

ALGORITMIA

Capítulo 1: Introducción a los sistemas de Cómputo

1.1 Definición de computadora

Una computadora es una máquina electrónica digital capaz de procesar información y producir datos de salida para lo cual requiere de ciertos datos de entrada. El término digital alude al hecho de que la información almacenada y procesada por la computadora esta representada mediante códigos numéricos binarios formados por ceros y unos (0 y 1) conocidos como bits.

Lo sorprendente de las computadoras es que pueden realizar operaciones complejas cuando sus circuitos electrónicos solo pueden comparar dos bits o cambiar un bit de 0 a 1. ¿Cómo es esto posible?, pues debido a las altas velocidades con que se ejecutan estas operaciones sencillas.

Para los informáticos hay una diferencia entre **datos** e **información**. **Dato** vendría a ser la representación de algún hecho, concepto o entidad real, es la materia prima de la información. **Información** vendría a ser el resultado del procesamiento de los datos. Para este curso no haremos distinción entre dato e información sino que hablaremos de datos de entrada y datos de salida.

Un sistema de procesamiento de la información involucra tres componentes: **datos de entrada**, **procesador** y **datos de salida**, como se muestra en la Fig.1.1. El procesador transforma los datos de entrada en datos de salida ejecutando instrucciones precisas y detalladas que se denominan **programas**.

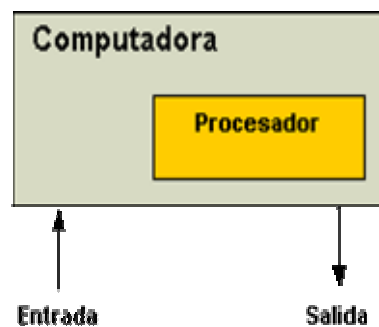


Figura 1.1 Proceso de información en una computadora

Una computadora esta compuesta por dos elementos fundamentales: **Hardware** y **Software**.

- El **Hardware** es la parte física de la computadora. Es aquello que podemos ver y tocar. Esta formado por el monitor, el teclado, el ratón, la unidad del sistema, la impresora, etc.
- El **Software** es la parte lógica de la computadora y esta formado por el conjunto de programas que controlan el funcionamiento de la computadora.

1.2 El hardware

Daremos aquí una breve descripción de las partes fundamentales del hardware sin entrar en detalles internos. Sin hacer distinción de tamaños, cualquier computadora, puede descomponerse físicamente en cuatro partes fundamentales:

- Procesador (llamado también Unidad Central de Proceso o CPU, del inglés, Central Processing Unit).

- Memoria principal.
- Dispositivos de entrada y salida E/S.
- Memoria auxiliar.

1.2.1 Procesador o Unidad Central de Proceso

El procesador es el cerebro de la computadora, el responsable de mantener en funcionamiento coordinado todas las partes de la computadora ejecutando instrucciones precisas y detalladas que son los programas. El procesador esta compuesto de dos partes fundamentales que son: La Unidad Aritmética-Lógica y la Unidad de Control.

La Unidad Aritmética-Lógica, es la responsable de efectuar operaciones aritméticas (suma, resta, multiplicación y división) y operaciones lógicas (comparaciones booleanas).

La Unidad de Control, es la responsable de la sincronización de todas las actividades de la computadora. Decide cuando se obtendrán datos de los dispositivos de entrada para ser llevados a la memoria, cuando se efectuarán cálculos con los datos almacenados en la memoria y cuando se llevarán datos de la memoria a los dispositivos de salida.

1.2.2 Memoria principal

La memoria principal es el espacio donde se almacenan los datos e instrucciones que se requieren para la realización de un proceso. En caso que los datos e instrucciones no quepan íntegramente en la memoria principal, entonces serán cargados hacia ella por etapas, pero en todo caso, cualquier operación se realiza en base al contenido actual de la memoria principal. A la memoria principal se la conoce también como memoria RAM (Random Access Memory, Memoria de Acceso Aleatorio).

Las características más relevantes de la memoria principal son las siguientes:

- La memoria principal es un espacio de almacenamiento temporal por lo que los datos no guardados en un medio de almacenamiento permanente se pierden al apagar la computadora o al salir un programa de la memoria.
- Está íntimamente ligada al procesador por lo que el acceso a la memoria principal es muy rápido lo que le permite al procesador acceder a la memoria principal millones de veces por segundo.
- Es de tamaño reducido en comparación a la memoria secundaria

La unidad más pequeña de memoria es el bit. Un bit puede almacenar un sólo dígito binario, 0 ó 1. Le sigue a esto el byte, que esta compuesto de 8 bits. Un byte tiene la capacidad de almacenar un carácter de información. En cambio para almacenar información numérica se requiere de mayor memoria que puede ser 2, 4 e incluso 8 bytes consecutivos, dependiendo del tipo de dato numérico.

Con cada byte de memoria se asocian dos cosas: **dirección** y **contenido** (la combinación de ceros y unos que puede almacenar). El procesador accede a una posición de memoria en base a su dirección.

Siempre que una nueva información se almacene en una posición de memoria, se destruye la información actual almacenada en esa posición y no se puede recuperar.

La memoria principal puede subdividirse en: memoria ROM, EPROM, RAM, DRAM, CMOS y CACHE. La memoria a que se hizo referencia en la descripción anterior se denomina memoria RAM (Random Access Memory, Memoria de Acceso Aleatorio). Para una descripción sobre los otros tipos de memoria puede recurrir a cualquier literatura técnica al respecto.

1.2.3 Dispositivos de entrada/salida (E/S)

También conocidos como periféricos de E/S. Estos dispositivos permiten comunicar la computadora con el usuario permitiendo el ingreso de datos a la computadora (dispositivos de entrada) y la salida de información de la computadora (dispositivos de salida). Es decir, los dispositivos de E/S son una interfaz entre el usuario y la computadora.

Entre los **dispositivos de entrada** más comunes se cuentan el teclado y el ratón. Existen otros dispositivos de entrada como el scanner, el lápiz óptico, etc.

Entre los **dispositivos de salida** más comunes tenemos: la pantalla y la impresora. Existen otros dispositivos de salida como los trazadores de gráficos, graficadores, etc.

1.2.4 Memoria auxiliar

Son dispositivos en los que se puede almacenar datos y programas de forma permanente. Entre los dispositivos más comunes de este tipo tenemos: los discos duros y los discos flexibles. Existen otros dispositivos de almacenamiento permanente como las unidades de cinta magnética, los discos compactos, los discos ópticos, etc.

Las características más relevantes de la memoria secundaria son las siguientes:

- Es un espacio de almacenamiento permanente.
- Un disco duro tiene un espacio de almacenamiento muchísimo más grande que el proporcionado por la memoria principal. Así un disco duro de 80 gigabytes es mucho más grande la capacidad de almacenamiento de la memoria principal (asumiendo una memoria principal típica de 32 megabytes de RAM).
- El procesador no actúa directamente con la memoria secundaria por lo que el acceso a la memoria secundaria es lento.
-

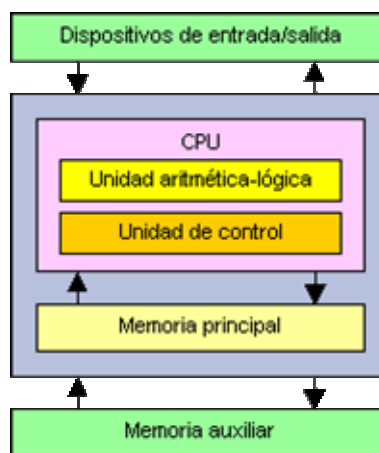


Figura 1.2. Organización física de una computadora

1.3 El software

El Software es el conjunto de datos y programas que usa la computadora y se guardan en algún dispositivo del hardware como, por ejemplo, un disco duro. El software es intangible (usted no lo puede tocar). Suponga que tiene un disco flexible conteniendo un programa, cuando usted borra el programa los átomos y moléculas del disco son los mismos que antes, pero ahora el programa ya no está. Se ha removido algo intangible sin alterar el medio tangible en el que estaba almacenado.

Un programa es un conjunto detallado de instrucciones que instruyen al procesador para realizar determinados procesos. Los datos pueden ser cualquier información que necesite el programa: caracteres, números, imágenes, etc. Para efectos de la memoria de la computadora (principal y secundaria) no hay ninguna distinción entre programas y datos.

1.3.1 Software específico o de aplicación

Son programas que tienen una aplicación específica tales como la preparación de nóminas, procesamiento de texto, procesamiento de imágenes, procesamiento de audio, etc.

Es decir, son programas que responden a una necesidad concreta y que ayudan a las persona a realizar sus trabajos. El mercado del software esta lleno de software de aplicación. Algunos programas de aplicación conocidos son: Microsoft Word, Microsoft Excel, Microsoft Power Point, Adobe Photoshop, Corel Draw, WinZip, Internet Explorer, etc.

1.3.2 Software de sistema

Son programas indispensables para el funcionamiento de la computadora. Estos programas son, básicamente, el sistema operativo, los compiladores e intérpretes y los programas de utilidad.

El software de sistema más importante es el sistema operativo. El sistema operativo es una colección compleja de muchos programas y es el encargado de coordinar el funcionamiento de los componentes hardware y software de un sistema de cómputo. El sistema operativo es responsable de iniciar la ejecución de otros programas proporcionando los recursos necesarios. Cuando un programa esta en ejecución, el sistema operativo maneja los detalles del hardware para dicho programa. Por ejemplo, cuando desde un programa procesador de textos, como Microsoft Word, usted decide guardar el documento que ha elaborado, es el sistema operativo el encargado de almacenar el documento como un archivo en el disco duro o disco flexible, según haya elegido. Así pues, entre muchas otras cosas, el sistema operativo se encarga del almacenamiento y recuperación de archivos. Los sistemas operativos pueden ser: **monousuarios** (un sólo usuario) y **multiusuarios** (diferentes usuarios), atendiendo al número de usuarios y **monocarga** (una sola tarea) o **multitarea** (múltiples tareas), atendiendo al número de tareas (procesos) que puede realizar simultáneamente

Los sistemas operativos modernos normalmente vienen con una interfaz gráfica de usuario que permite a los usuarios interactuar fácilmente con el sistema operativo mediante menús, íconos, el ratón y el teclado. Como ejemplos de sistemas operativos tenemos los siguientes: Unix, Windows 98, Windows NT, Windows XP, Linux, Solaris y System7.

Los compiladores e intérpretes son programas traductores que traducen un programa escrito en un determinado lenguaje de programación al lenguaje máquina que es el lenguaje del procesador. Cada lenguaje de programación tiene su propio compilador o intérprete.

1.4 Lenguajes de Programación

Lenguaje: Es una serie de símbolos que sirven para transmitir uno o mas mensajes (ideas) entre dos entidades diferentes. A la transmisión de mensajes se le conoce comúnmente como **comunicación**.

La **comunicación** es un proceso complejo que requiere una serie de reglas simples, pero indispensables para poderse llevar a cabo. Las dos principales son las siguientes:

- Los mensajes deben correr en un sentido a la vez.
- Debe forzosamente existir 4 elementos: Emisor, Receptor, Medio de Comunicación y Mensaje.

Lenguajes de Programación

Es un conjunto de símbolos, caracteres y reglas (programas) que le permiten a las personas comunicarse con la computadora.

Los lenguajes de programación tienen un conjunto de instrucciones que nos permiten realizar operaciones de entrada / salida, cálculo, manipulación de textos, lógica / comparación y almacenamiento / recuperación.

Los lenguajes de programación se clasifican en:

- **Lenguaje Máquina:** Son aquellos cuyas instrucciones son directamente entendibles por la computadora y no necesitan traducción posterior para que la CPU pueda comprender y ejecutar el programa. Las instrucciones en lenguaje máquina se expresan en términos de la unidad de memoria más pequeña el bit (dígito binario 0 o 1).
- **Lenguaje de Bajo Nivel (Ensamblador):** En este lenguaje las instrucciones se escriben en códigos alfabéticos conocidos como mnemotécnicos para las operaciones y direcciones simbólicas.
- **Lenguaje de Alto Nivel:** Los lenguajes de programación de alto nivel (BASIC, pascal, cobol, fortran, C, C++, etc.) son aquellos en los que las instrucciones o sentencias a la computadora son escritas con palabras similares a los lenguajes humanos (en general en inglés), lo que facilita la escritura y comprensión del programa.

Capítulo 2: Algoritmos, programas y conceptos fundamentales

2.1 Algoritmo vs. Programas

Un Algoritmo es un conjunto ordenado y finito de pasos o instrucciones que conducen a la solución de un problema. La naturaleza de los problemas varía con el ámbito o con el contexto donde están planteados; así, existen problemas matemáticos, químicos, filosóficos, etc. Según esto la naturaleza de los algoritmos también es variada y no todos ellos pueden ser ejecutados por la computadora. En este curso consideramos aquellos algoritmos que expresan soluciones usando reglas cuantitativas cuyas instrucciones pueden ser introducidas en la computadora, a este tipo de algoritmos se denominan **Algoritmos Computacionales**.

En la resolución de un problema con la computadora la parte pensante está en el algoritmo. Así pues la eficacia de un programador no está en conocer la herramienta de programación, cosa necesaria, sino en saber resolver problemas con la computadora para lo cual se requiere conocer un concepto conocido como **metodología de la programación** cuyo eje central es el algoritmo.

Una vez que la solución de un problema ha sido expresada mediante un algoritmo el paso siguiente es convertirlo a un programa para lo cual se elige un lenguaje de programación. De modo que un programa resulta ser la implementación de un algoritmo en un determinado lenguaje de programación. Esto significa, por otro lado, que un algoritmo es independiente del lenguaje de programación.



Figura 2.1 Problema, algoritmo y programa

2.2 El Seudo código

El seudo código es una herramienta algorítmica que permite escribir pseudo programas (una imitación de un programa real) utilizando un lenguaje de pseudo programación que es una imitación de los lenguajes de programación de alto nivel. Así, un pseudocódigo es una combinación de símbolos (+, -, *, /, %, >, >=, <, <=, !=, ==, y, o, no), términos (Leer, Imprimir, Abrir, Cerrar, Hacer...Mientras, Mientras...Hacer, Para...Mientras, etc) y otras características comúnmente utilizadas en uno o más lenguajes de alto nivel.

No existen reglas que determinen que es o no es un pseudocódigo, sino que varía de un programador a otro. El objetivo del pseudocódigo es permitir al programador centrarse en los aspectos lógicos de la solución evitando las reglas de sintaxis de un lenguaje de programación. Posteriormente el pseudocódi-

go debe ser traducido a programa usando un lenguaje de programación de alto nivel como Java, C++, C, etc.

Ejemplo 2.1:- Diseñe un algoritmo para preparar una limonada.

INICIO

- Llenar una jarra con un litro de agua
- Echar el jugo de tres limones
- Echar cuatro cucharadas de azúcar
- Remover el agua hasta disolver completamente el azúcar

FIN

Ejemplo 2.2:- Diseñe un algoritmo que permita hallar la suma y el promedio de tres números.

INICIO

- LEER** numero1, numero2, numero3
- suma = numero1 + numero2 + numero3
- promedio = suma / 3
- IMPRIMIR** suma, promedio

FIN

Notas:

El término **LEER** significa obtener un dato de algún dispositivo de entrada, como el teclado, y almacenarlo en una variable.

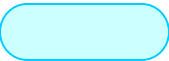


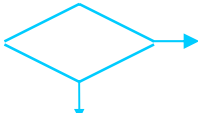
- Una variable es una localización en la memoria que tiene un nombre y cuyo contenido puede cambiar a lo largo de la ejecución de un programa. Así **numero1**, **numero2** y **numero3** son variables.
- El término **IMPRIMIR** significa mostrar el valor de una variable en algún dispositivo de salida, como la pantalla.

Para una explicación más detallada de estos conceptos vea [Variables](#) e [Instrucciones Básicas de Programación](#) más adelante en este mismo capítulo.

2.2.2 Diagramas de flujo

Diagrama de Flujo: Son símbolos gráficos que representan a un algoritmo. Estos símbolos están estandarizados, teniendo cada uno un significado universal.

Algunos de ellos son:

SÍMBOLO	DESCRIPCIÓN
	Indica el inicio y el final de nuestro programa diagrama de flujo
	Indica la entrada y salida de datos
	Símbolo de proceso y nos indica la asignación de un valor en la memoria y/o la ejecución de una operación aritmética
	Símbolo de decisión indica la realización de una comparación de valores



Se utiliza para representar los subprogramas



Conector dentro de página. Representa la continuidad del programa dentro de la misma página



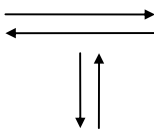
Conector fuera de página. Representa la continuidad del diagrama en otra página



Indica la salida de información por impresora



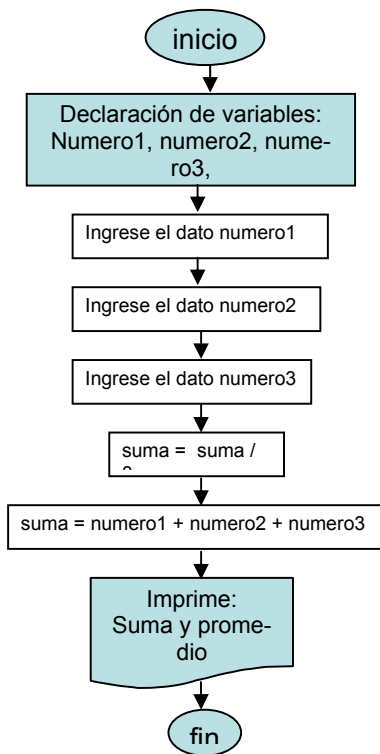
Indica la salida de información en la pantalla o monitor



Líneas de flujo o dirección. Indican la secuencia en que se realizar las operaciones

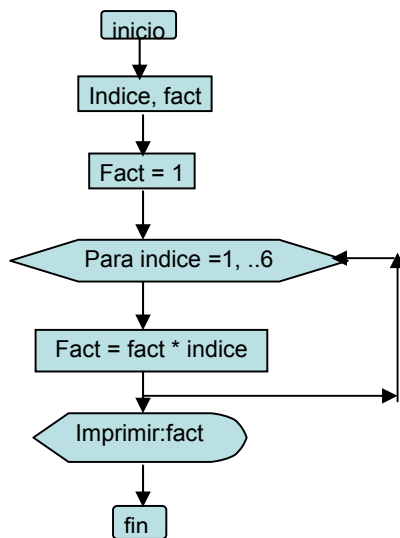
Ejemplo 1:

Ejemplificando el algoritmo anterior. El diagrama de flujo seria:



Ejemplo 2:

Se desea calcular el factorial del número 6.



2.3 Conceptos Fundamentales

El recurso fundamental utilizado por un programa es la memoria. Es en la memoria donde un programa almacena sus datos e instrucciones. Los datos se almacenan en localizaciones en la memoria denominadas variables. Cada variable puede almacenar un solo tipo de dato (entero, real, carácter, booleano, etc.) y requiere de un identificador para poder tener acceso a su contenido. En esta parte del curso veremos este y otros aspectos fundamentales.

2.3.1 Identificadores

Un *identificador* es un nombre que puede darse a una variable, a una constante y en general a cualquier elemento de un programa que necesite nombrarse.

Los lenguajes de programación imponen ciertas reglas para la creación de identificadores. Así, por ejemplo, en el lenguaje C, existen algunas reglas:

- El primer carácter debe ser una letra o un guión bajo, y puede contener casi cualquier combinación de letras, números, y guiones bajos pero no puede tener caracteres especiales ni blancos el cual se le considera como un carácter.
- No puede ser ninguna palabra reservada del lenguaje C.

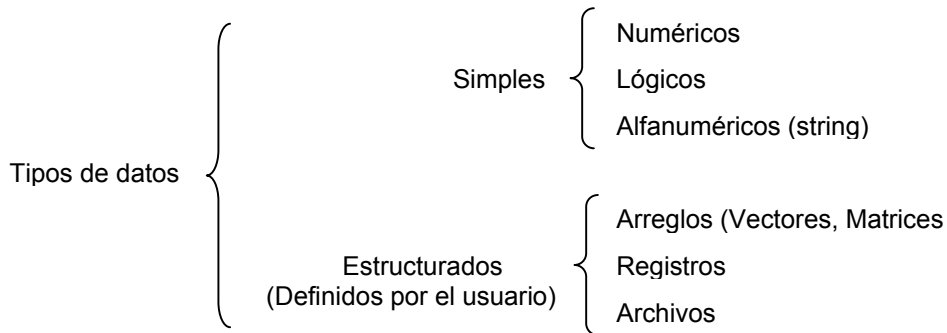
Así, por ejemplo, son válidos los siguientes identificadores:

SueldoBruto	Sueldo bruto de un empleado
Edad	Edad de una persona
numero_hijos	Número de hijos de un empleado
DIAS_SEMANA_7	Nombre de un días de la semana

2.3.2 Tipos de Datos

Los datos son los objetos de información sobre los que actúa un programa. Un dato puede ser un simple carácter como 'a', un valor entero tal como 35, un número real tal como 2.345 o una cadena tal como "algoritmia". En general puede ser cualquier cantidad que represente una medida, una magnitud, etc..

A nivel máquina los datos se representan internamente como una secuencia de bits (dígitos 0 y 1). Los lenguajes de alto nivel evitan estos manejos internos mediante el concepto de tipo de dato. En la mayoría de los lenguajes de programación los tipos de datos se pueden clasificar como:



Tipos de Datos Simples

- **Datos Numéricos:** Permiten representar valores escalares de forma numérica, esto incluye a los números enteros y los reales. Este tipo de datos permiten realizar operaciones aritméticas comunes.
- **Datos Lógicos:** Son aquellos que solo pueden tener dos valores (cierto o falso) ya que representan el resultado de una comparación entre otros datos (numéricos o alfanuméricos).
- **Datos Alfanuméricos (String):** Es una secuencia de caracteres alfanuméricos que permiten representar valores identificables de forma descriptiva, esto incluye nombres de personas, direcciones, etc. Es posible representar números como alfanuméricos, pero estos pierden su propiedad matemática, es decir no es posible hacer operaciones con ellos. Este tipo de datos se representan encerrados entre comillas.

2.3.3 Variables

Una variable es una localidad en la memoria principal que se relaciona con ella mediante una dirección, en la que se almacena un dato que puede cambiar a lo largo de la ejecución del programa.

Una variable tiene asociada dos cosas fundamentales: un **identificador** y un **tipo de dato**. El nombre identifica de manera única la localización de memoria donde se almacena el dato. El tipo de dato especifica la naturaleza del dato que puede almacenar la variable. Según el tipo de dato de la variable podemos tener variables enteras, variables reales, variables de carácter, variables booleanas, etc.

En el gráfico siguiente se muestran dos variables con sus contenidos en memoria. La variable llamada **Edad** (de tipo entero) cuyo contenido es 27 y la variable llamada **Descuento** (de tipo real) cuyo contenido es 23.57.

Edad	27
Descuento	23.57

2.3.4 Constantes

Se denominan constantes a todos aquellos valores que no cambian durante la ejecución de un programa. Según el tipo de dato podemos tener constantes enteras, constantes reales, constantes de carácter y constantes de cadena.

Constantes enteras

12, 20300, 15, etc.

Constantes reales

3.1416, 2345.456, etc.

Constantes de carácter

'a', 'B', ';', '<', '+', etc.

Constantes de cadena

"Hola", "Algoritmos Computacionales", etc.

Constantes lógicas

verdadero, falso

Normalmente los lenguajes de programación permiten dar nombres a determinadas constantes. Por ejemplo, el número de días de la semana (7) es un valor constante y puede recibir el nombre NUMDIAS.

2.3.5 Instrucciones Básicas de Programación

Las principales instrucciones básicas de programación son las siguientes: asignación, entrada y salida.

- **Instrucción de asignación:** consiste en dar a una variable el valor de una expresión, el valor de otra variable o el valor de una constante. *La asignación tiene efecto destructivo* en el sentido que destruye el valor previo de la variable que recibe la asignación.
- **Instrucción de entrada:** consiste en obtener un dato de un dispositivo de entrada, como el teclado, leerlo de un archivo, etc. y almacenarlo en una variable.
- **Instrucción de salida:** consiste en mostrar el valor de una variable en un dispositivo de salida como la pantalla, una impresora, o almacenarlo en un archivo.

Tabla 2.3 Instrucciones básicas de programación

Instrucción	Tipo de Instrucción
LEER variable	Entrada
variable = expresión	Asignación
IMPRIMIR variable	Salida

Ejemplo 2.3:- Instrucciones de entrada y de salida

- La siguiente **instrucción de entrada**, obtiene la edad de una persona desde un dispositivo de entrada, como el teclado, y lo almacena en la variable llamada **edad**:

LEER edad

- La siguiente **instrucción de entrada**, obtiene tres números desde un dispositivo de entrada, como el teclado, y los almacena en las variables **num1**, **num2** y **num3**:

LEER num1, num2, num3

- La siguiente **instrucción de salida**, muestra el contenido de la variable **sueldoNeto** en un dispositivo de salida como la pantalla:

IMPRIMIR sueldoNeto

- La siguiente **instrucción de salida**, muestra el contenido de las variables **montoPag** y **montoDes** en un dispositivo de salida como la pantalla:

IMPRIMIR montoPag, montoDes

Ejemplo 2.4:- Instrucción de asignación.

- La siguiente **instrucción de asignación**, asigna el valor constante 25 a la variable **kilometros**:

kilometros = 25

Antes de la asignación

Si asumimos que es la primera vez que la variable **kilometros** recibe una asignación, entonces su valor actual es desconocido:

kilometros

- Después de la asignación:

Kilómetros

- La siguiente **instrucción de asignación**, asigna a la variable **vara** el valor de la variable **varb**:

vara = varb

Luego de esto, ambas variables tienen el mismo valor.

Antes de la asignación

Asumamos que **vara** y **varb** tienen actualmente los valores mostrados

varb

vara

Después de la asignación

vara toma el valor de **varb**, luego de lo cual ambas variables tienen el mismo valor. Note que el valor anterior de **vara** se destruye de forma irrecuperable.

varb

vara

- La siguiente **instrucción de asignación**, calcula el valor de la expresión del lado derecho y lo asigna a la variable **montoPag**:

montoPag = montoBru - montoDes

Antes de la asignación

Asumamos que actualmente **montoBru** y **montoDes** tienen los valores mostrados y que **montoPag** tiene un valor desconocido al no haber recibido ninguna asignación anterior:

montoBru

montoDes

montoPag

Después de la asignación

montoBru

montoDes

montoDes

2.4 Expresiones Aritméticas

Una expresión aritmética es un conjunto de variables y/o constantes unidas o relacionadas por paréntesis y operadores aritméticos.

Los operadores aritméticos son los siguientes:

Tabla 2.4 Operadores aritméticos

Operador	Significado
+	Suma
-	Resta
*	Multiplicación
/	División
%	Residuo de división entera

En todos los lenguajes existen los operadores +, -, * y / entre otros, mientras que el operador % es propio de C, C++ y Java.

Notas

1. Si en una operación ambos operandos son enteros, entonces el resultado de la operación es un entero.
2. Si en una operación uno o ambos operandos son reales, entonces el resultado de la operación es un real.
3. El operador / produce un cociente entero si los dos operandos son enteros. Esto significa que se pierde la parte decimal si la división no es exacta. Esta es una consecuencia de la nota 1.
4. El operador / produce un cociente real si uno o los dos operandos son reales. Esta es una consecuencia de la nota 2.
5. Todas las expresiones entre paréntesis se evalúan primero. Las expresiones con paréntesis anidados se evalúan de dentro a fuera, el paréntesis mas interno se evalúa primero.
6. Dentro de una misma expresión los operadores se evalúan en el siguiente orden.
 1. ^ Exponenciación
 2. *, /, mod Multiplicación, división, modulo
 3. +, - Suma y resta.
7. Los operadores en una misma expresión con igual nivel de prioridad se evalúan de izquierda a derecha.

Por ejemplo:

7 / 2 es igual a 3 y no 3.5 como lo es matemáticamente. Esto debido a que 7 y 2 son enteros y al dividir dos enteros se pierde la parte fraccionaria, no se redondea.

En cambio:

7.0 / 2 es igual a 3.5 ya que si uno o los dos operandos son reales, entonces el resultado es real. En este caso 7.0 es real.

Ejemplo 2.5:- Expresiones aritméticas.

- sueldoBru = sueldoBas + 0.15*montoVen
- numero = centenas*100 + decenas*10 + unidades
- $e = a*b*b / 3 + (a*a + b) / (b + c)$

Ejemplo 2.6:- Diseñe un algoritmo que determine el cociente y el residuo de una división entera.

VARIABLES

ENTERO m, m

INICIO

LEER m, n

cociente = m/n

residuo = m%n

IMPRIMIR residuo, cociente

FIN

Nota: Observe que m/n es entero debido a que ambas variables son enteras.

2.5 Expresiones Lógicas

Una expresión lógica o booleana es un conjunto de variables y/o constantes unidas mediante *operadores lógicos* y *operadores relacionales*.

Una expresión lógica solo puede tomar uno de dos valores: verdadero o falso. Las expresiones lógicas son ampliamente utilizadas en las estructuras selectivas y las estructuras repetitivas.

En la tabla 2.4 se muestran los operadores relacionales y en la tabla 2.5 se muestran los operadores lógicos:

Tabla 2.4 Operadores relacionales

Operador	Significado
>	mayor
>=	mayor o igual que
<	menor
<=	menor o igual que
==	igual a
!=	diferente de

Tabla 2.5 Operadores lógicos

Operador	Significado
no (not)	negación
y (and)	conjunción
o (or)	disyunción

Ejemplo 2.7:- Expresar las siguientes condiciones como expresiones lógicas:

- B es mayor que 2.*
- M es menor ó igual que 5 pero mayor que 25.*
- P es igual a 6 ó mayor que Q.*
- N es par menor que 50.*
- M es mayor que A, B y C.*
- Z esta en el intervalo de 4 a 100.*
- T es igual a 2, 3 ó 4.*

- $B > 2$
- $(M \leq 5) \text{ y } (M > 25)$
- $(P == 6) \text{ o } (P > Q)$
- $((N \% 2) == 0) \text{ y } (N < 50)$
- $(M > A) \text{ y } (M > B) \text{ y } (M > C)$
- $(Z \geq 4) \text{ y } (Z \leq 100)$
- $(T == 2) \text{ o } (T == 3) \text{ o } (T == 4)$

Estos operadores se utilizan para establecer relaciones entre valores lógicos. Estos valores pueden ser resultado de una expresión relacional.

Capítulo 3: Solución de problemas con la computadora

3.1 Introducción

Etimológicamente, la palabra problema deriva del griego *proballein* y significa *algo lanzado hacia delante*. Un problema es un asunto o un conjunto de cuestiones que se plantean para ser resueltas. La naturaleza de los problemas varía con el ámbito o con el contexto donde están planteados; así, existen problemas matemáticos, químicos, filosóficos, etc. Consideramos aquí sólo aquellos problemas cuya solución se puede calcular utilizando una serie de reglas introducidas en la computadora.

No existe un método universal que permita resolver cualquier problema. En general, la resolución de problemas es un proceso creativo donde el conocimiento, la habilidad y la experiencia tienen un papel importante. El proceder de manera sistemática (sobre todo si se trata de problemas complejos) puede ayudar en la solución.

En general, la solución de problemas con la computadora se puede dividir en tres etapas:

- Análisis del problema
- Diseño del algoritmo
- Implementación del algoritmo en la computadora

3.2 Definición del Problema

Esta fase está dada por el enunciado del problema, el cual requiere una definición clara y precisa. Es importante que se conozca lo que se desea que realice la computadora; mientras esto no se conozca del todo no tiene mucho caso continuar con la siguiente etapa.

3.3 Análisis del Problema

El análisis del problema es la segunda fase en la resolución de un problema con la computadora. El objetivo del análisis es comprender y definir claramente la naturaleza del problema. En esta etapa es fundamental establecer con claridad que hará el programa que se pretende construir. No se puede abordar una solución mientras no se sepa a donde se quiere llegar.

Para poder definir el problema con precisión se requiere especificar con detalle cuales serán los datos de entrada y cuales los datos de salida. La forma como se procesará la información de entrada para producir los datos de salida es tarea de la etapa del diseño del algoritmo; pero, en esta etapa puede establecerse un esbozo de la solución.

Así pues, el análisis del problema comprende los siguientes aspectos:

- Definición del problema.
- Especificaciones de entrada.
- Especificaciones de salida.

Una recomendación muy práctica es el que nos pongamos en el lugar de la computadora y analicemos que es lo que necesitamos que nos ordenen y en que secuencia para producir los resultados esperados.

3.4 Diseño del Algoritmo

En esta etapa se construye un algoritmo que resuelva el problema analizado, utilizando una herramienta algorítmica como el pseudocódigo. Aquí se decide como hará el algoritmo para producir los datos de salida sobre la base de los datos de entrada. Esto puede requerir de acciones secuenciales, tomas de decisiones y repeticiones de procesos.

Para un diseño eficiente del algoritmo es importante abordar la solución mediante una **metodología de diseño**. Una estrategia popular consiste en aplicar el dicho romano "**divide y vencerás**". Este método consiste en dividir el problema en **subproblemas** más simples, para facilitar la comprensión y solución del problema. Si algún subproblema todavía es complejo puede a su vez subdividirse en otros subproblemas. Este proceso de subdivisión puede continuar hasta obtener subproblemas simples de resolver. A esta metodología de diseño se conoce como el **método del diseño modular descendente**, debido a que cada subproblema puede resolverse mediante un módulo de programa.

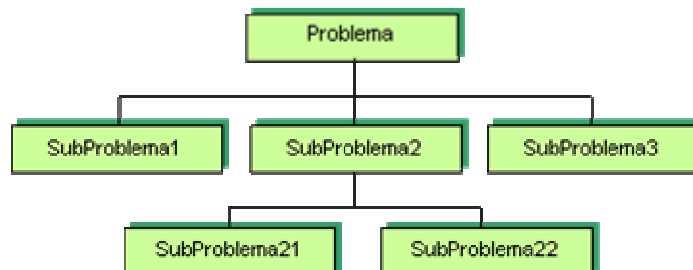


Figura 2.1 Diseño descendente

Las características de un buen algoritmo son:

- Debe tener un punto particular de inicio.
- Debe ser definido, no debe permitir dobles interpretaciones.
- Debe ser general, es decir, soportar la mayoría de las variantes que se puedan presentar en la definición del problema.
- Debe ser finito en tamaño y tiempo de ejecución.

3.5 Prueba y Depuración

Los errores humanos dentro de la programación de computadoras son muchos y aumentan considerablemente con la complejidad del problema. El proceso de identificar y eliminar errores, para dar paso a una solución sin errores se le llama **depuración**.

La **depuración o prueba** resulta una tarea tan creativa como el mismo desarrollo de la solución, por ello se debe considerar con el mismo interés y entusiasmo.

Resulta conveniente observar los siguientes principios al realizar una depuración, ya que de este trabajo depende el éxito de nuestra solución.

3.6 Documentación

Es la guía o comunicación escrita en sus variadas formas, ya sea en enunciados, procedimientos, dibujos o diagramas.

A menudo un programa escrito por una persona, es usado por otra. Por ello la documentación sirve para ayudar a comprender o usar un programa o para facilitar futuras modificaciones (mantenimiento).

La **documentación** se divide en tres partes:

- Documentación Interna: Son los comentarios o mensaje que se añaden al código fuente para hacer mas claro el entendimiento de un proceso.
- Documentación Externa: Se define en un documento escrito los siguientes puntos:

Descripción del Problema
Nombre del Autor
Algoritmo (diagrama de flujo o pseudo código)

Diccionario de Datos
Código Fuente (programa)

- Manual del Usuario: Describe paso a paso la manera como funciona el programa, con el fin de que el usuario obtenga el resultado deseado.

3.7 Mantenimiento

Se lleva a cabo después de terminado el programa, cuando se detecta que es necesario hacer algún cambio, ajuste o complementación al programa para que siga trabajando de manera correcta. Para poder realizar este trabajo se requiere que el programa este correctamente documentado.

3.8 Implementación del Algoritmo en la Computadora

Esta etapa es relativamente mecánica y consiste en codificar el algoritmo siguiendo las reglas sintácticas y semánticas de un determinado lenguaje de programación. Al resultado de la codificación se denomina código fuente o programa fuente. Luego de ello el programa fuente debe ser ejecutado y probado para verificar si los resultados obtenidos son los esperados.

La verificación del programa se efectúa con una amplia variedad de datos de entrada, llamados datos de test o datos de prueba, que determinarán si el programa tiene errores ("bugs"). Los datos de prueba comprenden: valores normales de entrada, valores extremos de entrada que comprueben los límites del programa y valores de entrada que comprueben aspectos especiales del programa. Si los resultados obtenidos no son los esperados se procede a depurar el programa. La depuración consiste en encontrar los errores del programa para efectuar las correcciones pertinentes.

Así pues, la implementación de un algoritmo en la computadora comprende los siguientes aspectos:

- Codificación del algoritmo.
- Verificación y depuración de la solución.

Ejemplo 3.1

Definición del problema

Se desea calcular el factorial de un número.

Afortunadamente para todos nos queda claro que es (por su definición) el factorial de un número.

Análisis del problema

De la lectura del problema encontramos los siguientes datos de entrada y datos de salida:

Datos de entrada:

- El número al que se le quiere calcular su factorial

Datos de salida:

- El número al que se le quiere calcular su factorial
- El factorial del número

Diseño del algoritmo

Primer diseño descendente

En un primer momento el problema puede descomponerse de manera bastante general. En este caso se ha descompuesto en tres pasos.

Inicio

1. Leer el número al que se le desea calcular el factorial **n**
2. Cálculo del factorial
3. Imprimir **n** y factorial

Fin

Refinamiento

Para refinar nuestro algoritmo es necesario recordar y plantear el modelo matemático de la función del factorial.

Factorial = 1 si **n** es igual a cero.

Factorial = $1 * 2 * 3 * \dots * n$ Si **n** > 0

Luego de un refinamiento en el subproblema **Cálculo del factorial** llegamos al siguiente diseño:

Inicio

1. Leer **n**

2. factorial = 1 // inicializamos la variable factorial

2. mientras **i** > **n** // mientras se cumpla que **i** < **n**

2.1 factorial = factorial * **i** // multiplicamos el factorial por **i**

2.2 **i** = **i** + 1 // incremento de la variable índice **i**

3. Imprimir **n** y factorial

Fin

Prueba y depuración

Una prueba para depurar el algoritmo es realizar la prueba de escritorio. Esta consiste ir asignando valores a las variables, de acuerdo al algoritmo durante toda su ejecución.

n **i** **factorial** **factorial * i**

6 ? 1

1 1 1

2 1 2

3 2 6

4 6 24

5 24 120

6 120 720

7 en este momento la condición no se cumple y se sale del ciclo por lo que imprime los que tienen las **Imprimir n** y **factorial** que son 6 y 720

Documentación

La documentación interna, son los comentarios que se agregaron al algoritmo y que se implementarían en el código cuando este se realizara. La documentación externa sería escribir un documento que indique al usuario como utilizar el programa, incluyendo los pasos que se siguieron para resolver el problema.

Mantenimiento

El mantenimiento consistiría en que, una vez que se tiene el programa y que por algún motivo se quiere modificar y/o que se quiera que el programa realice otras tareas, se procedería a realizar los cambios y/o adecuaciones necesarias. Con la documentación esta tarea será suficiente para que esta labor la pueda realizar cualquier persona que tenga conocimientos de programación.

Capítulo 4: Estructuras secuenciales y de E/S

4.1 Asignación

Asignación: La asignación consiste, en el paso de valores o resultados a una zona de la memoria. Dicha zona será reconocida con el nombre de la variable que recibe el valor. La asignación se puede clasificar de la siguiente forma:

- **Simples:** Consiste en pasar un valor constante a una variable ($a=15$)
- **Contador:** Consiste en usarla como un verificador del número de veces que se realiza un proceso ($a=a+1$)
- **Acumulador:** Consiste en usarla como un sumador en un proceso ($a=a+b$)
- **De trabajo:** Donde puede recibir el resultado de una operación matemática que involucre muchas variables ($a=c+b*2/4$).

4.2 Secuenciales

La estructura secuencial es aquella en la que una acción (instrucción) sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso. Una estructura secuencial se representa de la siguiente forma:

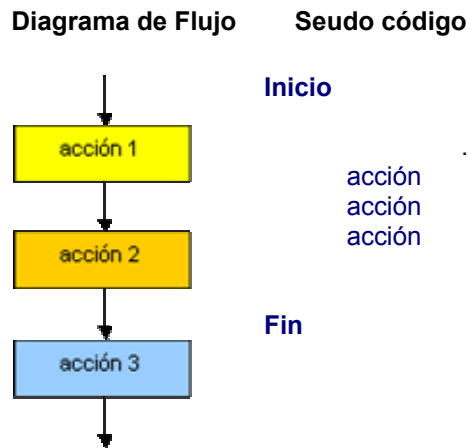
```

Inicio
  Accion1
  Accion2
  .
  .
  AccionN
Fin

```

La estructura secuencial tiene una entrada y una salida.

En la Figura 3.1 se muestra el diagrama de flujo y el pseudo código de una estructura secuencial.



4.3 Entrada (Lectura)

Lectura: La lectura consiste en recibir desde un dispositivo de entrada (por ejemplo el teclado) un valor. Esta operación se representa en un seudo código como sigue:

- Leer a, b
- Donde “a” y “b” son las variables que recibirán los valores

4.4 Salida (Escritura)

Escritura: Consiste en mandar por un dispositivo de salida (por ejemplo monitor o impresora) un resultado o mensaje. Este proceso se representa en un seudo código como sigue:

- Escribe “El resultado es:”, R
- Donde “El resultado es.” es un mensaje que se desea aparezca y R es una variable que contiene un valor.

4.5 Ejercicios

Ejercicio 1: Diseñe un seudo código que halle el área y el perímetro de un rectángulo. Considere las siguientes fórmulas: $\text{área} = \text{base} \times \text{altura}$, $\text{perímetro} = 2 \times (\text{base} + \text{altura})$.

Algoritmo

INICIO
REAL base, altura, area, perimetro
LEER base, altura
 area = base*altura
 perimetro = 2*(base+altura)
IMPRIMIR area, perimetro
FIN

Ejercicio 2: Diseñe un seudo código para convertir una longitud dada en centímetros a pies. Considere que: 1 pie = 30.48 centímetros.

Algoritmo**INICIO****REAL** cpies, ccent**LEER** ccent

cpies = ccent/30.48

IMPRIMIR cpies**FIN**

Ejercicio 3: Diseñe un seudo código para convertir una longitud dada en pies a centímetros. Considere que: 1 pie = 30.48 centímetros.

Algoritmo**INICIO****REAL** cpies, ccent**LEER** cpies

ccent = cpies*30.48

IMPRIMIR ccent**FIN**

Ejercicio 4: Una institución benéfica europea ha recibido tres donaciones en soles, dólares y marcos. La donación será repartida en tres rubros: 60% para la implementación de un centro de salud, 40% para un comedor de niños y el resto para gastos administrativos. Diseñe un algoritmo que determine el monto en euros que le corresponde a cada rubro. Considere que: 1 dólar = 3.52 soles, 1 dólar = 2.08 marcos, 1 dólar = 1.07 euros.

Algoritmo**INICIO****REAL** c soles, cdolares, c marcos, ceuros, rubro1, rubro2, rubro3**LEER** csoles, cdolares, cmarcos

// Determina el total en euros

ceuros = (csoles/3.52 + cdolares + cmarcos/2.08)*1.07

// Determina el monto de cada rubro

rubro1 = ceuros*0.60

rubro2 = ceuros*0.40

rubro3 = ceuros*0.20

IMPRIMIR rubro1, rubro2, rubro3**FIN**

Ejercicio 5: En una competencia atlética de velocidad el tiempo se mide en minutos, segundos y centésimas de segundo y, el espacio recorrido se mide en metros. Diseñe un algoritmo para determinar la velocidad promedio de un atleta en km/hr. Considere que: 1 hora = 60 minutos, 1 minuto = 60 segundos, 1 segundo = 100 centésimas de segundo, 1 kilómetro = 1000 metros.

Algoritmo**INICIO****ENTERO** tmin, tseg, tcsg, espmt**REAL** thor, velkmhr**LEER** tmin, tseg, tcsg, espmt

```
// Determina el tiempo total empleado en horas
thor = tmin/60 + tseg/3600 + tcsg/360000
// Determina el espacio recorrido en kilómetros
espkm = espmt/1000
// Determina la velocidad en km/hr
velkmhr = espkm/thor
```

```
IMPRIMIR velkmhr
FIN
```

Ejercicio 6: Diseñe un algoritmo que determine la cifra de las unidades de un número entero positivo.

Solución 1

Análisis

Puede comprobarse que la cifra de las unidades de un número es igual al resto de la división del número entre 10. Observe para ello las siguientes divisiones:

3245	10	
5	324	
756	10	
6	75	
8	10	
8	0	

Podemos concluir entonces que:

unidades = numero % 10

Siendo % el operador residuo. Este operador permite obtener el residuo de una división, así como / permite obtener el cociente.

Algoritmo

```
INICIO
  ENTERO numero, unidades

  LEER numero
  unidades = numero % 10
  IMPRIMIR unidades
FIN
```

Solución 2

Análisis

El residuo de una división entera puede obtenerse también sin recurrir al operador %, de la siguiente forma:

unidades = numero - (numero / 10) * 10

Observe para esto que en la división (**numero/10**) los operandos son enteros por lo que el cociente será un entero. Así por ejemplo, si **numero** es igual a **3245**, la división (**numero/10**) produce **324**, aunque matemáticamente sea **324.5**; es decir, se descarta la parte decimal.

Algoritmo

INICIO

ENTERO numero, unidades

LEER numero

unidades = numero - (numero/10)*10

IMPRIMIR unidades

FIN

Ejercicio 7: Diseñe un algoritmo que determine la suma de las cifras de un número entero positivo de 4 cifras.

Análisis

Para obtener las cifras de un número podemos proceder mediante divisiones sucesivas entre 10. Para el efecto, considere el caso de un número N igual a 3245:

3245	10
5	324
324	10
4	32
32	10
2	3

En la segunda y tercera división el dividendo es el cociente de la división anterior. Las cifras se obtienen como:

```

unidad = N%10
cociente = N/10
decena = cociente%10
cociente = cociente/10
centena = cociente%10
millar = cociente/10
    
```

Observe que a cambio de la variable cociente puede usarse la misma variable N ya que lo que interesa es que N guarde el nuevo dividendo para la próxima división. Así:

```

unidad = N%10
N = N/10
decena = N%10
N = N/10
centena = N%10
millar = N/10
    
```

Por otro lado, considerando que el número tiene 4 cifras, también podrían obtenerse las cifras por divisiones sucesivas entre 1000, 100 y 10. Así:

3245	1000
245	3
245	100
45	2
45	10
5	4

Que puede expresarse como:

```
millar = N/1000
resto = N%1000
centena = resto/100
resto = resto%100
decena = resto/10
unidad = resto%10
```

Algoritmo

INICIO

ENTERO N, suma, millar, centena, decena, unidad, resto

LEER N

```
millar = N/1000
resto = N%1000
centena = resto/100
resto = resto%100
decena = resto/10
unidad = resto%10
suma = unidad + decena + centena + millar
```

IMPRIMIR suma

FIN

Ejercicio 8: Diseñe un algoritmo que lea la hora actual del día HH:MM:SS y determine cuantas horas, minutos y segundos restan para culminar el día.

Algoritmo

INICIO

ENTERO hor1, min1, seg1, hor2, min2, seg2, segres, resto

LEER hor1, min1, seg1

// Determina la cantidad de segundos que restan para culminar el día
segres = 86400 - (hor1*3600 + min1*60 + seg1)

// Determina cuantas horas, minutos y segundos restan para culminar el día
hor2 = segres/3600
resto = segres%3600
min2 = resto/60
seg2 = resto%60

IMPRIMIR hor2, min2, seg2

FIN

Ejercicio 9: Diseñe un algoritmo para sumar dos tiempos dados en horas, minutos y segundos.

Algoritmo

INICIO

ENTERO hor1, min1, seg1, hor2, min2, seg2, hor3, min3, seg3, totseg, resto

LEER hor1, min1, seg1, hor2, min2, seg2

// Determina la cantidad total de segundos entre los dos tiempos
 totseg = (hor1+hor2)*3600 + (min1+min2)*60 + (seg1+seg2)

// Convierte totseg a horas, minutos y segundos

hor3 = totseg/3600
 resto = totseg%3600
 min3 = resto/60
 seg3 = resto%60

IMPRIMIR hor3, min3, seg3

FIN

Ejercicio 10: El sueldo neto de un vendedor se calcula como la suma de un sueldo básico de S/.250 más el 12% del monto total vendido. Diseñe un algoritmo que determine el sueldo neto de un vendedor sabiendo que hizo tres ventas en el mes.

Algoritmo

INICIO

REAL venta1, venta2, venta3, ventatot, comision, sueldoneto

LEER venta1, venta2, venta3

ventatot = venta1 + venta2 + venta3
 comision = 0.12*ventatot
 sueldoneto = 250 + comision

IMPRIMIR sueldoneto

FIN

Ejercicio 11: Diseñe un algoritmo que determine el porcentaje de varones y de mujeres que hay en un salón de clases.

Algoritmo

INICIO

REAL porcvar, porcuj
ENTERO varones, mujeres, total

LEER varones, mujeres

total = varones + mujeres
 porcvar = varones*100.0/total
 porcuj = mujeres*100.0/total

IMPRIMIR porcvar, porcuj

FIN

Ejercicio 12: En países de habla inglesa es común dar la estatura de una persona como la suma de una cantidad entera de pies más una cantidad entera de pulgadas. Así, la estatura de una persona podría ser 3' 2" (3 pies 2 pulgadas). Diseñe un algoritmo que determine la estatura de una persona en metros, conociendo su estatura en el formato inglés. Considere que: 1 pie = 12 plg, 1 plg = 2.54 cm, 1 m = 100 cm.

Algoritmo

INICIO

REAL estmt

ENTERO cpies, cplgs

LEER cpies, cplgs

estmt = ((cpies*12 + cplgs)*2.54)/100

IMPRIMIR estmt

FIN

Ejercicio 13: Diseñe un algoritmo que exprese la capacidad de un disco duro en megabytes, kilobytes y bytes, conociendo la capacidad del disco en gigabytes. Considere que: 1 kilobyte = 1024 bytes, 1 megabyte = 1024 kilobyte, 1 gigabyte = 1024 megabytes.

Algoritmo

INICIO

REAL cgigabyte, cmegabyte, ckilobyte, cbyte

LEER cgigabyte

cmegabyte = cgigabyte*1024

ckilobyte = cmegabyte*1024

cbyte = ckilobyte*1024

IMPRIMIR cmegabyte, ckilobyte, cbyte

FIN

Capítulo 5: Estructuras de Selección

5.1 Introducción

En la solución de la mayoría de los problemas algorítmicos se requieren efectuar tomas de decisiones que conducen a la ejecución de una o más acciones dependiendo de la Verdad o falsedad de una o más condiciones. Como consecuencia de esto se producen cambios en el flujo de control del programa. Dicho flujo de control implica rutas que deben ser seleccionadas. Para esto, se utilizan ciertas estructuras de programación conocidas como **estructuras selectivas** o **estructuras de decisión**.

Las estructuras selectivas o de selección se clasifican en:

- Estructura de selección simple (**SI**).
- Estructura de selección doble (**SI - SINO**).
- Estructura de selección múltiple (**EN CASO - SEA**)

5.2 Estructura de Selección Simple SI

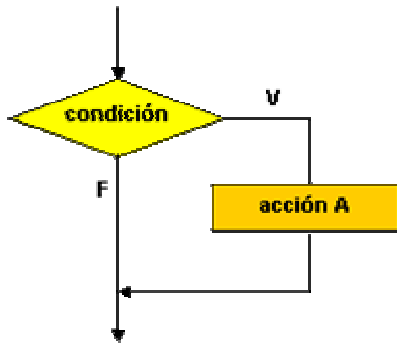
En la estructura de selección simple **SI**, evalúa una condición lógica y:

- Si la condición es verdadera se ejecuta la **acción A**. La **acción A** puede ser una *acción simple* (una sola acción) o una *acción compuesta* (un conjunto de acciones).

- Si la condición es falsa, no se hace nada.

Nota.- En el caso de acciones compuestas, estas serán encerradas entre llaves.

Diagrama de Flujo



Seudo código (acción simple)

```
SI( condición )
    acción A
```

Seudo código (acción compuesta)

```
SI( condición ){
    acción A1
    acción A2
    .
    .
    acción An
}
```

Figura 5.1 Estructura de Selección Simple

Por ejemplo, si se desea cambiar el signo de un número únicamente en caso que sea negativo, podemos escribir:

```
SI( numero < 0 )
    numero = -1 * numero
```

Si el número no es negativo, simplemente esta estructura se pasaría por alto y se continuaría en la siguiente instrucción después del **SI**.

5.3 Estructura de Selección Doble SI - SINO

La estructura de selección doble SI - SINO evalúa una condición lógica y:

- Si la condición es verdadera, ejecuta la acción A.
- Si la condición es falsa, ejecuta la acción B.

Tanto la acción A como la acción B pueden ser *acciones simples* (una sola acción) o *acciones compuestas* (un conjunto de acciones).

Nota.- En el caso de acciones compuestas, estas serán encerradas entre llaves.

Diagrama de Flujo

Seudo código (acciones simples)

```
SI(condición)
    acciónA
SINO
    acción B
```

Seudo código(acciones compuestas)

```
SI(condición){
```

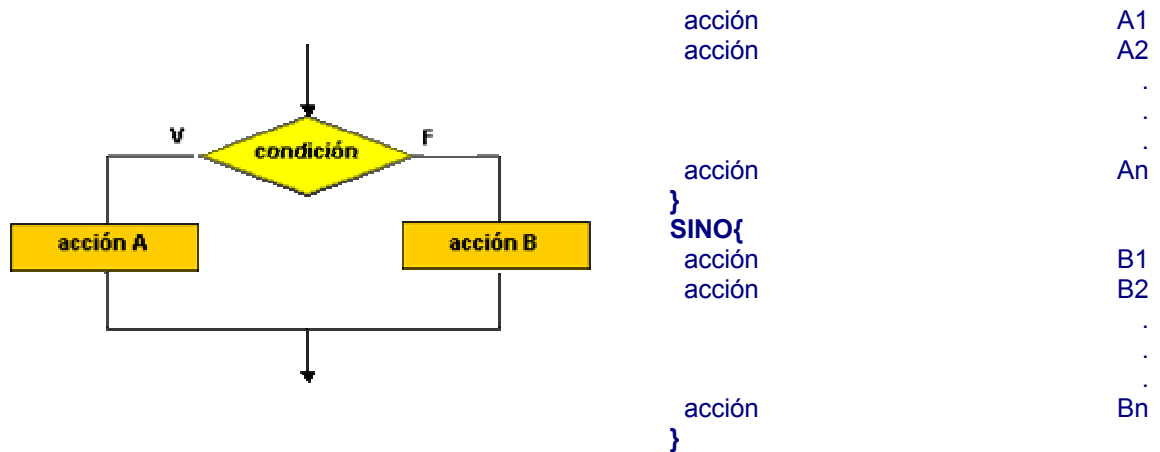


Figura 5.2 Estructura de Selección Doble

Por ejemplo, si se desea saber si una persona es mayor o menor de edad, podemos escribir:

```
SI( edad >= 18 )
    IMPRIMIR "Mayor de edad"
SINO
    IMPRIMIR "Menor de edad"
```

Esto imprime "Mayor de edad" si la persona tiene 18 años ó más e imprime "Menor de edad" si la persona tiene menos de 18 años. En cualquiera de los casos, después de efectuar la impresión, se ejecutará la primera instrucción que sigue a la estructura **SI...SINO**.

5.4 Operadores Lógicos y Relacionales

Para expresar condiciones como en el caso de las condiciones de las estructuras selectivas, se requieren de *operadores relacionales* y *operadores lógicos*, que se muestran en las tablas 1 y 2, respectivamente. Dichas condiciones solo puede tomar uno de los siguientes valores: verdadero o falso.

Tabla 1 Operadores relacionales

Operador	Significado
>	mayor
>=	mayor o igual que
<	menor
<=	menor o igual que
==	igual a
!=	diferente de

Tabla 2 Operadores lógicos

Operador	Significado
no	negación
y	conjunción
o	disyunción

Ejemplo 5.1: En una playa de estacionamiento cobran \$/ 2.5 por hora o fracción. Diseñe un algoritmo que determine cuanto debe pagar un cliente por el estacionamiento de su vehículo, conociendo el tiempo de estacionamiento en horas y minutos.

Algoritmo

```
INICIO
  ENTERO horas, minutos
  REAL pago

  LEER horas, minutos

  SI( minutos > 0 )
    horas = horas + 1
    pago = horas * 2.5

  IMPRIMIR pago
FIN
```

Ejemplo 5.2: Diseñe un algoritmo que determine si un número es o no, par positivo

Algoritmo

```
INICIO
  REAL n

  LEER n

  SI( (n%2==0) y (n>0) )
    IMPRIMIR "El número es par positivo"
  SINO
    IMPRIMIR "El número no es par positivo"
FIN
```

Ejemplo 5.3: Diseñe un algoritmo que determine el mayor valor de cuatro números a, b, c, d.

Algoritmo

```
INICIO
  REAL a, b, c, d, mayor

  LEER a, b, c, d

  mayor = a
  SI( b > mayor )
    mayor = b
  SI( c > mayor )
    mayor = c
  SI( d > mayor )
    mayor = d

  IMPRIMIR mayor
FIN
```

Observación

En caso que los cuatro números sean iguales entre sí, el algoritmo da como mayor a cualquiera de los cuatro.

Ejemplo 5.4: Una tienda ha puesto en oferta la venta al por mayor de cierto producto, ofreciendo un descuento del 15% por la compra de más de 3 docenas y 10% en caso contrario. Además por la compra de más de 3 docenas se obsequia una unidad del producto por cada docena en exceso sobre 3. Diseñe un algoritmo que determine el monto de la compra, el monto del descuento, el monto a pagar y el número de unidades de obsequio por la compra de cierta cantidad de docenas del producto.

INICIO

REAL montopag, montocom, montodes, precio

ENTERO docenas, obsequio

LEER docenas, precio

montocom = docenas*precio

```
SI( docenas > 3 ){
    montodes = 0.15*montocom
    obsequio = docenas-3
```

```
}
```

```
SINO{
    montodes = 0.10*montocom
    obsequio = 0
```

```
}
```

montopag = montocom - montodes

IMPRIMIR montocom, montodes, montopag

FIN

5.5 Selección Doble en Cascada

Se dice que varias estructuras de selección doble están en cascada cuando la instrucción que sigue a un **SINO** es otro **SI** a excepción del último **SINO**. No hay límite en cuanto al número de estructuras de selección doble que pueden ponerse en cascada.

En las figuras 5.3 y 5.4 se muestran dos formas de selección doble en cascada.

Funcionamiento

Las condiciones se evalúan en orden descendente pasando de una a otra si la anterior resulta falsa. En el momento que se encuentra una condición verdadera, se efectúa la acción correspondiente a dicha condición y se corta el resto de la estructura. Si todas las condiciones resultan falsas se efectúa la acción correspondiente al último SINO

Nota. En el caso de acciones compuestas, estas serán encerradas entre llaves.

```
SI( condición C1 )
    acción A1
SINO
    SI( condición C2 )
        acción A2
    SINO
        SI( condición C3 )
```

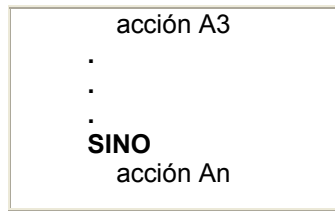


Figura 5.3 Selección doble en cascada: Forma 1

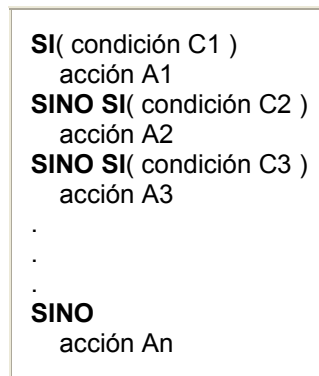


Figura 5.4 Selección doble en cascada: Forma 2

Ejemplo 5.5: Diseñe un algoritmo que determine si un número es negativo, positivo o cero.

Algoritmo (Formato 1)

```

INICIO
    REAL n

    LEER n
    SI( n > 0 )
        IMPRIMIR "Positivo"
    SINO
        SI( n < 0 )
            IMPRIMIR "Negativo"
        SINO
            IMPRIMIR "Cero"
FIN
    
```

Algoritmo (Formato 2)

```

INICIO
    REAL n

    LEER n
    SI( n > 0 )
        IMPRIMIR "Positivo"
    SINO SI( n < 0 )
        IMPRIMIR "Negativo"
    SINO
        IMPRIMIR "Cero"
FIN
    
```

Ejemplo 5.6: Considerando que las estaciones del año se numeran como 1 para primavera, 2 para verano, 3 para otoño y 4 para invierno; diseñe un algoritmo que determine el nombre de la estación del año conociendo el número de la estación.

Algoritmo

INICIO

ENTERO numero

CADENA nombre

LEER numero

SI(numero == 1)
 nombre = "Primavera"

SINO SI(numero == 2)
 nombre = "Verano"

SINO SI(numero == 3)
 nombre = "Otoño"

SINO
 nombre = "Invierno"

IMPRIMIR nombre

FIN

5.6 Estructuras de Selección Anidadas

Se dice que una estructura SI (o SI-SINO) esta anidada cuando esta contenida dentro de otra estructura SI o dentro de otra estructura SI-SINO. No existe límite en cuanto al nivel de anidamiento.

Por ejemplo, una estructura SI con tres niveles de anidamiento tendría el siguiente formato:

```
SI( condición C1 ){
    acción A1
    SI( condición C2 ){
        acción A2
        SI( condición C3 )
            acción A3
    }
}
```

En general, el anidamiento podría evitarse usando el operador lógico "y". Así, la anterior selección SI anidada puede descomponerse en tres estructuras de selección simple no anidadas consecutivas, como se muestra a continuación.

SI(condición C1)
 acción A1

SI(condición C1 **y** condición C2)
 acción A2

SI(condición C1 **y** condición C2 **y** condición C3)
 acción A3

La estructura de [selección doble en cascada](#) es un caso especial de la estructura SI..SINO anidada.

Ejemplo 5.7: Diseñe un algoritmo que lea un número de tres cifras y determine si es o no capicúa. Un número es capicúa si es igual al revés del número.

Observación

Observe que al ser el número de tres cifras, para ser capicúa es suficiente comprobar que la cifra de las **unidades** sea igual a la cifra de las **centenas**.

Algoritmo

INICIO

ENTERO numero, unidad, centena

LEER numero

SI(numero >= 100 && numero <= 999){

 unidad = numero%10

 centena = numero/100

SI(unidad == centena)

IMPRIMIR "El número es capicúa"

SINO

IMPRIMIR "El número no es capicúa"

 }

SINO

IMPRIMIR "El número no tiene tres cifras"

FIN

5.7 Estructura de Selección Múltiple SEGUN

La estructura de selección múltiple **SEGUN** permite elegir una ruta de entre varias rutas posibles, usando para ello una variable denominada **selector**. El selector se compara con una lista de constantes enteras o de carácter C1, C2, ..., Cn para cada una de las cuales hay una acción A1, A2, ..., An y:

- Si el selector coincide con una constante de la lista, se ejecuta la acción correspondiente a dicha constante.
- Si el selector no coincide con ninguna constante de la lista, se ejecuta la acción Df correspondiente al **SINO**, si es que existe.

Las acciones A1, A2, A3, ..., An pueden ser acciones simples(una sola acción) o acciones compuestas (un conjunto de acciones).

En la Figura 5.3 se muestra el seudo código de la *estructura de selección múltiple*.

Nota.- En el caso de acciones compuestas, estas no necesitan estar encerradas entre llaves.

```

SEGUN ( selector ){
  CASO C1 : acción A1
  CASO C2 : acción A2
  CASO C3 : acción A3
  .
  .
  .
  CASO Cn : acción An
  SINO : acción Df
}

```

Figura 5.3 Estructura de selección múltiple: pseudo código

Ejemplo 5.8: Resuelva el ejemplo 5.6 usando selección múltiple.

Algoritmo

INICIO

ENTERO numero

CADENA nombre

LEER numero

```

SEGUN(numero){
  CASO 1 : nombre = "Primavera"
  CASO 2 : nombre = "Verano"
  CASO 3 : nombre = "Otoño"
  SINO : nombre = "Invierno"
}

```

IMPRIMIR nombre

FIN

5.8 Ejercicios

Ejercicio 1: Diseñe un algoritmo que califique el puntaje obtenido en el lanzamiento de tres dados en función a la cantidad seis obtenidos, de acuerdo a lo siguiente:

- Seis en los tres dados, excelente.
- Seis en dos dados, muy bien.
- Seis en un dado, regular.
- Ningún seis, pésimo.

Algoritmo

INICIO

// Declaración de variables

ENTERO d1, d2, d3

// Entrada de datos

LEER d1, d2, d3

// Obtención de la calificación

```

SI( d1 + d2 + d3 == 18 )
  IMPRIMIR "Excelente"
SINO SI( d1+d2 == 12 ó d1+d3 == 12 ó d2+d3 == 12 )
  IMPRIMIR "Muy bien"
SINO SI( d1 == 6 ó d2 == 6 ó d3 == 6 )
  IMPRIMIR "Regular"
SINO
  IMPRIMIR "Pésimo"
FIN

```

Ejercicio 2: Una compañía dedicada al alquiler de automóviles cobra \$30 hasta un máximo de 300 Km. de distancia recorrida. Para más de 300 Km. y hasta 1000 Km., cobra \$30 más un monto adicional de \$ 0.15 por cada kilómetro en exceso sobre 300. Para más de 1000 Km. cobra \$30 más un monto adicional de \$ 0.10 por cada kilómetro en exceso sobre 1000. Los precios ya incluyen el 18% del impuesto general a las ventas, IGV. Diseñe un algoritmo que determine el monto a pagar por el alquiler de un vehículo y el monto incluido del impuesto.

Algoritmo

```

INICIO
  // Declaración de variables
  REAL kilomrec, montofijo, montoadic, montopag, montoigv
  REAL IGV = 0.18

  // Entrada de datos
  LEER kilomrec

  // Cálculo de montos
  montofijo = 30*kilomrec
  SI( kilomrec <= 300 )
    montoadic = 0
  SINO SI( kilomrec <= 1000 )
    montoadic = 0.15*(kilomrec-300)
  SINO
    montoadic = 0.15*700 + 0.10*(kilomrec-1000)
  montopag = montofijo + montoadic
  montoigv = IGV*montopag /(1+IGV)

  // Salida de resultados
  IMPRIMIR montoPag, montoIgv
FIN

```

Ejercicio 3: Diseñe un algoritmo que determine quienes son contemporáneos entre Juan, Mario y Pedro.

Algoritmo

```

INICIO
  ENTERO juan, mario, pedro

  LEER juan, mario, pedro

  SI( juan == mario y mario == pedro )
    IMPRIMIR "Los tres son contemporáneos"
  SINO SI( juan == mario )
    IMPRIMIR "Juan y Mario son contemporáneos"
  SINO SI( juan == pedro )
    IMPRIMIR "Juan y Pedro son contemporáneos"

```

```

SINO SI( mario == pedro )
    IMPRIMIR "Mario y Pedro son contemporáneos"
SINO
    IMPRIMIR "Los tres tienen edades diferentes entre sí"
FIN

```

Ejercicio 4: El promedio de prácticas de un curso se calcula en base a cuatro prácticas calificadas de las cuales se elimina la nota menor y se promedian las tres notas más altas. Diseñe un algoritmo que determine la nota eliminada y el promedio de prácticas de un estudiante.

Algoritmo

```

INICIO
    REAL pc1, pc2, pc3, pc4, pcmenor, promedio

    LEER pc1, pc2, pc3, pc4

    // Determina la nota menor
    pcmenor = pc1
    SI( pc2 < pcmenor )
        pcmenor = pc2
    SI( pc3 < pcmenor )
        pcmenor = pc3
    SI( pc4 < pcmenor )
        pcmenor = pc4

    // Determina el promedio descontando la nota menor
    promedio = (pc1 + pc2 + pc3 + pc4 - pcmenor )/3

    IMPRIMIR promedio, pcmenor
FIN

```

Ejercicio 5: Diseñe un algoritmo que lea tres longitudes y determine si forman o no un triángulo. Si es un triángulo determine de que tipo de triángulo se trata entre: equilátero (si tiene tres lados iguales), isósceles (si tiene dos lados iguales) o escaleno (si tiene tres lados desiguales). Considere que para formar un triángulo se requiere que: "el lado mayor sea menor que la suma de los otros dos lados".

Algoritmo

```

INICIO
    REAL L1, L2, L3, suma

    LEER L1, L2, L3

    // Determina el lado mayor
    mayor = L1
    SI( L2 > mayor )
        mayor = L2
    SI( L3 > mayor )
        mayor = L3

    // Determina la suma de los lados a excepción del lado mayor
    suma = L1 + L2 + L3 - mayor

    // Determina de que tipo de triángulo se trata
    SI( mayor < suma ){
        SI( ( L1 == L2 ) y ( L2 == L3 ) )

```

```

    IMPRIMIR "Triángulo equilátero"
  SINO SI( ( L1 == L2 ) o ( L1 == L3 ) o ( L2 == L3 ) )
    IMPRIMIR "Triángulo isósceles"
  SINO
    IMPRIMIR "Triángulo escaleno"
  }
SINO
  IMPRIMIR "No es un triángulo"
FIN

```

Ejercicio 6: Diseñe un algoritmo que lea tres números enteros y determine el menor valor positivo. Si los números positivos son iguales, dar como menor a cualquiera de ellos.

Algoritmo

```

INICIO
  ENTERO a, b, c

  LEER a, b, c

  SI( a > 0 ){ // Aquí a, b y c podrían ser positivos
    menor = a
    SI( b > 0 y b < menor )
      menor = b
    SI( c > 0 y c < menor )
      menor = c
    IMPRIMIR menor
  }
  SINO SI( b > 0 ){ // Aquí sólo b y c podrían ser positivos
    menor = b
    SI( c > 0 y c < menor )
      menor = c
    IMPRIMIR menor
  }
  SINO SI( c > 0 ){ // Aquí sólo c podría ser positivo
    menor = c
    IMPRIMIR menor
  }
  SINO
    IMPRIMIR "No hay números positivos"
FIN

```

Ejercicio 7: Diseñe un algoritmo que lea tres números y los imprima de mayor a menor y de menor a mayor.

Algoritmo

```

INICIO
  REAL n1, n2, n3, mayor, menor, medio

  LEER n1, n2, n3

  // Determina el menor
  menor = n1
  SI( n2 < menor )
    menor = n2
  SI( n3 < menor )

```

```

menor = n3

// Determina el mayor
mayor = n1
SI( n2 > mayor )
    mayor = n2
SI( n3 > mayor )
    mayor = n3

// Determina el medio
medio = n1+n2+n3-mayor-menor

// Imprime en orden ascendente
IMPRIMIR menor, medio, mayor

// Imprime en orden descendente
IMPRIMIR mayor, medio, menor
FIN

```

Ejercicio 8: Diseñe un algoritmo para obtener el grado de eficiencia de un operario de una fábrica de tornillos, de acuerdo a las siguientes condiciones, que se le imponen para un período de prueba:

- Menos de 200 tornillos defectuosos.
- Más de 10000 tornillos producidos.

El grado de eficiencia se determina de la siguiente manera:

- Si no cumple ninguna de las condiciones, grado 5.
- Si sólo cumple la primera condición, grado 6.
- Si sólo cumple la segunda condición, grado 7.
- Si cumple las dos condiciones, grado 8.

Algoritmo 1

```

INICIO
ENTERO torpro, tordef, grado

LEER torpro, tordef

// Determina el grado de eficiencia
SI( tordef < 200 ){
    SI( torpro > 10000 )
        grado = 8
    SINO
        grado = 6
}
SINO{
    SI( torpro > 10000 )
        grado = 7
    SINO
        grado = 5
}

IMPRIMIR grado
FIN

```

Algoritmo 2

INICIO**ENTERO** torpro, tordef, grado**LEER** torpro, tordef

// Determina el grado de eficiencia

SI(tordef < 200 y torpro > 10000)

grado = 8

SINO SI(tordef < 200)

grado = 6

SINO SI(torpro > 10000)

grado = 7

SINO

grado = 5

IMPRIMIR grado**FIN**

Ejercicio 9: Se cuenta con los votos obtenidos por Juan, Pedro y Maria en una elección democrática a la presidencia de un club. Para ganar la elección se debe obtener como mínimo el 50% de los votos más 1. En caso que no haya un ganador se repite la elección en una segunda vuelta. Van a la segunda vuelta los dos que obtengan la más alta votación ó, los tres en caso de producirse un empate doble (entre los dos con menor votación) o un empate triple. Diseñe un algoritmo que determine el resultado de la elección.

Algoritmo**INICIO****ENTERO** vjuan, vpedro, vmaria**LEER** vjuan, vpedro, vmaria

// Determina la votación total

vtotal = vjuan + vpedro + vmaria

// Determina la votación máxima

vmax = vjuan

SI(vpedro > vmax)

vmax = vpedro

SI(vmaria > vmax)

vmax = vmaria

// Determina el resultado de la elección

SI(vmax > vtotal/2 + 1){ // Hay ganador **SI**(vjuan == vmax) **IMPRIMIR** "Ganó Juan" **SINO SI**(vpedro == vmax) **IMPRIMIR** "Ganó Pedro" **SINO** **IMPRIMIR** "Ganó María"

}

SINO{ // No hay ganador **SI**(vjuan < vpedro y vjuan < vmaria) **IMPRIMIR** "Debe haber segunda vuelta entre Pedro y María" **SINO SI**(vpedro < vjuan y vpedro < vmaria) **IMPRIMIR** "Debe haber segunda vuelta entre Juan y María" **SINO SI**(vmaria < vjuan y vmaria < vpedro) **IMPRIMIR** "Debe haber segunda vuelta entre Juan y Pedro"

```

    SINO
      IMPRIMIR "Debe haber segunda vuelta entre los tres"
    }
  FIN

```

Ejercicio 10: Diseñe un algoritmo que lea un número entero de 3 cifras, y forme el mayor número posible con las cifras del número ingresado. El número formado debe tener el mismo signo que el número ingresado.

Algoritmo

INICIO

ENTERO num1, num2, numaux, uni, dec, cen, menor, mayor, medio

LEER num1

```

SI( ( num1 >= 100 y num1 <= 999 ) y ( num1 >= -999 y num1 <= -100 ) ){
  // Guarda el número en una variable auxiliar para preservar el signo
  numaux = num1

```

// Cambia el signo de num1 en caso de ser negativo

```

SI( num1 < 0 )
  num1 = -num1

```

// Determina las cifras del número

```

cen = num1/100
dec = (num1%100)/10
uni = (num1%100)%10

```

// Determina la cifra menor

```

menor = cen
SI( dec < menor )
  menor = dec
SI( uni < menor )
  menor = uni

```

// Determina la cifra mayor

```

mayor = cen
SI( dec > mayor )
  mayor = dec
SI( uni > mayor )
  mayor = uni

```

// Determina la cifra del medio

```

medio = cen+dec+uni-mayor-menor

```

// Forma el nuevo número

```

SI( numaux > 0 )
  num2 = mayor*100 + medio*10 + menor
SINO
  num2 = -1*(menor*100 + medio*10 + mayor*100)

```

// Imprime el nuevo número

```

IMPRIMIR num2

```

```

}
SINO
  IMPRIMIR "El número no tiene tres cifras"
FIN

```

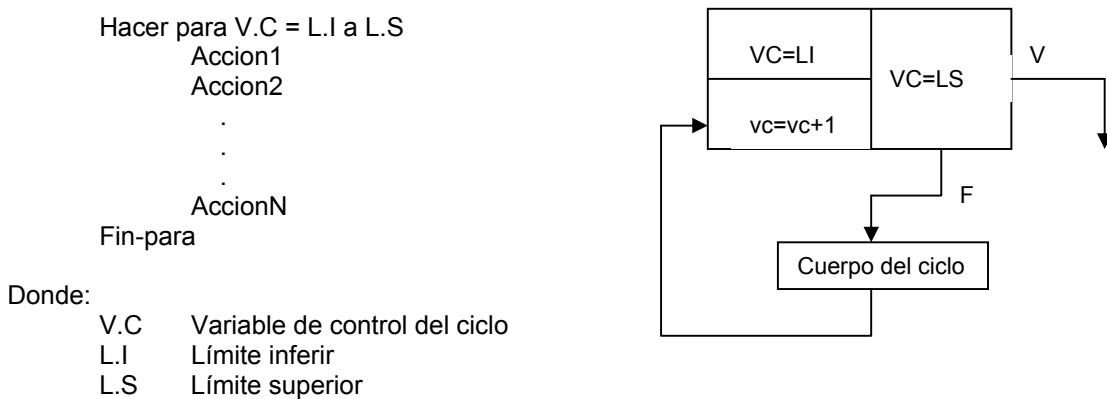

Capítulo 6: Estructuras de Repetición (Cíclicas)

2.1 Introducción

Se llaman problemas repetitivos o cíclicos a aquellos en cuya solución es necesario utilizar un mismo conjunto de acciones que se puedan ejecutar una cantidad específica de veces. Esta cantidad puede ser fija (previamente determinada por el programador) o puede ser variable (estar en función de algún dato dentro del programa). Los ciclos se clasifican en:

- **Ciclos con un Número Determinado de Iteraciones (Hacer-Para)**

Son aquellos en que el número de iteraciones se conoce antes de ejecutarse el ciclo. La forma de esta estructura es la siguiente:



En este ciclo la variable de control toma el valor inicial del ciclo y el ciclo se repite hasta que la variable de control llegue al límite superior.

Problemas (Hacer para)

- 1) Calcular el promedio de un alumno que tiene 7 calificaciones en la materia de Diseño Estructurado de Algoritmos.

```

Inicio
Sum=0
Leer Nom
Hacer para c = 1 a 7
    Leer calif
    Sum = sum + calif
Fin-para
prom = sum / 7
Imprimir prom
Fin.
  
```

- 2) Leer 10 números y obtener su cubo y su cuarta.

```

Inicio
Hacer para n = 1 a 10
    Leer num
    cubo = num * num * num
    cuarta = cubo * num
    Imprimir cubo, cuarta
Fin-para
Fin.
  
```

- 3) Leer 10 números e imprimir solamente los números positivos.

```

Inicio
  Hacer para n = 1 a 10
    Leer num
    Si num > 0 entonces
      Imprimir num
    fin-si
  Fin-para
Fin.

```

- 4) Leer 20 números e imprimir cuántos son positivos, cuántos negativos y cuántos neutros.

```

Inicio
  cn = 0
  cp = 0
  cneg = 0
  Hacer para x = 1 a 20
    Leer num
    Si num = 0 entonces
      cn = cn + 1
    si no
      Si num > 0 entonces
        cp = cp + 1
      si no
        cneg = cneg + 1
    Fin-si
  Fin-para
  Imprimir cn, cp, cneg
Fin.

```

- 5) Leer 15 números negativos y convertirlos a positivos e imprimir dichos números.

```

Inicio
  Hacer para x = 1 a 15
    Leer num
    pos = num * -1
    Imprimir num, pos
  Fin-para
Fin.

```

- 6) Suponga que se tiene un conjunto de calificaciones de un grupo de 40 alumnos. Realizar un algoritmo para calcular la calificación media y la calificación mas baja de todo el grupo.

```

Inicio
  sum = 0
  baja = 9999
  Hacer para a = 1 a 40
    Leer calif
    sum = sum + calif
    Si calif < baja entonces
      baja = calif
    fin-si
  Fin-para
  media = sum / 2
  Imprimir media, baja
fin

```

- 7) Calcular e imprimir la tabla de multiplicar de un número cualquiera. Imprimir el multiplicando, el multiplicador y el producto.

```

Inicio
  Leer num
  Hacer para X = 1 a 10
    resul = num * x
    Imprimir num, " * ", X, " = ", resul
  Fin-para
fin

```

- 8) Simular el comportamiento de un reloj digital, imprimiendo la hora, minutos y segundos de un día desde las 0:00:00 horas hasta las 23:59:59 horas.

```

Inicio
  Hacer para h = 1 a 23
    Hacer para m = 1 a 59
      Hacer para s = 1 a 59
        Imprimir h, m, s
      Fin-para
    Fin-para
  Fin-para
fin.

```

Problemas Propuestos

- 1) Una persona debe realizar un muestreo con 50 personas para determinar el promedio de peso de los niños, jóvenes, adultos y viejos que existen en su zona habitacional. Se determinan las categorías con base en la siguiente tabla:

CATEGORÍA	EDAD
Niños	0 - 12
Jóvenes	13 - 29
Adultos	30 - 59
Viejos	60 en adelante

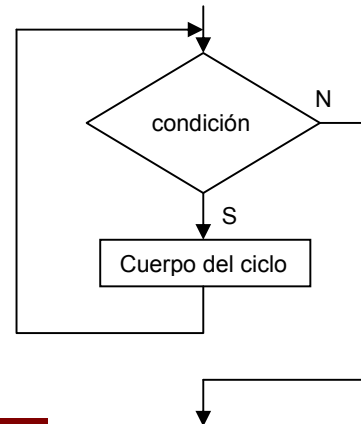
- 2) Al cerrar un expendio de naranjas, 15 clientes que aun no han pagado recibirán un 15% de descuento si compran más de 10 kilos. Determinar cuanto pagara cada cliente y cuanto percibirá la tienda por esas compras.
- 3) En un centro de verificación de automóviles se desea saber el promedio de puntos contaminantes de los primeros 25 automóviles que lleguen. Asimismo se desea saber los puntos contaminantes del carro que menos contamina y del que mas contamina.
- 4) Un entrenador le ha propuesto a un atleta recorrer una ruta de cinco kilómetros durante 10 días, para determinar si es apto para la prueba de 5 Kilómetros o debe buscar otra especialidad. Para considerarlo apto debe cumplir por lo menos una de las siguientes condiciones:
- Que en ninguna de las pruebas haga un tiempo mayor a 16 minutos.
 - Que al menos en una de las pruebas realice un tiempo mayor a 16 minutos.
 - Que su promedio de tiempos sea menor o igual a 15 minutos.
- 5) Un Zoólogo pretende determinar el porcentaje de animales que hay en las siguientes tres categorías de edades: de 0 a 1 año, de más de 1 año y menos de 3 y de 3 o más años. El zoológico todavía no esta seguro del animal que va a estudiar. Si se decide por elefantes solo tomara una muestra de 20 de ellos; si se decide por las jirafas, tomara 15 muestras, y si son chimpancés tomara 40.

- **Ciclos con un Numero Indeterminado de Iteraciones Hacer-Mientras, Repetir-Hasta)**

Son aquellos en que el número de iteraciones no se conoce con exactitud, ya que esta dado en función de un dato dentro del programa.

- **Hacer-Mientras:** Esta es una estructura que repetirá un proceso durante “N” veces, donde “N” puede ser fijo o variable. Para esto, la instrucción se vale de una condición que es la que debe cumplirse para que se siga ejecutando. Cuando la condición ya no se cumple, entonces ya no se ejecuta el proceso. La forma de esta estructura es la siguiente:

```
Hacer mientras <condición>
    Accion1
    Accion2
    .
    .
    AccionN
Fin-mientras
```



Problemas (Hacer Mientras)

- 1) Una compañía de seguros tiene contratados a n vendedores. Cada uno hace tres ventas a la semana. Su política de pagos es que un vendedor recibe un sueldo base, y un 10% extra por comisiones de sus ventas. El gerente de su compañía desea saber cuanto dinero obtendrá en la semana cada vendedor por concepto de comisiones por las tres ventas realizadas, y cuanto tomando en cuenta su sueldo base y sus comisiones.
- 2) En una empresa se requiere calcular el salario semanal de cada uno de los n obreros que laboran en ella. El salario se obtiene de la siguiente forma:

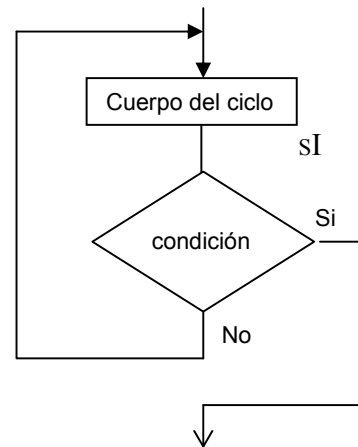
Si el obrero trabaja 40 horas o menos se le paga \$20 por hora

Si trabaja más de 40 horas se le paga \$20 por cada una de las primeras 40 horas y \$25 por cada hora extra.
- 3) Determinar cuantos hombres y cuantas mujeres se encuentran en un grupo de n personas, suponiendo que los datos son extraídos alumno por alumno.
- 4) El Departamento. de Seguridad Publica y Tránsito del DF. desea saber, de los n autos que entran a la ciudad de México, cuantos entran con calcomanía de cada color. Conociendo el último dígito de la placa de cada automóvil se puede determinar el color de la calcomanía utilizando la siguiente relación:

DÍGITO	COLOR
1 o 2	amarilla
3 o 4	rosa
5 o 6	roja
7 o 8	verde
9 o 0	azul
- 5) Obtener el promedio de calificaciones de un grupo de n alumnos.
- 6) Una persona desea invertir su dinero en un banco, el cual le otorga un 2% de interés. Cual será la cantidad de dinero que esta persona tendrá al cabo de un año si la ganancia de cada mes es reinvertida?
- 7) Calcular el promedio de edades de hombres, mujeres y de todo un grupo de alumnos.

- 8) Encontrar el menor valor de un conjunto de n números dados.
 - 9) Encontrar el mayor valor de un conjunto de n números dados.
 - 10) En un supermercado un cajero captura los precios de los artículos que los clientes compran e indica a cada cliente cual es el monto de lo que deben pagar. Al final del día le indica a su supervisor cuanto fue lo que cobro en total a todos los clientes que pasaron por su caja.
 - 11) Cinco miembros de un club contra la obesidad desean saber cuanto han bajado o subido de peso desde la ultima vez que se reunieron. Para esto se debe realizar un ritual de pesaje en donde cada uno se pesa en diez básculas distintas para así tener el promedio mas exacto de su peso. Si existe diferencia positiva entre este promedio de peso y el peso de la ultima vez que se reunieron, significa que subieron de peso. Pero si la diferencia es negativa, significa que bajaron. Lo que el problema requiere es que por cada persona se imprima un letrero que diga: "SUBIO" o "BAJO" y la cantidad de kilos que subió o bajo de peso.
 - 12) Se desea obtener el promedio de g grupos que están en un mismo año escolar; siendo que cada grupo puede tener n alumnos que cada alumno puede llevar m materias y que en todas las materias se promedian tres calificaciones para obtener el promedio de la materia. Lo que se desea desplegar es el promedio de los grupos, el promedio de cada grupo y el promedio de cada alumno.
- **Repetir-Hasta:** Esta es una estructura similar en algunas características, a la anterior. Repite un proceso una cantidad de veces, pero a diferencia del Hacer-Mientras, el Repetir-Hasta lo hace hasta que la condición se cumple y no mientras, como en el Hacer-Mientras. Por otra parte, esta estructura permite realizar el proceso cuando menos una vez, ya que la condición se evalúa al final del proceso, mientras que en el Hacer-Mientras puede ser que nunca llegue a entrar si la condición no se cumple desde un principio. La forma de esta estructura es la siguiente:

Repetir
 Accion1
 Accion2
 .
 .
 AccionN
 Hasta <condición>



Problemas Repetir - Hasta

- 1) En una tienda de descuento las personas que van a pagar el importe de su compra llegan a la caja y sacan una bolita de color, que les dirá que descuento tendrán sobre el total de su compra. Determinar la cantidad que pagara cada cliente desde que la tienda abre hasta que cierra. Se sabe que si el color de la bolita es rojo el cliente obtendrá un 40% de descuento; si es amarilla un 25% y si es blanca no obtendrá descuento.
- 2) En un supermercado una ama de casa pone en su carrito los artículos que va tomando de los estantes. La señora quiere asegurarse de que el cajero le cobre bien lo que ella ha comprado, por lo que cada vez que toma un articulo anota su precio junto con la cantidad de artículos iguales que ha tomado y determina cuanto dinero gastara en ese articulo; a esto le suma lo que ira

gastando en los demás artículos, hasta que decide que ya tomo todo lo que necesitaba. Ayúdale a esta señora a obtener el total de sus compras.

- 3) un teatro otorga descuentos según la edad del cliente. determinar la cantidad de dinero que el teatro deja de percibir por cada una de las categorías. Tomar en cuenta que los niños menores de 5 años no pueden entrar al teatro y que existe un precio único en los asientos. Los descuentos se hacen tomando en cuenta el siguiente cuadro:

	Edad	Descuento
Categoría 1	5 - 14	35 %
Categoría 2	15 - 19	25 %
Categoría 3	20 - 45	10 %
Categoría 4	46 - 65	25 %
Categoría 5	66 en adelante	35 %

Problemas Propuestos

- 1) La presión, volumen y temperatura de una masa de aire se relacionan por la formula:

$$\text{masa} = \frac{\text{presión} * \text{volumen}}{0.37 * (\text{temperatura} + 460)}$$

Calcular el promedio de masa de aire de los neumáticos de n vehículos que están en compostura en un servicio de alineación y balanceo. Los vehículos pueden ser motocicletas o automóviles.

- 2) Determinar la cantidad semanal de dinero que recibirá cada uno de los n obreros de una empresa. Se sabe que cuando las horas que trabajo un obrero exceden de 40, el resto se convierte en horas extras que se pagan al doble de una hora normal, cuando no exceden de 8; cuando las horas extras exceden de 8 se pagan las primeras 8 al doble de lo que se paga por una hora normal y el resto al triple.
- 3) En una granja se requiere saber alguna información para determinar el precio de venta por cada kilo de huevo. Es importante determinar el promedio de calidad de las n gallinas que hay en la granja. La calidad de cada gallina se obtiene según la formula:

$$\text{calidad} = \frac{\text{peso de la gallina} * \text{altura de la gallina}}{\text{número de huevos que pone}}$$

Finalmente para fijar el precio del kilo de huevo, se toma como base la siguiente tabla:

PRECIO TOTAL DE CALIDAD	PESO POR KILO DE HUEVO
mayor o igual que 15	1.2 * promedio de calidad
mayor que 8 y menor que 15	1.00 * promedio de calidad
menor o igual que 8	0.80 * promedio de calidad

- 4) En la Cámara de Diputados se levanta una encuesta con todos los integrantes con el fin de determinar que porcentaje de los n diputados está a favor del Tratado de Libre Comercio, que porcentaje está en contra y que porcentaje se abstiene de opinar.
- 5) Una persona que va de compras a la tienda “Enano, S.A.”, decide llevar un control sobre lo que va comprando, para saber la cantidad de dinero que tendrá que pagar al llegar a la caja. La tienda tiene una promoción del 20% de descuento sobre aquellos artículos cuya etiqueta sea roja. Determinar la cantidad de dinero que esta persona deberá pagar.
- 6) Un censador recopila ciertos datos aplicando encuestas para el último Censo Nacional de Población y Vivienda. Desea obtener de todas las personas que alcance a encuestar en un día, que porcentaje tiene estudios de primaria, secundaria, carrera técnica, estudios profesionales y estudios de postgrado.

- 7) Un jefe de casilla desea determinar cuantas personas de cada una de las secciones que componen su zona asisten el día de las votaciones. Las secciones son: norte, sur y centro. También desea determinar cuál es la sección con mayor número de votantes.
- 8) Un negocio de copias tiene un límite de producción diaria de 10 000 copias si el tipo de impresión es offset y de 50 000 si el tipo es estándar. Si hay una solicitud de un empleado que tiene que verificar que las copias pendientes hasta el momento y las copias solicitadas no excedan del límite de producción. Si el límite de producción se excediera el trabajo solicitado no podría ser aceptado. El empleado necesita llevar un buen control de las copias solicitadas hasta el momento para decidir en forma rápida si los trabajos que se soliciten en el día se deben aceptar o no.
- 9) Calcular la suma siguiente:
100 + 98 + 96 + 94 + . . . + 0 en este orden
- 10) Leer 50 calificaciones de un grupo de alumnos. Calcule y escriba el porcentaje de reprobados. Tomando en cuenta que la calificación mínima aprobatoria es de 70.
- 11) Leer por cada alumno de Diseño estructurado de algoritmos su número de control y su calificación en cada una de las 5 unidades de la materia. Al final que escriba el número de control del alumno que obtuvo mayor promedio. Suponga que los alumnos tienen diferentes promedios.
- 12) El profesor de una materia desea conocer la cantidad de sus alumnos que no tienen derecho al examen de nivelación.

Diseñe un algoritmo que lea las calificaciones obtenidas en las 5 unidades por cada uno de los 40 alumnos y escriba la cantidad de ellos que no tienen derecho al examen de nivelación.
- 13) Leer los 250,000 votos otorgados a los 3 candidatos a gobernador e imprimir el número del candidato ganador y su cantidad de votos.
- 14) Suponga que tiene usted una tienda y desea registrar las ventas en su computadora. Diseñe un algoritmo que lea por cada cliente, el monto total de su compra. Al final del día que escriba la cantidad total de ventas y el número de clientes atendidos.

Capítulo 7: MANEJO DE MÓDULOS

7.1 Definición, Función y Manipulación

Un problema complejo se puede dividir en pequeños subproblemas más sencillos. Estos subproblemas se conocen como “*Módulos*” y su complementación en un lenguaje se llama subprograma (procedimientos y funciones).

Un subprograma realiza las mismas acciones que un programa, sin embargo, un subprograma lo utiliza solamente un programa para un propósito específico.

Un subprograma recibe datos de un programa y le devuelve resultados (el programa “llama” o “invoca” al subprograma, este ejecuta una tarea específica y devuelve el “control” al programa que lo llama).

Función: Una función en matemáticas, es una operación que toma uno o más valores (argumentos) y devuelve un resultado (valor de la función para los argumentos dados). Por ejemplo:

$$F(x) = \frac{x}{1+x^2}$$

Donde:

F = Nombre de la función

x = Argumento (también conocido como parámetro formal)

Definición de funciones: Una definición de función se presenta de la siguiente manera:

```

Función nombre_funcion (p1, p2, ..., pn)
Inicio
  Bloque de instrucciones
Fin

```

Donde:

Función:	Es la palabra clave que nos indica una definición de función.
Nombre_funcion:	Es el identificador con el cual se reconoce a la función en el cuerpo del algoritmo principal.
P1,p2,...,pn	Es el grupo de parámetros que define a la función.

Llamado a una función

Cuando definimos una función sólo le indicamos al algoritmo que esta función existe, pero una definición de función no implica la realización de las instrucciones que la constituyen. Para hacer uso de una función, el algoritmo principal la debe llamar. Por ejemplo:

```

Función F(X)
  Inicio
    F = X / (1 + X^2)
  Fin
Inicio
  Imprimir "Este es el algoritmo principal"
  Leer N
  R = F(N) ← llamado de la función
  Imprimir "El resultado de la función es:",R
Fin

```

Problemas Propuestos

1) Diseñe un algoritmo que llene una matriz de $10 * 10$ y determine:

- A) El número mayor almacenado en la matriz
- B) El número mayor almacenado en cada renglón
- C) La columna que tuvo la máxima suma
- D) El renglón que tuvo la máxima suma

Diseña una función para cada inciso.

2) Diseñe un algoritmo que lea un número y mediante una función regrese el valor de 1 si el número es positivo y -1 si es negativo).