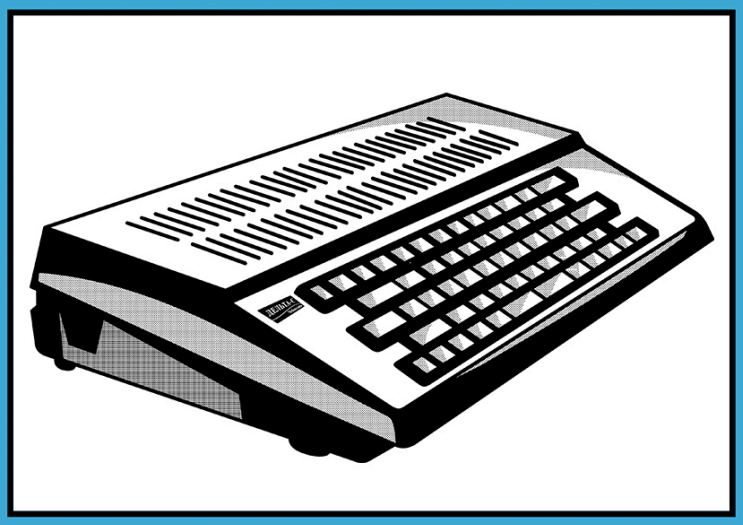
ПЕРСОНАЛЬНЫЙ БЫТОВОЙ

КОМПЬЮТЕР



ДЕЛЬТА-С

ТЕХНИЧЕСКОЕ ОПИСАНИЕ И ИНСТРУКЦИЯ ПО ЭКСПЛУАТАЦИИ

РЕПРОДУКЦИЯ
Сергей Old Fart Gamer Сухих
2021

ЧЕБОКСАРСКОЕ ПРЕДПРИЯТИЕ
«СЕСПЕЛЬ»

Техническое описание и инструкция по эксплуатации персонального бытового компьютера « Дельта - С »

Составитель:
Дьяконов В.П.
Калаева Т.А.

Репродукция Сергей OldFartGamer Сухих
2021

СОДЕРЖАНИЕ

страница

Репродукция оригинального руководства выполнена в новогодние «каникулы» 2021 года. За основу взяты сканированные страницы из нескольких источников (странно, что в 2020 году было сложно найти в интернете полноценный скан со всеми страницами в хорошем качестве). Текст был полностью перенабран заново, часть иллюстраций перерисована. Таблица кодов ассемблера позаимствована из книги ИНФОРКОМ «Персональный компьютер ZX-Spectrum - Программирование в машинных кодах и на языке Ассемблера».

Поддержать автора репродукции можно как морально в социальных сетях, так и материально любой суммой на кошелёк Юmoney - 41001700743790

Данное руководство по применению бытового персонального компьютера «Дельта-С» состоит из трех глав. В первой, ориентированной на пользователя малознакомого с ЭВМ, дается общее представление о работе с компьютером и его возможностях. Во второй описан язык программирования Бейсик, составляющий неотъемлемую часть встроенного программного обеспечения, и выполняющий функции операционной системы. Третья глава полезна более опытным пользователям и содержит описание некоторых секретов компьютера распределения памяти, назначения системных переменных.

ГЛАВА 1. ОБЩЕЕ ЗНАКОМСТВО С КОМПЬЮТЕРОМ.

1.1. Технические характеристики.

Персональный бытовой компьютер «Дельта - С» является 8-разрядной микро-ЭВМ, построенной на основе микропроцессора Z80A или его аналогов. По программному обеспечению он полностью совместим с одной из наиболее массовых моделей зарубежных персональных ЭВМ ZX-Spectrum фирмы SINCLAIR RESEARCH LTD (Великобритания). Для последнего разработаны тысячи прикладных программ - от простых и сложных игр до программы автоматизации сложных расчетов и проектирования различных устройств

На рис, 1 представлен компьютер «Дельта - С» с набором подключаемого к нему периферийного оборудования. В него входит: блок питания, кассетный магнитофон для записи и считывания программ и данных, дисплей или телевизор и печатающее устройство - принтер. На рис. 1 схематично показаны соединения компьютера с этими устройствами.

Компьютер "Дельта С" имеет следующие технические характеристики:

1. Разрядность шины данных микропроцессора - 8;
2. Разрядность шины адресов микропроцессора - 16;
3. Объем постоянного запоминающего устройства - 16 Кбайт;
4. Объем памяти оперативного запоминающего устройства - 48 Кбайт;
5. Тактовая частота микропроцессора 3.5 МГц;
6. Встроенное в ПЗУ программное обеспечение - системный монитор и язык Бейсик;
7. Разрешение графики 192*256 элементов при 8 цветах;

Монитор
или
Телевизор

Магнитофон

Компьютер

Принтер

Джойстик

Рис. 1

7. Разрешение графики 192*256 элементов при 8 цветах;
8. Габаритные размеры 50x219x334
9. Масса 2.2 кг.
10. Потребляемая мощность P = 10 Вт

1.2. Подготовка компьютера к работе.

Распакуйте компьютер и подключите к нему дисплей (телевизор), кассетный магнитофон и блок питания. Включите компьютер и дисплей (телевизор) и добейтесь изображения на экране в нижней его части сообщения об автоматической загрузке встроенного в ПЗУ Бейсик - интерпретатора.

©.1989 DELTA

Вставьте в отсек кассетного магнитофона кассету с записанными на ней программами и попробуйте задать команду:

LOAD ""

Для ввода команды **LOAD** надо нажать клавишу с соответствующей надписью и буквой **J**. Чтобы ввести кавычки, придется нажать одновременно клавиши **Symbol Shift** и **P** (на ней имеется и изображение кавычек "). После этого нажмите клавишу фиксации ввода **ENTER**.

Теперь включите магнитофон в режиме воспроизведения и наблюдайте процесс загрузки программы. Будет загружаться ближайшая программа из записанных на кассете. Рекомендуем начать загрузку с первой демонстрационной программы, для чего нужно предварительно перемотать ленту на начало стороны Б кассеты.

Как только начнется загрузка программы, появится звуковой сигнал, а по обрамлению экрана (бордюру) побегут характерные полосы. Затем на экране появится название загружаемой программы. Следует дождаться полной загрузки программы, на что указывает сообщение **OK** (O`Key - всё в порядке).

Теперь введите команду **RUN**, зафиксировав ввод нажатием клавиши **ENTER**. Программа начнёт исполняться и результат ее прогона Вы сможете наблюдать. Демонстрационная программа покажет Вам возможности компьютера. После её окончания введите команду **LIST**. Вы увидите текст (листинг) программы на языке Бейсик.

1.3. Программное обеспечение

В программное обеспечение компьютера, поставляемое на кассете, входит программа для контроли работоспособности отдельных узлов компьютера. С ее помощью можно более тщательно проверить компьютер, что (при прохождении всех тестов) является хорошей гарантией его правильной работоспособности.

Как отмечалось, язык Бейсик встроен в ПЗУ (постоянное запоминающее устройство) компьютера. Поэтому он загружается в течение 1-2 секунд после включения компьютера.

Примененная в компьютере версия Бейсика является одной из наиболее удобных. Отличительными чертами ее являются: возможность косвенной адресации при безусловных переходах GO TO и обращениям к подпрограммам GO SUB, расширенные возможности функции VAL (она преобразует в число значение любого арифметического выражения), расширенная система команд, синтаксический контроль строк до их ввода, ввод всех операторов разом (а не по отдельным буквам) и др.

Бейсик - основной язык программирования компьютера «Дельта С». Однако, при решении сложных профессиональных задач можно воспользоваться целым рядом других языков: Лого, Форт, Паскаль, Бета-Бейсик 3.0 и др. К Вашим услугам также ассемблер и дизассемблер, различные компиляторы и отладчики, программы для графических работ, синтеза звуков, решения многих научных, инженерных и учебных задач. «Дельта-С» имеет наиболее мощное программное обеспечение среди известных отечественных бытовых компьютеров.

Микропроцессор Z-80A (и его аналоги), примененный в компьютере, имеет расширенную систему команд, включая команды по перемещению обширных областей ОЗУ (оперативного запоминающего устройства).

Это позволяет реализовать сложные приемы машинной графики, создание быстродвигающихся по экрану графических объектов - графем и спрайтов. На этом основано создание разнообразных игровых программ, число которых достигает 4000 (для аналога ZX - Spectrum)

1. 4. Работа с клавишным пультом

Количество команд, которые можно вводить с клавиатурного пульта, в несколько раз превышает число клавиш. Внешний вид клавишного пульта показан на рис. 2.


TRUE VIDEO
INV VIDEO
DELETE
GRAPH
EXTEND MODE
EDIT
CAPS SHIFT
CAPS LOCK
SYMBOL SHIFT
CAPS SHIFT
SYMBOL SHIFT

Рис. 2

Выводимая функция или символ	Вспомогательная клавиша	Режим (изображение курсора)	Клавиша
	ENTER	K	
BORDER		K	BIN BRIGHT BORDER B * i
*	SYMBOL SHIFT	K	
BIN	EXTEND MODE	E	
BRIGHT	SYMBOL SHIFT	E	
B		C	
b	BP CAPS SHIFT	L	
и	РУССК LAT	L	
И	BP CAPS SHIFT	C	
	CAT) 9 GRAPH	G	
□		G	■ POINT (8
■	BP CAPS SHIFT	G	

Рис. 3

Из него видно, что каждая клавиша имеет до 5-6 надписей. Действие клавиши зависит от нажатия вспомогательных (префиксных) клавиш. Их две:

Simbol Shift (или **SS**) - клавиша переключения дополнительных режимов работы.

CAPS SHIFT - клавиша переключения регистров (верхний /нижний).

На рис. 3 представлены ко бинации клавиш, которые нужно нажимать, чтобы ввести ту или иную команду или символ из числа изображенных на основной клавише. Эти комбинации отображаются также буквой в мигающем курсоре, появляющемся на экране дисплея.

Нажав клавишу **ENTER**, получим изображение курсора с буквой **K** в нижнем левом углу экрана дисплея.

Буква **K** указывает, что компьютер находится в режиме ввода команд. В этом режиме нажатие любой клавиши приведет к вводу команды-слова, написанной на соответствующей клавише. (Например, **BORDER** при нажатии на клавишу B). Другую команду можно ввести, нажав одновременно клавишу **SS** и нужную клавишу. Одновременное нажатие клавиш **SS** и **CS** или **EXTED MODE** вводит еще один режим **E**. В нем также можно ввести одну из двух команд (без нажатия **SS** и с нажатием).

После ввода команд компьютер переключается на ввод символов. При этом курсор содержит букву **L** или **C**, соответствующую вводу либо малых букв (**L**), либо больших (**C**). Клавиша **CS** переключает режимы **L** или **C**.

Нажав одновременно клавиши **CS** и **9** или **GRAF** (с надписью **GRAPHICS**) можно задать еще один режим **G**. Это графический режим вывода графических элементов - графем.

На клавишах 1-8 нанесены знаки-графемы, которые можно выводить в режиме **G**. Таких знаков 16 и он образуют два комплекта графем. Один вводится при прямом нажатии клавиши, другой при одновременом нажатии ее и клавиши **CS**. Например, нажатие клавиши 8 вводит светлый прямоугольник, а клавиш **CS** и 8 одновременно - темный прямоугольник. Для отключения режима **G** надо повторно нажать клавиши **CS** и **9** или **GRAF**.

Совместное нажатие клавиши **CS** и клавиш **5, 6, 7** и **8** задает команды перемещения курсора влево (**5**), вниз (**6**), вверх (**7**) и вправо (**8**) или нажатием соответствующих клавиш ➡⬅ ⬇⬆ Перемещение курсора используется во многих программах.

Можно проверить перемещение курсора вправо и влево, введя в нижнюю сторону экрана любой текст, например PRINT PPPP… и т. д. Курсор будет перемещаться влево при нажатии клавиши **CS** и **5** и вправо при нажатии клавиш **CS** и **8**. Однако область, движения курсора ограничена лишь введенным текстом. Обратите внимание, что слова Бейсика (например,

PRINT) вместе с отделяющим пробелом курсор обегает, как один знак.

Стереть один знак слева от курсора можно, введя команду **DELETE**.

Теперь мы подготовлены к вводу целых предложений, содержащих команды микрокомпьютера. Эти предложения индицируются в двух нижних служебных строках экрана (22 строки сверху являются рабочими). Однако, если двух строк снизу не хватает, то их число автоматически возрастает при уменьшении количества рабочих строк. В каждой стране содержится по 32 знакоместа. Знаки (цифры, буквы, графемы) создаются в знакоместе, матрица которого содержит 8*8 элементов. пикселями.

Поэкспементируйте с вводом и исполнением различных команд. Например, с такими:

PRINT ``QWERTY``	Печатается слово Qwerty
PRINT 1/3	Печатается число 0,33333333
PRINT EXP 2	Печатается e^2 = 7.3890561
PLOT 10,20	Внизу и слева экрана строится точка
CIRCLE 130,80,70	В центре экрана строится круг

На этих примерах Вы убеждаетесь в работе с Бейсиком в режиме непосредственного исполнения команд. С возможностями этого языка можно ознакомиться по дальнейшим разделам данноґо описания.

Отметим лишь, что режим непосредственного исполнения команд резко расширяет диалоговые возможности Бейсика. В самом деле, любую команду или группу команд можно тут же опробовать. Активный эксперимент резко облегчает знакомство с языком. Старайтесь любые задачи (подсчет налогов или партвзносов, суммирование данных таблиц и т. д.) проводить с помощью Вашего компьютера. Вскоре Вы убедитесь, что Ваше мастерство общения с ним быстро возрастает, Вы почувствуете интерес к решению все более сложных и серьезных задач, а не только к использованию игр (хотя и это весьма увлекательное занятие как для детей, так и для убеленных сединой отцов семейств).

1.5 Первые навыки программирования

Сложно ли программировать? Ответ на этот вопрос, неоднозначен. Есть много школьников, ставших виртуозами программирования, несмотря на ограниченный доступ их к компьютерам. Но есть, увы, и люди, проработавшие с вычислительной техникой десятки лет, но так и не освоившие практики программирования.

Этот раздел рассчитан на лиц, которые смутно представляют себе программирование или находятся в блаженном неведении

о таковом вообще.

Программа для компьютера есть просто цепочка команд которые он последовательно, строго и пунктуально выполняет. Возможности нашего компьюстера в этом поразительны каждую секунду он выполняет почти миллион команд и не ошибается, считая в таком режиме многие часы подряд. Разумеется, если заданные ему команды корректны и не содержат ошибок.

Однако, при всей своей скорости и пунктуальности, компьютер - машина довольно глупая - его микропроцессор понимает только двоичные числа 0 и 1 или группы из этих чисел, например 01100011 или 11100101. Каждое такое число, содержащее 8 двоичных знаков (разрядов или битов) называется байтом.

Любая команда задается байтом. Так же задаются данные, символы алфавита, координаты точек графиков и графемы. Однако, задавать данные и команды программ в двоичном виде - дело бесконечно нудное.

Его можно немного упростить, если учесть, что байт имеет и десятичное значение, вычисляемое по формуле:

$$b = (2*d7) + (2*d6) + ... + (2*d1) + (2*d0)$$

где **d0-d7** - двоичные разряды (di=0 или 1).

Например, число 00001011 дает

$$b = (2*0)+(2*0)+(2*0)+(2*0)+(2*1)+(2*0)+(2*1)+(2*1) = 11$$

Байт может иметь $256=2^8$ десятичных значений, от 0 до 255. Десятичная форма задания команд довольно широко применяется опытными программистами для задания команд в машинных кодах. Применяют и шестнадцатеричные числа **HEX**, разряд которых обозначается знаками:

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
DEC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Шестнадцатиричные числа более компакны. чем десятичные. Например, FF эквивалентно числу 255, а FE числу 65534.

Для хранения цепочек байт (данных или команд программ) используется оперативное запоминающее устройство (ОЗУ). Если хранящую байт ячейку уподобить ящику, то ОЗУ- это просто набор ящиков. ОЗУ Вашего компьютера вмещает 48 Кбайт (1 Кбайт это 1024 байт).

Было бы неплохо отдавать команды компьютеру более понятным языком. Бейсик является специальной программой-интерпретатором. Она воспринимает команды в виде английских слов (например, PRINT - печать или INPUT - ввод) и тут же преобразует (транслирует) их в машинные коды, немедленно исполняемые.

Программа на Бейсике состоит из пронумерованных строк

(номера от 0 до 9999), содержащих предложения с командами. Вот пример простой программы, вычисляющей значение e^x по введенному X:

```
10  INPUT "X="; X
20  LET Y = EXP(X)
30  PRINT "Y="; Y
40  GO TO 10
```

Наберите эту программу, не забывая после ввода каждой строки нажимать клавишу **ENTER**. Попробуйте допустить ошибку, например не ввести кавычки- компьютер тут же откажется ввести строку и пометит место ошибки мигающим курсом с вопросительным знаком.

Теперь сотрите текст на экране, дав команду **CLS** и нажав клавишу **ENTER**. Программа, однако, сохраняется в ОЗУ. Дав команду **LIST**. в этом можно убедиться - текст программы тут же появится на экране.

Теперь посмотрим, что он из себя представляет. В строке 10 стоит команда ввода **INPUT**. По запросу в кавычках X= она останавливает счет до ввода X. В строке 20 оператор **LET** присваивает переменной значение **EXP (X)**, а в строке 30 команда **PRINT** печатает результат после вывода сообщения **Y=**. Команда безусловного перехода **GO TO 10** возвращает исполнение к строке 10, то есть вновь к вводу X.

Пуск программы осуществляется командой RUN. Этапы работы следующие:

Показания дисплея	**Проводимая операция**
RUN	Пуск программы
X=2	Запрос x= и ввод числа 2
Y=7.38905599	Печать y = exp (x)
X=	Новый запрос x

Вероятно, задав 3-4 значения X, Вам скоро наскучит вычисление экспоненты. Однако, попытки выйти из программы скорее всего окажутся безуспешными.

Подскажем путь выхода из цикла, заданного командой **GO TO 10**. Для этого в ответ на запрос **X=**, нажав клавиши **CS** и **A/Stop**, введем слово **STOP**. При нажатии клавиши **ENTER** вычисления прервутся.

Более подробные данные о программировании на Бейсике описаны далее - здесь отметим лишь, что стирание строки с заданным номером выполняется указанием ее номера. Полное стирание программы производится командой **NEW**. Команда **EDIT** выводит на редактирование строку листинга, помеченную курсором со знаком >. Вверх и вниз по листингу курсор перемещается с помощью клавиш 6 и 7. нажимаемых совместно с клавишей **CS**. Строка редактируется в процессе ее ввода или вывода на редактирование.

1.6. Работа с кассетным магнитофоном

Если Вы отладили свою программу, то у Вас, естественно, появится желание сохранить ее для использования на следующий день или даже через год. Для этого предназначен кассетный магнитофон компьютера.

Чтобы записать программу и значения переменных ее на магнитную ленту, дайте команду:

SAVE "Имя программы",

например

SAVE "EXP"

для нашей программы. Нажав клавишу **ENTER**, Вы получите предупреждающее сообщение:

Start tape, then press any key

(включите магнитофон, затем нажмите любую клавишу).

Подготовьте магнитофон к записи, вставьте в него пустую кассету и включите протяжку ленты. Затем нажмите любую клавишу компьютера.

Вы услышите примерно пятисекундный звуковой сигнал чистого тона. За это время надо установить нормальный уровень записи. Многие магнитофоны имеют автоматическую регулировку уровня записи (АРУЗ), так что этот сигнал нужен для работы системы АРУЗ. После этого сигнала слышится короткий (доли секунды) шумоподобный сигнал. Это сигнал записи на ленту заголовка программного файла. Он несет информацию об имени программы и содержит контрольный код для проверки правильности программы. Спустя еще примерно секунду появится шумоподобный сигнал - запись блока программы. Запись заканчивается появлением знака **OK** в служебной строке. В процессе записи на обрамлении экрана дисплея видны вначале бегущие полосы, а затем хаотично изменяющиеся полоски.

Вы можете проверить качество записи, дав команду верификации.

VERIFY "Имя программы"

Перемотайте ленту назад и пустите магнитофон в режиме воспроизведения. Если запись прошла удачно, то после считывания программы появится слово **OK**. Сообщение об ошибке указывает на то, что запись неудачна. Это может быть по причине неисправности магнитофона или плохого качества ленты.

Теперь сбросьте Вашу программу, введя команду **NEW** или **RANDOMIZE USR 0** или нажав клавишу **RESET**. Дав команду **LIST**, убедитесь, что программы уже нет в ОЗУ. Теперь, перемотав ленту, дайте команду:

LOAD "Имя программы"

и понаблюдайте за работой компьютера. Он разыщет нужную программу и выполнит её считывание. В процессе поиска печатается перечень файлов, имеющихся на кассете (если, разумеется,

они есть). Далее, как только встречается нужный файл, компьютер начинает его загрузку.

Загрузка контролируется двумя способами. Первый-визуальный. Вы видите появление по краям экрана характерных бегущих линий, точно таких же,что были в процессе записи. Второй способ - звуковой. Динамик компьютера воспроизводит звуковой сигнал - он также аналогичен слышимому при записи.

Система записи и считывания на кассетный магнитофон, примененная в компьютере «Дельта-С», уникальна. Она обеспечивает скорость записи и считывания в 1500 двоичных знаков в секунду - более чем в двое выше, чем, например, у популярных компьютеров «Атари». Система мало чувствительна к колебаниям скорости протяжки магнитной ленты, фону и шумам магнитофона (разумеется, если они не выходят за определенные пределы). Визуальный контроль и вывод перечня встречающихся на ленте файлов очень удобны для пользователя.

Мы рассмотрели лишь простейшие приемы использования магнитофона. Во второй главе Вы сможете более подробно ознакомиться с командами **SAVE**, **VERIFY**, **MERGE** и **LOAD**, обслуживающими кассетный накопитель, и разобраться с тем, как записывать программы с автозапуском, как объединять программы, записывать значения переменных, копии изображения с экрана и т. д.

1.7. Работа с дисплеем

Сам дисплей (или телевизор), если исправен, не требует особого внимания. Разумеется, надо по своему вкусу установить яркость и контрастность изображения. Однако, нередко бывает желание изменить некоторые параметры изображения, например цвет окантовки (бордюра) экрана, цвет знаков или цвет страницы. Некоторые пользователи любят работать с темными знаками на светлом фоне, другие - со светлым и на темном фоне. Как тут быть?

Бейсик компьютера имеет ряд команд, относящихся к управлению изображением. Вот они:

CLS - **очистка экрана**
BORDER n - **задает цвет бордюра с кодом n, где n - число от 0 до 7**
INK n - **задает цвет знаков**
PAPER n - **задает цвет страницы.**

Например, если Вы хотите получить белые знаки на темном общем фоне экрана, задайте

BORDER 0: PAPER 0: INK 7

Напротив, темные знаки на белом фоне получаются при исполнении команд:

BORDER 7: PAPER 7: INK 0

Поэкспериментируйте с этими командами и установите цвета на свой вкус. Компьютер использует следующие коды n для цветов:

0 - черный
1 - синий
2 - красный
3 - оранжевый
4 - зеленый
5 - голубой
6 - желтый
7 - белый

Цвета, разумеется, видны , если применяется цветной телевизор или дисплей. При черно-белом устройстве отображения информации (монохромном) цвета воспринимаются как градации яркости.

Постройте на экране изображение окружности. задав команды

CLS: CIRCLE 128,85,80

Если окружнось выглядит как эллипс, значит Ваш телевизор вносит геометрические искажения. Вы можете скорректировать их, регулируя размеры изображения по вертикали и горизонтали. Постарайтесь получить как можно правильное изображение окружности. Если это не удается - значит телевизор не вполне исправен.

1.8. Работа с принтером

Некоторые компьютеры поставляютс печатающим устройством- принтером. Подключите его к компьютеру и к сети. Командой **LPRINT "цепь знаков"** вы можете вывести на печать цепь знаков, команда **LPRINT 123** даст печать числа 123 и т.д. Команда **LLIST** выводит на печать листинг Вашей программы (если она есть в ОЗУ). Еще одна команда **COPY** позволяет получать копию любого (в том числе графического) изображения, наблюдаемого на экране дисплея (принтер должен работать в режиме графического отображения- плоттер).

ГЛАВА 2. ЯЗЫК ПРОГРАММИРОВАНИЯ БЕЙСИК.

2.1. Общие сведения.

Бейсик - наиболее массовый и популярный язык программирования персональных ЭВМ. Версия Бейсика ПЭВМ «Дельта-С» аналогична версии языка для ПЭВМ ZX Spectrum и имеет ряд особенностей.

В состав этой версии наряду с алфавитом (латинские буквы от a до z, от A до Z, спецзнаки +, -, *, / и др.) входят ключевые слова, т.е. команды, отдаваемые ПЭВМ для выполнения тех или иных операций. Все они нанесены на клавиши ПЭВМ. Ключевые слова вводятся не по буквам, а сразу (см. главу 1 описания), что резко ускоряет ввод программ и сокращает число ошибок.

В основном данная версия является существенно расширенной версией обычного Бейсика. Расширения касаются набора функций, операций над символьными переменными, графики и синтеза звуков. Бейсик встроен в ПЗУ ПЭВМ и загружается в течении 1-2 секунд после включения ПЭВМ или нажатия клавиши Reset.

Бейсик ПЭВМ «Дельта-С» имеет строчный редактор, обеспечивающий контроль синтаксических ошибок в процессе набора строки.

Команды Бейсика подразделяются на *директивы*, *операторы* и *функции*. Директивы - это команды, исполняемые с пульта, а операторы - это команды, исполняемые как с пульта, так и с программы. Все ключевые слова версии являются операторами, либо функциями. Функции обеспечивают преобразование аргумента в результат, например **SQR(x)** преобразует значение x в квадратный корень.

В Бейсик ПЭВМ «Дельта-С» заложена концепция расширения. Минимальный Бейсик не содержит ряда сервисных возможностей автоматической нумерации строк, их перенумерации, трассировки и т.д. Однако с помощью расширений (*Beta-Basic, Mega-Basic* и др.) и специальных программ - редакторов (*Tool Kit, Ultra Kit)* все недостающие возможности можно реализовать. Версия B*eta-Basic 3.0 к п*ример, по своим возможностям заметно превосходит стандартную версию Бейсика ПЭВМ PC дает возможность составлять сложные хорошо структурированные программы с обширными графическими возможностями.

Версия Бейсика ПЭВМ ZX-Spectrum (наряду с ее расширениями и другими версиями) подробно описана в книге Дьяконова В. П. «Применение персональных ЭВМ и программирование на языке Бейсик» (М., Радио и связь: 1989). В ней приведено также множество демострационных и прикладных программ на этой версии.

В связи с этим далее дается краткое описание версии Бейсика ПЭВМ ZX-Spectrum и «Дельта-С» с небольшим числом примеров

2.2. Режимы работы непосредственный и программный

Бейсик выполняет операторы в двух режимах: программном и командном, или непосредственном.

В программном режиме каждой программной строке предшествует номер, указывающий последовательность ее выполнения в программе. Наличие номера означает, что строка входит в программу и по нажатии клавиши ввода ENTER она не будет выполняться, а запишется в память программ. При этом в рабочей части экрана будет воспроизведен ее листинг. Знак >,появляющийся в программном листинге, называется пограммным курсом. Обычно он указывает последнюю введенную строку.

Строка, содержащая программный курсор, называется текущей.

В непосредственном режиме номер у операторной строки отсутствует. По нажатии клавиши ввода ENTER операторная строка выполняется немедленно.

Например:

```
PRINT 3+2
```

В ответ на ввод этой строки в нижней служебной зоне экрана мы получим (после нажатия клавиш ENTER) результат 5 (в верхней части экрана). Это пример работы в непосредственном режиме.

Программа на языке Бейсик представляет собой текстовый набор операторов и функций, объединенных в программные строки. Каждая программная строка может содержать один и более операторов и должна начинаться номером. Номера строк необходимы для управления порядком выполнения операторов и функций строк внутри программы а также для изменения обычного порядка выполнения операторов при условной и безусловной передачи управления (см. далее).

Номер строки должен быть натуральным числом в диапазоне от 1 до 9999.

При первом наборе программы рекомендуется пронумеровать строки с некоторым неединичным шагом, что позволит при отладке программы по необходимости вставлять дополнительные операторы между строками.

Пример:

```
10 LET a=10
15 LET b=15
20 PRINT a+b
```

Набирать строки можно в любом порядке. Бейсик пересортирует их в порядке возрастания номеров, что сразу отражается в листинге на экране.

В одну строку можно помещать несколько операторов, отделяя их друг от друга знаком двоеточия, например:

10 LET a=10: LET = 15: PRINT a+b

При необходимости редактирования служебной строки экрана курсор может быть перемещен вдоль нее:

влево клавишей ⬅

вправо клавишей ➡

Символ перед курсором стирается нажатием клавиши **DELETE**. Вся служебная строка может быть стерта последовательным нажатием клавиш **EDIT** и **ENTER**.

Для внесения изменений в программу можно заменить програмные строки, полностью перепечатав их заново.

Другим способом внесения изменений в программную строку является вызов её на редактирование в служебную строку экрана.

Для этого нужно переместить (используя клавиши перемещения курсора вверх и вниз) программный курсор в нужную строку, сделав тем самым ее текущей, затем нажать клавишу **EDIT**. При этом в служебной сроке экрана будет напечатана копия текущей строки программы, где она может быть отредактирована. По окончании редактирования нажать клавишу **ENTER**.

Отредактированная строка будет помещена в программу на место старой. Другой путь вызова строки на редактирование - указание ближайшего к ней несуществующего номера, например 9 для вызова строки 10.

Для удаления строки из программы следует набрать ее номер и нажать клавишу **ENTER**.

2.3. Элементы данных

Элементы данных Бейсика представлены числовыми и символьными константами, переменными, массивами.

В Бейсике используется два вида констант: числа и строки.

Числовые константы представляются в двоичной форме с плавающей запятой. Один байт используется для показателя степени и четыре байта для мантиссы. Диапазон представления чисел от 4E 39 до 1E 38.

Допустимы два формата записи чисел: нормальный формат (общепринятый) и экспоненциальный формат.

Пример нормального формата записи чисела:

123 1.23 -123

Нормальный формат может иметь до восьми значащих цифр. Целая часть от дробной отделяется точкой. Знак числа ставится первым слева.

Если число [*]<10 или x>=10 , то используется экспоненциальная форма записи числа: **[*]x.xxxxxxxE*xx**
где * - знак числа и порядка, после знака числа идет мантисса, имеющая до восьми значащих цифр. Показатель степени отделен от мантиссы буквой E, за которой стоит знак порядка + или - и два разряда показателя степени.

Пример:

1.2345679E+16

Строковая константа представляет собой набор символов, ограниченный с двух сторон кавычками. При отсутствии символов внутри кавычек строка называется пустой или нулевой: ````
Если внутри строки необходимо записать кавычки, их необходимо удвоить. При распечатке на экране такие двойные кавычки будут напечатаны как одна:

PRINT "Система""""ZX SPECTRUM""""""""

Строковая константа может иметь произвольную длину (насколько позволяет объем свободной области ОЗУ).

Переменные, в отличие от констант, могут иметь значения, меняемые по ходу выполнения программы. Бейсик допускает также два типа переменных: числовые и строковые.

Для присвоения переменным их значений применяется оператор LET. Простые числовые переменные могут иметь имена произвольной длины. В именах можно использовать только буквы и цифры, но *первым знаком в имени должна быть буква*. Пробелы и параметры управления цветом игнорируются, а все буквы преобразуются в строчные.

Управляющие переменные циклов **FOR-NEXT** имеют имена, состоящие из одной буквы. Числовые массивы обозначаются именами, также состоящими из одной буквы, причем эти имена могут совпадать с именами простых переменных.

Имя простой строковой переменной состоит из одной буквы и знака **$**. Имя строкового массива также содержит одну букву с последующим знаком **$**. Причем имена строкового массива и простой строковой переменной не должны совпадать.

Примеры:

LET ALPFA=123 - здесь ALPFA - числовая переменная, 123 - константа, все предложение означает присвоение переменной ALPFA значения 123

LET NAME$ = "ROBERT" - здесь NAME$ - строковая (символьная) переменная, "РОБЕРТ"- строковая константа, все предложение означает присвоение переменной NAME$ символьного значения ROBERT.

Массивом является упорядоченное множество величин одного (числового или строкового) типа, определенное одним именем.

Члены этого множества называются элементами массива. Элемент массива определяется именем массива с последующим списком индексных величин, заключенных в круглые скобки.

Индексы (если их два или более) разделяются между собой запятыми.

Количество индексов должно соответствовать объявленному оператором **DIM** количеству измерений (размерности) массива. Максимальное значение индекса не должно превышать величины соответствующего измерения.

Индексом может быть любое числовое выражение, значение которого должно быть больше или равно единице.

Если индексное выражение не целочисленное - оно округляется до целого.

Количество измерений и максимальное значение каждого измерения (индекса) массива объявляется оператором **DIM**.

Массивы занимают некоторую часть ОЗУ. Для ее резервирования они должны заранее (перед первым применением) объявляться с помощью оператора **DIM**.

Например:

DIM X(10) - объявляет массив из 10 элементов X1,X2… ,X10.

DIM a (2,3) - объявляет двухмерный массив с именем a, содержащий элементы:

a(1,1) a(1,2) a(1,3)
a (2,1) a(2:2) a(2,3)

Возможно задание и многомерных массивов, например:

DIM Y(10,10,10) задает трехмерный массив.

Массивы могут переобъявляться. При этом после каждого объявления элементы числовых массивов принимают нулевые значения. Индексы массивов целые числа, идущие с 1.

Массив строковых величин именуется одной латинской буквой с последующим знаком **$**.

Элемент строкового массива идентифицируется указанием имени массива, непосредственно за которым в круглых скобках следует список целочисленных индексов, разделенных запятыми.

При объявлении строкового массива последняя величина в списке измерений устанавливает длину элементов массива.

Пример:

DIM a$ (5,10)

устанавливает массив a$ из пяти строковых переменных длиной 10 символов каждая:

a$(1) = a$(1,1) a$(1,2) ... a$(1,10)
a$(2) = a$(2,1) a$(2,2) ... a$(2,10)
.....
a$(5) = a$(5,1) a$(5,2) ... a$(5,10)

Для вызова полной строки в списке индексов элемента массива опускается последний индекс, соответствующий в операторе **DIM** длине строки.

При полном списке индексов в имени переменной она вернет один символ строки, порядковый номер которого в строке задается значением последнего индекса.

Необходимо помнить, что в строковом массиве все строки имеют одинаковую длину. Поэтому в операции присвоения в случае недостаточной длины произойдет заполнение свободных мест пробелами справа, в случае превышения длины произойдет отрезание лишних символов справа до длины, определенной в операторе **DIM**.

Пример:

```
10 DIM a$(5,10)
20 LET a$(1)="1234567890"
30 LET a$(2)="abcdefghiklm"
40 LET a$(3)="abc"
50 PRINT a$(1), a$(1,7), a$(2), a$(3); "      "
60 STOP
```

На экран будет выведен следующий результат:

```
1234567890                7
abcdefghij                abc
```

Надо помнить, что в отличие от числового типа для строкового массива и простой строковой переменной недопустимо применение одинакового имени.

2.4. Выражения

Под выражением на Бейсике подразумевается некоторая запись, указывающая на характер выполняемых операций с использованием значений констант и переменных (числовых или строковых).

Так арифметическое или числовое выражение является эквивалентом математических формул, записываемых в строку с явным указанием всех операций. Например: 2*A*B есть выражение на Бейсике формулы 2AB. Обратите внимание на то, что применяемые в формуле 2AB по молчанию знаки умножения (2AB есть фактические (2*A*B) в выражении на Бейсике указываются явно (знаком *)

Для записи числовых выражений в Бейсике используются следующие знаки арифметических операций:

^ - возведение в степень **+ - сложение**
*** - умножение** **- - вычитание**
/ - деление

Последовательность вычисления выражений определяется приоритетом операций. Высший приоритет имеет операция возведения в степень (^). За ней следуют унарный минус (смена знака числа), затем умножение (*) и деление (/), имеющий равный приоритет: низший и равный приоритет имеют сложение (+) и вычитание (-).

При равном приоритете выполнение операций осуществляется слево направо.

Пример:

a=b*c +d^e

Сначала вычисляется d^e затем b*c и в последнюю очередь производится сложение.

Приоритет операции

Знак операции	Приоритет	Число операндов	Название операции
^	10	2 числовых	возведение в степень
-	9	1 числовой	унарный минус
*	8	2 числовых	умножение
/	8	2 числовых	деление
+	6	2 числовых	сложение
-	6	2 числовых	вычитание

Если в арифметическом выражении содержатся функции, то они вычисляются первыми.

Нарушить приоритет операций можно, используя круглые скобки, причем допустимо их многократное вложение. Сначала обрабатываются выражения, взятые в скобки. При многократном вложении первым вычисляется самое внутреннее выражение, затем внешнее по отношению к нему и т.д. до тех пор, пока не будет вычислено выражение в крайних скобках.

Пример:

f=10*(15 + 5^(1 + 2))

Вычисление правой части выражения будет осуществляться в следующем порядке:

1. 1 + 2
2. 5^3
3. 15+125
4. 10*140

При записи числовых выражений необходимо помнить следующее:

1. Два знака арифметических операций нельзя располагать рядом, за исключением комбинации какого-либо знака операции с унарным минусом или плюсом, например выражение 2*(- 1) правомерно записать в виде: 2*-1.

2. Для операции возведения в степень (^) левый операнд (основание степени) должен быть положительным числом.

Строковым выражением является запись, содержащая один или несколько операндов объединенных знаками конкатенации (объединение).

К строковым операндам относятся любые данные строкового типа: константы, переменные, функции, а также любые другие выражения, принимающие значения строкового типа.

Операция конкатенации обозначается знаком + и служит для объединения строковых операторов.

Значением строкового выражения, состоящего из двух строковых операндов и знака является константа, равная значению первого операнда, за которым следует значение второго операнда без каких-либо символов между ними.

Пример

```
10  LET f$= "con":      LET b$= "stant"
20  LET c$=f$+b$:       PRINT c$
```

Результат, выводимый на экран:

constant

Операция конкатенации имеет приоритет, равный 6.

Выражение отношения представляет собой запись двух выражений одного типа (числового или строкового), разделенных знаком операции отношения.

Для записи выражения отношения используются следующие знаки операций отношения:

Знак операции	Приоритет	Число операндов	Операция
=	5	2 однотипных	равно
>	5	2 однотипных	больше
<	5	2 однотипных	меньше
<=	5	2 однотипных	меньше или равно
=<	5	2 однотипных	больше или равно
<>	5	2 однотипных	не равно

Примечание: двойные знаки <=, >=, <> набираются как единый символ нажатием соответствующей клавиши. Попытка сформировать двойной знак из двух одиночных символов сообщение об ошибке.

Результатом выражения отношения является целое число 0, либо 1.

Значение выражения =1 («истина»), если отношение сравниваемых величин выполняется, и значение выражения равно 0 («ложь»), если отношение не выполняется.

Пример	Результат, выводимый на экран
PRINT 1 <> 2	1

```
PRINT "a" = "b"
PRINT "abc" < "abc"
```

В операциях отношения сравнение строковых величин осуществляется посимвольно слево направо. Сравнение идет до выявления двух первых несовпадающих символов.

Затем сравниваются как обычные числа коды этих несовпадающих символов (согласно кодовому набору системы ZX SPECTRUM): наибольший код соответствует большей строковой величине.

Две строки равны, если они имеют одинаковое количество символов и в одинаковых позициях содержатся одни и те же символы.

При сравнении строк разной длины короткая строка дополняется справа пробелами.

Поскольку результатом операций сравнения является число, выражения отношения могут участвовать как числовые величины во всех конструкциях Бейсика, где это допустимо.

Логическим выражением является формульная запись, формирующая из операндов и знаков логических операций. Логические операции осуществляют преобразования операндов соответственно правилам логики.

Значение логического выражения может быть числового и строкового типа.

Знаки логических операций

Знак операции	Приоритет	Число и тип операндов	Операция
NOT	4	1 числовой правый	отрицание
AND	3	правый числовой, левый числовой или строковый	конъюнкция
OR	2	2 числовых	дизъюнкция

Операция отрицания **NOT** является одноместной: имеет один операнд числового типа. Результатом операции является число 0 или 1.

Определение операции **NOT**:

$$\mathbf{NOT\ x} = \begin{cases} 0, \text{ если } x <> 0 \\ 1, \text{ если } x = 0 \end{cases}$$

Здесь x - любое выражение числового типа, в том числе выражение отношения.

Пример	Результат, выводимый на экран
PRINT NOT 1	0
PRINT NOT 0	1
PRINT NOT "a" <> "b"	0
PRINT NOT -12e34	0

Операция конъюнкции **AND** (логическое умножение) осуществляется над двумя операндами, причем правый операнд всегда должен иметь числовой тип, левый операнд может быть как числового, так и строкового типа. Тип результата операции определяется типом левого операнда.

Определение операции AND:

1. Левый операнд x - выражение числового типа.

$$x \text{ AND } y = \begin{cases} x, \text{ если } y <> 0 \\ 0, \text{ если } y = 0 \end{cases}$$

2. Левый операнд - выражение строкового типа.

$$a\$ \text{ AND } y = \begin{cases} a\$, \text{ если } y <> 0 \\ "", \text{ если } y = 0 \end{cases}$$

y - выражение числового типа.

Пример	Результат, выводимый на экран
PRINT "abc" AND 123	abc
PRINT "a+b" AND 1<>2	ab
PRINT 1234 AND "a = b"	0
PRINT 0 AND 0	0
PRINT 0 AND 1	0
PRINT 1 AND 0	0
PRINT 1 AND 1	1

Операция дизъюнкции **OR** (логическое сложение) осуществляется над двумя операндами числового типа. Результатом операции является число.

Определение операции **OR**:

$$x \text{ OR } y = \begin{cases} 1, \text{ если } y <> 0 \text{ и/или } x <> 0 \\ 0, \text{ если } x = y = 0 \end{cases}$$

Здесь x,y - выражение числового типа.

Пример	**Результат, выводимый на экран**
PRINT 1=0 OR 1<>0	1
PRINT 0 OR 0	0
PRINT 0 OR 1	1
PRINT 1 OR 0	1
PRINT 1 OR 1	1

2.5. Операторы ввода - вывода и их модификаторы

При решении любых задач нужно вводить в ПЭВМ данные и выводить их на экран дисплея (или на печать принтера). В данной версии Бейсика для этого служат операторы **PRINT** и **INPUT** и их модификаторы (элементы расширения).

Оператор **PRINT** выводит на экран дисплея значения элементов, перечисленные в списке вывода оператора.

Формат оператора:

PRINT список вывода

Элементами списка вывода могут быть:

- константы числовые и строковые
- переменные числовые и строковые
- функции строкового и числового типа
- выражения строкового и числового типа
- функции позиционирования печати **AT** и **TAB**
- операторы управления цветовым режимом
- управляющие символы

Перед выводом на экран элемента списка вычисляется его значение, а затем результат вычисления выводится на экран. Вывод осуществляется в порядке перечисления элементов слева направо.

Элементы в списке вывода оператора **PRINT** отделяются друг от друга *знаками-разделителями.* К ним относятся: *точка с запятой* (;),*запятая*(,) и *апостроф* (`)

Запятая используется для установки начал нечати в следующей зоне, т.е в начале строки (графа 0), либо в ее центре (графа 16), в зависимости от того, где до этого находился курсор.

Точка с запятой выводит на печать следующий элемент списка сразу за предыдущим.

Апостроф устанавливает начало печати следующего за ним элемента списка в начало следующей строки экрана.

Пример:

```
10 LET a=1: LET b=2: LET c=3
20 PRINT a,b,c
30 PRINT "a+b+c="; a+b+c
40 PRINT
50 PRINT "a;b;c"
60 PRINT a;b;c
70 PRINT a`b`c
```

Результатом работы операторов **PRINT** в этой программе будет следующая распечатка на экране:

```
1                    2
3
a+b+c=6

a;b;c                123
1
2
3
```

Предложение 20 распечатывает каждый элемент списка переменных, разделенных запятыми, в начале новой зоны.

Предложение 30 выводит на экран строковую константу и следом за ней в текущем знакоместе печатает значение выражения, стоящего за знаком (;).

Оператор PRINT в программной строке 40 оставляет пустую строку на экране.

В состав оператора PRINT могут входить модификаторы **AT** и **TAB**. Модификатор **AT (x,y)**, где x - номер строки и y - номер столбца, обеспечивает вывод с позиции (x,y), Например:

```
PRINT AT (10,12); "HELLO"
```

выводит надпись "HELLO" сразу в середину экрана (на десятую строку). Значения x==0-21 (т.е. допустимы 22 строки с номерами от 0 до 21), а y=0-31 (т.е. возможны 32 столбца с номерами от 0 до 31).

Модификатор горизонтальной табуляции **TAB (n)** перемещает вывод на n позиций по горизонтали. Применяется в виде

```
PRINT TAB n; "HELLO"
```

Здесь n (как и x и y в модификаторе AT) может быть константой, числовым значением или значением переменных. Используется только целая часть соответствующего значения.

В тексты программ можно вводить комментарии, которые выводятся только при выводе листинга программы (см. раздел 2.6). Для этого используется оператор **REM**. Например:

10 REM CALCULATION ERF (X) (Вычисление ERF (x))

Предложение с этим оператором должно быть последним или единственным в строке.

Оператор **INPUT** осуществляет присвоение введенных с клавиатуры значений поименным переменным списка ввода и вывода на экран значений элементов списка вывода.

Формат оператора **INPUT**:

INPUT список элементов

В список элементов оператора **INPUT** могут входить 3 вида элементов:

1. Элементы ввода - это элементы, которым оператор **INPUT** присваивает значения, вводимые с клавиатуры.
   К элементам ввода относятся:
   - имя числовой переменной,
   - имя строковой переменной;
   - **LINE** имя строковой переменной;

   Элементы ввода должны начинаться буквой или функцией LINE.
2. Элементы вывода-элементы, значения которых оператор **INPUT** вычисляет и печатает на экране подобно оператору **PRINT**.
   К элементам вывода относятся:
   - константы числовые и строковые;
   - выражения числового и строкового типа, заключенные в круглые скобки.

   Элементы вывода не должны начинаться буквой.
3. Модификатор печати AT. Отменяет работу в служебной строке и устанавливает позицию начала печати следующего за ней элемента по указанным координатам.

Элементы списка оператора **INPUT** разделяется между собой знаками-разделителями. К ним относятся: *точка с запятой* (;), *запятая* (,) и *апостроф* (') они позиционируют печать аналогично оператору **PRINT**.

Ввод-вывод элементов списка оператор **INPUT** осуществляет последовательно слева направо в порядке перечисления элементов.

Работа оператора **INPUT** отображается в служебной строке экрана. Отменяет ввод или вывод элемента в служебной строке функция **AT**.

В режиме вывода оператор **INPUT** работает аналогично оператору PRINT.

В режиме ввода оператор **INPUT**, встретив в списке элементов имя переменной, переходит в режим ожидания (курсор L на экране). Для ввода даных необходимо набрать с клавиатуры зна-

чение переменной (константу или выражение) и по окончании набора нажать клавишу **ENTER**.

По нажатии клавиши **ENTER** Бейсик присвоит введенное значение переменной. Если при наборе допущена синтаксическая ошибка, ввод будет блокирован и в строке ввода на ошибку укажет знак «?»

При вводе символьной переменной по оператору **INPUT** Бейсик печатает две строковые кавычки, помещая между ними курсор. Это освобождает пользователя от печати кавычек и является напоминанием о типе вводимых данных.

Особенностью работы оператора **INPUT** является возможность ввода не только строковых констант, но и выражений. Для этого необходимо удалить кавычки (сдвинуть курсор L вправо клавишей <> и дважды нажать **DELETE**) и ввести выражение строкового типа.

Пример:

```
10 LET a$="a"
20 INPUT b$
30 PRINT b$
```

Если при вводе по оператору **INPUT** напечатать в кавычках символ "b", это значение будет присвоено переменной b$ и распе чатано на экране. Если же при вводе удалить кавычки н напечатать имя a$, то оператор **INPUT b$** срабатывает как оператор присвоения **LET b$ = a$** и на экране будет напечатан символ a.

Для отмены вывода кавычек при вводе оператор **INPUT** используется совместно с модификатором **LINE**, например

```
20 INPUT LINE N
```

В ряде случаев ввод большого числа данных удобно задавать в виде их списка, вгзючаемого в текст программы. Для этого используются операторы **DATA**, **READ** и **RESTORE**.

Оператор **DATA** (данные) задает блок данных в виде:

```
DATA C1, C2, …,Cn
```

где C1, C2 … Cn - список данных в виде констант (числовых или строковых). Этот оператор может стоять в любом месте программы. Можно использовать несколько таких предложений, например:

```
10 DATA 1, 2, 3
20 DATA 4, 5, 6
```

Оператор READ в форме

```
READ VI, V2, …, Vn
```

Обеспечивает просмотр списка операторов **DATA** и переменным VI, V2, …, Vn соответствующих значений (числовых - числовым переменным и строчных - строчным). Порядок следования переменных и их тип должны строго следовать порядку данных.

Рассмотрим пример ввода данных с помощью операторов DATA и READ:

```
10 READ a,b,c
20 READ d,e,f$
30 PRINT a'b'c'd'e'f$
40 DATA 1,2,3,4,5, "end"
50 STOP
```

Оператор READ в строке 10 осуществляет последовательное считывание значений из оператора DATA строки 40 и присвоение их переменным списка:

a=1, b=2, c=3.

Следующий оператор READ (строка 20) начнет считывание с четвертой позиции блока данных и присвоит переменным следующие значения: d=4, e=5, f$="end". На экране будут распечатаны значения переменных:

1
2
3
3
5
end

После того, как блок весь считан последующая попытка чтения из него вызовет сообщение об ошибке.

Для повторного использования блока данных указатель блока должен быть установлен в начало списка данных. Это выполняется оператором RESTORE.

Формат оператора RESTORE:

RESTORE n

где n - номер строки оператора DATA ,на начало которого устанавливается указатель блока.

Если в операторе RESTORE отсутствует номер строки, то указатель блока данных устанавливается на начальный элемент первого в программе оператора DATA.

Пример:

```
10 READ x,y: PRINT x;y
20 RESTORE
30 READ a,b : PRINT a;b
40 RESTORE 70
50 READ c,d,e,f: PRINT c,d,e,f
60 DATA 0,1,2,3,4,5,6,7,8,9
70 DATA 1,2,3,5,8
80 STOP
```

Оператор READ (строка 10) читает и присваивает два первых числа из оператора DATA строки 60: x=0, y=1. Предложение 20 RESTORE возвращает указатель в начало блока данных и последующий оператор READ строки 30 считывает и присваивает те же числа другим переменным: a=0, b=1.

Оператор **RESTORE 70** (строка 40) перемещает указатель блока данных к начальному элементу блока данных в строке 70 и начиная с этого элемента следующий оператор **READ** (строка 50) считывает данные и осуществляет присвоение: c=1, d=2, e=3, f=5. В результате работы программы на экран будут выведены значения переменных:

01
01
1235

Таким образом Бейсик обеспечивает обширные возможности по вводу данных.

2.6. Директивы управления

С помощью описанных выше операторов уже можно решать на ПЭВМ некоторые важнейшие задачи (например те, что решаются на калькуляторах). Но для этого полезно знать некоторые директивы (операторы), позволяющие управлять ПЭВМ. Ниже они перечислены.

Директива **NEW** стирает все данные и программы и используется обычно перед вводом новой программы. При ее применении происходит полная очистка экрана и той части ОЗУ, которая лежит ниже установленного потолка **RAMTOP**.

Директива
CLEAR n
устанавливает потолок ОЗУ. Например, если задать **CLEAR 40000** , то область ОЗУ с ячейками памяти от 40001 до 65535 будет защищена от очистки командой **NEW**, а нижележащие области О.ЗУ будут очищаться. Директива **CLEAR** без n может использоваться для очистки переменных (их значения анулируются.

Директива
RUN n
где n - номер строки, запускает программу со строки n.Директива **RUN** (без n) пускает программу начиная со строки с наименьшим номером.

Директива
STOP
примененная в любом месте программы ведет к остановке вычислений в этом месте (с указанием в служебной строке где произошла остановка).

Директива
CONTINUE
обеспечивает продолжение выполнения программы с того места, где произошла остановка.

Команда

BREAK

(исполняемая только вводом с пульта) обеспечивает немедленное прерывание работы по программе. Исключением являются циклы **FOR-NEXT** (см.далее), исполнение которых командой **BREAK** не прерывается.

Оператор

PAUSE

приостанавливает вычислительный процесс на время воспроизведения указанного числа телевизионных кадров.

Формат оператора:

PAUSE n

где n - число кадров; может принимать значение натурального числа в диапазоне 0 <= n <=65535. Время паузы в секундах можно определить по формуле n/50 (в секунду воспроизводится 50 кадров).

Прерывается действие оператора **PAUSE** нажатием любой клавиши.

Если n = 0, то пауза будет длиться как угодно долго - до первого нажатия какой-либо клавиши.

Директива

LIST

распечатывает на экране текст текущей программы, находящейся в памяти.

Формат команды:

LIST nc

где nc - номер начальной строки листинга.

Если номер не указан, программа будет распечатываться начиная с минимального номера.

Распечатка текста производится постранично. Размер страницы составляет 22 строки экрана. После заполнения экрана очередной страницей текста, ввод приостанавливается и в служебной строке появляется сообщение:

scroll?
("перемещение?")

Нажатие одной из клавиш **N**, **BREAK** или **STOP** осуществит остановку программы с сообщением **D BREAK-Cont repeat**.

Любая же другая клавиша (кроме трех указанных) продолжает вывод листинга на экран.

2.7. Управляющие структуры

Для создания программ с разветвляющейся структурой используются управляющие структуры. Их три типа: безусловные переходы, условные выражения и циклы.

Оператор безусловного перехода:

GO TO n

где n - число (константа), имя переменной или арифметическое выражение обеспечивает безусловный переход к строке с номером n (или ближайшей строке снизу, если строки с номером n не оказалось). Итак, допустимы записи:

GO TO 100 **(переход к строке с номером 100)**
GO TO n **(переход к строке с номером --значением n)**
GO TO 2*50 **(переход к строке 2*50=100, заданной ариф метическим выражением).**

Оператор IF THEN осуществляет передачу управления в программе в зависимости от значения (истинно или ложно) выражения отношения.

Формат оператора:

IF условие THEN строка операторов

где условие - выражение отношения; может быть как арифметическим, так и строковым,

строка операторов - один или несколько операторов, разделенных двоеточием, выполняемых в случае истинности проверяемого условия.

Работа оператора IF THEN осуществляется следующим образом: если указанное условие (выражение отношения, стоящее между словами IF и THEN) истинно, то выполняются операторы оставшейся части строки, стоящие за словом THEN. В противном случае, если условие ложно, операторы после слова THEN игнорируются и управление передается строке, следующей за строкой с оператором IF THEN.

Пример:

```
10 INPUT a,b
20 IF a<b THEN PRINT a: STOP
30 PRINT b
```

Данная программа выбирает из двух вводимых чисел меньшее и печатает его.

Если условие в строке 20 истинно (a<b), то выполняется группа операторов, стоящих за THEN: печатается значение a и происходит останов по оператору STOP с сообщением

```
9 STOP. Statement, 20:3
(останов по третьей команде строки 20)
```

В противном случае, когда условие a<b не выполняется, операторы PRINT a: STOP оставшейся части строки 20 игнорируются и выполняется строка 30 PRINT b

Как видно из этих примеров, оператор **STOP** также можно рассматривать, как элемент управляющих структур, обеспечивающий остановку вычислений в заданном месте программы.

Для организации многократного выполнения одной и той же последовательности действий используются операторы организации циклов **FOR** и **NEXT**. Они всегда используются вместе, обрамляя тело цикла.

Заголовком цикла является оператор **FOR**. Его формат:

FOR a = A1 TO A2 STEP A3

где a - управляющая переменная цикла: должна иметь в своем имени только одну букву.

A1, A2, A3 - арифметические выражения, определяющие соответственно начальное значение (A1) переменной a, конечное значение (A2) и шаг (A3), с которым изменяется переменная a.

Все три параметра цикла могут быть числовыми константами, именами числовых переменных, арифметическими выражениями и числовыми функциями.

Цикл заканчивается оператором **NEXT**. Его формат

NEXT a

При каждом выполнении цикла текущее значение управляющей переменной a (начиная с начального A1) сравнивается с конечным значением A2.

Если a>A2 (при A3>0) или a<A2 (при A3<0), то цикл прекращается и происходит переход к следующей за оператором **NEXT** команде.

В противном случае выполняется тело цикла, после чего переменная a получает приращение A3 и цикл повторяется.

Пример:

```
10 FOR i = 1 TO 10 STEP 2
20 PRINT i;
30 NEXT i
40 PRINT "конец примера"
```

В этом примере программа выводит на экран нечетные числа в диапазоне от 1 до 10:

```
13579
конец примера
```

При входе в цикл управляющей переменной i присваивается начальное значение (i=1), сравнивается с конечным значением 10, т.к. условие i<=10 истинно, то выполняется тело цикла (строка 20) Затем в операторе **NEXT i** переменная i получает приращение (i==i+2); новое значение сравнивается с конечным и процесс повторяется до значений i=11, при котором тело цикла не выполняется и происходит выход из цикла к строке 40.

Если начальное A1 и конечное A2 значения управляющей переменной a выбраны неверно (например A1>A2 при A>0), то тело цикла не выполняется ни разу.

Недопустима передача управления внутрь цикла, минуя оператор **FOR**. При этом команда **NEXT** не будет воспринята и Бейсик выведет сообщение об ошибке.

Значение шага в операторе цикла может быть как положительным, так и отрицательным. Недопустимо задание шага = 0, т.к. при этом повторение станет бесконечным. Если в заголовке цикла отсутствует **STEP A3**, то приращение по умолчанию будет равно 1. Так, в результате работы программы:

```
10 FOR i=0 TO 9
20 PRINT i
30 NEXT i
```

На экран будет выведена строка цифр:

```
0123456789
```

Возможно применение вложенных друг в друга циклов - внутренний цикл должен заканчиваться до окончания внешнего.

Пример правильного построения вложенных циклов

```
10 FOR i=1 TO 10
20 FOR j=1 TO 10
30 INPUT a
40 NEXT j
50 NEXT i
```

Неправильного:

```
10 FOR i=1 TO 10
20 FOR j=1 TO 10
30 PRINT i; j
40 NEXT i
50 NEXT j
```

2.8. Подпрограммы

Часто бывает нужно произвольное число раз обращаться к отдельным фрагментам программ. Удобно вынести их в подпрограммы. Впрочем, они могут размещаться как под основной программой, так и над ней.

Оператор **GOSUB** осуществляет передачу управления подпрограмме.

Его формат:

```
GOSUB n
```

где n - номер строки, с которой начинается подпрограмма, выражение числового типа.

Оператор **RETURN** (используется без аргумента) осуществляет выход из подпрограммы в точку вызова подпрограммы, а именно к оператору, следующему за вызовом подпрограммы **GOSUB n**.

При выполнении оператора **GOSUB** его адрес заносится в специальное запоминающее устройство - стек возвратов. При этом содержимое стека "утапливается", а последний адрес хранится верхушкой стека. По оператору **RETURN** верхний адрес выбирается (набор данных в стеке при этом "всплывает", и управление передается оператору, следующему за этим адресом.

В программе вычисления числа сочетаний C_n^m из n элементов по m:

$$C_n^m = \frac{n!}{m!(n-m)!}$$

целесообразно вычисления факториала оформить подпрограммой:

```
10 INPUT "Введите числа N,M""  N="; N, "M="; M
20 LET K = N: GOSUB 100: LET A = F
30 LET K = M: GOSUB 100: LET B = F
40 K=N-M: GOSUB 100: LET C = F
50 LET D = A/B/C
60 PRINT "результат="; D: STOP
100 REM Вычисление факториала
110 LET F = 1
120 FOR I = 1 TO K
130 LET F = F*I
140 NEXT I
150 RETURN
```

Подпрограмма выполнения факториала К! располагается в строках 100-150. Обращение к ней происходит трижды: для вычисления N!, M! и (N-M)!, причем перед этим каждый раз переменной К присваивается соответствующее значение, а результат работы подпрограммы запоминается в переменной F. При возврате из подпрограммы это значение присваивается новой переменной (последовательно A, B и C) для запоминания.

2.9. Функции числовые

Бейсик содержит набор встроенных наиболее часто встречающихся математических и строковых функций, а также средства создания собственных функций пользователя.

Для обращения к функциям необходимо набрать имя функции и аргумент или список аргументов, если их несколько, в качестве которых могут использоваться константы, переменные, элементы массивов, другие функции и выражения. Для некоторых функций ограничивается диапазон изменения значений их аргументов.

Если аргументом является выражение, то его необходимо заключить в круглые скобки. В случае, если аргумент - одиночная константа, переменная или функция, круглые скобки можно опустить.

Функции используются в выражениях так же, как константы и переменные. Результатом вычисления функций является вещественное число или строковое значение.

Встроенные функции вещественного типа **Таблица**

Имя функции	Описание функции
ABS	Определение абсолюбной величины числа
SGN	Определение знака числа (сигнум)
INT	Определение целой части числа
SQR	Вычисление квадратного корня из числа
EXP	Вычисление экспоненты: возведение константы e (2,718281828) в степень, равную значению аргумента
LN	Вычисление натурального логарифма числа
SIN	Вычисление тригонометрического синуса числа
COS	Вычисление тригонометрического косинуса числа
TAN	Вычисление тригонометрического тангенса числа
ASN	Вычисление арксинуса числа
ACS	Вычисление арккосинуса числа
ATN	Вычисление арктангенса числа
PI	Числовая константа, равная 3,1415...
RND	Генератор случайных чисел
BIN	Преобразование десятичного числа в двоичное

Функция SGN имеет вид

SGN a

где a - аргумент функции, является выражением вещественного типа.

Значение функции определяется следующим образом:

SGN a = +1, если a > 0
SGN a = 0, если a = 0
SGN a = -1, если a < 0

Пример:	**Результат, выводимый на экран**
PRINT SGN 0	**0**
PRINT SGN (2,5-7,8)	**-1**
PRINT SGN VAL "1e2"	**1**

Функция ABS имеет вид:

ABS a

где a - аргумент функции; является выражением вещественного типа. Значение функции равно абсолютной величине аргумента и определяется следующим образом:

ABS a = a, если a >= 0
ABS a = -a, если a<0

Пример:	**Результат, выводимый на экран**
PRINT ABS 0,5	**0,5**
PRINT ABS SIN (-PI/2)	**1**
PRINT ABS -3,2	**3,2**

Функция INT имеет вид:

INT a

где a- аргумент функции, является выражением вещественного типа. Значением функции является целая часть значения аргумента. Напомнии,что целой частью числа является наибольшее целое число, не превышающее данного числа.

Пример:	**Результат, выводимый на экран**
PRINT INT 3,8	3
PRINT INT (3,2+ 7,1)	10
PRINT INT -3,2	-4
PRINT INT (-9-0,9)	-10

Функция SQR имеет вид:

SQR a

где a - аргумент, являющийся выражением вещественного типа, может принимать только положительные и нулевое значение. Значение функции равно квадратному корню из значения аргумента.

Примеры:	**Результат, выводимый на экран**
PRINT SQR 2	1,4142136
PRINT SQR INT 25,6	5
PRINT SQR (100 +5^3)	15

Если аргумент окажется отрицательным, на экран будет выведено сообщение об ошибке.

Формат функции **PI**

Формат функции RI является числовоой константой. Ее значение

PI = 3.1415927

Пример	**Результат, выводимый на экран**
PRINT PI	3.1415927
PRINT SIN PI	0
PRINT TAN (PI/4)	1

Функция **RND** порождает случайное число.

Значением функции является псевдослучайное число в диапазоне (0, 1). Аргумента функция не имеет.

При многократном использовании функции **RND**, например в цикле, она будет генерировать различные последовательности случайных чисел.

Пример

```
10 FOR i =1 TO 5
20 PRINT RND
30 NEXT i: PRINT
```

Эта программа при двух запусках распечатает на экране различные последовательности чисел.

Например:

```
0.37870789
0.40379333
0.28517151
0.38867188
0.15107727
0.33175659
0.88250732
0.18817139
0.11376953
0.53371192
```

Генерация чисел функцией **RND** осуществляется не совсем случайным образом, т.к. ее значения выбираются из фиксированной последовательности 65536 чисел.

Для генерации случайного числа в диапазоне (a,b) используется выражение:

a + (b - a)*RND

Например, следующая программа будет выводить на экран последовательности случайных целых чисел в диапазоне от 1 до 10:

```
10 FOR i=1 TO 5
20 PRINT 1 + INT (9*RND);
30 NEXT i : PRINT
40 GOTO 10
```

Результат, выводимый на экран

```
38988
23147
57246
76313
```

Для реализации запуска функции **RND** с определенного случайного числа используется оператор **RANDOMIZE**. На клавиатуре он обозначен аббревиатурой **RAND**.

Его формат:

RANDOMIZE n

где n - числовое выражение, которое может принимать любое целое положительное значение в диапазоне от 1 до 65535.

Этот параметр определяет точку запуска последовательности случайных чисел, генерируемых функций **RND**.

Так, в следующей программе после каждого выполнения оператора **RANDOMIZE** 1 последовательность, вырабатываемая функцией **RND**, заново запускается с числа 0022735596:

Пример:

```
10 RANDOMIZE 1
20 FOR i=1 TO 5: PRINT RND, : NEXT i
30 PRINT: GO TO 10
```

Результат, выводимый на экран:

```
0022735596          0.17164612
0.87440491          0.58050537
0.53837585
.0022735596         0.17164612
```

И т.д.

Бейсик содержит набор встроенных функций, которые являются реализацией наиболее часто встречающихся математических функций.

Обращение к функции осуществляется набором ее имени и аргумента. Например, **SIN X, COS (PI/2)** и т.д.

Встроенные математические функции относятся к числовому типу и имеют вещественный аргумент.

Аргумент функции может быть константой, переменной или выражением. В последнем случае его необходимо заключать в круглые скобки.

Функции **SIN** определяет синус аргумента, заданного в радианах. Значение функции заключается в интервале от -1 до +1.

Функции **COS** определяет косинус аргумента, заданного в радианах. Значение функции заключается в интервале от -1 до +1.

Функция **TAN** определяет тангенс аргумента, заданного в радианах.

Функция **ASN** определяет арксинус аргумента. Значение функции выражается в радианах и заключается в интервале -PI/2 до +PI/2. Аргумент может изменяться в интервале от -1 до +1

Функция **ACS** определяет арккосинус аргумента. Значение функции выражается в радианах и заключается в интервале от PI до 0. Аргумент может изменяться в интервале от -1 до +1.

Функция **ATN** определяет арктангенс аргумента. Значение функции выражается в радианах и заключается в интервале от -PI/2 до + PI/2.

Функция **EXP** определяет значение алгебраической константы (в математике принято обозначать буквой ***e***), возведенной в степень, равную аргументу функции. Значение константы

e = 2.7182818.

Функция LN определяет натуральный логарифм (логарифм по основанию e) аргумента. Аргумент функции может принимать только положительные значения.

Пример	Результат, выводимый на экран
PRINT EXP 1	2,7182818
PRINT LN(2,7- 2)	1,9865035
PRINT SIN (PI/2)	1
PRINT ATN 1*180/PI	45

Функция BIN используется для представления натуральных чисел в двоичном коде.

Ее формат:

BIN n

здесь n - число, записанное в двоичном коде (последовательность нулей и единиц). Сама функция не является преобразователем числа; она лишь указывает на двоичный формат его записи.

Аргумент может принимать значение в диапазоне от 0 до 65535, т.е. разрядность двоичного кода не должна превышать 16.

2. 10. Функции строковые

Большое число функций предназначено для работы с данными строчечного типа.

В Бейсике ПЭВМ «Дельта-С» реализован специальный прием для выделения из строки символов подстроки. Для этого используется функция - модификатор TO в виде:

строка (n1 TO n2)

где строка может быть строковой константой, именем строковой переменной или строковым выражением;

n1 - порядковый номер символа в обрабатываемой строке, которым начинается подстрока;

n2 - порядковый номер символа строки, который является последним в подстроке;

n1 и n2 - числовые выражения; могут принимать только положительные целые значения.

Пример:

PRINT "abcdef" (2 TO 4) **bcd**

Если отсутствует n1, то по умолчанию этот параметр принимается равным 1, например:

PRINT "abcdef" (TO 5) **abode**

Если функция не содержит n2, то конец подстроки определяется концом строки:

PRINT "abcdef" (3 TO) **cdef**

Очевидно, что если отсутствуют оба параметра (n1 и n2), то подстрока совпадает со строкой:

```
PRINT "abcdef" (TO)          abcdef
```

Если n1=n2, то подстроке присваивается один символ, например:

```
PRINT "abcdef" (3 TO 3)      c
```

То же самое получается, если в скобках присутствует только один параметр:

```
PRINT "abcdef" (3)           c
```

Начальная (n1) и конечная (n2) координаты должны ссылаться на существующие символы строки. Исключение составляет случай, когда начальное значение больше конечного (n1 > n2). Тогда подстрока оказызается пустой. В этом случае не будет сообщения об ошибке, даже если n1 больше длины строки. Если подстрока выделяется из строкового выражения, это выражение необходимо заключить в скобки.

Пример:

```
PRINT "abc" +"def"(1 2)                abcdef
PRINT ("abc" + "def")(1 TO 2)          ab
```

Пример:

```
        10 REM Действие функции TO
        20 LET f$ = "abcdefgh"
        30 FOR i = 1 TO 8
        40 PRINT f$ (1 TO i)
        50 NEXT i
ab
abc
abcd
abcde
abcdef
abcdefg
abcdefgh
```

С помощью функции TO возможно не только выделение подстроки, но и изменение значения исходной строки. Это осуществляется присвоением подстроке нового значения. При этом длина строки не меняется. Если новое значение короче подстроки, то оно дополняется справа пробелами, если длиннее - то обрезается справа до длины указанной подстроки.

Пример:

```
        10 LET a$="Я система ZX SPECTRUM"
```

```
30 LET a$ (3 TO 9) = "*******"
40 PRINT a$
50 LET a$ (3 TO 9) = "-"
60 PRINT a$
70 STOP

Я система ZX SPECTRUM
Я******* ZX SPECTRUM
Я
ZX SPECTRUM
```

Функция LEN определяет длину строки, т. е. число символов в значении строкового аргумента.

Формат функции:

LEN аргумент

Аргументом функции является строковое выражение, значением функции целое десятичное число.

Пример:

```
10 LET a$= "absdef"
20 LET b$ = "a b"
30 PRINT LEN a$, LEN b$, LEN (a$ + b$)
```

Результат выполнения программы:

```
6          4          10
```

Функция STR$ преобразует числовое значение аргумента в строку символов.

Формат функции: **STR$ аргумент**

Аргументом функции является выражение числового типа. Функция STR$ вычисляет значения аргумента и полученное число возвращает в виде символьной строки в формате оператора PRINT.

Пример

```
PRINT STR$ 1e2                  100
PRINT LEN STR$ 100,000          3
```

Функция VAL возвращает числовое значение выражения, заданного строкой. Ее формат

VAL аргумент

Аргументом функции является строка представляющая собой запись числового выражения, а результат вычисления этого выражения является значением функции VAL.

Если аргумент является сложным строковым выражением, его необходимо заключить в круглые кобки.

Пример:	**Результат, выводимый на экран**
PRINT VAL "3.5"	3.5
PRINT VAL "2*3"	6
PRINT VAL ("2" + "*3")	6

В последнем примере сначала вычисляется аргумен функции VAL как строка; его результатом является значение «2*3».

Затем снимаются кавычки и то, что остается, вычисляется как число: 2*3=6.

Если значением строкового выражения является пробел, то выдается сообщение об ошибке.

Функция **VAL$** вычисляет строковое значение аргумента и возвращает результат в виде строки.

Формат функции: **VAL$ аргумент**

Аргументом функции является строковое выражение (константа, имя строковой переменной, строковая функция или сложное строковое выражение). В свою очередь значение этого строкового аргумента также должно представлять собой запись строкового выражения.

Вычисление функции **VAL$** проходит в два этапа: на первом этапе вычисляется значение строкового аргумента и с него «снимаются» строковые кавычки. В результате этой операции должна быть получена запись строкового выражения. На втором этапе вычисляется полученное строковое выражение; его результат, являющийся строковой константой, и есть значение функции **VAL$**.

Необходимо помнить, что при многократном вложении кавычек каждый раз их должно быть в 2 раза больше, чем на предыдущем вложении.

Пример:

```
10 INPUT a$
20 PRINT VAL$ a$
30 STOP
```

Эта программа иллюстрирует действие функции **VAL$**.

Значение аргумента a$	**Результат, выводимый на экран**
"""a"""	a
"""""""привет!"""""""	"привет"
"STR$ 1e2"	100
"""a"" + ""b"""	ab

Функция **USR** определяет адрес первого байта массива, хранящего изображение символа.

Формат функции: **USR аргумент**

Аргумент функции имеет строковый тип и должен представлять собой одиночный символ, заключенный в кавычки. Это может быть любая буква от a до u, либо графический символ, определенный пользователем - символ, имеющий в таблице кодов системы значение от 144 до 164 включительно.

Значение функции - адрес первого байта массива, хранящего изображение символа - аргумента.

Функция **CODE** определяет десятичное значение кода первого символа аргумента.

Формат функции: **CODE аргумент**

где аргументом функции является выражение строкового типа. Если аргумент имеет значение "пусто", то функция имеет значение 0.

Значением функции является положительное целое число в диапазоне от 0 до 255 - десятичный код первого символа аргумента согласно таблице кодирования символов системы ZX Spectrum.

Пример:	**Результат, выводимый на экран**
PRINT CODE "a"	**97**
PRINT CODE "a"	**32**
PRINT CODE "123"	**49**
PRINT CODE ""	**0**

Функция **CHR$** определяет символ , соответствующий заданному коду.

Формат функции **CHR$ n**

Аргументом функции может быть выражение числового типа, принимающее целое положительное значение в диапазоне от 0 до 255. Если значение аргумента является дробным числом, оно округляется до ближайшего целого. Значением функции является символ, соответствующий заданному коду согласно таблице кодирования символов системы ZX Spectrum.

Пример:	**Результат, выводимый на экране:**
20 FOR a= 65 TO 67	**A**
20 PRINT CHR$ a	**B**
30 NEXT a: STOP	**C**

Функция **SCREEN$** читает символ на экране монитора по указанным координатам экрана.

Формат функции **SCREEN$**: **SCREEN$ (n,m)**

где n - числовое выражение; указывает номер строки экрана; может принимать целые значения в диапазоне от 0 <=n <= 23;

m -числовое выражение; указывает номер графы экрана; может принимать целые значения в диапазоне от 0 < m <= 31.

Значением функции **SCREEN$** является символ, который выведен на экране в позиции, указанной координатами n,m.

Если по указанным координатам расположены графические символы пользователя, а также фрагменты изображений, выведенные операторами **PLOT**, **DRAW**, **CIRCLE**, **OVER**, то значением функции **SCREEN$** будет нулевая (пустая) строка.

Функция **INKEY$** вырабатывает символьное значение нажатой в момент ее применения клавиши.

Так, в программе

```
10 PAUSE 0
20 LET F$ = INKEY$
30 PRINT F$
40 GO TO 10
```

При нажатии любой клавиши печатается соответствующий символ. Эту функцию удобно использовать при создании меню.

2. 11. Функции, определяемые пользователем

Чтобы избежать неоднократного повторения одинаковых формульных записей в различных точках программы, Бейсик позволяет определить новые функции, составленные самим пользователем, и обращаться к ним в программе, как к стандартным функциям Бейсика.

Функция пользователя определяется оператором **DEF**.

Его формат:

DEF FN name (список формальных аргументов) = выражение

где **FN name** - полное имя определяемой функции; **FN-** обязательное начало имени; **name** - одна латинская буква для функции числового типа или одна буква и знак **$** для строковой функции.

Список аргументов - имена одного или нескольких формальных параметров функции (аргументов), разделенных между собой запятыми; в имени параметра числового типа может использоваться только одна латинская буква, для строкового типа - одна буква с последующим знаком **$**. Кроме того, функция может вообще не иметь аргументов.

В отличие от стандартных функций Бейсика, список аргументов функции пользователя обязательно заключается в круглые скобки, даже если этот список пуст.

Выражение - формульная запись определяемой функции, включающая формальные параметры. Выражение может включать и переменные, которых нет в списке формальных аргументов. Эти переменные при вызове функции имеют присвоенное им в данный момент значение.

Операторы **DEF** могут располагаться в любом месте программы. Реализуется оператор **DEF** в расширенном режиме нажатием двух клавиш **SIMBOL SHIFT** и **1**. При этом автоматически выдается предложение **DEF FN**.

Тип выражения должен соответствовать типу функции.

Для вызова функции набирается ее полное имя с последующим списком фактических аргументов, разделенных запятыми, список обязательно заключается в круглые скобки. Функция имеет вид:

FN name (список фактических аргументов)

Фактические аргументы списка должны соответствовать по порядку, числу и типу соответствующим формальным параметрам в определении функции.

При обращении к функции в списке фактических аргументов могут быть константы, имена переменных, функции и выражения. Бейсик вычисляет значение выражения в операторе **DEF**, заменяя формальные аргументы функции соответствующими фактическими значениями из списка аргументов в обращении к функции.

При наборе имени функции слово **FN** печатается автоматически нажатием двух клавиш **SIMBOL SHIFT** и 2 в режиме E.

Пример:

следующая программа будет вычислять квадратный корень из суммы квадратов двух чисел.

```
10 DEF FN S (a,b) = SQR (a*a+b*b)
20 INPUT x, y
30 PRINT FN (x, y): STOP
```

В этой программе при вызове функции **FN S** формальные аргументы a, b заменяются соответственно значениями переменных x, y, для которых вычисляется выражение в операторе **DEF** и его значение присваивается функции **FN S**. Например, для x= 4, y=3 на экран будет выведен результат равный SQR (3*3 + 4*4) = 5

2. 12. Общение с памятью и портами микропроцессора

Бейсик ПЭВМ «Дельта -С» обеспечивает прямой доступ к ячейкам ОЗУ с помощью команд **PEEK** и **POKE**.

Функция **PEEK** в виде **PEEK A**

где A - адрес ячейки ОЗУ (от 0 до 65535) обеспечивает считывание содержания с указанным адресом. В каждой ячейке ОЗУ расположен байт, т.е. число от 0 до 2^8-1 (или от 0 до 255).

Оператор **POKE** в виде **POKE A,b**

заносит значение байта b от 0 до 255 в ячейку ОЗУ с адресом A (бесполезно задавать адреса A, характерные для ПЗУ, т.к. в этом случае действие оператора **POKE** просто игнорируется).

С помощью оператора **POKE** в ОЗУ можно вводить программы в машинных кодах. Нередко такие программы обеспечивают ускорение вычислений в десятки раз! Однако, необдуманное применение оператора **POKE** может резко нарушить работу ПЭВМ - вплоть до ее "зависания" и отказа от выполнения команд. К физическому отказу это не ведет, но может испортить программу, которую Вы отлаживаете много часов подряд.

Микропроцессор ПЭВМ имеет также множество так называемых портов - каналов для передачи информации. Порты также указываются адресом A. Для общения с портами служат операторы **OUT** и **IN**.

Оператор **IN A**

считывает данные с порта A, а оператор

OUT A, n

задает данные n по адресу порта A.

Значения b и n можно задавать и со знаком минус. При этом к таким значениям надо добавить 256. Итак, можно вместо 255 вводить -1, что короче.

2. 13. Операторы управления экраном и цветом

Символьный экран дисплея (телевизора) представляет собой прямоугольник, содержащий 768 позиций (знакомства для символов):

24 строки, в каждой строке 32 позиции (графы).

Каждая позиция представляет собой точечный квадрат размером 8*8 точек.

Нумерация строк идет сверху вниз от 0 до 23. Нумерация граф - слева направо от 0 до 31.

Таким образом, каждая позиция экрана однозначно определяется двумя координатами: строка, графа. Кроме координат экрана позиция символа характеризуется дополнительными четырьмя атрибутами:

- цвет бумаги
- цвет чернил
- режим яркости
- режим мерцания.

Окраску переднего плана позиции (цвет символа) будем называть чернильным цветом или чернилами, окраску фона в позиции будем называть бумагой или бумажным цветом.

Каждому воспроизводимому цвету ставится в соответствие числовой код.

Таблица цветовых кодов

Код	Воспроизводимый цвет
0	черный
1	синий
2	красный
3	пурпурный
4	зеленый
5	голубой
6	желтый
7	белый
8	прозрачный
9	контрастный

Примечание: На черно-белом экране коды будут располагаться в порядке увеличения яркости.

Код 8 (прозрачный) - означает, что значение указанного атрибута не изменяется.

Код 9 (контрасный) - означает смену цветового атрибута на контрастный по сравнению с другим по следующему правилу: цвет устанавливается белым, если другой темный (черный, синий, красный, пурпурный), цвет черным - если другой цвет светлый: зеленый, голубой, желтый, белый.

Операторы управления цветовым режимом

Управление атрибутами и режимами цветного воспроизведения символа экрана осуществляют операторы **PAPER**, **INK**, **BRIGHT**, **FLASH**, **INVERSE**, **OVER**, **BORDER**.

Формат операторов: **ОПЕРАТОР n**

где оператор - один из перечисленных операторов;

n - код, устанавливающий один из режимов цветового воспроизведения экрана; является выражением числового типа; принимает целые неотрицательные значения.

Перечисленные операторы управляют режимами работы верхней (рабочей) части экрана; служебные строки (22 и 23) и граница экрана не попадают в сферу их действия.

Операторы могут использоваться к качестве элементов операторов **PRINT**, **LPRINT**, **INPUT**, **DRAW**, **PLOT**, **CIRCLE**, с последующем разделителем (;), но в этом случае их действие будет временным и будет распространяться только до конца оператора, который их содержит. Включение их в список оператора **INPUT** дает возможность устанавливать и менять режим воспроизведения в нижней части экрана, но только на время действия оператора **INPUT**.

Оператор **PAPER** устанавливает бумажный цвет (фона) символа, согласно таблице цветовых кодов.

Формат оператора: **PAPER n**

где n - код цвета; целое число в диапазоне 0 <= n <= 9

Оператор **INK** устанавливает чернильный цвет - цвет букв и знаков, последовательно печатаемых элементов, согласно таблице цветовых кодов.

Формат оператора: **INK n**

где n - код цвета; целое число в диапазоне 0 <= n <= 9

Оператор **BRIGHT** управляет яркостью последовательно печатаемых символов.

Формат оператора: **BRIGHT n**

где n - код режима яркости; целое число, принимающее 3 значения;

n=0 - нормальная яркость символа

n=1 - повышенная яркость символа;

n=8 - сохраняется предыдущий режим яркости.

Оператор **FLASH** определяет режим мерцания символов, последовательно печатаемых на экране.

Формат оператора: **FLASH n**

где n - код режима мерцания; может принимать 3 значения:

n=0 - мерцание отсутствует

n=1 - есть мерцание

n=8 - сохраняется предыдущий режим мерцания

Оператор **INVERSE** устанавливает режим инверсии последовательно печатаемых символов.

Формат оператора: **INVERSE n**

где n - код режима инверсии. Может принимать два значения:

n = 0 - нет инверсии; символы печатаются при обычном видеосигнале: чернильный цвет на бумажном цвете;

n = 1 - инверсия включена; символы печатаются при инверсном видеосигнале: бумажный цвет на чернильном цвете.

Оператор инверсии управляет не атрибутом символа, а образами его точек. Включение инверсии означает: точки бумажного цвета будут заменены на точки чернильного цвета, а точки чернильного цвета окрасятся в бумажный цвет.

Значение цвета чернил и цвета бумаги сохраняются.

Оператор **BORDER** устанавливает цвет границы экрана согласно таблице цветовых кодов.

Формат оператора: **BORDER n**

где n - код цвета; числовое выражение, принимающее целое значение в диапазоне 0 <= n <= 7

Нижняя часть экрана, содержащая служебную строку, и его окаймление (бордюр) в качестве бумажного цвета имеет цвет границы экрана, устанавливаемый оператором **BORDER**.

Чернильный цвет служебной строки экрана устанавливается кодом 9 (контрастность) и является либо белым (при темных тонах бумаги), либо черным (при светлых тонах бумаги), при обычной яркости мерцания и без надпечаток. Таков постоянный режим служебной строки. Он изменяется, если в список оператора **INPUT** в качестве элементов списка включены операторы управления цветовым воспроизведением **PAPER, INK, BRIGHT, FLASH, INVERSE, OVER**. Эти операторы в списке **INPUT** устанавливают видеорежим только в течение работы оператора **INPUT**. Затем восстанавливается постоянный режим.

Оператор **OVER** управляет надпечатками для последовательно печатаемых символов.

Формат оператора: **OVER n**

где n - код режима надпечатки; принимает 2 целых значения 0 и 1; если n = 0, при печатании новых символов старые символы в занимаемых позициях стираются; по умолчанию принимается n=0 если n=1, при печатании новых символов, они смешиваются со старыми: новый символ печатается поверх старого.

Операторы **INK, PAPER** и **BORDER** при самостоятельном применении действует глобально (т.е. на весь экран). Однако, можно обеспечить локальное действие операторов **INK** и **PAPER** на любое знакоместо экрана. Для этого их используют как модификаторы оператора вывода **PRINT**. Например, следующая команда

PRINT INK 2; PAPER 6; FLASH 1; "HELLO"

дает вывод на экран слова HELLO красным цветом (INK 2), на желтом фоне знакомест (PAPER 6) в режиме мерцания (FLASH 1).

Функция **ATTR** определяет число, двоичная форма которого указывает атрибуты знакоместа экрана по заданным координатам.

Формат функции: **ATTR (x, y)**

где x,y - числовые выражения, значения которых задают координаты позиции экрана;

x - задает номер строки экрана; является целым числом в диапазоне 0 <= x <= 23;

y - задает номер графы экрана; является целым числом в диапазоне 0 <= y <= 31;

Двоичная запись значения функции дает атрибуты позиции с координатами (x, y) экрана

7 бит указывает код режима мерцания
6 бит указывает код режима яркости
5-3 биты содержат двоичный код бумажного цвета
2-0 биты содержат двоичный код чернильного цвета.

2. 14. Операторы машинной графики

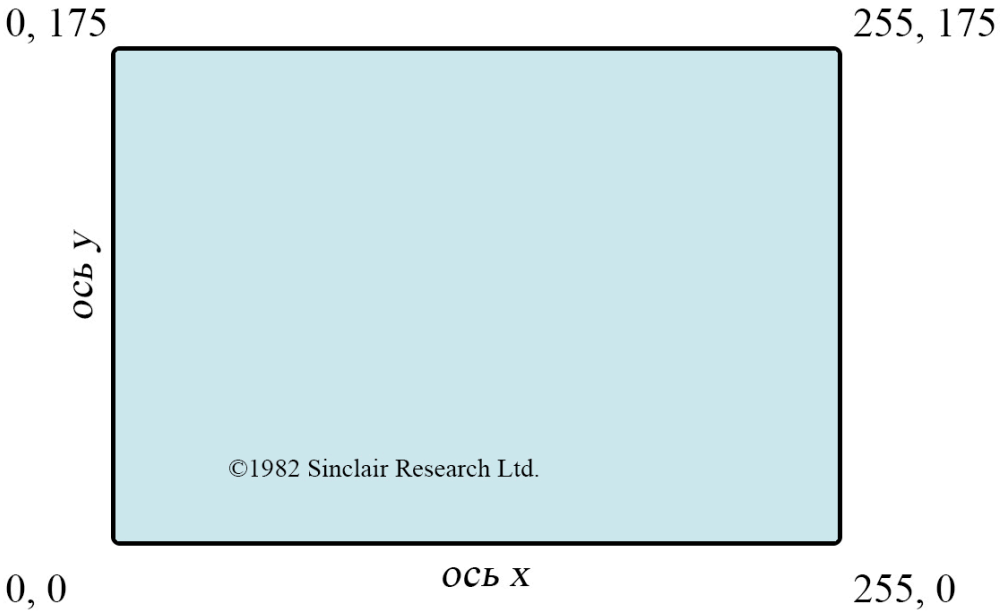
Графический экран - часть экрана видеомонитора, используемая для построения графических изображений. Занимает верхнюю часть символьного экрана из 22 строк (с 0 до 21) и 32 графы (с 0 до 31), что составляет 22*32=704 символьные позиции. Каждая символьная позиция образуется точечной матрицей 8*8 точек, которые называются элементами изображений (pixels). Таким образом, размеры графического экрана равны 176*256.

Каждому элементу изображения (точке) экрана ставится в соответствии два числа (x, y) - координаты точки на экране.

Первая координата x определяет расстояние от точки до левой границы графического экрана, вторая координата y определяет расстояние от точки нижней границы графического экрана.

Начало графических координат расположено в левом нижнем углу графического экрана и имест координату (0,0).

Координаты левого верхнего угла	(0, 175)
Координаты правого нижнего угла	(255, 0)
Координаты правого верхнего угла	(255, 175)



Операторы **RUN**, **CLEAR**, **CLS** и **NEW** устанавливают текущую позицию графика в начало координат (в точку графического экрана с координатами 0, 0- левый нижний угол).

Графические операторы Бейсика позволяют строить на экране монитора геометрические фигуры: устанавливать точку, рисовать отрезки прямых линий, дуги и окружности.

Помимо основных параметров, опереляющих работу графических параметров (координаты точки) в список параметров могут входить операторы управления цветовым режимом **PAPER**, **INK**, **FLASH**, **BRIGHT**, **INVERSE** и **OVER**.

Они располагаются между ключевым словом и координатами и разделяются занятой, либо точкой с запятой. Следует помнить, что операторы управления цветом закрашивают весь символ (т.е. всю символьную позицию) и не могут быть указаны для отдельных элементов изображений. Когда вычерчивается элемент изображения он представляется в своем черном цвете, а вся символьная позиция, содержащая этот элемент, дается в текущем бумажном цвете.

Оператор **PLOT** устанавливает точку (элемент изображения) на экране по указанным координатам.

Формат оператора: **PLOT c; m,n**

где c - список операторов управления цветовым режимом: может быть пустым (т.е. без c);

m - горизонтальная координата экрана (ось x);

0 <= m <= 255

n - вертикальная координата экрана (ось y);

0 <= n <= 175;

Оператор **PLOT** перемещает текущую позицию графика в элемент изображения с координатами m, n и печатает там пятно чернильного цвета.

Чернильный цвет в позиции символа, содержащего данный элемент изображения, изменяется на текущей постоянный чернильный цвет, а остальные цветовые атрибуты остаются без изменения (если только элементы списка с не задают другой режим).

Пример:

```
10 FOR n = TO 255
20 PLOT n, 88+80*SIN (n/128*PI)
30 NEXT n
```

Программа выводит на экран график одного периода (0,2PI) функции SIN.

Формат оператора **DRAW** для построения дуг

DRAW c;x,y,r

где c - список операторов управления цветовым режимом; может быть пустым;

x - приращение координаты x;

y - приращение по координате y;

r - угловой параметр дуги (r может отсутствовать)

Оператор **DRAW** вычерчивает отрезок прямой линии от текущей позиции графического экрана до вновь установленной позиции, где координата новой позиции = координате предыдуей позиции + приращение.

Пример:

PLOT 0,100: DRAW 80,-35

Рисует отрезок прямой между точками (0,100) и(80.65).

Если r присутствует, то значение r задает радиус дуги.

При r = 0 дуга вырождается в отрезок прямой, при r = PI получаем дугу в половину окружности:

PLOT 100,100: DRAW 50, 50, PI

при r > 0 поворот дуги идет против часовой стрелки, а при r < 0 по часовой стрелке.

Построение окружности обеспечивает оператор **CIRCLE**
Он задается в виде:

CIRCLE c; x, y, r

где c - список операторов управления цветовым режимом; может быть пуст;

x, y, r - выражения числового типа;

x, y - абсолютные координаты центра окружности;

r - радиус окружности.

Значения x, y и r должны быть такими, чтобы окружность вмещалась в рабочую зону экрана (иначе появится сообщение об ошибке). Пример применения оператора **CIRCLE**:

CIRCLE 130, 85, 80

- строится большой круг с центром в точке (130,85) и радиусом 80 пикселей.

Функция **POINT** определяет тип цвета (чернильный или бумажный) элемента изображения по графическим координатам.

Формат функции: **POINT (x, y)**

где x, y - выражения числового типа; определяют графические координаты элемента изображения; лежат в диапазоне 0<=x<=255, 0 <= y <= 175.

Функция принимает значение 0, если элемент изображения бумажного цвета, и значение 1, если элемент изображения чернильного цвета. В первом случае это означает, что точки нет на экране, во втором - что точка есть.

2. 15. Синтез звука

Бейсик ПЭВМ «Дельта-С» дает единственную возможность для синтеза звуковых сигналов с заданной длительностью t и высотой тона h. Она реализуется оператором **BEEP**.

BEEP t, h

где t лежит в пределах от 0 до 10 с, а h - от (-60) до (+69). Значение h задает число полутонов до ноты «до» первой октавы (h=0).

Регулировка громкости и тембра звука не предусмотрена. Однако с помощью специальных программ в машинных кодах можно обеспечить создание довольно сложных музыкальных мелодий.

2. 16. Графические символы пользователя.

Алфавит системы ZX Spectrum содержит 21 символ, конфигурация которых в графическом режиме может быть изменена пользователем .Это литеры английского алфавита от a до u, занимающие в кодовом наборе системы позиции от 144 до 164 включительно. По умолчанию при включении машины эти символы в графическом режиме устанавливаются в виде букв от a до u соответственно обозначениям на клавишах.

Образ символа формируется квадратом 8*8 точек, каждая из которых может иметь бумажный или чернильный цвет. Точка, окрашенная в бумажный цвет (фон) кодируется нулем, точка чернильного - цвета единицей.

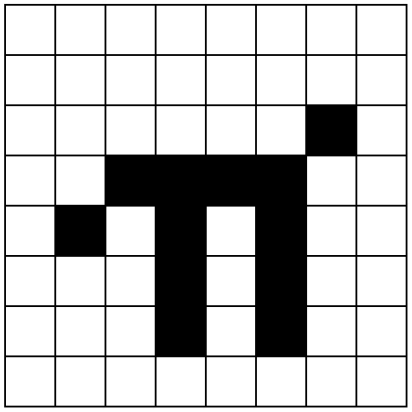
Полный образ символа, таким образом, формируется из восьми строк по 8 точек каждая. Каждой точечной строке ставится в соответствие восьмиразрядное двоичное число. Эти восемь двоичных байтов, несущие образ символа, хранятся в смежных восьми ячейках памяти машины, причем первая ячейка хранит верхнюю строку образа, следующадресу ячейка - вторую сверху строку образа и т.д. в порядке увеличения адреса на 1.

Запись байта в ячейку осуществляет оператор **POKE**; адрес первой (верхней) ячейки задается значением функции **USR** от символьного аргумента. Этот символьный аргумент задается клавишей, выбранной для синтезируемого знака. Адрес каждой следующей ячейки получается прибавлением единицы к адресу предыдущей ячейки. Каждое число, заносимое в ячейку, задается как функция **BIN** с последующим восьмиразрядным двоичным кодом точечной строки.

Пример:

Выберем клавишу P для синтеза греческой буквы Пи.

Рисунок Точечного образа буквы Пи



Двоичный код соответствующей строки образа

BIN 00000000
BIN 00000000
BIN 00000010
BIN 00111100
BIN 01010100
BIN 00010100
BIN 00010100
BIN 00000000

Адрес верхней строки образа задается значением функции **USR "P"**, второй строки сверху- значением **USR "P"** +1 и т.д., восьмой строке соответствует адрес **USR "P"+7**.

Следующая программа запишет образ символа Пи в память машины:

```
10 DATA BIN 0, BIN 0, BIN 00000010, BIN 00111100
20 DATA BIN 01010100,BIN 00010100, BIN 000101000, BIN 0
30 FOR n= 0 TO 7
40 READ i: POKE USR "P" +n, i
50 NEXT n
```

После прогона программы, клавиша P в графическом режиме будет выводить на экран символ PI.

Имеются специальные программы, облегчающие создание графических символов пользователя и даже полную замену алфавита ПЭВМ программным путем.

2. 17. Работа с кассетным магнитофоном

В состав аппаратных средств системы «Дельта-С» входит кассетный магнитофон, позволяющий записывать, считывать и хранить информацию на магнитной ленте.

Бейсик системы содержит программные средства записи на магнитную ленту в виде файла и чтения из него информации в четырех форматах: программы и переменные (вместе); числовые массивы, строковые массивы, машинные коды (двоичные байты).

Любому блоку данных (файлу), хранящемеся на ленте, присваивается имя. Имя файла заключается в кавычки и состоит из печатных символов американского стандартного кода **ASCII** может содержать от 1 до 10 символов… Если при обращении к магнитной ленте в команде отсутствует имя файла, то с ленты будет считываться первый файл указанного типа, независимо от его имени.

Для записи программ и данных на магнитную ленту используется оператор **SAVE** в следующих видах:

SAVE "Имя" - записывается вся программа и относящиеся к ней данные.

SAVE "Имя" LINE n - записывается вся программа и относящиеся к ней данные с указанием признака об автостарте со строки n (см. **LOAD**).

SAVE "Имя" DATA a () - записываются данные массива a () под указанным именем.

SAVE "Имя" CODE A, n- записывается под указанным именем массив из n машинных кодов с начальным адресом A.

SAVE "Имя" SCREEN$ - записывается (в машинных кодах) содержимое экрана.

Считывание файлов и их загрузка в ОЗУ осуществляется оператором **LOAD**. Он применяется в ряде форм:

LOAD "" - считывается первая программа и относищиеся к ней данные

LOAD "Имя" - считывается только программа с указанным именем и относящиеся к ней данные (имена других файлов отображаются на экране дисплея).

LOAD "Имя" DATA m ( ) - считывается массив данных с указанным именем и заполняется массив m ( )

LOAD "Имя" CODE A, n - считываются данные в виде порции n машинных кодов с начальным адресом A (A и n хранятся в заголовке файла).

LOAD "Имя" CODE A1, n - указанные выше данные считываются, но помещаются по начальному адресу A1.

Оператор **VERIFY** (проверка) позволяет проверить правильность записи, сличая ее с данными, хранящимися в ОЗУ ПЭВМ. Он применяется сразу после использования оператора **SAVE**. Возможны следующие формы применения оператора **VERIFY**:

VERIFY "Имя" - проверка считываемой под указанным именем программы.

VERIFY "Имя" DATA a ( ) - проверка считываемого массива со значением элементов массива a ( )

VERIFY "Имя" CODE A, n - проверка n байт с начальным адресом A.

VERIFY "Имя" CODE A - проверка всех байт считываемого файла с указанным именем со значениями байт начиная с адреса A.

К программе, хранящейся в ОЗУ, можно подсоединить другую программу, считав ее с ленты с помощью команды:

MERGE "Имя"

Если у объединяемых программ совпадают номера строк или имена переменных, то старые строки и переменные стираются и на их место записываются новые строки и переменные, загружаемые с магнитной ленты. Для байтов и массивов оператор **MERGE** не используется.

2. 18. Операторы управления принтером и другими устройствами

Если к ПЭВМ подключен принтер, то для управления им можно воспользоваться командами:

LLIST n - распечатка программы, начиная со строки n.

LLIST - распечатка всей программы.

LPRINT - печать данных (аналогично **PRINT**, но с выводом не на экран дисплея, а на принтер).

COPY - печать графической изображения с экрана.

В некоторых моделях ПЭВМ для управления принтером используются специальные программы драйверы. Они запускаются командой вида

RANDOMIZE USR A

где адрес A указывается в описание драйвера. Аналогично запускаются и программы, записанные в машинных кодах (иногда возможен запуск командами **PRINT USR A**, **GO TO USR A** или **GO SUB USR A**).

На клавиатуре ПЭВМ имеется и ряд других команд: **OPEN#, CLOSE#, MOVE, ERASE, CAT** и **FORMAT**. Они используются для обслуживания устройств (интерфейсов, микродрайверов, дисковых накопителей, которые не входят в нормальный комплект поставки ПЭВМ «Дельта-С».

Поэтому подробное описание этих команд опускается.

ГЛАВА 3. ДОПОЛНИТЕЛЬНЫЕ ДАННЫЕ

3.1. Распределение памяти

В данной главе мы рассмотрим некоторые дополнительные данные, полезные опытным программистам. Вероятно, на первых порах, Вы вполне обойдетесь без них.Однако, по мере усложнения решаемых задач эти данные несомненнно понадобятся.

Начнем с рассмотрения карты распределения памяти компьютера «Дельта-С». Как отмечалось, память компьютера, содержащая области ПЗУ и ОЗУ, представляется в виде 65536 ячеек - байт. Каждая ячейка имеет свой адрес в виде числа от 0 до 65535. Ниже приведена таблица распределения адресного пространства компьютера (указывается начальный адрес каждой области):

0	ПЗУ с интерпретатором языка Бейсик
16384	Файлы дисплея
22528	Атрибуты изображения (коды цветовых эффектов)
23296	Буфер принтера
23552	Системные переменные (см.3.2)
23734	Буфер микродрайва (в «Дельта-С» не используется)
CHANS	Информация о каналах передачи информации
PROG	Начало Бейсик - программы
VARS	Область переменных Бейсик - программы
ELINE	Команды строчного редактора
WORKSP	Входные данные и временная рабочая область.
STKROT	Вычислительный стек (начало)
STKEND	Конец стека
SD	Машинный стек
RASTOP	Высшая граница памяти (память с RAMTOP сохраняется при подаче команды NEW, RAMTOP устанавливается командой CLEAR, RAMTOP).
UDC	Область для хранения графических элементов пользователя графем.
FRAMT	Конец памяти (65535).

3.2. Системные переменные

Область ОЗУ от ячейки 23552 до ячейки 23734 отведена под системные переменные. Эти переменные обозначаются мнемоническими именами, например **PROG** - системная переменная, хранящая адрес ячейки ОЗУ, с которой размещается Бейсик-программа. Не следует путать системные переменные с командами Бейсика. Значения многих системных переменных (см.3.1) - просто адреса соответствующих ячеек ОЗУ. В этом случае системная переменная есть 2 ячейки, например, с адресами n и n1. Тогда адрес - значение системной переменной можно найти, исполнив команду

PRINT PEEK n+256*PEEK n1

Значения ряда системных переменных можно менять. Так, если системная переменная имеет значение байта b и размещена по адресу A, то команда

POKE A, b

позволяет занести новое значение b по адресу A.

Для двухбайтных значений можно применять команду

POKE A,V-256*INT (V/256)
POKE A +1, INT (V/256)

Здесь V - значение системной переменной.

Ниже в таблице приведены все системные переменные компьютера «Дельта-С». Указано: число байт, занятых системной переменной, начальный адрес ее, мнемоническое имя и содержание.

Таблица системных переменных

Обозначение	Адрес	Имя	Содержание
1	2	3	4
N8	23552	KSTATE	Используется при чтении клавиатуры
N1	23560	LAST K	Запоминает вновь нажатую клавишу
1	23561	REPDEL	Время/в 50-х долях секунды или в 60-х долях секунды для Сев. Америки/, на которое клавиша должна быть нажата перед повторением. Запуск со значением 35, но можно с помощью оператора POKE реализовать запуск с другими значениями.
1	23562	REPPER	Задержка /в 50-х долях секунды или в 60-х долях секунды для Сев. Америки/между последовательными повторами нажатой клавиши - первоначальное значение равно 5.

Обоз-начение	Адрес	Имя	Содержание
1	2	3	4
x2	23639	DATADD	Адрес разделителя последнего элемента данных.
x2	23641	E LINE	Адрес вводимой команды.
2	23643	C CUR	Адрес курсора.
x2	23645	CH ADD	Адрес следующего интерпретируемого символа: символ после аргумента оператора PEEK или атрибута NEWLINE в конце предложения POKE.
2	23647	X PTR	Адрес символа после маркера ?
x2	23645	WORKSP	Адрес временной рабочей области.
x2	23651	STKBOT	Адрес дна стека кулькулятора (вычислителя)
x2	23653	STKEND	Адрес начала резервного пространства
n1	23655	BREG	Регистр вычислителя.
n2	23656	MEM	Адрес области, используемой для памяти вычистлителя. (Обычно MEMOT,но не всегда).
1	23659	DFSZ	Число строк (включая одну пустую строку) и нижней части экрана.
2	23660	STOP	Номер верхней программной строки при автоматических листингах.
2	23662	OLDPPC	Номер строки, к которой осуществляется переход по оператору CONTINUE.
n1	23665	FLAGX	Различные признаки
n2	23666	STRLEN	Размер места назначения для печати строки при присвоении.
n2	23668	TADDR	Адрес следующего элемента в синтаксической таблице (очень нежелательно использовать).
2	23670	5EED	Начальное число для оператора RND /для выработки псевдослучайных чисел. Оно является переменной, которая устанавливаетс помощью оператора RANDOMIZE.
3	23672	FRAMES	Три байта (самый младший вначале) счетчика кадров. Приращение каждые 20 мс.
2	23675	UDG	Адрес первого определенного пользователем графического символа. Вы можете изменить этот адрес, например, для сохранения пространства памяти, посредством уменьшения определяемых пользователем графических символов.
1	23677	COORDS	Координата x последней построенной точки.
1	23678		Координата y последней построенной точки.
1	23679	P POSN	Номер 33 колонки позиции печатающего устройства.
1	23680	PR CC	Младший байт адреса следующей позиции для оператора LPRINT при печати (в буфере печатающего устройства).
1	23681		Не используется.
1	23664	OSPCC	Номер в строке предложения, к которому происходит переход по оператору CONTINUE
2	23682	ECHO E	Номер 33 колонки и номер 24 строки (в нижней половине) конца буфера ввода.

Обозначение	Адрес	Имя	Содержание
1	2	3	4
2	23684	DF CC	Адрес в воспроизводимом файле позиции
2	23686	DFCCL	Аналогично DF CC для нижней части экрана.
X1	23688	S POSN	Номер 33 колонки для позиции оператора PRINT
X1	23689		Номер 24 строки для позиции оператора PRINT.
X2	23690	SPOSNL	Аналогично S POSN для нижней части экрана.
1	23692	SCR CT	Вычисление числа перемещений: всегда на единицу больше числа необходимых перемещений, которое будет сделано перед остановкой с запросом на дальнейшее перемещение Если вы вставите сюда число больше единицы /скажем 255/, то содержимое экрана будет перемещаться без вашего решения
1	23692	SCR CT	Вычисление числа перемещений: всегда на единицу больше числа необходимых перемещений, которое будет сделано перед остановкой с запросом на дальнейшее перемещение. Если вы вставите сюда число больше единицы(скажем 255), то содержимое экрана будет перемещаться без вашего решения.
1	23693	ATTR P	Постоянно используемые цвета, и т.д. (как установлено предложениями по цвету).
1	23694	MASK P	Используется для прозрачных цветов и т.д. Любой бит, который находится в единице, показывает, что соответствующий бит атрибута взят не из переменной ATTR P, а из того что уже есть на экране.
N1	23695	ATTR T	Временные текущие цвета и т.д. (как установлены элементами цветов)
N1	23696	MASK T	Аналогично MASK P, но носит временный характер
1	23697	P FLAG	Дополнительные признаки.
N30	23698	MEMBOT	Область памяти вычислителя, используемый для хранения чисел, которые не могут быть надлежащим образом размещены в стеке вычислителя.
2	23728		Не используется (хранится 0)
2	23730	RAMTOP	Адрес последнего байта системной области BASIC
2	23732	P-RAMT	Адрес последнего байта физической памяти с произвольной выборкой.

3.3. Таблица кодов и команд ассемблера

Программирование компьютера «Дельта-С» возможно в машинных кодах и в командах ассемблера. Нередко это дает ускорение в сравнении с Бейсиком до 40-50 раз. Поскольку программирование в машинных кодах доступно лишь опытным программистам, ограничимся таблицей приведения кодов для компьютера. Она содержит десятичное значение кода (код), символ его (возможно управляющий), шестнадцатиричное значение кода HEX, мнемонику ассемблера Z80 и для составных команд их части, начинающихся с шестнадцатиричного адреса CB или ED.

3.3. Таблица кодов и команд ассемблера

Код		Символ	А С С Е М Б Л Е Р		
Десят.	Шестнад.	в БЕЙСИКЕ	-	После CB	После ED
0	00	не используется	NOP	RLC B	
1	01	не используется	LD BC,NN	RLC C	
2	02	не используется	LD (BC),A	RLC D	
3	03	не используется	INC BC	RLC E	
4	04	не используется	INC B	RLC H	
5	05	не используется	DEC B	RLC L	
6	06	PRINT запятая*	LD B,N	RLC (HL)	
7	07	EDIT	RLCA	RLC A	
8	08	курсор влево	EX AF,AF	RRC A	
9	09	курсор вправо	ADD HL,BC	RRC B	
10	0A	курсор вниз	LD A,(BC)	RRC D	
11	0B	курсор вверх	DEC BC	RRC E	
12	0C	DELETE	INC C	RRC H	
13	0D	ENTER	DEC C	RRC L	
14	0E	число	LD C,N	RRC (HL)	
15	0F	не используется	RRCA	RRC A	
16	10	управление INK	DJNZ S	RL B	
17	11	управление PAPER	LD DE,NN	RL C	
18	12	управление FLASH	LD (DE),A	RL D	
19	13	управл. BRIGHT	INC DE	RL E	
20	14	управл. INVERSE	INC D	RL H	
21	15	управление OVER	DEC D	RL L	
22	16	управление AT	LD D,N	RL (HL)	
23	17	управление TAB	RLA	RL A	
24	18	не используется	JR S	RR B	
25	19	не используется	ADD HL,DE	RR C	
26	1A	не используется	LD A,(DE)	RR D	
27	1B	не используется	DEC DE	RR E	
28	1C	не используется	INC E	RR H	
29	1D	не используется	DEC E	RR L	
30	1E	не используется	LD E,N	RR (HL)	
31	1F	не используется	RRA	RR A	
32	20	пробел	JR NZ,S	SLA B	
33	21	!	LD HL,NN	SLA C	
34	22	"	LD (NN),HL	SLA D	
35	23	#	INC HL	SLA E	
36	24	$	INC H	SLA H	

*Примечание: Символы 6...23 в БЕЙСИКе выполняют роль управляющих кодов для печати на экране.

Код		Символ	А С С Е М Б Л Е Р		
Десят.	Шестнад.	в БЕЙСИКЕ	-	После CB	После ED
37	25	%	DEC H	SLA L	
38	26	&	LD H,N	SLA (HL)	
39	27	'	DAA	SLA A	
40	28	(	JR Z,S	SRA B	
41	29	)	ADD HL,HL	SRA C	
42	2A	*	LD HL,(NN)	SRA D	
43	2B	+	DEC HL	SRA E	
44	2C	,	INC L	SRA H	
45	2D	_	DEC L	SRA L	
46	2E	.	LD L,N	SRA HL	
47	2F	/	CPL	SRA A	
48	30	0	JR NC,S		
49	31	1	LD SP,NN		
50	32	2	LD (NN),A		
51	33	3	INC SP		
52	34	4	INC (HL)		
53	35	5	DEC (HL)		
54	36	6	LD (HL),N		
55	37	7	SCF		
56	38	8	JR C,S	SRL B	
57	39	9	ADD HL,SP	SRL C	
58	3A	:	LD A,(NN)	SRL D	
59	3B	;	DEC SP	SRL E	
60	3C	<	INC A	SRL H	
61	3D	=	DEC A	SRL L	
62	3E	>	LD A,N	SRL (HL)	
63	3F	?	CCF	SRL A	
64	40	@	LD B,B	BIT 0,B	IN B,(C)
65	41	A	LD B,C	BIT 0,C	OUT (C),B
66	42	B	LD B,D	BIT 0,D	SBC HL,BC
67	43	C	LD B,E	BIT 0,E	LD (NN),BC
68	44	D	LD B,H	BIT 0,H	NEG
69	45	E	LD B,L	BIT 0,L	RETN
70	46	F	LD B,(HL)	BIT 0,(HL)	IM 0
71	47	G	LD B,A	BIT 0,A	LD I,A
72	48	H	LD C,B	BIT 1,B	IN C,(C)
73	49	I	LD C,C	BIT 1,C	OUT (C),C
74	4A	J	LD C,D	BIT 1,D	ADC HL,BC
75	4B	K	LD C,E	BIT 1,E	LD BC,(NN)
76	4C	L	LD C,H	BIT 1,H	
77	4D	M	LD C,L	BIT 1,L	RETI
78	4E	N	LD C,(HL)	BIT 1,(HL)	

Код		Символ	А С С Е М Б Л Е Р		
Десят.	Шестнад.	в БЕЙСИКЕ	-	После CB	После ED
79	4F	O	LD C,A	BIT 1,A	LD R,A
80	50	P	LD D,B	BIT 2,B	IN D,(C)
81	51	Q	LD D,C	BIT 2,C	OUT (C),D
82	52	R	LD D,D	BIT 2,D	SBC HL,DE
83	53	S	LD D,E	BIT 2,E	LD (NN),DE
84	54	T	LD D,H	BIT 2,H	
85	55	U	LD D,L	BIT 2,L	
86	56	V	LD D,(HL)	BIT 2,(HL)	IM 1
87	57	W	LD D A	BIT 2,A	LD A,I
88	58	X	LD E,B	BIT 3,B	IN E,(C)
89	59	Y	LD E,C	BIT 3,C	OUT (C),E
90	5A	Z	LD E,D	BIT 3,D	ADC HL,DE
91	5B	[	LD E,E	BIT 3,E	LD DE,(NN)
92	5C	/	LD E,H	BIT 3,H	
93	5D	]	LD E,L	BIT 3,L	
94	5E	...	LD E,(HL)	BIT 3,(HL)	IM 2
95	5F	_	LD E,A	BIT 3,A	LD A,R BC
96	60	...	LD H,B	BIT 4,B	IN H,(C)
97	61	a	LD H,C	BIT 4,C	OUT (C),H
98	62	b	LD H,D	BIT 4,D	SBC HL,HL
99	63	c	LD H,E	BIT 4,E	LD (NN),HL
100	64	d	LD H,H	BIT 4,H	
101	65	e	LD H,L	BIT 4,L	
102	66	f	LD H,(HL)	BIT 4,(HL)	
103	67	g	LD H,A	BIT 4,A	RRD
104	68	h	LD L,B	BIT 5,B	IN L,(C)
105	69	i	LD L,C	BIT 5,C	OUT (C),L
106	6A	j	LD L,D	BIT 5,D	ADC HL,HL
107	6B	k	LD L,EHL)	BIT 5,E	LD HL,(NN)
108	6C	l	LD L,H	BIT 5,H	
109	6D	m	LD L,L	BIT 5,L	
110	6E	n	LD L,(HL)	BIT 5,(HL)	
111	6F	o	LD L,A	BIT 5,A	RLD
112	70	p	LD (HL),B	BIT 6,B	IN F,(C)
113	71	q	LD (HL),C	BIT 6,C	
114	72	r	LD (HL),D	BIT 6,D	SBC HL,SP
115	73	s	LD (HL),E	BIT 6,E	LD (NN),SP
116	74	t	LD (HL),H	BIT 6,H	
117	75	u	LD (HL),L	BIT 6,L	
118	76	v	HALT	BIT 6,(HL)	
119	77	w	LD (HL),A	BIT 6,A	
120	78	x	LD A,B	BIT 7,B	IN A,(C)

Код		Символ	А С С Е М Б Л Е Р		
Десят.	Шестнад.	в БЕЙСИКЕ	-	После CB	После ED
121	79	y	LD A,C	BIT 7,C	OUT (C),A
122	7A	z	LD A,D	BIT 7,D	ADC HL,SP
123	7B	{	LD A,E	BIT 7,E	LD (SP),NN
124	7C	...	LD A,H	BIT 7,H	
125	7D	}	LD A,L	BIT 7,L	
126	7E	...	LD A,(HL)	BIT 7,(HL)	
127	7F	...	LD A,A	BIT 7,A	
128	80	...	ADD A,B	RES 0,B	
129	81	...	ADD A,C	RES 0,C	
130	82	...	ADD A,D	RES 0,D	
131	83	...	ADD A,E	RES 0,E	
132	84	...	ADD A,H	RES 0,H	
133	85	...	ADD A,L	RES 0,L	
134	86	...	ADD A,(HL)	RES 0,(HL)	
135	87	...	ADD A,A	RES 0,A	
136	88	...	ADC A,B	RES 1,B	
137	89	...	ADC A,C	RES 1,C	
138	8A	...	ADC A,D	RES 1,D	
139	8B	...	ADC A,E	RES 1,E	
140	8C	...	ADC A,H	RES 1,H	
141	8D	...	ADC A,L	RES 1,L	
142	8E	...	ADC A,(HL)	RES 1,(HL)	
143	8F	...	ADC A,A,D	RES 1,A	
144	90	граф. (A)	SUB B	RES 2,B	
145	91	граф. (B)	SUB C	RES 2,C	
146	92	граф. (C)	SUB D	RES 2,D	
147	93	граф. (D)	SUB E	RES 2,E	
148	94	граф. (E)	SUB H	RES 2,H	
149	95	граф. (F)	SUB L	RES 2,L	
150	96	граф. (G)	SUB (HL)	RES 2,(HL)	
151	97	граф. (H)	SUB A	RES 2,A	
152	98	граф. (I)	SBC A,B	RES 3,B	
153	99	граф. (J)	SBC A,C	RES 3,C	
154	9A	граф. (K)	SBC A,D	RES 3,D	
155	9B	граф. (L)	SBC A,E	RES 3,E	
156	9C	граф. (M)	SBC A,H	RES 3,H	
157	9D	граф. (N)	SBC A,L	RES 3,L	
158	9E	граф. (O)	SBC A,(HL)	RES 3,(HL)	
159	9F	граф. (P)	SBC A,A	RES 3,A	
160	A0	граф. (Q)	AND B	RES 4,B	
161	A1	граф. (R)	AND C	RES 4,C	
162	A2	граф. (S)	AND D	RES 4,D	

Код		Символ	А С С Е М Б Л Е Р		
Десят.	Шестнад.	в БЕЙСИК	-	После CB	После ED
163	A3	графическое(T)	AND E	RES 4,E	
164	A4	графическое(U)	AND H	RES 4,H	
165	A5	RND	AND L	RES 4,L	
166	A6	INKEY$	AND (HL)	RES 4,(HL)	
167	A7	PI	AND A	RES 4,A	
168	A8	FN	XOR B	RES 5,B	LDD
169	A9	POINT	XOR C	RES 5,C	CPD
170	AA	SCREEN$	XOR D	RES 5,D	IND
171	AB	ATTR	XOR E	RES 5,E	OUTD
172	AC	AT	XOR H	RES 5,H	
173	AD	TAB	XOR L	RES 5,L	
174	AE	VAL$	XOR (HL)	RES 5,(HL)	
175	AF	CODE	XOR A	RES 5,A	
176	B0	VAL	OR B	RES 6,B	LDIR
177	B1	LEN	OR C	RES 6,C	CPIR
178	B2	SIN	OR D	RES 6,D	INIR
179	B3	COS	OR E	RES 6,E	OTIR
180	B4	TAN	OR H	RES 6,H	
181	B5	ASN	OR L	RES 6,L	
182	B6	ACS	OR (HL)	RES 6,(HL)	
183	B7	ATN	OR A	RES 6,A	
184	B8	LN	CP B	RES 7,B	LDDR
185	B9	EXP	CP C	RES 7,C	CPDR
186	BA	INT	CP D	RES 7,D	INDR
187	BB	SQR	CP E	RES 7,E	OTDR
188	BC	SGN	CP H	RES 7,H	
189	BD	ABS	CP L	RES 7,L	
190	BE	PEEK	CP (HL)	RES 7,(HL)	
191	BF	IN	CP A	RES 7,A	
192	C0	USR	RET NZ	SET 0,B	
193	C1	STR$	POP BC	SET 0,C	
194	C2	CHR$	JP NZ,NN	SET 0,D	
195	C3	NOT	JP NN	SET 0,E	
196	C4	BIN	CALL NZ,NN	SET 0,H	
197	C5	OR	PUSH BC	SET 0,L	
198	C6	AND	ADD A,N	SET 0,(HL)	
199	C7	<==	RST 0	SET 0,A	
200	C8	>==	RET Z	SET 1,B	
201	C9	<>	RET	SET 1,C	
202	CA	LINE	JP Z,NN	SET 1,D	
203	CB	THEN		SET 1,E	
204	CC	TO	CALL Z,NN	SET 1,H	

Код		Символ	А С С Е М Б Л Е Р		
Десят.	Шестнад.	в БЕЙСИК	-	После CB	После ED
205	CD	STEP	CALL NN	SET 1,L	
206	CE	DEF FN	ADC A,N	SET 1,(HL)	
207	CF	CAT	RST 8	SET 1,A	
208	D0	FORMAT	RET NC	SET 2,B	
209	D1	MOVE	POP DE	SET 2,C	
210	D2	ERASE	JP NC,NN	SET 2,D	
211	D3	OPEN#	OUT (N),A	SET 2,E	
212	D4	CLOSE#	CALL NC,NN	SET 2,H	
213	D5	MERGE	PUSH DE	SET 2,L	
214	D6	VERIFY	SUB N	SET 2,(HL)	
215	D7	BEEP	RST 10	SET 2,A	
216	D8	CIRCLE	RET C	SET 3,B	
217	D9	INK	EXX	SET 3,C	
218	DA	PAPER	JP C,NN	SET 3,D	
219	DB	FLASH	IN A,(N)	SET 3,E	
220	DC	BRIGHT	CALL C,NN	SET 3,H	
221	DD	INVERSE	Предшествует операциям, в которых вместо регистра HL участвует IX.	SET 3,L	
222	DE	OVER	SBC A,N	SET 3,(HL)	
223	DF	OUT	RST 18	SET 3,A	
224	E0	LPRINT	RET PO	SET 4,B	
225	E1	LLIST	POP HL	SET 4,C	
226	E2	STOP	JP PO,NN	SET 4,D	
227	E3	READ	EX(SP),HL	SET 4,E	
228	E4	DATA	CALL PO,NN	SET 4,H	
229	E5	RESTORE	PUSH HL	SET 4,L	
230	E6	NEW	AND N	SET 4,(HL)	
231	E7	BORDER	RST 20	SET 4,A	
232	E8	CONTINUE	RET PE	SET 5,B	
233	E9	DIM	JP (HL)	SET 5,C	
234	EA	REM	JP PE,NN	SET 5,D	
235	EB	FOR	EX DE,HL	SET 5,E	
236	EC	GO TO	CALL PE,NN	SET 5,H	
237	ED	GO SUB		SET 5,L	
238	EE	INPUT	XOR N	SET 5,(HL)	
239	EF	LOAD	RST 28	SET 5,A	
240	F0	LIST	RET P	SET 6,B	
241	F1	LET	POP AF	SET 6,C	
242	F2	PAUSE	JP P,NN	SET 6,D	
243	F3	NEXT	DI	SET 6,E	
244	F4	POKE	CALL P,NN	SET 6,H	

Код		Символ	А С С Е М Б Л Е Р		
Десят.	Шестнад.	в БЕЙСИК	-	После CB	После ED
245	F5	PRINT	PUSH AF	SET 6,L	
246	F6	PLOT	OR N	SET 6,(HL)	
247	F7	RUN	RST 30	SET 6,A	
248	F8	SAVE	RET M	SET 7,B	
249	F9	RANDOMIZE	LD SP,HL	SET 7,C	
250	FA	IF	JP M,NN	SET 7,D	
251	FB	CLS	EI	SET 7,E	
252	FC	DRAW	CALL M,NN	SET 7,H	
253	FD	CLEAR	Предшествует операциям с реги-		
			стром IY.	SET 7,L	
254	FE	RETURN	CP N	SET 7,(HL)	
255	FF	COPY	RST 38	SET 7,A	

Другие мои работы можно скачать здесь:

Google Drive

Яндекс.Диск

Буду рад видеть Вас в числе подписчиков:

Instagram

YouTube