

EXAMPLE PROGRAMS

Probably the best way to learn how to program the HP38G is to study some interesting working programs that actually use many of the basic programming constructions. We provide a few examples here along with line-by-line documentation.

Simply entering these programs will give you some practice in finding commands and familiarize you with the most important elements of program syntax. Some suggestions are made for ways that you could adapt some of the programs to perform other tasks.

HP38G EXAMPLE 1.

This program looks for perfect numbers (positive integers N with the property that $N = S$, the sum of all positive integer factors $< N$). It tests all the positive integers from 2 to 1000, displaying each along with the sum of its divisors. When $N = S$, the program displays that value with the label "PERFECT". Hence, you will see the last perfect number found by the program as it runs.

The program illustrates the use of **WHILE REPEAT** loops as well as **IF THEN** statements. It also illustrates the use of the **DISP** command for displaying text and values on specified lines of the display screen.

Program PERFECTN

```
ERASE:
2 STO> N:
WHILE N ≤ 1000
REPEAT
2 STO> F:
1 STO> S:
WHILE F ≤ √N
REPEAT
IF N MOD F = 0 THEN
S+F+(N/F) STO> S:
END:
F+1 STO> F:
END:
IF N=S THEN
DISP 3; "PERFECT: "N:
END:
DISP 1;"N= ";N:
DISP 2;"S= ";S:
N+1 STO> N:
```

NOTE:

Here are some possible adaptations you could make to this program:

- a) Save the list of the perfect numbers found in LO.*
- b) Allow the user to input the value of N (the largest integer checked).*
- c) Search for prime numbers (or other integers with special properties).*

HP38G EXAMPLE 2.

This program illustrates the use of PIXON to light specific pixels on the display screen, as well as the use of nested WHILE REPEAT loops. It assumes that your viewing window is the default window [-6.5, 6.5] by [-3.1, 3.2]. The program checks the coordinates of each pixel in the third quadrant and uses the integer part of the value of $X^2 + Y^2 \text{ MOD } 2$ (i.e., the remainder after division by 2) to decide whether to light up the pixels or not (1 for yes, 0 for no). It also lights up the other three symmetrically located pixels in the other three quadrants. It takes a while to run, but the result is a striking picture exhibiting some interesting diffraction patterns. (The picture is stored in the SKETCH view of the current ApLet.)

NOTE: Try replacing the expression $X^2 + Y^2$ with another expression symmetric in X and Y to produce a more exotic picture.

Program RIPPLES

ERASE:	<i>Clear the display screen</i>
$(X_{\max}-X_{\min})/130 \text{ STO} > H:$	<i>Calculate pixel width H</i>
$(Y_{\max}-Y_{\min})/63 \text{ STO} > K:$	<i>Calculate pixel height K</i>
$X_{\min} \text{ STO} > X:$	<i>Initialize X to left edge of screen</i>
WHILE $X \leq 0$	<i>OuterWHILE loop begins</i>
REPEAT	
$Y_{\min} \text{ STO} > Y:$	<i>Initialize Y to bottom of the screen</i>
WHILE $Y \leq 0$	<i>InnerWHILE loop begins</i>
REPEAT	
IF $\text{INT}(X^2+Y^2) \text{ MOD } 2$	<i>If the integer part of</i>
THEN	<i>$X^2 + Y^2 \text{ MOD } 2$ is 1, then</i>
PIXON X;Y:	<i>light up that pixel</i>
PIXON X;Ymax-(Y-Ymin):	<i>and the symmetrically located</i>
PIXON Xmax,-(X-Xmin);Y:	<i>pixels in the other 3 quadrants</i>
PIXON Xmax,(X-Xmin);	
Ymax-(Y,-Ymin):	
END:	
$Y+K \text{ STO} > Y:$	<i>increment pixel y~coordinate</i>
END:	<i>inner WHILE loop ends</i>
$X+H \text{ STO} > X:$	<i>Increment pixel x-coordinate</i>
END:	<i>Outer WHILE loop ends</i>
DISPLAY→ Page	<i>Save finished picture in SKETCH</i>

HP38G EXAMPLE 3.

This program "animates" a numerical limit by displaying a simultaneous readout of X and F(X) values. (In this case, the particular function used is $\text{SIN}(X)/X$.) Pay particular attention to the INPUT statements, for they provide a very simple way to give a professional look to your programs, complete with a title bar, a highlighted input form (with a default value supplied if you wish), and a help message to the user to prompt for the necessary input.

NOTE: This program can be adapted to investigate the limit of an arbitrary function by adding an input statement that allows the user to select a function in the SYMB menu. This program can also be used to investigate limits "at infinity" by using MAXREAL or -MAXREAL as the TARGET value A, and setting a suitably large increment H.

Example for Program LIMITS shown on next page.

HP38G EXAMPLE 3.

(continued)

Program LIMITS

INPUT X;	<i>Begin INPUT statement for X</i>
"LIMIT ANIMATION";	<i>Title bar</i>
"INITIAL X";	<i>Label for input blank for X</i>
"ENTER STARTING X";	<i>Help message to prompt user</i>
1:	<i>Default value for X</i>
INPUT A;	<i>Begin INPUT statement for A</i>
"LIMIT ANIMATION";	<i>Title bar (same as before)</i>
"TARGET";	<i>Label for input blank for A</i>
"ENTER DESTINATION";	<i>Help message to prompt user</i>
0 :	<i>Default value for A</i>
INPUT H;	<i>Begin INPUT statement for H</i>
"LIMIT ANIMATION";	<i>Title bar (same as before)</i>
"DELTA X";	<i>Label for input blank for H</i>
"ENTER INCREMENT";	<i>Help message to prompt user</i>
.1:	<i>Default value for H</i>
ERASE:	<i>Clears the display screen</i>
IF (X-A)H>0	<i>If H has the same sign as X-A,</i>
THEN -H STO> H:	<i>then reverse the sign of H</i>
END:	
WHILE 1	<i>This WHILE loop will run</i>
REPEAT	<i>until the user presses ON</i>
DISP 1;"X" "X:	<i>Display the value of X on line 1</i>
IFERR SIN(X)/X STO> Z	<i>If the evaluation of SIN(X)/X results</i>
THEN 0 STO> Z:	<i>in an error, set the value to 0</i>
END:	
DISP 3;	<i>Display the value of SIN(X)/X on line 3</i>
"SIN(X)/X "Z:	
X STO> 0.	<i>Store current value of X in 0</i>
X+H STO> X:	<i>Increment X by H</i>
IF (A-X) (A-0) ≤ 0	<i>If the new value of X is equal to A or</i>
THEN	<i>on the opposite side as the previous</i>
ERASE:	<i>value of X then clear the display,</i>
0 STO> X:	<i>restore the old value of X and</i>
H/10 STO> H:	<i>reduce increment by a factor of 10</i>
END:	
END:	

HP38G EXAMPLE 4.

This is actually a "suite" of four programs (three are used as subroutines of the main program, called ARCHIMEDES). They provide an exploration of the Archimedean method for approximating the area of a circle. This program shows how the SETVIEWS command operates. Look at PLOT to see the graph of a regular polygon that can be inscribed in the unit circle. Look at VIEWS after running the program ARCHIMEDES for special interactive options to change the number of sides and to see the computed area of the regular polygon.

Program ARCHIMEDES

SELECT Polar:	<i>Activate Polar ApLet</i>
1 STO> Angle:	<i>Set ApLet mode to degrees</i>
1 STO> HAngle:	<i>Set Home mode to degrees</i>
1 STO> R1 (θ):	<i>Set unit circle polar equation in R 1</i>
0 STO> θ min:	<i>Set starting angle to 0</i>
361 STO> θ max:	<i>Set ending angle to 361</i>
CHECK 1:	<i>Check R1 in SYMB</i>
3 STO> N:	<i>Initialize number of sides to N = 3</i>
120 STO> θ step:	<i>Initialize angle step to $360/3= 120$</i>
SETVIEWS	<i>Set up special views</i>
"DOUBLE # SIDES" ;	<i>First special view allows user</i>
ARCHI .1;1;	<i> to double number of sides</i>
"INPUT # SIDES" ;	<i>Second special view allows user</i>
ARCHI .2;1;	<i> to set number of views</i>
"AREA A=?" ;	<i>Third special view will display</i>
ARCHI .3;1	<i> area of the inscribed N-gon</i>

Program ARCHI.1

2*N STO> N:	<i>Double size of N, the number of sides</i>
360/N STO> θ step:	<i>Set the corresponding new angle step</i>

Program ARCHI.2

INPUT N;	<i>INPUT for N, number of sides</i>
"ARCHIMEDES" ;	<i>Title bar</i>
"# SIDES" ;	<i>Label for input blank</i>
"ENTER NUMBER OF SIDES" ;N:	<i>Help message to prompt user</i>
360/N STO> θ step:	<i>Set the corresponding new angle step</i>

Program ARCHI.3

ERASE:	<i>Erase the display</i>
DISP 1; "A= ".5*N*SIN(360/N):	<i>Compute the area of the polygon</i>
FREEZE:	<i>Freeze the display screen</i>