAC 11	MAX.100 mA
12	GND 4	5

1	2	3
Pin	Type	Note
A	GND 4	

B	FLAG2 10	
C	PB0	
D	PB1	
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	
N	GND	

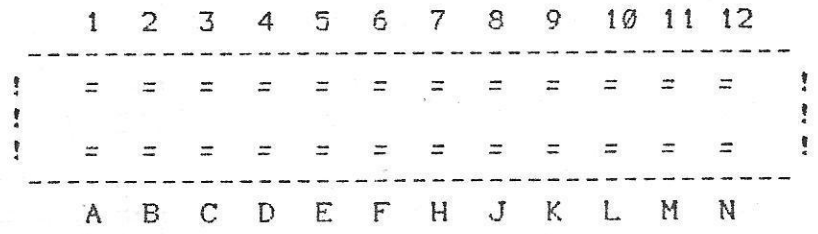


Рис.75. Вход/выход пользователя:
1 - вывод; 2 - тип; 3 - примечание; 4 - земля; 5 - максимум 100 мА; 6 - СБРОС; 7 - СЧЕТ 1; 8 - СЧЕТ 2; 9 - вход последовательного сигнала ATN; 10 - ФЛАГ 2; 11 - 9 вольт переменного тока

Приложение К

Программы для тренировки

В это Приложение мы включили несколько полезных программ для тренировки на компьютере "Коммодор-64". Помимо того что эти программы весьма полезны, они также довольно интересны.

```
1    100 print"<CRDN> jotto          jim butterfield"
2    120 input"<CRDN> want instructions";z¤:ifasc(z¤)=78goto250
3    130 print"<CRDN> try to guess the mystery 5-letter word"
4    140 print"<CRDN> you must guess only legal 5-letter"
5    150 print"words, too..."
6    160 print "you will be told the number of matches"
7    170 print"(or 'jots') of your guess."
8    180 print"<CRDN> hint: the trick is to vary slightly"
9    190 print" from one guess to the next; so that"
10   200 print" if you guess 'batch' and get 2 jots"
11   210 print" you might try 'botch' or 'chart'"
12   220 print" for the next guess..."
     250 data bxbsf,ipccz,dbdif,esfbe,pggbm
     260 data hpshf,ibudi,djwjm,kpmmz,lbzbl
     270 data sbkbi,mfwfm,njnjd,boofy,qjqfs
     280 data rvftu,sjwfs,qsftt,puufs,fwfou
     290 data xfbwf,fyupm,nvtiz,afcsb,qjaaz
     300 data uijdt,esvol,gmppe,ujhfs,qblfs
     310 data cppui,mzjoh,trvbu,hbvaf,pxjoh
     320 data uisff,tjhiu,bymft,hsvnq,bsfob
     330 data rvbsu,dsffg,cfmdi,qsftt,tqbsl
     340 data sbebs,svsdu,infmm,gspxo,esjgu
     400 n=50
     410 dim n¤(n),z(5),y(5)
     420 for j=1ton:readn¤(j):nextj
     430 t=ti
     440 t=t/1000:ift>=1thengoto440
     450 z=rnd(-t)
     500 g=0:n¤=n¤(rnd(1)*n+1)
13   510 print " i have a five letter word:":ifr>0goto560
14   520 print "guess (with legal words)"
15   530 print "and i'll tell you how many"
16   540 print "'jots', or matching letters,"
17   550 print "you have...."
18   560 g=g+1:input "your word";z¤
19   570 if.len(z¤)<>5thenprint"you must guess a
         5-letter word!":goto560
     580 v=0:h=0:m=0
     590 for j=1to5
     600 z=asc(mid¤(z¤,j,1)):y=asc(mid¤(n¤,j,1))-1:ify=64theny=
         90,
20   610 ifz<65orz>90thenprint"that's not a word!":goto560
     620 ifz=65orz=69orz=73orz=79orz=85orz=89thenv=v+1
     630 ifz=ythenm=m+1
```

```
    640 z(j)=z:y(j)=y:nextJ
    650 ifm=5goto800
21  660 ifv=0orv=5thenprint"come on...what kind of a
        word is that?":goto560
    670 for j=1to5:y=y(j)
    680 for k=1to5:ify=z(k)thenh=h+1:z(k)=0:goto700
    690 next k
    700 next j
    710 print "<CRUP><CRRT>...(9 раз)...<CRRT> ";H;"JOTS"
    720 ifg<30goto560
22  730 print "i'd.better tell you.. word was'";
    740 forj=1to5:printchr$(y(j));:nextj
    750 print"'":goto810
23  800 print "you got it in only";g;"guesses."
24  810 input " another word";z$
    820 r=1:lfasc(z$)<>78goto500
```

Рис.76

1 - "джот Джиму Баттерфилду"; 2 - ..."запрос команд..."; 3 - ..."постарайтесь угадать секретное слово из 5 букв"; 4 - ..."вы должны называть только разрешенные 5-буквенные; 5 - ..."слова, кроме того..."; 6 - ..."вам будет указано количество совпадений букв"; 7 - ...(или "джотов") вашей попытки; 8 - ..."подсказка: суть в том, чтобы незначительно изменять слово"; 9 - ..."от одной попытки к другой; так что"; 10 - ..."если вашей отгадкой было слово "batch" и вы получили два джота"; 11 - ..."вы можете дать слова "botch" или "chart"; 12 - ..."в качестве следующей отгадки секретного слова"; 13 - ..."у меня есть слово из 5 букв":"...; 14 - ..."угадай (из разрешенных слов)"; 15 - ..."и я скажу вам, сколько"; 16 - ..."джотов, или совпадающих букв"; 17 - ..."вы имеете..."; 18 - ..."ваше слово"...; 19 - ..."вы должны угадать слово из пяти букв"...; 20 - ..."это не слово..."; 21 - ..."ну, какое это слово?..."; 22 - ..."я лучше скажу вам...слово было"; 23 - ..."вы отгадали только за;g; угадываний"; 24 - "...другое слово"... .

```
1     1 rem *** sequence
      2 rem
2     3 rem *** from pet user group
3     4 rem *** software exchange
      5 rem *** po box 371
4     6 rem *** montgomeryville, pa 18936
      7 rem
      50 dim a$(26)
      100 z$="abcdefghijklmnopqrstuvwxyz"
      110 z1$="12345678901234567890123456"
5     200 print "<CLR><CRDN><CRDN> enter length of string
          to be sequenced "
6     220 input "maximum length is 26";s%
      230 if s%<1 or s%>26 then 200
      240 s=s%
      300 for i=1 to s
```

```
    310 a¤(1)=mid¤(z¤,i,1)
    320 next i
7   400 rem randomize string
    420 for i=1 to s
    430 k=int(rnd(1)*s+1)
    440 t¤=a¤(i)
    450 a¤(i)=a¤(k)
    460 a¤(k)=t¤
    470 next i
    480 gosub 950
    595 t=0
8   600 rem reverse substring
    605 t=t+1
9   610 input "hom many to reverse ";r%
    620 if r%=0 goto 900
    630 if r%>0 and r%<=s goto 650
10  640 print "must be between 1 and ":s: goto 610
    650 r=int(r%/2)
    660 for i=1 to r
    670 t¤=a¤(i)
    680 a¤(i)=a¤(r%-i+1)
    690 a¤(r%-i+1)=t¤
    700 next i
    750 gosub 950
    800 c=1: for i=2 to ¤
    810 if a¤(i)>a¤(i-1) goto 830
    820 c=0
    830 next i
    840 if c=0 goto 600
11  850 print "<CRDN> you did it in ":t;" tries"
12  900 rem check for another game
13  910 input "<CRDN> want to play again ";y¤
    920 if left¤ (y¤,1)="y" or y¤="ok" or y¤="i" goto 200
    930 end
    950 print
    960 print left¤(z1¤,s)
    970 for i=1 to s:print a¤(1);:next i
    980 print "<CRDN>"
    990 return

14  This program courlesy of Gene Deals
```

Рис.77

1 - упорядочение; 2 - ...от группы пользователей PET; 3 - обмен программным обеспечением; 4 - Монтгомерювиль, штат Пенсильвания 18936; 5 - ..."ввести длину цепочки, подлежащей упорядочению"; 6 - ..."максимальная длина - 26"...; 7 - "... рандомизируйте цепочку"; 8 - ..."реверсируйте субцепочку"; 9 - ..."сколько знаков следует реверсировать"...; 10 - ..."должно быть между 1 и"...; 11 - ..."вы осуществили это за ";t; попыток"; 12 - ... проверка для другой игры; 13 - ..."хотите сыграть еще раз"...; 14 - эта программа приведена с разрешения Джина Дилса

```
READY

1    90 REM PIANO KEYBOARD
   100 PRINT "<CLR><RUON><CRRT><CRRT><SH-B><CRRT><CRRT><CRRT>
             <SH-B><CRRT><CRRT><SH-B><CRRT><CRRT>"
   110 PRINT "<RUON><CRRT><CRRT><SH-B><CRRT><CRRT><CRRT>
             <SH-B><CRRT><CRRT><SH-B><CRRT><CRRT>"
   120 PRINT "<RUON><CRRT><CRRT><SH-B><CRRT><CRRT><CRRT>
             <SH-B><CRRT><CRRT><SH-B><CRRT><CRRT>"
   130 PRINT "<RUON><SH-B>...(12 раз)...<SH-B>"
   140 PRINT "<RUON>Q<SH-B>W<SH-B>E<SH-B>R<SH-B>T<SH-B>Y
       <SH-B>U<SH-B>I<SH-B>O<SH-B>P<SH-B>C<SH-B>*<SH-B>^"
2  150 PRINT "<CRDN>'SPACE' FOR SOLO OR POLYPHONIC"
3  160 PRINT "<CRDN>'F1, F3, F5, F7' OCTAVE SELECTION"
4  170 PRINT "<CRDN>'F2, F4, F6, F8' WAVEFORM Q"
5  180 PRINT "HANG ON, SETTING UP FREQUENCY TABLE..."
   190 S=13*4096+1024:DIMF<26>:DIMH<255>
   200 FORI=0TO28:POKES+I,0:NEXT
   210 F1=7040:FORI=1 TO26:F(27-1)=F1*5.8+30:F1=F1/2 <1/12>:
       NEXT
   220 K$="Q2W3EP5T6 Y7UI900P?-+   "
   230 FORI=1 TOLEN<K$>:K<A$C<MID(K$,I>>>=I:NEXT
   240 PRINT "<CRUP>
   250 AT=0:DE=0:SU=15:RE=9:SV=SU+1G+RE:AV=AT*16+DE:
       WV=16:W=0:M=1:OC=4:HB=256:Z=0
   260 FORI=0TO2:POKES+5+I*7,AT*16+DE:POKES+6+I*7,SU*16+RE
   270 POKES+2+I*7,4000AND255:POKES+3+I*7;4000/256:NEXT
   280 POKES+24,15:REM+16+64:POKES+23.7
   300 GETA $:IFA$=""THEN300
   310 FR=F(K(ASC(A$)))/M:T=V*7:CR=S+T+4:IFFR=ZTHEN500
6  320 POKES+6+T,Z,:REM FINISH DEC/SUS
   325 POKES+5+T,Z:REM FINISH ATT/REL
   330 POKECR,8:POKECR,0:REM FIX OFF
   340 POKES+T,FR-HB*INT<FR/HB>:REM SET LO
   350 POKES+1+T,FR/HB:REM SET HI
   360 POKES+6+T,SV:REM SET DEC/SUS
   365 POKES+5+T,AV:REM SET ATT/REL
   370 POKECR,WV+1:FORI=1TO50+AT:NEXT
7  375 POKECR,WV:REM PULSE
   390 IFP=1 THENV=V+1:IFV=3THENV=0
   400 GOTO300
   500 IFA$="<F1>"THENM=1:OC=4:GOTO300
   510 IFA$="<F3>" THENM=2:OC=3:GOTO300
   520 IFA$="<F5>" THENM=4:OC=2:GOTO300
   530 IFA$="<F7>" THENM=8:OC=1:GOTO300
   540 IFA$="<F2>" THENW=0:WV=16:GOTO300
   550 IFA$="<F4>" THENW=1:WV=32:GOTO300
   560 IFA$="<F6>" THENW=2:WV=64:GOTO300
   570 IFA$="<F8>" THENW=3:WV=128:GOTO300
   580 IFA$=" " THENP=1-P:GOTO300
   590 IFA$="<CLR>" THEN200
   600 GOTO300
8  800 PRINT "HIT A KEY"
9  810 GETA$:IFA$=" "THEN810:WAIT FOR A KEY
   820 PRINTA$:RETURN
```

Рис. 78

1 - ...клавиатура фортепиано; 2 - "...для соло или многоголосия"; 3 - "...выбор октавы"; 4 - "...форма сигнала"; 5 - ... "придерживайтесь частотной шкалы..."; 6 - ...окончание; 7 - ...импульсный сигнал; 8 - ..."нажмите на клавишу"; 9 - ... жмите клавишу

x)

Приложение М

Сообщения об ошибках

В этом Приложении содержится полный список сообщений об ошибках, которые вырабатываются компьютером "Коммодор-64", и приводится описание причин появления этих сообщений.

BAD DATA (НЕПРАВИЛЬНЫЕ ДАННЫЕ) - Из открытого файла были получены цепочечные данные, однако программа ожидала числовые данные.

BAD SUBSCRIPT (НЕПРАВИЛЬНЫЙ ИНДЕКС) - Программа осуществляла поиск элемента массива, номер которого находится за пределами диапазона, оговоренного оператором DIM.

CAN'T CONTINUE (ПРОДОЛЖЕНИЕ НЕВОЗМОЖНО) - Команда CONT (ПРОДОЛЖАТЬ) не работает, либо потому что ПРОГОН (RUN) программы не был запущен, так как имела место ошибка, либо строка была отредактирована.

DEVICE NOT PRESENT (УСТРОЙСТВО ОТСУТСТВУЕТ) - Требуемое устройство входа/выхода недоступно для операторов OPEN, CLOSE, CMD, PRINT#, INPUT# или GET#.

DIVISION BY ZERO (ДЕЛЕНИЕ НА НУЛЬ) - Деление на нуль является математической бессмыслицей и не допускается.

EXTRA IGNORED (ЛИШНИЕ ДАННЫЕ НЕ ВОСПРИНИМАЮТСЯ) - В ответ на оператор INPUT (ВВЕСТИ) было напечатано (набрано на клавиатуре) слишком много элементов информации.

FILE NOT FOUND (ФАЙЛ НЕ НАЙДЕН) - Появляется в случае, если Вы искали файл на ленте, и был обнаружен маркер END-OF-TAPE (КОНЕЦ ЛЕНТЫ). Если Вы искали файл на диске, то это сообщение появляется в случае, если файла с таким именем не существует.

FILE NOT OPEN (ФАЙЛ НЕ ОТКРЫТ) - файл, определенный операторами CLOSE, CMD, PRINT#, INPUT# или GET#, сначала должен быть ОТКРЫТ (OPEN).

FILE OPEN (ФАЙЛ ОТКРЫТ) - Была сделана попытка открыть файл с помощью номера уже открытого файла.

FORMULA TOO COMPLEX (ФОРМУЛА СЛИШКОМ СЛОЖНА) - оцениваемое последовательное выражение должно быть разбито по крайней мере на две части, для того чтобы система могла с ним работать.

ILLEGAL DIRECT (НЕПОСРЕДСТВЕННЫЙ РЕЖИМ НЕПРАВИЛЕН) - Оператор INPUT может использоваться только в программе, а не в непосредственном режиме.

ILLEGAL QUANTITY (НЕПРАВИЛЬНАЯ ВЕЛИЧИНА) - Число, исполь-

---

x)
В тексте оригинала стр. 147-148 отсутствуют (прим. пер.).

зуемое в качестве аргумента функции или оператора выходит за пределы допустимого диапазона.

LOAD (ЗАГРУЗКА) - Имеется ошибка в программе на ленте.

NEXT WITHOUT FOR (NEXT БЕЗ FOR) - Это вызывается либо неправильными вложенными циклами, либо в операторе NEXT имеется имя переменной, которое не соответствует имени переменной в операторе FOR.

NOT INPUT FILE (НЕ ВХОДНОЙ ФАЙЛ) Была сделана попытка ВВЕСТИ (INPUT) или ВЗЯТЬ (GET) данные из файла, который предназначен только для выхода.

NOT OUTPUT FILE (НЕ ВЫХОДНОЙ ФАЙЛ) - Была сделана попытка ВЫВЕСТИ НА ПЕЧАТЬ (PRINT) данные в файл, который предназначен только для ввода.

OUT OF DATA (НЕТ ДАННЫХ) - Был выполнен оператор READ (СЧИТЫВАНИЕ), но в операторе DATA (ДАННЫЕ) не осталось несчитанных данных.

OUT OF MEMORY (НЕХВАТКА ПАМЯТИ) - Больше не имеется свободных ячеек ОЗУ для программы или переменных. Это может также иметь место когда было осуществлено вложение очень многих циклов FOR или когда задействовано слишком много операторов GOSUB.

OVERFLOW (ПЕРЕПОЛНЕНИЕ) - Результатом вычисления является число, большее предельно допустимого числа в компьютере, т.е. большее, чем число 1,70141884E+38.

REDIMD ARRAY (ИЗМЕНЕНИЕ РАЗМЕРНОСТИ МАССИВА) - размерность массива может быть определена только один раз. Если переменная массива используется до того, как определена его размерность, то над этим массивом автоматически выполняется операция определения размерности (количество элементов устанавливается равным десяти), и любые последующие операторы DIM вызовут состояние ошибки.

REDO FROM START (ПЕРЕДЕЛАЙТЕ С НАЧАЛА) - При выполнении оператора INPUT, когда ожидались числовые данные, последовал набор с клавиатуры буквенных данных. Просто снова наберите ввод данных, так чтобы он был правильным, и программа будет продолжаться сама собой.

RETURN WITHOUT GOSUB (RETURN БЕЗ GOSUB) - Встретился оператор RETURN, а команды GOSUB не последовало.

STRING TOO LONG (СЛИШКОМ ДЛИННАЯ ЦЕПОЧКА) - Цепочка может содержать не более 255 знаков.

?SYNTAX ERROR (ОШИБКА В СИНТАКСИСЕ) - Компьютер не распознает оператор. Здесь могут быть случаи пропуска или добавления лишних скобок, неправильный ввод паролей и т.п.

TYPE MISMATCH (НЕСОГЛАСОВАННОСТЬ ТИПА ДАННЫХ) - Эта ошибка возникает, когда вместо цепочки используется число или наоборот.

UNDEF'D FUNCTION (ФУНКЦИЯ НЕ ОПРЕДЕЛЕНА) - Осуществляется поиск функции, определенной пользователем, однако функция не была определена с помощью оператора DEF FN.

UNDEF'D STATEMENT (ОПЕРАТОР НЕ ОПРЕДЕЛЕН) - Была сделана попытка реализовать операторы GOTO, или GOSUB или RUN в отношении номера строки, который не существует.

VERIFY (ПРОВЕРИТЬ) - Программа на ленте или диске не соответствует программе, находящейся в настоящее время в памяти.

Приложение H

Величины, относящиеся к музыкальным нотам

В этом Приложении приведен полный список номеров нот, действительных нот и величин, которые должны быть ВСТАВЛЕНЫ (POKE) в модули регистров HI FREQ (ВЫСОКАЯ ЧАСТОТА) и LOW FREQ (НИЗКАЯ ЧАСТОТА) для генерирования указанной ноты.

Таблица 10

Note	Note-Octave	Hi Freq	Low Freq
0	C-0	1	12
1	C#-0	1	28
2	D-0	1	45
3	D#-0	1	62
4	E-0	1	81
5	F-0	1	102
6	F#-0	1	123
7	G-0	1	145
8	G#-0	1	169
9	A-0	1	195
10	A#-0	1	221
11	B-0	1	250
16	C-1	2	24
17	C#-1	2	56
18	D-1	2	90
19	D#-1	2	125
20	E-1	2	163
21	F-1	2	204
22	F#-1	2	246
23	G-1	3	35
24	G#-1	3	83
25	A-1	3	134
26	A#-1	3	187
27	B-1	3	244
32	C-2	4	48
33	C#-2	4	112
34	D-2	4	180
35	D#-2	4	251
36	E-2	5	71
37	F-2	5	152
38	F#-2	5	237
39	G-2	6	71
40	G#-2	6	167
41	A-2	7	12
42	A#-2	7	119
43	B-2	7	233
48	C-3	8	97

Таблица 10 (продолжение)

1 Note	2 Note-Octave	3 Hi Freg	4 Low Freg
49	C#-3	8	225
50	D-3	9	104
51	D#-3	9	247
52	E-3	10	143
53	F-3	11	48
54	F#-3	11	218
55	G-3	12	143
56	G#-3	13	78
57	A-3	14	24
58	A#-3	14	239
59	B-3	15	210
64	C-4	16	195
65	C#-4	17	195
66	D-4	18	209
67	D#-4	19	239
68	E-4	21	31
69	F-4	22	96
70	F#-4	23	181
71	G-4	25	30
72	G#-4	26	156
73	A-4	28	49
74	A#-4	29	223
75	B-4	31	165
80	C-5	33	135
81	C#-5	35	134
82	D-5	37	162
83	D#-5	39	223
84	E-5	42	62
85	F-5	44	193
86	F#-5	47	107
87	G-5	50	60
88	G#-5	53	57
89	A-5	56	99
90	A#-5	59	190
91	B-5	63	75
96	C-6	67	15
97	C#-6	71	12
98	D-6	75	69
99	D#-6	79	191
100	E-6	84	125
101	F-6	89	131
102	F#-6	94	214
103	G-6	100	121
104	G#-6	106	115
105	A-6	112	199
106	A#-6	119	124

Таблица 10 (продолжение)

1 Note	2 Note-Octave	3 Hi Freq	4 Low Freq
107	B-6	126	151
112	C-7	134	30
113	C#-7	142	24
114	D-7	150	139
115	D#-7	159	126
116	E-7	168	250
117	F-7	179	6
118	F#-7	189	172
119	G-7	200	243
120	G#-7	212	230
121	A-7	225	143
122	A#-7	238	248
123	B-7	253	46

1 - нота; 2 - нота-октава; 3 - высокая частота; 4 - низкая частота
Обозначение нот в английском языке: A - ля, B - си, C - до, D - ре, E - ми, F - фа, C - соль (прим.пер.)

Приложение 0

Список литературы

Addison Wesley — "BASIC and the Personal Computer Dwyer and Critchfield

Dilithium Press — "BASIC Basic-English Dictionary for the Pet"; Larry Noonan

Faulk Baker Associates — "MOS Programming Manual", MOS Technology

Hayden Book Co — "BASIC From the Ground Up", David E.Simon

"I Speak BASIC to My PET",Aubrey Jones,Jr

Little, Brown and Co. — Computer Games for Businesses, Schools and Homes", J.Victor Nagigian and William S.Hodges

"The Computer Tutor: Learning Activities for Homes and Schools", Gary W.Orwig, University of Central Florida and William S. Hodges

McGraw Hill — "Hands-On BASIC With a PET", Herbert D. Packman

Osborne/McGraw Hill — "PET/CBM Personal Computer Guide", Corrol S. Donahue

"Osborne CP/M User Guide", Thom Hogan

"Some Common Basic Programs for the PET", L.Poole,M.Borchers

"The 8086 Book", Russell Rector and George Alexy

Reston Publishing Co. "Pet and the IEEE 488 Bus (GPIB)", Eugene Fisher and C.W.Jensen
"PET BASIC - Training Your PET Computer", Ramon Zamora, Wm.F.Carrie and Bod Albrecht
"Pet Games and Recreation", Mac Ogeisby, Len Lindsey and Dorothy Kunkin

В журналах фирмы "Коммодор" публикуется самая современная информация об использовании микрокомпьютера "Коммодор-64". Рекомендуем Вам рассмотреть вопрос о подписке на два наиболее популярных журнала.

COMMODORE - ежемесячный подписной журнал по микрокомпьютерам (стоимость полугодовой подписки - 15 долл. для США и 25 долл. вне США).

POWER PLAY - ежеквартальный подписной журнал по вычислительной технике (стоимость годовой подписки - 10 долл. для США и 15 долл. для других стран).

Приложение П

Карта регистра спрайтов

Таблица 11

1 Register # 2 Dec	Nex	D87	D86	D85	D84	D83	D82	D81	D80		
0	0	S0X7							S0X0	SPRITE 0X Component	4
1	1	S0Y7							S0Y0	SPRITE 0Y Component	5
2	2	S1X7							S1X0	SPRITE 1X	6
3	3	S1Y7							S1Y0	SPRITE 1Y	7
4	4	S2X7							S2X0	SPRITE 2X	8
5	5	S2Y7							S2Y0	SPRITE 2Y	9
6	6	S3X7							S3X0	SPRITE 3X	101
7	7	S3Y7							S3Y0	SPRITE 3Y	111
8	8	S4X7							S4X0	SPRITE 4X	12
9	9	S4Y7							S4Y0	SPRITE 4Y	13
10	A	S5X7							S5X0	SPRITE 5X	14

Таблица 11 (продолжение)

Register # Dec	Hex	D87	D86	D85	D84	D83	D82	D81	D80		
11	B	S5Y7							S5Y0	SPRITE 5Y	15
12	C	S6X7							S6X0	SPRITE 6X	16
13	D	S6Y7							S6Y0	SPRITE 6Y	17
14	E	S7X7							S7X0	SPRITE 7X Component	18
15	F	S7Y7							S7Y0	SPRITE 7Y Component	19
16	10	S7X8	S6X8	S5X8	S4X8	S3X8	S2X8	S1X8	S0X8	MSB of X COORD	20
17	11	RC8	ECM	BMM	BLNK	RSEL	YSCL2	YSCL 1	YSCL 0	Y SCROLL MODE	
18	12	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	RASTER	21
19	13	LPX7							LPX0	LIGHT PEN X	22
20	14	LPY7							LPY0	LIGHT PEN Y	23
21	15	SE7							SE0	SPRITE ENABLE (ON/OFF)	24
22	16	N.C.	N.C.	RST	MCM	CSEL	XSCL2	XSCL 1	XSCL 0	X SCROLL MODE	
23	17	SEXY 7							SEXY 0	SPRITE EXPAND Y	25
24	18	VS13	VS12	VS11	VS10	CB13	CB12	CB11	N.C.	SCREEN Character Memory	26
25	19	IRQ	N.C.	N.C.	N.C.	LPI-RQ	ISSC	ISBC	RIRQ	interupt Request's	27

Таблица 11 (окончание)

Register # 3 2Dec Hex	D87	D86	D85	D84	D83	D82	D81	D80	
26 1A	N.C.	N.C.	N.C.	N.C.	MLPI	MISSC	MISBC	MRIRQ	Interupt Request MASKS 28
27 1B	BSP7							BSPO	Backgro- und-Sprite PRIORITY 29
28 1C	SCM7							SCMO	MULTICOLOR SPRITE SELECT 30
29 1D	SEXX7							SEXX0	SPRITE EXPAND X 31
30 1E	SSC7							SSCO	Sprite- Sprite COLLISION 32
31 1F	SBC7							SBCO	Sprite- Background COLLISION 33

1 - номер регистра; 2 - десятичный код; 3 - шестнадцатиричный код; 4 - компонента X спрайта 0; 5 - компонента Y спрайта 0; 6 - компонента X спрайта 1; 7 - компонента Y спрайта 1; 8 - компонента X спрайта 2; 9 - компонента Y спрайта 2; 10 - компонента X спрайта 3; 11 - компонента Y спрайта 3; 12 - компонента X спрайта 4; 13 - компонента Y спрайта 4; 14 - компонента X спрайта 5; 15 - компонента Y спрайта 5; 16 - компонента X спрайта 6; 17 - компонента Y спрайта 6; 18 - компонента X спрайта 7; 19 - компонента Y спрайта 7; 20 - величина MSB координаты X; 21 - растр; 22 - координата X светового пера; 23 - координата Y светового пера; 24 - задействование спрайта (ON/OFF - ВКЛЮЧЕНО/ВЫКЛЮЧЕНО); 27 - запросы на прерывания; 28 - МАСКИ запросов на прерывания; 29 - ПРИОРИТЕТ фона-спрайта;

30 - выбор многоцветного спрайта; 31 - расширение спрайта по координате X; 32 - столкновение спрайта со спрайтом; 33 - столкновение спрайта с фоном

Таблица 12

1 COLOR CODES				2 DEC	3 HEX	4 COLOR
0	0	BLACK	5	32	20	BORDER COLOR
1	1	WHITE	6	33	21	BACKGROUND COLOR 0
2	2	RED	7	34	22	BACKGROUND COLOR 1
3	3	CYAN	8	35	23	BACKGROUND COLOR 2
4	4	PURPLE	9	36	24	BACKGROUND COLOR 3
5	5	GREEN	10	37	25	SPRITE MULTICOLOR 0
6	6	BLUE	11	38	26	SPRITE MULTICOLOR 1
7	7	YELLOW	12	39	27	SPRITE 0 COLOR
8	8	ORANGE	13	40	28	SPRITE 1 COLOR
9	9	BROWN	14	41	29	SPRITE 2 COLOR
10	A	LT RED	15	42	2A	SPRITE 3 COLOR
11	B	GRAY 1	16	43	2B	SPRITE 4 COLOR
12	C	GRAY 2	17	44	2C	SPRITE 5 COLOR
13	D	LT GREEN	18	45	2D	SPRITE 6 COLOR
14	E	LT BLUE	19	46	2E	SPRITE 7 COLOR
15	F	GRAY 3	20			

LEGEND:
21 ONLY COLORS 0-7 MAY BE USED IN MULTICOLOR CHARACTER MODE

1 - коды цвета; 2 - десятичный код; 3 - шестнадцатиричный код; 4 - цвет; 5 - черный; 6 - белый; 7 - красный; 8 - сине-зеленый; 9 - пурпурный; 10 - зеленый; 11 - синий; 12 - желтый; 13 - оранжевый; 14 - коричневый; 15 - светлокрасный; 16 - серый 1; 17 - серый 2; 18 - светлозеленый; 19 - голубой; 20 - серый 3; 21 - легенда: в режиме многоцветного знака могут использоваться только цвета 0-7

Приложение Р

Управление звуком в компьютере "Коммодор-64"

В этой компактной таблице, приведенной в этом Приложении указываются ключевые числа, которые понадобятся Вам при составлении звуковых программ, и которые относятся к тому из трех голосов компьютера, который Вы хотите выбрать. Для установки или регулировки звука в программе на языке БЕЙСИК Вам необходимо осуществить ВСТАВКУ (POKE) числа из второй колонки, за которым идет запятая (,) и число из таблицы, например, POKE 54276,17 (осуществляется выбор треугольной формы сигнала для ГОЛОСА 1)

Имейте в виду, что Вам необходимо установить ГРОМКОСТЬ (VOLUME) до того как Вы начнете генерировать звук. POKE54296, за которым идет номер от Ø до 15, устанавливает громкость для всех трех голосов.

Для генерирования каждой музыкальной ноты необходимо осуществить две отдельные ВСТАВКИ (POKE). Например, POKE54273,34:POKE54272,75 обозначает низкую ноту "до" в эталонной шкале, приведенной ниже.

Однако, Вы не ограничены числами, показанными в таблицах. Если 34 не дает "правильного" звука для низкой ноты "до", попробуйте, как звучит число 35. Для того чтобы обеспечить более высокую скорость SUSTAIN (ВЫДЕРЖИВАНИЕ) или ATTACK (НАРАСТАНИЕ) по сравнению с указанными в таблице, сложите два или более чисел SUSTAIN. Например, POKE54277,96 объединяет две скорости нарастания (32 и 64) для комбинированной более высокой скорости нарастания, в то время как POKE54277,2Ø обеспечивает малую скорость нарастания (16) и среднюю скорость затухания (4).

Таблица 13

1 SETTING VOLUME - SAME FOR ALL 3 VOICES		
VOLUME CONTROL	POKE54296	Settings range from Ø(off) to 15 (loudest)
2		3
TO CONTROL 4 THIS SETTING	POKE THIS NUMBER 5	6 FOLLOWED BY ONE OF THESE NUMBERS (Ø to 15... or ... Ø to 255 depending on range)

TO PLEY A NOTE	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#
HIGH FREQUENCY	54273 34	36	38	40	43	45	48	51	54	57	61	64	68	72

Таблица 13 (продолжение)

9 LOW FRE-QUENCY	54272	135	134	162	223	62	193	107	60	57	99	190	75
WAVEFORM 10	POKE	11 TRIANGLE		12 SAWTOOTH		13 PULSE		14 NOISE					
	54276	17		33		65		129					
15 PULSE RATE (Pulse Waveform)													18
16 HI PULSE	54275	A value of 0 to 15(for Pulse waveform only)											
17 LO PULSE	54274	A value of 0 to 255(for Pulse waveform only)											19
ATTACK/DECAY	POKE	ATK4	ATK3	ATK2	ATK1	DEC4	DEC3	DEC2	DEC1				
20	54277	128	64	32	16	8	4	2	1				
SUSTAIN/RELEASE	POKE	SUS4	SUS3	SUS2	SUS1	REL4	REL3	REL2	REL1				
21	54278	128	64	32	16	8	4	2	1				

7 TO PLAY A NOTE	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
8 HIGH FRE-QU-ENCY	54280 33	35	37	39	42	44	47	50	53	56	59	63
9 LOW FRE-QU-ENCY	54279 136	134	162	223	62	193	107	60	57	99	190	75

WAVEFORM	POKE	11 TRIANGLE	12 SAWTOOTH	13 PULSE	14 NOISE
10	54283	17	33	65	129

22 PULSE RATE			18
16 HI PULSE	54282	A value of 0 to 15(for Pulse waveform only)	
17 LO PULSE	54281	A value of 0 to 255(for Pulse waveform only)	19

Таблица 13 (продолжение)

ATTACK/DECAY	POKE	ATK4	ATK3	ATK2	ATK1	DEC4	DEC3	DEC2	DEC1
	54284	128	64	32	16	8	4	2	1
SUSTAIN/ RELEASE	POKE 54285	SUS4 128	SUS3 64	SUS2 32	SUS1 16	REL4 8	REL3 4	REL2 2	REL1 1

TO PLAY A NOTE		C	C#	D	D#	E	F	F#	G	G#	A	A#	B
HIGT FREQUENCY	54287	33	35	37	39	42	44	47	50	53	56	59	63
LOW FRE-QUENCY	54286	135	134	162	223	62	193	107	60	57	99	190	75

WAVEFORM	POKE	TRAINGLE	SAWTOOTH	PULSE	NOISE
	54290	17	33	65	129

PULSE RATE (Pulse Waveform)

HI PULSE	54289	A value of 0 to 15 (for Pulse waveform only)
LO PULSE	54288	A value of 0 to 255(for Pulse waveform only)

ATTAK/DECAY	POKE	ATK4	ATK3	ATK2	ATK1	DEC4	DEC3	DEC2	DEC1
	54291	128	64	32	16	8	4	2	1
SUSTAIN/ RELEASE	POKE 54292	SUS4 128	SUS3 64	SUS2 32	SUS1 16	REL4 8	REL3 4	REL2 2	REL1 1

1 - установка громкости - одна для всех трех голосов; 2 - регулировка громкости; 3 - установки находятся в диапазоне от 0 (включено) до 15 (максимальная громкость); 4 - для управления этой установкой; 5 - ВСТАВЬТЕ это число; 6 - за которым следует одно из этих чисел (от 0 до 15 или от 0 до 255 в зависимости от диапазона); 7 - для того чтобы сыграть ноту; 8 - высокая частота; 9 - низкая частота; 10 - форма сигнала; 11 - треугольная; 12 - пилообразная; 13 - импульсная; 14 - шум; 15 - скорость следования импульсов (для импульсной формы сигнала); 16 - высокая скорость импульсов; 17 - низкая скорость импульсов; 18 - величина от 0 до 15 (только для импульсной формы

сигнала); 19 - величина от 0 до 255 (только для импульсной формы сигнала); 20 - нарастание/затухание; 21 - выдерживание/отпускание; 22 - скорость следования импульсов.

Попробуйте использовать нижеприведенные установки для имитации звучания различных музыкальных инструментов.

Таблица 14

1 Instrument	2 Waveform	3 Attack/Decay	4 Sustain/Release	5 Pulse Rate
6 Piano	Pulse 7	9	0	Hi-0, Lo-255 8 9
9 Flute	Triangle 10	96	0	Not applicable 11
12 Horpsichord	Sawtooth 13	9	0	Not applicable 11
14 Xylophone	Triangle 10	9	0	Not applicable 11
15 Organ	Triangle 10	0	240	Not applicable 11
16 Colliape	Triangle 10	0	240	Not applicable 11
17 Accordian	Triangle 10	102	0	Not applicable 11
18 Trumpet	Sawtooth 13	96	0	Not applicable 11

1 - инструмент; 2 - форма сигнала; 3 - нарастание/затухание; 4 - выдерживание/отпускание; 5 - скорость следования импульсов; 6 - фортепиано; 7 - импульсная; 8 - высокая - 0, низкая - 255; 9 - флейта; 10 - треугольная; 11 - не применяется; 12 - клавикорды; 13 - пилообразная; 14 - ксилофон; 15 - орган; 16 - коллиап; 17 - аккордеон; 18 - труба

Значения звуковых терминов

ADSR - Нарастание/Затухание/Выдерживанне/Отпускание
Нарастание-скорость, с которой звук нарастает до максимального
Затухание - скорость, с которой звук спадает от максимального до уровня выдерживания
Выдерживание - звучание ноты при определенной громкости
Отпускание - скорость, с которой звук спадает от уровня выдерживания
Форма сигнала - "форма" звуковой волны
Импульс - характеризует качество тона импульсной формы сигнала
Примечание. Установки Attack/Decay (Нарастание/Затухание) и Sustain/Release (Выдерживание/Отпускание) должны всегда ВСТАВЛЯТЬСЯ (POKE) в Вашу программу ДО того как осуществлена ВСТАВКА (POKE) формы сигнала.

Фирма "Коммодор" надеется, что "Руководство по использованию микрокомпьютера "Коммодор-64" будет для Вас полезным. Хотя в этом руководстве содержится определенная информация и практические советы, оно НЕ предназначено для использования в качестве справочника для программиста. Для опытных программистов и специалистов, работающих с ЭВМ, большую пользу может принести "Справочник программиста для работы с компьютером "Коммодор-64", который Вы можете приобрести у местного представителя фирмы "Коммодор".

Справочный лист для работы на компьютере "Коммодор-64"

Простые переменные

Тип	Имя	Диапазон
Действительные	XY	+ 1,70141183E+38
		-
		+ 2,93873588E-39
		-
Целочисленные	XY%	+ 32767
		-
Цепочечные	XY$	от 0 до 255 знаков

X - это буква (A-Z), Y - буква или цифра (0-9). Имена переменных могут содержать более двух знаков, однако опознаются только два первых знака.

Переменные с массивами

Тип	Имя
Одномерные	XY(5)
Двумерные	XY(5,5)
Трехмерные	XY(5,5,5)

При необходимости могут быть использованы массивы с числом элементов до 11 (индексы 0-10). У массивов, имеющих более 11 элементов, должен быть определен размер (DIM).

Алгебраические операторы

=	Присваивает переменной величину
-	Отрицание
	Возведение в степень
*	Умножение
/	Деление
+	Сложение
-	Вычитание

Относительные и логические операторы

=	Равно
<>	Не равно
<	Меньше чем
>	Больше чем

<= Меньше или равно
>= Больше или равно
NOT Логическое "НЕ"
END Логическое "И"
Выражения равны 1, если они истинны, и Ø, если они ложны.

Системные команды

LOAD "NAME"	Загружает программу с ленты
SAVE "NAME"	Записывает программу на ленту
LOAD "NAME",8	Загружает программу с диска
SAVE "NAME",8	Записывает программу на диск
VERIFY "NAME"	Проверяет, нет ли ошибок в записанной программе
RUN	Выполняет программу
RUN xxx	Выполняет программу, начиная со строки xxx
STOP	Останавливает выполнение программы
END	Прекращает выполнение программы
CONT	Продолжает выполнение программы с той строки, на которой программа была остановлена
PEEK(X)	Выводит содержимое ячейки памяти X
POKE X,Y	Меняет содержимое ячейки X на величину
SYS xxxxx	Осуществляет переход к выполнению программы на машинном языке начиная с xxxxx
WAIT X,Y,Z	Программа ожидает, пока содержимое ячейки X при осуществлении операции FOR с Z и AND с Y станет не нулем
USR(X)	Передает величину X в стандартную подпрограмму машинного языка

Редактирующие и форматирующие команды

LIST	Выводит листинг всей программы
LIST A-B	Выводит листинг программы от строки A до строки B
REM Message	Комментирует сообщение, может быть выведено на листинг, но на выполнение программы не влияет
TAB(X)	Используется в операторах PRINT. Занимает на экране X позиций
SPC(X)	Осуществляет печать (PRINT) в строке X бланков
POS(X)	Показывает текущее положение курсора
CLR/HOME	Устанавливает курсор в левом верхнем углу экрана
SHIFT CLR/HOME	Экран очищается и курсор устанавливается в исходное положение
SHIFT INST/DEL	Вставляет пробел на то место, где в настоящее время находится курсор
CTRL	При использовании с клавишами цветовых чисел выбирает цвет текста. Может также использоваться в операторе PRINT.

CRSR клавиши	Перемещает курсор по экрану вверх, вниз, налево и направо
Клавиша "КОММОДОР"	Когда используется с клавишей SHIFT, осуществляет выбор между верхним и нижним регистрами и режимом графического дисплея. Когда используется с клавишей цветового числа, выбирает дополнительный цвет текста.

Массивы и цепочки

DIM A(X,Y,Z)	Устанавливает максимальное количество индексов для A, оставляет место для (X+1)*(Y+1)*(Z+1) элементов, начиная с A(Ø,Ø,Ø)
LEN (X$)	Выводит количество знаков в X$
STR$(X)	Выводит числовую величину X, превращенную в цепочку
VAL(X$)	Выводит числовую величину X до первого нечислового знака
CHR$(X)	Выводит знак ASCII, код которого равен X
ASC(X$)	Выводит код ASCII для первого знака X
LEFT (A$,X)	Выводит X крайних левых знаков A$
RIGHT(A$,X)	Выводит X крайних правых знаков A$
MID$(A$,X,Y)	Выводит Y знаков A$, начиная со знака X

Команды входа/выхода

INPUT A$ OR A	Выводит на экран "?", ожидает введения пользователем цепочки или величины
INPUT "ABC",A	Выводит сообщение и ожидает введения пользователем величины. Может также осуществлять операцию INPUT A$
GET A$ или A	Ожидает, пока пользователь не наберет на клавиатуре один знак. Нажатия клавиши RETURN не требуется
DATA A,"B",C	Устанавливает в начальное состояние набор величин, которые могут быть использованы оператором READ
READ A$ или A	Присваивает следующую величину DATA A$ или A
RESTORE	Сбрасывает указатель данных для повторного начала считывания данных
PRINT "A=";A	Выводит на печать (PRINT) цепочку "A=" и величину A;";"-ликвидирует пробелы-";" выводит табличные данные в новое поле

Ход выполнения программы

GOTO X	Осуществляет переход к строке X
IF A=3 THEN 1Ø	Если утверждение истинно, то выполняется следующая часть оператора. Если утверждение ложно, то выполняется следующая строка программы

FOR A=1 TO 1Ø STEP 2:NEXT	Выполняет все операторы между FOR и соответствующим NEXT, причем A меняется от 1 до 1Ø с величиной шага изменения, равным 2. Если шаг не определен, то он считается равным 1
NEXT A	Определяет окончание цикла. A - произвольно
GOSUB 2ØØØ	Осуществляется переход к стандартной подпрограмме со строки 2ØØØ
RETURN	Отмечает окончание стандартной подпрограммы. Возвращается к оператору, идущему непосредственно после последнего оператора GOSUB
ON X GOTO A,B	Осуществляется переход к строке с номером X в списке программы. Если X = 1, то осуществляется переход к A, и т.д.
ON X GOSUB A,B	Осуществляется переход к подпрограмме со строки с номером X в списке