## § 5. БЕЙСИК КАК ЯЗЫК ОПЕРАЦИОНАЛЬНО-ПРОБЛЕМНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

У языка Бейсик (Basic) весьма своеобразная судьба. Будучи созданным для, так называемых, непрофессиональных программистов, многократно раскритикованный почти каждым пишущим о программировании, он живет \же четверть века и продолжает иметь множество пусть не поклонников, но пользователей. В своих старших версиях он давно перестал быть столь «простым» как его принято почему-то представлять. Его возможности чрезвычайно велики, о чем можно судить хотя-бы по названию одной из недавно вышедших книг - «Разработка экспертных систем на языке Бейсик». На нем создают программы самой различной предметной ориентации. По-видимому, Бейсик продолжает лидировать по количеству пользователей, и хотя бы поэтому знакомство с ним необходимо.

В данном учебнике нет регулярного, «по-порядку». изложения Бейсика. Для человека, освоившего Паскаль, приведенного ниже в этом параграфе текста достаточно, чтобы составить себе отчетливое представление о Бейсике. Количество же учебников по нему столь велико, что нет смысла приводить их список - достаточно заглянуть в любой библиотечный каталог.

Даже при беглом знакомстве обращает на себя внимание некоторая «недисциплинированность» Бейсика - с точки зрения программиста, привыкшего к структурному языку семейства Паскаля. Бейсик относится к языкам операциональным, рожденным от вечно живого Фортрана, в которых необязательно (хотя и вполне возможно) организовывать строго упорядоченные программные структуры. Это и большой недостаток (особенно при разработке крупных программных комплексов), но иногда и достоинство - например, при разработке относительно небольшой диалоговой программы с регулярным обращением к внешним устройствам, сканированием клавиатуры и т.п.

Еще одна проблема, систематически возникающая при работе с Бейсиком - обилие версий и фактическое отсутствие базовой версии. Оставив обзор до конца данного параграфа, укажем лишь, что многие команды и функции в разных версиях сильно различаются, а иногда, существуя в одной, вовсе отсутствуют в другой. Это следует иметь в виду, если приведенные ниже примеры будут не просто анализироваться, а выполняться на ЭВМ, или по их аналогии будут разрабатываться собственные программы. Справочник по реально используемой версии в таком случае просто необходим.

Тексты приведенных в качестве примеров программ отлажены в широко распространенной версии языка QuickBasic.

### 5.1. ВВЕДЕНИЕ В БЕЙСИК

Выполнять в среде Бейсика элементарные операции и вычисления, особенно в ранних версиях типа GW-Basic или MSX-Basic, действительно нетрудно. Если компьютер включен и Бейсик загружен, можно смело приступить к работе. Начнем с того, что вы хотите что-то вычислить. Бейсик для этого лучше, чем любой калькулятор. Наберите команду

PRINT "Это команда вывода", 5\*5

и нажмите клавишу <Enter> (слово PRINT может заменить знак ?). Немедленно возникает ответ: 25. Команда PRINT выводит на экран результат вычислений или сообщений, заключенных в кавычки: PRINT "Привет!"- на экране появилось «Привет!» (без кавычек).

При вычислениях необязательно, чтобы операндом было выражение, содержащее только числа- Попробуйте ввести следующие команды (заканчивая каждую строчку нажатием на клавишу <Enter>):

а=5 b=4 ?а\*b

Компьютер немедленно выдаст результат: 20. , . ;

Режим работы, описанный выше, часто так и называют- режим калькулятора (или непосредственный режим).

А теперь каждую из представленных выше трех команд пронумеруем, см. программу 54.

*Программа 54*

10 а=5 20 Ь=4

30 ? a\*b

Обратим внимание, что после ввода этой программы команды не выполнялись, а записывались в память компьютера. Убедиться в этом можно введя команду LIST -текст этой маленькой программы тут же появится на экране.

Если теперь ввести команду RUN, то она запустит программу на исполнение. Таким образом, последовательный набор команд с номерами строк является программой на языке Бейсик. Программировать на языке Бейсик означает научиться составлять определенный набор команд для решения поставленной вами задачи. Какие имеются команды у Бейсика и как ими пользоваться - рассмотрим ниже. Режим, при котором команды не выполняются непосредственно, а «копятся», называют косвенным. В этом режиме, основном для Бейсика, он и является языком программирования.

Следует помнить, что существует множество версий языка Бейсик и все они имеют особенности. Описать здесь все версии не представляется возможным, да и нет смысла. В каждой из современных версий Бейсика можно выделить общее подмножество, в котором отражены характерные (стандартные) грамматика, синтаксис и семантика языка. В этой связи в последующих описаниях языка рассматривается лишь выделенное авторами подмножество, справедливое для наиболее популярных в настоящее время версий Бейсика: Basic-MSX, Qbasic, Turbo-Basic. Последние версии приобрели популярность благодаря удобному интерфейсу и предоставлению пользователю ряда сервисных возможностей, присущих современным системам программирования.

### *Контрольные вопросы*

1. Чем принципиально различаются прямой и косвенный режимы в Бейсике?

2. Обязательно ли в Бейсике явное описание типов данных?

### 5.2. БАЗОВЫЕ ОПЕРАТОРЫ

Основные базовые операторы (команды) языка Бейсик определяют ввод и вывод данных, присвоение, изменение порядка выполнения команд и циклические конструкции.

INPUT <список объектов ввода> - ввод данных;

PRINT < список объектов вывода> - вывод данных;

LET a= <арифметическое, логическое

или символьное выражение>

(служебное слово LET можно не писать) - присвоение;

IF <условие> THEN <оператор1>

ELSE <оператор2> - условный оператор;

GOTO <номер строки> -безусловный переход;

FOR х= 1 ТО n STEP h <оператор>

NEXTx - циклическая конструкция.

Часто используют, так называемый, внутренний ввод данных посредством операторов READ - DATA.

Добавим к этому списку несколько системных команд, с помощью которых программист и пользователь занимаются отладкой и обслуживанием программы:

RUN - команда запуска программы на выполнение;

LIST - команда вывода текста программы на экран дисплея;

SAVE - команда сохранения текста программы в виде файла;

LOAD - загрузка ранее сохраненной программы из существующего файла.

Этих операторов и команд обычно хватает, чтобы написать и отладить любую вычислительную программу. Ниже мы познакомимся и с другими командами Бейсика.

Как и во многих языках программирования, в Бейсике имеется набор встроенных функций: математических, логических, символьных и др. Можно сформировать собственные функции с помощью описания DEF, например

DEF FNA(x,y,z)=x\*x+y\*y+z\*z

Рассмотрим пример программы табуляции функции с целью определения ее максимального значения на заданном отрезке. Суть алгоритма заключается в вычислении значений функции Sin(.x) в 100 точках, определенных на задаваемом отрезке [а,Ь] с шагом h=(b-a)/100 и в выборе среди этих значений максимального.

*Программа 55*

10 REM максимум функции на отрезке

20 INPUT "введите отрезок a,b"; a, b

30 DEF fna (x) = SIN(x)

40 max = fna (a)

50 h = (b - a) / 100

60 FOR x *= a* TO b STEP h

70 IF max < fna(x) THEN max = fna(x)

80 NEXT x

85 CLS

90 PRINT "максимальное значение функции на отрезке ["; а;

","; Ь; "]="; max

100 END

Здесь в строке 10 - неисполнимый комментарий (который можно организовать и иначе - см. первые строки последующих программ), в строке 85 - команда очистки экрана, в строке 100 - команда «конец программы» (которую часто можно не писать без каких-либо видимых последствий).

Приведем примеры еще нескольких программ на Бейсике, являющихся аналогами программ из §3.

В следующем примере развилка организована с помощью сочетания операторов IF...THEN и GOTO. В современных версиях Бейсика есть конструкция IF...THEN...ELSE; при ее использовании программа 56 станет еще больше похожей на программу 1 (sqr(x) - корень квадратный).

*Программа 56 (см. программу 1)*

10 REM Квадратное уравнение

20 CLS

30 INPUT "введите коэффициенты А,В,С"; а, Ь, с

40 d *= Ъ\*Ь -* 4\*а\*с

50 IF d < 0 THEN GOTO 80

60 xl = (-b + sqr(d)) / (2 \* a): x2 *=* (-b - sqr(d)) / (2 \* ay

70 PRINT "корни уравнения xl="; xl; "x2="; х2

80 GOTO 100

90 PRINT "корней нет"

100 END

В следующем примере цикл (с предусловием) организован с помощью операторов IF...THEN и GOTO.

*Программа 57* (см. программу 5).

10 REM Машины, прибывающие на склад

20 CLS

30 num = 0: sum *= О*

40 IF sum >= 100 THEN GOTO 110

50 INPUT "Введите вес груза очередной машины"; w

60 sinn = sum + w

70 IF sum >= 100 THEN GOTO 90

80 num = num + 1: GOTO 90

90 PRINT "груз уже не поместится"

100 GOTO 40

110 PRINT "Количество разгруженных машин =", num

120 END

В программе 58 исходная последовательность задается в блоке данных (строка 30), а затем считывается в массив а. Отсутствие в Бейсике логических переменных преодолено с помощью текстовой переменной р. При выдаче результатов используется форматная печать PRINT USING (для дробных чисел форматы могли бы быть, например, ###.## - выдаст результат типа 345.98, *##.##лллл* - выдаст результат типа 34.17Е-02). В 20-й строке оператор DIM резервирует память под массив (аналог паскалевского описания array, но без явного указания типа элементов).

*Программа 58* (см. программу 10).

10 REM bubble (пузырек)

20 DIM a(10)

30 DATA 19, 8, 17, 6, 15, 4, 13, 2, 11, 0

40 CLS

50 PRINT "Исходный массив"

60 FOR i *=* 1 TO 10 STEP 1

70 READ a(i)

80 PRINT USING "»#»#"; a(i);

90 NEXT i

100 PRINT

110 p$ = "да"

120 FOR i = 10 TO 2 STEP -1

130 IF a(i) >= a(i-l) THEN GOTO 150

140 b = a(i): a(i) = a(i-l): a(i-l) = Ь: р$ = "нет"

150 NEXT i

160 IF p$ = "нет" GOTO 110

170 PRINT "Упорядоченный массив"

180 FOR i = 1 TO 10

190 PRINT USING "####"; a(i);

200 NEXT i

210 END

### *Контрольные вопросы ч задания*

1. Назовите основные операторы языка Бейсик и охарактеризуйте их назначение.

2. Поэкспрементируйте в приведенных выше программах с оператором форматной печати.

### 5.3. МУЗЫКАЛЬНЫЕ ВОЗМОЖНОСТИ

В ряд версий Бейсика встроен специальный подъязык, который позволяет программировать музыку. Это делается без труда командой PLAY. В команде указывается строковое выражение, составленное из символов, кодирующих ноты.

Общий формат команды PLAY таков:

PLAY AS, B$, CS,

где AS, BS, CS - строки символов, число которых не превышает числа параллельных звуковых каналов в системе. Каждый символ - команда специального языка, задающая ноту, длительность и громкость звучания, другие параметры. Следующие латинские буквы соответствуют основным нотам:

с - «до»,

d - «ре»,

е - «ми»,

f - «фа»,

g - «соль»,

а - «ля»,

b - «си»

Например,команда

**PLAY** "cdefgab"

осуществляет музыкальное исполнение гаммы (до-ре-ми-фа-соль-ля-си) четвертой октавы с длительностью 1/4 ноты (устанавливаемой, при отсутствии явных указаний, «по умолчанию»).

Нота может звучать на полтона выше (бемоль - знак «минус») или ниже (диез -знак «плюс», либо #). Высота звучания нотьГопределяется октавой. Октава кодируется символом с номером On. помещаемым перед нотой или группой последующих нот. Принято кодировать октавы по порядку с единицы в соответствии с фортепианным рядом, первая октава (01) - контроктава, вторая октава (02) - большая октава, третья октава (03) - малая октава, четвертая октава (04) - первая октава, пятая октава (05) - вторая октава и т.д. Например, команда

PLAY "оЗ а а а"

проигрывает три ноты ля-ля-ля в малой октаве.

Длительность звучания ноты измеряется в специальных единицах - одна целая нота (произнесите четко «раз-и. два-и, три-и, четыре-и», тогда время, затраченное на эту фразу, приблизительно соответствует длительности одной ноты). Выделяют длительность в полноты (1/2 ноты), четверть, одну восьмую, одну шестнадцатую и т.д. - вплоть до одна шестьдесят четвертая. Длительность звучания в программе устанавливают командой Ln (n = 1, 2, ... 64), включаемой в символьную последовательность, либо непосредственно в самих командах установки нот. Например,

PLAY"C8D2L16fbe"

означает звучание ноты «до» длительностью в одну восьмую ноты, «ре» - в одну вторую, а ноты фа-си-ми звучат длительностью в одн\ шестнадцатую ноты. Аналогичную размерность имеет пауза, обозначаемая Rn или Рп.

Уровень громкости может быть установлен командой Vn, предшествующей ноте (n = 0, 1,... 15); по умолчанию n = 8 - «средняя» громкость.

Ниже приведен пример кодирования музыкального произведения.

*Программа 59*

\*\*\*\*\*\*\*\*\*\*\* рпезы Шумана \*\*\*\*\*\*\*\*\*\*\*\*\*\*

10 DATA c8, f2f8, e8, f8, a8, o5c8, f8f2

20 DATA e8, d8, c8, f8, o4g8, a8, b-8, o5d8, o4f8, g8,a8

30 DATA o5c8, o4g2, c4, f2f8, e8, f8, a8, o5c8, a8a4a8

40 DATA g8, f8, d8, c8, f8, o4, g8, a8. b-8, o5, d8, o4, g8, a8, b-8

50 DATA o5d8, o4-d8, e8, f2

60 FOR i = 1 TO 46

70 READ f$

80 PLAY f$

90 NEXT i

Использование функции считывания п символов нажатых клавиш INPUTS(n) позволяет использовать клавиатуру как музыкальный инструмент.

*Программа 60*

5 \*\*\*\*\*\*\* компьютер - простейший музыкальный инструмент \*\*\*\*\*\*\*

10 f$ = INPUT$(1)

20 PLAY f$

30 GOTO 10

### *Контрольные вопросы и задания*

1. Составьте программу, исполняющую первые такты полонеза Огинского.

2. Разработайте компьютерный вариант популярной игры «Угадай мелодию».

### 5.4. ГРАФИЧЕСКИЕ ВОЗМОЖНОСТИ

Для получения графических образов необходимо задать графический экран (например, SCREEN 9) и масштаб или окно экрана WINDOW (х1,у1) - (х2,у2). Здесь (х1,у1), (х2,у2) - координаты угловых точек экрана, расположенных на диагонали. Если ограничиться этим, то цвет фона будет черным. Для задания фона другого цвета используется оператор COLOR zl, z2, где zl- цвет текста, z2- цвет фона. Например, зададим графический экран, с размерами по горизонтали и по вертикали, соответственно 32 и 24 единицы, с началом координат в центре экрана. Цвет фона- синий (z2=l), а цвет надписей -желтый (zl=14).

10 \*\*\*\*\*\*\*\*\*\*\* иыгтый лист *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

20 SCREEN 9

30 WINDOW (-16,-12)-(16, 12)

40 COLOR 14,1

Изображение точки с координатами (х,у) и заданным цветом осуществляется командой PSET(x,y),z. Здесь z - номер цвета. Точку на бумаге можно изобразить маленьким кружком. Иногда, когда требуется изображение точки покрупнее, буквально рисуют кружок и закрашивают его В этой связи, изображение точки на экране дисплея можно создавать, выводя образ окружности и закрашивая ее в тот или иной цвет. Для создания окружности используется оператор CIRCLE (\,y),r,z, где х,у - координаты центра окружности, г - радикс окружности, z - номер цвета окружности. Закрашивают окружность оператором PAINT (x,y),zl,z , где zl - номер цвета которым закрашивают, z - номер цвета окружности.

*Пример.* При запуске программы на экране появятся 4 «точки» в )гла\ квадрата 10\*10, окрашенные в зеленый цвет (10), и пятая «точка» красного цвета (12) вдвое большего размера в центре экрана Их границы окрашены в белый цвет (15).

*Программа 61*

10 '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Точки \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

20 SCREEN 9

30 WINDOW (-16,-12) - (16,12)

40 COLOR 14, 1

50 '\*\*\*\*\*\*\*\*\*\* изображение точек \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

60 CIRCLE (-5, -5). .25, 15 : PAINT (-5, -5), 10, 15

70 CIRCLE (-5, 5), .25, 15 : PAINT (-5, 5), 10, 15

80 CIRCLE (5, 5), .25, 15 : PAINT (5, 5), 10, 15

90 CIRCLE (5, -5), .25, 15 : PAINT (5, -5), 10, 15

100 CIRCLE (0, 0), .5, 15 : PAINT (0, 0), 12, 15

Допустим, что в компьютерной демонстрации надо обратить внимание на какую-либо точку графического образа. В этом сл>чае точку делают мигающей. Для этого меняют с определенной частотой ее цвет.

*Программа 62*

10 '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* чистый лист \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

20 SCREEN 9

30 WINDOW (-16, -12) - (16, 12)

40 COLOR 14, 1

50 '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* мигающая точка \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

60 FOR 1=1 TO 5

70 CIRCLE (0. 0), .5, 15 : PAINT (0, 0), 14, 15 : DELAY .2

80 'FOR k=l TO 200 : NEXT k

90 CIRCLE (0, 0),.5, 15 : PAINT (0, 0), 12, 15 : DELAY .5

100 'FOR k=l TO 500 : NEXT k

110 NEXT I

Обратим внимание на то, что в этой программе время выдержки красного цвета (12) полсекунды (DELAY .5, либо пустой цикл), а желтого цвета (14) - две десятых секунды. В результате получается красная мигающая точка.

Для акцентирования внимания мигание точки можно сопроводить звуковыми сигналами. Для этого используют оператор ВЕЕР стандартной частоты и длительности. Для издания звукового сигнала заданной частоты и длительности применяют оператор SOUND cl,t, где cl - частота сигнала, t -длительность.

*Программа 63*

1. \*\*\*\*\*\*\*\*\*\*\*\*\*\* Клаксон \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

20 SCREEN 9

30 WINDOW (-16, -12) - (16,12)

40 COLOR 14, 1

110 '\*\*\*\*\* мигающие точки со звуковым сигналом ВЕЕР \*\*\*\*\*\*\*

120 FOR i=l TO 5

130 CIRCLE (-5, -5), .25, 15 : PAINT (-5, -5), 6, 15

140 CIRCLE (-5, 5), .25. 15 : PAINT (-5, 5), 6, 15

150 CIRCLE (5, 5), .25, 15 : PAINT (5, 5). 6, 15

160 CIRCLE (5, -5), .25, 15 : PAINT (5, -5), 6, 15

170 DELAY .2 : BEEP

180 CIRCLE (-5, -5), .25, 15 : PAINT (-5, -5), 10, 15

190 CIRCLE (-5, 5), .25, 15 : PAINT (-5, 5), 10, 15

200 CIRCLE (5. 5), .25, 15 : PAINT (5, 5), 10, 15

210 CIRCLE (5, -5), .25, 15 : PAINT (5, -5). 10, 15

220 DELAY .5

230 NEXT i

240 '\*\*\*\*\* Мигающая точка со звуковым сигналом SOUND \*\*\*\*\*\*

250 FOR i=l TO 5

260 CIRCLE (0, 0), .5, 15 : PAINT (0, 0), 14, 15

270 SOUND 600, .2 : DELAY .2

280 CIRCLE (0,0), .5, 15 : PAINT (0, 0), 12, 15

290 SOUND 300, .5 : DELAY .5

300 NEXT i

В следующем примере демонстрируется работа с окружностями и эллипсами, которые достаточно часто встречаются в графических образах различных объектов. Изображение окружности или дуги окружности выводится тонкими линиями с помощью оператора CIRCLE.

*Программа 64*

'\*\*\*\*\*\*\*\*\*\*\*\*0кружности и дуги \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

20 SCREEN 9

30 WINDOW (-16,-12) - (16,12)

40 COLOR 14,1

50 г\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* задание числа PI \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

60 pi=4\*atn(l)

*iq* >\*\*\*\*\*\*\*\*\*\* изображение окружности и дуг \*\*\*\*\*\*\*\*\*\*

80 CIRCLE (0, 0), 4, 15

90 CIRCLE (5. 5), 4, 14, 0, pi/2 'дуга в первой четверти

100 CIRCLE (-5, 5). 4, 14. pi/2, pi 'во второй

110 CIRCLE (-5,-5) ,4, 14, 0, pi, 3\*pi/2 'в третьей

120 CIRCLE (5,-5), 4, 14, 3\*pi/2,2\*pi 'в четвертой

Для изображения линий используется оператор LINE, который строит линии одной и той же толщины. Однако, при создании графических образов в компьютерных Демонстрациях требуются линии различной толщины. Например, выделяются оси координат на фоне координатной сетки или рисуется график линейной зависимости и т.д. Существует очень быстрый способ получения изображений вертикальных и горизонтальных линий различной толщины. Это достигается выводом на экран узких прямоугольников. Этот прием иллюстрируется на примере построения осей координат:

LINE(-16.-.l)-(16,.l),15,bf LINE(-.l,-12)-(.l,12),15,bf

Линии произвольной формы и толщины, в том числе и прямые, выводятся на экран дисплея при помощи оператора CIRCLE. Заключенный в цикл с достаточно малым шагом, оператор CIRCLE действует аналогично плакатному перу с шириной линии, равной диаметру окружности:

FOR х =-4 ТО 4 STEP .01

у = *0.5\*\''2*

CIRCLE (x,y),.1,15 NEXTx

Если требуется провести очень тонкую линию, то вместо CIRCLE используется оператор PSET (\,y). z, где z - номер цвета, а (х,у) - координаты точки (PSET -точка). Иногда изображения соседних точек соединяют отрезками прямых линий. График функции в этом случае будет представляться ломаной линией.

*Программа 65*

*I* \*\*\*\*\*\*\*\*\* тонкие линии и ломаные кривые \*\*\*\*\*\*\*\*\*\*\*\*

10 SCREEN 9

20 WINDOW (-16,-12) - (16,12)

30 COLOR 14,1

4о у\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* тонкая линия \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

50 FOR х *=* -15 TO 15 STEP .01

60 у = 3\*sin(x/3)

70 PSET (х, у+7), 14

80 NEXT х

90 г\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ломаная линия - 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

100 FOR х = -15 TO 15 STEP .1

110 у = 3\*sin(x/3)

120 PSET (x,y), 11

130 IF x>-15 THEN LINE (x,y) - (xl.yl), 11

140 у1=-У : xl=x

150 NEXT х

160 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ломаная линия - 2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

170 FOR х = -15 TO 15.STEP .1

180 у = 3\*sin(x/3)

190 CIRCLE (х, у-7), .1, 15

200 IF x>-15 THEN LINE (х, у-7) - (xl, yl- 7), 10

210 CIRCLE (xl, yl- 7), .1, 15 : PAINT (xl.yl - 7), 14, 15

220 у1=У : х1=х

230 NEXT х

340 '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Оси \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

250 LINE (-16, 7) - (16, 7), 15

260 LINE (-16, 0) - (16, 0), 15

270 LINE (-16, -7) - (16, -7), 15

Выше упоминалось о способе закрашивания областей, ограниченных линией одного цвета, с помощью оператора PAINT. Оператор LINE (\l,yl)-(x2,y2),z,bf позволяет получать закрашенные прямоугольники. Существуют специальные приемы закрашивания. Они основаны на повторении оператора, закрашивающего элементарную площадку. Причем, это может производиться либо по определенному закону, либо случайным образом. Ниже приведена программа, иллюстрирующая два способа закрашивания (хотя можно придумать и другие).

*Программа 66*

10 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* закрашивание \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

20 SCREEN 9

30 WINDOW (-16, -12) -(16, 12)

40 COLOR 14, 1

50 '\*\*\*\*\*\*\*\*\*\*\*\* постепенное закрашивание \*\*\*\*\*\*\*\*\*\*\*\*

60 LINE (-10.1, -5.1) - (10.1, 5.1), 15, b

70 FOR x = -10 TO 10 STEP .01

80 LINE (x, -5) - (x, 5), 12

90 NEXT x

100 DELAY 2 : ' FOR k=l to 2000 : NEXT k

110 CLS

120 'а\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* гаситель экрана \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

130 FOR i = 1 ТО 2000

140 x = .5\*INT(RND\*64) - 16

150 у = INT(RND\*24) - 12

160 LINE (x, у) - (Х+.5, у+1), 15, bf

170 NEXT i

Следующая программа аналогична программе 31 на Паскале и строит столбчатую диаграмму, наглядно отражающую числовую информацию о населении 6 крупных городов мира: Токио, Гамбурга, Москвы, Бангкока, Мехико и Парижа.

*Программа 67* (см. программу 31)

Ю \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* столбчатая диаграмма \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

20 SCREEN 9

30 COLOR 14, 1

40 DATA 11500, 2300, 9700, 5100, 12400, 8200

45 DATA "Токио", "Гамбург", "Москва", "Бангкок", "Мехико", "Париж"

50 LINE (40, 300) - (550, 300): LINE (40, 300) - (40, 20)

60 FOR k *=* 1 ТО б

70 READ m(k)

80 NEXT k

90 a = m(l)

100 FOR k = 2 TO 6

110 IF m(k) > a THEN a = m(k)

120 NEXT k

130 FOR k = 1 TO 6

140 READ name$(k)

150 NEXT k

160 FOR k = 1 TO 6

170 n *=* 10 \* k + 3

180 LOCATE 23, n: PRINT name$(k)

190 m(k) = 300 - m(k) / a \* 200

200 LINE (k \* 80, 300) - (80 + k \* 80, 300 - m(k)), k + 3, BF

210 NEXT k

220 c$ = HEX$(a)

230 LOCATE 3, 1: PRINT c$

240 c$ *=* HEX$(0)

250 LOCATE 22, 3: PRINT c$ 260 END

Дадим короткие пояснения к использованным в данной программе и не использовавшимся ранее средствам. В представленном примере использован не встречавшийся ранее оператор LOCATE X,Y, останавливающий курсор на позицию х,у координатной сетки дисплея. Сочетание команд LOCATE ... PRINT позволяет выводить на экран, находящийся в графическом режиме, тексты. Функция HEXS преобразует числа в соответствующие символы. Отсутствие команды WINDOW приводит к максимально возможному графическому окну.

### *Контрольные вопросы и задания*

1. Перечислите основные операторы для построения элементов изображений и

охарактеризуйте их работу.

2. Замените в одном из приведенных выше примеров SCREEN 9 на другой экранный режим (например, 12). Что изменилось в работе программы? Если она перестала исполняться, отладьте ее.

3. Изобразите смешную «рожицу», попеременно моргающую глазами с звуковым эффектом.

1. Постройте орнаменты из геометрических фигур.

### 5.5. ОБРАБОТКА СИМВОЛЬНОЙ ИНФОРМАЦИИ

Кроме числовых значений переменные в Бейсике могут принимать значения в виде строки символов. Символьные значения (константы) заключаются в кавычки " ".

Имя символьной (строковой) переменной отличается от обычной числовой переменной добавлением справа специального символа, чаще знака доллара $, например, AS, Fl$, MaxS. Над символьными переменными допустима операция сложения (склейка строк символов); могут быть использованы функции и процедуры обработки символьных величин.

Начнем с примеров программ, аналогичных приведенным ранее на языке Паскаль.

*Программа 68* (см. программу 11)

10 ' \*\*\*\*\*\*\* Количество гласных в тексте \*\*\*\*\*\*\*\*\*\*\*\*

20 с$ *=* "аеиоуыэюяАЕИОУЫЭЮЯ"

30 INPUT "Введите текст"; а$

40 n » О

50 FOR k = 1 ТО LEN(c$)

6t> b$ = MID$(c$, k, 1)

70 FOR i = 1 TO LEN(a$)

80 IF b$ = MID$(a$, i, 1) THEN n = n + 1

90 NEXT i

100 NEXT k

110 PRINT "Количество гласных равно", n

120 END

Здесь использована строковая функция MID(a$,ij) - вырезка подстроки из строки а$, где первый аргумент - исходная строка, второй - начальная позиция подстроки, третий - длина подстроки. Функция LEN(aS) - длина текста.

В следующей программе дополнительно использованы строковые функции LEFT$(a$,i) - выделение подстроки длиной i от левого края заданной строки и RIGHT$(a$,i) - то же, но с правого края.

*Программа 69* (см. программу 12)

10 ' \*\*\*\*\*\*\* Замена в тексте sqr на ехр \*\*\*\*\*\*\*\*\*\*\*\*

20 INPUT "Введите текст"; а$

30 FOR i = 1 ТО LEN(a$) - 2

40 IF MID$(a$, i, 3) О "THEN GOTO 60

50 a$ = LEFT$(a$, i -1) + "+ RIGHT$(a$, LEN(a$) - i - 2)

60 NEXT i

70 PRINT a$

80 END

Приведем еще несколько примеров.

*Программа 70*

10 ' \*\*\*\*\*\*\* игра клавиатурный тренажер \*\*\*\*\*\*\*\*\*\*\*\*

12 DATA "азбука", "год"," .23\*45/56", letter, pc-ibm

15 DATA "дом-home", "Красноярск", "Омск", "Пермь", "Воронеж"

20 FOR i = 1 ТО 10

30 READ a$(i)

40 NEXT i

60 FOR i = 1 TO 10

70 CLS : PRINT : PRINT TAB(IO); a$(i);

100 PRINT : PRINT : PRINT

110 PRINT "введите слово и нажмите <ВК>";

120 LOCATE 8, 5 : х » TIMER : INPUT b$ : CLS

130 у = TIMER : PRINT : n =LEN(a$(i))

140 IF a$(i) <> b$ THEN 190

150 PRINT USING "###.#<"; (y - x) / П;

160 PRINT "сек. на один символ": GOTO 200

170 PRINT : PRINT "слово набрано неверно"

180 FOR m = 1 ТО 5000 : NEXT m

190 NEXT i

Функция TIMER возвращает текущее время системной среды в секундах. В строке 180 предусмотрен пустой цикл для осуществления временной задержки в исполнении программы.

Использование функций LEFTS(aS,n) и MID$(a$,n,k) еще раз продемонстрировано в следующем примере. Здесь же использованы математические функции INT(x) -целая часть числа х, RND(x) - случайное число в интервале от 0 до 1. В программе задается массив (оператор DIM aS(k)) из пяти слов, представленных в списке DATA (30 и 40 строки). Случайным образом выбирается одно из них и предлагается его угадать вводом подходящих символов.

*Программа 71*

5 ' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* угадай слово \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

10 k *=* 5

15 DIM a$(k)

30 DATA "вечер","информатика","университет"

40 DATA "эхо","тра-ля-ля"

50 FOR i = 1 ТО k

60 READ a$ (i)

70 NEXT i

80 CLS

90 r = INT(k \* RND(-TIMER)) + 1

100 n = LEN(a$(r))

110 PRINT "Я задумал слово из "; n; "букв"

120 LOCATE 5, 20: PRINT LEFT$(s$, n) *Ш*

130 t = о *т*

140 DO • 150 LOCATE 8, 12: PRINT "введи букву" *Щ*

160 INPUT h$

170 у = О

180 FOR i = 1 ТО n ] 190 IF h$ = MID$(a$(r), i, 1) THEN | 200 LOCATE 5, 19 + i: PRINT h$: | 210 t=t+l:y=l | 220 END IF |

230 NEXT i

240 LOCATE 8, 12: PRINT " "

250 IF у О 1 THEN LOCATE,10, 15: PRINT "нет такой буквы"

260 FOR m = 1 ТО 5000 : NEXT m 270 LOCATE 10, 15: PRINT " " |

280 LOOP WHILE t 0 n I 290 LOCATE 13, 13: PRINT "угадал, поздравляю!" *f*

Обратим внимание на использованный в данной программе цикл DO-LOOP-WHILE, который допустим в некоторых версиях Бейсика (QBASIC, Turbo-BASIC).

### *Контрольные вопросы и задания*

1. Охарактеризуйте использованные выше строковые функции.

### 5.6. ПОДПРОГРАММЫ

В Бейсике предусмотрена работа с подпрограммами, если необходимо использовать многократно сходные действия, или для блочно-структ\рпрованного построения программы. Для организации подпрограмм используют две команды: GOSUB <номер строки> - перейти на подпрограмм: RETURN - возврат.

Во многих версиях Бейсика предусмотрен переключатель подпрограмм ON k GOSUB 100,200,300. Здесь k может принимать значения 1.2.3, что соответствует обращению к одной из трех подпрограмм, начинающихся с номеров 100. 200, 300.

Начнем с примера, аналогичного одному из тех, которые сопровождали рассказ о процедурах Паскаля выше.

*Программа 72* (см. программу 17).

10 ' \*\*\*\*\*\* квадратные корни и логарифмы\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

15 CLS / 20 " а » 35: с$ = "-"

30 GOSUB 1000 40 PRINT : PRINT "Таблица квадратных корней"

50 GOSUB 1000

55 PRINT

60 FOR x = 1 TO 10

70 PRINT USING "x = ##» sqr(x) = t».#»#<"; x; SQR(x) :

80 NEXT x

90 GOSUB 1000

100 PRINT : PRINT "Таблица логарифмов" ;

110 GOSUB 1000 120 PRINT

130 FOR x = 1 TO 5 140 PRINT USING "x = ««» log(x) = »».»»»»"; X; LOG(x)

150 NEXT x

:60 с$ = "\*" 170 GOSUB 1000 180 END

1000 FOR i = 1 TO a : PRINT c$ : NEXT i

1010 RETURN

Обратим внимание на существенно иной механизм передачи параметров в подпрограмму на Бейсике, нежели в паскалевскую процедуру. Формирование входных параметров производится с помощью присваиваний до обращения к процедуре. Это гораздо менее удобно, чем использование формальных и фактических параметров в Паскале.

Приведем еще несколько примеров работы с подпрограммами.

*Программа 73*

'\*\*\*\*\*\*\*\*\*\*\* меню графиков функций \*\*\*\*\*\*\*\*\*\*\*

10 CLS : PRINT : PRINT

20 PRINT " ГРАФИКИ ФУНКЦИЙ"

30 PRINT

40 PRINT ". Синус"

50 PRINT ". Косинус"

60 PRINT ". Тангенс"

70 PRINT : PRINT

80 INPUT "введи номер"; j

90 SCREEN 2

100 LINE (200, 0) - -(200, 200) : LINE (0, 100) - (400, 100)

110 PSET (0, 0), 4

120 FOR x = - 4 TO 4 STEP 1/32

130 ON j GOSUB 500, 600, 700

140 PSET (200 + 32 \* x, 100 - 32 \* у)

150 NEXT x

160 r$ *=* INPUT$(1)

170 IF r$ = "К" OR r$ = *"к"* THEN END ELSE SCREEN 0: GOTO 10

500 у = SIN(x) : RETURN

600 у = COS(x) : RETURN

700 у = TAN(x) : RETURN

Подпрограммы, к которым обращаются по GOSUB, сильно уступают истинным процедурам Паскаля, поскольку у них отсутствуют возможности локализации переменных, нет формальных параметров, которые можно замещать при обращении фактическими. Однако, в более поздних версиях (например в Qbasic) предусмотрены подпрограммные единицы SUB - подпрограммы-процедуры и FUNCTION - подпрограммы-функции. В следующем примере использована подпрограмма решения квадратного уравнения SUB kvur(a,b,c) со списком входных данных. В основной программе можно объявить подпрограмму оператором DECLARE, а вызвать ее с помощью команды CALL.

*Программа 74*

5 ' \*\*\*\*\*\*\*\*\*\* пример работу с подпрограммой \*\*\*\*\*\*\*\*

10 DECLARE SUB kvur (а!, Ь!, с!)

20 INPUT "введи а, Ь, с"; а, Ь, с 30 CALL kvur (a, Ь, с)

40 END SUB kvur (a, b, c)

1000 d.=b\*b-4\*a\*c

1010 IF d < 0 THEN PRINT "корней нет": GOTO 1050

1020 xl = (- b + SQR(d)) / 2 / a

1030 x2 = (- b - SQR(d)) / 2 / a

1040 PRINT ="; xl, ="; x2

1050 END SUB

### *Контрольные вопросы*

1. В чем подпрограммы Бейсика принципиально отличаются от процедур Паскаля?

### 5.7. РАБОТА С ФАЙЛАМИ

Для осуществления связи с внешними периферийными устройствами и с файловой системой операционной среды в Бейсике предусмотрена работа с файлами. В следующем фрагменте программы показаны команды для открытия внешнего файла с именем "aaa3.dat" на активном устройстве внешней памяти для записи (OPEN-FOR OUTPLT-AS ff). Запись в файл осуществляется командой PRINT или WRITE с указанием номера файла (строка 30). По завершении работы с файлом его закрытие осуществляется оператором CLOSE.

5 ' \*\*\*\*\*\*\*\*\* работа с файлами для записи \*\*\*\*\*\*\*\*\*

10 OPEN "аааЗ.ааГ FOR OUTPUT AS ^3

20FORi=1TO10

30 PRINT #3, i:' либо эквивалентная команда WRITE #3, i

40 NEXT i

50 CLOSE #3

Аналогичные команды используют при извлечении данных из существующего файла. При этом в команде OPEN указывают служебное слово INPUT, а считывание информации происходит по команде INPUT с указанием номера файла.

' \*(\*\*\*\*\*\*\*\* работа с файлами доя чтения \*\*\*\*\*\*\*\*\*\*

60 OPEN "aa3.dat" FOR INPUT AS #2 'для чтения

70 FOR i = 1 TO 10

80 INPUT #2, a(i)

90 PRINT a(i)

100 NEXTi

110 CLOSE #2

Добавить данные в существующий файл можно, используя служебное слово **APPEND.**

' работа с файлами для продолжения записи в существующий файл

120 OPEN "aa3.dat" FOR APPEND AS #1

130 WRITE #1, 13

140 CLOSE #1

Ниже приведен пример двух программ, осуществляющих передачу и прием зашифрованного послания. Текст шифруется заменой символа его порядковым номером в русском алфавите со смещением на 10. Например, код буквы "а"равен 11, буквы "б" - 12 и т.д. Программы вполне аналогичны программам 26 на Паскале, разобранным в параграфе 3, хотя для разнообразия способ шифровки принят несколько иным.

*Программа 75* (см. программу 27 sekret)

5 ' \*\*\*\*\*\*«-\*\* шифровка текста \*\*\*\*\*\*\*\*\*\*\*\*\*\*

10 INPUT "введи текст"; а$ 23 с5 = "абвгдежзийклмнопрстуфхцчшшъыэюя "

30 х$ = " "

40 FOR i = 1 ТО LEN(a$)

50 FOR j = 1 TO LEN(c$)

60 IF MID$(a$, i, 1) - MID$(c$, j, 1) THEN k = j: GOTO 90

70 NEXT j

80 PRINT "таких символов в таблице кодов нет": GOTO 160

90 kk = k + 10

100 x$ *=* x$ + STR$(kk)

110 NEXT i

120 OPEN "letter.txt" FOR OUTPUT AS #1

130 PRINT #1, x$

140 CLOSE »1

150 PRINT "шифровка - "; x$

160 END

Результатом работы программы является зашифрованный текст, записанный в файл с именем letter.txt.

Следующая программа считывает информацию из этого файла и проводит его

раскодирование, используя указанный выше ключ.

*Программа* 76 (см. программу 27 retsek)

5 '\*\*\*\*\*\*\*\*\*•\* расшифровка текста \*\*\*\*\*\*\*•\*\*\*«\*»

10 OPEN "letter.txt" FOR INPUT AS #1

20 INPUT tl, a$

30 CLS

40 PRINT a$

50 c$ *=* "абвгдежзийклшюпрстуфхцчшщъыэюя "

60 x$ *=* " "

70 FOR i *=* 1 TO LEM(a$) STEP 3

SO k = VAL(MID$(a$, i, 2))

90 k *=* k - 10

100 x$ *=* x$ + MID$(c$, k, 1)

110 NEXT i

120 CLOSE #1

130 PRINT "шифровка - "; x$

140 END

### 5.8. СРЕДСТВА И МЕТОДЫ ОРГАНИЗАЦИИ ДИАЛОГА

В **Бейсике** сильно развиты команды и функции ввода-вывода данных, позволяющие осуществлять диалог (интерактивный интерфейс) с пользователем.

Для осуществления клавиатурного управления программой, помимо уже известных операторов ввода INPUT и вывода PRINT, используют функцию INPUT$(n), возвращающею значение введенной символьной строки из п символов с ожиданием, функцию INKEYS, принимающею символьное значение нажатой клавиши без ожидания, функции STICK(n), STRIG(O), определяющие коды стрелок или направления джойстика (внешнего управляющего устройства для компьютерных игр и тренажеров).

Часто при разработке компьютерных программ необходимо использовать коды клавиш в ASCI-представлении. Следующая программа печатает код нажатой клавиши и может оказаться полезной при отсутствии под рукой таблицы ASCI кодов.

*Программа 77* 5 ' \*\*\*\*\*\*\*\*\*\*\*\* коды символов \*\*\*\*\*\*\*\*\*\*\*\*

10 cls

20 f$ *=* INPUT$(1)

30 PRINT "символ="; f$; "код="; ASC(f$)

40 PRINT 50 GOTO 20

Полную таблицу кодов символов можно получить и другим способом, см. программу 78.

*Программа 78* 5 \*\*\*\*\*\*\*\*\*\*\*\* до коду печать символа \*\*\*\*\*\*\*\*\*\*\*\*\*\*

10 FOR i = 1 ТО 256

20 PRINT "код="; i, "символ="; CHR$(i)

30 a$ = INPUT$(1)

40 NEXT I

Функцию INPUTS часто используют для приостановки выполнения программы до нажатия любой клавиши:

п г$ = INPUTS(l): IF r$ = "" THEN n

В отличие от оператора INPLT ф\ нкцпя не ставит знак "?"и не высвечивает вводимые символы. Это свойство удобно использовать для защиты доступа к какой-либо информации. В следующем примере демонстрируется фрагмент проверки «своего» клиента с помощью пароля. Если при вводе четырехсимвольного пароля ("") трижды допущена ошибка, программа приостанавливает работу.

*Программа 79 ;*

5 \*\*\*\*\*\*\*\*\*\* пароль \*\*\*\*\*\*\*\*\*'\*\*

7 CLS

8 PRINT "нажми любую клавишу"

9 r$=INPUT$(l) : IF r$=""THEN 9

10 code$ = "1996" : k = О

20 INPUT "Имя"; а$

30 PRINT "Пароль"

40 р$ = INPUT$(4)

50 IF p$ = code$ THEN GOTO 80

60 k *=* k + 1 70 IF k < 3 THEN GOTO 30 ELSE PRINT "До свидания" : END

80 . PRINT 90 PRINT a$, "добро пожаловать"

100 END

С помощью строковой переменной INKEYS аналогично можно осуществлять временную задержку программы до нажатия любой клавиши:

n sS=INKEYS: IF sS=""THEN n

Ниже приведен пример назначения некоторых клавиш, управляющих процессом рисования точек (простейший графический редактор). Здесь код 27 (строка 20) означает клавишу Esc.

*Программа 80*

*•^ i* \*\*\*\*\*\*\*\*\*\* простейший графический редактор \*\*\*\*\*\*\*\*\*\*\*\* 5 SCREEN 2

6 х *=* 100: у = 100: h = 2

7 PSET (х, у)

10 g$ = INKEY$

20 IF g$ = CHR$(27) THEN 80

30 IF g$ = "A" OR g$ = "a " THEN х = х - h

40 IF g$ *= "S"* OR g$ = "s " THEN х = х + h

50 IF g$ = "Z" OR g$ *= "z"* THEN у = у + h

60 IF g$ = "W" OR g$ *=* "w" THEN у = у - h

70 GOTO 7

80 END

При разработке компьютерных программ полезно бывает использовать функциональные клавиши в качестве «горячих» (НОТ KEY). Назначение команд функциональным клавишам осуществляет оператор KEY. Его работу можно изучить по следующей демонстрационной программе.

*Программа 81*

5 ' \*\*\*\*\*\*\*\*\*\* назначение функциональных клавиш \*\*\*\*\*\*\*\*

10 FOR i = 1 ТО 10

20 KEY i, "+ STR$(i)

30 NEXT i

40 KEY 30, "

50 KEY 31, "+ CHR$(13)

60 KEY LIST

70 KEY ON

80 r$ = INPUT$(1): IF r$ = ""THEN 80

### *Контрольные задания*

1. Составьте программу, подсчитывающую количество слогов в заданном слове.

2. Разработайте собственный компьютерный вариант игры «Поле чудес».

3. Придумайте новые варианты программ шифрования и дешифрования текста.

4. Разработайте клавиатурный тренажер, используя игровые элементы.

5. Разработайте игру «Крестики-нолики».

1. Разработайте компьютерный вариант карточной игры «21».

### 5.9. ВЕРСИИ БЕЙСИКА

Бейсик характерен необычайным обилием версий, которые иногда настолько отличаются друг от друга, что могли бы считаться разными языками.

Хотя первая официальная публикация по Бейсику создавшей его группы разработчиков Дартмутского колледжа (США) относится к 1967 г., к этому времени уже было создано три «Дартмутских версии»; первая из них относится к 1964 г. После 1967 г. к разработке этого языка подключились крупные фирмы, и были созданы версии для систем с разделением времени, для сетевых информационных систем, для мощных ЭВМ третьего поколения (включая отечественную версию «Бейсик ЕС»), для малых ЭВМ.

Большой толчок развитию Бейсика дало появление персональных компьютеров. На какое-то время Бейсик с его естественной ориентацией на трансляторы интерпретирующего типа оказался доминирующим для «персоналок» языком программирования. Бейсик оказался пригоден для первых учебных и бытовых ЭВМ с малым объемом оперативной памяти и невысоким уровнем системного программного обеспечения. Зачастую Бейсик-интерпретатор выполнял и функции операционной системы (например, бывшая популярной в России в конце 80-х - начале 90-х годов версия MSX-Бейсик учебных компьютеров «Ямаха»).

Бейсик начала 60-х годов был прост и примитивен. Он насчитывал всего порядка двух десятков команд. Несмотря на простоту работы с Бейсиком, интерфейс пользователя оставлял желать лучшего. Вся информация (текст программы, «мусор», результаты работы, сообщения) размещались на скроллирующем экране. В силу несложной процедуры разработки транслятора-интерпретатора Бейсика каждый разработчик персональных компьютеров встраивал (обычно в ПЗУ) свой вариант Бейсика.

Бурное развитие микропроцессорной техники в 70-х, затем в 80-х годах и связанные с этим проблемы совместимости заставили разработчиков задуматься о стандарте Бейсика для персональных компьютеров. Аппаратное и программное развитие позволило существенно расширить возможности языка. Бейсик стал системой программирования со всеми сервисными утилитами и инструментарием программиста. Наибольшей популярностью для школьного и педвузовского образования приобрел Бейсик-MSX. Для большинства учебных компьютеров (Ямаха, Корвет, УКНЦ) были разработаны трансляторы версии Бейсик-MSX. Описание и практика работы с этим языком достаточно полно раскрыты в существующих популярных учебниках для вузов и школ.

Бейсик последних версий, в основном, отличается интерфейсом. Например, основное меню Турбо-Бейсика расположено в верхней строке экрана и представляет систему ниспадающих подменю, команды которых позволяют выполнять все необходимые действия по составлению, отладке и выполнению программ. В Турбо-Бейсике существуют несколько экранных режимов. Основной исходный экран представляет поле редактирования текста программы. Встроенный текстовый редактор выполняет две функции - редактирования текста и синхронной интерпретации строки. В этой связи строки нумеровать необязательно. Результаты работы программы, комментарии и сообщения интерпретатора представляются в текстовом экране (по умолчанию режим SCREEN 0). При работе с графикой изображения строятся в графическом экране, который имеет несколько режимов (режимы SCREEN n, п=2,3,...). Аналогичные возможности предусмотрены и в QBasic. Следует отметить, что в последних версиях Бейсика предусмотрены управляющие структуры. Введены операторы выбора SELECT CASE, циклические конструкции с пост- и предусловием WHILE-WEND, DO-LOOP-WHILE, DO-WHILE-LOOP, развит аппарат подпрограмм (SUB-CALL).

Особый интерес представляет система помощи - help (подсказок), имеющая контекстную структуру с возможностью копирования примеров из текста подсказок, что позволяет осваивать работу с языком программирования практически самостоятельно.

### 5.10. БЕЙСИК И ПАСКАЛЬ

Сопоставим эти популярные языки - не в деталях, а в целом. В Бейсике отсутствуют:

• ряд структур данных (множества, записи, перечисляемые и интервальные типы);

• ссылочные типы и динамические переменные (хотя в старших версиях возможно динамическое описание массивов);

• в большинстве версий - процедуры и функции (их слабыми аналогами являются подпрограммы);

• модули (и, следовательно, возможности организации больших внешних библиотек).

По работе с графикой, возможностям обработки строк (текстов) Бейсик не уступает Паскалю, а по возможностям организации диалога, пожалуй, даже превосходит.

Одна из неприятных особенностей Бейсика для тех, кто привык к структурной записи программного кода, состоит в том, что Бейсик провоцирует программиста к неструктурности. Впрочем, это вопрос навыков и привычек. По большому счету, однако, язык, в котором нет настоящих процедур и средств создания внешних библиотек, не может быть языком структурного программирования.

Усложненные версии Бейсика, в которых фигурируют истинные процедуры и другие средства, заимствованные из более мощных языков, производят неоднозначное впечатление, так как теряется одно из главных достоинств языка - простота (лучше уж тогда пользоваться во многих отношениях превосходящим Бейсик языком операционального программирования Фортран-77). При написании же коротких простых программ Бейсик реально не уступает Паскалю (и многие специалисты утверждают, что процедура программирования на Бейсике проще, хотя это спорно). Поскольку современные версии Бейсик-систем используют не интерпретаторы, а компиляторы, эффективность программ не ниже, чем аналогичных на Паскале.