HEWLETT PACKARD

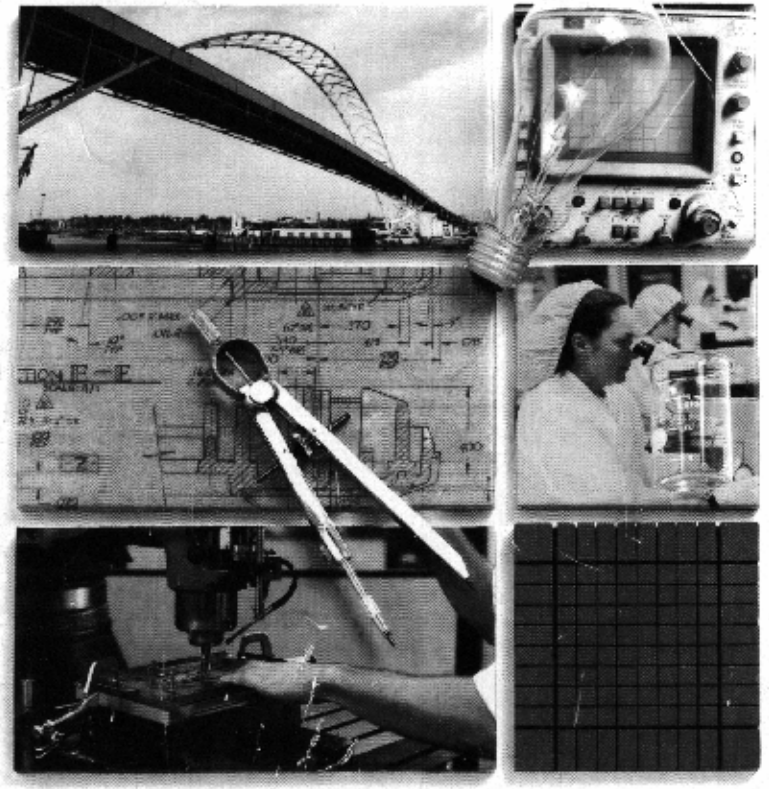
Portable Computer Division
1000 N.E. Circle Blvd., Corvallis, OR 97330, U.S.A.

00011-90001 Rev. G English Printed in Canada 11/85

HEWLETT-PACKARD

HP-11C

OWNER'S HANDBOOK AND PROBLEM-SOLVING GUIDE



Notice

Hewlett-Packard Company makes no express or implied warranty with regard to the keystroke procedures and program material offered or their merchantability or their fitness for any particular purpose. The keystroke procedures and program material are made available solely on an "as is" basis, and the entire risk as to their quality and performance is with the user. Should the keystroke procedures or program material prove defective, the user (and not Hewlett-Packard Company nor any other party) shall bear the entire cost of all necessary correction and all incidental or consequential damages. Hewlett-Packard Company shall not be liable for any incidental or consequential damages in connection with or arising out of the furnishing, use, or performance of the keystroke procedures or program material.

HEWLETT PACKARD

HP-11C

Owner's Handbook and Problem Solving Guide

November 1985

00011-90001 Rev. G

Printed in Canada 11/85

Introduction

Congratulations! Your selection of an HP-11C calculator with Continuous Memory demonstrates your interest in quality, capability, and ease of use. This handbook describes the calculator's many features and can help you to quickly learn how to use the features you are not already familiar with.

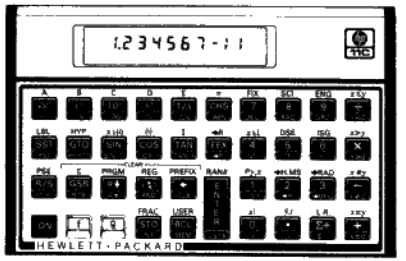
Fundamentals, Programming, Applications Programs. Your HP-11C handbook is divided into three main parts. Parts I and II cover the use of keyboard and programming features you may be familiar with if you have used other HP programmable calculators. Part III provides you with a variety of applications programs, plus some additional information on fundamental programming practices. However, before you begin reading in any of the three main parts, we suggest that you gain some experience using your HP-11C by working through the introductory material entitled Your HP-11C, A Problem Solver, on page 9.

Programming Sections in Brief. At or near the beginning of each section in part II (sections five through nine) is a brief discussion of the operating characteristics covered in that section. This information may be all you need to read in the section if you already know how to create, run, and edit programs on another HP calculator. If you are new to HP calculators, you'll want to also read the remaining material in each section for a more tutorial discussion and/or examples that highlight selected topics.

Additional Applications. Also available from many authorized Hewlett-Packard dealers is the HP-11C Solutions Handbook. This accessory handbook contains a comprehensive collection of HP-11C programs covering applications in mathematics, statistics, electrical engineering, chemistry, business, and games.

Contents

The HP-11C Keyboard and Continuous Memory



AUTOMATIC MEMORY STACK

T →
Z →
Y →
X → Displayed

LAST X

PROGRAM MEMORY

Permanent	Shared
000–	064–
001–	065–
002–	066–
⋮	⋮
061–	201–
062–	202–
063–	203–

STORAGE REGISTERS

Permanent	Shared	
R_I	R_0	$R_{.0}$
	R_1	$R_{.1}$
	R_2	$R_{.2}$
	R_3	$R_{.3}$
	R_4	$R_{.4}$
	R_5	$R_{.5}$
	R_6	$R_{.6}$
	R_7	$R_{.7}$
	R_8	$R_{.8}$
	R_9	$R_{.9}$

The basic program memory-storage register allocation is 63 lines of programming and 20 data storage registers, plus the Index register (R_I). The calculator automatically converts one data storage register into seven lines of program memory, one register at a time, as you need them. Conversion begins with $R_{.9}$ and ends with R_0.

Your HP-11C: A Problem Solver

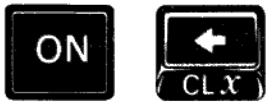
Your HP-11C Programmable Scientific Calculator is a powerful problem solver you can carry with you almost anywhere to handle problems ranging from the simple to the complex, and to remember important data. The HP-11C is so easy to program and use that it requires no prior programming experience or knowledge of programming languages.

The HP-11C helps you to conserve power by automatically shutting itself off if it is left on and inactive for more than 8 to 17 minutes. But don't worry about losing data—any information you have keyed into your HP-11C is saved by Continuous Memory.

We're different! Your Hewlett-Packard calculator uses a unique operating logic, represented by the [ENTER] key, that differs from the logic in most other calculators. The power in HP calculator logic becomes obvious through use. Later we will cover the details of [ENTER] and your HP-11C's logic, but right now let's get acquainted with [ENTER] by seeing how easy it is to perform calculations with your HP-11C.

Answers appear immediately after you press a numerical function key. For example, let's look at the arithmetic functions. The [+] key adds the last entry to whatever is already in the machine, and the [−] key subtracts the last entry; the [×] key multiplies what's in the machine by the last entry, and the [÷] key divides by the last entry. First we have to get the numbers into the machine. To do this, key in the first number, press [ENTER] to separate the first number from the second, then key in the second number and press [+], [−], [×], or [÷].

To get the feel of your new calculator turn it on by pressing the [ON] key.* If any nonzero digits appear, press [←] to clear the display to **0.0000**. If four zeroes are not displayed to the right of the decimal point, press [f] [FIX] 4 so your display will match those in the following problems. (All displays illustrated in this handbook are set to the [FIX] 4 display setting unless otherwise specified.)



0.0000

Note: An asterisk (*) flashing in the lower-left corner of the display when the calculator is turned on signifies that the available battery power is nearly exhausted. To install new batteries, refer to appendix D.

Manual Solutions

It is not necessary to clear the calculator between problems. But if you make a digit entry mistake, press [←] and key in the correct digit.

To Solve	Keystrokes	Display
$9+6=15$	9 [ENTER] 6 [+]	**15.0000**
$9-6=3$	9 [ENTER] 6 [−]	**3.0000**
$9\times6=54$	9 [ENTER] 6 [×]	**54.0000**
$9\div6=1.5$	9 [ENTER] 6 [÷]	**1.5000**

Notice that in the four examples:

- Both numbers are in the calculator before you press [+], [−], [×], or [÷].
- [ENTER] is used only to separate two numbers that are keyed in one after the other.
- Pressing a numerical function key, in this case [+], [−], [×], or [÷], causes the function to execute immediately and the result to be displayed.

* Note that the [ON] key is lower than the other keys to help prevent its being pressed inadvertantly.

To see the close relationship between manual and programmed problem solving, let's first calculate the solution to a problem manually, that is, from the keyboard. Then we'll use a program to calculate the solution to the same problem and others like it.

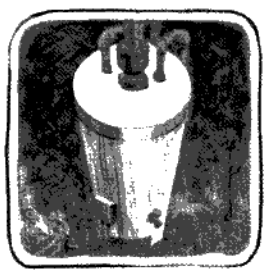
Most conventional home water heaters are cylindrical in shape. You can easily calculate the heat loss from such a tank using the formula $q = h \times A \times T$, where:

q is the heat loss from the water heater (Btu per hour).

h is the heat-transfer coefficient.

A is the total surface area of the cylinder.

T is the temperature difference between the cylinder and the surrounding air.



Example: Assume you have a 52-gallon cylindrical water heater and you wish to determine how much energy is being lost because of poor insulation. In initial measurements, you found an average temperature difference between the heater surface and surrounding air of 15 degrees Fahrenheit. The surface area of the tank is 30 square feet and the heat transfer coefficient is approximately 0.47. To calculate the heat loss of the water heater, simply press the following keys in order.

Keystrokes	Display	
15 [ENTER]	**15.0000**	Input temperature difference (T) and area of water heater (A).
30	**30**	
[×]	**450.0000**	Calculates $A \times T$.
.47	**0.47**	Heat-transfer coefficient (h).
[×]	**211.5000**	Heat loss in Btu per hour ($h \times (AT)$).
[←]	**0.0000**	Clears display.

Programmed Solutions

The heat loss for the water heater in the preceding example was calculated for a 15-degree temperature difference. But suppose you want to calculate the heat loss for several temperature differences? You could perform each heat loss calculation manually. However, an easier and faster method is to write a program that will calculate the heat loss for any temperature difference.

Writing the Program. The program is the same series of keystrokes you executed to solve the problem manually. Two additional instructions, a label and a return, are used to define the beginning and end of the program.

Loading the Program. To load the instructions of the program into the HP-11C press the following keys in order. The calculator records (remembers) the instructions as you key them in. (The display gives you information you will find useful later, but which you can ignore for now.)

Keystrokes	Display		
g P/R	000-		Places HP-11C in Program mode. (Program annunciator appears.)
f CLEAR PRGM	000-		Clears program memory.
f LBL A	001-42,21,11		Label "A" defines the beginning of the program.
3	002-	3	The same keys you pressed to solve the problem manually.
0	003-	0	
×	004-	20	
.	005-	48	
4	006-	4	
7	007-	7	
×	008-	20	
g RTN	009-	43 32	"Return" defines the end of the program.
g P/R	0.0000		Places HP-11C in Run mode. (Program annunciator is cleared.)

Running the Program. Press the following keys to run the program.

Keystrokes	Display	
15	15	The first temperature difference.
f A	211.5000	The Btu heat loss you calculated earlier by hand.
18 f A	253.8000	The Btu heat loss for a new temperature difference.

With the program you have loaded, you can now quickly calculate the Btu heat loss for many temperature differences. Simply key in the desired difference and press f A. For example, complete the table at the right.

Temp. Diff.	Btu Heat Loss
10	?
12	?
14	?
16	?
18	?
20	?

The answers you should see are 141.0000, 169.2000, 197.4000, 225.6000, 253.8000, and 282.0000.

Programming is that easy! The calculator remembers a series of keystrokes and then executes them whenever you wish. Now that you have had some experience in using your HP-11C, let's take a look at some of the calculator's important operating details.

Part I

HP-11C Fundamentals

Section 1

Getting Started

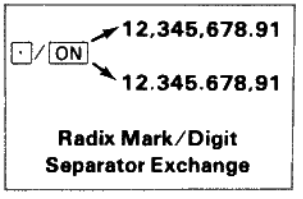
Power On and Off

The [ON] key turns the HP-11C on and off. And, to conserve power, the HP-11C automatically turns itself off (time-out) after 8 to 17 minutes of inactivity.

Display

Radix Mark and Digit Separator

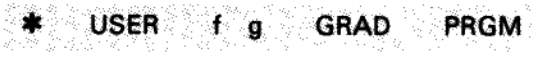
A radix mark is the divider between the integer and fractional portions of a number. A digit separator distinguishes the groups of digits in a large number. In some countries the radix is a decimal point and the digit separator is a comma, while in other countries, the reverse is true. To change the radix/digit separator convention on your HP-11C, turn off the calculator, then hold down the [.] key, turn the calculator back on, and release the [.] key ([.]/[ON]).



Radix Mark/Digit Separator Exchange

Annunciators

Your HP-11C's display contains six annunciators that tell you the status of the calculator during certain operations. The annunciators are described, with the operations they refer to, in the appropriate sections of this handbook.



Display Annunciator Set

Negative Numbers

To make a displayed number negative—either one that has just been keyed in or one that has resulted from a calculation— simply press [CHS] *(change sign)*. When the display shows a negative number—that is, the number is preceded by a minus sign—pressing [CHS] removes the minus sign from the display, making the number positive.

Display Clearing: [CLx] and [←]

The HP-11C has two types of display clearing operations, [CLx] *(clear X)* and [←] *(back arrow)*. When pressed in Run mode, [g] [CLx] clears any displayed numbers to zero. When pressed in Program mode, [g] [CLx] is stored in the calculator as a program instruction. [←] is a nonprogrammable function that enables you to clear the display in either Program or Run mode, as follows:

1. In Run mode:
   A. Pressing [←] *after* executing almost any function clears all digits in the display to zero.* (Executing almost any HP-11C function terminates digit entry—that is, tells the calculator the number in the display is complete—and causes [←] to act on the complete number.)

Keystrokes	Display	
12345	12,345	
[$\sqrt{x}$]	111.1081	
[←]	0.0000	Pressing [←] after executing a function clears all digits in the display to zero.

* Unseen trailing digits of the number in the display may be held internally and will also be cleared.

B. After keying in a new number, if you press [←] *before* executing a function (that is, before terminating digit entry) the last digit you keyed in is deleted. After you delete one or more digits, you can, if you want, key in new digits to replace them.

Keystrokes	Display	
12345	**12,345**	
[←]	**1,234**	When digit entry has not
[←]	**123**	been terminated, [←] acts
9	**1,239**	on each digit separately.

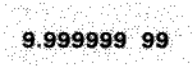
2. In Program mode, pressing [←] deletes the entire program instruction currently in the display.

Running

While a program is running, or during execution of [$P_{y,x}$] *(permutation)* or [$C_{y,x}$] *(combination)*, **running** flashes in the display.

Overflow and Underflow

Overflow. When the result of a calculation in the displayed X-register is a number with a magnitude greater than $9.999999999 \times 10^{99}$, all 9's are displayed with the appropriate sign. When overflow occurs in a running program, execution halts and the overflow display appears.



Overflow Display

Underflow. If the result of a calculation is a number with a magnitude less than $1.000000000 \times 10^{-99}$, zero will be substituted for that number. Underflow will not halt the execution of a running program.

Error Messages

If you attempt a calculation using an improper parameter, such as attempting to find the square root of a negative number, an error message will appear in the display.

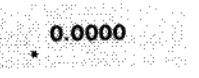
Keystrokes	Display
4 [CHS]	**-4**
[$\sqrt{x}$]	**Error 0**
[←]	**-4.0000**

For a complete listing of error messages and their causes, refer to appendix A, Error Conditions. A summary of error messages is printed on the calculator's back label.

> To clear any error message, press [←] (or any other key), then resume normal calculator operation.

Low Battery Indicator

Whenever a flashing asterisk, which indicates low power, appears in the lower left-hand side of the display, refer to Installing New Batteries, page 231.



Memory

Continuous Memory

The Continuous Memory feature in your HP-11C maintains the following even when the calculator is turned off:

- All numeric data stored in the calculator.
- All programs stored in the calculator.
- Display mode and setting.
- Flag settings.
- Position of the calculator in program memory.
- Any pending subroutine returns.
- Trig mode (Degree, Radian, or Grad).

When the HP-11C is turned on, it always "wakes up" in Run mode (**PRGM** annunciator cleared), even if it was in Program mode (**PRGM** annunciator displayed) when last turned off.

Continuous Memory is preserved for a short time when the batteries are removed, allowing you to replace a set of low batteries without losing any data or programs you want preserved in the calculator.

Resetting Memory

If at any time you want to reset (entirely clear) the HP-11C's Continuous Memory, do the following:

1. Turn the HP-11C off.
2. Press and hold the [ON] key.
3. Press and hold the [−] key.
4. Release the [ON] key and then release the [−] key.

Resetting Continuous Memory

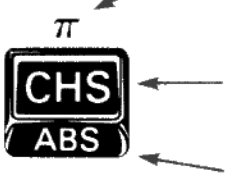
When you perform the memory reset operation the error message shown to the right is displayed. Press [←] (or any other key) to clear the message.

Note: Continuous Memory can be inadvertently reset if the calculator is dropped or otherwise traumatized, or if power is interrupted.

Keyboard Operation

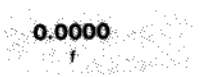
Primary and Alternate Functions

Most keys on your HP-11C perform one primary and two alternate functions. The primary function of any key is indicated by the character on the horizontal face of the key. The two alternate functions are indicated by the characters above and on the slanted face of the key.



- To select the alternate function printed in gold above the key, first press the gold prefix key [f], then press the function key; for example: [f] [π].
- To select the primary function on the face of the key, press only that key; for example: [CHS].
- To select the alternate function printed in blue on the slanted face of the key, first press the blue prefix key [g], then press the function key; for example: [g] [ABS].

Notice that when you press the [f] or [g] prefix keys, the **f** or **g** annunciator appears and remains in the display until a function key is pressed to complete the sequence.



Clearing Prefixes

If you make a mistake while keying in a prefix for a function, press [f] CLEAR [PREFIX] to cancel the error. (CLEAR [PREFIX] also cancels the [STO], [RCL], [GTO], [GSB], [HYP], and [HYP^{-1}] keys.) Since the [PREFIX] key is also used to display the mantissa of a displayed number, all ten digits of the number in the display will appear for a moment after [PREFIX] is pressed (in Run mode only).

One-Number Functions

A one-number function is any numerical function that performs an operation using only one number. To use any one-number function:

1. Key in the number (if it is not already in the display).
2. Press the function key(s).

Keystrokes	Display
45	45
[g] [LOG]	1.6532

Two-Number Functions

A two-number function must have two numbers present in the calculator before executing the function. [+], [−], [×], and [÷] are examples of two-number functions.

The [ENTER] Key. If one of the numbers you need for a two-number function is already in the calculator as the result of a previous function, you do not need to use the [ENTER] key. However, when you must key in two numbers before performing a function, use the [ENTER] key to separate the two numbers.

To place two numbers into the calculator and perform a two-number function such as $2 \div 3$:

1. Key in the first number.
2. Press [ENTER] to separate the first number from the second.
3. Key in the second number.
4. Press the function key(s).

Keystrokes	Display
2	2
[ENTER]	2.0000
3	3
[÷]	0.6667

Order of Entry. As you know from basic arithmetic, reversing the order of the numbers in addition and multiplication examples will not affect the answer. But for subtraction or division, the number you are subtracting or dividing by is always the *second* number keyed in.

To perform	Keystrokes	Display
$10 - 3$	10 [ENTER] 3 [−]	7.0000
$3 - 10$	3 [ENTER] 10 [−]	−7.0000
$10 \div 3$	10 [ENTER] 3 [÷]	3.3333
$3 \div 10$	3 [ENTER] 10 [÷]	0.3000

When working with your HP-11C's other two-number functions (such as [y^x]), remember that the number designated by x on the key is always the last number to be keyed in. For example, to calculate the value of 2 raised to the power of three (2^3), key in 2, press [ENTER], key in the exponent, 3, then press [y^x].

Keystrokes	Display	
2 [ENTER]	2.0000	
3	3	3 is the x-value.
[y^x]	8.0000	2 (y) raised to the third (x) power.

Now try these problems. Notice that you have to press [ENTER] to separate numbers only when they are being keyed in one immediately after the other. A previously calculated result (intermediate result) will be automatically separated from a new number you key in.

To solve $(2 + 4) \div 8$:

Keystrokes	Display	
2 [ENTER]	2.0000	
4	4	
[+]	6.0000	$(2 + 4)$
8	8	
[÷]	0.7500	$(2 + 4) \div 8$

To solve $(9 + 17 - 4 + 23) \div 4$:

Keystrokes	Display	
9 [ENTER]	9.0000	
17 [+]	26.0000	$(9 + 17)$
4 [−]	22.0000	$(9 + 17 - 4)$
23 [+]	45.0000	$(9 + 17 - 4 + 23)$
4 [÷]	11.2500	$(9 + 17 - 4 + 23) \div 4$

Even more complicated problems are solved in the same simple manner—using automatic storage of intermediate results.

Example: Solve $(6 + 7) \times (9 - 3)$.

First solve for the intermediate result of $(6 + 7)$:

Keystrokes	Display	
6	6	
ENTER	6.0000	
7	7	
+	13.0000	$(6 + 7)$

Now perform $(9 - 3)$. Since another pair of numbers must be keyed in, one immediately after the other, use the ENTER key again to separate the first number (9) from the second (3). (There is no need to press ENTER to separate the 9 from the previous intermediate result of 13 that is already in the calculator—the results of previous calculations are stored automatically.) To solve $(9 - 3)$:

Keystrokes	Display	
9	9	
ENTER	9.0000	
3	3	
−	6.0000	$(9 - 3)$

Then multiply the intermediate results (13 and 6) together for the final answer:

Keystroke	Display	
×	78.0000	$(6 + 7) \times (9 - 3) = 78$

Notice that the HP-11C automatically stored the intermediate results for you and used them on a last-in, first-out basis when it was time to multiply. No matter how complicated a problem may look, it can always be reduced to a series of one- and two-number operations.

Remember:

- The ENTER key is used for separating the second number from the first in any operation requiring the sequential entry of two numbers.
- Any new digits keyed in following a calculation are automatically treated as a new number.

- Intermediate results are stored on a last-in, first-out basis.

Now try these problems. Work through them as you would with pencil and paper. Don't be concerned about intermediate answers—they are handled automatically by your HP-11C.

$$(16 \times 38) - (13 \times 11) = 465.0000$$

$$(27 + 63) \div (33 \times 9) = 0.3030$$

$$(\sqrt{(16.38 \times 0.55)}) \div .05 = 60.0300$$

$$4 \times (17 - 12) \div (10 - 5) = 4.0000$$

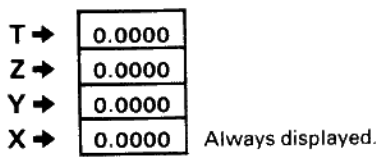
Section 2

The Automatic Memory Stack, LAST X, and Data Storage

The Automatic Memory Stack and Stack Manipulation

Automatic retention and return of intermediate results is the reason your HP-11C takes you through complex calculations so easily. The features supporting this ease of use are the automatic memory stack and the [ENTER] key.

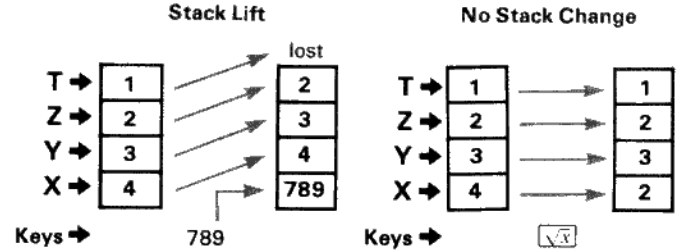
The Automatic Memory Stack Registers

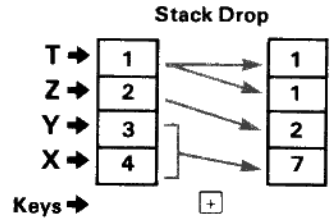


When your HP-11C is in Run mode (when the **PRGM** annunciator is not displayed), the number that appears in the display is *always* the number in the X-register.

Any number keyed in and the result of executing a numeric function is placed in the displayed X-register. Executing a function or keying in a number will cause numbers already in the stack to lift, remain in the same register, or drop, depending upon the type of operation being performed. Numbers in the stack are available on a last-in, first-out basis. If the stack was loaded as shown on the left of the following illustrations (as the result of previous

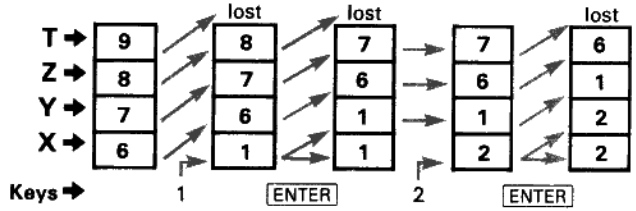
calculations), pressing the indicated keys would result in the stack arrangement shown on the right of each illustration.*





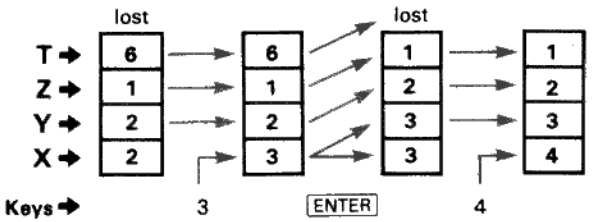
Stack Manipulation Functions

[ENTER] separates two numbers keyed in one after the other. When [ENTER] is pressed the calculator lifts the stack by copying the number in the displayed X-register into the Y-register. For example, to fill the stack with the numbers 1, 2, 3, 4 (assume that the stack registers have already been loaded with the numbers shown as the result of previous calculations):

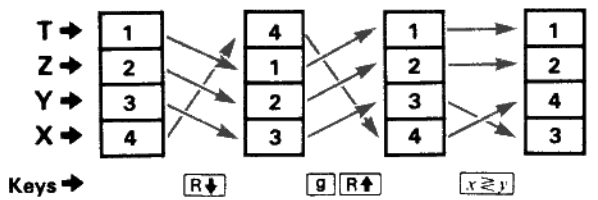


(Illustration continued on next page.)

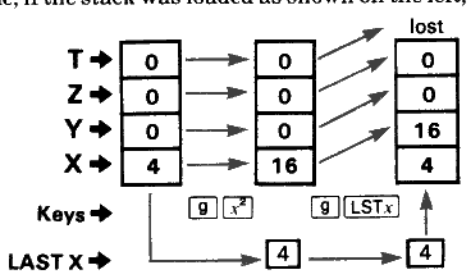
* To simplify the illustration of features described in this section, a single-digit number format is used in most of the diagrams instead of the decimal number ([FIX] 4) format used elsewhere in this handbook.



R↓ *(roll down)*, R↑ *(roll up)*, and $x \gtrless y$ *(X exchange Y)*. R↓ and R↑ rotate the contents of the stack registers down or up one register. No values are lost. $x \gtrless y$ exchanges the numbers in the X- and Y-registers. If the stack were loaded with the sequence 1, 2, 3, 4, the following shifts would result from pressing R↓, g R↑, and $x \gtrless y$.



LSTx *(LAST X)*. When a numeric function is executed, a copy of the value occupying the displayed X-register before the function was executed is stored in the LAST X register. Pressing g LSTx places a copy of the current contents of the LAST X register into the displayed X-register. (Refer to appendix B, Stack Lift and LAST X for a listing of the functions that save x in the LAST X register.) For example, if the stack was loaded as shown on the left, below:

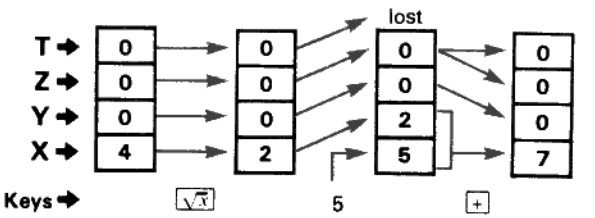


Calculator Functions and the Stack

When you want to key in two numbers, one after the other, you press ENTER between entries of the numbers. However, when you want to key in a number when the number already in the displayed X-register is the result of a previous calculation or other function (like $x \gtrless y$, R↑, etc.), you do not need to use ENTER. Why? Executing almost any HP-11C function has two results:

1. The specified function is executed.
2. The automatic memory stack is *enabled*; that is, the stack will lift automatically when the *next* number is keyed in.

For example, with 4 already keyed into the X-register:



There are four functions—ENTER, CLx, Σ+, and Σ−—which *disable* the stack.* They do *not* provide for the lifting of the stack when the *next* number is keyed in. Following the execution of one of these functions, keying in a new number will simply write over the currently displayed number instead of causing the stack to lift. (Although the stack lifts when ENTER is pressed, it will *not* lift when the *next* number is keyed in. The operation of ENTER illustrated on pages 27 and 28 shows how ENTER thus disables the stack.) In most cases, the above effects will come so naturally that you won't even think about them.

Two-Number Functions

An important aspect of two-number functions is the positioning of the numbers in the stack. To execute an arithmetic function, the numbers should be positioned in the same way that you would

* When pressing ← clears the entire display it operates the same as CLx and disables the stack. Otherwise, ← is neutral; that is, it does not affect the stack. For a further discussion of the stack, refer to appendix B, Stack Lift and LAST X.

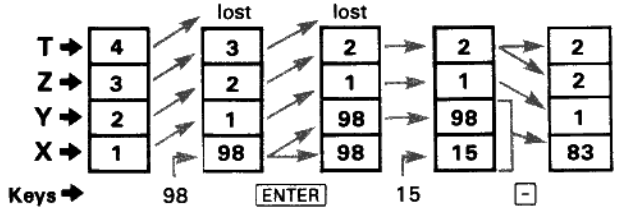
write them on paper. For example, to subtract 15 from 98 you first write 98 on paper, then write 15 underneath it, like this:

```
 98
-15
```

Then you would perform the subtraction, like this:

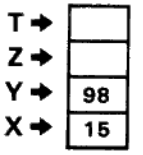
```
 98
-15
----
 83
```

The numbers are positioned in the calculator in the same way, with the first number, the minuend, in the Y-register and the second number, the subtrahend, in the displayed X-register. When the subtraction function is executed, the 15 in the X-register is subtracted from the 98 in the Y-register, and the stack drops, leaving the result in the X-register. Here is how the entire operation is executed (assume that the stack registers have already been loaded with the numbers shown as the result of previous calculations):



For any arithmetic function, the numbers are always positioned in their natural order first, then the function is executed and the stack drops. In the above example, we subtracted 15 from 98. The same number positioning would be used to add 15 to 98, multiply 98 by 15, and to divide 98 by 15, that is:

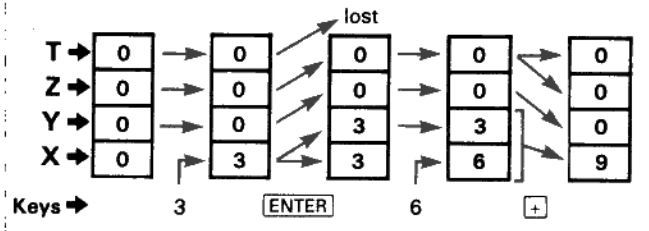

```
 98
+15
----

 98
x 15
----

 98
----
 15
```

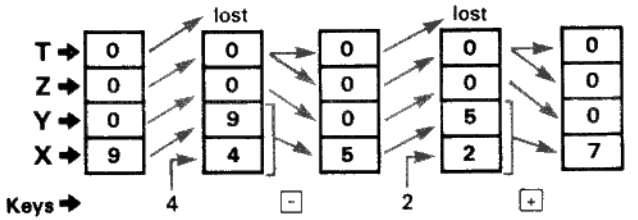


Chain Calculations

Whether you use your HP-11C mostly for direct keyboard solutions or programmed solutions, you are likely to use chain calculations frequently. It is in this area that the simplicity and power of your HP-11C's logic system becomes very apparent. Even during extremely long calculations, you still perform only one operation at a time. The automatic memory stack stores up to four intermediate results until you need them, then inserts them into the calculation. Thus, working through a problem is as natural as if you were working it out with pencil and paper.

You have already learned how to key in a pair of numbers using the ENTER key and then perform a calculation. You have seen how the stack drops as a result of executing some functions and how the stack lifts automatically when you key in a number after executing a function. To see how these features operate in a chain calculation, let's solve $3+6-4+2=?$ (Assume the stack cleared to zeros by pressing ← ENTER ENTER ENTER.)





As you can see, we worked through the problem one operation at a time. The stack automatically dropped after each two-number calculation. And, after each calculation, the stack automatically lifted when a new number was keyed in. Even more complicated problems are solved in the same simple manner.

Example: Instead of the arrow diagrams we've used earlier in this section, we'll use a table to follow stack operation as we solve the expression

$$\frac{(3+4)\times(6-4)}{2}$$

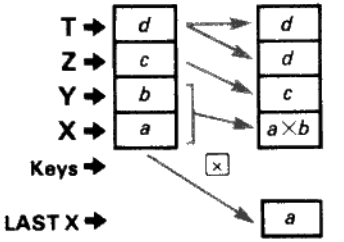
T →	0	0	0	0	0	0
Z →	0	0	0	0	0	7
Y →	0	3	3	0	7	6
X →	3	3	4	7	6	6
Keys →	3	ENTER	4	+	6	ENTER

T →	0	0	0	0	0
Z →	7	0	0	0	0
Y →	6	7	0	14	0
X →	4	2	14	2	7
Keys →	4	−	×	2	÷

LAST X

The HP-11C's LAST X register, a separate data storage register, preserves the value that was last in the display before execution of

a numeric function.*



This feature saves you from having to re-enter numbers you want to use again and can assist you in error recovery.

Example: To multiply two separate values, such as 45.575 meters and 25.331 meters by 0.175:

T →	0.0000	0.0000	0.0000	0.0000
Z →	0.0000	0.0000	0.0000	0.0000
Y →	0.0000	45.5750	45.5750	0.0000
X →	45.575	45.5750	0.175	7.9756
Keys →	45 · 575	ENTER	· 175	×
LAST X →				0.1750

T →	0.0000	0.0000	0.0000
Z →	0.0000	7.9756	0.0000
Y →	7.9756	25.3310	7.9756
X →	25.331	0.1750	4.4329
Keys →	25.331	g LSTx	×
LAST X →	0.1750	0.1750	0.1750

* The exceptions are the statistics functions x̄, s, and L.R.

[LSTx] makes it easy to recover from keystroke mistakes, such as executing the wrong function or keying in the wrong number. For example, divide 287 by 13.9 after you have mistakenly divided by 12.9:

Keystrokes	Display	
287 [ENTER]	287.0000	
12.9 [÷]	22.2481	Oops! The wrong divisor.
[g] [LSTx]	12.9000	Retrieves from LAST X the last entry to the X-register (the incorrect divisor) before [÷] was executed.
[×]	287.0000	Perform the reverse of the function that produced the wrong answer.
13.9 [÷]	20.6475	The correct answer.

Constant Arithmetic

Because the number in the T-register remains there when the stack drops, this number can be used as a constant in arithmetic operations.

T →	*c*	*c*
Z →	*c*	*c*
Y →	*c*	*c*
X →	*x*	*cx*
Keys →		[×]

To insert a constant into a calculation, load the stack with the constant by keying the constant into the X-register and pressing [ENTER] three times. Use the constant by keying in your initial argument and executing your planned series of arithmetic operations. Each time the stack drops, a copy of the constant will be made available for your next calculation and a new copy of the constant is reproduced in the T-register.

Example: A bacteriologist tests a certain strain of microorganisms whose population typically increases by 15% each day (a growth factor of 1.15). If he starts a sample culture of 1000, what will be the bacteria population at the end of each day for five consecutive days?

Method: Use [ENTER] to put the constant growth factor (1.15) in the Y-, Z-, and T-registers. Then put the original population (1000) in the displayed X-register. Thereafter, you calculate the new daily population whenever you press [×]. To set your calculator to the same display format as is shown in the following example, press [f] [FIX] 2.

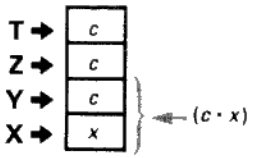
T →	0.00	0.00	0.00	1.15	1.15
Z →	0.00	0.00	1.15	1.15	1.15
Y →	0.00	1.15	1.15	1.15	1.15
X →	1.15	1.15	1.15	1.15	1,000
Keys →	1.15	[ENTER]	[ENTER]	[ENTER]	1000

T →	1.15	1.15	1.15	1.15	1.15
Z →	1.15	1.15	1.15	1.15	1.15
Y →	1.15	1.15	1.15	1.15	1.15
X →	1,150.00	1,322.50	1,520.88	1,749.01	2,011.36
Keys →	[×]	[×]	[×]	[×]	[×]

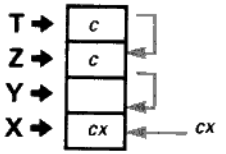
When you press [×] the first time, you calculate 1.15 × 1000. The result (1,150.00) is displayed in the X-register, the stack drops, and

a new copy of the constant is generated in the T-register, that is, each time you press [×]:

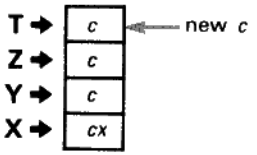
1. A new calculation involving the X- and Y-registers takes place.



2. The result of the calculation is placed in the displayed X-register and the contents of the rest of the stack drop.



3. A new copy of the number last in T (in this case, our constant) is generated in T.



Since a new copy of the growth factor is duplicated in the T-register each time the stack drops, you never have to re-enter it.

Press [f] [FIX] 4 to return the HP-11C to the [FIX] 4 display format.

Alternate Method: Constant arithmetic can also be performed using the LAST X register. To use this method to calculate the result of the preceding example:

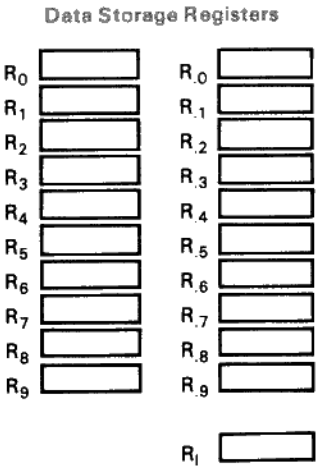
1. Key in the original population (1,000) and press [ENTER].
2. Key in the constant growth factor (1.15).
3. Press [×] to calculate the population at the end of one day.
4. Press [g] [LSTx] [×] to calculate the population at the end of each succeeding day.

Storage Register Operations

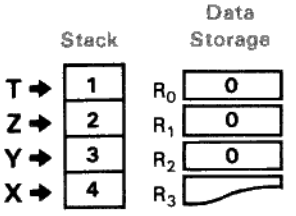
Storing and recalling numbers are operations involving the displayed X-register and the HP-11C's 21 data storage registers. Data storage registers are entirely separate from the stack and LAST X registers.

Storing Numbers

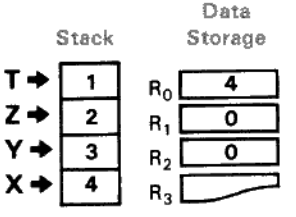
[STO] *(store)*. When followed by a storage register address (0 through 9, [.] 0 through [.] 9, or [I]), copies a number from the displayed X-register into the data storage register specified by the storage register address.



If...



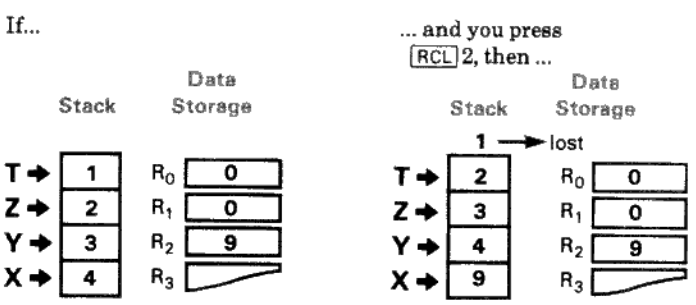
... and you press [STO] 0, then ...



A copy of the stored number remains in the storage register until a new number is stored there or until the storage registers are cleared or Continuous Memory is reset.

Recalling Numbers

RCL *(recall).* When followed by a storage register *address* (0 through 9, · 0 through · 9, or I) places a copy of the number in the specified data storage register into the displayed X-register. If the stack is not disabled, executing a RCL operation causes the stack to lift.



Storage and Recall Exercises

Execute the following operations:

Keystrokes	Display	
123	**123**	
STO 4	**123.0000**	Stores 123 in R_4.
678	**678**	
STO · 7	**678.0000**	Stores 678 in $R_{.7}$.
RCL 4	**123.0000**	Recalls 123 from R_4.
RCL · 7	**678.0000**	Recalls 678 from $R_{.7}$.

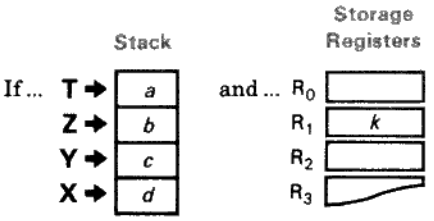
Clearing Data Storage Registers

CLEAR REG *(clear registers).* Clears the contents of all data storage registers to zero. CLEAR REG does *not* affect the stack or

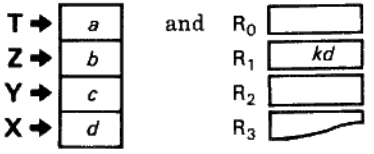
LAST X registers. (To clear a single data storage register, store zero in that register.)

Storage Register Arithmetic

STO (+, −, ×, ÷) *n (storage register arithmetic).* Uses the number in the displayed X-register to perform arithmetic upon the contents of a specified storage register *n*. The key sequence is STO followed by an arithmetic function key, followed in turn by the register address (0 through 9).* The result of any storage register arithmetic operation is placed in the specified data storage register.



represent the current status of the memories, executing STO × 1 results in:



The same numbers in the stack registers.

Contents of R_1 changed from k to $(k \times d)$.

*Storage register arithmetic can be performed in $R_{.0}$ through $R_{.9}$ using indirect addressing, which is covered in section 9, The Index Register. Storage register arithmetic cannot be performed in R_I.

Storage Register Arithmetic Exercises

Keystrokes	Display	
18 [STO] 0	18.0000	Stores 18 in R_0.
3 [STO] [÷] 0	3.0000	Divides number in R_0(18) by 3.
[RCL] 0	6.0000	Recalls copy of new number in R_0.
4 [STO] [×] 0	4.0000	Multiplies new number in R_0 (6.0000) by 4.
[RCL] 0	24.0000	Recalls copy of new number in R_0.
[STO] [+] 0	24.0000	Adds 24 to number in R_0.
[RCL] 0	48.0000	Recalls copy of new number in R_0.
40 [STO] [−] 0	40.0000	Subtracts 40 from number in R_0.
[RCL] 0	8.0000	Recalls copy of new number in R_0.

Problems

1. Calculate the value of x in the following equation.

$$x = \sqrt{\frac{8.33(4 - 5.2) \div ((8.33 - 7.46) \times 0.32)}{4.3(3.15 - 2.75) - (1.71 \times 2.01)}}$$

Answer: 4.5728

A possible keystroke solution is:

4 [ENTER] 5.2 [−]
8.33 [×] [g] [LSTx] 7.46
[−] .32 [×] [÷]
3.15 [ENTER] 2.75 [−]
4.3 [×] 1.71 [ENTER]
2.01 [×] [−] [÷] [$\sqrt{x}$]

2. Use constant arithmetic to calculate the remaining balance of a $1000 loan after six payments of $100 each and an interest rate of 1% per payment period.

Procedure: Load the stack with $(1 + i)$ and key in the initial loan balance. Use the following formula to find the new balance after each payment:

$$\text{New Balance} = ((\text{Old Balance}) \times (1 + i)) - \text{Payment}$$

Answer: 446.3186

3. Store 100 in R_5. Then:
   1. Divide the contents of R_5 by 25.
   2. Subtract 2 from the contents of R_5.
   3. Multiply the contents of R_5 by 0.75.
   4. Add 1.75 to the contents of R_5.
   5. Recall the contents of R_5.

Answer: 3.2500.

Section 3

Numeric Functions

Your HP-11C's numeric function set enables you to perform a wide range of operations involving number alteration, math, and statistics. Each function is used in the same way for both keyboard and program execution.

Pi

Pressing [f] [π] places a 10-digit approximation of the value of pi (3.141592654) in the displayed X-register. If the stack is not disabled, pressing [f] [π] causes the stack to lift.

Number Alteration Functions

In addition to [CHS] (refer to page 17, Negative Numbers) your HP-11C has four functions for altering numbers: [ABS], [INT], [FRAC], and [RND].

Absolute Value. Pressing [g] [ABS] changes the number in the displayed X-register to the absolute value of that number.

Integer Portion. Pressing [g] [INT] replaces the number in the displayed X-register with its integer portion, that is, it replaces all digits to the right of the decimal with zeroes.

Fractional Portion. Pressing [f] [FRAC] replaces the number in the displayed X-register with its decimal portion, that is, it replaces any digits to the left of the decimal with zeroes.

Rounding. Pressing [g] [RND] rounds the internally held 10-digit mantissa of any displayed value to the number of digits specified by the current [FIX], [SCI], or [ENG] display setting.

To Calculate	Example Keystrokes	Display
Absolute value	12345 [CHS]	**-12,345**
	[g] [ABS]	**12,345.0000**
Integer portion	123.4567	**123.4567**
	[g] [INT]	**123.0000**

To Calculate	Example Keystrokes	Display
Fractional portion	123.4567	**123.4567**
	[f] [FRAC]	**0.4567**
Round (Assumes [FIX] 4 display setting.)	1.23456789	**1.23456789**
	[g] [RND]	**1.2346**
Inspect rounding.	[f] [FIX] 8	**1.23460000**
Reset to [FIX] 4.	[f] [FIX] 4	**1.2346**

One-Number Functions

The HP-11C's one-number math functions have the following characteristics:

- Use the number in the displayed X-register as the argument for the function.
- Replace the number in the displayed X-register with the result of executing the function.
- Do not affect numbers in the Y-, Z-, and T-registers.

General Functions

Reciprocal. Pressing [1/x] calculates the reciprocal of the number in the displayed X-register, that is, it divides 1 by the number in the displayed X-register.

Factorial and Gamma. Pressing [f] [x!] calculates the factorial or Gamma value as follows:

1. Factorial.

   When executed with a nonnegative integer n ($0 \leq n \leq 69$) in the displayed X-register, [x!] calculates the factorial of n, that is, it calculates the product of the integers from 1 to n.

2. Gamma Function.

   The [x!] key can also be used to calculate the Gamma function, denoted by $\Gamma(x)$, which occurs in certain problems in advanced mathematics and statistics.* Pressing [x!] gives you $\Gamma(x+1)$. To calculate the Gamma function of a number, subtract 1 from the number. Then, with the result in the displayed X-register, press [f] [x!].

Square Root. Pressing [√x] calculates the square root of the number in the displayed X-register.

Squaring. Pressing [g] [x²] calculates the square of the number in the displayed X-register.

To Calculate	Keystroke Example	Display
Reciprocal	25 [1/x]	25 0.0400
Factorial	8 [f] [x!]	8 40,320.0000
Gamma	2.7 [ENTER] 1 [−] [f] [x!]	2.7 1.7000 1.5447
Square Root	3.9 [√x]	3.9 1.9748
Square	12.3 [g] [x²]	12.3 151.2900

*[x!] can be used for both the Factorial and Gamma functions because when x is a nonnegative integer n, $\Gamma(x+1) = \Gamma(n+1) = n!$. The Gamma function can be regarded as a generalization of the factorial function, since the number in the X-register is not limited to nonnegative integers. Conversely, the factorial function can be regarded as a special case of the Gamma function.

Trigonometric Operations

The six basic trigonometric functions operate in the trigonometric mode you select.

Trigonometric Modes. Selecting a specific trig mode does not convert any number already in the calculator to that mode. Selecting a specific trig mode is simply telling the calculator what unit of measure (degree, radian, or grad) to use when executing a trig function.

Pressing [g] [DEG] selects the Degree trig mode. No annunciator appears in the display.

Pressing [g] [RAD] selects the Radian trig mode. While [RAD] mode is set, the **RAD** annunciator appears in the display.

[g] [RAD] → 0.0000 RAD

Pressing [g] [GRD] selects the Grad trig mode. While [GRD] mode is set, the **GRAD** annunciator appears in the display.

[g] [GRD] → 0.0000 GRAD

The calculator is always set to one of the three trig modes. Continuous Memory maintains the last trig mode selected, even when the calculator is turned off, then on again. If a power loss occurs, or if you reset Continuous Memory (refer to page 20), the calculator will automatically reset to [DEG] mode.

Trigonometric Functions.

Pressing	Calculates
[SIN]	sine
[g] [SIN⁻¹]	arc sine
[COS]	cosine
[g] [COS⁻¹]	arc cosine
[TAN]	tangent
[g] [TAN⁻¹]	arc tangent

To use any of the trig functions, ensure that the calculator is set to the desired trig mode ([DEG], [RAD], or [GRD]), then execute the desired trig function.

To Calculate	Keystroke Example	Display
(Examples assume [DEG] trig mode.)		
All Trig functions, for example:		
Sine	33.5 [SIN]	33.5 0.5519
Arc Sine	.7982 [g] [SIN^{-1}]	0.7982 52.9586

Time and Angle Conversions

Numbers representing time or angles are interpreted by the HP-11C in a decimal or minutes-seconds format, depending upon the conversion being executed:

Hours.Decimal Hours (H.h)	Hours.Minutes Seconds Decimal Seconds (H.MMSSs)
or	or
Degrees.Decimal Degrees (D.d)	Degrees.Minutes Seconds Decimal Seconds (D.MMSSs)

Hours (or Degrees), Minutes, Seconds Conversion. Pressing [f] [→H.MS] converts the number in the displayed X-register from a decimal hours (or decimal degrees) format to an hours (or degrees), minutes, seconds, decimal seconds format.

H.h → H.MMSSs
or
D.d → D.MMSSs

Decimal Hours (or Degrees) Conversion. Pressing [g] [→H] converts the number in the displayed X-register from an hours (or degrees), minutes, seconds, decimal seconds format to a decimal hour (or degrees) format.

H.MMSSs → H.h
or
D.MMSSs → D.d

Degrees/Radians Conversions

The [→DEG] and [→RAD] functions are used to convert angles between *decimal* degrees and radians (D.d → R.r and R.r → D.d).

Degrees to Radians Conversions. Pressing [f] [→RAD] converts the number in the displayed X-register from a decimal degree value to its radian equivalent.

Radians to Degrees Conversion. Pressing [g] [→DEG] converts the number in the displayed X-register from a radian value to its decimal degree equivalent.

To Convert	Example Keystrokes	Display
Decimal hours (H.h) or degrees (D.d) to H.MMSSs or D.MMSSs format.	17.553 [f] [→H.MS]	17.553 17.3311
To view decimal seconds while in [FIX] 4 setting, press:	[f] [PREFIX]	1733108000 17.3311
H.MMSSs or D.MMSSs to decimal hours (H.h) or degrees (D.d) format.	12.3045 [g] [→H]	12.3045 12.5125
Degrees to Radians	40.5 [f] [→RAD]	40.5 0.7069
Radians to Degrees	1.1746 [g] [→DEG]	1.1746 67.2996

Logarithmic Functions

Natural Logarithm. Pressing [g] [LN] calculates the natural logarithm of the number in the displayed X-register, that is, the logarithm to the base *e* (2.718281828) of the number in the displayed X-register.

Natural Antilogarithm. Pressing [e^x] calculates the natural antilogarithm of the number in the displayed X-register, that is, it raises *e* (2.718281828) to the power of the value in the X-register.

Common Logarithm. Pressing [g] [LOG] calculates the common logarithm of the number in the displayed X-register, that is, the logarithm to the base 10.

Common Antilogarithm. Pressing [10^x] calculates the common antilogarithm of the number in the displayed X-register, that is, it raises 10 to the power of that number.

To Calculate	Example Keystrokes	Display
Natural Log	45 [g] [LN]	45 3.8067
Natural Antilog	3.4012 [e^x]	3.4012 30.0001
Common Log	12.4578 [g] [LOG]	12.4578 1.0954
Common Antilog	3.1354 [10^x]	3.1354 1,365.8405

Hyperbolic Functions

Pressing	Calculates
[f] [HYP] [SIN]	Hyperbolic sine *(sinh)*
[g] [HYP^{-1}] [SIN]	Inverse hyperbolic sine *($\sinh^{-1}$)*
[f] [HYP] [COS]	Hyperbolic cosine *(cosh)*

Pressing	Calculates
[g] [HYP^{-1}] [COS]	Inverse hyperbolic cosine *($\cosh^{-1}$)*
[f] [HYP] [TAN]	Hyperbolic tangent *(tanh)*
[g] [HYP^{-1}] [TAN]	Inverse hyperbolic tangent *($\tanh^{-1}$)*

To Calculate	Example Keystrokes	Display
All hyperbolic functions; for example:		
Hyperbolic sine	2.53 [f] [HYP] [SIN]	2.53 6.2369
Inverse hyperbolic sine	1.95 [g] [HYP^{-1}] [SIN]	1.95 1.4210

Two-Number Functions

Your HP-11C's two-number math functions use the values in the displayed X-register and in the Y-register to calculate a result. To use any of these functions, key in the Y-register value first, press [ENTER] to lift the value into the Y-register, key in the displayed X-register value, then execute the function.

Exponential

Pressing [y^x] raises the number in the Y-register to the power of the number in the X-register.

To Calculate	Example Keystrokes	Display
Exponential	2 [ENTER] 3 [y^x]	2.0000 3 8.0000

Percentages

Percent. To find a specified percentage of a number:

1. Key in the base number.
2. Press [ENTER].
3. Key in the percent rate.
4. Press [g] [%].

T →				
Z →				
Y →		150	150	150
X →	150	150	25	37.5
Keys →	150	ENTER	25	g %
LAST X →				25

The percentage will appear in the displayed X-register, the base number will remain in the Y-register, and the percentage *rate* will be placed in LAST X. The stack does not lift, so any values held in the Z- and T-registers before pressing g % will remain. The above illustration shows the use of the % key to calculate 25% of 150.

Percent Difference. The Δ% function calculates the percent *difference*—the relative increase or decrease—between two numbers. To find the percent difference:

1. Key in the base number (typically, the number that occurs first in time).
2. Press ENTER.
3. Key in the second number.
4. Press g Δ%.

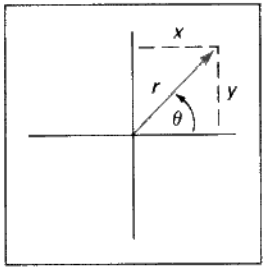
T →				
Z →				
Y →		150	150	150
X →	150	150	225	50
Keys →	150	ENTER	225	g Δ%
LAST X →				225

Using the above order of entry, a positive result signifies an increase of the second number over the first; a negative result signifies a decrease of the second number below the first. The preceding illustration shows the use of Δ% to calculate the percent increase realized between 150 (the base number) and 225 (the second number in time).

To Calculate	Example Keystrokes	Display
Percent		
Base Number	200 ENTER	200.0000
Percent Rate	75	75
Percentage	g %	150.0000
Percent Difference		
Base Number	40 ENTER	40.0000
Second Number	160	160
Percent Increase	g Δ%	300.0000

Polar-Rectangular Coordinate Conversions

Two functions (→P, →R) are provided in your HP-11C for polar/rectangular coordinate conversions.

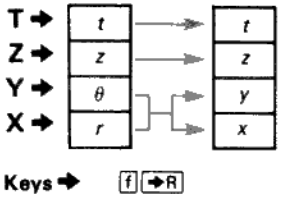


Angle θ is assumed to be in *decimal* degrees, radians, or grads, depending upon which trigonometric mode (DEG, RAD, or GRD) the calculator is set to. Angle θ is measured as shown in the illustration to the right. The answer returned for θ is between 180° and −180°.

Polar Conversion. Pressing g →P (*polar*) converts values in the X- and Y-registers representing rectangular coordinates (*x*, *y*) to polar coordinates (magnitude *r*, angle θ).

T →	t	t
Z →	z	z
Y →	y	θ
X →	x	r
Keys →	g →P	

Rectangular Conversion. Pressing f →R (*rectangular*) converts values in the X- and Y-registers representing polar coordinates (magnitude r, angle θ), to rectangular coordinates (x, y).



To Convert	Example Keystrokes	Display
Rectangular coordinates to polar:		
y	5 ENTER	5.0000
x	10	10
r	g →P	11.1803
θ	$x \gtrless y$	26.5651
Polar coordinates to rectangular:		
θ	30 ENTER	30.0000
r	12	12
x	f →R	10.3923
y	$x \gtrless y$	6.0000

Probability

Permutation. Pressing f $P_{y,x}$ calculates the number of possible *arrangements* of y different items taken in quantities of x items at a time, where different orders of the same x items are counted separately. (No item occurs more than once in an arrangement.) $P_{y,x}$ calculates permutations by the following formula:

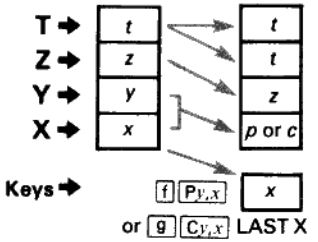
$$P_{y,x} = \frac{y!}{(y-x)!}$$

Combination. Pressing g $C_{y,x}$ calculates the number of possible *sets* of y different items taken in quantities of x items at a time regardless of order (where no item occurs more than once in a set). $C_{y,x}$ calculates combinations by the following formula:

$$C_{y,x} = \frac{y!}{x!(y-x)!}$$

To execute a permutation or combination:

1. Key in the number of items (y).
2. Press ENTER.
3. Key in the quantity of items required (x) per arrangement or set.
4. Press f $P_{y,x}$ or g $C_{y,x}$.



The result, p or c, will appear in the displayed X-register, the stack will drop, and the quantity per arrangement or set (x) is placed in the LAST X register.

> All permutation and combination inputs must be non-negative integers.

Note: The execution times for $P_{y,x}$ and $C_{y,x}$ calculations can be several seconds or longer, depending on the magnitude of your y and x inputs. The maximum value you can input for a $P_{y,x}$ or $C_{y,x}$ calculation is $10^{10} - 1$. The **running** message will flash in the display during execution of $P_{y,x}$ and $C_{y,x}$ calculations.

To Calculate	Example Keystrokes	Display
Permutation Example: 10 items in arrangements of 3 at a time.	10 [ENTER] 3 [f] [$P_{y,x}$]	10.0000 720.0000
Combination Example: 10 items in sets of 3 at a time.	10 [ENTER] 3 [g] [$C_{y,x}$]	10.0000 120.0000

Statistics Functions

Random Number Generator

The HP-11C's random number generator uses either an automatically stored seed (zero), or a seed you key in, to initiate a uniformly distributed pseudo-random number sequence in the range $0 \leq r < 1$.* Because of Continuous Memory, the sequence will continue until a new initial seed is stored.

Random Number Seed. To initiate a new random number sequence at any time, place a new seed in the random number generator by keying in any number $0 \leq n < 1$† and pressing [STO] [f] [RAN#]. To generate a random number, simply press [f] [RAN#]. The new random number will appear in the displayed X-register. ([f] [RAN#] affects the stack in the same way as recalling a number from a storage register.) The newly generated random number also becomes the seed for the next random number in the current sequence. Whenever Continuous Memory is reset, or when a power failure occurs, the random number seed is set to zero. (Repeated use of the same value for a random number seed will produce repetitions of the same random number sequence.)

* Passes the spectral test (Knuth, Vol. 2).

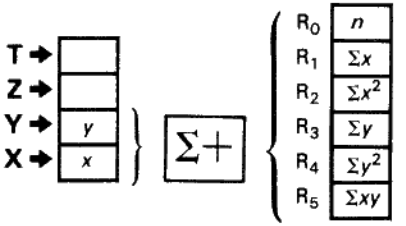
† If a number whose magnitude is ≥ 1 is used as a seed, the decimal point in the stored version of the number will be shifted to the left of the first significant digit. For example, 123.0, 12.3, and 0.123 would all be stored as 0.123 by [STO] [f] [RAN#].

Random Number Generation	Example Keystrokes	Display
To store 0.5764 as a random number seed: (Arbitary Seed)	.5764 [STO] [f]	0.5764
Seed stored	[RAN#]	0.5764
To generate a random number series based on the above seed:	[f] [RAN#] [f] [RAN#] ⋮	0.3422 0.2809 ⋮

For a further discussion of the random number generator, refer to page 217, Random Numbers, in part III of this handbook.

Accumulating Statistics

[Σ+] is a two-number function that calculates statistics of the values in the X- and Y-registers. The results are automatically accumulated in storage registers R_0 through R_5.

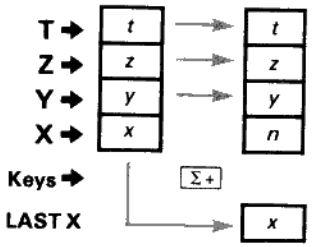


Any values remaining in R_0 through R_5 from previous calculations or storage operations will be included in subsequent statistics accumulations. To ensure that all accumulations registers are set to zero before beginning a new series of accumulations, press [f] CLEAR [Σ] (clears R_0 through R_5 and the stack) before beginning the operation.

When you press the [Σ+] key, the following statistics are placed in the indicated data storage registers:

Register	Contents
R_0	n : Number of data points (pairs) accumulated. (n also appears in the displayed X-register.)
R_1	Σx : Summation of x values.
R_2	Σx^2 : Summation of squares of x values.
R_3	Σy : Summation of y values.
R_4	Σy^2 : Summation of squares of y values.
R_5	Σxy : Summation of products of x and y values.

When you execute [Σ+], the number previously in the X-register is placed in the LAST X register and the updated n-value is placed in X. *The number previously in the Y-register is not changed.*



When you key in a new number, the n-value in the displayed X-register will be written over; the stack does not lift.

If your statistics problem involves only one variable (x) instead of two (x and y), ensure that the Y-register holds zero for each execution of the [Σ+] function. (Pressing [f] CLEAR [Σ] once, just before beginning a new accumulations series, ensures a clear Y-register for a one-number series by clearing the stack as well as the Σ registers (R_0 through R_5).

Some sets of data points consist of a series of x-values (or y-values) that differ from each other by a comparatively small amount. You can maximize the precision of any statistical calculation involving

such data by keying in only the differences between each value and a number approximating the average of the values. This number must be added to the result of calculating $\bar{x}$, $\hat{y}$, or the y-intercept of L.R. For example, if your x values consist of 665999, 666000, and 666001, you should enter the data as -1, 0, and 1. If afterwards you calculate $\bar{x}$, add 666000 to the answer. In some cases the calculator cannot compute s, r, L.R., or $\hat{y}$ with data values that are too close to each other; and if you attempt to do so, the calculator will display **Error 2.** This will not happen, however, if you normalize the data as described above.

Note: Unlike storage register arithmetic, the [Σ+] and [Σ-] operations allow overflow to occur in storage registers R_0 through R_5 without indicating **Error 1** in the display.

You can recall any of the statistics accumulations to the displayed X-register by pressing [RCL] and the number of the data storage register holding the desired statistical accumulation. If you want to recall both the Σx and Σy statistics, press [RCL] [Σ+]. This simultaneously copies Σx from R_1 into the displayed X-register and copies Σy from R_3 into the Y-register. (Pressing [RCL] [Σ+] causes the stack to lift in the same way that it would if you keyed in two numbers in sequence.)

Example. Electrical energy researcher Helen I. Voltz suspects a possible relationship between the rise in worldwide coal production in the years 1972 through 1976 and a similar rise in worldwide electricity output for the same period. To assist in a study of the data, Voltz will use her HP-11C to accumulate the coal production and electrical output statistics. Find Σx, Σx^2, Σy, Σy^2, and Σxy for the paired x and y values of Voltz's data.

Year	1972	1973	1974	1975	1976
Coal Production (y) (Billions of Metric Tons)	1.761	1.775	1.792	1.884	1.943
Electricity Output (x) (Billions of Megawatt Hours)	5.552	5.963	6.135	6.313	6.713

Keystrokes	Display	
f CLEAR Σ	0.0000	Clear statistical data storage registers (R_0 through R_5 and stack).
1.761 ENTER	1.7610	
5.552 Σ+	1.0000	1972 data
1.775 ENTER	1.7750	
5.963 Σ+	2.0000	1973 data
1.792 ENTER	1.7920	
6.135 Σ+	3.0000	1974 data
1.884 ENTER	1.8840	
6.313 Σ+	4.0000	1975 data
1.943 ENTER	1.9430	
6.713 Σ+	5.0000	1976 data
RCL 1	30.6760	Sum of x values (Σx) from register R_1.
RCL 2	188.9386	Sum of squares of x values (Σx^2) from register R_2.
RCL 3	9.1550	Sum of y values (Σy) from register R_3.
RCL 4	16.7877	Sum of squares of y values (Σy^2) from register R_4.
RCL 5	56.2924	Sum of products of x and y values (Σxy) from register R_5.

Correcting Statistics Accumulations

If you discover that you have entered data incorrectly, the accumulated statistics can be easily corrected.

1. Key the *incorrect* data pair into the X- and Y-registers.
2. Press Σ− to delete the incorrect data.
3. Key in the correct values for x and y. If one value of an (x, y) data pair is incorrect, you must delete and re-enter both values.
4. Press Σ+.

Note: Although g Σ− can be used to delete an erroneous (x, y) pair, it will not delete any rounding errors that may have occurred when the statistics of that pair were added into accumulating registers R_1 through R_5. Consequently, subsequent results may be different than they would have been if the erroneous pair had not been entered via Σ+ and then deleted via Σ−. However, the difference will not be serious unless the erroneous pair has a magnitude that is enormous compared with the correct pair; and in such a case it may be wise to start over again and re-enter the data again (and more carefully!).

Example. After keying in the preceding data, Voltz found new information indicating that the coal output for the last data pair should have been 1.946 instead of 1.943. Use Σ− to remove the statistical data that was accumulated as a result of using the older, incorrect data pair. Then input the correct data pair.

Keystrokes	Display	
1.943 ENTER	1.9430	Key in the data pair we
6.713 g Σ−	4.0000	want to replace and delete the pair's unwanted statistics. Number of pair entries then drops to 4.
1.946 ENTER	1.9460	Key in and accumulate
6.713 Σ+	5.0000	the replacement data pair. Number of pairs accumulated is again five.

Retain the preceding statistics in your calculator for use in the following examples.

Mean

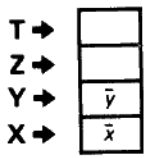
The [x̄] function computes the arithmetic *mean* (average) of the x and y statistics accumulated in registers R_1 and R_3, respectively. When you press [g] [x̄]:

1. The contents of the stack registers lift in the same way as when you press [RCL] [Σ+], as described on page 57.
2. The mean of the x values ($\bar{x}$) is calculated using the statistics accumulated in $R_1(\Sigma x)$ and $R_0(n)$. The mean of the y values ($\bar{y}$) is calculated using the data accumulated in registers $R_3(\Sigma y)$ and $R_0(n)$. The formulas used are shown below.

$$\bar{x} = \frac{\Sigma x}{n} \qquad \bar{y} = \frac{\Sigma y}{n}$$

3. The values for $\bar{x}$ and $\bar{y}$ are placed in the X- and Y-registers of the stack.

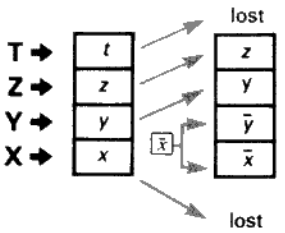
[g] [x̄] =



Example. From the five year statistical data you accumulated (and corrected) in the previous example, calculate the average coal production and electrical output for the entire period.

Keystrokes	Display	
[g] [x̄]	**6.1352**	Average electrical output (average of X-register inputs) for the five-year period.
[x≷y]	**1.8316**	Average coal production (average Y-register inputs) for the five-year period.

The illustration below shows what happens in the stack when you execute [x̄] (assumes stack disabled, as it would be following a [Σ+] operation):



Retain the preceding statistics in your HP-11C for use in the next example.

Standard Deviation

Pressing [g] [s] computes the *standard deviation* (a measure of dispersion around the mean) of the accumulated statistics data. The formulas used by the HP-11C to compute s_x, the standard deviation of the accumulated x values, and s_y, the standard deviation of the accumulated y values are:

$$s_x = \sqrt{\frac{n\Sigma x^2 - (\Sigma x)^2}{n(n-1)}} \qquad s_y = \sqrt{\frac{n\Sigma y^2 - (\Sigma y)^2}{n(n-1)}}$$

These formulas give the *best estimates* of the *population* standard deviations from the *sample* data. Consequently, the standard deviation given by these formulas is termed by convention the *sample* standard deviation. When you press [g] [s]:

1. The contents of the stack registers are lifted in the same way as they are when you press [RCL] [Σ+], as described on page 57.
2. The standard deviation of the x values (s_x) is calculated using the data accumulated in registers $R_2(\Sigma x^2)$, $R_1(\Sigma x)$, and $R_0(n)$ according to the formula shown above. The resultant value for s_x is placed in the X-register.

3. The standard deviation of the y values (s_y) is calculated using the statistical data accumulated in registers $R_4(\Sigma y^2)$, $R_3(\Sigma y)$, and $R_0(n)$ according to the formula shown above. The resultant value for s_y is available in the Y-register.

Example. Calculate the standard deviation for the corrected coal production and electrical output accumulations used in the previous examples.

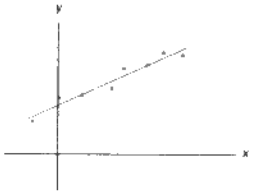
Keystrokes	Display	
g s	**0.4287**	Standard deviation of electrical output (X-register inputs) for the five-year period.
x≷y	**0.0800**	Standard deviation of coal production (Y-register inputs) for the five-year period.

Retain the preceding statistics in your HP-11C for use in the next example.

When your data constitutes not just a sample of a population but rather *all* of the population, the standard deviation of the data is the *true* population standard deviation (denoted σ). The formula for the true population standard deviation differs by a factor of $[(n-1)/n]^{1/2}$ from the formula used for the s function. The difference between the values is small, and for most applications can be ignored. Nevertheless, if you want to calculate the exact value of the population standard deviation for an entire population, you can easily do so with just a few keystrokes on your HP-11C. Simply add, using the Σ+ key, the mean ($\bar{x}$) of the data to the data and then press g s. The result will be the true population standard deviation of the original data.

Linear Regression

Linear regression is a statistical method for finding a straight line that best fits a set of two or more data pairs, thus providing a relationship between two variables. After the statistics of a group of data pairs has been accumulated in register R_0 through R_5, you can calculate the coefficients in the linear equation $y = Ax + B$ using the least squares method by pressing f L.R.



To use the linear regression function on your HP-11C, use the Σ+ key to accumulate the statistics of a series of two or more data pairs. Then execute L.R.. When you press f L.R.:

1. The contents of the stack registers are lifted just as they are when you press RCL Σ+, as described on page 57.
2. The slope (A) and the y-intercept (B) of the least squares line of the data are calculated using the equations:

$$A = \frac{n\Sigma xy - \Sigma x \Sigma y}{n\Sigma x^2 - (\Sigma x)^2} \qquad B = \frac{\Sigma y \Sigma x^2 - \Sigma x \Sigma xy}{n\Sigma x^2 - (\Sigma x)^2}$$

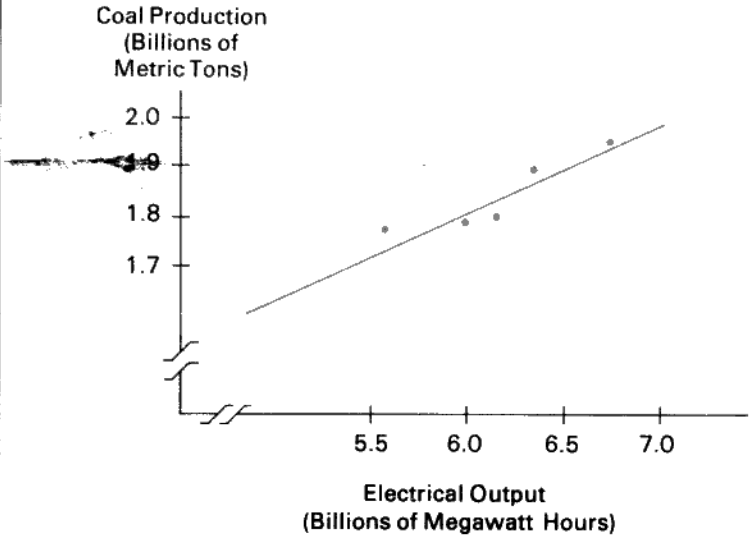
The slope A is placed in the Y-register; the y-intercept, B, is placed in the displayed X-register.

f L.R. =

T →	
Z →	
Y →	slope (A)
X →	y-intercept (B)

Example: Calculate the y-intercept and slope of Voltz's corrected data.

Solution: Voltz *could* draw a plot of coal production against electrical output, like the one in the following illustration. However, with her HP-11C, Voltz has only to accumulate the statistics (as we have already done) using the Σ+ key, then press f L.R.



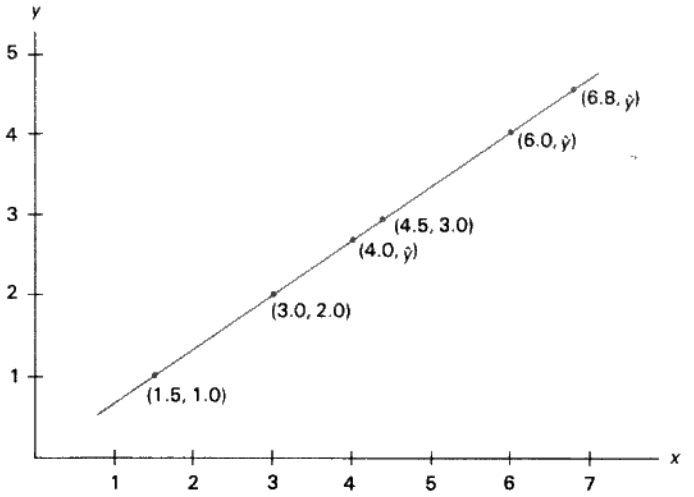
Keystrokes	Display	
f L.R.	0.7773	*Y*-intercept of the line.
x≷y	0.1718	Slope of the line.

Retain the preceding statistical accumulations in your calculator for use in the next example.

Linear Estimation and Correlation Coefficient

When you execute the ŷ,r function, the *linear estimate* ($\hat{y}$) is placed in the displayed X-register, and the *correlation coefficient* r is placed in the Y-register.

Linear Estimation. With statistics accumulated in registers R_0 through R_5, an estimated value for y (denoted $\hat{y}$) can be calculated by keying in a known value for x and pressing f ŷ,r.

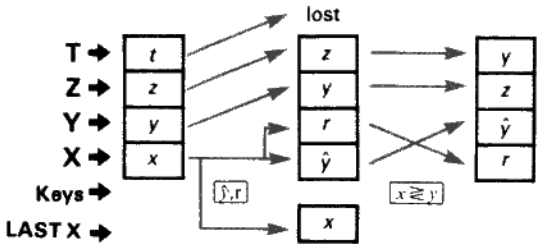


An estimated value for x (denoted $\hat{x}$) can be calculated as follows:

1. Press f L.R.
2. Key in the known y-value.
3. Press x≷y − x≷y +.

Correlation Coefficient. Both linear regression and linear estimation presume that the relationship between the x and y data values can be approximated, to some degree, by a linear function (that is, a straight line). The correlation coefficient (r) is a determination of how closely your data "fits" a straight line. The correlation coefficient can range from $r = +1$ to $r = -1$. At $r = +1$ the data falls exactly onto a straight line with positive slope. At $r = -1$, the data falls exactly on a straight line with negative slope. At $r = 0$, the data cannot be approximated at all by a straight line. With statistics accumulated in registers R_0 through R_5, the correlation coefficient r is calculated by pressing f ŷ,r. The number that appears in the displayed X-register will be a $\hat{y}$ value (meaningless, unless you keyed in a specific x-value, as described above). To view the correlation coefficient value (r), exchange the contents of the X- and Y-registers by pressing x≷y.

Recall from the discussion of LAST X in section 2 that [x̄], [s], and [L.R.] do not place a copy of the x-value in the LAST X register. However, because $\hat{y}$ is calculated from the value in the displayed X-register, when you press [f] [ŷ,r], a copy of the x-value is placed in the LAST X register and the stack—in all cases—lifts only once.



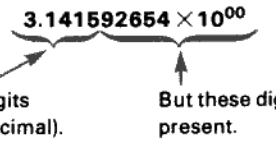
Example. Using the statistics saved from the previous example, if Voltz wishes to predict coal production ($\hat{y}$) for 1977, she keys in an estimate of electrical production (a "known" x-value) for 1977 and presses [f] [ŷ,r]. Because the correlation coefficient for Voltz's data is automatically included in the calculation, she can view how closely her data fits a straight line by simply pressing [x≷y] after the $\hat{y}$ prediction appears in the display.

Keystrokes	Display	
7.1417	**7.1417**	Voltz's estimate of 1977 electrical output.
[f] [ŷ,r]	**2.0046**	Predicted coal production for 1977.
[x≷y]	**0.9211**	The data closely approximates a straight line.

Section 4

Display Control

When you turn on your HP-11C, because of Continuous Memory, the display setting will be the same as it was before you last turned off the calculator. But regardless of the display options in effect, the HP-11C always internally represents each number as a 10-digit mantissa and a two-digit exponent of 10. Thus, when the calculator is set to display only four digits past the decimal point, the fixed constant pi, for example, appears in the display as 3.1416. However, pi is always represented internally as $3.141592654 \times 10^{00}$.



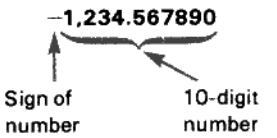
Display Mode Control

Your HP-11C has three display modes, [FIX], [SCI], and [ENG], that use a variable (0 through 9) to specify display setting. The following illustration shows how the number 123,456 would be displayed by a 4-digit setting in each of the three modes.

```
[f] [FIX] 4 : 123,456.0000
[f] [SCI] 4 :   1.2346  05
[f] [ENG] 4 : 123.46    03
```

Fixed Decimal Display

[FIX] *(fixed decimal)* displays numbers using a fixed decimal mode without exponents. In any [FIX] display setting, the calculator will automatically switch to [SCI] mode to allow viewing of a displayed number that is too large or too small to be viewed in the current [FIX] mode. The calculator will automatically switch back to the specified [FIX] mode when a number is displayed that can be viewed in that particular [FIX] mode display setting.



Fixed decimal display is selected or modified by pressing [f] [FIX] followed by the appropriate number key to specify the number of decimal places (0 to 9) to which you want the display rounded.

Keystrokes	Display	
123.45678 [ENTER]	**123.4568**	Display is rounded to four decimal places. However, internally the number is maintained in its original value to 10 digits.
[f] [FIX] 6	**123.456780**	The display is rounded upward if the first undisplayed digit is 5 or greater.
[f] [FIX] 0	**123.**	
[f] [FIX] 4	**123.4568**	Usual [FIX] 4 display.

Scientific Notation Display

[SCI] *(scientific)* displays numbers in scientific notation mode. To select or modify a [SCI] mode, press [f] [SCI] followed by the number key (0 through 6) that specifies the number of decimal places you want the display rounded to. For display rounding, 7, 8, and 9 can also be used, but no more than six digits can be displayed to the right of the decimal while in [SCI] mode.*

Keystrokes	Display		
123.4567895 [ENTER]	**123.4568**		Display is rounded to 4 decimal places.
[f] [SCI] 2	**1.23**	**02**	1.23×10^2; display rounded down.

* [SCI] 8 or 9 and [ENG] 8 or 9 are stored in program memory as [SCI] 7 and [ENG] 7.

Keystrokes	Display		
[f] [SCI] 4	**1.2346**	**02**	1.2346×10^2; display rounded up.
[f] [SCI] 6	**1.234568**	**02**	1.234568×10^2; display rounded up.

As indicated in the above examples, display rounding occurs on the last decimal place you *specify* when you place the calculator in [SCI] mode. Specifying more than six digits to the right of the decimal will not increase the number of digits displayed to the right of the decimal beyond six. However, specifying seven or more to the right of the decimal will move rounding into the digits (held internally) that follow those allowed by the largest [SCI] display. Using the display remaining from the previous example, the following operation does not increase the number of digits in the display, but does move rounding beyond the displayed digits.*

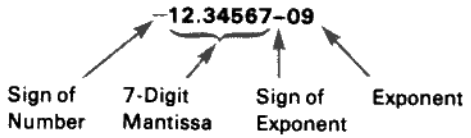
Keystrokes	Display		
[f] [SCI] 7	**1.234567**	**02**	Rounding occurs at seventh decimal place; display cannot show seventh decimal place in [SCI] mode, so no rounding occurs in the display.
[f] [SCI] 8	**1.234567**	**02**	Rounds to 8^{th} decimal place. No change in displayed digits.
[f] [SCI] 9	**1.234567**	**02**	Rounds to 9^{th} decimal place. No change in displayed digits.

* If one or more trailing 9's exist internally following the last digit allowed in the display setting, rounding may be propagated in the displayed digits for [SCI] 7 and 8 display settings. For example, 1.00000094 in [SCI] 7 will not cause rounding in the displayed version of the number, but 1.00000095 (...95 to ...99) in [SCI] 7 will cause rounding in the displayed digits.

Engineering Notation Display

ENG *(engineering)* displays numbers in an engineering notation format that operates the same as SCI notation format except:

- Engineering notation shows all exponents in multiples of three.
- The number of digits specified for the display setting refers to the number of significant digits you want to appear after the *leading digit.*



In engineering notation, the first significant digit is always present in the display. The number key you press after f ENG specifies the number of additional significant digits to which you want the display to be rounded. For example:

Keystrokes	Display		
.012345	**0.012345**		
f ENG 1	**12.**	**–03**	Engineering notation. Display is rounded to one significant digit after the leading digit. Power of 10 is proper multiple of three.
f ENG 3	**12.35**	**–03**	Display is rounded to third significant digit after the leading digit.
f ENG 6	**12.34500–03**		
f ENG 0	**10.**	**–03**	Display is rounded to first significant digit.

Notice that in ENG display mode the decimal automatically shifts to maintain the exponent of 10 as a multiple of three, as in the case of the following example:

Keystrokes	Display		
f ENG 2	**12.3**	**–03**	Display from previous example changed to ENG 2 format.
10 ×	**123.**	**–03**	Decimal shifts to maintain multiple of 3 in exponent.

Keying In Exponents

EEX *(enter exponent)* is used whenever an exponent is part of a number you are keying in. To use EEX, first key in the mantissa, then press EEX and key in the exponent. For example, divide 95,600 by Avogadro's number (6.0222×10^{26} kmol^{-1}):

Keystrokes	Display		
f FIX 4			Reset to FIX 4 display mode.
95600 ENTER	**95.600.0000**		
6.0222	**6.0222**		
EEX	**6.0222**	**00**	The 00 prompts you to key in the exponent.
26	**6.0222**	**26**	(6.0222×10^{26})
÷	**1.5875**	**–22**	kmol

To key in a number having a negative exponent of 10, first key in the number and press EEX, then press CHS *(change sign)* to make the exponent negative, and key in the exponent. For example, key in Planck's constant (6.6262×10^{-34} Joule-seconds) and multiply it by 50:

Keystrokes	Display		
6.6262 EEX	**6.6262**	**00**	
CHS	**6.6262**	**–00**	
3	**6.6262**	**–03**	
4	**6.6262**	**–34**	
ENTER	**6.6262**	**–34**	
50 ×	**3.3131**	**–32**	Joule-seconds.

Note: Decimal digits keyed into the exponent field will disappear from the display when you press EEX, but will be retained internally.

[EEX] will not operate with a number having more than seven integer digits or a decimal number having more than five zeros preceding the first significant digit. To key in such a number, use a form having a higher or lower exponent value, as appropriate. For example, $123456789.8 \times 10^{23}$ can be keyed in as $1234567.898 \times 10^{25}$; $0.00000025 \times 10^{-15}$ can be keyed in as 2.5×10^{-22}.

Mantissa. All numbers held in the calculator's stack and data storage registers are represented internally as 10-digit mantissas with a two-digit exponent. When you want to view the full ten-digit mantissa of a number held in the displayed X-register, press [f] CLEAR [PREFIX] and hold the [PREFIX] key. The mantissa of the currently displayed number will appear and remain in the display until you release the [PREFIX] key.

Keystrokes	Display
[f][π]	3.1416
[f]CLEAR [PREFIX]	
(hold)	3141592654

Rounding at the Tenth Digit

As you read earlier, your HP-11C holds every value to 10 digits internally, regardless of the number of digits specified in the current [FIX], [SCI], or [ENG] display setting. The final result of every calculation or series of calculations is rounded to the tenth digit. For example, pi and 2/3 have nonterminating decimal representations (3.1415926535... and 0.6666666666...) Because the HP-11C can provide only a finite approximation of such numbers (10 digits), a small error due to rounding can occur in the tenth digit. This error can be increased through lengthy calculations, but in the majority of cases, it does not enter the practical range of significant digits for a particular application. Accurately assessing the effects of roundoff error for a given calculation requires the use of numerical analysis methods that are beyond the scope of this handbook.

Part II
HP-11C Programming

Section 5

Programming Basics

The Basics In Brief

What Is a Program?

A program is a sequence of keystrokes that is remembered by the calculator. You can execute a given program as often as you like—with just one or two keystrokes. The stack responds to instructions in a running program in exactly the same way it responds to an identical set of instructions executed from the keyboard. The answer displayed at the end of program execution is likewise the same as the one you would obtain by executing the instructions from the keyboard. No prior programming experience is necessary to learn HP-11C programming.

Why Write Programs?

Programs save you time on repetitive calculations. Once you have written the keystroke procedure for solving a particular problem and recorded it in the calculator, you need no longer devote attention to the individual keystrokes that make up the procedure. You can let the calculator solve each problem for you. And because you can easily check the procedure in your program, you have more confidence in your final answer since you don't have to worry each time about whether or not you have pressed an incorrect key.

The following information covers the operation of your HP-11C's programming features. For guidelines that can help you in planning and developing your programs, refer to page 206, Structure, in part III of this handbook.

Program Control

Automatic Memory Reallocation

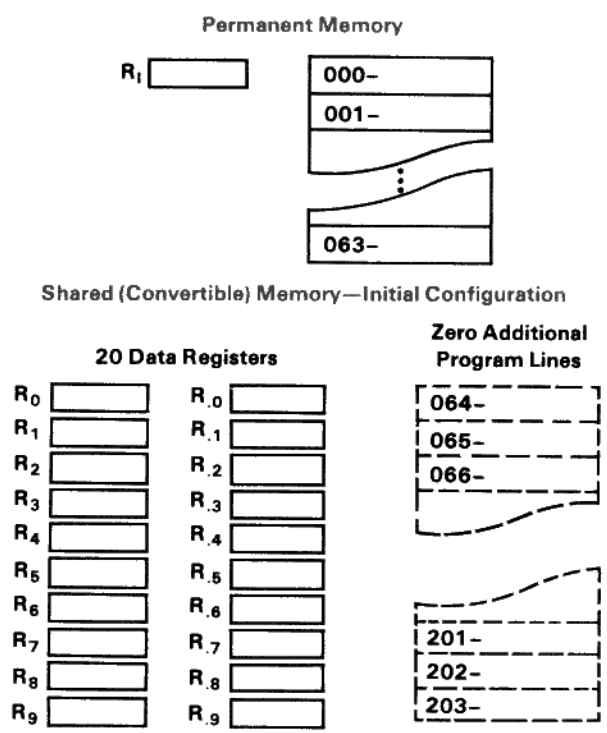
The HP-11C's reallocation of memory space between program memory and data storage is controlled automatically by the calculator. Because of this internal control, memory reallocation

does not affect the display or your keyboard operations. For this reason, you need be concerned only with:

1. What causes reallocations, and
2. What happens in memory when a reallocation takes place.

When program memory is cleared, or when Continuous Memory is reset, the calculator's memory configuration is 20 data storage registers (plus the I-register) and 63 lines of available program memory.

HP-11C MEMORY CONFIGURATION



As instructions are keyed into program memory, they are stored sequentially in the available space. If all 63 lines of the initial program space are already occupied and you key in a 64th program instruction, data storage register $R_{.9}$ is automatically reallocated to seven more lines of program memory to make room for the new instruction (plus up to six more). The HP-11C's memory configuration would then be 70 lines of program memory and 20 data storage registers. Keying in a 71st program instruction automatically reallocates storage register $R_{.8}$ to seven additional lines of program memory. This pattern can be repeated until all 20 convertible data storage register ($R_{.9}$ through $R_{.0}$, R_9 through R_0) have been reallocated to program memory.

If all convertible data storage registers are reallocated to program memory, the memory configuration will be 203 program lines and one data storage register (the Index—R_I—register). The following table shows the allocation of the lines of program memory to their respective storage registers.

$R_{.9}$	064—070	R_9	134—140
$R_{.8}$	071—077	R_8	141—147
$R_{.7}$	078—084	R_7	148—154
$R_{.6}$	085—091	R_6	155—161
$R_{.5}$	092—098	R_5	162—168
$R_{.4}$	099—105	R_4	169—175
$R_{.3}$	106—112	R_3	176—182
$R_{.2}$	113—119	R_2	183—189
$R_{.1}$	120—126	R_1	190—196
$R_{.0}$	127—133	R_0	197—203

Storage Register/Program Memory Allocation

Deleting lines of program memory one by one from any location in program memory causes the calculator to automatically reallocate

program memory to data storage registers in the reverse of the order described above.

Data Storage to Program Memory Conversion Order

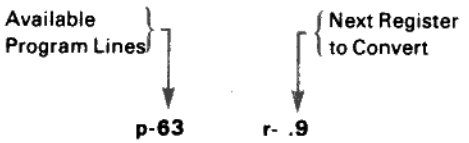
$R_{.9}\,R_{.8}\,R_{.7}\ldots R_{.1}\,R_{.0}\,R_9\,R_8\ldots R_1\,R_0$

Program Memory to Data Storage Conversion Order

For additional text and illustrations describing Automatic Memory Reallocation, refer to appendix C, How Automatic Memory Reallocation Operates.

[MEM]

To display the current allocation of memory at any time in or out of program mode, press [g], then press and hold [MEM]. While you hold [MEM], the calculator will display (1), the number of available program lines to fill before conversion of the next storage register and (2), the name of the next storage register to be converted. (For a more graphic coverage of [MEM] operation, refer to appendix C, How Automatic Memory Reallocation Operates.

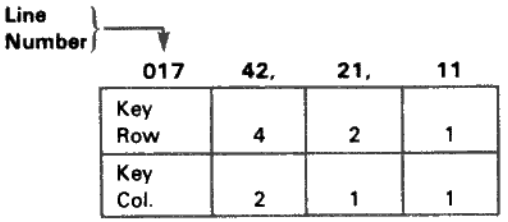


[MEM] Display With Program Memory Cleared

Keycodes and Line Numbers

When you place the calculator in Program mode and key in a program instruction, the keycode for the keys you pressed and the line number of the complete instruction will appear in the display. The keycode for any program instruction will have one, two, or

three elements, depending upon whether the instruction required one, two, or three keystrokes. Each element in a keycode is composed of two digits that describe the row/column matrix position of the key represented by the element (except numeric keys, which are represented by a single digit element).



Abbreviated Key Sequences

In Run or Program modes, the f prefix keystroke you would expect to include in the keystroke sequences for some instructions is not needed. (An unnecessary f prefix keystroke pressed as part of a program instruction will not appear in the keycode for that instruction.) For example, pressing STO RAN# will produce the same results as pressing STO f RAN#. References to other keys that can be used in abbreviated key sequences are included in the appropriate sections.

Program Control Functions

Program/Run. Pressing g P/R switches the calculator between Program and Run modes. When the calculator is in Program mode, the **PRGM** annunciator appears in the display and program instructions can be inserted or deleted. In Run mode either programs stored in program memory or individual keyboard functions can be executed.

Clear Program Memory. Pressing f CLEAR PRGM in *Program* mode clears all programs from Continuous Memory and automatically reallocates Continuous Memory to 21 data storage registers and 63 lines of available program memory. Pressing f CLEAR PRGM in *Run* mode resets the calculator to line 000 but does not clear program memory.

Go to Line 000. Pressing GTO · 000 in Program or Run mode sets the calculator to line 000 (top of program memory).

Labels. The HP-11C's labels are *addresses* for programs, program branches, and program subroutines. The alpha labels (A through E) and the numeric labels (0 through 9) are keyed into program memory by pressing f LBL *(label)* and the desired alpha or numeric key. With the calculator in Run mode, a program addressed by an alpha label is executed by pressing the f shift key and the label key. Labels 0 through 9 can also be used to address programs, but are usually reserved for program subdivisions (branches and subroutines). Numeric labels can be executed by pressing GSB and the desired number key.

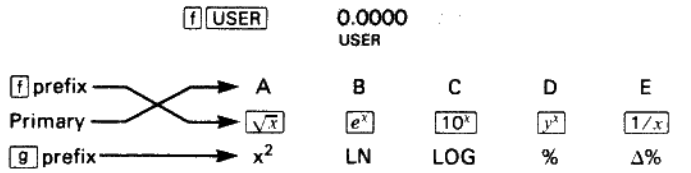
Return. The RTN *(return)* instruction, when used to end a program, causes program execution to transfer to line 000 and halt.

Run/Stop. When encountered in a running program, R/S *(Run/Stop)* causes program execution to halt. When a program is halted, pressing R/S causes program execution to begin with the line of program memory the calculator is currently positioned to.

Pause. When f PSE *(pause)* is encountered in a running program, execution halts temporarily (approximately one second) to allow viewing of the number currently in the displayed X-register. Program execution then resumes.

User Mode

User mode is a convenience feature you can use to save keystrokes during program operations. Pressing f USER exchanges the primary math and the alternate f prefix alpha key assignments on the calculator's top row keys. While USER mode is set, the **USER** annunciator appears in the display.



In Run mode, this exchange enables you to execute any programs labeled A through E by pressing only the appropriate alpha

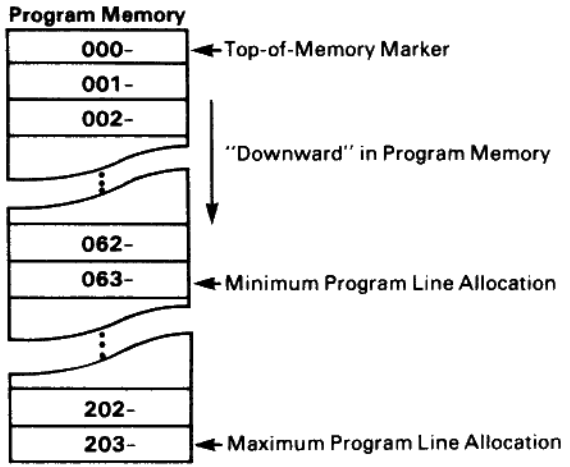
function key instead of having to first press the [f] shift key.

> Note: To avoid inadvertently executing or programming a wrong function, User mode should be activated only when specifically desired.

To deactivate User mode, press [f] [USER] again.

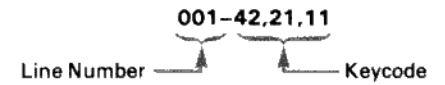
Program Memory

As you may remember from the heat loss program you keyed in at the beginning of this handbook, the keystrokes used to calculate a solution manually are also used when you write a program to calculate the solution automatically. These keystrokes are stored in the calculator's *program memory*. Press [GTO] [.] 000 now to return the calculator to the top of program memory. If you have not already done so, set the calculator to Program mode by pressing [9] [P/R]. (Remember, whenever the calculator is in Program mode, the **PRGM** annunciator will be visible in the display.) The display should now show 000-, which is the top of (program) memory marker.

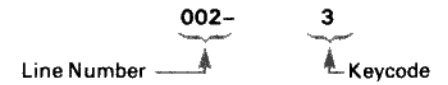


Program memory is separate from the stack, LAST X, R_I, and any data storage registers that have not been converted to lines of program memory.

When the calculator is in program mode, the number that you see on the left side of the display indicates the line number in program memory to which the calculator is set. Press [f] CLEAR [PRGM], then [f] [LBL] [A]—the first keystroke of the heat loss program (page 12)— and the display will change to:



The calculator is now set to line 001 of program memory, as indicated by the **001** that you see on the left side of the display. The other numbers in the display are keycodes for the keystrokes that have been loaded into that line of program memory. Press 3. Your display shows:

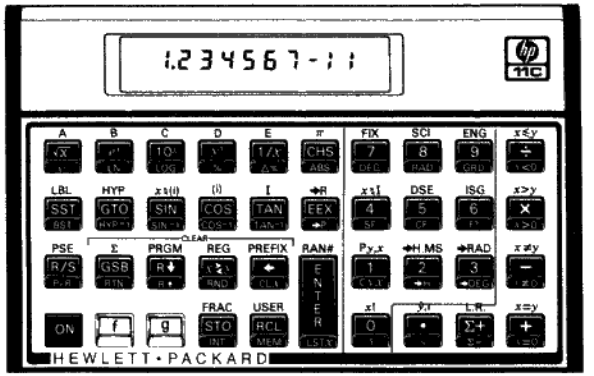


The **002** on the left side of the display indicates that you are now at line two of the program.

Each line of program memory "remembers" a single program instruction, whether that instruction consists of one, two, or three keystrokes. Thus, one line of program memory might contain a single-keystroke instruction like [CHS], while another line of program memory could contain the three-keystroke instruction [STO] [+] 6 (adds the number in the displayed X-register to the number in R_6). The keystrokes in program instructions are represented in the calculator by keycodes.

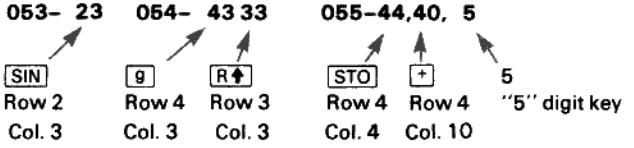
Interpreting Keycodes

Most keycodes for HP-11C key positions are determined by a simple row/column matrix. The key rows are numbered 1 through 4. The key columns are numbered 1 through 10. (The tenth column is represented in HP-11C keycodes as a 0, for example, "20" represents row 2, column 10, and corresponds to the [×] key. The only key positions which do not conform to the matrix code are functions assigned to the 0 through 9 digit keys. The codes for functions on these keys are simply the single digit on the face of the key.

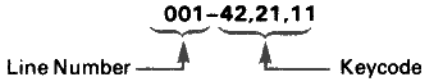


Single-Digit Keycodes

A program line can consist of one to three keycode elements. For example:



Let's take another look at the program instructions we just keyed in (page 81). Press [g] [BST]. Your display will now show the first line of the heat loss program:



The number code 001- designates the line number of program memory. The next digit pair, 42, represents [f] (row 4, column 2); 21 represents [LBL] (row 2, column 1); 11 represents [A] (row 1, column 1). In this manner all programmable keystrokes except functions assigned to digit keys are represented by a two-digit keycode. Let's see an example. Press [SST] once. Your HP-11C's display will now show the keycode for the second instruction of the heat loss program:

We know that 002- is the program line number. The "3" denotes, in this case, the number 3. The box at the right shows how the keycode would change when the "3" key is used with or without the [f] and [g] prefix keys.

Keys		Keycode
[f] [→RAD]	→	42 3
3	→	3
[g] [→DEG]	→	43 3

The remaining keystrokes for the heat loss program are shown below with their corresponding displays. Press each key in turn and verify the keycodes shown in the display.

Keystrokes	Display		
0	003-	0	The "0" digit key.
[×]	004-	20	The 2nd row, 10th key.
[·]	005-	48	The 4th row, 8th key.
4	006-	4	The "4" digit key.
7	007-	7	The "7" digit key.
[×]	008-	20	The 2nd row, 10th key.
[g] [RTN]	009-	43 32	Denotes end of program.
[g] [P/R]			Sets calculator to Run mode.

Programming Operations

The preceding topic, keycodes, covered individual program instructions. Now let's take some time to examine the details of a complete programming process. The following paragraph describes a new program we can create to help illustrate the steps involved in programming.

If you want to manually calculate the area of a circle using the formula $A = \pi r^2$ you could first key in the radius r, then square it by pressing [g] [x^2]. Next you would place π in the display by pressing

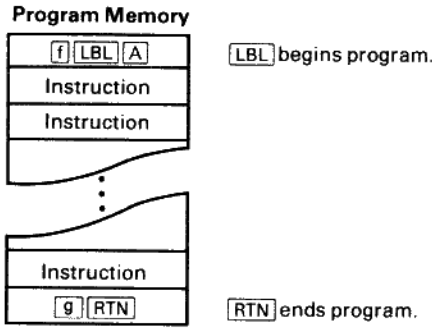
[f] [π]. Finally you would multiply the squared radius and π together by pressing [×]. The resulting keystroke sequence is shown below.

[g] [x^2]
[f] [π]
[×]

Beginning and Ending a Program

To define the beginning of a program use an [f] [LBL] *(label)* instruction followed by one of the alpha or numeric keys to specify *which* label. The use of labels permits you to have several different programs or parts of programs loaded into the calculator at any time, and to run them in the order you choose.

To define the end of a program, you can use a [g] [RTN] *(return)* instruction. In a running program, a [RTN] instruction used in this manner causes the calculator to immediately transfer execution to line 000 and halt.



Note: When a running program encounters the end of *occupied* program memory, the effect is the same as if a [g] [RTN] had been encountered. This means that if your *last* instruction in occupied program memory will be a [g] [RTN], it can be eliminated, saving you one line of memory space.

The Complete Program

The complete program to calculate the area of any circle given its radius is:

[f] [LBL] [A]	Assigns name to and defines beginning of program.
[g] [x^2]	Squares the radius you input (stack does not lift or drop).
[f] [π]	Summons π into the display (stack lifts).
[×]	Multiplies the squared radius by π (stack drops) and displays answer.
[g] [RTN]	Defines end of program; calculator returns to line 000 and halts.

Loading a Program

A program can be loaded in memory ahead of or after other programs already in memory. If a new program is loaded ahead of an existing program (by going to line 000 and keying in the new instructions), the existing program will be bumped downward in memory, one line at a time, as you key in the new program's instructions.

To prepare for loading the preceding Area of a Circle program:

1. Press [g] [P/R] to toggle into Program mode. The PRGM annunciator will appear in the display.
2. Press [f] CLEAR [PRGM] to clear program memory of previous, unwanted programs. (If you want to save a program already in the calculator, press [GTO] [·] 000 instead of [f] CLEAR [PRGM]. [GTO] [·] 000 sets the calculator to line 000 without affecting the contents of program memory.)

You can tell that the calculator is at the top of program memory because the digits 000 appear at the left of the display.

The keys you press to load the program to calculate the area of a circle are:

[f] [LBL] [A]
[g] [x^2]
[f] [π]
[×]
[g] [RTN]

Press the first key, [f], of the program.

Keystroke	Display
f	000-

You can see that the display of program memory has not changed. It will not change until you press all of the keys required for the complete instruction. Now press the rest of the keys for the first instruction.

Keystrokes	Display	
LBL	000-	
A	001-42,21,11	f LBL A loaded into program memory.

When a new program memory line number and keycode appear in the display, they indicate that a complete operation has been loaded into that line. Remember, nothing is loaded into program memory until a complete instruction (whether one, two, or three keystrokes) has been keyed in.

Now load the remainder of the program by pressing the following keys.

Keystrokes	Display	
g x^2	002-	43 11
f π	003-	42 16
×	004-	20
g RTN	005-	43 32

The program for solving the area of a circle given its radius is now loaded into your HP-11C's program memory.

Running a Program

Programs are executed in Run mode only. To prepare to run the Area of a Circle program you loaded in the preceding example, set the HP-11C to Run mode now by pressing g P/R.

To run a program, you need only key in any required data and press f and the alpha key (A through E) that labels your program. To run the circle area program, key in the radius data and press f A.

Example Execution. Calculate the areas of circles having radii of 7.5 centimeters, 9 inches, and 15.3 meters:

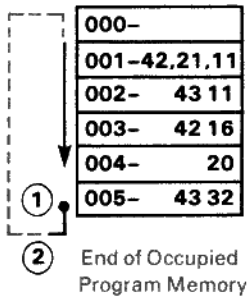
Keystrokes	Display	
7.5 f A	176.7146	Square Centimeters
9 f A	254.4690	Square Inches
15.3 f A	735.4154	Square Meters

How the Calculator Searches For a Label. When you set the HP-11C to Run mode, the calculator was positioned at line 005 of program memory (the last line you filled with an instruction when you were loading the program.) When you pressed f A, the calculator began to *search* sequentially downward through program memory, beginning with line 005, for a LBL A instruction. When the calculator searches, it does not execute program instructions.

Because:

1. Line 005 did not contain the LBL A instruction, and
2. No further lines of program memory were occupied,

your HP-11C returned to line 000 and resumed searching downward through program memory. When the calculator found the f LBL A instruction at line 001 it then began *executing* your program.



Executing Program Instructions. The calculator executes the instructions in the order you keyed them in, performing the g x^2 operation in line 002 first, then f π in line 003, etc., until it executes a g RTN instruction, a R/S *(run/stop)* instruction, or encounters the end of occupied program memory. Since there is a g RTN instruction in line 005, execution returns to line 000 and halts. The result of the calculation, the value in the X-register, is

then displayed. In programs having lengthier execution times, running will flash in the display while execution is in progress.

Non-programmable Functions. When the calculator is in program mode (**PRGM** annunciator displayed) almost every function on the keyboard can be recorded as an instruction in program memory. However, the following keyboard instructions are designed for use as non-programmable functions.

f CLEAR PRGM	g P/R	SST
f CLEAR PREFIX	g MEM	←
GTO · *nnn*	· / ON	g BST
f USER		

User Mode Operation

Let's set the calculator to User mode now and run the Area of a Circle program you just ran in the preceding example, then execute the keyboard functions affected by User mode.

Keystrokes	Display	
f USER	**User**	Activates User mode; **USER** annunciator appears.
7.5 A	**176.7146**	In User mode A through E become the primary functions of their respective keys.
9 A	**254.4690**	
15.3 A	**735.4154**	
4 f $\sqrt{x}$	**2.0000**	
1 f e^x	**2.7183**	In User mode, the top row math functions become the alternate f prefix functions of their respective keys.
1 f 10^x	**10.0000**	
2 ENTER	**2.0000**	
8 f y^x	**256.0000**	
· 5 f $1/x$	**2.0000**	
f USER	**2.0000**	Deactivates User mode.

Program Stops and Pauses

When programming, there may be occasions when you want a program to halt during execution so that you can key in data. Or you may want the program to pause so that you can quickly view results before the program automatically resumes running. Two keys, R/S (*run/stop*) and PSE (*pause*), are used for program interruptions.

Planned Stops During Program Execution

The R/S *(run/stop)* function can be used either as an instruction in a program or as an operation pressed from the keyboard.

When pressed from the keyboard:

1. If a program is running, R/S halts program execution.
2. If a program is stopped or not running, and the calculator is in Run mode, pressing R/S starts the program running. Execution then begins with the first line of program memory following the R/S instruction. (When R/S is pressed and held in Run mode, it displays the line number and keycode of that current line—when released, execution begins with that line.)

You can use these features of the R/S instruction to stop a running program at points where you want to key in data. After the data has been keyed in, restart the program using the R/S key from the keyboard.

Example: Universal Tins, a canning company, needs to calculate the volumes of various cylindrically shaped cans. Universal would also like to be able to record the area of the base of each can before the volume is calculated.

The following program calculates the area of the base of each can and then stops. After you have written down the result, the program can be restarted to calculate the final volume. The formula used is:

$$\text{Volume} = \text{base area} \times \text{height} = \pi r^2 \times h$$

The radius (r) and the height (h) of the can are keyed into the X-and Y-registers, respectively before the program is run.

To record this program, set the HP-11C to Program mode, then key in the following program instructions.

Keystrokes	Display	
f CLEAR PRGM	000-	Clears program memory and displays line 000.
f LBL A	001-42,21,11	
g x^2	002- 43 11	Square the radius.
f π	003- 42 16	Place π in X.
×	004- 20	Calculate the area of the base.
R/S	005- 31	Stop to record the area.
×	006- 20	Calculate the final volume.
g RTN	007- 43 32	

Set the HP-11C to Run mode. Then use the program to complete the table below:

Height	Radius	Area of Base	Volume
25	10.0	?	?
8	4.5	?	?

Keystrokes	Display	
25 ENTER	25.0000	Enter the height into the Y-register.
10 f A	314.1593	Key the radius into the X-register and calculate area. Program stops to display the area.
R/S	7,853.9816	Volume of first can is calculated.
8 ENTER	8.0000	Enter the height into the Y-register.

Keystrokes	Display	
4.5 f A	63.6173	Key the radius into the X-register and calculate area. Program stops to display the area.
R/S	508.9380	Second volume is calculated.

With the height in the Y-register and the radius in the X-register, pressing f A in Run mode calculates the area of the can's base; the program stops at the first R/S instruction encountered. Pressing R/S calculates the volume of the can. Program execution then returns to line 000 and halts.

For a discussion of techniques for data inputs in programs, refer to page 212, Data Input, and page 217, the User-Definable Keys, in part III of this handbook.

Pausing During Program Execution

An f PSE instruction executed in a program interrupts program execution to display results momentarily before execution is resumed. The length of the pause is about 1 second, but you can use more than one consecutive f PSE instruction to lengthen the pause duration.

To see how f PSE can be used in a program, we'll modify the cylinder volume program in the previous example. In the new program the area of the base will be briefly displayed before the volume is calculated. This example will also show how different programming approaches can be taken to solve the same problem.

To key in the program, set the HP-11C to Program mode. Press f CLEAR PRGM to clear program memory and display line 000. Then key in the following program instructions.

Keystrokes	Display	
f CLEAR PRGM	000-	
f LBL A	001-42,21,11	
g x^2	002- 43 11	Squares the radius in X.

Keystrokes	Display		
f π	003–	42 16	Places π in X.
×	004–	20	Calculates the area of the base.
f PSE	005–	42 31	Pauses to show the base area for one second.
×	006–	20	Calculates final volume of can.
g RTN	007–	43 32	

This program also assumes the height has been entered into the Y-register and the radius has been keyed into the X-register. If you have stored the instructions, set the HP-11C to Run mode. Now complete the table below using the new program.

Height	Radius	Area of Base	Volume
20	15	?	?
10	5	?	?

Keystrokes	Display	
20 ENTER	**20.0000**	Enter the height into the Y-register.
15 f A	**706.8583**	Key the radius into the X-register and calculate. Area of base is displayed for 1 second.
	14,137.1669	Program stops, displaying the volume.
10 ENTER	**10.0000**	Enter the second height into Y.
5 f A	**78.5398**	Key the radius into the X-register and calculate. Area of base is displayed for 1 second.
	785.3982	Program stops, displaying the volume.

Unexpected Program Stops

At times a mistake of some kind in your program will stop program execution. To help you determine why the calculator stopped in the middle of a program, possible reasons are listed below.

Executing g RTN. Unless in a subroutine, whenever g RTN is executed in a program, the calculator immediately returns to line 000 and halts.

Encountering the End of Program Memory. When the final instruction in program memory is not GTO, GSB, RTN or R/S, and is not in a subroutine, a running program will encounter the end of occupied program memory, transfer immediately to line 000, and halt.

Pressing Any Key. Pressing any key halts program execution. The calculator has been designed so that program execution will *not* halt in the middle of a digit entry sequence. If you press any key while a number is being placed in the X-register by a running program, the entire number will be "written" and the following line will be executed by the program before the calculator halts.

When a program is halted, you can resume execution by pressing R/S from the keyboard in Run mode. When you press R/S, the program resumes execution where it left off as though it had never stopped at all.

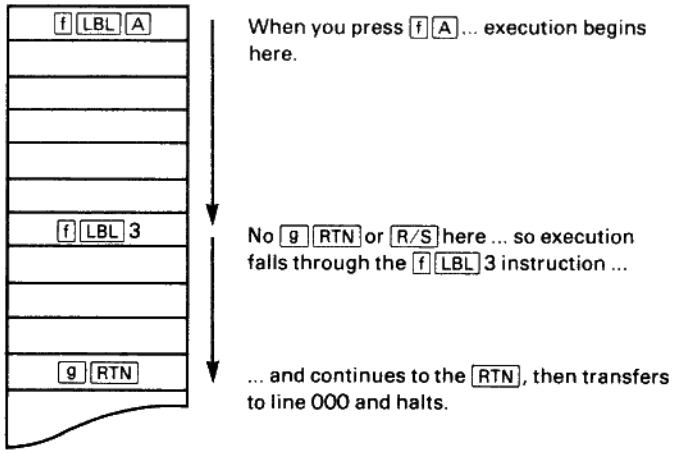
Error Stops. If the calculator attempts to execute any error-causing operation (refer to appendix A, Error Conditions) during a running program, execution immediately halts and the calculator displays the word **Error** and a number. To see the line number and keycode of the error-causing instruction, you can set the HP-11C to Program mode by first pressing any key to clear the error message, then pressing g P/R.

If an attempted storage register arithmetic operation would result in overflow in a storage register, the calculator halts and displays **Error 1**. The number in the affected storage register remains unchanged from its previous value. When you clear the error message, the last number in the display returns.

If the result of a calculation is a number with a magnitude less than $1.000000000 \times 10^{-99}$, zero will be substituted for that number and a running program will continue to execute normally. This is known as an underflow.

Labels

As you read earlier, the labels in your programs act as addresses—they tell the calculator where to begin or resume execution. Notice that when a label is encountered as part of a program, execution merely "falls through" the label and continues onward. For example, in the program segment shown below, if you press [f] [A], execution would begin at [f] [LBL] [A] and continue downward through program memory, on through the [f] [LBL] 3 instruction, until the [RTN] was encountered and execution returned to line 000 and halted.



Problems

1. Write and load a program that converts temperature in degrees Celsius to Fahrenheit, according to the formula $F = 1.8°C + 32$. Define the program with [f] [LBL] [A] and [g] [RTN] and run it to convert Celsius temperatures of -40°, 0°, and 72°.

   Answers: -40.0000°F, 32.0000°F, 161.6000°F.

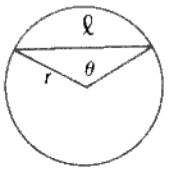
2. Create a program to calculate the length of a chord ℓ subtended by angle θ on a circle of r radius using the equation

$$\ell = 2r \sin \frac{\theta}{2}.$$

   Design your program for an r, θ order of data entry.

   Define this new program with [f] [LBL] [B] and use it to complete the following table:

r (meters)	θ	ℓ
25	30	?
50	45	?
100	90	?



   (Answers: 12.9410 meters, 38.2683 meters, 141.4214 meters.)

3. What are the program keycodes for the following instructions? [g] [%], [f] [→RAD], [STO] [+] 1, [SIN], [f] [LBL] [D], [STO] [RAN#].

   Answers: 43 14; 42 3; 44, 40, 1; 23; 42, 21, 14; 44 36*.

4. How many lines of program memory are required to load the following sections of programs?

   a. 2 [ENTER] 3 [+].
   b. 10 [STO] 6 [RCL] 5 [×].
   c. 100 [STO] 1 50 [STO] [×] 1 [RCL] 2 [f] [π] [÷].

   Answers: a, 4; b, 5; c, 10.

---

* Refer to page 78, Abbreviated Key Sequences.

Section 6

Program Editing

Editing In Brief

Even the most experienced programmers find errors in their programs. These errors range from mistakes in the original formulas to mistakes in recording the program. Whenever errors occur they need to be found and corrected. Your HP-11C is designed to make this error-checking process as easy as possible.

A program that uses storage registers or status settings (such as flags or trigonometric modes) may give rise to errors if such information is incorrect when you run the program. However, you can eliminate the possibility of this type error using an *initializing*—clearing or resetting—procedure. One method of initializing a program is to execute the necessary clearing and resetting instructions from the keyboard before beginning program execution. Another method is to make the program self-initializing by including the necessary instructions at the beginning of the program.

Finding Program Errors

One of the easiest ways to help verify that a program is working properly is to run a test case in which you either know the answer or the answer can be easily determined. Another option is to test a program for proper responses at its intended limits of applications and accuracy. For some types of calculations, you may even want to test the program for proper responses to illegal data. To help locate any potential problems in a self-initializing program, try running the program several times with all data registers loaded with meaningless data, in different trig modes, and with the flags set, and then cleared.

The editing features of your HP-11C have been designed to provide you with quick and easy access to any part of a program, whether for editing, debugging, or documentation. If a program stops running because of an error or because of an overflow, you can simply clear the error message, then switch the calculator to

Program mode to see the line number and keycode of the operation that caused the error or overflow. If you suspect that a portion of your program is faulty you can check execution step by step, then use the other editing features to make any necessary changes.

Editing Functions

Your HP-11C's function set includes the following four nonprogrammable editing and manipulation functions to aid you in modifying and correcting your programs:

[SST] [BST] [GTO] [.] *nnn* [←]

[SST] *(Single-Step).*

In Program Mode:

When you press and release [SST] the calculator moves to and displays the next line in occupied program memory. If you press and hold [SST] in program mode, the calculator will continuously scroll through the instructions in program memory until you release the [SST] key. When using [SST] in program mode, no program instructions are executed.

```
000-
  [SST]
001-42,21,11
```

In Run Mode:

When you press [SST] the calculator moves to and displays the next line of program memory. When you release [SST], the calculator executes the instruction loaded in that line.

[BST] *(Back Step).* Pressing [g] [BST] causes the calculator to step or scroll backwards through program memory in the same way that [SST] causes the calculator to step or scroll forward. (No program instructions are executed.)

```
002-  43 13
  [g] [BST]
001-42,21,11
```

[GTO] [.] *nnn (Go To Line nnn).* Pressing [GTO] [.] *nnn* in Program or Run mode causes the calculator to go to the program line number

specified by *nnn*. (In Run mode only, pressing [GTO] and an alpha or numeric label name causes the calculator to go to the specified label in the same way that [GTO] [·] *nnn* causes the calculator to go to a specified line number.)

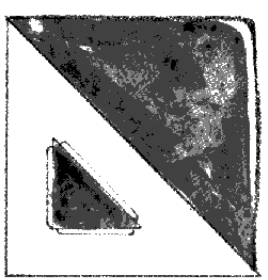
[←] ***(Backarrow).*** Pressing [←] in Program mode deletes the displayed program instruction from program memory. Any program instructions following the deleted instruction will then automatically move upward one line and be renumbered. Pressing [←] in Run mode does not affect program memory, but does affect the contents of the displayed X-register (refer to Display Clearing: [CLx] and [←], page 17).

Example Program Memory

	Before Pressing [←]:		After Pressing [←]:	
	001-42,21,11	→	001-42,21,11	(Display)
(Display)	002- 43 13	→	002- 23	
	003- 23	→	003- 45 0	
	004- 45 0	→	004- 45 1	
	005- 45 1			

Editing Example

To provide an example for editing, load the following Pythagorean theorem program for calculating the length of the hypoteneuse (side c) of a right triangle, given the lengths of sides a and b. The formula used is $c = \sqrt{a^2 + b^2}$. Assume that the calculation begins with side a in the Y-register and side b in the displayed X-register.



To begin, set the HP-11C to Program mode.

Keystrokes	Display		
[f] CLEAR [PRGM]	000-		Clears program memory.
[f] [LBL] [E]	001-42,21,15		Labels the program.
[g] [x^2]	002-	43 11	Squares side 2 (b^2).
[$x \gtrless y$]	003-	34	Moves b^2 from displayed X-register to Y-register and a from Y-register to displayed X-register.
[g] [x^2]	004-	43 11	Squares side 1 (a^2).
[+]	005-	40	$(a^2 + b^2)$.
[$\sqrt{x}$]	006-	11	$\sqrt{(a^2+b^2)}$.
[g] [RTN]	007-	43 32	Terminates the program.

Set the HP-11C to Run mode.

To test the program, calculate the hypoteneuse of a right triangle with side a of 22 meters and side b of 9 meters.

Keystrokes	Display	
22 [ENTER]	22.0000	Key in a.
9	9	Key in b.
[f] [E]	23.7697	Length in meters of the hypoteneuse.

Single-Step Execution of a Program

In longer programs a wrong test-case answer will seldom pinpoint a mistake. For these cases, you can slow down program execution by using [SST] in Run mode.

Single-step execution begins with the line the calculator is currently positioned to. Since the [RTN] instruction at the end of the program positioned the calculator to line 000 after you ran the program, we need only key in our initial values to begin single-step execution. Each time you press [SST], hold the key down momentarily to view the line number and keycode of the next instruction to be executed; then release the [SST] key to execute the instruction.

Keystrokes	Display	
22 [ENTER]	22.0000	Load *a* into the Y-register.
9	9	Load *b* into the X-register.
[SST]	001-42,21,15	While [SST] is held, keycode in line 001 (for [f] [LBL] [E]) is displayed.
	9.0000	When [SST] is released, the label in line 001 is executed.
[SST]	002- 43 11	Keycode for [x^2].
	81.0000	[SST] released, [x^2] executed.
[SST]	003- 34	Keycode for [$x \geq y$].
	22.0000	[$x \geq y$] executed.
[SST]	004- 43 11	
	484.0000	[g] [x^2] executed.
[SST]	005- 40	
	565.0000	[+] executed.
[SST]	006- 11	
	23.7697	[$\sqrt{x}$] executed.
[SST]	007- 43 32	
	23.7697	[g] [RTN] executed. Program completed; calculator returns to line 000 and halts.

Note: [SST] will not advance into unoccupied lines of program memory. If you single-step from the last *occupied* line of program memory, and the last instruction is not [GTO] *(go to)* or [GSB] *(go to subroutine)*, the calculator will "wrap around" to line 000. If the last instruction is a [GTO] or [GSB] the calculator will execute the [GTO] or [GSB] instruction by moving to the designated position in program memory.

Using [SST] and [BST] in Program Mode

Line by Line Stepping Through Program Memory. In Program mode, when you press and *release* [SST], the calculator moves to and displays the next line of occupied program memory. No instructions are executed. ([g] [BST] operates in the same way, except that the previous line of program memory is displayed.)

Keystrokes	Display	
[g] [P/R]	000-	Set the HP-11C to Program mode.
[SST]	001-42,21,15	
[SST]	002- 43 11	
[SST]	003- 34	
[SST]	004- 43 11	
[SST]	005- 40	
[SST]	006- 11	
[SST]	007- 43 32	
[g] [P/R]		Set the HP-11C to Run mode.

Scrolling Through Program Memory. In Program mode, pressing and *holding* [SST] or [BST] scrolls the calculator forward or backward through program memory. (The calculator displays the current program line for approximately two seconds, then displays each succeeding line for approximately one-half second.)

Modifying a Program

To illustrate how to use the HP-11C's editing features, let's modify the Pythagorean Theorem program shown on page 99 so that the X-register contents will automatically be displayed at certain points in the program. We will do this by inserting an [f] [PSE] *(pause)* instruction in the indicated locations in program memory.

[f] [LBL] [E]	001-42,21,15	
[g] [x^2]	002- 43 11	We will insert an [f] [PSE] instruction after these three instructions.
[$x \geq y$]	003- 34	
[g] [x^2]	004- 43 11	
[+]	005- 40	
[$\sqrt{x}$]	006- 11	
[g] [RTN]	007- 43 32	

Inserting Instructions. A new program instruction can be inserted anywhere from line 000 through the end of occupied program memory. To insert a new instruction:

1. Set the calculator to Program mode.
2. Move to the program line number immediately *preceding* the position where you want to insert the new instruction.
3. Key in the new instruction. It will be loaded into the *next* line of program memory. All subsequent instructions will be "bumped" down (renumbered) one line in program memory.

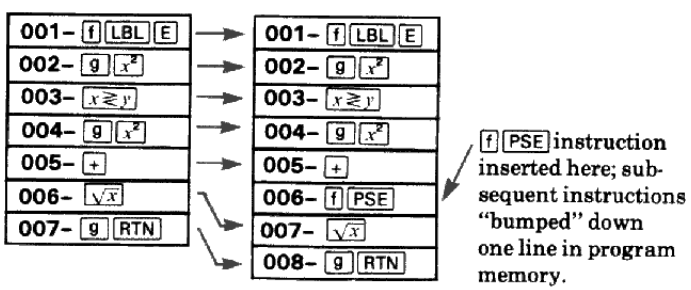
Because each new instruction you insert causes all subsequent instructions to be renumbered, inserting changes in a program is simplified by beginning with the change farthest from line 000, then working back toward the beginning of the program.

Example. To insert a [PSE] instruction after the [+] instruction:

Keystrokes	Display		
[GTO] [·] 000			Set the calculator to program line 000.
[g] [P/R]	000–		Line 000.
[SST]	001–	42,21,15	Go to line 005.
[SST]	002–	43 11	
[SST]	003–	34	
[SST]	004–	43 11	
[SST]	005–	40	Line 005.
[f] [PSE]	006–	42 31	[PSE] instruction inserted at line 006.

With the calculator set at line 005, when you pressed [f] [PSE], program memory was altered...

...from this... ...to this.

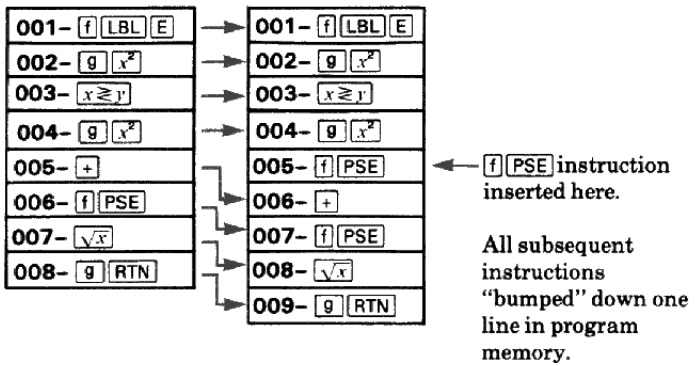


Going To a Line Number. It is easy to see that if you wanted to single-step or scroll to some remote line in program memory it could take an inconvenient amount of time. However, using the [GTO] [·] *nnn* procedure you used previously to jump to line 000, you can avoid unnecessary waiting. When you press [GTO] [·] and the desired three-digit line number in occupied program memory, the calculator immediately jumps to the line number specified by the three digits. (You can use [GTO] [·] *nnn* in Program or Run mode—no program instructions are executed.) For example, to go to line 004 to insert a [PSE] instruction after the second [g] [x²] instruction:

Keystrokes	Display		
[GTO] [·] 004	004–	43 11	Go to line 004.
[f] [PSE]	005–	42 31	Insert [PSE] after line 004.

When you inserted the [f] [PSE] instruction after line 004, program memory was altered...

...from this... ...to this.



Now we will use [BST] to insert the last [f] [PSE] in our example.

Keystrokes	Display		
	005-	42 31	Current position in program memory.
[g] [BST]	004-	43 11	
[g] [BST]	003-	34	
[g] [BST]	002-	43 11	Insert the last [PSE] instruction after this line.
[f] [PSE]	003-	42 31	Inserts [PSE] at line 004 and "bumps" all subsequent instructions downward one line in program memory.

The modified example program now looks like the program illustrated below. If you wish, you can use [SST] in program mode to verify that the program now in your calculator matches the modified version.

Keystrokes	Display
[f] [LBL] [E]	001-42,21,15
[g] [x^2]	002- 43 11
[f] [PSE]	003- 42 31
[$x \geqslant y$]	004- 34
[g] [x^2]	005- 43 11
[f] [PSE]	006- 42 31
[+]	007- 40
[f] [PSE]	008- 42 31
[$\sqrt{x}$]	009- 11
[g] [RTN]	010- 43 32

Now run the program again. Use our previous example of 22 and 9 for sides a and b.

Keystrokes	Display	
[g] [P/R]		Set the HP-11C to Run mode.
22 [ENTER]	22.0000	Side a keyed into the Y-register.
9	9	Side b keyed into the X-register.
[f] [E]	81.0000	Pause to show result of [x^2] (b^2) instruction at line 003.
	484.0000	Pause to show result of [x^2] (a^2) instruction at line 006.
	565.0000	Pause to show result of [+] ($a^2 + b^2$) instruction at line 007.
	23.7697	The final result, side c ($c = \sqrt{a^2 + b^2}$).

Inserting Instructions in Longer Programs. If all 203 program lines are *occupied*, the calculator will not accept any additional program instructions. If you attempt to insert a new instruction at any point in program memory with all 203 lines already occupied, **Error 4** will appear in the display and program memory will remain unchanged. (Refer to appendix C, How Automatic Memory Reallocation Operates.)

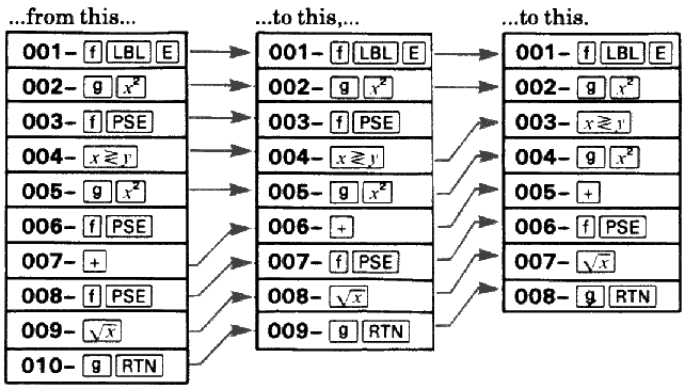
Deleting Instructions. Often in modifying or correcting a program you may wish to delete an instruction from program memory. To delete an instruction, use [GTO], [SST], or [BST] to set the calculator to the instruction, then press the nonprogrammable operation [←] with the calculator set to Program mode. (When you delete an instruction from program memory using [←], all subsequent instructions in program memory are moved *up* (renumbered) one line. The calculator then displays the line *preceding* the line that held the instruction you deleted.)

Example. To modify the Pythagorean Theorem program from the previous example so that only one pause remains (to display the

sum of the squares) you would have to delete the [f] [PSE] instructions that are presently loaded in lines 003 and 006 of program memory.

Keystrokes	Display		
[g] [P/R]	000-		Sets the HP-11C to Program mode.
[GTO] [·] 006	006-	42 31	Moves calculator to line 006.
[←]	005-	43 11	Deletes [f] [PSE] instruction; calculator displays preceding line (005). Subsequent instructions are renumbered (move up one line).
[GTO] [·] 003	003-	42 31	Moves calculator to line 003.
[←]	002-	43 11	Deletes [f] [PSE] instruction; calculator displays preceding line (002). Subsequent instructions are renumbered (move up one line).

When you deleted the [PSE] instructions at lines 006 and 003, memory was altered...



If you modified the Pythagorean Theorem program as described above, the program now pauses only once, to display the sum of the squares. The length of the hypoteneuse is then calculated and execution halts.

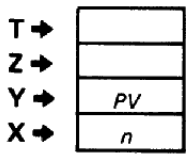
Run the program for a right triangle having sides a and b of 17 and 34 meters respectively.

Keystrokes	Display	
[g] [P/R]		Set the HP-11C to Run mode.
17 [ENTER] 34	34	
[f] [E]	1,445.0000	Pause to display sum of the squares of sides a and b.
	38.0132	Length of hypoteneuse.

When deleting instructions from a program of more than 63 lines, the process of automatically allocating storage registers to program lines works in reverse. For example, deleting *any* instruction from a 71-line program automatically converts program lines 71-77 back to storage register $R_{.8}$. (Refer to appendix C, How Automatic Memory Reallocation Operates.)

Problems

1. The following program is used by the manager of a savings and loan company to compute the future values of savings accounts according to the formula $FV = PV(1 + i)^n$, where FV is future value or amount, PV is present value, i is the *periodic* interest rate expressed as a decimal, and n is the number of periods. Assuming PV and n keyed in as shown before beginning execution, and an annual interest rate of 7.5%, the program is:



Keystrokes	Display
[f] [LBL] [A]	001-42,21,11
[f] [FIX] 2	002-42, 7, 2

Keystrokes	Display		
1	003-	1	
.	004-	48	
0	005-	0	Interest
7	006-	7	
5	007-	5	
x≷y	008-	34	
y^x	009-	14	$(1+i)^n$
×	010-	20	$PV(1+i)^n$
g RTN	011-	43 32	

a. Load the program into the calculator.

b. Run the program to find the future amount of $1,000 invested for 5 years;

(Answer: $1,435.63)

of $2,300 invested for 4 years.

(Answer: $3,071.58)

c. Alter the program to account for a change of the annual interest rate from 7.5% to 8%.

d. Run the program for the new interest rate to find the future value of $500 invested for 4 years; of $2,000 invested for 10 years.

(Answers: $680.24; $4,317.85)

2. The following program calculates the time it takes for an object to fall to the earth when dropped from a given height. (Friction from the air is not taken into account.) When the height h in meters is keyed into the displayed X-register and B is pressed, the time t in seconds the object takes to fall to earth is computed according to the formula:

$$t = \sqrt{\frac{2h}{9.8}}$$

a. Clear all previously recorded programs from the calculator, reset the display mode to FIX 4, and load the following program.

Keystrokes	Display	
f LBL B	001-42,21,12	
2	002-	2
×	003-	20
9	004-	9
.	005-	48
8	006-	8
÷	007-	10
$\sqrt{x}$	008-	11
g RTN	009-	43 32

b. Run the program to compute the time taken by a stone to fall from the top of the Eiffel Tower, 300.51 meters high; and from a blimp stationed 1000 meters in the air.

(Answers: 7.8313 seconds; 14.2857 seconds.)

c. Alter the program to compute the time of descent when the height in *feet* is known, according to the formula:

$$t = \sqrt{\frac{2h}{32.1740}}$$

d. Run the altered program to compute the time taken by a stone to fall from the top of the Grand Coulee Dam, 550 feet high; and from the 1350-foot height of the World Trade Center buildings in New York City.

(Answers: 5.8471 seconds; 9.1607 seconds.)

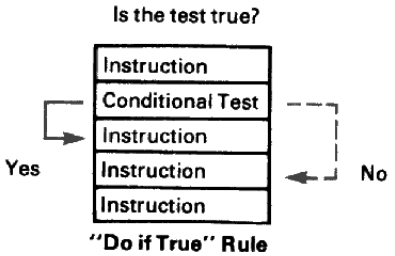
Section 7

Program Decisions and Control

Decisions and Control In Brief

Program Conditional Tests

The HP-11C's eight conditional tests are true/false tests used in programs to enable your HP-11C to make decisions. In a running program, when the result of a conditional test is true, program execution continues with the first instruction following the conditional test. When the result of a conditional test is false, execution *bypasses* the first instruction following the test and resumes with the second instruction following the test.



"Do if True" Rule

Your HP-11C's conditionals operate by comparing the value in the X-register to either the number in the Y-register or to zero in the following manner:

[f] [x≤y] tests to see if the value in the X-register is less than or equal to the value in the Y-register.

[f] [x>y] tests to see if the value in the X-register is greater than the value in the Y-register.

[f] [x≠y] tests to see if the value in the X-register is not equal to the value in the Y-register.

[f] [x=y] tests to see if the value in the X-register is equal to the value in the Y-register.

[g] [x<0] tests to see if the value in the X-register is less than zero.

[g] [x>0] tests to see if the value in the X-register is greater than zero.

[g] [x≠0] tests to see if the value in the X-register is not equal to zero.

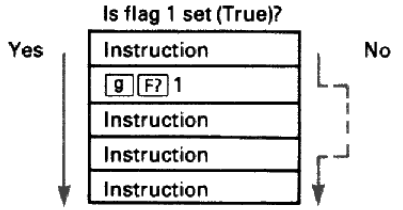
[g] [x=0] tests to see if the value in the X-register is equal to zero.

Flags

Another type of decision-making test for use in programs is a flag test. A flag is actually a status indicator that is in either a *set* (true) or *clear* (false) status. A running program can test a flag and make a decision based upon whether the flag is set or clear. Flag tests affect program execution in the same way as conditional tests.

The two flags in your HP-11C are numbered 0 and 1. To set a flag, press [g] [SF] *(set flag)* followed by the proper digit key (0 or 1) of the desired flag. To clear a flag, press [g] [CF] *(clear flag)* followed by the proper digit key. To test a flag use [g] [F?] *(is flag set?)* followed by the digit key specifying the flag to be tested.

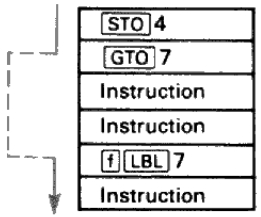
A flag that has been set by a [g] [SF] *n* command remains set until it is cleared by a [g] [CF] *n* command or until Continuous Memory is reset.



Program Control

Go To

The GTO instruction you have used earlier to position the calculator to a specific program line for viewing or editing is also used to transfer execution to a label elsewhere in program memory. When used as an instruction in a program GTO is followed by an alpha or numeric label address.



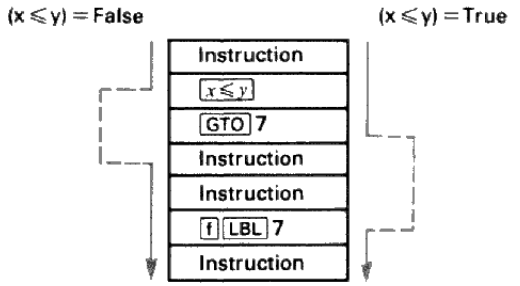
When a running program encounters a GTO *label* instruction, execution is suspended while the calculator searches downward in memory for the specified label. Program execution resumes at the first line found to contain the label.

Branching and Looping

Unconditional Branching. An unconditional branch is simply a GTO instruction that is always executed in a running program to transfer program execution elsewhere in the program, regardless of data status. The preceding illustration of GTO operation is an unconditional branch.

Conditional Branching. When GTO *label* is used in conjunction with a conditional test, as in the following illustration, the GTO *label* instruction becomes a *conditional branch*. That is, the result of the conditional test preceding the GTO determines whether or not the GTO instruction will be executed.

Conditional Branch

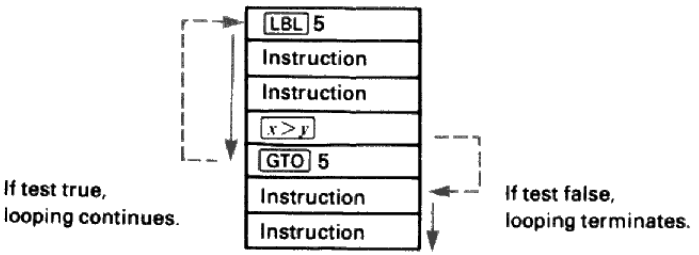


When the calculator is in Run mode, pressing GTO *label* causes the calculator to go to the specified label and halt. This feature is convenient when you want to review or edit lines of memory following a certain label instead of executing them as part of a program.

Looping. Looping is a special case of branching in which a GTO instruction is used to repeat the execution of a series of instructions one or more times. Loops are frequently used for counters and for sequentially calculating a series of results using the same set of program instructions. The continuation of a loop for a new iteration or the transfer of execution out of a loop is controlled by a conditional branch.

In the following illustration, as long as the result of the conditional test preceding the GTO is true the calculator continues successive iterations of the loop. When the result of the loop conditional test becomes false, execution skips the GTO instruction and continues with the rest of the program. Use of an unconditional branch to control a loop results in an infinite loop, that is, a loop that is reexecuted indefinitely.

Conditional Branch
Control of Program Loop



Example of Conditonal Loop Control. The following program calculates and displays the square roots of consecutive whole numbers from 1 to 10. After the square root of an integer is calculated and displayed, an [x>y] conditional test determines whether or not that integer was less than 10. If the integer was less than 10, a [GTO] 0 conditional branch is executed and the loop is repeated using the next highest integer. If the test finds that the integer was not less than 10, program execution bypasses the [GTO] 0 conditional branch and the loop is terminated. (The loop portion of the following program is included in lines 004 through 015.)

Keystrokes	Display	
[g] [P/R]	000-	Sets HP-11C to Program mode.
[f] CLEAR [PRGM]	000-	
[f] [LBL] [C]	001-42,21,13	
0	002- 0	Clears R_1 of any
[STO] 1	003- 44 1	unwanted data from previous calculations.
[f] [LBL] 0	004-42,21, 0	Begins loop.
1	005- 1	
[STO] [+] 1	006-44,40, 1	Integer counter/ increment.

Keystrokes	Display	
[RCL] 1	007- 45 1	
[f] [PSE]	008- 42 31	Displays next integer.
[√x]	009- 11	Square root of integer.
[f] [PSE]	010- 42 31	Displays square root.
[g] [LST x]	011- 43 36	Recalls copy of integer from LAST X register.
1	012- 1	Compares last integer to
0	013- 0	highest integer allowed by
[f] [x>y]	014- 42 20	the program.
[GTO] 0	015- 22 0	Conditional branch. If last integer is less than 10, go to [LBL] 0 and resume...
[g] [CLx]	016- 43 35	...otherwise, avoid another
[g] [RTN]	017- 43 32	loop, clear displayed X-register and halt program.
[g] [P/R]		Sets HP-11C to Run mode.

To run the program press [f] [C]. The calculator will display a table of integers from 1 to 10 with their corresponding square roots. When 0.0000 is displayed, program execution is completed.

How It Works: When you press [f] [C], the calculator searches through program memory until it encounters the [f] [LBL] [C] instruction. It executes that label and each subsequent instruction in sequential order through line 014, the [x>y] conditional test. If the result of the [x>y] conditional test is true—that is, the last integer used was less than 10—the following [GTO] 0 instruction is executed, the calculator returns to the [LBL] 0 instruction at line 004, and execution resumes as a new iteration of the loop. However, if the result of the [x>y] conditional test is false, meaning the last integer used was 10, the [GTO] 0 instruction is bypassed and the loop is terminated.

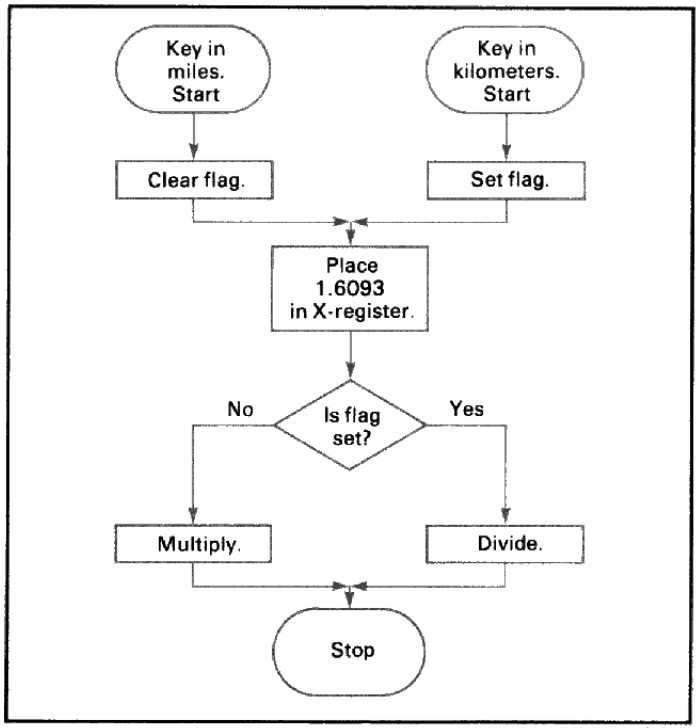
For an additional discusssion of looping and loop control techniques, refer to page 214, Looping, in part III of this handbook.

Problem. Write a program to calculate and display sales commissions. For sales of less than $100 the commission should be 10% of the sale. For sales of $100 or more the commission should be 15%. Use a conditional branch in your solution.

Using Flags

Like the $x{:}y$ and $x{:}0$ conditional tests, flags give you the capability to either skip or execute individual lines in program memory. However, while the $x{:}y$ and $x{:}0$ conditionals function by comparing values, flags function by telling the calculator status. That is, if a set flag ([g] [SF] n) instruction is placed in a program branch, testing that flag elsewhere ([g] [F?] n) can tell the calculator whether or not that branch has been executed.

Example. The following program closely approximates the conversion of an input in miles to kilometers ([f] [A]), or the conversion of an input in kilometers to miles ([f] [B]). The calculator uses the status of flag 0 to decide whether to convert a user input to kilometers (multiply input by conversion factor) or to miles (divide input by conversion factor).



Keystrokes	Display	
[g] [P/R]	000-	Sets HP-11C to Program mode.
[f] CLEAR [PRGM]	000-	Clears program memory.
[f] [LBL] [A]	001-42,21,11	Program begins with input in miles.
[g] [CF] 0	002-43, 5, 0	Flag 0 cleared ("remembers" miles input for later program control).
[GTO] 1	003- 22 1	Unconditional branch to [LBL] 1.
[f] [LBL] [B]	004-42,21,12	Program begins with input in kilometers.
[g] [SF] 0	005-43, 4, 0	Flag 0 set ("remembers" kilometers input for later in program).
[f] [LBL] 1	006-42,21, 1	[LBL] 1 begins calculation portion of program.
1	007- 1	Place conversion factor in X-register.
.	008- 48	
6	009- 6	
0	010- 0	
9	011- 9	
3	012- 3	
[g] [F?] 0	013-43, 6, 0	Flag test. If true (kilometers input), go to [LBL] 2.
[GTO] 2	014- 22 2	
×	015- 20	If flag test false (miles input), calculate conversion to kilometers and halt program.
[g] [RTN]	016- 43 32	
[f] [LBL] 2	017-42,21, 2	Calculate conversion to miles and halt program.
÷	018- 10	
[g] [RTN]	019- 43 32	
[g] [P/R]		Sets HP-11C to Run mode.

Run the program to convert 26 miles into kilometers; to convert 1.5 kilometers into miles.

Keystrokes	Display	
26	26	Input miles.
f A	41.8418	Execute Miles to Kilometers option. Result of conversion displayed.
1.5	1.5	Input kilometers.
f B	0.9321	Execute Kilometers to Miles option. Result of conversion displayed.

How It Works. When you input a value and identify the type of conversion you want by pressing f A (miles to kilometers) or f B (kilometers to miles), the calculator sets or clears flag 0, then executes the main program (beginning at label 1). After placing the conversion factor in the X-register the program tests flag 0. If the flag is set (true), execution transfers to label 2, the conversion to miles is calculated, and execution halts. If the flag is clear (false), the conditional branch to LBL 2 is bypassed, a conversion to kilometers is calculated, and execution halts. The function of the flag is to tell the calculator which of the two labels has been executed and, therefore, which conversion is desired.

For a further discussion of flag use, refer to page 215, Flags, in part III of this handbook.

Section 8

Subroutines

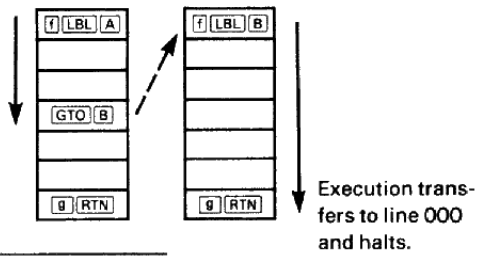
Subroutines In Brief

Often a program contains a certain series of instructions that are executed several times during a single execution of the program. When the same set of instructions occurs more than once in a program, memory space can be conserved by executing the instructions as a subroutine.

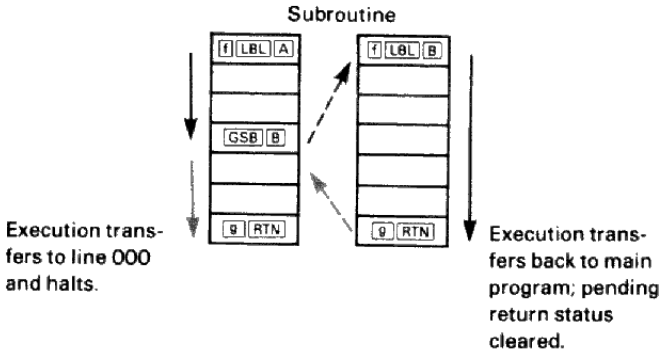
Go To Subroutine

A subroutine is executed using GSB with the desired alpha or numeric label address.* A GSB instruction transfers execution to the specified label address in the same way as GTO. However, when a GSB instruction is executed in a running program, a *pending return* condition is set in the calculator. When a pending return condition exists, the first subsequent RTN *(return)* instruction encountered by a running program will cause program execution to transfer back to the program instruction immediately following the last GSB. Execution then continues sequentially downward through program memory. (When the RTN is executed, the "pending" condition is cleared.) Compare the following illustrations of a branch and a subroutine.

Branch



* A GSB followed by an alpha label address is an abbreviated key sequence. Refer to page 78, Abbreviated Key Sequences.

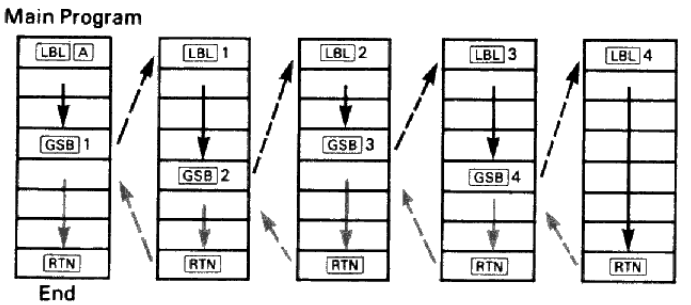


As you can see, the only difference between using GSB *(go to subroutine)* and GTO *(go to branch)* is the transfer of execution *after* the RTN.

> After a GTO, the next RTN causes execution to transfer to line 000 and halt; after a GSB, the next RTN causes execution to transfer back to the main program and resume.

Subroutine Limits

A subroutine can call up another subroutine, and that subroutine can call up yet another subroutine. This "subroutine nesting"—the execution of a subroutine within a subroutine—is limited only by the number of *returns* (RTN) that can be held pending at any one time in the HP-11C. Here is how the HP-11C operates with nested subroutines.

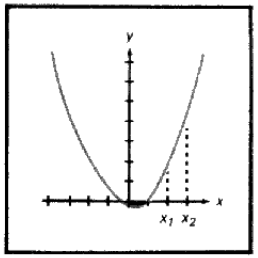


The calculator can return back to the main program from subroutines that are nested four deep, as shown. If you attempt to call a subroutine that is nested five levels deep, the calculator will halt and display **Error 5** when it encounters the instruction calling the fifth subroutine level.

> Note: While any one set of nested subroutines can be up to four levels deep, there is no limit to the number of nested subroutine sets or non-nested subroutines you can include in a program.

Subroutine Usage

Example. Write a program for calculating the average slope between x_1 and x_2 on the graph shown, where $y = x^2 - \sin x$.

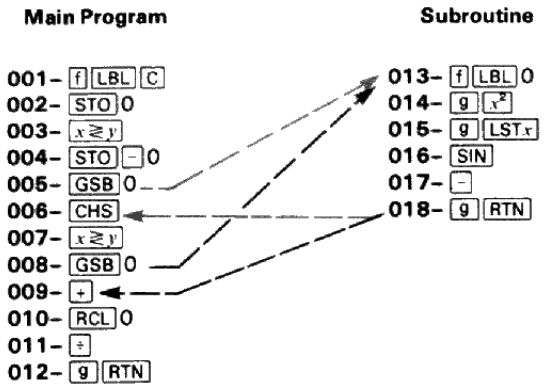


Solution: The average slope is given by the formula

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{(x_2^2 - \sin x_2) - (x_1^2 - \sin x_1)}{x_2 - x_1}$$

Notice that the solution requires two computations of the expression $x^2 - \sin x$ (once for $x = x_1$ and again for $x = x_2$).

Since the solution includes an expression that must be repeated for both values of x, we can create a subroutine to execute the repetition and save space in program memory.



(Program assumes DEG trig mode.)

When you press f C with x_1 in the Y-register and x_2 in the displayed X-register, execution begins with the LBL C instruction in line 001. When the GSB 0 instruction in line 005 is encountered, execution transfers to the LBL 0 instruction in line 013 and calculates y_1. If for example, we used a value of 2 for x_1 and a value of 3 for x_2, here is how the solution would be calculated.

	001	002	003	004
T →				
Z →				
Y →	2	2	3	3
X →	3	3	2	2
Keys →	f LBL C (x_1 in Y-reg. x_2 in X-reg.)	STO 0 (x_2 in R_0)	$x \gtrless y$ (Exchange)	STO - 0 ($x_2 - x_1$ in R_0)

	005	013	014	015
T →				
Z →				3
Y →	3	3	3	4
X →	2	2	4	2
Keys →	GSB 0 (Go to label 0)	f LBL 0 (Begin subroutine)	g x^2 (x_1^2)	g LSTx (x_1)

	016	017	018	006
T →				
Z →	3			
Y →	4	3	3	3
X →	0.0349	3.9651	3.9651	-3.9651
Keys →	SIN ($\sin x_1$)	- ($x_1^2 - \sin x_1 = y_1$)	g RTN (Return to main program)	CHS ($-(x_1^2 - \sin x_1)$)

From line 018 execution transfers back to the main program and continues with the first line after the last GSB instruction. When the GSB 0 instruction in line 008 is encountered, execution again transfers to the LBL 0 instruction in line 013.

	007	008	013	014
T →				
Z →				
Y →	-3.9651	-3.9651	-3.9651	-3.9651
X →	3	3	3	9
Keys →	$x \gtrless y$ (Exchange)	GSB 0 (Go to label 0)	f LBL 0 (Begin Subroutine)	g x^2 (x_2^2)

	015	016	017	018
T →				
Z →	-3.9651	-3.9651		
Y →	9	9	-3.9651	-3.9651
X →	3	0.0523	8.9477	8.9477
Keys →	g LST x	SIN	−	g RTN
	(x_2)	($\sin x_2$)	($x_2 - \sin x_2 = y_2$)	(Return to main program.)

After the calculator passes through the subroutine under LBL 0 a second time to compute y_2, the f RTN instruction at line 018 causes execution to return to line 009, the first instruction after the most recent GSB 0 instruction. At this point, y_2 is in the X-register; $-y_1$ is in the Y-register. The remaining instructions complete the slope calculation.

	009	010	011	012
T →				
Z →				
Y →		4.9826		
X →	4.9826	1	4.9826	4.9826
Keys →	+	RCL 0	÷	g RTN
	($y_2 - y_1$)	($x_2 - x_1$)	$\frac{y_2 - y_1}{x_2 - x_1}$	(End of program)

When calculation halts, the slope of the line between x_1 and x_2 appears in the display.

To execute the program yourself, key in the instructions listed on page 122 and calculate the average slope for following values of x_1 and x_2: 0.5, 1.25; 2.52, 3.72; 5, 7.

Answers: 1.7325, 6.2226, 11.9826.

For additional guidelines concerning the use of subroutines, refer to page 210, Subroutines, in part III of this handbook.

Notice that GTO and GSB instructions always cause the calculator to search *downward* in program memory for the specified label. This feature often allows you to write a program in such a way that it uses a given label more than once.

Example: The following program to calculate the value of the expression $\sqrt{x^2 + y^2 + z^2 + t^2}$ uses LBL A to identify both the beginning of the program and a subroutine within the program. The program is executed by placing the variables x, y, z, and t in the stack and pressing A.*

Keystrokes	Display		
g P/R	000-		
f CLEAR PRGM	000-		
f LBL A	001-	42,21,11	
g x^2	002-	43 11	x^2
GSB A	003-	32 11	Calculates y^2 and $x^2 + y^2$
GSB A	004-	32 11	Calculates z^2 and $x^2 + y^2 + z^2$
GSB A	005-	32 11	Calculates t^2 and $x^2 + y^2 + z^2 + t^2$
$\sqrt{x}$	006-	11	Calculates $\sqrt{x^2 + y^2 + z^2 + t^2}$
g RTN	007-	43 32	
f LBL A	008-	42,21,11	
$x \gtrless y$	009-	34	
g x^2	010-	43 11	
+	011-	40	
g RTN	012-	43 32	
g P/R			

* While it is generally best to avoid the use of identical labels to reduce the possibility of confusion or programming errors, the following program illustrates one way identical labels might be used, if necessary.

Key in the following set of variables:

$$x=4.3,\ y=7.9,\ z=1.3,\ t=8.0$$

Keystrokes	Display
8 [ENTER]	**8.0000**
1.3 [ENTER]	**1.3000**
7.9 [ENTER] 4.3 [f] [A]	**12.1074**

Section 9

The Index Register

The Index Register In Brief

The Index register (R_I) is one of the most powerful programming tools available on your HP-11C. In addition to simple storage and recall of data, the Index register can also be used for:

- Program loop counter and control functions.
- Indirectly addressing data storage registers, branches, and subroutines.

Index register control of loops and indirect addressing is performed using a control number you place in the Index register itself. The *integer* portion of the control number determines the result of each loop iteration or indirect addressing operation. The *decimal* portion of the control number contains the parameters for altering and limiting the integer portion.

[I] and [(i)] Abbreviated Key Sequences. The calculator is designed so that you can omit the [f] prefix key in [I] or [(i)] key sequences. Even when the [f] prefix key is pressed in an [I] or [(i)] key sequence, the calculator will automatically exclude the code for [f].

Direct Index Register Functions

Direct Index register functions act on the contents of the Index register itself.

Index Register Store and Recall. [STO] [I] *(store in R_I)* and [RCL] [I] *(recall from R_I)* operate in the same way as [STO] and [RCL] operate with data storage registers R_0 through R_9 and $R_{.0}$ through $R_{.9}$.

Exchanging X and I. Pressing [f] [$x \gtrless$ I] exchanges the contents of the displayed X-register and the Index register in the same way that pressing [$x \gtrless y$] exchanges the contents of the displayed X-register and the Y-register.

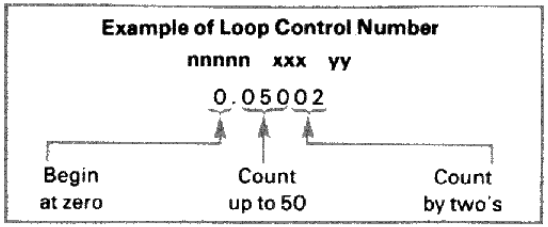
Incrementing and Decrementing the Index Register. The [ISG] *(increment, then skip if greater)* and [DSE] *(decrement, then skip if less than or equal)* functions use the loop control number you place in the Index register.

Both [ISG] and [DSE] interpret and compare the components of the loop control number according to the following format:

nnnnn.xxxyy, where	**±nnnnn**	is the current counter value,
	xxx	is the counter test value, and
	yy	is the increment or decrement value.

nnnnn is the integer portion of the control number and is used for counting successive iterations of a program loop, for determining when to exit from a loop, and for indirectly addressing branches, subroutines, and storage registers. **nnnnn** can be from 1 to 5 digits and, if unspecified, defaults to zero. **nnnnn** is incremented or decremented by executions of [ISG] or [DSE].

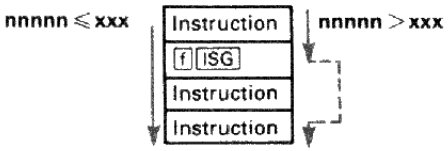
Example of Loop Control Number



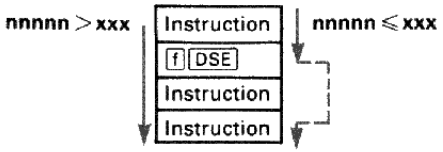
xxx is in the decimal portion of the control number. When you use [ISG] or [DSE] to increment or decrement **nnnnn**, **xxx** is then compared internally to **nnnnn** by the calculator to determine when the desired number of increments or decrements have been executed. **xxx** must be specified as a three-digit number. For example, an **xxx** value of 10 would be specified as 010. (**xxx** is a reference value and is not changed by executions of [ISG] or [DSE].)

yy is also in the decimal portion of the control number. **yy** tells the calculator what increments to count in, that is, **nnnnn** will be incremented or decremented by the value of **yy**. A specified **yy** must be two digits (for example, 02, 03, 55). An unspecified **yy** defaults to 01. (**yy** is a reference value and is not changed by executions of [ISG] or [DSE].)

[ISG] **and** [DSE] **Operation.** The increment and decrement functions use **nnnnn/xxx** comparisons to control a running program in a manner similar to the conditional tests described in section 7. Each execution of [f] [ISG] increments **nnnnn** by **yy**. In a running program a test is then made to see if **nnnnn** is greater than **xxx**. If it is, the HP-11C skips the next line in the program before resuming program execution.



Each execution of [f] [DSE] decrements **nnnnn** by **yy**. In a running program it then tests to see if **nnnnn** is equal to (or less than) **xxx**. If it is, the HP-11C skips the next line in the program before resuming program execution.



	If Initial R_i Content is:	Changes in nnnnn.xxxyy during successive iterations of [ISG] or [DSE].			
Iterations →	**0**	**1**	**2**	**3**	**4**
[ISG] →	0.00602	2.00602	4.00602	6.00602	8.00602 (Skip)
[DSE] →	6.00002	4.00002	2.00002	0.00002 (Skip)	-2.00002 (Skip)

Indirect Index Register Functions

Indirect Index register functions do not affect the contents of R_I. Instead, they use the **nnnnn** (integer) portion of the number stored in R_I as an address for determining where in memory to execute the function. Indirect functions are frequently used in conjunction with [ISG] or [DSE] in programs when it is desirable to use the same program instruction repeatedly to sequentially address storage registers, branches, or subroutines.

Indirect Storage and Recall. [STO] [(i)] and [RCL] [(i)]* store or recall numbers using the data storage register addressed by the absolute value of **nnnnn**. For addressing R_0 through R_9, **nnnnn** = 0 through 9; for $R_{.0}$ through $R_{.9}$, **nnnnn** = 10 through 19; for R_I, **nnnnn** = 20. (Refer to the Indirect Addressing table on the following page.)

Exchange X, Indirect. [f] [$x \gtrless$ (i)] exchanges the contents of the X-register with the contents of the storage register addressed by the absolute value of **nnnnn**.

Indirect Storage Register Arithmetic. [STO] ([+], [−], [×], or [÷]) [(i)] performs storage register arithmetic on the contents of the storage register addressed by the absolute value of **nnnnn**.

Examples

If 5.01202 is stored in the Index register:

nnnnn = 5 xxx = 012 yy = 02

[STO] [(i)] = [STO] 5

[f] [$x \gtrless$ (i)] = Exchange X-register and R_5 Contents

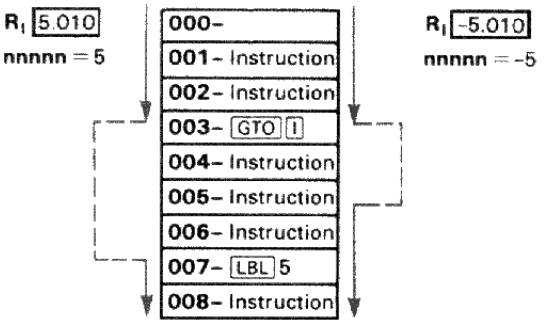
[STO] [+] [(i)] = [STO] [+] 5

Indirect Branching to Label or Line Number. In a running program:

1. If **nnnnn** is ≥ 0, [GTO] [I] branches execution downward in program memory to the next label specified by **nnnnn**.

* Refer to page 127, [I] and [(i)] Abbreviated Key Sequences.

2. If **nnnnn** is < 0, [GTO] [I] branches execution directly to the occupied *line number* specified by the absolute value of **nnnnn**.
For example:



Indirect Subroutine Label or Line Number. In a running program, [GSB] [I] indirectly transfers execution to a subroutine in the same way that [GTO] [I] indirectly transfers execution to a branch.

Indirect Addressing Table

If nnnnn is:	[GTO] or [GSB] [I] transfers execution to:	[STO] [(i)] or [RCL] [(i)] addresses storage register:
0	[f] [LBL] 0	R_0
⋮	⋮	⋮
9	[f] [LBL] 9	R_9
10	" " [A]	$R_{.0}$
11	" " [B]	$R_{.1}$
12	" " [C]	$R_{.2}$
13	" " [D]	$R_{.3}$
14	" " [E]	$R_{.4}$
15	—	$R_{.5}$
16	—	$R_{.6}$
17	—	$R_{.7}$
18	—	$R_{.8}$
19	—	$R_{.9}$
20	—	R_I

Loop Control Using [ISG]

Example: Here is a program that illustrates how [ISG] works. It contains a loop that pauses to display the current value in R_I and uses [ISG] to control the number of passes through the loop and the value of a squared number. The program generates a table of squares of even numbers from 2 through 50.

Keystrokes	Display		
[g] [P/R]	000-		Program mode.
[f] CLEAR [PRGM]	000-		
[f] [LBL] [A]	001-	42,21,11	Program label.
2	002-	2	Current counter value (**nnnnn**).
[.]	003-	48	
0	004-	0	
5	005-	5	Counter test value (**xxx**).
0	006-	0	
0	007-	0	
2	008-	2	Increment value (**yy**).
[STO] [I]	009-	44 25	Store loop control value in R_I.
[f] [LBL] 1	010-	42,21, 1	Begin the loop.
[RCL] [I]	011-	45 25	Recall the number in R_I.
[g] [INT]	012-	43 44	Take the integer portion.
[f] [PSE]	013-	42 31	Pause to display the integer.
[g] [x^2]	014-	43 11	Square the number.
[f] [PSE]	015-	42 31	Display the square of the number.
[f] [ISG]	016-	42 6	Increment R_I by 2 and check to see that the counter is not greater than the final number (50). If the counter is greater than the final number, skip the next line in the program.
[GTO] 1	017-	22 1	Loop back to label 1.
[g] [RTN]	018-	43 32	Halts the program.
[g] [P/R]			Run mode.

Now run the program by pressing [f] [A].

Keystrokes	Display	
[f] [A]	2.0000 4.0000 4.0000 16.0000 ⋮ 50.0000 2,500.0000	When the calculator begins executing, it first pauses to display the number to be squared, then pauses to display the square of the number. When the loop counter increments beyond 50, the program halts.

Here is what happens when you run the above program.

1. Under label [A], the number 2.05002 is stored in the I-register as the loop control value:

nnnnn	xxx	yy
(0000)2 Current Counter Value	050 Test Value	02 Increment Value

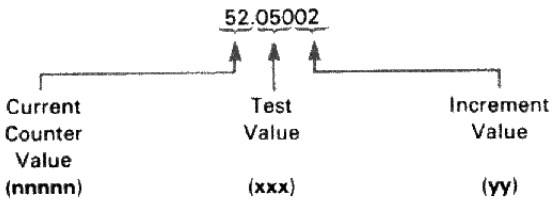
2. Under label 1, the following sequence occurs:
   After 2 and 4 (the square of 2) are displayed, the current counter value in R_I, 00002 (**nnnnn**), is incremented by the increment value 02 (**yy**). The new number in R_I is 4.05002, which is interpreted by your calculator as:

nnnnn	xxx	yy
(0000)4 Current Counter Value	050 Test Value	02 Increment Value

   The new counter value is then compared to the test value 050 (**xxx**). As the counter value has not exceeded the test value, the calculator proceeds to the next line, [GTO] 1, and the process is repeated with the new number.

3. After 25 even numbers (2-50) and their squares are displayed, the current counter value finally increments beyond 50. This causes the calculator to skip one line after the [f] [ISG] at line 16. As a result, the [GTO] 1 command at line 17 is bypassed and the [RTN] command at line 18 is executed, causing the calculator to return to line 000 and halt.

After running the program, press [f] [FIX] 5, then [RCL] [I]. The recalled I-register value in your display should now look like this:



Press [f] [FIX] 4 to return the calculator to [FIX] 4 display mode.

ISG and DSE Limits. Note that [ISG] and [DSE] can be used to increment and decrement any number that the calculator can display. However, the decimal portion of the loop control value will be affected by current counter values exceeding the five-digit **nnnnn** value.

For example, the number 99,950.50055, when incremented using [ISG] would become 100,005.5006. The initial number was incremented by 55. But since the new number 100,005.50055, cannot be fully displayed, the decimal portion of the number was rounded. As the calculator assumes a two-digit number for the increment value (**yy**), the next increment would be by 60, not 55. And when the number becomes 999,945.5006, the next number would be 1,000,005.501, thus rounding the decimal portion of the number again. Since no increment value **yy** is present, the next increment would default to 01 instead of remaining at 60.

Direct R_I Operations

To Execute	Example Keystrokes	Display
Store in R_I	12345 [STO] [I]	12,345 12,345.0000

To Execute	Example Keystrokes	Display
(Clear display)	[←]	0.0000
Recall from R_I (the value stored earlier using [STO] [I]).	[RCL] [I]	12,345.0000
(Clear display)	[←]	0.0000
Exchange X and I (using the value remaining in [I] from above examples)	[f] [x≥I]	0.0000 12,345.0000

Indirect R_I Operations

To Execute	Example Keystrokes	Display
(Store 3 in R_I as an address, that is, **nnnnn** = 3.)	3 [STO] [I]	3.0000
Indirect storage and recall of $\sqrt{7}$ using R_3 (indirectly addressed by first storing 3 in R_I, above).	7 [√x] [STO] [(i)] [←] [RCL] [(i)] [←]	2.6458 2.6458 0.0000 2.6458 0.0000
Exchange, indirect, the 0.0000 in the X-register and the 2.6458 remaining in R_3. (The **nnnnn** address 3 remains in R_I.)	[f] [x≥(i)] [f] [x≥(i)]	2.6458 0.0000

To Execute	Example Keystrokes	Display
Indirect storage register arithmetic; divide the contents of R_3 by pi.	f π	3.1416
	STO ÷ (i)	3.1416
	RCL (i)	0.8422

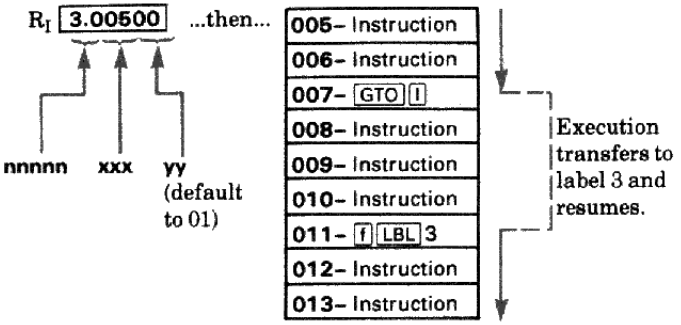
For an additional discussion of indirect storage register addressing techniques, refer to page 211, ISG With RCL (i), in part III of this handbook.

Indirect Program Control

Indirect Label Branches and Subroutines

You can indirectly branch to line numbers and labels in the same way that you indirectly address data storage registers. The table on page 131 shows the numerical address that corresponds to each possible label. Each possible address is the **nnnnn** portion of a *positive* loop control value stored in R_I.

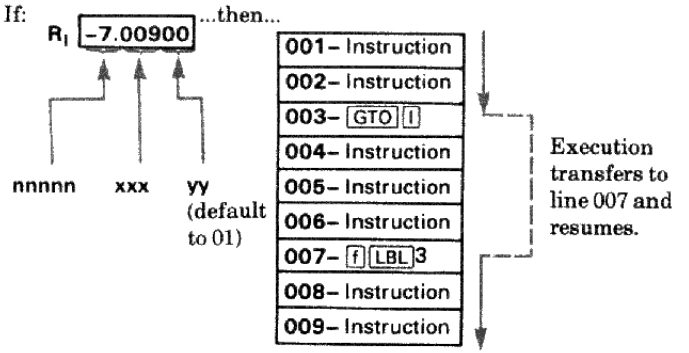
To indirectly *branch* to a label use GTO I. When a running program encounters GTO I, execution is transferred downward in program memory to the label that is indirectly addressed by the current **nnnnn** value. For example, if 3.005 is the current counter value in the Index register...



The same method is used to indirectly execute a subroutine, except that GSB I is used instead of GTO I.

Indirect Line Number Branches and Subroutines

Program line numbers can be indirectly addressed by a branch or subroutine instruction in almost the same way that labels are indirectly addressed. As covered earlier, when an indirect branch or subroutine instruction is executed with a *positive* **nnnnn** value in R_I, execution transfers to the *label* addressed by **nnnnn** and resumes. However, when an indirect branch or subroutine instruction is executed with a *negative* **nnnnn** value in R_I, execution transfers to the *line number* addressed by the *absolute value* of **–nnnnn**.



Part III

Programmed Problem Solving

Section 10

Applications Programs

Matrix Algebra

This program calculates the determinant and inverse of a 3×3 matrix (if it exists, i.e. det. $\neq 0$) and, with a minor program manipulation, the solution to the system of linear equations represented by the matrix.

Equations:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$C = \frac{1}{\det A} \begin{bmatrix} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & - \begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & \begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} \\ - \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & - \begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} \\ \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} & - \begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{bmatrix}$$

where C is the inverse of A.

$$\det A = a_{11}(a_{22}a_{23} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31})$$

and $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$

$AX = B$ is solved by $X = CB$, viz.,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \frac{1}{\det A} \begin{bmatrix} b_1 \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - b_2 \begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} + b_3 \begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} \\ - b_1 \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + b_2 \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} - b_3 \begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} \\ b_1 \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} - b_2 \begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} + b_3 \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{bmatrix}$$

Note:

- Matrix operations can be especially prone to roundoff error. In certain cases this error may significantly affect the program results.
- If you have several sets of simultaneous equations to solve, key in the determinant routine to find the determinants for each 3×3 matrix. Replace the determinant routine with the simultaneous equations routine and store the respective determinant in R_0 for each calculation.
- A 2×2 matrix should be stored as $\begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- **Error 0** will be displayed if the matrix entered is singular.

Main Program

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
f CLEAR PRGM	000–	RCL · 1	027– 45 .1
f LBL 9	001–42,21, 9	RCL · 0	028– 45 .0
f ISG	002– 42 6	RCL 8	029– 45 8
f PSE	003– 42 31	RCL · 3	030– 45 .3
RCL (i)	004– 45 24	GSB 8	031– 32 8
×	005– 20	RCL 5	032– 45 5
×	006– 20	RCL · 3	033– 45 .3
+	007– 40	RCL · 1	034– 45 .1
STO 0	008– 44 0	RCL 7	035– 45 7
g RTN	009– 43 32	GSB 8	036– 32 8
f LBL B	010–42,21,12	RCL 8	037– 45 8
RCL 9	011– 45 9	RCL 7	038– 45 7
RCL · 3	012– 45 .3	RCL 5	039– 45 5
RCL · 2	013– 45 .2	RCL · 0	040– 45 .0
RCL · 0	014– 45 .0	GSB 8	041– 32 8
GSB 8	015– 32 8	GSB 2	042– 32 2
RCL · 2	016– 45 .2	RCL 8	043– 45 8
RCL 7	017– 45 7	RCL · 2	044– 45 .2
RCL 6	018– 45 6	RCL · 1	045– 45 .1
RCL · 3	019– 45 .3	RCL 9	046– 45 9
GSB 8	020– 32 8	GSB 8	047– 32 8
RCL 6	021– 45 6	RCL · 1	048– 45 .1
RCL · 0	022– 45 .0	RCL 6	049– 45 6
RCL 9	023– 45 9	RCL 5	050– 45 5
RCL 7	024– 45 7	RCL · 2	051– 45 .2
GSB 8	025– 32 8	GSB 8	052– 32 8
GSB 2	026– 32 2	RCL 5	053– 45 5

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
RCL 9	054– 45 9	R/S	067– 31
RCL 8	055– 45 8	g RTN	068– 43 32
RCL 6	056– 45 6	f LBL 2	069–42,21, 2
f LBL 8	057–42,21, 8	RCL 4	070– 45 4
×	058– 20	g F? 0	071–43, 6, 0
R↓	059– 33	R/S	072– 31
×	060– 20	f LBL 3	073–42,21, 3
g R↑	061– 43 33	1	074– 1
−	062– 30	STO I	075– 44 25
RCL 0	063– 45 0	0	076– 0
+	064– 10	STO 4	077– 44 4
g F? 0	065–43, 6, 0	g RTN	078– 43 32
GTO 1	066– 22 1		

REGISTERS			R_i: Index
R_0: Det A	R_1: b_1	R_2: b_2	R_3: b_3
R_4: x_1, x_2, x_3	R_5: a_{11}	R_6: a_{12}	R_7: a_{13}
R_8: a_{21}	R_9: a_{22}	$R_{.0}$: a_{23}	$R_{.1}$: a_{31}
$R_{.2}$: a_{32}	$R_{.3}$: a_{33}		

Determinant Subroutine

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
f LBL A	079–42,21,11	0	083– 0
g CF 0	080–43, 5, 0	RCL · 0	084– 45 .0
4	081– 4	RCL · 2	085– 45 .2
STO I	082– 44 25	GSB 9	086– 32 9

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
RCL 8	087- 45 8	RCL · 2	095- 45 .2
RCL · 3	088- 45 .3	GSB 9	096- 32 9
GSB 9	089- 32 9	RCL 5	097- 45 5
RCL 9	090- 45 9	RCL · 3	098- 45 .3
RCL · 1	091- 45 .1	GSB 9	099- 32 9
GSB 9	092- 32 9	RCL 6	100- 45 6
CHS	093- 16	RCL · 1	101- 45 .1
RCL 7	094- 45 7	GSB 9	102- 32 9

Simultaneous Equations Subroutine

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
f LBL 1	079-42,21, 1	STO 3	088- 44 3
RCL (i)	080- 45 24	R↓	089- 33
×	081- 20	STO 2	090- 44 2
STO + 4	082-44,40, 4	R↓	091- 33
f ISG	083- 42 6	STO 1	092- 44 1
f PSE	084- 42 31	GSB 3	093- 32 3
g RTN	085- 43 32	GSB B	094- 32 12
f LBL C	086-42,21,13	g CF 0	095-43, 5, 0
g SF 0	087-43, 4, 0	RCL 4	096- 45 4

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
1	Key in the main program			
	(lines 000-078).			
2	**Determinant**			
	Key in subroutine LBL A			
	(lines 079-102).		GTO · 078	
			g P/R ... g P/R	

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
3	Set User mode.			
4	Store the elements of the matrix.			
		a_{11}	STO 5	
		a_{12}	STO 6	
		a_{13}	STO 7	
		a_{21}	STO 8	
		a_{22}	STO 9	
		a_{23}	STO · 0	
		a_{31}	STO · 1	
		a_{32}	STO · 2	
		a_{33}	STO · 3	
5	Find the determinant.		A	Det A
6	**Inverse** (Optional)			
	Find the inverse.		B	c_{11}
			R/S	c_{12}
			R/S	c_{13}
			R/S	c_{21}
			R/S	c_{22}
			R/S	c_{23}
			R/S	c_{31}
			R/S	c_{32}
			R/S	c_{33}
7	**Simultaneous Equations** (Optional)			
	Delete lines 079 to 102.		g RTN g BST	
			g P/R	102- 32 9
			← ... ←	078- 43 32
8	Key in subroutine under LBL 1		f LBL 1	079-42,21, 1
	(lines 079 to 096).		... g P/R	

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
9	Input the column matrix and find x.	b_1	ENTER	
		b_2	ENTER	
		b_3	C	x_1
			R/S	x_2
			R/S	x_3
10	For new column matrix go to			
	step 9.			
11	For new determinant or inverse			
	delete lines 079 to 096.		g RTN g BST	
			g P/R	096- 45 4
			← ... ←	078- 43 32
	Go to step 2.			

Example 1: Find the inverse of $\begin{bmatrix} 14 & -8 \\ -8 & 12 \end{bmatrix}$ then solve $\begin{bmatrix} 14 & -8 \\ -8 & 12 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 20 \\ 5 \end{bmatrix}$

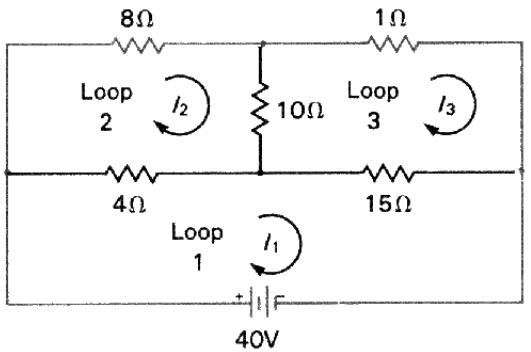
Keystrokes	Display	
Set User mode.		
f FIX 4		
Key in the determinant routine.		
14 STO 5		
8 CHS STO 6		
0 STO 7		
8 CHS STO 8		
12 STO 9		
0 STO · 0		
STO · 1 STO · 2		
1 STO · 3		
A	**104.0000**	Det. A
B	**0.1154**	c_{11}

Keystrokes	Display	
R/S	**0.0769**	c_{12}
R/S	**0.0000**	c_{13}
R/S	**0.0769**	c_{21}
R/S	**0.1346**	c_{22}
R/S	**0.0000**	c_{23}
R/S	**0.0000**	c_{31}
R/S	**0.0000**	c_{32}
R/S	**1.0000**	c_{33}
g RTN g BST		
g P/R	**102- 32 9**	
← ← ... ←	**078- 43 32**	

Key in the simultaneous equations subroutine.

Keystrokes	Display	
g P/R		
20 ENTER 5 ENTER		
0 C	**2.6923**	x_1
R/S	**2.2115**	x_2
R/S	**0.0000**	x_3

Example 2: Solve for the loop currents in the following circuit.



The three loop equations are:

Loop 1 $4I_1 - 4I_2 + 15I_1 - 15I_3 - 40 = 0$

Loop 2 $4I_2 - 4I_1 + 8I_2 + 10I_2 - 10I_3 = 0$

Loop 3 $10I_3 - 10I_2 + 1I_3 + 15I_3 - 15I_1 = 0$

or

$$19I_1 - 4I_2 - 15I_3 = 40$$
$$-4I_1 + 22I_2 - 10I_3 = 0$$
$$-15I_1 - 10I_2 + 26I_3 = 0$$

or in matrix form

$$\begin{bmatrix} 19 & -4 & -15 \\ -4 & 22 & -10 \\ -15 & -10 & 26 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} 40 \\ 0 \\ 0 \end{bmatrix}.$$

Keystrokes	Display
19 STO 5	
4 CHS STO 6	
15 CHS STO 7	
4 CHS STO 8	
22 STO 9	
10 CHS STO · 0	
15 CHS STO · 1	
10 CHS STO · 2	
26 STO · 3	

Delete the simultaneous equations routine if it is still in program memory, and insert the determinant routine.

A	**2,402.0000**

Delete the determinant routine and insert the simultaneous equations routine.

40 ENTER	
0 ENTER C	**7.8601**
R/S	**4.2298**
R/S	**6.1615**

Systems of Linear Equations With Three Unknowns

This program uses Cramer's rule to solve systems of linear equations with three unknowns. It has been included here because of the relative ease with which it can be used to solve systems of linear equations compared to the Matrix Algebra program.

Equations: A system of linear equations can be expressed as

$$AX = B.$$

For three unknowns,

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$\det A = a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31}).$$

x_i's are solved by

$$x_i = \frac{\det(i)}{\text{Det } A}$$

for Det $A \neq 0$ where det (i) is the determinant of the A matrix with the i^{th} column replaced by B.

Remarks: If Det $A = 0$, then the system is linearly dependent and this program is not applicable. The program will terminate with **Error 0**. If Det A is very close to zero, the calculator representation of the number will contain significant round-off error and the ratio (det (i)/Det A) will be in error.

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
f CLEAR PRGM	000-	RCL 5	029- 45 5
f LBL A	001-42,21,11	RCL 7	030- 45 7
1	002- 1	GSB 9	031- 32 9
.	003- 48	CHS	032- 16
0	004- 0	RCL 3	033- 45 3
0	005- 0	RCL 8	034- 45 8
9	006- 9	GSB 9	035- 32 9
STO I	007- 44 25	RCL 1	036- 45 1
f LBL 2	008-42,21, 2	RCL 9	037- 45 9
RCL I	009- 45 25	GSB 9	038- 32 9
g INT	010- 43 44	RCL 2	039- 45 2
f PSE	011- 42 31	RCL 7	040- 45 7
RCL (i)	012- 45 24	f LBL 9	041-42,21, 9
R/S	013- 31	f DSE	042- 42 5
STO (i)	014- 44 24	f PSE	043- 42 31
f ISG	015- 42 6	RCL (i)	044- 45 24
GTO 2	016- 22 2	×	045- 20
GSB 0	017- 32 0	×	046- 20
STO 0	018- 44 0	+	047- 40
R/S	019- 31	g RTN	048- 43 32
f LBL 0	020-42,21, 0	f LBL B	049-42,21,12
0	021- 0	STO . 3	050- 44 .3
STO I	022- 44 25	R↓	051- 33
RCL 6	023- 45 6	STO . 2	052- 44 .2
RCL 8	024- 45 8	R↓	053- 33
GSB 9	025- 32 9	STO . 1	054- 44 .1
RCL 4	026- 45 4	1	055- 1
RCL 9	027- 45 9	GSB 7	056- 32 7
GSB 9	028- 32 9	R/S	057- 31

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
4	058- 4	RCL . 0	075- 45 .0
GSB 7	059- 32 7	g RTN	076- 43 32
R/S	060- 31	f LBL 8	077-42,21, 8
7	061- 7	RCL . 1	078- 45 .1
GSB 7	062- 32 7	f x≷(i)	079- 42 23
R/S	063- 31	STO . 1	080- 44 .1
f LBL 7	064-42,21, 7	f ISG	081- 42 6
STO . 4	065- 44 .4	f PSE	082- 42 31
STO I	066- 44 25	RCL . 2	083- 45 .2
GSB 8	067- 32 8	f x≷(i)	084- 42 23
GSB 0	068- 32 0	STO . 2	085- 44 .2
RCL 0	069- 45 0	f ISG	086- 42 6
+	070- 10	f PSE	087- 42 31
STO . 0	071- 44 .0	RCL . 3	088- 45 .3
RCL . 4	072- 45 .4	f x≷(i)	089- 42 23
STO I	073- 44 25	STO . 3	090- 44 .3
GSB 8	074- 32 8	g RTN	091- 43 32

REGISTERS			R_i: Index
R_0: Det	R_1: a_{11}	R_2: a_{21}	R_3: a_{31}
R_4: a_{12}	R_5: a_{22}	R_6: a_{32}	R_7: a_{13}
R_8: a_{23}	R_9: a_{33}	$R_{.0}$: det (i)	$R_{.1}$: b_1
$R_{.2}$: b_2	$R_{.3}$: b_3	$R_{.4}$: Index	

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
1	Key in the program.			
2	Set User mode.			

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
3	Key in the 3 × 3 matrix		A	1, R_1
	in column order.	a_{11}	R/S	2, R_2
	(Note: to leave the displayed value	a_{21}	R/S	3, R_3
	in the register press R/S).	a_{31}	R/S	4, R_4
		a_{12}	R/S	5, R_5
		a_{22}	R/S	6, R_6
		a_{32}	R/S	7, R_7
		a_{13}	R/S	8, R_8
		a_{23}	R/S	9, R_9
		a_{33}	R/S	Det.
	To review the matrix press A.			
4	Input the column matrix.	b_1	ENTER	b_1
		b_2	ENTER	b_2
		b_3	B	x_1
			R/S	x_2
			R/S	x_3
5	For new column matrix go to step 4.			
6	For new 3 × 3 matrix go to step 3.			

Example 1: Find the solution matrix for the following:

$$\begin{bmatrix} 19 & -4 & -15 \\ -4 & 22 & -10 \\ -15 & -10 & 26 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 40 \\ 0 \\ 0 \end{bmatrix}$$

Keystrokes	Display		
f CLEAR REG			
Set User mode.			
f FIX 4			
A	1.0000	0.0000	

Keystrokes	Display		
19 R/S	2.0000	0.0000	
4 CHS R/S	3.0000	0.0000	
15 CHS R/S	4.0000	0.0000	
4 CHS R/S	5.0000	0.0000	
22 R/S	6.0000	0.0000	
10 CHS R/S	7.0000	0.0000	
15 CHS R/S	8.0000	0.0000	
10 CHS R/S	9.0000	0.0000	
26 R/S	2,402.0000		Det.
40 ENTER			
0 ENTER B	7.8601		x_1
R/S	4.2298		x_2
R/S	6.1615		x_3

Example 2: Find the solution matrix for:

$$\begin{bmatrix} 19 & -4 & 4 \\ 5 & -12 & -10 \\ -15 & 8 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ -3 \\ 4 \end{bmatrix}$$

Keystrokes	Display		
A	1.0000	19.0000	
R/S	2.0000	-4.0000	
5 R/S	3.0000	-15.0000	
R/S	4.0000	-4.0000	
R/S	5.0000	22.0000	
12 CHS R/S	6.0000	-10.0000	
8 R/S	7.0000	-15.0000	
4 R/S	8.0000	-10.0000	
R/S	9.0000	26.0000	
3 R/S	-264.0000		Det.
5 ENTER			
3 CHS ENTER			
4 B	-1.6667		x_1
R/S	-4.4091		x_2
R/S	4.7576		x_3

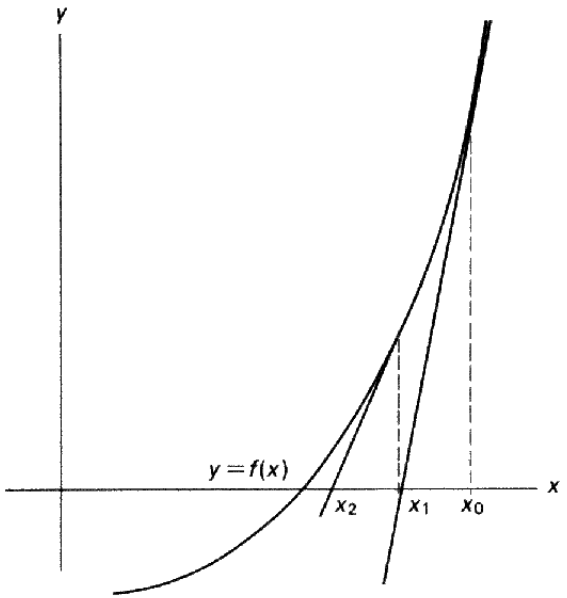
Newton's Method Solution to $f(x) = 0$

One of the most common and frustrating problems in algebra is the solution of an equation like:

$$\ln x + 3x = 10.8074,$$

in which the x's refuse to conveniently migrate to one side of the equation and isolate themselves. That is, there is no simple algebraic solution. The following program uses Newton's method to find a solution for $f(x) = 0$, where $f(x)$ is specified by the user.

The user must define the function $f(x)$ by entering into program memory the keystrokes required to find $f(x)$, assuming x is in the X-register. The main program is 68 steps long and requires registers 0-4. The remaining memory and available registers may be used for defining $f(x)$ by the user. In addition, the user must provide the program with an initial guess, x_0, for the solution. The closer the initial guess is to the actual solution, the faster the program will converge to an answer.



The program will halt when the following conditions are satisfied:

1. Loop count ≥ Loop Limit
2. $|f(x)| \leq$ tolerance (ϵ)
3. $|x_{new} - x_{old}| \leq \Delta x$ *limit*

The user may choose values for ϵ and Δx or the program will use default values.

Equations:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

This program makes a numerical approximation for the derivative $f'(x)$ to give the following equation:

$$x_{i+1} = x_i - \delta_i \frac{f(x_i)}{f(x_i + \delta_i) - f(x_i)}$$

where $\delta_0 = 10^{-5} x_0$ and $\delta_i = x_i - x_{i-1}$

Note:

- After the program has finished, x for $f(x) = 0$ will be displayed and will also be in register 2.
- **Error 0** will be displayed if, in the attempt to find a root, there is a division by zero. If this occurs, try a new guess that is close to your last guess.
- This program also keeps track of the number of iterations that take place when trying to find a root. R_1 is initialized to 50 (lines 020, 021 and 022) and an error (**Error 4**) will occur if that many iterations have taken place. The user may change the maximum number of iterations by deleting lines 020 and 021 and keying in the desired number.
- The user may like to see each x_{i+1} as it is found by Newton's method. To do this key in [f] [PSE] after [STO] 2 (at line 047 if no other modifications have been made to the program).
- If the tolerance is too small, round-off error within the calculator will become significant and the roots become erroneous.

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
f CLEAR PRGM	000–	GSB C	029– 32 13
f LBL A	001–42,21,11	STO 2	030– 44 2
EEX	002– 26	RCL 0	031– 45 0
CHS	003– 16	RCL 1	032– 45 1
5	004– 5	+	033– 40
STO 0	005– 44 0	GSB C	034– 32 13
EEX	006– 26	RCL 2	035– 45 2
CHS	007– 16	−	036– 30
8	008– 8	g x=0	037– 43 40
STO 3	009– 44 3	STO · 9	038– 44 .9
STO 4	010– 44 4	RCL 2	039– 45 2
R/S	011– 31	x≷y	040– 34
STO 3	012– 44 3	+	041– 10
R/S	013– 31	RCL 0	042– 45 0
STO 4	014– 44 4	×	043– 20
g CLx	015– 43 35	CHS	044– 16
g RTN	016– 43 32	RCL 1	045– 45 1
f LBL B	017–42,21,12	+	046– 40
STO 2	018– 44 2	STO 2	047– 44 2
STO × 0	019–44,20, 0	g LSTx	048– 43 36
5	020– 5	−	049– 30
0	021– 0	STO 0	050– 44 0
STO I	022– 44 25	f DSE	051– 42 5
GSB 1	023– 32 1	GTO 2	052– 22 2
RCL 2	024– 45 2	GSB 9	053– 32 9
g RTN	025– 43 32	f LBL 2	054–42,21, 2
f LBL 1	026–42,21, 1	RCL 2	055– 45 2
RCL 2	027– 45 2	GSB C	056– 32 13
STO 1	028– 44 1	g ABS	057– 43 16

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
RCL 3	058– 45 3	g ABS	064– 43 16
x≥y	059– 34	f x≤y	065– 42 10
f x>y	060– 42 20	g RTN	066– 43 32
GTO 1	061– 22 1	GTO 1	067– 22 1
RCL 4	062– 45 4	f LBL C	068–42,21,13
RCL 0	063– 45 0		

REGISTERS			R_i: Loop counter
R_0: δ_i	R_1: Temporary	R_2: Temporary	R_3: ϵ
R_4: Δx limit	R_5—R_8: Unused		

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
1	Key in the program.			
2	Go to LBL C.		GTO C	
3	Set Program mode.		g P/R	068-42,21,13
4	Key in the function to be solved.	$f(x)$		
5	Return to Run mode.		g P/R	
6	Set User mode.			
7	Initialize tolerances.		A	10^{-8}
8	(Optional) Input ϵ.	ϵ	R/S	ϵ
	If you do not wish to alter the			
	default value of ϵ (10^{-8}) but wish to			
	perform step 9:		R/S	10^{-8}
9	(Optional) Input Δx limit.	*Δx limit*	R/S	*Δx limit*
10	Input the initial guess and compute			
	the root.	x_0	B	x_i

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
11	For a new initial guess go to			
	step 10.			
12	For new tolerances go to step 8.			
13	For a new $f(x)$ go to the end of the			
	program.		g RTN g BST	
	Set program mode and delete lines		g P/R	
	back to LBL C.		← ... ←	068–42,21,13
	Go to step 4.			

Example: Find a root of

$$f(x) = x^6 - x - 1 = 0$$

with $x_0 = 2$.

Keystrokes	Display	
GTO C		
g P/R	068–42,21,13	
STO 5		
6 y^x		
RCL 5 − 1 −	075– 30	
g P/R		
f FIX 9		
Set User mode.		
A	0.000000010	Default ϵ and Δx.
2 B	1.134724138	x_i
C	−0.000000004	$f(x_i)$

Numerical Integration by Discrete Points

This program will perform numerical integration when a function is known at a finite number of equally spaced points (discrete case). The integrals are approximated by either the trapezoidal rule or Simpson's rule.

Equations:

Let $x_0, x_1, \ldots, x_n$ be $n+1$ equally spaced points ($x_j = x_0 + jh$, $j = 0, 1, 2, \ldots, n$) at which corresponding values $f(x_0)$, $f(x_1)$ ..., $f(x_n)$ of the function $f(x)$ are known.

The integral: $\int_{x_0}^{x_n} f(x)dx$ may be approximated using:

1. The trapezoidal rule:

$$\int_{x_0}^{x_n} f(x)dx \sim \frac{h}{2}\left[f(x_0) + 2\left(\sum_{j=1}^{n-1} f(x_j)\right) + f(x_n)\right]$$

2. Simpson's rule:

$$\int_{x_0}^{x_n} f(x)dx \sim \frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + \ldots 4f(x_{n-3})\right.$$

$$\left. + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\right]$$

In order to apply Simpson's rule, n must be even. If it is not, **Error 0** will be displayed.

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
f CLEAR PRGM	000–	STO 0	005– 44 0
f LBL A	001–42,21,11	STO 5	006– 44 5
STO 4	002– 44 4	0	007– 0
R/S	003– 31	STO 3	008– 44 3
f LBL B	004–42,21,12	f LBL 1	009–42,21, 1

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
R/S	010- 31	GSB 3	035- 32 3
f LBL B	011-42,21,12	3	036- 3
STO 1	012- 44 1	RCL 5	037- 45 5
GSB 2	013- 32 2	f LBL 0	038-42,21, 0
ENTER	014- 36	RCL 1	039- 45 1
+	015- 40	−	040- 30
STO + 5	016-44,40, 5	RCL 4	041- 45 4
1	017- 1	×	042- 20
STO + 3	018-44,40, 3	x≷y	043- 34
RCL 3	019- 45 3	+	044- 10
R/S	020- 31	R/S	045- 31
f LBL B	021-42,21,12	f LBL 2	046-42,21, 2
STO 1	022- 44 1	ENTER	047- 36
GSB 2	023- 32 2	+	048- 40
STO + 5	024-44,40, 5	STO + 0	049-44,40, 0
1	025- 1	g RTN	050- 43 32
STO + 3	026-44,40, 3	f LBL 3	051-42,21, 3
RCL 3	027- 45 3	2	052- 2
GTO 1	028- 22 1	÷	053- 10
f LBL C	029-42,21,13	f FRAC	054- 42 44
2	030- 2	g x=0	055- 43 40
RCL 0	031- 45 0	g RTN	056- 43 32
GTO 0	032- 22 0	0	057- 0
f LBL D	033-42,21,14	÷	058- 10
RCL 3	034- 45 3	g RTN	059- 43 32

REGISTERS			R_i: Unused
R_0: Used	R_1: $f(x_j)$	R_2: Unused	R_3: j
R_4: h	R_5: Used	R_6—R_9: Unused	

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
1	Key in the program.			
2	Set User mode.			
3	Input the interval between x-values.	h	A	h
4	Input the value of the function at x_j			
	for j=0, 1, ..., n.	$f(x_j)$	B	j
5	Calculate the integral via the			
	trapezoidal rule.		C	Trap $\int$
	OR,			
	by Simpson's rule (n must have			
	been even).		D	Simp $\int$
6	For new case, go to step 3.			

Example: Given the values below for $f(x_j)$, $j = 0, 1, ..., 8$, calculate the approximations to the integral

$$\int_0^2 f(x)dx$$

by using the trapezoidal rule and by using Simpson's rule. The value for h is 0.25.

j	0	1	2	3	4	5	6	7	8
x_j	0	0.25	0.5	0.75	1	1.25	1.5	1.75	2
$f(x_j)$	2	2.8	3.8	5.2	7	9.2	12.1	15.6	20

Keystrokes	Display
f FIX 2	
Set User mode.	
.25 A	0.25
2 B	0.00
2.8 B	1.00
3.8 B	2.00
5.2 B	3.00

Keystrokes	Display	
7 [B]	4.00	
9.2 [B]	5.00	
12.1 [B]	6.00	
15.6 [B]	7.00	
20 [B]	8.00	
[C]	16.68	Trapezoidal.
[D]	16.58	Simpson's.

Curve Fitting

Your HP-11C calculator is equipped with a powerful built-in function, linear regression, [L.R.], which quickly and conveniently fits data to a straight line. (See page 63.)

This capability is used here in a program to fit data to other types of curves:

1. Exponential curves; $y = ae^{bx}$ $(a > 0)$
2. Logarithmic curves; $y = a + b \ln x$
3. Power curves; $y = ax^b$ $(a > 0)$

which may be transformed to the general linear form $Y = A + bX$.

The regression coefficients A and b are found by solving the following system of linear equations

$$\begin{bmatrix} n & \Sigma X_i \\ \Sigma X_i & \Sigma X_i^2 \end{bmatrix} \begin{bmatrix} A \\ b \end{bmatrix} = \begin{bmatrix} \Sigma Y_i \\ \Sigma (Y_i X_i) \end{bmatrix}$$

where n is the number of data pairs.

The relations of the variables are defined as the following:

Regression	A	X_i	Y_i	Code
Exponential	$\ln a$	x_i	$\ln y_i$	1
Logarithmic	a	$\ln x_i$	y_i	2
Power	$\ln a$	$\ln x_i$	$\ln y_i$	3

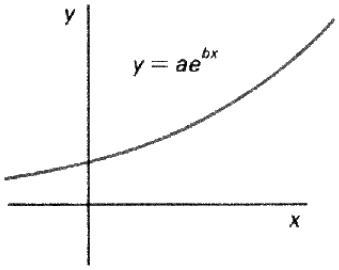
The coefficient of determination is:

$$r^2 = \frac{A\Sigma Y_i + b\Sigma X_i Y_i - \frac{1}{n}(\Sigma Y_i)^2}{\Sigma(Y_i^2) - \frac{1}{n}(\Sigma Y_i)^2}$$

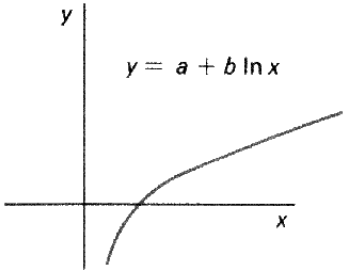
The type of curve fit to be run is determined before data input begins by inputting the code number.

The coefficient of determination indicates the quality of fit achieved by the regression. Values of r^2 close to 1.00 indicate a better fit than values close to zero. The regression coefficients a and b define the curve generated, according to the equations at the beginning of this discussion.

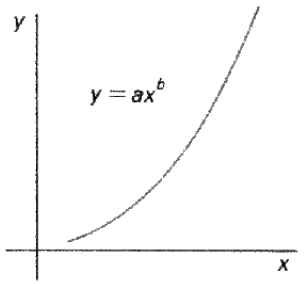
Exponential Curve Fit
Code = 1



Logarithmic Curve Fit
Code = 2



Power Curve Fit
Code = 3



Remarks:

- The program applies the least squares method, either to the original equations (logarithmic curve) or to the transformed equations (exponential curve and power curve).
- Negative and zero values of x_i will cause a machine error for logarithmic curve fits. Negative and zero values of y_i will cause a machine error for exponential curve fits. For power curve fits, both x_i and y_i must be positive, non-zero values.
- As the differences between x and/or y values become small, the accuracy of the regression coefficients will decrease.
- During operation of the program all storage registers are cleared. Any data stored in extra registers will therefore be destroyed.

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
f CLEAR PRGM	000-	6	006- 6
f LBL A	001-42,21,11	+	007- 40
f CLEAR REG	002- 42 34	STO I	008- 44 25
f FIX 2	003-42, 7, 2	R↓	009- 33
ENTER	004- 36	g SF 0	010-43, 4, 0
ENTER	005- 36	g SF 1	011-43, 4, 1

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
GTO I	012- 22 25	g Σ-	036- 43 49
f LBL 7	013-42,21, 7	GTO 9	037- 22 9
g CF 1	014-43, 5, 1	f LBL B	038-42,21,12
GTO 9	015- 22 9	f LR	039- 42 49
f LBL 8	016-42,21, 8	g F? 0	040-43, 6, 0
g CF 0	017-43, 5, 0	e^x	041- 12
f LBL 9	018-42,21, 9	R/S	042- 31
R/S	019- 31	$x \geq y$	043- 34
g F? 0	020-43, 6, 0	R/S	044- 31
g LN	021- 43 12	f $\hat{y},r$	045- 42 48
$x \geq y$	022- 34	$x \geq y$	046- 34
g F? 1	023-43, 6, 1	g x^2	047- 43 11
g LN	024- 43 12	g RTN	048- 43 32
RCL 6	025- 45 6	f LBL D	049-42,21,14
g $x \neq 0$	026- 43 30	1	050- 1
GTO 6	027- 22 6	STO 6	051- 44 6
R↓	028- 33	GTO 9	052- 22 9
Σ+	029- 49	f LBL C	053-42,21,13
GTO 9	030- 22 9	g F? 1	054-43, 6, 1
f LBL 6	031-42,21, 6	g LN	055- 43 12
R↓	032- 33	f $\hat{y},r$	056- 42 48
0	033- 0	g F? 0	057-43, 6, 0
STO 6	034- 44 6	e^x	058- 12
R↓	035- 33		

REGISTERS			R_i: Code
R_0: i	R_1: Σx_i	R_2: Σx_i^2	R_3: Σy_i
R_4: Σy_i^2	R_5: $\Sigma x_i y_i$	R_6: 1 or 0	R_7—R_9: Unused

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
1	Key in the program.			
2	Set User mode.			
3	Select the type of curve fit:			
	Exponential	1	[A]	1.00
	Logarithmic	2	[A]	2.00
	Power	3	[A]	3.00
4	Input x_i value and y_i value.	x_i	[ENTER]	
		y_i	[R/S]	i
	(Repeat step 4 for all data points.)			
5	Calculate regression coefficients		[B]	a
			[R/S]	b
	and coefficient of determination.		[R/S]	r^2
6	Make projection of new $\hat{y}$ for a			
	known x value.	x	[C]	$\hat{y}$
	(Repeat step 6 for all x values of			
	interest.)			
7	Error correction:			
	Erroneous inputs at step 4 may be		[D]	
	corrected by pressing [D] and	$x_i$err	[ENTER]	
	reinputting the erroneous data.	$y_i$err	[R/S]	$i-1$
	Then return to step 4 and enter the			
	correct data.			

Example 1:

(Exponential, Code = 1)

x_i	0.72	1.31	1.95	2.58	3.14
y_i	2.16	1.61	1.16	0.85	0.5

Solution:

$a = 3.45, \quad b = -0.58$

$y = 3.45\, e^{-0.58x}$

$r^2 = 0.98$

Keystrokes	Display	
Set User mode.		
1 [A]	1.00	
.72 [ENTER] 2.16 [R/S]		
1.31 [ENTER] 1.61 [R/S]		
1.95 [ENTER] 1.16 [R/S]		
2.58 [ENTER] .85 [R/S]		
3.14 [ENTER] .5 [R/S]		
[B]	3.45	a
[R/S]	-0.58	b
[R/S]	0.98	r^2
1.5 [C]	1.44	$\hat{y}$

Example 2:

(Logarithmic, Code = 2)

x_i	3	4	6	10	12
y_i	1.5	9.3	23.4	45.8	60.1

Solution:

$a = -47.02, \quad b = 41.39$

$y = 47.02 + 41.39 \ln x$

$r^2 = 0.98$

For $x = 8, \quad \hat{y} = 39.06$

For $x = 14.5, \quad \hat{y} = 63.67$

Keystrokes	Display
2 [A]	2.00
3 [ENTER] 1.5 [R/S]	
4 [ENTER] 9.3 [R/S]	
6 [ENTER] 23.4 [R/S]	

Keystrokes	Display	
10 [ENTER] 45.8 [R/S]		
12 [ENTER] 60.1 [R/S]		
[B]	-47.02	a
[R/S]	41.39	b
[R/S]	0.98	r^2
8 [C]	39.06	$\hat{y}$
14.5 [C]	63.67	$\hat{y}$

Example 3:

(Power, Code = 3)

x_i	1.3	2.6	4	5.4	7.1
Y_i	3.74	6.15	7.21	8.03	8.84

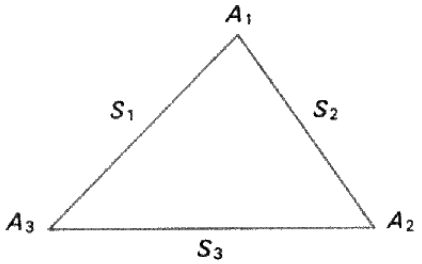
Solution:

$a = 3.49, b = 0.50$
$y = 3.49\,x^{0.50}$
$r^2 = 0.97$

Keystrokes	Display	
3 [A]	3.00	
1.3 [ENTER] 3.74 [R/S]		
2.6 [ENTER] 6.15 [R/S]		
4 [ENTER] 7.21 [R/S]		
5.4 [ENTER] 8.03 [R/S]		
7.1 [ENTER] 8.84 [R/S]		
[B]	3.49	a
[R/S]	0.50	b
[R/S]	0.97	r^2

Triangle Solutions

This program may be used to find the sides, the angles, and the area of a plane triangle.



In general, the specification of any three of the six parameters of a triangle (3 sides, 3 angles) is sufficient to define a triangle. (The exception is that three angles will not define a triangle). There are thus five possible cases that this program will handle: two sides and the included angle (SAS), two angles and the included side (ASA), two sides and the adjacent angle (SSA—an ambiguous case), two angles and the adjacent side (AAS), and three sides (SSS).

If the three known input values are selected in a clockwise order around the triangle, the outputs will also follow a clockwise order.

Remarks:

- Inputs may be in any angular mode (i.e., DEG, RAD, GRAD). Be sure to set the angular mode of the calculator to match that of the inputs.
- Note that the triangle described by the program does not conform to standard triangle notation; i.e., A_1 is not opposite S_1.
- Angles must be entered as decimals. The [g] [→H] conversion can be used to convert degrees, minutes, and seconds to decimal degrees.
- Accuracy of solution may degenerate for triangles containing extremely small angles.

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
f CLEAR PRGM	000-	RCL 2	029- 45 2
f LBL A	001-42,21,11	RCL 6	030- 45 6
STO 5	002- 44 5	+	031- 40
R↓	003- 33	SIN	032- 23
STO 3	004- 44 3	÷	033- 10
R↓	005- 33	RCL 1	034- 45 1
STO 1	006- 44 1	×	035- 20
RCL 3	007- 45 3	STO 3	036- 44 3
g →P	008- 43 26	GTO 0	037- 22 0
g x^2	009- 43 11	f LBL C	038-42,21,13
RCL 5	010- 45 5	STO 4	039- 44 4
g x^2	011- 43 11	R↓	040- 33
−	012- 30	STO 2	041- 44 2
RCL 1	013- 45 1	R↓	042- 33
RCL 3	014- 45 3	STO 1	043- 44 1
×	015- 20	RCL 4	044- 45 4
2	016- 2	RCL 2	045- 45 2
×	017- 20	+	046- 40
+	018- 10	SIN	047- 23
g COS^{-1}	019- 43 24	RCL 4	048- 45 4
STO 2	020- 44 2	SIN	049- 23
GTO 0	021- 22 0	+	050- 10
f LBL B	022-42,21,12	RCL 1	051- 45 1
STO 2	023- 44 2	×	052- 20
R↓	024- 33	STO 3	053- 44 3
STO 1	025- 44 1	GTO 0	054- 22 0
R↓	026- 33	f LBL E	055-42,21,15
STO 6	027- 44 6	STO 4	056- 44 4
SIN	028- 23	R↓	057- 33

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
STO 3	058- 44 3	f LBL D	087-42,21,14
R↓	059- 33	STO 3	088- 44 3
STO 1	060- 44 1	R↓	089- 33
RCL 3	061- 45 3	STO 2	090- 44 2
RCL 4	062- 45 4	R↓	091- 33
SIN	063- 23	STO 1	092- 44 1
RCL 1	064- 45 1	f LBL 0	093-42,21, 0
+	065- 10	RCL 2	094- 45 2
×	066- 20	RCL 1	095- 45 1
g SIN^{-1}	067- 43 23	f →R	096- 42 26
RCL 4	068- 45 4	RCL 3	097- 45 3
+	069- 40	$x \geq y$	098- 34
GSB 9	070- 32 9	−	099- 30
STO 2	071- 44 2	g →P	100- 43 26
GSB 0	072- 32 0	STO 5	101- 44 5
RCL 1	073- 45 1	$x \geq y$	102- 34
RCL 3	074- 45 3	STO 4	103- 44 4
f $x \leq y$	075- 42 10	RCL 2	104- 45 2
GTO 8	076- 22 8	+	105- 40
2	077- 2	GSB 9	106- 32 9
R/S	078- 31	STO 6	107- 44 6
RCL 6	079- 45 6	SIN	108- 23
GSB 9	080- 32 9	×	109- 20
STO 6	081- 44 6	RCL 1	110- 45 1
RCL 4	082- 45 4	×	111- 20
+	083- 40	2	112- 2
GSB 9	084- 32 9	÷	113- 10
STO 2	085- 44 2	STO 0	114- 44 0
GTO 0	086- 22 0	.	115- 48

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
0	116- 0	GTO 2	124- 22 2
0	117- 0	f LBL 8	125-42,21, 8
6	118- 6	g CLΣ	126- 43 35
STO I	119- 44 25	g RTN	127- 43 32
f LBL 2	120-42,21, 2	f LBL 9	128-42,21, 9
RCL (i)	121- 45 24	COS	129- 24
R/S	122- 31	CHS	130- 16
f ISG	123- 42 6	g COS^{-1}	131- 43 24

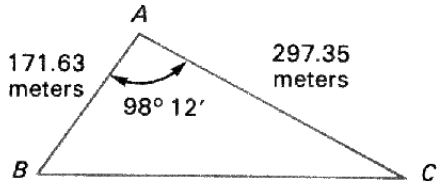
REGISTERS			R_i: Index
R_0: Area	R_1: S_1	R_2: A_1	R_3: S_2
R_4: A_2	R_5: S_3	R_6: A_3	R_7—R_8: Unused

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
1	Key in the program.			
2	Set User mode.			
3	Set desired angular mode			
	[DEG, RAD, GRAD].			

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
4	**SSS [3 sides]**			
	Enter:			
	Side 1	S_1	ENTER	
	Side 2	S_2	ENTER	
	Side 3	S_3	A	Area
	Go to step 9.			
5	**ASA [2 angles + included side]**			
	Enter:			
	Angle 3	A_3	ENTER	
	Side 1	S_1	ENTER	
	Angle 1	A_1	B	Area
	Go to step 9.			
6	**SAA [2 angles + adjacent side]**			
	Enter:			
	Side 1	S_1	ENTER	
	Angle 1	A_1	ENTER	
	Angle 2	A_2	C	Area
	Go to step 9.			
7	**SAS [2 sides + included angle]**			
	Enter:			
	Side 1	S_1	ENTER	
	Angle 1	A_1	ENTER	
	Side 2	S_2	D	Area
	Go to step 9.			
8	**SSA [2 sides + adjacent angle]**			
	Enter:			
	Side 1	S_1	ENTER	
	Side 2	S_2	ENTER	

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
	Angle 2	A_2	E	Area
9	Dimensions:			
	To find the remaining sides and			
	angles:		R/S	S_1
			R/S	A_1
			R/S	S_2
			R/S	A_2
			R/S	S_3
			R/S	A_3
9a	If no alternative triangle is possible:		R/S	0
9b	If second triangle is possible:		R/S	2
			R/S	Area
			R/S	S_1
			R/S	A_1
			R/S	S_2
			R/S	A_2
			R/S	S_3
			R/S	A_3

Example 1: A surveyor is to find the area and dimensions of a triangular land parcel. From point A, the distances to B and C are measured with an electronic distance meter. The angle between AB and AC is also measured. Find the area and other dimensions of the triangle.



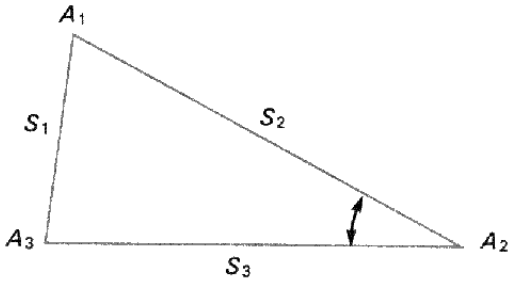
This is a side-angle-side problem where:

$$S_1 = 171.63, \quad A_1 = 98°12' \quad \text{and} \quad S_2 = 297.35.$$

Keystrokes	Display	
g DEG		
f FIX 2		
Set User mode.		
171.63 ENTER	171.63	
98.12 g →H	98.20	
297.35 D	25,256.21	Area.
R/S	171.63	S_1
R/S	98.20	A_1
R/S	297.35	S_2
R/S	27.83	A_2
R/S	363.91	S_3
R/S	53.97	A_3

Example 2: Given 2 sides and a nonincluded angle, solve for the triangle:

$$\text{Side}_1 = 25.6$$
$$\text{Side}_2 = 32.8$$
$$\text{Angle}_2 = 42.3°$$



Keystrokes	Display	
25.6 [ENTER]	25.60	
32.8 [ENTER]	32.80	
42.3 [E]	410.85	Area.
[R/S]	25.60	S_1
[R/S]	78.12	A_1
[R/S]	32.80	S_2
[R/S]	42.30	A_2
[R/S]	37.22	S_3
[R/S]	59.58	A_3
[R/S]	2.00	Second Solution.
[R/S]	124.68	Area.
[R/S]	25.60	S_1
[R/S]	17.28	A_1
[R/S]	32.80	S_2
[R/S]	42.30	A_2
[R/S]	11.30	S_3
[R/S]	120.42	A_3

t Statistics

Paired t Statistics

Given a set of paired observations from two normal populations with means μ_1, μ_2 (unknown)

x_i	x_1	$x_2 \ldots x_n$
y_i	y_1	$y_2 \cdots y_n$

let $$D_i = x_i - y_i$$

$$\bar{D} = \frac{1}{n}\sum_{i=1}^{n} D_i$$

$$s_D = \sqrt{\frac{\Sigma D_i^2 - \frac{1}{n}(\Sigma D_i)^2}{n-1}}$$

The test statistic

$$t = \frac{\bar{D}}{s_D}\sqrt{n}$$

which has $n-1$ degrees of freedom (df) can be used to test the null hypothesis, H_0: $\mu_1 = \mu_2$.

t Statistic for Two Means

Suppose $\{x_1, x_2, \ldots, x_{n_1}\}$ and $\{y_1, y_2, \ldots, y_{n_2}\}$ are independent random samples from two normal populations having means μ_1 and μ_2 (unknown) and the same unknown variance σ^2.

We want to test the null hypothesis H_0: $\mu_1 - \mu_2 = d$.

Equations:

$$\bar{x} = \frac{1}{n_1}\sum_{i=1}^{n_1} x_i$$

$$\bar{y} = \frac{1}{n_2}\sum_{i=1}^{n_2} y_i$$

$$t = \frac{\bar{x} - \bar{y} - d}{\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}\sqrt{\frac{\Sigma x_i^2 - n_1\bar{x}^2 + \Sigma y_i^2 - n_2\bar{y}^2}{n_1 + n_2 - 2}}}$$

We can use this t statistic (which has the t distribution with $n_1 + n_2 - 2$ degrees of freedom (df)) to test the null hypothesis H_0.

References:

1. *Statistics in Research*, B. Ostle, Iowa State University Press, 1963.
2. *Statistical Theory and Methodology in Science and Engineering*, K.A. Brownlee, John Wiley and Sons, 1965.

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
f CLEAR PRGM	000–	STO · 2	029– 44 .2
f LBL A	001–42,21,11	RCL 0	030– 45 0
−	002– 30	STO · 0	031– 44 .0
Σ+	003– 49	0	032– 0
R/S	004– 31	f CLEAR Σ	033– 42 32
f LBL C	005–42,21,13	R/S	034– 31
g x̄	006– 43 0	f LBL D	035–42,21,14
STO 6	007– 44 6	STO 6	036– 44 6
R/S	008– 31	RCL · 1	037– 45 .1
g s	009– 43 48	RCL · 2	038– 45 .2
R/S	010– 31	STO 4	039– 44 4
RCL 6	011– 45 6	x≷y	040– 34
RCL 0	012– 45 0	STO 3	041– 44 3
√x	013– 11	g x̄	042– 43 0
×	014– 20	STO 8	043– 44 8
x≷y	015– 34	x≷y	044– 34
÷	016– 10	RCL 0	045– 45 0
R/S	017– 31	×	046– 20
RCL 0	018– 45 0	RCL · 0	047– 45 .0
1	019– 1	÷	048– 10
−	020– 30	STO 7	049– 44 7
R/S	021– 31	x≷y	050– 34
GTO C	022– 22 13	−	051– 30
f LBL B	023–42,21,12	RCL 6	052– 45 6
Σ+	024– 49	−	053– 30
R/S	025– 31	RCL 4	054– 45 4
RCL 1	026– 45 1	RCL 3	055– 45 3
STO · 1	027– 44 .1	RCL 7	056– 45 7
RCL 2	028– 45 2	×	057– 20

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
−	058– 30	√x	072– 11
RCL 2	059– 45 2	RCL 0	073– 45 0
+	060– 40	1/x	074– 15
RCL 1	061– 45 1	RCL · 0	075– 45 .0
RCL 8	062– 45 8	1/x	076– 15
×	063– 20	+	077– 40
−	064– 30	√x	078– 11
RCL 0	065– 45 0	×	079– 20
RCL · 0	066– 45 .0	−	080– 10
+	067– 40	R/S	081– 31
2	068– 2	RCL 9	082– 45 9
−	069– 30	R/S	083– 31
STO 9	070– 44 9	RCL 6	084– 45 6
+	071– 10	GTO D	085– 22 14

REGISTERS*			
R_0: n_1, n_2; n	R_1: Σx, Σy; ΣD	R_2: Σx^2, Σy^2; ΣD^2	R_i: Unused
R_4: Σx^2;	R_5: Unused	R_6: d; $\bar{D}$	R_7: $\bar{x}$;
R_8: $\bar{y}$;	R_9: df;	R_0: n_1;	R_1: Σx;
R_2: Σx^2;	R_3—R_4: Unused		

* Entries before the semicolon indicate register usage for the t Statistics for Two Means program. Entries after the semicolon are for the Paired t Statistics program. A blank entry represents unused; a single entry without a semicolon is valid for both programs.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
	Paired t Statistics			
1	Key in lines 001-022.			
2	Initialize the program.		f CLEAR REG	
3	Set User mode.			
	Repeat steps 4-5 for $i = 1, 2, \ldots, n$.			

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
4	Input			
		x_i	ENTER	
		y_i	A	i
5	(Optional) To correct erroneous			
	x_k or y_k:	x_k	ENTER	
		y_k	- g Σ-	$k-1$
6	To calculate test statistics:			
	$\bar{D}$		C	$\bar{D}$
	s_D		R/S	s_D
	t		R/S	t
	df		R/S	df
7	Repeat step 6 to review the results.			
	t Statistic for Two Means			
1	Key in lines 023-085.			
2	Initialize the program.		f CLEAR REG	
3	Set User mode.			
	Repeat steps 4-5 for i=1, 2, ..., n_1.			
4	Input x_i	x_i	B	i
5	To correct erroneous input x_k:	x_k	g Σ-	$k-1$
6	Initialize for the second array of			
	data.		R/S	
	Repeat steps 7-8 for i=1, 2, ..., n_2.			
7	Input y_i	y_i	B	i
8	To correct erroneous input y_k:	y_k	g Σ-	$k-1$
9	Input d to calculate test statistic:			
		d	D	t
			R/S	df
10	Repeat step 9 to review the results.			

Example 1:

x_i	14	17.5	17	17.5	15.4
y_i	17	20.7	21.6	20.9	17.2

Find $\bar{D}$, s_D, t and df for the above data.

Keystrokes	Display	
f FIX 4		
Set User Mode.		
f CLEAR REG		
14 ENTER 17 A	1.0000	
17 ENTER 15 A	2.0000	Error.
17 ENTER		
15 - g Σ-	1.0000	Correction.
17.5 ENTER 20.7 A	2.0000	
17 ENTER 21.6 A	3.0000	
17.5 ENTER 20.9 A	4.0000	
15.4 ENTER 17.2 A	5.0000	
C	-3.2000	$\bar{D}$
R/S	1.0000	s_D
R/S	-7.1554	t
R/S	4.0000	df

Example 2:

x	79	84	108	114	120	103	122	120		
y	91	103	90	113	108	87	100	80	99	54

If $d = 0$ (i.e., H_0: $\mu_1 = \mu_2$) find t and df for the above data.

Keystrokes	Display
f CLEAR REG	
79 B 84 B 99	
B 99 g Σ-	2.0000
108 B 114 B	
120 B 103 B	
122 B 120 B	8.0000
R/S	0.0000
91 B 103 B	

Keystrokes	Display	
90 [B] 113 [B]		
108 [B] 87 [B]		
100 [B] 80 [B]		
99 [B] 54 [B]	10.0000	
0 [D]	1.7316	t
[R/S]	16.0000	df

Chi-Square Evaluation

This program calculates the value of the χ^2 statistic for the goodness of fit test by the equation:

$$\chi_1{}^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \qquad (df = n - 1)$$

where O_i = observed frequency

E_i = expected frequency.

If the expected values are equal,

$$E = E_i = \frac{\Sigma O_i}{n} \text{ for all } i,$$

then

$$\chi_2{}^2 = \frac{n\Sigma O_i^2}{\Sigma O_i} - \Sigma O_i.$$

Note: In order to apply the goodness of fit test to a set of given data, combining some classes may be necessary to ensure that each expected frequency is not too small (say, not less than 5).

Reference:

Mathematical Statistics, J.E. Freund, Prentice Hall, 1962.

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
[f] CLEAR [PRGM]	000–	[GTO] [C]	022– 22 13
[f] [LBL] [A]	001–42,21,11	[f] [LBL] [B]	023–42,21,12
[g] [CF] 0	002–43, 5, 0	[Σ+]	024– 49
[f] [LBL] 0	003–42,21, 0	[R/S]	025– 31
[ENTER]	004– 36	[f] [LBL] [D]	026–42,21,14
[R↓]	005– 33	[RCL] 0	027– 45 0
[−]	006– 30	[RCL] 2	028– 45 2
[g] [x^2]	007– 43 11	[×]	029– 20
[g] [R↑]	008– 43 33	[RCL] 1	030– 45 1
[÷]	009– 10	[÷]	031– 10
[g] [F?] 0	010–43, 6, 0	[RCL] 1	032– 45 1
[CHS]	011– 16	[−]	033– 30
[STO] [+] 1	012–44,40, 1	[R/S]	034– 31
1	013– 1	[RCL] 1	035– 45 1
[g] [F?] 0	014–43, 6, 0	[RCL] 0	036– 45 0
[CHS]	015– 16	[÷]	037– 10
[STO] [+] 0	016–44,40, 0	[R/S]	038– 31
[RCL] 0	017– 45 0	[GTO] [D]	039– 22 14
[R/S]	018– 31	[f] [LBL] [E]	040–42,21,15
[f] [LBL] [C]	019–42,21,13	[g] [SF] 0	041–43, 4, 0
[RCL] 1	020– 45 1	[GTO] 0	042– 22 0
[R/S]	021– 31		

REGISTERS			
R_0: Index	R_1: $\chi_2{}^2$, ΣO_i	R_2: $\Sigma O_i{}^2$	R_3: Used
R_4: Used	R_5: Used	R_6—R_9: Unused	R_i: Unused

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
	Unequal Expected Frequency			
1	Key in the program.			
2	Initialize the program.		f CLEAR REG	
3	Set User mode.			
	Repeat steps 4-5 for i=1, 2, ..., n.			
4	Input			
	O_i	O_i	ENTER	
	E_i	E_i	A	i
5	To correct erroneous O_k or E_k	O_k	ENTER	
		E_k	E	k–1
6	Calculate χ_1^2.		C	χ_1^2
7	Repeat step 6 to review the result.			
	Equal Expected Frequency			
1	Initialize the program.		f CLEAR REG	
2	Set User mode.			
	Repeat steps 3-4 for i=1, 2, ..., n.			
3	Input O_i	O_i	B	i
4	To correct erroneous O_h	O_h	g Σ-	h–1
5	Calculate			
	χ_2^2		D	χ_2^2
	E		R/S	E
6	Repeat step 5 to review the results.			

Example 1: Find the value of the χ^2 statistic for the goodness of fit for the following data set:

O_i	8	50	47	56	5	14
E_i	9.6	46.75	51.85	54.4	8.25	9.15

Keystrokes	Display	
f CLEAR REG		
f FIX 4		
Set User Mode.		
8 ENTER 9.6 A	1.0000	
50 ENTER 46.75 A	2.0000	
47 ENTER 51.85 A	3.0000	
56 ENTER 54.4 A	4.0000	
100 ENTER A	5.0000	Error.
100 ENTER E	4.0000	Correction.
5 ENTER 8.25 A	5.0000	
14 ENTER 9.15 A	6.0000	
C	4.8444	χ_1^2

Example 2: The following table shows the observed frequencies in tossing a die 120 times. χ^2 can be used to test if the die is fair. (df = 5.) Note: Assume that the expected frequencies are equal.

Number	1	2	3	4	5	6
Frequency O_i	25	17	15	23	24	16

Keystrokes	Display	
f CLEAR REG		
25 B 17 B 19 B	3.0000	Error.
19 g Σ-	2.0000	Correction.
15 B 23 B 24 B		
16 B	6.0000	
D	5.0000	χ_2^2
R/S	20.0000	E

The value of χ^2 for df = 5 and 5% significance (found on p. 438 of Freund's *Mathematical Statistics*, 2nd edition) is 11.070. Since 5.00 is less than 11.070, no statistically significant differences exist between the observed and expected frequencies.

Finance: Annuities and Compound Amounts

This program can be used to solve a variety of problems involving money and time. The following variables (except the interest rate,

which must be given) can be either inputs or outputs:

n: The number of compounding periods. (For a 30 year loan with monthly payments, $n = 12 \times 30 = 360$.)

i: The periodic interest rate expressed as a percentage. (For other than annual compounding, divide the annual percentage rate by the number of compounding periods in a year, i.e., 10% annual interest compounded monthly equals 10/12 or 0.83%.)

PV: The present value of the cash flow or compound amounts.

PMT: The periodic payment.

FV: The future value; the final cash flow (balloon payment or remaining balance) or the compounded value of a series of cash flows.

Accumulated interest and remaining balance may also be computed with this program.

The program accommodates payments which are made at the beginning or end of compounding periods. Payments made at the end of compounding periods (ordinary annuity) are common in direct reduction loans and mortgages while payments at the beginning of compounding periods (annuity due) are common in leasing. For ordinary annuity press [A] and then [R/S] until 1 is displayed. For annuity due press [A] then [R/S] until 0 is displayed.

This program uses the convention that cash outlays are input as negative, and cash incomes are input as positive.

Pressing [f] CLEAR [REG] provides a safe, convenient, easy to remember method of preparing the calculator for a new problem. However, it is not necessary to use [f] CLEAR [REG] between problems containing the same combination of variables. For instance, any number of *n, i, PV, PMT, FV* problems involving different numbers and/or different combinations of knowns can be done in succession without clearing the registers. Only the values which change from problem to problem would have to be keyed in. To change the combination of variables without using [f] CLEAR [REG], simply input zero for any variable which is no longer applicable. To go from a problem where *n, i, PMT, PV* are knowns to a problem where *n, i, PV, FV* are knowns, a zero would be stored in place of *PMT*. The following table summarizes these procedures.

Possible Solutions Using Annuities and Compound Amounts (Interest is given)

Allowable Combination of Variables	Applications		Initial Procedure
	Ordinary Annuity	Annuity Due	
n, PV, PMT, (input any two and calculate the third).	Direct reduction loan Discounted notes Mortgages	Leases	Use [f] CLEAR [REG] or set *FV* to zero
n, PV, PMT, FV (input any three and calculate the fourth).	Direct reduction loan with balloon Discounted notes with balloon	Lease with residual values	None
n, PMT, FV (input any two and calculate the third).	Sinking fund	Periodic savings, Insurance	Use [f] CLEAR [REG] or set *FV* to zero
n, PV, FV (input any two and calculate the third).	Compound amount Savings (Annuity mode is not applicable and has no effect)		Use [f] CLEAR [REG] or set *PMT* to zero

Equations:

$$-PV = \frac{PMT}{i} A\left[1-(1+i)^{-n}\right] + (FV)(1+i)^{-n}$$

where $A = \begin{cases} 1 & \text{ordinary annuity} \\ (1+i) & \text{annuity due} \end{cases}$

Note:

- Problems with an interest rate of 0 will result in **Error 0**.
- Problems with extremely high (10^6) or low values (10^{-6}) for *n* or *i* may give invalid results.

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
f CLEAR PRGM	000-	STO 1	029- 44 1
f LBL A	001-42,21,11	g RTN	030- 43 32
f FIX 2	002-42, 7, 2	f LBL C	031-42,21,13
STO 2	003- 44 2	STO 3	032- 44 3
f LBL 4	004-42,21, 4	R/S	033- 31
R/S	005- 31	GSB 1	034- 32 1
0	006- 0	GSB 2	035- 32 2
g F? 0	007-43, 6, 0	CHS	036- 16
1	008- 1	STO 3	037- 44 3
g CF 0	009-43, 5, 0	g RTN	038- 43 32
g x=0	010- 43 40	f LBL D	039-42,21,14
g SF 0	011-43, 4, 0	STO 4	040- 44 4
GTO 4	012- 22 4	R/S	041- 31
f LBL B	013-42,21,12	1	042- 1
STO 1	014- 44 1	STO 4	043- 44 4
R/S	015- 31	GSB 1	044- 32 1
GSB 1	016- 32 1	1/x	045- 15
RCL 5	017- 45 5	RCL 3	046- 45 3
g LSTx	018- 43 36	GSB 2	047- 32 2
−	019- 30	×	048- 20
RCL 3	020- 45 3	CHS	049- 16
g LSTx	021- 43 36	STO 4	050- 44 4
+	022- 40	g RTN	051- 43 32
÷	023- 10	f LBL E	052-42,21,15
CHS	024- 16	STO 5	053- 44 5
g LN	025- 43 12	R/S	054- 31
RCL 6	026- 45 6	GSB 1	055- 32 1
g LN	027- 43 12	RCL 3	056- 45 3
÷	028- 10	+	057- 40

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
RCL 7	058- 45 7	y^x	076- 14
÷	059- 10	STO 7	077- 44 7
CHS	060- 16	1	078- 1
STO 5	061- 44 5	x≷y	079- 34
g RTN	062- 43 32	−	080- 30
f LBL 1	063-42,21, 1	RCL 4	081- 45 4
1	064- 1	RCL 8	082- 45 8
RCL 2	065- 45 2	÷	083- 10
g %	066- 43 14	RCL 0	084- 45 0
STO 8	067- 44 8	×	085- 20
1	068- 1	×	086- 20
STO 0	069- 44 0	g RTN	087- 43 32
+	070- 40	f LBL 2	088-42,21, 2
STO 6	071- 44 6	RCL 5	089- 45 5
g F? 0	072-43, 6, 0	RCL 7	090- 45 7
STO 0	073- 44 0	×	091- 20
RCL 1	074- 45 1	÷	092- 40
CHS	075- 16		

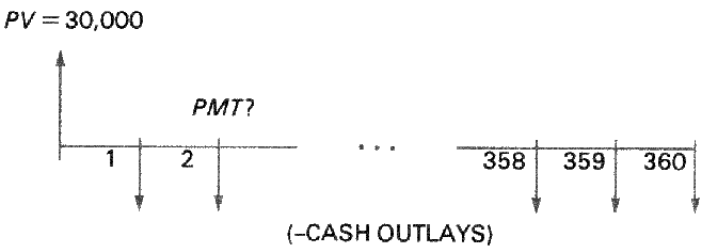
REGISTERS			R_i: Unused
R_0: 1 or 1+i/100	R_1: *n*	R_2: *i*(%)	R_3: *PV*
R_4: *PMT*	R_5: *FV*	R_6: $1+i/100$	R_7: $(1+i/100)^{-n}$
R_8: $i/100$	R_9—$R_{.4}$: Unused		

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
1	Key in the program.			
2	Set User mode.			
3	Initialize.		f CLEAR REG	
4	Input periodic interest rate.	*i*(%)	A	*i*(%)
5	Set ordinary annuity mode		R/S	1.00
	or annuity due mode		R/S	0.00
	(repeating this step toggles			
	between the two modes).			
6	Input the known values:			
	• Number of periods	*n*	B	*n*
	• Present value	*PV*	C	*PV*
	• Periodic payment	*PMT*	D	*PMT*
	• Future value	*FV*	E	*FV*
7	Calculate the unknown value:			
	• Number of periods		B R/S	*n*
	• Present value		C R/S	*PV*
	• Periodic payment		D R/S	*PMT*
	• Future value		E R/S	*FV*
8	To modify the problem, go to step 4			
	and change the appropriate values.			
	Input zero for any value not appli-			
	cable in the new problem.			
9	For new case, go to step 3.			

Example 1: If you place $155 in a savings account paying 5¾% compounded monthly, what sum of money can you withdraw at the end of 9 years?

Keystrokes	Display	
f CLEAR REG		
Set User mode.		
5.75 ENTER 12 ÷ A	0.48	% monthly interest rate.
R/S	0.00	Annuity due.
9 ENTER 12 × B	108.00	# of months.
155 CHS C	−155.00	Initial deposit.
E R/S	259.74	*FV*

Example 2: You receive $30,000 from a bank as a 30 year, 13% mortgage. What monthly payment must you make to the bank to fully amortize the mortgage?



Keystrokes	Display	
f CLEAR REG		
13 ENTER 12 ÷ A	1.08	% monthly interest rate.
R/S	1.00	Ordinary annuity.
30 ENTER 12 × B	360.00	*n*
30000 C	30,000.00	*PV*
D R/S	-331.86	*PMT*

Example 3: Two individuals are constructing a loan with a balloon payment. The loan amount is $3,600, and it is agreed that the annual interest rate will be 10% with 36 monthly payments of $100. What balloon payment amount, to be paid coincident with the 36th payment, is required to fulfill the loan agreement?

(Note the cash flow diagram below is with respect to the lender. For the borrower, the appropriate diagram will be exactly the opposite.)

Keystrokes	Display	
f CLEAR REG		
10 ENTER 12 ÷ A	0.83	% monthly interest rate.
R/S	1.00	Ordinary annuity.
36 B	36.00	*n*
3600 CHS C	-3,600.00	*PV*
100 D E R/S	675.27	*FV*

(Note that the final payment is $675.27 + $100.00 = $775.27 since the final payment falls at the end of the last period.)

Example 4: This program may also be used to calculate accumulated interest/remaining balance for loans. The accumulated interest between two points in time, n_1 and n_2, is just the total payments made in that period less the principal reduction in that period. The principal reduction is the difference of the remaining balances for the two points in time.

For a 360 month, $50,000 loan at a 14% annual interest rate, find the remaining balance after the 24th payment and the accrued interest for payments 13-24 (between the 12th and 24th payments).

First we must calculate the payment on the loan:

Keystrokes	Display	
f CLEAR REG		
360 B	360.00	*n*
14 ENTER 12 ÷ A	1.17	*i*
R/S	1.00	Ordinary annuity.
50000 CHS C	-50,000.00	*PV*
D R/S	592.44	*PMT*

The remaining balance at month 24 is:

24 B E R/S	49,749.56	*FV* at month 24.

Store this remaining balance and calculate the remaining balance at month 12:

STO 9		
12 B E R/S	49,883.48	*FV* at month 12.

The principal reduction between payments 12 and 24 is:

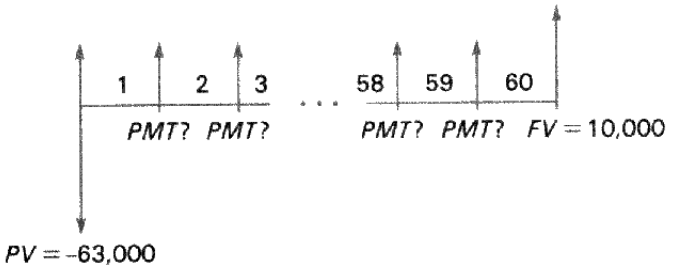
RCL 9 -	133.92

The accrued interest is 12 payments less the principal reduction:

RCL 4 12 ×	7,109.23	Total paid out.
x≷y -	6,975.31	Accrued interest.

Example 5: A third party leasing firm is considering the purchase of a minicomputer priced at $63,000 and intends to achieve a 13% annual yield by leasing the computer to a customer for a 5-year period. Ownership is retained by the leasing firm, and at the end of the lease they expect to be able to sell the equipment for at least $10,000. (Since lease payments occur at the start of each

period, this is an annuity due problem.) What monthly payment should the firm charge in order to achieve the 13% yield?



Keystrokes	Display	
f CLEAR REG		
13 ENTER 12 ÷ A	1.08	*i*
R/S	0.00	Annuity due.
5 ENTER 12 × B	60.00	*n*
63000 CHS C	-63,000.00	*FV*
10000 E D R/S	1,300.16	*PMT*

If the price is increased to $70,000 what should the payments be?

70000 CHS C	-70,000.00	
D R/S	1,457.73	*PMT*

Submarine Hunt

Using your destroyer, try to locate the position of the enemy submarine in a 10 × 10 grid, then destroy it with a depth charge.

Input a seed (between 0 and 1) and the calculator will position the submarine in the center of the 100 squares (R,C), where R = row, and C = column, and where R and C can each be 0, 1, 2, ..., 9.

Make guesses as to where you think the submarine is hiding by taking sonar readings. Input the location of your destroyer (R,C) and press B. If the submarine is in one of the 8 adjacent squares (or directly under your destroyer), the calculator will display "1." Otherwise, a "0" will be shown.

When you think you've located the submarine, move your destroyer directly over it (move to the same square) and drop a depth charge. Blinking "1's" indicate a hit, while a "0" shows a miss. If you miss, the submarine will move randomly to one of the 4 adjacent squares in the same row or column.

To make a more challenging game, press C immediately after inputting the seed. This allows the submarine to move after each sonar echo as well as after each depth charge miss. The submarine always moves randomly to an adjacent square in the same row or column.

A depth charge has a range of 0.9 squares. When you position your destroyer for a depth charge drop, you may move anywhere on the board, not just to the center of a square. For instance, a depth charge dropped from a (2.5, 6.5) location would destroy any submarine in the center of squares (2, 6) (2, 7) (3, 6) and (3, 7).

Try to destroy the submarine using no more than 10 sonar readings and 1 depth charge. You can check your status any time the display is steady by pressing D.

Status format is XX.YY

where: XX = Number of depth charges fired.

YY = Number of sonar readings.

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
f CLEAR PRGM	000-	g RTN	005- 43 32
f LBL C	001-42,21,13	f LBL E	006-42,21,15
1	002- 1	f CLEAR REG	007- 42 34
STO 0	003- 44 0	g CF 0	008-43, 5, 0
g SF 0	004-43, 4, 0	STO RAN#	009- 44 36

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
GSB 9	010- 32 9	f PSE	039- 42 31
STO 1	011- 44 1	g RTN	040- 43 32
GSB 9	012- 32 9	f LBL B	041-42,21,12
STO 2	013- 44 2	1	042- 1
f FIX 0	014-42, 7, 0	STO + 8	043-44,40, 8
g CLx	015- 43 35	R↓	044- 33
g RTN	016- 43 32	f FIX 0	045-42, 7, 0
f LBL A	017-42,21,11	g CF 1	046-43, 5, 1
1	018- 1	GSB 6	047- 32 6
STO + 7	019-44,40, 7	RCL 0	048- 45 0
R↓	020- 33	STO 5	049- 44 5
g SF 1	021-43, 4, 1	g F? 0	050-43, 6, 0
GSB 6	022- 32 6	GSB 5	051- 32 5
g x≠0	023- 43 30	RCL 3	052- 45 3
GTO 0	024- 22 0	g RTN	053- 43 32
1	025- 1	f LBL 5	054-42,21, 5
STO 5	026- 44 5	GSB 9	055- 32 9
GSB 5	027- 32 5	4	056- 4
g RTN	028- 43 32	x≷y	057- 34
f LBL 0	029-42,21, 0	f x>y	058- 42 20
9	030- 9	GTO 0	059- 22 0
1/x	031- 15	RCL 5	060- 45 5
f FIX 3	032-42, 7, 3	CHS	061- 16
f PSE	033- 42 31	GTO 1	062- 22 1
f FIX 5	034-42, 7, 5	f LBL 0	063-42,21, 0
f PSE	035- 42 31	RCL 5	064- 45 5
f FIX 7	036-42, 7, 7	f LBL 1	065-42,21, 1
f PSE	037- 42 31	STO 6	066- 44 6
f FIX 9	038-42, 7, 9	GSB 9	067- 32 9

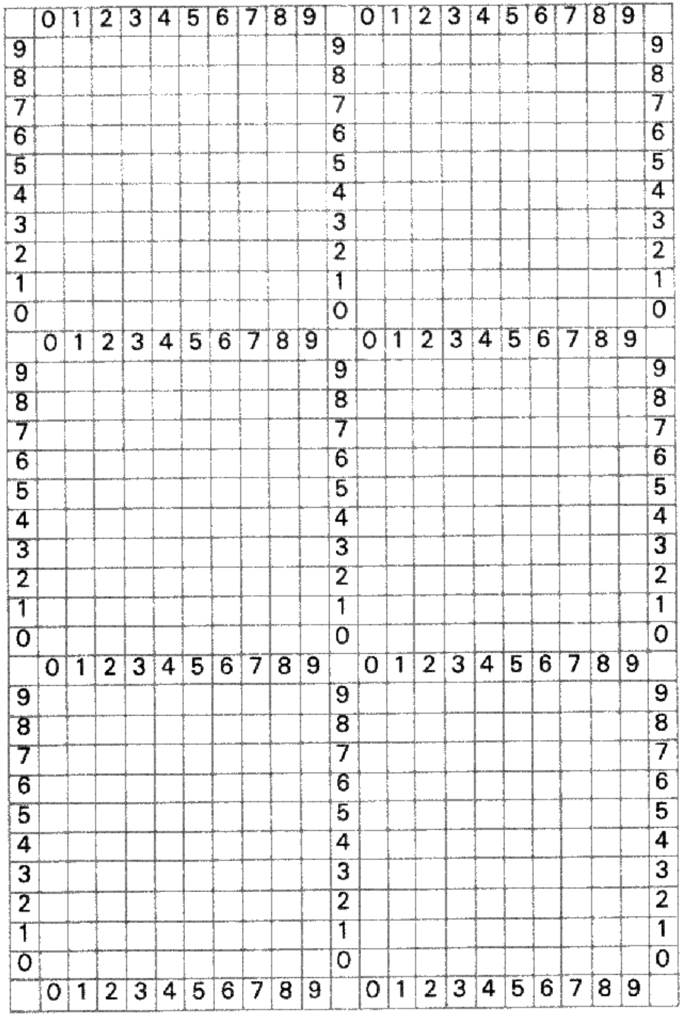
KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
5	068- 5	f FIX 0	097-42, 7, 0
f x>y	069- 42 20	g RTN	098- 43 32
GTO 0	070- 22 0	f LBL 6	099-42,21, 6
RCL 1	071- 45 1	RCL 2	100- 45 2
GSB 1	072- 32 1	−	101- 30
STO 1	073- 44 1	x≷y	102- 34
GTO 2	074- 22 2	RCL 1	103- 45 1
f LBL 0	075-42,21, 0	−	104- 30
RCL 2	076- 45 2	g →P	105- 43 26
GSB 1	077- 32 1	STO 4	106- 44 4
STO 2	078- 44 2	g F? 1	107-43, 6, 1
GTO 2	079- 22 2	GTO 0	108- 22 0
f LBL 1	080-42,21, 1	1	109- 1
RCL 6	081- 45 6	−	110- 30
+	082- 40	f LBL 0	111-42,21, 0
g x<0	083- 43 10	·	112- 48
GTO 0	084- 22 0	9	113- 9
9	085- 9	−	114- 30
x≷y	086- 34	g x<0	115- 43 10
f x≤y	087- 42 10	GTO 0	116- 22 0
g RTN	088- 43 32	0	117- 0
f LBL 0	089-42,21, 0	GTO 1	118- 22 1
RCL 6	090- 45 6	f LBL 0	119-42,21, 0
2	091- 2	1	120- 1
×	092- 20	f LBL 1	121-42,21, 1
−	093- 30	STO 3	122- 44 3
g RTN	094- 43 32	g RTN	123- 43 32
f LBL 2	095-42,21, 2	f LBL D	124-42,21,14
RCL 3	096- 45 3	f FIX 2	125-42, 7, 2

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
RCL 7	126- 45 7	f LBL 9	133-42,21, 9
RCL 8	127- 45 8	f RAN#	134- 42 36
EEX	128- 26	1	135- 1
2	129- 2	0	136- 0
÷	130- 10	×	137- 20
+	131- 40	g INT	138- 43 44
g RTN	132- 43 32	g RTN	139- 43 32

REGISTERS			R_i: Unused
R_0: 0, 1	R_1: P_1	R_2: P_2	R_3: Response
R_4: d	R_5: Used	R_6: Used	R_7: Used
R_8: Used			

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
1	Key in the program.			
2	Set User mode.			
3	Input seed (between 0 and 1).	*n*	E	0
4	For regular game, go to step 6.			
5	Select different game (submarine			
	always moving).		C	1
6	**Sonar**	*Row*	ENTER	
	"0" means no echo.	*Col.*	B	0 or 1
	"1" means echo received.			
	OR			
	Depth Charge	*Row*	ENTER	
	"0" means miss.	*Col.*	A	0 or
	Blinking "1's" means HIT!			blink

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYSTROKES	OUTPUT DATA/UNITS
7	Repeat step 6 until submarine is			
	hit.			
8	To review status at any time:		D	XX.YY
	XX = number of depth charges fired.			
	YY = number of sonar readings.			
9	For new game go to step 3.			



Playing boards for Submarine Hunt. You might wish to use copies of this page for your games.

Example 1:

Keystrokes	**Display**	
Set User mode.		
0.58 [E]	0.	
First move:		
3 [ENTER] 8 [B]	1.	Echo.

You now know your enemy is in one of the x squares below.

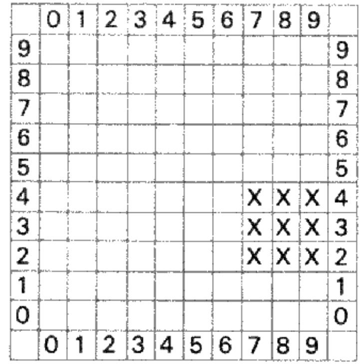


Diagram of 1st move

Second move:		
4 [ENTER] 7 [B]	0.	No echo.

The submarine cannot be in the (X) squares below.

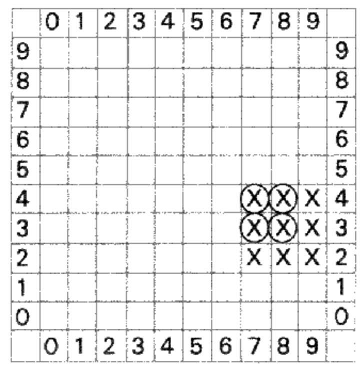


Diagram of 2nd move

Third move:

2 [ENTER] 9 [B] **0.** No echo.

You've narrowed down the submarine's location to just 2 squares, those containing an x with no circle.

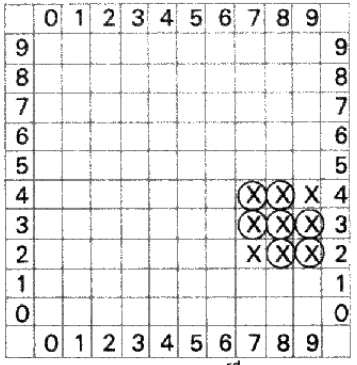


Diagram of 3rd move

Fourth move:

4 [ENTER] 9 [B] **1.** Echo.

This eliminates (2, 7) as a submarine location, so you've found it.

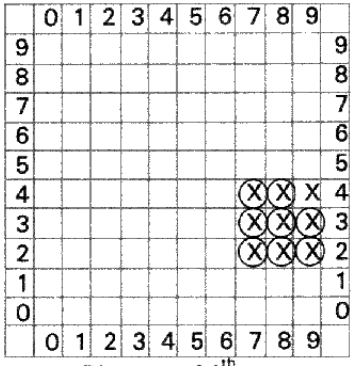


Diagram of 4th move

Fifth move:

4 [ENTER] 9 [A]

0.111
0.11111
0.1111111
0.111111111
0.111111111 A hit!

Example 2:

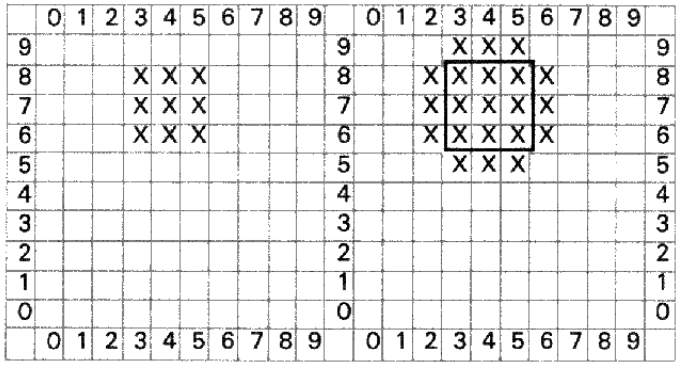
Keystrokes	**Display**
0.6 [E]	**0**
[C]	**1.**

Submarine will now move on sonar echoes as well as on depth charge misses.

First move:

7 [ENTER] 4 [B] **1.** Echo.

The submarine is in one of the x squares in the left diagram below. But the submarine moves, so now it could be in any of the x squares in the right diagram below.



Diagrams of 1st move

Second move:

6 [ENTER] 4 [B] **0.** No echo.

You've eliminated some positions (left diagram: (x)), but new possible positions have been created by the enemy's random move (right diagram).

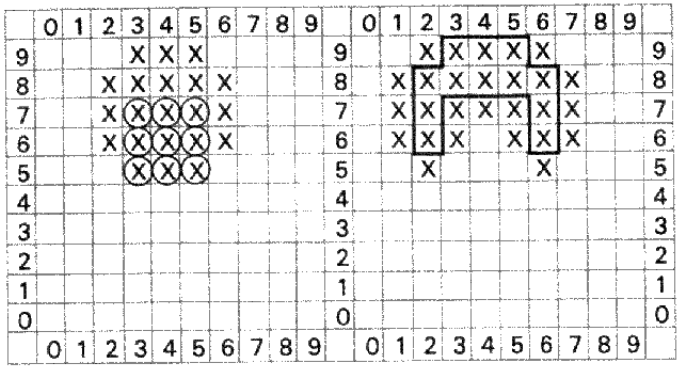


Diagram of 2nd move

Third move:

7 ENTER 3 B **1.** Echo.

This eliminates many possible positions (left diagram), but again, new ones are created (right diagram).

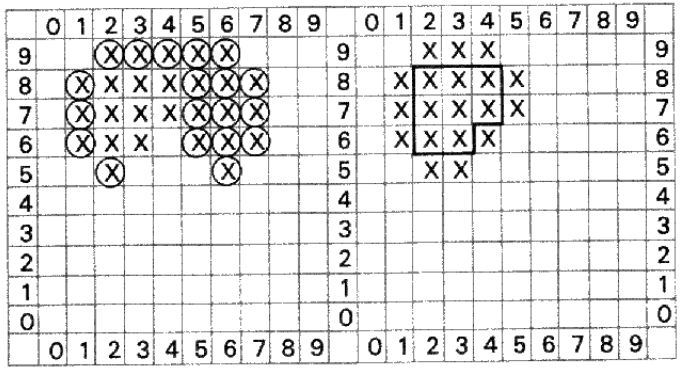


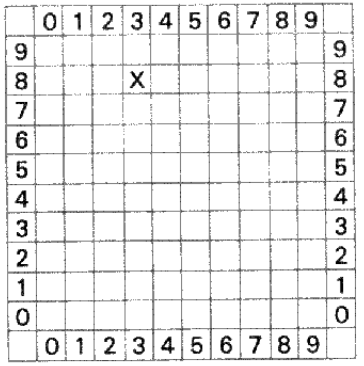
Diagram of 3rd move

Fourth move: You try a depth charge.

8 ENTER 3 A **0.111**
0.11111
0.1111111
0.111111111
0.111111111 A hit!

It pays to be lucky.

The submarine used to be here:



Section 11

Programming Techniques

Structure

What we mean by "structure" is the notion that even in a language as far from English as the HP-11C programming language there can be organization. We want you to realize that the programs you run on the HP-11C can be both useful and "friendly". But it is up to you, the programmer, to make them that way. What is involved is a little forethought and planning to make your programs efficient, readable and correctable. This section is a compilation of techniques and examples that we hope will aid you in writing such programs.

The Problem Statement

The first step involved in writing a program is stating the problem to be solved. This may seem an obvious step, but often it is completely omitted. The programmer then finds that the program does not produce the desired results because the original objective was not clear. This first step is always important because it gives both a clear idea of the problem and a definite direction to its solution. It is only then that one can start the logical development of the program.

Example: Suppose we want to find the roots of the equation $ax^2 + bx + c = 0$, where a, b and c are constants. Our problem statement might then be: "With inputs of a, b and c, find both solutions to the equation $ax^2 + bx + c = 0$."

Again, such a statement may seem obvious, but it supplies the two fundamental quantities: 1) the data we must supply and 2) the output we desire. With these two essentials and the concept of a solution we can go on to the second step, that of designing the algorithm.

The Algorithm

An algorithm is not a program, but an outline of the logical steps required to solve the problem. Such an algorithm should be non-specific at first, solving the problem through logical steps but

leaving the details out. The necessary details will be filled in later as programming language, space, and personal preference dictate. For now, all we are trying to accomplish is to lay down the foundations of the solution. In the case at hand, we might choose to solve the problem using the quadratic equation, namely:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

With this equation to guide us, our initial algorithm might look like this:

Find $b^2 - 4ac$.

If the difference is positive, find $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

If the difference is negative, find $\frac{-b}{2a}$ and $\frac{\pm\sqrt{|b^2 - 4ac|}}{2a}$

In the case where $b^2 - 4ac$ is positive, the solution is in the form of two real roots. When the difference is negative, the solution is in the form of two complex numbers. Note that none of the above statements define actual program steps. What is defined is the sequence of events required to arrive at the solution.

Refinement of the basic algorithm will bring out repeating patterns and condensable sequences that suggest the actual program steps that will ultimately be used. The revision process will also help to keep your programming objectives in mind. One such refinement might be:

Using the hypothetical registers R_A, R_B, and R_C do the following:

1. Take the negative of b and divide it by twice a.
2. Store this result in R_A.
3. Square b, subtract four a times c from it and store the result in R_B.
4. Take the square root of the absolute value of R_B and divide it by twice a.

5. Store this result in R_C.
6. If the value in R_B is positive or zero add the numbers in R_A and R_C together and display the (real) result.
7. If the value in R_B is less than zero display the contents of R_A and R_C separately (the complex root).
8. Change the sign of the value in R_C and repeat the previous two steps.

In this refinement, we realize the importance of using registers to store intermediate results. Also, the operations are better defined and consideration is given to the actual output.

The degree of refinement of the final algorithm is up to personal preference, but the more refined the algorithm the less likely the program itself will have to be later modified in order to work properly. Our final algorithm takes us to a satisfactory level.

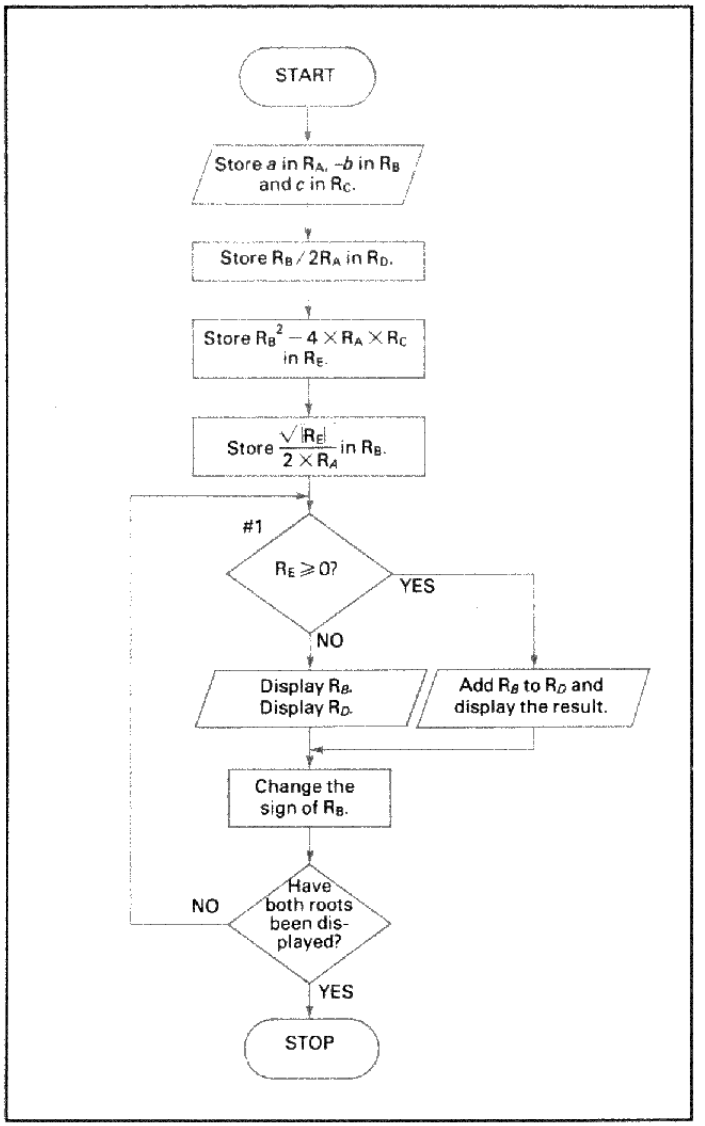
Using the hypothetical registers R_A, R_B, R_C, R_D and R_E do the following:

1. Store a in R_A, negative b in R_B and c in R_C.
2. Divide R_B by twice R_A and store the result in R_D.
3. Subtract four R_A times R_C from R_B squared and store the result in R_E.
4. Find the square root of the absolute value of R_E and divide it by twice R_A. Store this value in R_B.
5. If R_E is positive or zero add R_B to R_D and display the result.
6. If R_E is negative display R_B and R_D separately.
7. Change the sign of R_B and repeat the previous two steps.

This algorithm takes into account the initial storage of data and reuses R_B, thereby reducing the total number of storage registers needed.

Flowcharts

A useful way of visualizing your developing algorithm is to use a "flowchart". A flowchart is a diagram of the flow of the algorithm, giving form to the train of thought you are developing. An example that uses our developed algorithm is:



At first glance this may look complicated, but closer inspection reveals a simple logic. To read the chart, start at the upper left-hand corner and follow the arrows. Each box is connected to the next by one-way arrows. In other words, the flow is in one direction, generally from top to bottom.

Note that at the box marked #1 there is a choice of exits or a "branch". The direction of flow at this point is determined by the answer to the question asked within the box, in this case, "Is the value in R_E greater than or equal to zero?" An answer of "yes" channels the flow to the right while an answer of "no" channels the flow downward.

As you can see, a flowchart can be very useful in eliminating confusion, especially around branches and loops, where there is an option in the direction of program flow.

Subroutines

Perhaps the first thing you notice when previewing the Matrix Algebra program is its surprising length. It is so long that it cannot be contained, in its entirety, in the memory of the calculator. What is not immediately obvious is that it is also a highly condensed program. This is because the program takes advantage of the many repeated patterns involved in solving the problem. In fact, even the operation of taking the matrix's inverse is repeated. Therefore, this and several lesser functions have been incorporated into the program as subroutines.

Technically, in the language of the calculator, a subroutine can be any series of keystrokes that begins with a label ([LBL] *n*) and ends with a return ([RTN]) or the end of the program. These boundaries allow entry into and exit from the subroutine. (In fact, entry into a subroutine may be achieved at any line number. Refer to page 137, Indirect Line Number Branches and Subroutines.)

Access is gained to the subroutine via the [GSB] *n* (go to subroutine *n*) command. Notice that in the main body (steps 000 through 078) of the Matrix Algebra program [GSB] 8 is encountered eight times. Each time, the flow jumps from [GSB] 8 to [LBL] 8, completes the series of steps between [LBL] 8 and [RTN], and returns to the step immediately following [GSB] 8.

The usefulness of the subroutine becomes obvious first in its space savings. It is better to call subroutine 8 with eight [GSB]'s than to

rewrite the subroutine's steps eight times. What you also notice is an increase in organization from using a subroutine. The program has been broken down into its component parts. Each part, by itself, is easier to read and understand than the unsectioned program. Once the function of each part is understood, the program can be read as a whole with better understanding. This compartmentalization also simplifies error correction. Errors can be more easily isolated and the corrections are less likely to have adverse effects on other parts of the program.

[ISG] With [RCL] [(i)]

When looking for repeating patterns in your algorithms, it is also good practice to look for sequential storage and recall, i.e., [STO] 1 ... [STO] 2 ... [STO] 3, etc. Such sequences may be incorporated into subroutines via the [STO] [(i)] and [RCL] [(i)] functions of the HP-11C. This technique is effectively used in the Matrix Algebra program.

Routine "A" (lines 079 through 101) calculates the determinant of the 3 × 3 matrix stored in R_5 through $R_{.3}$. Mathematically, the determinant is calculated as:

$$\begin{vmatrix} R_5 & R_6 & R_7 \\ R_8 & R_9 & R_{.0} \\ R_{.1} & R_{.2} & R_{.3} \end{vmatrix} = \begin{aligned} & R_5(R_9 \times R_{.3} - R_{.0} \times R_{.2}) \\ & -R_6(R_8 \times R_{.3} - R_{.0} \times R_{.1}) \\ & +R_7(R_8 \times R_{.2} - R_9 \times R_{.1}). \end{aligned}$$

Rearrange this to bring out the pattern:

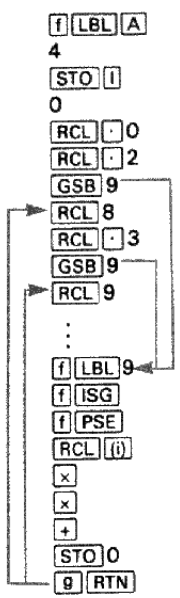
$$\begin{aligned} & -(R_{.0} \times R_{.2} \times R_5 + R_8 \times R_{.3} \times R_6 + R_{.1} \times R_9 \times R_7) \\ & +R_7 \times R_{.2} \times R_8 + R_5 \times R_{.3} \times R_9 + R_6 \times R_{.1} \times R_{.0} \end{aligned}$$

What results is not just the pattern R × R × R + but also the sequential recall R_5 to $R_{.0}$. These two features have been combined into subroutine 9 (lines 001 to 009).

In this subroutine [ISG] (Increment and Skip of Greater) is used to increment the I-register (refer to page 128, Incrementing and Decrementing the Index Register). Each call of the subroutine increases the value stored in R_I by 1 and recalls the value stored in the register indicated by the value in R_I ([RCL] [(i)]). Since the fractional portion of the number in the I-register is always zero and

the integer portion is always greater than zero, the step immediately following [ISG] is always skipped. For this reason a dummy step, [PSE] in this case, occupies that spot.

Stepping through the program will illustrate the subroutine's function.



Instructions	First Execution	Second Execution
f LBL A		
4		
STO I		
0		
RCL · 0		
RCL · 2		
GSB 9		
RCL 8		
RCL · 3		
GSB 9		
RCL 9		
⋮		
f LBL 9		
f ISG	I = 5	I = 6
f PSE	Skipped.	Skipped.
RCL (i)	Recalls R_5.	Recalls R_6.
×	$R_2 \times R_5$	$R_3 \times R_6$
×	$R_0 \times R_2 \times R_5$	$R_8 \times R_3 \times R_6$
+	$0 + R_0 \times R_2 \times R_5$	Adds to previous total.
STO 0	Stores total in R_0.	Stores new total in R_0.
g RTN		

The space savings is clear. Four steps would have been repeated six times, a total of twenty-four steps. The subroutine condensed this to ten (including 4 [STO] [I] and excluding [STO] 0 which would have been performed anyway).

Data Input

In writing a program, once you have decided on what data your program needs, you must then decide on how it will be stored. Indeed, there are several options depending on the space you have and the number of inputs to be stored.

In the Triangle Solutions program we have three inputs for each of the following five cases: *SSS*, *ASA*, *SAA*, *SAS*, and *SSA*. It is clear that we need five similar but definitely different sequences to input the data. In this case, since the HP-11C has five user-definable keys each key may be given the function of inputting the three variables in a particular case.

There are several methods of loading the variables via the user keys. One would be to store the three knowns in their appropriate registers by hand (e.g., S_1 [STO] 1, A_1 [STO] 2, etc.), then select the appropriate routine via the user keys to solve the particular case. Although this is a satisfactory method when dealing with a few variables or when there is not enough space for an input routine, it is tedious and does not take advantage of the time and effort-saving features of the HP-11C.

Another, more common method is the "stop-and-store". What is done is to select the user key corresponding to the case at hand. The program immediately halts waiting for you to key in the first variable. You do so and press [R/S] to resume program execution. The program stores the data in the appropriate register and halts awaiting the next input. This process repeats until all of the variables have been keyed in. More sophisticated versions of stop-and-store include input loops that prompt you for the input variables, and even allow you to review and modify them (see [LBL] A in The Systems of Linear Equations with Three Unknowns program).

The method we have chosen in the Triangle Solutions program is to load the stack with our input values (we only have three) and select the appropriate user key. The first function of each user key is to store the three inputs in appropriate registers via the sequence [STO] R_A, [R↓], [STO] R_B, [R↓], [STO] R_C. This method, though limited to programs requiring few inputs, is quick and easy, and does not require many program steps.

Looping

The Newton's Method program deals with the common problem of approximating a solution to an equation. Such approximations are necessary because finding an exact root to certain functions is often difficult and sometimes impossible.

In Newton's Method, we choose a point x at which the function $f(x)$ exists and, we hope, is close to a root of the function. (Applying a little basic algebra will help narrow the guessing range.) What goes on next is to adjust the guess, calculating a closer approximation of the root based on the initial guess. The calculated value is then used to calculate another closer approximation. This process is repeated with the new guess until, in the limit of an infinite number of repetitions, the exact solution may be reached.

We see that a section of the program must be repeated over and over again in succession until we get the answer we desire. Such a section is called a "loop". (The main loop in this program is between lines 026 and 052.) Of course, an infinite number of repetitions implies an infinite period of time, so we must set a boundary on the number of iterations performed.

One method of setting this boundary would be to insert a counter in the loop such that after each pass through the loop a certain register will have been incremented by one. The total number of repetitions is compared with a desired maximum. The program will then exit from the loop when the two values are equal. (In this program, the maximum is decremented each time and the loop terminates when the value has gone to zero. See line 051.) Notice that this method does not ensure any standard of accuracy. Also, depending on the function and the guess, finding a reasonable answer may take an unreasonable number of repetitions. In other words, stopping the process at even one-hundred repetitions may yield an answer so far from the actual root as to be useless.

A preferable method is to test the last two computed values to see whether or not they are significantly different. This difference, called the Δx *limit*, is up to the user and is input when initializing the program.

The format for such a loop is as follows:

1. Store the Δx *limit* and the initial guess.
2. Compute the first approximation from the guess and store it.

3. Compute the next approximation from the modified guess and store it.
4. Recall the value of the previous approximation and subtract it from the value of the present approximation.
5. Recall the Δx *limit* and compare its value with the magnitude of the difference just computed.
6. If the magnitude of the difference is greater than that of the Δx *limit* continue looping.
7. If the magnitude of the difference is less than that of the Δx *limit*, exit from the loop.

Notice that this method does not guarantee an exit from the loop. The first guess may have been very far from the actual root, or the root may not even exist. The best method then is a combination of the two, as is the method used by this program.

Not at all obvious is the fact that the nature of certain functions is such that phantom roots may appear. These phantom roots occur when the slope of the function is so large that the difference between two consecutive approximations falls within the Δx *limit*. This program provides a safeguard against this by testing not only the loop count and the Δx *limit* but also the value of the function at the point in question. The functional value is compared with the tolerance to see how close to zero the user wants the solution to be.

Flags

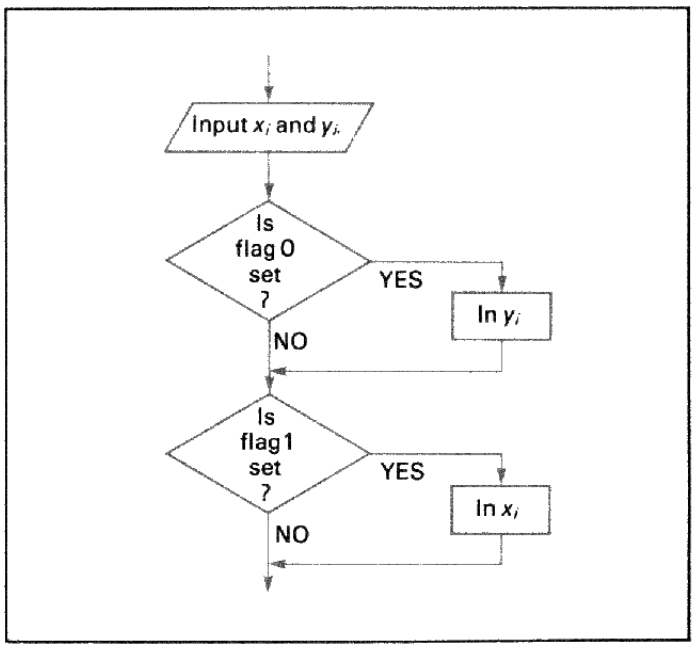
When writing a program that deals with different problems in similar but not identical fashions, flags can be used to control the differences in the treatment of the data.

Flags serve as reminders. For instance, when the program performs an optional initial calculation, a flag may be set. Later in the program we may need to know whether or not the operation was performed, so we test the flag, i.e., Is flag 0 set? (Was the operation performed?) In this way, flags are part of the decision-making power of the calcualtor.

In the Curve Fitting program, three very similar processes may be performed depending on the choice of curve to be fit. To illustrate the role of the flags, two more columns may be added to the chart on page 162.

Regression	A	X_i	Y_i	Code	Flag 0	Flag 1
Exponential	$\ln a$	x_i	$\ln y_i$	1	Set	Clear
Logarithmic	a	$\ln x_i$	y_i	2	Clear	Set
Power	$\ln a$	$\ln x_i$	$\ln y_i$	3	Set	Set

The two flags are initially set by the program and, depending on the choice of curve fit, one or the other may be cleared in accordance with the above table. The status of the two flags determines the treatment of the input data (x_i and y_i) as follows:



The treatment of the output value (A) is similar.

The use of flags here is very convenient. A problem that otherwise might have required three separate programs is solved very neatly in one. The use of flags can become a very powerful tool and will enhance your own personal programs.

Random Numbers

The HP-11C has a convenient built-in function that generates pseudo-random numbers ([RAN#]). By pseudo-random we mean that no calculator or computer, no matter how powerful, can generate a totally random number. The nature of the machine is such that it performs a sequence of established steps on an input and produces a predictable result. So the random number generator on the HP-11C takes a random "seed", stored by the user, and performs an operation on it to produce a generally unpredictable (by the user) result. This result is always a number between 0 and 1, exclusive of 1.

There are several common uses for random numbers, but they are very frequently found in game programs. This seems reasonable because no one likes to play against a completely predictable and redundant opponent. Submarine Hunt is a good example of such a game. The game requires a random initial placement of the submarine and random evasive moves.

Notice that the program requires two random integers, the x and y coordinates, that fall between 0 and 9 inclusive. But the random number generator only generates numbers between 0 and 1 exclusive of 1. This turns out to be no problem. Multiplying the output of the random number generator by 10 yields numbers from 0 to 10 exclusive of 10. Taking the integer portion of this result gives the set of whole numbers between 0 and 9 inclusive, which is what we want. The subroutine following [LBL] 9 shows the actual keystrokes.

To illustrate the ease in which different ranges of random numbers may be generated let's take another example. Suppose you need a real number between the limits 34.5 and 98.36 including the lower bound and excluding the upper bound. The routine need only generate values between 0 and 63.86 (98.36 - 34.50) to which the value 34.5 will be added. This range is easily generated by multiplying 63.86 by the output from the random number generator.

User-Definable Keys

Among the most useful features on the HP-11C are the five user-definable keys: [A], [B], [C], [D], and [E]. These keys are particularly useful for three applications:

1. Storing data in specific registers (see page 213).

2. Selecting execution of different routines within a program.
3. Selecting execution of different programs in program memory.

The first two applications are used in the Finance program.

Storing Data

In the Finance program, there are five possible inputs: periodic interest (i), number of periods (n), present value (PV), periodic payment (PMT), and future value (FV). Because there are five user-definable keys, we can assign a routine that stores input data in specific registers to each of these keys. For example, the routine associated with key [C] begins with [f] [LBL] [C], [STO] [3], [R/S]. Each time key [C] is pressed, the value in stack register X is stored in R_3. Up to 5 values, one associated with each key, may be stored in this way. In addition, these values may be stored in any order. Just key in the value to be stored and press the appropriate user-definable key.

Selecting Different Routines

A program may be written to calculate more than one value. The selection of which value is to be calculated may be done using the user-definable keys. The key pressed would indicate which value is to be calculated. In the Finance program, for example, the value PMT is calculated by pressing the following sequence: [D] (select PMT), [R/S] (calculate). When [D] is pressed, the user is telling the calculator to execute the routine beginning with [LBL] [D]. In this program, an inconsequential value is stored in R_4 and the calculator is positioned to the proper place to calculate PMT. The value stored is inconsequential because when [R/S] is pressed, PMT will be calculated and the value in R_4 will be over-written.

As you can see, the user-definable keys are very useful and can serve more than one purpose.

Appendix A

Error Conditions

If you attempt a calculation containing an improper operation—say division by zero—the display will show **Error** and a number. To clear an error message, press any key.

The following operations will display **Error** plus a number:

Error 0: Improper Mathematical Operation

Illegal argument to math routine;
[÷], where $x = 0$.
[y^x], where $y = 0$ and $x \leq 0$, or $y < 0$ and x is non-integer.
[$\sqrt{x}$], where $x < 0$.
[$1/x$], where $x = 0$.
[LOG], where $x \leq 0$.
[LN], where $x \leq 0$.
[SIN^{-1}], where $|x|$ is > 1.
[COS^{-1}], where $|x|$ is > 1.
[STO] [÷], where $x = 0$.
[Δ%], where the value in the Y-register is 0.
[HYP^{-1}] [COS], where $|x| < 1$.
[HYP^{-1}] [TAN], where $|x| > 1$.
[$C_{y,x}$], where:
1. x or y non-integer.
2. x or y less than zero.
3. $x > y$.
4. $x \geq 10^{10}$.

[$P_{y,x}$] same as [$C_{y,x}$]

Error 1: Storage Register Overflow

Storage register overflow (except [Σ+], [Σ−]). Magnitude of number in storage register would be larger than $9.999999999 \times 10^{99}$.

Error 2: Improper Statistical Operation

[$\bar{x}$] $n = 0$
[s] $n \leq 1$
[$\hat{y}$,r] $n \leq 1$
[L.R.] $n \leq 1$

Note: **Error 2** is also displayed if division by zero or the square root of a negative number would be required during computation with any of the following formulas:

$$s_x = \sqrt{\frac{M}{n(n-1)}} \qquad s_y = \sqrt{\frac{N}{n(n-1)}} \qquad r = \frac{P}{\sqrt{M \cdot N}}$$

$$A = \frac{P}{M} \qquad B = \frac{M\,\Sigma y - P\,\Sigma x}{n \cdot M}$$

(A and B are the values returned by the operation [L.R.], where $y = Ax + B$.)

$$\hat{y} = \frac{M\,\Sigma y + P\,(n \cdot x - \Sigma x)}{n \cdot M}$$

where:

$M = n\,\Sigma x^2 - (\Sigma x)^2$
$N = n\,\Sigma y^2 - (\Sigma y)^2$
$P = n\,\Sigma xy - \Sigma x\,\Sigma y$

Error 3: Improper Register Number

Named storage register currently converted to program memory, or nonexistent storage register.

Error 4: Improper Line Number or Label Call

Line number called for is currently unoccupied, or nonexistent (>203), attempt to load more than 203 lines of program memory, or label called does not exist.

Error 5: Subroutine Level Too Deep.

Subroutine nested more than four deep.

Error 6: Improper Flag Number:

Attempted flag name > 1.

Error 9: Service
Refer to page 233, Verifying Proper Operation
Pr Error

Continuous memory reset because of power failure.

Appendix B

Stack Lift and LAST X

Your HP-11C calculator has been designed to operate in a natural manner. As you have seen as you worked through this handbook, you are seldom required to think about the operation of the automatic memory stack—you merely work through calculations in the same way you would with a pencil and paper, performing one operation at a time.

There may be occasions, however, particularly as you program the HP-11C, when you wish to know the effect of a particular operation upon the stack. The following explanation should help you.

Digit Entry Termination

Most operations on the calculator, whether executed as instructions in a program or pressed from the keyboard, terminate digit entry. This means that the calculator knows that any digits you key in after any of these operations are part of a new number. (The [CHS], [.], [EEX], and [←] operations do not terminate digit entry.)

Stack Lift

There are three types of operations on the calculator, depending upon how they affect the stack lift. These are stack-*disabling* operations, stack-*enabling* operations, and *neutral* operations.

Disabling Operations

There are four stack-disabling operations on the calculator.* These operations *disable* the stack lift, so that a number keyed in after one of these disabling operations writes over the current number in the displayed X-register and the stack does not lift. These special disabling operations are:

[ENTER] [CLΣ] [Σ+] [Σ−]

* Refer to footnote, page 29.

Enabling Operations

Most of the operations on the keyboard, including one- and two-number mathematical functions like [x^2] and [×], are stack enabling operations. These operations *enable* the stack lift, so that a number keyed in after one of the enabling operations lifts the stack. Note that switching from Program mode to Run mode and turning the calculator off, then on, are enabling operations.

T →				
Z →				
Y →		4.0000	4.0000	53.1301
X →	4	4.0000	3	5.0000
Keys →	4	[ENTER] Stack disabled.	3 No stack lift.	[g] [→P] Stack enabled.

T →				
Z →				
Y →	53.1301	53.1301	53.1301	53.1301
X →	0.0000	7	1.0000	9
Keys →	[g] [CLx] Stack disabled.	7 No stack lift.	[Σ+] Stack disabled.	9 No stack lift.

Neutral Operations

Some operations, like [CHS] and [FIX], are neutral; that is, they do not alter the previous status of the stack lift. Thus, if you disable the stack lift by pressing [ENTER], then press [f] [FIX] *n* and key in a new number, that number will write over the number in the X-register and the stack will not lift. Similarly, if you have previously enabled the stack lift by executing, say [x^2], then execute a [FIX] instruction followed by a digit entry sequence, the stack will lift.

The following operations are neutral on the HP-11C:

[FIX]
[SCI]

[GTO] [·] *nnn*
[BST]

CLEAR [PREFIX]
CLEAR [REG]

[ENG]
[DEG]
[RAD]
[GRD]

[SST] (In Run mode [SST] may execute an instruction that does enable the stack.)

[MEM]
[USER]
[P/R]

CLEAR [Σ]
[CHS]*
[PREFIX]
[R/S]
[PSE]

LAST X

The following operations save *x* in LAST X:

[−]
[+]
[×]
[÷]
[→H.MS]
[→H]
[ABS]
[→R]
[→D]

[Σ+]
[Σ−]
[%]
[Δ%]
[x̂,r]
[x̄]
[FRAC]
[INT]
[LN]

[e^x]
[LOG]
[10^x]
[SIN]
[SIN^{-1}]
[COS]
[COS^{-1}]
[TAN]
[TAN^{-1}]

[$\sqrt{x}$]
[x^2]
[$1/x$]
[y^x]
[→R]
[→P]
[$P_{y,x}$]
[$C_{y,x}$]

[RND]
[HYP] [SIN]
[HYP] [COS]
[HYP] [TAN]
[HYP^{-1}] [SIN]
[HYP^{-1}] [COS]
[HYP^{-1}] [TAN]

---

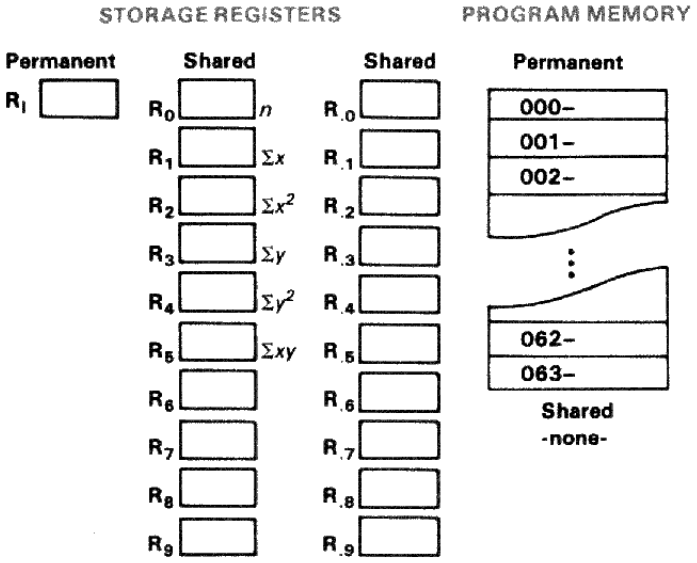
* [CHS] is neutral during digit entry of a number from keys, as in 1, 2, 3, [CHS] to enter −123; or 123 [EEX] 6 [CHS] to enter 123×10^{-6}. But otherwise, [CHS] enables the stack.

Appendix C

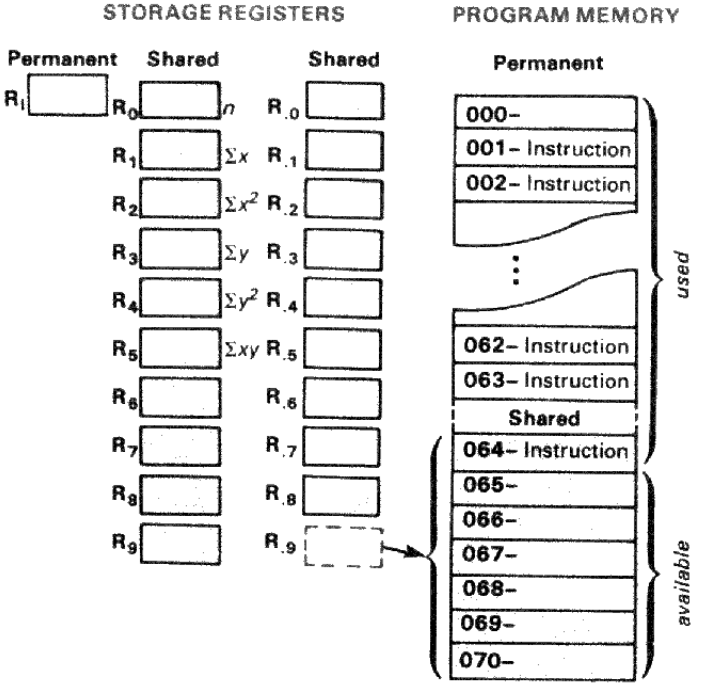
How Automatic Memory Reallocation Operates

Converting Storage Registers to Program Memory

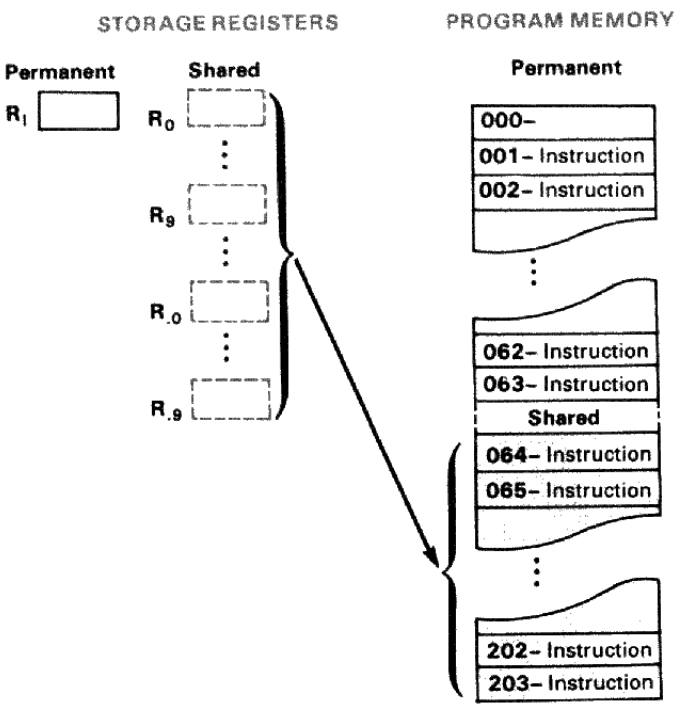
The automatic memory reallocation designed into your HP-11C gives you increased versatility by converting storage registers to lines of program memory only as needed. You begin programming with 63 lines of program memory and 20 storage registers (plus the Index register described in section 9). With up to 63 instructions in program memory, the allocation looks like this:



When you key in a 64th program instruction, storage register $R_{.9}$ converts to 7 lines of additional program memory. Now the memory allocation looks like this:



When you record a full 203 lines of program memory, the calculator's memory registers look like this:



Notice that instead of the original 21 storage registers (R_0 through R_9, $R_{.0}$ through $R_{.9}$, and R_I) we now have just the non-convertible R_I. What happened to storage registers R_0 through R_9 and $R_{.0}$ through $R_{.9}$? They were converted to program memory at the rate of seven lines per register. The table on page 76 shows the allocation of the lines of program memory to their respective storage registers.

As you can see, each time currently available programming space is filled, keying in another command automatically converts the next remaining storage register to seven more lines of program memory. For example, filling the first 70 lines and then keying a command into line 71 converts register $R_{.8}$ to 7 more lines of program memory (lines 71–77), and so on.

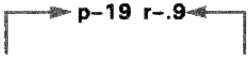
Note: Your HP-11C converts storage registers to program lines in reverse numerical order, from $R_{.9}$ to $R_{.0}$ and then from R_9 to R_0. For this reason it is good practice to program your [STO] and [RCL] operations using data registers in the opposite order; that is, beginning with register R_0. This procedure helps avoid accidentally programming [STO] and [RCL] for data registers which have been converted to lines of program memory. Remember also that the calculator does not retain data previously stored in registers that are later converted to lines of program memory.

Converting Program Memory to Storage Registers

Pressing [f] CLEAR [PRGM] in Program mode converts all shared program memory (lines 064–203) to storage registers R_0 through $R_{.9}$. However, deleting individual lines of program memory allows you to convert portions of shared memory to storage registers without clearing all of program memory. (Refer to page 105, Deleting Instructions.)

Using [MEM]

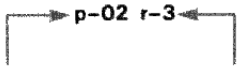
The [MEM] *(memory)* function on your calculator describes the current memory allocation in or out of program mode. For example, if you press [g] [MEM] with 44 lines of program memory occupied, you will see the following display:



Lines remaining to be occupied before the calculator automatically converts a storage register to 7 more program lines.

The next storage register to be converted.

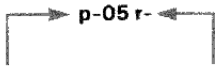
If you press [g] [MEM] with 173 lines of program memory occupied, you will see this display:



Lines remaining to be occupied before the calculator automatically converts a storage register to 7 more program lines.

The next storage register to be converted.

If you press [g] [MEM] with 198 lines of program memory occupied, you will see this display:



Lines remaining to be occupied before *all* lines of program memory are occupied.

No more storage registers can be converted to program memory.

Because R_I is a permanent storage register with special functions, it is not covered by the [MEM] operation.

> Note: Remember that the statistical functions involve registers R_0 through R_5. If one or more of these last six registers are converted to lines of program memory, attempts to execute statistical functions will result in an **Error 3** display.

Appendix D

Battery, Warranty, and Service Information

Batteries

The HP-11C is powered by three batteries. In "typical" use, the HP-11C has been designed to operate 6 months or more on a set of alkaline batteries. The batteries supplied with the calculator are alkaline, but silver-oxide batteries (which should last twice as long) can also be used.

A set of three fresh alkaline batteries will provide at least 80 hours of *continuous* program running (the most power-consuming kind of calculator use*). A set of three fresh silver-oxide batteries will provide at least 180 hours of *continuous* program running. If the calculator is being used to perform operations other than running programs, it uses much less power. When only the display is on—that is, if you are not pressing keys or running programs—very little power is consumed.

If the calculator remains turned off, a set of fresh batteries will preserve the contents of Continuous Memory for as long as the batteries would last outside of the calculator—at least 1½ years for alkaline batteries or at least 2 years for silver-oxide batteries.

The actual lifetime of the batteries depends on how often you use the calculator, whether you use it more for running programs or more for manual calculations, and which functions vou use.*

WARNING

Do not attempt to recharge the batteries; do not store batteries near a source of high heat; do not dispose of batteries in fire. Doing so may cause the batteries to leak or explode.

* Power consumption in the HP-11C depends on the mode of calculator use: off (with Continuous Memory preserved); idle (with only the display on); or "operating" (running a program, performing a calculation, or having a key pressed). While the calculator is turned on, typical calculator use is a mixture of idle time and "operating" time. Therefore, the actual lifetime of the batteries depends on how much time the calculator spends in each of the three modes.

The batteries supplied with the calculator, as well as the batteries listed below for replacement, are *not* rechargeable.

The following batteries are recommended for replacement in your HP-11C:

Alkaline	Silver-Oxide
Eveready A76*	Eveready 357*
Union Carbide (UCAR) A76	Union Carbide (UCAR) 357
National or Panasonic LR44	

Low-Power Indication

An asterisk (*) flashing in the lower-left corner of the display when the calculator is on signifies that the available battery power is nearly exhausted.

With alkaline batteries installed:

- The calculator can be used for at least 2 hours of continuous program running after the asterisk first appears.†
- If the calculator remains turned off, the contents of its Continuous Memory will be preserved for at least 1 month after the asterisk first appears.

With silver-oxide batteries installed:

- The calculator can be used for at least 15 minutes of continuous program running after the asterisk first appears.†
- If the calculator remains turned off, the contents of its Continuous Memory will be preserved for at least 1 week after the asterisk first appears.

Installing New Batteries

The contents of the calculator's Continuous Memory are preserved for a short time while the batteries are out of the calculator (provided that you turn off the calculator before removing the batteries). This allows you ample time to replace the batteries

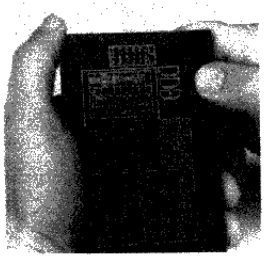
* Not available in the United Kingdom or Republic of Ireland.

† Note that this time is the minimum available for *continuous program running*—that is, while continuously "operating" (as described in the footnote on the preceeding page). If you are using the calculator for manual calculations—a mixture of the idle and "operating" modes—the calculator can be used for a much longer time after the asterisk first appears.

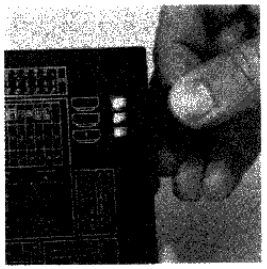
without losing data or programs. If the batteries are left out of the calculator for an extended period, the contents of Continuous Memory may be lost.

To install new batteries, use the following procedure:

1. **Ensure that the calculator is off.**
2. **Holding the calculator as shown, press outward on the battery compartment door until it opens slightly.**

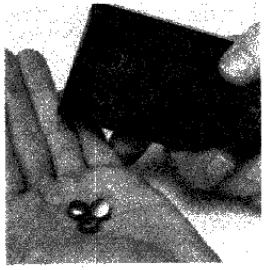


3. **Grasp the outer edge of the battery compartment door, then tilt it up and out of the calculator.**



Note: In the next two steps, be careful not to press any keys while batteries are out of the calculator. If you do so, the contents of Continuous Memory may be lost and keyboard control may be lost (that is, the calculator may not respond to keystrokes).

4. **Turn the calculator over and gently shake, allowing the batteries to fall into the palm of your hand.**

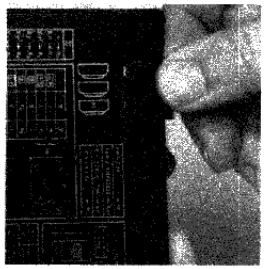


CAUTION

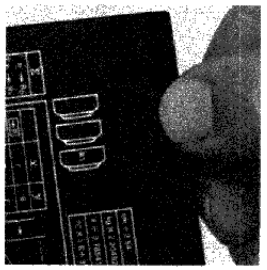
In the next step, replace *all three* batteries with fresh ones. If you replace only one or two of the batteries, an old battery may leak. Furthermore, be careful not to insert any battery backwards. If you do so, the contents of Continuous Memory may be lost and the batteries may be damaged.

5. **Insert three new batteries under the plastic flap or flaps shielding the battery compartment. They should be positioned with their flat sides (the sides marked +) facing *toward* the nearby rubber foot, as shown in the illustration on the calculator case.**

6. **Insert the tab of the battery compartment door into the slot in the calculator case.**



7. **Lower the battery compartment door until it is flush with the case, then push the door inward until it is tightly shut.**
8. **Press [ON] to turn the calculator on. If for any reason Continuous Memory has been reset (that is, if its contents have been lost), the display will show Pr Error. Pressing any key will clear this message from the display.**



Verifying Proper Operation (Self-Tests)

If it appears that the calculator will not turn on or otherwise is not operating properly, use the following procedures.

For a calculator that does not respond to keystrokes:

1. Press the [y^x] and [ON] keys simultaneously and release them. This will alter the contents of the X-register, so clear the X-register afterward.

2. If the calculator still does not respond to keystrokes, remove and reinsert the batteries. Make sure that the batteries are properly positioned in the battery compartment: the flat sides (the sides marked +) should all be facing *toward* the nearby rubber foot.

3. If the calculator still does not respond to keystrokes, *leave the batteries in the compartment* and short the battery terminals together. (The batteries must remain in place to prevent possible internal damage to the calculator.) With a paper clip or a piece of wire, briefly connect the terminals. *Only momentary contact is required.* The terminals are matching metal strips, or a combination of one spring and one hard edged tab located at either end of the battery compartment. After you do this, the contents of Continuous Memory will be lost, and you may need to press the [ON] key more than once to turn the calculator back on.

4. If the calculator does not turn on, install fresh batteries. If there is still no response, the calculator requires service.

For a calculator that does respond to keystrokes:

1. With the calculator off, hold down the [ON] key and press [×].

2. Release the [ON] key, then release the [×] key. This initiates a complete test of the calculator's electronic circuitry. If everything is working correctly, within about 25 seconds (during which the word **running** flashes) the display should show **−8.8.8.8.8.8.8.8.8.8.**, and all of the status indicators (except the * low-power indicator) should turn on.* If the display shows **Error 9**, goes blank, or otherwise does not show the proper result, the calculator requires service.†

Note: Tests of the calculator's electronics are also performed if the [+] key or the [÷] key is held down when [ON] is released.†‡ These tests are included in the calculator to be used in verifying that it is operating properly during manufacture and service.

If you had suspected that the calculator was not working properly but the proper display was obtained in step 2, it is likely that you made an error in operating the calculator. We suggest you reread the section in this handbook applicable to your calculation. If you still experience difficulty, write or telephone Hewlett-Packard at an address or phone number listed under Service (page 237).

---

*The status indicators turned on at the end of this test include some that normally are not displayed on the HP-11C.

†If the calculator displays **Error 9** as a result of the [ON]/[×] test or the [ON]/[+] test but you wish to continue using your calculator, you should reset Continuous Memory as described on page 20.

‡The [ON]/[÷] combination initiates a test that is similar to that described above, but continues indefinitely. The test can be terminated by pressing any key, which will halt the test within 25 seconds. The [ON]/[÷] combination initiates a test of the keyboard and the display. When the [ON] key is released, certain segments in the display will be lit. To run the test, the keys are pressed in order from left to right along each row, from the top row to the bottom row. As each key is pressed, different segments in the display are lit. If the calculator is operating properly *and all the keys are pressed in the proper order*, the calculator will display **11** after the last key is pressed. (The [ENTER] key should be pressed both with the third-row keys and with the fourth-row keys.) If the calculator is not working properly, *or if a key is pressed out of order*, the calculator will display **Error 9**. *Note that if this error display results from an incorrect key being pressed, this does* ***not*** *indicate that your calculator requires service.* This test can be terminated by pressing any key out of order (which will, of course, result in the **Error 9** display). Both the **Error 9** display and the **11** display can be cleared by pressing any key.

Limited One-Year Warranty

What We Will Do

The HP-11C (except for the batteries, or damage caused by the batteries) is warranted by Hewlett-Packard against defects in materials and workmanship for one year from the date of original purchase. If you sell your unit or give it as a gift, the warranty is automatically transferred to the new owner and remains in effect for the original one-year period. During the warranty period, we will repair or, at our option, replace at no charge a product that proves to be defective, provided you return the product, shipping prepaid, to a Hewlett-Packard service center.

What Is Not Covered

Batteries, and damage caused by the batteries, are not covered by the Hewlett-Packard warranty. Check with the battery manufacturer about battery and battery leakage warranties.

This warranty does not apply if the product has been damaged by accident or misuse or as the result of service or modification by other than an authorized Hewlett-Packard service center.

No other express warranty is given. The repair or replacement of a product is your exclusive remedy. **ANY OTHER IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS IS LIMITED TO THE ONE-YEAR DURATION OF THIS WRITTEN WARRANTY.** Some states, provinces, or countries do not allow limitations on how long an implied warranty lasts, so the above limitation may not apply to you. **IN NO EVENT SHALL HEWLETT-PACKARD COMPANY BE LIABLE FOR CONSEQUENTIAL DAMAGES.** Some states, provinces, or countries do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

This warranty gives you specific legal rights, and you may also have other rights which vary from state to state, province to province, or country to country.

Warranty for Consumer Transactions in the United Kingdom

This warranty shall not apply to consumer transactions and shall not affect the statutory rights of a consumer. In relation to such transactions, the rights and obligations of Seller and Buyer shall be determined by statute.

Obligation to Make Changes

Products are sold on the basis of specifications applicable at the time of manufacture. Hewlett-Packard shall have no obligation to modify or update products once sold.

Warranty Information

If you have any questions concerning this warranty, please contact an authorized Hewlett-Packard dealer or a Hewlett-Packard sales and service office. Should you be unable to contact them, please contact:

- In the United States:

Hewlett-Packard
Calculator Service Center
1030 N.E. Circle Blvd.
Corvallis, OR 97330
Telephone: (503) 757-2002

- In Europe:

Hewlett-Packard S.A.
150 route du Nant-d'Avril
P.O. Box
CH-1217 Meyrin
Geneva
Switzerland
Telephone: (022) 83 81 11

Note: Do *not* send calculators to this address for repair.

- In other countries:

Hewlett-Packard Intercontinental
3495 Deer Creek Rd.
Palo Alto, California 94304
U.S.A.
Telephone: (415) 857-1501

Note: Do *not* send calculators to this address for repair.

Service

Hewlett-Packard maintains service centers in most major countries throughout the world. You may have your unit repaired at a Hewlett-Packard service center any time it needs service, whether the unit is under warranty or not. There is a charge for repairs after the one-year warranty period.

Hewlett-Packard calculator products normally are repaired and reshipped within five (5) working days of receipt at any service center. This is an average time and could possibly vary depending upon the time of year and work load at the service center. The total time you are without your unit will depend largely on the shipping time.

Obtaining Repair Service in the United States

The Hewlett-Packard United States Service Center for handheld and portable calculator products is located in Corvallis, Oregon:

Hewlett-Packard Company
P.O. Box 999
Corvallis, Oregon 97339, U.S.A.
or
1030 N.E. Circle Blvd.
Corvallis, Oregon 97330, U.S.A.
Telephone: (503) 757-2000

Obtaining Repair Service in Europe

Service centers are maintained at the following locations. For countries not listed, contact the dealer where you purchased your calculator.

AUSTRIA
HEWLETT-PACKARD Ges.m.b.H.
Kleinrechner-Service
Wagramerstrasse-Lieblgasse 1
A-1220 Wien (Vienna)
Telephone: (0222) 23 65 11

BELGIUM
HEWLETT-PACKARD BELGIUM SA/NV
Woluwedal 100
B-1200 Brussels
Telephone: (02) 762 32 00

DENMARK
HEWLETT-PACKARD A/S
Datavej 52
DK-3460 Birkerod (Copenhagen)
Telephone: (02) 81 66 40

EASTERN EUROPE
Refer to the address listed under Austria.

FINLAND
HEWLETT-PACKARD OY
Revontulentie 7
SF-02100 Espoo 10 (Helsinki)
Telephone: (90) 455 02 11

FRANCE
HEWLETT-PACKARD FRANCE
Division Informatique Personnelle
S.A.V. Calculateurs de Poche
F-91947 Les Ulis Cedex
Telephone: (6) 907 78 25

GERMANY
HEWLETT-PACKARD GmbH
Kleinrechner-Service
Vertriebszentrale
Berner Strasse 117
Postfach 560 140
D-6000 Frankfurt 56
Telephone: (611) 50041

ITALY
HEWLETT-PACKARD ITALIANA S P.A.
Casella postale 3645 (Milano)
Via G. Di Vittorio, 9
I-20063 Cernusco Sul Naviglio (Milan)
Telephone: (2) 90 36 91

NETHERLANDS
HEWLETT-PACKARD NEDERLAND B.V.
Van Heuven Goedhartlaan 121
NL-1181 KK Amstelveen (Amsterdam)
P.O. Box 667
Telephone: (020) 472021

NORWAY
HEWLETT-PACKARD NORGE A/S
P.O. Box 34
Oesterndalen 18
N-1345 Oesteraas (Oslo)
Telephone: (2) 17 11 80

SPAIN
HEWLETT-PACKARD ESPANOLA S.A
Calle Jerez 3
E-Madrid 16
Telephone: (1) 458 2600

SWEDEN
HEWLETT-PACKARD SVERIGE AB
Skalholtsgatan 9, Kista
Box 19
S-163 93 Spanga (Stockholm)
Telephone: (08) 750 20 00

SWITZERLAND
HEWLETT-PACKARD (SCHWEIZ) AG
Kleinrechner-Service
Allmend 2
CH-8967 Widen
Telephone: (057) 31 21 11

UNITED KINGDOM
HEWLETT-PACKARD Ltd
King Street Lane
GB-Winnersh, Wokingham
Berkshire RG11 5AR
Telephone: (0734) 784 774

International Service Information

Not all Hewlett-Packard service centers offer service for all models of HP calculator products. However, if you bought your product from an authorized Hewlett-Packard dealer, you can be sure that service is available in the country where you bought it.

If you happen to be outside of the country where you bought your unit, you can contact the local Hewlett-Packard service center to see if service is available for it. If service is unavailable, please ship the unit to the address listed above under Obtaining Repair Service in the United States. A list of service centers for other countries can be obtained by writing to that address.

All shipping, reimportation arrangements, and customs costs are your responsibility.

Service Repair Charge

There is a standard repair charge for out-of-warranty repairs. The repair charges include all labor and materials. In the United States, the full charge is subject to the customer's local sales tax. In

European countries, the full charge is subject to Value Added Tax (VAT) and similar taxes wherever applicable. All such taxes will appear as separate items on invoiced amounts.

Calculator products damaged by accident or misuse are not covered by the fixed repair charges. In these situations, repair charges will be individually determined based on time and material.

Service Warranty

Any out-of-warranty repairs are warranted against defects in materials and workmanship for a period of 90 days from date of service.

Shipping Instructions

Should your unit require service, return it with the following items:

- A completed Service Card, including a description of the problem.
- A sales receipt or other documentary proof of purchase date if the one-year warranty has not expired.

The product, the Service Card, a brief description of the problem, and (if required) the proof of purchase date should be packaged in the original shipping case or other adequate protective packaging to prevent in-transit damage. Such damage is not covered by the one-year limited warranty; Hewlett-Packard suggests that you insure the shipment to the service center. The packaged unit should be shipped to the nearest Hewlett-Packard designated collection point or service center. Contact your dealer for assistance. (If you are not in the country where you originally purchased the unit, refer to International Service Information above.)

Whether the unit is under warranty or not, it is your responsibility to pay shipping charges for delivery to the Hewlett-Packard service center.

After warranty repairs are completed, the service center returns the unit with postage prepaid. On out-of-warranty repairs in the United States and some other countries, the unit is returned C.O.D. (covering shipping costs and the service charge).

Further Information

Service contracts are available. For information about service contracts, please contact the Calculator Service Center in Corvallis, Oregon.

Calculator product circuitry and design are proprietary to Hewlett-Packard, and service manuals are not available to customers.

Should other problems or questions arise regarding repairs, please call your nearest Hewlett-Packard service center.

When You Need Help

Technical Assistance. For technical assistance with this product, call:

(503) 757-2004
8 a.m. to 3 p.m.
Pacific time

or write to:

Hewlett-Packard Co.
Portable Computer Division
Calculator Technical Support
1000 N.E. Circle Blvd.
Corvallis, OR 97330

Product Information. For information about Hewlett-Packard products and prices, contact your local Hewlett-Packard dealer. For the name of the dealer nearest you, or to order free literature about Hewlett-Packard products, call toll free:

(800) FOR-HPPC
(800) 367-4772

or write to:

Hewlett-Packard Co.
Personal Computer Group
PCG Telemarketing
10520 Ridgeview Court
Cupertino, CA 95014

Temperature Specifications

- Operating: 0° to 55° C (32° to 131° F)
- Storage: −40° to 65° C (−40° to 149° F)

Potential for Radio/Television Interference (for U.S.A. Only)

The HP-11C generates and uses radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is not guarantee that interference will not occur in a particular installation. If your HP-11C does cause interference to radio or television reception, which can be determined by turning the calculator off and on, you are encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving antenna.
- Relocate the calculator with respect to the receiver.
- Move the calculator away from the receiver.

If necessary, you should consult your dealer or an experienced radio/television technician for additional suggestions. You may find the following booklet prepared by the Federal Communications Commission helpful: *How to Identify and Resolve Radio-TV Interference Problems*. This booklet is available from the U.S. Government Printing Office, Washington, D.C. 20402, Stock No. 004-000-00345-4.

Programming Techniques Index

The following is a short index of several useful functions and practices used in Part III: Programmed Problem Solving. Each entry is followed by the page numbers (in **bold** type) and the respective program line numbers (in regular type) that illustrate the use of the technique.

Function Key Index

inverse hyperbolic sine, inverse hyperbolic cosine, or inverse hyperbolic tangent, respectively, of number in displayed X-register **(Page 48)**.

Index Register Control

[I] Index (R_I)-register. Used with [STO] and [RCL] as a simple data storage register. Used also to hold control number for increment/decrement operations and for indirect control of display and program execution **(Page 127)**.

[x≷I] Exchanges number in displayed X-register with number in R_I **(Page 127)**.

[(i)] Indirect Operations command. Used with [STO] and [RCL] for indirect data storage, recall, and storage register arithmetic **(Page 130)**.

[x≷(i)] Exchanges contents of displayed X-register with those of the register addressed by the value stored in R_I **(Page 130)**.

[DSE] Decrement, Skip If Equal or Less Than. Subtracts specified decrement value from counter value. Skips one program line if new counter value is equal to or less than specified test value **(Page 128)**.

[ISG] Increment, Skip If Greater. Adds specified increment value to counter value. Skips one program line if new counter value is greater than specified test value **(Page 128)**.

Logarithmic and Exponential

[LN] Computes natural logarithm of number in displayed X-register **(Page 48)**.

[e^x] Natural Antilogarithm. Raises *e* to power of number in displayed X-register **(Page 48)**.

[LOG] Computes common logarithm (base 10) of number in displayed X-register **(Page 48)**.

[10^x] Common Antilogarithm. Raises 10 to power of number in displayed X-register **(Page 48)**.

[y^x] Raises number in Y-register to power of number in displayed X-register **(Page 49)**.

Mathematics

[−] [+] [×] [÷] Arithmetic operators **(Page 22)**.

[$\sqrt{x}$] Computes square root of number in displayed X-register **(Page 44)**.

[x^2] Computes square of number in displayed X-register **(Page 44)**.

[x!] Calculates factorial *x*! or Gamma function $\Gamma(1+x)$ **(Page 43)**.

[1/x] Computes reciprocal of number in displayed X-register **(Page 43)**.

[π] Places value of pi (3.141592654) in displayed X-register **(Page 42)**.

Number Alteration

[RND] Rounds mantissa of 10-digit number in X-register to match display setting **(Page 42)**.

[ABS] Gives absolute value of number in displayed X-register **(Page 42)**.

[INT] Leaves only integer portion of number in displayed X-register by truncating fractional portion **(Page 42)**.

[FRAC] Leaves only fractional portion of number in displayed X-register by truncating integer portion **(Page 42)**.

Percentage

[%] Percent. Computes *x*% of value in the Y-register **(Page 49)**.

[Δ%] Percent Difference. Computes percent of change between number in Y-register and number in displayed X-register **(Page 50)**.

Prefix Keys

[f] Pressed before a function key, selects gold function printed above that key **(Page 21)**.

[g] Pressed before a function key, selects blue function printed on the slanted face of that key **(Page 21)**.

CLEAR [PREFIX] cancels [f] and [g] prefix keystrokes and partially entered instructions such as [f] [SCI] or [g] [HYP^{-1}]. Also displays 10-digit mantissa of number in displayed X-register **(Page 21)**.

Probability

[$P_{y,x}$] Permutation. Computes the number of possible ordered choices of *y* different items taken in quantities of *x* items at a time without repetitions **(Page 52)**.

[$C_{y,x}$] Combination. Computes the number of possible sets of *y* different items taken in quantities of *x* items at a time without repetitions or order **(Page 53)**.

Stack Manipulation

[x≷y] Exchanges contents of X- and Y-stack registers **(Page 28)**.

[R↓] Rolls down contents of stack **(Page 28)**.

[R↑] Rolls up contents of stack **(Page 28)**.

[CLx] Clears contents of displayed X-register to zero **(Page 17)**.

Statistics

CLEAR [Σ] Clears the statistics registers (R_0 through R_5) and the stack registers to zero **(Page 55)**.

[Σ+] Accumulates statistics of numbers from X- and Y-registers into storage registers R_0 through R_5 **(Page 55)**.

[Σ−] Subtracts statistics of numbers in X- and Y-registers from storage registers R_0 through

R_5 for correcting [Σ+] accumulations **(Page 58)**.

[x̄,ȳ] Computes mean (average) of *x* and *y* values accumulated by [Σ+] **(Page 60)**.

[s] Computes sample standard deviations of *x* and *y* values accumulated by [Σ+] **(Page 61)**.

[ŷ,r] Linear Estimate and Correlation Coefficient. Computes estimated value of *y* ($\hat{y}$) for a given value of *x* by least squares method and places result in displayed X-register. Computes the correlation coefficient (*r*) of the linear estimate data by measuring how closely the data pairs would, if plotted on a graph, represent a straight line, and places result in Y-register **(Page 64)**.

[L.R.] Linear Regression. Computes y-intercept (*A*) and slope (*A*) for linear function $y = AX + B$ that best approximates *x* and *y* values accumulated using [Σ+]. The value of the y-intercept is placed in the X-register; the value of the slope is placed in the Y-register **(Page 63)**.

[RAN#] Places a pseudo-random number in the displayed X-register; the pseudo random sequence is generated from a seed stored using [STO] [RAN#] **(Page 54)**.

Storage

[STO] Store. Followed by register address (0 through 9, .0 through .9, [I], or [(i)]), stores displayed number in the storage register specified. Also used to perform storage register arithmetic **(Page 37)**.

[RCL] Recall. Followed by address (0 through 9, .0 through .9, [I], or [(i)], recalls number from storage register specified into the displayed X-register **(Page 38)**.

CLEAR [REG] Clears contents of all storage registers to zero **(Page 38)**.

[LST x] Recalls number displayed before the previous function back into the displayed X-register **(Page 28)**.

Trigonometry

[DEG] Sets decimal degrees mode for trigonometric functions—indicated by absence of **GRAD** or **RAD** annunciator **(Page 45)**.

[RAD] Sets radians mode for trigonometric functions—indicated by **RAD** annunciator **(Page 45)**.

[GRD] Sets grads mode for trigonometric functions—indicated by **GRAD** annunciator **(Page 45)**.

[SIN], [COS], [TAN] Compute sine, cosine, or tangent, respectively, of number in displayed X-register **(Page 45)**.

[SIN^{-1}], [COS^{-1}], [TAN^{-1}] Compute arc sine, arc cosine, or arc tangent, respectively, of number in displayed X-register **(Page 45)**.

Programming Key Index

Program/Run mode. Pressing [g] [P/R] sets the calculator to Program mode—**PRGM** annunciator on—or Run mode—**PRGM** annunciator cleared—**(Page 78)**.

[MEM] Displays current status of program memory/storage register allocation **(Page 77)**.

[A] [B] [C] [D] [E] User-defined program keys for both program labels and program execution **(Page 79)**.

[USER] Places calculator in and out of User mode. In User mode the primary [√x̄], [e^x], [10^x], [y^x], and [1/x] key functions and the shifted [A] through [E] user-defined key functions are exchanged **(Page 79)**.

0 1 2 3 4 5 6 7 8 9 Label designations. When preceded by [LBL], define beginning of a routine **(Page 79)**.

[LBL] Label. When used with [A] through [E] or 0 through 9, denote the beginning of a program or subroutine **(Page 79)**.

[GTO] Go To. Used with [A] through [E], 0 through 9, or [I]. From the keyboard; causes calculator to search downward in program memory for designated label and halt. In a running program; causes calculator to transfer downward in program memory to designated label and resume program execution **(Page 112)**.

[GSB] Go to Subroutine. Used with [A] through [E], 0 through 9, or [I]. From the keyboard; causes calculator to search downward in program memory for designated label and begin program execution. In a running program;

causes calculator to transfer downward in program memory to designated label and continue execution until a [RTN] instruction transfers execution back to the first instruction after the [GSB] **(Page 119).**

[GTO] [·] *nnn* Go To Line Number. Positions calculator to the occupied line number specified by *nnn* **(Page 97).**

[BST] Back Step. Moves calculator back one line in program memory **(Page 97).**

[SST] Single Step. Moves calculator forward one line in program memory **(Page 97).**

[←] Backarrow. In Program mode, deletes displayed instruction from program memory. All subsequent instructions are moved up one line. In run mode, deletes digits or numbers from displayed X-register **(Page 17).**

CLEAR [PRGM] In Program mode, clears all instructions from program memory and resets calculator to line 000. In Run mode, only resets calculator to line 000 **(Page 78).**

[PSE] Pause. Halts program execution for about one second to display contents of X-register, then resumes execution **(Page 79).**

[R/S] Run/Stop. Begins program execution from current line number in program memory. Stops execution if program is running **(Page 79).**

[RTN] Return. Causes calculator to return from any line in program memory to line 000, or from subroutine to appropriate line elsewhere in program memory **(Page 79).**

[SF] Set Flag. Followed by flag designator (0 or 1), sets flag—true— **(Page 111).**

[CF] Clear Flag. Followed by flag designator (0 or 1), clears flag **(Page 111).**

[F?] Is Flag *n* True? Followed by flag designator (0 or 1), tests designated flag. If flag is set (true) the calculator executes the instruction in the next line of program memory. If flag is cleared (false), calculator skips one line in program memory before resuming execution **(Page 111).**

[$x \leq y$] [$x > y$] [$x \neq y$] [$x = y$] [$x < 0$] [$x > 0$] [$x \neq 0$] [$x = 0$] Conditionals. Each tests value in X-register against 0 or value in Y-register as indicated. If true, calculator executes instruction in next line of memory. If false, calculator skips one line in program memory before resuming execution **(Page 110).**

Subject Index

Page numbers in **bold** type indicate primary references; page numbers in regular type indicate secondary references.

S

T