Using the TI-92+ Calculator as a Tool for Illustrating Programming Concepts

By Paul Abraham
Kent State University Stark Campus

Introduction

As a mathematics professor who occasionally teaches introductory computer science courses, I have found the TI-92+ calculator to be valuable in illustrating basic and advanced programming concepts, especially when mathematics is an important underlying issue. The computer science courses I teach are an Introduction to Computer Science (CS 10051) and an Introduction to Object-Oriented Programming (CS 23021) at Kent State University Stark Campus.

CS 10051 surveys major areas of computer science like development of pseudocode and elementary structured (C++) programs, analysis of algorithms, data representation, basic computer architecture and assembly language and programming paradigms. The course lays the foundation for entry into many other computer science courses and is also the required computer science class for mathematics majors. Since the TI-92+ programming language is so natural it was particularly useful in demonstrating the progression in program development from pseudocode to code. A small segment of the course allows for a discussion of symbolic programming. As with the adopted text [1] for the course I presented a few examples of *Maple* code, but I also continued with TI-92+ examples to give students an idea of the current power of hand-held technology.

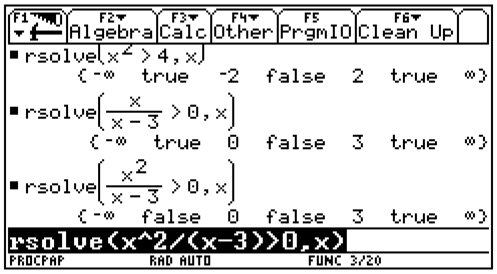
CS 23021 follows CS 10051 and is the first programming-directed course in the computer science major. As the title suggests, course discussions and student assignments progress to involve critical and varied use of objects. In particular string objects, vector objects (which behave much like TI-92+ lists), rational number objects and geometrical window objects like RectangleShape objects play a major role in the course and in the adopted text [2]. In this course a gave a few TI-92+ demonstrations which incorporated lists into symbolic programming to show how lists could be used to solve substantial mathematical problems. The function rsolve() (discussed in detail below) which determines the regions on the number line that pass/fail a given rational inequality was such an example. With College Algebra (Math 11011) as a prerequisite for CS 23021, students could certainly appreciate the significance of rsolve(). Since TI-92+ fractions and imaginary numbers are represented in standard form, I also used the TI-92+ to motivate the development of C++ rational and complex number objects.

Example Functions: rsolve(), mysort(), ispos() and gsolve()

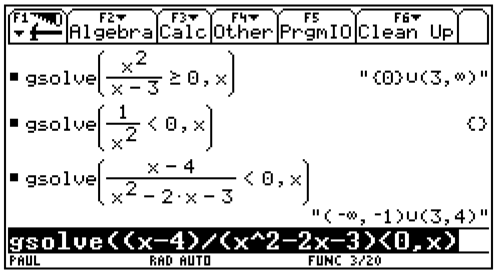
In this section the functions rsolve(), mysort() and ispos() are presented in entirety and a fourth function gsolve() is discussed. The code for all of these functions is available from the author and perhaps the Calculator Program Archive [4].

The function rsolve() solves any given rational inequality. The solution returned by rsolve() is only partial in the sense that it determines the regions on the number line that pass/fail the inequality, but does not decide which endpoints should be included. The function rsolve() returns the solution as a list which mimics the bottom line of sign chart solutions commonly constructed in textbooks, e.g. see [3].

The function gsolve() solves any given rational inequality completely. It returns the final answer using standard interval notation, singleton sets or the empty set as needed. The code for gsolve() is somewhat more tedious than rsolve() and will not be presented here since it occupies several pages. Runs of both rsolve() and gsolve() follow:



```
■ rsolve(x^2 > 4, x)
      {-∞  true  -2  false  2  true  ∞}
■ rsolve(x/(x-3) > 0, x)
      {-∞  true  0  false  3  true  ∞}
■ rsolve(x^2/(x-3) > 0, x)
      {-∞  false  0  false  3  true  ∞}
rsolve(x^2/(x-3))0,x)
PROCPAP        RAD AUTO        FUNC 3/20
```



```
■ gsolve(x^2/(x-3) ≥ 0, x)            "{0}∪(3,∞)"
■ gsolve(1/x^2 < 0, x)                         {}
■ gsolve((x-4)/(x^2-2·x-3) < 0, x)
                                 "(-∞,-1)∪(3,4)"
gsolve((x-4)/(x^2-2x-3)<0,x)
PAUL           RAD AUTO        FUNC 3/20
```

The comments in the rsolve() function code below explain fully how the function works. As the output of rsolve() illustrates, the solution is a list which contains different data types: extended real numbers and Boolean values. The ability for a list to contain values of different base types is extremely powerful and rare in common languages.

```
rsolve(ineq,var)
Func

¨This function determines the regions on the number line that satisfy the given rational inequality ineq in
variable var.

¨The result answ is returned as an ordered list of numbers with true or false between adjacent numbers. A
true between adjacent numbers indicates the corresponding region satisfies the inequality, else a false
appears.

Local  oneside,combined,top,bottom

¨The left and right sides of ineq will be combined with a common denominator into one side called
combined. 0 is understood to be the new right-hand side.  top is the numerator of combined and bottom is
the denominator of combined.

left(ineq)-right(ineq)»oneside
comDenom(oneside,var)»combined
getNum(combined)»top
getDenom(combined)»bottom

Local  s,z,endpt,answ

¨These variables represent lists. s will hold the zeros of bottom, ª¸ and ¸, z the zeros of top, and endpt will be
the ascending sorted list sUz of endpoints.
```

```
zeros(top,var)»z
zeros(bottom,var)»s
augment(s,{ª¸,¸})»s
z»endpt
augment(endpt,s)»endpt

mysort(endpt)»endpt

¨The code for mysort() occurs later in this article. It was necessary to write mysort since a function can not
call the built-in ascending sort program.

Local  a,b,i,testv

¨These variables are used to construct answ. a represents the left endpoint for the current region, b
represents the right endpoint, i maintains the current position in endpt and testv represents the test value for
the current region.

¨answ begins with left endpoint a -¸.  Looping ends when a reaches ¸.

1»i
ª¸»a
{ª¸}»answ

While  a ¸
  endpt[i+1]»b

¨ this block of code picks a test value appropriately.

  If a=ª¸ Then
    If b=¸ Then
      0»testv
    Else
      b-1»testv
    EndIf
  Else
    If b=¸ Then
      a+1»testv
    Else
      (a+b)/2»testv
    EndIf
  EndIf

¨ The next two lines insert true or false and the right endpoint into answ.  testv is plugged into ineq to decide
between true or false.

  augment(answ,{ineq|var=testv})»answ
  augment(answ,{b})»answ

¨ Finally, the right endpoint becomes the new left endpoint and the current position in endpt increases.

  b»a
  i+1»i
  EndWhile

Return  answ
EndFunc
```

As you may have noticed rsolve() requires a list to be sorted and calls the function mysort(). The function mysort() is a routine bubble sort. The code for mysort() follows:

```
mysort(l)
Func
¨This function bubble sorts list l.

Local  i,j,n,temp
dim(l)»n
1»i
While  i<n
1»j
  While  jœn-i
  If l[j]>l[j+1] Then
    l[j]»temp
    l[j+1]»l[j]
    temp»l[j+1]
  EndIf
  j+1»j
  EndWhile
i+1»i
EndWhile
Return  l
EndFunc
```

We conclude with the code for the function ispos() that illustrates another nice feature of lists on the TI-92+: the ability to apply a function or operator across a list. In calculus this feature could be used for example to differentiate a list of functions in one step, e.g., *d*({x,x^2,x^3},x) produces the list {1,2x,3x^2}. Similarly, the function ispos() determines which list entries are positive without explicit looping through the list and returns a list of true/false outcomes. The call ispos({1,-2,0}) produces {true,false,false}.

```
ispos(l)
Func
Local  z,n
dim(l)»n
¨n is the size of l
newList(n)»z
¨Created a list z of size n and by default filled it with 0's. The comparison of l and z entries is done next
coordinate-wise and the list of results is returned
Return  l>z
EndFunc
```

References:

[1] J.P. Cohoon and J.W. Davidson, C**++ Program Design: An Introduction to Programming and Object-Oriented Design**, 2nd edition, McGraw-Hill Inc., 1999.

[2] G.M. Schneider and J.L. Gersting**, An Invitation to Computer Science**, 2nd edition, Brooks/Cole Publishing, 1999.

[3] M. Sullivan, **College Algebra**, 5th edition, Prentice Hall, 1999.

[4] Texas Instruments Inc., Calculator Program Archive, Texas Instruments Inc., <http://www.ti.com/calc/docs/arch.htm>, 22 December 1999.

Author's Postal Address:

KSU Stark Campus
6000 Frank Avenue NW
Canton OH 44720

Email Address: pabraham@stark.kent.edu