FI

Richard J. Mathar

Generated by Doxygen 1.8.2

Thu Mar 28 2013 12:22:46

Contents

1 FI Documentation

2 Module Index

2.1 Modules

Here is a list of all modules:

3 Namespace Index

3.1 Packages

Here are the packages with brief descriptions (if available):

4 Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

5 Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

6 File Index

6.1 File List

Here is a list of all files with brief descriptions:

7 Module Documentation

7.1 online integer number calculator with Forth type syntax

Author

Richard J. Mathar

7.1.1 Scope

This is a calculator which mainly operates on signed integer numbers. Numbers and operators are typed into the "Input" field in Reverse Polish Notation (RPN) and submitted with carriage return. The TOS is the top-of-stack, which is the most recent number pushed on it; the NOS is the next number below the top-of-stack. There is an auxiliary return stack. Both stacks grow downwards in the GUI.

7.1.2 Main Layout

The GUI contains a `History` field on top, which essentially contains what has been typed in before and is useful to copy'n paste coding pieces from there back into the `Input`.

The next field is an `Input` line in which operators and numbers etc. (that is blank separated words) are typed in in Forth-type fashion.

7.1.3 Standard Forth Operators

`Operators` are one of

1. `+` add top 2 items of the stack
2. `1+` increase TOS by 1
3. `2+` increase TOS by 2
4. `-` subtract top 2 items of the stack
5. `1-` decrease TOS by 1
6. `2-` decrease TOS by 2
7. `*` multiply top 2 items of the stack
8. `2*` multiply TOS by 2
9. `/` integer division of top 2 items of the stack, NOS/TOS
10. `2/` divide TOS by 2 (arithmetic shift right)
11. `*/` multiply 2nd and 3rd item from top of the stack and divide through top of the stack
12. `MOD` remainder of top 2 items of the stack, NOS mod TOS.
13. `/MOD` remainder and divisor of top 2 items of the stack
14. `*/MOD` multiply 2nd and 3rd item of stack and leave remainder and divisor of dividing the product by top item of stack
15. `AND` bitwise AND of top 2 items of the stack
16. `OR` bitwise OR of top 2 items of the stack
17. `XOR` bitwise XOR of top 2 items of the stack
18. `NEGATE` flip sign of TOS
19. `ABS` build positive absolute value of TOS
20. `MIN` maximum of the top 2 items of the stack
21. `MAX` minimum of the top 2 items of the stack
22. `DUP` duplicate TOS
23. `?DUP` duplicate TOS if nonzero
24. `DROP` remove TOS
25. `SWAP` exchange top 2 items on the stack
26. `OVER` duplicate NOS
27. `TUCK` copy a duplicate of top stack item before first but last item
28. `NIP` remove NOS
29. `ROT` rotate the 3 topmost items of the stack
30. `ROLL` rotate as many items of the stack is indicated by the TOS (plus 1)
31. `PICK` copy an item from the stack at the position indicated by the top item
32. `2DUP` copy 2 items of the top of the stack
33. `2DROP` remove NOS and TOS

34. 2SWAP swap the two pairs of the 4 topmost items on the stack
35. 2OVER copy the pair of numbers below TOS and NOS on top of the stack
36. TRUE Push TRUE (-1) on the TOS
37. FALSE Push FALSE (0) on the TOS
38. >R push item from stack to return stack
39. R@ copy top of return stack to stack
40. R> move top of return stack to stack
41. ! Store NOS to the variable named by the TOS
42. +! Add the value of the NOS to the variable named by the TOS
43. @ Replace the variable named by the TOS by its value.
44. QUIT remove all items from return stack
45. 2>R move 2 items from stack to return stack
46. 2R> move 2 items from return stack to stack
47. < evaluate to true if 2nd item of stack less than top of stack
48. > evaluate to true if 2nd item of stack is greater than top of stack
49. = evaluate to true if 2nd item of stack equals top of stack
50. <> evaluate to true if 2nd item of stack differs from top of stack
51. 0= evaluate to true if top of stack equals zero
52. 0<> evaluate to true if top of stack differs from zero
53. within evaluate to true if n3<=n2<n1 where n1, n2 and n3 are the topmost items on the stack
54. DECIMAL switch to number base 10
55. HEX switch to number base 16

Operators may be written in uppercase or lowercase letters, and act in RPN style on numbers on the stack. There is limited support for fractions: if a word keyed in contains a single slash, it will be reduced to a fraction, on which the operators will operate according to the arithmetic standards. If the result of such an operation is integer, it will be treated as such by the followup operations. Characters between parenthesis (surrounded by blanks) or after a backslash in a line are comments and ignored (but copied to the history).

7.1.4 Variables and Constants

New variables are declared by typing variable followed by the name of the variable in the next blank-separated input word. The variable BASE is predefined for the number basis, the variable DIGITS for the precision of the secondary display of fractions, and POLYDEGREE for the cut-off power of the Taylor expansion of polynomial division. New constants are declared in the format

initvalue VARIABLE name

which means the initial value is put on the stack followed by the word VARIABLE and the variable name. Pushing the name of the variable later on on the TOS will replace the name by the value.

7.1.5 User defined Functions

Functions are defined by a blank-separated colon, the function name (case sensitive), the list of words to be executed, and finished with a blank-separated semicolon. They are executed by placing their name on the stack. Examples:

1. What is called the INVERT transform takes a sequence, flips all signs, adds a 1 in front computes the inverse of the sequence, and drops the initial 1 of the constant term. The implementation is done by multplication by -x (negates and shifts right in one go) adding 1, inserting the 1 of the numerator and swapping it with the TOS, dividing the 1 (now NOS) by the TOS, and dropping the initial term by subtracting 1 and division through x:

   ```
   : invrt 0,-1 * 1 + 1 swap / 1 - 0,1 / ; ( INVERT transform )
   ```

   This is then called like

   ```
   1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20 invrt
   ```

   and leaves essentially a bisection of Fibonaccis on the stack. The inverse operation is of course implementing these calculations in reverse order:

   ```
   : invrti 0,1 * 1 + 1 swap / 1 - 0,-1 / ; ( INVERTi transform )
   ```

   This example of the function is the closest equivalent of an existing Maple implementation, as documented in Some canconical sequences of integers. This implementation assumes that an known generating function a(x) = 1+a_1∗x+a_2∗x^2+a_3∗x^3+.... is turned into another function b(x) = 1/(1-a(x)) = 1+b_1∗x+b_2∗x^2+.. where the constant terms, that is the leading 1's, are not specified on input and removed on output. Other versions could follow eq. 3.6.16 of Abramowitz-Stegun without a sign flip

   ```
   : invrt2 0,1 * 1 + 1 swap / 1 - 0,1 / ; ( 2nd INVERT transform )
   ```

   or could expect that the input and output maintain the constant term explicitly

   ```
   : invrtB 1 swap / ; ( basic INVERT transform )
   ```

2. To calculate the Tribonacci sequence, define a function

   ```
   : trib 2dup + 3 pick + ;
   ```

   which creates a copy of the top 2 items on the stack, adds them, then creates a copy by reaching to what had been the 3rd item on the stack, adds this to the sum of the others, and leaves this on the stack. The sequence is then created by a call like

   ```
   0 0 1 trib trib trib trib trib trib trib trib trib trib
   ```

   A simpler way is of course to use the generating function x^2/(1-x-x^2-x^3) with the technology of polynomial division as described below:

   ```
   0,0,1
   1,-1,-1,-1
   /
   ```

3. To calculate the pentagonal numbers 5n(n+1)/2+1, define a function

   ```
   : cpo dup 5 * swap 1+ * 2 / 1+ ;
   ```

   and then use

   ```
   0 cpo 1 cpo 2 cpo 3 cpo 4 cpo 5 cpo 6 cpo 7 cpo 8 cpo 9 cpo
   ```

   to get a list on the stack.

4. To calculate the sum of the divisors of a number smaller than the number n itself A001065, define a function

   ```
   : aliqu dup sigma swap - ;
   ```

   which duplicates the TOS, calculates (n) with the precompiled function defined below, exchanges the two elements, so (n) becomes NOS and n becomes TOS, then computes the difference. An aliquot sequence of a number, like 276, is then calculated in the style

   ```
   276 aliqu aliqu aliqu aliqu aliqu
   ```

7.1.6 Precompiled Functions

The following number theoretic functions are available without the need to implement them by the user program. They can be typed in without regard to the letter cases:

- FACTORIAL replace TOS by its factorial. Examples:

  10 factorial **replaces the 10 by** 3628800**.**

0 factorial **replaces the 0 by** 1**.**

- DOUBLEFACTORIAL replace TOS by its double factorial
- PRIME replace TOS by the n'th prime number Examples:
  1. 1 prime replaces the 1 by 2.
  2. 2 prime replaces the 2 by 3.
  3. 3 prime replaces the 3 by 5.
  4. 4 prime replaces the 4 by 7.
  5. 5 prime replaces the 5 by 11.
  6. 3 7 prime dup >R 1- pow 1- R> / computes the Fermat quotient of the 7th prime with base 3, A146211. It pushes 3 and 7 on the stack, replaces 7 by the 7th prime (that is 17), uses dup >R to create a copy on the return stack (for later use), moves from 17 to 16 on the stack by 1-, replaces the 3 and 16 by 3^16=43046721 with pow, subtracts one which leaves 43046720 on the stack, moves the value on the return stack back (that was 17) on top of the stack with R>, and divides finally 43046720 thru 17 with the slash, to leave 2532160 on the stack.
- PRIMORIAL replace TOS by the product of the first prime numbers
- NEXTPRIME smallest prime larger than the number on TOS
- PREVPRIME largest prime smaller than the number on TOS
- PI The count of primes smaller or equal to the number on TOS
- ISPRIME check for primeness of the TOS. The TOS is replaced by -1 if the value is a prime, else by 0. Examples:
  1. 10 isprime replaces the 10 by 0.
  2. 333 isprime replaces the 333 by 0.
  3. 19 isprime replaces the 19 by -1.
- BINOMIAL binomial of top 2 times on the stack, binomial(NOS,TOS). This is one of the few functions which also handles signed fractions in the numerator, so one can easily use this to derive the Taylor expansions of expressions like $(1+x)^{\beta}$ with fractional β. Examples:
  1. 5 3 binomial replaces the 5 and 3 by 10 which is 5!/(2!3!).
  2. 6 3 binomial replaces the 6 and 3 by 20 which is 6!/(3!3!).
  3. -1/2 5 binomial replaces the -1/2 and 5 by -63/256 which is (-1/2)(-3/2)(-5/2)(-7/2)(-9/2)/5!.
- POCHHAMMER Pochhammer's symbol: Gamma(NOS+TOS)/Gamma(TOS). This is one of the few functions which accepts a signed fraction as the NOS.
- GCD Greatest common divisor of NOS and TOS
- LCM Least common multiple of NOS and TOS
- POW power NOS^TOS. This is one of the few functions which accept a signed fraction at the NOS. If the result has a representation in the rationals, the TOS can also be a signed fraction. Examples:

1. `5 3 pow` replaces the 5 and 3 by `125` which is 5^3.
2. `5 -3 pow` replaces the 5 and -3 by `1/125` which is 1/5^3.
3. `2/3 3 pow` replaces the 2/3 and 3 by `8/27` which is (2/3)^3.
4. `-2/3 3 pow` replaces the -2/3 and 3 by `-8/27` which is (-2/3)^3.
5. `1/1024 1/5 pow` replaces the 1/1024=2^(-10) and 1/5 by `1/4` which is 2^(-2).

- `MOEBIUS` `Mobius function` of TOS, 0 or +/-1
- `IFACTORS` integer factorization of TOS = p1^e1*p2^e2*... returned as a comma/semicolon separated list p1,e1;p2,e2;... of primes and their powers. Example

  1. `999 ifactors`
     first puts 999 on the stack, and the function replaces this by `3,3;37,1;`
     which means 3 cubed multiplied by 37.
  2. `234890890 ifactors`
     puts 234890890 on the stack, and the function replaces this by `2,1;5,1;13,1;1171,1;1543,1;`
     which means the product of 2 multiplied by 5 and by 13 and by 1171 and by 1543.

- `SIGMA` `Sum of divisors` of TOS
- `SIGMAK` Sum of the TOS-th powers of the divisors of NOS Examples:

  1. `3 2 sigmak`
     leaves 10 = 1^2+3^2 on the stack.
  2. `10 4 sigmak`
     leaves 10642 = 1^4+2^4+5^4+10^4 on the stack.

- `TAU` `Number of divisors` of TOS
- `FIBONACCI` `Fibonacci number` with index equal to TOS
- `LUCAS` `Lucas number` with index equal to TOS
- `BELL` `Bell number` with index equal to TOS
- `CATALAN` `Catalan number` of TOS
- `PHI` `Euler's totient function` of TOS
- `NUMBPART` `Number of unrestricted partitions` of TOS
- `CORE` `square-free part` of TOS
- `BIGOMEGA` `Number of prime factors` of TOS (counted with multiplicity)
- `OMEGA` `Number of distinct prime factors` of TOS
- `MOTZKIN` `Motzkin number` indexed by TOS
- `SCHRODER` `Large Schroder number` indexed by TOS
- `HAMMING` `Number of bits set` in TOS
- `THUEMORSE` `Thue-Morse value` of TOS
- `STIRLING1` `Stirling numbers of the first kind` of NOS and TOS. Example:

  1. `12 9 stirling1`
     leaves -32670 on the stack.

- `STIRLING2` `Stirling numbers of the second kind` of NOS and TOS. Examples:

1. `8 5 stirling2`
   leaves 1050 on the stack.
2. `20 4 stirling2`
   leaves 45232115901 on the stack.

- `BERNOULLI` Bernoulli numbers at the index given by the TOS
- `EULER` `Euler numbers` at the index given by the TOS
- `HARMONIC` Harmonic numbers at the index given by the TOS `Numerator`/ `Denominator`.
- `JACOBSTHAL` `Jacobsthal number` with index equal to TOS
- `CAUCHY2` Cauchy numbers of the 2nd kind as represented by the division of `A002657` and `A002790`. Example

  1. `4 cauchy2`
     leaves `251/30` on the stack.
  2. `9 cauchy2`
     leaves `2082753/20` on the stack.

The pre-defined functions can be hidden by a user-defined function as described further up.

7.1.7 Operations on polynomials (or sequences)

Polynomials are represented by comma-separated (no white-space) sequences of numbers (integers or fractions) which represent the coefficients in front of the constant, the linear, the quadratic term etc. Besides the commands that move stack items around (`DUP`, `DROP` etc), the currently implemented actions on one or two polynomials are

- $*$ multiplication. Note that this is the convolution of the sequences of the coefficients. Example

  1. To multiply 1+2$*$x^2+7$*$x^5 by 3$*$x^2+x^6/4 push both polynomials on the stack and use the star (from the common set of operators) to multiply these two items:
     `1,0,2,0,0,7 0,0,3,0,0,0,1/4 *`
     which should leave `0,0,3,0,6,0,1/4,21,1/2,0,0,7/4,...` on the stack and is to be interpreted as 3$*$x^2+6$*$x^4+x^6/4+21x^7...
  2. To generate `A090826`, that is `0,1,2,5,12,31,85,...` note that this is the convolution of `A000045` = 0,1,1,2,3,5,8,... by `A000108` = 1,1,2,5,14,42,132,429,... and can be generated by multiplication:
     `1,1,2,5,14,42,132,429,1430,4862,16796,58786,208012,742900,2674440`
     `0,1 1,-1,-1 / ( this generates 0,1,1,2, etc from x/(1-x-x^2) ) *`

- $+$ addition.
- $-$ subtraction.
- `pow` raising to a power. Example

  1. `1,1 4 pow`
     calculates (1+x)^4 and represents this as `1,4,6,4,1`
     on the stack. This is also a quick way of generating a sequence of binomials, of course.
  2. Negative integer powers (in truncated power series representation) are calculated by division. 1/(1+2x)^5 is generated by
     `1 1,2 5 pow /`
  3. As long as the absolute term of the polynomial raised to the power is in the rationals, the power may also be a rational:
     `4,3,5,6 1/2 pow /`
     pulls the square root out of 4+3$*$x+5$*$x^2+6$*$x^4, which is evaluated since 4^(1/2) can be reduced to a rational value (namely 2).

- / division (as in the generating function interpretation). This covers the series inversion [**?** , 3.6.16], if one uses the polynomial 1 in the numerator, and the deconvolutions. Example

  1. To calculate the sequence with the generating function (1+x)/(1-x+x^3) as in A078013 from the 7th term on, push the coefficients of 1+x, then the coefficients of 1-x+x^3 on the stack, and divide:

     ```
     1,1
     1,-1,0,1
     /
     ```

     which leaves 1,2,2,1,-1,-3 etc on the stack, which in turn is the Taylor expansion 1+2x+2x^2+x^3-x^4+...
  2. To calculate the sequence of squared Fibonacci numbers with the generating function x(1-x)/((1+x)(1-3x+x^2)) as in A007598, push the coefficients of x, 1-x, on the stack, multiply, then 1+x and 1-3x+x^2 on the stack, multiply, and divide

     ```
     0,1 1,-1 * ( now have x-x^2 )
     1,1 1,-3,1 * ( now have x-x^2 as NOS and (1+x)(1-3x+x^2) as TOS )
     / ( got the taylor expansion on TOS )
     ```

     which leaves 0,1,1,4,9,25,64,169 etc on the stack.

- POLYDERIV computes the first derivative,
- POLYBINOMIALT binomial transform of the coefficients
- POLYBINOMIALTINV inverse binomial transform of the coefficients
- POLYSTIR1T Stirling-1 transform of the coefficients. Example: to obtain a polynomial with the coefficients of A086672 use

  ```
  1,1,2,5,14,42,132,429,1430,4862,16796,58786,....  polystir1t.
  ```

- POLYSTIR2T Stirling-2 transform of the coefficients. Example: To obtain the numbers of A069321, use the input line

  ```
  0,1,4,18,96,600,4320,35280,322560,3265930,...  polystir2t.
  ```

- POLYMOBIUST Mobius transform of the coefficients. Example:

  1. The input

     ```
     0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16 polymobiust
     ```

     generates on the stack essentially A000010, 0,1,1,2,2,4,2,6,4,6,4,10,4,12, etc. As usual, to get a correct output in all orders, it is easier to use generating function technology to represent the natural numbers by x/(1-x)^2 like

     ```
     0,1 1,-1 2 pow / polymobiust
     ```

     The absolute term of the input polynomial is ignored and replaced in all cases by a zero absolute term on output.

- POLYMOBIUSTINV Inverse Mobius transform of the coefficients. Example

  1. To generate the inverse Mobius transform of the triangular numbers 1,3,6,10,..., generate them first on the stack and call polymobiustinv,

     ```
     0,1
     1,-1
     dup dup *
     /
     polymobiustinv
     ```

     which leaves essentially A007437 on the stack.

- POLYMOTZKINT Motzkin transform of the coefficients. The polynomial that results is computed from the original polynomial assuming that each occurrence of the independent variable x is replaced by x times the generating function of the Motzkin series. Examples:

1. To obtain the numbers of A001006, use the input line
   `0,1 polymotzkint.`
2. To generate A002026, use the input line
   `0,1,1,1 polymotzkint.`
3. To generate A036908, use the input line
   `1,2,3,4,5,6,7,8,9,10 polymotzkint` or, using the generating function 1/(1-x)^2 for the integer sequnce
   ```
   1
   1,-1
   dup
   *
   /
   polymotzkint
   ```
4. To generate effectively A005774, use
   `1,0,0,1,0,0,1,0,0,1,0,0,1 polymotzkint`
   More experienced users would use the generating function 1/(1-x^3) for the integer sequence to ensure that the polynomial is correct to all `POLYDEGREE` orders
   `1 1,0,0,-1 / polymotzkint`
5. To generate A000244, use the input line
   `0,1,2,3,4,5,6,7,8,9,10 polymotzkint`
   or again the generating function for the integers,
   `0,1 1,-1 dup * / polymotzkint`
6. To generate essentially A005717, use the input line
   `1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0 polymotzkint`
   or generate the periodic input coefficients from the generating function 1/(1-x^2) more efficiently to the order given by `POLYDEGREE`,
   `1 1,0,-1 / polymotzkint`
7. To generate essentially A002426, use the input line
   `0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0 polymotzkint`
   or generate the periodic input coefficients from the generating function x/(1-x^2) more efficiently to the full order given by `POLYDEGREE`,
   `0,1 1,0,-1 / polymotzkint`

- `POLYMOTZKINTINV` Inverse Motzkin transform of the coefficients. This is the inverse functionality of `POLYMOTZKINT`.
- `POLYROOTS` which dumps the polynomial roots in Fortran complex-notation (that is real and imaginary part enclosed by parentheses, separated by a comma) on the stack. This is a kind of dead end in the sense that complex numbers are not used anywhere else. Also, it has not been tested whether the implementation can handle roots of higher multiplicity at all. The variable `DIGITS` can be set with the same `!` syntax as the other variables to control the number of digits in the internal handling of the floating point representations. Examples

  1. To calculate the tribonacci constant A058265, define the polynomial 1+x+x^2-x^3 and calculate the roots with
     `1,1,1,-1 POLYROOTS`
     which should replace the polynomial by three numbers, one of them close to 1.839, on the stack.
  2. To calculate `sqrt(3)/2`, observe that this equals a root of 4x^2-3, so use the input
     `-3,0,4 polyroots`
     to get a numbers close to +-0.8660 back on the stack.

- `POLYCATALANT` Catalan transform of the coefficients. Example:

  1. To calculate A100320 use
     `1,2 1,-2 / polycatalant`

2. To calculate A126983, the Catalan transform of A033999, use
   `1 1,1 / polycatalant`
3. To calculate the Catalan transform of A011655, use
   `0,1,1 1,0,0,-1 / polycatalant`
4. To calculate A127632, generate a polynomial which contains the Catalan numbers, drop the leading 0 by dividing through x, and use the Catalan transform,
   `0,1 polycatalant 0,1 / polycatalant`

- `POLYCATALANTINV` Inverse Catalan transform of the coefficients. Example:

1. To calculate the inverse Catalan transform of A143464 use `0,1,3,11,42,164,649,2591,10408,41998 polycatalanTInv`

- `POLYCONVEXT` which extends a polynomial by higher order terms, a(n)=sum(i=0..n-1) a(i)a(n-i-1). Examples

1. The call
   `2 polyconvext`
   generates (see A025225 and A115125) on the stack,
   `2,4,16,80,448,2688,16896,109824,732160,4978688,...`
2. The call
   `2,1 polyconvext`
   generates (see A025228) on the stack
   `2,1,4,17,76,354,1704,8421,42508,218318,1137400,...`
3. The call
   `0,1,1 polyconvext`
   generates (see A007477) on the stack
   `0,1,1,1,2,3,6,11,22,44,90,187,392,832,1778,3831,8304,...`

- `POLYCONVEX1` which extends a polynomial by higher order terms, a(n)=a(n-1)+sum(i=0..n-2) a(i)a(n-i-1). Examples

1. The input
   `0,1,2 polyconvex1`
   generates A143013 on the stack,
   `0,1,2,3,7,17,43,114,310,861,2433,6970,20198,...`
2. The input
   `2,2 polyconvex1`
   are altenatively
   `2 polyconvex1`
   generates essentially A006318 on the stack,
   `2,2,6,22,90,394,1806,8558,41586,206098,1037718,...`
3. The input
   `0,1,1 polyconvex1`
   generates essentially the Motzking numbers A001006 on the stack,
   `0,1,1,2,4,9,21,51,127,323,835,2188,5798,15511,41835,113634,...`

As an aid of interpretation, these polynomials are listed also in a different notation, more similar to a generating function expansion, as a comment after the comma-separated list on the stack.

8 Namespace Documentation

8.1 Package org

Packages

- package nevec

8.2 Package org.nevec

Packages

- package rjm

8.3 Package org.nevec.rjm

Classes

- class Bernoulli

  Bernoulli numbers.
- class BigComplex

  Complex numbers with BigDecimal real and imaginary components.
- class BigDecimalMath

  BigDecimal special functions.
- class BigDecimalPoly

  Polynomial with BigDecimal coefficients.
- class BigIntegerMath

  BigInteger special functions and Number theory.
- class BigIntegerPoly

  Polynomial with integer coefficients.
- class BigSurd

  Square roots on the real line.
- class BigSurdVec

  A BigSurdVec represents an algebraic sum or differences of values which each term an instance of BigSurd.
- class **Blas**

  Basic Linear Algebra subroutines (BLAS)
- class Euler

  Euler numbers.
- class EulerPhi

  Euler totient function.
- class Factorial

  Factorials.
- class **FIFunc**

  A function, which is a string (=function name) and a sequence of words to be executed.
- class **FILoop**

  A inner body of a loop between DO and LOOP or +LOOP This is mainly needed to save the text back to the DO even if the loop is defined outside a function, which means even if the pieces of the loop body are forwarded individually from the input line.
- class FI

  Applet of an integer calculator with Forth-alike syntax.
- class Harmonic

  Harmonic numbers.
- class Ifactor

  Factored integers.
- class IinHRecGui
- class **A000005**

  Number of divisors of the number.
- class **A000027**

  The integers.
- class **A000032**

Lucas numbers.

- class **A000045**

  Fibonacci numbers.
- class **A000108**

  Catalan numbers.
- class **A000110**

  Bell numbers.
- class **A000120**

  Number of 1's in binary expansion.
- class **A000290**

  Squares.
- class **A000367**

  Numerators of Bernoulli numbers of even index.
- class **A000720**

  Pi, the number of primes less than or equal to n.
- class **A001006**

  Motzkin numbers.
- class **A001045**

  Jacobsthal sequence.
- class **A001065**

  Sum of proper divisors of n.
- class **A001147**

  Double factorials.
- class **A002110**

  Primorials.
- class **Cauchy2**

  Cauchy numbers of the second type.
- class **A005114**

  Untouchable numbers.
- class **A006318**

  Large Schroder numbers.
- class **A006954**

  Denominators of Bernoulli numbers.
- class **A007947**

  Largest square-free number dividing n.
- class **A008275**

  Stirling numbers of the 1st kind.
- class **A008683**

  Moebius function.
- class **A010060**

  Thue-Morse function .
- class **A048993**

  Stirling numbers of the 2nd kind.
- class **A047994**

  Unitary Phi.
- class **A078923**

  Complement set to the untouchable numbers.
- class **A111278**

  Untouchable squares.
- class **A119829**

  Diagonal sum of number triangle A119828 .

- class PartitionsP

  Number of partitions.
- class Prime

  Prime numbers.
- class Rational

  Fractions (rational numbers).
- class RationalPoly

  Ratio of two univariate polynomials with rational coefficients.
- class **RatPoly**

  A uni-variater polynomial with rational coefficients.
- class Wigner3j

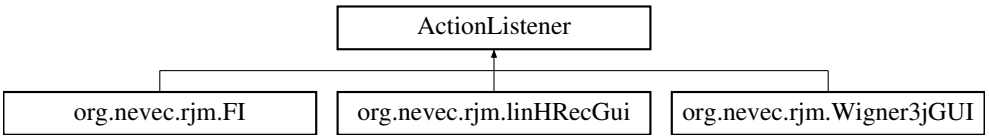
  Exact representations of Wigner 3jm and 3nj values of half-integer arguments.
- class Wigner3jGUI

  An interactive interface to the Wigner3j class.

9 Class Documentation

9.1 ActionListener Class Reference

Inheritance diagram for ActionListener:



9.2 org.nevec.rjm.Bernoulli Class Reference

Bernoulli numbers.

Public Member Functions

- Bernoulli ()
- Rational at (int n)

  The Bernoulli number at the index provided.

Static Public Member Functions

- static void main (String args[])

Protected Member Functions

- void set (final int n, final Rational value)

  Set a coefficient in the internal table.

Private Member Functions

- Rational doubleSum (int n)

9.2.1 Detailed Description

Bernoulli numbers.

See Also

[**?**] [**?**]

Since

2006-06-25

Author

Richard J. Mathar

9.2.2 Constructor & Destructor Documentation

9.2.2.1 org.nevec.rjm.Bernoulli.Bernoulli ( )

9.2.3 Member Function Documentation

9.2.3.1 Rational org.nevec.rjm.Bernoulli.at (int *n*)

The Bernoulli number at the index provided.

Parameters

n	the index, non-negative.

Returns

the B_0=1 for n=0, B_1=-1/2 for n=1, B_2=1/6 for n=2 etc

9.2.3.2 Rational org.nevec.rjm.Bernoulli.doubleSum (int *n*) [private]

9.2.3.3 static void org.nevec.rjm.Bernoulli.main (String *args[]*) [static]

9.2.3.4 void org.nevec.rjm.Bernoulli.set (final int *n,* final Rational *value*) [protected]

Set a coefficient in the internal table.

Parameters

n	the zero-based index of the coefficient. n=0 for the constant term.
value	the new value of the coefficient.

9.3 org.nevec.rjm.BigComplex Class Reference

Complex numbers with BigDecimal real and imaginary components.

Public Member Functions

- BigComplex ()

  Default ctor equivalent to zero.
- BigComplex (BigDecimal x, BigDecimal y)

ctor with real and imaginary parts

- BigComplex (BigDecimal x)

  ctor with real part.
- BigComplex (double x, double y)

  ctor with real and imaginary parts
- String toString ()

  Human-readable Fortran-type display.
- String toString (MathContext mc)

  Human-readable Fortran-type display.

9.3.1 Detailed Description

Complex numbers with BigDecimal real and imaginary components.

See Also

BigComplex

Since

2008-10-26

Author

Richard J. Mathar

9.3.2 Constructor & Destructor Documentation

9.3.2.1 org.nevec.rjm.BigComplex.BigComplex ( )

Default ctor equivalent to zero.

9.3.2.2 org.nevec.rjm.BigComplex.BigComplex (BigDecimal *x*, BigDecimal *y*)

ctor with real and imaginary parts

Parameters

x	real part
y	imaginary part

9.3.2.3 org.nevec.rjm.BigComplex.BigComplex (BigDecimal *x*)

ctor with real part.

Parameters

x	real part. The imaginary part is set to zero.

9.3.2.4 org.nevec.rjm.BigComplex.BigComplex (double *x*, double *y*)

ctor with real and imaginary parts

Parameters

x	real part
y	imaginary part

9.3.3 Member Function Documentation

9.3.3.1 String org.nevec.rjm.BigComplex.toString ( )

Human-readable Fortran-type display.

Returns

real and imaginary part in parenthesis, divided by a comma.

9.3.3.2 String org.nevec.rjm.BigComplex.toString (MathContext *mc*)

Human-readable Fortran-type display.

Returns

real and imaginary part in parenthesis, divided by a comma.

9.4 org.nevec.rjm.BigDecimalMath Class Reference

BigDecimal special functions.

Static Public Member Functions

- static void main (String args[])
  Test programs.
- static BigDecimal pi (final MathContext mc)
  Euler's constant.
- static BigDecimal gamma (MathContext mc)
  Euler-Mascheroni constant.
- static BigDecimal sqrt (final BigDecimal x, final MathContext mc)
  The square root.
- static BigDecimal sqrt (final BigDecimal x)
  The square root.
- static BigDecimal cbrt (final BigDecimal x)
  The cube root.
- static BigDecimal root (final int n, final BigDecimal x)
  The integer root.
- static BigDecimal hypot (final BigDecimal x, final BigDecimal y)
  The hypotenuse.
- static BigDecimal hypot (final int n, final BigDecimal x)
  The hypotenuse.
- static BigDecimal exp (BigDecimal x)
  The exponential function.
- static BigDecimal exp (final MathContext mc)
  The base of the natural logarithm.
- static BigDecimal log (BigDecimal x)
  The natural logarithm.
- static BigDecimal log (int n, final MathContext mc)

The natural logarithm.

- static BigDecimal log (final Rational r, final MathContext mc)

  The natural logarithm.
- static BigDecimal pow (final BigDecimal x, final BigDecimal y)

  Power function.
- static BigDecimal powRound (final BigDecimal x, final int n)

  Raise to an integer power and round.
- static BigDecimal powRound (final BigDecimal x, final BigInteger n)

  Raise to an integer power and round.
- static BigDecimal powRound (final BigDecimal x, final Rational q)

  Raise to a fractional power and round.
- static BigDecimal sin (final BigDecimal x)

  Trigonometric sine.
- static BigDecimal cos (final BigDecimal x)

  Trigonometric cosine.
- static BigDecimal tan (final BigDecimal x)

  The trigonometric tangent.
- static BigDecimal cot (final BigDecimal x)

  The trigonometric co-tangent.
- static BigDecimal asin (final BigDecimal x)

  The inverse trigonometric sine.
- static BigDecimal acos (final BigDecimal x)

  The inverse trigonometric cosine.
- static BigDecimal atan (final BigDecimal x)

  The inverse trigonometric tangent.
- static BigDecimal cosh (final BigDecimal x)

  The hyperbolic cosine.
- static BigDecimal sinh (final BigDecimal x)

  The hyperbolic sine.
- static BigDecimal tanh (final BigDecimal x)

  The hyperbolic tangent.
- static BigDecimal asinh (final BigDecimal x)

  The inverse hyperbolic sine.
- static BigDecimal acosh (final BigDecimal x)

  The inverse hyperbolic cosine.
- static BigDecimal Gamma (final BigDecimal x)

  The Gamma function.
- static BigDecimal Gamma (final Rational q, final MathContext mc)

  The Gamma function.
- static BigDecimal pochhammer (final BigDecimal x, final int n)

  Pochhammer's function.
- static BigDecimal mod2pi (BigDecimal x)

  Reduce value to the interval [0,2∗Pi].
- static BigDecimal modpi (BigDecimal x)

  Reduce value to the interval [-Pi/2,Pi/2].
- static BigDecimal zeta (final int n, final MathContext mc)

  Riemann zeta function.
- static double zeta1 (final int n)

  Riemann zeta function.
- static double cot (final double x)

  trigonometric cot.

- static double psi (final double x)

  Digamma function.
- static double psi (final int n, final double x)

  Polygamma functions.
- static BigDecimal EllipticK (final BigDecimal m)

  Complete Elliptic Integral of the First Kind.
- static BigDecimal EllipticE (final BigDecimal m)

  Complete Elliptic Integral of the Second Kind.
- static BigDecimal EllipticE (final BigDecimal m, final BigDecimal phi)

  Incomplete Elliptic Integral of the Second Kind.
- static BigDecimal Hypergeometric21 (final Rational a, final Rational b, final Rational c, final BigDecimal x)

  Gaussian Hypergeometric function with three rational parameters.
- static Rational toRational (BigDecimal x)

  Convert the finite representation of a floating point value to its fraction.
- static Vector< BigInteger > cfrac (final BigDecimal x)

  Continued fraction.
- static BigDecimal add (final BigDecimal x, final BigInteger y)

  Add a BigDecimal and a BigInteger.
- static BigDecimal addRound (final BigDecimal x, final BigDecimal y)

  Add and round according to the larger of the two ulp's.
- static BigComplex addRound (final BigComplex x, final BigDecimal y)

  Add and round according to the larger of the two ulp's.
- static BigComplex addRound (final BigComplex x, final BigComplex y)

  Add and round according to the larger of the two ulp's.
- static BigDecimal subtractRound (final BigDecimal x, final BigDecimal y)

  Subtract and round according to the larger of the two ulp's.
- static BigComplex subtractRound (final BigComplex x, final BigComplex y)

  Subtract and round according to the larger of the two ulp's.
- static BigDecimal multiplyRound (final BigDecimal x, final BigDecimal y)

  Multiply and round.
- static BigComplex multiplyRound (final BigComplex x, final BigDecimal y)

  Multiply and round.
- static BigComplex multiplyRound (final BigComplex x, final BigComplex y)

  Multiply and round.
- static BigDecimal multiplyRound (final BigDecimal x, final Rational f)

  Multiply and round.
- static BigDecimal multiplyRound (final BigDecimal x, final int n)

  Multiply and round.
- static BigDecimal multiplyRound (final BigDecimal x, final BigInteger n)

  Multiply and round.
- static BigDecimal divideRound (final BigDecimal x, final BigDecimal y)

  Divide and round.
- static BigComplex invertRound (final BigComplex z)

  Build the inverse and maintain the approximate accuracy.
- static BigComplex divideRound (final BigComplex x, final BigComplex y)

  Divide and round.
- static BigDecimal divideRound (final BigDecimal x, final int n)

  Divide and round.
- static BigDecimal divideRound (final BigDecimal x, final BigInteger n)

  Divide and round.
- static BigDecimal divideRound (final BigInteger n, final BigDecimal x)

Divide and round.

- static BigComplex divideRound (final BigInteger n, final BigComplex x)

  Divide and round.
- static BigDecimal divideRound (final int n, final BigDecimal x)

  Divide and round.
- static BigDecimal scalePrec (final BigDecimal x, int d)

  Append decimal zeros to the value.
- static BigComplex scalePrec (final BigComplex x, int d)

  Append decimal zeros to the value.
- static BigDecimal scalePrec (final BigDecimal x, final MathContext mc)

  Boost the precision by appending decimal zeros to the value.
- static int err2prec (BigDecimal x, BigDecimal xerr)

  Convert an absolute error to a precision.
- static int err2prec (double x, double xerr)

  Convert an absolute error to a precision.
- static int err2prec (double xerr)

  Convert a relative error to a precision.
- static double prec2err (final double x, final int prec)

  Convert a precision (relative error) to an absolute error.

Static Protected Member Functions

- static BigDecimal broadhurstBBP (final int n, final int p, final int a[], MathContext mc)

  Broadhurst ladder sequence.

Static Private Member Functions

- static BigDecimal F21series (final Rational a, final Rational b, final Rational c, final BigDecimal x)

  2F1(a,b;c;z) Hypergeometric function.

Static Private Attributes

- static int TAYLOR_NTERM = 8

  A suggestion for the maximum numter of terms in the Taylor expansion of the exponential.

9.4.1 Detailed Description

BigDecimal special functions.

`A Java Math.BigDecimal Implementation of Core Mathematical Functions`

Since

2009-05-22

Author

Richard J. Mathar

See Also

`apfloat`
`dfp`
`JScience`

9.4.2 Member Function Documentation

9.4.2.1 static BigDecimal org.nevec.rjm.BigDecimalMath.acos (final BigDecimal *x*) [static]

The inverse trigonometric cosine.

Parameters

x	the argument.

Returns

the arccos(x) in radians.

Since

2009-09-29

9.4.2.2 static BigDecimal org.nevec.rjm.BigDecimalMath.acosh (final BigDecimal *x*) [static]

The inverse hyperbolic cosine.

Parameters

x	The argument.

Returns

The arccosh(x) .

Author

Richard J. Mathar

Since

2009-08-20

9.4.2.3 static BigDecimal org.nevec.rjm.BigDecimalMath.add (final BigDecimal *x*, final BigInteger *y*) [static]

Add a BigDecimal and a BigInteger.

Parameters

x	The left summand
y	The right summand

Returns

The sum x+y.

Since

2012-03-02

9.4.2.4 static BigDecimal org.nevec.rjm.BigDecimalMath.addRound (final BigDecimal *x*, final BigDecimal *y*) [static]

Add and round according to the larger of the two ulp's.

Parameters

x	The left summand
y	The right summand

Returns

The sum x+y.

Since

2009-07-30

9.4.2.5 static BigComplex org.nevec.rjm.BigDecimalMath.addRound (final BigComplex *x*, final BigDecimal *y*) [static]

Add and round according to the larger of the two ulp's.

Parameters

x	The left summand
y	The right summand

Returns

The sum x+y.

Since

2010-07-19

9.4.2.6 static BigComplex org.nevec.rjm.BigDecimalMath.addRound (final BigComplex *x*, final BigComplex *y*) [static]

Add and round according to the larger of the two ulp's.

Parameters

x	The left summand
y	The right summand

Returns

The sum x+y.

Since

2010-07-19

9.4.2.7 static BigDecimal org.nevec.rjm.BigDecimalMath.asin (final BigDecimal *x*) [static]

The inverse trigonometric sine.

Parameters

x	the argument.

Returns

the arcsin(x) in radians.

9.4.2.8 static BigDecimal org.nevec.rjm.BigDecimalMath.asinh (final BigDecimal *x*) [static]

The inverse hyperbolic sine.

Parameters

x	The argument.

Returns

The arcsinh(x) .

Author

Richard J. Mathar

Since

2009-08-20

9.4.2.9 static BigDecimal org.nevec.rjm.BigDecimalMath.atan (final BigDecimal *x*) [static]

The inverse trigonometric tangent.

Parameters

x	the argument.

Returns

the principal value of arctan(x) in radians in the range -pi/2 to +pi/2.

Since

2009-08-03

9.4.2.10 static BigDecimal org.nevec.rjm.BigDecimalMath.broadhurstBBP (final int *n*, final int *p*, final int *a[]*, MathContext *mc*) [static],[protected]

Broadhurst ladder sequence.

Parameters

a	The vector of 8 integer arguments
mc	Specification of the accuracy of the result

Returns

S_(n,p)(a)

Since

2009-08-09

See Also

arXiv:math/9803067

9.4.2.11 static BigDecimal org.nevec.rjm.BigDecimalMath.cbrt (final BigDecimal *x*) [static]

The cube root.

Parameters

x	The argument.

Returns

The cubic root of the BigDecimal rounded to the precision implied by x. The sign of the result is the sign of the argument.

Since

2009-08-16

9.4.2.12 static Vector<BigInteger> org.nevec.rjm.BigDecimalMath.cfrac (final BigDecimal *x*) [static]

Continued fraction.

Parameters

x	The number the absolute value of which will be decomposed.

Returns

A list of the form [a0,a1,a2,a3,...] where The decomposition is |x| = a0+1/(a1+1/(a2+1/(a3+...))).

Since

2012-03-09

9.4.2.13 static BigDecimal org.nevec.rjm.BigDecimalMath.cos (final BigDecimal *x*) [static]

Trigonometric cosine.

Parameters

x	The argument in radians.

Returns

cos(x) in the range -1 to 1.

Since

2009-06-01

9.4.2.14 static BigDecimal org.nevec.rjm.BigDecimalMath.cosh (final BigDecimal *x*) [static]

The hyperbolic cosine.

Parameters

x	The argument.

Returns

The cosh(x) = (exp(x)+exp(-x))/2 .

Author

Richard J. Mathar

Since

2009-08-19

9.4.2.15 static BigDecimal org.nevec.rjm.BigDecimalMath.cot (final BigDecimal *x*) [static]

The trigonometric co-tangent.

Parameters

x	the argument in radians.

Returns

the cot(x)

Since

2009-07-31

9.4.2.16 static double org.nevec.rjm.BigDecimalMath.cot (final double *x*) [static]

trigonometric cot.

Parameters

x	The argument.

Returns

cot(x) = 1/tan(x).

9.4.2.17 static BigDecimal org.nevec.rjm.BigDecimalMath.divideRound (final BigDecimal *x*, final BigDecimal *y*) [static]

Divide and round.

Parameters

x	The numerator
y	The denominator

Returns

the divided x/y

Since

2009-07-30

9.4.2.18 static BigComplex org.nevec.rjm.BigDecimalMath.divideRound (final BigComplex *x*, final BigComplex *y*) [static]

Divide and round.

Parameters

x	The numerator
y	The denominator

Returns

the divided x/y

Since

2010-07-19

9.4.2.19 static BigDecimal org.nevec.rjm.BigDecimalMath.divideRound (final BigDecimal *x*, final int *n*) [static]

Divide and round.

Parameters

x	The numerator
n	The denominator

Returns

the divided x/n

Since

2009-07-30

9.4.2.20 static BigDecimal org.nevec.rjm.BigDecimalMath.divideRound (final BigDecimal *x*, final BigInteger *n*) [static]

Divide and round.

Parameters

x	The numerator
n	The denominator

Returns

the divided x/n

Since

2009-07-30

9.4.2.21 static BigDecimal org.nevec.rjm.BigDecimalMath.divideRound (final BigInteger *n*, final BigDecimal *x*) [static]

Divide and round.

Parameters

n	The numerator
x	The denominator

Returns

the divided n/x

Since

2009-08-05

9.4.2.22 static BigComplex org.nevec.rjm.BigDecimalMath.divideRound (final BigInteger *n*, final BigComplex *x*) [static]

Divide and round.

Parameters

n	The numerator
x	The denominator

Returns

the divided n/x

Since

2012-03-01

9.4.2.23 static BigDecimal org.nevec.rjm.BigDecimalMath.divideRound (final int *n*, final BigDecimal *x*) [static]

Divide and round.

Parameters

n	The numerator.
x	The denominator.

Returns

the divided n/x.

Since

2009-08-05

9.4.2.24 static BigDecimal org.nevec.rjm.BigDecimalMath.EllipticE (final BigDecimal *m*) [static]

Complete Elliptic Integral of the Second Kind.

Parameters

m	The parameter, equals squared sine of the modular angle. This often written k^2 in the reference works. The current implementation only allows input in the range |m|<1.

Returns

E(m)

Author

Richard J. Mathar

Since

2010-05-31

9.4.2.25 static BigDecimal org.nevec.rjm.BigDecimalMath.EllipticE (final BigDecimal *m,* final BigDecimal *phi*) [static]

Incomplete Elliptic Integral of the Second Kind.

This is implemented as a Taylor series in the parameter m, and therefore slow if m approaches 1: E(m,phi) = sum_{n>=0} binomial(1/2,n)*(-m)^n*int_{t=0..phi} sin^{2n}t dt.

Parameters

m	The parameter, equals squared sine of the modular angle. This often written k^2 in the reference works. The current implementation only allows input in the range |m|<1.
phi	the amplitude in radians

Returns

E(m,phi)

Author

Richard J. Mathar

Since

2010-05-31

9.4.2.26 static BigDecimal org.nevec.rjm.BigDecimalMath.EllipticK (final BigDecimal *m*) [static]

Complete Elliptic Integral of the First Kind.

Parameters

m	The parameter, equals squared sine of the modular angle. This often written k^2 in the reference works. The current implementation only allows input in the range |m|<1.

Returns

K(m)

Author

Richard J. Mathar

Since

2010-05-25

9.4.2.27 static int org.nevec.rjm.BigDecimalMath.err2prec (BigDecimal *x,* BigDecimal *xerr*) [static]

Convert an absolute error to a precision.

Parameters

x	The value of the variable
xerr	The absolute error in the variable

Returns

The number of valid digits in x. The value is rounded down, and on the pessimistic side for that reason.

Since

2009-06-25

9.4.2.28 static int org.nevec.rjm.BigDecimalMath.err2prec (double *x,* double *xerr*) [static]

Convert an absolute error to a precision.

Parameters

x	The value of the variable The value returned depends only on the absolute value, not on the sign.
xerr	The absolute error in the variable The value returned depends only on the absolute value, not on the sign.

Returns

The number of valid digits in x. Derived from the representation x+- xerr, as if the error was represented in a "half width" (half of the error bar) form. The value is rounded down, and on the pessimistic side for that reason.

Since

2009-05-30

9.4.2.29 static int org.nevec.rjm.BigDecimalMath.err2prec (double *xerr*) [static]

Convert a relative error to a precision.

Parameters

xerr	The relative error in the variable. The value returned depends only on the absolute value, not on the sign.

Returns

The number of valid digits in x. The value is rounded down, and on the pessimistic side for that reason.

Since

2009-08-05

9.4.2.30 static BigDecimal org.nevec.rjm.BigDecimalMath.exp (BigDecimal *x*) [static]

The exponential function.

Parameters

x	the argument.

Returns

exp(x). The precision of the result is implicitly defined by the precision in the argument. In particular this means that "Invalid Operation" errors are thrown if catastrophic cancellation of digits causes the result to have no valid digits left.

Since

2009-05-29

Author

Richard J. Mathar

9.4.2.31 static BigDecimal org.nevec.rjm.BigDecimalMath.exp (final MathContext *mc*) [static]

The base of the natural logarithm.

Parameters

mc	the required precision of the result

Returns

exp(1) = 2.71828....

Since

2009-05-29

9.4.2.32 static BigDecimal org.nevec.rjm.BigDecimalMath.F21series (final Rational *a,* final Rational *b,* final Rational *c,* final BigDecimal *x*) [static],[private]

2F1(a,b;c;z) Hypergeometric function.

This is the blind execution of the defining series. To be called once all other optimization options have been exhausted.

Parameters

a	The first numerator
b	The second numerator
c	The denominator
z	The argument

Author

Richard J. Mathar

Since

2010-05-25

9.4.2.33 static BigDecimal org.nevec.rjm.BigDecimalMath.gamma (MathContext *mc*) [static]

Euler-Mascheroni constant.

Parameters

mc	The required precision of the result.

Returns

0.577...

Since

2009-08-13

9.4.2.34 static BigDecimal org.nevec.rjm.BigDecimalMath.Gamma (final BigDecimal *x*) [static]

The Gamma function.

Parameters

x	The argument.

Returns

Gamma(x).

Since

2009-08-06

9.4.2.35 static BigDecimal org.nevec.rjm.BigDecimalMath.Gamma (final Rational *q*, final MathContext *mc*) [static]

The Gamma function.

Parameters

q	The argument.
mc	The required accuracy in the result.

Returns

Gamma(x).

Since

2010-05-26

9.4.2.36 static BigDecimal org.nevec.rjm.BigDecimalMath.Hypergeometric21 (final Rational *a,* final Rational *b,* final Rational *c,* final BigDecimal *x*) [static]

Gaussian Hypergeometric function with three rational parameters.

Parameters

a	The first numerator
b	The second numerator
c	The denominator
z	The argument

Returns

2F1(a,b;c;z) = sum_{n>=0} Pochhammer(a,n)*Pochhammer(b,n)*z^k/(Pochhammer(c,n)*n!)

Author

Richard J. Mathar

Since

2010-05-25

This is basically another implementation of RatPoly.valueOf(x,MathContext mc), and implements many of the standard orthogonal polynomials.

9.4.2.37 static BigDecimal org.nevec.rjm.BigDecimalMath.hypot (final BigDecimal *x,* final BigDecimal *y*) [static]

The hypotenuse.

Parameters

x	the first argument.
y	the second argument.

Returns

the square root of the sum of the squares of the two arguments, sqrt(x^2+y^2).

Since

2009-06-25

9.4.2.38 static BigDecimal org.nevec.rjm.BigDecimalMath.hypot (final int *n,* final BigDecimal *x*) [static]

The hypotenuse.

Parameters

n	the first argument.
x	the second argument.

Returns

the square root of the sum of the squares of the two arguments, sqrt(n^2+x^2).

Since

2009-08-05

9.4.2.39 static BigComplex org.nevec.rjm.BigDecimalMath.invertRound (final BigComplex *z*) [static]

Build the inverse and maintain the approximate accuracy.

Parameters

z	The denominator

Returns

The divided 1/z = [Re(z)-i∗Im(z)]/ [Re^2 z + Im^2 z]

Since

2010-07-19

9.4.2.40 static BigDecimal org.nevec.rjm.BigDecimalMath.log (BigDecimal *x*) [static]

The natural logarithm.

Parameters

x	the argument.

Returns

ln(x). The precision of the result is implicitly defined by the precision in the argument.

Since

2009-05-29

Author

Richard J. Mathar

9.4.2.41 static BigDecimal org.nevec.rjm.BigDecimalMath.log (int *n,* final MathContext *mc*) [static]

The natural logarithm.

Parameters

n	The main argument, a strictly positive integer.
mc	The requirements on the precision.

Returns

ln(n).

Since

2009-08-08

Author

Richard J. Mathar

9.4.2.42 static BigDecimal org.nevec.rjm.BigDecimalMath.log (final Rational *r*, final MathContext *mc*) [static]

The natural logarithm.

Parameters

r	The main argument, a strictly positive value.
mc	The requirements on the precision.

Returns

ln(r).

Since

2009-08-09

Author

Richard J. Mathar

9.4.2.43 static void org.nevec.rjm.BigDecimalMath.main (String *args[]*) [static]

Test programs.

This is the interface to printing numbers from the command line, like java -cp . org.nevec.rjm.BigDecimalMath sin 1.000000000

Parameters

args	The command line arguments, blank-separated. These consist of a function name like cos, exp, log, sin, sqrt etc, and one or two arguments depending on whether the function takes one or two arguments to calculate a value. The trigonometric functions take one argument in units of radians. The hypot takes two real-valued arguments. The pow takes two real-valued arguments, the base followed by the exponent. The root takes an integer (say 3 for the cubic root) plus a non-negative value for the basis.

Since

2009-07-30

Author

Richard J. Mathar

9.4.2.44 static BigDecimal org.nevec.rjm.BigDecimalMath.mod2pi (BigDecimal *x*) [static]

Reduce value to the interval [0,2∗Pi].

Parameters

x	the original value

Returns

the value modulo 2∗pi in the interval from 0 to 2∗pi.

Since

2009-06-01

9.4.2.45 static BigDecimal org.nevec.rjm.BigDecimalMath.modpi (BigDecimal *x*) [static]

Reduce value to the interval [-Pi/2,Pi/2].

Parameters

x	The original value

Returns

The value modulo pi, shifted to the interval from -Pi/2 to Pi/2.

Since

2009-07-31

9.4.2.46 static BigDecimal org.nevec.rjm.BigDecimalMath.multiplyRound (final BigDecimal *x*, final BigDecimal *y*) [static]

Multiply and round.

Parameters

x	The left factor.
y	The right factor.

Returns

The product x∗y.

Since

2009-07-30

9.4.2.47 static BigComplex org.nevec.rjm.BigDecimalMath.multiplyRound (final BigComplex *x*, final BigDecimal *y*) [static]

Multiply and round.

Parameters

x	The left factor.
y	The right factor.

Returns

The product x∗y.

Since

2010-07-19

9.4.2.48 static BigComplex org.nevec.rjm.BigDecimalMath.multiplyRound (final BigComplex *x*, final BigComplex *y*) [static]

Multiply and round.

Parameters

x	The left factor.
y	The right factor.

Returns

The product x∗y.

Since

2010-07-19

9.4.2.49 static BigDecimal org.nevec.rjm.BigDecimalMath.multiplyRound (final BigDecimal *x*, final Rational *f*) [static]

Multiply and round.

Parameters

x	The left factor.
f	The right factor.

Returns

The product x∗f.

Since

2009-07-30

9.4.2.50 static BigDecimal org.nevec.rjm.BigDecimalMath.multiplyRound (final BigDecimal *x*, final int *n*) [static]

Multiply and round.

Parameters

x	The left factor.
n	The right factor.

Returns

The product x∗n.

Since

2009-07-30

9.4.2.51 static BigDecimal org.nevec.rjm.BigDecimalMath.multiplyRound (final BigDecimal *x*, final BigInteger *n*) [static]

Multiply and round.

Parameters

x	The left factor.
n	The right factor.

Returns

the product x∗n

Since

2009-07-30

9.4.2.52 static BigDecimal org.nevec.rjm.BigDecimalMath.pi (final MathContext *mc*) [static]

Euler's constant.

Parameters

mc	The required precision of the result.

Returns

3.14159...

Since

2009-05-29

9.4.2.53 static BigDecimal org.nevec.rjm.BigDecimalMath.pochhammer (final BigDecimal *x*, final int *n*) [static]

Pochhammer's function.

Parameters

x	The main argument.
n	The non-negative index.

Returns

(x)_n = x(x+1)(x+2)∗...∗(x+n-1).

Since

2009-08-19

9.4.2.54 static BigDecimal org.nevec.rjm.BigDecimalMath.pow (final BigDecimal *x*, final BigDecimal *y*) [static]

Power function.

Parameters

x	Base of the power.
y	Exponent of the power.

Returns

x^y. The estimation of the relative error in the result is |log(x)∗err(y)|+|y∗err(x)/x|

Since

2009-06-01

9.4.2.55 static BigDecimal org.nevec.rjm.BigDecimalMath.powRound (final BigDecimal *x*, final int *n*) [static]

Raise to an integer power and round.

Parameters

x	The base.
n	The exponent.

Returns

x^n.

Since

2009-08-13
2010-05-26 handle also n<0 cases.

Special cases: x^1=x and x^0 = 1

9.4.2.56 static BigDecimal org.nevec.rjm.BigDecimalMath.powRound (final BigDecimal *x*, final BigInteger *n*) [static]

Raise to an integer power and round.

Parameters

x	The base.
n	The exponent. The current implementation allows n only in the interval of the standard int values.

Returns

x^n.

Since

2010-05-26

For now, the implementation forwards to the cases where n is in the range of the standard integers. This might, however, be implemented to decompose larger powers into cascaded calls to smaller ones.

9.4.2.57 static BigDecimal org.nevec.rjm.BigDecimalMath.powRound (final BigDecimal *x*, final Rational *q*) [static]

Raise to a fractional power and round.

Parameters

x	The base. Generally enforced to be positive, with the exception of integer exponents where the sign is carried over according to the parity of the exponent.
q	The exponent.

Returns

x^q.

Since

2010-05-26

Special cases: x^1=x and x^0 = 1

9.4.2.58 static double org.nevec.rjm.BigDecimalMath.prec2err (final double *x*, final int *prec*) [static]

Convert a precision (relative error) to an absolute error.

The is the inverse functionality of err2prec().

Parameters

x	The value of the variable The value returned depends only on the absolute value, not on the sign.
prec	The number of valid digits of the variable.

Returns

the absolute error in x. Derived from the an accuracy of one half of the ulp.

Since

2009-08-09

9.4.2.59 static double org.nevec.rjm.BigDecimalMath.psi (final double *x*) [static]

Digamma function.

Parameters

x	The main argument.

Returns

psi(x). The error is sometimes up to 10 ulp, where AS 6.3.15 suffers from cancellation of digits and psi=0

Since

2009-08-26

9.4.2.60 static double org.nevec.rjm.BigDecimalMath.psi (final int *n*, final double *x*) [static]

Polygamma functions.

Parameters

n	The positive integer argument. The digamma funcion requires n=0, the trigamma function n=1 etc.
x	The main argument.

Returns

psi^(n)(x).

Since

2009-08-20

9.4.2.61 static BigDecimal org.nevec.rjm.BigDecimalMath.root (final int *n*, final BigDecimal *x*) [static]

The integer root.

Parameters

n	the positive argument.
x	the non-negative argument.

Returns

The n-th root of the BigDecimal rounded to the precision implied by x, x^(1/n).

Since

2009-07-30

9.4.2.62 static BigDecimal org.nevec.rjm.BigDecimalMath.scalePrec (final BigDecimal *x*, int *d*) [static]

Append decimal zeros to the value.

This returns a value which appears to have a higher precision than the input.

Parameters

x	The input value
d	The (positive) value of zeros to be added as least significant digits.

Returns

The same value as the input but with increased (pseudo) precision.

9.4.2.63 static BigComplex org.nevec.rjm.BigDecimalMath.scalePrec (final BigComplex *x*, int *d*) [static]

Append decimal zeros to the value.

This returns a value which appears to have a higher precision than the input.

Parameters

x	The input value
d	The (positive) value of zeros to be added as least significant digits.

Returns

The same value as the input but with increased (pseudo) precision.

9.4.2.64 static BigDecimal org.nevec.rjm.BigDecimalMath.scalePrec (final BigDecimal *x*, final MathContext *mc*) [static]

Boost the precision by appending decimal zeros to the value.

This returns a value which appears to have a higher precision than the input.

Parameters

x	The input value
mc	The requirement on the minimum precision on return.

Returns

The same value as the input but with increased (pseudo) precision.

9.4.2.65 static BigDecimal org.nevec.rjm.BigDecimalMath.sin (final BigDecimal *x*) [static]

Trigonometric sine.

Parameters

x	The argument in radians.

Returns

sin(x) in the range -1 to 1.

Since

2009-06-01

9.4.2.66 static BigDecimal org.nevec.rjm.BigDecimalMath.sinh (final BigDecimal *x*) [static]

The hyperbolic sine.

Parameters

x	the argument.

Returns

the sinh(x) = (exp(x)-exp(-x))/2 .

Author

Richard J. Mathar

Since

2009-08-19

9.4.2.67 static BigDecimal org.nevec.rjm.BigDecimalMath.sqrt (final BigDecimal *x*, final MathContext *mc*) [static]

The square root.

Parameters

x	the non-negative argument.
mc	

Returns

the square root of the BigDecimal.

Since

2008-10-27

9.4.2.68 static BigDecimal org.nevec.rjm.BigDecimalMath.sqrt (final BigDecimal *x*) [static]

The square root.

Parameters

x	the non-negative argument.

Returns

the square root of the BigDecimal rounded to the precision implied by x.

Since

2009-06-25

9.4.2.69 static BigDecimal org.nevec.rjm.BigDecimalMath.subtractRound (final BigDecimal *x*, final BigDecimal *y*) [static]

Subtract and round according to the larger of the two ulp's.

Parameters

x	The left term.
y	The right term.

Returns

The difference x-y.

Since

2009-07-30

9.4.2.70 static BigComplex org.nevec.rjm.BigDecimalMath.subtractRound (final BigComplex *x*, final BigComplex *y*) [static]

Subtract and round according to the larger of the two ulp's.

Parameters

x	The left summand
y	The right summand

Returns

The difference x-y.

Since

2010-07-19

9.4.2.71 static BigDecimal org.nevec.rjm.BigDecimalMath.tan (final BigDecimal *x*) [static]

The trigonometric tangent.

Parameters

x	the argument in radians.

Returns

the tan(x)

9.4.2.72 static BigDecimal org.nevec.rjm.BigDecimalMath.tanh (final BigDecimal *x*) [static]

The hyperbolic tangent.

Parameters

x	The argument.

Returns

The tanh(x) = sinh(x)/cosh(x).

Author

Richard J. Mathar

Since

2009-08-20

9.4.2.73 static Rational org.nevec.rjm.BigDecimalMath.toRational (BigDecimal *x*) [static]

Convert the finite representation of a floating point value to its fraction.

Parameters

x	The number to be translated.

Returns

The rational number with the same decimal expansion as x.

Since

2012-03-09

9.4.2.74 static BigDecimal org.nevec.rjm.BigDecimalMath.zeta (final int *n*, final MathContext *mc*) [static]

Riemann zeta function.

Parameters

n	The positive integer argument.
mc	Specification of the accuracy of the result.

Returns

zeta(n).

Since

2009-08-05

9.4.2.75 static double org.nevec.rjm.BigDecimalMath.zeta1 (final int *n*) [static]

Riemann zeta function.

Parameters

n	The positive integer argument.

Returns

zeta(n)-1.

Since

2009-08-20

9.4.3 Member Data Documentation

9.4.3.1 int org.nevec.rjm.BigDecimalMath.TAYLOR_NTERM = 8 [static], [private]

A suggestion for the maximum numter of terms in the Taylor expansion of the exponential.

9.5 org.nevec.rjm.BigDecimalPoly Class Reference

Polynomial with BigDecimal coefficients.

Public Member Functions

- BigDecimalPoly ()
  Default ctor.
- BigDecimalPoly (final String L) throws NumberFormatException
  Creator with a comma-separated list as the list of coefficients.
- BigDecimalPoly (final BigIntegerPoly L, int d)
  Creator with a list of integer coefficients.

- BigDecimalPoly clone ()

  Create a copy of this.
- BigDecimal at (final int n)

  Retrieve a polynomial coefficient.
- BigDecimal valueOf (final BigDecimal x)

  Horner scheme with the function value at the argument x.
- BigComplex valueOf (final BigComplex x)

  Horner scheme with the function value at the argument x.
- BigDecimal valueOf (int x)

  Horner scheme to find the function value at the argument x.
- void set (final int n, final BigDecimal value)
- void setx ()

  Set to the taylor series representing 0+x.
- BigDecimalPoly multiply (final BigDecimal val)

  Multiply by a constant factor.
- BigDecimalPoly multiply (final BigDecimalPoly val)

  Multiply by another polynomial.
- BigDecimalPoly pow (final int n) throws ArithmeticException

  Raise to a positive power.
- BigDecimalPoly add (final BigDecimalPoly val)

  Add another polynomial.
- BigDecimalPoly subtract (final BigDecimalPoly val)

  Subtract another polynomial.
- BigDecimalPoly divide (final BigDecimal val)

  Divide by a constant.
- String toString ()

  Print as a comma-separated list of coefficients.
- String toPString ()

  Print as a polyomial in x.
- BigDecimalPoly derive ()

  First derivative.
- Vector< BigComplex > roots ()

  Generate the roots of the polynomial in floating point arithmetic.

Static Public Member Functions

- static void main (String args[])

Private Member Functions

- void simplify ()

  Simplify the representation.

9.5.1 Detailed Description

Polynomial with BigDecimal coefficients.

Alternatively to be interpreted as a sequence which has the polynomial as an (approximate) generating function. The internal representation is a vector of coefficients with at least one element (which may equal zero to represent the polynomial which always evaluates to zero).

Since

2010-07-19

Author

Richard J. Mathar

9.5.2 Constructor & Destructor Documentation

9.5.2.1 org.nevec.rjm.BigDecimalPoly.BigDecimalPoly ( )

Default ctor.

Creates the polynomial x=0.

9.5.2.2 org.nevec.rjm.BigDecimalPoly.BigDecimalPoly (final String *L*) throws NumberFormatException

Creator with a comma-separated list as the list of coefficients.

Parameters

L	the string of the form a0,a1,a2,a3 with the coefficients

9.5.2.3 org.nevec.rjm.BigDecimalPoly.BigDecimalPoly (final **BigIntegerPoly** *L*, int *d*)

Creator with a list of integer coefficients.

Parameters

p	The polynomial with integer coefficients.
d	The number of zeros to be appended to the integers to indicate their precision.

Since

2012-03-01

9.5.3 Member Function Documentation

9.5.3.1 **BigDecimalPoly** org.nevec.rjm.BigDecimalPoly.add (final **BigDecimalPoly** *val*)

Add another polynomial.

Parameters

val	the other polynomial

Returns

the sum of this with the other polynomial

Since

2010-07-19

9.5.3.2 BigDecimal org.nevec.rjm.BigDecimalPoly.at (final int *n*)

Retrieve a polynomial coefficient.

Parameters

n	the zero-based index of the coefficient. n=0 for the constant term.

Returns

the polynomial coefficient in front of x^n.

9.5.3.3 BigDecimalPoly org.nevec.rjm.BigDecimalPoly.clone ( )

Create a copy of this.

Since

2010-07-19

9.5.3.4 BigDecimalPoly org.nevec.rjm.BigDecimalPoly.derive ( )

First derivative.

Returns

The first derivative with respect to the indeterminate variable.

Since

2010-07-19

9.5.3.5 BigDecimalPoly org.nevec.rjm.BigDecimalPoly.divide (final BigDecimal *val*)

Divide by a constant.

Parameters

val	the constant through which the coefficients are divided.

Returns

the new polynomial this/val .

Since

2010-07-19

9.5.3.6 static void org.nevec.rjm.BigDecimalPoly.main (String *args[]*) [static]

9.5.3.7 BigDecimalPoly org.nevec.rjm.BigDecimalPoly.multiply (final BigDecimal *val*)

Multiply by a constant factor.

Parameters

val	the factor

Returns

the product of this with the factor. All coefficients of this have been multiplied individually by the factor.

9.5.3.8 BigDecimalPoly org.nevec.rjm.BigDecimalPoly.multiply (final BigDecimalPoly *val*)

Multiply by another polynomial.

Parameters

val	the other polynomial.

Returns

the product of this with the other polynomial. The coefficients of the new polynomial are a sum of the products of the coefficients of this and 'val'.

9.5.3.9 BigDecimalPoly org.nevec.rjm.BigDecimalPoly.pow (final int *n*) throws ArithmeticException

Raise to a positive power.

Parameters

n	the exponent of the power

Returns

the n-th power of this.

9.5.3.10 Vector<BigComplex> org.nevec.rjm.BigDecimalPoly.roots ( )

Generate the roots of the polynomial in floating point arithmetic.

See Also

Durand Kerner method

Since

2010-07-19

9.5.3.11 void org.nevec.rjm.BigDecimalPoly.set (final int *n*, final BigDecimal *value*)

9.5.3.12 void org.nevec.rjm.BigDecimalPoly.setx ( )

Set to the taylor series representing 0+x.

9.5.3.13 void org.nevec.rjm.BigDecimalPoly.simplify ( ) [private]

Simplify the representation.

Trailing values with zero coefficients (at high powers) are deleted.

9.5.3.14 BigDecimalPoly org.nevec.rjm.BigDecimalPoly.subtract (final BigDecimalPoly *val*)

Subtract another polynomial.

Parameters

val	the other polynomial

Returns

the difference between this and the other polynomial

Since

2010-07-19

9.5.3.15 String org.nevec.rjm.BigDecimalPoly.toPString ( )

Print as a polyomial in x.

Returns

To representation a0+a1∗x+a2∗x^2+... This does not print the terms with coefficients equal to zero.

Since

2010-07-19

9.5.3.16 String org.nevec.rjm.BigDecimalPoly.toString ( )

Print as a comma-separated list of coefficients.

Returns

the representation a0,a1,a2,a3,... This is a sort of opposite of the ctor that takes a string as an argument.

Since

2010-07-19

9.5.3.17 BigDecimal org.nevec.rjm.BigDecimalPoly.valueOf (final BigDecimal *x*)

Horner scheme with the function value at the argument x.

Parameters

x	The abscissa value of the argument.

Returns

The value of the polynomial at the argument.

Since

2010-07-19

9.5.3.18 BigComplex org.nevec.rjm.BigDecimalPoly.valueOf (final BigComplex *x*)

Horner scheme with the function value at the argument x.

Parameters

x	The abscissa value of the argument.

Returns

The value of the polynomial at the argument.

Since

2010-07-19

9.5.3.19 BigDecimal org.nevec.rjm.BigDecimalPoly.valueOf (int *x*)

Horner scheme to find the function value at the argument x.

Parameters

x	The abscissa value of the argument.

Returns

The value of the polynomial at the argument.

Since

2010-07-19

9.6 org.nevec.rjm.BigIntegerMath Class Reference

BigInteger special functions and Number theory.

Static Public Member Functions

- static void main (String args[])
  Test programs.
- static BigInteger binomial (final int n, final int k)
  Evaluate binomial(n,k).
- static BigInteger binomial (final BigInteger n, final BigInteger k)
  Evaluate binomial(n,k).
- static BigInteger sigmak (final BigInteger n, final int k)
  Evaluate sigma_k(n).
- static BigInteger sigma (int n)
  Evaluate sigma(n).
- static Vector< BigInteger > divisors (final BigInteger n)
  Compute the list of positive divisors.
- static BigInteger sigma (final BigInteger n)
  Evaluate sigma(n).
- static int isqrt (final int n)
  Evaluate floor(sqrt(n)).
- static long isqrt (final long n)
  Evaluate floor(sqrt(n)).
- static BigInteger isqrt (final BigInteger n)
  Evaluate floor(sqrt(n)).
- static BigInteger iroot (final BigInteger x, final int n)
  Evaluate floor(root[n](x)).
- static BigInteger core (final BigInteger n)

Evaluate core(n).

- static BigInteger[][] minor (final BigInteger[][] A, final int r, final int c) throws ArithmeticException

  Minor of an integer matrix.
- static BigInteger det (final BigInteger[][] A) throws ArithmeticException

  Determinant of an integer square matrix.
- static Rational[] solve (final BigInteger[][]A, final BigInteger[] rhs) throws ArithmeticException

  Solve a linear system of equations.
- static BigInteger lcm (final BigInteger a, final BigInteger b)

  The lowest common multiple.
- static Rational[][] linHRec (final BigInteger[]a, final int o) throws ArithmeticException

  Try to find a linear homogeneous recurrence of a sequence.
- static Rational[] linHRec (final BigInteger[] a, Integer[] firstval) throws ArithmeticException

  Try to find a linear homogeneous recurrence of a sequence.
- static Rational[] linHRec (final Vector< BigInteger > a, Integer[] firstval) throws ArithmeticException

  Try to find a linear homogeneous recurrence of a sequence.
- static BigInteger valueOf (final Vector< BigInteger >c, final BigInteger x)

  Evaluate the value of an integer polynomial at some integer argument.
- static Rational centrlFactNumt (int n, int k)

  The central factorial number t(n,k) number at the indices provided.
- static Rational centrlFactNumT (int n, int k)

  The central factorial number T(n,k) number at the indices provided.

Static Private Member Functions

- static BigInteger[][] colSubs (final BigInteger[][] A, final int c, final BigInteger[] v) throws ArithmeticException

  Replace column of a matrix with a column vector.

9.6.1 Detailed Description

BigInteger special functions and Number theory.

Since

2009-08-06

Author

Richard J. Mathar

9.6.2 Member Function Documentation

9.6.2.1 static BigInteger org.nevec.rjm.BigIntegerMath.binomial (final int *n*, final int *k*) [static]

Evaluate binomial(n,k).

Parameters

n	The upper index
k	The lower index

Returns

The binomial coefficient

9.6.2.2 static BigInteger org.nevec.rjm.BigIntegerMath.binomial (final BigInteger *n*, final BigInteger *k*) [static]

Evaluate binomial(n,k).

Parameters

n	The upper index
k	The lower index

Returns

The binomial coefficient

Since

2008-10-15

9.6.2.3 static Rational org.nevec.rjm.BigIntegerMath.centrlFactNumt (int *n*, int *k*) [static]

The central factorial number t(n,k) number at the indices provided.

Parameters

n	the first parameter, non-negative.
k	the second index, non-negative.

Returns

t(n,k)

Since

2009-08-06

Author

Richard J. Mathar

See Also

P. L. Butzer et al, Num. Funct. Anal. Opt. 10 (5)(1989) 419-488

9.6.2.4 static Rational org.nevec.rjm.BigIntegerMath.centrlFactNumT (int *n*, int *k*) [static]

The central factorial number T(n,k) number at the indices provided.

Parameters

n	the first parameter, non-negative.
k	the second index, non-negative.

Returns

T(n,k)

Since

2009-08-06

Author

Richard J. Mathar

See Also

P. L. Butzer et al, Num. Funct. Anal. Opt. 10 (5)(1989) 419-488

9.6.2.5 static BigInteger [][] org.nevec.rjm.BigIntegerMath.colSubs (final BigInteger *A[][]*, final int *c*, final BigInteger[] *v*) throws ArithmeticException [static], [private]

Replace column of a matrix with a column vector.

Parameters

A	The matrix.
c	The column index of the column to be substituted (0-based).
v	The column vector to be inserted. With the current implementation, it must be at least as long as the row count, and its elements that exceed that count are ignored.

Returns

The modified matrix. This is not a deep copy but contains references to the original.

Since

2010-08-27

Author

Richard J. Mathar

9.6.2.6 static BigInteger org.nevec.rjm.BigIntegerMath.core (final BigInteger *n*) [static]

Evaluate core(n).

Returns the smallest positive integer m such that n/m is a perfect square.

Parameters

n	The non-negative argument.

Returns

The square-free part of n.

Since

2011-02-12

Author

Richard J. Mathar

9.6.2.7 static BigInteger org.nevec.rjm.BigIntegerMath.det (final BigInteger *A[][]*) throws ArithmeticException [static]

Determinant of an integer square matrix.

Parameters

A	The square matrix. If column and row dimensions are unequal, an ArithmeticException is thrown.

Returns

The determinant.

Since

2010-08-27

Author

Richard J. Mathar

9.6.2.8 static Vector<BigInteger> org.nevec.rjm.BigIntegerMath.divisors (final BigInteger *n*) [static]

Compute the list of positive divisors.

Parameters

n	The integer of which the divisors are to be found.

Returns

The sorted list of positive divisors.

Since

2010-08-27

Author

Richard J. Mathar

9.6.2.9 static BigInteger org.nevec.rjm.BigIntegerMath.iroot (final BigInteger *x*, final int *n*) [static]

Evaluate floor(root[n](x)).

Parameters

x	The non-negative argument. Arguments less than zero throw an ArithmeticException.
n	The positive inverse power.

Returns

The integer n-th root of x, rounded down.

Since

2012-11-29

Author

Richard J. Mathar

9.6.2.10 static int org.nevec.rjm.BigIntegerMath.isqrt (final int *n*) [static]

Evaluate floor(sqrt(n)).

Parameters

n	The non-negative argument.

Returns

The integer square root. The square root rounded down.

Since

2010-08-27

Author

Richard J. Mathar

9.6.2.11 static long org.nevec.rjm.BigIntegerMath.isqrt (final long *n*) [static]

Evaluate floor(sqrt(n)).

Parameters

n	The non-negative argument. Arguments less than zero throw an ArithmeticException.

Returns

The integer square root, the square root rounded down.

Since

2010-08-27

Author

Richard J. Mathar

9.6.2.12 static BigInteger org.nevec.rjm.BigIntegerMath.isqrt (final BigInteger *n*) [static]

Evaluate floor(sqrt(n)).

Parameters

n	The non-negative argument. Arguments less than zero throw an ArithmeticException.

Returns

The integer square root, the square root rounded down.

Since

2011-02-12

Author

Richard J. Mathar

9.6.2.13 static BigInteger org.nevec.rjm.BigIntegerMath.lcm (final BigInteger *a*, final BigInteger *b*) [static]

The lowest common multiple.

Parameters

a	The first argument
b	The second argument

Returns

lcm(|a|,|b|)

Since

2010-08-27

Author

Richard J. Mathar

9.6.2.14 static Rational [][] org.nevec.rjm.BigIntegerMath.linHRec (final BigInteger[] *a*, final int *o*) throws ArithmeticException [static]

Try to find a linear homogeneous recurrence of a sequence.

Parameters

a	The (finite) sequence.
o	The attempted order of the recurrence.

Returns

The matrix of the individual coefficients. This contains 0 rows and columns if one of the linear system of equations was inconsistent.

Since

2010-08-27

Author

Richard J. Mathar

9.6.2.15 static Rational [] org.nevec.rjm.BigIntegerMath.linHRec (final BigInteger[] *a,* Integer[] *firstval*) throws ArithmeticException [static]

Try to find a linear homogeneous recurrence of a sequence.

Parameters

a	The (finite) sequence.
firstval	On return indicates that only valid a(n) = ... for n >= firstval.

Returns

The expansion coefficients. This vector has zero length if no solutions are found.

Since

2010-08-27

Author

Richard J. Mathar

9.6.2.16 static Rational [] org.nevec.rjm.BigIntegerMath.linHRec (final Vector< BigInteger > *a,* Integer[] *firstval*) throws ArithmeticException [static]

Try to find a linear homogeneous recurrence of a sequence.

Parameters

a	The (finite) sequence.

Returns

The expansion coefficients. This vector has zero length if no solutions are found.

Since

2010-08-27

Author

Richard J. Mathar

9.6.2.17 static void org.nevec.rjm.BigIntegerMath.main (String *args[]*) [static]

Test programs.

Since

2009-08-06

Author

Richard J. Mathar

9.6.2.18 static BigInteger [][] org.nevec.rjm.BigIntegerMath.minor (final BigInteger *A[][],* final int *r,* final int *c*) throws ArithmeticException [static]

Minor of an integer matrix.

Parameters

A	The matrix.
r	The row index of the row to be removed (0-based). An exception is thrown if this is outside the range 0 to the upper row index of A.
c	The column index of the column to be removed (0-based). An exception is thrown if this is outside the range 0 to the upper column index of A.

Returns

The depleted matrix. This is not a deep copy but contains references to the original.

Since

2010-08-27

Author

Richard J. Mathar

9.6.2.19 static BigInteger org.nevec.rjm.BigIntegerMath.sigma (int *n*) [static]

Evaluate sigma(n).

Parameters

n	the argument for which divisors will be searched.

Returns

the sigma function. Sum of the positive divisors of the argument.

Since

2006-08-14

Author

Richard J. Mathar

9.6.2.20 static BigInteger org.nevec.rjm.BigIntegerMath.sigma (final BigInteger *n*) [static]

Evaluate sigma(n).

Parameters

n	the argument for which divisors will be searched.

Returns

the sigma function. Sum of the divisors of the argument.

Since

2006-08-14

Author

Richard J. Mathar

9.6.2.21 static BigInteger org.nevec.rjm.BigIntegerMath.sigmak (final BigInteger *n*, final int *k*) [static]

Evaluate sigma_k(n).

Parameters

n	the main argument which defines the divisors
k	the lower index, which defines the power

Returns

The sum of the k-th powers of the positive divisors

9.6.2.22 static Rational [] org.nevec.rjm.BigIntegerMath.solve (final *BigIntegerA[][]*, final BigInteger[] *rhs*) throws ArithmeticException [static]

Solve a linear system of equations.

Parameters

A	The square matrix. If it is not of full rank, an ArithmeticException is thrown.
rhs	The right hand side. The length of this vector must match the matrix size; else an ArithmeticException is thrown.

Returns

The vector of x in A∗x=rhs.

Since

2010-08-28

Author

Richard J. Mathar

9.6.2.23 static BigInteger org.nevec.rjm.BigIntegerMath.valueOf (final Vector< BigInteger > *c*, final BigInteger *x*) [static]

Evaluate the value of an integer polynomial at some integer argument.

Parameters

c	Represents the coefficients c[0]+c[1]∗x+c[2]∗x^2+.. of the polynomial
x	The abscissa point of the evaluation

Returns

The polynomial value.

Since

2010-08-27

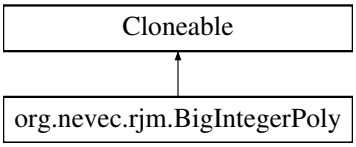
Author

Richard J. Mathar

9.7 org.nevec.rjm.BigIntegerPoly Class Reference

Polynomial with integer coefficients.

Inheritance diagram for org.nevec.rjm.BigIntegerPoly:



Public Member Functions

- BigIntegerPoly ()
  Default ctor.
- BigIntegerPoly (final String L) throws NumberFormatException
  Ctor with a comma-separated list as the list of coefficients.
- BigIntegerPoly (final Vector< BigInteger > c)
  Ctor with a list of coefficients.
- BigIntegerPoly (final BigInteger[] c)
  Ctor with a list of coefficients.
- BigIntegerPoly clone ()
  Create a copy of this.
- BigDecimalPoly toBigDecimalPoly (int d)
  Translate into a BigDecimal copy.
- RatPoly toRatPoly ()
  Translate into a RatPoly copy.
- BigInteger at (final int n)
  Retrieve a polynomial coefficient.
- BigInteger valueOf (final BigInteger x)
  Evaluate at some integer argument.
- BigInteger valueOf (int x)
  Horner scheme to find the function value at the argument x.
- void set (final int n, final BigInteger value)
  Set a polynomial coefficient.
- void set (final int n, final int value)
  Set a polynomial coefficient.
- int size ()
  Count of coefficients.
- int degree ()
  Polynomial degree.
- int ldegree ()
  Polynomial lower degree.

- BigIntegerPoly multiply (final BigInteger val)

  Multiply by a constant factor.
- BigIntegerPoly multiply (final BigIntegerPoly val)

  Multiply by another polynomial.
- BigIntegerPoly pow (final int n) throws ArithmeticException

  Raise to a positive power.
- BigIntegerPoly add (final BigIntegerPoly val)

  Add another polynomial.
- BigIntegerPoly subtract (final BigIntegerPoly val)

  Subtract another polynomial.
- BigIntegerPoly divide (final BigIntegerPoly val, int nmax)

  Divide by another polynomial.
- BigIntegerPoly[] divideAndRemainder (final BigIntegerPoly val)

  Divide by another polynomial.
- String toString ()

  Print as a comma-separated list of coefficients.
- String toPString ()

  Print as a polyomial in x.
- BigIntegerPoly derive ()

  First derivative.
- BigIntegerPoly trunc (int newdeg)

  Truncate polynomial degree.
- BigIntegerPoly binomialTInv (int maxdeg)

  Inverse Binomial transform.
- int rootDeg (final BigInteger r)

  Compute the order of the root r.
- Vector< BigInteger > iroots ()

  Generate the integer roots of the polynomial.
- boolean isZero ()

  Test whether this polynomial value is zero.
- Vector< BigIntegerPoly > ifactor ()

  Factorization into integer polynomials.

Static Public Member Functions

- static void main (String args[])

  Hermite polynomial.

Static Public Attributes

- static final BigIntegerPoly ONE = new BigIntegerPoly("1")

  The constant 1.
- static final BigIntegerPoly X = new BigIntegerPoly("0,1")

  The linear function x.

Protected Member Functions

- void simplify ()

  Simplify the representation.
- Vector< BigIntegerPoly > i2roots ()

  Generate the factors which are 2nd degree polynomials.

9.7.1 Detailed Description

Polynomial with integer coefficients.

Alternatively to be interpreted as a sequence which has the polynomial as an (approximate) generating function.

Since

2010-08-27

Author

Richard J. Mathar

9.7.2 Constructor & Destructor Documentation

9.7.2.1 org.nevec.rjm.BigIntegerPoly.BigIntegerPoly ( )

Default ctor.

Creates the polynomial p(x)=0.

9.7.2.2 org.nevec.rjm.BigIntegerPoly.BigIntegerPoly (final String *L*) throws NumberFormatException

Ctor with a comma-separated list as the list of coefficients.

Parameters

L	the string of the form a0,a1,a2,a3 with the coefficients

9.7.2.3 org.nevec.rjm.BigIntegerPoly.BigIntegerPoly (final Vector< BigInteger > *c*)

Ctor with a list of coefficients.

Parameters

c	The coefficients a0, a1, a2 etc in a0+a1∗x+a2∗x^2+...

9.7.2.4 org.nevec.rjm.BigIntegerPoly.BigIntegerPoly (final BigInteger[] *c*)

Ctor with a list of coefficients.

Parameters

c	The coefficients a0, a1, a2 etc in a0+a1∗x+a2∗x^2+...

9.7.3 Member Function Documentation

9.7.3.1 BigIntegerPoly org.nevec.rjm.BigIntegerPoly.add (final BigIntegerPoly *val*)

Add another polynomial.

Parameters

val	the other polynomial

Returns

the sum of this with the other polynomial

Since

2010-08-27

9.7.3.2 BigInteger org.nevec.rjm.BigIntegerPoly.at (final int *n*)

Retrieve a polynomial coefficient.

Parameters

n	the zero-based index of the coefficient. n=0 for the constant term.

Returns

the polynomial coefficient in front of x^n.

9.7.3.3 BigIntegerPoly org.nevec.rjm.BigIntegerPoly.binomialTInv (int *maxdeg*)

Inverse Binomial transform.

Parameters

maxdeg	the maximum polynomial degree of the result

Returns

the sequence of coefficients is the inverse binomial transform of the original sequence.

Since

2010-08-29

9.7.3.4 BigIntegerPoly org.nevec.rjm.BigIntegerPoly.clone ( )

Create a copy of this.

Since

2010-08-27

9.7.3.5 int org.nevec.rjm.BigIntegerPoly.degree ( )

Polynomial degree.

Returns

the polynomial degree.

9.7.3.6 BigIntegerPoly org.nevec.rjm.BigIntegerPoly.derive ( )

First derivative.

Returns

The first derivative with respect to the indeterminate variable.

Since

2008-10-26

9.7.3.7 BigIntegerPoly org.nevec.rjm.BigIntegerPoly.divide (final BigIntegerPoly *val,* int *nmax*)

Divide by another polynomial.

Parameters

val	the other polynomial
nmax	the maximum degree of the Taylor expansion of the result.

Returns

the Taylor expansion of this/val up to degree nmax.

9.7.3.8 BigIntegerPoly [] org.nevec.rjm.BigIntegerPoly.divideAndRemainder (final BigIntegerPoly *val*)

Divide by another polynomial.

Parameters

val	the other polynomial

Returns

A vector with [0] containg the polynomial of degree which is the difference of the degree of this and the degree of val. [1] the remainder polynomial. This = returnvalue[0] + returnvalue[1]/val .

Since

2012-03-01

9.7.3.9 Vector<BigIntegerPoly> org.nevec.rjm.BigIntegerPoly.i2roots ( ) [protected]

Generate the factors which are 2nd degree polynomials.

Returns

A (potentially empty) vector of factors, without multiplicity. Only factors with non-zero absolute coefficient are generated. This means the factors are of the form x^2+a*x+b=0 with nonzero b.

Since

2012-03-01

9.7.3.10 Vector<BigIntegerPoly> org.nevec.rjm.BigIntegerPoly.ifactor ( )

Factorization into integer polynomials.

The current factorization detects only factors which are polynomials of order up to 2.

Returns

The vector of factors. Factors with higher multiplicity are represented by repetition.

Since

2012-03-01

9.7.3.11 Vector<BigInteger> org.nevec.rjm.BigIntegerPoly.iroots ( )

Generate the integer roots of the polynomial.

Returns

The vector of integer roots, without their multiplicity.

Since

2010-08-27

9.7.3.12 boolean org.nevec.rjm.BigIntegerPoly.isZero ( )

Test whether this polynomial value is zero.

Returns

If this is a polynomial p(x)=0 for all x.

9.7.3.13 int org.nevec.rjm.BigIntegerPoly.ldegree ( )

Polynomial lower degree.

Returns

power of the smallest non-zero coefficient. If the polynomial is identical to 0, 0 is returned.

9.7.3.14 static void org.nevec.rjm.BigIntegerPoly.main (String *args[]*) [static]

Hermite polynomial.

Chebysehv polynomialLaguerre polynomial

9.7.3.15 BigIntegerPoly org.nevec.rjm.BigIntegerPoly.multiply (final BigInteger *val*)

Multiply by a constant factor.

Parameters

val	the factor

Returns

the product of this with the factor. All coefficients of this have been multiplied individually by the factor.

Since

2010-08-27

9.7.3.16 BigIntegerPoly org.nevec.rjm.BigIntegerPoly.multiply (final BigIntegerPoly *val*)

Multiply by another polynomial.

Equivalent to a convolution of the polynomial coefficients.

Parameters

val	the other polynomial

Returns

the product of this with the other polynomial

9.7.3.17 BigIntegerPoly org.nevec.rjm.BigIntegerPoly.pow (final int *n*) throws ArithmeticException

Raise to a positive power.

Parameters

n	the exponent of the power

Returns

the n-th power of this.

9.7.3.18 int org.nevec.rjm.BigIntegerPoly.rootDeg (final BigInteger *r*)

Compute the order of the root r.

Returns

1 for simple roots, 2 for order 2 etc., 0 if not a root

Since

2010-08-27

9.7.3.19 void org.nevec.rjm.BigIntegerPoly.set (final int *n,* final BigInteger *value*)

Set a polynomial coefficient.

Parameters

n	the zero-based index of the coefficient. n=0 for the constant term. If the polynomial has not yet the degree to need this coefficient, the intermediate coefficients are set to zero.
value	the new value of the coefficient.

9.7.3.20 void org.nevec.rjm.BigIntegerPoly.set (final int *n,* final int *value*)

Set a polynomial coefficient.

Parameters

n	the zero-based index of the coefficient. n=0 for the constant term. If the polynomial has not yet the degree to need this coefficient, the intermediate coefficients are implicitly set to zero.
value	the new value of the coefficient.

9.7.3.21 void org.nevec.rjm.BigIntegerPoly.simplify ( ) [protected]

Simplify the representation.

Trailing values with zero coefficients (at high powers) are deleted.

9.7.3.22 int org.nevec.rjm.BigIntegerPoly.size ( )

Count of coefficients.

Returns

the number of polynomial coefficients. Differs from the polynomial degree by one.

9.7.3.23 BigIntegerPoly org.nevec.rjm.BigIntegerPoly.subtract (final BigIntegerPoly *val*)

Subtract another polynomial.

Parameters

val	the other polynomial

Returns

the difference between this and the other polynomial

Since

2008-10-25

9.7.3.24 BigDecimalPoly org.nevec.rjm.BigIntegerPoly.toBigDecimalPoly (int *d*)

Translate into a BigDecimal copy.

Parameters

d	The number of zeros to be appended to each coefficient.

Since

2012-03-01

9.7.3.25 String org.nevec.rjm.BigIntegerPoly.toPString ( )

Print as a polyomial in x.

Returns

The representation a0+a1∗x+a2∗x^2+... The terms with zero coefficients are not mentioned.

Since

2008-10-26

9.7.3.26 RatPoly org.nevec.rjm.BigIntegerPoly.toRatPoly ( )

Translate into a RatPoly copy.

Since

2012-03-02

9.7.3.27 String org.nevec.rjm.BigIntegerPoly.toString ( )

Print as a comma-separated list of coefficients.

Returns

the representation a0,a1,a2,a3,...

Since

2010-08-27

9.7.3.28 BigIntegerPoly org.nevec.rjm.BigIntegerPoly.trunc (int *newdeg*)

Truncate polynomial degree.

Returns

The polynomial with all coefficients beyond deg set to zero.

Since

2010-08-27

9.7.3.29 BigInteger org.nevec.rjm.BigIntegerPoly.valueOf (final BigInteger *x*)

Evaluate at some integer argument.

Parameters

x	The abscissa point of the evaluation

Returns

The polynomial value.

Since

2010-08-27

Author

Richard J. Mathar

9.7.3.30 BigInteger org.nevec.rjm.BigIntegerPoly.valueOf (int *x*)

Horner scheme to find the function value at the argument x.

Parameters

x	The argument x.

Returns

Value of the polynomial at x.

Since

2008-11-13

9.7.4 Member Data Documentation

9.7.4.1 final BigIntegerPoly org.nevec.rjm.BigIntegerPoly.ONE = new BigIntegerPoly("1") [static]

The constant 1.

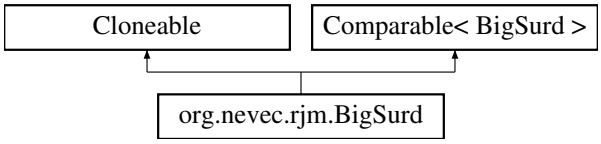
9.7.4.2 final BigIntegerPoly org.nevec.rjm.BigIntegerPoly.X = new BigIntegerPoly("0,1") [static]

The linear function x.

9.8 org.nevec.rjm.BigSurd Class Reference

Square roots on the real line.

Inheritance diagram for org.nevec.rjm.BigSurd:



Public Member Functions

- BigSurd ()
  Default ctor, which represents the zero.
- BigSurd (Rational a, Rational b)
  ctor given the prefactor and the basis of the root.
- BigSurd (int a, int b)
  ctor given the numerator and denominator of the root.
- BigSurd (BigInteger a)
  ctor given the value under the root.
- BigSurd clone ()
  Create a deep copy.
- BigSurdVec add (final BigSurd val)
  Add two surds of compatible discriminant.
- BigSurd multiply (final BigSurd val)
  Multiply by another square root.
- BigSurd multiply (final Rational val)
  Multiply by a rational number.
- BigSurd multiply (final BigInteger val)
  Multiply by a BigInteger.
- BigSurd multiply (final int val)
  Multiply by an integer.
- Rational sqr ()
  Compute the square.
- BigSurd divide (final BigSurd val)
  Divide by another square root.
- BigSurd divide (final BigInteger val)
  Divide by an integer.
- BigSurd divide (int val)
  Divide by an integer.

- BigSurd negate ()

  Compute the negative.
- BigSurd abs ()

  Absolute value.
- int compareTo (final BigSurd val)

  Compares the value of this with another constant.
- String toString ()

  Return a string in the format (number/denom)∗()^(1/2).
- double doubleValue ()

  Return a double value representation.
- float floatValue ()

  Return a float value representation.
- boolean isBigInteger ()

  True if the value is integer.
- boolean isRational ()

  True if the value is rational.
- Rational toRational ()

  Convert to a rational value if possible.
- int signum ()

  The sign: 1 if the number is >0, 0 if ==0, -1 if <0.
- BigDecimal BigDecimalValue (MathContext mc)

  Return the approximate floating point representation.

Static Public Member Functions

- static void main (String args[])

  Test programs.

Static Public Attributes

- static BigSurd ZERO = new BigSurd()

  The value of zero.
- static BigSurd ONE = new BigSurd(Rational.ONE,Rational.ONE)

  The value of one.

Protected Member Functions

- void normalize ()

  Normalize to squarefree discriminant.
- void normalizeG ()

  Normalize to coprime numerator and denominator in prefactor and discriminant.

9.8.1 Detailed Description

Square roots on the real line.

These represent numbers which are a product of a (signed) fraction by a square root of a non-negative fraction. This might be extended to values on the imaginary axis by allowing negative values underneath the square root, but this is not yet implemented.

Since

2011-02-12

Author

Richard J. Mathar

9.8.2 Constructor & Destructor Documentation

9.8.2.1 org.nevec.rjm.BigSurd.BigSurd ( )

Default ctor, which represents the zero.

Since

2011-02-12

9.8.2.2 org.nevec.rjm.BigSurd.BigSurd (Rational *a*, Rational *b*)

ctor given the prefactor and the basis of the root.

This creates an object of value a∗sqrt(b).

Parameters

a	the prefactor.
b	the discriminant.

Since

2011-02-12

9.8.2.3 org.nevec.rjm.BigSurd.BigSurd (int *a*, int *b*)

ctor given the numerator and denominator of the root.

This creates an object of value sqrt(a/b).

Parameters

a	the numerator
b	the denominator.

Since

2011-02-12

9.8.2.4 org.nevec.rjm.BigSurd.BigSurd (BigInteger *a*)

ctor given the value under the root.

This creates an object of value sqrt(a).

Parameters

a	the discriminant.

Since

2011-02-12

9.8.3 Member Function Documentation

9.8.3.1 BigSurd org.nevec.rjm.BigSurd.abs ( )

Absolute value.

Returns

The absolute (non-negative) value of this.

Since

2011-02-12

9.8.3.2 BigSurdVec org.nevec.rjm.BigSurd.add (final BigSurd *val*)

Add two surds of compatible discriminant.

Parameters

val	The value to be added to this.

9.8.3.3 BigDecimal org.nevec.rjm.BigSurd.BigDecimalValue (MathContext *mc*)

Return the approximate floating point representation.

Parameters

mc	Description of the accuracy needed.

Returns

A representation with digits valid as described by mc

Since

2012-02-15

9.8.3.4 BigSurd org.nevec.rjm.BigSurd.clone ( )

Create a deep copy.

Since

2011-02-12

9.8.3.5 int org.nevec.rjm.BigSurd.compareTo (final BigSurd *val*)

Compares the value of this with another constant.

Parameters

val	the other constant to compare with

Returns

-1, 0 or 1 if this number is numerically less than, equal to, or greater than val.

Since

2011-02-12

9.8.3.6 BigSurd org.nevec.rjm.BigSurd.divide (final BigSurd *val*)

Divide by another square root.

Parameters

val	A second number of this type.

Returns

The value of this/val

Since

2011-02-12

9.8.3.7 BigSurd org.nevec.rjm.BigSurd.divide (final BigInteger *val*)

Divide by an integer.

Parameters

val	a second number.

Returns

the value of this/val

Since

2011-02-12

9.8.3.8 BigSurd org.nevec.rjm.BigSurd.divide (int *val*)

Divide by an integer.

Parameters

val	A second number.

Returns

The value of this/val

Since

2011-02-12

9.8.3.9 double org.nevec.rjm.BigSurd.doubleValue ( )

Return a double value representation.

Returns

The value with double precision.

Since

2011-02-12

9.8.3.10 float org.nevec.rjm.BigSurd.floatValue ( )

Return a float value representation.

Returns

The value with single precision.

Since

2011-02-12

9.8.3.11 boolean org.nevec.rjm.BigSurd.isBigInteger ( )

True if the value is integer.

Equivalent to the indication whether a conversion to an integer can be exact.

Since

2011-02-12

9.8.3.12 boolean org.nevec.rjm.BigSurd.isRational ( )

True if the value is rational.

Equivalent to the indication whether a conversion to a Rational can be exact.

Since

2011-02-12

9.8.3.13 static void org.nevec.rjm.BigSurd.main (String *args[]*) [static]

Test programs.

Since

2011-02-13

Author

Richard J. Mathar

9.8.3.14 BigSurd org.nevec.rjm.BigSurd.multiply (final BigSurd *val*)

Multiply by another square root.

Parameters

val	a second number of this type.

Returns

the product of this with the val.

Since

2011-02-12

9.8.3.15 BigSurd org.nevec.rjm.BigSurd.multiply (final Rational *val*)

Multiply by a rational number.

Parameters

val	the factor.

Returns

the product of this with the val.

Since

2011-02-15

9.8.3.16 BigSurd org.nevec.rjm.BigSurd.multiply (final BigInteger *val*)

Multiply by a BigInteger.

Parameters

val	a second number.

Returns

the product of this with the value.

Since

2011-02-12

9.8.3.17 BigSurd org.nevec.rjm.BigSurd.multiply (final int *val*)

Multiply by an integer.

Parameters

val	a second number.

Returns

the product of this with the value.

Since

2011-02-12

9.8.3.18 BigSurd org.nevec.rjm.BigSurd.negate ( )

Compute the negative.

Returns

-this.

Since

2011-02-12

9.8.3.19 void org.nevec.rjm.BigSurd.normalize ( ) [protected]

Normalize to squarefree discriminant.

Since

2011-02-12

9.8.3.20 void org.nevec.rjm.BigSurd.normalizeG ( ) [protected]

Normalize to coprime numerator and denominator in prefactor and discriminant.

Since

2011-02-12

9.8.3.21 int org.nevec.rjm.BigSurd.signum ( )

The sign: 1 if the number is >0, 0 if ==0, -1 if <0.

Returns

the signum of the value.

Since

2011-02-12

9.8.3.22 Rational org.nevec.rjm.BigSurd.sqr ( )

Compute the square.

Returns

this value squared.

Since

2011-02-12

9.8.3.23 Rational org.nevec.rjm.BigSurd.toRational ( )

Convert to a rational value if possible.

Since

2012-02-15

9.8.3.24 String org.nevec.rjm.BigSurd.toString ( )

Return a string in the format (number/denom)∗()^(1/2).

If the discriminant equals 1, print just the prefactor.

Returns

the human-readable version in base 10

Since

2011-02-12

9.8.4 Member Data Documentation

9.8.4.1 BigSurd org.nevec.rjm.BigSurd.ONE = new BigSurd(Rational.ONE,Rational.ONE) [static]

The value of one.

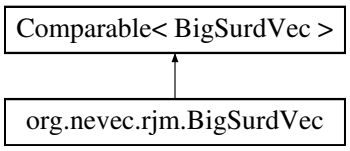
9.8.4.2 BigSurd org.nevec.rjm.BigSurd.ZERO = new BigSurd() [static]

The value of zero.

9.9 org.nevec.rjm.BigSurdVec Class Reference

A BigSurdVec represents an algebraic sum or differences of values which each term an instance of BigSurd.

Inheritance diagram for org.nevec.rjm.BigSurdVec:



Public Member Functions

- BigSurdVec ()
  Default ctor, which represents the zero.
- BigSurdVec (BigSurd a)
  ctor given the value of a BigSurd.
- BigSurdVec (BigSurd a, BigSurd b)
  ctor given two values, which (when added) represent this number a+b.
- BigSurdVec (Vector< BigInteger > a) throws NumberFormatException
  ctor given a periodic continued fraction.
- int compareTo (BigSurdVec oth)

Compare algebraic value with oth.

- int signum ()

  Sign function.
- BigDecimal BigDecimalValue (MathContext mc)

  Construct an approximate floating point representation.
- double doubleValue ()

  Construct an approximate floating point representation.
- double floatValue ()

  Construct an approximate floating point representation.
- BigSurdVec add (final Rational val)

  Add two vectors algebraically.
- BigSurdVec add (final BigSurdVec val)

  Add two vectors algebraically.
- BigSurdVec add (final BigSurd val)

  Add two vectors algebraically.
- BigSurdVec subtract (final BigSurdVec val)

  Subtract another number.
- BigSurdVec subtract (final BigSurd val)

  Subtract another number.
- BigSurdVec negate ()

  Compute the negative.
- BigSurdVec sqr ()

  Compute the square.
- BigSurdVec multiply (final BigSurd val)

  Multiply by another square root.
- String toString ()

  Return a string in the format (number/denom)∗()^(1/2).

Static Public Member Functions

- static void main (String args[])

  Test programs.

Static Public Attributes

- static BigSurdVec ZERO = new BigSurdVec()

  The value of zero.
- static BigSurdVec ONE = new BigSurdVec(BigSurd.ONE)

  The value of one.

Protected Member Functions

- void normalize ()

  Combine terms that can be written as a single surd.

9.9.1 Detailed Description

A BigSurdVec represents an algebraic sum or differences of values which each term an instance of BigSurd.

This mainly means that sums or differences of two BigSurd (or two BigSurdVec) can be represented (exactly) as a BigSurdVec.

Since

2012-02-15

Author

Richard J. Mathar

9.9.2 Constructor & Destructor Documentation

9.9.2.1 org.nevec.rjm.BigSurdVec.BigSurdVec ( )

Default ctor, which represents the zero.

Since

2012-02-15

9.9.2.2 org.nevec.rjm.BigSurdVec.BigSurdVec (BigSurd *a*)

ctor given the value of a BigSurd.

Parameters

a	The value to be represented by this vector.

Since

2012-02-15

9.9.2.3 org.nevec.rjm.BigSurdVec.BigSurdVec (BigSurd *a*, BigSurd *b*)

ctor given two values, which (when added) represent this number a+b.

Parameters

a	The value to be represented by the first term of the vector.
b	The value to be represented by the second term of the vector.

Since

2012-02-15

9.9.2.4 org.nevec.rjm.BigSurdVec.BigSurdVec (Vector< BigInteger > *a*) throws NumberFormatException

ctor given a periodic continued fraction.

The value to be represented has a cont. fraction [a0,a1,a2,...,a0,a1,a2,...]

Parameters

a	The coefficients a[0], a[1],.. of the period of the continued fraction.

Since

2012-03-08

9.9.3 Member Function Documentation

9.9.3.1 BigSurdVec org.nevec.rjm.BigSurdVec.add (final Rational *val*)

Add two vectors algebraically.

Parameters

val	The value to be added to this.

Returns

The new value representing this+val.

9.9.3.2 BigSurdVec org.nevec.rjm.BigSurdVec.add (final BigSurdVec *val*)

Add two vectors algebraically.

Parameters

val	The value to be added to this.

Returns

The new value representing this+val.

9.9.3.3 BigSurdVec org.nevec.rjm.BigSurdVec.add (final BigSurd *val*)

Add two vectors algebraically.

Parameters

val	The value to be added to this.

Returns

The new value representing this+val.

9.9.3.4 BigDecimal org.nevec.rjm.BigSurdVec.BigDecimalValue (MathContext *mc*)

Construct an approximate floating point representation.

Parameters

mc	The intended accuracy of the result.

Returns

A truncated version with the precision described by mc

9.9.3.5 int org.nevec.rjm.BigSurdVec.compareTo (BigSurdVec *oth*)

Compare algebraic value with oth.

Returns -1, 0 or +1 depending on whether this is smaller, equal to or larger than oth.

Parameters

oth	The value with which this is to be compared.

Returns

0 or +-1.

Since

2012-02-15

9.9.3.6 double org.nevec.rjm.BigSurdVec.doubleValue ( )

Construct an approximate floating point representation.

Returns

A truncated version with the precision described by mc

9.9.3.7 double org.nevec.rjm.BigSurdVec.floatValue ( )

Construct an approximate floating point representation.

Returns

A truncated version with the precision described by mc

9.9.3.8 static void org.nevec.rjm.BigSurdVec.main (String *args[]*) [static]

Test programs.

Since

2011-02-13

Author

Richard J. Mathar

9.9.3.9 BigSurdVec org.nevec.rjm.BigSurdVec.multiply (final BigSurd *val*)

Multiply by another square root.

Parameters

val	a second number of this type.

Returns

the product of this with the val.

Since

2011-02-12

9.9.3.10 BigSurdVec org.nevec.rjm.BigSurdVec.negate ( )

Compute the negative.

Returns

-this.

Since

2012-02-15

9.9.3.11 void org.nevec.rjm.BigSurdVec.normalize ( ) [protected]

Combine terms that can be written as a single surd.

This unites for example the terms sqrt(90) and sqrt(10) to 4∗sqrt(10).

Since

2012-02-15

9.9.3.12 int org.nevec.rjm.BigSurdVec.signum ( )

Sign function.

Returns -1, 0 or +1 depending on whether this is smaller, equal to or larger than zero.

Returns

0 or +-1.

Since

2012-02-15

9.9.3.13 BigSurdVec org.nevec.rjm.BigSurdVec.sqr ( )

Compute the square.

Returns

this value squared.

Since

2012-02-15

9.9.3.14 BigSurdVec org.nevec.rjm.BigSurdVec.subtract (final BigSurdVec *val*)

Subtract another number.

Parameters

val	The value to be subtracted from this.

Returns

The new value representing this-val.

9.9.3.15 BigSurdVec org.nevec.rjm.BigSurdVec.subtract (final BigSurd *val*)

Subtract another number.

Parameters

val	The value to be subtracted from this.

Returns

The new value representing this-val.

9.9.3.16 String org.nevec.rjm.BigSurdVec.toString ( )

Return a string in the format (number/denom)∗()^(1/2).

If the discriminant equals 1, print just the prefactor.

Returns

the human-readable version in base 10

Since

2012-02-16

9.9.4 Member Data Documentation

9.9.4.1 BigSurdVec org.nevec.rjm.BigSurdVec.ONE = new BigSurdVec(BigSurd.ONE) [static]

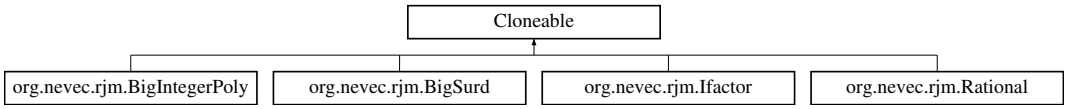
The value of one.

9.9.4.2 BigSurdVec org.nevec.rjm.BigSurdVec.ZERO = new BigSurdVec() [static]

The value of zero.

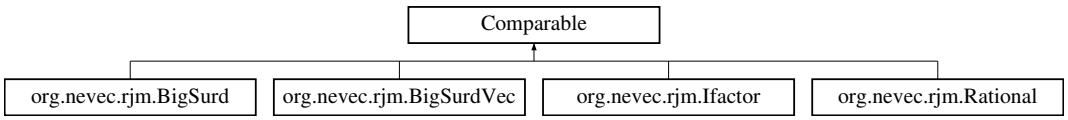
9.10 Cloneable Class Reference

Inheritance diagram for Cloneable:



9.11 Comparable Class Reference

Inheritance diagram for Comparable:



9.12 org.nevec.rjm.Euler Class Reference

Euler numbers.

Public Member Functions

- Euler ()
  Ctor().
- BigInteger at (int n)
  The Euler number at the index provided.

Protected Member Functions

- void set (final int n)
  Compute a coefficient in the internal table.

Static Protected Attributes

- static Vector< BigInteger > a = new Vector<BigInteger>()

9.12.1 Detailed Description

Euler numbers.

See Also

A000364 in the OEIS.

Since

2008-10-30

Author

Richard J. Mathar

9.12.2 Constructor & Destructor Documentation

9.12.2.1 org.nevec.rjm.Euler.Euler ( )

Ctor().

Fill the hash list initially with E_0 to E_3.

9.12.3 Member Function Documentation

9.12.3.1 BigInteger org.nevec.rjm.Euler.at (int *n*)

The Euler number at the index provided.

Parameters

n	the index, non-negative.

Returns

the E_0=E_1=1 , E_2=5, E_3=61 etc

9.12.3.2 void org.nevec.rjm.Euler.set (final int *n*) [protected]

Compute a coefficient in the internal table.

Parameters

n	the zero-based index of the coefficient. n=0 for the E_0 term.

9.12.4 Member Data Documentation

9.12.4.1 Vector<BigInteger> org.nevec.rjm.Euler.a = new Vector<BigInteger>() [static], [protected]

9.13 org.nevec.rjm.EulerPhi Class Reference

Euler totient function.

Public Member Functions

- EulerPhi ()
  Default constructor.
- BigInteger at (int n)
  Compute phi(n).
- BigInteger at (BigInteger n)
  Compute phi(n).

Static Public Member Functions

- static void main (String[] args) throws ArithmeticException
  Test program.

9.13.1 Detailed Description

Euler totient function.

See Also

A000010 in the OEIS.

Since

2008-10-14
2012-03-04 Adapted to new Ifactor representation.

Author

Richard J. Mathar

9.13.2 Constructor & Destructor Documentation

9.13.2.1 org.nevec.rjm.EulerPhi.EulerPhi ( )

Default constructor.

Does nothing().

9.13.3 Member Function Documentation

9.13.3.1 BigInteger org.nevec.rjm.EulerPhi.at (int *n*)

Compute phi(n).

Parameters

n	The positive argument of the function.

Returns

phi(n)

9.13.3.2 BigInteger org.nevec.rjm.EulerPhi.at (BigInteger *n*)

Compute phi(n).

Parameters

n	The positive argument of the function.

Returns

phi(n)

9.13.3.3 static void org.nevec.rjm.EulerPhi.main (String[] *args*) throws ArithmeticException [static]

Test program.

It takes one argument n and prints the value phi(n).

java -cp . org.nevec.rjm.EulerPhi n

Since

2006-08-14

9.14 org.nevec.rjm.Factorial Class Reference

Factorials.

Public Member Functions

- Factorial ()
  ctor().
- BigInteger at (int n)
  Compute the factorial of the non-negative integer.
- lfactor tolfactor (int n)
  Compute the factorial of the non-negative integer.

Static Public Member Functions

- static void main (String[] args) throws Exception

Private Member Functions

- void growto (int n)

  Extend the internal table to cover up to n!

9.14.1 Detailed Description

Factorials.

Since

2006-06-25
2012-02-15 Storage of the values based on lfactor, not BigInteger.

Author

Richard J. Mathar

9.14.2 Constructor & Destructor Documentation

9.14.2.1 org.nevec.rjm.Factorial.Factorial ( )

ctor().

Initialize the vector of the factorials with 0!=1 and 1!=1.

9.14.3 Member Function Documentation

9.14.3.1 BigInteger org.nevec.rjm.Factorial.at (int *n*)

Compute the factorial of the non-negative integer.

Parameters

n	the argument to the factorial, non-negative.

Returns

the factorial of n.

9.14.3.2 void org.nevec.rjm.Factorial.growto (int *n*) [private]

Extend the internal table to cover up to n!

Parameters

n	The maximum factorial to be supported.

Since

2012-02-15

9.14.3.3 static void org.nevec.rjm.Factorial.main (String[] *args*) throws Exception [static]

9.14.3.4 lfactor org.nevec.rjm.Factorial.tolfactor (int *n*)

Compute the factorial of the non-negative integer.

Parameters

n	the argument to the factorial, non-negative.

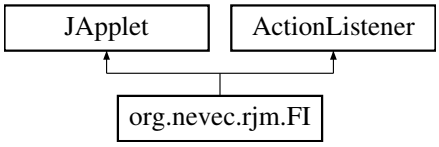
Returns

the factorial of n.

9.15 org.nevec.rjm.FI Class Reference

Applet of an integer calculator with Forth-alike syntax.

Inheritance diagram for org.nevec.rjm.FI:



Public Member Functions

- void start ()
- void init ()
- void actionPerformed (ActionEvent e)

  Interpreter parser loop.

9.15.1 Detailed Description

Applet of an integer calculator with Forth-alike syntax.

Author

Richard J. Mathar www.strw.leidenuniv.nl/∼mathar

Since

2008-10-13

9.15.2 Member Function Documentation

9.15.2.1 void org.nevec.rjm.FI.actionPerformed (ActionEvent *e*)

Interpreter parser loop.

Parameters

e	the information on which button had been pressed in the GUI

9.15.2.2 void org.nevec.rjm.FI.init ( )

9.15.2.3 void org.nevec.rjm.FI.start ( )

9.16 org.nevec.rjm.Harmonic Class Reference

Harmonic numbers.

Public Member Functions

- Harmonic ()

  ctor() Does nothing.
- Rational at (int n)

  The Harmonic number at the index specified.

9.16.1 Detailed Description

Harmonic numbers.

H(n) is the sum of the inverses of the integers from 1 to n.

Since

2008-10-19

Author

Richard J. Mathar

9.16.2 Constructor & Destructor Documentation

9.16.2.1 org.nevec.rjm.Harmonic.Harmonic ( )

ctor() Does nothing.

9.16.3 Member Function Documentation

9.16.3.1 Rational org.nevec.rjm.Harmonic.at (int *n*)

The Harmonic number at the index specified.

Parameters

n	the index, non-negative.

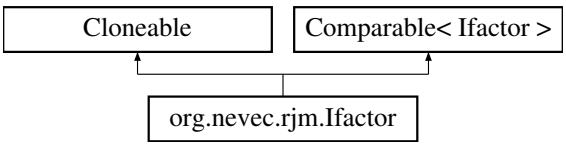
Returns

the H_1=1 for n=1, H_2=3/2 for n=2 etc. For values of n less than 1, zero is returned.

9.17 org.nevec.rjm.Ifactor Class Reference

Factored integers.

Inheritance diagram for org.nevec.rjm.Ifactor:



Public Member Functions

- Ifactor (int number)

  Constructor given an integer.
- Ifactor (BigInteger number)

  Constructor given a BigInteger .
- Ifactor (Vector< Integer > pows)

  Constructor given a list of exponents of the prime factor decomposition.
- Ifactor (Ifactor oth)

  Copy constructor.
- Ifactor clone ()

  Deep copy.
- boolean equals (final Ifactor oth)

  Comparison of two numbers.
- Ifactor multiply (final BigInteger oth)

  Multiply with another positive integer.
- Ifactor multiply (final int oth)

  Multiply with another positive integer.
- Ifactor multiply (final Ifactor oth)

  Multiply with another positive integer.
- Ifactor lcm (final Ifactor oth)

  Lowest common multiple of this with oth.
- Ifactor gcd (final Ifactor oth)

  Greatest common divisor of this and oth.
- Ifactor divide (final Ifactor oth)

  Integer division through another positive integer.
- Ifactor add (final BigInteger oth)

  Summation with another positive integer.
- Ifactor pow (final int exponent) throws ArithmeticException

  Exponentiation with a positive integer.
- Rational root (final int r) throws ArithmeticException

  Pulling the r-th root.
- Vector< BigInteger > divisors ()

  The set of positive divisors.
- Ifactor sigma ()

  Sum of the divisors of the number.
- Ifactor sigma (int k)

  Sum of the k-th powers of divisors of the number.
- Ifactor dropPrime ()

  Divide through the highest possible power of the highest prime.
- boolean issquare ()

  Test whether this is a square of an integer (perfect square).
- int bigomega ()

  The sum of the prime factor exponents, with multiplicity.

- int omega ()

  The sum of the prime factor exponents, without multiplicity.
- BigInteger core ()

  The square-free part.
- int moebius ()

  The Moebius function.
- Ifactor max (final Ifactor oth)

  Maximum of two values.
- Ifactor min (final Ifactor oth)

  Minimum of two values.
- int compareTo (final Ifactor oth)

  Compare value against another Ifactor.
- String toString ()

  Convert to printable format.

Static Public Member Functions

- static Ifactor max (final Vector< Ifactor > set)

  Maximum of a list of values.
- static Ifactor min (final Vector< Ifactor > set)

  Minimum of a list of values.
- static void main (String[] args) throws Exception

  Test program.

Public Attributes

- BigInteger n

  The standard representation of the number.
- Vector< Integer > primeexp

Static Public Attributes

- final static Ifactor ONE = new Ifactor(1)
- final static Ifactor ZERO = new Ifactor(0)

Protected Member Functions

- Ifactor multGcdLcm (final Ifactor oth, int type)

  Multiply with another positive integer.

9.17.1 Detailed Description

Factored integers.

This class contains a non-negative integer with the prime factor decomposition attached.

Since

2006-08-14
2012-02-14 The internal representation contains the bases, and becomes sparser if few prime factors are present.

Author

Richard J. Mathar

9.17.2 Constructor & Destructor Documentation

9.17.2.1 org.nevec.rjm.Ifactor.Ifactor (int *number*)

Constructor given an integer.

constructor with an ordinary integer

Parameters

number	the standard representation of the integer

9.17.2.2 org.nevec.rjm.Ifactor.Ifactor (BigInteger *number*)

Constructor given a BigInteger .

Constructor with an ordinary integer, calling a prime factor decomposition.

Parameters

number	the BigInteger representation of the integer

9.17.2.3 org.nevec.rjm.Ifactor.Ifactor (Vector< Integer > *pows*)

Constructor given a list of exponents of the prime factor decomposition.

Parameters

pows	the vector with the sorted list of exponents. pows[0] is the exponent of 2, pows[1] the exponent of 3, pows[2] the exponent of 5 etc. Note that this list does not include the primes, but assumes a continuous prime-smooth basis.

9.17.2.4 org.nevec.rjm.Ifactor.Ifactor (Ifactor *oth*)

Copy constructor.

Parameters

oth	the value to be copied

9.17.3 Member Function Documentation

9.17.3.1 Ifactor org.nevec.rjm.Ifactor.add (final BigInteger *oth*)

Summation with another positive integer.

Parameters

oth	the other term.

Returns

the sum of both numbers

9.17.3.2 int org.nevec.rjm.Ifactor.bigomega ( )

The sum of the prime factor exponents, with multiplicity.

Returns

the sum over the primeexp numbers

9.17.3.3 Ifactor org.nevec.rjm.Ifactor.clone ( )

Deep copy.

Since

2009-08-14

9.17.3.4 int org.nevec.rjm.Ifactor.compareTo (final Ifactor *oth*)

Compare value against another Ifactor.

Parameters

oth	The value to be compared agains.

Returns

1, 0 or -1 according to being larger, equal to or smaller than oth.

Since

2012-02-15

9.17.3.5 BigInteger org.nevec.rjm.Ifactor.core ( )

The square-free part.

Returns

the minimum m such that m times this number is a square.

Since

2008-10-16

9.17.3.6 Ifactor org.nevec.rjm.Ifactor.divide (final Ifactor *oth*)

Integer division through another positive integer.

Parameters

oth	the denominator.

Returns

the division of this through the oth, discarding the remainder.

9.17.3.7 Vector<BigInteger> org.nevec.rjm.Ifactor.divisors ( )

The set of positive divisors.

Returns

the vector of divisors of the absolute value, sorted.

Since

2010-08-27

9.17.3.8 Ifactor org.nevec.rjm.Ifactor.dropPrime ( )

Divide through the highest possible power of the highest prime.

If the current number is the prime factor product p1^e1 * p2*e2* p3^e3*...*py^ey * pz^ez, the value returned has the final factor pz^ez eliminated, which gives p1^e1 * p2*e2* p3^e3*...*py^ey.

Returns

the new integer obtained by removing the highest prime power. If this here represents 0 or 1, it is returned without change.

Since

2006-08-20

9.17.3.9 boolean org.nevec.rjm.Ifactor.equals (final Ifactor *oth*)

Comparison of two numbers.

The value of this method is in allowing the Vector<>.contains() calls that use the value, not the reference for comparison.

Parameters

oth	the number to compare this with.

Returns

true if both are the same numbers, false otherwise.

9.17.3.10 Ifactor org.nevec.rjm.Ifactor.gcd (final Ifactor *oth*)

Greatest common divisor of this and oth.

Parameters

oth	the second parameter of gcd(this,oth)

Returns

the lowest common multiple of both numbers. Returns zero if any of both arguments is zero.

9.17.3.11 boolean org.nevec.rjm.Ifactor.issquare ( )

Test whether this is a square of an integer (perfect square).

Returns

true if this is an integer squared (including 0), else false

9.17.3.12 Ifactor org.nevec.rjm.Ifactor.lcm (final Ifactor *oth*)

Lowest common multiple of this with oth.

Parameters

oth	the second parameter of lcm(this,oth)

Returns

the lowest common multiple of both numbers. Returns zero if any of both arguments is zero.

9.17.3.13 static void org.nevec.rjm.Ifactor.main (String[] *args*) throws Exception [static]

Test program.

It takes a single argument n and prints the integer factorizaton.

java -cp . org.nevec.rjm.Ifactor n

9.17.3.14 Ifactor org.nevec.rjm.Ifactor.max (final Ifactor *oth*)

Maximum of two values.

Parameters

oth	the number to compare this with.

Returns

the larger of the two values.

9.17.3.15 static Ifactor org.nevec.rjm.Ifactor.max (final Vector< Ifactor > *set*) [static]

Maximum of a list of values.

Parameters

set	list of numbers.

Returns

the largest in the list.

9.17.3.16 Ifactor org.nevec.rjm.Ifactor.min (final Ifactor *oth*)

Minimum of two values.

Parameters

oth	the number to compare this with.

Returns

the smaller of the two values.

9.17.3.17 static Ifactor org.nevec.rjm.Ifactor.min (final Vector< Ifactor > *set*) [static]

Minimum of a list of values.

Parameters

set	list of numbers.

Returns

the smallest in the list.

9.17.3.18 int org.nevec.rjm.Ifactor.moebius ( )

The Moebius function.

1 if n=1, else, if k is the number of distinct prime factors, return (-1)^k, else, if k has repeated prime factors, return 0.

Returns

the moebius function.

9.17.3.19 Ifactor org.nevec.rjm.Ifactor.multGcdLcm (final Ifactor *oth,* int *type*) [protected]

Multiply with another positive integer.

Parameters

oth	the second factor.
type	0 to multiply, 1 for gcd, 2 for lcm

Returns

the product, gcd or lcm of both numbers.

9.17.3.20 Ifactor org.nevec.rjm.Ifactor.multiply (final BigInteger *oth*)

Multiply with another positive integer.

Parameters

oth	the second factor.

Returns

the product of both numbers.

9.17.3.21 Ifactor org.nevec.rjm.Ifactor.multiply (final int *oth*)

Multiply with another positive integer.

Parameters

oth	the second factor.

Returns

the product of both numbers.

9.17.3.22 Ifactor org.nevec.rjm.Ifactor.multiply (final Ifactor *oth*)

Multiply with another positive integer.

Parameters

oth	the second factor.

Returns

the product of both numbers.

9.17.3.23 int org.nevec.rjm.Ifactor.omega ( )

The sum of the prime factor exponents, without multiplicity.

Returns

the number of distinct prime factors.

Since

2008-10-16

9.17.3.24 Ifactor org.nevec.rjm.Ifactor.pow (final int *exponent*) throws ArithmeticException

Exponentiation with a positive integer.

Parameters

exponent	the non-negative exponent

Returns

n^exponent. If exponent=0, the result is 1.

9.17.3.25 Rational org.nevec.rjm.Ifactor.root (final int *r*) throws ArithmeticException

Pulling the r-th root.

Parameters

r	the positive or negative (nonzero) root.

Returns

n^(1/r). The return value falls into the Ifactor class if r is positive, but if r is negative a Rational type is needed.

Since

2009-05-18

9.17.3.26 Ifactor org.nevec.rjm.Ifactor.sigma ( )

Sum of the divisors of the number.

Returns

the sum of all divisors of the number, 1+....+n.

9.17.3.27 Ifactor org.nevec.rjm.Ifactor.sigma (int *k*)

Sum of the k-th powers of divisors of the number.

Returns

the sum of all divisors of the number, 1^k+....+n^k.

9.17.3.28 String org.nevec.rjm.Ifactor.toString ( )

Convert to printable format.

Returns

a string of the form n:prime^pow∗prime^pow∗prime^pow...

9.17.4 Member Data Documentation

9.17.4.1 BigInteger org.nevec.rjm.Ifactor.n

The standard representation of the number.

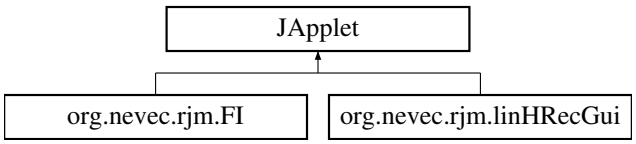
9.17.4.2 final static Ifactor org.nevec.rjm.Ifactor.ONE = new Ifactor(1) [static]

9.17.4.3 Vector<Integer> org.nevec.rjm.Ifactor.primeexp

9.17.4.4 final static Ifactor org.nevec.rjm.Ifactor.ZERO = new Ifactor(0) [static]

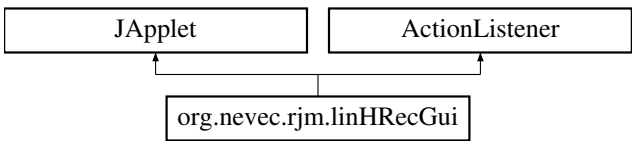
9.18 JApplet Class Reference

Inheritance diagram for JApplet:



9.19 org.nevec.rjm.linHRecGui Class Reference

Inheritance diagram for org.nevec.rjm.linHRecGui:



Public Member Functions

- void init ()
- void actionPerformed (ActionEvent e)

  Interpreter parser loop.

Static Public Member Functions

- static void main (String[] args)

  Main entry point.

Private Member Functions

- String modPri (final Vector< BigInteger > a, String in, int p)

  Append the string with a report on remainders of the vector, mod p.

9.19.1 Detailed Description

Author

Richard J. Mathar `home page`

Since

2010-08-27

9.19.2 Member Function Documentation

9.19.2.1 void org.nevec.rjm.linHRecGui.actionPerformed (ActionEvent *e*)

Interpreter parser loop.

Parameters

e	the information on which button had been pressed in the GUI

Since

2010-08-27

9.19.2.2 void org.nevec.rjm.linHRecGui.init ( )

Since

2010-08-27

9.19.2.3 static void org.nevec.rjm.linHRecGui.main (String[] *args*) `[static]`

Main entry point.

The usage for interactive choices and interactive parameter selection is not taking any command line options:

java -jar linHRecGui.jar

This is a solver to detect homogeneous linear recurrences with constant coefficients in sequences of integer numbers. The input is a sequence of integers, separated by white space (including line breaks) and/or commas. This is fed into the black "Input" area. (Clicking on "Clear Input" erases this input area.) Clicking on "Search" tests for a

linear recurrence with rational coefficients. It only returns a positive result if the *all* terms involved match the same recurrence; so it may be advisable to omit the first few terms of a sequence if these corrupt the recurrence.

Examples of inputs that obey linear recurrences are in the OEIS.

See Also

- Other transformations of series (Binomial, inverse Binomial, Catalan, Motzkin, Euler, ...) are implemented in my online calculator FI.html.
- The API. The backbone are the BigIntegerMath.solve() calls to see whether the coefficients that fit consecutive terms to a linear recurrence are consistent for all terms.
- R. J. Mathar, Some linear recurrences with constant coefficients
- Anonymous, Recurrence equation, Wikipedia
- Herbert Wilf, generatingfunctionology
- Maple users have gfun with guessgf()

Since

2012-02-14

Author

Richard J. Mathar

9.19.2.4 String org.nevec.rjm.linHRecGui.modPri (final Vector< BigInteger > *a*, String *in*, int *p*) [private]

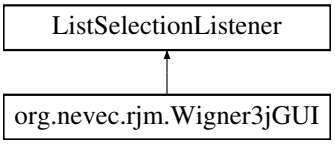
Append the string with a report on remainders of the vector, mod p.

Since

2010-08-28

9.20 ListSelectionListener Class Reference

Inheritance diagram for ListSelectionListener:



9.21 org.nevec.rjm.PartitionsP Class Reference

Number of partitions.

Public Member Functions

- PartitionsP ()

  Default constructor initializing a list of partitions up to 7.
- BigInteger at (int i)

  return the number of partitions of i

Static Public Member Functions

- static void main (String[] args) throws Exception

  Test program.

Static Protected Attributes

- static Vector< BigInteger > a = new Vector<BigInteger>()

  The list of all partitions as a vector.
- static BigInteger nMax =new BigInteger("-1")

  The maximum integer covered by the high end of the list.

Private Member Functions

- void growto (BigInteger n)

  extend the list of known partitions up to n

9.21.1 Detailed Description

Number of partitions.

Since

2008-10-15

Author

Richard J. Mathar

9.21.2 Constructor & Destructor Documentation

9.21.2.1 org.nevec.rjm.PartitionsP.PartitionsP ( )

Default constructor initializing a list of partitions up to 7.

9.21.3 Member Function Documentation

9.21.3.1 BigInteger org.nevec.rjm.PartitionsP.at (int *i*)

return the number of partitions of i

Parameters

i	the zero-based index into the list of partitions

Returns

the ith partition number. This is 1 if i=0 or 1, 2 if i=2 and so forth.

9.21.3.2 void org.nevec.rjm.PartitionsP.growto (BigInteger *n*) [private]

extend the list of known partitions up to n

Parameters

n	the maximum integer hashed after the call.

9.21.3.3 static void org.nevec.rjm.PartitionsP.main (String[] *args*) throws Exception [static]

Test program.

It takes one integer argument n and prints P(n).

java -cp . org.nevec.rjm.PartitionsP n

Since

2008-10-15

9.21.4 Member Data Documentation

9.21.4.1 Vector<BigInteger> org.nevec.rjm.PartitionsP.a = new Vector<BigInteger>() [static], [protected]

The list of all partitions as a vector.

9.21.4.2 BigInteger org.nevec.rjm.PartitionsP.nMax =new BigInteger("-1") [static], [protected]

The maximum integer covered by the high end of the list.

9.22 org.nevec.rjm.Prime Class Reference

Prime numbers.

Public Member Functions

- Prime ()
  Default constructor initializing a list of primes up to 17.
- boolean contains (BigInteger n)
  Test if a number is a prime.
- boolean isSPP (final BigInteger n, final BigInteger a)
  Test whether a number n is a strong pseudoprime to base a.
- int millerRabin (final BigInteger n)
  Miller-Rabin primality tests.
- BigInteger at (int i)
  return the ithprime
- BigInteger pi (BigInteger n)
  return the count of primes <= n
- BigInteger nextprime (BigInteger n)
  return the smallest prime larger than n
- BigInteger prevprime (BigInteger n)
  return the largest prime smaller than n

Static Public Member Functions

- static void main (String[] args) throws Exception
  Test program.

Protected Member Functions

- void growto (BigInteger n)

  extend the list of known primes up to n

Static Protected Attributes

- static BigInteger nMax = new BigInteger("-1")

  The maximum integer covered by the high end of the list.

9.22.1 Detailed Description

Prime numbers.

The implementation is a very basic computation of the set of all primes on demand, growing infinitely without any defined upper limit. The effects of such scheme are (i) the lookup-times become shorter after a while as more and more primes have been used and stored. The applications appear to become faster. (ii) Using the implementation for factorizations may easily require all available memory and stall finally, because indeed a dense list of primes with growing upper bound is kept without any hashing or lagging scheme.

Since

2006-08-11

Author

Richard J. Mathar

9.22.2 Constructor & Destructor Documentation

9.22.2.1 org.nevec.rjm.Prime.Prime ( )

Default constructor initializing a list of primes up to 17.

17 is enough to call the Miller-Rabin tests on the first 7 primes without further action.

9.22.3 Member Function Documentation

9.22.3.1 BigInteger org.nevec.rjm.Prime.at (int *i*)

return the ithprime

Parameters

i	the zero-based index into the list of primes

Returns

the ith prime. This is 2 if i=0, 3 if i=1 and so forth.

9.22.3.2 boolean org.nevec.rjm.Prime.contains (BigInteger *n*)

Test if a number is a prime.

Parameters

n	the integer to be tested for primality

Returns

true if prime, false if not

9.22.3.3 void org.nevec.rjm.Prime.growto (BigInteger *n*) [protected]

extend the list of known primes up to n

Parameters

n	the maximum integer known to be prime or not prime after the call.

9.22.3.4 boolean org.nevec.rjm.Prime.isSPP (final BigInteger *n*, final BigInteger *a*)

Test whether a number n is a strong pseudoprime to base a.

Parameters

n	the integer to be tested for primality
a	the base

Returns

true if the test is passed, so n may be a prime. false if the test is not passed, so n is not a prime.

Since

2010-02-25

9.22.3.5 static void org.nevec.rjm.Prime.main (String[] *args*) throws Exception [static]

Test program.

Usage: java -cp . org.nevec.rjm.Prime n

This takes a single argument (n) and prints prime(n), the previous and next prime, and pi(n).

Since

2006-08-14

9.22.3.6 int org.nevec.rjm.Prime.millerRabin (final BigInteger *n*)

Miller-Rabin primality tests.

Parameters

n	The prime candidate

Returns

-1 if n is a composite, 1 if it is a prime, 0 if it may be a prime.

Since

2010-02-25

9.22.3.7 BigInteger org.nevec.rjm.Prime.nextprime (BigInteger *n*)

return the smallest prime larger than n

Parameters

n	lower limit of the search

Returns

the next larger prime.

Since

2008-10-16

9.22.3.8 BigInteger org.nevec.rjm.Prime.pi (BigInteger *n*)

return the count of primes <= n

Parameters

n	the upper limit of the scan

Returns

the ith prime. This is 2 if i=0, 3 if i=1 and so forth.

9.22.3.9 BigInteger org.nevec.rjm.Prime.prevprime (BigInteger *n*)

return the largest prime smaller than n

Parameters

n	upper limit of the search

Returns

the next smaller prime.

Since

2008-10-17

9.22.4 Member Data Documentation

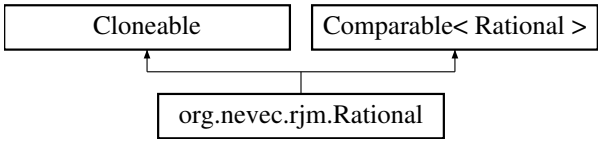
9.22.4.1 BigInteger org.nevec.rjm.Prime.nMax = new BigInteger("-1") [static], [protected]

The maximum integer covered by the high end of the list.

9.23 org.nevec.rjm.Rational Class Reference

Fractions (rational numbers).

Inheritance diagram for org.nevec.rjm.Rational:



Public Member Functions

- Rational ()

  Default ctor, which represents the zero.
- Rational (BigInteger a, BigInteger b)

  ctor from a numerator and denominator.
- Rational (BigInteger a)

  ctor from a numerator.
- Rational (int a, int b)

  ctor from a numerator and denominator.
- Rational (int n)

  ctor from an integer.
- Rational (String str) throws NumberFormatException

  ctor from a string representation.
- Rational (String str, int radix) throws NumberFormatException

  ctor from a string representation in a specified base.
- Rational (Vector< BigInteger > cfr)

  ctor from a terminating continued fraction.
- Rational clone ()

  Create a copy.
- Rational multiply (final Rational val)

  Multiply by another fraction.
- Rational multiply (final BigInteger val)

  Multiply by a BigInteger.
- Rational multiply (final int val)

  Multiply by an integer.
- Rational pow (int exponent)

  Power to an integer.
- Rational pow (BigInteger exponent) throws NumberFormatException

  Power to an integer.
- Rational root (BigInteger r) throws NumberFormatException

  r-th root.
- Rational pow (Rational exponent) throws NumberFormatException

  Raise to a rational power.
- Rational divide (final Rational val)

  Divide by another fraction.
- Rational divide (BigInteger val)

  Divide by an integer.
- Rational divide (int val)

  Divide by an integer.
- Rational add (Rational val)

  Add another fraction.
- Rational add (BigInteger val)

  Add another integer.

- Rational add (int val)
  Add another integer.
- Rational negate ()
  Compute the negative.
- Rational subtract (Rational val)
  Subtract another fraction.
- Rational subtract (BigInteger val)
  Subtract an integer.
- Rational subtract (int val)
  Subtract an integer.
- BigInteger numer ()
  Get the numerator.
- BigInteger denom ()
  Get the denominator.
- Rational abs ()
  Absolute value.
- BigInteger floor ()
  floor(): the nearest integer not greater than this.
- BigInteger ceil ()
  ceil(): the nearest integer not smaller than this.
- BigInteger trunc ()
  Remove the fractional part.
- int compareTo (final Rational val)
  Compares the value of this with another constant.
- int compareTo (final BigInteger val)
  Compares the value of this with another constant.
- String toString ()
  Return a string in the format number/denom.
- double doubleValue ()
  Return a double value representation.
- float floatValue ()
  Return a float value representation.
- BigDecimal BigDecimalValue (MathContext mc)
  Return a representation as BigDecimal.
- String toFString (int digits)
  Return a string in floating point format.
- Rational max (final Rational val)
  Compares the value of this with another constant.
- Rational min (final Rational val)
  Compares the value of this with another constant.
- Rational Pochhammer (final BigInteger n)
  Compute Pochhammer's symbol (this)_n.
- Rational Pochhammer (int n)
  Compute pochhammer's symbol (this)_n.
- boolean isBigInteger ()
  True if the value is integer.
- boolean isInteger ()
  True if the value is integer and in the range of the standard integer.
- boolean isIntegerFrac ()
  True if the value is a fraction of two integers in the range of the standard integer.
- int signum ()
  The sign: 1 if the number is >0, 0 if ==0, -1 if <0.
- Vector< BigInteger > cfrac ()
  Terminating continued fractions.

Static Public Member Functions

- static Rational binomial (Rational n, BigInteger m)

  binomial (n choose m).
- static Rational binomial (Rational n, int m)

  binomial (n choose m).
- static Rational hankelSymb (Rational n, int k)

  Hankel's symbol (n,k)
- static BigInteger lcmDenom (final Rational[] vals)

  Common lcm of the denominators of a set of rational values.

Static Public Attributes

- static BigInteger MAX_INT = new BigInteger("2147483647")

  The maximum and minimum value of a standard Java integer, 2^31.
- static BigInteger MIN_INT = new BigInteger("-2147483648")
- static Rational ZERO = new Rational()

  The constant 0.
- static Rational HALF = new Rational(1,2)

  The constant 1/2.

Protected Member Functions

- void normalize ()

  Normalize to coprime numerator and denominator.

9.23.1 Detailed Description

Fractions (rational numbers).

They are divisions of two BigInteger numbers, reduced to coprime numerator and denominator.

Since

2006-06-25

Author

Richard J. Mathar

9.23.2 Constructor & Destructor Documentation

9.23.2.1 org.nevec.rjm.Rational.Rational ( )

Default ctor, which represents the zero.

Since

2007-11-17

9.23.2.2 org.nevec.rjm.Rational.Rational (BigInteger *a*, BigInteger *b*)

ctor from a numerator and denominator.

Parameters

a	the numerator.
b	the denominator.

9.23.2.3 org.nevec.rjm.Rational.Rational (BigInteger *a*)

ctor from a numerator.

Parameters

a	the BigInteger.

9.23.2.4 org.nevec.rjm.Rational.Rational (int *a,* int *b*)

ctor from a numerator and denominator.

Parameters

a	the numerator.
b	the denominator.

9.23.2.5 org.nevec.rjm.Rational.Rational (int *n*)

ctor from an integer.

Parameters

n	the integer to be represented by the new instance.

Since

2010-07-18

9.23.2.6 org.nevec.rjm.Rational.Rational (String *str*) throws NumberFormatException

ctor from a string representation.

Parameters

str	the string. This either has a slash in it, separating two integers, or, if there is no slash, is representing the numerator with implicit denominator equal to 1. Warning: this does not yet test for a denominator equal to zero

9.23.2.7 org.nevec.rjm.Rational.Rational (String *str,* int *radix*) throws NumberFormatException

ctor from a string representation in a specified base.

Parameters

str	the string. This either has a slash in it, separating two integers, or, if there is no slash, is just representing the numerator.
radix	the number base for numerator and denominator Warning: this does not yet test for a denominator equal to zero

9.23.2.8 org.nevec.rjm.Rational.Rational (Vector< BigInteger > *cfr*)

ctor from a terminating continued fraction.

Constructs the value of cfr[0]+1/(cfr[1]+1/(cfr[2]+...))).

Parameters

cfr	The coefficients cfr[0], cfr[1],... of the continued fraction. An exception is thrown if any of these is zero.

Since

2012-03-08

9.23.3 Member Function Documentation

9.23.3.1 Rational org.nevec.rjm.Rational.abs ( )

Absolute value.

Returns

The absolute (non-negative) value of this.

9.23.3.2 Rational org.nevec.rjm.Rational.add (Rational *val*)

Add another fraction.

Parameters

val	The number to be added

Returns

this+val.

9.23.3.3 Rational org.nevec.rjm.Rational.add (BigInteger *val*)

Add another integer.

Parameters

val	The number to be added

Returns

this+val.

9.23.3.4 Rational org.nevec.rjm.Rational.add (int *val*)

Add another integer.

Parameters

val	The number to be added

Returns

this+val.

Since

May 26 2010

9.23.3.5 BigDecimal org.nevec.rjm.Rational.BigDecimalValue (MathContext *mc*)

Return a representation as BigDecimal.

Parameters

mc	the mathematical context which determines precision, rounding mode etc

Returns

A representation as a BigDecimal floating point number.

Since

2008-10-26

9.23.3.6 static Rational org.nevec.rjm.Rational.binomial (Rational *n,* BigInteger *m*) [static]

binomial (n choose m).

Parameters

n	the numerator. Equals the size of the set to choose from.
m	the denominator. Equals the number of elements to select.

Returns

the binomial coefficient.

Since

2006-06-27

Author

Richard J. Mathar

9.23.3.7 static Rational org.nevec.rjm.Rational.binomial (Rational *n,* int *m*) [static]

binomial (n choose m).

Parameters

n	the numerator. Equals the size of the set to choose from.
m	the denominator. Equals the number of elements to select.

Returns

the binomial coefficient.

Since

2009-05-19

Author

Richard J. Mathar

9.23.3.8 BigInteger org.nevec.rjm.Rational.ceil ( )

ceil(): the nearest integer not smaller than this.

Returns

The integer rounded towards positive infinity.

Since

2010-05-26

9.23.3.9 Vector<BigInteger> org.nevec.rjm.Rational.cfrac ( )

Terminating continued fractions.

Returns

The list of a0, a1, a2,... in this =a0+1/(a1+1/(a2+1/(a3+...)))). If this here is zero, the list is empty.

Since

2012-03-09

9.23.3.10 Rational org.nevec.rjm.Rational.clone ( )

Create a copy.

Since

2008-11-07

9.23.3.11 int org.nevec.rjm.Rational.compareTo (final Rational *val*)

Compares the value of this with another constant.

Parameters

val	the other constant to compare with

Returns

-1, 0 or 1 if this number is numerically less than, equal to, or greater than val.

9.23.3.12 int org.nevec.rjm.Rational.compareTo (final BigInteger *val*)

Compares the value of this with another constant.

Parameters

val	the other constant to compare with

Returns

-1, 0 or 1 if this number is numerically less than, equal to, or greater than val.

9.23.3.13 BigInteger org.nevec.rjm.Rational.denom ( )

Get the denominator.

Returns

The denominator of the reduced fraction.

9.23.3.14 Rational org.nevec.rjm.Rational.divide (final Rational *val*)

Divide by another fraction.

Parameters

val	A second rational number.

Returns

The value of this/val

9.23.3.15 Rational org.nevec.rjm.Rational.divide (BigInteger *val*)

Divide by an integer.

Parameters

val	a second number.

Returns

the value of this/val

9.23.3.16 Rational org.nevec.rjm.Rational.divide (int *val*)

Divide by an integer.

Parameters

val	A second number.

Returns

The value of this/val

9.23.3.17 double org.nevec.rjm.Rational.doubleValue ( )

Return a double value representation.

Returns

The value with double precision.

Since

2008-10-26

9.23.3.18 float org.nevec.rjm.Rational.floatValue ( )

Return a float value representation.

Returns

The value with single precision.

Since

2009-08-06

9.23.3.19 BigInteger org.nevec.rjm.Rational.floor ( )

floor(): the nearest integer not greater than this.

Returns

The integer rounded towards negative infinity.

9.23.3.20 static Rational org.nevec.rjm.Rational.hankelSymb (Rational *n,* int *k*) [static]

Hankel's symbol (n,k)

Parameters

n	the first parameter.
k	the second parameter, greater or equal to 0.

Returns

Gamma(n+k+1/2)/k!/GAMMA(n-k+1/2)

Since

2010-07-18

Author

Richard J. Mathar

9.23.3.21 boolean org.nevec.rjm.Rational.isBigInteger ( )

True if the value is integer.

Equivalent to the indication whether a conversion to an integer can be exact.

Since

2010-05-26

9.23.3.22 boolean org.nevec.rjm.Rational.isInteger ( )

True if the value is integer and in the range of the standard integer.

Equivalent to the indication whether a conversion to an integer can be exact.

Since

2010-05-26

9.23.3.23 boolean org.nevec.rjm.Rational.isIntegerFrac ( )

True if the value is a fraction of two integers in the range of the standard integer.

Since

2010-05-26

9.23.3.24 static BigInteger org.nevec.rjm.Rational.lcmDenom (final Rational[] *vals*) [static]

Common lcm of the denominators of a set of rational values.

Parameters

vals	The list/set of the rational values.

Returns

LCM(denom of first, denom of second, ..,denom of last)

Since

2012-03-02

9.23.3.25 Rational org.nevec.rjm.Rational.max (final Rational *val*)

Compares the value of this with another constant.

Parameters

val	The other constant to compare with

Returns

The arithmetic maximum of this and val.

Since

2008-10-19

9.23.3.26 Rational org.nevec.rjm.Rational.min (final Rational *val*)

Compares the value of this with another constant.

Parameters

val	The other constant to compare with

Returns

The arithmetic minimum of this and val.

Since

2008-10-19

9.23.3.27 Rational org.nevec.rjm.Rational.multiply (final Rational *val*)

Multiply by another fraction.

Parameters

val	a second rational number.

Returns

the product of this with the val.

9.23.3.28 Rational org.nevec.rjm.Rational.multiply (final BigInteger *val*)

Multiply by a BigInteger.

Parameters

val	a second number.

Returns

the product of this with the value.

9.23.3.29 Rational org.nevec.rjm.Rational.multiply (final int *val*)

Multiply by an integer.

Parameters

val	a second number.

Returns

the product of this with the value.

9.23.3.30 Rational org.nevec.rjm.Rational.negate ( )

Compute the negative.

Returns

-this.

9.23.3.31 void org.nevec.rjm.Rational.normalize ( ) [protected]

Normalize to coprime numerator and denominator.

Also copy a negative sign of the denominator to the numerator.

Since

2008-10-19

9.23.3.32 BigInteger org.nevec.rjm.Rational.numer ( )

Get the numerator.

Returns

The numerator of the reduced fraction.

9.23.3.33 Rational org.nevec.rjm.Rational.Pochhammer (final BigInteger *n*)

Compute Pochhammer's symbol (this)_n.

Parameters

n	The number of product terms in the evaluation.

Returns

Gamma(this+n)/Gamma(this) = this∗(this+1)∗...∗(this+n-1).

Since

2008-10-25

9.23.3.34 Rational org.nevec.rjm.Rational.Pochhammer (int *n*)

Compute pochhammer's symbol (this)_n.

Parameters

n	The number of product terms in the evaluation.

Returns

Gamma(this+n)/GAMMA(this).

Since

2008-11-13

9.23.3.35 Rational org.nevec.rjm.Rational.pow (int *exponent*)

Power to an integer.

Parameters

exponent	the exponent.

Returns

this value raised to the power given by the exponent. If the exponent is 0, the value 1 is returned.

9.23.3.36 Rational org.nevec.rjm.Rational.pow (BigInteger *exponent*) throws NumberFormatException

Power to an integer.

Parameters

exponent	the exponent.

Returns

this value raised to the power given by the exponent. If the exponent is 0, the value 1 is returned.

Since

2009-05-18

9.23.3.37 Rational org.nevec.rjm.Rational.pow (Rational *exponent*) throws NumberFormatException

Raise to a rational power.

Parameters

exponent	The exponent.

Returns

This value raised to the power given by the exponent. If the exponent is 0, the value 1 is returned.

Since

2009-05-18

9.23.3.38 Rational org.nevec.rjm.Rational.root (BigInteger *r*) throws NumberFormatException

r-th root.

Parameters

r	the inverse of the exponent. 2 for the square root, 3 for the third root etc

Returns

this value raised to the inverse power given by the root argument, this^(1/r).

Since

2009-05-18

9.23.3.39 int org.nevec.rjm.Rational.signum ( )

The sign: 1 if the number is >0, 0 if ==0, -1 if <0.

Returns

the signum of the value.

Since

2010-05-26

9.23.3.40 Rational org.nevec.rjm.Rational.subtract (Rational *val*)

Subtract another fraction.

Parameters

val	the number to be subtracted from this

Returns

this - val.

9.23.3.41 Rational org.nevec.rjm.Rational.subtract (BigInteger *val*)

Subtract an integer.

Parameters

val	the number to be subtracted from this

Returns

this - val.

9.23.3.42 Rational org.nevec.rjm.Rational.subtract (int *val*)

Subtract an integer.

Parameters

val	the number to be subtracted from this

Returns

this - val.

9.23.3.43 String org.nevec.rjm.Rational.toFString (int *digits*)

Return a string in floating point format.

Parameters

digits	The precision (number of digits)

Returns

The human-readable version in base 10.

Since

2008-10-25

9.23.3.44 String org.nevec.rjm.Rational.toString ( )

Return a string in the format number/denom.

If the denominator equals 1, print just the numerator without a slash.

Returns

the human-readable version in base 10

9.23.3.45 BigInteger org.nevec.rjm.Rational.trunc ( )

Remove the fractional part.

Returns

The integer rounded towards zero.

9.23.4 Member Data Documentation

9.23.4.1 Rational org.nevec.rjm.Rational.HALF = new Rational(1,2) [static]

The constant 1/2.

Since

2010-05-25

9.23.4.2 BigInteger org.nevec.rjm.Rational.MAX_INT = new BigInteger("2147483647") [static]

The maximum and minimum value of a standard Java integer, 2^31.

Since

2009-05-18

9.23.4.3 BigInteger org.nevec.rjm.Rational.MIN_INT = new BigInteger("-2147483648") [static]

9.23.4.4 Rational org.nevec.rjm.Rational.ZERO = new Rational() [static]

The constant 0.

9.24 org.nevec.rjm.RationalPoly Class Reference

Ratio of two univariate polynomials with rational coefficients.

Public Member Functions

- RationalPoly (RatPoly num, RatPoly den)
  Ctor with explicit numerator and denominator polynomials.
- RationalPoly (BigIntegerPoly num, BigIntegerPoly den)
  Ctor with explicit numerator and denominator polynomials.
- RationalPoly (RatPoly num)
  Ctor with explicit numerator.
- RationalPoly minus (Rational oth)
  Subtract a constant.
- RationalPoly minus (BigInteger oth)
  Subtract a constant.
- RationalPoly inverse ()
  Build the inverse, 1 divided by this.

Static Public Member Functions

- static void main (String args[])

9.24.1 Detailed Description

Ratio of two univariate polynomials with rational coefficients.

Since

2012-03-08

Author

Richard J. Mathar

9.24.2 Constructor & Destructor Documentation

9.24.2.1 org.nevec.rjm.RationalPoly.RationalPoly (RatPoly *num,* RatPoly *den*)

Ctor with explicit numerator and denominator polynomials.

Parameters

num	The numerator.
den	The denominator.

9.24.2.2 org.nevec.rjm.RationalPoly.RationalPoly (**BigIntegerPoly** *num,* **BigIntegerPoly** *den*)

Ctor with explicit numerator and denominator polynomials.

Parameters

num	The numerator.
den	The denominator.

9.24.2.3 org.nevec.rjm.RationalPoly.RationalPoly (RatPoly *num*)

Ctor with explicit numerator.

The denominator is set to 1.

Parameters

num	The numerator.

9.24.3 Member Function Documentation

9.24.3.1 **RationalPoly** org.nevec.rjm.RationalPoly.inverse ( )

Build the inverse, 1 divided by this.

Returns

1/this.

9.24.3.2 static void org.nevec.rjm.RationalPoly.main (String *args[]*) [static]

9.24.3.3 **RationalPoly** org.nevec.rjm.RationalPoly.minus (**Rational** *oth*)

Subtract a constant.

Parameters

oth	The number to be subtracted

Returns

this-oth.

9.24.3.4 RationalPoly org.nevec.rjm.RationalPoly.minus (BigInteger *oth*)

Subtract a constant.

Parameters

oth	The number to be subtracted

Returns

this-oth.

9.25 org.nevec.rjm.Wigner3j Class Reference

Exact representations of Wigner 3jm and 3nj values of half-integer arguments.

Static Public Member Functions

- static void main (String args[])
  Test programs.
- static BigSurd wigner3jm (int j1, int j2, int j3, int m1, int m2, int m3)
  The Wigner 3jm symbol (j1,j2,j3,m1,m2,m3).
- static BigSurdVec wigner3j (String m1, String t1, String t2, String j)
  Wigner 3jn symbol.

Static Protected Member Functions

- static BigSurd wigner3jm (Rational j1, Rational j2, Rational j3, Rational m1, Rational m2, Rational m3)
  The Wigner 3jm symbol (j1,j2,j3,m1,m2,m3).

Static Private Member Functions

- static BigSurdVec wigner3j (final int[] tvec, final Rational[] J, final Rational[] M, final int[] triadidx)
  Wigner 3jn symbol.

9.25.1 Detailed Description

Exact representations of Wigner 3jm and 3nj values of half-integer arguments.

See Also

R. J. Mathar, Corrigendum to "Universal factorzation fo 3n-j (j>2) symbols ..[J. Phys. A: Math. Gen.37 (2004) 3259]"
R. J. Mathar, Symmetries in Wigner 18-j and 21-j Symbols

Since

2011-02-15

Author

Richard J. Mathar

9.25.2 Member Function Documentation

9.25.2.1 static void org.nevec.rjm.Wigner3j.main (String *args[]*) [static]

Test programs.

This supports three types of direct evaluations:

java -cp . org.nevec.rjm.Wigner3j 3jm 2j1+1 2j2+1 2j3+1 2m1+1 2m2+1 2m3+1

java -cp . org.nevec.rjm.Wigner3j 6j 2j1+1 2j2+2 .. 2j6+1

java -cp . org.nevec.rjm.Wigner3j 9j 2j1+1 2j2+2 .. 2j9+1

The first command line argument is one of the three tags which determine whether a 3jm, a 6j or a 9j symbol will be computed. The other arguments are 6 or 9 integer values, which are the physical (half-integer) values multplied by 2 and augmented by 1. The order of the 6 or 9 values is as reading the corresponding standard symbol as first row, then second row (and for the 9j symbol) third row.

Since

2011-02-15

Author

Richard J. Mathar

9.25.2.2 static BigSurdVec org.nevec.rjm.Wigner3j.wigner3j (String *m1,* String *t1,* String *t2,* String *j*) [static]

Wigner 3jn symbol.

For the 6j symbol, the input of the 3 lines is "1 2 3 1 5 6", "4 5 3 4 2 6" "2j1+1 2j2+1 2j3+1 2l1+1 2l2+1 2l3+1"

Parameters

m1	The information on the number of angular momenta.
t1	The list of one half of the triads, indexing j, whitespace separated
t2	The list of the second half of the triads, indexing j, whitespace separated
j	The list of the integer values of the angular momenta. They are actually the doubled j-values plus 1, whitespace separated. Only as many as announced by the m1 parameter are used; trailing numbers are ignored.

See Also

A. Bar-Shalom and M. Klapisch, NJGRAF..., Comp. Phys Comm. 50 (3) (1988) 375

Since

2011-02-13
2012-02-15 Upgraded return value to BigSurdVec

Author

Richard J. Mathar

9.25.2.3 static BigSurdVec org.nevec.rjm.Wigner3j.wigner3j (final int[] *tvec,* final Rational[] *J,* final Rational[] *M,* final int[] *triadidx*) [static],[private]

Wigner 3jn symbol.

Computes sum_{mi} (-1)^(j1-m1+j2-m2+...) triad(triadidx[0..2])∗triad(triadidx[3..5])∗... where each factor is a Wigner-3jm symbol with each sign of m_i occurring once at the corresponding l-value.

Parameters

triadidx	0-based indices into the list of J
J	The list of J-values
M	The list of M-values associated with the J. This contains null where the parameter has not yet been set by an outer loop.

Since

2011-02-13
2012-02-15 Upgraded to return BigSurdVec

Author

Richard J. Mathar

9.25.2.4 static BigSurd org.nevec.rjm.Wigner3j.wigner3jm (int *j1,* int *j2,* int *j3,* int *m1,* int *m2,* int *m3*) [static]

The Wigner 3jm symbol (j1,j2,j3,m1,m2,m3).

All arguments of the function are the actual parameters multiplied by 2, so they all allow an integer representation.

Parameters

j1	integer representing 2∗j1
j2	integer representing 2∗j2
j3	integer representing 2∗j3
m1	integer representing 2∗m1
m2	integer representing 2∗m2
m3	integer representing 2∗m3

Returns

The value of the symbol. Zero if any of the triangular inequalities is violated or some parameters are out of range.

Since

2011-02-13

Author

Richard J. Mathar

9.25.2.5 static BigSurd org.nevec.rjm.Wigner3j.wigner3jm (Rational *j1,* Rational *j2,* Rational *j3,* Rational *m1,* Rational *m2,* Rational *m3*) [static],[protected]

The Wigner 3jm symbol (j1,j2,j3,m1,m2,m3).

Warning: there is no check that each argument is indeed half-integer.

Parameters

j1	integer or half-integer j1
j2	integer or half-integer j2
j3	integer or half-integer j3
m1	integer or half-integer m1
m2	integer or half-integer m2
m3	integer or half-integer m3

Returns

The value of the symbol. Zero if any of the triangular inequalities is violated or some parameters are out of range.

Since

2011-02-13

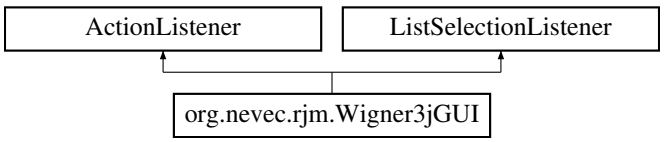
Author

Richard J. Mathar

9.26 org.nevec.rjm.Wigner3jGUI Class Reference

An interactive interface to the Wigner3j class.

Inheritance diagram for org.nevec.rjm.Wigner3jGUI:



Public Member Functions

- void init ()
- void compute ()
- void actionPerformed (ActionEvent e)

  Interpreter parser loop.
- void valueChanged (ListSelectionEvent e)

  Interpreter parser loop.

Static Public Member Functions

- static void main (String[] args)

  Main entry point.

9.26.1 Detailed Description

An interactive interface to the Wigner3j class.

The GUI allows to preselect one of the symbols if the number of j-terms is small (6j up to 15j), or to enter any other connectivity for the triads of j-values. The actual j-values are entered as integers (2j+1) and the computation of one value (in exact square root representation) is started manually.

Since

2011-02-15

9.26.2 Member Function Documentation

9.26.2.1 void org.nevec.rjm.Wigner3jGUI.actionPerformed (ActionEvent *e*)

Interpreter parser loop.

Parameters

e	the information on which button had been pressed in the GUI

Since

2011-02-15

9.26.2.2 void org.nevec.rjm.Wigner3jGUI.compute ( )

Since

2010-08-27

9.26.2.3 void org.nevec.rjm.Wigner3jGUI.init ( )

Since

2011-02-15

9.26.2.4 static void org.nevec.rjm.Wigner3jGUI.main (String[] *args*) [static]

Main entry point.

not taking any command line options:

java -jar Wigner3jGUI.jar

Since

2012-02-16

Author

Richard J. Mathar

9.26.2.5 void org.nevec.rjm.Wigner3jGUI.valueChanged (ListSelectionEvent *e*)

Interpreter parser loop.

Parameters

e	the information on which of the 3jn templates had been selected in the Menu

Since

2011-02-18

10 File Documentation

10.1 Bernoulli.java File Reference

Classes

- class org.nevec.rjm.Bernoulli
  Bernoulli numbers.

Packages

- package org.nevec.rjm

10.2 BigComplex.java File Reference

Classes

- class org.nevec.rjm.BigComplex
  Complex numbers with BigDecimal real and imaginary components.

Packages

- package org.nevec.rjm

10.3 BigDecimalMath.java File Reference

Classes

- class org.nevec.rjm.BigDecimalMath
  BigDecimal special functions.

Packages

- package org.nevec.rjm

10.4 BigDecimalPoly.java File Reference

Classes

- class org.nevec.rjm.BigDecimalPoly
  Polynomial with BigDecimal coefficients.

Packages

- package org.nevec.rjm

10.5 BigIntegerMath.java File Reference

Classes

- class org.nevec.rjm.BigIntegerMath
  BigInteger special functions and Number theory.

Packages

- package org.nevec.rjm

10.6 BigIntegerPoly.java File Reference

Classes

- class org.nevec.rjm.BigIntegerPoly
  Polynomial with integer coefficients.

Packages

- package org.nevec.rjm

10.7 BigSurd.java File Reference

Classes

- class org.nevec.rjm.BigSurd
  Square roots on the real line.

Packages

- package org.nevec.rjm

10.8 BigSurdVec.java File Reference

Classes

- class org.nevec.rjm.BigSurdVec
  A BigSurdVec represents an algebraic sum or differences of values which each term an instance of BigSurd.

Packages

- package org.nevec.rjm

10.9 Blas.java File Reference

Classes

- class **org.nevec.rjm.Blas**

  Basic Linear Algebra subroutines (BLAS)
- interface **org.nevec.rjm.Blas.Transform**

  The possible on-the-fly transformations of an input: normal, transposed or complex-conjugated.

Packages

- package org.nevec.rjm

10.10 Euler.java File Reference

Classes

- class org.nevec.rjm.Euler

  Euler numbers.

Packages

- package org.nevec.rjm

10.11 EulerPhi.java File Reference

Classes

- class org.nevec.rjm.EulerPhi

  Euler totient function.

Packages

- package org.nevec.rjm

10.12 Factorial.java File Reference

Classes

- class org.nevec.rjm.Factorial

  Factorials.

Packages

- package org.nevec.rjm

10.13 FI.java File Reference

Classes

- class **org.nevec.rjm.FIFunc**

  A function, which is a string (=function name) and a sequence of words to be executed.
- class **org.nevec.rjm.FILoop**

  A inner body of a loop between DO and LOOP or +LOOP This is mainly needed to save the text back to the DO even if the loop is defined outside a function, which means even if the pieces of the loop body are forwarded individually from the input line.
- class org.nevec.rjm.FI

  Applet of an integer calculator with Forth-alike syntax.

Packages

- package org.nevec.rjm

10.14 Harmonic.java File Reference

Classes

- class org.nevec.rjm.Harmonic

  Harmonic numbers.

Packages

- package org.nevec.rjm

10.15 Ifactor.java File Reference

Classes

- class org.nevec.rjm.Ifactor

  Factored integers.

Packages

- package org.nevec.rjm

10.16 linHRecGui.java File Reference

Classes

- class org.nevec.rjm.linHRecGui

Packages

- package org.nevec.rjm

10.17 oeis.java File Reference

Classes

- class **org.nevec.rjm.A000005**
  Number of divisors of the number.
- class **org.nevec.rjm.A000027**
  The integers.
- class **org.nevec.rjm.A000032**
  Lucas numbers.
- class **org.nevec.rjm.A000045**
  Fibonacci numbers.
- class **org.nevec.rjm.A000108**
  Catalan numbers.
- class **org.nevec.rjm.A000110**
  Bell numbers.
- class **org.nevec.rjm.A000120**
  Number of 1's in binary expansion.
- class **org.nevec.rjm.A000290**
  Squares.
- class **org.nevec.rjm.A000367**
  Numerators of Bernoulli numbers of even index.
- class **org.nevec.rjm.A000720**
  Pi, the number of primes less than or equal to n.
- class **org.nevec.rjm.A001006**
  Motzkin numbers.
- class **org.nevec.rjm.A001045**
  Jacobsthal sequence.
- class **org.nevec.rjm.A001065**
  Sum of proper divisors of n.
- class **org.nevec.rjm.A001147**
  Double factorials.
- class **org.nevec.rjm.A002110**
  Primorials.
- class **org.nevec.rjm.Cauchy2**
  Cauchy numbers of the second type.
- class **org.nevec.rjm.A005114**
  Untouchable numbers.
- class **org.nevec.rjm.A006318**
  Large Schroder numbers.
- class **org.nevec.rjm.A006954**
  Denominators of Bernoulli numbers.
- class **org.nevec.rjm.A007947**
  Largest square-free number dividing n.
- class **org.nevec.rjm.A008275**
  Stirling numbers of the 1st kind.
- class **org.nevec.rjm.A008683**
  Moebius function.
- class **org.nevec.rjm.A010060**
  Thue-Morse function .
- class **org.nevec.rjm.A048993**

Stirling numbers of the 2nd kind.

- class **org.nevec.rjm.A047994**

  Unitary Phi.

- class **org.nevec.rjm.A078923**

  Complement set to the untouchable numbers.

- class **org.nevec.rjm.A111278**

  Untouchable squares.

- class **org.nevec.rjm.A119829**

  Diagonal sum of number triangle A119828 .

Packages

- package org.nevec.rjm

10.18 PartitionsP.java File Reference

Classes

- class org.nevec.rjm.PartitionsP

  Number of partitions.

Packages

- package org.nevec.rjm

10.19 Prime.java File Reference

Classes

- class org.nevec.rjm.Prime

  Prime numbers.

Packages

- package org.nevec.rjm

10.20 Rational.java File Reference

Classes

- class org.nevec.rjm.Rational

  Fractions (rational numbers).

Packages

- package org.nevec.rjm

10.21 RationalPoly.java File Reference

Classes

- class org.nevec.rjm.RationalPoly

  Ratio of two univariate polynomials with rational coefficients.

Packages

- package org.nevec.rjm

10.22 RatPoly.java File Reference

Classes

- class **org.nevec.rjm.RatPoly**

  A uni-variater polynomial with rational coefficients.

Packages

- package org.nevec.rjm

10.23 Wigner3j.java File Reference

Classes

- class org.nevec.rjm.Wigner3j

  Exact representations of Wigner 3jm and 3nj values of half-integer arguments.

Packages

- package org.nevec.rjm

10.24 Wigner3jGUI.java File Reference

Classes

- class org.nevec.rjm.Wigner3jGUI

  An interactive interface to the Wigner3j class.

Packages

- package org.nevec.rjm

Index