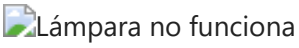
Lenguajes de Programación para Matemáticas

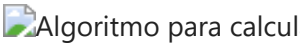
Un concepto muy importante que hay que tener en cuenta antes de comenzar a programar es el concepto de **algoritmo**. Un algoritmo es un conjunto de pasos, sentencias o reglas bien definidas que permite llevar a cabo una tarea determinada, ya sea solucionar un problema, realizar un cómputo, procesar datos y llevar a cabo otras tareas o actividades.

Según la RAE, es un conjunto ordenado y finito de operaciones que permite hallar la solución de un problema. Otras definiciones restringen el concepto de algoritmo al contexto de la Matemática o Ingeniería. El diccionario de Cambridge define algoritmo como: "a set of mathematical instructions or rules that, especially if given to a computer, will help to calculate an answer to a problem".

En nuestros días, el concepto de algoritmo se ha vuelto común y se aplica a muchas situaciones fuera del campo científico-técnico: una receta de cocina puede considerarse un algoritmo, una guía para usar un equipo electrodoméstico o para instalarlo, etc.

 Lámpara no funciona

Algunas características importantes de los algoritmos son la claridad, el orden de los pasos del mismo, que incluya todas las posibilidades a tener en cuenta y que ofrezca solución a todas y que permita realizarlo en un tiempo finito.

 Algoritmo para calcular la raíz cuadrada de un número

Para poder comunicarnos con los ordenadores y otros equipos informáticos se han desarrollado diferentes lenguajes artificiales con reglas gramaticales bien definidas que permiten a las personas comunicarse con el ordenador y trasmitirle una serie de instrucciones u órdenes para que dicho equipo realice una tarea determinada bajo ciertas condiciones.

Al conjunto de órdenes expresadas mediante un lenguaje de programación se le denomina programa informático o simplemente programa.

No se debe confundir 'lenguaje de programación' y 'lenguaje informático'. Lenguaje informático es un término más general que engloba a los lenguajes de programación y a otros más, como por ejemplo HTML (lenguaje para el marcado de páginas web, que no es propiamente un lenguaje de programación, sino un conjunto de sentencias que permiten estructurar el contenido de los documentos). Existen diferentes tipos de lenguajes relacionados con los ordenadores:

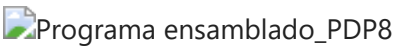
- Lenguaje de programación: un lenguaje formal diseñado para comunicar instrucciones a una máquina, particularmente a una computadora
- Lenguaje de consulta: un lenguaje utilizado para realizar consultas y búsquedas en bases de datos y sistemas de información.

- Lenguaje de marcado: un lenguaje para anotar un documento de manera que las anotaciones se diferencien sintácticamente del texto, como HTML y permitan un mejor uso del texto tanto para los humanos como para las máquinas.

entre otros muchos.

Lenguaje de Máquina

El lenguaje de máquina, que se utiliza para controlar la unidad central de procesamiento (CPU) de la computadora, es un lenguaje de programación de bajo nivel que puede ser comprendido directamente por el hardware de la computadora. Cada instrucción hace que la CPU realice una tarea muy específica, como cargar o almacenar un dato, realizar un cálculo con valores contenidos en la memoria o los registros de la CPU, utilizando para ello la unidad aritmético-lógica (ALU), tomar una decisión en dependencia de los valores calculados, etc. El lenguaje de máquina es estrictamente numérico y es la interfaz de más bajo nivel con la CPU a la que tiene acceso un programador.



En la imagen anterior se muestra un programa en PAL III (Program Assembly Language version III) para la familia de computadoras PDP-8. El PDP-8 (Programmed Data Processor - 8) fue un miniordenador de 12 bits creada por Digital Equipment Corporation (DEC) en abril de 1965, de la serie PDP. Fue la primera minicomputadora comercialmente exitosa, con más de 50.000 unidades vendidas durante la vida útil del modelo.

En la imagen se aprecian varias columnas. En la columna de la extrema izquierda aparecen numeradas en forma secuencial las localizaciones de la memoria de una PDP-8 4K en el año 1970. En la segunda columna de izquierda a derecha podemos observar los códigos en lenguaje de máquina de las instrucciones que aparecen en la siguiente columna.

Si bien es posible escribir programas directamente en código de máquina, es tedioso y propenso a errores. Por esta razón, los programas rara vez se escriben directamente en código de máquina en contextos modernos.

Debido a las dificultades antes mencionadas, ya desde la década del 40 del siglo XX se comenzaron a definir y desarrollar lenguajes, cuyas sentencias representaran expresiones matemáticas en una forma más comprensible.

Lenguaje Ensamblador

El lenguaje ensamblador (en inglés: assembler language y la abreviación asm) es un lenguaje de programación que se usa en los microprocesadores. Implementa una representación simbólica de los códigos de las instrucciones de máquina y otras constantes necesarias para programar una arquitectura de procesador y es la representación más directa del código máquina.

Cada arquitectura de procesador es definida por el fabricante de hardware y tiene asociado su propio lenguaje de máquina y su correspondiente lenguaje ensamblador que está basado en los códigos mnemotécnicos que simbolizan los pasos de procesamiento (las

instrucciones), los registros del procesador, las posiciones de memoria y otras características del lenguaje.

Un lenguaje ensamblador es por lo tanto específico de cierta arquitectura de computador física (o virtual), a diferencia de lo que ocurre con la mayoría de los lenguajes de programación de alto nivel, que idealmente son portables.

El programa diseñado en lenguaje ensamblador es trasladado a lenguaje de máquina (en binario) mediante el programa ensamblador, el cual se carga directamente en la memoria de la computadora para ser ejecutado

Ejemplo de programa en lenguaje ensamblador Intel 8086. El programa busca en una cadena un 0 y si lo encuentra rellena las siguientes 5 posiciones con ceros.

LEA DI, ES:CADENA; carga en el registro DI la dirección inicial de la cadena

MOV AX, 0; borra (pone a cero) el registro AX

MOV CX, 200; asigna al registro CX el valor 200

REPNE SCASW; si CX es distinto de 0, ejecuta SCASW y decrementa CX

JCXZ no_encon; si CX = 0, salta a la instrucción con la etiqueta 'no_encon'

SUB DI, 2; substrae (resta) 2 al registro DI

MOV CX, 6; coloca el valor 6 en el registro CX

REP STOS CADENA; si CX es distinto de 0, ejecuta STOS y decrementa CX

no_encon: ... ...

; STOS transfiere el contenido de AL a una cadena (destino).

; SCASW localiza el valor contenido en AX en una cadena; si encuentra el elemento, devuelve en DI el desplazamiento del siguiente elemento.

Ejemplo de programa en lenguaje ensamblador Motorola 6809. El MC6809 es un procesador creado por Motorola en 1977. Es un procesador de 8 bits, con algunas características de 16 bits.

El programa imprime en la pantalla las palabras Hola mundo.

; Programa de ejemplo: hola.asm

```
        .area PROG (ABS)

        .org 0x100


cadena: .ascii "Hola, mundo."

        .byte   10      ; 10 es CTRL-J: salto de lInea
```

```
        .byte   0       ; 0 es CTRL-@: fin de cadena

        .globl programa
```

programa:

```
        ldx #cadena     ; carga en el registro X la dirección de la
    cadena
```

bucle: lda ,x+ ; carga en el registro A la dirección de la cadena

```
        beq acabar      ; salta si cero a la etiqueta 'acabar'

        sta 0xFF00      ; salida por pantalla

        bra bucle       ; salta a la etiqueta 'bucle'
```

acabar: clra ; borra (clear) a

```
        sta 0xFF01      ; salida por pantalla

        .org 0xFFFE     ; Vector de RESET

        .word programa
```

FORTRAN

En 1954, John Backus inventó el lenguaje FORTRAN mientras trabajaba en IBM. En aquellos años, los ordenadores no permitían escribir letras minúsculas, por lo cual se acostumbró a escribir el nombre en mayúsculas. El primer manual de FORTRAN lo describe como un sistema de traducción de fórmulas (Formula Translating System).

Fortran fue el primer lenguaje de programación de propósito general de alto nivel ampliamente utilizado en tener una implementación funcional, en lugar de solo un diseño en papel. Sigue siendo un lenguaje popular para la computación de alto rendimiento y se usa para programas que comparan y clasifican las supercomputadoras más rápidas del mundo.

 IBM Blue Gene/P supercomputer

Fortran ha estado en uso durante más de seis décadas en áreas computacionalmente intensivas como la predicción meteorológica numérica, el análisis de elementos finitos, la dinámica de fluidos computacional, la geofísica, la física computacional, la cristalografía y la química computacional. Es un lenguaje popular para computación de alto rendimiento y se usa para programas que comparan y clasifican las supercomputadoras más rápidas del mundo.

Desde agosto de 2021, Fortran se encuentra entre los quince principales idiomas en el índice TIOBE, el cual mide la popularidad de los lenguajes de programación.

En julio de 2023, se espera que se publique la nueva versión standard de Fortran.

Ejemplo

El siguiente programa escrito en FORTRAN 90 calcula un promedio sobre los datos ingresados de forma interactiva. Este ejemplo ilustra la asignación de memoria dinámica y las operaciones basadas en matrices. Conviene destacar la ausencia de bucles (lazos, ciclos) DO y sentencias IF/THEN en la manipulación de la matriz; las operaciones matemáticas se aplican a la matriz como un todo. También es evidente el uso de nombres de variables descriptivos y formato de código general que se ajusta al estilo de programación contemporáneo.

```
program average

 ! Read in some numbers and take the average

 ! As written, if there are no data points, an average of zero is returned

 ! While this may not be desired behavior, it keeps this example simple

 implicit none

 real, dimension(:), allocatable :: points

 integer :: number_of_points

 real :: average_points, positive_average, negative_average

 average_points = 0.0

 positive_average = 0.0

 negative_average = 0.0

 write (,) "Input number of points to average:"

 read (,) number_of_points

 allocate (points(number_of_points))

 write (,) "Enter the points to average:"

 read (,) points

 ! Take the average by summing points and dividing by number_of_points

 if (number_of_points > 0) average_points = sum(points) / number_of_points

 ! Now form average over positive and negative points only

 if (count(points > 0.) > 0) positive_average = sum(points, points > 0.) / count(points > 0.)
```

```
  if (count(points < 0.) > 0) negative_average = sum(points, points < 0.) / count(points < 0.)

  ! Print result to terminal stdout unit 6

  write (*,'(a,g12.4)') 'Average = ', average_points

  write (*,'(a,g12.4)') 'Average of positive points = ', positive_average

  write (*,'(a,g12.4)') 'Average of negative points = ', negative_average

  deallocate (points) ! free memory

end program average
```

MATLAB

MATLAB fue inventado por el matemático y programador informático Cleve Moler. La idea de MATLAB se basó en su tesis doctoral de la década de 1960. Moler se convirtió en profesor de matemáticas en la Universidad de Nuevo México y comenzó a desarrollar MATLAB para sus alumnos como pasatiempo. Desarrolló la versión inicial de los programas de álgebra lineal de MATLAB en 1967 con su antiguo asesor de tesis, George Forsythe.

La primera versión temprana de MATLAB se completó a fines de la década de 1970. El software se divulgó al público por primera vez en febrero de 1979. En ese momento, MATLAB se distribuía de forma gratuita a las universidades. Moler dejaba copias en las universidades que visitaba y el software desarrolló muchos seguidores en los departamentos de matemáticas de los campus universitarios.

En la década de 1980, Cleve Moler y John N. Little decidieron reprogramar MATLAB utlizando el lenguaje C y comercializarlo para las computadoras de escritorio de IBM. De esta manera, crearon el lenguaje de programación MATLAB y desarrollaron funciones para cajas de herramientas.

MATLAB se lanzó por primera vez como producto comercial en 1984. Se fundó MathWorks, Inc. para desarrollar el software y se lanzó el lenguaje de programación MATLAB. A fines de la década de 1980, se habían vendido varios cientos de copias de MATLAB a universidades para uso de los estudiantes. Muchas de las cajas de herramientas se desarrollaron como resultado del uso de MATLAB por los estudiantes de Stanford, los cuales difundieron así el software al sector privado.

MATLAB (abreviatura de "MATrix LABoratory") es un lenguaje de programación multiparadigma y un entorno de computación numérica patentado por MathWorks. MATLAB permite la manipulación de matrices, el trazado de funciones y datos, la implementación de algoritmos, la creación de interfaces de usuario y la interfaz con programas escritos en otros lenguajes.

Aunque MATLAB está destinado principalmente a la computación numérica, una caja de herramientas opcional utiliza el motor simbólico MuPAD que permite computación simbólica. Otro paquete adicional, Simulink, agrega simulación gráfica multidominio y diseño basado en modelos para sistemas dinámicos e integrados.

En 2000, MathWorks agregó una biblioteca basada en Fortran para álgebra lineal, reemplazando las subrutinas LINPACK y EISPACK originales que estaban en C. La caja de herramientas de computación paralela de MATLAB se lanzó en 2004 y se le agregó soporte para unidades de procesamiento de gráficos (GPU) en 2010.

A partir de 2020, MATLAB tiene más de 4 millones de usuarios en todo el mundo de ingeniería, ciencia y economía.

Ejemplo

MATLAB tiene funciones de gráficación integradas. Por ejemplo, la función $plot$ permite crear un gráfico de la función seno a partir de dos vectores x e y con facilidad:

```
x = 0:pi/100:2*pi;

y = sin(x);

plot(x,y)
```


Matlab sin plot

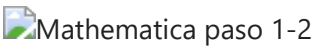
Wolfram Mathematica

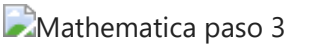
Wolfram Mathematica es un sistema de software con bibliotecas integradas para varias áreas de computación tales como aprendizaje automático, estadísticas, computación simbólica, análisis de redes y de series de tiempo, procesamiento de lenguaje natural, optimización, implementación de algoritmos, creación de interfaces de usuario e interfaz con programas escritos en otros lenguajes de programación.

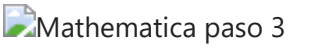
Fue concebido por Stephen Wolfram y desarrollado por Wolfram Research. Wolfram Language es el lenguaje de programación utilizado en Mathematica. Mathematica 1.0 fue lanzado el 23 de junio de 1988.

Wolfram Language es un lenguaje de programación multiparadigma patentado, general y de alto nivel que hace hincapié en la computación simbólica, la programación funcional y la programación basada en reglas y puede emplear estructuras y datos arbitrarios. Los aspectos simbólicos lo convierten en un sistema de álgebra computacional. El lenguaje puede realizar integración, diferenciación, manipulaciones de matrices y resolver ecuaciones diferenciales utilizando un conjunto de reglas.

Ejemplo de operaciones simbólicas para simplificar una expresión aritmética:


Mathematica paso 1-2


Mathematica paso 3


Mathematica paso 3

Características de los Lenguajes de Programación

Los lenguajes de programación están constituídos por un alfabeto (conjunto de símbolos), reglas gramaticales (léxico/morfológicas y sintácticas) y semánticas, que en su conjunto

definen las estructuras válidas del lenguaje y su significado.

El lenguaje de programación permite especificar de manera precisa sobre qué datos debe operar un software específico, cómo deben ser almacenados o transmitidos dichos datos, y qué acciones debe tomar el software bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural.

Una característica relevante de los lenguajes de programación es que diferentes programadores pueden usar y comprender un mismo conjunto de sentencias para elaborar programas de forma colaborativa.

Variables

Las variables son nombres asignados a espacios en memoria para almacenar datos específicos. Las variables constan de dos elementos fundamentales: su identificador y su contenido. Según el contenido que almacene la variable, el espacio de memoria que ocupa la misma será mayor o menor.

El identificador de una variable permite hacer referencia a la misma, mencionarla como parte del programa. El identificador de una variable es lo que nos permite acceder a su contenido para utilizarlo o cambiarlo.

El contenido de una variable puede ser de diferentes tipos. Los tipos de contenido más comunes son:

- Char: Estas variables contienen un único carácter, es decir, una letra, un signo o un número.
- Int: Contienen un número entero.
- Float: Contienen un número decimal.
- String: Contienen cadenas de texto, o lo que es lo mismo, es un vector con varias variables del tipo Char.
- Boolean: Solo pueden contener un valor lógico False o True que usualmente se representan mediante un cero o un uno respectivamente.

Al igual que las variables de tipo string hacen referencia a un vector de variables de tipo char, podemos tener variables que hacen referencia a vectores de valores enteros, reales, booleanos, entre otros.

En algunos lenguajes de programación se requiere asociar cada variable a un tipo específico de datos. De esta manera, a la variable se le asigna una cantidad de memoria específica acorde con el tipo de dato que va a almacenar y se verifica en todo momento que las operaciones donde se use la variable sean acordes con su tipo de dato. Estos lenguajes se denominan tipados, aunque hay distintas categorías de tipado. Un lenguaje es fuertemente tipado "si no se permiten violaciones de los tipos de datos"

Por ejemplo, a los valores lógicos o de caracteres no tiene sentido aplicarles operadores aritméticos (i.e., suma, resta, mutiplicación, etc.) Por tanto, una variable de tipo char no se puede utilizar en expresiones aritméticas.

Por otra parte, un lenguaje débilmente tipado tiene reglas de tipeo menos estrictas y puede producir resultados impredecibles o incluso erróneos o puede realizar una conversión de tipo implícita en tiempo de ejecución. Los lenguajes de programación no tipados no controlan los tipos de las variables que declaran, de este modo, es posible usar variables de cualquier tipo en un mismo escenario.

Las diferentes formas de tipado presentan ventajas y desventajas particulares, por lo que hay una gran diversidad de tipados según unos u otros lenguajes. Por esta razón, hay que prestarle especial atención a la forma de tipado del lenguaje que estemos utilizando.

Sentencias Condicionales

Las sentencias condicionales son estructuras de código que permiten expresar ciertas premisas que deben cumplirse para que una cierta parte del programa se ejecute. Por ejemplo: una condición puede ser que dos valores sean iguales o que un valor exista o que un valor sea mayor que otro, etc.

Las sentencias condicionales más usuales en programación son:

- If: Indica una condición para que se ejecute una parte del programa.
- Else if: Siempre va precedido de un "If" e indica una condición para que se ejecute una parte del programa siempre que no cumpla la condición del if previo y sí se cumpla la condición especificada por el "else if".
- Else: Siempre precedido de "If" y en ocasiones de "Else If". Indica que debe ejecutarse cuando no se cumplan las condiciones previas.

Ejemplo de programa utilizando la sentencia IF

Sean a, b y c variables que contienen la número de grados que forman cada uno de los tres ángulos de un triángulo. Se quiere saber si el triángulo es equilátero o no.

%verificamos si el ángulo a es recto (= 90 grados)

if a == 90

```
  % si la condición se cumple imprimimos que es un triángulo
rectángulo

  fprintf('El triángulo es rectángulo\n' );
```

elseif b == 90 %verificamos si el ángulo b es recto (= 90 grados)

```
  % si la condición se cumple imprimimos que es un triángulo
rectángulo
```

```
  fprintf('El triángulo es rectángulo\n' );
```

elseif c == 90 %verificamos si el ángulo c es recto (= 90 grados)

```
  % si la condición se cumple imprimimos que es un triángulo
rectángulo

  fprintf('El triángulo es rectángulo\n' );
```

else

```
  % si ninguno de los tres ángulos es recto, entonces no es
rectángulo

  fprintf('El triángulo NO es rectángulo\n');
```

end

Una versión más compacta del programa anterior sería la siguiente:

```
if a == 90 | b == 90 | c == 90 % si (a=90 ó b=90 ó c=90)

  % si la condición se cumple imprimimos que es un triángulo
rectángulo

  fprintf('El triángulo es rectángulo\n' );

else

  % si ninguno de los tres ángulos es recto, entonces no es
rectángulo

  fprintf('El triángulo NO es rectángulo\n');

end
```

Sentencias Bucle (Lazo, Ciclo)

Las sentencias de tipo bucle (loops en inglés) permiten conformar una serie de sentencias cualesquiera, las cuales se ejecutarán durante un número específico de repeticiones o hasta que se cumplen las condiciones especificadas. Estas sentencias se presentan en dos formas principales:

- los bucles controlados mediante un contador que determina el número de repeticiones máximo que se ejecutará el bucle. Se denominan bucles tipo FOR, ya que es el nombre que recibe este tipo de sentencia en muchos lenguajes
- los bucles controlados mediante una condición, la cual determina si se repite de nuevo el bucle o no. Se denominan bucles tipo WHILE por

Un bucle tipo FOR es una sentencia que controla el flujo del programa y permite especificar una sección del código que se ejecutará una y otra vez hasta que el contador alcance un determinado valor. Por ello se dice que el contador controla el bucle.

Los bucles tipo FOR tienen dos partes: un encabezado y un cuerpo. El encabezado define las condiciones de las iteraciones y el cuerpo es el código que se ejecuta una vez por cada iteración. El encabezado a menudo declara un contador de bucle explícito o una variable de bucle. Esto permite que el cuerpo sepa qué iteración se está ejecutando. Los bucles for se utilizan normalmente cuando se conoce el número de iteraciones antes de entrar en el bucle.

Ejemplo de bucle tipo FOR en MATLAB

El programa a continuación permite calcular el factorial del número 5.

```
counter = 5; %valor del cual se quiere obtener el factorial

factorial = 1; %variable auxiliar para ir acumulando los productos

% la variable n toma los valores desde 1 hasta counter sucesivamente.

for n = 1:counter % para cada uno de dichos valores se repite el bucle

    %en cada iteración se multiplica la variable auxiliar por n

    %y el resultado se copia en la variable auxiliar

    factorial = factorial * n

    %al llegar al 'end' del bucle, se incrementa el valor de n y

    %se verifica que sea menor o igual que counter, en cuyo caso

    %se repite el bucle. Si n > counter, se termina el bucle


end

factorial %al terminar el bucle, la variable contiene el valor del factorial
```

Un bucle tipo WHILE es una sentencia de control de flujo que permite que el código se ejecute repetidamente en función de una condición booleana determinada. El bucle tipo WHILE se puede considerar como una sentencia IF que se repite una y otra vez mientras l condición es verdadera.

Los bucles tipo WHILE tienen dos partes: una condición (o expresión) y un bloque de código. Se evalúa la condición/expresión, y si la condición/expresión es verdadera, se ejecuta el bloque de código. Esto se repite hasta que la condición/expresión se vuelve falsa, i.e. deje de ser verdadera.

Ejemplo de bucle tipo WHILE en MATLAB

El programa a continuación permite calcular el factorial del número 5.

counter = 5; %valor del cual se quiere obtener el factorial

factorial = 1; %variable auxiliar para ir acumulando los productos

% mientras el counter sea mayor que cero se repite el bloque del bucle

while (counter > 0) %si la condición no se cumple, se termina el bucle

```
%en cada iteración se multiplica la variable auxiliar por el
counter

%y el resultado se copia en la variable auxiliar

factorial = factorial * counter;

%a continuación se decrementa el contador

counter = counter - 1;
```

end

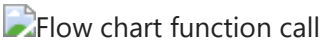
factorial %al terminar el bucle, la variable contiene el valor del factorial

Subprogramas

Un subprograma es un segmento de código que realiza una tarea específica y está empaquetado como una unidad. El subprograma se puede definir dentro de un programa determinado o formar parte de bibliotecas de subprogramas que pueden ser utilizadas a su vez por diferentes programas. El subprograma puede ser invocado en cualquier momento desde el programa principal o desde otro subprograma.

Los subprogramas reciben diferentes nombres tales como subrutina, procedimiento, función, rutina o método, según el lenguaje de programación o el contexto en que se utilice el término

Un subprograma, al ser llamado dentro de un programa, hace que la ejecución del código principal se detenga y se dirija a ejecutar el código del subprograma.

 Flow chart function call

Los subprogramas se crearon para evitar tener que repetir constantemente fragmentos de código. Un subprograma podría considerarse como una variable que encierra código dentro de si. Por lo tanto, cuando accedemos a dicha variable en realidad lo que estamos haciendo es ordenar al programa que ejecute un determinado código predefinido anteriormente.

La idea de una subrutina fue inicialmente concebida por John Mauchly y Kathleen Antonelli durante su trabajo en ENIAC y registrada en un simposio de Harvard de enero de 1947.

 ENIAC

Uno de los primeros lenguajes de programación que admitió subrutinas y funciones escritas por el usuario fue FORTRAN II, cuyo compilador para IBM fue lanzado en 1958.

El siguiente programa muestra como escribir una función externa para calcular el promedio de tres números

```
      PROGRAM FUNDEM


C Declarations for main program

      REAL A,B,C

      REAL AV, AVSQ1, AVSQ2

      REAL AVRAGE


C Enter the data

      DATA A,B,C/5.0,2.0,3.0/


C Calculate the average of the numbers

      AV = AVRAGE(A,B,C)

      AVSQ1 = AVRAGE(A,B,C) **2

      AVSQ2 = AVRAGE(A**2,B**2,C**2)

      PRINT *,'Statistical Analysis'

      PRINT *,'The average of the numbers is:',AV

      PRINT *,'The average squared of the numbers: ',AVSQl

      PRINT *,'The average of the squares is: ', AVSQ2

      END

     REAL FUNCTION AVRAGE(X,Y,Z)

     REAL X,Y,Z,SUM

     SUM = X + Y + Z

     AVRAGE = SUM /3.0

     RETURN

     END
```

Bibliografía

- Algoritmo
- Lenguaje de programación
- Fortran
- MATLAB
- MATLAB
- Wolfram Mathematica
- Strong and weak typing
- Tipado fuerte
- Sentencia condicional
- Bucle for
- For loop
- While loop
- Bucle while
- Statement (computer science)
- Introduction to Programming May70
- PDP-8
- Subutina
- Function (computer programming))
- Funciones y Subrutinas
- Ensamblador 8086/88
- Motorola 6809
- Motorola 6809
- Intel 8086 y 8088

In [ ]: