NUMBRT1 – Factorization, GCD and Decimal to Fraction

© 2019 Valentín Albillo

Abstract

NUMBRT1 is a program written in 1979 for the HP-34C programmable calculator to factorize an integer into its prime factors, compute the greatest common divisor (GCD) of two integers, and find integer fractions which best approximate a given real value. It will also work in the HP-41C series with trivial changes. Three worked examples are included.

Keywords: *factorization, greatest common divisor, primes, decimal to fraction, programmable calculator, RPN, HP-34C, HP-41C.*

1. Introduction

NUMBERT1 is a *127-step RPN* program that I wrote in 1979 for the *HP-34C* calculator (will also run in the *HP-41C series* with trivial changes, see ***Note 1***), which implements several *Number Theory* algorithms to factorize a given integer into its prime factors, compute the greatest common divisor (GCD) of two integers, and find a series of integer fractions that best approximate a given real (decimal) value.

- *Factorization*

  Given an integer *N* as input, it will find and display all its prime factors. Each factor will be briefly paused in turn, in increasing value (repetitions allowed), and the last one will be marked negative to signal the end of the factorization. If the input number is prime, it will be displayed as its last and only factor, marked negative as well. The program checks all divisors up to the square root of *N*, skipping multiples of *2*, *3* and *5* for extra speed, but for large *N* the process might still take a long while.

- *Greatest common divisor (GCD)*
  Given two integers *M* and *N* as input, it will find their greatest common divisor using the *Euclidean algorithm*. If they're coprime (no common divisor) then the *GCD* will be returned as ***1***. Computation time is very fast regardless of the size of the input.

- *Decimal to fraction*

  Given a decimal (real) value *R* as input, it will find and display the integer numerator and denominator of fractions (called *convergents*) whose value approximate the input value with increasing accuracy. It uses an algorithm based in continued fractions, iteratively generating ever more accurate fractions and pausing for every iteration the corresponding numerator, denominator, the value of the fraction and the error of the approximation. The error will alternate between positive and negative, and finally zero.

  When the error becomes zero the program halts, displaying the numerator and the denominator of the last fraction generated. The user can stop the program early at any time by pressing **R/S** and the numerator and denominator of the last computed fraction can be displayed.

Remarks:

- *N* and *M* must be integer and positive; *R* (real value) can be negative.
- $N, M \leq 10^8$ is recommended.
- in the *HP-34C*, register $\boldsymbol{R_{.1}}$ can be recovered for storage if *step 127* is *deleted* and no steps follow.
- the three routines **A**, **B** and **9** have no common dependencies and thus can be loaded individually.

2. Program Listing

```
01 ♦LBL A     26   2        51 CHS      76 STO 6    101 PSE
02 STO 0      27 GSB 0 ▸    52 R/S      77 STO 7    102 RCL 7
03   0        28   6        53 ♦LBL B   78 R↓       103 RCL 5
04 STO 1      29 GSB 0 ▸    54 ENTER↑   79 ♦LBL 3   104 STO 7
05   2        30 GTO 1 ▸    55 ENTER↑   80 STO 0    105 RCL 2
06 GSB 0 ▸    31 ♦LBL 0     56 CLx      81 x↔y      106   x
07   1        32 STO+ 1     57   +      82 STO 1    107   +
08 GSB 0 ▸    33 RCL 0      58 R↓       83 x↔y      108 STO 5
09   2        34 RCL 1      59   ÷      84   ÷      109 PSE
10 GSB 0 ▸    35   ÷        60 INT      85 INT      110   ÷
11   2        36 LST x      61 R↑       86 STO 2    111 FIX 9
12 GSB 0 ▸    37 x>y        62   x      87 RCL 1    112 PSE
13 ♦LBL 1     38 GTO 2 ▸    63   -      88 x↔y      113 RCL 8
14   4        39 x↔y        64 x≠0      89 RCL 0    114 x↔y
15 GSB 0 ▸    40 FRAC       65 GTO B ▸  90   x      115   -
16   2        41 x≠0        66   +      91   -      116 PSE
17 GSB 0 ▸    42 RTN        67 RTN      92 STO 3    117 x=0
18   4        43 LST x      68 ♦LBL 9   93 RCL 4    118 GTO 8 ▸
19 GSB 0 ▸    44 STO 0      69 STO 8    94 RCL 6    119 RCL 0
20   2        45 RCL 1      70 CLΣ      95 STO 4    120 RCL 3
21 GSB 0 ▸    46 PSE        71 EEX      96 RCL 2    121 GTO 3 ▸
22   4        47   0        72   9      97   x      122 ♦LBL 8
23 GSB 0 ▸    48 GTO 0 ▸    73   x      98   +      123 FIX 0
24   6        49 ♦LBL 2     74 LST x    99 STO 6    124 RCL 6
25 GSB 0 ▸    50 RCL 0      75   1     100 FIX 0    125 PSE
                                                    126 RCL 5
                                                    127 RTN
```

- *127 program steps*
- *uses registers $R_0 - R_7$*
- *uses labels A,B,0,1,2,3,8,9*
- *FIX 0 recommended*
- *the symbols ♦ and ▸ are purely cosmetic to indicate branching*

- *all **PSE** steps can be changed to **R/S** to allow for more time to write down the results. Once done, press **R/S** to continue.*

3. Usage Instructions

- To factorize an integer *N*:

  N **A** → *first factor* → *2nd factor* → ... → *last factor (marked negative)*

- To find the greatest common divisor of two integers *M* and *N*:

  M **ENTER↑** *N* **B** → *greatest common divisor* *(will be **1** if M and N are coprime)*

- To find the best rational approximations to a decimal (real) value *R*:

  R **GOSUB 9** → $(num._1) \rightarrow (denom._1) \rightarrow (num._1/denom._1) \rightarrow (error_1)$
  → $(num._2) \rightarrow (denom._2) \rightarrow (num._2/denom._2) \rightarrow (error_2)$
  ... ...
  {when error$_n$ is finally 0} → $(num._n) \rightarrow (denom._n)$

  The computation can be halted at any time by pressing **R/S** and the last numerator and denominator computed can be shown by executing **GOSUB 8** → $(num._k) \rightarrow (denom._k)$

4. Examples

The following examples can be useful to check that the program is correctly entered and to understand its usage.

4.1 Example 1

Factorize 5040, 111121111 and 7332197, and check whether 72727, 10001, 11111 and 5555551 are prime or not.

5040	**A**	→ *2 → 2 → 2 → 2 → 3 → 3 → 5 → -7,*	*5040 =* 2^4 *x* 3^2 *x 5 x 7*
111121111	**A**	→ *41 → 73 → 137 → -271 {last factor},*	*111121111 = 41 x 73 x 137 x 271*
7332197	**A**	→ *983 → -7459 {ditto},*	*7332197 = 983 x 7459*

72727	**A**	→ *-72727,*	*72727*	*is prime*
10001	**A**	→ *73 → -137,*	*10001 = 73 x 137,*	*so not prime*
11111	**A**	→ *41 → -271,*	*11111 = 41 x 271,*	*so not prime*
5555551	**A**	→ *773 → -7187,*	*5555551 = 773 x 7187,*	*so not prime*

4.2 Example 2

Find the greatest common divisor (GCD) of 22519237 and 52940839.

22519237 **ENTER↑** 52940839 **B** → *527,* so *GCD(22519237, 52940839) = 527*

4.3 Example 3

Find fractions whose values best approximate π ≈ 3.141592654.

π **GOSUB 9** → *{we get the following results, shown here in tabular form}*

Numerator	*Denominator*	*Quotient*	*Error*
3	*1*	*3.000000000*	*+0.141592654*
22	*7*	*3.142857143*	*-0.001264489*
333	*106*	*3.141509434*	*+0.000083220*
355	*113*	*3.141592920*	*-0.000000266*
104348	***33215***	*3.**141592654***	***0***

and the program halts displaying ***104348*** and ***33215***, so the best fraction is *104348/33215 ≈ 3.141592654* whose value fully coincides with π when rounded to 10 digits.

Notes

*1.*This program will also run in the *HP-41C* series by just changing all **GSB** instructions to **XEQ** and executing **ΣREG 00**.

*2.*This *127 step* program effectively demonstrates the benefits of the *HP-34C*'s *expandable* program memory. For instance, though the *HP-29C* has more RAM, it can't be allocated flexibly and it's limited to 98 steps, so this program won't fit in.

*3.*This program is published in the *Hewlett-Packard's Solution Book "HP-34C Matemática Avanzada" (Spanish)*. The program listing there includes keycodes for all steps and fully commented source code.

Copyrights

Copyright for this paper and its contents is retained by the author. Permission to use it for non-profit purposes is granted as long as the contents aren't modified in any way and the copyright is acknowledged.