Programming the TI-8n where $0 \leq n \leq 6$

W. Ethan Duckworth, Loyola College, 2007

This guide gives you some directions on programming the TI-8n where $1 \leq n \leq 6$. Actually, most of it applies for $1 \leq n \leq 4$, but the TI-85 and TI-86 have a lot of the same commands, it's just that you can type them in directly.

The first thing to watch out for is that on the TI-8n, **programming commands must be entered from buttons and menus**, not just typed in directly. For instance, `Input` ≠`INPUT`. The first is the "input" command that you get from the buttons, the second is what it would look like if you typed it in one letter at a time (this is where the TI-85 and TI-86 are different, you can just type the commands in one letter at a time, but you need to be in lower case).

Anyway, here's how to enter programs and some of the commands for them.

Entering letters, quote marks, spaces

To enter a letter, hit `ALPHA`, then the button with the letter. On the TI-84 you can also hit `2ND`, `ALPHA` to call up `A-LOCK`. This keeps the calculator in alpha mode (until you hit `ALPHA` again) so you can keep hitting letters.

Quote marks: `"`. While in `ALPHA` or `A-LOCK`, hit the `+` button.

Spaces: While in `ALPHA` or `A-LOCK`, hit the `0` (zero) button.

Creating a new programm

To create a new program hit `PRGM`, go right to menu `NEW`, hit `1:Create New`, then type in the name, and hit enter.

Running a program

On the main screen you hit `PRGM`, then you choose the name of the program that you want to run. This should put the program name on the main screen, together with a prefix `prgm`. Hit enter (again), and the program should start running.

Various commands and where to find them

All of these commands need to be entered through single buttons and/or menus on the calculator. You can not type them in letter by letter.

Command	how to get it	when you can get it and use it
`Input`	Hit `PRGM`, go right to menu `I/O` (that stands for "Input/Output"), choose `1:Input`	While editing a program
$\mathtt{Y}_1$	You hit `VARS`, go right to menu `Y-VARS`, choose `1:Function`, then choose $\mathtt{Y}_1$	You can enter this in programs, or on the main screen, or even within other functions (like $\mathtt{Y}_2 = 3\mathtt{Y}_1$)
`>`	`2CND`, `MATH` (i.e. `TEST`) `3:>`	Anytime

Command	how to get it	when you can get it and use it
`If`	`PRGM`, `If`	While editing a program
`Disp`	`PRGM`, `I/O`, `3:Disp`	While editing a program
`While`	`PRGM`, go right to menu `CTL` (that stands for "Control"), `5:While`	While editing a program
`→`	`STO▸`	Anytime
`Then`	`PRGM`, `CTL`, `1:If`	While editing a program
`Else`	`PRGM`, `CTL`, `3:Else`	While editing a program
`End`	`PRGM`, `CTL`, `7:End`	While editing a program

Here are some sample programs to try running. The first one or two are very silly, but are worth practicing if you have no idea how to enter and run a program.

Program 1

```
:Input "NUMBER ",N
:Disp N+3
```

Comments: This program displays "NUMBER" on the screen and waits for the user to input a number. The input is stored in the variable N. Then the program displays $N + 3$.

Try running this program with a couple of different inputs.

Program 2

```
:Input "NUMBER ",N
:While N≤10
:N+1→N    } These steps get repeated until N > 10
:Disp N   }
:End
```

Comments: After this program gets the input it starts a loop. A "loop" is a set of commands that gets repeated. The command $N + 1 \rightarrow N$ is equivalent to saying "add 1 to N" (actually what it says is, add 1 to N, and then replace the old value of N with this new value).

Try running this program with various inputs (the best ones should be at least a few less than 10).

Program 3

This program actually does something and so I explain some of the code below.

When you run this program, it gets a number as input, and it checks to see of the number is too close to 0 (i.e. it checks if $|N| < 1$). If it is too close then the program displays the number and stops. Otherwise, the program moves N closer to zero one step at a time. It adds 1 if N is negative, and it substracts 1 if N is positive. It stops when N gets within a distance of 1 from zero. It displays N at each step so that you can see it moving closer to zero.

```
:Input "NUMBER ",N
:If abs(N)<1
:Then
:Disp N       } These commands apply if |N| < 1
:Stop
:End
:While abs(N)≥ 1        (The commands between here and the last End get repeated)
:If N>0
:Then
:N-1→N
:Else          } These commands are the If, Then  part
:N+1→N
:End
:Disp N                 (This displays N at each step)
:End                    (This is the end of the loop)
```

Program 4

This program is somewhat similar to the one that finds roots of an equation using the Intermediate Value Theorem and the method of bisection. The main difference is that here the program always homes in on an x-value, specifically $x = 0$. In fact, there is no mention of y-values at all in this program. Try running it with a few different inputs.

```
:Input "L NUMBER",L
:Input "R NUMBER",R
:If L*R>0
:Then
:Disp "ERROR"
:Disp "NEED L, R"
:Disp "TO BE OPPOSITE"   } These steps only get done if L * R > 0
:Disp "SIGNS"
:Stop
:End                        (This ends the If statement)
:While R-L≥.001
:(R+L)/2→M                  (This calculates the point half-way between L and R)
:If L*M>0
:Then
:M→L                        (This only gets executed if L * M > 0 (what does that mean?))
:Else
:M→R                        (This only gets executed if L * M ≤ 0)
:End                        (This ends the If statement)
:End                        (This ends the While loop)
:Disp (R+L)/2
```