The HP-41C: A Literate Calculator?

Brian P Hayes
Scientific American
415 Madison Ave
New York NY 10017

Calculator vs Computer

The computer and the programmable calculator seem to be following paths of convergent evolution. As the one is made smaller while the other gains in capability, the line of demarcation between them becomes more and more arbitrary. For now at least, the programmable calculator remains a distinct and lesser species, but it shares many of the attributes of the computer. Moreover, the shared attributes are chiefly the ones that make the computer an interesting machine. Both devices offer an intimate acquaintance with the powers and pleasures of algorithms. Both exhibit an enigmatic unpredictability: the response of the machine to any given stimulus is wholly deterministic, yet the behavior of a large program can be full of surprises, often to the frustration of the programmer.

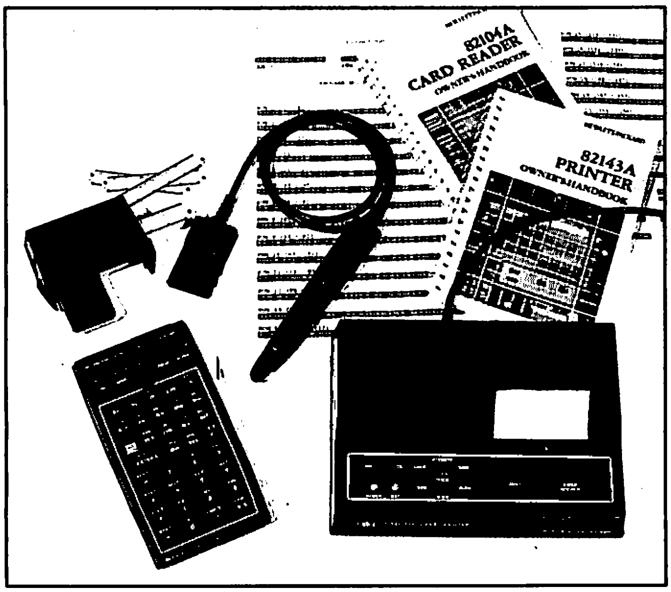


Photo 1: *Components of the Hewlett-Packard HP-41C calculator system. Shown here are the calculator itself and three peripheral devices: a magnetic-card reader, a wand for reading printed bar codes, and a thermal dot-matrix printer. The peripheral units plug into four ports at the top of the calculator, which can also receive modules containing additional memory or precoded applications programs. The HP-41C alone costs about $300; a system including all three peripheral devices and two memory or applications modules is about $1000. (Photo by Ed Crabtree.)*

The HP-41C, which was introduced by the Hewlett-Packard Company about a year ago, is among the programmable calculators that lie closest to the computer borderline. It comes close enough for the jargon of computers to be useful in describing it. At the Corvallis Division of Hewlett-Packard, where the HP-41C is made, they refer to the calculator itself as the "mainframe" and to its accessory devices as the "peripherals." The calculator comes equipped with four input/output (I/O) ports, through which the various elements of the system are interconnected. Because the peripherals do some data processing internally, the system might even be said to have "distributed intelligence."

When compared with a computer, most programmable calculators have a rich instruction set, but they are deficient in memory capacity and in facilities for communication with the user. A calculator comes with such amenities as trigonometric, logarithmic, and statistical functions built in; with a computer, even floating-point arithmetic must usually be constructed out of software. On the other hand, no calculator has the memory needed to store large tables or other data structures. And it is the communication problem that most seriously limits the utility of the calculator. A display that can represent only the 10 digits, a decimal point, and a minus sign does not have much range of expression. Even for problems that have entirely numerical results, such a display is not always adequate, since without labeling of any kind it is easy to become confused about what a number means.

The HP-41C

In the HP-41C, the instruction set is at least the equal of that in any other calculator and the potential memory space is large (although it can never be large enough). The most conspicuous distinguishing features, however, have to do with communications and "human factors" (or, in other words, those things that aid in writing programs and in interpreting their results).

All three of the peripheral units now available serve to get information into or out of the HP-41C; they are a printer, a magnetic-card reader, and a wand for reading bar codes. But perhaps the most significant innovation of all is in the calculator itself: a liquid-crystal display that can represent not only numerals but also the complete uppercase alphabet and a few lowercase letters and other symbols. The letterforms are crude but perfectly legible; what they bring to the calculator is literacy, and it makes all the difference in man-machine relations.

The architecture of the HP-41C is not fundamentally different from that of its predecessors in the Hewlett-Packard line. There is a four-level stack of registers where pending operands are generally held; other registers are identified by a 2- or 3-digit address. The internal memory consists of 63 registers, but this number can be increased by plugging memory modules into the ports. Each module adds 64 registers, so that a full complement of four modules yields a total capacity of 319 registers; with all the ports occupied, however, no peripheral devices can be connected.

The memory available can be divided in any way desired between data storage and program storage. When allocated to data memory, a register holds a single floating-point number (10-digit mantissa and 2-digit exponent). Program capacity is more difficult to measure because instructions have varying space requirements. Without extra memory and with a reasonable allowance for data storage, the maximum for an unassisted HP-41C usually falls between 150 and 200 program lines. By adding three modules and keeping the same data space, the program capacity is expanded to about 1200 lines.

An additional wider register is dedicated to alphabetic operations. Up to 24 characters can be accumulated in the alpha register, although only 12 at a time fit in the liquid-crystal display; the extra characters scroll in to the left, marquee-style. The alphabetic capability is not a mere frill. The extent to which it is called upon in the everyday operation of the calculator can be illustrated by considering one of the curious challenges of calculator design.

Mnemonic Functions

The problem is that most scientific calculators have more instructions than they have keys; in the case of the HP-41C, there are more than 130 instructions and only thirty-five keys. A *shift* function doubles the number of distinguishable key sequences, but that still leaves almost half the instruction set without a home on the keyboard. Rather than further increase the number of keys or the number of shifted modes, Hewlett-Packard has adopted a solution familiar in larger systems: all instructions, whether or not they appear on the keyboard, can be executed by spelling out their mnemonic in the display. Programs resident in memory and instructions associated with peripheral devices can be executed in the same way.

Execution of a mnemonic label has the significant advantage of eliminating all dependence of the instruction set on the layout of the keyboard. It also has certain potential drawbacks that the designers of the HP-41C have gone to some lengths to remedy, largely by exploiting the alphabetic display. For example, if the spelling of a mnemonic is forgotten, a complete listing of the instruction set can be called up by the CATALOG function.

Another objection is that repeatedly spelling out a function can be tiresome on a keyboard smaller than the human hand. This burden has been relieved by the radical strategy of allowing all the keys to be redefined by the user. Any instruction (with the exception of a few program-editing pseudoinstructions) and any program can be assigned to any key.

The fluid indeterminacy of the keyboard leads to a further possible complaint: the user may lose track of what function has been assigned to a particular key. Two devices come to the aid of the forgetful. A keyboard overlay slides into place to relabel the keys according to the chosen assignments; if several programs require different key assignments, a separate overlay can be made up for each one. The second aid is more elegant: the current function of any key can be verified merely by pressing the key and holding it down a moment. The mnemonic of the function appears in the display. If the key is released, the function is executed; otherwise, the word "null" appears and the command is canceled.

[A third aid to the use of the HP-41C keyboard is the selection of the user/standard mode. The key redefinitions are valid only when the calculator is in the user mode. To use a key that has been redefined for its original function, the user has only to press the USER key to toggle the calculator back to its standard mode. In the standard mode, the HP-41C behaves as it would before any keys were assigned, thus giving the user the best of both worlds. . . . GW]

Further Features for the Programmer

The versatility of the liquid-crystal display is exploited in several other ways to make the HP-41C friendly and fool-resistant. A row of indicators below the main display provides various indications of mode and status. Error messages can be reasonably explicit: an attempt to divide by 0 elicits "data error," and a number greater than 10^{99} is flagged as "out of range." When a conditional test, such as "X = 0?", is executed from the keyboard, the display answers the question "yes" or "no."

Alphabetic text can also have a valuable role within a program. How it is employed is largely up to the programmer, but two obvious uses are prompting for inputs and labeling outputs.

Even with the best of keyboard technologies, entering a long program is inevitably tedious. A feature of the HP-41C that helps in avoiding needless repetition of effort is a *continuous memory*, which maintains all data and programs even when the calculator is turned off. Key assignments, the settings of flags, and other status information (such as the angular mode) are also preserved. A program that is run frequently can be kept in the calculator. Memory resources are finite, however, and on occasion a program must be cleared to make room for another and later reloaded. It is for such purposes that the magnetic-card reader and the bar-code reader are intended.

Using Cards

The magnetic-card reader, which occupies one port, is a small unit that clips onto the top of the calculator and can be left in place. The cards are the standard 1 by 7 cm magnetic strips (slightly smaller than a stick of chewing gum) that are also employed by the HP-67 and HP-97 and by some Texas Instruments calculators. They are inserted in a slot at the side of the reader and pulled through by a motor for retrieval on the other side. Each card has two tracks and each track holds the contents of 16 registers, which can be either data or programs. A long program requires several cards, and a routine that saves the state of the entire machine sometimes calls for a whole deck of them.

Cues provided by the calculator make operations with the cards almost mindless. When writing a program onto cards, a message in the display indicates how many tracks will be needed; when reading a program, the same message gives the lowest-numbered track that has yet to be read. The cards can be inserted in any sequence, and the information is sorted out internally. A defective card or an unsuccessful pass through the slot generates an appropriate error message.

Cards can be both written and read at the command of a running program. For example, a data card might be requested during an initialization routine, and new values might be written onto the card at the end of a calculation. Or one of several possible subroutines might be appended to a running program once the program had determined which subroutine was needed. Unfortunately, all these procedures still require human intervention for the actual insertion of the card. Thus, the user must attend the machine and feed it by spoonfuls on demand.

An amusing feature of the card reader is its ability to create "private" program cards. When such a card is read back into the calculator, the program appears in the catalog and becomes available for execution, but it cannot be examined, modified, or copied onto another card. Any attempt to do so is blocked by the imperious message "private." The security measures seem to be effective (although I have not worked seriously at penetrating them); how often they will be needed is another question. In the realm of very-small-scale systems, the major worry is theft of hardware, not software.

Software Compatibility

The introduction of a new model computer often raises questions of software compatibility. In this case, Hewlett-Packard has made the new machine compatible with the old software by including a translator routine in the card reader. Magnetic cards written on the HP-67 or HP-97 can be entered into the HP-41C and, with no intervention by the user, will be converted into HP-41C programs. Thus, the machine has access to the large body of software written for the earlier calculators, including more than 3000 programs in a users' library administered by Hewlett-Packard.

An incidental benefit is the addition of more than a dozen instructions peculiar to the HP-67 and HP-97 that become available on the HP-41C whenever the card reader is plugged in, even though most of those instructions have nothing directly to do with card operations. For example, there is a block-memory swap that comes in handy occasionally.

Bar-Code Wand

One drawback of magnetic-card recording is the cost of the medium: roughly fifty cents a card, plus the considerable expense of the card reader itself. There is also the delicacy of the iron-oxide surface, which necessitates careful storage and the maintenance of duplicate copies for backup. A second input device for the HP-41C, the bar-code reader, relies on the most inexpensive of all known storage media, ink on paper. The reader is a hand-held wand similar to a general-purpose one introduced some months ago (the Hewlett-Packard HEDS-3000), but it has an interface and a plug specifically adapted to the HP-41C.

With programs encoded and printed by Hewlett-Packard, the wand works extremely well. A line of code can be scanned in either direction, although multiple lines must be read in sequence. The calculator display prompts for the lowest-numbered line not yet read. Even more helpful is audible confirmation. After each successful pass, the calculator emits a high-pitched beep; a failure results in a lower-pitched tone. The speed and orientation of the wand are not critical, and with practice the success rate becomes quite high.

The wand can also do a few things besides the straightforward loading of programs. Individual instructions can be executed from a "paper keyboard" (which is a table of bar codes, each of which is a single HP-41C instruction); data can be entered directly into designated storage registers; subroutines can be appended and programs merged. One wand function, instead of translating the scanned bar code into HP-41C operation codes, displays the actual binary value represented by the bars.

Printed machine-readable code is an ideal medium for the mass distribution of programs, and Hewlett-Packard will reportedly make all its software for the HP-41C available in this form. Programs from the users' library will also be offered in bar code, presumably at a lower price than programs on magnetic cards. For frequent users of such prepared software, bar code seems to be the medium of choice.

The situation is somewhat different, however, for those whose main interest is in writing their own programs rather than in running other people's. The trouble is that bar code, for now, remains largely a one-way channel of communication.

It is possible to assemble by hand a bar-code representation of a program. The basic materials are adhesive labels, each bearing the code for a single instruction or a single numeric or alphabetic character. [*The "paper keyboard" can also be photocopied, with a program being created by cutting and pasting photocopied bar-code keystrokes. . . .* GW] A long program, however, would require several hundred labels; moreover, they must be scanned as a series of many short strokes. The ability to reproduce the program by photocopying might sometimes compensate for this inconvenience, although the wand owner's manual warns that such copies may not always give acceptable results. (Three copying machines I tried all produced readable images, although the error rate was somewhat higher than with originals.)

For those who have access to a computer system that includes a daisy-wheel printer or a plotter, Hewlett-Packard will supply programs in BASIC or FORTRAN that will generate bar code in the HP-41C format. A far more appealing method would be to produce the bar code on the printer in the HP-41C system; if that could be done, the wand might entirely displace the magnetic-card reader. The HP-41C printer can readily be made to generate patterns that superficially resemble bar codes. In several weeks of experimenting, however, I have been unable to persuade the wand to recognize those patterns reliably. The printer output itself, which is made up of blue or purple characters, is not recognized at all by the wand, and photocopies give erratic results.

Even if the problems of color, contrast, and resolution could be solved, there would remain other impediments. The bar pattern for most of the instruction codes exceeds the capacity of the print buffer; what is more, with no means of summoning up operation codes from program memory, printing the bar-code representation of a program would necessarily entail manual translation. With the system in its present configuration, bar-code output from the printer does not seem to be practical, although it is tantalizingly close.

The mere possibility of obtaining hard copy greatly enhances the utility of the calculator . . .

The Printer

The printer is easily the most engaging component of the HP-41C system. The mere possibility of obtaining hard copy greatly enhances the utility of the calculator, since it relieves the operator of the need to transcribe results as they become available. The printer for the HP-41C does more than that: it will reproduce anything that appears in the display and much else besides.

The print mechanism is a thermal, dot-matrix one; 24-character lines are printed on rolls of heat-sensitive paper about 6 cm wide. There is a standard set of 127 characters, including full uppercase and lowercase alphabets, the ten numerals, a few Greek letters, and miscellaneous other symbols and punctuation marks. All characters can be printed in a standard 5 by 7 matrix or in a double-width format. A few of the standard calculator instructions trigger printing and, in addition, the printer has its own repertoire of about twenty-five instructions.

Programs can be listed in their entirety, or a designated number of lines can be printed out; in either case, the listing shows the same mnemonics that appear in the display. The path followed by the calculator through a program being executed can be traced, providing a record of all instructions and operands; this is a useful facility when the program does not function as expected. The contents of the operand stack can be printed out with a single command; so can the contents of all allocated memory registers, or of a defined block of registers. In addition, assignments of nonstandard functions to the keyboard and the status of all flags can be listed. All of these functions can be executed manually or within a program.

The most commonly invoked print functions are those that print the contents of the X register (roughly equivalent to an accumulator), the alpha register, or a print buffer. The variations offered by these instructions allow the output of a program to take almost any format within the physical capabilities of the printer. The main limitations are the time and space the programmer wishes to dedicate to format commands. It is easy to list a series of variable names, each followed by a colon or an equals sign and a value. Tabulating two or three columns of numbers so they line up vertically on their decimal points demands a somewhat larger investment of program memory and execution time.

The dot-matrix print head is a single vertical row of print elements that sweeps across the paper forming characters as a series of columns (see table 1a). A special set of printer instructions brings this process under program control so that nonstandard characters can be created. Indeed, the printer reproduces any pattern that can be defined by a matrix 7 dots high and no more than 40 dots wide. If the pattern fits in a 7 by 7 box, it can be treated as a special character, stored in a register, and called up as needed. In principle, a complete font could be built up in this way, although its usefulness might be somewhat impaired by the limited capacity of the print buffer: only 6 special characters per line can be printed. A more practical application is the creation of schematic symbols and markers, such as playing-card suits, chess pieces, or the phases of the moon (see table 1b).

Another capability of the printer is the plotting of graphs for any function that can be expressed in the form $y = f(x)$. The graph is drawn under the direction of a program called PRPLOT (print plot), which is committed to read-only memory in the printer. When PRPLOT is executed (see listing 1), it first asks the user to supply certain information that determines the form of the graph, such as the range of x and y. It then calls on a named program, also supplied by the user, that for each given value of x must return a value $f(x)$. The resulting graphs cannot compare to the product of an x,y plotter, but they can be run off quickly and are adequate for gauging the basic form and range of a function. PRPLOT can also be executed from within a program without the prompting for input values, and various parts of it can be called independently.

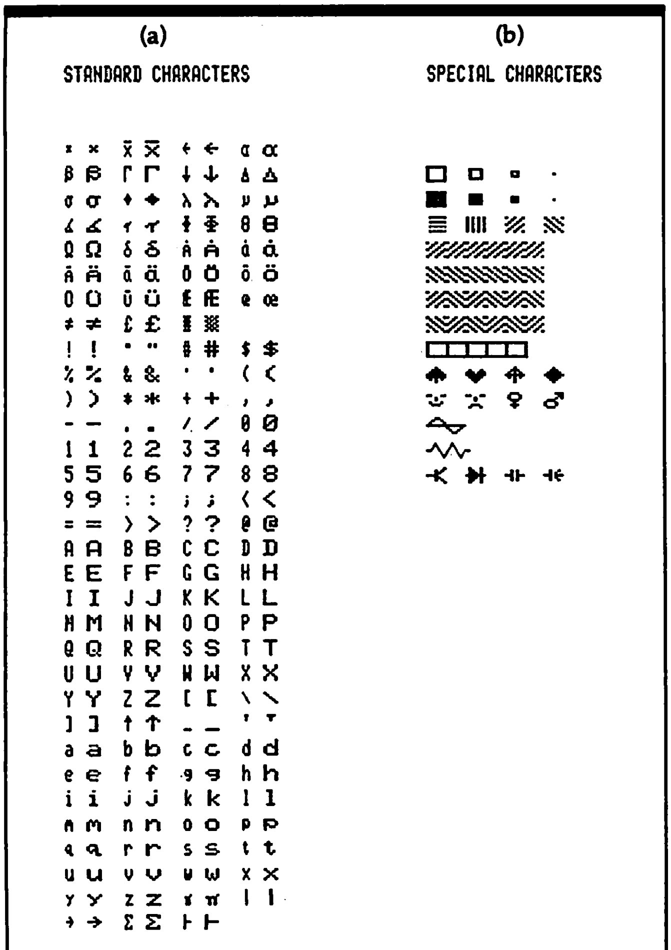


Table 1: *Character set as printed by the HP-41C printer. The standard character set, shown in table 1a, contains 127 letters, numbers, and other symbols. About sixty of them, including the full uppercase alphabet, can also be represented in a somewhat different form in the display of the HP-41C itself. Each character can be printed in a standard 5 by 7 dot matrix or in a double-width format. Special characters (table 1b) can also be created by specifying the pattern of dots in each column of the character.*

Programming with Labels

An organizing principle of programs for the HP-41C is that all references and transfers of control are made by means of *labels*. The name given to a program constitutes a global label, one that can be accessed from any point in program memory. By invoking the name, a program can be called as a subroutine and can even call itself, although there are limits to such recursion.

Labels within programs are generally local, so that the same labels can be repeated in different programs without interference. Subroutine calls and branches can be made only to a label; there is no absolute addressing by line number. As a result, all programs and procedures within programs can be relocated at will. Lines can also be freely inserted or deleted without adjusting references elsewhere.

Instructions that require an address or a numerical argument can be given it either directly or indirectly. The addressing modes are uniform for all memory operations, subroutine calls, branching, loop control, the setting, clearing, and testing of flags, and even such functions as setting the display format and determining the pitch of the beeper. A subroutine is called by the XEQ (execute) function, which must be followed by a local label or the name of a program.

If the instruction is an indirect one (XEQ IND), the 2-digit number that follows is interpreted as the register where the subroutine name or label will be found. Any register, including those of the stack, can hold the indirect address. Subroutines can be nested six levels deep before the return address of the highest-level routine is lost.

Conditional tests of numerical data include various combinations of "less than," "greater than," "equal to," and "not equal to"; alphabetic strings can also be compared, but only for equivalence. All the tests have the same format, in which a false result causes the instruction following the test to be skipped. Tests of flags (set or clear) employ the same scheme. The complement of fifty-six flags seems particularly generous. Eleven flags are completely unencumbered for use in programs; the rest control the status of the HP-41C and its peripherals, thereby affording the calculator a valuable amount of self-knowledge.

Loops

The control of loops in HP-41C programs is facilitated by two instructions that store all the needed information in a single register. The instructions, ISG (increment, skip if greater) and DSE (decrement, skip if equal), refer directly or indirectly to a register holding a number of the form *nnnnn.tttcc*. Here *nnnnn* is the number to be tested, *ttt* is the value against which it is tested, and *cc* is the amount by which *nnnnn* is incremented, or decremented. The compacted form is a convenience, although I find it odd that the incremented number has a range of up to 99,999, whereas a jump must take place whenever it exceeds 999.

Other Programming Features

The HP-41C cannot realistically be said to support structured programming, not as I understand the term. The rule that all procedures should have a single entry point and a single exit, which is one of the precepts of structured programming, cannot be observed without extreme awkwardness. On the other hand, the program-control structures of the HP-41C strongly encourage the composition of modular programs, where each procedure is a self-contained unit, small enough to be fully understood and capable of being tested independently. In a program longer than a few hundred lines, some such technique for imposing order is obligatory.

In the end, the capabilities of the HP-41C can be exhibited best by real programs and their output. A few short utility routines and a longer program, called CHART, are given in listings 2 and 3. CHART, which incidentally shows off to good advantage the versatility of the printer, produces a bar graph, a form of display that is more appropriate for some kinds of data than the line graphs of PRPLOT.

The main program in CHART (listing 2), which is confined to the first 20 lines, is little more than a list of XEQ statements. It first prompts the user for needed information, then does some preliminary calculations and prints a header that will identify the graph. An external program (see listing 4) is then called once for each bar; it is expected to return a value defining the length of the bar and a label of not more than 4 characters.

It is worth noting that the actual calculation of the bar length is a trivial operation. The bulk of the program is taken up with input and output routines, which are intended to minimize the burden on the user's memory and faculties of interpretation. A bar graph generated by the CHART program is shown for data on the distribution of digits obtained from the RDM LN pseudorandom-number generator; see listing 5.

Next Generations

What more can one ask for in a programmable calculator? Quite a lot; there is much to look forward to in the next generation. More memory is always near the top of such a wish list. One way of supplying it, which might be compatible with the present mainframe, would be in a double-density memory module. The entire address space could then be utilized without filling all the ports.

The very existence of ports inspires thoughts of other

Text continued on page 136

Listing 1: *Graph of the function (sin x)/x was drawn by PRPLOT, a program that resides in read-only memory in the HP-41C printer. The function itself is defined by a separate program (at bottom), which evaluates the expression each time it is supplied with a value of x and called PRPLOT.*

```
       PLOT OF SIN/X
     X <UNITS= 1.> ↓
     Y <UNITS= E-2.> →
      -0.50          2.00
         0.00
      |---|--------------|
-360.     x
-331.    x¦
-302.   x ¦
-274.  x  ¦
-245.  x  ¦
-216.   x ¦
-187.    x¦
-158.     ¦ x
-130.     ¦   x
-101.     ¦      x
 -72.     ¦         x
 -43.     ¦           x
 -14.     ¦            x
  14.     ¦            x
  43.     ¦           x
  72.     ¦         x
 101.     ¦      x
 130.     ¦   x
 158.     ¦ x
 187.    x¦
 216.   x ¦                        01♦LBL "SIN/X"
 245.  x  ¦                        02 RCL X
 274.  x  ¦                        03 SIN
 302.   x ¦                        04 X<>Y
 331.    x¦                        05 /
 360.     x                        06 END
```

Listing 2: *A bar-graph program. CHART, the HP-41C program for generating bar graphs, is written as a series of modules. The first of these prompts the user to supply certain initial information that will determine the form of the graph. An alternative entry point, CHARTP, is intended for occasions when the bar-graph routine is called from another program; this entry point bypasses the prompting. For each bar drawn, CHART calls on a user-supplied program, which must return two items, the value to be plotted in the X register and a label for the bar no more than 4 characters long in the alpha register. The bar is actually formed in subroutine 08 out of a standard character and additional print columns for fine adjustment of the length.*

```
01♦LBL "CHART"
02♦LBL a
03 XEQ 00
04♦LBL "CHARTP"
05 XEQ 01
06 XEQ 02
07 XEQ "BAR"
```

Initialization; can be executed from the keyboard by pressing "A."

```
08♦LBL A
09 XEQ 03
10 XEQ 04
11 XEQ 05
12♦LBL 30
13 XEQ 07
14 RCL 18
15 INT
16 XEQ IND 11
17 XEQ 08
18 ISG 18
19 GTO 30
20 XEQ 07
21 GTO 50
```

Main calculation and printing of bars. Calls a user program whose name is stored in register 11.

```
22♦LBL 00
23 CF 23
24 "PGM NAME?"
25 AON
26 PROMPT
27 FS? 23
28 ASTO 11
29 AOFF
30 CF 22
31 "NO. OF BARS?"
32 PROMPT
33 FS?C 22
34 STO 12
35 "Y MIN?"
36 PROMPT
37 FS?C 22
38 STO 13
39 "Y MAX?"
40 PROMPT
41 FS?C 22
42 STO 14
43 "AXIS?"
44 PROMPT
45 FS? 22
46 STO 15
47 RTN
```

Subroutine that prompts for inputs. In each case the prompting message appears in the display but is not printed. If no value is input following the prompt, the program assumes the value supplied on the previous run is still valid.

```
48♦LBL 01
49 RCL 12
50 1
51 -
52 1 E3
53 /
54 STO 18
```

Set up register for looped calls to user program.

```
55 137
56 RCL 14
57 RCL 13
58 -
59 /
60 STO 16
```

Calculate coefficient relating *Y*-axis scale to graph width of 137 columns.

```
61 RCL 15
62 XEQ 10
63 STO 17
64 5
65 X<>Y
66 X<=Y?
67 ST- 17
68 132
69 X<>Y
70 X>Y?
71 ST- 17
72 RTN
```

Calculate absolute position of axis; if beyond the range of the graph, axis is suppressed.

```
73♦LBL 02
74 ADV
75 ADV
76 "P"
77 ACA
78 SF 13
79 "LOT OF "
80 ACA
81 CF 13
82 SF 12
83 RCL 11
84 ACX
85 CF 12
86 PRBUF
87 RTN
```

Print identifying header: "Plot of 'PGM NAME' "

```
88♦LBL 03
89 SF 12
90 "X"
91 ACA
92 7
93 ACCHR
94 29
95 SKPCOL
96 "Y"
97 ACA
98 125
99 ACCHR
100 CF 12
101 PRBUF
102 RTN
```

Print labels for *X* and *Y* axes.

```
103♦LBL 04
104 RCL 13
105 ACX
106 XEQ 11
107 STO 10
108 RCL 14
109 XEQ 11
110 ST+ 10
111.144
112 RCL 10
113 -
114 SKPCOL
115 RCL 14
116 ACX
117 ADV
118 RTN
```

Labels extrema of *Y* axis.

```
119♦LBL 05
120 0
121 STO 10
122 RCL 17
123 X=0?
124 RTN
125 119
126 ACCOL
127 RDN
128 RCL 15
129 XEQ 11
130 ST+ 10
131 2
132 /
133 -
134 5
135 X>Y?
136 GTO 52
137 RDN
138 132
139 RCL 10
140 -
141 X<Y?
142 GTO 52
143 RDN
144♦LBL 52
145 INT
146 SKPCOL
147 ST+ 10
148 RCL 15
149 ACX
150 XEQ 12
151 RTN
```

Labels axis within graph, if it has not been suppressed.

```
152♦LBL 07
153 119
154 ACCOL
155 0
156 STO 10
157 XEQ 17
158 XEQ 12
159 RTN
```

Accumulates markers for the extrema points and the axis in spaces between bars.

```
160♦LBL 08
161 ACA
162 3
163 SKPCOL
164 RDN
165 XEQ 10
166 X<=0?
167 GTO 07
168 127
169 ACCOL
170 RDN
171 136
```

Master subroutine for accumulating and printing a bar. Checks if the length is zero; if so, executes LBL 07. Checks if the length is

Listing 2 continued on page 134

Listing 2 continued:

```
172 X<=Y?
173 GTO 09
174 RDN
175 STO 10
176 XEQ 15
177 RDN
178 XEQ 16
179 127
180 ACCOL
181 XEQ 17
182 XEQ 12
183 RTN
```

greater than the maximum; if so, executes LBL 09. Otherwise, the bar is built up by LBL 15 and LBL 16.

```
184♦LBL 09
185 STO 10
186 XEQ 15
187 RDN
188 XEQ 16
189 127
190 ACCOL
191 ADV
192 RTN
```

Special routine for a bar that must fill the entire width of the graph.

```
193♦LBL 15
194 7
195 X>Y?
196 RTN
197 X=Y?
198 RTN
199 31
200 ACCHR
201 RDN
202 -
203 GTO 15
```

Accumulates the maximum integer number of gray-tone characters (standard character 31) that will fit in the bar.

```
204♦LBL 16
205 1
206 X>Y?
207 RTN
208 X=Y?
209 RTN
210 42
211 ACCOL
212 RDN
213 -
214 1
215 X>Y?
216 RTN
217 X=Y?
218 RTN
219 85
220 ACCOL
221 RDN
222 -
223 GTO 16
```

Finishes a bar by accumulating individual columns until actual length equals specified length.

```
224♦LBL 17
225 RCL 10
226 1
227 +
228 RCL 17
229 X≠0?
230 X<=Y?
231 RTN
232 STO 10
233 X<>Y
234 -
235 SKPCOL
236 119
237 ACCOL
238 RTN
```

Inserts space from end of bar to maximum *Y* then adds a marker for maximum *Y*

```
239♦LBL 10
240 RCL 13
241 -
242 RCL 16
243 *
244 FIX 0
245 RND
246 FIX 2
247 RTN
```

Calculates the length of the bar.

```
248♦LBL 11
249 ABS
250 SF 25
251 LOG
252 CF 25
253 INT
254 5
255 +
256 .7
257 *
258 RTN
```

Calculates width of a number (eg: axis or extrema labels) in number of columns.

```
259♦LBL 12
260 135
261 RCL 10
262 -
263 SKPCOL
264 119
265 ACCOL
266 ADV
267 RTN
```

Adds space to fill out a line, other than a line with a bar, then prints a *Y*-maximum marker.

```
268♦LBL 50
269 ADV
270 ADV
271 BEEP
272 END
```

Beeps to mark finish.

Circle 83 on inquiry card.

Circle 84 on inquiry card. →

Text continued from page 130:

peripheral devices. A cassette recorder could provide mass storage and would make feasible operations on large blocks of data. An x,y plotter could be driven very efficiently by the HP-41C, albeit at a leisurely pace. With a fairly simple interface, it should be possible to connect the calculator to a computer system. The likelihood that any of these products will ever be forthcoming is unknown. It is probably too much to ask that Hewlett-Packard release technical information on the signals available at the ports so that others could develop plug-compatible devices. Some intrepid experimenter with a logic probe may do it anyway.

There are a few gaps in the instruction set of the HP-41C that should not be perpetuated in future calculators. For example, there are tests for $x < y$, for $x \le y$ and for $x > y$, but there is no test for $x \ge y$. Of course, any desired logic function can be fabricated out of the existing instructions, but the programmer should not have to go to that trouble and should not have to remember which of the tests is the missing one.

The most fundamental defect in the architecture of the HP-41C, inadequate numerical precision, is a serious flaw indeed. Numbers are represented, both internally and in the display, with 10 decimal digits; there are no guard digits. As a result, inaccuracies are quite often introduced into the least-significant digit. For example, $(\sqrt{2})^2$ is evaluated by the calculator as 1.999999999. For operations on some data, the corruption goes still deeper and 2 or 3 digits become suspect. There is something absurd about the world's fanciest calculator not being able to give results accurate to more than seven or eight decimal places.

Actually, a subsidiary problem is more serious than that. Conditional tests on data are carried out on the full 10-digit representation. Consequently, a test that effectively asks "Is $(\sqrt{2})^2$ equal to 2?" will give a false result, which can lead a program far astray.

Listing 3: *Utility routines for the HP-41C. These two routines are the kinds of programs that can remain in memory as resources to be drawn on by other programs, somewhat like macro instructions in an assembly language. BAR simply prints a heavy bar across the width of the paper to separate different kinds of information. TAB handles the spacing of numbers to be printed in vertical columns. It must be supplied with the number to be printed (in the X register) and the number of character spaces to be measured from the present position in the line of print to the decimal point. TAB was employed in formatting the random-number data in listing 2.*

```
01♦LBL "BAR"
02 ADV
03 .023
04 31
05♦LBL 01
06 ACCHR
07 ISG Y
08 GTO 01
09 PRBUF
10 ADV
11 ADV
12 END
```

```
01♦LBL "TAB"
02 ABS
03 SF 25
04 LOG
05 CF 25
06 X<=0?
07 CLX
08 INT
09 1
10 +
11 RCL X
12 3.1
13 /
14 INT
15 +
16 CHS
17 +
18 SKPCHR
19 END
```

Listing 4: *Random-number routines for the HP-41C. These two random-number generators, standard coding exercises for programmable calculators, both calculate a pseudorandom real value, then select a single pseudorandom digit for return to the calling program. RDM LC employs the standard linear-congruential method, which has virtues and failings that are well understood. In this example, R_{n+1} is equal to $[24{,}298R_n + 99{,}991]_{\mathrm{mod}\ 199{,}017}$.*

RDM LN is an algorithm the author stumbled upon but has not seen in the literature. R_{n+1} is defined as $1/\ln R_n$. Experimental runs of up to several thousand iterations have given good results, but the behavior of the algorithm is not understood. A sample test is shown in listing 5.

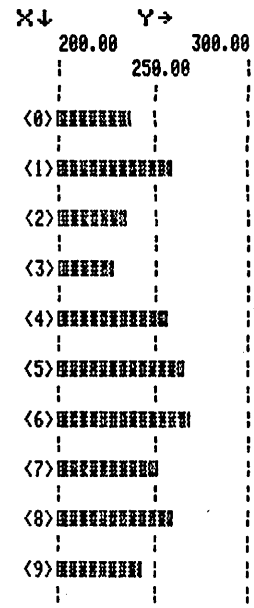
```
01♦LBL "RDM LN"
02 RCL 20
03 ABS
04 LN
05 1/X
06 STO 20
07 1 E3
08 *
09 FRC
10 10
11 *
12 INT
13 ABS
14 END
```

```
01♦LBL "RDM LC"
02 RCL 20
03 24298
04 *
05 99991
06 +
07 199017
08 MOD
09 STO 20
10 1 E3
11 /
12 FRC
13 10
14 *
15 INT
16 END
```

Listing 5: *Bar-graph results of the CHART program, given in listing 2. The graph represents pictorially the distribution of the 10 digits in a sample of 2500 pseudorandom numbers. The numbers were generated by another program, RDM LN (shown in listing 4), with the bookkeeping done by a third program.*

```
Test of "RDM  LN"

Number of trials 2500

Seed =  1.234567890

IIIIIIIIIIIIIIIIIIIIIIII


 DIGIT  COUNTS

MEAN =          250.00

 <0> =          237.
 <1> =          259.
 <2> =          234.
 <3> =          228.
 <4> =          256.
 <5> =          265.
 <6> =          268.
 <7> =          251.
 <8> =          259.
 <9> =          243.


RUNS OF 2:      211.
RUNS OF 3:       14.
RUNS OF 4:        3.
RUNS OF 5:        0.


  STATISTICS

CHI SQUARED =     6.8240
HIGH/LOW =        1.0593
ODD/EVEN =        0.9936

IIIIIIIIIIIIIIIIIIIIIIII
```

Plot of "RANDOM"



It is easy to imagine that some programmable calculator evolved from the HP-41C would have instructions much like those of a higher-level language. Having introduced named programs, the next obvious step is named variables, which would relieve the programmer of much tedious worry over memory allocation. Let the machine keep track of where the numbers are; it does so better than people can. The existing conditional tests, which act directly on particular registers, might be recast as a more general *if . . . then . . . else* construction, employing the named variables. Also, *do . . . while* and *repeat . . . until* commands would be a welcome addition; indeed, the loop-control instructions of the HP-41C already come close.

One essential capability must be added to the calculator before such higher-level commands can be made available. A higher-level language is a program whose output is another program, and so it is necessary that instructions be allowed to operate not only on data but also on other instructions. In this context, it seems significant that the inability of a calculator to alter its own instructions is what most clearly distinguishes calculators from computers. ■