Manual de usuario de la Calculadora multiprecisión v1.2

# Índice

[0. Índice 1](#_Toc62346463)

[1. Introducción 5](#_Toc62346464)

[1.1. Aspecto visual de la pantalla 6](#_Toc62346465)

[2. Inicio rápido 6](#_Toc62346466)

[2.1. Requisitos 6](#_Toc62346467)

[2.2. Inicio rápido 6](#_Toc62346468)

[2.3. Ejemplos 7](#_Toc62346469)

[3. Interfaz de usuario 8](#_Toc62346470)

[3.1. Pantalla principal 8](#_Toc62346471)

[3.2. Menú de la aplicación 10](#_Toc62346472)

[3.2.1. Menú archivo 10](#_Toc62346473)

[3.2.2. Menú vista 10](#_Toc62346474)

[3.2.3. Menú herramientas 10](#_Toc62346475)

[3.2.4. Menú ayuda 12](#_Toc62346476)

[3.3. Configuración 13](#_Toc62346477)

[3.3.1. Motor de cálculo 13](#_Toc62346478)

[3.3.2. Interfaz de usuario 14](#_Toc62346479)

[3.3.3. Parámetros configuración de idiomas de la aplicación 15](#_Toc62346480)

[3.3.4. Parámetros configuración de vista de la aplicación 16](#_Toc62346481)

[3.4. Ventana de autocompletación 17](#_Toc62346482)

[3.5. Acerca de … 18](#_Toc62346483)

[3.6. Licencia 19](#_Toc62346484)

[4. Lista de operaciones 20](#_Toc62346485)

[4.1. Palabras reservadas 20](#_Toc62346486)

[4.1.1. print 20](#_Toc62346487)

[4.1.2. Constantes 20](#_Toc62346488)

[4.1.2.1. e 20](#_Toc62346489)

[4.1.2.2. pi 20](#_Toc62346490)

[4.1.2.3. ln2 20](#_Toc62346491)

[4.1.3. Comandos de variables de usuario 21](#_Toc62346492)

[4.1.3.1. Asignación de variables 21](#_Toc62346493)

[4.1.3.2. clearvars 21](#_Toc62346494)

[4.1.3.3. erasevar 21](#_Toc62346495)

[4.1.4. Comandos y funciones de funciones de usuario 22](#_Toc62346496)

[4.1.4.1. Asignación de funciones 22](#_Toc62346497)

[4.1.4.2. Composición de funciones 23](#_Toc62346498)

[4.1.4.3. val 23](#_Toc62346499)

[4.1.4.4. subst 24](#_Toc62346500)

[4.1.4.5. derivative 24](#_Toc62346501)

[4.1.4.6. simplify 24](#_Toc62346502)

[4.1.4.7. eraseafunc 24](#_Toc62346503)

[4.1.4.8. clearfuncs 25](#_Toc62346504)

[4.1.5. Operadores 26](#_Toc62346505)

[4.1.5.1. Operador de suma (+) 26](#_Toc62346506)

[4.1.5.2. Operador de resta (-) 26](#_Toc62346507)

[4.1.5.3. Operador de multiplicación (\*) 26](#_Toc62346508)

[4.1.5.4. Operador de división (/) 26](#_Toc62346509)

[4.1.5.5. Operador de potencia (^) 26](#_Toc62346510)

[4.1.5.6. Paréntesis ( ( ) ) 27](#_Toc62346511)

[4.1.5.7. Precedencia de operadores 27](#_Toc62346512)

[4.1.6. Otros comandos 28](#_Toc62346513)

[4.1.6.1. exit 28](#_Toc62346514)

[4.1.7. Funciones matemáticas predefinidas 29](#_Toc62346515)

[4.1.7.1. abs 29](#_Toc62346516)

[4.1.7.2. absolute 29](#_Toc62346517)

[4.1.7.3. acos 29](#_Toc62346518)

[4.1.7.4. acosh 29](#_Toc62346519)

[4.1.7.5. add 29](#_Toc62346520)

[4.1.7.6. arccos 30](#_Toc62346521)

[4.1.7.7. arccosh 30](#_Toc62346522)

[4.1.7.8. arcsin 30](#_Toc62346523)

[4.1.7.9. arcsinh 30](#_Toc62346524)

[4.1.7.10. arctan 30](#_Toc62346525)

[4.1.7.11. arctanh 31](#_Toc62346526)

[4.1.7.12. asin 31](#_Toc62346527)

[4.1.7.13. asinh 31](#_Toc62346528)

[4.1.7.14. atan 31](#_Toc62346529)

[4.1.7.15. atanh 31](#_Toc62346530)

[4.1.7.16. ceil 32](#_Toc62346531)

[4.1.7.17. ceiling 32](#_Toc62346532)

[4.1.7.18. cos 32](#_Toc62346533)

[4.1.7.19. cosh 32](#_Toc62346534)

[4.1.7.20. divide 32](#_Toc62346535)

[4.1.7.21. down 33](#_Toc62346536)

[4.1.7.22. exp 33](#_Toc62346537)

[4.1.7.23. floor 33](#_Toc62346538)

[4.1.7.24. gcd 33](#_Toc62346539)

[4.1.7.25. ln 34](#_Toc62346540)

[4.1.7.26. log 34](#_Toc62346541)

[4.1.7.27. max 34](#_Toc62346542)

[4.1.7.28. min 34](#_Toc62346543)

[4.1.7.29. multiply 34](#_Toc62346544)

[4.1.7.30. power 35](#_Toc62346545)

[4.1.7.31. quotient 35](#_Toc62346546)

[4.1.7.32. root 35](#_Toc62346547)

[4.1.7.33. round 35](#_Toc62346548)

[4.1.7.34. sgn 35](#_Toc62346549)

[4.1.7.35. sin 35](#_Toc62346550)

[4.1.7.36. sinh 36](#_Toc62346551)

[4.1.7.37. sqrt 36](#_Toc62346552)

[4.1.7.38. subtract 36](#_Toc62346553)

[4.1.7.39. sum 36](#_Toc62346554)

[4.1.7.40. tan 36](#_Toc62346555)

[4.1.7.41. tanh 37](#_Toc62346556)

[4.1.7.42. up 37](#_Toc62346557)

[4.1.7.43. value 37](#_Toc62346558)

[4.2. Comando help 38](#_Toc62346559)

[4.3. Tabla resumen de palabras reservadas 39](#_Toc62346560)

[5. Invocación desde la interfaz de comandos 41](#_Toc62346561)

# Introducción

La calculadora multiprecisión es una aplicación que permite realizar cálculos con precisión configurable.

Realmente no es demasiado útil disponer de una precisión mayor de 15 dígitos decimales en los cálculos habituales, pero es una curiosidad y un reto programar una calculadora de este tipo.

Hace unos años programé la primera versión de esta calculadora multiprecisión y mirando hacia atrás, me parecía una lástima que se quedara tal y como estaba, con una arquitectura de clases muy mejorable.

Rescaté de mi pasado lejano el reto de la derivación algebraica, y me decidí a programar una nueva versión de la aplicación.

La funcionalidad anterior sigue funcionando, y lo que se ha hecho es añadir nuevas opciones para el usuario.

La funcionalidad que ofrecía la versión anterior es esta:

* Motor de cálculo
	+ Funciones de usuario, en una o múltiples variables.
	+ Variables de usuario.
	+ Cálculos numéricos con precisión variable.

Las mejoras que introduce la versión v1.1 son estas:

* Nuevas funcionalidades:
	+ Cálculo de la función derivada de manera algebraica.
	+ Máximo común divisor.
	+ Cálculo utilizando múltiples hilos en paralelo.
	+ Versión invocable desde la interfaz de comandos.
* Mejoras en la interfaz de usuario:
	+ Múltiples idiomas.
	+ Nivel de zoom configurable.
	+ Autocompletación.
	+ Ayuda mejorada.
	+ Formateo de texto.
	+ Deshacer / rehacer en el panel de texto de entrada
* Mejoras en el código:
	+ Mejor arquitectura de clases.
	+ Posibilidad de crecimiento de manera fácil y estructurada.

Las mejoras que introduce la versión v1.2 son estas:

* Nuevas funcionalidades:
	+ Consulta de nueva versión al servidor.

## Aspecto visual de la pantalla

Aspecto visual de la pantalla:



La pantalla tiene los siguientes componentes:

* Menú principal en la línea superior
* Línea inferior:
	+ Precisión. Permite configurar la precisión para los cálculos.
	+ Botón Cancelar. Permite cancelar los cálculos pendientes.
* Paneles de texto:
	+ Panel **1**. Panel de entrada, ahí deben escribirse los comandos a ejecutar.
	+ Panel **2**. Panel de salida. La aplicación escribe los resultados de las operaciones en este panel.
	+ Panel **3**. Panel de funciones y variables de usuario. En este panel se muestran las funciones y variables definidas por el usuario.

# Inicio rápido

## Requisitos

Para ejecutar la aplicación es necesario tener el entorno de java instalado (Java Runtime Environment), en la versión 8 o superior.

De esa manera, se podrá abrir el fichero .jar existente en la carpeta ./\_binary/

## Inicio rápido

Para comenzar a utilizar la aplicación, sigue los siguientes pasos:

* Abre la Calculadora multiprecisión, haciendo doble click en: el .jar de la aplicación: que está en .../\_binary/ multicalcu-gui-main-v1.2-SNAPSHOT-all.jar
* Elige la precisión de los cálculos (componente de texto: “**Precisión:**”)
* Ejecuta un comando (escribiendo en el panel de texto de abajo a la derecha):
	+ Recibe ayuda. P. ej. help + return
	+ Define una función. P.ej. f(x) = 2 \* x ^ 2 + return
	+ Escribe una operación numérica. P.ej. print( 2 \* 4 ^ 2 ) + return
	+ Calcula una función derivada. P.ej. g(x) = subst( derivative( x, 1, f(x) ) ) + return

## Ejemplos

* help

Muestra la ayuda de la aplicación en el idioma escogido.

* help functions

Muestra la ayuda de la palabra clave functions ( es la ayuda de las funciones ).

* print( 2 \* pi )

Muestra el resultado de la operación 2 \* pi con la precisión seleccionada

* a = 1.1

Asigna el valor 1.1 a la variable a, que podrá ser utilizada en cálculos numéricos de expresiones, pero no dentro de la definición de funciones.

* fun(A, f, alpha, t ) = A \* cos( 2 \* pi \* f \* t + alpha )

Define la función fun en función de cuatro variables.

* g(f, t ) = fun( 2.7, f, pi / 2, t )

Define la función g en función de fun, pero sin substituir la expresión.

Es decir, siempre que se calcule un valor de la función g, se tomará la definición de fun, y se realizará la composición de variables definida.

Esto quiere decir, que si f cambia tras haber definido g, g también queda afectada.

* h(f, f2, t) = subst( fun( 2.7 \* sin( 2 \* pi \* f2 \* t + pi / 3), f, pi/2, t ) )

Define la función h en función de fun, realizando la composición de funciones y tomando la expresión resultante.

Esto quiere decir que la relación entre h y fun es sólo en el momento de definición de h.

Si fun cambia tras la definción de h, h quedará inalterada, porque la expresión ya se había substituido ( gracias a la función subst ).

* print( h( 10, 0.26, 3.7 ) )

Muestra el resultado de la función h para sus valores de entrada.

* print( 10 \* log( 10, ( 25.2 \* h( 10, 0.26, 3.7 ) ) ^ 2 ) )

Muestra el resultado de la operación. Para más detalle, consultar el apartado: ***4.1.7.26-log***.

* print( 10 \* arccos( tanh( 3 \* atan( 500 / pi \* 3.6 ) ) ) + sqrt( exp( 5.3 ) ^ ( 30 \* sin( 1 / 30.7 ) ) ) / ( 7.3 + sin( 27 + pi ) + sin( 27 ) ) )

Calcula el resultado numérico de una expresión compleja.

* i(f, f2, t) = subst( derivative( t, 2, h(f, f2, t ) ) )

Calcula la derivada parcial de segundo orden de h en función de t y la asigna a la función i.

# Interfaz de usuario

En este capítulo verás cómo usar los paneles de la ventana principal.

## Pantalla principal

Al arrancar la aplicación gráfica, aparece una pantalla como la siguiente:



La pantalla tiene los siguientes componentes:

* Menú principal en la línea superior
* Línea inferior:
	+ Precisión. Permite configurar la precisión para los cálculos.
	+ Botón Cancelar. Permite cancelar los cálculos pendientes.
* Paneles de texto:
	+ Panel **1**. Panel de entrada, ahí deben escribirse los comandos a ejecutar.
	+ Panel **2**. Panel de salida. La aplicación escribe los resultados de las operaciones en este panel.
	+ Panel **3**. Panel de funciones y variables de usuario. En este panel se muestran las funciones y variables definidas por el usuario.

Ejemplo de ventana en un uso normal:



## Menú de la aplicación

El menú es la barra de opciones que está en la línea de arriba de la ventana.

Tiene diferentes opciones que veremos a continuación.

### Menú archivo

El menú Archivo tiene este aspecto:



Este menú tiene una opción:

* Salir. Permite salir de la aplicación.

### Menú vista

El menú Vista tiene este aspecto:



Este menú sólo tiene una opción:

* Zoom. Permite escoger el tamaño de la parte visual de la aplicación, con un porcentaje que indica el tamaño de los componentes en comparación con la visualización normal de las ventanas (100%).

### Menú herramientas

El menú Herramientas tiene este aspecto:



Este menú tiene dos opciones:

* Idioma. Al abrir este submenú aparecen los idiomas disponibles. Sirve para poder cambiar el idioma de la aplicación.

Por defecto hay disponibles los siguientes idiomas:

* "EN". Inglés
* "ES". Castellano
* "CAT". Catalán
* Configuración. Cuando se pulsa este elemento de menú, aparece un formulario en el que se permite modificar los parámetros de configuración generales de la aplicación. Ver punto: ***3.3-Configuración***

### Menú ayuda

El menú Ayuda tiene este aspecto:



Este menú tiene tres opciones:

* Ayuda. Abre este archivo de ayuda.
* Acerca de. Cuando se pulsa este elemento de menú, aparece un formulario con los datos de la aplicación, los agradecimientos y los datos de contacto. Ver apartado: **3.5-Acerca de …**
* Licencia. Muestra la licencia que se aceptó la primera vez que se ejecutó la aplicación. Ver apartado: ***3.6-Licencia***

## Configuración

Este formulario permite modificar los parámetros de configuración generales de la aplicación.

El formulario tiene varias pestañas que detallaremos a continuación.

### Motor de cálculo

La pestaña tiene el siguiente aspecto:



Los parámetros que pueden configurarse son los siguientes:

* Precisión para los cálculos. Es la precisión en la que se mostrarán los cálculos realizados. Internamente los cálculos se realizan con una precisión ligeramente mayor, para intentar evitar inexactitudes en el resultado mostrado (cosa que no siempre se consigue).
* Máximo número de hilos a usar para los cálculos. Indica el número máximo de hilos que se va a utilizar para realizar los cálculos.

Se aconseja utilizar un número igual al número de hilos simultáneos soportados por el procesador menos uno.

Por ejemplo, si tienes un i7 con 4 núcleos y dos hilos por núcleo, el número aconsejado sería 7 (a menos que quieras reservar la capacidad de proceso para otros programas que se estén ejecutando simultáneamente).

### Interfaz de usuario

La pestaña tiene el siguiente aspecto:



Los parámetros que pueden configurarse son los siguientes:

* Mostrar autocompletación de comandos.

Cuando se marca esta opción, a medida que se va escribiendo en el panel de texto de entrada, aparece una ventana con las opciones de autocompletado.

Cuando se muestra esta ventana y hay opciones, se puede mover con los cursores arriba y abajo, para elegir otra posible opción, o continuar tecleando (se irá cerrando la búsqueda). Si pulsas return mientras esta ventana tiene opciones posibles, se escribe en el panel de texto la opción seleccionada y se cierra esa ventana.

También es posible elegir mediante el puntero del ratón la opción de autocompletación que desees.

* Mostrar ayuda del parámetro actual.

Cuando se marca esta opción, a medida que se va escribiendo en el panel de texto de entrada, aparece una ventana con el parámetro actual que estamos rellenando.

Esto es útil para saber, por ejemplo, en una función cual es el parámetro que estamos rellenando.

### Parámetros configuración de idiomas de la aplicación

La pestaña tiene el siguiente aspecto:



Los parámetros que pueden configurarse son los siguientes:

* Idioma. Es el idioma de los textos que escribe la aplicación.

Los idiomas que pueden escogerse son:

* EN. Inglés
* ES. Español
* CAT. Catalán
* Locale de idioma. Es el locale de Java que va a utilizar la aplicación para este idioma.

La aplicación lo utiliza para convertir números a cadenas numéricas formateadas.

* Idioma adicional. Es un nuevo idioma adicional que aparece a la hora de poder cambiar de idioma.

Para el nuevo idioma, tendrás que seleccionar el locale de Java a utilizar.

Si quieres poner un idioma del que no existen textos en la aplicación, puedes añadir tu idioma traduciendo los ficheros que hay dentro del directorio que se crea cuando pulsas el botón de ”Añadir idioma”.

Los ficheros a traducir se copian al directorio indicado en: "Directorio de idioma adicional"

El formato de estos ficheros de textos es el de properties de java.

Para los que no conozcáis este formato, os diré que un fichero properties, tiene un título, y tras este, un número variable de etiquetas con su valor, parecido a lo que sigue:

# TITULO

# xxxxxxxxxx

ETIQUETA1=texto 1

ETIQUETA2=texto 2

...

Las etiquetas deben permanecer sin modificar, y los textos deben modificarse dependiendo de la traducción en el idioma escogido.

Además, también hay algunos archivos en formato RTF, que tendrás que traducir con un editor de RTF (un editor típico para ese formato es el Word de Office).

Si creas la traducción para un idioma adicional distinto a los que hay disponibles en la aplicación, si quieres puedes enviármelo (frojasg1@hotmail.com) y lo incluiré en las siguientes versiones de la aplicación.

### Parámetros configuración de vista de la aplicación

La pestaña tiene el siguiente aspecto:



* Tamaño de las ventanas de la aplicación. Este parámetro permite controlar el aspecto de las ventanas que aparecen en la aplicación, existiendo la posibilidad de escoger el tamaño pequeño, el normal, o bien el tamaño grande.

## Ventana de autocompletación

La ventana de autocompletación es una ventana de ayuda que aparece a medida que se va escribiendo, y siempre que esté configurado para mostrarse.

La idea se ha sacado de la funcionalidad que tienen muchos Entornos integrados de desarrollo, en los que a medida que vas escribiendo, te aparecen las opciones, que te refrescan como es la sintaxis de lo que quieres escribir, así como cuales son los argumentos que toman en caso de que la opción sea una función.

Mientras se esté mostrando esa ventana con opciones (panel superior), podrás moverte por esas opciones con los cursores, o bien pulsando sobre la opción deseada con el ratón.

La manera de elegir una opción, es pulsando con el puntero del ratón encima de ella, o pulsando return. Entonces se escribirá la opción completa en el panel de texto de entrada y desaparecerá esta ventana.

La ventana también desaparece cuando el foco sale de la propia ventana, o de la ventana principal, y también cuando se minimiza la ventana principal.

Este es un ejemplo de pantalla de autocompletación:



En este caso, están activos los dos parámetros de configuración.

La ventana de autocompletación tiene dos paneles

* Panel superior: indica las opciones de autocompletación.
* Panel inferior, indica el parámetro actual en el que nos encontramos escribiendo.

## Acerca de …

La opción Acerca de …, muestra una ventana con un resumen de las novedades en esta versión. También se incluyen los agradecimientos.

Tiene un aspecto como este:



## Licencia

La opción Licencia, dentro del menú Ayuda, permite mostrar la licencia que se aceptó la primera vez que se ejecutó la aplicación.

Tiene un aspecto como este:



# Lista de operaciones

La calculadora multiprecisión permite hacer cálculos en precisión configurable, así como realizar algunas operaciones con funciones y variables.

Todo ello se realiza mediante la introducción de instrucciones a través del teclado.

Esas instrucciones están formadas por distintas palabras reservadas y parámetros.

##  Palabras reservadas

Las palabras reservadas son palabras o signos que tienen un significado concreto para el analizador de expresiones.

Muchas de esas palabras reservadas son comandos o funciones, aunque hay de otros tipos.

En este apartado daremos una vuelta por todas esas palabras.

### print

El comando print, permite mostrar el resultado numérico de la expresión que tiene como argumento.

Formato: **print**( **expresión** )

Ejemplo: print( 2 + 2 )

Resultado: 4

### Constantes

Las constantes son elementos como las variables, pero que tienen un valor fijo calculado automáticamente por la aplicación.

### e

Número de Euler.

Resultado: **2.718281828459045235360287471352662**…

### pi

Constante pi.

Resultado: **3.141592653589793238462643383279502**…

### ln2

Logaritmo neperiano de 2.

Resultado: **0.6931471805599453094172321214581**…

### Comandos de variables de usuario

Este tipo de comandos permite manejar la definición/borrado de variables con valor numérico concreto.

### Asignación de variables

Permite asignar un valor numérico a una variable.

Formato: **nombreVariable** = expresión numérica

Ejemplo: aa = 2 + 2

Resultado: Se asigna el valor 4 a la variable aa.

### clearvars

Borra todas las variables de usuario.

Formato: **clearvars**

Ejemplo: clearvars

Resultado: borra todas las variables.

### erasevar

Borra la variable de usuario indicada.

Formato: **erasevar nombreVariable**

Ejemplo: **erasevar aa**

Resultado: borra la variable aa.

### Comandos y funciones de funciones de usuario

Este tipo de comandos permite manejar la definición/borrado de funciones y hay algunos de ellos que no son realmente comandos, sino que son funciones que tienen que formar parte de una expresión y por sí mismos no forman ninguna instrucción completa.

### Asignación de funciones

Permite guardar una función en la lista de funciones de usuario.

Formato: **nombreFunción( variable1, …, variableN )** = expresión( variable1, …, variableN )

Ejemplo: **fun**( x, y, z ) = 2 \* x ^ 2 \* y ^ 2 \* z ^ 2

Resultado: Se asigna la expresión en función de tres variables al nombre de función **fun**.

Notas: La expresión de la función, permite definir relaciones entre funciones, es decir, podríamos definir la función: fun2(x, y) = fun( x, y, x\*y )

 En ese caso, si queremos calcular el valor de fun2( 1, 1 ), el motor de cálculo cogerá la función fun(x, y, z), y creará las operaciones dinámicamente.

 La expresión de fun puede cambiar tras la definición de fun2, pero en la definición de fun2 no hemos puesto ninguna expresión explícitamente, así que se resolvería de manera satisfactoria en el momento de la ejecución, cogiendo la definición de fun(x, y, z) más nueva.

 Si queremos que fun2 tenga una expresión concreta que no dependa de la definición de otras funciones, tendremos que utilizar el comando (función) subst, que hace que se substituya la expresión que toma como parámetro en el momento de la invocación.

 Así por ejemplo, si definimos: fun2(x, y) = subst( fun( x, y , x\*y ) ), directamente obtendremos una expresión concreta para fun2, y aunque después cambie la definición de fun, no notaríamos ningún cambio en la definición de fun2.

 El ese caso, el resultado sería directamente:

 fun2(x, y) = 2 \* x ^ 2 \* y ^ 2 \* ( x \* y ) ^ 2

 y la expresión de fun2 ya no tendría relación con la expresión de fun.

 Algo similar ocurre con el comando (función) derivative.

 Este comando permite obtener la función derivada de la expresión que toma como argumento.

 Así pues, podemos definir una función tomando como entrada otras funciones o incluso sus funciones derivadas, a modo de la expresión de una ecuación diferencial, aunque en este caso, la parte izquierda de la expresión, sería una función simple (no puede estar derivada).

 (La aplicación no resuelve ecuaciones diferenciales!!).

 Por ejemplo, para ilustrar lo que os comento, podríamos definir:

 fun3(x, y, z) = derivative( x, 2, fun( x, y, z ) ) – f(x, y, z) \* derivative( x, 1, fun( 2\*x, x\*y, x\*z ) )

 La expresión matemática de esa función sería esta:

 $fun3\left(x, y, z\right)= \frac{δ^{2}fun\left(x,y,z\right)}{δ^{2}x}-fun\left(x,y,z\right)\* \frac{δfun\left(2\*x, x\*y,x\*z\right)}{δx}$

Sin embargo, si lo que deseamos es concretar la expresión de fun2 en el tiempo de la definición, podemos poner:

fun3(x, y, z)= subst( derivative( x, 2, fun(x, y, z ) ) – subst( f(x, y, z ) ) \* subst( derivative( x, 1, fun( 2\*x, x\*y, x\*z ) ) )

En ese caso, el resultado obtenido para la definición de la función, sería este:

fun3( x, y, z ) = 4 \* y ^ 2 \* z ^ 2 – 2 \* x ^ 2 \* y ^ 2 \* z ^ 2 \* d fun( 2\*x, x\*y, x\*z ) / dx =

4 \* y ^ 2 \* z ^ 2 – 2 \* x ^ 2 \* y ^ 2 \* z ^ 2 \* d ( 2 \* ( 2 \* x ) ^ 2 \* ( x \* y ) ^ 2 \* ( x \* z ) ^ 2 ) / dx =

4 \* y ^ 2 \* z ^ 2 – 2 \* x ^ 2 \* y ^ 2 \* z ^ 2 \* 48 \* x ^ 5 \* y ^ 2 \* z ^ 2

En este caso, la expresión de fun3, se habría obtenido a partir de la definición actual de fun, pero ya no tendría dependencia con fun.

### Composición de funciones

Esta posibilidad permite calcular una expresión substituyendo las variables de una función por expresiones.

Formato: **nombreFunción( expresion1, …, expresionN )**

Ejemplo: **fun( 2\*x, x\*y, x\*z )**

Resultado: Si fun( x, y, z ) = 2 \* x ^ 2 \* y ^ 2 \* z ^ 2

 Entonces: fun( 2\*x, x\*y, x\*z ) = 2 \* ( 2 \* x ) ^ 2 \* ( x \* y ) ^ 2 \* ( x \* z ) ^ 2

### val

Este comando (que en realidad es una función), permite substituir una expresión que da como resultado un número, por ese número.

Esto es útil, porque, en caso de que estemos definiendo una función, podría ser útil substituir una expresión numérica compleja, por su resultado (para que se realicen menos operaciones cada vez que se calcule un valor de la función).

La parte negativa, es que esta operación está ligada a la precisión que tengamos en el momento de utilizarla, y si luego cambiamos de precisión, el valor calculado, no se recalcula.

Formato: **val( expresión )**

Ejemplo: f( x ) = **val( tan( pi / 4 ) / 2 )** \* x ^ 2

Resultado: f( x ) = 0.5 \* x ^ 2

### subst

Este comando (que en realidad es una función), permite substituir una composición de funciones o un cálculo de derivada por sus expresiones concretas.

Si no se usa esta función, el resultado quedará expresado en base a las funciones referenciadas en la composición o cálculo de derivada.

( Para más detalle, ver el apartado Notas, del punto: ***4.1.4.1-Asignación de funciones***)

Formato: **subst( composición de función ) ó**

 **subst( cálculo de función derivada )**

Ejemplo: f( x ) = **subst( g( 2 \* x ) )**

Resultado: si g(x) = 2 \* x entonces f( x ) = 2 \* 2 \* x

### derivative

Este comando (que en realidad es una función), permite calcular la derivada n-ésima sobre una de las variables de una función que puede estar definida en función de múltiples variables.

De momento no se ha implementado la posibilidad de calcular derivadas parciales cruzadas.

Si esta función no se utiliza como argumento de una función subst, entonces el resultado quedará expresado en base a las funciones referenciadas en el cálculo de derivada.

( Para más detalle, ver el apartado Notas, del punto: ***4.1.4.1-Asignación de funciones***)

Formato: **derivative( variable, orden, expresión )**

Ejemplo: g( x ) = **derivative( x, 1, f(x) )**

Resultado: si f( x )= x ^ 2 entonces g( x ) = 2 \* x

### simplify

Este comando (que en realidad es una función), permite simplificar una expresión. No substituye funciones, incluso aunque sean argumento de un comando subst.

Se creó con propósito de depuración, ya que esta función de simplificación se invoca cada vez que se calcula una derivada con el comando derivative.

Formato: **simplify( expresión )**

Ejemplo: f( x ) = **simplify( x ^ 2 / ( 2 \* x ) - 3 + 1 / 2 )**

Resultado: f( x )= x / 2 - 5 / 2

### eraseafunc

Este comando permite borrar una función de usuario.

Formato: **erasefunc( nombreFuncion )**

Ejemplo: erasefunc( f )

Resultado: borra la función de usuario f

### clearfuncs

Este comando permite borrar todas las funciones de usuario.

Formato: **clearfuncs**

Ejemplo: clearfuncs

Resultado: borra todas las funciones de usuario

### Operadores

Los operadores son cadenas, formadas normalmente por un solo carácter, que permiten concatenar expresiones.

### Operador de suma (+)

Suma dos números.

Formato: **sumando1 + sumando2**

Ejemplo: **1.01 + 3.45**

Resultado: 4.46

### Operador de resta (-)

Resta dos números.

Formato: **minuendo + sustraendo**

Ejemplo: **7.25 - 3.743**

Resultado: 3.507

### Operador de multiplicación (\*)

Multiplica dos números.

Formato: **factor1 \* factor2**

Ejemplo: **7.25 \* 3**

Resultado: 21.75

### Operador de división (/)

Divide dos números.

Formato: **dividendo / divisor**

Ejemplo: **16 / 4**

Resultado: 4

### Operador de potencia (^)

Eleva un número a otro.

Formato: **base ^ exponente**

Ejemplo: **16 / 4**

Resultado: 4

### Paréntesis ( ( ) )

Da prioridad a la operación entre paréntesis.

Formato: **( expresión )**

Ejemplo: **( 3 + 4 ) \* ( 2 - 7 )**

Resultado: 7 \* ( -5 )

### Precedencia de operadores

Los operadores se evaluarán en el siguiente orden:

1. Las expresiones entre paréntesis ( ... )
2. El operador de potencia ^
3. La multiplicación y división: \*, /
4. La suma y resta: +, -

### Otros comandos

Otros comandos, tiene que ver con comandos implementados que no tienen nada que ver con el motor de cálculo.

### exit

Salir de la aplicación.

Formato: **exit**

Ejemplo: **exit**

Resultado: Se sale de la aplicación.

### Funciones matemáticas predefinidas

Son funciones matemáticas que existen sin que el usuario tenga que definirlas.

Producen un resultado en función de los parámetros de entrada.

### abs

Calcula el valor absoluto del argumento.

Formato: **abs( argumento )**

Ejemplo: **abs( -2.35 )**

Resultado: 2.35

### absolute

Calcula el valor absoluto del argumento.

Formato: **absolute( argumento )**

Ejemplo: **absolute( -2.35 )**

Resultado: 2.35

### acos

Calcula el arco coseno del argumento, dando el resultado en radianes.

Formato: **acos( argumento )**

Ejemplo: **acos( 1 )**

Resultado: 0

### acosh

Calcula el arco coseno hiperbólico del argumento.

Formato: **acosh( argumento )**

Ejemplo: **acosh( 1 )**

Resultado: 0

### add

Calcula la suma de los dos sumandos.

Formato: **add( sumando1, sumando2 )**

Ejemplo: **add( 2, 3 )**

Resultado: 5

### arccos

Calcula el arco coseno del argumento, dando el resultado en radianes.

Formato: **arccos( argumento )**

Ejemplo: **arccos( 1 )**

Resultado: 0

### arccosh

Calcula el arco coseno hiperbólico del argumento.

Formato: **arccosh( argumento )**

Ejemplo: **arccosh( 1 )**

Resultado: 0

### arcsin

Calcula el arco seno del argumento, dando el resultado en radianes.

Formato: **arcsin( argumento )**

Ejemplo: **arcsin( 0 )**

Resultado: 0

### arcsinh

Calcula el arco seno hiperbólico del argumento.

Formato: **arcsinh( argumento )**

Ejemplo: **arcsinh( 0 )**

Resultado: 0

### arctan

Calcula el arco tangente del argumento, dando el resultado en radianes.

Formato: **arctan( argumento )**

Ejemplo: **arctan( 0 )**

Resultado: 0

### arctanh

Calcula el arco tangente hiperbólica del argumento.

Formato: **arctanh( argumento )**

Ejemplo: **arctanh( 0 )**

Resultado: 0

### asin

Calcula el arco seno del argumento, dando el resultado en radianes.

Formato: **asin( argumento )**

Ejemplo: **asin( 0 )**

Resultado: 0

### asinh

Calcula el arco seno hiperbólico del argumento.

Formato: **asinh( argumento )**

Ejemplo: **asinh( 0 )**

Resultado: 0

### atan

Calcula el arco tangente del argumento, dando el resultado en radianes.

Formato: **atan( argumento )**

Ejemplo: **atan( 0 )**

Resultado: 0

### atanh

Calcula el arco tangente hiperbólica del argumento.

Formato: **atanh( argumento )**

Ejemplo: **atanh( 0 )**

Resultado: 0

### ceil

Calcula el siguiente número con n decimales mayor o igual al argumento (dirección más infinito).

Formato: **ceil( n, argumento )**

Ejemplo: **ceil( 0, 1.01 )**

Resultado: 2

Ejemplo: **ceil( 0, -1.01 )**

Resultado: -1

### ceiling

Calcula el siguiente número con n decimales mayor o igual al argumento (dirección más infinito).

Formato: **ceiling( n, argumento )**

Ejemplo: **ceiling( 0, 1.01 )**

Resultado: 2

Ejemplo: **ceiling( 0, -1.01 )**

Resultado: -1

### cos

Calcula el coseno del argumento en radianes.

Formato: **cos( argumento )**

Ejemplo: **cos( 0 )**

Resultado: 1

### cosh

Calcula el coseno hiperbólico del argumento.

Formato: **cosh( argumento )**

Ejemplo: **cosh( 0 )**

Resultado: 1

### divide

Calcula la división de los argumentos.

Formato: **divide( dividendo, divisor )**

Ejemplo: **divide( 16, 4 )**

Resultado: 4

### down

Calcula el siguiente número con n decimales con menor o igual valor absoluto que el argumento (dirección a 0).

Formato: **down( n, argumento )**

Ejemplo: **down( 0, 1.01 )**

Resultado: 1

Ejemplo: **down( 0, -1.01 )**

Resultado: -1

### exp

Exponencial. Calcula la potencia del número e elevado al argumento.

Formato: **exp( argumento )**

Ejemplo: **exp( 0 )**

Resultado: 1

### floor

Calcula el siguiente número con n decimales menor o igual que el argumento (dirección a menos infinito)

Formato: **floor( n, argumento )**

Ejemplo: **floor( 0, 1.01 )**

Resultado: 1

Ejemplo: **floor( 0, -1.01 )**

Resultado: -2

### gcd

Calcula el máximo común divisor de los argumentos.

Formato: **gcd( argumento1, ..., argumentoN )**

Ejemplo: **gcd( 26, 39 )**

Resultado: 13

### ln

Logaritmo neperiano.

Formato: **ln( argumento )**

Ejemplo: **ln( 1 )**

Resultado: 0

### log

Calcula el logaritmo del argumento en la base indicada.

Formato: **log( base, argumento )**

Ejemplo: **log( 10, 100 )**

Resultado: 2

### max

Calcula el máximo de los argumentos.

Formato: **max( argumento1, ..., argumentoN )**

Ejemplo: **max( 7, 16, 5 )**

Resultado: 16

### min

Calcula el mínimo de los argumentos.

Formato: **min( argumento1, ..., argumentoN )**

Ejemplo: **min( 1, 7, -1 )**

Resultado: -1

### multiply

Calcula el producto de los factores.

Formato: **multiply( factor1, factor2 )**

Ejemplo: **multiply( 1, 7 )**

Resultado: 7

### power

Potencia. Calcula la potencia: base elevado a exponente.

Formato: **power( base, exponente )**

Ejemplo: **power( 3, 4 )**

Resultado: 81

### quotient

División. Calcula la división: dividendo dividido por divisor.

Formato: **quotient( dividendo, divisor )**

Ejemplo: **quotient( 16, 4 )**

Resultado: 4

### root

Calcula la raíz n-ésima del argumento.

Formato: **root( n, argumento )**

Ejemplo: **root( 4, 81 )**

Resultado: 3

### round

Calcula el número más cercano al argumento con n decimales.

Formato: **round( n, argumento )**

Ejemplo: **round( 2, 0.513 )**

Resultado: 0.51

### sgn

Calcula el signo del argumento (devuelve -1 si es negativo, 0 si es 0 y 1 si es positivo).

Formato: **sgn( argumento )**

Ejemplo: **sgn( -0.1 )**

Resultado: -1

### sin

Calcula el seno del argumento en radianes.

Formato: **sin( argumento )**

Ejemplo: **sin( pi / 2 )**

Resultado: 1

### sinh

Calcula el seno hiperbólico del argumento.

Formato: **sinh( argumento )**

Ejemplo: **sinh( 0 )**

Resultado: 0

### sqrt

Calcula la raíz cuadrada del argumento.

Formato: **sqrt( argumento )**

Ejemplo: **sqrt( 9 )**

Resultado: 3

### subtract

Resta. Calcula la resta: minuendo menos sustraendo.

Formato: **subtract( minuendo, sustraendo )**

Ejemplo: **subtract( 3, -2 )**

Resultado: 5

### sum

Calcula la suma de los dos sumandos.

Formato: **sum( sumando1, sumando2 )**

Ejemplo: **sum( 2, 3 )**

Resultado: 5

### tan

Calcula la tangente del argumento en radianes.

Formato: **tan( argumento )**

Ejemplo: **tan( pi / 4 )**

Resultado: 1

### tanh

Calcula la tangente hiperbólica del argumento.

Formato: **tanh( argumento )**

Ejemplo: **tanh( 0 )**

Resultado: 0

### up

Calcula el siguiente número con n decimales con un valor absoluto mayor o igual que el argumento (dirección a menos infinito si el argumento es negativo, o a más infinito si el argumento es positivo).

Formato: **up( n, argumento )**

Ejemplo: **up( 1, 1.01 )**

Resultado: 1.1

### value

Calcula el valor numérico del argumento.

Formato: **value( argumento )**

Ejemplo: **value(** sqrt(2)/ 2 ^ (1/2) **)**

Resultado: 1

##  Comando help

El comando help, muestra la ayuda de la aplicación, cuyo resumen se encuentra en la tabla detallada en el siguiente apartado. Referencia-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Formato: **help** nombre

Ejemplo: help functions

Resultado: Muestra la ayuda de las funciones, junto con un resumen de cada una de ellas.

La ayuda está organizada en apartados de manera jerárquica.

Si el comando help no toma argumentos, se muestra la ayuda global de la aplicación, que es un resumen de cada uno de los apartados que tienen ayuda.

Si el comando help toma como argumento un nombre, se muestra un texto explicativo del tag nombre junto a un resumen de cada uno de los contenidos derivados de él, cada contenido derivado a su vez con un **nombre** en color azul y negrita, que permite obtener ayuda más detallada de cada uno de esos tags, simplemente haciendo: help **nombre**

Al usar el comando help, normalmente aparecen nuevos contenidos derivados por los que se puede navegar.

Si te mueves con el puntero del ratón en el panel de salida sobre los nombres derivados (**nombre**), verás que cuando el puntero está encima de un **nombre** “hijo”, el puntero del ratón muestra el icono del tipo “seguir link”, y si en ese momento pulsas el botón izquierdo del ratón, directamente obtienes ayuda de ese tag, como si escribieras: help **nombre** + return.

Esta característica facilita mucho la navegación por la ayuda que de otra forma sería muy costosa.

Los nombres posibles que reconoce el comando help, son los que están en la primera columna de la tabla del apartado siguiente.

##  Tabla resumen de palabras reservadas

A continuación resumimos en una tabla el significado de las palabras reservadas.

|  |  |  |
| --- | --- | --- |
| **Palabra reservada** | **Formato** | **Descripción** |
| **help** | help | Muestra la ayuda de la aplicación (cuyo resumen está en esta tabla).  |
| **help** | help nombre | Muestra la ayuda concreta del tag “nombre”.  |
|  **Commands** |  |  |
|  **Print** |  print( expresion ) |  Imprime |
|  **user\_variables** |   |   |
|  **Assignvariable** |  variable = expresion |  Asigna variable |
|  **Clearvars** |  clearvars |  Limpia variables |
|  **Erasevar** |  erasevar( variable ) |  Elimina variable |
|  **user\_functions** |   |   |
|  **Assignfunction** |  f( x, y ) = expresion |  Define función |
|  **Composition** |  f( x ) = g( h( x ), i( x ) ) |  Composición |
|  **Val** |  val( expresion ) |  Valor |
|  **Subst** |  subst( f( x ) ) |  Substituye |
|  **Derivative** |  derivative( variable, n, function ) |  Función derivada |
|  **Simplify** |  simplify( expresion ) |  Simplifica expresión |
|  **Erasefunc** |  erasefunc( f ) |  Borra funciones |
|  **Clearfuncs** |  clearfuncs |  Limpia funciones |
|  **Operators** |   |   |
|  **Addition** |  sumando1 + sumando2 |  Suma |
|  **Subtraction** |  minuendo - sustraendo |  Resta |
|  **Multiplication** |  factor1 \* factor2 |  Multiplicación |
|  **División** |  dividendo / divisor |  División |
|  **circumflex** |  base ^ exponente |  Potencia |
|  **parenthesis** |  ( expresión ) |  Paréntesis |
|  **precedence** |   |  Precedencia de operadores |
|  **constants** |   |   |
|  **e** |  e |  Número de Euler |
|  **pi** |  pi |  Constante pi |
|  **ln2** |  ln2 |  Logaritmo neperiano de 2 |
|  **other\_commands** |   |   |
|  **exit** |  exit |  Salir |
|  **functions** |   |   |
|  **abs** |  abs( argumento ) |  Valor absoluto |
|  **absolute** |  absolute( argumento ) |  Valor absoluto |
|  **acos** |  acos( argumento ) |  Arco coseno |
|  **acosh** |  acosh( argumento ) |  Arco coseno hiperbólico |
|  **add** |  add( sumando1, sumando2 ) |  Suma |
|  **arccos** |  arccos( argumento ) |  Arco coseno |
|  **arccosh** |  arccosh( argumento ) |  Arco coseno hiperbólico |
|  **arcsin** |  arcsin( argumento ) |  Arco seno |
|  **arcsinh** |  arcsinh( argumento ) |  Arco seno hiperbólico |
|  **arctan** |  arctan( argumento ) |  Arco tangente |
|  **arctanh** |  arctanh( argumento ) |  Arco tangente hiperbólica |
|  **asin** |  asin( argumento ) |  Arco seno |
|  **asinh** |  asinh( argumento ) |  Arco seno hiperbólico |
|  **atan** |  atan( argumento ) |  Arco tangente |
|  **atanh** |  atanh( argumento ) |  Arco tangente hiperbólica |
|  **ceil** |  ceil( n, argumento ) |  Techo |
|  **ceiling** |  ceiling( n, argumento ) |  Techo |
|  **cos** |  cos( argumento ) |  Coseno |
|  **cosh** |  cosh( argumento ) |  Coseno hiperbólico |
|  **divide** |  divide( dividendo, divisor ) |  División |
|  **down** |  down( n, argumento ) |  Abajo |
|  **exp** |  exp( argumento ) |  Exponencial |
|  **floor** |  floor( n, argumento ) |  Suelo |
|  **gcd** |  gcd( argumento1, ..., argumentoN ) |  Máximo común divisor (greatest common divisor) |
|  **ln** |  ln( argumento ) |  Logaritmo neperiano |
|  **log** |  log( base, argumento ) |  Logaritmo |
|  **max** |  max( argumento1, ..., argumentoN ) |  Máximo |
|  **min** |  min( argumento1, ..., argumentoN ) |  Mínimo |
|  **multiply** |  multiply( factor1, factor2 ) |  Multiplicación |
|  **power** |  power( base, exponente ) |  Potencia |
|  **quotient** |  quotient( dividendo, divisor ) |  División |
|  **root** |  root( n, argumento ) |  Raíz |
|  **round** |  round( n, argumento ) |  Redondea |
|  **sgn** |  sgn( argumento ) |  Signo |
|  **sin** |  sin( argumento ) |  Seno |
|  **sinh** |  sinh( argumento ) |  Seno hiperbólico |
|  **sqrt** |  sqrt( argumento ) |  Raíz cuadrada |
|  **subtract** |  subtract( minuendo, sustraendo ) |  Resta |
|  **sum** |  sum( sumando1, sumando2 ) |  Suma |
|  **tan** |  tan( argumento ) |  Tangente |
|  **tanh** |  tanh( argumento ) |  Tangente hiperbólica |
|  **up** |  up( n, argumento ) |  Arriba |
|  **value** |  value( argumento ) |  Valor |

# Invocación desde la interfaz de comandos

En esta versión de la aplicación, se ha creado un binario .jar mínimo que puede ser invocado desde la interfaz de comandos, aunque no permite la ejecución interactiva.

La aplicación de interfaz de comandos ejecuta un solo comando que se le pasa por parámetro en la invocación.

Esta aplicación acepta tres parámetros de entrada:

* -precision (-precision=valor). Precisión a utilizar
* -threads (-threads=valor). Hilos a utilizar en la ejecución
* -command (“-command=comando”). Comando a ejecutar en la calculadora.

Notar que normalmente este parámetro tendrá que ir entre comillas dobles, ya que lo más normal es que se incluyan espacios, y sin las comillas, esos espacios no podrían formar parte del mismo parámetro.

Ejemplo:

java -jar multicalcu-commandline-v1.2-SNAPSHOT-all.jar -precision=1000 -threads=7 "-command=print( pi )"

Resultado:

3.141592653589793238462643383279502884197169399375105820974944592307816406286208998628034825342117067982148086513282306647093844609550582231725359408128481117450284102701938521105559644622948954930381964428810975665933446128475648233786783165271201909145648566923460348610454326648213393607260249141273724587006606315588174881520920962829254091715364367892590360011330530548820466521384146951941511609433057270365759591953092186117381932611793105118548074462379962749567351885752724891227938183011949129833673362440656643086021394946395224737190702179860943702770539217176293176752384674818467669405132000568127145263560827785771342757789609173637178721468440901224953430146549585371050792279689258923542019956112129021960864034418159813629774771309960518707211349999998372978049951059731732816096318595024459455346908302642522308253344685035261931188171010003137838752886587533208381420617177669147303598253490428755468731159562863882353787593751957781857780532171226806613001927876611195909216420199