UTILIZACIÓN DEL COMPUTADOR EN LA ENSEÑANZA DE LA MATEMÁTICA FUNDAMENTAL

FERNANDO PÍO BETANCOURT LÓPEZ

UNIVERSIDAD NACIONAL DE COLOMBIA

SEDE MANIZALES

1994

UTILIZACION DEL COMPUTADOR EN LA ENSEÑANZA DE LA MATEMATICA FUNDAMENTAL.

PAQUETE muMATH-83

A través del estudio que se ha hecho con este y otros paquetes, se ha podido observar que su utilidad es de gran importancia para propender por un mejoramiento continuo y sustancial de la educación matemática en todos los niveles del sistema educativo.

Es importante darse cuenta que toda la aritmética en muMATH es aritmética racional exacta, por lo tanto aritmética aproximada no está incluida. Sin embargo, muMATH trata de simplificar las expresiones que contienen operaciones irracionales. En algunos casos la expresión racional exacta o al menos a una expresión que es mucho más simple.

La educación matemática en nuestra Universidad y el medio educativo que nos rodea, está todavia fundamentada en exposiciones que sacrifican a nuestros docentes, a largas horas de tablero, y para los alumnos no existe una ayuda que sea más motivación que presión, por las cuales se puedan guiar.

En los últimos años el empleo de los computadores ha ido adquiriendo gran importancia y se ha visto la necesidad de que en la educación se les de la utilización debida facilitando el desarrollo de programas y contenidos. Creo que la elaboración de esta cartilla contribuye de alguna forma a llenar éste vacio, constituido en una necesidad cada vez más creciente.

muMATH-83 es un sistema matemático simbólico, totalmente interactivo que efectúa eficiente y acertadamente operaciones algebráicas y analiticas. Estas operaciones están más allá de las incluídas en lenguajes de programación científicos tradicionales tales como: APL, BASIC, FORTRAN, PASCAL ó PL/I. A diferencia de estos lenguajes y como subrutinas típicas escritas para ellos, muMATH puede evaluar

y simplificar expresiones que contienen variables a las que no se han asignado valores numéricos. Por ejemplo, muMATH puede automáticamente reducir expresiones a común denominador, emplear identidades trigonométricas para simplificar expresiones, e integrar exactamente expresiones simbólicas. Operaciones tales como inversión y producto de matrices pueden ser computadas en muMATH, aún cuando las matrices contengan entradas no numéricas. Variables no asignadas en los datos son tratadas algebráicamente tal como se hace en cursos de álgebra, trigonometría y cálculo. Además, los cálculos aritméticos realizados sobre coeficientes numéricos de tales variables son de alta precisión, aritmética racional exacta.

muSIMP-83 es el lenguaje de programación de propósito general en el cual muMATH está escrito. muSIMP fue diseñado especialmente para implementar sistemas de álgebra computacional. Se provee junto con muMATH para que los usuarios que deseen puedan usar muSIMP para extender muMATH o para implementar otras aplicaciones de inteligencia artificial. Sin embargo, la mayoría de usuarios encontrarán los integrantes de muMATH suficientemente poderosos para manejar todos sus problemas matemáticos.

muMATH y muSIMP fueron diseñados e implementados por David Stoutemyer y Albert Rich de Soft Warehouse, Honolulú, y Hawaii. El proyecto se comenzó en 1976 por Rich con el diseño y codificación de un interpretador de lenguaje LISP en un microcomputador 8080. Convencido del poder y utilidad de los símbolos matemáticos Stoutemyer colaboró con Rich para mejorar grandemente el interpretador, tanto en velocidad como habilidad numérica. La base del lenguaje muSIMP fué ideado durante éste período para dar al usuario el poder de un lenguaje aplicativo como LISP, pero con una sintaxis más natural. Finalmente muMATH-79, sistema de matemática simbólica, se escribió en muSIMP, haciéndolo el primero de tales sistemas disponible en microcomputador. La versión 83 de muMATH proporciona al usuario un mejor desempeño y capacidad matemática aumentada. Nuevos paquetes de matemáticas fueron agregados, incluyendo un paquete de límites, un paquete de sumatorias, un paquete de ecuaciones diferenciales ordinarias y un paquete de álgebra vectorial y cálculo.

muMATH-83 puede ser implementado por un procesador 8086 u 8088 sobre el sistema operativo MS-DOS (por ejemplo el computador personal IBM), y computadores basados en el 8080 y Z80 que corran el sistema operativo CP/M-80 (es decir el computador Apple II con una tarjeta de lenguaje Z80).

muMATH y muSIMP atraen grandemente a gentes que tienen diferentes conocimientos matemáticos y experiencia en el manejo de un computador. Esto se extiende desde un ingeniero profesional ó un maestro de matemáticas con poca experiencia en el computador hasta un "gomoso" de los microcomputadores con una base matemática limitada.

UN VISTAZO A muMATH Y muSIMP

La capacidad matemática de los lenguajes de programación tradicionales tales como ALGOL, APL, BASIC, FORTRAN, PASCAL y PL/I está esencialmente limitada a operaciones matemáticas. Programas escritos en estos lenguajes contienen fórmulas consistentes de variables, funciones y operadores. Sin embargo Ud. puede desilusionarse al enterarse de que un error ocurre cuando evalúa una fórmula que contiene una variable a la cual no se le ha asignado previamente un valor.

En contraste, muMATH puede evaluar y simplificar fórmulas que contienen variables a las que no se les ha asignado valores numéricos. Tales variables son tratadas algebráicamente como lo hace un matemático utilizando álgebra, trigonometría y cálculo. muMATH puede transformar y simplificar fórmulas en otras fórmulas equivalentes, automáticamente o al requerimiento explícito del usuario.

Implícitamente ningún conocimiento de programación de computadores se requiere para usar muMATH en el modo calculador, bajo las condiciones siguientes:

1. El operador "*" se usa para indicar multiplicación en vez de "." o "x". El "*" se puede omitir salvo entre una variable y una expresión entre paréntesis, que de otra forma puede ser interpretada como una llamada de función.

2. El operador "^" es usado para denotar exponenciación o "elevar una expresión a una potencia ".

3. Expresiones entradas por el usuario deben terminar en un PUNTO Y COMA (;) seguido de un <RETURN>.

Una de las grandes ventajas del paquete muMATH es la forma y precisión con que trabaja la Aritmética racional de precisión infinita o aritmética exacta, que es usada internamente por muMATH en todos los campos numéricos. Por ejemplo, si ud. entra la expresión

5/6 - 0.25;

muMATH mostrará

7/12

Note que las respuestas en forma de fracciones son reducidas a términos mínimos cuyos resultados son siempre exactos. Al eliminar la necesidad de explicar errores de redondeo se hace de muMATH una herramienta excepcional para enseñar matemáticas sobre un computador. Como un ejemplo de la capacidad de la aritmética de muMATH, para computar la octava potencia de diez factorial entre

10!^8

y muMATH mostrará

300679807141675805997423113304381849600000000000000

El número y tamaño de expresiones que muMATH puede manipular está limitado únicamente por la cantidad de memoria total RAM disponible en su computador. Cálculos involucrando cerca de 100 términos con coeficientes de cientos de dígitos son muy posibles.

Como se ha ilustrado en los ejemplos anteriores, muMATH tiene un amplio alcance de capacidad matemática, desde el álgebra elemental hasta el cálculo. Sin embargo muMATH es tan manejable que ud. no necesita tener un alto nivel matemático para usar las capacidades de bajo nivel tales como álgebra o aritmética racional exacta.

Usar muMATH es tan conveniente e interactivo como usar una calculadora de bolsillo. Las expresiones son transformadas y simplificadas cuando entran. Así, usando muMATH en este modo de calcular se elimina la necesidad de escribir y utilizar largos programas. Ud. puede libremente experimentar y manipular fórmulas y podrá ver inmediatamente los resultados.

Los errores se descubren rápidamente de tal manera que una fórmula puede ser reentrada sin tener que correr nuevamente el programa. Viendo cómo se calcula cada resultado, sugiero nuevas aventuras para explorar que no pueden ser abandonadas en la secuencia completa de cálculo que se ha preplaneado. El modo de calcular es análogo al correspondiente a una conversación por teléfono más que por correo: esto permite un diálogo en vez de una secuencia de monólogos separados por largos silencios.

Las facilidades internas de muMATH son tan poderosas que la mayoría de los usuarios encontrarán el modo de calcular suficiente para solucionar sus problemas matemáticos. Sin embargo, muMATH es escrito en el lenguaje de programación muSIMP, el cual fué diseñado especialmente para manejar las demandas de proceso matemático simbólico. Además, los archivos fuente de muMATH se proporcionan con el sistema en su forma original, de tal forma que ellos pueden ser modificados como se desee o usados como modelos para extensiones adicionales.

muMATH viene con un conjunto de lecciones que explican cómo programar en muSIMP. Sin embargo, a excepción del modo de calcular el uso del modo de programación requiere experiencia previa de programación; ningún lenguaje en particular, solo cierto grado de conocimiento de las técnicas de la programación.

Para satisfacer la curiosidad de aquellos interesados en muSIMP, he aquí, sin explicaciones adicionales en este momento, una definición de función muSIMP la cual da el número de Fibonacci correspondiente a su argumento entero. En matemáticas los números de Fibonacci son usualmente definidos como sigue:

$$F(0)=1,$$
$$F(1)=1,$$
$$F(n)=F(n-1) + F(n-2), \text{ para } n>1$$

Lo anterior es una definición recursiva válida, ya que ésta requiere únicamente el cálculo de números de Fibonacci pequeños. Para definir la correspondiente función muSIMP, también recursiva, entre la función:

```
FUNCTION F(n),
  WHEN  n=0, 1 EXIT,
  WHEN  n=1, 1 EXIT,
  F(n-1) + F(n-2),
ENDFUN;
```

Aunque sin entrar en detalles, es claro que la definición muSIMP para números de Fibonacci es muy parecida a la matemática. La definición de funciones en muSIMP puede hacerse desde muSIMP mismo o leerse de un archivo fuente preparado por un editor de texto. En cualquier caso, una vez la función se ha definido, es disponible para su uso. Si después de definir la función F, desea encontrar el 5 número de Fibonacci, entre la expresión

```
F(5);
```

y muMATH retornará un 8. (Aunque elegante, la definición dada anteriormente es extremadamente ineficiente ya que el tiempo requerido para encontrar el n-ésimo (enésimo) número

de Fibonacci crece exponencialmente con el tamaño de n. (Escribir una función de Fibonacci recursiva, pero eficiente, es un buen ejercicio en programación muSIMP).

El álgebra de computadores originalmente creció sobre el campo de inteligencia artificial. Así, no es sorprendente que un lenguaje de programación bien establecido para álgebra de computadores, sea bueno también para investigaciones de inteligencia artificial. muSIMP es en efecto una buena escogencia para trabajo de inteligencia artificial en general que involucra microcomputadores. Las lecciones de programación muSIMP enseñan técnicas que son útiles para programación de inteligencia artificial.

REQUERIMIENTOS DEL SISTEMA

muMATH-83 se ha implementado sobre una amplia variedad de microcomputadores. Aquí se lista el mínimo "hardware" y el "software" del sistema operativo requerido para correr muMATH en cada computador para los cuales se ha implementado.

COMPUTADOR PERSONAL IBM

1. Un computador personal IBM con teclado y monitor.
2. Al menos 128K (128Kw) de memoria RAM (si es más, mejor).
3. Al menos un drive standard para IBM PC, de disco flexible de 5 1/4 pulgadas.
4. El diskette que contenga "software" IBM-DOS (ya sea la versión 1.1 o 2.0) que es entregada con el computador.

COMPUTADORES MS-DOS

1. Un computador basado en el 8086 o 8088 que corra el sistema operativo MS-DOS de la Microsoft's.
2. Al menos 128K(Kw) de memoria RAM (entre más mejor).
3. Al menos un "drive" de disco flexible standard capaz de leer el diskette master muMATH (póngase en contacto con su distribuidor de muMATH para determinar si tienen diskette muMATH formateado para su drive de disco).

Realizado por: Fernando Pio Betancourt López
Profesor Asociado Universidad Nacional Manizales

CONTENIDOS DEL SISTEMA

EL SISTEMA muMATH-83: SISTEMA MATEMATICO SIMBOLICO

muMATH-83 consta de los siguientes temas:

1. LOS ARCHIVOS FUENTE muMATH-83 están impresos en archivos de texto ASCII. Cada archivo provee un paquete modular de capacidades matemáticas. Así, Ud. necesita únicamente leer en el archivo fuente requerido para resolver un problema matemático. Esto hace posible correr muMATH sobre computadores de memoria limitada. Las tres letras tipo dadas al nombre de un archivo fuente consisten de las tres primeras letras del archivo que es prerrequisito inmediato.

ARITH.MUS	Aritmética racional
ALGEBRA.ARI	Algebra elemental
EQN.ALG	Simplificación de ecuaciones
SOLVE.EQN	Solución de ecuaciones
ARRAY.ARI	Operaciones de vectores
MATRIX.ARR	Operaciones de matrices
LINEQN.MAT	Ecuaciones algebráicas lineales simultáneas
ABSVAL.ALG	Simplificación de valor absoluto
LOG.ALG	Simplificación Logarítmica
TRG.ALG	Simplificación Trigonométrica
ATRG.TRG	Simplificación trigonométrica inversa
HYPER.ALG	Simplificación trigonométrica hiperbólica
DIF.ALG	Diferenciación simbólica y series de Taylor
INT.DIF	Integración simbólica
INTMORE.INT	Integración simbólica extendida
LIM.DIF	Límites de funciones
SIGMA.DIF	Sumatorias y productos
ODE.SOL	Ec. diferenciales ordinarias de primer orden
ODENTH.ODE	Ec. diferenciales ordinarias de orden sup.
ODEMORE.ODE	Otros métodos para ec. diferenciales ordinarias de primer orden
VEC.ARR	Algebra vectorial
VECDIF.VEC	Cálculo vectorial

1. ARCHIVOS SYS muMATH son archivos no impresos en memoria. Un archivo SYS es creado por un comando SAVE y cargado mediante un comando LOAD. Cada archivo SYS contiene las capacidades matemáticas presentes en el momento en el cual fué grabado.

ALGEBRA.SYS	Incluye: ARITH.MUS y ALGEBRA.ARI.
CALCULUS.SYS	Incluye: ALGEBRA.SYS, LOG.ALG, TRG.ALG, ATRG.TRG, DIF.ALG, INT.DIF, INTMORE.INT, LIM.DIF, SIGMA.DIF HYPER.ALG y ABSVAL.ALG.

MATSOL.SYS Incluye: ALGEBRA.SYS, EQN.ALG, SOLVE.EQN, ARRAY.ARI, MATRIX.ARR y LINEQN.MAT.

3. Las lecciones del modo calculador muMATH son archivos de lecciones interactivas que enseñan cómo usar muMATH en el modo calculador.

CLES1.ALG Aritmética racional y asignación de variables.
CLES2.ALG Factoriales y potencias fraccionarias.
CLES3.ALG Expansión y factorización de polinomios.
CLES4.ALG Variable compleja y sustitución de expresiones

DESCRIPCION Y EJEMPLOS DEL CONTENIDO DEL PAQUETE ARITH.MUS Y LOS SUB-PAQUETES OPCIONALES

1. OPERADORES ARITMETICOS:

a. Los operadores " + " y " - " denotan adición y sutracción respectivamente. " - " se usa también como el operador de negación.

```
? 4+2-3;        ?--------456;
@: 3            @:-456
```

b. El operador " * " denota multiplicación. Generalmente el " * " se puede omitir de una expresión puesto que la multiplicación está implícita.

```
? 4 3 (-5);
@:-60
```

" * " debe utilizarse explícitamente después de una variable y antes de una expresión entre paréntesis, de otra forma la variable se interpreta como una función y la expresión entre paréntesis como su argumento.

```
? EG*(6-2); en lugar de EG (6-2);
```

c. " / " denota division. Las fracciones siempre se reducen a términos mínimos (un entero que divide al numerador y al denominador es simplificado).

```
? 4/6;          ? 5/9 + 7/12;
@:2/3           @:41/36
```

d. " ^ " denota potenciación

```
? 2^4;
@:16
```

e. " : " asigna a la variable del lado izquierdo el valor de la expresión del lado derecho. Si a una variable se le ha asignado un valor, se puede usar la variable en lugar de su valor.

```
? FOO:20-7;   ? FOO + 2;
@:13          @:15
```

Una variable conserva su valor hasta que sea reasignada. La expresión del lado derecho del operador " : " se e[illegible] que la variable cambie de valor.

```
? FOO:-FOO;
@:-13
```

PRECEDENCIA DE LOS OPERADORES

(ORDEN DECRECIENTE)

OPERADOR	OPERACION
`	comilla (para acotar)
!	factorial
^	potenciación
-	negativo
* /	multiplicación y división
+ -	adición y sustracción

Cuando hay más de un operador en una expresión, la precedencia relativa de los operadores determina la operación que se calcula primero.

```
? 6+3,2;
@: 12
```

La multiplicación tiene mayor precedencia que la adición.

Si dos operadores tienen la misma precedencia, la expresión se calcula de izquierda a derecha.

```
? 14/2 2;
@:14
```

La única excepción a la regla "izquierda - derecha" es el operador de potenciación "^" , el cual evalúa de "derecha a izquierda".

```
? 3^3^2;
@:19683   ----> en lugar de 729
```

Para cambiar el orden de evaluación, deben utilizarse paréntesis para agrupar los términos considerados. por ejemplo, para encontrar el valor de 7+8/6 entrar la expresión:

```
? (7+8)/6;  en lugar de ? 7 + 8/6
@:5/12                  @:25/3
```

Como una conveniencia adicional, muMATH conserva el resultado previo asignándolo a la variable " @ ".Como advertencia de esto, en muMATH los resultados se dan mediante la asignación " @ "

```
? 1234567890123456789 9;
@:11111111011111111101

? @^2;
@:12345678790123457767654321189876543220 1
```

FUNCIONES ARITMETICAS

a. **ABS (Expr)** retorna el valor absoluto de (expr) si (expr) es un número racional. Además el paquete de valor absoluto realiza simplificaciones algebráicas para que se pueda evaluar la función ABS.

```
? ABS(-5+4);    ? ABS((3*2)^2);    ? ABS(4/3*(-5));
@: 1            @: 36              @: 20/3
```

b. **DEN (Expr)** retorna simplificado el denominador de (expr) si en (expr) aparece un denominador; en otro caso retorna 1.

```
? DEN(8/12);  ? DEN(22);    ? DEN(3/5);
@: 3          @: 1          @: 5
```

c. **GCD(entero 1, entero 2)** retorna el máximo común divisor de (entero 1) y (entero 2). Si a GCD se le dá un argumento no entero se produce el mensaje de error "Non numeric argument" (argumento no numérico).

```
? GCD(-15,25);      ? GCD(2,4,6,8,10,12);
@: 5                @: 2
```

```
? GCD(-40,-20,-120,-4)↵  ? GCD(2,3,5,7,11,13)↵
@: 4                     @: 1
```

d. **LCM (entero 1, entero 2)** retorna el mínimo común múltiplo de (entero 1) y (entero 2). Ocurre un mensaje de error: "Non numeric argument" si a LCM se le dá un argumento no numérico.

```
? LCM (-3,12)↵  ? LCM(3,8)↵  ? LCM(-20,-13)↵
@: 12           @: 24        @: 260
```

NOTA: LCM no retorna en forma inmediata el mínimo común múltiplo de tres ó mas enteros, en este caso retorna el mínimo común múltiplo de los dos primeros enteros.

```
? LCM(5,7,3)↵
@: 35, pero se puede calcular:

? LCM(LCM(5,7), 3)↵
@: 105
```

e. **MIN (número 1, número 2)** retorna el mínimo de sus dos argumentos racionales.

```
? MIN(-85,-96)↵      ? MIN(4/8,3/6)↵
@:-96                @: 1/2
```

NOTA: MIN no retorna en forma inmediata el mínimo de tres ó más argumentos. En este caso retorna el mínimo de los dos primeros números.

```
? MIN(60,30,2)↵
@: 30, pero se puede calcular:

? MIN(MIN(60,30),2)↵
@: 2
```

f. **MOD (entero 1, entero 2)** retorna (entero 1) módulo (entero 2). MOD es equivalente a la función residuo encontrada en muchos lenguajes de programación. Ocurre un error si a MOD se le da un argumento no numérico.

```
? MOD(13,8)↵  ? MOD(-50,-20)↵  ? MOD(12,-6)↵
@: 5          @: 10            @: 0
```

g. **NUM(Expr)** retorna el numerador simplificado de (expr) si en (expr) aparece un numerador. En otro caso retorna (expr).

```
? NUM(8/4)↵  ? NUM(3/5)↵  ? NUM(45)↵
@: 2         @: 3         @: 45
```

h. **RADIX (entero)** transforma la representación INPUT y OUTPUT a base (entero), si (entero) está en el intervalo 2-36 y retorna la base original en representación base (entero).

Si (entero) no está en el intervalo 2-36, se retorna la base corriente.

Para bases mayores que 10, se usan las letras "A" hasta la "Z" para representar los dígitos 11 a 36, teniendo el cuidado de colocar un cero al comienzo para distinguirlos de nombres.

Una vez hecho el cambio de base, pueden realizarse cálculos en dicha base, teniendo presente que las cantidades que se introduzcan deben estar en la base en la cual se desea trabajar.

Si se quiere hacer otro cambio de base, se debe salir de la base anterior y ésto se puede hacer de la siguiente manera:

RADIX(TEN); ó RADIX(*);

Donde * corresponde a la base 10 representada en la base (entero).

```
A: 32$
TEN:RADIX(3);
@: 101

A;
@:1012

RADIX(101);
@: 3

D: 98 $              X:3456$
TEN:RADIX(5);        TEN:RADIX(16);
@: 20                @:0A

D;          X;
@: 343      @:0D80

RADIX (20); ? 5+0A;
@: 5        @: 0F

RADIX(0A);
@: 16
```

SUBPAQUETES OPCIONALES

A. SUBPAQUETE DE NOTACION DECIMAL.

Los números racionales se pueden entrar al computador como fracciones usando el operador " / ". Este subpaquete: punto decimal, permite expresar los números racionales usando notación del punto decimal y/o notación cientifica; la variable de control POINT controla la exhibición en pantalla de los números racionales.

1. Si **POINT** es cero ó un entero positivo, los números racionales se muestran usando notación del punto decimal. El máximo número de dígitos que se muestran después del punto decimal, está dado por el valor de POINT.

```
? POINT: 20$

? 5/3;
@: 1.66666666666666666666

? 1/4;
@: 0.25
```

2. Si POINT es un entero negativo, los números racionales mayores o iguales que la base elevada a la cuarta potencia ó menores que uno sobre la base a la cuarta potencia se muestran usando notación científica.

```
? POINT:-10$

? 5342900000;
@:5.3429 10^9

? 0.0000001234;
@: 1.234 10^-7
```

3. Si POINT no es un entero,los números racionales se exhiben como una fracción; los números se pueden entrar usando notacion decimal o científica.

```
? POINT: FALSE$

? 0.25;
@: 1/4
```

4. Si la base usada en muMATH para entrar y sacar una cantidad se ha transformado usando la función RADIX y si POINT tiene un valor entero, los números racionales se exhibe correctamente en notación de punto decimal de la base corriente.

SUBPAQUETE POTENCIA FRACCIONARIA.

Este subpaquete de ARITH.MUS proporciona los medios para la simplificación de potencias fraccionarias de números exponeciales complejos.

1. Si la variable de control **PBRCH** no es falsa, muMATH automáticamente escoge una de las diversas raíces de una potencia fraccionaria. Si PBRCH es falsa y hay varias raíces muMATH se abstiene de seleccionar una. PBRCH es generalmente verdadera.

2. Supuesto que PBRCH es verdadera, un número elevado a una potencia fraccionaria retorna una raíz real no negativa si existe alguna; en otro caso, retorna una raíz real negativa, si existe alguna. Si no existe raíz real, la expresión se muestra tal y como se introdujo.

```
? 9^(1/2);          ? (-64)^(1/3);          ? 6^(1/2);
@: 3                @: -4                   @:6^(1/2)
```

3. Como en cómputos manuales, escoger una raiz de una potencia fraccionaria implica una selección arbitraria que puede producir resultados no válidos. Así, el usuario está cuestionado a verificar los resultados obtenidos por tal operación.

4. Cuando PBRCH es verdadera y no existe raíz real, la potencia fraccionaria retorna la raíz compleja que tiene el menor ángulo positivo, medido desde el eje real positivo hasta el el eje imaginario positivo. Este número complejo está representado en forma polar, usando #I para representar (-1)^(1/2), #E para representar la base del logaritmo natural (2.71828...) y #PI para representar la circunferencia de un círculo de diámetro unitario (3.14159...);

```
? (4 #I)^(1/2);
@: 2#E^(1/4 #I #PI)
```

5. La variable global **PRIMES** contiene una lista de primos desde 2 hasta 19. Para potencias fraccionarias, el numerador y el denominador del radicando se factorizan en un producto de los números comprendidos en PRIMES, la potencia fraccionaria se distribuye sobre este producto; utiliza una variación discreta del método de Newton para determinar si la potencia fraccionaria de algún residuo es un entero. Así, la simplificación de potencias fracionarias de enteros grandes puede ser incompleta si la lista de PRIMES no contiene una cantidad suficiente de primos.

```
? 24^(1/3);          ? 24334^(1/3);
@: 2 3^(1/3)         @: 23 2^(1/3)
```

6. Exponenciales complejos son reducidos al módulo (2#PI #I);

```
? #E^(15#I #PI/7);
@: #E^(#I #PI/7)
```

Sin embargo, debe tenerse en cuenta que como en cómputos manuales, ésta reducción es inconsistente con la identidad:

LN(Z W) = LN(Z) + LN(W)

Se debe tener cuidado para verificar resultados obtenidos usando ambas transformaciones a la vez.

SUBPAQUETE FACTORIAL:

Este subpaquete proporciona medios para calcular factoriales.

1. El factorial de un entero no negativo, N, se define recursivamente como:

$$N!=1, \text{ si } N=0$$
$$N!=N*(N-1)!, \text{ para } N>0$$

2. " ! " denota el operador factorial.

```
? 8!;
@: 40320
```

AL USUARIO

Tiene en sus manos uno de los paquetes de muMATH más interesante, sobre todo porque la capacidad matemática de su lenguaje de programación es increible. muMATH puede evaluar y simplificar fórmulas que contienen variables a las que no se les ha asignado valores numéricos. Tales variables se tratan algebráicamente como lo hace cualquier estudiante utilizando álgebra, trigonometria y cálculo. muMATH puede transformar y simplificar fórmulas equivalentes, automáticamente o a requerimiento explícito del usuario.

No es necesario ningún conocimiento de programación de computadores para usar muMATH, pero debe tener en cuenta:

1. El operador " * " se usa para indicar multiplicación en vez de " . " o " x ". El " * " se puede omitir salvo entre una variable y una expresión entre paréntesis.

2. El operador " ^ " es usado para denotar potenciación o "elevar una expresión a una potencia"

3. Las expresiones entradas deben terminar en punto y coma (;) seguido de <RETURN> (o <ENTER>). Si la expresión no ha sido terminada en punto y coma, proceda así:

   - Escriba únicamente " ; " y luego <ENTER>, y la expresión se evalúa.

   - Complétela con un error intencional de sintaxis, tal como " _; " para prevenir una evaluación.

4. Si no necesita la exhibición del resultado computado y asignado, termine la asignación con un signo de pesos ($) en vez de un punto y coma.

5. Cuando una expresión tiene más de un operador es importante saber el orden en el cual se están haciendo. El orden es consistente con las convenciones matemáticas estándar.

FUNCION	OPERADOR
Para acotar	'
Factorial	!
Potenciación	^
Multiplicación	*
División	/
Suma	+
Resta	-

" ' " tiene precedencia más alta que " ! ", el cual tiene precedencia más alta que " ^ ", el cual tiene precedencia más alta que " * " ó " / ", los cuales tienen precedencia más alta que " + " ó " - ".

6. Los errores computacionales como por ejemplo: una división por cero producen una interrupción, dándole la oportunidad de

A (Abort)	Salirse.
B (Break)	Interrumpir, para suplirlo por un resultado terminado por "_".
C (Continue)	Continuar, con un valor específico que falta.
D (Disk)	Para volver al sistema operativo del disco. (sale de MUSIMP).

7. La variable nombrada " @ " te guarda automáticamente la última respuesta; puedes usar tal valor en la próxima expresión. Por ejemplo

```
? 5 + 7;
@ : 12

? 3 * @ + 6;
@ : 42
```

y ahora el valor de @ es 42.

8. También puede dar el valor a una variable comenzando siempre por una letra y contener solamente letras, dígitos y " # ". Las letras mayúsculas se distinguen de las minúsculas. Observe estos ejemplos.

```
Aa: 3;        G1: @;        Ele: 76;        C: 300000000;
```

y las variables las puede emplear como el valor que les ha asignado.

También puede entrar expresiones y comenzar a cambiar el valor de la variable, asi:

```
? EG**3

? EG: 2;          ? EG: -2;          ? EG: @;
@: 8              @: -8              @: 512
```

RECUERDA QUE: " * " no se puede omitir entre un nombre (una variable) y una expresión entre paréntesis, ya que un nombre seguido por una expresión entre paréntesis denota una función con su argumento. Por ejemplo: A1 (2 + 3X) se interpretaría como la función A1 de argumento (2 + 3X) en vez de A1 * (2 + 3X).

9. Al realizar alguna evaluación puede obtener un resultado tan largo que en un principio se sale de la pantalla antes de que haya tenido oportunidad de estudiarlo. Para superar esto, congele temporalmente la exhibición del resultado, asi: oprima la tecla marcada CTRL al mismo tiempo que escribe S. Cuando desee seguir presione cualquier tecla, diferente a CTRL-S. De otra forma podría quedar incierto del estado de la máquina sin saber cómo salirse o proseguir. Ensaye entrando ABS (- 295!); congele la exhibición del resultado una vez comience a aparecer y luego prosiga.

COMO ENTRAR AL PROGRAMA MUMATH

Primero cargue el sistema operativo del computador, luego introduzca un diskette que contenga una copia de muSIMP. Después del prompt (indicador de listo) del sistema operativo,el cual es el signo >, entre el comando

muSIMP

y presione la tecla marcada <RETURN> (o <ENTER>).

Después de unos segundos aparece en la pantalla el siguiente mensaje:

```
muSIMP-83 4.06(12/18/83)
MS-DOS Versión
Copyright (C) 1982 The Soft Warehouse
Licensed by Microsoft Corp.
```

seguido de un signo de interrogación (?) el cual le indica a usted que está listo para cargar el paquete de estudio.

Una forma alterna para entrar y cargar archivos SYS, por ejemplo, ALGEBRA.SYS o CALCULUS.SYS es desde el nivel del sistema operativo, junto con muSIMP. Si MUSIMP.COM está en <drive A> y el archivo <name>.SYS está en <drive B> un comando operativo de la forma

A:MUSIMP B:name

primero corre MUSIMP.COM y luego carga <name>.SYS. Como siempre, en tales comandos, los prefijos de drive se pueden omitir del nombre del archivo si el archivo está registrado en el drive. Por ejemplo, si MUSIMP.COM está registrado en el drive A y usted desea cargar ALGEBRA.SYS del drive B, use el comando:

MUSIMP B:ALGEBRA

EDITANDO LINEA Y ERRORES DE SINTAXIS

Las facilidades del editor de línea provisto por el sistema operativo son inherentes a muSIMP y muMATH. Esas facilidades incluyen unos medios de borrado de caracteres retrocediendo el cursor. Usualmente, la tecla devolutora está marcada <BACKSPACE>, <DELETE>, <RUB>, <<--- >. Entrando <CTRL-H> (presionando la tecla H mientras está oprimiendo la tecla CTRL) es otra manera de retroceder.

El sistema operativo también provee unos medios para borrar el texto sobre la línea de entrada. Algunos sistemas operativos (e.g. MS-DOS) tienen unos medios de restablecer la línea previa del texto, haciendo posible la reedición.

Entre un <CTRL-S> para parar temporalmente la presentación, para evitar que el listado del texto se salga de la pantalla antes de que pueda leer. Entre un <CTRL-P> para producir una copia escrita de una sesión muSIMP.

Si su computador tiene impresora, después de presionar <CTRL-P> (en el IBM PC puede usar también <CTRL-PrtSc>), los caracteres remitidos por muSIMP a la pantalla también serán remitidos a la impresora. Los manuales de referencia del sistema operacional del computador describen en forma más detallada las facilidades que provee el editor de línea.

Usted puede únicamente corregir errores sobre la línea de entrada usando las facilidades del editor de línea. Una vez que usted presione la tecla <RETURN>, muSIMP lee y analiza la expresión sobre la línea de impresión.

Si muSIMP detecta un error de sintaxis (i.e. una forma gramaticalmente incorrecta), el mensaje amonestador:

```
Syntax Error!
```

se muestra en la pantalla.Este se sigue por una frase que da la causa del error. En la línea debajo del mensaje de error, el resto de la expresión se reescribe, comenzando desde el lugar donde se detectó el error. Esto ayudará a identificar rápidamente la causa del problema. Por ejemplo, si usted introduce:

```
4 = /5;
```

muSIMP mostrará

```
Syntax Error: / usado como operador prefijo /5;
```

para indicar un error de sintaxis causado por un uso ilegal del operador "/".

Si un punto y coma no se ha entrado en seguida de una expresión indeseada comenzando una línea previa, un error de sintaxis se puede introducir deliberadamente para evitar la evaluación. Por ejemplo, una expresión se podría terminar con (;.

INTERRUMPIENDO LA EVALUACION

Cuando se entra una expresión sintácticamente correcta, muSIMP intentará evaluarla. Si la expresión no es la que pensó entrar y desea salirse de la evaluación por consumir mucho tiempo, usted puede usar la tecla <ESC> para interrumpir; puede lograr lo mismo presionando la tecla <C> mientras oprime la tecla <CTRL> (entrar <CTRL-C>).

Cuando muSIMP detecta la tecla intrruptora, el mensaje interruptor:

Interrupt
Abort, Break, Continue, Dos?

aparece en la pantalla y muSIMP espera para que usted seleccione una opción, **A, B, C ó D.** La elección usual es entrando una A, la cual inicia un nuevo ciclo de interacción. Todas las funciones definidas y valores existentes en el momento de la interrupción se preservan.

Entre C si usted decide que la evaluación prosiga después de todo. Entre D para salir de muSIMP y retornar el control al sistema operativo. Esta será la respuesta razonable si el medio muSIMP se ha desordenado tanto y usted decide salir a una nueva sesión.

La opción B rompe temporalmente la evaluación de una expresión, permite usar comandos muSIMP y luego reasume la evaluación de la expresión.

Si usted es novato en muMATH, puede posponer el estudio del uso de esta característica. El paquete muSIMP para "romper" se describe en el capítulo 12.

Si por alguna razón no hay respuesta a una tecla interruptora, siempre es posible interrumpir el sistema por un resetting en el computador. Sin embargo, un reset hecho en la mitad de una evaluación deja a muSIMP en un estado impredecible, haciendo imposible restablecer el medio muSIMP presente al momento del reset. muSIMP y muMATH se tendrán que recargar.

COMO SALIR DEL PROGRAMA MUMATH

El comando muSIMP

```
SYSTEM ();
```

termina el ciclo de interacción muSIMP y retorna el control al sistema operativo.

Realizado por: Fernando Pio Betancourt López
Profesor Asociado Universidad Nacional Manizales

ALGUNOS CONCEPTOS BASICOS DEL PAQUETE MATEMATICO

"DERIVE"

COMANDOS DE EDICION

TECLAS DE COMANDOS ESPECIALES:

Backspace - Elimina caracter a la izquierda del cursor
Del - Elimina el caracter sobre el cursor
Enter - Da entrada a línea de texto
Ctrl-Enter - Entra & simplifica línea de texto
Esc - Aborta edición & retorna al menú
Ins - Intercambia los modos insertar / sobreescribir
F3 - Inserta expresión iluminada
F4 - Inserta expresión destacada entre paréntesis

COMANDOS DE MOVIMIENTO DEL CURSOR:

Ctrl-S - mueva a la izquierda un carácter
Ctrl-D - mueva a la derecha un carácter
Ctrl-A - mueva a la izquierda una palabra
Ctrl-F - mueva a la derecha una palabra
Ctrl-Q S - mueve al principio de la línea
Ctrl-Q D - mueve al fin de la línea

COMANDOS DE BORRADO DE TEXTO:

Ctrl-H - Borra un caracter a la izquierda del cursor (= Backspace)
Ctrl-G - Elimina el caracter sobre el cursor (= Del)
Ctrl-T - Borra la palabra o fracción, desde el cursor
Ctrl-Y - Borra la línea donde está el cursor
Ctrl-Q Y - Borra todo desde el cursor hasta el fin de la línea
Ctrl-Q H - Borra desde el inicio de la línea hasta el punto donde está el cursor

OTROS COMANDOS:

Ctrl-M - Da entrada a línea de texto (= Enter)
Ctrl-J - Entra & simplifica línea de texto (= Ctrl-Enter)
Ctrl-[- Aborta edición & retorna al menú (= Esc)
Ctrl-V - Intercambia los modos insertar / sobreescribir (= Ins)
Ctrl-U - Inserte línea previa de texto

CARACTERES ESPECIALES (ALFABETO GRIEGO)

Alt-A – alpha
Alt-B – beta
Alt-G – gamma
Alt-D – delta
Alt-N – epsilon
Alt-F – phi
Alt-H – theta
Alt-M – mu
Alt-P – pi
Alt-S – sigma
Alt-T – tau
Alt-O – omega
Alt-E – #e – base de los logaritmos naturales
Alt-I – #i – raíz cuadrada de -1
Alt-Q – SQRT – Operador raíz cuadrada

FUNCIONES, CONSTANTES Y OPERADORES:

El **Derive** puede aproximar y/o simplificar las siguientes funciones, constantes y operadores. Los argumentos numéricos dados en modo {aproXimation}, son numéricamente aproximados para la precisión indicada en {Option} {Precision}. De otra manera las simplificaciones algebraicas y trigonométricas más complicadas, se resuelven con la opción {Manage}.

CONSTANTES:

 # e = Alt-e – base de logaritmos naturales
 # i = Alt-i – raíz-cuadrada de -1
pi = Alt-p – área de disco de unidad
deg – radianes por grado (pi / 180)
inf – más infinito

OPERADORES:

+ más
– menos o diferencia
***** por (a*b = ab)
/ cociente
^ exponenciación (a^2, a^3,...)
% percent (a % = a/100)
! factorial
= igual
/= diferente
< menor que
<= menor o igual que
> mayor que
>= mayor o igual que

LAS FUNCIONES EXPONENCIALES Y LOGARITMICAS

```
EXP (z)      - # e elevado a la z
LN (z)       - logaritmo natural de z
LOG (z)      - logaritmo natural de z
LOG (z, w)   - logaritmo de z en base w
SQRT (z)     - raíz cuadrada de z
```

LAS FUNCIONES TRIGONOMETRICAS

```
SIN (z*deg) - seno de z (en grados)
SIN (z)      - seno de z en radianes
COS (z)      - coseno de z (radianes)
TAN (z)      - tangenta de z (radianes)
COT (z)      - cotangente de z (radianes)
SEC (z)      - secante de z (radianes)
CSC (z)      - cosecante de z (radianes)
ATAN (z)     - arco cuya tangente es z
ATAN (y,x)   - ángulo del vector (x,y)
ACOT (z)     - arco cuya cotangente es z
ACOT (x,y)   - ángulo del vector (x,y)
ASIN (z)     - arco cuyo seno es z
ACOS (Z)     - arco cuyo coseno es z
ASEC (z)     - arco cuya secante es z
ACSC (z)     - arco cuya cosecante es z
```

FUNCIONES HIPERBOLICAS

```
SINH (z)  - seno hiperbólico de z
COSH (z)  - coseno hiperbólico de z
TANH (z)  - tangente hiperbólica de z
COTH (z)  - cotangente hiperbólica de z
SECH (z)  - secante hiperbólica de z
CSCH (z)  - cosecante hiperbólica de z
ASINH (z) - arcoseno hiperbólico de z
ACOSH (z) - arcocoseno hiperbólico de z
ATANH (z) - arcotangente hiperbólica de z
ACOTH (z) - arcocotangente hiperbólica de z
ASECH (z) - arcosecante hiperbólica de z
ACSCH (z) - arcocosecante hiperbólica de z
```

FUNCIONES ESPECIALES:

```
ABS (x)          - Valor absoluto de x
SIGN (x)         - signo de x
MAX (x, y, ...)  - máximo de los argumentos
MIN (x, y, ...)  - mínimo de los argumentos
STEP (x)         - 1 si x>0, 0 si x<0
CHI (a, x, b)    - 1 si a<x<b, 0 si x<a or x>b
FLOOR (m, n)     - el mayor entero <= m/n
FLOOR (m)        - parte entera de m
MOD (m, n)       - m modulo n (nonegativo residuo de m/n)
MOD (m)          - parte fraccionaria de m
NEXT_PRIME (n)   - Siguiente primo mayor que n
```

```
FIBONACCI ( n ) = n-simo número de Fibonacci
BERNOULLI ( n ) = n-simo número de Bernoulli
CATALAN (n)     = n-simo número de Catalán
NTH_PRIME (n)   = n-simo número primo
POLY_COEFF (u,x,n) = coeficiente del término en x^n en el
                     polinomio u(x)
POLY_DEGREE (u,x)  = grado del polinomio u(x)
```

FUNCIONES DE VARIABLE COMPLEJA:

```
ABS (z)   = Valor absoluto o magnitud de z
SIGN (z)  = proyección radial de z en círculo unitario
RE (z)    = parte real de z
IM (z)    = parte imaginaria de z
CONJ (z)  = complejo conjugado de z
PHASE (z) = argumento principal de z
```

FUNCIONES ESTADISTICAS Y DE PROBABILIDAD:

```
z!           = z factorial
GAMMA (z)    = Función gamma de z
PERM (z,w)   = permutaciones de z tomando w a la vez
COMB (z,w)   = combinaciones de z tomando w a la vez
RANDOM (n)   = si n>1, un Entero aleatorio en el intervalo
               [0,n)
RANDOM (n)   = si n=1, un Número aleatorio en el intervalo
               [0,1)
AVERAGE (z1, ..., zn) = Promedio aritmético (la media)
RMS (z1, ..., zn)     = Valor medio cuadrático
VAR (z1, ..., zn)     = varianza
STDEV (z1, ..., zn)   = desviación estandard
FID (m)               = Regresión de mínimos cuadrados de la
                        matriz m.
```

FUNCIONES DE ERROR:

```
ERF (z)          = función de error
ERF (z, w)       = función de error generalizada
ERFC (z)         = función de error complementaria
NORMAL (z, m, s) = distribución normal con media a y de
                   desviación estándar s
NORMAL ( z )     = distribución acumulativa de z
```

SOLUCION DE RELACIONES

```
SOLVE (u,x)          = Resuelve u=0 para x
SOLVE (u=v,x)        = Resuelve u=v para x
SOLVE (u<v,x)        = Resuelve u<v para x
SOLVE (u=v,x,a,b)    = Resuelve u=v para x en [a,b]
SOLVE ([u1=v1,u2=v2,...], [x1,x2,...]) = Resuelve el sistema
                     de ecuaciones lineales
                     u1=v1,u2=v2,...con respecto a las
                     variables x1, x2, ...
```

FUNCIONES PARA CALCULO

```
LIM (u,x,a) -        límite de u cuando x tiende a a
LIM (u,x,a,1) -      límite de u cuando x tiende a a a derecha
LIM (u,x,a,-1) -     límite de u cuando x tiende a a por la
                     izquierda
DIF (u,x) -          derivada de u con respecto a x
DIF (u,x,n) -        n-ésima derivada de u con respecto a x
TAYLOR (u,x,a,n) -   Polinomio de Taylor de orden n de la
                     función u alrededor de x=a
INT (u,x) -          antiderivada de u con respecto
INT (u,x,a,b) -      integral definida de u, respecto a x, de a
                     hasta b
SUM (u,n) -          antidiferencia de u con respecto a n
SUM (u,n,k,m) -      suma de u para n desde k hasta m
DIF_NUM (y,x,x0,h) - primera derivada de y respecto a x en
                     x0 usando pasos de tamaño h
DIF2_NUM (y,x,x0,h)- segunda derivada de y respecto a x en
                     x0 usando pasos de tamaño h
```

FUNCIONES VECTORIALES

```
[x1, x2, ..., xn] -  vector de n elementos
VECTOR (u,k,n) -     define el vector u, donde k va desde 1
                     hasta n en saltos de 1
VECTOR (u,k,m,n) -   define el vector u, donde k va desde m
                     hasta n en saltos de 1
VECTOR (u,k,m,n,s) - define el vector u, donde k va desde m
                     hasta n en saltos de s
ELEMENT (v,n) -      elemento n del vector v
DIMENSION (v) -      número de elementos del vector v
ABS (v) -            magnitud o longitud del vector v
v . w -              producto punto o escalar de vectores
CROSS (v,w) -        producto cruz o vectorial
```

FUNCIONES MATRICIALES

```
[[a, b], [c, d]] -   matriz 2x2
IDENTITY_MATRIX (n) - matriz identidad nxn
ELEMENT (A,j,k) -    elemento en la fila j y columna k de
                     la matriz A
A.B -                producto de matrices
A' -                 matriz transpuesta de A
DET (A) -            determinante de A
TRACE (A) -          traza de A (suma de los elementos de
                     la diagonal)
A^-1 -               inversa de A
MINOR (A,i,j) -      borra fila i & columna j de A
CHARPOLY (A, x) -    polinomio característico de A, con
                     respecto a la variable x
EIGENVALUES (A) -    eigenvalores de A
```

CALCULO VECTORIAL

POTENCIAL (v) - potencial escalar de v
GRAD (expn) - gradiente de expn dependiendo de **x, y** y **z**
GRAD (expn,v) - gradiente de expn dependiendo de las variables en el vector v
GRAD (expn,A) - gradiente de expn en el sistema de coordenadas dado en A.
DIV (v,A) - divergencia del vector v en el sistema de coordenadas dado en A.
LAPLACIAN (expn,A) - divergencia del gradiente de expn = laplaciano de expn
CURL (v,A) - rotacional del vector v

Nota: Para más información sobre otras funciones que tiene implementadas el paquete Derive, favor consultar las ayudas internas.

COMO ENTRAR AL PROGRAMA DERIVE

Primero cargue el sistema operativo del computador, luego introduzca un diskette que contenga una copia del paquete derive. Después del prompt (indicador de listo) del sistema operativo, el cual es el signo >, entre el comando

derive

y presione la tecla marcada <RETURN> (o <ENTER>). Asi, puede ya entrar a operar y utilizar las distintas órdenes y funciones que contiene este paquete. A continuación entra a seleccionar alguno de los comandos propios del paquete:

Author Build Calculus Declare Expand Factor Help Jump soLve Manage Options Plot Quit Remove Simplify Transfer moVe Window approX

Para pasar de un comando a otro podemos pulsar la tecla en mayúsculas o movernos con la **barra espaciadora** ó con la tecla **Tab** ó con la tecla **Backspace**.

Un pequeño vistazo a los comandos:

Author - entre nueva expresión
Build - combine expresiones
Calculus
Differentiate - deriva expresión
Integrate - integra expresión
Limit - encuentre límite de una expresión
Product - encuentre producto de forma cerrada
Sum - encuentre suma de forma cerrada
Taylor - encuentre aproximación de Taylor

Declare
Function – defin@ función
Variable
Domain – declare el dominio de las variables
Value – declare el valor de las variables
Matrix – declare una matriz
vectoR – declare un vector
Expand – expande (sub)expresión
Factor – factoriza (sub)expresión
Help Ubicándose sobre este comando:
Editing – commandos de edición de línea
Functions – functions & constants
Algebra – commandos de la ventana de algebra
2D-plot – commandos de ventana de gráficos en 2D
3D-plot – commandos de ventana de gráficos en 3D
Utility – funciones utilitarias de archivo
State – estado corriente del sistema
Return – retorna al Derive
Jump – expresión resaltada
soLve – resuelve ecuación, relación o vector de ecuaciones lineales
Manage
Exponential – transformaciones exponenciales
Logarithm – transformaciones logarítmicas
Ordering – ordenando variables
Substitute – reemplazo para variables o una subexpresión
Trigonometry – transformaciones trigonométricas
Options
Color
Notation – muestra estilo de notación numérica
Precisión – muestra modo de precisión aritmética
Radix – muestra input & output radix bases
Plot – dibuja (sub)expresión
Quit – sale del Derive
Remove – remueve expresiones
Simplify – simplifica (sub)expresión
Transfer
load
Derive – carga expresiones de archivos MTH
State – carga estado de archivos ENI
daTa – carga datos numéricos de DAT archivo
Utility – carga definiciones de MTH archivo
Save
Derive – graba expresiones en archivo MTH
Basic – graba expresiones en archivo BAS
C – graba expresiones en a archivo C
Fortran – graba expresiones en archivo FOR
Pascal – graba expresiones en archivo PAS
Options – selecciona rango de grabación y longitud de línea de archivo

Window
Close – cierra ventana
Designate – designa el tipo de ventana
Flip – retira ventanas superpuestas (= tecla F2)
Goto – ir a una ventana dada
Next – ir a la siguiente ventana (= tecla F1)
Open – abrir ventana
Previous – ir a la ventana previa
Split – parte ventana
Horizontal –horizontalmente
Vertical – verticalmente
approX – aproxima (sub)expresión

Teclas de función para la ventana álgebra:

F1 – ir a la siguiente ventana
F2 – retira ventanas superpuestas
F3 – inserta expresión resaltada
F4 – inserta expresión resaltada entre paréntesis
F5 – cambia al modo previo display

COMO SALIR DEL DERIVE

Ubicarnos en el comando **Quit** y dar Enter o, pulsar la tecla Q. A continuación el programa pregunta:
abandon expressions: (Y/N)
a lo cual si se le da Y, automáticamente salimos al sistema operativo. De lo contrario, vuelve al modo de espera.

COMO ESCRIBIR EN LA PILA:

Seleccionando **Author**, entramos lo que queramos entrar y damos **Return**.

COMO BORRAR TODO:

Transfer – Clear.

PARA BORRAR UNA PARTE:

Remove y damos desde qué línea queremos borrar hasta la última línea que vamos a omitir.

PARA TRASLADAR BLOQUES:

moVe y le decimos la dirección a donde vamos a llevar unas determinadas líneas de la pila.

COMO ENCONTRAR UN RACIONAL QUE COINCIDA CON UN IRRACIONAL EN LAS PRIMERAS n CIFRAS:

Options – Precision – Mode: Mixed entramos n **Return – Options – Notation – Style: Rational – Author** y escribimos el irracional. **Simplify**.

PARA ENCONTRAR n PRIMERAS CIFRAS DE UN IRRACIONAL:

Options - Precision - escribimos n - **Author** - escribimos el número - **ApproX** - número de la línea en que está el número.

Comandos Expand y Simplify:

Permite expandir expresiones algebráicas o numéricas. Por ejemplo, suma de fracciones, racionalización, descomposición en fracciones parciales, desarrollos de polinomios.

Comando Manage:

Permite simplificaciones con funciones exponenciales, logaritmicas, trigonométricas. Para ello se combinan los comandos:
Manage - Exponential - con una de las alternativas **Collect, Expand** o **Auto** (ensayar las 3 para notar la diferencia). Luego damos el comando **Author** para dar entrada a la expresión que queremos simplificar. Anexamos, en algunos casos el comando **Simplify**.
Manage - Logarithm - con las mismas alternativas del anterior: **Collect, Expand** y **Auto**. A continuación el comando para dar entrada a las expresiones: **Author** y posteriormente **Simplify**.
Manage - Trigonometry - con las mismas alternativas de los anteriores. A continuación se selecciona entre los comandos: **Auto, Sines** o **Cosines**. Después entramos la expresión a reducir por medio del comando **Author** y simplificamos: **Simplify**.

SOLUCION DE ECUACIONES:

Author y entramos la expresión a resolver, dando entonces el comando: **soLve** - y puede requerirse simplificación con **Simplify** o escribir la solución de manera aproximada: **ApproX**.

FACTORIZACION:

Author y a continuación entramos la expresión. Después el comando **Factor** y el número de la línea en que está la expresión que queremos factorizar.
En el caso de polinomios, se emplean los mismos comandos y a continuación se escoge el tipo de factorización: **Trivial, Squarefree, Rational** (sobre Q), **raDical** (irracionales) o **Complex**.

PARA DECLARAR VARIABLES Y CONSTANTES:

Declare - Variable y damos el nombre de la variable. Luego, el dominio: **Domain** escogiendo una opción: **Positive, Nonnegative, Real, Complex** o **Interval**. En ésta última opción es necesario dar los límites del intervalo.

Para declarar las constantes: **Declare - Variable** y damos el nombre, para por medio de **Value** entramos el valor.

FUNCIONES

Author y entramos f(...)=.... También se puede entrar: **Declare - Function** f y a continuación ....
Para obtener la función inversa, entramos la siguiente función: **INVERSA(f,x):=ITERATE(f,x,x,-1)**
donde f es la función a invertir, siendo explícita y dependiente de la variable x.

SUMATORIAS:

Author y escribimos SUM(f,k,n,m), donde f es la expresión a sumar, k la variable de la sumatoria y n y m los límites de la suma.
También se puede utilizar el comando **Calculus** seguido del comando **Sum**, para entrar la función f, la variable k, y los límites inferior y superior, respectivamente.

VECTORES Y MATRICES:

Para construir un vector: **Author** y entramos el vector de la forma: [a_1, a_2, ... a_n]. También con: **Declare - vectoR** y a continuación cada componente: a_1 <return>, a_2 <return>, ...
Una tercera forma de definir un vector es: **Author** y a continuación la función **VECTOR(f(i),i,m,n,s)**, donde i se reemplaza desde m hasta n en saltos de s en la expresión f(i) para obtener las distintas componentes del vector.
Luego, damos el comando **Simplify**.
Producto escalar: v . w
Producto vectorial: **cross** (v,w)
Elemento n-ésimo de un vector: **ELEMENT**(v,n)
Norma de un vector: **ABS**(v)
Para una matriz: **Author** y
[[a11,a21,...an1],[a12,a22,...an2],...,[a1m,a2m,...anm]]
También se puede: **Author** y
VECTOR(VECTOR(f(i,j),i,m,n,s),j,p,q,r) y luego **Simplify**.

Otra forma es declarando la matriz: **Declare - Matrix** y entramos el número de filas <return> y luego el número de columnas <return> siguiendo con los diferentes elementos separados por <return>.
Para operar con las matrices:
Suma= A+B;
Producto por escalar: α*A;
Producto A.B
Determinante de una matriz cuadrada: det(A)
Traza de una matriz cuadrada: Trace (A)
Traspuesta de una matriz: A`
Matriz inversa: A^(-1)
Matriz adjunta: La función **ADJOINT** (A) obtiene la adjunta de una matriz A.

La función **COFACTOR** (A,i,j) entrega el cofactor del elemento de la fila i columna j de A.
Reducción a la forma escalonada reducida el sistema de ecuaciones Ax=B, se le da la función **ROW_REDUCE** (A,B)
Para obtener el polinomio característico de A en función de x: se da la función **CHARPOLY** (A,x)
Para obtener los valores propios de una matriz cuadrada: utilizamos la función **EIGENVALUES** (A,x)
La función **EXACT_EIGENVECTOR** (A,σ) produce para el valor propio σ, la forma general que tendrán los correspondientes vectores propios.

Operaciones elementales entre las filas de una matriz:

La función **SWAP_ELEMENTS** (A,i,j) intercambia los elementos de las filas i y j de la matriz A.
La función **SCALE_ELEMENTS** (A,i,m) multiplica los elementos de la fila i por m.
La función **SUBSTRACT_ELEMENTS** (A,i,j) le resta a los elementos de la fila i los correspondientes elementos de la fila j.
La función **FORCEO** (A,i,m,j) le suma a los elementos de la fila i los correspondientes elementos de la fila j multiplicados por m.
La función **PIVOT** (A,i,j) anula todos los elementos de la columna j debajo de la fila i.
La función **RANGO_MATRIX** (A) reduce la matriz A a la forma escalonada reducida.

Solución de sistemas de ecuaciones

Entramos con **Author** y a continuación escribimos el sistema de ecuaciones. Luego, entramos el comando **soLve** para obtener la solución, especificando las distintas variables.

Cáculo con derive:

1. Cálculo de límites: Varias formas:
   a. **Author** y entramos la función. Seguimos con **Calculus** y **Limit**. Especificamos la variable y el valor a que tiende la variable con el comando **Point**. Luego se especifica entre las opciones **Both**, **Left** y **Right** si el límite no depende del camino, si es por la izquierda o si es por la derecha. Y entramos a simplificar con **Simplify**.

   b. Entrando directamente: **Lim(f(x₁,x₂,,...xₙ),x₁,a,0)** si el límite es independiente del camino; **Lim(f(x₁,x₂,,...xₙ),x₁,a,-1)** si es por la izquierda y **Lim(f(x₁,x₂,,...xₙ),x₁,a,1)** si es por la derecha.

2. Cálculo de derivadas: Varias formas:
   a. **Author** y entramos la función. Seguimos con **Calculus** y **Differenciate**, especificamos la variable y el número de veces que deseamos la derivada (el orden), y por último simplificamos, si es posible, con el comando **Simplify**.

   b. **Author** y entramos la función: **DIF** $(f(x_1,x_2,...x_n),x_1,k)$

Nota: Para aplicaciones de la derivada se carga el archivo DIF_APPS.MTH por medio de los comandos: **Transfer – Load**. Entre las aplicaciones que provee el Derive están: ecuación de la recta que pasa por un punto y de pendiente dada, ecuación de la recta tangente a una curva en un punto, ecuación de la recta normal, curvatura de una función en un punto, ecuación paramétrica de la circunferencia con centro y radio conocidos, ecuación paramétrica de la circunferencia de curvatura de f en un punto, centro de curvatura de f en x, las derivadas implícitas, curvaturas y centros de curvatura en ecuaciones dadas paramétricamente, ...

3. Cálculo de integrales: Varias formas:
   a. **Author** y entramos la función. Seguimos con **Calculus** y **Integrate**, especificamos la variable y damos <return> si la integral es indefinida o entramos los limites, si es definida. Por último simplificamos, si es posible, con el comando **Simplify**.

   b. **Author** y entramos una de laa funciones: **INT** $(f(x_1,x_2,...x_n),x_1)$, si la integral es indefinida ó **INT** $(f(x_1,x_2,...x_n),x_1,a,b)$ si la integral es definida entre a y b.

Nota: Para aplicaciones de la integral se carga el archivo INT_APPS.MTH por medio de los comandos: **Transfer – Load**. Entre las aplicaciones que provee el Derive están: longitud de arco, vector longitud de arco, volumen de un sólido de revolución, áreas de superficies de revolución, aproximación de series de Fourier, transformadas de Laplace, área bajo una curva, volumen de un sólido, área de una superficie, volumen de un sólido en coordenadas esféricas, volumen de un sólido en coordenadas cilíndricas, centroide de una región, integrales dobles y triples como áreas y volúmenes, ...