UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

A P L

UN NUEVO CONCEPTO EN LENGUAJES INTERACTIVOS

T E S I S

QUE PARA OBTENER EL TITULO DE

A C T U A R I O

PRESENTA

JOSE LUIS GARCIA LUNA MARTINEZ

MEXICO, D.F. 1979

INTRODUCCION

Es indiscutible que en el campo de la tecnología actual la computación es una de las áreas que tiene un desarrollo más dinámico, ya que constantemente se están haciendo innovaciones.

Tal es el caso del teleproceso interactivo que es, sin duda, uno de los adelantos más significativos, ya que permite procesar información de manera remota por medio de terminales conectadas a un computador central por líneas telefónicas comunes.

Los lenguajes de programación usados actualmente en este tipo de proceso se pueden dividir en dos grupos básicos:

1.- Aquellos que originalmente fueron creados para manejar procesos BATCH * y que han sido adoptados a esta nueva forma de manejo de información.

2.- Aquellos que desde su concepción fueron planeados pensando en el concepto de teleproceso interactivo.

Del primer grupo podemos mencionar algunos como FORTRAN Y COBOL, y del segundo a APL, siendo éste el lenguaje más poderoso y eficiente que existe actualmente para este tipo de proceso.

Aunque originalmente APL fue creado para resolver algoritmos complejos, en la actualidad se ha encontrado que este lenguaje puede ser empleado como una herramienta ideal para resolver problemas de investigación de operaciones, estadísticas y, en general, de cualquier área que implique cálculos numéricos desde los más simples hasta los más complejos.

Este trabajo tiene como objetivo presentar el funcionamiento del lenguaje, así como su aplicación a la Planeación Financiera de una empresa manufacturera.

Con objeto de seguir una secuencia lógica, este trabajo ha sido estructurado en cuatro partes, que son:

I ANTECEDENTES
II EL LENGUAJE APL
III COMUNICACION CON EL SISTEMA QUE SOPORTA A APL
IV APL APLICADO A LA PLANEACION FINANCIERA DE UNA EMPRESA MANUFACTURERA

A continuación se hace una descripción general de cada una de estas partes:

ANTECEDENTES

En esta parte se presentan los antecedentes del teleproceso, así como una clasificación de los diferentes tipos del mismo.

EL LENGUAJE APL

Aquí se hace una descripción de las reglas de sintáxis de APL, así como de cada uno de los operadores que la componen.

Esta parte está compuesta por seis secciones que contienen los siguientes puntos:

1a. Sección: Contiene los conceptos básicos del lenguaje, tales como modo de empleo, las operaciones básicas, reglas de sintáxis,

formas de construcción, los tipos de operadores existentes, el concepto de variables y los tipos de errores existentes.

2a. Sección: Contiene las indicaciones para crear y manejar arreglos.

3a. Sección: Se refiere a las funciones u operadores más comúnmente empleados en operaciones con arreglos elemento a elemento.

4a. Sección: Se refiere a operadores cuya función involucra operaciones con arreglos completos, tal es el caso de inversiones de matrices o de multiplicación de matrices, desde el punto de vista de las matemáticas.

5a. Sección: Se refiere a la definición y adición de funciones y contiene todas las indicaciones necesarias para la elaboración, corrección e impresión de las mencionadas funciones.

6a. Sección: En esta sección se presenta un grupo de operadores, los cuales están enfocados a seleccionar partes específicas de arreglos previamente definidos o a cambiar la estructura de los mismos.

COMUNICACION CON EL LENGUAJE QUE SOPORTA A APL

Se presenta la relación que mantiene el lenguaje APL con el sistema que lo soporta, así como con la biblioteca y el área de trabajo donde funciona.

Esta parte a su vez está subdividida en ocho puntos, que son:

a) Contenido de las Areas de Trabajo

El área donde el usuario puede trabajar con APL se conoce como área de trabajo; en este punto se analizan los elementos que pueden estar contenidos dentro de las áreas de trabajo y que son las variables y las funciones; así mismo, se presenta cómo estos elementos pueden quedar agrupados bajo un mismo nombre, cómo pueden ser copiados de otras áreas de trabajo o bien, cómo borrarlos cuando se requiera.

b) La Biblioteca

Se refiere al lugar donde están contenidas las áreas de trabajo dentro del computador, así como la posibilidad de hacer ciertos movimientos con ellas, tales como: borrarlas, llamarlas, salvarlas, copiarlas o cambiarles el nombre o la clave.

c) El Area de Trabajo; la Unidad de Trabajo de APL

En este punto se presentan las características internas de las áreas de trabajo en cuanto a su dimensión, tabla de símbolos, despliegue de decimales, impresión. Así como las variables del sistema que sirven para modificarlas.

d) Entrada y Salida

Existen distintas posibilidades de conectarse o desconectarse del sistema que soporta a APL; en este punto, se analizan cada una de ellas, así como sus implicaciones.

e) Mensajes

APL nos provee de la capacidad de enviar o recibir mensajes, tanto del operador como de otros usuarios que estén trabajando simultáneamente desde otras terminales; en este punto se explican las distintas alternativas existentes.

f) Reportes de Problemas

Se presenta una tabla que contiene los reportes de problemas existentes, los cuales están asociados con comandos del sistema y una serie de sugerencias para resolver estos problemas.

g) Información del Sistema

Se presentan aquellas variables cuya función es proveernos de información de las características del procesador o de la terminal en la que estamos trabajando.

h) Las Funciones del Sistema

Se muestran aquellas funciones del sistema que manejan funciones que han sido definidas por el usuario afectando su ejecución normal. También se presenta el vector con los elementos del código-z, código en el cual opera APL.

APL APLICADO A LA PLANEACION FINANCIERA DE UNA EMPRESA MANUFACTURERA

En este capítulo se presenta la aplicación de este lenguaje a la Planeación Financiera de una empresa manufacturera, enfocándose principalmente a la obtención de los costos de producción y ventas de los

equipos producidos en ellas; así como, las ventajas que se obtienen por la utilización de un sistema de este tipo.

CONCLUSIONES

Por último se presentan las conclusiones obtenidas de esta investigación.

TABLA DE CONTENIDO

ANTECEDENTES

ANTECEDENTES

Es de todos sabido que a partir de la Segunda Guerra Mundial el desarrollo tecnológico, en general, ha sido explosivo; ya que lo que se ha logrado en estos últimos cuarenta años es comparable con el avance logrado en los siglos anteriores.

Uno de los campos que se ha visto más beneficiado por este desarrollo acelerado es, sin duda alguna, el campo de procesamiento y manejo de información.

Este campo se ha diversificado en tal forma que lo mismo encontramos computadoras que sirven para elaborar los cálculos necesarios para controlar vuelos espaciales, que microcomputadoras de bolsillo que empleamos en cálculos matemáticos simples.

Actualmente, las computadoras se encuentran clasificadas en tres grandes grupos que son:

a) Computadoras de bolsillo

b) Minicomputadoras o de mediana capacidad

c) Computadoras o de gran capacidad

Las computadoras de bolsillo son pequeñas máquinas manuables que sirven para resolver, por medio de programas sencillos, problemas matemáticos simples como cálculos de regresión lineal, modelos simples de investigación de operaciones, cálculos de valor presente, tablas de amortización, etc. Estas computadoras no tienen normalmente capacidad de almacenamiento de datos, o cuando la tienen, ésta está muy restringida.

Su principal característica es que su manejo es muy simple y que están al alcance de cualquier persona.

Las minicomputadoras son procesadoras que tienen un campo de aplicación más amplio, ya que aunado a la capacidad de las computadoras de bolsillo ya cuentan con un sistema formal, aunque aún restringido, de almacenamiento de datos.

Este tipo de computador se utiliza principalmente en procesos administrativos como pueden ser el cálculo de la nómina de empleados, control de inventarios, control de las cuentas por cobrar, facturación, etc.; aunque también existen con aplicaciones científicas.

Para el manejo de este tipo de computador, es necesario contar con un grupo de gentes especializadas, ya que por un lado se requiere de un operador de la máquina y por otro, un grupo que se encargue de análisis y la programación de los sistemas.

En este último punto, nos encontramos con que si queremos obtener un resultado específico del computador vamos a necesitar de terceras personas para lograrlo, ya que este grupo de gente especializada será al que le tengamos que explicar cuál es el proceso y los resultados deseados para que ellos se encarguen del análisis y la programación del mismo. Aquí aparece un problema de dependencia que va a mermar la utilización del computador o por lo menos va a retrasar la implementación de nuestros sistemas.

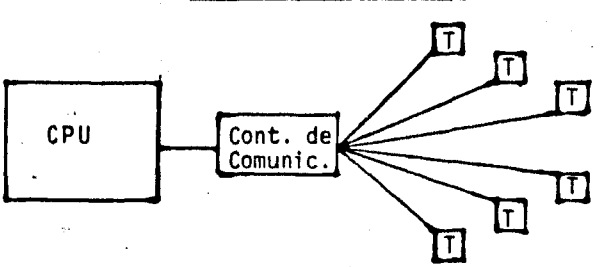
Este tipo de computador está enfocado a pequeñas y medianas empresas, cuyas necesidades de procesamiento no requieren de una gran capacidad de máquina.

Por último, nos encontramos con los computadores que pueden procesar grandes volúmenes de información y que cuentan con una gran capacidad de almacenamiento de datos.

Este tipo de computador está capacitado para realizar, además de procesos administrativos, procesos científicos muy sofisticados. El principal inconveniente que presentan, aunado al de la dependencia de un departamento de sistemas, es el de la centralización ya que normalmente las empresas que cuentan con este tipo de computador, cuentan también con fuentes de información dispersas, ya sea en una misma localidad o en varias, lo que tiene como consecuencia la necesidad de trasladar dicha información de manera física para ser procesada, lo cual tiene su impacto en eficiencia y tiempo. En base a lo anterior, se crea el concepto de teleproceso.

El teleproceso es el que nos permite procesar información de manera remota para lo cual se utilizan terminales conectadas a un computador central por medio de líneas telefónicas,

TELEPROCESO



lo cual viene a eliminar el problema de centralización y dependencia, ya que si en un momento determinado deseamos procesar u obtener información, lo único que tendremos que hacer es utilizar nuestra terminal bajo el entendido de que esto está conectado a un gran computador central que cuenta

con una capacidad casi ilimitada de memoria. También nos permitirá, en un momento dado, consultar o actualizar información que esté contenida en el archivo de esta gran máquina.

El teleproceso se divide en dos grandes grupos:

a) Teleproceso de datos
b) Teleproceso interactivo

El teleproceso de datos es aquel que aunque se haga desde una terminal, el computador central lo manejará como un proceso común; esto es, tendrá una cierta prioridad de ejecución, o sea, que tendrá que formar una cola de espera para ser procesado. La forma en que se introducirá la información será por medio de tarjetas perforadas o de diskettes y normalmente el resultado se obtendrá por una impresora de baja o mediana velocidad.

El teleproceso interactivo no hace cola de espera sino que interactúa directamente con el precesador central bajo el concepto de pregunta y respuesta. Para este tipo de proceso se emplean normalmente terminales de video con un teclado o parecidas a máquinas de escribir; introduciéndose la información por medio del teclado y obteniendo el resultado en la pantalla o en pequeños listados, según sea el caso.

El proceso interactivo está dividido en dos modalidades:

a) Para consulta o actualización de bancos de información
b) Para proceso utilizando lenguajes interactivos

La modalidad de consulta y actualización de bancos de información es muy fácil de manejar y tiene un gran alcance, ya que por medio de este tipo de teleproceso podremos conectarnos a bancos de datos que hayan sido creados previamente y consultar o actualizar la información que en ellos esté contenida.

El teleproceso, utilizando lenguajes interactivos, nos va a permitir el desarrollo de cálculos y sistemas de una manera ágil y eficiente, ya que como vimos anteriormente, este tipo de procesos no entra en cola de espera sino que es procesado al momento, lográndose así el concepto de interactivo.

Existen en la actualidad una serie de lenguajes que aunque originalmente fueron creados bajo el concepto no interactivo han sido adaptados a esta nueva modalidad, debido básicamente a las grandes ventajas que esto representa; algunos ejemplos de estos lenguajes son: FORTRAN, BASIC, COBOL, etc.

Sin embargo, al no ser concebidos de manera original como interactivos, sino adaptados posteriormente, tienen problemas en su estructura que hacen que su programación sea complicada y laboriosa.

En 1962, IBM lanzó al mercado un lenguaje de computación que fue concebido desde su inicio como un lenguaje netamente interactivo, el cual fue clasificado por los especialistas de computación como el adelanto más significativo de los últimos años en materia de programación. Este lenguaje es APL y es tan fácil de manejar que cualquier persona que cuente con conocimientos básicos de matemáticas, puede aprender y utilizarlo como una herramienta invaluable en su trabajo cotidiano.

El objeto de este trabajo es presentar ¿qué es?, ¿cómo funciona? y para qué sirve este lenguaje de computación, así como una serie de aplicaciones específicas y las conclusiones obtenidas de esta investigación.

ANTECEDENTES HISTORICOS

APL (A Programming Language) fue inventado en la Universidad de Harvard por el Doctor Kenneth E. Iverson alrededor de 1957, cuando encontró que no existía notación matemática adecuada para expresar claramente algoritmos complejos. Como el Doctor Iverson lo declara en el libro que resultó de sus investigaciones (A PROGRAMMING LANGUAGE - WILEY, 1962)

> "El tratamiento sistemático de algoritmos complejos requiere de un lenguaje para programar apropiado para sus descripciones, este lenguaje debe ser conciso, preciso, consistente, sobre un amplio rango de aplicación, nemónico y económico en símbolos; éste debe exhibir claramente las necesidades sobre la secuencia en la cual las operaciones son desarrolladas, y esto debe permitir la descripción de un proceso para ser independiente de una representación particular escogida para los datos."

El lenguaje que desarrolló durante 10 años ha llenado grandemente estas metas, las simples reglas uniformes de sintáxis y entrada numérica de forma libre hacen al lenguaje fácil de aprender, aún más poderoso en su uso. APL además provee un gran conjunto de operadores primitivos, los cuales trabajan indistintamente con escaleras, vectores, matrices y arreglos multidimensionales.

Si cierta función no existe como operador primitivo, ésta puede ser definida en un programa simple y utilizado posteriormente como operador. Como lenguaje de programación, APL es considerado como más conciso y

eficiente para definiciones que cualquier otra notación. En algunos casos un caracter en notación APL puede ser equivalente a una o varias líneas en la mayoría de los otros lenguajes.

EL LENGUAJE APL

A.- CONCEPTOS BASICOS

Operaciones Básicas

Al igual que cualquier calculadora, APL puede manejar las cuatro operaciones básicas(+, -, ×, ÷)

```
      95 + 117
212
      338 - 182
156
      3965 × 217
860405
      500 ÷ 10
50
```

Sin embargo, debido a una de las caracterfsticas de APL que nos permite utilizar los operadores con escalares, vectores, matrices y arreglos multidimensionales, estamos facultados para hacer operaciones de la forma siguiente:

```
      2 4 6 + 15 20 25
17 24 31
      3 × 10 20 30
30 60 90
      1 2 3 + 2 3 4
      4 5 6   5 6 7
3 5 7
9 11 13
```

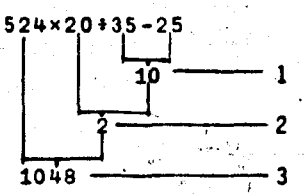
o también nos permite manejar varias operaciones en una sola instrucción.

```
      524×20÷35-25
1048
```

Tanto las operaciones con arreglos como el manejar varias operaciones, están sujetas a reglas de sintaxis que se describen a continuación.

La Regla de Derecha a Izquierda

Como vimos, APL puede desarrollar varias operaciones en un solo renglón, cuando éste sea el caso, APL maneja los operadores de derecha a izquierda sin importar el tipo de operador involucrado.



En el diagrama se presenta la evolución de un grupo de operaciones en un renglón.

Algebraicamente lo podemos representar como:

1er. paso 35 - 25 = 10

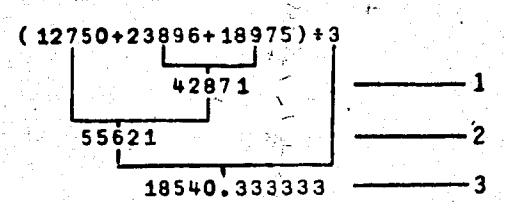
2nd. paso 20 ÷ 10 = 2

3er. paso 524 x 2 = 1048 = resultado final

o utilizando paréntesis:

524(20 ÷(35 - 25)) = 1048

Si se quiere cambiar la secuencia de una operación en APL, es necesario usar paréntesis en cuyo caso primero se desarrollaran las operaciones que están entre paréntesis y una vez que éstas se hayan realizado, la operación seguirá su secuencia normal, o sea, de derecha a izquierda.



Formas de Construcción

Todas las operaciones con arreglos deben de cumplir con cualquiera de las siguientes normas:

1.- Los argumentos son escalares (resultado escalar)

```
      4+4-2×6
0
```

2.- Si los argumentos son vectores u otro tipo de arreglos, éstos deben tener el mismo tamaño y dimensión

```
      5 6 8 + 1 2 7
6 8 15
```

3.- Si un argumento es escalar y el otro es un vector u otro tipo de arreglo, el resultado es un arreglo con el mismo tamaño y dimensión que el arreglo argumento

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

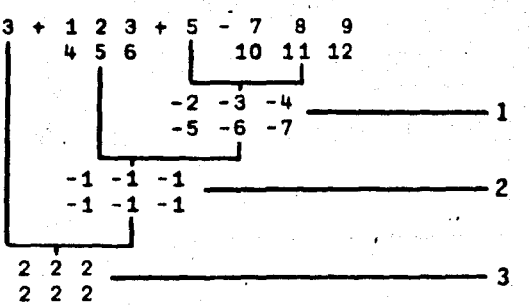
En el caso de vectores y arreglos, las operaciones se harán de elemento a elemento, o sea,

```
      1 2 3 + 4 5 6

5 7 9
```

también podremos hacer operaciones que combinen estas reglas

```
      3 + 1 2 3 + 5 - 7 8 9
          4 5 6       10 11 12

2 2 2
2 2 2
```

analicemos paso a paso la operación anterior



En el primer paso estamos restando una matriz de un escalar; en el segundo paso, la matriz resultante más otra matriz del mismo tamaño y dimensión (2 x 3) y por último en el tercero, la matriz resultante más otro escalar siendo el resultado final una matriz del mismo tamaño y dimensión que las otras dos anteriores.

Tipos de Funciones

Cuando el Doctor Iverson empezó a desarrollar este lenguaje

decidió que debido a que él quería que fuera interactivo, se manejara desde una terminal que fuera del tipo de una máquina de escribir con su correspondiente teclado, por medio del cual se enviaría la información al computador central y se recibieran los resultados impresos en la misma.

Fue por esto, que al ir creando los operadores primitivos, se encontró con que en poco tiempo había saturado el teclado de la mencionada terminal. Como consecuencia de esta saturación, ideó la manera de que cada operador tuviera la posibilidad de tener dos aplicaciones, naciendo así el concepto de operador monádico y operador diádico.

El operador monádico es aquel que tiene únicamente un argumento del lado derecho.

El operador diádico es el que tiene un argumento de cada lado.

Para efectos de ilustración, definamos al operador ÷ de manera monádica, el cual nos dará como resultado el inverso del argumento.

```
      ÷ 5 3 4
0.2 0.333333 0.25
```

Como ya hemos visto, el operador ÷ manejado de manera diádica nos dará la división del argumento izquierdo entre el derecho

```
      10 15 30 ÷ 5 3 6
2 5 5
```

Variables

Una variable es un nombre que representa un valor donde este valor puede ser cambiado a diferencia de una constante en la que el valor es siempre el mismo, (3 es siempre 3).

Todas las variables deben de empezar siempre con una letra pudiendo ser los demás componentes de las mismas letras o números.

Para dar valor a estas variables, se utiliza la flecha de asignación; esta flecha tiene la punta hacia la izquierda (←) y como ya dijimos, sirve para asignar un valor a una serie de valores a la variable.

Sea una serie de productos con su utilidad respectiva

```
      UTILIDAD ← 2.8 1.5 6.4 2.08 4.1
```

las ventas en unidades de cada uno de estos productos fue

```
      VENTAS ← 50 75 23 80 49
```

para obtener la utilidad total por producto multiplicamos

```
      VENTAS × UTILIDAD
140 112.5 147.2 166.4 220.5
```

Si queremos saber cuál es el valor que tiene una variable en un momento, simplemente tecleamos el nombre de la misma y automáticamente se despliega

```
      UTILIDAD
2.8 1.5 6.4 2.08 4.1
```

Errores

APL es un lenguaje que fue construido basado en las reglas de las matemáticas comunes, por lo cual si infringimos cualquiera de ellas, se presentará un error. Los errores más comunes son los siguientes:

SYNTAX ERROR: es cuando la expresión no ha sido bien construida en base a su sintaxis como cuando los paréntesis no están completos

```
      (3+4÷5
SYNTAX ERROR
(3+4÷5
^
```

el tipo de error es desplegado por el sistema y en seguida, la instrucción vuelve a ser desplegada y el sistema nos informa por medio de un ^ la posición del error, lo cual nos permite fácilmente hacer la corrección.

DOMAIN ERROR: es cuando la función no está definida para un argumento dado; un número dividido entre cero

```
      5÷0
DOMAIN ERROR
5÷0
 ^
```

LENGTH ERROR: en una operación con dos vectores, éstos tienen diferente número de elementos

```
      2 3 4 + 2 3
LENGTH ERROR
2 3 4 + 2 3
    ^
```

VALUE ERROR: no existe valor asociado con una variable dada

```
      UTILIDADES
VALUE ERROR
UTILIDADES
^
```

RANK ERROR: es el mismo caso que el LENGTH ERROR, sólo que aplicado a arreglos de dos o más dimensiones.

Representación de Datos

APL nos provee de la capacidad de representar los valores numéricos de dos formas: una, como decimales ordinarios y otra, de forma exponencial. Esta última tiene una gran aplicación sobre todo, cuando manejamos valores numéricos muy grandes o decimales muy pequeños. La forma en que son representados los números de manera exponencial es por medio de una *E*

```
      10 E 5 = 1000000
      10 E ¯5 = .00001
```

Generador de Indices

El primer operador específico de APL que es analizado, es el generador de índices; éste es un operador monádico, o sea, que sólo tiene argumento del lado derecho, está representado por una iota (⍳) y su función es precisamente generar índices

```
      ⍳5
1 2 3 4 5
```

es aplicable a cualquier número entero positivo y el resultado será el conjunto de números enteros comprendidos entre el uno y el argumento.

La razón por la que la presento en primer lugar, es porque, aunque su concepto es muy simple, tiene un gran rango de aplicación. Veamos el caso en que queremos obtener los enteros pares comprendidos entre el dos y el diez

```
      2 × ⍳5
2 4 6 8 10
```

o los números impares del mismo conjunto

```
      ¯1 + 2 × ⍳5
1 3 5 7 9
```

En estos dos casos nos encontramos con que se presentan dos operadores juntos entre los argumentos; sin embargo, partiendo de la regla de derecha a izquierda, vemos que en ambos casos la operación que primero se realiza es ⍳5 y una vez realizada ésta, el resultado es multiplicado ×2 ; ésta es otra de las características de APL que lo provee de una gran flexibilidad, ya que en un momento determinado existe la posibilidad de intercalar, no sólo dos, sino un número indefinido de operadores entre dos argumentos, conscientes de que de acuerdo con la regla de derecha a izquierda, esto sería totalmente válido.

B.- ARREGLO DE DATOS

Arreglos

Una forma de arreglo puede ser un vector, el cual es de dimensión uno, ya que está arreglado linealmente.

Otro puede ser una matriz que tiene dos dimensiones, ya que sus componentes están arreglados por renglones y columnas.

Arreglos de tres o más dimensiones no tienen nombre específico y sus componentes están arreglados en renglones, columnas, planos, hiperplanos, etc.

La importancia de un grupo de datos ordenados se hace sentir cuando hay que hacer operaciones de componente a componente.

Dimensión y Rango

El término rango es usado en APL para identificar una forma de arreglo de otra. El rango de un arreglo es el número de índices independientes, que son necesarios para identificar un componente.

En APL, el nombre de una variable puede representar un elemento o un arreglo. Si queremos conocer "el tamaño" de un arreglo, la función (ρ) nos lo dá. La forma general de shape es ρA , donde puede ser cualquier arreglo. El resultado de ρA es siempre un vector, cuyos componentes representan el número de elementos en cada dimensión del arreglo.

```
      PRECIO←3 5 7 9
      ⍴PRECIO
4
      INT
2 4 6
10 12 14
      ⍴INT
2 3
      ⍴⍴PRECIO
1
```

Construcción y Arreglos

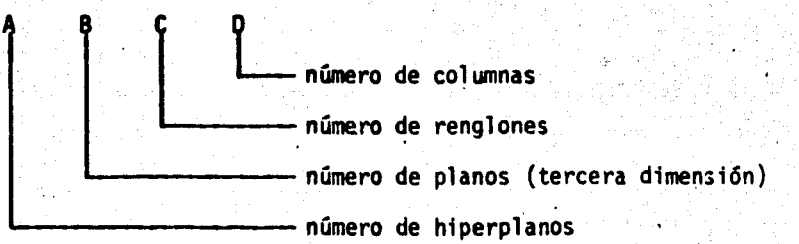
Para construir un arreglo se emplea el operador ⍴ en forma diádica, donde el argumento del lado izquierdo representa el número de elementos de cada dimensión del arreglo y el argumento del lado derecho es un vector de datos.

```
      B←2 3⍴2 4 6 10 12 14
      B
2 4 6
10 12 14
```

El número de componentes en el argumento izquierdo, define el rango del arreglo deseado. El valor de cada componente define el grado de libertad disponible en el índice de la dimensión correspondiente. Si el argumento izquierdo es un escalar o un vector de un componente, el resultado es un vector, y el valor del argumento

izquierdo define el número de componentes en el resultado y así sucesivamente.

En general, el argumento izquierdo A, B, C, D, de la función puede ser interpretada como sigue:



Arreglos Indexados

Es común que una vez que tengamos un arreglo definido (de cualquier dimensión), querramos conocer uno o varios de los componentes de éste, o bien, que querramos modificarlos. En esta sección se analizan estas posibilidades utilizando índices.

Supongamos que tenemos un arreglo *B* con los siguientes componentes.

```
      B
15 30 45 60 85
```

Este arreglo, como se puede ver, es un vector, esto es; tiene rango uno. Para que nosotros podamos identificar qué componente se encuentra en una posición dada, hacemos lo siguiente:

```
      B[4]
60
```

que es equivalente a preguntar cuál es el componente del vector *B* que se encuentra en la cuarta posición. APL también nos permite preguntar por varios componentes a la vez, esto es:

```
      B[4 3 1]
60 45 15
```

Donde cada posición o índice irá separado por un blanco. Los índices contenidos entre [] siempre deberán de representar números enteros y positivos, se usa el concepto representar, ya que existe la posibilidad de que se ponga como índice una variable numérica

```
      K← 1 2 3
      K
1 2 3
      B[K]
15 30 45
```

esta última posibilidad, nos provee de una gran flexibilidad, ya que al manejar los índices como variables podremos cambiarlos de una manera dinámica.

Como se ha venido diciendo, APL maneja por igual cualquier tipo de arreglo.

Con objeto de generalizar lo anterior, decimos que para poder identificar una dimensión de otra en el caso de arreglos multidimensionales, éstos deberán de ir separados por un punto y coma; en el caso de una matriz tendremos:

```
      A
10 20 30
40 50 60
      A[2;3]
60
```

Donde el índice del lado izquierdo del punto y coma es relativo a los renglones y el del lado derecho a las columnas. Al igual que en el caso de vectores, se pueden pedir varios componentes del arreglo a la vez

```
      A[1;3 2]
30 20
```

APL también nos permite que asignemos parte de un arreglo; lo anterior se logra por medio de índices

```
      C← A[1;1 2 3]
      C
10 20 30
```

Si queremos obtener un renglón o una columna completa de un arreglo, simplemente dejamos en blanco la posición correspondiente; esto es:

```
      A[;3]
30 60
```

Ahora, si queremos modificar uno o varios componentes de un arreglo, esto lo podremos hacer utilizando índices

```
      A
10 20 30
40 50 60

      A[1;1]
10
```

Si queremos modificar el componente que se encuentra en la primera columna del primer renglón (10), simplemente definimos

```
      A[1;1]←100
```

quedándonos el arreglo como

```
      A
100 20 30
 40 50 60
```

Si queremos substituir varios componentes de un arreglo por un valor común, hacemos lo siguiente:

```
      A[1;1 2]←80
      A
80 80 30
40 50 60
```

de lo anterior, podemos deducir que si queremos modificar componentes de un arreglo, tendremos dos opciones:

1.- Definir un nuevo valor para cada componente a modificar.

2.- Definir un valor común para todos los componentes a modificar.

Antes de continuar con este trabajo, es importante hacer notar que aunque hasta el momento únicamente hemos trabajado con arreglos numéricos, APL también maneja arreglos de caracteres, lo cual es de gran utilidad sobre todo para el caso en el que manejamos títulos.

Para definir arreglos de caracteres, éstos tendrán que estar entre apóstrofes.

```
      TITUL←3 6 ρ'LINEA1LINEA2LINEA3'
      TITUL
LINEA1
LINEA2
LINEA3
```

Como puede verse en el ejemplo, todo lo que fue definido entre apóstrofes es manejado como caracteres, sin importar que sea numérico o no.

También debemos mencionar que cuando estamos definiendo arreglos de caracteres, todas las posiciones entre los apóstrofes son significativas, incluyendo a los espacios en blanco.

```
      CAR← 'A B C D'
      ρCAR
7
      CAR[3]
B
```

Contrario a lo anterior, existen ocasiones en que sabemos que hay un dato que nos interesa dentro de un arreglo; sin embargo, no conocemos la posición ni queremos desplegar todo el arreglo para

localizarlo. APL nos provee de un operador que soluciona lo anterior, este operador es la iota (⍳) , pero utilizada de manera diádica; el argumento izquierdo será el arreglo y el derecho el dato que queremos localizar, el resultado será la posición dentro del arreglo donde se localiza esta información.

```
      'ABCD' ⍳ 'A'
1
```

Ravel (, *A*)

Ocasionalmente, queremos que un arreglo de dimensión M x N sea desplegado en forma de vector, o sea, asignado a una nueva variable como tal. Para esta situación, APL cuenta con el operador denominado RAVEL que es la coma presentada de manera monádica (, *A*) donde el argumento de la función será el arreglo que queramos representar de forma vectorial.

```
      A
1 2 3
4 5 6

      ,A
1 2 3 4 5 6
```

Posteriormente, veremos que esto será de gran utilidad cuando querramos obtener el gran total de los valores contenidos en un arreglo de dimensión M x N.

Ordenación de Datos

APL nos provee de unos operadores diádicos (⍋⍒) que nos permiten conocer la posición de los valores contenidos en un vector en orden creciente o decreciente, lo cual aplicado como índices al mismo vector, nos permite ordenar éste en cualquiera de los dos sentidos.

```
      VEC← 5 3 9 7
      VEC
5 3 9 7
      ⍋VEC
2 1 4 3
      VEC[⍋VEC]
3 5 7 9
      VEC[⍒VEC]
9 7 5 3
```

Estos operadores son compuestos, ya que están formados por la sobreposición de ∆ y | en un caso y de ∇ y | en el otro.

Arreglos Aleatorios

En el punto anterior, vimos producir dos permutaciones específicas de un grupo de índices. Otra permutación útil de índices es un arreglo aleatorio de los mismos.

El operador para esta función es ? y la forma general de la función se describe como

```
A?B
```

Donde *A* es el número de elementos del vector resultante y *B* es el grupo de enteros de donde va a ser seleccionado

```
      5?5
1 5 2 3 4
```

dando en cada ocasión una combinación distinta de los mismos

```
      5?5
5 3 4 2 1
```

los enteros son seleccionados sin reemplazo, lo cual quiere decir que una vez que un entero ha sido seleccionado no puede ser seleccionado nuevamente, lo anterior aplicado a índices nos dará una ordenación aleatoria de los componentes contenidos en el arreglo

```
      VEC← 30 20 50 80
      VEC[4?⍴VEC]
30 50 80 20
```

C. FUNCIONES ELEMENTALES

Funciones Monádicas Escalares

La función *?A* analizada en el tema anterior, fue la primera función monádica escalar discutida. Una función escalar monádica se aplica a cada elemento del argumento de un arreglo componente por componente.

Las funciones para econtrar el valor absoluto, el recíproco y la negación de un número son tres funciones monádicas escalares comunes.

La función valor absoluto (|) es igual que el de los matemáticos tradicionales - su forma general es:

```
|B
```

donde *B* es cualquier arreglo, el valor absoluto quita el signo del argumento dejando únicamente su magnitud.

```
|¯23 23 4.6 ¯3.7
23 23 4.6 3.7
```

La función negación se refiere a que el valor del número afectado sea negativo

```
¯3
```

o sea, el valor negativo de 3.

El resultado de la función recíproco ÷ es 1 dividido entre el argumento.

La forma general es:

```
+B
```

donde *B* es cualquier arreglo

```
+5 .25 ¯2
.2 4 ¯0.5
```

Funciones de Rango Limitado

El rango de una función es el conjunto de todos los posibles valores del resultado de la función.

El dominio de una función es el conjunto de todos los posibles valores de los argumentos de la función.

Muchas funciones de APL tienen un dominio y/o rango limitado, algunas de ellas son discutidas en este tema.

Techo y Suelo

Dos funciones escalares monádicas son usadas para truncar números fraccionarios a enteros, éstas son techo (⌈) y suelo (⌊); la forma general de suelo es:

```
⌊B
```

donde *B* es cualquier arreglo. El resultado de la función suelo ⌊*B* es el mayor número entero menor o igual que *B*

```
⌊6.2 6.8 ¯6.2 8
6 6 ¯7 8
```

La forma general de techo es:

```
⌈B
```

donde *B* es cualquier arreglo y el resultado de la función techo ⌈*B* es el menor número, mayor o igual que *B*

```
      ⌈6.2 6.8 ¯6.2 6
7 7 ¯6 6
```

Funciones Relacionales

El conjunto de funciones diádicas cuyo resultado tienen un rango más limitado que techo y suelo, son las seis funciones relacionales <, ≤, =, ≠, >, ≥, cada una de las cuales nos da "1" si la relación presentada es verdadera y "0" si es falsa. En seguida se presenta una tabla general de estas funciones.

NOMBRE	SIMBOLO	SINTAXIS	EJEMPLOS
Menor que	<	*A*<*B*	9 ¯8<11 ¯1 1 1
Menor o igual que	≤	*A*≤*B*	6 5≤5 5 0 1
Igual que	=	*A*=*B*	6 5=5 5 0 1 '*FIX*'='*SIX*' 0 1 1
Diferente que	≠	*A*≠*B*	6 5≠5 5 1 0 '*FIX*'≠'*SIX*' 1 0 0

NOMBRE	SIMBOLO	SINTAXIS	EJEMPLOS
Mayor que	>	*A>B*	6 7>5 9 1 0
Mayor o igual que	≥	*A≥B*	6 7≥6 9 1 0

Sólo las funciones igual y diferente pueden emplearse con caracteres.

Membresía

Las funciones relacionales pueden ser usadas para determinar que los componentes de un vector cumplan con ciertos requisitos, algunas veces es necesario determinar sólo si un elemento está presente en un conjunto de datos: por ejemplo, nos interesa saber si 8 es elemento del conjunto *M* . La función diádica de membresía (∈) se usa para lograr lo anterior.

Su forma general es:

A∈B

donde *A* y *B* pueden ser arreglos cualquiera. No existe ninguna relación en la construcción de ambos, el resultado tendrá el mismo número de elementos que el argumento izquierdo, el rango será 0, 1 .

```
      8∈8
1
      9∈6 2 1
0
      8 9 6∈9
0 1 0
      'A'∈'BROAD'
1
```

Funciones Lógicas

Las funciones lógicas son $Y\wedge$, $O\vee$, $Y\ NO\tilde{\wedge}$, $O\ NO\tilde{\vee}$, $NO\sim$.

El resultado de $A\wedge B$ es 1 si y sólo si A y B son ciertas, el resultado de $A\vee B$ es 1 si cualquiera de las dos es cierta.

En seguida se presenta una sumaria con las distintas alternativas y los resultados posibles.

A	B	$A\wedge B$	$A\vee B$	$A\tilde{\wedge}B$	$A\tilde{\vee}B$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	1	0	0

Signo

La función monádica escalar signo (×) es usada cuando la única información que necesitamos es saber si el número es positivo, negativo o cero.

La forma general de signo es:

$\times B$

donde B es cualquier arreglo. El resultado de $\times B$ es como sigue:

RESULTADO	SIGNIFICADO
1	B Positivo
¯1	B Negativo
0	B Cero

```
      ×6 ¯9 0 2.1 ¯7.3
1 ¯1 0 1 ¯1
```

Funciones más Comunes

Algunas funciones de APL que son comunes a la aritmética y al algebra serán discutidas en este capítulo.

Exponencial

La forma general de exponenciación es:

$A*B$

donde A es la base y B el exponente, si el exponente es un entero positivo, el resultado es el de multiplicar A B-veces por si misma.

$A*3 = A \times A \times A$

Si el exponente es fraccionario positivo, el resultado es la raiz D-esima

```
      25*.5
5
      64*÷3
4
      27*2÷3
9
```

Si el exponente B es negativo, el resultado es equivalente al recíproco de A elevado a la tasa positiva y esto es $(÷A)*¯B$

```
      5*¯2
0.04
```

Usando de manera monádica $*A$ nos dará el exponencial del argumento.

Logaritmos

La función diádica escalar logaritmo ($A \bullet B$) nos da el logaritmo de B en base A

$A \bullet B$

Usado en forma monádica, el operador $\bullet B$ nos da el logaritmo natural del argumento.

Factorial, Combinaciones y Residuo

Factorial en matemáticas se define como la multiplicación de un número consecutivo de números, esto es:

$3! = 3 \times 2 \times 1$

En el caso de APL, tiene el mismo significado con la diferencia de que la función factorial ($!A$) va a la izquierda del argumento; existen dos formas de emplear ($!$).

1) Cuando el argumento es entero positivo en el que produce el producto.

$1 \times 2 \times 3 \times \ldots \times B$.

```
      !4
24
      !2 5 4
2 120 24
```

2) Si B no es entero, el resultado es el valor de la función gama elevada en $B+1$

```
      !.25
0.9064
```

El argumento *B* no puede ser entero negativo, si *B* es cero *B* es 1.

Combinaciones

Partiendo de la forma general de las combinaciones que dice, dado un conjunto encontrar las posibles combinaciones de C elementos en el conjunto B, la fórmula matemática es:

$$= \frac{B!}{(B - N)!N!}$$

en APL se representa simplemente como:

N!*B*

donde *N* y *B* pueden ser arreglos. El resultado es el número de combinaciones de *B* elementos tomados de *N* maneras.

Ejemplo:

```
      4!8
70
      2!26 10
325 45
```

Para terminar, supongamos que seis personas están jugando FAN-TAN, un juego de cartas en que todas ellas van a ser distribuidas, la idea es conocer a cuántas de estas personas les tocará una carta extra.

La función diádica escalar RESIDUO (|) puede ser para resolver este problema. Su forma general es:

A|*B*

donde *A* y *B* pueden ser arreglos. La idea de la función residuo, es obtener el residuo después de que *B* ha sido dividido entre *A* .

Volviendo al caso de las cartas:

```
      6|52
4
```

O sea, que cuatro personas van a tener una carta de más.

Adicionalmente a los operadores presentados en este último párrafo, existe un grupo de ellos con aplicaciones específicas en el campo de la geometría, debido a que sus aplicaciones son bien claras; únicamente, se presenta un cuadro de éstos, así como una breve descripción de su funcionamiento.

OPERADOR	FUNCION
0Y	w veces Y
0oY	(1-Y*2)*.5
1oY	seno de Y
2oY	coseno de Y
3oY	tangente de Y
4oY	(1+Y*2)*.5
5oY	seno hiperbólico de Y
6oY	coseno hiperbólico de Y
7oY	tangente hiperbólica de Y
¯1oY	arcoseno de Y
¯2oY	arcocoseno de Y
¯3oY	arcotangente de Y

OPERADOR	FUNCION
⁻4oY	(⁻1+Y*2)*.5
⁻5oY	arcoseno hiperbólico de Y
⁻6oY	arcocoseno hiperbólico de Y
⁻7oY	arcotangente hiperbólico de Y

D. EXTENSION SOBRE ARREGLOS

En el inciso anterior fueron analizados una serie de operadores que aunque podrían ser aplicados a cualquier tipo de arreglo, el resultado obtenido era particular para cada componente del mismo.

En este inciso presento un grupo de operadores, los cuales nos permiten desarrollar operaciones en las que para obtener un resultado, se involucra a todos los componentes de arreglo; tal es el caso de las sumatorias, la multiplicación, la división, la inversa de matrices desde el punto de vista algebráico e incluso la solución de sistemas de ecuaciones lineales.

Estos operadores tienen una gran aplicación, sobre todo en el campo de la investigación de operaciones y en el de la estadística, donde operaciones de este tipo son requeridas constantemente.

Reducción

Si tenemos el vector

```
ENROLL←250 207 189 170
```

y queremos obtener la suma de todos sus elementos, una manera de hacerlo es sumar elemento a elemento.

```
      250+207+189+170
816
```

Y otro puede ser

```
      ENROLL[1]+ENROLL[2]+ENROLL[3]+ENROLL[4]
816
```

Sin embargo, necesitamos conocer en el primero, el valor de cada elemento y en el segundo, el número de elementos que contiene la variable *ENROLL* .

En APL existe una forma de sumar todos los elementos de un arreglo sin necesidad de conocer de antemano ni el número de elementos ni el valor de cada uno de ellos; esto se puede hacer, gracias a la reducción sobre suma (+ /) donde ésta es equivalente a un signo + entre cada elemento del arreglo. Para obtener la suma del vector *ENROLL* simplemente definimos:

```
      +/ENROLL
816
```

Como la suma de un conjunto de números, muchas veces se requiere obtener el producto de un conjunto de números, el más grande de un conjunto de números, el más pequeño, etc.

El operador reducción representa de manera general estos tipos de funciones. Su forma general es:

FN/A .

donde *FN* puede ser cualquier función diádica primitiva y *A* es cualquier arreglo.

En general, la reducción se define junto con el nombre de la función (reducción sobre suma, reducción sobre producto, reducción sobre máximo, etc.) y es equivalente a insertar la función entre cada elemento del arreglo *FN/ A,B,C,D = AFNBFNCFND* . La función reducción también se extiende a operadores lógicos y relacionales.

Cuando trabajamos con matrices y queremos hacer la reducción sobre renglones, simplemente indexamos +/[1]*A* ; si queremos hacer la reducción sobre columnas, empleamos +/[2]*A* , donde el número entre corchetes se refiere a la dimensión sobre la cual se quiere hacer la reducción.

Acumulación

Supongamos el caso en que tengamos una información mensual sobre las ventas de un producto y queremos obtener el acumulado mes a mes.

En APL para poder lograr lo anterior, utilizamos la función acumulación, cuya forma general es:

*FN**A*

donde *FN* puede ser cualquier función primitiva diádica y *A* cualquier arreglo.

Si queremos obtener el acumulado mensual de:

```
PRODUC←10 14 25 10 30 0 15 50
```

Simplemente definimos

```
+\PRODUC
10 24 49 59 89 89 105 154
```

el arreglo puede ser acumulado sobre la dimensión que se quiera +\[*I*]*A* donde *I* es la dimensión requerida.

Producto Exterior

Los primeros tres números nones elevados al cubo se obtienen:

```
      3*1 3 5
3 27 243
```

Pero ahora supongamos que queremos elevar los tres primeros nones a las tres primeras potencias nones, si nosotros hacemos:

```
      1 3 5*3
1 27 125
```

No cumple con lo solicitado.

El resultado que deseamos es:

*	1	3	5
1	1	1	1
3	3	27	243
5	5	125	3125

o sea, multiplicar cada elemento izquierdo por cada uno de los elementos del derecho.

En APL, el producto exterior realiza esta función. Su forma general es:

A∘.*FN B*

donde *A* y *B* pueden ser cualquier arreglo y *FN* una función primitiva diádica.

Para obtener el resultado del ejemplo inicial

```
      1 3 5 ∘.*1 3 5
1   1   1
3   27  243
5 125 3125
```

donde la tabla resultante va a tener la dimensión del argumento izquierdo, por el derecho.

Producto Interior

El siguiente problema puede ser resuelto simplemente por reducción.

Un stock contiene acciones de tres distintas compañías. ¿Cuál es el valor total del stock?

Si *NUM* es el vector que contiene el número de acciones por compañía, *VM* el valor de mercado de cada una de ellas *+/VM×NUM*, nos dá el valor total; por ejemplo:

```
      NUM←100 15 25
      VM←18.5 301 120.75
      +/VM×NUM
9383.75
```

Pero supongamos que queremos saber cuál ha sido el valor del paquete, en base a los precios promedio de los últimos 4 años.

La matriz *PRECIO* (3 x 4) contempla estos precios, siendo los renglones las compañías y las columnas los años.

Tomamos el vector *NUM* y lo multiplicamos por cada año y luego la reducimos sobre suma.

```
      PRECIO
  18.5   17.25  21.5   20.6
301     285    321    331
120.75  115    96     94
      +/NUM×PRECIO[;1]          +/NUM×PRECIO[;4]
9383.7                    9382.5
      +/NUM×PRECIO[;2]
8875
      +/NUM×PRECIO[;3]
9365
```

Como se puede ver, tuvimos que efectuar cuatro cálculos, en APL existe una función que simplifica lo anterior, esta función es el producto interior, cuya forma general es:

$$A\ FN_2\ .\ FN_1\ B$$

donde A y B son arreglos, y FN_1 y FN_2 son funciones diádicas primitivas.

Algunos ejemplos de producto interior son:

+.× ∧.∨ ⌈.× ⌈.⌊

El nombre de ésta son: Producto interior más - por, producto interior Y - O, producto interior máximo - por y producto interior máximo - mínimo.

La solución al ejemplo queda como:

```
      NUM+.× PRECIO
9383.7 8875 9365 9382.5
```

La ejecución del producto interior es básicamente un proceso de dos etapas, en la primera la función FN_1 es aplicada a los argumentos elemento por elemento, en seguida la reducción sobre la FN_2 es aplicada al resultado obtenido de la FN_1 .

En el caso de matrices, el argumento izquierdo es operado por renglones y el argumento derecho por columnas. Si uno de los argumentos es un vector y el otro una matriz, el vector es considerado como renglón o como columna, dependiendo de su posición. En el ejemplo de arriba, *NUM* es considerado como un vector renglón, el cual es combinado con las columnas de *PRECIO*.

Hemos visto cómo el producto interior más-por puede ser usado, para recordarnos que cualquier función primitiva diádica puede ser empleada en el producto interior. Supongamos por ejemplo, que queremos conocer el valor.

```
      NUM⌈.×PRECIO
4515 4275 4815 4972.5
```

Funciones de Dos Matrices

. Matriz Inversa

La inversa de una matriz dada es una

matriz que cuando es multiplicada para la matriz original, da como resultado la matriz idéntica. Una matriz idéntica es aquella que contiene unos en la diagonal central y ceros en las demás posiciones.

La matriz idéntica juega el mismo papel en álgebra lineal que el número uno en el sistema de números reales; a diferencia de los números reales en que todos los números excepto el cero, pueden ser divididos entre uno (recíproco) no todas las matrices tienen inverso. Las matrices que no tienen inverso son llamadas matrices singulares, las que sí, se llaman matrices no singulares.

Se han desarrollado muchos métodos para obtener la inversa de una matriz, los cuales no son tan simples como encontrar el recíproco de un número.

En APL, el proceso para encontrar la inversa de una matriz se ha reducido a un simple operador (⌹), cuya sintaxis es:

```
⌹A
```

donde *A* debe ser de rango dos o menor y si *A* es una matriz, sus columnas deben ser linealmente independientes.

Si *A* es una matriz cuadrada, no singular, entonces ⌹*A* es la inversa de *A* . Por ejemplo:

```
      A
¯0.5  0.2  ¯1.6
 0    0.2   0.4
 0.5  0     1

      ⌹A
 2   ¯2   4
 2    3   2
¯1    1  ¯1
```

Para verificar tomemos:

```
      (⊞A)+.×A
1.000E0 8.326E¯17 4.441E¯16
¯3.331E¯16 1.000E0 ¯4.441E¯16
1.665E¯16 ¯6.939E¯17 1.000E0
```

Es importante hacer notar que un número con exponente ¯16 o menor, debe ser tomado como cero, esto queda más claro si aplicamos la función suelo (⌊) a la matriz resultado.

```
      ⌊(⊞A)+.×A
1 0 0
0 1 0
0 0 1
```

El argumento no tiene que ser necesariamente una matriz cuadrada, pero sí debe de tener más renglones que columnas, en otra cosa un ERROR DE DOMINIO es reportado.

El argumento *A* de ⊞*A* puede ser también un vector o un escalar.

Si *A* es un vector, éste es tratado como una matriz de una columna (nunca como una matriz de un renglón, ya que rompería con el requerimiento de que las columnas deben ser independientes).

Por ejemplo:

```
      V←15
      ⊞V
0.018182 0.036364 0.054545 0.072727 0.090909
```

Si *A* es un escalar, es tratado como una matriz del 1 x 1 para el argumento escalar *S* , la expresión ⌹*S* es equivalente a ÷*S* , por ejemplo:

```
      ⌹5
.2
      ÷5
.2
```

Si *A* tiene un rango mayor de dos, un ERROR DE RANGO es reportado.

. División de Matrices

La división de matrices es usada para resolver sistemas de ecuaciones lineales. Su forma general es:

N⌹*M*

donde el argumento *N* puede ser un vector o una matriz y el argumento *M* debe ser una matriz.

Existe relación entre la solución de una simple ecuación ax = b y la solución de un sistema de ecuaciones Ax = B. La solución de una ecuación lineal simple es *B*÷*A* . La solución de un conjunto de ecuaciones es *B*⌹*A* , consideramos un conjunto de tres ecuaciones simultáneas.

$$4W + Y + 2Z = 16$$
$$W + 2Y + 5Z = 12$$
$$W + 3Y + Z = 10$$

El vector de soluciones S para este conjunto de ecuaciones es:

```
      S←B⌹A
```

donde B es un vector de constantes y A es la matriz de coeficientes de las incognitas.

```
      B
16 12 10
      A
4 1 2
1 2 5
1 3 1
```

entonces,

```
      S←B⌹A
3 2 1
```

para verificar $A+.\times S$ debe ser igual a B

```
      A+.×S
16 12 10
```

La expresión

```
      S←B⌹A
```

se ejecuta si:

1) La primera dimensión de A y B son iguales.
2) Las columnas de A son linealmente independientes.

El vector solución S producido por $B⌹A$ es el resultado de minimizar la expresión de mínimos cuadrados $+/,(B-A+.\times S)*2$

El argumento izquierdo B puede ser una matriz, el tamaño del resultado S es el mismo que el de B por ejemplo:

```
      A
4 1 2
1 2 5
1 3 1
      B
16 1
12 2
10 3
      B⌹A
3.0000E0 4.9179E¯17
2.0000E0 1.0000E0
1.0000E0 1.2162E¯16
      ⌊B⌹A
3 0
2 1
1 0
```

Cada columna de la matriz de soluciones es la solución del conjunto de ecuaciones simultaneas cuyos coeficientes son dados por la matriz A y los valores constantes por las correspondientes columnas de B . Por ejemplo: $(B⌹A)[;1]$ es idéntico a $16\ 12\ 10⌹A$

Si B es una matriz idéntica, la expresión $B⌹A$ es equivalente $⌹A$.

. Cuando *A* y *B* son escalares, $B \boxplus A$ es equivalente a $B+A$ excepto para $0 \boxplus 0$ en que es error de dominio y no 1.

E. DEFINICION Y EDICION DE FUNCIONES

Definición de Funciones

En los incisos anteriores, fueron analizados una serie de operadores primitivos en cuanto a su aplicación y manejo, hemos visto que cada vez que definimos una operación, ésta es ejecutada al momento; sin embargo, existen ocasiones en las que queremos que una serie de operaciones queden definidas dentro de una función o programa para que posteriormente podamos ejecutarla con distintos datos, esta posibilidad es conocida en APL como DEFINICION DE FUNCIONES.

Una función definida está compuesta por una colección de expresiones formuladas por el usuario y que están agrupadas bajo un nombre, el cual en lo sucesivo puede o no ser usado como operador primitivo.

Las expresiones contenidas en la función son ejecutadas cuando el nombre de ésta, así como los argumentos, si es que existen, son introducidos en el sistema.

En APL existe una forma o modo de definir las mencionadas funciones, con objeto de presentar de manera objetiva cómo se debe definir una función. A continuación se presenta un ejemplo en el que se desean hacer una serie de cálculos sobre un paquete de acciones; estos cálculos son:

1.- Obtener el incremento o decremento del precio actual sobre el precio anterior.

2.- Rendimiento por acción

3.- Relación precio utilidad

Para tal efecto, definimos unas variables de manera arbitraria:

EARN: Utilidades del último año

CMV: Valor actual en el mercado

OMV: Valor antiguo en el mercado

DIV: Dividendos pagados

los cálculos que vamos a hacer son:

CMV - OMV: incremento/decremento sobre el precio anterior

100 x DIV ÷ CMV: Rendimiento

CMV ÷ EARN: Relación Precio - Utilidad

estas tres expresiones van a constituir el cuerpo de la función.

Para entrar en modo de definición, existe un operador ∇ (del), el cual debe de teclarse antes del nombre de la función que vamos a definir con este operador, automáticamente la computadora se pone en este modo y espera la información; para el ejemplo denominamos a la función como *STOCK* al teclear ∇*STOCK* , la computadora nos pregunta por la primera instrucción poniendo [1] .

```
[1]CMV-OMV
```

Al terminar de teclear la primera instrucción, la computadora nos imprime [2] en espera de la siguiente instrucción y así sucesivamente.

```
[2]  100×DIV÷CMV
[3]  CMV÷EARN
[4]
```

Una vez tecleada la instrucción [3] , la computadora nos pregunta por la [4] , pero como ya hemos acabado de definir nuestra función en la etiqueta [4] tecleamos simplemente ∇ con lo que le decimos al computador que hemos terminado de emplear el modo de ejecución.

Para esta última instrucción, existen dos alternativas siendo la primera la que se expresa en el párrafo anterior y la segunda tecleando ∇ a continuación de la última expresión.

```
[3]  CMV÷EARN∇
```

Después de este proceso, la función *STOCK* ha quedado definida como:

```
     ∇STOCK
[1]  CMV-OMV
[2]  100×DIV÷CMV
[3]  CMV÷EARN
     ∇
```

Antes de llamar (ejecutar) la función es necesario que asignemos valores a las variables implicadas, con el fin de evitar *VALUE ERROR* .

```
OMV←71 97.5 22.5 29.125
CMV←69.75 96 25 32.5
EARN←2.45 4.58 1.10 1.30
DIV←.49 1.3 .6 .66
```

para ejecutar la función, simplemente hay que teclear su nombre

```
      STOCK
¯1.25 ¯1.5 2.5 3.375
0.68817 1.3542 2.4 2.0308
24.474 26.816 22.727 25
```

donde cada vector representa el resultado de los cálculos previstos.

Edición de Funciones

Una vez que hemos definido una función, si se quiere editar o hacer algunas modificaciones, se emplea la edición de funciones, la forma más general es:

```
      ∇STOCK[□]∇
```

donde el (□) quad entre corchetes implica que toda la función *STOCK* va a ser desplegada.

Examinamos por un momento la estructura de lo que escribimos el primer ∇ implica que entramos en modo de definición, luego llamamos a la función *STOCK*, en seguida le pedimos que nos la despliegue [□] y por último nos salimos del modo de definición ∇

El quad es usado de tres formas distintas:

. ∇*NAME*[□]∇	Despliega la función entera
. ∇*NAME*[*N*□]∇	Despliega la línea N de la función
. ∇*NAME*[□*N*]∇	Despliega de la línea N en adelante

Ahora supongamos que deseamos intercalar una nueva línea en la función.

Para poder hacerlo, llamamos nuevamente a la función y entre corchetes ponemos un número fraccionario donde la parte entera del número es la línea anterior a la que queremos introducir.

```
      ∇STOCK[1.1]
```

Volviendo al ejemplo de *STOCK* vamos a introducir títulos a cada uno de los cálculos realizados.

```
      ∇STOCK[.1]'INCREMENTO SOBRE EL PERIODO ANTERIOR'
[.2]  [1.5]'RENDIMIENTO'
[1.6] [2.7] 'RELACION PRECIO UTILIDAD'
[2.8] ∇
```

después de cerrar con ∇ , las líneas de la función vuelven a ser reetiquetadas automáticamente quedando como sigue:

```
      ∇STOCK[□]∇
      ∇STOCK
[1]   'INCREMENTO SOBRE EL PERIODO ANTERIOR'
[2]   CMV-OMV
[3]   'RENDIMIENTO'
[4]   100×DIV÷CMV
[5]   'RELACION PRECIO UTILIDAD'
[6]   CMV÷EARN
      ∇
```

Si queremos cambiar una instrucción completa, hacemos lo siguiente:

```
∇STOCK[1]'CAMBIOS SOBRE EL PERIODO ANTERIOR'
```

Si queremos borrar una instrucción:

```
∇STOCK[1]
```

y damos un ATTN

Si queremos borrar la función completa:

```
)ERASE STOCK
```

Argumentos de una Función Definida

En la función *STOCK* no fue definido ningún argumento ni ningún resultado específico.

Pero ahora veamos la función *TEMP* en la que se convierten los grados farenheit en centígrados.

Esta se define como:

```
∇C←TEMP F
```

donde F va a ser el argumento y el resultado será C .

```
∇C←TEMP F
[1]  C←(F-32)×5÷9
∇
```

Durante la función *TEMP* , F será sustituida por el valor dado en el argumento y el resultado será dado automáticamente.

Este argumento puede ser un escalar o una variable definida anteriormente.

```
      TEMP 68
20
      MANY←50 75 90
      TEMP MANY
0 10 28.889 32.222
```

En el ejemplo anterior se vio una función con un argumento, ahora veamos otra función con dos argumentos.

Supongamos la función *VEL* para determinar el recorrido en millas que hace un automóvil basado en el diámetro de las llantas (en pulgadas) y el número de revoluciones por minuto de estas llantas.

Esta función es una función con dos argumentos y se define como:

```
      ∇MPH←RPM VEL DIAM
[1]   MPH←RPM×DIAM×3.14159×60÷528×12
      ∇
```

los argumentos deben ser llamados en el mismo orden en que fueron establecidos en el encabezado.

```
      500 VEL 25
37.187
      500 1000 1500 VEL 25
37.187 74.375 111.56
```

Existen seis posibles encabezados para funciones definidas, las cuales se pueden ver en la tabla siguiente:

NUMERO DE ARGUMENTOS	NOMBRE	SIN RESULTADO EXPLICITO	CON RESULTADO EXPLICITO
0	Niládica	∇*FN*	∇*Z*←*FN*
1	Monádica	∇*FN X*	∇*Z*←*FN X*
2	Diádica	∇*A FN B*	∇*Z*←*A FN B*

Variables Locales y Globales

Una variable es llamada local cuando únicamente existe durante la ejecución de la función; estas variables se definen en el encabezado enseguida del argumento derecho de la función y van separadas entre si por punto y coma.

∇*Z*←*FNC X*;*I*;*J*;*K*

siendo *I*, *J* y *K* las variables internas.

Las variables globales son aquellas que al no estar definidas como locales persisten después de la ejecución de la función o bien que han sido definidas antes de la misma.

Funciones Anidadas

Las funciones anidadas son aquellas que pueden ser llamadas durante el proceso de otra función (subrutinas).

Teniendo cada una de ellas sus propias variables locales.

Edición de Líneas

Anteriormente vimos como se podían editar, intercalar, cambiar y borrar líneas completas de una función.

Ahora supongamos la posibilidad de cambiar solo una parte de una línea o dentro de la misma introducir un valor.

En general para poder hacer lo anterior, existen tres pasos cada uno de ellos seguido de un RETURN.

1.- Entrar a la función y entre corchetes especificar del lado derecho del quad la línea que queremos modificar y del lado izquierdo la posición aproximada de la modificación. Si la posición es cero, saltamos el paso 2 y seguimos en el 3.

2.- Indicamos el cambio teclando cualquiera de las siguientes alternativas debajo del caracter implicado.

SIMBOLO	PROPOSITO
Un dígito del 1 al 9	Inserta el número de espacios especificados a la izquierda del caracter.
Una letra de la *A* a la *Z*	*A* deja cinco espacios, *B* deja diez y así sucesivamente
Un slash (/)	Borra el caracter o los caracteres.

Es permitido correr o regresar el carro.

3.- Escribir los nuevos caracteres en la línea, si existen, si no, sólo se borran caracteres.

Técnicas Adicionales y Otras Posibilidades para la Definición de Funciones

Como hemos visto hasta ahora, la definición de funciones permite al usuario escribir programas para elaborar procesos específicos.

El rango de problemas que pueden ser resueltos por medio de definición de funciones, puede ser ampliado por medio de técnicas adicionales. Las más importantes son poder cambiar la secuencia de un programa y la otra, poder controlar los formatos de salida.

a) Cambios en la secuencia de un programa, la manera que podemos contar con la posibilidad de cambiar la secuencia normal de un programa es usando "branching" en la definición del mismo.

La sintaxis para este estatuto es:

→*EXPRESION*

La flecha con dirección derecha seguida por una expresión, cuyo valor determina la siguiente acción a tomar.

La relación entre la expresión y la acción a tomar se indica en la tabla siguiente:

Valor de la Expresión	Acción
1. Un entero positivo I o un vector cuyo primer elemento es un entero positivo I	
a) Si I es menor o igual que el número de estatuto de la función	El estatuto I es el siguiente que se ejecuta.
b) Si I es mayor que el número de estatutos de la función	La ejecución de la función termina
2. Si es un cero (escalar) o el primer elemento de un vector es cero	La ejecución de la función termina

Valor de la Expresión	Acción
3.- Un arreglo vacío	Continúa con la siguiente línea

Para ilustrar como la expresión "branch" puede afectar a la ejecución de un programa, supongamos una función que ha sido definida para presentar varios tipos de problemas a un alumno, su respuesta determina el tipo de mensaje que recibirá. Una vez que el mensaje es dado, el alumno tendrá "x" posibilidades de fallar cuando se han agotado estas posibilidades, el programa terminará con el mensaje de POSIBILIDADES AGOTADAS.

La parte de la función involucrada se ve de la forma siguiente:

```
[15]   'BIEN, AHORA TRATE CON OTRA CLASE DE PROBLEMA'
[16]   →7
[17]   'NO, INTENTE EL PROBLEMA OTRA VEZ'
[18]   →3
[19]   'POSIBILIDADES AGOTADAS'
[20]   →0
```

Las líneas 16, 18 y 20 son instrucciones direccionadas, en la 16 dice "ejecute la línea 7", en la 18 "ejecute la línea 3" y en la 20 "termine con la ejecución".

Sin embargo, existen ocasiones en que una ejecución se va a ejecutar sólo si cumple con una restricción y si no se ejecuta otra, en este caso, decimos que la dirección está condicionada; este tipo de condiciones se expresan usualmente por la relación cierto (1) o falso (0).

Formas de expresar las direcciones

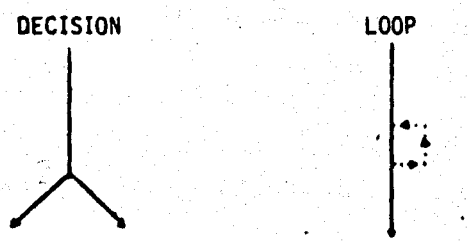
$\rightarrow(CONDICION)\rho NUMERO\ DE\ LINEA$

$\rightarrow NUMERO\ DE\ LINEA\times\iota CONDICION$

Las dos formas anteriores trabajan de manera similar a un I F con una dirección.

Existen dos formas de direccionar, la primera que se usa para tomar una decisión y la segunda que sirve para producir un "LOOP".

La representación gráfica de lo anterior es la siguiente:



Cuando se define una decisión con un loop, normalmente va acompañada de un contador (para hacer crecer o decrecer un índice dado.)

$C\leftarrow C+1$

$C\leftarrow C-1$

Dirección por Etiquetas

Normalmente cuando se está definiendo una función, es necesario introducir o quitar líneas en el cuerpo de la misma, lo que trae como consecuencia lógica la reetiquetación de las líneas, esto es muy perjudicial cuando existen instrucciones que direccionan a cierta línea del programa, ya que en un momento dado la lógica de este cambio , es el cambiar el número de una instrucción.

Con el objeto de evitar lo anterior, APL cuenta con la posibilidad de direccionar a etiquetas, independientemente del número de línea que le corresponda.

La forma general es:

```
      ETIQUETA:EXPRESION
      →(CONDICION)ρETIQUETA
[3]   ETI1:C←C+1
 :
[25]  →(C>N)ρETI1
```

Comentarios

Existen ocasiones en las que un programa es muy complicado y para hacerlo más claro en su entendimiento, es necesario introducir una serie de comentarios para no perder la secuencia del mismo.

Esto se logra en APL por medio del símbolo (⍝) (∩ sobre ∘).

Este símbolo debe escribirse al principio de la línea de comentarios y todo lo que está a su derecha lo considera como tal.

Ejemplo:

```
[3]   ⍝PARA ENCONTRAR TASA DE INTERES
```

Input/Output Conversacional

. Salida de una Función Dada

La forma más simple de desplegar el valor de una expresión de una línea durante la ejecución de una función es:

```
[4]    R←+/SCORE
[5]    'EL TOTAL ES'
[6]    R
```

Donde será desplegado es:

```
EL TOTAL ES
  . . . . . (EL VALOR DE R)
```

Una forma más general y recomendable de desplegar una expresión es el quad ⎕ usado inmediatamente a la izquierda de la flecha de asignación.

```
      ⎕←'EL TOTAL ES'
EL TOTAL ES
  . . . . . (EL VALOR DE R)
```

Formateo Diádico

Usando el quad específico es posible obtener el valor de una expresión dada; sin embargo, no puede controlar la apariencia de la salida de la información; por ejemplo, las matrices y los vectores siempre son desplegados pegados al margen izquierdo. La función de formateo diádico $A \blacktriangledown B$ es usado para controlar la apariencia de las salidas. Su forma general es:

```
      A▼B
```

donde B es el arreglo de datos a ser formateados y A es un vector escalar que describe el formato del resultado. El resultado es un arreglo de caracteres.

El formato de salida es controlado por pares de números donde el primero especifica el tamaño del campo y el segundo el número de decimales deseado.

Es posible controlar cada columna del resultado tomando un par de escalares para cada una de ellos.

Cuando el segundo elemento de la pareja es negativa, nos dice que va a imprimir de manera exponencial.

Ejemplo:

```
      M
¯31.8629  ¯39
934.8       0
  6.217   ¯467.8191
¯345          0.621
        10 2▼M
¯31.86    ¯39.00
934.80        .00
  6.22    ¯467.82
¯345.00       .62
```

El escalar que sirve para identificar el tamaño del campo en el vector de formateo puede ser cero o no existir, en cuyo caso a la hora del reporte de salida siempre existirá un espacio en blanco entre columna y columna.

```
      2▼M                              0 2▼M
 ¯31.86    ¯39.00                 ¯31.86    ¯39.00
934.80       .00                 934.80       .00
  6.22    ¯467.82                   6.22    ¯467.82
¯345.00       .62                ¯345.00       .62
```

Formateo Monádico

Existen ocasiones en que queremos imprimir resultados que contengan mezclados números y caracteres en una misma línea como por ejemplo:

EL PROMEDIO ES 6

donde 6 es el resultado de un cálculo; en APL, esto es posible mediante la concatenación y el uso del formateo monádico, cuya forma general es:

```
▼B
```

donde *B* es un arrelgo.

La forma monádica de formatear produce un arreglo de caracteres que es idéntico al argumento numérico, por ejemplo:

```
      'EL PROMEDIO ES',▼(+/9 8 3 4)÷4
EL PROMEDIO ES 6
```

Entrada Durante la Ejecución de una Función

Las funciones que fueron definidas en los capítulos anteriores requerían que los datos fueran definidos antes o a la hora de llamar la función. Sin embargo, muchas veces se requiere introducir datos durante la ejecución de la función, con el objeto de hacerla más ágil.

APL cubre esta posibilidad. Usando el quad como parte de una expresión, podemos lograr lo anterior.

```
ANS←⎕          ST←(2×⎕)÷3          →0×ι12≥⎕
```

Cuando a la derecha de una especificación nos encontramos con un ⎕: durante la ejecución de una expresión, ésta para, y el símbolo es impreso. La ejecución no continúa hasta que no se introduzca un dato desde afuera.

Una vez hecho esto, el dato reemplazará al ⎕ y la expresión es evaluada por completo.

Por ejemplo:

```
      ANS←⎕                          ST←(2×⎕)÷3
⎕:                             ⎕:
      31 72                          8 6 5
      ANS                            ST
31 72                          5.333 4 3.333
```

Es generalmente usado para introducir datos numéricos.

Si queremos introducir datos en forma de caracteres, se usa el quad con un apóstrofe en el interior (⍞) .

Rastreo

Normalmente existe necesidad de conocer el desarrollo de una variable durante la ejecución de una función; sin embargo, pocas veces podemos tener acceso a ese tipo de información. En APL podemos hacerlo empleando la función:

*T*Δ*FN*← NUMEROS DE LINEA (o una expresión cuyo valor es un número de línea).

*T*Δ seguido por el nombre de la función establece un rastreo en los números de línea indicados para esta función. Cada vez que se ejecuta esta función, el nombre de la función, el número de línea y el valor de cada línea indicada es impreso. Por ejemplo:

Para rastrear las líneas 3 y 4 de *APRIN* se hace como sigue:

```
      T∆APRIN←3 4
      100 5 APRIN 1968 1971
APRIN[3]   105
APRIN[4]   →3
APRIN[3]   110.25
APRIN[4]   →3
APRIN[3]   115.76
APRIN[4]   →0
115.76
```

El rastreo de una función es eliminado por:

```
      T∆FN←⍳0
```

Continuando con el ejemplo de *APRIN*

```
      T∆APRIN←⍳0
      100 5 APRIN 1968 1971
115.76
```

Suspensión de la Ejecución de una Función

Cada una de las siguientes circunstancias interrumpen la ejecución de una función.

- Una función llama a otra durante su ejecución.
- Un quad ocurre en la función
- Un error es encontrado durante la ejecución de la función o la tecla de ATTN es oprimida.

Funciones con Llave

Existen ocasiones en que no deseamos que otras personas conozcan como fue definida una función.

Para estos casos, existe lo que llamaremos función con llave (▼) .

Cuando nos encontramos con una función que ha sido definida, usando el símbolo ▼ la función sólo puede ser ejecutada o borrada, no puede ser editada ni rastreada.

Una vez cerrada, no puede ser abierta. Las funciones no pueden ser cerradas caprichosamente.

F) SELECCION DE DATOS Y REARREGLOS

El objeto de este tema es presentar otra manera de seleccionar datos específicos de un arreglo y otra forma de cambiar la estructura de los componentes de un arreglo.

Selección de Datos

- Compresión

La función compresión *L/A* es usada para eliminar componentes de un arreglo que no satisfagan con condiciones establecidas, dejando sólo los que si cumplan. El argumento derecho *A* define los datos originales y el argumento izquierdo *L* es un vector lógico que define la regla de compresión. Los elementos de *A* son comparados con los correspondientes componentes de *L*; donde un componente de *L* es 1, el correspondiente elemento de *A* es retenido y si es *0*, el elemento *A* es descartado. Por ejemplo:

```
      0 1 0 1 0 1 0 1/6 2 ¯3 ¯4 1 5 ¯2 9
2 ¯4 5 9
      1 1 0 1 0 0 1/'CULTURA'
CUTA
```

Generalmente, el argumento izquierdo puede ser creado por una expresión, supongamos que *V* es un vector, la expresión *(V≥0)/V* seleccionará los elementos positivos de *V*

```
      V←9 ¯8 3.2 ¯4.1 ¯7
      (V≥0)/V
9 3.2
```

El argumento derecho puede ser cualquier arreglo, y la compresión puede ser hecha sobre cualquier dimensión, usando un especificador de ejes

```
      L/[I]A
```

donde *A* es el arreglo ha ser definido y puede tener cualquier rango y tamaño; *I* indica la dimensión a través de la cual va a ser hecha la compresión; *L* es el vector lógico que define la regla de compresión.

```
      MAT
ABCD
EFGH
IJKL
      0 1 0/[1]MAT
EFGH
0 1 1 0/[2]MAT
BC
FG
JK
```

Si el argumento izquierdo es un cero o un uno, el argumento derecho es totalmente rechazado o aceptado.

```
      1/'ALL'
ALL
      0/'ALL'
```

. **Expansión**

La función compresión como vimos en el inciso anterior, sirve para eliminar o comprimir elementos de un arreglo. A veces es necesario el caso contrario: Una expansión del arreglo.

La función expansión introduce blancos en los lugares donde en el vector lógico aparezcan ceros, por ejemplo:

```
      1 1 1 0 0 0 1 1 0 0 1 1 1\'RETLR4.3'
RET   LR    4.3
```

De la misma forma que compresión, también podemos expresar un arreglo sobre cualquier dimensión, indicándolo por un especificador de ejes $L\backslash[I]A$.

. **Toma**

Compresión como vimos anteriormente, sirve para seleccionar elementos en cualquier parte de un arreglo. En ocasiones, únicamente queremos seleccionar los primeros I **ó los últimos** I **elementos de un arreglo; esto es posible en APL, gracias a la función diádica "toma"** $I\uparrow B$ **, por ejemplo:**

```
      Q←'MARSHALL'
      5↑Q
MARSH
      ¯5↑Q
SHALL
```

El argumento derecho de "toma" puede ser cualquier arreglo.

Para el caso de arreglos de dos dimensiones, trabaja de la forma siguiente:

```
      M
NOWHERE
BEDEVIL
BARKING
      ⍴M
4 7
      2 3↑M
NOW
BED
```

```
      ¯3 ¯4↑M
EVIL
NEED
KING
      2 ¯4↑M
HERE
EVIL
      ¯2 3↑M
INK
BAR
```

. Quita

Justo como la función "toma", se especializa en seleccionar los primeros o los últimos elementos de un arreglo. La función "quita" elimina todos los elementos de un arreglo, los primeros o los últimos; está definido como:

```
      A↓B
      Q←'MARSHALL'
```

Trabaja de la misma forma que "toma".

. Concatenación

Al inicio de este trabajo, el operador concatenación fue utilizado para anexar un escalar o una expresión vectorial a otra. Ahora

discutiremos cómo este operador puede ser empleado para concatenar arreglos; para tal efecto, consideramos la matriz *STOCKS* que contiene información sobre cuatro emisores de acciones.

```
      STOCKS
31.625 27.72 14.25 301.5
 2       0.2   0.17  11.08
```

El primer renglón representa al valor actual por acción y el segundo las utilidades correspondientes a los 12 últimos meses, también por acción.

Si queremos introducir un nuevo emisor de acciones, hacemos lo siguiente:

```
      NEWSTOCK
116.375 6.31
      STOCKS←STOCKS,[2]NEWSTOCK
      STOCKS
31.625 27.72 14.25 301.5 116.375
 2       0.2   0.17  11.08   6.31
```

Ahora, supongamos que como información adicional queremos incluir los dividendos pagados por acción por cada emisor.

```
      DIVIDENDOS
1.11 0.6 0 5.2 0.8
      STOCKS←STOCKS,[1]DIVIDENDOS
      STOCKS
31.625 27.75 14.25 301.5 116.375
 2       0.2   0.17  11.08   6.31
 1.11    0.6   0      5.2    0.8
```

Como se puede ver en el ejemplo anterior, es posible concatenar sobre cualquier dimensión de un arreglo. La forma general es:

```
A,[I]B
```

donde *A* es el dato al cual se le va a concatenar, *A* puede ser un arreglo de cualquier rango y tamaño; *B* es el dato a ser concatenado y puede asumir cualquier rango o tamaño que tenga *A*; *I* indica a través de qué dimensión va a ocurrir la concentración.

Rearreglo de Datos

. Reversión

A este momento, si quisiéramos reversar los componentes de un vector, tendríamos que hacer lo siguiente:

```
      V←'ARROZ'
      V[⍒⍳⍴V]
ZORRA
```

Este mismo resultado puede ser obtenido de una forma más directa, por medio del operador monádico reversión (⌽*B*)

```
      ⌽V
ZORRA
```

La forma general de este operador es:

```
      ⌽[I]A
```

donde *A* es el arreglo a ser reversado e *I* indica la dimensión sobre la cual se hará la reversión.

Si el especificador de ejes es omitido, la reversión se hará sobre la última dimensión. El símbolo ⊖ puede ser usado para hacer reversiones sobre la primera dimensión. La reversión rearregla la información sin cambiar los valores de ésta; esto es, permuta los índices asociados con la información.

```
sea     CH
        CH

ABC
DEF
GHI

        ρCH

3 3
```

Si representamos la matriz *CH* con los índices correspondientes a la posición de cada uno de sus valores, tendremos:

$$\begin{matrix} A_{11} & B_{12} & C_{13} \\ D_{21} & E_{22} & F_{23} \\ G_{31} & H_{32} & I_{33} \end{matrix}$$

Si aplicamos la reversión sobre la primera dimensión de *CH* (⌽[1]*CH* ó ⊖*CH*), los índices de los valores de la matriz queda como sigue:

A_{31} B_{32} C_{33}

D_{21} E_{22} F_{23}

G_{11} H_{12} I_{13}

que al reordenar la matriz de acuerdo a estos índices nos da:

G_{11} H_{12} I_{13}

D_{21} E_{22} F_{23}

A_{31} B_{32} C_{33}

Ahora, si aplicamos la reversión sobre la segunda dimensión de CH, los índices nos quedarán:

A_{13} B_{12} C_{11}

D_{23} E_{22} F_{12}

G_{33} H_{32} I_{13}

que reordenado nos da:

C_{11} B_{12} A_{13}

F_{21} E_{22} D_{23}

I_{31} H_{32} G_{33}

. Rotación

En el punto anterior, vimos como podemos reversar un arreglo en forma completa; sin embargo, existe la posibilidad de que queramos

controlar esta rotación; esto es, que queramos rotar "n" posiciones a un arreglo; para lo anterior, existe la función diádica conocida como rotación ($A \phi B$), donde A es el número de posiciones que queremos rotar al arreglo B.

Definimos un vector W

```
      W←'HOLA BUENOS DIAS'
      ⍴W
16
```

Si queremos rotar este vector en cuatro posiciones definimos:

```
      4 ⌽ W
BUENOS  DIASHOLA
```

si lo queremos rotar en sentido contrario:

```
      ¯4 ⌽ W
DIASHOLA BUENOS
```

En este ejemplo, vimos cómo podemos rotar un arreglo de una dimensión, pero también es aplicable a arreglos de más dimensiones. Su forma general es:

```
      A ⌽ [I]B
```

donde A es un escalar o un arreglo que nos indicará cómo será rotado el arreglo B bajo la dimensión I.

Si el argumento derecho es una matriz, ésta podrá ser rotada sobre cualquiera de sus dimensiones.

La rotación definida como $\phi[1] \circ \bullet$ produce una rotación sobre la primera dimensión (renglones). Si el argumento izquierdo es positivo, la rotación se hará de arriba para abajo; si es negativo, será de abajo para arriba.

```
      CHR
A B C D
E F G H
I J K L

      1φ[1]CHR  ∘  1 ⊖ CHR
E F G H
I J K L
A B C D

      ¯1φ[1]CHR
I J K L
A B C D
E F G H
```

La rotación definida como $\phi[2] \circ \phi$ produce una rotación sobre la segunda dimensión (columnas) si el argumento izquierdo es positivo, la rotación se hace de izquierda a derecha; y si es negativo, de derecha a izquierda.

```
      1φCHR  ∘  1φ[2]CHR
B C D A
F G H E
J K L I

      ¯1φCHR
D A B C
H E F G
L I J K
```

También es posible especificar el número de posiciones a ser rotada para cada uno de los renglones o columnas.

```
      1 2 1⌽ CHR
B C D A
G H E F
J K L I

      1 2 1 2 ⌽[1]CHR
E J G L
I B K D
A F C H

      ¯1 ¯2 1 ⌽ CHR
D A B C
G H E F
J K L I
```

. Transposición Monádica

La transposición monádica reversa el orden de todas las dimensiones de un arreglo. Su forma es: ⍉ ⍤

```
      M
 1  2  3  4
 5  6  7  8
 9 10 11 12

      PM
3 4

      N←⍉M

      PN
4 3

      N
 1  5  9
 2  6 10
 3  7 11
 4  8 12
```

Veamos ahora el caso de un arreglo de tres dimensiones:

```
      AR                    T
A B C D                     AM
E F G H                     EQ
I J K L                     IU

M N O P                     BN
Q R S T                     FR
U V W X                     JV

      PAR                   CO
                            GS
2 3 4                       KW

      T←QAR                 DP
      PT                    HT

4 3 2                       LY
```

COMUNICACION CON EL SISTEMA QUE SOPORTA A APL

COMUNICACION CON EL SISTEMA QUE SOPORTA A APL

Existen dos formas de comunicación con el sistema: A través de comandos que son independientes del lenguaje y que deben de ser usados de manera manual desde la terminal cada vez que son requeridos; y a través de variables y funciones del sistema que proveen de medios para utilizar las facilidades del lenguaje para manejar porciones del sistema, esto es, ciertos elementos de comunicación entre el lenguaje y el sistema que son identificados y definidos como variables y funciones del sistema.

La comunicación con el sistema de APL vía comandos del sistema para movimientos como salvar o llamar áreas de trabajo, están identificados con un paréntesis derecho al principio de la instrucción; este tipo de comandos no pueden formar parte de una función definida.

La comunicación con el sistema de APL por medio de variables y funciones del sistema para realizar instrucciones como cambiar las características de los espacios de trabajo, posee nombres especiales los cuales se identifican con un □ al principio. Estos nombres especiales están reservados para la comunicación con el sistema y tienen un significado específico. Como cualquier otra variable o función del lenguaje, las variables y funciones del sistema pueden ser empleadas en expresiones y funciones definidas.

En este capítulo se hace una breve descripción de todos los comandos, variables y funciones del sistema que están permitidos a los usuarios de APL .S.V.

A.- CONTENIDO DE LAS AREAS DE TRABAJO

El contenido de las áreas de trabajo son variables y funciones definidas, este contenido puede ser modificado por:

1) Especificación de variables
2) Definición de funciones
3) Copiar variables y funciones de otras áreas de trabajo
4) Agrupar variables y funciones en otras unidades más útiles

Copiado

En los capítulos anteriores vimos que la manera más común de crear variables y funciones en un área de trabajo es especificándolas y definiéndolas. Otra forma de traer funciones y variables al área de trabajo es copiándolas de otras áreas, utilizando los comandos *)COPY O)PCOPY* .

El comando *)COPY* es empleado para traer al área activa del espacio de trabajo cualquier contenido de otro espacio de trabajo.

)COPY NOMB o un conjunto de elementos de otro espacio de trabajo *)COPY NOMB OBJ1 OBJ2*, una vez que se ha copiado estos elementos un mensaje en el cual se indica la hora y la fecha en que fue salvado. Si el objeto requerido no es encontrado, un mensaje de *NOT FOUND* es enviado, por ejemplo:

```
     )VARS
V1 V2 V3
     )FNS
F1 F2
     )COPY THIS NEWF
SAVED 11.33.04 11/29/99
```

```
     )VARS
V1 V2 V3
     )FNS
F1 F2 NEWF
     )COPY CONTINUE W Z
SAVED 11.33.09 11/29/99
NOT FOUND:Z
     )VARS
V1 V2 V3 W
```

La función *NEWF* del área de trabajo *THIS* fue copiado a el área de trabajo activa, así como la variable *W* del área de trabajo *CONTINUE* , pero la variable *Z* no fue encontrado en *CONTINUE* .

```
     )COPY THIS
SAVED 12.12.12 08/10/99
     )VARS
V1 V2 V3 V4 V5 V6 V7
     )FNS
F1 F2 F3 NEWF
```

Todas las variables y las funciones del área de trabajo fueron copiadas a nuestra área de trabajo activa.

En el caso en que una variable o una función copiada tenga el mismo nombre que una ya existente en el área de trabajo activa, la variable toma el valor de la que fue copiada y el valor antiguo se pierde.

```
     A←'PRUEBA'
     )COPY THIS A
SAVED 8.11.10 12/20/99
     A
763
```

El uso del comando para copiar protegido)*PCOPY* evita el problema anterior, ya que únicamente van a ser copiados los elementos que no tengan el mismo nombre de los ya existentes en el área de trabajo activa. Los nombres que ya existan en el área activa no serán copiados y aparecerán listados después del reporte de *SAVED* como *NOT COPIED* .

```
      A←'PRUEBA'
      )PCOPY THIS A
SAVED 8.11.10 12/20/99
NOT COPIED: A
      A
PRUEBA
```

Agrupaciones

Agrupar es juntar un conjunto de variables y/o funciones bajo un mismo nombre. El primer propósito de agrupar es facilitar el copiado de un conjunto de variables y funciones en otra área de trabajo. Por ejemplo: un área de trabajo llamado *LECCIONES* contiene tres conjuntos de variables y funciones, supongamos que durante la sesión de clases sólo un conjunto es requerido.

Si *LECCION*1 es este conjunto, el comando

)*COPY LECCIONES LECCION*1

trae a nuestra área de trabajo las funciones y variables que han sido agrupadas bajo el nombre de *LECCION*1 .

Un grupo es definido por)*GROUP NOMBRE LISTA* , donde *NOMBRE* es el nombre del grupo y *LISTA* es la lista de nombres que identifican a las variables, a las funciones y a los grupos; por ejemplo:

)*GROUP LECCION1 DRILL PRAC PRUEBA ESTAT A B ANS COMPAR*

El nombre del grupo debe ser único en el área de trabajo. El mensaje *NOT GROUPED,NAME IN USE* es dado si una función o variable en el área de trabajo tenían ya ese nombre.

```
    )FNS
F1 F2 F3
    )GROUP F1 A B
NOT GROUPED,NAME IN USE
```

Usando el comando)*ERASE* en un grupo, borramos la identificación del grupo, así como los nombres de los objetos contenidos en el y sus significados.

El comando)*GRP NOMBRE* lista los nombres contenidos en el grupo *NOMBRE* .

```
    )GRP LECCION1
DRILL PRAC PRUEBA ESTAT A B ANS COMPAR
```

Nuevos nombres pueden ser añadidos a un grupo repitiendo el nombre del grupo como primer elemento de la lista del comando)*GROUP NOMBRE LISTA* .

```
)GROUP LECCION1 LECCION1 TRATAR
)GRP LECCION1
DRILL PRAC ESTAT A B ANS COMPAR TRATAR
```

Si un grupo es copiado en un área de trabajo que ya tiene un objeto con ese nombre, lo referente a ese grupo es copiado pero su definición como grupo no. El mensaje *NOT GROUPED,NAME IN USE* es desplegado.

```
)LOAD WS1
SAVED 12.12.12 08/10/99
)GRPS
SET
)GRPS SET
X Y Z
)LOAD WS2
SAVED 11.11.11. 08/10/99
)VARS
SET
)COPY WS1 SET
NOT GROUPED,NAME IN USE
)VARS
SET X Y Z
```

Si un grupo es copiado con el comando)*PCOPY*, sólo los objetos que no crean conflicto son copiados.

Preguntando sobre Objetos en un Espacio de Trabajo

Existen tres comandos básicos para preguntar sobre el contenido de un espacio de trabajo.

)VARS NOS DA LA LISTA DE VARIABLES

)FNS NOS DA LA LISTA DE FUNCIONES

)GRPS NOS DA LA LISTA DE GRUPOS

El comando *)VARS* lista únicamente las variables globales.

Cada uno de estos comandos lista los nombres en orden alfabético de la *A* a la *Z* y de la *A* a la *Z*

```
)VARS
AL AME GRY MILT RESET SET TEST TRIAL
)FNS
GULLY STY UNCL VALUE ZILCH
)GRPS
METHOD STYLE
```

Opcionalmente, cada uno de estos comandos puede ir seguido por una letra en cuyo caso únicamente serán listados los nombres que empiecen con esta letra y los subsecuentes.

```
)VARS R
RESET SET TEST TRIAL
)FNS U
UNCL VALUE ZILCH
)GRPS
STYLE
```

Los comandos anteriores no pueden ofrecernos información sobre si la variable es numérica o de caracteres, o sobre cuál es su tamaño o rango, o de cuál de las seis distintas maneras de definición de funciones es una función particular.

Borrando Objetos

Anteriormente ya hemos usado el comando)*ERASE* para borrar variables y funciones de un área de trabajo. Esto se hace listando los nombres a ser borrados.

)*ERASE FN1 VAR1 FN2 VAR2 VAR3*

El borrado de un grupo borra la definición de éste y lo referente al mismo.

Si un grupo es elemento de otro grupo que va a ser borrado, su definición es eliminada, pero lo referente a él no.

```
    )GROUP ADJ ART IVE ING AL
    )GROUP ART AN THE
    )GRPS
ADJ ART
    )VARS
AL AN ING IVE THE
    )ERASE ADJ
    )GRPS
    )VARS
AN THE
```

B.- LA BIBLIOTECA

En la sección anterior se discutió el contenido de las áreas de trabajo. La estructura que contiene y donde se pueden crear áreas de trabajo, se conoce como BIBLIOTECA. En APL, existen dos tipos de bibliotecas: las bibliotecas públicas y las privadas. Cada biblioteca pública contiene áreas de trabajo que pueden ser usadas por todos los usuarios. Las bibliotecas privadas exclusivamente pueden ser usadas por los poseedores del número de la biblioteca. Las bibliotecas están referidas por números en vez de por nombres, como es el caso de las áreas de trabajo.

Cada usuario tiene su biblioteca privada en donde almacenar sus áreas de trabajo. La identificación de esta librería es el número de entrada del usuario, (generalmente la clave de entrada del usuario consiste de dos partes: el número de entrada y una llave). Cualquiera que conozca este número puede tener acceso a la biblioteca y a sus áreas de trabajo, excepto a aquellas áreas de trabajo que están protegidas por un nombre. Normalmente nadie más que el usuario puede almacenar dentro de su biblioteca.

La cuota de áreas de trabajo es el número de éstas que puede un usuario almacenar en su biblioteca, la cual puede ser incrementada si se requiere al administrador de APL; además, de la cuota de áreas de trabajo, cada biblioteca tiene espacio para otra área llamada *CONTINUE*. Esta área de trabajo es creada automáticamente cuando la línea del usuario es desconectada o de forma explícita con)*SAVE CONTINUE* o)*CONTINUE*. Si la línea del usuario es desconectada, el trabajo en el área activa es automáticamente colocada en el área *CONTINUE* .

A partir de ese momento, la biblioteca del usuario tiene un área de trabajo llamada *CONTINUE*. Cuando el usuario vuelve a entrar en el sistema, aparece un mensaje indicando que el área de trabajo *CONTINUE* ha sido automáticamente reactivado.

Si el trabajo contenido en el área de trabajo no es necesario, el comando *)CLEAR* puede ser activado para definir nuevamente el área como un área limpia.

El comando *)SAVE COTINUE* es ejecutado a pesar de que el nombre del área de trabajo activa no es *CONTINUE*; comparemos:

```
)SAVE LECCIONES                    )SAVE CONTINUE
NOT SAVED,THIS WS IS CLEAR WS      SAVED 12.12.12 08/10/99
```

CONTINUE puede ser usada como cualquier otra área de trabajo en la biblioteca del usuario.

Las bibliotecas públicas no están asociadas con números de entrada. Estas están numeradas del 1 al 999, pero no todas contienen áreas de trabajo. Cualquiera puede usar estas bibliotecas, y las restricciones sobre éstas pueden ser establecidas por la instalación. Los espacios de trabajo pueden ser puestos dentro de bibliotecas públicas.

Lista de Bibliotecas

El comando *)LIB* lista los nombres de las áreas de trabajo en la biblioteca del usuario. El comando *LIB NUMERO* lista el nombre

de las áreas de trabajo contenidas en la biblioteca pública referenciada. No es posible para el usuario, listar el contenido de otra biblioteca privada.

```
    SIGN ON
    )LIB
WS1
CONTINUE
LECCION2
    )LIB 10
PLOT FORMAT
    )LIB 246810
IMPROPER LIBRARY REFERENCE
```

Cómo Salvar Areas de Trabajo

Salvar áreas de trabajo es la única forma de conservarlos en la biblioteca, exepto para el caso de *CONTINUE*. El primer comando)*SAVE NOMBRE* asigna el *NOMBRE* al área de trabajo activa y guarda una copia de esta área de trabajo en la biblioteca bajo el título *NOMBRE*. Una vez que *NOMBRE* es una área de trabajo establecida en la biblioteca un)*SAVE NOMBRE* es ejecutado sólo si el área de trabajo es *NOMBRE*. Si esto ocurre, el comando abreviado)*SAVE* puede ser usado siendo su efecto el mismo que el de)*SAVE NOMBRE*.

Los primeros once caracteres del nombre del área de trabajo son significativos.

Un área de trabajo es salvada en una biblioteca pública usando el comando.

)SAVE NUMERO NOMBRE

Donde el número es menor que 1000. Si el nombre de un área de trabajo activa es *CONTINUE* , éste no puede ser conservado en una biblioteca pública. Sólo el creador de un área de trabajo pública puede salvarlo. El nombre no aparece en la biblioteca privada *)LIB* , pero sí en la biblioteca pública *)LIB NOMBRE LISTA* ; no es posible salvar áreas de trabajo en otras bibliotecas privadas.

Uno puede salvar áreas de trabajo únicamente en su biblioteca y en bibliotecas públicas.

Un área de trabajo puede ser protegida de uso inautorizado por medio de una palabra clave. Esta palabra clave debe consistir de ocho caracteres o menos. Está separado del nombre del área de trabajo por dos puntos, por ejemplo:

```
)SAVE PORTAFOLIO:MIO
SAVED 12.12.12. 08/10/99
```

Después de esto, si queremos llamar esta área de trabajo, únicamente por su nombre *)LOAD PORTAFOLIO*, el mensaje *WS LOCKED* es desplegado, *)SAVE NOMBRE* sin la llave quita la protección, *)SAVE* la retiene; para cambiar la llave, simplemente hay que volver a salvar el área de trabajo con el nuevo nombre.

)SAVE PORTAFOLIO:FAMILIAR

La protección de áreas de trabajo es muy común cuando varias personas conocen el número del usuario o cuando existen áreas de trabajo con información confidencial. Un área de trabajo con llave puede ser salvada en bibliotecas públicas.

Cómo Llamar Areas de Trabajo

El comando)*LOAD NOMBRE* trae al área de trabajo activa una copia del área de trabajo *NOMBRE* que se encuentra en la biblioteca.

Si el área de trabajo *NOMBRE* ha sido salvado con una palabra clave, esta palabra clave es necesaria al usar el comando

)*LOAD NOMBRE*

)*LOAD PORTAFOLIO:FAMILIA*.

SAVED 12.15.12. 08/10/99

el olvido de esta palabra clave implica un mensaje de *WS LOCKED*.

Si conocemos el nombre, el número de biblioteca, y la llave del área de trabajo, esta área de trabajo puede ser requerida.

)*LOAD* 234567 *EL:TRABAJO*

SAVED 12.12.12.08/10/99

Cómo Borrar Areas de Trabajo

El comando)*DROP NOMBRE* borra el área de trabajo *NOMBRE* de la biblioteca. Cualquier área de trabajo en la biblioteca del

usuario puede ser borrada en cualquier momento, a pesar de que tenga una palabra clave. Conocer la palabra clave es necesario para borrar áreas de trabajo. El comando)*DROP* también puede ser usado para borrar áreas de trabajo públicas que hayan sido creadas por el usuario, pero no puede emplearse para borrar áreas de trabajo de otras bibliotecas privadas ni aquellas públicas que no hayan sido creadas por él.

Cómo Copiar de Otras Bibliotecas

Para poder copiar áreas de trabajo u objetos de otras bibliotecas, es necesario especificar el número y el nombre de la otra área o el objeto.

```
)COPY 246810 TUYO
```

C.- EL AREA DE TRABAJO, LA UNIDAD DE TRABAJO DE APL

Un nuevo espacio de trabajo tiene los siguientes atributos:

1) Su nombre *CLEAR WS*
2) Un espacio de almacenamiento (asignado por el operador del sistema) para funciones, variables, cálculos, etc.
3) Una tabla de símbolos que permite 256 nombres para ser usados en el área de trabajo.
4) Desplegado de 10 dígitos significativos, si es necesario.
5) Un máximo de 120 caracteres de salida por línea.
6) Indices en origen 1 (lo que significa que el índice del primer componente de un vector es 1).
7) Una generación aleatoria de 7*5

Las variables y las funciones del sistema permiten cambiar los atributos de su área de trabajo (excepto para el espacio de almacenamiento que sólo puede ser modificado por el operador del sistema y generalmente se aplica a todos los usuarios). Sin embargo, mientras que las variables y comandos del sistema pueden ser usadas en forma manual desde la terminal (en modo de ejecución), sólo las variables del sistema pueden ser especificadas dentro de una función definida. Comandos del sistema no pueden ser incluidas dentro de una función definida.

Las variables del sistema son valores asignados por medio de la flecha de especificación de la misma forma que se asignan valores a cualquier variable en APL. Por ejemplo: $\square PW \leftarrow 100$, nos imprimirá 100 caracteres por línea, para preguntar por el valor de $\square PW$, se hace como cualquier variable.

```
      ⎕PW
     100
```

Las variables del sistema siempre tienen un valor, si no se ha determinado un valor específico, el valor de la variable del sistema del sistema es su valor de emisión.

```
      ⎕PP
     10
```

La precisión de impresión es 10 su valor de emisión.

Dadas funciones definidas, las variables del sistema deben ser declaradas como locales. Si una variable del sistema es local, cualquier uso de ella antes de una previa especificación resulta un reporte de error *IMPLICIT* . Este reporte también resulta si se especifica una variable del sistema ignorando el conjunto de restricciones del sistema.

Los comandos y variables del sistema para modificar áreas de trabajo se muestran en la tabla siguiente:

MODIFICABLE POR

ATRIBUTO	VARIABLE DEL SISTEMA	COMANDO DEL SISTEMA	RESTRICCIONES
NOMBRE		)*WSID NOMBRE*)*SAVE NOMBRE*)*LOAD NOMBRE*)*CLEAR*	Once caracteres por *NOMBRE*
ALMACENAMIENTO Espacio de almacenamiento			Sólo por el operador, no puede ser hecho en forma individual
TABLA DE SIMBOLOS (falta 256)		)*SYMBOLS N*	$26 \leq N \leq 4241$ debe de ser ejecutado antes de usar cualquier nombre en el área de trabajo
DESPLIEGUE Dígitos significativos (falta 10)	□*PP* Print precisión	)*DIGITS N*	$1 \leq N \leq 16$
Caracteres por línea (falta 120)	□*PW* Page width	)*WIDTH N*	$30 \leq N \leq 390$
CALCULOS Indice origen (falta 1)	□*IO* Index origin	)*ORIGIN N*	$N \in 0\ 1$
FUZZ (falta $1E^{-}13$)	□*CT* Comparison Tolerance		$0 \leq N < 1$
Generación aleatoria (falta $7*5$)	□*RL* Random Link		$N \in \iota^{-}1+3*31$

Nombre

-Identificación de áreas de trabajo

El nombre de un área de trabajo puede ser determinado por el comando del sistema)*WSID*.

)*WSID* es un comando que nos sirve para preguntar el nombre del área de trabajo.

```
)WSID
TOSSES
)WSID
10 PLOT FORMAT
```

Es usado para conocer el nombre y el número de la biblioteca asociada con un área de trabajo activa; si el número de biblioteca es el mismo que el número de contabilidad, éste es omitido.

Un área de trabajo fresca siempre tiene el nombre *CLEAR WS* asociado. El nombre *CONTINUE* es asociado a un área de trabajo, cuando la última sesión fue terminada de manera prematura por una desconexión de las líneas o cuando el comando)*CONTINUE* fue usado para firmar.

Para cambiar el nombre asociado con un área de trabajo, el comando del sistema)*WSID NOMBRE* es usado donde *NOMBRE* es el nombre asignado.

```
)WSID TRIAL
WAS TOSSES
```

El cambio de nombre es frecuentemente hecho si el área de trabajo *CONTINUE* es llamado automáticamente al firmar la entrada.

El comando)*WSID* puede ser empleado para proteger un área de trabajo, anexando al nombre una palabra clave

```
)WSID TRIAL:LK
```

Cualquier pregunta vía)*WSID*, sólo proporciona el nombre y no la palabra clave.

El comando)*SAVE NOMBRE* sirve para cambiar el nombre de un área de trabajo activa.

```
     )WSID
   CLEAR WS
     )SAVE NUEVO
 12.12,12 05/10/99
     )WSID
   NUEVO
```

Almacenamiento

La tabla de símbolos es la localización donde están almacenados los nombres usados en un área de trabajo.

La tabla de símbolos contiene todos los nombres locales y globales que fueron creados en el área de trabajo desde el momento que ésta fue usada por primera vez. Con la primera especificación de un

nombre, se crea un área para él dentro de la tabla de símbolos. Aunque un nombre sea borrado, éste no desaparece de la tabla de símbolos.

El tamaño de la tabla de símbolos puede ser cambiado por el comando

)SYMBOLS N

donde *N* corre de 26 a 4241. Este comando puede ser ejecutado dentro de un área de trabajo, únicamente cuando no existe en ella ningún nombre definido.

Si es necesario cambiar el tamaño de la tabla de símbolos, el procedimiento a seguir es el siguiente:

)SAVE WS1
)CLEAR
)SYMBOLS N (UNICAMENTE SE VA A CAMBIAR EL TAMANO)
)COPY WS1

Todos los nombres aparecen en la tabla de *WS1*, a excepción de aquellos que hace tiempo no han sido usados, después de esto damos *)SAVE WS1*.

Para conocer el tamaño de la tabla de símbolos, usamos el comando:

)SYMBOLS
IS 256

Despliegue de Reportes

. Dígitos Significativos

Los cálculos en APL se hacen con aproximadamente 17 decimales, el número de decimales que va a ser desplegado va de 1 a 16, siendo la forma normal la decimal. La variable del sistema $\square PP$, precisión de impresión, su forma general es:

```
      □PP←N
```

donde N es el número de decimales que queremos que sean desplegados (entre 1 y 16); cualquier número que sea mayor o igual a $10*\square PP$ es desplegado de manera exponencial.

```
      □PP←5
      2÷3
0.66667
      V←7 1E6
7 1000000
```

Con $\square PP$ es posible hacer cambios de decimales de manera dinámica durante la ejecución de una función.

```
      ∇F 1;□PP
[4] □PP←5
  •
  •
  •
[7] □PP←12
      ∇
```

El comando del sistema *)DIGITS N* hace la misma función de *⎕PP*; para preguntar por el número de decimales empleamos *)DIGITS*.

```
    )DIGITS 5
WAS 10
    2÷3
0.66667
```

. Líneas de Salida

La densidad de la línea de salida puede variar de 30 a 390 caracteres por línea, usando la variable del sistema *⎕PW*, densidad de página, en una expresión de la forma *⎕PW←N* con *N*=30,390 .

Si la impresión excede al tamaño de la línea, ésta continua a partir de la doceava posición de la siguiente línea.

El comando del sistema *)WIDTH N* puede ser empleado también.

Cálculos

. Indice Origen

El índice origen es el número asignado a la primera posición de cada coordenada en un arreglo. Esta debe ser 0 ó 1; la posición original de APL es 1.

```
    V
1.9 6 3 5 4
```

El índice de 1.9 es *V*[1], el último componente de *V* es

V[⍴*V*], en el caso de una matriz *M*

```
    M
8 3 2
1 7 4
```

el índice de 8 es *M*[1;1]

La variable del sistema ⎕*IO* cambia el índice del origen

```
⎕IO←0
```

En origen 0, el índice de 1.9 en el vector *V* es *V*[0], el último componente de *V* es *V*[¯1+⍴*V*], el índice de 8 en la matriz *M* es *M*[0;0].

El comando del sistema)*ORIGIN N* puede ser usado también para el mismo fin, la respuesta es el índice que se tenía definido anteriormente

```
    )ORIGIN 0
WAS 1
```

Todas las funciones que tengan que ver con índices, son afectadas por el cambio de origen.

FUNCION		
Generador de índices		
⍳7	0 1 2 3 4 5 6	1 2 3 4 5 6 7
Index-of		
'ABC'⍳'CAT'	2 0 3	3 1 4
Grade up		
⍋9 6 3 8 4	2 4 1 3 0	3 5 2 4 1
Grade down		
⍒9 6 3 8 4	0 3 1 4 2	1 4 2 5 3
Roll		
?10 10 10 10	0 4 9 0	10 1 1 5
Deal		
5?5	3 1 0 2 4	3 5 1 4 2
Transposición Diádica		
1 2 0⍉2 2⍴⍳8	0 2 4 6 1 3 5 7	*DOMAIN ERROR*
2 3 1⍉2 2⍴⍳8	*DOMAIN ERROR*	1 3 5 7 2 4 6 8

El especificador de ejes que puede aparecer en la reducción, acumulación, laminación, concatenación, rotación, compresión, expansión, etc.; también son afectados por el cambio de origen.

D.- ENTRADA Y SALIDA.

Hasta ahora sabemos como firmar la entrada por medio del comando *)NUMERO* y la salida por medio de *)OFF* . Esta sección considera cómo firmar la entrada con una palabra clave y otros medios para firmar la salida.

Entrada

La parte principal del comando de entrada al sistema es el número de contabilidad del usuario *)NUMERO* , que pone a disposición de la terminal APL si éste es reconocido y si el número de entrada no está protegido por una palabra clave. *)NUMERO :PALABRA CLAVE* pone el sistema a disposición de la terminal, si éste está protegido.

```
      )999999:NIKAP
018) 12.12.12 10/10/99 RPEREZ
A P L . S V
```

La computadora responde al comando válido de entrada con:

1) El número del puerto de acceso,
2) la hora en horas, minutos y segundos,
3) la fecha,
4) el nombre del usuario tal como aparece para efecto de contabilidad-usualmente una o dos iniciales y el apellido.

Salida

Existen cuatro comandos para firmar la salida. Un reporte final es recibido al terminar una sección de trabajo:

018 15.15.15 10/10/99 *RPK*
CONNECTED 2.21.45 *TO DATE* 15.12.15
CPU TIME 0.10.10 *TO DATE* 1.13.07

La primera línea del reporte de salida incluye el número de puerto, la hora de salida, la fecha y el código del usuario que son las primeras tres letras de su nombre en la firma de entrada. La segunda línea y la tercera son estadísticas del tiempo de conexión, el tiempo en que la terminal estuvo prendida y el tiempo de CPU.

En cada renglón un par de valores es dado, el primero establece el tiempo usado durante la última sesión y el segundo es el acumulado. Cada valor es dado en horas, minutos y segundos.

Los cuatro comandos de salida son:

1) .)*OFF* Este comando simplemente firma la salida del sistema y desconecta la línea. El contenido del área de trabajo se pierde.

.)*OFF*;*LOCK* Este comando es el mismo)*OFF*, excepto que *LOCK* es una palabra clave que debe ser usada en la firma de entrada para regresar al sistema. La palabra clave debe contener ocho caracteres.)*OFF*: elimina la palabra clave)*OFF*;*NEWLOCK* la cambia.

2) .)*OFF HOLD* Este comando es parecido a)*OFF*, excepto que mantiene la línea por un minuto para que otro pueda usarla sin tener que volver a marcar.

.)*OFF HOLD*;*LOCK* Es lo mismo que)*OFF HOLD*, nada mas que incluye una llave.

3) .)*CONTINUE* Este comando salva el área de trabajo activa en *CONTINUE*, de un reporte de *SAVED* y en seguida desconecta la terminal.

.)*CONTINUE*;*LOCK* Es lo mismo que)*CONTINUE*, excepto que se establece una palabra clave.

4) .)*CONTINUE HOLD* Lo mismo que)*CONTINUE*, excepto que mantiene la línea por un minuto.

.)*CONTINUE HOLD*;*LOCK* Lo mismo que)*CONTINUE HOLD*, excepto que lleva una llave.

Si los comandos)*CONTINUE* o)*CONTINUE HOLD* son usados, la siguiente firma de entrada activa automáticamente el área de trabajo *CONTINUE*.

```
    )FNS
F1 F2 F3
    )VARS
A B C
    )CONTINUE
    )123456:LK1
015) 17.54.30. 10/22/99 ESTUDIANTE
    A P L . S V
SAVED 15.21.47 10/21/99
    )FNS
F1 F2 F3
    )VARS
A B C
```

Sin embargo, si una palabra clave está asociada con el área de trabajo *CONTINUE* , *CONTINUE* no es automáticamente activado.

E.- MENSAJES

Existe la posibilidad de enviar mensajes a otras terminales por medio de comandos del sistema. Hay dos tipos de mensajes: *)OPR* , *)OPRN* para enviar mensajes al sistema y *)MSG* , *)MSGN* para enviar mensajes a usuarios de otras terminales. Los mensajes pueden ser recibidos sólo cuando el teclado de la terminal esté bloqueado; la terminal del usuario que envía un mensaje se mantiene bloqueado hasta que el mensaje *SENT* es desplegado. Si la tecla de *ATTENTION* es oprimida antes de que el mensaje sea recibido, el reporte *MESSAGE LOST* es desplegado y la terminal del que lo envió se desbloquea.

Un mensaje ordinario enviado por el operador es antecedido por *OPR:* , cuando se envía un mensaje por el operador a todos los usuarios es antecedido por *PA:* . Un mensaje *PA:* interrumpe la ejecución de una función, otros no.

Un mensaje enviado por un usuario es precedido por su número de puerto.

Para conocer qué usuarios se encuentran usando el APL en un momento determinado, así como su número de puerto y su código asociado, el comando *)PORTS* nos da esta información.

```
)PORTS
18 RJO
21 SPA
25 RJB
31 STP
42 RJO
51 RPO
```

El comando)*PORTS CODIGO* lista todos los puertos asociados a un mismo código.

```
)PORTS RJO
18 RJO
42 RJO
```

Mensajes al Operador

El comando)*OPRN* seguido de algún texto envía un mensaje a la terminal del operador. La terminal queda libre una vez que el mensaje ha sido enviado.

El comando)*OPR* seguido por algún texto envía un mensaje a la terminal del operador. La terminal del usuario se mantiene bloqueada hasta que el mensaje ha sido enviado o la tecla de *ATTN* es oprimida.

Mensajes a Otras Terminales

El comando)*MSGN NUMERO* seguido por algún texto, envía un mensaje a la terminal cuyo número de puerto es dado. La terminal del

que lo envía se desbloquea al momento en que el mensaje es enviado. Si el mensaje es enviado a un puerto que está apagado, el número de puerto y el mensaje del que lo envió son impresos.

El comando)*MSG NUMERO* seguido de algún texto, envía un mensaje al *NUMERO* de puerto. La terminal del usuario permanece bloqueada hasta que el mensaje es regresado o la tecla de *ATTN* es oprimida.

La terminal receptora recibe el mensaje precedido por el número de puerto del que la envió, seguido por la letra *R* 21:*R* ó sólo por el número de puerto 21: . La letra *R* aparece cuando el que envió el mensaje está esperando con su terminal bloqueada por la respuesta, o sea, que el que lo envió usó el comando)*MSG*, si éste usa el comando)*MSGN*, la letra *R* no aparece.

Consideramos la siguiente secuencia como una ilustración del uso de los comandos para mensajes.

```
    )OPR OLVIDE EL NOMBRE DE MI LLAVE,AUXILIO
OPR:LO SIENTO NO TE PUEDO AYUDAR
    )OPRN GRACIAS
    )PORTS
18 RJO
21 SPA
    )MSG 21 TOMAS,ERES TU? GAD
021:R SI QUE QUIERES
    )MSG 21 CUAL ES LA LLAVE DEL WS DE PROBLEMAS?
021:PATY
    )MSGN CLARO GRACIAS
```

Control de Mensajes

Con los comandos *)MSG* y *)MSGN* cualquier usuario puede enviar mensajes a otras terminales. Sin embargo, este mensaje puede aparecer de manera inoportuna entre las líneas de un reporte final. En algunos casos, el usuario desea evitar recibir mensajes, el comando *)MSG OFF* hace esto posible.

El comando *)MSG ON* reestablece la posibilidad de recibir mensajes, después del *)MSG ON* el último *PA:* introducido durante el *)MSG OFF* es desplegado.

El comando *)MSG OFF* no puede ser salvado por lo que cuando un área de trabajo es llamada, siempre puede recibir mensajes.

Mensajes Asociados con Areas de Trabajo

Así como las áreas de trabajo pueden contener diferentes aplicaciones, éstas pueden ser accesadas por diferentes personas ajenas a su creación.

Frecuentemente, información sobre un área de trabajo tanto como sus detalles de operación, necesitan ser conocidos por la persona que la llama.

La variable del sistema *□LX* 'expresión latente', permite obtener información de manera automática al activar un área de trabajo. Después de que ha sido asignada la información a *□LX* al área de trabajo, ésta debe ser salvada. En lo sucesivo, cuando el área de trabajo es activada, la expresión previamente asignada *□LX* es ejecutada.

```
    )LOAD WS1
    □LX←'''CONTENIDO CAMBIADO EN 6/30/76'''
    )SAVE
    )LOAD WS1
CONTENIDO CAMBIADO EN 6/30/76
    ∇D
[1] 'PARA MAYOR INFORMACION'
[2] 'TECLEE DESCRIBE'
    ∇
    □LX←'D'
    )SAVE WS1
    )LOAD WS1
PARA MAYOR INFORMACION
TECLEE DESCRIBE
```

Note que en el primer ejemplo la expresión empieza y termina con tres apóstrofes, esto es necesario cuando la expresión especifica da por □*LX* va a aparecer como una cadena de caracteres después de ejecutarse. También hay que recalcar que un apóstrofe en una cadena de caracteres debe de establecerse como un par de apóstrofes.

□*LX* trabaja como si fuera el argumento derecho de una función de ejecución; esto es, puede ser comparado con ⍎□*LX*. Por lo que cualquier expresión asignada a □*LX* debe ser una cadena de caracteres. Las restricciones de esta variable son las mismas que la de la función de ejecución. En este caso particular, la cadena de caracteres asignada a □*LX*, debe ser la representación en caracteres

de una expresión ejecutable, el nombre de una función o un comentario a base de caracteres.

F.- REPORTES DE PROBLEMAS

La tabla siguiente contiene los reportes de problemas asociados con los comandos del sistema. El problema que motiva el reporte, normalmente está explícito en dicho reporte. Sugerencias para resolver el problema son dadas.

REPORTE	SUGERENCIA PARA RESOLVERLO
ALREADY SIGNED ON	1↑□*AI* para determinar el número de entrada. Encontrar al dueño del número.
IMPROPER LIBRARY REFERENCE	Evitando)*SAVE*,)*DROP* y)*LIB* número para una biblioteca privada diferente a la que hemos entrado.
INCORRECT SIGN ON	Introducir un paréntesis derecho seguido por el número de contabilidad y una palabra clave, si existe.
MESSAGE LOST	Volver a enviar el mensaje, no teclear *ATTN* hasta que el reporte de *SENT* aparezca.
NOT GROUPED,NAME IN USE	Cambiar el nombre del grupo. Borrar o cambiar el nombre del objeto conflictivo.
NOT SAVED,THIS WS IS WSID	Cambiar el nombre del área de trabajo activa usando)*WSID*
NOT SAVED,WS QUOTA USED UP	Incrementar el espacio en el área de trabajo. Usar el área de trabajo *CONTINUE*, o borrar un o área de trabajo sin uso en su librería.
NOT WITH OPEN DEFINITION	Cerrando la definición.
NUMBER IN USE	Checar el número, volviendo a meter, o notificar al operador.
NUMBER LOCKED OUT	Ver a la persona que dió la autorización.

REPORTE	SUGERENCIA PARA RESOLVERLO
NUMBER NOT IN SYSTEM	Introducir el número con la palabra clave indicada. Hablar con el operador.
OBJECT NOT FOUND	Checar variables, funciones y listas de grupos en el área de trabajo que se van a copiar.
WS LOCKED	Llamarlo o copiarlo con la palabra clave indicada.
WS NOT FOUND	Checar el nombre en los comandos de *)LOAD* o *)COPY*
SYMBOL TABLE FULL	Agrandar la tabla de símbolos, usando *)SYMBOLS* o borrando de la tabla de símbolos, nombres sin especificación por medio de *)CLEAR* y *)COPY*

G.- INFORMACION DEL SISTEMA

Dependencia del Sistema de Variables del Sistema

Un conjunto de variables del sistema está enfocado a proveer información del procesador que soporta el lenguaje. Cada variable está identificada por un nombre. Cada variable puede ser localizada por el usuario. Tal acción es inmaterial, ya que el procesador siempre lo resitúa antes de que el usuario lo pueda volver a usar. En la tabla siguiente podemos ver estas variables y la información que proporcionan:

REPRESENTACION	SIGNIFICADO	VALOR
□*WA*	Work Area	El número de bytes sin usar en el área de trabajo activa.
□*UL*	User Load	Número de usuarios activos en el sistema en un momento.
□*TS*	Time Stamp	Un vector de siete componentes que contiene: el año, el mes, el día, la hora, los minutos, los segundos y los milisegundos.
□*AI*	Account Information	Un vector de cuatro componentes conteniendo: 1. Número del usuario 2. Tiempo de computadora 3. Tiempo de conexión 4. Tiempo de llave Todos los tiempos se encuentran en milésimas de segundo y son acumulativos.
□*TT*	Terminal Type	3 para 1050, 1 para selectric y 2 para PTTC/BCD

REPRESENTACION	SIGNIFICADO	VALOR
□*LC*	Line Counter	Nos da información sobre el número de funciones en operación, suspendidas o pendientes.
□*AV*	Atomic Vector	Un vector de 256 componentes que contiene todos los caracteres representables en el sistema /370

. Variables del Sistema para Tiempo y Contabilidad

Las variables del sistema □*AI* contienen información consistente en:

1) El número de contabilidad del usuario
2) El tiempo que ha sido usado el computador desde que fue encendida la terminal
3) El tiempo de conexión
4) El tiempo acumulado que ha sido empleado para meter información por la terminal

Todos los tiempos son en milisegundos y acumulativos desde el momento en que se entró al sistema.

La variable del sistema □*TS*, time-stamp nos proporciona un vector de siete elementos donde el primero es el año y las milésimas de segundo, el último; esto es, el vector está compuesto por el año, el mes, el día, la hora, los minutos, los segundos y las milésimas de segundo.

. Ejemplos:

La expresión siguiente sirve para determinar cuánto tiempo ha

estado prendida la terminal del usuario expresado éste en horas, minutos, segundos y milésimas de segundo.

```
      0 60 60 1000⊤□AI[3]
0 1 0 567
LA EXPRESION
      0 60 60 1000⊤□AI[3 2]
0 8 37 933
0 0 9 583
```

Produce una matriz de horas, minutos, segundos y milésimas de segundo, donde el primer renglón nos da el tiempo de conexión y el segundo nos dice cuánto tiempo de CPU ha sido consumido.

Las siguientes expresiones nos permiten imprimir la fecha de distintas formas:

```
      □TS[2 3 1]
1 15 1977          (NUMERICO)
      ▼□TS[2 3 1]
1 15 1977          (CARACTER)
      (' '≠T)/T←,(0,1↓▼3 1⍴□TS[2 3 1]),'/'
1/15/1977
      MES,[□TS[2];], (▼□TS[3]),',',▼1↓□TS
ENERO 15,1977
```

En la última expresión se definió una matriz MES cuyos renglones son los nombres de los meses y cuya última columna está definida como un blanco.

Variables del sistema para obtener información del área de trabajo.

La variable $\square WA$ **, working area, determina el número de bytes que se encuentran disponibles en el área de trabajo activa. El número de espacios ocupados depende de su representación interna, ésta depende del sistema. Para el caso de APL en el sistema/370, su representación es como sigue:**

- **Cada caracter ocupa un byte**
- **Cada entero con magnitud menor o igual a 2*31 ocupa 4 bytes, cada número entero o decimal con magnitud mayor que 2*31 ocupa 8 bytes.**
- **Cada número binario ocupa un octavo de byte.**

. Ejemplo:

□WA NOS PERMITE DETERMINAR LA CANTIDAD DE ESPACIO OCUPADO POR DISTINTAS REPRESENTACIONES DE LOS MISMOS DATOS

```
    )CLEAR
CLEAR WS
    □WA
60300
    SP←□WA
    SP
60300
    M←⍳0
    SP-□WA
32
    SP←□WA
    M←5⍴1.0
    SP-□WA
40
    )CLEAR
CLEAR WS
    SP←□WA
    M←⍳0
    SP-□WA
32
    M←1=5⍴1.0
    SP-□WA
36
    )CLEAR
CLEAR WS
    SP←□WA
    SP
60300
    M←⍳0
    SP←□WA
    M←1=5⍴1.0
    SP-□WA
4
    )CLEAR
CLEAR WS
    SP←□WA
    SP
60300
    M←⍳0
    SP←□WA
    M←L5⍴1.0
    SP-□WA
20
```

. Variables del Sistema Relativos al Uso de la Terminal

□*UL*, user list, define el número de usuarios que están trabajando en ese momento.

Ejecutando □*UL* antes de)*PORTS* nos dice cuántos nombres son esperados en la lista de puertos.

□*TT*, terminal type, define la naturaleza de la terminal en uso de la forma siguiente:

3 para 1050
1 para Selectric
2 para PTTC/BCD

. Variable del Sistema para Secuencias de Funciones

□*LC*, line counter, produce un vector con el número de las líneas en que se encuentran las funciones en ejecución, suspendidas o pendientes.

□*LC*, otorga información instantánea del número de líneas de la más reciente instrucción ejecutada de una función.

. Variable del Sistema para Representación Interna

La variable □*AV*, atomic vector, es un vector de 256 componentes que contiene todos los caracteres representados en el sistema /370.

Cada caracter es representado como un byte de 8 dígitos binarios.

Actualmente, no todos los componentes tienen asignados representación de caracteres. La asignación en el sistema /370 se llama código-Z.

□*AV*, permite al usuario tener acceso a cualquiera de los caracteres definidos en código-Z, lo que le da oportunidad de accesar códigos de control como regresar el carro de la terminal, regresar espacios, cambiar de línea, etc. La tabla siguiente muestra la representación del Código-Z de los caracteres definidos.

CODIGO	CARACTER APL	SIGNIFICADO
0		Parar
1		Fin de un estatuto de caracteres
2		Fin de un estatuto que contiene una etiqueta
3		Flecha izquierda en el encabezado de una función
4		Final del caracter en el buffer
5		Sin uso
6		Sin uso
7	¨	Falsos dos puntos en representación interna
8	.	Falso punto en representación interna
9		Constante de punto flotante (Formato-E)
10		Constante lógico
11		Constante entera
12		Constante flotante
13		Constante caracter
14	[	Corchete izquierdo
15	]	Corchete derecho
16	(	Paréntesis izquierdo
17	)	Paréntesis derecho
18	;	Punto y coma
19	/	Diagonal; comprensión; reducción

CODIGO	CARACTER APL	SIGNIFICADO
20	\	Diagonal inversa; expansión; acumulación
21	←	Flecha izquierda; especificación
22	→	Flecha derecha; interacción
23	*E*	Para números en formato-E
24	¯	Signo negativo o menos alto
25	¨	Diérisis
26	+	Signo más: suma
27	-	Signo menos: resta; negación
28	×	Signo por: multiplicación
29	÷	Signo entre: división; recíproco
30	*	Estrella; exponente; exponencial
31	⌈	Techo; máximo
32	⌊	Suelo; mínimo
33	|	Residuo; módulo
34	∧	Y
35	∨	O
36	<	Menos que
37	≤	Menor o igual
38	=	Igual
39	≥	Mayor o igual

CODIGO	CARACTER APL	SIGNIFICADO
40	>	Mayor que
41	≠	Diferente
42	⍺	Alfa
43	∈	Epsilón; elemento de
44	⍳	Iota; generador de índices; index-off
45	⍴	Ro; restructuración; forma
46	⍵	Omega
47	,	Coma
48	!	Signo de exclamación; combinaciones; factorial
49	⌽	Rotación; reversa
50	⊥	Decodificar
51	⊤	Encodificar
52	○	Círculo; circular; Piveces
53	?	Aleatorio
54	~	No
55	↑	Tomar
56	↓	Quitar
57	⊂	Subordinación
58	⊃	Implantación
59	∩	Gorro
60	∪	Copa
61	_	Subrayar

CODIGO	CARACTER APL	SIGNIFICADO
62	⍉	Transpuesto
63	⌶	I-bao
64	⍬	Nulo
65	⎕	Cuad
66	⍞	Quote Cuad
67	⍟	Logaritmo
68	⍲	Y no
69	⍱	O no
70	⍝	Comentarios
71	⍋	Ordenación hacia arriba
72	⍒	Ordenación hacia abajo
73	⊖	Rotación sobre la primera dimensión, reverso
74	⌿	Comprensión, reducción sobre la primera dimensión
75	⍀	Expansión; acumulación sobre la primera dimensión
76	⌹	Multiplicación matrices; inversas
77	⍕	Formato
78	⍎	Ejecutar
79		Sin uso
80		Sin uso
81		Sin uso
82		Sin uso

CODIGO	CARACTER APL	SIGNIFICADO
83		Sin uso
84	$T\Delta$	Rastreo
85	$S\Delta$	Alto programado
86	*A*	A
87	*B*	B
88	*C*	C
89	*D*	D
90	*E*	E
91	*F*	F
92	*G*	G
93	*H*	H
94	*I*	I
95	*J*	J
96	*K*	K
97	*L*	L
98	*M*	M
99	*N*	N
100	*O*	O
101	*P*	P
102	*Q*	Q
103	*R*	R
104	*S*	S
105	*T*	T
106	*U*	U
107	*V*	V

CODIGO	CARACTER APL	SIGNIFICADO
108	*W*	W
109	*X*	X
110	*Y*	Y
111	*Z*	Z
112	Δ	Delta
113	*A*	A
114	*B*	B
115	*C*	C
116	*D*	D
117	*E*	E
118	*F*	F
119	*G*	G
120	*H*	H
121	*I*	I
122	*J*	J
123	*K*	K
124	*L*	L
125	*M*	M
126	*N*	N
127	*O*	O
128	*P*	P
129	*Q*	Q
130	*R*	R
131	*S*	S
132	*T*	T

CODIGO	CARACTER APL	SIGNIFICADO
133	U	U
134	V	V
135	W	W
136	X	X
137	Y	Y
138	Z	Z
139	Δ	Delta Subrayado
140	0	0
141	1	1
142	2	2
143	3	3
144	4	4
145	5	5
146	6	6
147	7	7
148	8	8
149	9	9
150	.	Punto
151	-	Raya sobre
152		Espacio
153	'	Apóstrofe
154	:	Dos puntos
155	∇	Del
156		Regreso de Carro

CODIGO	CARACTER APL	SIGNIFICADO
157		Fin de bloque
158		Regresa un espacio
159		Salto de línea
160		Del protegido
161		Prefijo (Círculo D para 1050)
162		Tab
164		
165		Regresa media línea
166		
167		Ejecutor de tarjetas
168		Longitud de la tabla

Entonces, para el número binario 1 0 1 0 1 representado por el vector binario *V*←0 0 0 1 0 1 0 1, la expresión □*AV*[2⊥*V*] o □*AV*[21] en origen cero admite el caracter ←. En general □*AV*[2⊥*V*] admite la representación de caracteres por el vector binario de 8 componentes *V* recíprocamente, □*AV*⍳'*Q*' permite el origen cero del caracter *Q*, Y (8⍴2)⊤□*AV*⍳'*Q*' produce la codificación de 8 elementos binarios del caracter.

. Ejemplo:

El signo de dólares

Uno no puede encimar directamente caracteres para crear un símbolo especial. Sin embargo, con □*AV*[158+□*IO*] cualquier símbolo puede ser construído. Entonces el vector de caracteres 'S',□*AV*[158+□*IO*],'|', produce $ que es una aproximación razonable al signo de dólares.

```
      COST←432.57
      'S',□AV[158+□IO],'|',0 2▼COST
$ 432.57
```

H.- LAS FUNCIONES DEL SISTEMA

Adicionalmente al conjunto de variables del sistema, APL.S.V. provee de un conjunto de funciones del sistema para manejar funciones definidas, proporcionar información sobre los nombres en un área de trabajo y para afectar la ejecución de una función, al igual que las variables, las funciones del sistema están definidas empezando con un □, de manera similar a las funciones definidas con llave; este tipo de funciones no pueden ser editadas. Estas funciones pueden ser empleadas de manera monádica o diádica con un espacio separando el nombre de la función del argumento. Cada una de éstas tienen un resultado explícito y algunas de ellas tienen un resultado implícito adicional; esto es, una acción ocurre durante la ejecución que no es aparente en el resultado explícito.

Manipulación de Funciones por Medio de Funciones del Sistema

. Representación Canónica y Establecimiento de Funciones

Existen ocasiones en que tiene grandes ventajas el poder convertir una función definida a caracteres. Por ejemplo, cuando es necesario almacenar información en otras unidades de almacenamiento. Como veremos más adelante, una función definida tiene que ser primero convertida en caracteres y luego almacenada como dato. La función □*CR* (canonical representation) nos permite convertir las funciones de definición en caracteres. La reprsentación canónica es una función monádica, cuya forma es:

```
R←□CR V
```

Donde *V* debe ser un caracter o un vector de caracteres que represente el nombre de función. Por ejemplo: si en nuestra área de trabajo existe una función llamada *IGUAL*, *V* , debe ser representada como *'IGUAL'*.

El resultado es una matriz de caracteres que representa la función.

Ni los ∇ ni los números de línea encerrados entre corchetes, aparecen en el resultado.

```
     ∇ Z←A IGUAL B;I;J
[1]  Z←(⍴J←⍴A)=⍴I←⍴B
[2]  →0×⍳~Z
[3]  →0×⍳~Z←J∧.=I
[4]  Z←~0∈A=B
     ∇
     V←⍳8
     A←8 1⍴⍳8
     A IGUAL V
0
     Q←⎕CR 'IGUAL'
     Q
Z←A IGUAL B;I;J
Z←(⍴J←⍴A)=⍴I←⍴B
→0×⍳~Z
→0×⍳~Z←I∧.=I
Z←~0∈A=B
     ⍴Q
5 15
```

La *Q* resultante es una matriz de 5×15 . La primera línea representa el encabezado de la función. Cada renglón es tan largo como el más largo de las líneas definidas y los espacios son llenados con blancos. Una vez que la función ha sido convertida en una matriz de caracteres, esto puede ser manipulado como cualquier información de caracteres.

```
      □CR      no borra la función que convierte a caracteres
 )FNS
IGUAL
 )VARS
Q
```

Si la función ya no va a ser requerida en el área de trabajo, puede ser borrada.

Si el argumento no representa el nombre de una función definida, *□CR* regresa una matriz nula de 0×0 . Si el argumento derecho no es una matriz de caracteres, un *DOMAIN ERROR* es desplegado, y si el argumento no es ni escalar ni vector, un *RANK ERROR* es el resultado.

La función del sistema *□FX* (function establishment) es el inverso de *□CR* ; esto es, transforma la matriz de caracteres en una función ejecutable.

El formato de esta función es:

```
R←□FX X
```

Donde el argumento derecho *X* es la matriz de caracteres que va a ser convertido en función ejecutable. La función □*FX* produce un resultado explícito y uno implícito. El resultado explícito es un vector de caracteres que representa el nombre de la función. Simultáneamente y de manera implícita, esta función vuelve a ser constituída como una función ejecutable.

```
     R
Z←TEST A
Z←(A>100)∧A<1000
     ρR
2 16
     )VARS
R
     )FNS
```

Para cambiar la matriz *R* a una función:

```
     □FX R
TEST
     TEST
SYNTAX ERROR
     TEST
     ^
     )VARS
R
     )FNS
TEST
```

```
      ∇TEST[□]∇
    ∇Z←TEST A
[1] Z←(A>100)∧1000
    ∇
      TEST 75
0
```

El argumento *□FX* debe ser una matriz de caracteres. Si el argumento no es un caracter, un *DOMAIN ERROR* ocurre. Si el argumento no es una matriz, un *RANK ERROR* ocurre. Aún más, la matriz de argumentos debe tener una representación válida de una función, si lo anterior no se cumple, *□FX* regresa el número del renglón en la matriz donde ocurre la falta.

```
      M
Z←FLIN V
Z←1+(ρV)-(ΦV≠0)ι1∇
      □FX M
2
```

Si existe una función en el área de trabajo cuyo nombre es el mismo que el producido por *□FX*, éste último reemplaza al anterior.

```
      ∇EQUALITY[□]∇
      ∇Z←A EQUAL B
[1] Z←A=B
      ∇
      □FX Q
EQUAL
      ∇EQUAL[□]∇
     ∇ Z←A EQUAL' B;I;J
```

```
[1] Z←(⍴J←⍴A)=⍴I←⍴B
[2] →0×⍳~Z
[3] →0×⍳~Z←J∧.=I
[4] Z←~0∈A=B
    ∇
```

⎕*FX* no puede establecer una función en el área de trabajo si el nombre de la función resultante es el nombre:

1) Una función actualmente suspéndida
2) Una etiqueta
3) El nombre de un grupo
4) El nombre de una variable

En estos casos, ⎕*FX* regresa el número del renglón de la matriz de argumentos donde el conflicto aparece.

```
EQUAL←5
⎕FX Q
1
```

El número uno indica que fue en la primera línea de la matriz donde apareció el problema.

⎕*FX* no borra la variable que va a ser convertida a función, si esta variable no va a ser usada posteriormente, debe ser borrada del área de trabajo.

Si el nombre establecido por ⎕*FX* es declarado como local en una

función, la función se establece únicamente durante la ejecución de la función , una vez que ésta ha terminado, desaparece.

```
      )CLEAR
CLEAR WS
      ∇ Z←A EQ B
[1]Z←A=B
      ∇
      M←⎕CR 'EQ'
      )ERASE EQ
      )FNS
      )VARS
M
      ∇ NONS;EQ
[1]   N←⎕FX M
[2]   C←3
[3]   A←4
[4]   ⍞←'A  ',N,' C'
      ∇
      S∆NONS←2
      NONS
NONS[2]
      )FNS
NONS
      3 EQ 4
0
      →2
0     )FNS
NONS
    3 EQ 4
SYNTAX ERROR
   3 EQ 4
     ^
```

Borrado Dinámico □EX

En la discusión de □CR y □FX , notamos que ninguno de éstos borra el objeto que ha sido cambiado de forma; para borrar un objeto sin uso, tiene que hacerse de manera explícita. El comando del sistema)ERASE puede ser usado para quitar objetos del área de trabajo, pero siempre con la intervención del usuario. El borrado dinámico nos provee de esta capacidad por medio de la función □EX, (expugne).

Es una función monádica que se representa como R←□EX M , donde M es un escalar, vector o matriz de caracteres, que representa el nombre del objeto a ser borrado.

Si el argumento es una matriz, cada renglón representa el nombre de un objeto a borrar. □EX nos da un resultado explícito de ceros y unos, donde el uno indica que el objeto fue borrado y el cero que no. Para una matriz de argumentos, □EX resulta un vector lógico de ceros y unos, donde el I-ésimo elemento de éste no va a decir si el I-ésimo renglón de la matriz fue borrado o no.

```
     )FNS
)EN1 FN2 FN3
     )VARS
A B C M
     M1
A
FN2
FN3
C
     □EX M1
1 1 1 1
    )FNS
FN1
    )VARS
B M
```

Un objeto no es borrado si:

1) El nombre identifica a una etiqueta, un grupo, una función pendiente o suspendida o funciones o variables del sistema.

2) El nombre no está bien establecido.

Si el argumento de □*EX* no es un arreglo de caracteres, resulta un *DOMAIN ERROR* . Si el rango del argumento es mayor que 2, un *RANK ERROR* ocurre □*EX* difiere de)*ERASE* en que □*EX* puede ser usado dentro de una función definida y puede borrar variables locales y globales;)*ERASE* , sólo borra variables locales. En la siguiente secuencia se vé claramente esta diferencia:

```
     ∇ F1;A;B;I
[1]  A←10
[2]  B←50
[3]  I←□EX 'A'
[4]  B
     ∇
     A←B←2
     S∆F1←3 4
     F1
F1[3]
     )VARS
A B
     A,B
10 50
     )ERASE B
     )VARS
A
     B
50
     →3
F1[4]
     )VARS
A
     A
VALUE ERROR
     A
     ∧
     B
50
     →4
50
     )VARS
A
     A
2
     B
VALUE ERROR
     B
     ∧
```

Listado de Nombres · □*NL*

El desplegado dinámico de las variables y funciones contenidas en un área de trabajo dada, es posible gracias a la función del sistema □*NL* (name list), que proporciona una matriz de caracteres con el nombre de los objetos en el área de trabajo, su formato es:

R←□*NL* *X*

Donde el argumento *X* indica los objetos que son requeridos de la forma siguiente:

1) Para requerir etiquetas
2) Para requerir variables
3) Para requerir funciones

X puede ser un escalar o un vector, cuyos valores pueden ser 1, 2 ó 3.

El resultado es una matriz que contiene la lista de nombres, donde los nombres no están en orden alfabético.

```
     )VARS
A AT BET STOP VIE WAIT ZERO
     □NL 2
STOP
VIE
A
AT
BET
ZERO
WAIT
```

Si el área de trabajo no contiene objetos del tipo especificado, $\square NL$ regresa una matriz nula de dimensión 0×0 .

$\square NL$ también puede ser empleado de forma diádica que nos permite hacer selecciones de la lista por medio del argumento izquierdo.

Su formato es:

$R\ C\ \square NL\ X$

Donde el argumento X tiene la misma aplicación que monádicamente; el argumento C , las letras con las que deben de empezar los nombres a ser listados.

```
      'ASZ' □NL 2
ZERO
AT
STOP
A
```

Clasificación de Nombres $\square NC$

La función del sistema $\square NC$ (name classification), nos permite determinar qué uso se le ha dado a un nombre. $\square NC$ es una función monádica de la forma

$R{\leftarrow}\square NC\ A$

donde A es un arreglo de caracteres de rango menor o igual que 2.

El argumento contiene un nombre en cada línea los cuales se quieren clasificar. Para cada línea $\square NC$ regresa el número de clasificación;

este número va de cero a cuatro; *R[I]* corresponde a la I-ésima línea de *A* .

El significado de los números es el siguiente:

0 Indica que el nombre no está asociado con ningún objeto en el área de trabajo.

1 Indica que el nombre está asociado con una etiqueta.

2 Indica que el nombre está asociado con una variable.

3 Indica que el nombre está asociado con una función.

4 Indica que el nombre no es permitido porque es el nombre de un grupo, de una variable del sistema o un nombre construído de forma inválida.

```
    )FNS
EQUAL TAX WA
    )VARS
A AM RATE TRY
    )GRPS
PACK
    A
TAX
PACK
RATE
DO-IT
NEW
    ⎕NC A
3 4 2 4 0
```

Funciones del Sistema que Afectan la Ejecución de una Función

. Suspensión de Funciones $\square DL$

En una situación donde dos funciones en distintas áreas de trabajo pueden interactuar, una de éstas debe esperar por la otra o en una instrucción dada al computador. Una pausa es necesaria para considerar la respuesta antes de presentar una nueva alternativa. La función del sistema $\square DL$ (delay) presentada en forma monádica produce una pausa en la ejecución de la instrucción donde ésta sea definida, su formato es:

$R \leftarrow \square DL\ S$

La pausa en segundos es dada por el argumento S .

El resultado explícito de $\square DL$ es un valor escalar que representa el tiempo en segundos en que estuvo detenida la función.

La función $\square DL$ usa una cantidad despreciable del tiempo de computadora, relacionada con el valor de S .

Si una función es iniciada y no completada, el usuario puede terminarla por medio del *ATTN* . Si el argumento S no es un escalar o un vector de componente, ocurre un *RANK ERROR* . Si el argumento no es numérico, un *DOMAIN ERROR* ocurre.

. Ejemplo:

Existen ocasiones en que es necesario desplegar un reporte cada determinado número de segundos, aproximadamente cada N segundos, el

elemento de escritura se moverá dos espacios hacia adelante y hacia atrás para dar la impresión de movimiento.

```
     ∇ MUESTRA N;⎕IO⎕TOD;J;K;I
[1] ⎕IO←0
[2] TOD←3 9⍴'DIAS TARDESNOCHES'
[3] K←'OA'
[4] R: 'BUEN' ,N[2],TOD[+/⎕TS[3]>12 18;]
[5] PROSA N[1]
[6] J←⎕NL N[1]÷2
[7] ⎕AV[152 158 152 158]
[8] J←⎕DL N[1]÷2
[9] →R
     ∇
     ∇ PROSA N[1]
[1] 'ESTO ES UNA DEMOSTRACION'
[2] 'ESTE REPORTE SERA REPETIDO CADA ',(⍕N[1]),' SEGUNDOS'
     ∇
```

APL APLICADO A LA PLANEACION FINANCIERA DE UNA EMPRESA MANUFACTURERA

El objeto de este capítulo es presentar la ventaja que representa el uso de modelos en APL para el desarrollo de la Planeación Financiera de una empresa manufacturera.

Antes de empezar con aplicaciones específicas es conveniente presentar las diferentes etapas de planeación de una empresa en general.

La Planeación Financiera cuenta con tres fases que son:

- A largo plazo (cinco a diez años)
- A mediano plazo (uno o dos años)
- A corto plazo (menos de un año)

La planeación a largo plazo es aquella que se desarrolla tomando como base principal las estrategias corporativas; esto es, tomando en consideración en forma global qué es lo que se espera de la empresa en un futuro lejano. Para desarrollar este tipo de planeación hay que tomar en cuenta tendencias de los resultados obtenidos en el pasado, a las cuales deben de adicionarse las mencionadas estrategias y factores económicos y financieros esperados.

En base a lo anterior, en la planeación a largo plazo únicamente se obtendrán datos globales de lo que se espera en el futuro.

La planeación a mediano plazo es la fase más importante en este proceso, ya que es aquí donde se dearrollará el presupuesto de operación que regirá a la empresa durante el siguiente año y será contra este presupuesto contra el que se comparen los resultados reales obtenidos en la operación.

Debido a lo anterior, este presupuesto deberá elaborarse con un gran

detalle y utilizando las mejores técnicas de planeación, así mismo considerando todos los factores internos como externos que de una u otra forma puedan afectar a la operación futura de la empresa.

Por útlimo se tiene la planeación a corto plazo cuyo objetivo es hacer las correcciones al presupuesto en base a los resultados reales obtenidos o a cambios que afecten los resultados de la empresa y que sean detectados antes de que se realicen.

Con el propósito de centrarnos en el objetivo de este capítulo, a continuación se hace la descripción de la empresa manufacturera en cuestión, así como de la metodología empleada en la planeación de la misma:

Esta empresa produce equipos electrónicos los cuales son vendidos tanto localmente como en el extranjero. Los costos en los que se incurre para producir estos equipos están agrupados de la siguiente forma:

a) Costos de materia prima (partes)
b) Costos de mano de obra directa
c) Costos y gastos indirectos

Como es claro entender, para poder producir un equipo lo primero que se necesita son las partes que lo integran, segundo la mano de obra que armará el equipo y por último el personal y los gastos indirectos necesarios para operar la empresa, los cuales se deben ver reflejados en el costo de los equipos a producir.

El costo de producción de los equipos se obtiene en base a movimientos de inventarios utilizando costos promedio. Una vez que estos equipos han sido producidos pasan a un inventario de productos terminados, en el

cual permanecen hasta que son vendidos, este inventario también se maneja por costos promedio.

De lo anterior, podemos ver que para vender equipos es necesario haberlos producido previamente; esto es, que la producción va en función de las ventas esperadas, también se puede ver que para poder producir los equipos necesarios para poder satisfacer la demanda es necesario contar con las partes suficientes para poder llevar a cabo esta producción; esto quiere decir, que deberemos mantener un nivel de inventarios adecuado de cada una de las partes que componen el equipo.

En el caso particular de esta empresa, las partes tienen cuatro orígenes que son:

- Local
- Brasil
- Argentina
- Estados Unidos

Esto es muy importante considerarlo, ya que como veremos más adelante para efectos de planeación agruparemos todas las partes en sólo cuatro inventarios de acuerdo a su origen.

Una vez obtenidos los ingresos y el costo de ventas relativo a los mismos, se podrá obtener si existe o no utilidad en la operación de la empresa.

Antes de aplicarle el pago del impuesto sobre el ingreso global de las empresas para obtener la utilidad neta, es necesario considerar otros ingresos y gastos en los que se haya incurrido por operar la empresa,

tales como los intereses que tendrán que pagarse sobre créditos obtenidos para financiar la operación.

Al igual que en la operación real de la empresa para poder desarrollar la planeación de ésta, se sigue el mismo procedimiento nada mas que en vez de manejar cifras reales se manejan cifras estimadas.

Al igual que en la operación real, en planeación el renglón que se deberá obtener son las ventas. En el caso de nuestra empresa, éste se obtiene principalmente en base a tendencias utilizando series de tiempo o un método similar, una vez obtenidas las ventas proyectadas de acuerdo a la experiencia se le impactarán las nuevas estrategias de mercadotecnia, planeadas a llevarse a cabo en el período en cuestión tales como nuevos alicientes a los vendedores, campañas de publicidad, incorporación de nueva tecnología, etc. La elaboración de esta parte corresponde al área de mercadotecnia.

Una vez que se han obtenido estos pronósticos, corresponderá al área de financiera determinar cuáles serán los costos y gastos de estas ventas para determinar si existirá utilidad o pérdida en la operación de nuestra empresa.

Si los resultados obtenidos no están de acuerdo con lo esperado por la dirección, se procederá a presentar nuevas alternativas hasta encontrar una que los satisfaga.

Sin duda la parte más compleja de calcular en este proceso, es la que se refiere al movimiento de inventarios tanto para obtener el costo de producción como el costo de ventas, ya que es en donde se maneja un

mayor volumen de información y se llevan a cabo gran cantidad de cálculos, los cuales sin embargo, son siempre hechos bajo la misma rutina.

Es en este punto donde APL tiene una gran aplicación, ya que con un modelo de inventarios simple se pueden obtener resultados rápidos y confiables, lo cual nos permitirá presentar varias alternativas en un mínimo de tiempo.

Para poder hacer más eficiente este sistema de planeación, las partes se agrupan de acuerdo a su origen; lo anterior, con objeto de evitar el hacer los movimientos de inventarios para cada una de las partes que componen el equipo, con la cual proyectaremos únicamente cuatro inventarios, uno por las partes de México, otro por las de Brasil, otro por las de Argentina y otro por las partes provenientes de los Estados Unidos.

A continuación se hace una descripción del sistema propuesto, esta descripción se divide en tres partes que son:

- ° Datos de entrada
- ° Proceso
- ° Reportes de salida

Datos de Entrada

A) Datos Generales

- ° Número de meses a proyectar
- ° Número de productos a considerar
- ° Nombre de los productos

B) Datos Comunes a todos los Productos

- ° Tipo de cambio
- ° Porcentajes de fletes por origen
- ° Meses de inventarios por origen
- ° Porcentajes de derechos por origen

C) Datos Específicos a cada Producto

- ° Costo de compra de juegos de partes por origen
- ° Producción de unidades
- ° Inventario inicial de partes por origen en miles de dólares, miles de pesos y número de juegos
- ° Compra mínima preestablecida por origen
- ° Inventario inicial de productos terminados en miles de dólares, miles de pesos y unidades
- ° Estándares anuales de partes por origen, mano de obra directa y gastos indirectos
- ° Venta de unidades por destino

El objeto de manejar dólares y pesos se debe a que esta empresa compra partes y vende sus unidades, tanto localmente como al exterior, ya que la moneda internacional es indiscutiblemente el dólar.

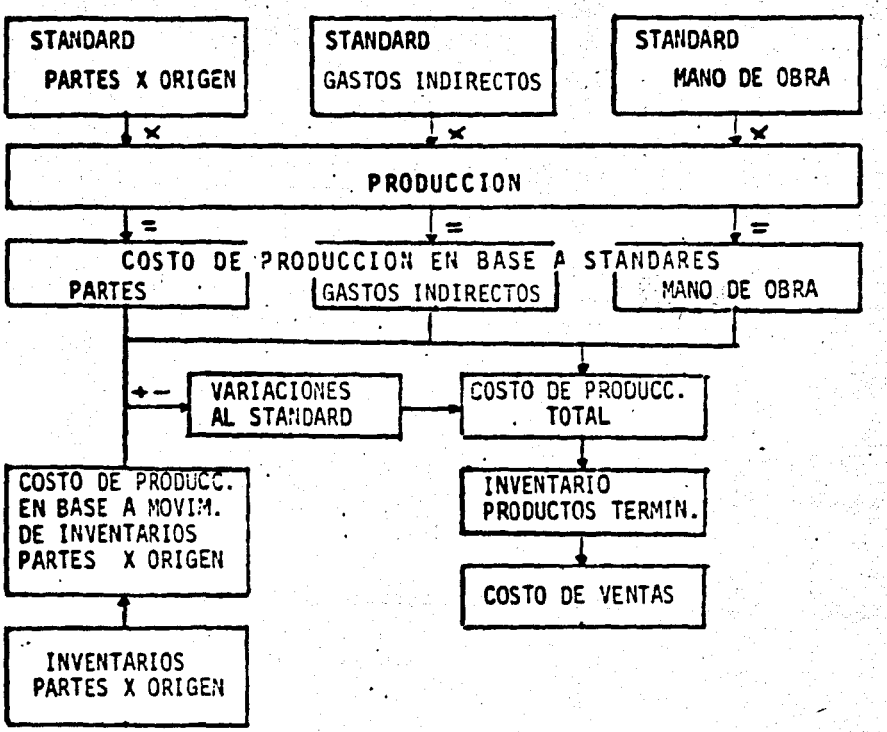
Proceso

El proceso de este sistema se ha dividido en cuatro partes, que son:

- ° Explosión de inventarios de partes por origen
- ° Integración del costo de producción por producto
- ° Caso especial del costo de producción
- ° Inventario de productos terminados y costo de ventas.

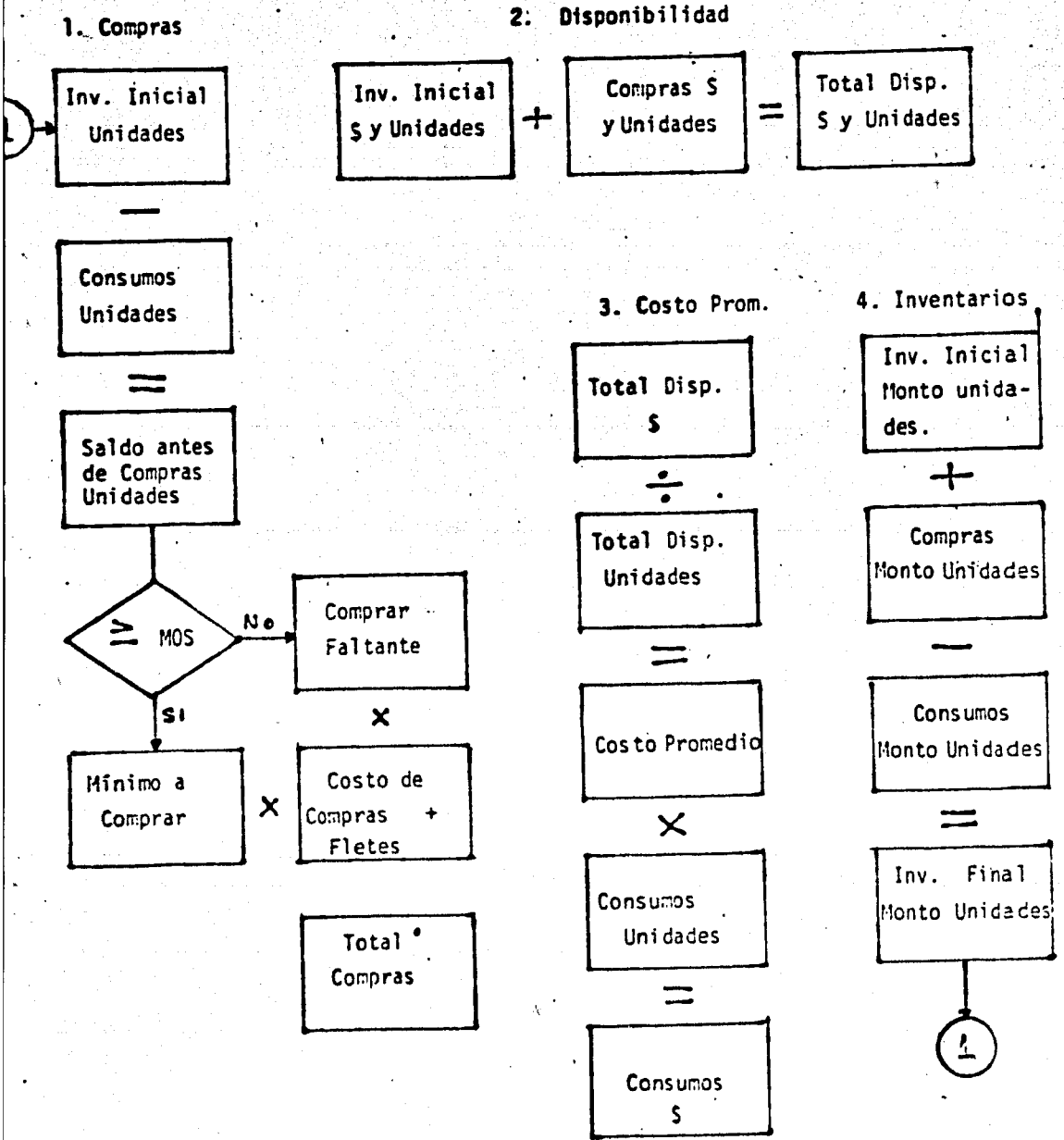
A continuación se presenta, primero, el diagrama general del sistema y en seguida los diagramas y una breve explicación de cada uno de los sub-procesos que lo componen.

DIAGRAMA GENERAL



INVENTARIOS PARTES POR ORIGEN

- D I A G R A M A -



Esta primera etapa del proceso se divide a su vez en cuatro que son:

1) Determinación de necesidades de compra en la cual se calcula en base a los niveles de inventarios en juegos de partes previstos si es necesario o no hacer nuevas compras para mantener los niveles deseados.

2) En esta etapa se obtiene el total de partes disponibles en monto y juegos sumando el inventario inicial y las compras.

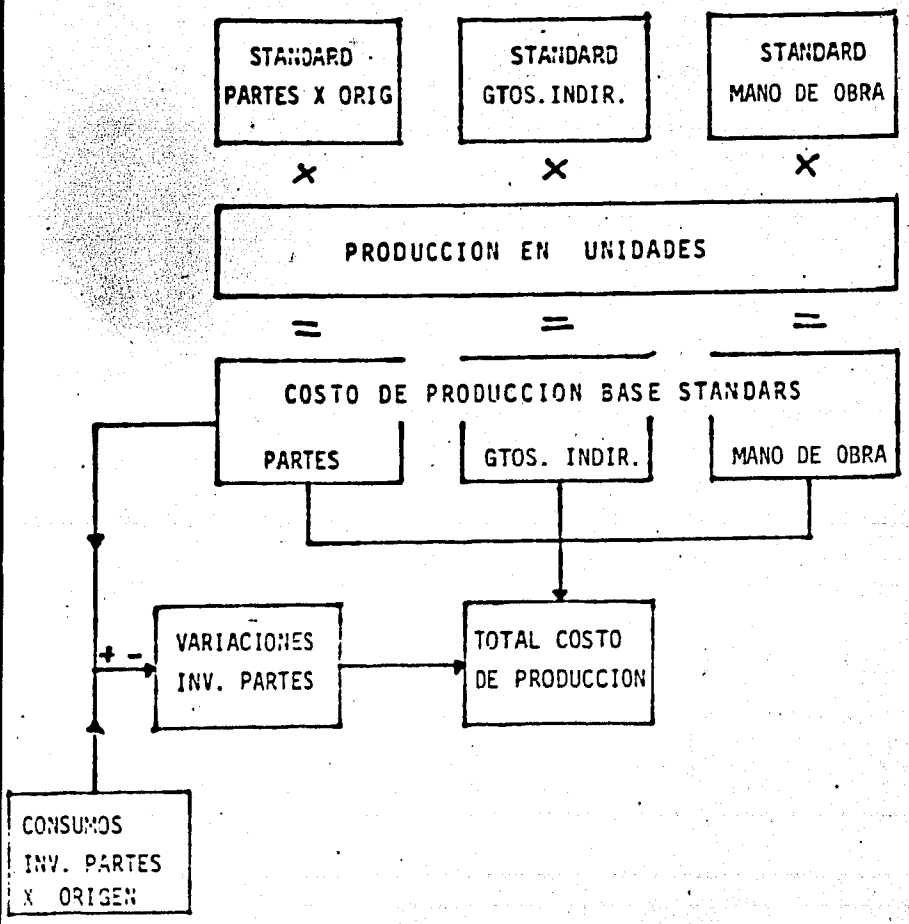
3) En esta parte, en base al total disponible se obtiene el costo promedio por juego de partes, el cual se multiplica por los consumos de juegos (producción), obteniéndose así el consumo en monto valuado a costos promedio.

4) Aquí se hace el movimiento de inventarios, con objeto de obtener el inventario final, mismo que será el inicial del mes siguiente.

Después de este paso, se inicia nuevamente el proceso.

INTEGRACION COSTO DE PRODUCCION X PRODUCTO

- D I A G R A M A -



Este proceso tiene como objeto integrar el costo de producción, el cual como se dijo anteriormente, está compuesto por:

- ° Partes por origen
- ° Mano de obra directa
- ° Gastos indirectos

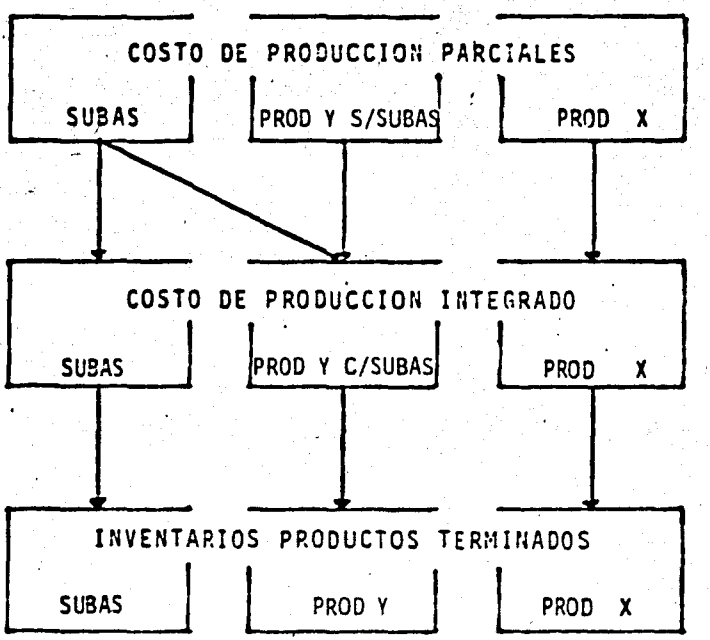
1) Se determinan estándares para cada uno de los conceptos arriba mencionados, los cuales son multiplicados por la producción presupuestada, obteniéndose así el costo de producción en base a estándares.

2) Para el caso de partes, el resultado obtenido es comparado con los consumos o promedios obtenidos del movimiento de inventarios (diagrama anterior), la variación resultante se integra también al costo de producción quedando finalmente el costo del producto integrado de la siguiente forma:

- ° Partes por origen
- ° Mano de obra directa — En base a estándares
- ° Gastos indirectos
- ° Variaciones al estándar — (+ -)

CASO ESPECIAL COSTO DE PRODUCCION

- D I A G R A M A -



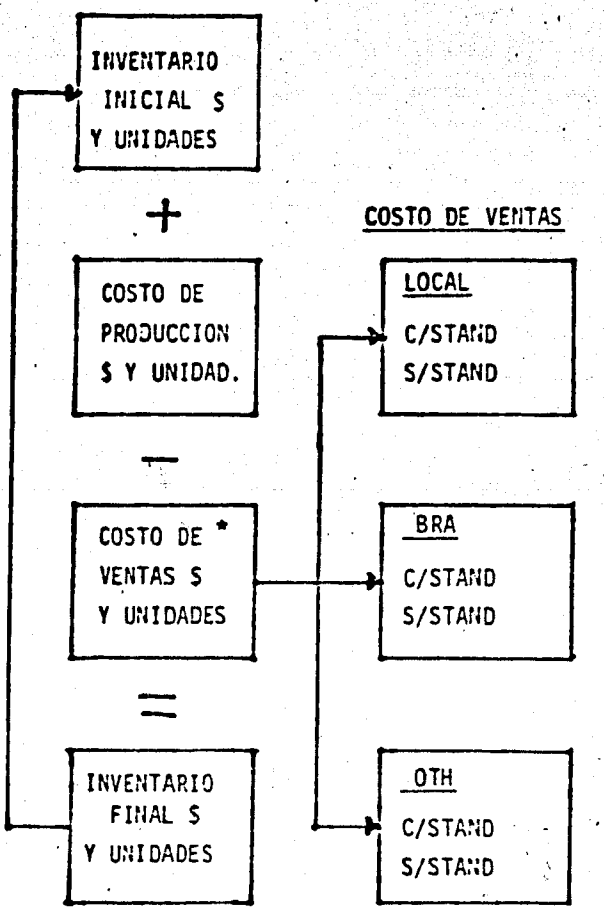
Existen ocasiones en que algunos de nuestros productos tengan subensambles que puedan ser vendidos como tales, o sea, sin necesidad de que estén integrados al producto final o también en que estos subensambles sean comunes a varios productos.

Previendo lo anterior, el sistema debe de tomar en consideración esas alternativas.

El problema básico en estos casos radica en los inventarios de partes, ya que si queremos mantener el costo promedio deberemos de manejar estos subensambles por separado, con objeto de obtener su costo de producción específico, una vez que hayamos obtenido este costo, deberemos de integrarlo al costo total de las máquinas a las que éste sea común, o en el caso en que éste se venda, directamente pasará a un inventario propio de productos terminados.

INVENTARIO DE PRODUCTOS TERMINADOS Y COSTO DE VENTAS

- DIAGRAMA -



* Adicional al resultado obtenido del movimiento de inventarios, en este paso se calculan derechos para ventas locales y el costo de stands cuando la máquina lo incluya

Esta última etapa del proceso es la que se encarga de calcular los inventarios de productos terminados, siendo las entradas a este inventario los costos de producción obtenidos en las etapas anteriores y las salidas, el costo de ventas.

El costo de ventas obtenido en esta etapa será el que irá al estado de resultados.

Este sistema tiene una gran flexibilidad, ya que una vez que tengamos guardado en archivo los datos de entrada por medio de matrices, podremos cambiar cualquier dato yendo directamente a éste y así con los datos actualizados podremos volver a correr el sistema, obteniendo así gran número de alternativas, lo que nos permitirá decidirnos por la que la dirección considere como óptima.

Este sistema también proveerá de información al balance en lo que se refiere a saldos de inventarios, tanto de partes como de productos terminados, y al estado de flujo de fondos en lo referente a las compras de estas partes.

Reportes de Salida

Por último, presento los reportes que, desde mi punto de vista, son los que este sistema deberá emitir:

- Inventarios de partes por origen por producto
- Inventario de partes por producto
- Inventario de partes total compañía

- Reporte de costo de producción integrado por producto
- Inventario de productos terminados por producto
- Inventario de productos terminados total compañía
- Reporte de costo de ventas dividido por producto, destino y total

A continuación se presentan copias de los reportes de un sistema que ha sido desarrollado por mí y que se apega a las características del mencionado en este apéndice.

EMPRESA MANUFACTURERA X
COSTO DE VENTAS
(000 PS)

	AGO	SEP	OCT	NOV	TOTAL
3100					
LOCAL /STAND	632	141	93	274	1139
BRA /STAND	0	0	0	0	0
OTR /STAND	40	0	0	0	40
TOTAL /STAND	672	141	93	274	1179
LOCAL /STAND	0	0	0	0	0
BRA /STAND	913	612	519	0	2044
OTR /STAND	182	162	0	402	746
TOTAL /STAND	1095	774	519	402	2790
TOTAL	1767	915	612	676	3969
SUBAS					
LOCAL /STAND	520	430	498	453	1901
BRA /STAND	0	0	0	0	0
OTR /STAND	0	0	0	0	0
TOTAL /STAND	520	430	498	453	1901
TOTAL	520	430	498	453	1901
COSTO DE VENTAS	2287	1344	1110	1129	5870

EMPRESA MANUFACTURERA X
INVENTARIO DE PRODUCTOS TERMINADOS
(000 PS)

	AGO	SEP	OCT	NOV	TOTAL
INVENTARIO INIC:					
UNIDADES	265	0	0	268	265
LOCAL	0	0	0	0	0
BRASIL	0	0	0	0	0
ARGENTINA	0	0	0	0	0
USA	0	0	0	0	0
DERECHOS	26	0	0	0	26
GASTO DE COMPRAS	13	0	0	0	13
VARIACION STD	458	0	0	475	458
TOTAL PARTES	497	0	0	475	497
G. INDIRECTOS	107	0	0	104	107
M. DE OBRA	10	0	0	9	10
TOTAL	614	0	0	587	614
COSTO DE PROD:					
UNIDADES	667	551	638	580	2436
LOCAL	0	0	0	0	0
BRASIL	0	0	0	0	0
ARGENTINA	0	0	0	0	0
USA	0	0	0	0	0
DERECHOS	0	0	0	0	0
GASTO DE COMPRAS	0	0	0	0	0
VARIACION STD	1182	977	1130	1018	4307
TOTAL PARTES	1182	977	1130	1018	4307
G. INDIRECTOS	259	214	248	225	945
M. DE OBRA	21	18	20	19	78
TOTAL	1462	1209	1398	1262	5330
COSTO DE VENTAS					
UNIDADES	932	551	370	396	2249
LOCAL	0	0	0	0	0
BRASIL	0	0	0	0	0
ARGENTINA	0	0	0	0	0
USA	0	0	0	0	0
DERECHOS	26	0	0	0	26
GASTO DE COMPRAS	13	0	0	0	13
VARIACION STD	1640	977	655	697	3969
TOTAL PARTES	1679	977	655	697	4008
G. INDIRECTOS	366	214	144	154	877
M. DE OBRA	31	18	12	13	73
TOTAL	2076	1209	811	864	4959
DERECHOS	57	19	12	36	125
STANDS	28	6	4	12	51
TOTAL C/VENTAS	2161	1234	827	912	5135
INVENTARIO FINAL					
UNIDADES	0	0	268	452	452
LOCAL	0	0	0	0	0
BRASIL	0	0	0	0	0
ARGENTINA	0	0	0	0	0
USA	0	0	0	0	0
DERECHOS	0	0	0	0	0
GASTO DE COMPRAS	0	0	0	0	0
VARIACION STD	0	0	475	796	796
TOTAL PARTES	0	0	475	796	796
G. INDIRECTOS	0	0	104	175	175
M. DE OBRA	0	0	9	14	14
TOTAL	0	0	587	986	986

FECHA: SEP.7,1978
HORA : 13:19:13

EMPRESA MANUFACTURERA X
COSTO DE PRODUCCION
(000 PS)

	AGO	SEP	OCT	NOV	TOTAL
LOCAL	47	42	53	52	194
BRASIL	143	120	140	128	530
ARGENTINA	323	268	309	274	1175
USA	288	233	263	231	1014
TOTAL PARTES	801	663	764	685	2913
GASTO DE COMPRAS	0	0	0	0	0
TOTAL	801	663	764	685	2913

FECHA: SEP,7,1978
HORA : 13:34:55

EMPRESA MANUFACTURERA X
INVENTARIO DE PARTES
(000 PS)

	AGO	SEP	OCT	NOV	TOTAL
INVENTARIO INIC:					
LOCAL	831	784	764	712	831
BRASIL	223	804	936	1046	223
ARGENTINA	1274	1241	1260	1223	1274
USA	1820	1723	1718	1643	1820
TOTAL	4148	4551	4678	4625	4148
COMPRAS:					
LOCAL	140	140	140	150	570
BRASIL	734	260	260	260	1512
ARGENTINA	319	312	300	530	1460
USA	392	393	377	693	1855
TOTAL	1585	1104	1076	1632	5398
CONSUMOS:					
LOCAL	187	160	193	185	724
BRASIL	153	128	149	136	566
ARGENTINA	352	292	336	299	1280
USA	489	398	452	398	1737
TOTAL	1182	977	1130	1018	4307
GASTO DE COMPRAS	0	0	0	0	0
TOTAL	1182	977	1130	1018	4307
INVENTARIO FINAL:					
LOCAL	784	764	712	677	677
BRASIL	804	936	1046	1170	1170
ARGENTINA	1241	1260	1223	1454	1454
USA	1723	1718	1643	1937	1937
TOTAL	4551	4678	4625	5238	5238
MEMO:					
C SIN FLETES	1529	1069	1042	1580	5220
FLETES	56	35	34	53	177

EMPRESA MANUFACTURERA X
INVENTARIO DE PARTES POR ORIGEN
(000 PS)

LOCAL

	AGO	SEP	OCT	NOV	TOTAL
INVENTARIO INIC:					
MONTO (000)	193	216	245	262	193
SETS	3022	3100	3216	3162	3022
C/SET	64	70	76	83	64
COMPRAS:					
MONTO (000)	70	70	70	70	280
SETS	745	667	584	523	2519
C/SET	94	105	120	134	111
CONSUMOS:					
MONTO (000)	47	42	53	52	194
SETS	667	551	638	580	2436
C/SET	70	76	83	90	79
INVENTARIO FINAL:					
MONTO /(000)	216	245	262	280	280
SETS	3100	3216	3162	3105	3105
C/SET	70	76	83	90	90
MEMO:					
C SIN/FLETES	70	70	70	70	280
FLETES	0	0	0	0	0

BRASIL

	AGO	SEP	OCT	NOV	TOTAL
INVENTARIO INIC:					
MONTO (000)	208	753	877	981	208
SETS	1083	3509	4036	4480	1083
C/SET	192	215	217	219	192
COMPRAS:					
MONTO (000)	688	244	244	244	1420
SETS	3093	1080	1080	1080	6333
C/SET	223	226	226	226	224
CONSUMOS:					
MONTO (000)	143	120	140	128	530
SETS	667	551	638	580	2436
C/SET	215	217	219	220	218
INVENTARIO FINAL:					
MONTO (000)	753	877	981	1097	1097
SETS	3509	4038	4480	4980	4980
C/SET	215	217	219	220	220
MEMO:					
C SIN/FLETES	659	233	233	233	1359
FLETES	30	10	10	10	61

ARGENTINA

	AGO	SEP	OCT	NOV	TOTAL
INVENTARIO INIC:					
MONTO (000)	1170	1139	1157	1123	1170
SETS	2422	2349	2378	2320	2422
C/SET	483	485	486	484	483
COMPRAS:					
MONTO (000)	293	286	274	486	1339
SETS	594	580	580	1080	2834
C/SET	493	493	473	450	472
CONSUMOS:					
MONTO (000)	323	268	309	274	1175
SETS	667	551	638	580	2436
C/SET	485	486	484	473	482
INVENTARIO FINAL:					
MONTO (000)	1139	1157	1123	1334	1334
SETS	2349	2378	2320	2820	2820
C/SET	485	486	484	473	473
MEMO:					
C SIN/FLETES	263	276	265	470	1294
FLETES	10	10	9	16	45

USA

	AGO	SEP	OCT	NOV	TOTAL
INVENTARIO INIC:					
MONTO (000)	1078	1014	1005	957	1078
SETS	2473	2349	2378	2320	2473
C/SET	436	432	423	412	436
COMPRAS:					
MONTO (000)	224	224	215	395	1057
SETS	543	580	580	1080	2783
C/SET	412	386	371	365	380
CONSUMOS:					
MONTO (000)	288	233	263	231	1014
SETS	667	551	638	580	2436
C/SET	432	423	412	397	416
INVENTARIO FINAL:					
MONTO (000)	1014	1005	957	1121	1121
SETS	2349	2378	2320	2820	2820
C/SET	432	423	412	397	397
MEMO:					
C SIN/FLETES	216	216	208	381	1021
FLETES	8	8	7	13	36

CONCLUSIONES

Como vimos en el primer capítulo, es sin duda el Teleproceso uno de los adelantos más significativos en computación, ya que permite llevar el procesamiento de datos a cualquier parte, sin necesidad de tener que desembolsar grandes sumas de dinero y nos provee de la capacidad de manejar un gran computador por medio de una pequeña terminal de una manera simple y eficiente.

Dentro de los lenguajes que existen disponibles a la fecha para ser usados de manera interactiva, podemos concluir que APL es el más poderoso en base a los siguientes puntos:

- Fue pensado desde su origen como un lenguaje interactivo, lo cual tiene como ventaja que su lógica interna y su forma de uso es más simple que la de aquellos lenguajes que, aunque en la actualidad han sido adaptados para ser usados como interactivos, originalmente fueron creados para manejarse de forma tradicional.

- La estructura bajo la cual está hecho nos permite manejarlo de una manera fácil y eficiente, ya que como vimos sus operadores primitivos son una serie de funciones que por sí solas facilitan dramáticamente cualquier problema a ser resuelto por medio del lenguaje, asimismo la capacidad que tiene APL para manejar lo mismo escalares, que vectores, que matrices o que arreglos de dimensión, N nos permite eficientar al máximo la programación.

- Otra característica que tiene APL es que el concepto de dimensionamiento y definición de archivos no existe, por lo cual no es necesario tener experiencia anterior en programación para poder manejarlo.

./.

° Si dividimos los trabajos que puede efectuar un computador en dos grandes grupos:

1.- Aquellos cuyos datos de entrada/salida representan un gran volumen.

2.- Aquellos cuyos datos de entrada/salida representan poco volumen.

Podemos concluir que APL se encasilla dentro del segundo grupo, ya que es manejado por medio de una terminal cuya única forma de introducción de datos es un teclado similar al de una máquina de escribir y que cuenta con una impresora de baja velocidad (30 caracteres por segundo como máximo).

Sin embargo, para el campo de aplicación para el que fue creado, no es necesario que cuente con otro tipo de aditamentos.

Este campo de aplicación es como se dijo en la introducción, la Investigación de Operaciones, la Estadística, los negocios y en general el de los cálculos matemáticos cuyos datos de entrada/salida son de poco volumen.

B I B L I O G R A F I A

INVERSON K. E., A PROGRAMMING LANGUAGE, WILEY, 1972

APL SHARED VARIABLE (APLSV) USERS GUIDE, IBM, 1975

POLIVKA R. Y S. PAKIN., APL: THE LANGUAGE AND ITS USAGE, PRENTICE HALL, 1975

GILMAN L., AND A. J. ROSE, APL - AN INTERACTIVE APPROACH, 2ND ED. WILEY, 1974

XEROX - APL LANGUAGE & OPERATIONS REFERENCE MANUAL, XEROX, 1975

INTRODUCTION TO SHARP APL, I. P. SHARP, 1976