Introduction to Programming the TI-92

John Hanna
Teaneck High School
100 Elizabeth Ave
Teaneck, NJ 07666
(201) 833-5567
email: tejohhan@bergen.org

Here's what a program screen will look like when you create a new program:

```
:xxx()
:Prgm
:
:  ¨  All code goes between Prgm and
:  ¨  EndPrgm
:  ¨ <<comment is "2nd-x"
:EndPrgm
:
```

To run a program

On the HOME SCREEN, type the name of the program WITH parentheses at the end. A program/function can have arguments, See the fib(n) function below. If a program or function expects arguments and you do not supply any, the error message 'Too few arguments' appears

Here are sample programs to copy and paste into programs that you must define. To copy these samples, put your TI-92 into "split screen" mode. Open this file on one side and a program window on the other side. Switch between the two windows with 2nd APPS and use ♦**C** and ♦**V** to copy and paste between the two windows.

Entering programs

When editing a program, type the program statements using the keyboard or select the commands from the menus on the toolbar, When you select from menus,both parts of a two-part statement (such as For() and EndFor) will be pasted into your program on two different lines with a space in between to begin typing statements.

Program I/O appears on its own screen, the PrgmIO screen. To switch between the PrgmIO screen and the HOME screen press F5.

Local variables are used so that variables do not get 'created' in the current folder. Local variables get destroyed when the program stops running.

comments are a neat way of writing notes in a program. The comment symbol, ©, is entered in a program by pressing 2-nd **x**

The 'Sequence' structure...

The programming sequence is one statement a time, top to bottom

```
:
:local a,b,c,d
:prompt a
:prompt b
:prompt c
:b^2-4*a*c»d
:disp "d=",d
:
```

The block statements are best entered from the F2:Control menu since the menu item will paste both the start and end of the block so you don't forget!

Branching with the If structure...

Paste this If...EndIf structure into the last program.

```
:
:If d<0 Then
:  disp "2 complex roots."
:ElseIf d=0 Then
:  disp "One real root."
:Else  ¨ d must be greater than 0
:  disp "Two real roots."
:EndIf
:
```

Looping structures

The while loop...

This is my favorite. Note the way the output can be formatted using strings and concatenation using **&** (2nd-h).

```
:  Input "Please enter a number. Enter 0 to end",x
:  While x≠ 0
:    Disp string(x)&" squared is "&string(x^2)
:    Input "Please enter another number. Enter 0 to end",x
:  EndWhile
:
```

The For loop…

A table can be built using 'output', but the positions are pixels, hence the 8*x.

```
:
:ClrIO
:For x,1,10,1
:  Output 8*x,1,x
:  Output 8*x,100,x^2
:EndFor
:
```

Generating Perfect Numbers using the "definition"...

Note the lines with the comment © (2-nd **x**) at the beginning. These were for 'debugging' purposes.

```
:
:Local total,perf,fact
:ClrIO
:For perf,3,500,1
:© Disp perf
:  1→total
:  For fact,2,perf/2,1
:    If mod(perf,fact)=0 Then
:      total+fact→total
:    EndIf
:  EndFor
:© Disp total
:© Pause
:  If total=perf Then
:    Disp perf
:  EndIf
:EndFor
:
```

Fibonacci Numbers: a program with only two variables.

Note the two store statements in the loop. They do not store the same thing in a and b because the first one changes the value of a.

```
:
:local a,b
:1→a
:1→b
:while b<100
:  disp a,b
:  a+b→a
```

```
: a+b→b
:endwhile
:disp a,b
:
```

Fibonacci function using recursion...

Create a new function using the program editor. Put n in as a parameter for the function (in the () after the functon name). The entire program is shown here:

```
:
:fib(n)
:Func
:If n<3 Then
: 1→n
:Else
: fib(n-1)+fib(n-2)→n
:EndIf
:Return n
:EndFunc
:
```

The "Return n" on the last line returns that value to the function call (the place where the function was used).

Creating a "custom" menu bar.

Edit and run this program, then press the custom (2-nd **3**)key. Don't confuse this command with the "ToolBar" structure. Any program can program the custom key. Name this program mycustm(). When you run the program the custom key brings up this configuration. Pressing the custom key again puts the regular menu back. This is "my" custom program

```
:
:Custom
: Title "John Hanna, T3"
: Title "Programs"
:   Item "customp()"
:   Item "fib(4)"
:   Item "hanoi(3)"
:   Item "han(3,1,3,2)"
:   Item "pyth(3,4)"
:   Item "implrel()"
: Title "Functions"
:   Item "vxx(u,l,a,b)"
:   Item "vxy(u,l,a,b)"
:EndCustm
:
```

When you choose an item from the custom menu while on the home screen, it gets pasted on the edit line.

Dialog boxes

```
:
:special=""
:  Dialog
:    Title "Hooray...dessert!"
:    text "Here's where you select dessert."
:    DropDown "Choose a topping",{"fudge","sprinkles","chocolate"},m
:    Request "Special topping",special
:  EndDlog
:
```

After the above code is executed, the variable 'm' contains a number (the number of the item chosen in the list) and variable 'special' contains the string entered. This works like the APPS / NEW dialog box

Graphing

Graphing functions in programs turns off the "y=" functions and creates an "internal" list of functions that will be graphed (see "Smart-Graph"). Experiment by graphing a function in a program, changing the function, and run the program again. Both functions will be graphed and the "y=" functions will not be graphed.

```
:
:Prgm
:  Graph ª2*sin(3*x)
:  DispG
:EndPrgm
:
:The following program creates 12
:pictures for  use with the "cyclepic"
:command...
:
: Local j
: For i,1,12,1
:  Graph (i-6)/2*sin(x)
:  string(i)→j
:  StoPic #("pict"&j)
:  ClrGraph
: EndFor
: CyclePic "pict",12,.25,5,ª1
:
```

Error Trapping

The "Try...Else...EndTry" feature will catch errors and allow you to handle the error condition. In the above program, we should "Try" the **cyclepic** command first, and if it fails, then we will regenerate the pictures.

```
:
: Try
:   CyclePic "pict",12,.25,5,ª1
: Else
:   ¨ the rest of the program here
: EndTry
:
:
:the end
```