ДСО "ПРИБОРОСТРОЕНЕ И АВТОМАТИЗАЦИЯ"

НАЧАЛНО РЪКОВОДСТВО ЗА РАБОТА С

П Р А В Е Ц - 82

София, март 1984 г.

С Ъ Д Ъ Р Ж А Н И Е

	стр.
ПРЕДГОВОР	4
ГЛАВА I	
НАЧАЛНИ ПОЗНАНИЯ ЗА КОМПЮТЪРА	
Увод	6
Необходими принадлежности	6
Свързване на монитора	7
Включване на принадлежностите за игра	7
Свързване на дисковото устройство	8
Свързване на касетофона	8
Клавиатурата на ПРАВЕЦ	8
Означения при инструкциите за работа с клавиатурата	11
Контрол и други странни символи	11
Работа с касетофон	12
Стандартна процедура при зареждане от касетофон	14
Полезно указание	15
Използуване на дисковото устройство	16
Прекъсване на изпълнението на програмата	19
Настройка на цветовете на екрана	19
Разруши стената	20
ГЛАВА II	
НАЧАЛНИ ПОЗНАНИЯ ПО РАЗШИРЕН БЕЙСИК	22
Първо запознаване с командата PRINT	23
Формат при числата	26
Допълнителни сведения за RETURN	28
Лесни начини за редактиране	29
Цветове по екрана	31
Съобщения за грешка при команда PLOT	34
Рисуване на прави линии	35
Устройствата за игра	37
Памет: запознаване с променливите	38
Приоритет при аритметичните операции или кой има предимство?	43
Как да избягваме грешки свързани с приоритета	46

стр.

ГЛАВА III

ПРОСТО ПРОГРАМИРАНЕ

Отложено изпълнение: 49
Просто редактиране: 53
Лесна акробатика: цикъл чрез GOTO 55
Някои неща облекчават живота: допълнителни моменти при редактирането 56
Подвижният курсор: редактиране чрез клавиша ESC 57
Няколко думи за изучаването на разширения БЕЙСИК 59
Предотвратен нещастен случай 60
Истината: аритметични и логически твърдения 61
Приоритет при операциите 65
Командата IF 65
Записване на програми върху диск: 67
Записване на програми с касетофон: 68
Други графични програми 69
Образуване на цикли посредством FOR/NEXT 71
Една неправилна програма 76
Механичен начин за преорганизиране на цикли 77
Последен пример на вложени цикли 78
С блясък в очите 79
Художествено оформление: запетая, точка и запетая 80

ГЛАВА IV

ГРАФИЧНИ ИЗОБРАЖЕНИЯ

Разговор с програмата 85
Рикошет: програма с много скокове 89
Компютърът издава звуци: 91
Озвучаване на отскачащата топка 92
Случайни числа 92
Имитиране на двойка зарове 95
Подпрограми 96
Проследяване 99
Подобрена програма за рисуване на коне 100
Графики с висока разрешаваща способност 103

ГЛАВА V стр.

НИЗОВЕ И МАСИВИ

Низ от операции: 110
Думата "конкатенация": събиране на низове 115
Други низови функции 116
Увод в масивите 119
Съобщения за грешки, свързани с използуване на масиви 122
Заключение 122

ДОПЪЛНЕНИЯ

Допълнение А: Списък на командите 124
Добълнение Б: Запазени думи в ПРАВЕЦ-версията на БЕЙСИК 136
Допълнение В: Средства за редактиране 138
Клавиши с лява и дясна стрелка 138
Чисти движения на курсора 139
Премахване на редове от програмата 140
Списък на средствата за редактиране 141
Допълнение Г: Съобщения за грешка 142

ПРЕДИСЛОВИЕ

Това ръководство е предназначено за тези, които желаят да се научат да програмират на предназначената за ПРАВЕЦ-версия на езика БЕЙСИК. С това ръководство, с един компютър ПРАВЕЦ и с малко време и внимание от Ваша страна, ще откриете, че не е трудно да се научите да програмирате компютър. В началото, като всяко ново занимание, програмирането ще бъде непривично за вас. Това ръководство е предназначено да премахне всяка боязън, която може да възникне у вас. Преди всичко, за четенето му не са необходими никакви специални познания. Всичко е открито и във всеки момент се обяснява само едно нещо. Ако сте начинаещ, проверявайте всяко нещо на практика и още от самото начало вземете решение да не бързате. Това е достатъчна гаранция, че ще се научите да програмирате.

Истинската тайна е да не бързате и да проверявате всичко на практика. Човек не може да се научи да програмира просто като чете тази или онази книга. Вие трябва да се учите практикувайки, така както се учи свирането на цигулка или карането на велосипед. Трябва да правите грешки и да ги поправяте, и да не се дразните, когато грешите.

Ако вече умеете да програмирате, едно бързо преглеждане на тази книга ще Ви запознае с особеностите на ПРАВЕЦ-версията на БЕЙСИК. Подозираме, че ще Ви направи силно впечатление лекотата, с която се създават графични изображения с висока разрешаваща способност, както и другите особености, които правят ПРАВЕЦ чудесен компютър с широк диапазон на приложение.

Тази книга е ръководство. За справки и специални подробности трябва да ползувате съпътствуващата литература към ПРАВЕЦ. Поради обширните си азбучни указатели и многобройните технически детайли тази литература трябва да се използува след изучаване на основните моменти в ръководството. Надяваме се, че работата с ръководството ще бъде за Вас такова удоволствие, каквото беше за нас написването му.

ГЛАВА I

НАЧАЛНИ ПОЗНАНИЯ ЗА КОМПЮТЪРА

Увод
Необходими принадлежности
Свързване на монитора
Включване на принадлежностите за игра
Свързване на дисковото устройство
Свързване на касетофона
Клавиатурата на ПРАВЕЦ
Означения при инструкциите за работа с клавиатурата
Контрол и други странни символи: клавишите
Работа с касетофон
Стандартна процедура при зареждане от касетофон
Полезно указание
Използуване на дисковото устройство
Прекъсване на изпълнението на програмата
Настройка на цветовете на екрана
Разруши стената

УВОД

Това ръководство ще Ви покаже как лесно да приведете в действие своя ПРАВЕЦ и как /също лесно/ да се научите да го програмирате. Ако сте стари познати с програмирането, ще откриете в ПРАВЕЦ-версията на БЕЙСИК някои нови особености и улеснения, които правят програмирането много по-приятно. Ако сте начинаещ, приятните моменти и улеснения, които ще откриете са също много. Начинаещите обаче трябва да бъдат предупредени, че програмирането, макар и не трудно, може да бъде научено само чрез практикуване. По-нататък ще бъде казано повече по този въпрос, но трябва да се помни, че тази книга е предназначена за употреба, а не за прочит.

Привеждането на компютъра в действие е не по-трудно от привеждането в действие на стерео-уредба, поради което не са необходими технически познания.

При работата с ръководството трябва да се внимава при появяването на знака

▽

Предназначението на този знак е да насочи Вашето внимание към някоя по-необикновена особеност на езика. Описаната в тези случаи ситуация може да предизвика унищожаването на Вашата програма.

НЕОБХОДИМИ ПРИНАДЛЕЖНОСТИ

Това ръководство се намира в опаковката на компютъра. Тази опаковка трябва да съдържа още

1. Кабел за захранването /кабелът, който се включва в контакта на стената/
2. Комплект устройства за игра /малки черни кутии с бутони и потенциометри, свързани с кабел/.
3. Кабел за свързване на компютъра с касетофон. Този кабел трябва да има куплунг на всеки край.
4. Няколко касети за касетофон.

Освен самия ПРАВЕЦ и съдържанието на опаковката му, ще Ви бъдат необходими още две неща по избор от направения подолу списък:

Имате нужда от
а. Касетофон
ИЛИ
б. Дисково устройство

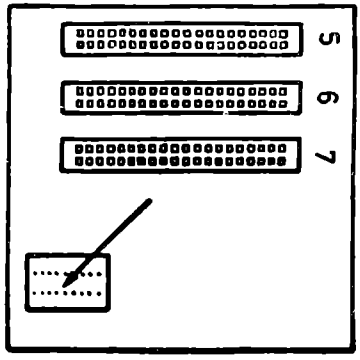
ПРАВЕЦ работи нормално с черно-бял монитор, но в този случай не може да бъде изпълзувана възможността му да създава цветни изображения. Цветовете, описани в ръководството, ще се появяват на черно-бял монитор като различни оттенъци на сивото.

СВЪРЗВАНЕ НА МОНИТОРА

Свързването се извършва, като излизащият от задната страна на компютъра кабел се включва в куплунга на монитора.

ВКЛЮЧВАНЕ НА УСТРОЙСТВАТА ЗА ИГРА

За целта е необходимо да отворите горния капак на компютъра. Това става като се издърпа задния му край с две ръце право нагоре. След това твърде нежният куплунг на принадлежностите за игра се включва в предназначеното за целта гнездо, намиращо се в десния заден край /гледано отпред/. Трябва много да внимавате, щото всички игли на куплунга да попаднат на местата си.



СВЪРЗВАНЕ НА ДИСКОВОТО УСТРОЙСТВО

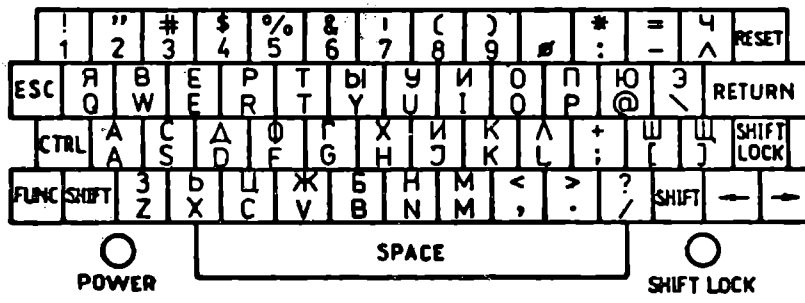
Дисковото устройство /ако имате такова/ трябва да се ра зопакова внимателно. След това прочетете предговора и първата глава на ДОС-ръководството към ПРАВЕЦ. Там ще намерите всички необходими инструкции за инсталиране на дисковото устройство.

СВЪРЗВАНЕ НА КАСЕТОФОНА

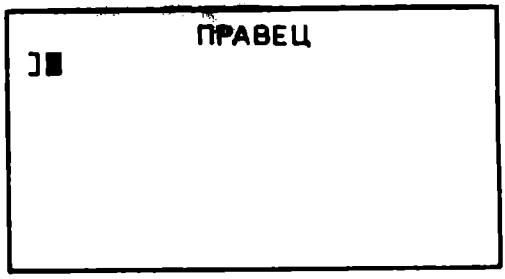
Напълно аналогично на свързването на касетофона с друг касетофон или радиоприемник. Куплунгът се намира на задната страна на компютъра.

КЛАВИАТУРАТА НА ПРАВЕЦ

След като всички свързвания са направени, можете да включите компютъра. За целта е достатъчно да натиснете ключа, който се намира на задната му страна. В резултат на това се запалва сигналната лампа в левия долен край на клавиатурния блок, компютърът подава къс звуков сигнал и в горния край на екрана на монитора /в случай, че последният е включен и работи/ се появява надписът ПРАВЕЦ.



В случая, когато компютърът не е свързан с дискови устройства, в левия край на екрана, малко под надписа ПРАВЕЦ се появява символът] и непосредствено след него едно мигащо бяло квадратче, наречено "курсор".



Ако след включването съдържанието на екрана не отговаря на даденото описание, натиснете клавиша с надпис RESET /намира се в десния горен ъгъл на клавиатурата/. При отпускането на този клавиш се чува къс звуков сигнал. Ако екранът все още не изглежда добре, изключете компютъра и след това го включете отново. Това обикновено оправя нещата.

Когато към компютъра е включено дисково устройство, след включването то издава няколко щракащи звука, последвани от тихо бръмчене. Ако дисковото устройство е празно/в него няма поставен диск/, щракащите звуци се повтарят още няколко пъти, след което прекратява работата си и на екрана се появява символът] следван от мигащия курсор. Ако в дисковото устройство има диск, съдържащ Дисковата Операционна Система /ДОС вж. ръководството по ДОС към ПРАВЕЦ/ щракащите звуци не се повтарят, надписът ПРАВЕЦ изчезва от екрана и след около 10 секунди бръмченето спира, устройството прекратява работата си и на екрана се появява символът] следван от мигащия курсор. Независимо от това дали е поставен диск, докато дисковото устройство работи на предния му панел свети червена сигнална лампа.

Разгледайте клавиатурата. Тя се различава малко от клавиатурата на пишещите машини. Преди всичко се забелязва, че няма малки букви. За програмирането на ПРАВЕЦ големите букви са напълно достатъчни. Намерете с помощта на диаграмата двата клавиша с надпис SHIFT. Те играят аналогична роля на клавишите за превключване на големи и малки букви при пишещите машини. Разликата се състои в това, че що се отнася до буквите

при ПРАВЕЦ тези клавиши служат за превключване на кирилицата и латиницата. Както и при пишещите машини, когато желаете да работите по-дълго време с горните символи от клавиатурата, можете да натиснете веднаж клавиша с надпис SHIFT LOCK вместо през цялото време да държите натиснат някой от клавишите SHIFT Последователните натискания на клавиша SHIFT LOCK превключват алтернативно клавиатурата за постоянна работа само с горни или само с долни символи. Когато системата се намира в режим на постоянна работа с горни символи, в долния десен ъгъл на клавиатурния блок свети жълта сигнална лампа.

Индийските математици са означили цифрата "нула" със символ, който не се среща в тяхната писменост, но изглежда точно като буквата О от кирилицата и латиницата. Тъй като компютърът обикновено работи с твърде сложна смес от букви и цифри, за него е твърде важно да различава "от пръв поглед" буквата О от цифрата "нула". Когато напреднете още малко в програмирането, Вие ще се убедите, че тази възможност е необходима и за самия програмист. За целта в компютърния свят единият от двата символа се изписва пресечен с наклонена черта. При ПРАВЕЦ и при повечето компютри това е символът "нула" - Ø. Означенията на клавиатурата и по екрана са винаги съобразени с този начин на различаване на двата символа. Опитайте.

След като сте писали известно време по екрана, той започва да се задръства със символи. За да го изчистите, трябва да използувате клавиша с надпис ESC . При натискането му на екрана не се появява символ. Подобно на клавиша SHIFT LOCK , последователното натискане на ESC превключва алтернативно компютъра в "редакторски" и нормален режим. За редакторския режим ще поговорим достатъчно подробно по-късно в това ръководство. Влизането в редакторски режим става с натискане на клавиша ESC а излизането от него може да стане с натискането на който и да било клавиш /включително и с повторно натискане на ESC /, с изключение на клавишите I, J, K, M, SHIFT, CTRL, SHIFT LOCK и RPT. За да изчистите екрана преминете в редакторски режим, като натиснете ESC и след това натиснете @. Ако екранът не се изчисти, това означава, че преди опита за изчистване сте натискали ESC и сте влезли в редакторски режим, при което новото натискане на ESC е върнало системата в нормално състояние. Следователно, ако натиснете отново първо ESC и след това @ ще получите така желаното изчистване на екрана.

ОЗНАЧЕНИЯ ПРИ ИНСТРУКЦИИТЕ ЗА РАБОТА С КЛАВИАТУРАТА

Ще въведем прости означения при указанията за работа с клавиатурата.

Преди всичко, когато искаме да укажем, че даден клавиш трябва да бъде натиснат, като в същото време е натиснат друг клавиш /например за написването на @ е необходимо да натиснем клавиша с надпис

като едновременно държим натиснат клавиша SHIFT /ще пишем означенията на двата клавиша един над друг/, в случая

SHIFT

При натискането на долния клавиш трябва да държите натиснат горния.

Когато искаме да укажем, че трябва да натиснете последователно няколко клавиша, ще пишем означенията на клавишите в реда, в който трябва да бъдат натискани, например

В смисъла на горните означения инструкцията за изчистване на екрана изглежда по следния начин

SHIFT

ESC 

Опитайте.

КОНТРОЛ И ДРУГИ СТРАННИ СИМВОЛИ

Когато натиснете клавиша 

, на екрана се появява цифрата 5. Може би вярвате, че е така, но все пак опитайте. Ако в момента, в който натискате клавиша 

държите натиснат клавиша SHIFT, на екрана се появява знакът за процент /%/. Нали?

По същия начин, клавишът CTRL дава възможност на някои клавиши да изпълняват и трета функция. В този случай на екрана не се появяват символи, но за сметка на това компютърът извършва различни действия

Натиснете клавиша CTRL и без да го пускате натиснете G:

Компютърът отвръща със звуков сигнал.

Друг клавиш, който обикновено не се среща при пишещите машини е RPT . Ако държите клавиша RPT натиснат и едновременно с това натиснете някой друг клавиш, символът, съответствуващ на този клавиш се появява последователно на екрана, докато и двата клавиша са натиснати.

Клавиатурата на ПРАВЕЦ има също един клавиш с надпис RETURN. В някои отношения той съответствува на клавиша, който при пишещите машини връща валяка в начално положение. При ПРАВЕЦ той връща курсора в левия край на екрана, но също така дава специално съобщение за компютъра. По-късно ще се запознаем по-подробно с този въпрос. Понякога, когато натиснете RETURN компютърът ще отговори със звуков сигнал и на екрана ще се появи съобщението

? SYNTAX ERROR

Засега не обръщайте внимание на това съобщение.

Единствените неразгледани досега клавиши са ← и →. Те движат курсора наляво и надясно. Подробности по тяхното действие ще бъдат дадени по-късно. Изпробвайте тези клавиши и всички други, които откриете. Работата с клавиатурата не може да повреди компютъра по никакъв начин. Освен ако не работите с чук. Така че не се стеснявайте да експериментирате. С пръсти.

РАБОТА С КАСЕТОФОН

/Ако не използувате касетофон, направо преминете към раздела, наречен ИЗПОЛЗУВАНЕ НА ДИСКОВО УСТРОЙСТВО/

Сега натиснете клавиша RETURN . Дясната средна скоба, следвана от мигащия курсор се разполагат в левия край на екрана.

Обикновено човек използува касетофон с намерението да слуша различни звуци. Ако касетофонът е пуснат прекалено тихо, част от думите или музиката не се чуват добре. Прекалено силното възпроизвеждане е досадно за повечето хора.

Целта на работата с касетофон при ПРАВЕЦ е прехвърлянето на информация от лентата в паметта на компютъра или прехвърлянето на информация от паметта на компютъра върху лентата. Засега ще разглеждаме първия процес. Ако силата на звука е твърде малка, ПРАВЕЦ ще пропусне част от подаваната информация и ще се оплаче, като даде съобщение за грешка. Оплаквания ще получите и ако силата на звука е твърде голяма.

Правилното положение на регулатора за силата на звука се получава по метода на пробите и грешките. Пускате тихо касетофона на компютъра и следите дали информацията се прочита правилно. Ако това не е така /компютърът издава съобщение за грешка/, почвате от начало, като увеличавате малко силата на звука. При нов неуспех усилвате още малко звука и т.н. След известен брой опити силата на звука ще бъде точно по вкуса на ПРАВЕЦ и той ще съобщи това със звуков сигнал.

Сега да опитаме всичко на практика. Изчистете екрана с

Поставете касетата с надпис ЦВЕТОВЕ в касетофона. За всяко експериментално положение на регулатора за силата на звука трябва да правите следното:

1. Превиване на касетата до начално положение.
2. Пуснете касетофона на просвирване.
3. Напишете: 

Когато направите това, курсорът изчезва. Около 15 секунди нищо не се случва. След това са налице следните възможности:

а. Поява на съобщението ?SYNTAX ERROR
б. Нищо не се случва
в. Появява се съобщението ERR или ERRERR /с или без звуков сигнал/.
г. Компютърът подава само звуков сигнал.

В случая /а/ се върнете на точка 1 без да променяте силата на звука.

В случаите /б/ и /в/ си спомнете дали сте чакали 15 секунди преди да се откажете. Ако няма средна скоба или курсор и ПРАВЕЦ не реагира на клавиатурата, натиснете RESET, увеличете малко силата на звука и се върнете на точка 1. Поня-

кога, макар и много рядко, командата LOAD не сработва и курсорът се появява на екрана веднага, без да чака прочитането на лентата. В такъв случай изключете и отново включете компютъра и започнете отново.

Когато се реализира случая /г/, Вие сте на прав път. Когато чуете звуковия сигнал, почакайте още 15 секунди. Или ще получите съобщение за грешка /случай в./, или на екрана ще се появи средната скоба, последвана от курсора. Ако се случи последното, спрете касетофона и пренавийте лентата. Отбележете си положението на регулатора за силата на звука, така че да можете да го използувате, когато следващия път работите с касетофона. След това напишете P R У U H N RETURN

Екранът трябва да изглежда по следния начин

```
ДЕМОНСТРАЦИОННИ ПРОГРАМИ ПРАВЕЦ
ЗА РАБОТА С ДЕМОНСТРАЦИЯТА, НАПИШЕТЕ
НОМЕРА И. СЛЕД ТОВА НАТИСНЕТЕ КЛАВИША
С НАДПИС 'RETURN' В ДЕСНИЯ КРАЙ НА
КЛАВИАТУРАТА. НАТИСНЕТЕ 'RETURN' И КОГАТО
ИСКАТЕ ДА СПРЕТЕ ДАДЕНА ДЕМОНСТРАЦИЯ.
1.  СТАНДАРТНИ ИМЕНА НА ЦВЕТОВЕТЕ
2.  СТАНДАРТНИ НОМЕРА НА ЦВЕТОВЕТЕ
3.          КАЛЕЙДОСКОП
4.      РИСУВАНЕ ПО ЕКРАНА

КАКВО ЩЕ ЖЕЛАЕТЕ ? ■
```

ОБИЧАЙНАТА ПРОЦЕДУРА ЗА ЗАРЕЖДАНЕ ОТ ЛЕНТА
/след като веднаж регулаторът за силата на звука е правилно нагласен/

1. Пренавиване на лентата.
2. Пускане на касетофона на просвирване.
3. Подаване на команда LOAD

След като натиснете RETURN курсорът изчезва. Нищо не се случва в продължение на 5 до 20 секунди, след което компютърът подава звуков сигнал. Това означава, че информацията от лентата е започнала да преминава в паметта на компютъра. След известно време /зависи от това колко информация има върху лентата, но обикновено не повече от няколко минути /ПРАВЕЦ отново подава звуков сигнал и на екрана се появяват средната дясна скоба и курсорът.

4. Спиране на касетофона и превиване на лентата. Информацията е била прехвърлена и засега нямате повече работа с касетофона.

5. Подаване команда RUN

△

Ако компютърът е в режим на разширен БЕЙСИК, записаната върху лентата информация трябва да бъде на същия език. Опитът да се превхърли в паметта на компютъра информация, записана на несъответстващ компютърен език, води до непредвидими резултати. Върху екрана се появяват странни символи и съобщения за грешка, можете да загубите контрола от клавиатурата, както и да се случат други загадъчни неща. При това положение единственият изход е да изключите и отново да включите компютъра.

За означаването на процеса на вземане на информация от лента и поставянето ѝ в компютъра, в компютърния свят се използуват различни термини. Казва се, че компютърът "чете" /"прочита"/ лентата, а информацията бива "прочетена" или "вкарана" в компютъра. Самият процес на четене също се нарича "качване" на лентата /или информацията/ в компютъра. Всички тези изрази означават едно и също нещо.

ПОЛЕЗНО УКАЗАНИЕ

Какво толкова интересно намира компютърът в тези ленти? Прослушайте някоя от тях. Не звучи много приятно. Все пак вие различавате някои от звуците, които компютърът слуша. Информацията започва с устойчив тон. Следва късо "бип" и същият устойчив тон продължава още известно време. Този звук има 1000 трептения в секунда. След него идва звук, който твърде напомня шума от пороен дъжд.

Ако сте свикнали да слушате добри магнетофонни записи, вие лесно ще определите на слух дали дадена лента съдържа компютърен запис. Ако можете да възприемате информацията, която такава лента съдържа, Вие сте мутант и ще отидете далече в света на компютрите.

ИЗПОЛЗУВАНЕ НА ДИСКОВО УСТРОЙСТВО

/Пропуснете този раздел, ако не използвате дисково устройство/

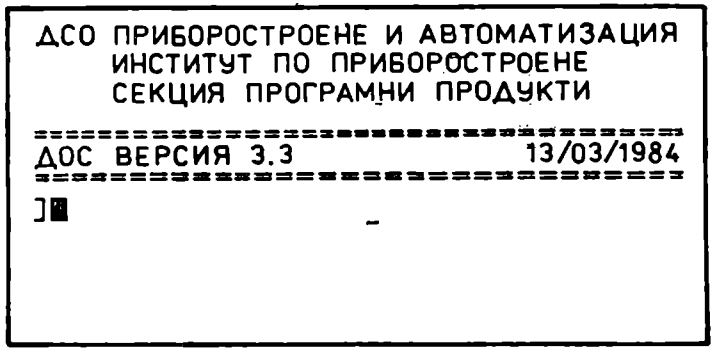
Работата с дисково устройство е много по-лесна и бърза от работата с касетофон. Трябва да се има предвид обаче, че дисковете и дисковите устройства са деликатни създания и трябва да полагате известни грижи, за да ги предпазите от повреда. Информация по този въпрос ще намерите в ръководството по ДОС.

Последният раздел на първа глава на ръководството по ДОС се нарича ПОСТАВЯНЕ И ИЗВАЖДАНЕ НА ДИСКОВЕТЕ. Извадете системния диск от опаковката му и го поставете в дисковото устройство, като спазвате изискването етикетът да бъде отгоре и дискът да влиза в отвора на дисковото устройство с овалния изрез напред, както е описано в ръководството по ДОС.

Една от характеристиките на дисковото устройство, които го правят толкова удобно за работа е възможността му да записва и изтрива няколко различни групи от информация /файлове/, всяка от които има свое име. Например една програма, която съдържа списък на адреси, би могла да бъде наречена АДРЕСИ.

Системата от програми, която се занимава с изтриването и записването на файловете върху диска, съставянето и актуализирането на списъка на файловете, както и с много други стопански дейности се нарича Дискова Операционна Система или ДОС. Процесът на добавяне на възможностите на ДОС към разширения БЕЙСИК /или към който и да било език, с който ПРАВЕЦ работи/ се нарича "зареждане на ДОС" или "зареждане на системата".

Има няколко начина за зареждане на ДОС. Един от тях е просто да изключите компютъра и след това да го включите отново. Червената лампа върху дисковото устройство ще светне отново и то ще издаде същите щракащи и бръмчащи звуци, както и при първото включване на компютъра; когато бръмченето спре и червената лампа угасне, надписът ПРАВЕЦ изчезва от екрана и се появява съобщение:



Това съобщение означава, че Дисковата Операционна Система е заредена.

Друг начин за зареждане на ДОС е да напишете

SHIFT

И I | H N | 3 # | 6 & | RETURN

Ако контролерът не е включен в шесто гнездо на компютъра, напишете

SHIFT

И I | H N | 3 #

последвано от номера на гнездото, в което е включен контролера и след това натиснете RETURN.

Системният диск изпълнява специални функции. Той съдържа програми, които ще Ви бъдат необходими за ефективна работа с това ръководство, както и други полезни програми. За да видите какви програми съдържа даден диск, използувайте командата CATALOG. Просто напишете

C A T A L O G RETURN

и върху екрана ще се появи списък от имена на файлове:

```
DISK VOLUME 254
 A 003 ЗДРАВЕЙТЕ
*A 028 ЦБЕЙСИК
*B 011 МЕНЮ
*B 020 ОНФ
*A 004 ПРАВИ ТЕКСТ
*A 003 ЧЕТЕ ТЕКСТ
*A 010 КОПИРАНЕ
*B 006 КОПЧ.ОБКФ
*A 009 ЦВЕТОВЕ
*A 006 ДАНТЕЛА
*A 005 ДИАПОЗИТИВИ
*B 034 ЕДНА НЕПОЗНАТА
*B 034 ДОБРА ПОЗНАТА
*B 034 СЛАДУРЧЕ
*B 034 ИЗСТИСКВАНЕ
*A 005 НРБ
*A 025 РАЗРУШИ СТЕНАТА
```

Първата програма, която Ви е необходима се нарича ЦВЕТОВЕ. Намерете името ЦВЕТОВЕ в списъка на файловете / в каталога/. Сега напишете

RUN ЦВЕТОВЕ

след което натиснете RETURN. Екранът трябва да добие следния вид.

```
   ДЕМОНСТРАЦИОННИ ПРОГРАМИ ПРАВЕЦ
ЗА РАБОТА С ДЕМОНСТРАЦИЯТА, НАПИШЕТЕ
НОМЕРА И. СЛЕД ТОВА НАТИСНЕТЕ КЛАВИША
С НАДПИС 'RETURN' В ДЕСНИЯ КРАЙ НА
КЛАВИАТУРАТА. НАТИСНЕТЕ 'RETURN' И КОГАТО
ИСКАТЕ ДА СПРЕТЕ ДАДЕНА ДЕМОНСТРАЦИЯ.
1.  СТАНДАРТНИ ИМЕНА НА ЦВЕТОВЕТЕ
2.  СТАНДАРТНИ НОМЕРА НА ЦВЕТОВЕТЕ
3.        КАЛЕЙДОСКОП
4.     РИСУВАНЕ ПО ЕКРАНА

КАКВО ЩЕ ЖЕЛАЕТЕ ? ■
```

В света на компютрите такъв списък от номерирани описания се нарича "меню". Той работи подобно на номерираното меню в някои заведения. Вместо да изреждате "телешко варено, бифтек, малеби и кафе" вие просто си избирате №5.

Опитайте да изберете някоя от цветните демонстрации, като напишете номера й /след което, разбира се натискате RETURN/. За да се върнете обратно на "менюто", натиснете RETURN.

ПРЕКЪСВАНЕ НА ИЗПЪЛНЕНИЕТО НА ПРОГРАМАТА

За да прекъснете изпълнението на дадена програма използувайте

Това ще предизвика появата на средната дясна скоба и курсора. Средната дясна скоба е указание за това, че можете да продължите да въвеждате информация в компютъра.

След като изпълнението на програмата е било прекратено, то може да започне отначало, ако напишете

RUN

/и, разбира се RETURN, което едва ли има смисъл да Ви напомняме. В същност повече няма да правим това/.

Използувайте

за да спрете изпълнението на програмата и

RUN

за да я накарате да започне отначало. Опитайте това няколко пъти.

НАСТРОЙВАНЕ НА ЦВЕТОВЕТЕ НА ЕКРАНА

Ако "менюто" не е на екрана, заредете ДОС и предизвикайте изпълнението на програмата, наречена ЦВЕТОВЕ / в случай, че използувате дисково устройство/. Ако работите с касетофон, използувайте обичайната процедура и заредете касетата с надпис ЦВЕТОВЕ. Една от възможностите за избор в "менюто" се нарича ИМЕНА НА ЦВЕТОВЕТЕ. Ще използуваме тази част на програмата за настройка на цветовете на екрана. Напишете номера на ИМЕНА НА ЦВЕТОВЕТЕ и натиснете RETURN. На екрана се появяват известен брой светлинни ивици /евентуално цветни/. Под всяка от тях има четирибуквено съкращение на

име на цвят. Пълните имена са

0 черен	8 кафяв
1 тъмнопурпурен	9 оранжев
2 тъмносин	10 сив
3 пурпурен	11 розов
4 тъмнозелен	12 зелен
5 сив	13 жълт
6 син	14 аквамарин
7 светлосин	15 бял

Ако имате черно-бял монитор, нагласете контраста и яркостта докато изображението ви хареса. При цветните монитори са необходими малко повече настройки.

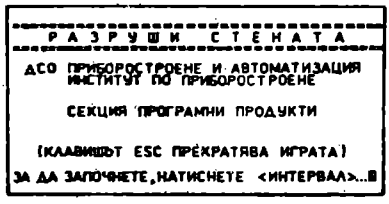
Когато цветовете на монитора са настроени добре, натиснете RETURN и на екрана ще се появи менюто. Сега опитайте № 2, който показва цветните ивици с техния числов код. Опитайте също и другите демонстрации.

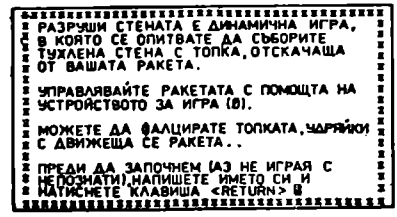
РАЗРУШИ СТЕНАТА

Предизвикайте изпълнението на програмата, която в каталога на системния диск е наречена РАЗРУШИ СТЕНАТА. Ако работите с касетофон качете в паметта на компютъра съдържанието на касетата със същия надпис и подайте команда

RUN

Екранът трябва да приеме вида, указан на лявата фотография. Сега натиснете интервалния клавиш и на екрана ще се появи описание на играта.





Когато бъдете помолени за това, напишете името си и натиснете RETURN. Върху екрана се появява полето за игра. Числата в долния край са точките, които получавате, когато ударите тухла от съответната вертикална ивица. Колкото повече навътре отивате в "стената", толкова повече точки получавате за една съборена тухла.

Ако името ви е по-дълго от 12 символа /заедно с интервалите/, програмата ще го съкрати до дължина, с която й е по-удобно да работи /12 символа/.

Случва се понякога неволно да натиснете RESET вместо RETURN; екранът започва целият да свети. Не изпадайте в паника. Просто напишете

RUN

Не забравяйте, че вече не споменаваме RETURN всеки път, когато е необходим.

Опитайте да направите нарочно няколко грешки, като например натиснете "случайно" RESET, за да добиете повече увереност във възможностите си за излизане от неприятни ситуации.

Да се върнем сега на нашата игра. Програмата ви дава указанието

ЗА ЗАПОЧВАНЕ НА ИГРАТА НАТИСНЕТЕ БУТОНА НА УСТРОЙСТВОТО ЗА ИГРА

така, че вземете в ръка въпросното устройство и започнете да играете.

Питате кое устройство ли? Опитайте и двете и ще разберете. Компютърът ще Ви съобщи, ако изберете не това, което трябва.

ГЛАВА II

НАЧАЛНИ ПОЗНАНИЯ ПО РАЗШИРЕН БЕЙСИК

Първо запознаване с командата PRINT
Формат при числата
Допълнителни сведения за RETURN
Лесни начини за редактиране:клавишите със стрелки
Цветове по екрана
Съобщения за грешка при команда PLOT
Рисуване на прави линии
Устройствата за игра
Памет: запознаване с променливите
Приоритет при аритметичните операции или
кой има предимство?
Как да избягваме грешки свързани с приоритета

ГЛАВА II

В случай, че имате дискови устройства, качете ДОС-а по някои от начините, описани в ГЛАВА I. Предполага се, че компютърът е в режим на БЕЙСИК-плаваща точка. В такъв случай в лявата част на екрана стои символът] / дясна средна скоба/, следван от мигащия курсор.

ПЪРВО ЗАПОЗНАВАНЕ С КОМАНДАТА PRINT

След като символът] заедно с курсора са в лявата част на екрана /и ДОС-ът е качен в случай, че имате дисково устройство/, можете да започнете използуването на РАЗШИРЕНИЯ БЕЙСИК. Напишете

```
PRINT "НАЧАЛО"
```

и на екрана ще се появи думата

```
НАЧАЛО
```

на следващия ред. В случай, че тя не се появи, си задайте въпроса "Не забравих ли да натисна RETURN?" Ако напишете неправилно думата"PRINT" на екрана ще се появи съобщението за грешка

```
? SYNTAX ERROR
```

Ако забравите първата или двете кавички, компютърът ще напише нула /познава се, че е нула по пресечната черта/:

```
Ø
```

Ако крайната кавичка е последният символ преди RETURN не е необходимо да я пишете: думата "НАЧАЛО" ще бъде написана независимо от това. Препоръчваме, все пак, да поставяте последните кавички. Такъв навик ще се окаже полезен ло-късно. В това ръководство ще предполагаме, че поставяте последните кавички.

Командата

```
PRINT "НАЧАЛО"
```

е указание за компютъра, което му съобщава да разположи върху екрана всички символи, намиращи се между кавичките. Можете да използувате командата PRINT, за да накарате компютъра да разположи върху екрана всяко желано от вас съобщение. Ако все

пак напишете много повече от 240 символа, компютърът ще започне да подава звуков сигнал, след това на екрана ще се появи обратна наклонена черта и ще можете да започнете отново.

```
]PRINT "РАБОТАТА С ТОЗИ КОМПЮТЪР ПРОИЗВЕ-
ДЕН В ПРАВЕЦ И НОСЕЩ СЪЩОТО ИМЕ ЧИЕТО
ПРОГРАМНО ОСИГУРЯВАНЕ СЕ ИЗРАБОТВА ОТ
СЕКЦИЯ ПРОГРАМНИ ПРОДУКТИ ПРИ ИНСТИТУТА
ПО ПРИБОРОСТРОЕНЕ КЪМ ДСО ПРИБОРОСТРОЕНЕ
И АВТОМАТИЗАЦИЯ КОЕТО Е В СИСТЕМАТА НА
МИНИСТЕРСТВОТО НА\
]
```

Сега опитайте командата

```
PRINT "150"
```

Компютърът послушно изписва числото 150 на следващия ред, както се очакваше.

Ако напишете

```
PRINT 150
```

компютърът отново написва числото, без да капризничи или да издава съобщение за грешка за липсващите кавички. Всъщност ПРАВЕЦ ще Ви позволи да изпълните командата PRINT, последвана от кое да е число, без да го заграждате в кавички.

Без по-нататъшно изучаване ПРАВЕЦ може да бъде използуван като прост настолен калкулатор.

Опитайте следната команда

```
PRINT 3 + 4
```

Отговорът 7 се появява на следващия ред. ПРАВЕЦ може да извършва пет различни елементарни аритметични операции:

1. СЪБИРАНЕ. Означава се с обиктовения знак "плюс" /+/.
2. ИЗВАЖДАНЕ. Използува се обикновеният знак"минус" /-/.
3. УМНОЖЕНИЕ. Много Хора използуват знака "X", за да оз-

начат операцията умножение. Този знак може да бъде объркан с буквата "Х". Някои хора използуват за тази цел точка /./, но тя може да бъде объркана с граматична точка или с десетична точка. Поради това ПРАВЕЦ използува звезда /*/. За да намерите на колко е равно 7 по 8 / в случай, че не си спомняте това/, просто напишете

```
PRINT  7 * 8
```

и опреснете паметта си.

4. ДЕЛЕНИЕ. Както обикновено, за целта се използува наклонена черта (/). За да разделите 63 на 7 напишете

```
PRINT  63 / 7
```

и правилният отговор ще се появи на екрана.
Опитайте да разделите 3 на 2. Отговорът е едно и половина. ПРАВЕЦ ви дава отговора в десетична форма: 1.5.
Трябва да кажем, че можете да извършите няколко аритметични операции в една команда. Например операторът

```
PRINT  3 + 5 + 9 + 4
```

е коректен. Точните правила, определящи начина на подобно използуване ще бъдат дадени по-късно, но ако желаете, можете да експериментирате и сега.

5. ПОВДИГАНЕ НА СТЕПЕН. Понякога се налага едно число да бъде умножено само по себе си определен брой пъти. Вместо да си правите труда да пишете

```
PRINT  4 * 4 * 4 * 4 * 4
```

можете да използувате стенографския запис

```
PRINT  4^5
```

Горната стрелка се набира от клавиатурата по следния начин:

Няма нищо странно в операцията повдигане на степен. Тя е просто съкратен запис на многократно умножение. В некомпютърни означения горният пример би изглеждал: 4^5.

```
150
]PRINT 150
150
]PRINT 3+4
7
]PRINT 7*8
56
]PRINT 63/7
9
]PRINT 3+5+9+4
21
]PRINT 4*4*4*4*4
1024
]PRINT 4^5
1024
]
```

Навсякъде, където използувате думата PRINT като команда, а не като част от текст, заключен между кавички, можете да използувате вместо нея символа ? /въпросителна/.

ФОРМАТ ПРИ ЧИСЛАТА

Напишете

PRINT 45.340

Вашият компютър ще отговори с

45.34

и няма да отпечата последната нула. ПРАВЕЦ не пише незначещи цифри /т.е. водещи преди десетичната точка и завършващи след десетичната точка нули/. Много, много малки по абсолютна стойност числа /намиращи се например между 0000000000000000000003 и -.000000000000000000000000000000003 ще бъдат превърнати в нула. /Надяваме се, че това е верният брой нули/. По-лесен начин да напишете тези числа е 3*10^-39 и -3*10^-39. Не го приемайте на доверие. Опитайте сами. Сега напишете

PRINT 985788.6898

Изненада. Последните две цифри са премахнати и числото, което е останало е най-близкото приближение до началното число, което компютърът може да измисли. Този процес се нарича "закръгляване". Опитайте да напишете

```
PRINT 788.6898
```

Вашият компютър този път не закръглява числото, а Ви го връща такова, каквото сте му го дали. Казвате, че това е лудост? Но в тази лудост има система. Числата се заклюглява само ако имат повече от девет значещи цифри. Всяко число, което има помалко от десет значещи цифри няма да бъде закръглявано. Компютърът прави всичко, на което е способен, но той разполага само с девет значещи цифри.

Ако подадете команда PRINT с дълго число, например

```
1234567890
```

ПРАВЕЦ отговаря с

```
1.23456789E+09
```

Означенията 1234567890 и 1.23456789E+09 представят едно и също число. Наистина. Числото, написано от Вашия компютър е зададено в "научен вид". Ако имате нужда от такива числа, сигурно знаете и как да ги четете. Подробната литература към ПРАВЕЦ ще даде по-точна информация за тези, които се интересуват от подобни странни означения.

Опитайте същото и с други числа. Колко цифри трябва да има едно число без десетична точка, преди ПРАВЕЦ да го превърне в "научен"вид? Ако научните означения ви изглеждат твърде сложни, не се тревожете. По всяка вероятност няма да ви се налага да ги използувате за известно време. Помнете, че всяко число може да бъде написано от компютъра във вида, в който Вие сте го задали, ако в командата PRINT бъде заградено с кавички. Трябва да се знае все пак, че ПРАВЕЦ не може да използува числата в кавички за аритметични операции.

ДОПЪЛНИТЕЛНИ СВЕДЕНИЯ ЗА RETURN

До сега Вие упорито натискахте RETURN след всеки ред. Смятаме, че е време да Ви съобщим защо този клавиш е толкова претоварен. Причината е проста: без RETURN компютърът няма да знае, че сте завършили писането на командата. Например, ако започнете да пишете

```
PRINT  4 + 5
```

и компютърът незабавно напише 9, това ще Ви бъде неприятно в случай, че сте искари да напишете

```
PRINT 4 + 5 + 346
```

което очевидно дава друг резултат. Тъй като компютърът не може да отгатне кога писането на една команда или инструкция е напълно завършено, това трябва да му бъде указано от Вас. Именно това и правите, когато натиснете клавиша RETURN. Тъй като трябва да правите това след всяко указание за компютъра, то ние /както вече сте забелязали/ не Ви напомняме за това всеки път. Смятаме, че ако сте правили на практика всички примери до тук, натискането на RETURN трябва да Ви е станало вече навик.

Ние наистина се надяваме, че сте опитвали всички примери. Обучението по програмиране много прилича на обучението по каране на велосипед, свирене на пиано или игра на футбол. Можете да изчетете всички книги в света, свързани с карането на велосипед и да бъдете голям "книжен специалист". Но цялото това четене малко ще ви помогне, когато се качите на велосипед за пръв път в живота си. След като веднаж сте се научили да карате велосипед от собствен опит /което може да бъде малко болезнено/, Вие можете да отидете почти навсякъде. Същото се отнася и до програмирането. Можете да прочетете това ръководство и да мислите, че го разбирате. Да програмирате обаче няма да можете. Само ако правите всеки от дадените примери ще се научите да програмирате. Истината е такава.

ЛЕСНИ НАЧИНИ ЗА РЕДАКТИРАНЕ :
КАКВО ДА ПРАВИМ ПРЕДИ ДА НАТИСНЕМ RETURN

Никой не е съвършен машинописец. Всички правим грешки/ Е..., разбирате какво искам да кажа/. ПРАВЕЦ има няколко особености, които помагат да поправите грешките си и по такъв начин Ви спестяват усилието да преписвате цял ред за всяка глупава грешка. Именно тук влизат в ролята си клавишите с лява и дясна стрелка.

Клавишът ⟵ твърде много прилича на възвратния клавиш на пишещата машина и затова ние ще го наричаме "възвратен клавиш". Няколко опита ще поясният това. Напишете /точно както е показано/ командата:

```
PRINT  KOMPUTER "
```

и, както обиктовено, натиснете RETURN. Компютърът ще отговори

```
Ø
```

поради липсващата кавичка. Ако беше написано

```
PRINT "COMPUTER"
```

компютърът щеше да отговори с

```
COMPUTER
```

Не приемайте казаното на доверие. Опитайте. Сега, без да натискате RETURN, напишете "грешното" указание:

```
PRINT "КОМПУТЪР"
```

Тъй като не сте натиснали RETURN още нищо не се е случило. Както е показано на снимката, курсорът стои отдясно на последната кавичка. /За съжаление не можем да накараме фотографията да мига/.

```
]PRINT COMPUTER"
Ø
]PRINT "COMPUTER"
COMPUTER
]PRINT "COMPUTER" ■
```

За да промените думата КОМПУТЪР и да я направите КОМПЮТЪР, можете да използувате [←]. Забележете, че всеки път, когато натиснете този клавиш, мигащият курсор се премества с една позиция назад /вляво/. Придвижете курсора до У. Сега напишете Ю. Както вишдате Ю замества У. Натиснете RETURN. Получихте

КОМПЮ

от компютъра. Това е защото сте преминали със заден ход на курсора през "ТЪР". Ако преминете със заден ход на курсора през даден символ от реда, който пишете в момента, то при натискане на RETURN този символ няма да бъде подаден на компютъра. Едно решение на проблема е да поправите У като се върнете с курсора до него и след това напишете

[SHIFT]

[Т] [Ъ] [P] [2] [RETURN]

Опитайте.

Работи! Има обаче и по-десен начин. Когато натиснете клавиша [→], курсорът се движи надясно. Когато курсорът се движи надясно през някой символ, това има ефект на написване на този символ. Клавиша [→] ще наричаме "възстановяващ клавиш". Напишете отново

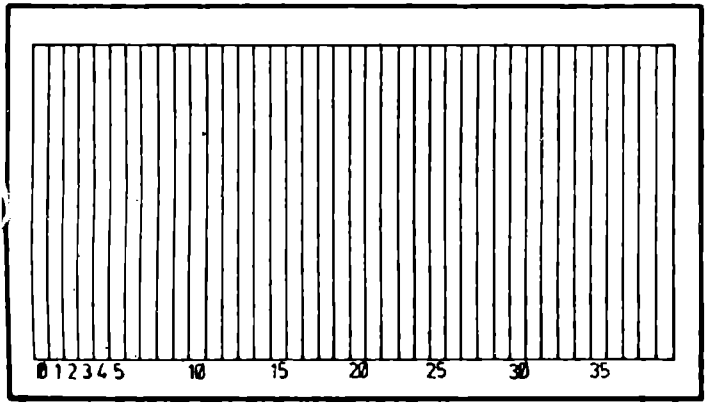
PRINT "КОМПУТЪР"

след това върнете курсора до У и го променете на Ю. За да завършите поправката просто натиснете дясната стрелка пет пъти и след това натиснете RETURN.

Всичко в ред ли е? Употребата на възвратния и възстановяващия клавиш ще Ви спести много време. Отработете този въпрос, като ги използувате няколко пъти за поправка на Ваши собствени "грешки", така че да свикнете добре с тези клавиши.

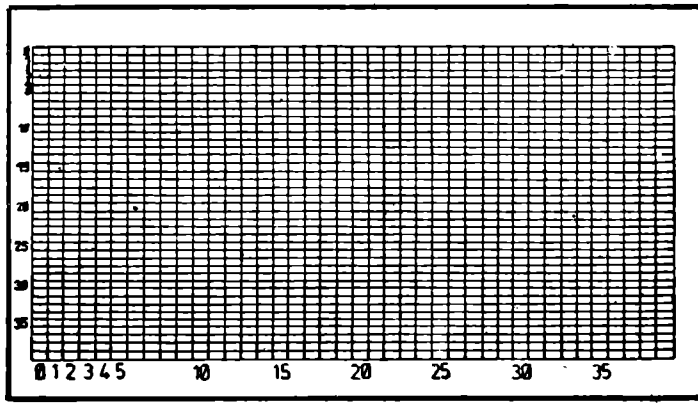
ЦВЕТОВЕ НА ЕКРАНА

За да разположим цветно изображение върху екрана, трябва да имаме някакъв начин да опишем кой от 16-те достъпни цветове желаем да използуваме и къде искаме да го разположим върху екрана. За да можем да укажем мястото на даден цвят, ние разделяме екрана на 40 вертикални ивици, номерирани от 0 до 39. Ивицата с номер 0 се намира в най-левия край на екрана и номерата на ивиците растат надясно. Може би се учудвате защо номерирането не е направено от 1 до 40, вместо от 0 до 39. Когато добиете повече опит в програмирането, ще откриете, че изборът, който сме направили, е в известен смисъл по-удобен, макар, че може би, на пръв поглед не изглежда така.



Екранът е също така разделен на 40 хоризонтални редове, отново номерирани от 0 до 39. Редовете започват с ред 0, разположен в горната част на екрана. Тези редове пресичат вертикалните ивици и по такъв начин разделят всяка от тях на 40

"тухлички", номерирани от 0 /най-горната/ до 39 /най-долната/. Тези, които предпочитат стандартната терминология, ще разберат, че това е просто правоъгълна декартова координатна система. Тези, които не обичат да си служат със специални термини, могат да мислят в термините на тухли и ивици.



За да получите възможност да използувате екрана по живописен начин, напишете следната команда:

GR

Разбира се, не сте забравили да натиснете RETURN. Когато използувате тази команда екранът се изчиства, като оставя за текст само най-долните четири реда. За да възстановите старото положение на нещата /т.е. положението преди да напишете GR/ използувайте командата

TEXT

Когато подадете тази команда, целият екран внезапно се изпълва с много знаци (@) Това е нормално. Опитайте да напишете командата TEXT и след това да се върнете обратно в графичен режим, написвайки командата GR.

Преди да разположите цветна точка върху екрана, трябва да укажете на компютъра какъв е желаният от вас цвят. Имате на разположение шестнадесет цвята. Вече сте ги виждали: те са номерирани от 0 до 15, както е показано в

ЦВЕТОВЕ

Да предположим, че искате да поставите някъде зелена точка. Трябва първо да напишете командата GR и след това да напишете

```
COLOR = 12
```

Това означава, че всяка цветна точка /или петно, или тухла/, която разполагате върху екрана, ще бъде зелена. В същност, докато не получи друга инструкция, компютърът ще разполага всяко нещо върху екрана в зелен цвят. С изключение, разбира се, на малката зона под екрана, която е запазена за Вашите команди. За да разположите цветно петно в горния ляв ъгъл на екрана /най-лява или нулева ивица, най-горна или нулева тухла/, напишете

```
PLOT 0,0
```

За да поставите петно със същия цвят в горния десен ъгъл, трябва да укажете ивица 39, тухла 0. Така че напишете

```
PLOT 39,0
```

Забележете, че винаги задавате първо номера на ивицата. Сега поставете оранжева тухла в долния ляв ъгъл. Първо променете цвета. Запомнете - трябва наистина да правите тези упражнения, а не само да си мислите за тях. Така че, протегнете пръсти и напишете

```
COLOR = 9
```

Нищо не се променя в горната графична част на екрана /даже, ако не сте забравили да натиснете RETURN/. Но компютърът помни, че когато следващият път получи команда PLOT, тя трябва да бъде изпълнена в оранжево, а не в зелено. След като сте избрали цвета, можете да поставите една точка в долния ляв ъгъл. Това е ивица 0, тухла 39:

```
PLOT 0, 39
```

Всичко в ред ли е? Забравихте ли да натиснете RETURN? Оранжевото Ваш любим цвят ли е?

Сега поставете точка в тъмнопурпурен цвят в долния десен ъгъл. Помислете сами как да го направите.

СЪОБЩЕНИЯ ЗА ГРЕШКА ПРИ КОМАНДА PLOT

Има две съобщения за грежка, които лесно могат да се появят, когато използувате командата PLOT. Вече знаете, че ако напишете

```
PLAT
```

или

```
PLOP
```

вместо

```
PLOT
```

ще получите съобщението

```
? SYNTAX ERROR
```

Ново съобщение за грешка се появява, когато в PLOT-команда зададете координата по-голяма или по-малка от разрешените за това стойности.

Напишете

```
PLOT 13,85
```

и ще получите съобщението

```
? ILLEGAL QUANTITY ERROR
```

Това съобщение означава, че сте се опитали да поставите точка с недопустими координати и извън екрана. Най-големите числа, които можете да използувате в команда PLOT са 39 за първата координата и 47 за втората. Използуването на числа по-големи от 29 за втора координата, като например

```
PLOT 2Ø,45
```

ще доведе просто до появата на особени символи в текстовата област в долната част на екрана.

Всеки опит да се използуват отрицателни числа в команда PLOT е друг начин да се получи съобщението

```
? ILLEGAL QUANTITY ERROR
```

РИСУВАНЕ НА ПРАВИ ЛИНИИ

Да предположим, че искате да нарисувате светлосиня хоризонтална черта от ивица 5-та до ивица 9-та на нивото на тухлата с номер 14. Бихте могли да напишете

```
COLOR = 7
PLOT 5,14
PLOT 6,14
PLOT 7,14
PLOT 8,14
PLOT 9,14
```

Забележете, че междините между две съседни тухли не се виждат и те образуват непрекъсната линия. Има обаче и по-лесен начин да се нарисува хоризонтална черта. И добре, че има. Да предположим, че искате да прекарате тъмнозелена хоризонтална черта по средата на екрана. Ако използувате дългия начин, ще ви се наложи да напишете четиридесет команди:

```
COLOR = 4
PLOT 0,20
PLOT 1,20
PLOT 2,20
```

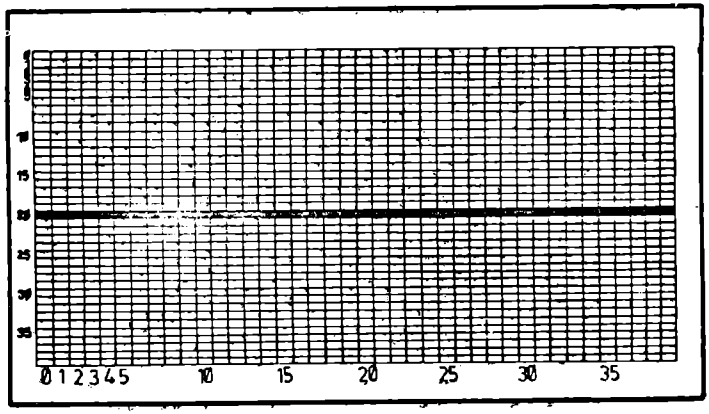
и т.н- до

```
PLOT 39,20
```

По-лесният начин е този. Просто напишете

```
COLOR = 4
HLIN  0,39 AT 20
```

Натиснете RETURN и ето я пред вас: незабавно появяваща се хоризонтална черта от ивица 0 до ивица 39 на нивото на тухла 20.

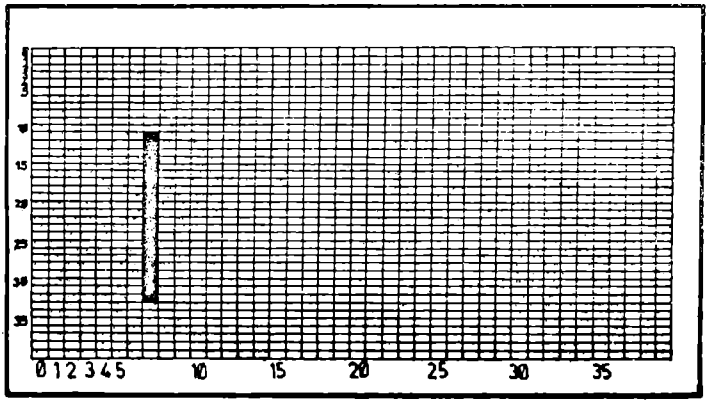


Сега опитайте пурпурна черта от ивица 19 до ивица 28 на нивото на тухла 18. Опитайте и няколко други. След като направите около 6 различни хоризонтални черти, това занимание би трябвало да почне да ви омръзва.

Забележете, че когато поставите цветна точка или линия на мястото на вече съществуваща точка или линия, новият цвят се налага, а старият изчезва. За да изчистите целия екран от всичко едновременно, използувайте командата GR.

Предвидено е автоматично поставяне на вертикални черти, аналогично на хоризонталните. За да прекараме оранжева вертикална черта на нивото на ивица 7-ма от тухла 12 до тухла 33 пишем:

```
COLOR = 9
VLIN 12,33 AT 7
```



Направете още няколко вертикални линии, като променяте номерата на извиците и редовете. Можете да изпитате сръчността си едновременно с вертикалните и хоризонталните линии, като нарисувате пурпурна рамка около екрана с пет команди. След това поставете върху екрана зелен кръст. Опитайте да нарисувате няколко линии с COLOR = Ø. Поиграйте си с PLOT, HLIN, VLIN. Ползата за Вас от това ръководство ще изчезне за пет секунди, ако не експериментирате с тези команди.

УСТРОЙСТВАТА ЗА ИГРА

Вземете в едната си ръка устройството за игра, с което си служехте при играта РАЗРУШИ СТЕНАТА. С другата ръка напишете

```
PRINT PDL (Ø)
```

и на екрана ще се появи число. Завъртете малко потенциометъра. Сега напишете

```
PRINT PDL (Ø)
```

отново. Правете опити с последователно въртене на потенциометъра и подаване на командата

```
PRINT PDL (Ø)
```

Ако числото не се изменя, вие сте взели не това устройство за игра, което е било нужно. Кое е най-голямото и най-малкото число, което можете да получите по този начин? Коя е най-малката промяна, която можете да предизвикате?

Можете да установите положението на другото устройство за игра, като подадете команда PRINT с PDL (1).

PDL е функция. Под функция се разбира нещо, което взема едно или няколко числа, извършва някакви операции върху тях и връща някаква стойност. Числата, които функцията използува се наричат нейни аргументи и винаги се поставят в скоби след името на функцията. PDL е функция, която има един аргумент - номера на устройството за игра. В програмирането се казва, че числото, което една функция намира, се връща в програмата.

ПАМЕТ
ДРУГИ ВЪЗМОЖНОСТИ ЗА СМЯТАНЕ

При много прости калкулатори е възможно да запишете в паметта число, което ще Ви потрябва по-късно. За да направите това, поставяте числото на специално място в калкулатора - място, което засега ще наричаме преградка. Обикновено това се прави като се натисне клавиш, означен "M". Същата операция можете да извършите и с ПРАВЕЦ. Например, за да запазим стойността 77, написваме

```
M = 77
```

Стойността 77 не се изписва на екрана. Тя просто е складирана в преградката, наречена M. Ако сега напишете

```
PRINT M
```

компютърът ще напише стойността на M. Опитайте да подадете на компютъра горните две команди.

Сега напишете

```
M = 324
```

и подайте команда за изписване върху екрана на стойността на M. Тя е 324, нали? Какво се е случило с числото 77? То си е отишло завинаги. Преградката може да пази в себе си само една стойност. Когато поставите в нея нова стойност, старата стойност се изтрива.

Напишете

```
PRINT "M"
```

Какво става?Има голяма разлика между

```
M
```

и

```
"M"
```

Тя е напълно аналогична на разликата между следните две изречения:

```
ГОШО ИМА ЧЕТИРИ ЛЕВА
"ГОШО" ИМА ЧЕТИРИ БУКВИ
```

В единия случай става дума за нашия познат Гошо, а в другия - за самата дума. Такъв смисъл имат кавичките на компютърен език. Когато кажем

```
PRINT "M"
```

имаме пред вид отпечатването на самата буква. Когато кажем

```
PRINT M
```

имаме предвид отпечатването на това, което тя изразява. Вие никога няма да объркате някого, когото обичате със собственото му име.

Можете да складирате в преградка и резултата от някакво пресмятане. Например:

```
M = 4 + 5
```

Можете да се убедите, че отговорът е бил складиран в M като отпечатате стойността на M.

Можете също да използувате стойността на M за по-нататъшни пресмятания.

Например опитайте на вашия ПРАВЕЦ това:

```
PRINT  M + 2
```

Получи ли се това, което очаквахте? Опитайте да направите и други пресмятания с M.

Простият калкулатор има само една преградка. Компютрите имат стотици преградки /ПРАВЕЦ има /. Официалното название на преградките е"променливи".В това название има нещо подвеждащо, тъй като преградките не се държат като "променливите" в математиката. Те са много по-прости. Всяка от тях е просто място, където се складира някаква стойност. Ние все пак ще се съобразим с общоприетото название. Когато става въпрос за променливи в програмирането, ще имаме пред вид горното описание. При ПРАВЕЦ всички променливи имат стойност нула, докато не поставите нещо в тях.

Една преградка може да има почти всяко име, стига то да започва с буква.

Например:

```
SUM = 56 + 34 + 1525 + 8
GAMEPOINTS  = 45
PLAYER  = 9
```

Някои имена не са позволени, тъй като съдържат в себе си дума, която за ПРАВЕЦ има специално значение. Такива думи се наричат "запазени". Една от тях е "COLOR". Опитйте да напишете

```
THISCOLOR = 6
```

или

```
COLORFUL = 9
```

Всичко, което ще получите за тези усилия, е съобщение за грешка. Когато използуването на име на променлива ви даде съобщението ? SYNTAX ERROR, това означава, че неразумно сте се опитали да включите запазена дума в име на променлива. В такъв случай не се притеснявайте. Просто изберете друго име.

Списък на "запазените" думи, които не могат да бъдат използувани като имена на променливи или като част от имена на променливи, може да бъде намерен в Приложение Б в края на това ръководство.

Освен запазените думи, имената на променливите не могат да съдържат контролни символи и кирилица. Голяма част от буквите на кирилицата са неразличими върху екрана от съответните букви на латиницата. Компютърът обаче ги възприема по различен начин и резултатът от такова объркване /вмъкване на символи от кирилицата в име на променлива /неизбежно е съобщението ? SYNTAX ERROR. Същото съобщение се получава и когато символи от кирилицата са вмъкнати в самите запазени думи като TEXT, PLOT, PRINT, COLOR и т.н. Например напишете

```
PRINT "А"
```

като при буквите Р и А компютърът е превключен на кирилица.
Отговорът е

```
?SYNTAX ERROR
```

Сега напишете отново

```
PRINT "A"
```

като този път сте на кирилица само когато пишете буквата А. Този път компютърът послушно изписва

A

Това е така, защото буквите от кирилицата, както и всички останали символи, могат безпрепятствено да бъдат включвани между кавичките при команда PRINT.
Ако напишете обаче

PRINT A

като отново само символът А е от кирилицата, ще получите съобщение за грешка ?SYNTAX ERROR. Проверете как реагира ПРАВЕЦ на други Ваши "грешки" от този вид, като за целта използвате в подходящи команди запазените думи TEXT, PRINT, COLOR.

Несъмнено Вас не ви блазни перспективата да коригирате всяка подобна грешка чрез повторно написване на целия оператор с повишено внимание. И не е нужно. Съществува значително по-ефективен начин да направите това. Напишете без да натискате RETURN оператора

PRINT "COMPUTER"

като навсякъде, където това е възможно, пишете вместо латинските букви графично съвпадащите с тях символи от кирилицата. Сега върнете курсора в началото на оператора и с помощта на дясната стрелка ⇨ го накарайте отново да премине последователно през всеки символ. Какво става? "Неподправените" символи се държат нормално, когато курсорът е върху тях, а с "фалшивите" настъпват странни промени. Именно това е начинът те да бъдат разпознавани, а тъй като точно в този момент курсорът е върху тях, поправянето им не представлява трудност. Направете още няколко експеримента с подобни "грешки" и поправянето им. Забележете, че не е наложително да поправяте такива грешки, когато става въпрос за символи, заградени с кавички - резултатът от действието на оператора в този случай не се променя.

Когато избирате имена на променливите, старайте се те да изразяват смисъла, в който съответната променлива ще бъде употребявана. Това ще ги прави по-лесни за запомняне.

Напишете

```
BIRD = 11
```

и след това

```
PRINT BIRD
```

Това ли е, което очаквахте? Сега напишете

```
PRINT BITE
```

Какво става? Опитайте

```
PRINT BILL
```

и

```
PRINT BILLOW
```

Ако разгледате тези имена на променливи, ще установите, че всички започват с "BI". ПРАВЕЦ-версията на БЕЙСИК използува само първите два символа от името на всяка променлива, за да я различава от другите променливи. Затова името

```
BIRD
```

обозначава същата променлива, както и

```
BITE
```

и т.н.

Ето едно полезно средство. Да предположим, че имате някаква стойност в променливата PRICE и искате да увеличите тази стойност с 5. Един начин да направите това е да отпечатате на екрана стойността на PRICE, след това да добавите 5 към тази стойност и накрая да запишете получената стойност обратно в PRICE. Например

```
PRICE = 28
PRINT PRICE
PRINT 28 + 5
PRICE = 33
```

Но вижте колко по-лесно е да напишете

```
PRICE = 28
PRICE = PRICE + 5
```

Опитайте следните команди по реда, в който са дадени:

```
PRICE = 2
PRINT PRICE
PRICE = PRICE + 3
PRINT PRICE
PRICE = PRICE * 6
PRINT PRICE
PRICE = PRICE / 10
PRINT PRICE
```

В края на тази поредица от команди по всяка вероятност ще получите стойност 3. Правилно ли е това? Това ли е, което очаквахте? Опитайте следната последователност:

```
APPLES = 55
BANANAS = 11
QUOTIENT = APPLES / BANANAS
PRINT QUOTIENT
```

Първо помислете какъв отговор очаквате и след това проверете дали е така. Ако не е, разберете защо. Накрая опитайте следните команди:

```
HELLO = 128
PRINT"HELLO"
HELLO = HELLO / 2
PRINT "HELLO"
HELLO = HELLO / 2
PRINT HELLO
```

Какво очаквахте? Какво получихте?

ПРИОРИТЕТ НА АРИТМЕТИЧНИТЕ ОПЕРАЦИИ, ИЛИ КОЙ ИМА ПРЕДИМСТВО?

На приемите в миналото сервирането е ставало съгласно установен и стриктно спазван план: първо на почетния гост,

след това на дамите /по реда на ранговете на техните съпрузи/, след това на мъжете /по реда на ранговете им/ и накрая се е сервирало на домакина. Независимо от това кой къде е седнал, сервитьорът минавал между тях, намирайки този гост, чийто ред е било да бъде обслужен. Бихме могли да кажем, че между събралите се е имало отношение на приоритет. При едно просто пресмятане като

```
PRINT 4 + 8 / 2
```

не бихте могли да кажете дали резултатът би трябвало да е 6 или 8, без да знаете реда /или приоритета/ на аритметичните операции. Ако добавите 4 към 8 получавате 12. Ако след това разделите 12 на 2, получавате 6. Това е единият възможен отговор. Ако обаче добавите 4 към осем-делено-на-две, ще получите 4 плюс 4 или 8. Това е другият възможен отговор. Осем е отговорът, който ще даде Вашия ПРАВЕЦ. Ето как той избира реда, по който да извършва аритметичните операции:

1. Когато е използуван знак минус, за да укаже отрицателно число, например

```
-3+2
```

компютърът първо прилага знака минус към съответното число или променлива. По такъв начин -3 + 2 става равно на -1. Ако ПРАВЕЦ извършваше първо събирането -3 + 2 щеше да стане равно на -5, а това не е така. Друг пример е

```
BRIAN = 6
PRINT -BRIAN + 10
```

Отговорът е 4. /Забележете обаче, че в израза 5 - 3 знакът минус означава изваждане, а не отрицателно число/.

2. След като е приложил всички знаци минус, ПРАВЕЦ извършва повдиганията на степен. Изразът

```
4 + 3 ^ 2
```

се пресмята, като 3 се повдигне на квадрат /три по три е девет/, след което се прибавя 4, за да се получи забележителният окончателен резултат 13. Когато има няколко последователни повдигания на степен, те се извършват отляво надясно, така че

```
2 ^ 3 ^ 2
```

се пресмята, като 2 се умножава само по себе си три пъти /2*2*2/, което е осем и последното се умножава само по себе си. Отговорът е 64.

3. След като всички повдигания на степен са били извършени, се извършват всички умножения и деления отляво надясно. Аритметичните операции с еднакъв приоритет винаги се извършват отляво надясно. Умножението /*/ и делението /// имат еднакъв приоритет.

4. Накрая се извършват всички събирания и изваждания отляво надясно. Събирането /+/ и изваждането /-/ имат еднакъв приоритет.

Да резюмираме приоритета на аритметичните операции при ПРАВЕЦ:

Първо: - /знаците минус, означаващи отрицателни числа/
Второ: ^ /повдигания на степен, отляво надясно/
Трето:/* /умножения и деления, отляво надясно/
Четвърто: +- /събирания и изваждания, отляво надясно/

По-долу ще намерите няколко аритметични израза за пресмятане. Пресметнете всеки от тях първо на ум /или с джобен калкулатор, или с лист и молив/ и след това го пресметнете с помощта на ПРАВЕЦ. Ако Вашият резултат е различен от този на компютъра, опитайте се да откриете защо. Ние ще Ви дадем само аритметичните изрази. Вие ще трябва да поставите по едно PRINT или ? пред всяко от тях, за да накарате компютъра да ги пресметне. Ако нямате богат опит в тази област трябва да направите предложените упражнения на практика. Не бива да пресметнете всичко наведнаж и след това да правите проверка с компютъра. Извършете всяко упражнение първо на ръка и след това го пресметнете с компютъра. След това преминете на следващото и т.н.

```
3 + 2
4 + 6 - 2 + 1
8 * 4
4 ^ 2 + 1
6 / 4 + 1
5 - 4 / 2
4 / 2 - 2
6 * -2 + 6 / 3 + 8
4 + -2
```

12 + 2 + 8 = -2

```
2 ^ 2 ^ 3 + 1
2 * 2 * 3 + 1
2 * 2 + 1 * 3
2 * 2 * 1 + 3
8 / 2 / 2 / 1
8 * 2 / 2 + 3 * 2   2 * 1
20 / 2 * 5
```

Няма да даваме решения. Вашият ПРАВЕЦ ще Ви даде правилните отговори.

КАК ДА ИЗБЯГВАМЕ ГРЕШКИ, СВЪРЗАНИ С ПРИОРИТЕТА

Да предположим, че искате да разделите 12 на четири-плюс-две. Ако напишете

```
12 / 4 + 2
```

ще получите 12-делено-на-четири и след това добавено две. Но Вие искате не това. За да постигнете целта си, можете да напишете

```
12 / (4 +2)
```

Скобите изменят приоритета. Правилото, което компютърът следва, е просто: извършвай първо това, което е в скоби. Ако има скоби в скоби, извършвай първо това, което е в най-вътрешните скоби. Ето един пример:

```
12 / (3 + (1 + 2)^2)
```

В този случай, извършвайки действията в най-вътрешните скоби, Вие първо събирате 1 + 2. Сега изразът вече е

```
12 / (3 + 3 ^ 2)
```

Но вие знаете, че 3 + 3^2 е 3 + 9 или 12, така че изразът се опростява до 12 / 12, което е едно.

В случай като (9 + 4) * (1 + 2), където има повече от една двойка скоби, но те не са вложени една в друга, действията се извършват просто отляво надясно. В случая изразът се редуцира до 13 * 3, което е 39.

Ето няколко израза за пресмятане. Отново, ако не сте запознати с компютрите, няколкото минути, които в същност ще за-

губите и проверката чрез ПРАВЕЦ ще бъдат много ценни. Вашите усилия ще бъдат добре възнаградени с това, че ще можете да използувате компютъра по-ефективно. Повечето от тези правила за приоритет и скоби случайно са валидни за почти всички компютърни системи навсякъде по света, а не само за ПРАВЕЦ.

```
44 / ( 2 + 2 )
(44 / 2 ) + 2
3 + ( -2x2 )
(3 + -2)  x 2
100 / (200 / (1 x (9 - 5)))
32 / (1 + (7 / 3)  + (5 / 4))
```

ГЛАВА III

ПРОСТО ПРОГРАМИРАНЕ

Отложено изпълнение:
Просто редактиране:
Лесна акробатика: цикъл чрез GOTO
Някои неща облекчават живота: допълнителни моменти при редактирането
Подвижният курсор:редактиране чрез клавиша ESC
Няколко думи за изучаването на разширения БЕЙСИК
Предотвратен нещастен случай
Истината: аритметични и логически твърдения
Приоритет при операциите
Командата IF
Записване на програми върху диск:
Записване на програми с касетофон:
Други графични програми
Образуване на цикли посредством FOR/NEXT
Една неправилна програма
Механичен начин за преорганизиране на цикли
Последен пример на вложени цикли
С блясък в очите
Художествено оформление:запетая, точка и запетая

ГЛАВА III
ОТЛОЖЕНО ИЗПЪЛНЕНИЕ

Не, в този раздел не става дума за отлагане на екзекуцията на престъпник в последния момент. Досега, когато напишехте

```
PRINT 3 + 4
```

и натиснете RETURN, компютърът правеше незабавно това, което му е било указано. В този случай той прави непосредствено изпълнение на всяка команда, която напишете посредством клавиатурата.

На път сте да научите как да натрупвате команди, които да бъдат изпълнени по-късно /отложено изпълнение/. За да сте сигурни, че паметта на компютъра е изчистена от всякакви предишни програми, напишете

```
NEW
```

Както почти всичко, което сте виждали, NEW трябва да бъде последвано от RETURN. За да накарате компютъра да запази в паметта си дадена команда, напишете пред нея някакво число. Например, ако напишете

```
100 PRINT 3 + 4
```

видимо нищо не се случва, даже ако натиснете RETURN. ПРАВЕЦ е запазил командата. За да се убедите, че я е запазил, подайте инструкцията

```
LIST
```

Опитайте. Освен ако не сте направили печатна грешка /наградата за това по всяка вероятност е съобщението ?SYNTAX ERROR/, на екрана се появява

```
100 PRINT 3 + 4
```

Сега напишете командата

```
RUN
```

и отговорът

7

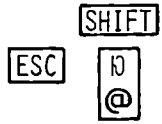
се появява на екрана.

Написването на командата RUN предизвиква изпълнението на Вашата записана команда, но компютърът продължава да я помни. Можете да го накарате /по същия начин/ да я изпълни колкото пъти пожелаете. Опитайте.

Нещо повече. Компютърът не забравя записаната команда, дори когато изчистите екрана. Ето един нов начин да изчистите екрана:

HOME

Командата HOME има същия ефект, както и



което ви е известно отпреди, но нейното изпълнение може да бъде както непосредствено, така и отложено. За да проверите това, напишете

100 HOME

Сега, когато напишете

RUN

компютърът предано изпълнява записаната команда и изчиства екрана. Напишете

NEW

и след това

LIST

и вижте какво се е случило. Командата NEW е предизвикала загубата на записаната команда. Напишете

RUN

и нищо не се появява на екрана. Това е защото Вашата стара команда е била изтрита от командата NEW.

Възможно е да натрупате много команди, ако давате на всяка от тях различен номер. Напишете например това:

```
1 PRINT "HELLO"
2 PRINT  4^5
3 PRINT 67 / 12
```

За сега нищо не се случва. Сега напишете

```
RUN
```

и наблюдавайте появата на резултата.

```
]NEW
]LIST

]RUN
]1 PRINT "HELLO"
]2 PRINT 4^5
]3 PRINT 67/12
]RUN
HELLO
1024
5.58333334
]
```

Числата, които пишем пред командите /операторите/ се наричат номера на редовете. Други разпространени наименования на същото нещо са номер на оператор и адрес. Компютърът натрупва и изпълнява командите по реда на нарастващите номера на редовете. За да проверите това на практика, изтрийте чрез командата NEW операторите, които са се натрупали досега в паметта на компютъра и след това напишете операторите:

```
1 PRINT   "Р"
Ø PRINT   "П"
4 PRINT   "Ц"
5 PRINT   "Е"
2 PRINT   "В"
```

За да видите какво се е получило в паметта, напишете

```
LIST
```

Забележете, че не е наложително да разглеждате на набора от команди, преди да подадете командата за изпълнение RUN. Това просто е една полезна практика.

Всеки набор от команди, който се изпълнява, когато напишете RUN, се нарича програма.

Програмата, която току-що написахме имаше за цел да напише

```
П
Р
А
В
Е
Ц
```

върху екрана, но, както изглежда, някоя команда PRINT е изпусната. Как можем да я добавим? Само като препишем операторите с номера 2,3 и 4 като оператори с имена 3,4 и 5 и добавим нов оператор с номер 2. За да извършите поправката напишете следното:

```
2 PRINT   "А"
3 PRINT   "В"
4 PRINT   "Е"
5 PRINT   "Ц"
```

Разгледайте програмата/с команда LIST/, за да видите какво се е получило. Сигурно ви е направило впечатление, че в какъвто и ред да въвеждате операторите, ПРАВЕЦ ги подрежда по реда на техните номера. Сега подайте команда RUN за изпълнение на програмата.

Досадно е, че трябваше да преписваме толкова оператори, само за да добавим още един по средата на програмата. Поради това е добре да си създадете полезния навик да оставяте свободни номера между номерата на два последователни оператора, а също така да оставяте свободни номера преди първия оператор. Напишете

```
NEW
RUN
```

за да изтриете старата програма и въведете друга:

```
100 PRINT "К"
110 PRINT "Ш"
```

Изпълнението на тази програма не довежда до изписването на цялата дума "КОШ" по вертикала. Но сега можете да се върнете и да напишете

```
105 PRINT  "O"
```

Подайте последователно командите LIST и RUN. Отсега нататък в тази книга ще започваме да пишем програмите от разумно голям номер на първия ред и ще оставяме много място между номерата на два последователни оператора, за да можем винаги да добавяме между тях нови оператори.

ПРОСТО РЕДАКТИРАНЕ

По-рано открихте, че командата

```
PRINT PDL (0)
```

изписва на екрана число, което съответствува на текущото положение на някое от устройствата за игра. Необходими бяха твърде много PRINT-ове, за да научите достатъчно за това устройство. Сега, когато можете да пишете програми, животът е много по-лесен. Изчистете паметта на компютъра с

```
NEW
```

и пишете

```
100 PRINT PDL (0)
```

Всеки път, когато напишете RUN, тази къса програма се изпълнява и Вие можете да видите положението на устройството за игра.

Записаната програма вече Ви спестява известен труд, когато трябва да извършите едно нещо няколко пъти. Можете да промените част от нея, без да изменяте останалата част и без да преписвате цялата програма отначало. Например:

```
NEW
200 P = PDL (0)
210 PRINT P
220 PRINT "ЗАВЪРТЕТЕ ПОТЕНЦИОМЕТЪРА"
230 PRINT "НА УСТРОЙСТВОТО ЗА ИГРА"
```

Накарайте компютъра да изпълни няколко пъти тази програма, като междувременно променяте положението на потенциометъра на устройството за игра. Проверете дали програмата реагира и на двете устройства. Трябва да работи само за едното от тях. Можете да използувате тази възможност и да го маркирате с числото нула.

С нищожна промяна тази програма може да се използува и за поглед върху другото устройство. Предизвикайте изпълнението на програмата както е в момента и след това напишете

```
200 P = PDL (1)
```

Когато въведете оператор с номер, съвпадащ с номера на вече съществуващ оператор, новият оператор застава на мястото на стария. Разгледайте програмата посредством команда LIST, за да видите колко се е променила. Предизвикайте няколко пъти изпълнението й и вижте какво ще стане. Изменяйте състоянието на другото устройство между изпълненията. Реагира ли тази програма и на двете устройства? Бележете с номер едно устройството, на което тя реагира.

Изменянето на една програма по този начин е само пример за редактиране. Друг начин да направите това е да използувате току-що наученото за изтриване на оператори, от които вече нямате нужда. Ако искате да изтриете оператора с номер 230 от предната програма, трябва да напишете

```
230
```

и след това на натиснете RETURN.

За същата цел можете да използувате и функцията DEL. Изтриването на оператор номер 230 от Вашата програма може да стане с командата DEL 230, 230.

```
DEL 200, 230
```

Тази операция изтрива всеки оператор, чийто номер е 200 или по-голям, но е по-малък или равен на 230.
Опитайте тези команди и след това разгледайте програмата, за да видите какво са й причинили. Възможността за изтриване цели блокове ще бъде не малко удобство за Вас, когато започнете да пишете големи програми.

Както вече видяхте има няколко команди, които ви позволяват да работите с цялата програма едновременно. Те са

NEW

която изтрива програмите,

LIST

която разполага текста на програмите върху екрана и

RUN

която предизвиква изпълнението на програмите, като започва от оператора с най-малък номер. Възможно е, също така, да започнете изпълнението от друго място и да разгледате само част от програмата. Тези възможности ще бъдат разяснени по-късно.

ПРОСТА АКРОБАТИКА

От този момент започвате да летите и затова в този пасаж ще се говори за "лупинги" /цикли/.

Най-добрият начин да видите как работи функцията PDL и да разберете програмните "лупинги" е да използувате команда, която не сме използували досега. Тя е много проста. Напишете следните оператори /след като сте написали NEW, за да изтриете всички стари програми, които могат да се случат в паметта/:

```
110 PRINT PDL (0)
120 GOTO 110
```

Оператор номер 110 пише на екрана числото, което представлява текущото състояние на устройството за игра. Операторът с номер 120 праща отново изпълнението на оператор 110. Какво става тогава? Програмата пише текущата стойност на устройството за игра. След това се изпълнява оператор 120, който указва да се изпълни отново оператор 110 и т.н. Завинаги. Това е цикъл. Цикълът е програмна структура, която съдържа команда, връщаща изпълнението на оператор, който вече е бил изпълняван. Сега стартирайте програмата. Повъртете потенциометъра на устройството за игра. В следващия раздел ще Ви кажем как да спрете тази програма. Дотогава

у Вас ще буди възхищение фактът, че Вашият ПРАВЕЦ е изпълнил вече няколко стотин пъти командата PRINT PDL (Ø). Вече възможностите на записаната програма превъзхождат значително това, което можете да правите на ръка. Вашите възможности за работа с компютъра ще нараснат драматично в следващите няколко раздела, поради добрата основа, която вече имате.

НЯКОИ НЕЩА ОБЛЕКЧАВАТ ЖИВОТА

Но първо, вие вероятно се питате как да спрете програмата. Вече сте забелязали как числата се движат бързо нагоре по екрана, когато движите потенциометъра. Това е така, защото те се изписват в долния край на екрана и с появяването на всяко ново число останалите се повдигат с един ред. Вие сте наблюдавали това явление през цялото време, но с много по-малка скорост. За да прекратите изпълнението на програмата просто използувайте

CTRL
Ц
C

Командата CTRL Ц C ви позволява да разберете къде точно е спряно изпълнението на програмата, като изписва на екрана

BREAK IN 110

или изобщо номера на оператора, който се е изпълнявал, когато е станало прекъсването. /Опитайте/. Между другото, това е изключение от правилото да се натиска RETURN след всяка команда. Натискането на RETURN обикновено не е нужно, когато спирате изпълнението на дадена програма с CTRL Ц C

Можете също да използувате клавиша RESET с цел да спрете изпълняваща се програма, но тогава няма да получите съобщение за мястото на спирането.

Когато спрете програмата с CTRL Ц C или RESET, можете да продължите изпълнението й като напишете командата

CONT

Опитайте да направите това, а след това опитайте следната програма:

```
NEW
100 X = PDL (0)
110 PRINT "УСТРОЙСТВО НОМЕР ЕДНО Е В ПОЛОЖЕНИЕ"
120 PRINT X
130 Y = PDL (1)
140 PRINT "УСТРОЙСТВО НОМЕР ДВЕ Е В ПОЛОЖЕНИЕ"
150 PRINT Y
```

По-горе казахме, че когато напишете RUN програмата започва да се изпълнява от оператора с най-малък номер. Това е вярно. Ако искате обаче да започнете изпълнението от някой друг оператор, например оператор 130, вие просто трябва да напишете

```
RUN 130
```

Можете също да указвате номера на оператори в командата LIST. Ако напишете

```
LIST 130
```

ПРАВЕЦ ще ви покаже оператор 130 от Вашата програма /ако такъв има, разбира се/. Ако напишете

```
LIST 110, 130
```

ПРАВЕЦ ще Ви покаже всички оператори от Вашата програма, започвайки от 110 и завършвайки със 130. Командата RUN не притежава тази особеност.

```
ПОДВИЖНИЯТ КУРСОР
ЧИСТИ ДВИЖЕНИЯ НА КУРСОРА
```

Когато бъдат натиснати, клавишите със стрелки движат курсора. Едновременно с това те изтриват или възстановяват символи, както вече научихте това в ГЛАВА II. Има възможност курсорът да бъде преместван, без това да оказва влияние на каквото и да било /с изключение на положението на курсора/. Можете да правите това, като използвате чистите движения на курсора.

За извършването на чистите движения на курсора се използуват пет клавиша. Те са: ESC, I, J, K, и M. Ето как се използуват.

Първо прехвърляте компютъра в редакторски режим, чрез натискане на клавиша ESC, след това използувате I, за да движите курсора нагоре, J за да движите курсора наляво, K за да го движите надясно и M за да го движите надолу. За да получите непрекъснато движение на курсора, натиснете някой от клавишите за управление /I, J, K или M/ и при това натискайте клавиша RPT. Ако курсорът стигне горния край на екрана, той ще спре. Ако достигне долния край на екрана, той ще спре и цялото съдържание на екрана ще започне последователно да се придвижва нагоре. Ако курсорът достигне десния край на екрана, той изчезва и се появява отново в началото на следващия ред. Правете известно време експерименти с тези пет клавиша.

Когато чистите движения на курсорът Ви омръзнат, достатъчно е да натиснете кой да е клавиш и ще се върнете в нормалния режим за писане.

Чистите движения на курсора, предизвикани от клавишите J и K изглеждат върху екрана еднакви с движенията, предизвикани от клавишите със стрелки, но ефектът е различен. Лесно ще се убедите в това, като разгледате резултата с команда LIST. Чистите движения на курсора не оказват никакво въздействие на текста, върху който преминават, докато движенията на курсора посредством клавишите със стрелки изтриват или възстановяват символите, през които минават.

Тези чисти движения на курсора имат сериозно приложение, така че в края на краищата не са толкова чисти. Напишете например

```
130 PPUNT "СМИЛЕТЕ СЕ"
140 PRINT "НАД БЕДНИТЕ ДЕЧИЦА"
```

като, разбира се, натискате RETURN след всеки оператор. По всичко изглежда, че вашият ПРАВЕЦ приема тази програма, но когато предизвикате изпълнението и получавате съобщението,

```
? SYNTAX ERROR IN 130
```

като награда за труда си.

За да направите необходимата поправка, без да преписвате целия погрешен оператор, можете да използувате следния трик. Направете първо LIST на програмата и след това натиснете ESC. Сега натиснете I достатъчен брой пъти, за да доведете курсора върху реда с погрешния оператор. Натиснете J, за да преместите курсора в началото на този ред и след това използувайте дясната стрелка и възстановете всички символи от ред, предшествуващи "U" от "PRUNT". Поправете "U" с "I" и възстановете с дясната стрелка останалите символи от оператора. След това натиснете RETURN и разгледайте програмата, за да видите дали е правилно коригирана.

Компютърът доста често се смилява над бедните програмисти. Когато трябва да препишете част от текст, който се намира някъде върху екрана, чистите движения на курсора, съчетани с изчистващите и възстановяващите му функции могат значително да ускорят преписването. Няколко минути, посветени на опити с тези неща, ще Ви спестят по-късно много излишен труд.

▽

Лявата стрелка действува само върху реда, който пишете в момента. Ако пишете някакъв оператор и извършите чисто движение на курсора преди да сте натиснали RETURN ,лявата стрелка влияе /изтриващо/ върху оператора, който пишете, а не върху символите, по които се движи курсорът. За разлика от нея, дясната стрелка възстановява винаги точно символите, върху които се намира курсорът.

НАКОЛКО ДУМИ ОТНОСНО ИЗУЧАВАНЕТО НА РАЗШИРЕНИЯ БЕЙСИК

Много често възникват въпроси във връзка с разширения БЕЙСИК, на който тази книга не дава пряк отговор. Например дали в командата

```
PRINT "HELLO"
```

трябва да се остави интервал /шпация/ след думата "PRINT". Вместо да Ви дадем наготово отговора, ние Ви препоръчваме да опитате с Вашия ПРАВЕЦ и двата начина. Обикновено един прост експеримент дава отговор на подобни въпроси, а тъй

като до отговора сте достигнали сами, Вие ще го запомните много по-добре, отколкото ако го бяхте прочели.

ПРЕДОТВРАТЕН НЕЩАСТЕН СЛУЧАЙ

В предните части на тази глава се научихте да изтривате даден оператор от програмата, като наберете номера му и след това натиснете RETURN . Това е популярен начин за въвеждане на грешки в програмата. Помислете си какво ще стане, ако поискате да премахнете от Вашата програма оператора с номер 1100, но поради неволна грешка напишете

```
110
```

RETURN. Нашите поздравления, Вие току-що изтрихте оператор 110. Случва се. Представете си също, че искате да въведете някаква промяна в оператор 450, поради което пишете

```
450
```

след което размисляте и решавате, че няма да променяте оператора. Не натискайте RETURN . Или се върнете с лявата стрелка по току-що написания номер, или използувайте специална команда, имаща смисъл "забрави това, което написах":

CTRL
X

Използуването на CTRL X поставя обратна наклонена черта в края на написаното от Вас и можете да смятате, че не сте писали нищо.

```
]LIST
200 PRINT "HELLO"
210 PRINT "345+765"
220 PRINT "67/4"
]210 PRIN "БЕЗ ПОСЛЕДСТВИЯ"\
]LIST
200 PRINT "HELLO"
210 PRINT 345+765
220 PRINT "67/4"
]
```

ИСТИНАТА

ПРАВЕЦ може винаги да каже кое е истина и кое не. Тъй като това превъзхожда възможностите на повечето от нас, необходими са няколко обяснителни думи. Символът > означава "по-голямо от". Твърдението 6 > 2 /което се чете "шест е по-голямо от две" /е, разбира се вярно. ПРАВЕЦ използува числото 1 вместо езиковото обозначение "ИСТИНА".

Ако напишете

```
PRINT 6 > 2
```

компютърът ще отговори с 1. Твърдението 55 > 78 е невярно. ПРАВЕЦ изпълзува числото 0 за означаване на "НЕИСТИНА". Ако напишете

```
PRINT 55 > 78
```

компютърът ще отговори с 0.

Символът < означава "по-малко от" и можете да образувате твърдения, като използувате и него. Ето пълният набор от символи, използувани при образуването на логически твърдения:

>	по-голямо от
<	по-малко от
=	равно на
>=	по-малко или равно
<=	по-голямо или равно
<>	различно от

За да напишете на клавиатурата на ПРАВЕЦ символите "по-голямо или равно" и "по-малко или равно", трябва първо да напишете < или > и след това да напишете =. Символът "различно от" се получава, като първо напишете < и след това >

Първо помислете, а след това проверете на ПРАВЕЦ кои от следните твърдения са верни и кои не.

```
5 <> 5
6  > 2
8  > 8
9534 = 4359
5  < 8
45 >= 4
```

```
-8 < -7
-2 >= -5
9 <>-9
```

Логическите твърдения могат да съдържат променливи и изрази, не само числа. Например

```
PRINT (45 * 6) <> 645 + 6)
```

ще напише на екрана стойност 1, тъй като 270 не е равно на 51 /спомнете си, че с 1 се обозначава вярно твърдение/.

Вече видахте, че ПРАВЕЦ може да различи истината от лъжата при прости твърдения, отнасящи се до числа. Твърдения от вида на A > B обаче могат да бъдат верни или не в зависимост от стойностите на променливите A и B. Ако

```
A = 5
```

и

```
B = 9
```

твърдението A>B не е вярно. Но ако

```
A = -8
```

и

```
B = -15
```

твърдението

```
A > B
```

е вярно.

Логическите твърдения имат числена стойност нула или едно. Те могат да бъдат използувани в аритметични изрази вместо нули или единици. Например

```
PRINT 3 + (4 > 2)
```

ще даде стойност 4. Операторът

```
T = 4<>3
```

зарежда T със стойност 1, тъй като 4 не е равно на три и по такъв начин логическото твърдение 4 <> 3 има стойност 1. Командата

```
H = 67 = 19
```

е доста смущаваща на пръв поглед, но не е трудно да бъде раз-

брана. Тъй като 67 не е равно на 19, твърдението 67 = 19 е невярно и има стойност нула. Тази стойност се зарежда в променливата Н.

Както вече видахме, ПРАВЕЦ използува 1 за "истина" и Ø за "неистина". Ако едно нещо не е вярно, то е лъжливо. Ако едно нещо не е лъжливо, то е вярно. Това не винаги е така в живота, но винаги е изпълнено при компютрите. Опитайте с ПРАВЕЦ следната команда:

```
PRINT NOT 1
```

и след това опитайте

```
PRINT NOT Ø
```

Думата "NOT ", поставена пред 1 и Ø е съответната английска граматична форма за "НЕ". Компютърът се съгласява: неистина е лъжа и нелъжа е истина. Разбира се, можете да използувате изрази от нули и единици.

Например

```
PRINT NOT (45 > 3)
```

Изречението

ТРИЪГЪЛНИЦИТЕ ИМАТ ТРИ СТРАНИ.

е вярно. Също така вярно е и изречението

ТАЗИ КНИГА Е НА БЪЛГАРСКИ.

Разгледайте изречението

ТРИЪГЪЛНИЦИТЕ ИМАТ ТРИ СТРАНИ И ТАЗИ КНИГА Е НА БЪЛГАРСКИ.

Вярно или невярно е това твърдение? Вярно е. Разгледайте изречението

ТРИЪГЪЛНИЦИТЕ ИМАТ ОСЕМ СТРАНИ И ТАЗИ КНИГА Е НА БЪЛГАРСКИ.

Това твърдение като цяло е невярно. Накрая разгледайте твърдението

ТРИЪГЪЛНИЦИТЕ ИМАТ ОСЕМ СТРАНИ И ТАЗИ КНИГА Е НА ЯПОНСКИ.

Това твърдение също не е вярно. Изобщо, когато комбинирате две изречения или твърдения, като ги свързвате с думата И, Вие откривате, че

а. Новото изречение е вярно, ако двете начални твърдения са верни
б. Новото изречение е погрешно, ако поне едното от началните изречения е погрешно

ПРАВЕЦ умее да определя кога едно твърдение, съдържащо думата AND/AND означава И на български/ е вярно и кога - погрешно. Можете да изпробвате Вашия компютър със следните команди; опитвайте се да предсказвате отговора:

```
PRINT 1 AND 1
PRINT 1 AND Ø
PRINT Ø/AND Ø
PRINT (3 > 2) AND Ø
PRINT (NOT Ø) AND (4 = 5)
```

Вижте дали следното изречение е вярно:
ТРИЪГЪЛНИКЪТ ИМА ТРИ СТРАНИ ИЛИ ТАЗИ КНИГА Е НА ЛАТИНСКИ
Изречението е вярно. Триъгълникът има три страни, дори ако тази книга не е на латински, така че изречението като цяло е вярно.
Изобщо, когато комбинирате две изречения, като ги свързвате с думата ИЛИ, Вие откривате, че

а. Новото изречение е вярно, ако поне едното от двете начални изречения е вярно.
б. Новото изречение е погрешно, ако и двете начални изречения са погрешни.

ПРАВЕЦ може също да определи кога едно логическо твърдение, съдържащо думата OR/OR означава ИЛИ на български/ е вярно и кога - погрешно. Опитайте на компютъра всяко от следните логически твърдения, като предварително съобразявате какъв трябва да бъде отговорът.

```
PRINT 1 OR 1
PRINT 1 OR Ø
PRINT Ø OR 1
PRINT Ø OR Ø
PRINT (4<>5) OR (4 = 5)
PRINT 1 OR (Ø AND 1)
PRINT ((3>4) OR (54<337)) AND (NOT Ø)
```

Командите AND, OR и NOT ще станат много полезни в следващия пасаж.

Вие вече сте открили, че в командата

```
PRINT 1 OR Ø
```

компютърът счита 1 за "ИСТИНА" и Ø за "НЕИСТИНА". Сега опитайте следното:

```
PRINT 23 OR Ø
```

и това:

```
PRINT -247 AND 32767,61
```

Когато става дума за логически твърдения, ПРАВЕЦ разглежда не само 1, а и всяко ненулево число като "ИСТИНА". Когато обаче компютърът определя стойността на твърдението, тя винаги е Ø или 1. Следващите няколко реда ви дават приоритета при логическите операции. Настойчиво ви съветваме да внасяте яснота в логическите си твърдения, като използувате скоби.

ПРИОРИТЕТ /РЕД НА ИЗПЪЛНЕНИЕ/
НА ОПЕРАЦИИТЕ, ИЗПОЛЗУВАНИ ДОСЕГА В ТЕКСТА

1. ( )
2. NOT - /за отрицателни стойности/
3. ^
4. * /
5. + -
6. > < = >= <>
7. OR
8.

КОМАНДАТА IF

Представете си, че трябва да напишете на екрана целите числа от 1 до 10, като разполагате по едно в ред. Един очевиден начин да направите това е

```
NEW
210  PRINT 1
220  PRINT 2
230  PRINT 3
```

и така нататък. Но така ще трябва да напишете 10 оператора, а ако трябваше да напишете числата от 1 до 200 по този начин, щеше да Ви се наложи да напишете 200 оператора. Като използувате вече наученото, можете да напишете числата от 1 нагоре, като построите цикъл:

```
200  N = 1
210  PRIN N
220  N = N + 1
230  GOTO 210
```

Съществува възможност да се определи до кога да се върти цикълът. Това, което ви трябва, е оператор, който изпълнява командата GOTO когато е например по-малко от 11 и не я изпълнява, когато N е по-голямо или равно на 11. Командата IF задоволява напълно Вашите желания. Ако определено условие се изпълнява, компютърът ще пропусне операцията GOTO и ще премине към изпълнението на оператора на следващия ред. Ако няма следващ ред, програмата ще спре.

Ето една програма, която брои от 1 до 10 и спира:

```
200  N = 1
210  PRINT N
220  N = N + 1
230  IF N < 10 THEN GOTO 210
```

В общия случай командата IF действува по следния начин:

IF логически израз THEN - каква да е команда

/Думата IF на български означава АКО/.

Първо се дава стойност /0 или 1/ на логическия израз. Когато той има стойност 0, цялата останала част на оператора се пренебрегва. Ако логическият израз има стойност 1, тогава се изпълнява командата след думата THEN. /THEN на български означава ТОГАВА или В ТАКЪВ СЛУЧАЙ/.

Операторът IF е много мощен и ще се появява неизбежно в почти всяка програма, която пишете. Опитайте за развлечение следната програма:

```
NEW
400  GR
410  ROW = 1
420  COLOR = ROW
430  HLIN 0,39 AT ROW
440  ROW = ROW + 1
450  IF ROW < 16 THEN GOTO 420
```

ЗАПИСВАНЕ НА ПРОГРАМИ ВЪРХУ ДИСК

/Можете да пропуснете тази част, ако не използвате дисково устройство/

На този етап от работата може да поискате да запишете върху диск някои от програмите, които сте използували. Просто напишете цялата програма /след като сте написали NEW / след това напишете

SAVE

последвано от някакво име, с което ще наречете тази програма и след това, разбира се, RETURN . Ако например искате да запишете горната програма и да я наречете ИВИЦИ, трябва да напишете

SAVE ИВИЦИ

и програмата ще бъде записана на диск под името ИВИЦИ. След като програмата вече е записана, напишете

CATALOG

последвано от RETURN, за да видите името на Вашата програма, написано заедно с другите имена на програми от диска. Сега вече можете да предизвикате изпълнението на програмата от диска, наречена ИВИЦИ всеки път, когато пожелаете, просто като напишете

RUN ИВИЦИ

Опитайте да запишете програмата, която най-много Ви харесва. Ако случайно направите грешка при писането на командата /при което по всяка вероятност ще получите съобщението ? SYNTAX ERROR/, можете да поправите грешката си, като напишете командата отново.

Понякога е желателно една програма да бъде заредена в паметта на компютъра, без да се стига до изпълнение. Например, ако искате да промените някоя програма, преди да предизвикате изпълнението й. В такива случаи е полезна командата LOAD За да я използувате, просто напишете

LOAD

последвано от името на програмата,която искате да заредите в паметта. Ако например искате да заредите програмата, наречена ИВИЦИ, просто напишете

LOAD ИВИЦИ

Ако промените програмата и след това поискате тя да се изпълни, без да записвате новата версия върху диска, не забравяйте да напишете само

RUN

Ако забравите и напишете

RUN ИВИЦИ

старата версия на програмата ще бъде презаредена и ще изтрие новата версия от паметта.

Можете да използувате командите LOAD и SAVE с цел да прехвърляте програми от един диск на друг, като зареждате програмата от един диск и я записвате върху друг. Направете няколко упражнения с употребата на командите LOAD и SAVE.

ЗАПИСВАНЕ НА ПРОГРАМИ С КАСЕТОФОН

/Можете да пропуснете тази част, ако не използувате касетофон/

За да запишете върху касета една програма, която ще използувате по-късно, първо поставете празна касета в касетофона и пренавийте лентата до началото, където ще бъде лесно да я намерите. Включете касетофона на запис и наберете върху клавиатурата на ПРАВЕЦ

SAVE

Когато натиснете RETURN , мигащият курсор ще изчезне. След около 10 или 15 секунди компютърът ще подаде звуков сигнал, с което ще покаже, че записването е започнало. Следва нов звуков сигнал в края на процеса на записване и курсорът се появява отново. Спрете касетофона, превийте лентата до началото и можете да продължите да програмирате. Записът не е повлиял по никакъв начин върху програмата в паметта.

ДРУГИ ГРАФИЧНИ ПРОГРАМИ

По-рано поставихте четири цвята в ъглите на екрана. Сега напишете програмата:

```
NEW
190  GR
200  COLOR = 9
210  PLOT 0,0
220  PLOT 0,39
230  PLOT 39,39
240  PLOT 39,0
```

Разгледайте програмата с команда LIST , за да се убедите, че сте я написали правилно и след това я стартирайте. Става бързо, нали? За да промените цветовете, променете оператор 200 и отново стартирайте програмата.
Опитайте се да подадете команда LIST. Забелязвате, че текстът на програмата преминава през тясната текстова ивица в долния край на екрана. Това ще продължава да се случва докато не напишете

```
TEXT
```

и с това излезете от графичния режим.

Следната програма запълва целия екран с еднакъв цвят.

```
NEW
200  GR
210  COLOR = 9
220  COLUMN = 0
230  VLIN 0,39 AT COLUMN.
240  COLUMN = COLUMN + 1
250  IF COLUM<40 THEN GOTO 230
```

Ще обясним оператор по оператор действието на тази програма. Оператор 200 установява графичен режим. Цветът се избира в оператор 210. Програмата започва работа от нулевата ивица на екрана и продължава до 39-та ивица. Оператор 220 осигурява започването от ивица 0. Оператор 230 покрива ивица 0 с установения в оператор 210 цвят. След като това е направено, оператор 240 увеличава номера на ивицата с 1. Оператор 250 проверяве дали новият номер на ивицата е по-малък от 40. Ако това е така, изпълнението се връща на оператор 230, за да запълни тази ивица. Когато обаче стойността на COLUM стане 40 /на екрана на ПРАВЕЦ най-дясната ивица е с номер 39/, изпълнението на програмата не се предава на оператор 230, а, както ние казваме, "пропада надолу" и се прекратява, тъй като в случая това е краят на програмата.

За да избегнете нуждата да пишете RUN всеки път, когато искате да запълните екрана, напишете

```
260  GOTO 210
```

Наблюдавайте какво става. Кога ще спре тази програма? Разгледайте програмата и се убедете, че разбирате добре какво прави, преди да продължите по-нататък с това ръководство.

Когато свършите заниманията си с боядисващата програма, изчистете паметта на компютъра и опитайте следната програма. Тя съдържа една нова и много важна инструкция: инструкцията REM Думата "REM"е първата част от английската дума REMark /забележка/.

Тази команда Ви позволява да поставяте коментари в програмата. Компютърът пренебрегва REM-командите; те са предназначени изключително за употреба от страна на човешките същества. Вижте колко лесно е да се анализира тази програма, в която REM-овете са използувани широко. В REM -операторите могат да бъдат използувани всички символи от клавиатурата.

```
200  REM  УСТАНОВЯВАНЕ НА ГРАФИЧЕН РЕЖИМ
210  GR
220  REM  ИЗБОР НА ЦВЯТ
230  COLOR = 1
240  REM  ТЕКУЩО СЪСТОЯНИЕ НА НУЛЕВО У-ВО ЗА ИГРА
250  X =  PDL(0)
260  REM  ДЕЛЕНИЕ НА 7 С ЦЕЛ МАКСИМУМЪТ НА X
          ДА БЪДЕ 36
```

```
270  X = Y / 7
280  REM  ТЕКУЩО СЪСТОЯНИЕ НА УСТРОЙСТВО НОМЕР 1
290  Y = PDL (1)
300  REM  ОГРАНИЧАВАНЕ НА СТОЙНОСТИТЕ НА Y
310  Y = Y / 7
320  REM  НАНАСЯНЕ НА ТОЧКАТА ВЪРХУ ЕКРАНА
330  PLOT X,Y
340  GOTO 250
```

След командата RUN повъртете устройствата за игра. Деленето на 7 е необходимо, тъй като функцията PDL заема стойности между 0 и 39. Деленето на 7 дава стойности между 0/7 = 0 и 255/7 = 36,4285715. Графичната система автоматично закръгля координатните стойности с недостиг, т.е. взема най-близкото цяло число, което е по-малко от зададеното. С други думи X и Y -координатите биват закръгляни до цели числа между 0 и 36, така че да бъдат в диапазона на екрана. За да използувате цялата ширина на екрана, вместо

```
270 X = X / 7
```

използувайте двата оператора

```
200 IF X > 239 THEN  X=239
275 X = X / 6
```

За да използувате цялата височина на екрана, трябва да направите същото и с Y -координатата.

Оператор 270 сега ограничава стойностите на X до 239. Графичната система на IIPABEЦ закръгля 239/6 от 39,8333333 на 39. Използуването на IF оператор за ограничаване на диапазона на някоя променлива е много разпространено.

ОБРАЗУВАНЕ НА ЦИКЛИ ПОСРЕДСТВОМ FOR / NEXT

Всеки цикъл има начало и край. В програмата

```
NEW
100  NUMBER = 0
110  PRINT NUMBER
120  NUMBER = NUMBER  1
130  IF NUMBER < = 12 THEN GOTO 110
```

оператор 110 е началото, а оператор 130 - краят на цикъла. Програмата изписва на екрана числата от 0 до 12 включително. Числото 12 е граница на цикъла. Друг начин да се организира цикъл е използуването на оператор, който още не сме разглеждали: операторът FOR Можем да използуваме този оператор и да напишем по друг начин предната програма:

```
200  FOR NUMBER = 0 TO 12
210  PRINT NUMBER
220  NEXT NUMBER
```

За да предизвикате изпълнението на тази програма напишете

```
RUN 200
```

Ако напишете само RUN, изпълнението ще започне от оператор 100, който в момента е най-ниският номер на оператор от тези, които се намират в паметта.

Новият оператор има номер 200. Неговата дейност започва с това, че зарежда с 0 променливата NUMBER. Точно това извършва и оператор 100. След това се изпълнява оператор 210, който по нищо не се различава от оператор 110. Краят на цикъла е оператор 220. Стойността на променливата NUMBER се увеличава с единица и след това тази стойност се сравнява с границата на цикъла, която е зададена във оператора FOR :12. Ако NUMBER не превишава тази граница, управлението се предава на оператора, който е непосредствено след FOR -оператора. Ако стойността на NUMBERнадминава границата на цикъла, управлението се предава на оператора, който е непосредствено след NEXT В нашия случай такъв оператор няма и програмата прекратява изпълнението си.

Най-очевидното преимущество на FOR/NEXT-метода за организиране на цикли е спестяването на един оператор. Най-важното му преимущество е, че организирането на цикъл по този начин става с по-малко мислене. Ако например искате да нарисувате на екрана поредица от хоризонтални ивици, оцветени последователно с всичките 15 достъпни цвята, можете да направите това по следния начин:

```
3000 GR
3010 FOR N = 0 TO 15
3020 COLOR = N
3030 HLIN 0,39 AT N
3040 NEXT N
```

Друго преимущество на метода е по-лесното четене на програми, съдържащи цикли. Всичко, което е необходимо да направите за да откриете края на един FOR / NEXT цикъл е да потърсите такъв NEXT ,който има същата променлива като FOR -оператора. Този процес на търсене обаче се затруднява, когато в програмата има няколко цикъла с една и съща променлива.

Може би трябва да спомена, че познаването на оператора FOR е наложително /най-малкото, за да можем да четем чужди програми/, но използуването му - не. Той не прибавя нови възможности, към тези, които вече имате, а само създава известни улеснения за някои хора.

FOR / NEXT-методът има и недостатъци, някои от които са сериозни и непознаването им може да доведе в някои случаи до значително объркване. Недостатъкът, на който човек най-лесно се натъква е невъзможността за непосредствено влизане в тялото на цикъла /т.е. в областта на оператори, намираща се между FOR и NEXT/. Например опитът да предизвикате изпълнението на програмата

```
4000 GOTO 4020
4010 FOR I = 1 TO 45
4020 PRINT I
4030 NEXT I
```

ще доведе до съобщението за грешка

```
? NEXT WITHOUT FOR ERROR IN 4030
```

Горната програма е, разбира се, безсмислена, но в някои случаи се налага да бъде използувана само определена част от тялото на даден цикъл и тогава подобни операции имат смисъл.

Много често се налага принудително излизане от тялото на цикъла /т.е. преминаването в друга област на програмата, преди цикълът да се е извъртял необходимия брой пъти/. Например да предположим, че трябва да открием кое е първото цяло число N , за което поредицата умножения

$$1 \times 2 \times 3 \quad \text{и т.н. } N$$

дава резултат по-голям от 5000000000 или, в "научен" запис 5 E + 9. Ние предполагаме /твърде основателно/, че това число е по-малко от 100 и пишем следната програма:

```
100  I = 1
110  FOR N = 1 TO 100
120  I = I * N
130  IF I > 5 E + 09 THEN 160
140  NEXT N
150  GOTO 170
160  PRINT "N = ";N
170  A = (A + (A + (A + (A + (A + (A + A)))))))))))
```

Проверете внимателно дали сте написали всичко правилно и предизвикайте изпълнението на програмата. Може би противно на Вашите очаквания тя работи и дава забележителния резултат 13. Сега добавете безсмисления за нашата малка задача оператор

```
180 GOTO 100
```

и отново предизвикайте изпълнение и проявете известно търпение, като почакате малко. Отначало всичко върви според Вашите очаквания. Програмата отново и отново решава нашата задача и изписва отговора на екрана. Съдейки от текста на програмата, Вие очаквате този процес да бъде безкраен и точно тук грешите. Развръзката, както в някои пиеси, настъпва внезапно: компютърът подава звуков сигнал, изпълнението на програмата се прекратява и на екрана се появява съобщението за грешка

```
?OUT OF MEMORY ERROR IN 170
```

всичко това едновременно. Оказва се, че компютърът допуска известен брой пъти "нарушаване на етикета", т.е. принудително излизане от FOR / NEXT - цикъл, но през цялото време има в буквалния смисъл на думата "едно на ум". По-точно, компютърът в известен смисъл брои колко пъти в процеса на изпълнение на програмата се е случило принудително напускане на FOR - цикъл, и когато броят на тези напускания стане твърде голям се стига до прекъсване на изпълнението на програмата и издаване на съобщението

```
? OUT OF MEMORY ERROR
```

Този случай може да бъде много неприятен и объркващ, особено в големи програми, в които принудителните напускания на различни цикли зависят от непредвидимия резултат на сложни пресмятания. Може да се случи такива програми да работят нормално

при едни начални стойности на някои променливи и да правят на пръв поглед безпричинно прекъсване при други начални стойности. Съобщението ?OUT OF MEMORY означава в превод на български "недостатъчна памет" и в случая е самонасочващо указание за твърде напреднали програмисти.

По-долу ще посочим лесен начин за механично заместване на FOR / NEXTоператори с IF -оператори.

За да напишете само четните числа от 0 до 12 можете да използувате програмата

```
100  TH = 0
110  PRINT TH
120  TH = TH + 2
130  IF TH<= 12 THEN GOTO 110
```

Тайната е в оператор 120, където към TH се добавя 2. Казваме, че цикълът има стъпка две. За да имате стъпка две при FOR-цикъл, към оператора FOR трябва да добавите указанието STEP 2. Например горната програма ще има вида

```
200  FOR TH = 0 TO 12 STEP 2
210  PRINT TH
220  NEXT TH
```

Не бива да смятате, че стъпката на FOR / NEXT -цикъл може да бъде само 1 или 2. Тя може да бъде всяко число от тези, които ПРАВЕЦ използува, в частност и отрицателно число. Например напишете

```
200  FOR TH = 39 TO 15 STEP -3
```

след това

```
RUN 200
```

Трябва да се поупражнявате с FOR / NEXT -операторите, ако искате да се научите да ги използувате. Не малка част от примерите, давани по-нататък в ръководството, включват между другото FOR / NEXT -оператори.

FOR -циклите могат да бъдат влагани един в друг, но не бива да се пресичат. Ето няколко примера, които илюстрират казаното.

```
NEW
300  GR
310  FOR H = 1 TO 15
320  COLOR = H
330  FOR R = 0 TO 39
340  HLIN 0,39 AT R
350  NEXT R
360  COLOR = H - 1
370  FOR G = 0 TO 39
380  VLIN 0,39 AT C
390  NEXT C
400  NEXT H
```

Тази програма е пример за влагане от второ ниво. Анализирайте нейната работа и предизвикайте изпълнението и преди да преминете към следващия пример. Запомнете, че когато пишете програма, в която има FOR -оператори, всеки FORтрябва да има съответствуващ му NEXT .

ПОГРЕШНА ПРОГРАМА

```
NEW
500  FOR N = 1 TO 20
510  PRINT N
520  FOR J = 30 TO 40
530  PRINT J
540  NEXT N
550  NEXT J
```

Тази програма няма да работи. Нейните цикли се пресичат, което освен че дава съобщение за грешка, е безсмислено. Обикновено създаването на пресичащи се цикли е указание, че нещо в логиката на програмата /и програмиста/ се е объркало. В случай, че сте сигурни, че пресичането на някои цикли е необходимо, образувайте тези цикли, като използувате оператора IF.

МЕХАНИЧЕН НАЧИН ЗА ПРЕОРГАНИЗИРАНЕ НА ЦИКЛИ

Поради описаните по-горе недостатъци на FOR / NEXT - метода за организиране на цикли, в много случаи се налага някои от тях да бъдат организирани по друг начин. Това е особено неприятно, когато програмата е вече написана и циклите, които трябва да бъдат преорганизирани са много. Понякога се случва като допълнително затруднение да не разполагаме със свободни адреси /номера на редове/. Ще посочим един икономичен начин за преорганизиране, който, разбирасе, може да бъде използуван и при първо писане и който не изисква почти никакво мислене и съобразяване.

Можем да разгледаме следния общ пример:

```
100  FOR N = 1 TO 100
110
120 /други оператори/
1120 NEXT N
```

Ако сега напишем

```
100  N = 1
110  IF N < 100 THEN  N = N + 1   GOTO 110
```

нищо няма да се промени в работата на програмата. Не приемайте това на доверие. Проверете сами.

В оператор 120 употребихме средство, което досега не беше споменавано - написването на няколко последователни оператори, отделени с две точки /:/. Такава група оператори може да бъде включена както след думата THEN в оператор IF /което току-що направихме/, така и след номер на ред. Например

```
100 A = 5:B = 6:PRINT A:PRINT B
```

или

```
100 P = PDL(0):PRINT P:GOTO 100
```

Писането на няколко оператора с общ операторен номер има следните предимства:

1. Операторите се изпълняват по-бързо. /Това е предимство само, ако имате нужда от по-голяма скорост на изпълнение/.

2. По-голяма част от Вашата програма се събира наведнаж на екрана.

3. Спестява се известно количество писане.

4. Можете да групирате на едно място няколко оператора, които изпълняват заедно някаква особена работа.

5. Такъв начин на организация изисква по-малко памет. /Това е преимущество само ако не Ви достигне паметта на компютъра и в процеса на писане на програмата получите съобщение ? OUT OF MEMORY ERROR или PROGRAM TOO LARGE/.

Налице са също и няколко недостатъка:

1. Така написана, програмата се чете по-трудно.

2. По трудно се нанасят промени и поправки.

3. Можете да подадете управлението само на първия оператор от групата.

4. Твърде обезкуражаващо е, когато напишете дълга поредица от оператори с общ номер и при изпълнение получите съобщение за грешка, да трябва да преписвате цялата група оператори наново.

ПОСЛЕДЕН ПРИМЕР НА ВЛОЖЕНИ ЦИКЛИ

```
NEW
300  GR
310  H = 0
320  FOR C = 0 TO 35 STEP 5
330  FOR L = 0 TO 30 STEP 10
340  H = H + 1
350  IF H >15 THEN H = 0
360  COLOR = H
370  FOR R = L TO L + 9
380  HLIN C,C + 4 AT R
390  NEXT R
400  NEXT L
410  NEXT C
```

В тази програма има влагане на цикли от трето ниво. Тя рисува юргани. Забележете, че COLOR не може да бъде използувана като променлива на FOR / NEXT-цикъл. COLOR е запазена

дума в разширения БЕЙСИК. Опитайте се да изпълните програмата в текстов ружим, като премахнете оператор 300. Какво става в такъв случай?

С БЛЯСЪК В ОЧИТЕ

Ако старият начин на писане с бели букви върху черен екран вече Ви е омръзнал, този раздел ще Ви достави особено удоволствие. Напишете

```
INVERSE
```

и погледнете новата средна скоба, която се е появила в левия край на екрана. Тя трябва да бъде черна на бял фон. Сега напишете проста програма като тази:

```
NEW
100 PRINT "И ЧЕРНО-БЯЛОТО МОЖЕ ДА БЪДЕ ИНТЕРЕСНО"
```

и я пуснете да се изпълни. Не е ли по-очарователно? сега напишете

```
FLASH
```

и отново пуснете програмата. Сега вече наистина гледате екрана с блясък в очите.

Забележете, че INVERSE и FLASH оказват влияние само върху това, което компютърът изписва върху екрана. Символите, които се появяват върху екрана, когато Вие ги пишете, не се променят. Тези команди могат да бъдат използувани в режим както на непосредствено, така и на отложено изпълнение. Направете няколко опита с тях. След като известно време сте използували командите INVERSE и FLASH, възможно е да решите, че в края на краищата белите букви на черен фон не са чак толкова досадни. Ако наистина решите да се върнете в доброто старо време, напишете

```
NORMAL
```

и всичко ще бъде наред.

ХУДОЖЕСТВЕНО ОФОРМЛЕНИЕ

Просто за експеримент напишете следната малка програма и проверете какво прави при изпълнение.

```
NEW
100 PRINT "ПРАВЕЦ"
110 GOTO 100
```

Спрете изпълнението с CTRL C. След това променете оператор 100 само с един символ:

```
100 PRINT  "ПРАВЕЦ",
```

и отново пуснете програмата. Както виждате, сега тя печата думата в колони. Сега заменете запетаята /,/ с точка и запетая /;/

```
100 PRINT  "ПРАВЕЦ";
```

и пуснете програмата отново. Този път печатът е компактен. Това означава, че между думите, които сте наредили на компютъра да напише, няма празнини. Компютърът пише ПРАВЕЦ след ПРАВЕЦ и екранът бързо се запълва.

Променете програмата, като добавите оператора

```
90 V = 99
```

и променете оператор 100 на

```
100 PRINT V
```

Пуснете тази програма за изпълнение. Сега променете оператор 100 на

```
100 PRINT V,
```

и отново я пуснете за изпълнение. Сега променете оператор 100 на

```
100 PRINT V;
```

и се убедете, че запетаята, и точката и запетая могат също да бъдат използувани и за числови стойности. Възможността да разполагаме числа едно след друго, без да оставяме празни места между тях, е понякога много полезна. Запетаи, и

точки и запетаи могат да бъдат използувани и вътре в оператор PRINT. Изчистете старата програма с NEW и напишете

```
100 ST = 2
110 BA = 3
120 PRINT ST.BA
```

Можете да организирате по-ясен печат, като включите съобщение в оператора PRINT. Например променете оператор 120 на

```
120 PRINT "УДАРИТЕ И ТОЧКИТЕ СА ";ST,BA"
```

Предполагаме, че не възразявате срещу оставената празна позиция след думата СА. В противен случай първото число ще бъде "залепено" за надписа. Ако пък мислите, че големият промеждутък между двете числа не изглежда добре, можете да използувате оператора

```
120 PRINT "УДАРИТЕ И ТОЧКИТЕ СА"; ST;"    ";BA
```

При този вариант Вие организирате оставянето на празно място между числата. Може би най-добрият начин да направите всичко /правите ли всичко това на компютъра?/ е

```
120       "УДАРИ ";ST;"    ТОЧКИ ";BA
```

Този начин кара екрана да заприлича на спортно табло.

Да предположим, че искате да напишете думата ТУК, като започнете от десета колона на екрана /между другото, екранът е 40 колони широк/; можете да използувате оператора

```
120 PRINT  "ТУК"
```

/Трябва да приемете на доверие, че сме оставили девет празни позиции преди думата"тук/. Можете също да използувате командата TAB. Както и при пишещите машини, на ПРАВЕЦ е възможно да се въведе табулация. Операторът

```
120 PRINT TAB(10)"ТУК"
```

има същия ефект, както и поставянето на девет празни позиции между кавичкитеОпитайте, ще ви хареса.

Ако използувате TAB в цикъл, можете да постигнете някои интересни визуални ефекти. Например

```
NEW
200   FOR N = 1 TO 24
210   PRINT TAB (N)"X"
220   NEXT N
```

В текстов режим екранът разполага с 24 /а не с 40/ реда. Поради тази причина границата на цикъла в горната програма е 24.TAB не може да бъде използуван за придвижване назад /наляво/ по реда. Компютърът изпълнява само указанията на TAB за придвижване по посока на писането /надясно/. За да пишете върху предварително избран ред, трябва да използуване командата VTAB, последвана от номера на реда. Най-горният ред има номер 1, а най-долният - номер 24. За разлика от ТАБ, командата VTAB не се пише вътре в PRINT -оператора.

За хоризонтална табулация можете да използувате и командата HTAB, вместо TAB. Разликата между тях е, че HTAB не се пише вътре в PRINT -оператора и може да установява позицията за писане на произволно място върху реда, без да се интересува какви са били предните указания в това отношение. Най-лявата позиция върху един ред има номер 1, а най-дясната - номер 40. Ето една къса програма, която демонстрира действието и употребата на VTAB и HTAB:

```
NEW
590 HOME
600 FOR X = 1 TO 24
620 FOR Y = 1 TO X
630 HTAB X
640 VTAB Y
650 PRINT "ПРАВЕЦ"
660 NEXT X : NEXT Y
670 GOTO 600
```

Преди да предизвикате изпълнението на тази програма, опитайте /не е лесно/ да предвидите нейните действия. Те са изненадващи и красиви.

TAB работи и при директна команда от клавиатурата, а VTAB и HTAB действуват като координатите в PLOT ,но има и някои разлики. Четиридесетте колони или "вертикални ивици" при TAB и HTAB са номерирани от 1 до 40, докато при PLOT номерацията е от 0 до 39, което е по-удобно при програмиране на

графични изображения. Тъй като символите са по-високи от "тухлите", с които строим графичните изображения, върху екрана се събират само 24 реда от символи. Поради това VTAB е ограничено от 1 до 24. Една нула или изобщо число, което е твърде голямо или твърде малко за TAB, VTAB или HTAB ще предизвика съобщение за грешка

```
? ILLEGAL QUANTITY ERROR
```

Най-голямата стойност за VTAB е 24, но за ТАБ и HTAB тя е 255. TAB и HTAB табулират по дължината на реда, след това "минават зад екрана" и продължават на следващия ред. За да видите това в действие напишете

```
NEW
310   FOR K = 1 TO 255
320   PRINT TAB(K) K
330   NEXT K
```

След това заменете операторите 320 и 330 с

```
320   HTAB K
330   PRINT K
```

и добавете

```
340   NEXT K
```

Какво става с програмата, когато заместите HTAB с VTAB?

ГЛАВА IV

ГРАФИЧНИ ИЗОБРАЖЕНИЯ

Разговор с програмата
Рикошет: програма с много скокове
Компютърът издава звуци:
Озвучаване на отскачащата топка
Случайни числа
Имитиране на двойка зарове
Подпрограми
Проследяване
Подобрена програма за рисуване на коне
Графики с висока разрешаваща способност

ГЛАВА IV

ГРАФИЧНИ ИЗОБРАЖЕНИЯ

РАЗГОВОР С ПРОГРАМАТА

Пред Вас е една програма, която кара едно цветно петно да се движи по екрана и да отскача от левия и десния му край.

```
400   REM       ЦВЯТ НА ТОПКАТА
420   BA = 9
440   REM       ПРЕМИНАВАНЕ В ГРАФИЧЕН РЕЖИМ
460   GR
480   REM       НАЧАЛНО ПОЛОЖЕНИЕ
500   XO = 20
520   REM       ПРИДВИЖВАНЕ НА ТОПКАТА
540   XM = 1
560   REM       НОВА X-КООРДИНАТА
580   XN = XO + XM
600   REM       ТОПКАТА ВЪРХУ ЕКРАНА ЛИ Е?
620   IF(XN>=0) AND (XN < 40) THEN GOTO 720
640   REM       СМЯНА НА ПОСОКАТА НА ДВИЖЕНИЕ
660   XM = -1 * XM
680   GOTO 580
700   REM       РИСУВАНЕ НА НОВАТА ТОПКА
720   COLOR = BA
740   PLOT XN,20
760   REM       ИЗТРИВАНЕ НА СТАРАТА ТОПКА
780   COLOR = 0
800   PLOT XO,20
820   REM       ЗАПОМНЯНЕ НА НОВОТО ПОЛОЖЕНИЕ
840   XO = XN
860   REM       НОВО ПРИДВИЖВАНЕ
880   GOTO 580
```

Този вид програми са в основата на много телевизионни игри. Заслужава си да поекспериментирате известно време с тази програма, като я променяте тук-там, за да видите какво да се получи от нея. Би било добра идея да я запишете върху диск или касета, за да не се налага да я преписвате цялата, в случай, че направите фатална промяна.

Когато използувате команда LIST за да разгледате тази програма, тя не се събира цялата на екрана, а редовете се движат толкова бързо, че не успявате да ги прочетете. Един начин да видите цялата програма е да я разглеждате на части. Например

```
LIST 400, 680
```

и на екрана ще излязат само операторите с номера между 400 и 680. След това можете да разгледате и останалата част на програмата. Можете също да използувате

CTRL

S

за да прекъснете движението на текста на програмата по екрана. Подайте команда LIST и след това бързо натиснете

CTRL

S

преди началото на текста на програмата да е изчезнало в горния край на екрана. За да го накарате да продължи да се движи, натиснете отново.

Можете да спрете движението на текста на програмата и с CTRL U Това обаче отменя командата LIST CTRL S само я привежда в режим на изчакване / и Вие не можете да продължите непосредствено да разглеждате текста на програмата от там, откъдето сте спрели.

Вие вече имате възможност да разберете действието на горната програма, но може би имате приятели, които не могат. Представете си, че искате вашият приятел сам да избере цвета на топката. Бихте могли да му обясните как да промени оператор 420, но ще трябва също да обясните и възможните съобщения за грешка и какво да прави, ако... изобщо ще трябват много обяснения. По-добре ще бъде да направите така, че

Вашият приятел да контактува непосредствено с програмата. Можете да постигнете това, като използувате команда INPUT. Променете оператор 420 на

```
420 INPUT BA
```

Когато този оператор се изпълнява, на екрана ще се появи една въпросителна /?/, последвана от мигащия курсор. ПРАВЕЦ ще чака, докато някой не напише някакво число и след това натисне RETURN . Написаното число ще стане стойността на променливата BA и програмата ще продължи изпълнението си. Не би било лошо да накарате компютъра да каже на Вашия приятел какво се очаква от него. Това може да стане посредством PRINT -оператори, като например следните

```
280  REM      ЗАДАВАНЕ НА ТЕКСТОВ РЕЖИМ
300  TEXT
320  REM      "ЗА ДА ИЗБЕРЕТЕ ЦВЯТ НА ТОПКАТА"
340  PRINT    "НАПИШЕТЕ ЕДНО ЧИСЛО МЕЖДУ 1 И 15"
360  PRINT    "СЛЕД ВЪПРОСИТЕЛНАТА"
380  PRINT    "СЛЕД ТОВА НАТИСНЕТЕ RETURN"
```

Можете обаче да включите съобщение или надпис и в оператора

```
420 INPUT "КАКЪВ ЦВЯТ ДА БЪДЕ ТОПКАТА/1-15/?";BA
```

Забележете, че в оператор INPUT съобщението трябва да бъде в кавички и трябва да има точка и запетая между него и името на променливата. Когато оператор INPUT съдържа съобщение, той не поставя при изпълнение въпросителен знак /освен когато въпросителният знак се съдържа в съобщението/.

Приятелите Ви могат да използуват клавишите със стрелки за поправка на грешки при написването на числото, но ако направят грешка и след това натиснат RETURN , ще получат съобщение за грешка. Ако някой от символите не е цифра, на екрана ще се появи

```
?REENTER
?
```

Ако бъде подадено прекалено малко или прекалено голямо число, то или програмата ще придвижи топката до десния край на екрана и ще спре, или на екрана ще се появи съобщението за грешка

```
?ILLEGAL QUANTITY ERROR IN 720
```

и програмата отново ще спре. В повечето случаи потребителят няма да знае как отново да пусне програмата в действие - и не е длъжен. Следователно трябва да накарате програмата да проверява дали данните, подавани от потребителя са правилни. Следващите няколко оператора правят това:

```
424   REM  ВА МЕЖДУ Ø И 15 ЛИ Е?
428   IF (BA>Ø) AND (BA<16) THEN GOTO 46Ø
432   PRINT "ТОВА НЕ БЕШЕ МЕЖДУ Ø И 15"
436   GOTO 42Ø
```

Започвате ли вече да разбирате защо Ви съветвахме да оставяте толкова много място между операторите?

Полезен навик е да правите програмите си колкото е възможно по-устойчиви на грешки от страна на потребителя. Вече сте напреднали дотам, че пишете съобщения за грешка, предназначени за другиго. Може би е нормално за програмист като Вас да чете жаргон от сорта на ?SYNTAX ERROR, но определено не е в реда на нещата да карате невинния потребител да се занимава с такива глупости.

Всеки път, когато използвате оператор INPUT , Вашата програма трябва да проверява дали това, което потребителят подава се намира в определени граници, така че програмата да не "гръмне" и да работи нормално. Работата с неопитния потребител /трябва да приемете, че потребителите не са програмисти/ е сама по себе си изкуство. Трябва винаги да използвате ясен и точен език при съобщенията и внимателно и предвидливо да проверявате данните, идващи от потребителя.

Можете, между другото, да вкарвате в компютъра няколко числови стойности с един INPUT Операторът

```
3ØØØ  INPUT X,Y,Z
```

ще разположи както винаги на керана въпросителна и след това ще чака да бъдат написани три числа. Първото число ще бъде заредено в променливата X, второто - в променливата У и третото - в променливата Z . Трите числа трябва да бъдат разделени едно от друго с натискания на RETURN или запетаи.

Запишете за всеки случай Вашата най-добра версия на програмата със скачащата топка. Ако още не сте направили това, опитайте се да добавите към нея вертикално движение. По-долу е дадено решение на този проблем, но е желателно да се опи-

тате да се справите сами преди да погледнете.

Когато тази програма започне да работи така, както вие искате, най-добре е да я запишете на диск или касета. По-късно тя ще ни потрябва отново.

РИКОШЕТ

Ето един начин да накарате топката да отскача от четирите страни на екрана. Операторите в черно са тези, които са били променени или добавени към програмата, при която топката отскачаше от две страни на екрана.

```
280   REM ПРЕМИНАВАНЕ НА ТЕКСТОВ РЕЖИМ
300   TEXT
320   PRINT     "ЗА ДА ИЗБЕРЕТЕ ЦВЯТ НА ТОПКАТА"
340   PRINT     "НАПИШЕТЕ СЛЕД ВЪПРОСИТЕЛНАТА"
360   PRINT     "ЕДНО ЧИСЛО МЕЖДУ 1 И 15"
380   PRINT     "И СЛЕД ТОВА НАТИСНЕТЕ RETURN"
400   REM        ЦВЯТ НА ТОПКАТА
420   INPUT     "КАКЪВ ЦВЯТ ИСКАТЕ?";
424   REM       ЗАДАДЕНАТА СТОЙНОСТ МЕЖДУ 1 И 15 ЛИ Е?
428   IF (BA<16) THEN GOTO 460
432   PRINT     "ТОВА НЕ БЕШЕ МЕЖДУ 1 И 15"
436   GOTO  420
440   REM       ПРЕМИНАВАНЕ В ГРАФИЧЕН РЕЖИМ
460   GR
480   REM       НАЧАЛНО ПОЛОЖЕНИЕ
500   XO = 20
510   YO = 38
520   REM       ПРИДВИЖВАНЕ НА ТОПКАТА ПО ХОРИЗОНТАЛА
540   XM = 1
545   REM       ПРИДВИЖВАНЕ НА ТОПКАТА ПО ВЕРТИКАЛА
550   YM = 1
560   REM       НОВА X-КООРДИНАТА
580   XN = XO    XM
600   REM       ТОПКАТА ВЪРХУ ЕКРАНА ЛИ Е?
620   IF (XN>= 0) AND (XN < 40) THEN GOTO 686
640   REM       СМЯНА НА ПОСОКАТА НА ДВИЖЕНИЕ ПО ХОРИЗОНТАЛА
660   XM = -1   XM
680   GOTO 580
```

КОМПЮТЪРЪТ ИЗДАВА ЗВУЦИ

Можете да накарате Вашия ПРАВЕЦ да издава звуци, като го ударите с нещо, драснете по него с нокът или го пуснете на пода, но в това ръководство става дума за програмиране на звуци. Затова отидете някъде, където е по-тихо и започнете да работите по този раздел.

За създаването на каквато и да била звучаща програма, ще Ви бъде необходима следната магическа формула:

```
150   SO = PEEK(-16336)
```

Не е лесно да се обясни тази формула. Числото -16336 има връзка с "адреса в паметта" на високоговорителя на ПРАВЕЦ и е вградено в елекрониката на компютъра. Човек просто трябва да потърси това число, когато му потрябва.

PEEK връща числовия код, записан на дадено място в компютъра. При повечето такива места PEEK може да предизвика и други явления. В този случай то кара говорителя да издаде късо щракане. Пуснете програмата,като в същото време се вслушвате внимателно в говорителя.

Сега добавете следния оператор:

```
160   GOTO 150
```

и приведете програмата в изпълнение. Това вече се чува без проблеми!

За да накарате компютъра да издава звуци за определен период от време, добавете някои оператори като например

```
140   FOR BEEP = 1 TO 100
160   NEXT BEEP
```

Опитайте.

Всеки тон се генерира чрез бърза поредица от щракания. Всяка програма, която използува последователни повторения на PEEK /-16336/ ще произвежда някакъв шум. Тъй като е досадно да пишем всеки път - 16336, ние ще го заменим със символ, който е по-лесен за писане. Въведете оператора

```
100   S = -16336
```

За да получите хубав, резониращ звук, заменете оператор 150 с

```
150   SD = PEEK(S)-PEEK(S)+PEEK(S)-PEEK(S)+PEEK(S)-PEEK(S)
```

Различният брой PEEK-ове в оператора ще довежда до различно качество на звука. Опитайте няколко варианта. Поставете някои от Вашите варианти в цикъл. Като правило, колкото по-бързо се върти цикълът, толкова по-висок е тонът и обратно.

За да използувате всички тези звуци, качете отново програмата с отскачащата топка в паметта на компютъра. Опитайте се да я накарате да издава подходящ звук всеки път, когато топката се удари в стената.

Едно възможно решение е дадено по-долу, но първо се опитайте да решите въпроса сами. /Подсещане: отскок имаме точно когато ХМ или УМ сменя стойността си/.

ОЗВУЧАВАНЕ НА ОТСКАЧАЩАТА ТОПКА

Ето един начин да направите отскачането достъпно за слуха. Добавете тези оператори към програмата с отскачащата топка:

```
240   REM  ЗАРЕЖДАНЕ НА    С АДРЕСА НА ГОВОРИТЕЛЯ
260   S = -16336
263   REM  ЗВУК ПРИ УДАР
265   FOR B = 1 TO 5
670   BO = PEEK(S)-PEEK(S)+PEEK(S)-PEEK(S)
675   NEXT B
695   REM   ЗВУК ПРИ УДАР
696   FOR B = 1 TO 5
697   BO = PEEK(S)-PEEK(S)+PEEK(S)-PEEK(S)
698   NEXT B
```

Сега опитайте тези звуци. Защо не направите звука различен за всяка стена?

СЛУЧАЙНИ ЧИСЛА

Опитайте следната къса програма

```
NEW
100   PRINT RND(1)
110   GOTO 100
RUN
```

Означението RND в оператор 100 е означението на функцията, която произвежда случайни числа. Спрете изпълнението на програмата с

CTRL
Ц
C

Числата, които тази функция дава, са случайно избрани числа между 0 и 1.

Променете оператор 100 на

```
100   PRINT RND(6)
```

и пуснете програмата. Отново я спрете с CTRL Ц C . Интересно. Случайните числа отново са между нула и едно. Запишете си някъде последното генерирано число, за да не го забравите.

Сега отново променете оператор 100, този път на

```
100   PRINT RND(0)
```

и отново пуснете програмата. Спрете я и сравнете това, което си записахте, с това, което е сега на екрана. RND(0) дава последното случайно число, което е било генерирано.

Работата със случайни числа между нула и едно може да бъде малко тромава. Често целите числа /такива като 3 и 6 или 10/ са по-удобни за употреба. За да получим случайни цели числа между 0 и 9, трябва да добавим още няколко оператора към програмата. Напишете

```
 90   REM       ЗАРЕЖДА X СЪС СЛУЧАЙНО ЧИСЛО
100   X = RND(1)
110   REM       УМНОЖАВА X С 10
120   X = X * 10
130   REM       ОТСИЧА ДРОБНАТА ЧАСТ
140   X = INT(X)
150   PRINT X
160   GOTO 100
```

Оператор 140 въвежда функцията INT. Командата INT(X) дава най-голямото цяло число, което е по-малко или равно на стойността на променливата X. Например, ако стойността X е 3.6754, то стойността на INT (X)е 3. Скобите след INT могат да съдържат какъвто и да било аритметичен израз или числова променлива.

Сега предизвикайте изпълнението на програмата. Това ли е, което очаквахте? За да промените програмата, така че да генерира цели числа между едно и десет, вместо между нула и девет, просто добавете единица към стойността на X, като добавите следния оператор към програмата:

```
145 X = X + 1
```

Опитайте.

Тази програма може да изглежда отначало малко сложна. За да проследите в детайли какво става, можете да добавите множество PRINT -оператори. Изменете програмата по следния начин:

```
 90   REM   ЗАРЕЖДА X СЪС СЛУЧАЙНО ЧИСЛО
100   X = RND(1)   PRINT "X = RND(1)",X
105   PRINT
110   REM УМНОЖАВА X С 10
120   X = X * 10   PRINT "X = 10 * X",X
125   PRINT
130   REM  ОТСИЧА ДРОБНАТА ЧАСТ
140   X = INT(X)   PRINT "X = INT(X)",X
145   PRINT
150   REM    ДОБАВЯ 1 КЪМ СТОЙНОСТТА НА X
160   X = X + 1 : PRINT "X = X +1"
170   PRINT X
175   PRINT
180   FOR PA = 1 TO 2000   NEXT PA
190   GOTO 100
```

Предизвикайте изпълнението на тази програма и вижте какво прави.

Ако искате да станете интересен, можете да съберете цялата тази програма на един ред:

```
100   PRINT INT(10 * RND(1)) + 1 : GOTO 100
```

Наясно ли сте как работи този ред?

ИМИТИРАНЕ НА ДВОЙКА ЗАРОВЕ

Можете да използувате това, което научихте за случайните числа, за да напишете програма, която се преструва, че е двойка зарове.

```
NEW
100   PRINT      "БЯЛ ЗАР"
110   PRINT INT (6 * RND(1)) + 1
120   PRINT      "ЧЕРВЕН ЗАР"
130   PRINT INT (6 * RND(1)) + 1
```

Тази програма генерира случайни цели числа от едно до шест за всеки зар. Можете ли да напишете програма, която използува тези "зарове", за да играе някаква игра? Опитайте.

Опитайте да напишете едноредова програма, която да генерира случайни цели числа между 1 и 50. Между 0 и 25. Измислете свои числа. Не забравяйте да добавите 1, ако не искате да генерирате нули.

Ето един живописен начин да използувате случайните цели числа.

```
NEW
200   GR
210   REM        ИЗБИРАНЕ НА СЛУЧАЕН ЦВЯТ
220   COLOR = INT(16 * RND(1))
230   REM        ИЗБИРАНЕ НА СЛУЧАЙНО МЯСТО
240   X = INT(40 * RND(1))
250   Y = INT(40 * RND(1))
260   REM        НАНАСЯНЕ НА ТОЧКАТА
270   PLOT X,Y
280   REM        ВСИЧКО ОТНОВО
290   GOTO 220
```

Опитайте да използувате RND в други програми. Можете ли да напишете програма, която рисува по екрана черти със случайни цветове?

ПОДПРОГРАМИ

Представете си, че за някоя игра Ви потрябва рисунка, която изглежда като син кон с оранжеви крака и бяла глава. Ето една програма, която рисува такъв кон:

```
NEW
1000  REM       ПРОГРАМА ЗА РИСУВАНЕ НА СИН КОН С
                БЯЛА ГЛАВА И ОРАНЖЕВИ КРАКА

1010  GR
1020  PLOT = 7        REM    СВЕТЛОСИН ЦВЯТ
1030  PLOT 15,15
1040  HLIN 15,17 AT 16
1050  COLOR = 9       REM    ОРАНЖЕВ ЦВЯТ
1060  PLOT 15,17
1070  PLOT 17,17
1080  COLOR = 15      REM    БЯЛ ЦВЯТ
1090  PLOT 14,15
```

В тази програма няма нищо лошо; тя наистина рисува син кон с оранжеви крака и бяло лице. Сега си представете, че трябва да нарисувате кон някъде другаде по екрана. Бихте могли да напишете отново програмата с други стойности на координатите, но това е досадна работа. Трябва да има някакъв начин една и съща програма да рисува кон по екрана, където Вие пожелаете, без да е необходимо всеки път да я преписвате.

Ключовият момент в подобна дейност е наблюдението, че една точка с координаки (A,B)може да бъде преместена надясно като към първата й координата - в случая A - се прибави някакво число. Например точката (4,17) се премества с десет позиции надясно, ако прибавите 10 към първата координата, при което точката става (14,17). По същият начин една точка се премества наляво, ако от първата й координата извадите някаква стойност /или добавите отрицателна стойност/. Няколко прости експеримента ще Ви покажат, че прибавянето и изваждането на различни стойности от втората координата движат точката нагоре-надолу.

Като имате предвид тези факти, можете да преправите програмата така, че да може да "позиционира" кон почти във всяка точка (X,Y) от екрана. Защо "почти" във всяка точка?

Защото, ако изберете централната точка на коня в края на екрана, конят ще излезе от него и това може да доведе до съобщението за грешка ?ILLEGAL QUANTITY ERRORIN1030 /или в някой друг оператор/. Ето една подобрена програма.

```
NEW
1000  REM       ПОСТАВЯНЕ НА КОН НАВСЯКЪДЕ ПО ЕКРАНА
1010  COLOR 7                  REM  СВЕТЛОСИН ЦВЯТ
1020  PLOT X,Y - 1
1030  HLIN X,X + 2 AT Y
1040  COLOR = 9                REM  ОРАНЖЕВ ЦВЯТ
1050  PLOT X,Y + 1
1060  PLOT X +2,Y + 1
1070  COLOR = 15               REM  БЯЛ ЦВЯТ
1080  PLOT X - 1,Y - 1
```

Забелязвате, че командата GR е изпусната. Това е така, защото искаме да използуваме тази програма за поставянето на няколко коня върху екрана. Командата GR би изчиствала екрана преди поставянето на всеки нов кон.

Тази програма не може да бъде изпълнена така, както е в момента. Първо трябва да зададете графичен режим и да изберете X и У. Добър първи опит в използуването на конската програма би бил

```
20    GR
30    REM ЦЕНТЪР НА ПЪРВИЯ КОН
40    X = 12
50    Y = 35
```

Ако накарате компютъра да изпълни тази програма, Вие наистина ще получите кон на желаното място, но програмата свършва дотук. Ние искаме да поставим два коня върху екрана. Какво ще стане, ако напишете

```
60  ИЗПЪЛНИ ПРОГРАМАТА ОТ ОПЕРАТОР 1000 НАТАТЪК
    И СЛЕД ТОВА 70
70   REM  ЦЕНТЪР НА ВТОРИЯ КОН
80    X = 33
90    У = 2
100 ИЗПЪЛНИ ПРОГРАМАТА ОТ ОПЕРАТОР 1000 НАТАТЪК
    И СЛЕД ТОВА КРАЙ
```

Би било чудесно и просто. Само че компютърът не може да чете странните инсрукции в редове 60 и 100. Той обаче може да чете

```
GOSUB 1000
```

в разширения БЕЙСИК. Програма като тази, която започва от оператор 1000 се нарича подпрограма. GOSUB 1000 указва на компютъра да предаде управлението на подпрограмата, започваща с оператор 1000 и да започне изпълнението от този оператор. На компютъра също се указва да върне управлението обратно на оператора, който се намира непосредствено след GOSUB-оператора, когато подпрограмата приключи своята работа. Компютърът разбира, че подпрограмата е приключила работата си, когато в процеса на нейното изпълнение срещне оператор RETURN. За да превърнете своята частична програма за рисуване на кон в пълна подпрограма, добавете оператора

```
1090 RETURN
```

Сега вече нищо не Ви струва да напишете

```
  20   GR
  30   REM       ЦЕНТЪР НА ПЪРВИЯ КОН
  40   X = 12
  50   Y = 35
  60   GOSUB 1000
  70   REM       ЦЕНТЪР НА ВТОРИЯ КОН
  80   X = 33
  90   Y = 2
 100   GOSUB 1000
```

Приведете сега програмата в изпълнение. Получавате съобщение за грешка:

```
?RETURN WITHOUT GOSUB ERROR
```

но, както изглежда, във всяко друго отношение програмата работи чудесно. По същество Вие добавихте нов оператор в ПРАВЕЦ-версията на БЕЙСИК: оператор, който рисува кон. Сега вече можете да използувате оператора,

GOSUB 1000, за да нарисувате кон, в която и да било точка X,Y на екрана.

ПРОСЛЕДЯВАНЕ

Участъкът от програмата между операторите 1000 и 1090 се нарича подпрограма. Участъкът от програмата между операторите 20 и 100 се нарича главна програма. Можете да проследите изпълнението на програмата, като използувате съответното средство за проследяване, включено в разширения БЕЙСИК. То може да Ви покаже защо компютърът дава съобщение за грешка при изпълнението на програмата за рисуване на коня. Добавете този оператор към главната програма:

```
10 TRACE
```

и временно изтрийте оператор 20. Преминете в текстов режим и пуснете програмата да се изпълнява.

Числата, които се появяват на екрана, са номерата на съответните оператори в момента на изпълнението им. По този начин можете да видите как програмата започва изпълнението си от оператор 10, продължава през главната програма до обръщението към подпрограмата, връща се в главната програма и отново минава през подпрограмата. Именно тук възниква проблемът. Може би вече разбирате защо се получава съобщение за грешка

За да се справите с този проблем, прибавете към програмата оператора

```
110 END
```

END означава КРАЙ на български и когато програмата срещне оператор 110, тя ще направи именно това: ще прекрати изпълнението си. Нека програмата се изпълни още веднаж. Край на съобщенията за грешка. Както тощу-що видяхте, TRACE е много удобна команда, когато си имате неприятности с някоя програма. Ако искате да проследите само част от дадена програма, можете да използувате командата NOTRACE, която отменя действието на TRACE. Добавете оператора.

```
65 NOTRACE
```

и изпълнението на програмата ще бъде проследено до изпълнението на оператор 65.

TRACE може също да бъде използувано и като директна команда от клавиатурата. Просто напишете

```
TRACE
RUN
```

и програмата ще бъде проследена.

След като веднаж сте подали командата TRACE, Вашата програма ще бъде проследявана всеки път, когато предизвикате изпълнението и. За да излезете от това състояние, подайте командата NOTRACE, като можете да направите това както програмно, така и направо от клавиатурата.

ПОДОБРЕНА ПРОГРАМА ЗА РИСУВАНЕ НА КОНЕ

Подпрограмите трябва да бъдат писани така, че в тях да не възникват грешки при изпълнение. Един от проблемите при нашата рисуваща подпрограма е, че при някои стойности на X и Y конят излиза извън екрана. Това може да бъде предотвратено от набор оператори като

```
1012   IF X < 1 THEN X = 1
1014   IF X >37 THEN X = 37
1016   IF Y < 1 THEN Y = 1
1018   IF Y >38 THEN Y = 38
```

/Защо ограничаваме максималната стойност на Y до 38, а тази на X до 37?/

При всеки опит конят да бъде поставен извън екрана, той ще бъде поставян до най-близкия му край. Има и други стратегии, като например издаването на съобщение за грешка и последвано от спиране на програмата. Нашият избор обаче има предимството едновременно да не спира програмата и да Ви дава възможност да видите, че става нещо нередно.

Понякога е необходимо да можете да променяте стойностите в дадена подпрограма при различните обръщения към нея. Например вторият играч може да иска да разположи кон с различен цвят. Един от начините да направите това е да напишете още един път цялата подпрограма с различни стойности за цветовете. Нека обаче се опитаме да използуваме променливи вместо числа. Вместо оператор 1010 да указва COLOR = 7, той

би могъл да указва

```
1010 COLOR = BO
```

Аналогично можете да напишете

```
1040   COLOR = FE
1070   COLOR = FA
```

Тогава главната програма може да бъде такава:

```
  20   GR
  30   REM       ЦВЯТ  НА ПЪРВИЯ ИГРАЧ
  40   BO = 7             : REM    СВЕТЛОСИНЬО ТЯЛО
  50   FE = 9             : REM    ОРАНЖЕВИ КРАКА
  60   FA = 15            : REM    БЯЛА ГЛАВА
  70   REM       ЦЕНТЪР НА КОНЯ НА ПЪРВИЯ ИГРАЧ
  80   X = 15
  90   Y = 30
 100   GOSUB 1000
```

и така нататък /не забравяйте да завършите с оператор END преди да пуснете програмата за изпълнение/. Това са много оператори за всеки кон, но все пак са по-малко, отколкото, ако трябваше да пишете всеки път цялата програма наново. За още по-голямо удобство при програмирането, можете да използвате един още по-тънък прийом, като въведете подпрограми, които зареждат цветовете на коня на всеки играч и след това всяка от тях се обръща към подпрограмата, която рисува кон.

```
2000   REM  РИСУВА СИН КОН С ОРАНЖЕВА ГЛАВА И БЯЛА ГЛАВА
1010   BO = 7           REM   СВЕТЛОСИНЬО ТЯЛО
2020   FE = 9         : REM   ОРАНЖЕВИ КРАКА
2030   FA = 15        : REM   БЯЛА ГЛАВА
2040   GOSUB 1000
2050   RETURN

2500   REM  РИСУВА ОРАНЖЕВ КОН С РОЗОВИ КРАКА И ЗЕЛЕНА ГЛАВА
2510   BO = 9         : REM   ОРАНЖЕВО ТЯЛО
2520   FE = 11        : REM   РОЗОВИ КРАКА
2530   FA = 12        : REM   ЗЕЛЕНА ГЛАВА
2540   GOSUB 1000
2550   RETURN
```

Сега всичко, което трябва да направите, за да поставите син кон с бяла глава и оранжеви крака с център точка /10,11/ е

```
30    REM       КОНЯТ НА ПЪРВИЯ ИГРАЧ
40    X = 10
50    Y = 11
60    GOSUB 2000
```

Всичко необходимо за поставянето на оранжев кон с център /19,2/ е

```
70    REM       КОНЯТ НА ВТОРИЯ ИГРАЧ
80    X = 19
90    Y = 12
100   GOSUB 2500
```

И двете рисуващи подпрограми, с начални оператори съответно 2000 и 2500, се обръщат към друга подпрограма с начален оператор 1000. Организацията вече става твърде ефективна. След като веднаж сте направили добра подпрограма, която проверява за грешки и използува променливи, зареждани в обръщащата се към нея програма /която може да бъде главната програма или някоя друга подпрограма/, Вие можете да строите на тази основа цяла структура от подпрограми. По този начин е много по-леко да бъдат писани сложни главни програми. С помощта на трите подпрограми е твърде лесно да се организира оригинално разполагане на конете по екрана.

Но първо още една удобна подпрограма:

```
3000  REM       ИЗБИРА СЛУЧАЙНИ X И Y
3010  X = INT(RND(1) * 38) + 1
3020  Y = INT(RND(1) * 37) + 1
3030  RETURN
```

И сега отново главната програма.

```
10    REM  ПРЕМИНАВАНЕ В ГРАФИЧЕН РЕЖИМ
20    GR
30    REM  ИЗБИРАНЕ НА СЛУЧАЙНА ТОЧКА
40    GOSUB 3000
50    REM  ПОСТАВЯНЕ НА СИН КОН С ЦЕНТЪР ТАЗИ ТОЧКА
60    GOSUB 2000
70     REM ИЗБИРАНЕ НА ДРУГА СЛУЧАЙНА ТОЧКА
```

```
 80 GOSUB 3000
 90 REM    ПОСТАВЯНЕ НА ОРАНЖЕВ КОН С ЦЕНТЪР ТАЗИ ТОЧКА
100 GOSUB 2500
110 REM    ВСИЧКО ЗАПОЧВА ОТНАЧАЛО
120 GOTO 30
```

Ето така трябва да изглежда една главна програма, ако сте добър програмист: главно REM-овете и GOSUB-овете. Работата трябва да бъде извършвана от относително къси подпрограми, всяка от които е пълна сама по себе си и е лесно да бъде написана. Не се стеснявайте да използувате TRACE, за да проследите как Вашата програма си върши работата.

ГРАФИЧНИ ИЗОБРАЖЕНИЯ С ВИСОКА РАЗРЕШАВАЩА СПОСОБНОСТ

Изображенията върху екрана на компютъра, които използувахте досега, се наричат изображения с <u>ниска разрешаваща способност</u>. В този раздел ще се научите да използувате друг вид изображения, наречени изображения с <u>висока разрешаваща способност</u>. Този нов начин на изобразяване ви позволява да нанасяте върху екрана много повече детайли, отколкото това позволява по-грубата решетка, която използувахте досега. При новите изображения екранът е разграфен на 280 на 160 точки /вместо 40 на 40, както беше при старите/. Хоризонталните координати започват с 0 в лявата част на екрана и завършват с 279 в дясната. Аналогично, вертикалните координати започват с 0 в горната част на екрана и завършват със 159 в долната му част.

Работата с висока разрешаваща способност не е трудна за разбиране. В повечето случаи командите се различават от съответните команди при работа с ниска разрешаваща способност само при буквата H в началото. Доброто познаване на изображенията с ниска разрешаваща способност ще бъде от голямо значение при работата с този раздел.

Напишете

```
HGR
```

за да преминете в графичен режим с висока разрешаваща способност. Тази команда изчиства екрана до черно, като оставя за работа в текстов режим последните четири реда в долната

част на екрана.
Както и в случая с ниска разрешаваща способност, графичните изображения с висока разрешаваща способност позволяват да използувате вертикални координати, които попадат в областта на текста /максимумът е 192/, но тези точки изобщо не се появяват на екрана. Ако курсорът не се вижда /защото е"покрит" от графичната част на екрана/, натиснете няколко пъти RETURN, докато се появи в текстовата област.

Графиките с висока разрешаваща способност са наистина чудесни, но човек трябва да направи някои жертви, когато ги използува. В този графичен режим има по-малко цветове. Тук цветовете вървят от 0 /черно/ до 7 /бяло/ и са следните:

0 черно 1	4 черно 2
1 зелено	5 оранжево /червено/
2 виолетово	6 синьо
3 бяло 1	7 бяло 2

Тези цветове варират в зависимост от монитора и в зависимост от мястото на точката върху екрана. Точка, нарисувана например с цвят номер 3, ще бъде синя, ако хоризонталната и координата е черна, зелена, ако хоризонталната и координатната е нечетна и бяла само ако бъдат нанесени една до друга по хоризонтала две точки. Това се дължи на начина, по който работят някои монитори.

Единствената команда за работа с висока разрешаваща способност е HPLOT. За да проверите това, след като сте подали командата HGR, напишете

```
HCOLOR = 3
HPLOT 130, 100
```

Последната команда ще нанасе малка бяла точка с координати X = 130, Y = 100.

Рисуването на прави линии става дори по-бързо при високата разрешаваща способност. Достатъчно е в HPLOT-оператора да се укаже свързването на две точки от екрана с права линия. Например, за да прекарате права по горния край на екрана е достатъчно да напишете

```
HPLOT 0,0 TO 279,0
```

Ако искате сега да свържете точката с координати 279,0 с долния десен ъгъл на екрана, трябва просто да напишете

```
HPLOT TO 279,159
```

и желаната линия се появява на екрана. Когато бъде подадена последната команда, новата права взема за свое начало и цвят тези на последната поставяна върху екрана точка /дори ако междувременно е била подадена друга команда HCOLOR за установяване на цвета/. Можете при желание да "навържете" няколко такива команди и да прекарате няколко прави с един оператор.

Изчистете екрана с HGR и опитайте следната команда:

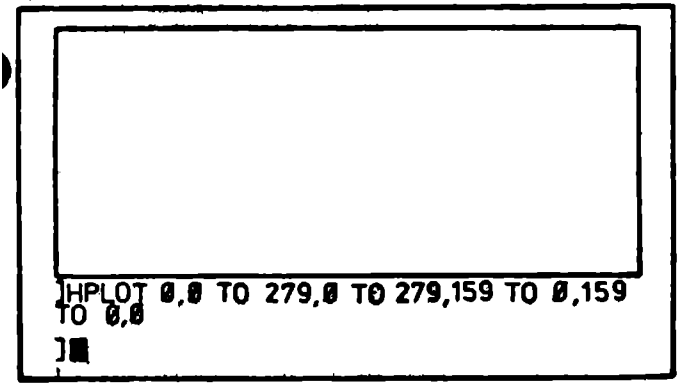
```
HPLOT 0,0 TO 279,0 TO 279,159 TO 0, 159 TO 0,0
```

След изпълнението й трябва да се получи рамка около екрана. Ако по краищата на екрана няма непрекъсната линия, първо проверете дали правилно сте написали командата. Ако не сте направили грешка при писането, сменете цвета посредством команда HCOLOR и опитайте отново. При някои монитори и при някои цветове изображението се появява само в някои части на екрана.

Ако искате да прекарате права от горния ляв ъгъл на екрана до долния му десен ъгъл напишете

```
HPLOT 0,0 TO 279,159
```

Упражнявайте се известно време да рисувате прави, като променяте дължината, направлението и цвета.



Ето една програма, която превръща екрана на Вашия компютър в статив на художник:

```
NEW
200   HGR
210   HCOLOR = 3
220   X = PDL(0)
230   Y = PDL(1)
240   IF Y > 159 THEN Y = 159
250   HPLOT X,Y
260   GOTO 220
```

Оператор 240 е включен в програмата тъй като функцията PDL дава стойности до 255, а Y-координатата трябва да бъде не по-голяма от 159, тъй като в противен случай точката излиза от графичната част на екрана. Опитайте тази програма. Тя работи, но би била по-добра, ако можеше да рисува непрекъсната линия. Празниците между точките не са винаги желателни. Можете да подобрите програмата, като напишете следните оператори:

```
220   GOSUB 1000
230   HPLOT X,Y
240   GOSUB 1000
250   HPLOT TO X,Y
260   GOTO 240
1000  X = PDL(0) / .913
1010  Y = PDL(1) / 1.6
1020  RETURN
```

Проверете дали сте написали всичко вярно.

Тази програма прави следното. Установява графичен режим с висока разрешаваща способност и след това се обръща към подпрограмата с начален оператор 1000. Тази подпрограма определя стойностите на координатите X и Y. Функцията PDL дава максимална стойност 255. Тъй като хоризонталната координата на екрана има област на изменение между 0 и 279, подпрограмата дели стойността, получена от функцията PDL на .913 и по такъв начин разтяга стойностите, използувани за X-координата до 279. Аналогично, стойностите, получени отPDL(1) се разделят на 1.6, за да бъдат свити в граници от 0 до 159.

Както и при изображенията с ниска разрешаваща способност, координатата, която в същност се използува за нанасяне на точка върху екрана е най-голямото цяло число, което е по-малко или равно на подаваната стойност. Оператор 1020 връща в главната програма и след това оператор 230 нанася точката върху екрана. След това главната програма отново се връща към подпрограмата, последната дава /евентуално/ нови стойности на X и Y и предава управлението този път на оператор 250 от главната програма, който свързва старата и новата точка с права линия. Оператор 260 предизвиква ново изпълнение на цялата програма, с изключение на командите HGR и HCOLOR. Използувайте CTRL C за спиране на програмата.

Има една съществена причина да прекарваме линии, а не да рисуваме отделни точки. Нанасянето на точка върху екрана отнема определено време и когато компютърът нанася точките върху екрана една по една, той не смогва да нанася цялата подавана от устройствата за игра информация. Именно затова първата версия на рисуващата програма оставяше празни места между точките при бързо въртене на потенциометрите. Това се поправя, като всяка нова точка, подадена от устройствата за игра, се свързва с къса линия със старата: нанасянето на свързваща линия между две точки става автоматично и много по-бързо, отколкото ако точките биват нанасяни поотделно. Запишете тази програма върху диск или касета и след това я приведете в действие.

Ето една програма, която рисува красиво "моаре" върху екрана:

```
 NEW
 90 HOME
100 VTAB 24: REM  ПРИДВИЖВАНЕ НА КУРСОРА ДО НАЙ-ДОЛНАТА
                  ЧАСТ НА ЕКРАНА
120 HGR      REM  ПРЕМИНАВАНЕ В ГРАФИЧЕН РЕЖИМ С ВИСОКА
                  РАЗРЕШАВАЩА СПОСОБНОСТ
140 A = RND(1) * 279 : REM  СЛУЧАЕН ИЗБОР НА "A" ЗА ЦЕНТЪР
160 B = RND(1) * 159 : REM  СЛУЧАЕН ИЗБОР НА "B" ЗА ЦЕНТЪР
180 N = INT(RND(1) * 4)+2 : REM СЛУЧАЕН ИЗБОР НА СТЪПКАТА
200 HTAB 15    : PRINT "СТЪПКА" N
220 FOR X = 0 TO 278 STEP N : REM  СТЪПКА ПО ХОРИЗОНТАЛА
240 FOR S = 0 TO 1    : REM  ДВЕ ПРАВИ ОТ X И X + 1
260 HCOLOR = 7 * S    : REM  ПЪРВАТА ПРАВА ЧЕРНА,
                             ВТОРАТА БЯЛА
```

```
280 REM  ПРЕКАРВА ЛИНИЯ ПРЕЗ "ЦЕНТЪРА" ДО ПРОТИВОПО-
         ЛОЖНАТА СТРАНА
300  HPLOT X +S,0 TO,B * 279 - X - S,159
320 NEXT S,X
340 FOR Y = 0 TO 158 STEP N: REM  СТЪПКА ПО ВЕРТИКАЛА
360 FOR S = 0 TO 1        : REM  ДВЕ ПРАВИ, ОТ   И + 1
380 HCOLOR = 7 * S       REM  ПЪРВАТА ПРАВА ЧЕРНА,
                              ВТОРАТА БЯЛА
400 REM   ПРЕКАРВА ЛИНИЯ ПРЕЗ "ЦЕНТЪРА" ДО ПРОТИВОПО-
          ЛОЖНАТА СТРАНА
420 HPLOT 279,Y + S TO A,B TO 0,159 - Y - S
440 NEXT S,Y
460 FOR PA = 1 TO 1500   NEXT PA          :REM  ЗАБАВЯНЕ
480 GOTO 120
```

Това е доста дълга програма; напишете я внимателно и я разгледайте на части /например LIST 0,320/, за да проверите всичко дали е написано правилно. Когато се убедите, че всичко е в ред, можете да я пуснете за изпълнение.

Както сигурно ви е направило впечатление при оператори 320 и 440, един оператор може да служи за NEXT на няколко цикъла. В такива случаи обаче трябва да внимавате да изреждате променливите в NEXT-оператора в съответния ред, за да не допуснете пресичане на цикли.

Прекъснете изпълнението на програмата с

и след това подайте команда

```
TEXT
```

Можете ли да измислите някакви начини за промяна на програмата? След като запишете тази версия на диск или касета се опитайте да накарате стойността на HCOLOR да се променя по случаен начин. Опитайте да рисувате първо червени и след това сини линии или само сини линии.

Има много повече неща, свързани с изображенията с висока разрешаваща способност от тия, които са споменати тук. Когато се почувствувате достатъчно сигурни в употребата на графичните команди, изложени в този раздел, можете да се обърнете към по-подробната литература, свързана с ПРАВЕЦ.

ГЛАВА V

НИЗОВЕ И МАСИВИ

Низ от операции:
Думата "конкатенация": събиране на низове
Други низови функции
Увод в масивите
Съобщения за грешки, свързани с използуване на масиви
Заключение

НИЗ ОТ ОПЕРАЦИИ

Искате ли да видите името си написано наопаки? До сега си играехме с графики и числа, но компютрите могат също да манипулират с букви и символи. Нашият компютър може да борави както със самостоятелни символи, така и с цели низове от символи едновременно. Това изглежда съвсем естествено, след като и хората работят повечето време с групи от символи. Променливите, които съдържат низове от символи, както и числовите променливи имат имена. Правилата, на които се подчиняват имената на низовите променливи са същите, каквито са и правилата, на които се подчиняват числовите, с тази разлика, че имената на низовите променливи завършват обезателно с доларовия знак /$/. Ето няколко примера на имена на низови променливи:

```
NAME$
A$
LIBERAL$
```

Променливата А няма нищо общо с променливата А$ и двете могат да бъдат използувани в една програма.

Ако искате низовата променлива N$ /произнася се "ен-долар"/ да съдържа буквите "ИВАН М. ВАЗОВ", можете да напишете

```
N$ =      "ИВАН М.ВАЗОВ"
```

Обърнете внимание на това, че символите, които зареждате в низова променлива трябва да са заградени в кавички. Командата

```
PRINT N$
```

ще напише на екрана съдържанието на променливата N$ в случая името на патриарха на българската литература. По такъв начин, когато имате низ от символи, който често ви трябва, можете да го заредите в променлива с късо име.

В разширеия БЕЙСИК са предвидени различни операции за обработка на низове. Да допуснем например, че искате да узнаете дължината на даден низ /т.е. броя на символите, които съдържа/. Можете да направите това като напишете

```
PRINT LEN ( "ИВАН М.ВАЗОВ" )
```

или еквивалентното в случая

```
PRINT LEN (N$)
```

и ПРАВЕЦ ще напише на екрана дължината на низа, в този случай 13. Забележете, че интервалите /шпациите/ също се броят за символи.

Броят на символите в един низ може да бъде от 0 до 255. Ако опитате да използувате повече от 255 символа в един низ, ще получите съобщение ? SYNTAX ERROR или ? SRING TOO LONG ERROR. Низ с 0 съдържащи се символи се нарича празен.

В някои случаи може да поискате да работите само с част от даден низ. За тази цел можете да използувате много удобните функции LEFT$, RIGHT$ и MID$.

Ако например искате на екрана да излязат първите четири букви на N$ можете да напишете

```
PRINT LEFT$ (N$,4)
```

и на екрана ще се появи

```
ИВАН
```

Ако напишете

```
PRINT  RINGHT$ (N$,4)
```

на екрана ще се появи

```
АЗОВ
```

Когато в някоя програма използувате низови променливи, те трябва да бъдат заредени вътре в програмата. При командата RUN всички числови променливи се зареждат автоматично с 0 и всички низови променливи се зареждат с празния низ. Ето една къса програма, в която са използувани функциите LEN и LEFT$.

```
NEW
 90 N$ =    "ИВАН М. ВАЗОВ"
100 FOR N = 1 TO LEN (N$)
110 PRINT LEFT$(N$,N)
120 NEXT N
```

Функцията RIGHT$ действува по същия начин, както и LEFT$, с тази разлика, че взема десните символи на низа, с който работи. Променете горната програма, като заместите LEFT$ с

RIGHT$. Какво се получава при изпълнение?

Ако искате да използувате символи, взети от средата на низа, а не от левия или десния края на Вас Ви трябва MID$.

Напишете

```
PRINT MILD$ (N$,6)
```

и компютърът ще отговори с

```
M. ВАЗОВ
```

тъй като М е шестият символ в низа. Сега опитайте тази програма

```
NEW
[illegible]
200 FOR N = 1 TO LEN (N$)
210 PRINT MID$ (N$,N)
220 [illegible]
```

Можете ли да предвидите резултата от изпълнението на тази програма?

Да предположим, че трябва да извадите само "Й.М.В" от низовата променлива N$. В такъв случай трябва да използувате още един аргумент при функцията MID$.

```
PRINT MID$ (N$,4,6)
```

Първото число /4/ указва от кой символ /броено от началото на низа/ трябва да започне вадене на символи. Второто число /6/ указва колко символи да бъдат извадени. Командата се интерпретира от ПРАВЕЦ като "намери четвъртия символ на N$ и, започвайки от него /включително/, напиши следващите шест символа". Променете оператор 210 от предната програма, както е показано по-долу и я приведете в изпълнение.

```
210 PRINT MID$ (N$,N,6)
```

Не продължавайте по-нататък с тази книга, докато не сте проверили напълно действието на функциите LEFT$, RIGHT$ и MID$.

Разгледайте също следната програма:

```
NEW
200   A$ =  "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЬЮЯ"
210   PRINT
220   PRINT "НАПИШЕТЕ ЕДНО ЧИСЛО МЕЖДУ 1 И ";LEN(A$);" "
230   PRINT "И АЗ ЩЕ ВИ КАЖА КОЯ БУКВА Е ТОВА ОТ АЗБУ-
          КАТА.";
240   INPUT P
250   IF P > LEN(A$) OR P < 1 THEN GOTO 210
260   PRINT
270   PRINT MID$(A$,P,1)"Е БУКВА НОМЕР "P"ОТ АЗБУКАТА."
280   PRINT = PRINT
290   PRINT "НАПИШЕТЕ НЯКОЯ БУКВА И АЗ ЩЕ ВИ КАЖА"
300   INPUT "НА КОЕ МЯСТО СЕ НАМИРА В АЗБУКАТА. ";X$
310   FOR N = 1 TO LEN (A$)
320   IF MID$ (A$,N,1) = X$ THEN GOTO 370
330   NEXT N
340   PRINT
350   PRINT "ТОВА НЕ Е БУКВА ОТ АЗБУКАТА."  PRINT
360   GOTO 210
370   PRINT
380 PRINT X$ ;" Е БУКВА НОМЕР ";N;" ОТ АЗБУКАТА."
390   PRINT
400   GOTO 210
```

Тази програма илюстрира някои разпространени техники в програмирането. Обърнете внимание на начина, по който тя намира местоположението на даден символ в низа. Използуването на цикъл за "претърсване" на даден низ, като се върви символ по символ, е много разпространен метод. Обърнете също внимание на ролята на празните места, оставяни в началото и в края на надписите в операторите PRINT и INPUT. Какво би станало, ако премахнем тези празни места? Забележете накрая, че границите, в които действува програмата винаги се задават от LEN(A), а не от предварително известния брой на буквите в азбуката. Това позволява програмата да работи дори ако зададете някой друг набор от символи в оператор 200. Опитайте и ще видите.

Можете да запазите даден низ в различни променливи, като използувате стандартния оператор за присвояване на стойност. Например

```
X$ = A$
```

В случая съдържанието на A$ се копира в X $. Не можете обаче да използувате низова функция или изобщо каквото и да било различно от низова променлива за лява страна на такъв оператор. Например командата

```
MID$ (A$,3,5) = "XYZ"
```

е неправилна и предизвиква съобщение за грешка, но с командата

```
X$ = MID$ (A$,24,3)
```

всичко е наред.

А, да - все още ли искате да видите името си, написано наопаки? Следващата програма прави точно това.

```
NEW
100   REM   ПРОГРАМА,КОЯТО ПИШЕ ИМЕТО ВИ НАОПАКИ
110   INPUT  "НАПИШЕТЕ ИМЕТО СИ И АЗ ЩЕ ВИ ГО ПОКАЖА
             НАПИСАНО НАОПАКИ" ; N$
120   REM   ОБРЪЩАНЕ НА РЕДА НА БУКВИТЕ
130   FOR T = LEN(N$) TO 1 STEP - 1
140   R$ = R$ + MID$ (N$,T,1)
150   NEXT T
160   PRINT   PRINT "ИМЕТОВИ,НАПИСАНО НАОПАКИ Е";R$
170   PRINT : PRINT
180   GOTO 110
```

Опитайте тази програма с няколко имена. След няколко опита ще установите, че нещо не е в ред. Цялата работа е в оператор 140. Ако името Ви например е Таня и го напишете в отговор на предложението на програмата, променливата N$ ще се зареди с ТАНЯ, а променливата R$ - с ЯНАТ. Може би Вашият приятел Гошо е при Вас в този момент и също иска да види името си написано наопаки. Когато програмата за пореден път поиска име, той ще напише името си и по такъв начин ще зареди променливата N$ с ГОШО. След това оператор 140 ще зареди R$ със старото му съдържание плюс съдържанието на N$ взето в обратен ред или с други думи ЯНАТОШОГ. Необходима е команда, която да анулира съдържанието на низовите променливи, така че R$ да може да бъде зареждано всеки път с ново съдържание.

За щастие в разширения БЕЙСИК има такава команда. Това е командата CLEAR. Тя анулира съдържанието на променливите от всякакъв вид, цвят и размер. Добавете към Вашата програма оператора

```
175 CLEAR
```

и отново я приведете в изпълнение.

Командата CLEAR може да бъде подавана и директно от клавиатурата.

Напишете

```
N = 254
PRINT N
```

Сега напишете

```
CLEAR
```

и след това отново

```
PRINT N
```

Както виждате, сега ПРАВЕЦ дава 0.

ДУМАТА "КОНКАТЕНАЦИЯ"

Съществува възможност да добавяте един низ към края на друг, като за целта използувате знака "плюс"/+/. Този процес се нарича конкатенация. Подайте на Вашия компютър следната поредица от команди:

```
C$ = "ДОБРО УТРО"
D$ = C$ + " " + "ПЕШО"
PRINT D$
```

Той отговаря

```
ДОБРО УТРО ПЕШО
```

Конкатенацията е особено полезна, когато се налага даден низ да бъде разделен на части и след това тези части да бъдат отново събрани след някои промени. Например, ако искате са създадете нов низ, който се отличава от D$ по това, че празните места са заменени с тирета, можете да напишете

```
E$ = RIGHT$(D$,5) + "-" + LEFT$(D$,4) + "-" + MID$,7,4)
     PRINT E$
```

и на екрана ще се появи

```
     ПЕШО-ДОБРО-УТРО
```

Ето една програма, която използува конкатенация

```
     NEW
     100 INPUT "ДАЙТЕ МИ ОКОЛО ПОЛОВИН ИЗРЕЧЕНИЕ.  ";НА $
     110 INPUT "СЕГА МИ ДАЙТЕ ДРУГАТА ПОЛОВИНА ОТ ИЗРЕЧЕ-
               НИЕТО.  ";ОТ$
     120 WH$ = НА$ + ОТ$
     130 PRINT
     140 PRINT WH$
     150 PRINT   PRINT   PRINT   GOTO 100
```

Знакът +, когато става дума за конкатенация, няма същите свойства, каквито има аритметичният знак +. Например за всеки две числа A и B A + B е същото, каквото е и B + A. Когато става дума за "събиране" на низове обаче подобна симетрия се наблюдава твърде рядко. Опитайте сами да измислите примери, когато такава симетрия при "събирането" е налице и други, при които няма симетрия.

ДРУГИ НИЗОВИ ФУНКЦИИ

Низове могат да бъдат образувани от почти всички символи, включително и от цифри. Както и при оператора PRINT обаче, символите, заключени в кавички не могат да служат за аритметични операции, дори когато са цифри. За да видите какво става, когато се опитате да направите това, напишете

```
     C$ = "125"
     PRINT C$ + 7
```

ПРАВЕЦ смутено отговаря с

```
     ?TYPE MISMATCH ERROR
```

и не може да се справи с последната команда. Имаме нужда от помощта на функцията

```
     VAL
```

за да можем да се справим с този проблем.

Функцията VAL дава числовата стойност на съдържанието на низа. Напишете

```
PRINT C$
```

и след това

```
PRINT VAL(C$)
```

На пръв поглед и двете команди дават еднакъв резултат. Външният вид обаче мами. Вече знаете, че ако напишете

```
PRINT C$ + 5
```

компютърът отговаря с

```
?TYPE MISMATCH ERROR
```

Сега напишете

```
PRINT VAL(C$) + 5
```

и на екрана се появява победоносно

```
128
```

Обърнете внимание на това, че името на низовата променлива, което е аргумент на функцията VAL, трябва да бъде заключено в скоби.

Какво ще стане, ако поискате да зададете числовата стойност на C$ минус 21 в обикновена /не низова/ променлива? Всичко е просто. Напишете

```
Q = VAL(C$) - 21
```

и след това

```
PRINT Q
```

и вижте какво се е получило. Съдържанието на Q съвпада ли с очакваните Ви? Можете, разбира се, да използувате функцията VAL, за да съберете числовите стойности на два различни низа. За целта създайте нов низ:

```
K$ = "12"
```

и след това напишете

```
P = VAL(C$) - VAL(K$)
PRINT P
```

Опитайте VAL с различни низове, включително и такива, които започват или завършват с букви.

Понякога е необходимо обратното - да превърнем някое число в низ. За тази цел можем да използуваме функцията STR$, която прави именно обратното на това, което прави VAL. Да предположим, че е необходимо числовата променлива P да бъде превърната в низова променлива. Напишете

```
P$ = STR$ (P)
PRINT P
```

и ще разберете как работи STR$. Ето една програма, която използува STR$ и VAL.

```
300 INPUT "НАПИШЕТЕ ЕДНО ЧИСЛО МЕЖДУ 1 И 999999999. ";N$
310 N = VAL(N$)
320 IF N < 1 OR N >999999999 THEN GOTO 300
330 N$ = STR$(N)
340 FOR T = LEN(N$) TO 1 STEP -1
350 P$ = P$ + MID$(N$,T,1)
360 NEXT T
370 PRINT   PRINT    "В ОРИГИНАЛ",N$
380 PRINT : PRINT    "В ОБЪРНАТ ВИД",P$
390 P = VAL(P$)
400 PRINT   PRINT  "ОРИГИНАЛНОТО+ОБЪРНАТОТО=";N+P
410 CLEAR
420 PRINT   PRINT   GOTO 300
```

Разбирате ли как работи тази програма? За какво служат запетайките в операторите 370 и 380? Изтрийте оператор 330, за да проверите каква роля играе в програмата. Първите четири оператора в тази програма демонстрират първите стъпки в създаването на наистина "дуракоустойчива" при вход на програма.Открийте какви входни данни могат да спрат тази програма или да я изведат от равновесие и след това измислете как да хващате подобни входни данни, преди да са успели да нарушат нормалната работа на програмата.

УВОД В МАСИВИТЕ

В този раздел, посветен на масивите, ние използуваме някои математически примери, но те са от областта на популярната математика и не изискват познания извън елементарната аритметика.

Масивите Ви дават възможност да работите с произволно избран елемент от дадена таблица от числа и програмните възможности, които получавате по този начин компенсират с излишък необходимостта от известна доза мислене и експериментиране при запознаването с тях.

Масивът е таблица от числа. Името на тази таблица се нарича име на масива и за целта може да послужи всяко допустимо име на променлива, например A. Името на масива A е различно и отделно от името на простата променлива A.

За да създадете масив, трябва първо да укажете на компютъра максималния брой елементи, с които този масив трябва да разполага. За целта се използува операторът за оразмеряване DIM. Номерата на елементите на един масив започват от 0, така че, за да укажете на компютъра, че масивът A трябва да има най-много 16 елемента, напишете

```
DIM A(15)
```

Горният оператор DIM ни дава на разположение 16 нови променливи. Те имат същите свойства, каквито имат и променливите, които вече познавате и обичате. Тези променливи са:

```
A(0)
A(1)
A(2)
```

и така нататък до

```
A(15)
```

Въпреки, че могат да Ви се сторят неудобни за писане, те могат да бъдат използувани така, както се използуват всички други променливи. Командата

```
A(9) = 45 + A(12)
```

е съвършено правилна. Числото в скобите се нарича индекс, а

означението се чете "А-дванадесет" или "А от дванадесет". Индексът може да бъде аритметичен израз, променлива или израз от променливи и числа.

Опитайте следната програма. Тя илюстрира използуването на променливи за задаване на индекси и пише на екрана съдържанието на всеки елемент на масива.

```
100 REM  ОРАЗМЕРЯВАНЕ НА МАСИВ,НАРЕЧЕН "D" със 7
         ЕЛЕМЕНТА
110 DIM D(6)
120 REM  ЗАРЕЖДАНЕ НА ЕЛЕМЕНТА
130   FOR NU = 0 TO 6
140 D(NU) = NU + 1
150 NEXT NU
160 REM  ПИСАНЕ ВЪРХУ ЕКРАНА НА ЕЛЕМЕНТИТЕ НА МАСИВА
170 FOR I = 0 TO 6
180 PRINT "D(";I;") =M";D(I)
190 NEXT I.
```

Ако даден масив бъде използуван от програмата преди да е бил оразмерен, /т.е. преди в хода на изпълнението на програмата да е бил срещнат оператор DIM, оразмеряващ този масив /ПРАВЕЦ-версията на БЕЙСИК запазва за него 11 елемента/ с индекси от 0 до 10./ Въпреки това, оразмеряването на всеки масив е полезен навик.

Да предположим, че искате да напишете програма, която да дава на изхода си числата от едно до осем в случаен ред. За да постигнете това Ви е необходимо да можете да работите с таблици от данни. В подобни случаи масивите са чудесно средство. Вижте например следната програма:

```
NEW
200 REM   ОРАЗМЕРЯВАНЕ НА МАСИВА
210 DIM GL(8)
220 REM   ЗАРЕЖДАНЕ НА МАСИВА
230 FOR I = 1 TO 8
240 GL(I) = I
250 NEXT I
260 REM  РАЗМЕСТВАНЕ НА ЕЛЕМЕНТИТЕ НА МАСИВА
270 FOR WI = 1 TO 8      : REM ПОРЕДЕН ИЗБОР НА ЕЛЕМЕНТ
280 REM  СЛУЧАЕН ИЗБОР НА ДРУГ ЕЛЕМЕНТ
290 MI = INT(RND(1) * 8) + 1
```

```
300 REM   РАЗЛИЧНИ ЛИ СА ДВАТА ЕЛЕМЕНТА?
310 REM   АКО НЕ, НОВ СЛУЧАЕН ИЗБОР
320 IF MI = WI THEN GOTO 280
330 REM   ДВАТА ЕЛЕМЕНТА СМЕНЯТ МЕСТАТА СИ
340 TE = GL(WI)   GL(WI) = GL(MI)   GL(MI) = TE
350 NEXT WI
360 REM   ИЗВОД ВЪРХУ ЕКРАНА НА СЪДЪРЖАНИЕТО НА МАСИВА
370 FOR C = 1 TO 8
380 PRINT GL(C)
390 NEXT C
```

Разбирате ли как работи тази програма? Първо тя зарежда един масив с числа и след това разбърква елементите му по случаен начин. Забележете, че не е задължително да започнете да зареждате от нулевия индекс,

Ето описание на дейността на някои по-трудни за разбиране елементи на програмата. Оператори 230,240 и 250 зареждат масива, като присвояват на всеки елемент стойност, съответстваща на индекса му. /GL(1) = 1./. Оператор 270 зарежда последователно променливата WI с числата от едно до осем. Оператор 290 присвоява на променливата WI случайно цяло число между 1 и 8. Оператор 320 проверява всеки път дали стойността MI не е равна на стойността на WI. Съдържанията на GL(WI) и GL(MI) се разменят в оператор 340. Накрая съдържанието на масива се извежда на екрана от оператори 370,380 и 390.

Размяната на стойности, която се извършва в оператор 340 може да се разглежда по следния начин. Да си представим, че имаме две чаши - едната пълна с вино, а другата с мляко и искаме да разменим съдържанието им. За щастие в този момент разполагаме и с трета чаша, която е празна. Сега можем да прелеем виното в празната чаша, след това да прелеем млякото в чашата, в която преди малко имаше вино и накрая да прелеем виното от третата чаша в чашата, в която в началото имаше мляко. В резултат на тези манипулации съдържанието на първите две чаши е разменено. Ако искате да се убедите с очите си, че този метод на размяна работи, можете да извършите горния експеримент на практика. Полученият винено-млечен коктейл в едната чаша и млечно-виненият коктейл в другата, ще бъдат една от наградите за Вашите усилия.

СЪОБЩЕНИЯ ЗА ГРЕШКИ,СВЪРЗАНИ С ИЗПОЛЗУВАНЕ НА МАСИВИ

Ето няколко съобщения за грешка, които можете да предизвикате при работа с масиви.

?REDIM'D ARRAY

Това съобщение за грешка се получава, когато един масив бъде оразмерен за втори път в една и съща програма. Често тази грешка се получава, когато даден масив е бил използуван без оразмеряване /спомнете си, че ПРАВЕЦ-версията на БЕЙСИК автоматично дава на всеки срещнат в програмата масив размереност 10/ и след това в хода на изпълнението на програмата е срещнат оператор DIM, оразмеряващ същия масив.

?BAD SUBSCRIPT ERROR

Ако бъде направен опит да се използува елемент, който е извън размерността на масива се получава това съобщение. Например, ако на масива A е била дадена размерност 25 с оператор DIMA(25), опитът за обръщение към елемент A(52) или който и да било елемент с индекс по-голям от 25 или по-малък от 0 ще даде съобщение ?BAD SUBSCRIPT ERROR.

?ILLEGAL QUANTITY ERROR

Това съобщение ще получите, ако използувате отрицателно число за индекс на масив.

Това са някои от начините, по които можете да използувате масиви. Масивите, използувани в тази книга са едномерни. Могат да бъдат използувани и масиви с две или повече размерности. За целта вижте по-подробната литература към ПРАВЕЦ.

ЗАКЛЮЧЕНИЕ

Тази книга дава основата на разширения БЕЙСИК. Ако сега отново разгледате книгата, като пишете свои програми с командите, които са дадени в нея, Вие значително ще затвърдите познанията си. ПРАВЕЦ има много повече възможности и след като веднаж добре сте овладели тези, които са дадени тук, пред Вас стоят за изследване цели нови светове.

ДОПЪЛНЕНИЯ

Допълнение А: Списък на командите
Добълнение Б: Запазени думи в ПРАВЕЦ-версията на БЕЙСИК
Допълнение В: Средства за редактиране
Клавиши с лява и дясна стрелка
Чисти движения на курсора
Премахване на редове от програмата
Списък на средствата за редактиране
Допълнение Г: Съобщения за грешка

ДОПЪЛНЕНИЕ А: СПИСЪК НА КОМАНДИТЕ

/Това допълнение съдържа команди както от БЕЙСИК, така и от ДОС /

Тук е изложен списък на команди, които могат да бъдат използувани в разширения БЕЙСИК. За по-големи подробности по тях вижте подробната литература към ПРАВЕЦ.

CALL -151

Предизвиква появата на звезда в левия край на екрана, показваща, че сега ПРАВЕЦ разговаря на своя роден език, наречен машинен език. Ако не се твърде напреднали в програмирането, по всяка вероятност няма да се нуждаете от тази команда.

CATALOG

Разполага върху екрана списък на всички файлове записани върху диска, който се намира в текущото или в указаното в горната команда дисково устройство. Вида на файла и броя на секторите, които той заема се указват от компютъра вляво от името на файла върху екрана. Видовете файлове са

I Файлът представлява програма,написана на целочислен БЕЙСИК.

A Файлът представлява програма, написана на разширен БЕЙСИК.

T Файлът съдържа текстов запис: бил е създаден с команда WRITE.

B Файлът представлява запис,направен бит по бит на част от паметта на ПРАВЕЦ.

R Файлът представлява програма на машинен език в относителни адреси.

Командата CATALOG е команда от ДОС /Дисковата Операционна Система/. Тя не е елемент на разширения БЕЙСИК, но може да бъде използувана програмно, при положение, че дисковата операционна система е в действие.

CLEAR

Зарежда с нула всички числови променливи и опразва всички низови променливи.

COLOR = 12

Установява цвета при графичния режим с ниска разрешаваща способност. В дадения случай установеният цвят е зелен. Командата GR установява цвят Ø /черен/.

Имената на цветовете и съответните им номера са:

Ø черен	4 тъмнозелен	8 кафяв	12 зелен
1 тъмнопурпурен	5 сив	9 оранжев	13 жълт
2 тъмносин	6 син	1Ø сив	14 аквамарин
3 пурпурен	7 светлосин	11 розов	15 бял

CONT

Ако изпълнението на програмата е било преустановено чрез STOP, END или CTRL C, след команда CONT изпълнението продължава от следващия оператор, а не от следващия ред. Всички променливи остават непроменени. Не можете да използувате CONT, ако

а/ сте променили, добавили или изтрили нещо в програмата, или

б/ сте получили съобщение за грешка при спирането на програмата.

CTRL C

Може да бъде използуван за прекъсване на изпълнението на програма или за прекъсване на изпълнението на команда LIST. Също така може да бъде прекъснат INPUT, ако първият входен символ е CTRL C. Прекъсването в този случай става след натискане на RETURN.

CTRL X

Дава указание на компютъра да игнорира реда, който се пише в момента, без да изтрива реда от програмата, ако има такъв, със същия номер.

В момента на подаването на CTRL X на мястото на текущата позиция върху екрана се появява обратна наклонена черта (\).

DEL 23,56

Премахва всички оператори или групи от оператори, чиито номера са в указаната област. В дадения случай всички оператори или групи от оператори, чиито номера са между 23 и 56 включително ще бъдат изтрити. За да изтриете отделен оператор /група от оператори/ с даден номер, например 350, можете да използувате DEL 350,350 или просто да напишете 350 и да натиснете RETURN.

DIM N%(50)

При изпълнението на оператор DIM в паметта на компютъра се запазва място за указания масив с индекси от 0 до зададеното в оператора DIM число включително. В дадения случай за масива N% ще бъдат запазени 50 + 1 или 51 низови променливи. Ако елемент на масив бъде използуван преди да бъде оразмерен с оператор DIM, за всяка от размерностите му се задава максимален индекс 10. Елементите на всички числови масиви се зареждат с 0, а тези на низовите масиви се опразват при командите RUN и CLEAR.

END

Предизвиква прекратяване на изпълнението на програмата и връща управлението на потребителя. Не дава никакви съобщения.

ESCI или ESCJ или ESCK или ESCM

Клавишът ESC може да бъде използуван заедно с клавишите I или J или K или M за придвижване на курсора по екрана, без това да има някакво отношение към символите през които се преминава. За да движите курсора по този начин, натиснете един път клавиша ESC и влезте по този начин в редакторски режим. След това, като използувате клавиша със съответната буква, можете да придвижвате курсора по екрана във всяка желана от Вас посока. Всяко натискане на такъв клавиш придвижва курсора на една позиция в съответната посока. За да ускорите движението, можете да използувате кла-

вива RPT, като го държите натиснат заедно с някой от клавишите I, J, K или M.

команда	придвижва курсора една позиция
I	нагоре
J	наляво
K	надясно
M	надолу

FLASH

Изменя режима на работа с видеомонитора така, че изходът на компютъра върху екрана бива показван последователно с бели символи на черен фон и черни символи на бял фон. За връщане в стандартното /нормалното/ състояние, използувайте командата NORMAL.

```
FOR W = 1 TO 20N...NEXT W
FOR Q = 2 TO -3 STEP -2 ... NEXT Q
FOR Z = 5 TO 4 STEP 3 ... NEXT Z
```

Дава възможност за построяването на "цикъл", който да изпълнява указан от програмиста брой пъти набора от команди между началото на цикъла /командата FOR и края на цикъла /командата NEXT/. В първия пример променливата W "брои" колко пъти се изпълнява наборът от команди в цикъла; този набор команди ще бъде изпълнен за W равно на 1,2,3...,20, след което цикълът завършва работата си / с W + 21/ и се изпълнява операторът след NEXT W. Вторият пример показва проверка и затова наборът от команди в третия цикъл ще се изпълни само веднаж.

GOSUB 250

Предизвиква предаване на управлението на оператора, чийто номер е указан / в случая 250/. Когато в по-нататъшния ход на програмата бъде срещнат оператор RETURN, управлението се предава на оператора, непосредствено следващ последния изпълнен оператор GOSUB.

GOTO 250

Предава управлението на оператора с указания номер

/ в случая 250/.

Установява графичен режим с ниска разрешаваща способност /40 на 40/, като оставя в долния край на екрана четири реда от текстовата страница. Екранът се изчиства до черно, курсорът се премества в тектовия прозорец /последните четири реда/ и цветът се установява на 0 /черен цвят/.

HCOLOR = 4

Установява цвета при графичния режим с висока разрешаваща способност. Имената на цветовете и съответните им номера са:

0 черен1	4 черен2
1 зелен	5 червен
2 виолетов	6 син
3 бял1	7 бял2

HGR

Установява графичен режим с висока разрешаваща способност /280 на 160/, като оставя в долния край на екрана четири реда от текстовата страница. Екранът се изчиства до черно и се работи с първа страница от паметта. Изпълнението на HRG не влияе нито върху HCOLOR, нито върху текстовата страница. Курсорът не се премества в текстовия прозорец /последните четири реда/.

HGR2

Установява графичен режим с висока разрешаваща способност за целия екран /280 на 192/, като не оставя редове от текстовата страница. Екранът се изчиства до черно и се работи с втора страница от паметтта. Изпълнението на HGR2 не влияе нито върху HCOLOR, нито върху текстовата страница.

HLIN 10,20 AT 30

Служи за прекарване на хоризонтални прави линии по екрана при графичен режим с ниска разрешаваща способност. Използува цвета, който последен е бил указан в оператора COLOR. Началото /X ± 0 и Y = 0/ на координатната система е горният ляв ъгъл на екрана. В дадения пример се чертае права линия от X = 10 до X = 20, като Y през цялото време е 30.

Можем да кажем още, че точките (10,20) и (20,30)са били съединени с права линия.

```
HOME
```

Придвижва курсора в горния ляв ъгъл на екрана и изчиства екрана от текстово съдържание.

```
HPLOT 10,20
HPLOT 30,40, TO 50,60
HPLOT TO 70,80
```

Рисува върху екрана точки и прави при графичен режим с висока разрешаваща способност, като използува цвета, който последен е бил указан в оператор HLOR. Началото /X = 0, Y = 0/ на координатната система е горният ляв ъгъл на екрана.Операторът в първия пример рисува малка точка с координати X = 10, Y =20.Операторът от втория пример рисува тънка права линия от точката с координати X = 30, У = 40 до точката с координати X = 50, У = 60. Операторът от третия пример прекарва права линия от последната точка, която е била нанесена върху екрана, до точката с координати X=70, У=80, като използува цвета на точката, която е била нанесена последна, който не съвпада задължително с последното указание в HCOLOR.

```
HTAB 23
```

Променя положението на курсора по хоризонтала, като го премества в указаната вертикална ивица от екрана /между 1 и 40/. В дадения случай курсорът ще бъде позициониран във вертикалната ивица с номер 23, без да се променя текущото му положение по вертикала.

```
IF A < 18 THEN A = Q : B = 1   C = 2
IF AN$ = "YES" THEN GOTO 100
IF N<MA THEN GOTO 25
```

Ако изразът след IF има стойност "истина" /т.е. ненулева/, ще бъде изпълнен тогава операторът или наборът от намиращи се след THEN и отделени един от друг с двоеточие /:/. В противен случай всички оператори след THEN се пренеб-

регват и управлението се предава на номерирания оператор, който е непосредствено след IF-оператора. Низовите изрази се сравняват и оценяват лексикографски.

```
INPUT A
INPUT "A = "; B,C$
```

В първия пример на екрана се появява въпросителен знак и компютърът изчаква потребителя да набере число, което след това се присвоява на числовата променлива A. При втория пример на екрана се повявява надписът в кавичките, който може да бъде произволен, след това компютърът изчаква потребителя да набере число/което се присвоява на числовата променлива B/, след това запетая и след това някакъв низ от символи /който се присвоява на низовата променлива C$. Многократните влизания в INPUT могат да бъдат разделяни със запетаи или с натискания на клавиша RETURN.

```
INT(N)
```

Връща най-голямото цяло число, което е по-малко или равно на зададения аргумент. В случая, ако N е 2.389, ще бъде върнато числото 2. Ако N е -45.12376, ще бъде върнато числото -46.

```
INVERSE
```

Установява режим на видеомонитора, при който изходът на компютъра върху екрана бива показван с черни символи на бял фон. За връщане в стандартното /нормалното/ състояние използувайте командата NORMAL.

```
LEFT$ /ПОЛИФОНИЯ",4/
```

Връща отрязък с дължина указания брой символи, взет от левия край на зададения низ. В нашия случай функцията LEFT$ връща ПОЛИ /първите четири символа от ляво/.

```
LEN ("ЕТО ДВА КАМЪКА")
LEN(B$)
```

Връща броя на символите в зададения низ, който може да бъде от 0 до 255 включително. В пръвия случай върнатата стойност ще бъде 14.

```
A = 23,567
A$ = "ВКУСНО"
```

На променливата, представена от името си, написано вляво от знака за равенство /=/ се присвоява числовата или низовата стойност на израза, стоящ отдясно на знака за равенства.

```
LIST
LIST 200,3000
LIST 200-3000
```

Първият оператор предизвиква последователното разполагане върху екрана на операторите на програмата, намираща се в паметта на компютъра. Вторият и третият оператор извършват същото, но само за операторите, чиито номера са между 200 и 3000 включително. За да започнете преглеждането на програмата от началото до оператор 200, използувайте LIST-200 или LIST 200. За да прегледате програмата от оператор 200 до края, използувайте LIST200- или LIST 200. Командата LIST се отменя по време на изпълнение с натискане на STRL C; командата CONT обаче не продължава изпълнението на командата LISTот мястото на прекъсването. Ако искате да извършите временно спиране на движението на операторите по екрана, след което да можете да продължите отново, използувайте CTRL S. За да продължите след това преглеждането на програмата е достатъчно да натиснете който и да било клавиш.

```
LOAD
```

Чете програма от касетофон, написана на разширен БЕЙСИК и я прехвърля в паметта на компютъра. Потребителят трябва да пренавие лентата и да включи касетофона в режим на възпроизвеждане, преди да подаде командата LOAD. Когато компютърът открие върху лентата разбираема за него информация, тъй подава къс звуков сигнал. Същият звуков сигнал се подава след като четенето приключи успешно; на екрана се появява символът], следван от мигащия курсор, което показва на потребителя, че отново има операторски контрол при разширения БЕЙСИК. Изпълнението на командата LOAD може да бъде прекъснато само от RESET.

LOAD ИГРИ

Опитва да намери програмен файл, нареченИГРИ върху диск, който се намира в указаното в самата команда или в текущото дисково устройство. Ако програмата бъде намерена,тя ще бъде прехвърлена в паметта на компютъра; преди това обаче, всяка програма, която евентуално се намира в паметта на компютера ще бъде изтрита. Командата LOAD последвана от име е команда от ДОС /Дискова Операционна Система/.

MID$ /"ЕТО ДВА КАМЪКА",7/
MID$ /"ЕТО ДВА КАМЪКА",3,8/

Връща указаната част от низа. В първия случай ще бъдат върнати символите от седмия символ до края на низа: А КАМЪКА. Във втория случай се получават осемте поредни символа, които започват от третия: О ДВА КА.

NEW

Изтрива текущата програма и всички променливи.

NEXT

Виж обяснението на FOR TO STEP.

NORMAL

Връща текстовия режим на монитора в стандартно състояние /бели символи на черен фон/.

NOTRACE

Отменя режима TRACE. Виж TRACE.

PDL(1)

Връща текущата стойност /тя може да заема стойности между 0 и 255 включително /на указаното устройство за игра. Валидни са номера на устройства за игра от 0 до 3.

PLOT 10,20

Нанася точка върху екрана при графичен режим с ниска разрешаваща способност. В случая точката ще има координати X = 10, У = 20. Цветът на точката се определя от последния изпълнен оператор COLOR, а ако такъв няма цветът е 0 /черен/.

```
PRINT
PRINT A$;"X = ";X
```

Първият оператор предизвиква преминаване на нов ред върху екрана. При положение, че няколко неща трябва да бъдат изведени върху екрана с един оператор PRINT, те трябва да бъдат разделени със запетаи, ако трябва да бъдат нанасяни всяко в отделно поле, а ако всяко трябва да следва непосредствено след предното, разделянето става с точка и запетая. Ако A$ "МОНА", а X е 3, то вторият оператор ще напише на екрана

```
MOHAX = 3
```

```
REM  ТОВА Е ЗАБЕЛЕЖКА
```

Дава възможност в текста на програмата да се вмъкват обяснителни или други бележки, които се игнорират при изпълнение.

```
RETURN
```

Предава управлението на оператора, който непосредствено следва последния изпълнен оператор GOSUB.

```
RIGHT$ /"ПОЛИМЕРИ",4/
```

```
RIGHT$ (S$,2)
```

Връща отрязък с дължина указания брой символи, взет от десния край на зададения низ. В първия случай функциата RIGHT$ връща МЕРИ /последните четири символа отдясно/.

```
RND(5)
```

Връща случайно реално число по-голямо или равно на нула и по-малко от едно. RND(0) връща последното генерирано случайно число. Всеки <u>отрицателен</u> аргумент предизвиква генерирането на съответстващо му случайно число, което е едно и също всеки път, когато се използува този отрицателен аргумент. Ако <u>след това</u> функцията RND бъде използувана с положителни аргументи, това води до получаването на особена периодична поредица от числа. Всеки път, когато RND бъде използувана с положителен аргумент, тя генерира <u>ново</u> случайно число между 0 и 1, освен в случая, когато то е част от поредица, започната от използуването на отрицателен аргумент.

RUN 500

Изчиства всички променливи, стекове и поинтъри и започва изпълнението от оператора с указания номер /в случая 500/. Ако не бъде указан конкретен номер на оператор, изпълнението започва от този оператор в програмата, който има най-малък номер.

RUN ИГРИ

Изпълнява първо командата LOAD, последвана от указаното име /в случая ИГРИ/ и в случай на успешно изпълнение на тази команда, изпълнява команда RUN. Когато е следвана от име, RUN е команда от ДОС /Дисковата Операционна Система/.

SAVE

Записва на касета програма, написана на разширения БЕиСИК и намираща се в момента в паметта на компютъра. Потребителят трябва да включи касетофона в режим на запис и след това да подаде командата SAVE. Компютърът не извършва проверка дали действията на потребителя са правилни; в началото и в края на процеса на записване се подава къс звуков сигнал.

SAVE ИГРИ

Записва върху диск програма, написана на разширения БЕЙСИК и намираща се в момента в паметта на компютъра. Ако върху диска, намиращ се в указаното от горната команда или в текущото дисково устройство не бъде намерен файл на име ИГРИ /в дадения случай/, компютърът създава върху диска нов файл с указаното в командата SAVE име и под това име записва намиращата се в паметта му програма. Ако дискът съдържа файл със същото име и тип, записаната в този файл програма се изтрива и на нейно място се записва новата програма. Не се издава предупредително съобщение. Когато е последвана от име, SAVE е команда от ДОС /Дисковата Операционна Система/

STR$ (12.45)

Връща низ, който представя в низов вид стойността на аргумента. В този случай се получава "12.45".

TAB(23)

Може да бъде използуван само в команда PRINT; аргументът трябва да бъде между Ø и 255 и да бъде затворен в скоби. Ако аргументът е по-голям от текущата стойност на положението на курсора по хоризонтала, курсорът се премества в указаната пишеща позиция, отчитана от левия край на текущия ред. Ако аргументът е по-малък или равен на текущата стойност на положението на курсора по хоризонтала, курсорът не се премества. TAB (Ø) поставя курсора в позиция 256, броено от началото на текущия ред.

TEXT

Установява обичайния не-графичен текстов режим с 4Ø символа на ред и 24 реда.

TRACE

Предизвиква появата на екрана на номера на всеки оператор, в момента на изпълнението му. TRACE не се изключва от RUN, CLEAR, NEW, DEL или RESET. TRACE се изключва от NOTRACE.

VAL("-3.7E4A5ZKI")

Опитва да интерпретира зададения низ като число от първия срещнат не-числов символ и да върне стойността на това число. Ако още първият символ не е числов, връща Ø. В дадения случай се получава - 37ØØØ.

VLIN 1Ø,2Ø AT 3Ø

Служи за прекарване на вертикални прави линии по екрана при графичен режим с ниска разрешаваща способност. Използува цвета, който последен е бил указан в оператор COLOR. Началото /X = Ø и Y = Ø/ на координатната система е горния ляв ъгъл на екрана. В дадения случай се чертае права линия от Y = 1Ø до Y = 2Ø, като през цялото време X = 3Ø. Можем да кажем още, че точките (3Ø,3Ø) и (3Ø,2Ø) са били свързани с права.

VTAB 15

Придвижва курсора до реда върху екрана с указания номер; най-горният ред има номер 1, а най-долният - 24. VTAB не променя положението на курсора по хоризонтала.

ЗАПАЗЕНИ ДУМИ В ПРАВЕЦ-ВЕРСИЯТА НА БЕЙСИК

Следващият списък съдържа <u>всички</u> запазени думи в ПРАВЕЦ-версията на БЕЙСИК. Въпреки, че повечето от тези думи не са разглеждани другаде в това ръководство, списъкът е полезен при създаване на имена на променливи. Отнесете се към по-подробната литература към ПРАВЕЦ, за да разберете как да използувате повече от тези команди.

&	GET	NEW	SAVE
	GOSUB	NEXT	SCALE=
ABS	GOTO	NORMAL	SCRN(
AND	GR	NOT	SGN
ASC		NOTRACE	SHLOAD
AT	HCOLOR=		SIN
ATN	HGR	ON	SPC(
	HGR2	ONERR	SPEED=
CALL	HIMEM:	OR	SQR
CHR$	HLIN		STEP
CLEAR	HOME	PDL	STOP
COLOR=	HPLOT	PEEK	STORE
CONT	HTAB	PLOT	STR$
COS		POKE	
	IF	POP	TAB(
DATA	IN#	POS	TAN
DEF	INPUT	PRINT	TEXT
DEL	INT	PR#	THEN
DIM	INVERSE		TO
DRAW		READ	TRACE
	LEFT	RECALL	
END	LEN	REM	USR
EXP	LET	RESTORE	
	LIST	RESUME	VAL

FLASH	LOAD	RETURN	VLIN
FN	LOG	RIGHT$	VTAB
FOR	LOMEM:	RND	
FRE		ROT=	WAIT
	MID$	RUN	
			XPLOT
			XDRAW

Разширеният БЕЙСИК превръща всяка от тези думи във вътрешен символ и по този начин тя заема само един байт в паметта. Останалите символи от програмата заемат по един байт всеки. Амперсандът /&/ е предназначен за вътрешна употреба от компютъра и не е правилна команда в разширения БЕЙСИК. При изпълнението на този символ като команда управлението се предава на клетка 83F5.

XPLOT е запазена дума, която не съответствува на команда от разширения БЕЙСИК.

Някои запазени думи биват разпознавани като такива от разширения БЕЙСИК само в определени ситуации.

CLOR,HCOLOR,SCALE,SPEED,ROT,
: се интерпретират като запазени думи само ако първият непразен символ след тях е знакът за равенство/=/. Ползата от това не е голяма в случая с COLOR и HCOLOR, тъй като включената в тях запазена дума OR при всички случаи пречи те да бъдат използувани като имена на променливи.

SCRN,SPC,TAB
: се интерпретират като запазени думи само ако първият непразен символ след тях е лява скоба /(/.

HIMEM:
: трябва да включва двоеточието в края си, за да се интерпретира като запазена дума.

LOMEM:
: трябва да включва двоеточието в края си, за да се интерпретира като запазена дума.

ATN — се интерпретира като запазена сума само ако няма интервал между T и N. В противен случай се взема запазената дума AT иТсе разглежда като име на променлива.

TO — се интерпретира като запазена дума, освен случая, когато последният непразен символ преди нея е A и между T и O има интервал /празен символ/. В този случай компютърът образува запазената дума AT и разглежда O като име на променлива.

THEN — се интерпретира като запазена дума освен случая, когато последният непразен символ преди нея е A. В този случай компютърът образува запазената думат AT и разглеждан HEN като име на променлива.

В някои случаи употребата на скоби оправя нещата, но не винаги. Например

```
100 FOR A = LOFT OR CAT TO 15
```

се интерпретира като

```
100 FOR A = LOF TO RC AT TO 15
```

а

```
100 FOR A = (LOFT) OR (CAT) TO 15
```

се интерпретира като

```
100 FOR A = (LOFT) OR (C AT) TO 15
```

ДОПЪЛНЕНИЕ В: СРЕДСТВА ЗА РЕДАКТИРАНЕ
/Клавиши с лява и дясна стрелка/

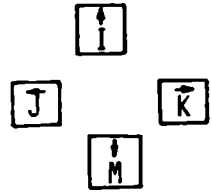
Клавишът с лява стрелка, наричан още възвратен клавиш при всяко натискане придвижва курсора назад/вляво/ на една позиция, като изтрива от текущата програмна памет /но не и от екрана/ символа, през който минава. Изтриването се отнася само до оператора или групата оператори с общ номер, които пишете в момента.

Едно натискане на клавиша с дясна стрелка, наричан още възстановяващ клавиш, придвижва курсора една позиция напред /вдясно/, като възстановява в текущата програмна памет символа от екрана, през който минава. Ако възстановите по този начин група символи, и след това натиснете RETURN, компютърът се държи така, като че сте написали символите на ръка.

Можете да накарате курсора да се движи по-бързо по екрана, като едновременно със съответния клавиш със стрелка държите натиснат клавиша RPT.

Чисти движения на курсора

Клавишите ESC, I, J, K, M се използуват за движение на курсора, без това да има отношение към символите от екрана, през които се минава. Представете си, че върху последните четири клавиша са нарисувани стрелки, както е показано:



Едно натискане на ESC превключва компютъра в редакторски режим. След като веднъж сте влезли в този режим, всяко натискане на някой от горните четири клавиша ще кара курсора да се придвижва по екрана с една позиция по посока на съответната стрелка. Можете да използувате тези клавиши за движения на курсора по целия екран.

За по-бързо придвижване на курсора използувайте клавиша RPT, като го държите натиснат заедно със съответния движещ клавиш /I, J, K или M/. Докато и двата клавиша са натиснати, курсорът се движи бързо по екрана. Когато достигне горния му край той спира. Ако курсорът достигне долния край на екрана, той спира и цялото съдържание на екрана започва да се движи нагоре. Ако достигне десния край, курсорът изчезва и се появява отново откъм левия край на екрана, един ред по-долу. Когато достигне левия край на екрана, курсорът изчезва и се появява отново откъм десния край, един ред по-горе. За връщане в нормален режим натиснете

който и да било клавиш, различен от I, J, K или M.

Премахване на редове от програмата

Един лесен начин за премахване на програмен ред оператор или група оператори с общ номер /от дадена програма е да напишете номера на реда и след това да натиснете RETURN. Ако трябва да изтриете повече редове, можете да използувате командата DEL. Например, за да премахнете всички редове, чиито номера са между 100 и 200 включително, трябва да напишете

```
DEL 100,200
```

Натискането на

CTRL X

изтрива от текущата програмна памет програмния ред, който пишете в момента. Това е полезно в случай, че откриете грешка, преди да сте натиснали RETURN.

Изчистване на екрана

Следващите команди влияят само върху това, което се намира върху екрана, но не и върху програмната памет. Едно натискане на клавиша ESC превключва компютъра в редакторски режим.

За да изчистите целия екран натиснете ESC и след това натиснете знака

SHIFT

ESC 0 @

Екранът се изчиства от всички символи и курсорът се появява в горния ъгъл, непредшествуван от средна скоба. Тя се появява след като натиснете RETURN.

Ако вече сте в редакторски режим, просто натиснете @. ПРАВЕЦ изпълнява командата и се връща в нормален режим.

Командата HOME изчиства екрана.

Има възможност за изчистване само на части от екрана. Ако искате да премахнете символите от дадена точка на екрана до края на екрана, влезте в редакторски режим, като натиснете ESC. След това използувайте чистите движения на курсора и го придвижете до първия символ, който искате да премахнете от екрана. Натиснете клавиша F и всички символи от мястото на курсора до края на екрана ще бъдат премахнати. За да изчистите символите само до края на даден ред, трябва първо да влезете в редакторски режим. След това придвижете курсора до първия символ, който трябва да бъде премахнат и натиснете клавиша E. И в двата случая ПРАВЕЦ изпълнява желаната команда и се връща в нормален режим.

Списък на средствата за редактиране

Влизане в редакторски режим	Натиснете ESC
Излизане от редакторски режим	Натиснете който и да било клавиш,различен от E,F,I,J, K,M,SHIFT,RPT,SHIFT LOCK, или CTRL.
Придвижване на курсора	Натиснете I,J,K или M
Изтриване на символ	Натиснете ←
Възстановяване на символ	Натиснете →
Изчистване на мястото на курсора до края на реда	Натиснете ESC и след това E
Изчистване от мястото на курсора до края на екрана	Натиснете ESC и след това F
Изчистване на целия екран	Натиснете ESC и след това SHIFT и 0 @
Временно спиране на изпълнението на команда LIST	Натиснете CTRL и S
Продължаване на изпълнението на команда LIST	Натиснете кой да е клавиш различен от RPT,SHIFT,CTRL или SHIFT LOCK

ДОПЪЛНЕНИЕ Г: СЪОБЩЕНИЯ ЗА ГРЕШКА

Това е списък на всички съобщения за грешка, които могат да бъдат давани от разширения БЕЙСИК, заедно с техните описания. Повече сведения по въпроса можете да получите от по-подробната литература към ПРАВЕЦ.

При възникване на грешка разширеният БЕЙСИК се връща на командно ниво, индикация за което е символът] в лявата част на екрана, последван от мигащия курсор. Стойностите на променливите остават непроменени, но изпълнението на програмата не може да бъде продължено и всички броячи на подпрограми и цикли се зареждат с Ø.

Когато грешката е възникнала в директна команда съобщението за грешка не съдържа номер на реда.

Формат при съобщенията за грешка:

При директна команда ?XX ERROR

При индиректна команда ?XX ERROR IN YY
/програмна грешка/

И в двата случая XX е названието на конкретната грешка. YY е номера на програмния ред, когато става дума за програмна грешка. Грешките при програмните оператори биват откривани при изпълнението на съответния оператор.

Следва изложение на възможните съобщения за грешка, свързани с разширения БЕЙСИК и тяхното значение.

?CAN'T CONTINUE ERROR

Опит за продължаване на изпълнението на програмата, когато такава липсва или след възникване на грешка, или след промени в програмата.

?DIVISION BY ZERO ERROR

Опит за делене на нула.

?ILLEGAL DIRECT ERROR

Командите INPUT, DEF FN, GET и DATA не могат да бъдат подавани директно от клавиатурата..

?ILLEGAL QUANTITY ERROR

Опит за използуване на числова или низова функция с аргумент извън съответния диапазон. Такава грешка може да възникне поради:

а/ отрицателен ИНДЕКС на масив /напр. A(-1) = Ø/
б/ използуване на функцията LOG с отрицателен аргумент
в/ използуване на функцията SQR с отрицателен аргумент
г/ A^B, при което A е отрицателно и B не е цяло
д/ използуване на MID$, LEFT$, RIGHT$, WAIT, PEEK, POKE, TAB, SPC, ON... GOTO или някоя от графичните функции с неправилен аргумент

?NEXT WITHOUT FOR ERROR

Променливата в команда NEXT не съответствува на променливата на действуващата в момента команда FOR или изобщо в процеса на изпълнението на програмата е била срещната команда NEXT без в момента да има действуваща команда FOR.

?OUT OF DATA ERROR

Изпълнението на команда READ, когато съдържанието на всички оператори DATA в програмата вече е било прочетено. Програмата се опитва да чете твърде много данни или пък в нея са били включени недостатъчно /по брой/ данни.

?OUT OF MEMORY ERROR

Всяко от следните неща може да предизвика това съобщение: прекалено голяма програма; прекалено много променливи; цикли FOR, вложени по-дълбоко от десето ниво; подпрограми GOSUB, вложени по-дълбоко от 24-то ниво; прекалено сложен израз; влагане на скоби по-дълбоко от 36-то ниво; опит да се установи LOMEM: прекалено високо; опит да се установи HIMEM: прекалено ниско.

?OVERFLOW ERROR

Резултатът от някакви изчисления е станал прекалено голям по абсолютна стойност, за да може да бъде представен

чрез предвидения в разширения БЕЙСИК формат на числата. Ако резултатът от изчисленията е твърде малък по абнолютна стойност, той се заменя с Ø и изпълнението продължава без издаване на съобщение за грешка.

?REDIM'D ARRAY ERROR

След като даден масив е бил оразмерен, е направен опит за повторното му оразмеряване. Тази грешка често възниква, когато въпросният масив е бил автоматично оразмерен, поради използуването му в израз от сорта на A(I) = 3 и след това в хода на изпълнението на програмата е срещната оразмеряващата инструкция DIM A(1ØØ). Това съобщение за грешка може да се окаже полезно, ако искате да разберете в кой оператор е бил оразмерен даден масив: въведете оразмеряващ същия масив оператор като първи в дадената програма и съобщението за грешка ще ви покаже /след като приведете програмата в изпълнение/ къде е оригиналният оразмеряващ оператор за този масив.

?RETURN WITHOUT GOSUB ERROR

В хода на изпълнението на програмата е срещнат оператор RETURN, без да е била изпълнена преди това съответната команда GOSUB.

?STRING TOO LONG ERROR

Направен е опит чрез конкатенация да се създаде подълъг от 256 символа.

?BAD SUBSGRIPT ERROR

Опит за обръщане към елемент на масив с индекс, който е по-голям от зададената на масива размерност; опит за обръщане към елемент на масив, като се използува различен брой индекси от този, с който е бил оразмерен масивът. Например A(7,11) = Ø, при положение, че масивът A е бил оразмерен с оператора DIM A(21,21,21).

?SYNTAX ERROR

Липсваща скоба в израз, оператор, съдържащ недопустим символ, неправилна пунктуация и т.н.

?TIPE MISMATCH ERROR

Лявата страна на оператор за присвояване е била числова променлива, а дясната е била низ или обратно. Дадена функция очаква числов аргумент, а е бил подаден низов или обратно.

?UNDEF'D STATEMENT ERROR

Направен е опит чрез оператор GOTO, GOSUB или THEN да бъде предадено управлението на оператор с несъществуващ номер.

?UNDEF'D FUNCTION ERROR

Опит за използуване на потребителска функция, която не е била дефинирана.