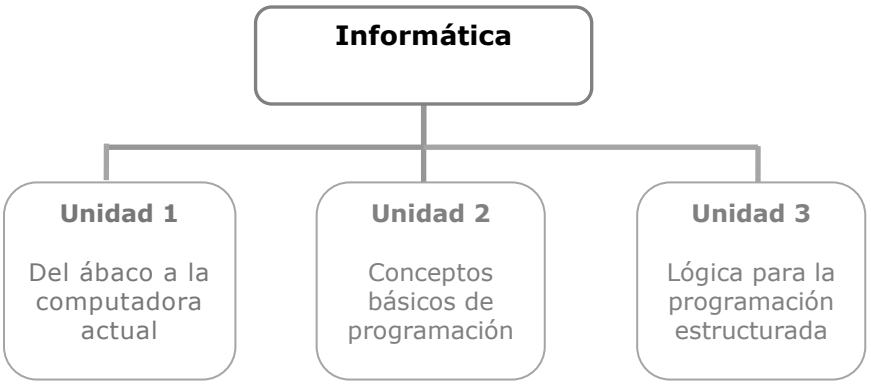
FACULTAD DE TECNOLOGIA INFORMÁTICA

MANUAL DE APOYO PARA INGRESANTES

Asignatura: INFORMÁTICA

Autor de contenidos: Prof. Mario L. A. Zani

Coordinador: Dr. Marcelo De Vincenzi



Índice (puede hacer clic sobre el título para ir al contenido)

Unidad 1 - Del ábaco a la computadora actual

Presentación

En esta unidad usted tendrá la oportunidad de conocer la historia de la computación desde sus orígenes hasta nuestros días; abordaremos los cambios tecnológicos y de paradigmas que se fueron desarrollando a lo largo de su breve pero intensa historia.

También nos abocaremos al estudio de los componentes que integran las computadoras tal como las conocemos actualmente: el *hardware* y el *software*. Y para ello apelaremos tanto a su experiencia como a su capacidad de indagación en tanto le pediremos que realice relevamientos de información en el mercado acerca de estos temas.

Tenga en cuenta que los contenidos que trabajaremos a lo largo de esta primera unidad son básicos y fundamentales para su formación, y los iremos retomando y profundizando en los siguientes desarrollos.

Nos proponemos que a través del estudio de los contenidos enunciados y de las actividades propuestas adquiera capacidad para:

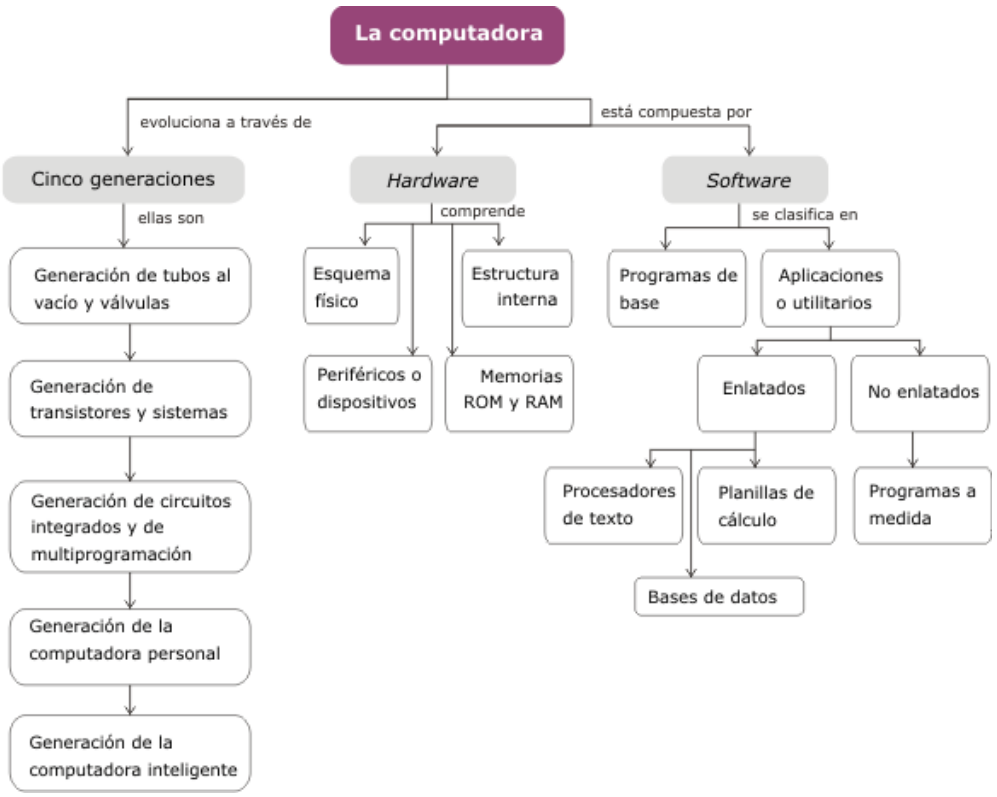
- Saber cómo evolucionaron las computadoras a través de la historia.
- Conocer los componentes de una computadora y sus principales características.
- Comprender las diferentes clasificaciones de *software*.

A continuación, encontrará a modo de índice un detalle de los contenidos y actividades que integran esta unidad. Usted deberá ir avanzando en el estudio y profundización de los diferentes temas, realizando las lecturas requeridas y elaborando las actividades propuestas.

Deseándole éxito en esta etapa que inicia, lo invitamos a comenzar.

Organizador Gráfico de la Unidad 1

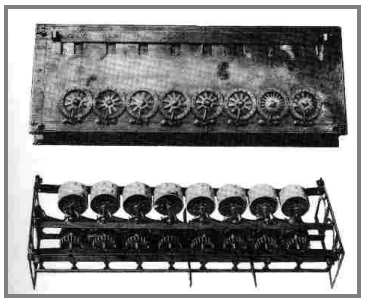
El siguiente esquema le permitirá visualizar la interrelación entre los conceptos que a continuación abordaremos.



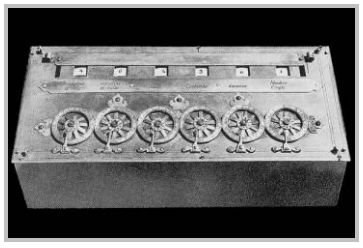
1. La computadora: breve reseña de su evolución histórica

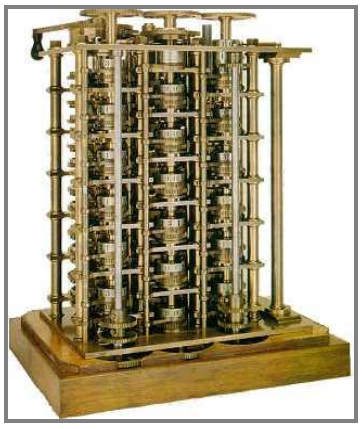
Las computadoras no se crearon en los últimos años, en realidad el hombre siempre buscó tener dispositivos que le ayudaran a efectuar cálculos precisos y rápidos. Una breve reseña histórica nos permitirá comprender su evolución a lo largo del tiempo hasta llegar a las computadoras tal como se han desarrollado actualmente.

Los chinos hace más de 3000 años diseñaron **el ábaco**. Con este instrumento, con bolillas que corrían de izquierda a derecha, hacían cálculos y todavía hoy es usado en Japón y en China.



El matemático francés **Blaise Pascal** crea, en 1642, una máquina mecánica de sumar, parecida a los cuentakilómetros de los autos, que presentaba algunas dificultades con las sumas largas. En 1671, el filósofo y matemático alemán **Gottfried Wilhelm Leibniz** le agregó la posibilidad de restar, multiplicar y dividir. Era una máquina con ruedas dentadas; cada rueda tenía 10 dientes y cada uno de ellos correspondía a los dígitos del 0 a 9. Los conceptos de esta máquina se utilizaron durante mucho tiempo, pero exigían la intervención de un operador que debía copiar los resultados parciales en un papel.

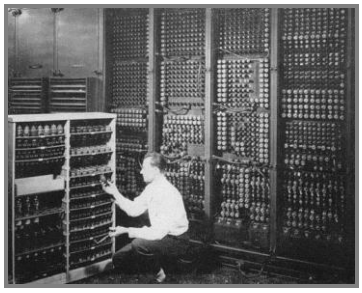




También en el siglo XIX el matemático e inventor británico **Charles Babbage** elaboró los principios de la computadora digital moderna. Babbage **diseñó y desarrolló** la primera computadora de uso general a la que llamó "**Máquina de las diferencias".** Un genio para ese momento histórico a quien el contexto no ayudó para terminar de construirla.

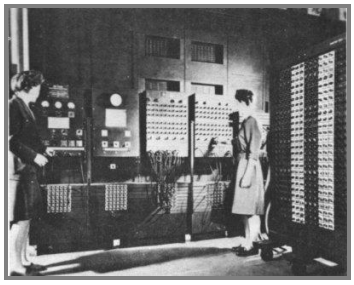
La primera operación de procesamiento de datos fue lograda en 1890 por el estadístico estadounidense Herman Hollerith. Él desarrolló un sistema mecánico para calcular y agrupar datos de censo basado en tarjetas perforadas. Este sistema fue utilizado en el censo de la población en Estados Unidos, donde se logró por primera vez que los resultados fueran conocidos a los dos años y medio, cuando en el censo anterior se demoraron siete años.

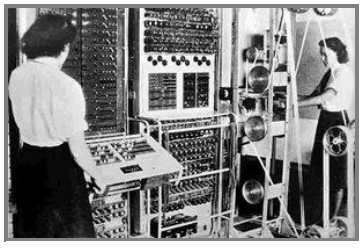
La primera mujer programadora fue la matemática británica Augusta Ada Byron (1815-1852), hija del poeta inglés Lord Byron, quien se interesó en los estudios de Babbage a quien ayudó.



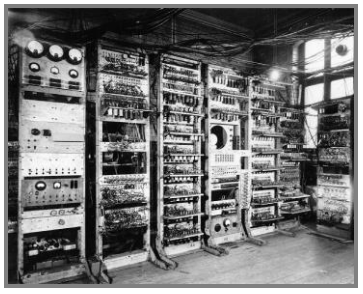
La primera computadora totalmente electrónica, llamada **ENIAC** *(Electronic Numerical Integrator And Computer),* fue construida entre 1943 y 1945 por el Dr. John W. Mauchly y J. Presper Eckert.

La ENIAC, mil veces más veloz que sus predecesoras electromecánicas, irrumpió como un importante descubrimiento en la tecnología de la computación. Pesaba 30 toneladas y ocupaba un espacio de 450 mts cuadrados, llenaba un cuarto de 6 m x 12 m y contenía 18,000 bulbos, tenía que programarse manualmente conectándola a 3 tableros que contenían más de 6000 interruptores. Ingresar un nuevo programa era un proceso muy tedioso que requería días o incluso semanas. A diferencia de las computadoras actuales que operan con un sistema binario (0,1) la ENIAC operaba con uno decimal (0, 1, 2…9). Pero también tenía sus limitaciones: estaba construida totalmente por válvulas, consumía gran cantidad de energía y producía mucho calor. Esto hacía que las válvulas se quemaran con facilidad y que las casas del vecindario sufrieran cortes de luz.





El primer intento de sobreponerse a estos errores y limitaciones de velocidad fue de Howard Airen quien junto a ingenieros de IBM diseñó, en 1944, una calculadora automática denominada **Mark I**.



Luego se construyó **Mark II** sirviendo sólo de base para cuando se crearan las válvulas al vacío y comenzara la computación electrónica.

Luego de esta breve reseña histórica, le proponemos la lectura del siguiente texto que describe las distintas generaciones de computadoras.

Generación de Computadoras

Primera Generación

(1945 –1955)
Se llama así a la generación de **tubos al vacío y válvulas**.
Se caracterizó por máquinas muy grandes, pesadas y lentas en sus procesos, tanto que la resolución de programas largos implicaba varios días de espera. Pero igual fue muy útil pues resolvía 5000 cálculos por segundo.

Segunda Generación

(1955-1965)
Se llamaba la generación de los **transistores y sistemas en lote.**
En las computadoras de esta generación se reemplazaron las válvulas por los transistores. Con esto se pudo reducir el tamaño de los ordenadores y aumentar su velocidad de trabajo, aunque todavía eran lentas.

Tercera Generación

(1965-1980)
Se la denomina de **circuitos integrados y de multiprogramación**.
El gran descubrimiento de este período fueron los circuitos integrados denominados CHIP. El circuito integrado consiste en un gran número de componentes electrónicos (transistores, resistencias, etc.) miniaturizados y encapsulados en un espacio de pocos centímetros. Este descubrimiento produjo grandes cambios en cuanto al tamaño de las computadoras, en velocidad, en compatibilidad, introduciendo nuevas técnicas de programación.

Cuarta Generación

(1980-1990)
Se la denomina de la **computadora personal o computadora hogareña**. Se llama así ya que los microprocesadores son chips más pequeños que contienen en un centímetro cuadrado miles de transistores. Si las comparamos con la ENIAC, cuyas válvulas ocupaban toda una habitación, veremos cómo ahora todo se resume en un centímetro cuadrado. De esta forma muchas familias comenzaron a tener computadoras en sus hogares.

Quinta Generación

(1980 a la actualidad)
Con el advenimiento de los microprocesadores integrados en un chip, estos constantemente han ido aumentando su velocidad de operación (hoy cercana a los 4 GHz, así como en cantidad de otras funcionan integradas en el chip (memorias caché, controlador de memoria, y hoy día el procesador de video y el puente intercomunicador Northbridge), siendo que en el presente en la cabeza de un alfiler pueden integrarse 100 millones de transistores.
Dada la disipación de calor que aumenta con los GHz, fue necesario desarrollar los chips multicore, para ejecutar dos o mas programas simultáneamente, y así darle más velocidad a los distintos tipos de computadoras. Con un solo procesador (núcleo o "core") sólo se puede ejecutar un programa por vea.

La evolución de la informática afecta a todos los aspectos de la vida, teniendo muchísimas aplicaciones: en medicina, bancos, escuelas, oficinas, medios de transporte, todo pasa hoy por una computadora. Son la fuente de grandes entradas de datos reemplazando papeles, agilizando búsquedas y trámites, que años atrás eran dificultosos, proporcionando grandes adelantos en lo que respecta al bienestar del individuo.

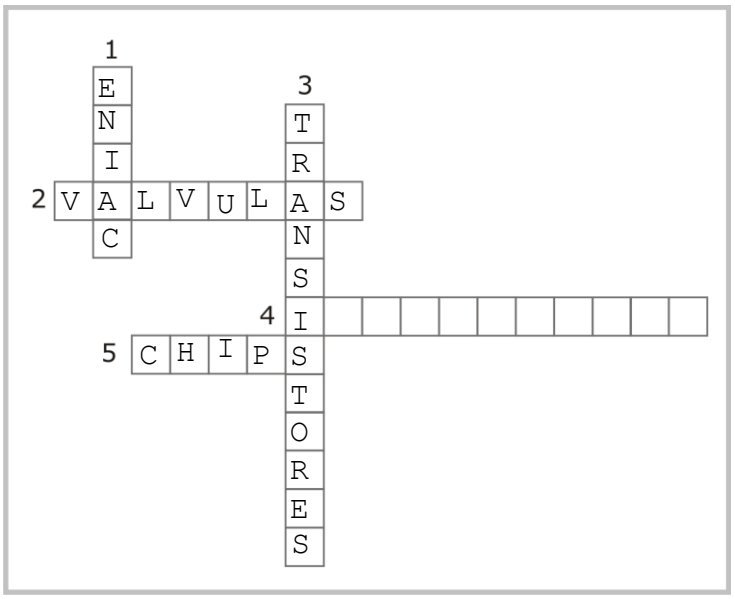
Guía de revisión de conceptos

Como guía de revisión del texto le sugerimos que procure responder las siguientes preguntas:

- ¿Cuántas etapas se consideran en la historia de la computación?
- ¿Cuáles son las características técnicas de cada etapa?
- ¿Cuáles son las características de cada etapa según el software?

Actividades para la facilitación de los aprendizajes

Antes de avanzar en el estudio de los siguientes temas, le proponemos que complete este crucigrama sobre la base de las referencias que seguidamente le presentamos.



Referencias:

1) ¿Cómo se llamó la primera computadora?
2) ¿Cuáles eran sus componentes?
3) ¿Por qué fueron reemplazadas?
4) ¿Cómo se denomina a las computadoras de la quinta generación?
5) ¿Qué reemplazó a los transistores?

2. ¿Qué es una computadora?

Una computadora es una máquina que procesa datos, transformándolos en información.

En 1981, IBM lanzó al mercado el IBM PC, el primer ordenador personal. La sigla PC que corresponde a *Personal Computer*, es decir, *Computadora Personal* se convirtió en un estándar informático.

La computadora ha podido evolucionar y lo sigue haciendo a un ritmo vertiginoso que parece no tener fin. Pocos hubiesen apostado, en sus inicios, que sería el modelo de ordenador que nos iba acompañar hasta el siglo XXI.

Actividades para la facilitación de los aprendizajes

Le proponemos detenerse por un momento y reflexionar acerca de los contenidos a cuyo estudio se ha abocado hasta ahora. La idea es que usted pueda confrontar lo que aquí se expone con su propia visión acerca de las temáticas abordadas y que pueda elaborar sus propias conclusiones. Por ello le proponemos, en primer lugar, leer y reflexionar acerca de la siguiente afirmación y luego responder los interrogantes que le planteamos teniendo en cuenta el uso que usted hace de la PC en su vida cotidiana y/o laboral.

La computadora es una herramienta que nos facilita la resolución de problemas y nos posibilita realizar una gran cantidad de veces una misma operación sin demostrar fatiga ni error. Es sólo una herramienta al servicio del hombre.

- ¿Coincide con la afirmación que acaba de leer? Fundamente su respuesta.
- ¿Cómo definiría usted a la computadora? ¿Cuáles considera que son sus principales usos?
- ¿Usted la utiliza con frecuencia?
- ¿Qué tipo de problemas considera que le permite solucionar?

2.1. Componentes de una computadora

Toda computadora posee un componente externo, llamado ***hardware,*** y otro interno, denominado ***software****.*

Definámoslos…

- El *hardware* es la parte material de la computadora: "la ferretería", que evolucionó a pasos agigantados a medida que avanzaban las nuevas tecnologías, desde los tubos al vacío y máquinas que ocupaban una habitación entera hasta los chips que se desarrollan hoy en día.

- El *software* es la parte inteligente de la computadora: los programas que sirven para el control y manejo del *hardware* como también aquellos que permiten resolver con mayor facilidad y eficiencia los problemas que le plantea el hombre.

Le proponemos enriquecer estas descripciones con la siguiente lectura.

¿Qué es *Hardware*?

Esta palabra de origen inglés quiere decir "material de ferretería" y es el nombre que se da a la parte física de la computadora, es decir todo componente externo, incluyendo los elementos que se encuentran dentro del gabinete, así también como aquellos visibles que constituyen un equipo informático.

La configuración de un equipo informático es el conjunto de todos los dispositivos conectados a una computadora en particular.

Esquema físico de una computadora

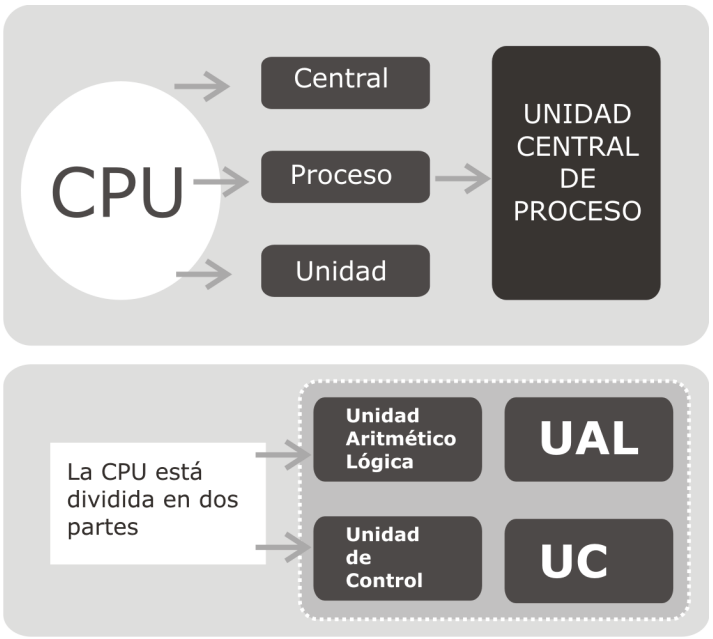
El *hardware* se compone por la unidad central y los dispositivos periféricos.

La **unidad central de proceso (UCP)** es el conjunto de circuitos que gobiernan el funcionamiento de la computadora en lo referente al pedido y ejecución de instrucciones, y es donde se hacen las operaciones sobre los datos que se quieren procesar.
Con el advenimiento de los chip integrados la UCP están integrados en un chip, denominado **microprocesador**. Ejemplo: Pentium 4, Opteron, Dual Core, etc. Los dispositivos periféricos se encargan de dar entrada/salida de datos, almacenarlos, o mostrar resultados.

Núcleo central de una Computadora

Todas las computadoras presentan un núcleo central formado por la UCP y la memoria principal.

La capacidad para pedir y ejecutar instrucciones se concentra en los circuitos del denominado **C.P.U.** (Central Process Unit) o **U P.C**. en castellano.



Unidad de control

U.C. (Control Unit). Esta área pide y ejecuta las instrucciones, ordena a la UAL que opere datos, a los registros que almacenan, y a la memoria que sea leída o escrita.

Unidad Aritmético-Lógica

La **U.A.L** (Aritmetic Logic Unit). Su función es realizar todas las operaciones aritméticas como suma, resta, potencias, multiplicación, etc. y las operaciones lógicas como negación, And, Or.
La denominación "Lógica" no implica ninguna inteligencia, la cual está en la UC, que concentra la inteligencia, sino que la UAL realiza operaciones lógicas.

Una tercer porción de la UCP son los **registros** para almacenar transitoriamente datos y resultados, instrucciones y direcciones de memoria.

MEMORIAS

Las memorias de las **computadoras** son circuitos que guardan **información** en forma momentánea o permanente.
Dentro de las computadoras hay dos tipos de **memorias**: una llamada **ROM** y la otra **RAM.**

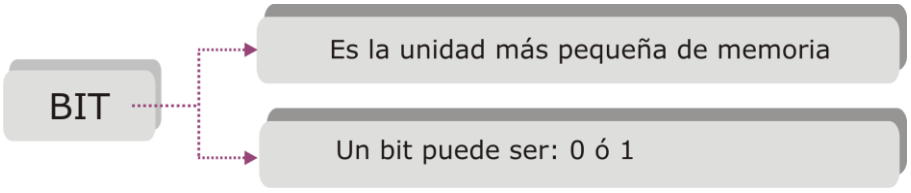
La memoria ROM

Red Only Memory, memoria que conserva la información aunque se corte la energía que la alimenta, y se vuelve a recuperar cuando vuelve la energía.
Ejemplos Rom Bios, pendrives, discos SSD.

La memoria RAM

Random Access Memory, memoria de acceso aleatorio (azar) es como un pizarrón, ya que escribimos todo lo que queremos y luego, al borrar, volvemos a encontrar nuestro pizarrón vacío para comenzar otra vez. Es decir que si estamos trabajando en nuestra **PC** y se corta la luz perdemos toda nuestra información hasta allí elaborada.

Pero ¿cuántos datos podemos cargar? Eso depende de la capacidad que la memoria **RAM** tenga, ya que hay distintos tipos.
Esta capacidad se mide y su unidad de medida es el **bit.**



La combinación de 8 bits forma un **byte** y cada byte representa una letra del abecedario, un número, un carácter especial, etc.

Por ejemplo:

Letras	Bytes
A	01000001
B	01000010
C	01000011
D	01000100
E	01000101

Entonces, podemos decir que la memoria de una computadora es la capacidad de guardar bytes.

Las computadoras actuales reciben millones de datos por eso para medir esta capacidad se utilizan los múltiplos del byte.

Tabla de equivalencias de capacidad

1 Bit	Dígito Binario	
1 Byte	8 Bits	
1 Kilobyte	1.024 Bytes	
1 Megabyte	1.024 Kilobytes	Aprox. 1.050.000 Bytes
1 Gigabyte	1.024 Megabytes	Aprox. 1.070.000.000 Bytes
1 Terabyte	1.024 Gigabytes	Aprox.1.100.000.000.000 Bytes

Completar el siguiente cuadro:

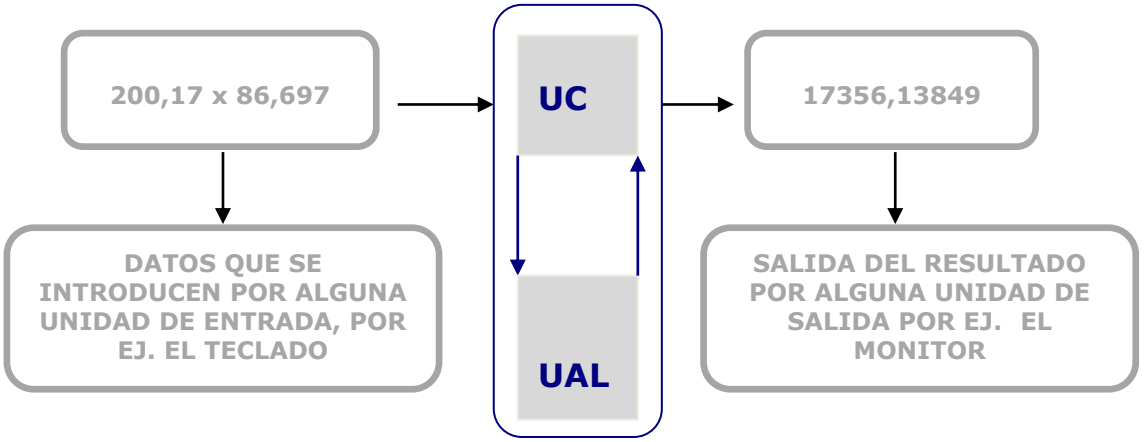
BYTES	KILOBYTE	MEGABYTE
1024		
2048		
	4	
		8
		16

Ejemplo de un procesamiento de datos

De lo dicho hasta aquí comprendemos que si queremos multiplicar 200,17 por 86,697 en la vida diaria nuestros pasos serían:

- **Observar** en la hoja de trabajo la tarea a resolver.
- **Comprender** la tarea, **resolverla** en hoja aparte, y **obtener** el resultado.
- **Volcar** el resultado a nuestra hoja de trabajo

Un computador lo resolvería de forma semejante



Unidades Periféricas (dispositivos)

Los periféricos son dispositivos mediante los cuales se realiza la entrada y salida de datos. Se distinguen tres tipos:

- **ENTRADA**
- **SALIDA**
- **ENTRADA / SALIDA**

Periféricos de entrada

Permiten ingresar información (datos y programas) a la computadora. Algunos de ellos son:

- **Teclado:** utilizado para ingresar datos, enviar órdenes, comandos o instrucciones a la CPU o poner en funcionamiento los programas.

- **Mouse:** es un dispositivo manual a bolilla u óptico que dispone de dos o tres botones, de múltiples aplicaciones y de bajo costo.

- **Scanner:** es un lector electrónico que posibilita la captura de imágenes, firmas, fotos, logotipos o textos que se visualizan en la pantalla; funcionan de manera similar a una fotocopiadora con distintos grados de definición.

- **Lápiz óptico (*light Pen*):** es un instrumento con forma de lápiz que controla el movimiento del cursor basado en la intensidad de la luz de la pantalla por medio de un sensor.

- **Touch screen:** Para entrar información usando la pantalla del monitor

- **Joystick:** controla el movimiento del cursor en la pantalla permitiendo ubicarse rápidamente en el lugar deseado.

- **Unidad lectora de CD-ROM:** dispositivo que permite la lectura de la información contenida en un disco óptico CD-ROM.

Periféricos de salida

Muestran los resultados de algún proceso de los datos ingresados a la computadora. Algunos de ellos son:

- **Monitor y tarjeta gráfica:** el monitor o pantalla de video se utiliza para visualizar datos, instrucciones o comandos, caracteres o gráficos dados por la computadora o entrados a través de algún periférico. A la visualización de caracteres en pantalla se la conoce con el nombre de **modo texto** y a la de gráficos, **modo gráfico**; en este último, cada imagen es descompuesta en puntos, cada uno de ellos se denomina **píxel**. La definición del monitor está dada por la cantidad de pixeles que hay en $1mm^2$, a mayor cantidad de pixeles mayor definición.

- **Impresora:** es un dispositivo necesario pues es el papel aún la forma en que se suele presentar la información. Existen en el mercado distintos tipos:
  - Matriciales: poseen una cabeza con agujas que golpean sobre una cinta entintada.
  - De papel termo sensible (más económicas y portátiles)
  - De chorro a tinta (poseen muy buena calidad de impresión)
  - Láser: tienen microprocesador y memoria propia, mas velocidad, mejor calidad de impresión (símil fotocopiadora)

- **Plotter:** es un periférico de salida que se conecta a una computadora para el trazado de diagramas, gráficos y planos. Está constituido por plumillas o rotuladores, encargados de realizar los trazos sobre el papel. Es aplicado, fundamentalmente, en el Diseño Asistido por Computadora.

Periféricos de Entrada / Salida

Son los que permiten guardar y leer información; algunos de ellos son:

- **Unidades de Discos:** son dispositivos que posibilitan el almacenamiento herma-nente de información aunque se apague el equipo. Ejemplo dispositivos de almacena-miento masivo (discos rígidos y los actuales SSD (Solid State Device, CD, DVDs).

- **Pendrive:** es una Flash Rom, constituida por transistores especiales.

- **Módem:** (modulador – demodulador) es un integrado que permite la intercomunicación entre computadoras a través de la línea telefónica. De esta forma los usuarios pueden intercambiar información en forma telemática.

Unidades de Almacenamiento masivo

Las unidades de almacenamiento que son los discos rígidos, los CDs, DVDs y pendrives, que nos permiten el acopio de información, la posibilidad de trasladar dicha información.

Disco rígido magnético

Es un dispositivo de almacenamiento de información, constituido por uno o más discos de material duro (aleación de aluminio) recubierto cada uno de ellos por una capa magnética, colocados unos sobre otros, unidos por un eje de rotación, cuyas caras pueden ser escritas/leídas por cabezas magnéticas de lectura-escritura, encargadas de leer y grabar la información.

Está generalmente instalado en el interior del gabinete de la computadora y funcionan en forma permanente a lo largo de una sesión de trabajo, aumentando de esta forma la rapidez de acceso y transferencia de datos. Siempre va acompañado por un componente circuital, denominado controlador, que es el encargado de regir toda operación que se lleve a cabo sobre la superficie del rígido.

Los beneficios más importantes son: gran velocidad de lectura y escritura, capacidad de almacenamiento y confiabilidad.

CD-ROM

- Es un disco compacto sólo de lectura (*Compact Disk Read Only Memory*), que constituye un soporte para el almacenamiento de los datos no modificable.
- Físicamente, es idéntico a un disco compacto de sonido de **policarbonato** (fibra plástica muy dura) de 120 mm de diámetro por 1,2 mm de grosor. Posee una sola pista en forma de espiral que produce una densidad estimativa de 16.000 **TPI** y tiene una capacidad que oscila entre 500 y 600 Mb (según el fabricante). Su capacidad puede almacenar el equivalente a 250.000 páginas.

SOFTWARE

¿Qué es *Software*?
Fuera de lo tangible y visible existen los programas que constituyen el *software*. (*soft*: blando e intangible).

Software es el conjunto de **órdenes y procedimientos** relacionados entre sí para poder ejecutar un programa dentro de la computadora.

***Hardware* y *Software* se hallan íntimamente ligados ya que no hay *software* que funcione sin hardware, ni *hardware* que funcione sin el *software* adecuado.**

Clasificación de los *Software*

Dentro del *software* existen distintos tipos, según las prestaciones que brinden. Haremos, a continuación, un breve detalle de algunos de los grupos más utilizados. La primera división se puede hacer entre aquellos denominados **de base** y los de aplicaciones; los primeros son programas que sirven de sostén para las **aplicaciones**, o sea, que los de base no nos brindan una prestación en particular. En cambio, las aplicaciones son todos aquellos programas que nos permiten obtener un resultado concreto (procesadores de texto, bases de datos, planillas de cálculo, agendas, etc.).

Un **sistema operativo** forma parte del software de base. Hacia el exterior facilita el manejo de un computador por parte de los usuarios. En relación con el interior gestiona los 4 recursos de un sistema:

1) Qué programa se va a ejecutar en caso de existir una UCP, o cuáles programas si se tiene un chip multicore (dual core, cuádruple core…).

2) La memoria principal
3) Los archivos
4) Los periféricos (mediante los programas "drivers")

WINDOWS

Desde sus comienzos el hombre buscó medios; métodos que le permitieran agilizar y simplificar las tareas que debía hacer.

En la historia de la informática vemos que en la década del '80 surge la primera computadora personal con un sistema operativo propio **D.O.S.**, compitiendo con **MAC** que poseía otro sistema operativo el **UNIX** y otro tipo de computadoras. La conjunción de estos hechos inició una verdadera revolución en el área.

Por su parte, algunos usuarios aprendieron con dificultad a usar algunos comandos del **D.O.S.** utilizando generalmente el modo texto, y también a manejar una variedad de programas de aplicación (procesadores de textos, planillas de Cálculo y bases de datos).

Cabe destacar que, en ese momento, no se disponía de métodos sencillos para el intercambio de información entre esos programas, ni había forma de hacerlo sin tener que cerrar aquel con el que se estaba trabajando.

A comienzos del '85 se desarrolla como un reto y desafío a los cambios, la interface gráfica **Windows**; en 1990 Microsoft presentó Windows 3.0 con el advenimiento del entorno Windows que revolucionó nuevamente el mundo de la informática desde la gran empresa hasta la escuela. Más adelante, en 1995, aparece **Windows '95** que se presenta como sistema operativo aunque aún algunos de nuestros equipos necesitan del tradicional **D.O.S.**, para funcionar correctamente.

Windows '95 presenta características especiales que le permiten acelerar su trabajo. Además del botón principal del mouse, se puede utilizar el botón secundario para acceder a información, o moverla más rápidamente y crear accesos directos a documentos, programas y otros elementos.

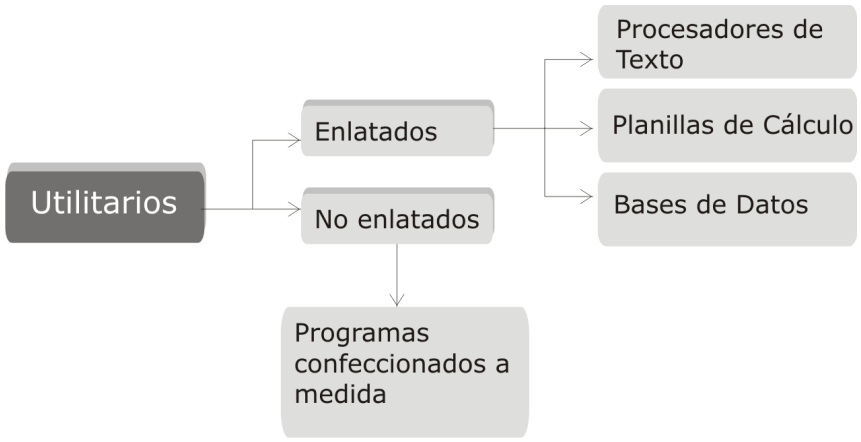
En los años siguientes aparecieron Windows NT, XP, 7 y 8, cada vez más orientados a agilizar las comunicaciones con Internet y funcionar en aplicaciones de computación móviles.

¿Qué es un utilitario?

Se define como **utilitario** a los **programas** que el usuario compra para trabajar o, sencillamente, para jugar. Algunos ejemplos que podemos mencionar son: Ms Word, PowerPoint, Exel, Corel Draw, software sobre crucigramas, etc.

UTILITARIO = PROGRAMA = SOFTWARE

Clasificación de utilitarios



Todos los utilitarios se van mejorando en sus ediciones. A estas mejoras se las denomina **versiones**.

ENLATADOS

Son programas desarrollados a medida, por grandes empresas (IBM, Microsoft, Borland, etc.) que se dedican al diseño de *software*.

Entre los más usados encontramos:

- Procesador de textos
- Planilla de cálculo
- Bases de datos
- Graficadores

Estos *software* **no tienen la posibilidad de modificarse** según nuestra necesidad. Es por eso que se los llama **ENLATADOS**. Es decir no se puede adaptar a nuestros pedidos. Por ejemplo realizar una liquidación de sueldos de nuestro personal con beneficios o bien un programa que permita controlar los gastos de nuestra casa.

Procesador de texto

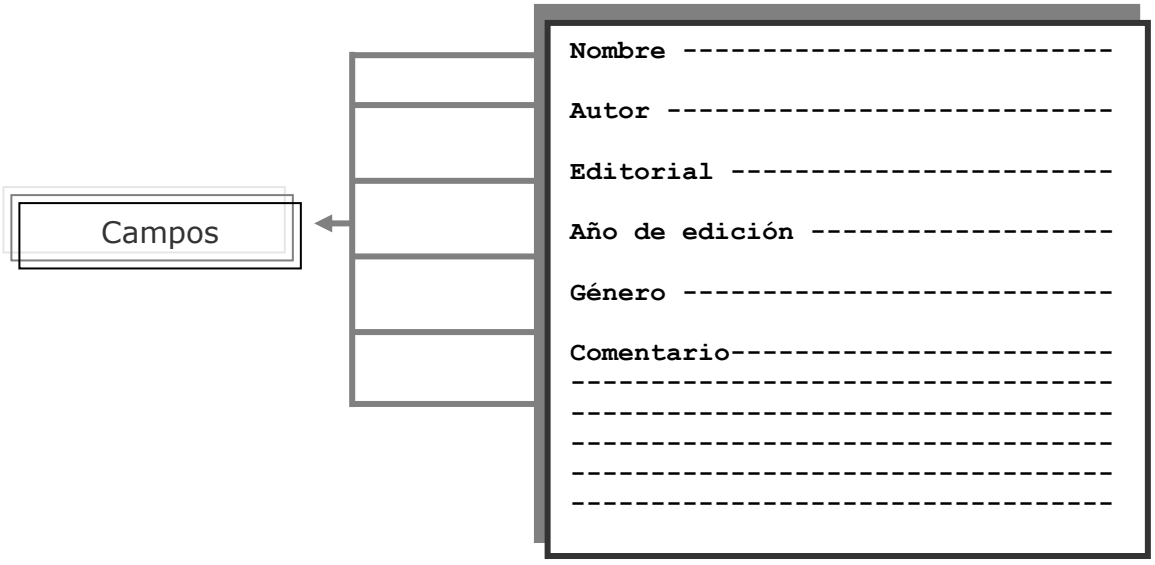
Es un programa creado para reemplazar el uso de la máquina de escribir con muchos beneficios, como corrección de errores, agregados de dibujos, selección de tipos de letra, etc.

Planilla de cálculo

Es un programa que se utiliza, generalmente, para registrar la contabilidad de una casa, empresa o negocio y así poder calcular los balances, sus sueldos, etc. También podemos realizar gráficos estadísticos.

Base de datos

En el presente que vivimos observamos que en todas las oficinas, empresas, etc. se maneja mucha información organizada en ficheros, que también se denominan archivos. Por ejemplo: cuando vamos a una visita médica automáticamente el profesional completa los datos de nuestros síntomas en una ficha que pasa a formar parte del archivo médico, y así sucede también en un club, biblioteca, etc.
Por ejemplo, ésta sería la ficha de una biblioteca:



Esta es una ficha con seis campos, es decir, es el diseño de un registro de seis campos: nombre, autor, editorial, año de edición, género y comentario.
Para desarrollar estas tareas en forma rápida y ordenada, se crearon los programas de **Base de Datos** que permiten manejar, guardar y organizar datos, permitiendo realizar todo tipo de consultas en cualquier momento.
En el mercado existen diferentes tipos de bases de datos, la más utilizada por usuarios particulares es Access que viene con el paquete de Office de Microsoft.

NO ENLATADOS

Son programas confeccionados a medida. Estos se elaboran a pedido de una **empresa** o establecimiento cuando los programas enlatados no satisfacen sus requerimientos. Estos programas son realizados por **analistas de sistemas** de acuerdo con las necesidades de la empresa que los contrató.

Estos programas se desarrollan durante largo tiempo ya que el programador debe primero comprender muy bien las necesidades del usuario para poder solucionar su problema. Para esto se utilizan diagramas y se realizan pruebas en la computadora hasta que se logra resolverlos.

Los programas no enlatados se escriben en la computadora y para esto el programador selecciona un lenguaje de programación, logrando así comunicarse con la computadora.

Programas de esparcimiento

Son los utilitarios creados para el tiempo libre como, por ejemplo, ajedrez, carreras de auto, simuladores de vuelo, etc.

¿Qué es un Archivo?

Podemos considerar al **"archivo"** como información almacenada en una unidad almacenamiento, recuperable por su nombre.

En función del tipo de información que éstos posean se pueden agrupar en: archivos de programa, de texto, de manejo de video, de manejo de impresora, etc. Los archivos se identifican por un nombre y una extensión separados por un punto. Por ejemplo, **CARTAS.TXT:** el nombre es CARTA y TXT es la extensión que identifica a los archivos de texto.

Las restricciones que tienen los nombres están dadas por la cantidad de letras o números (no más de 8) y no pueden tener puntos en el medio o nombres que coincidan con alguno de los comandos; en lo que respecta a la extensión como máximo puede tener tres letras o números.

Las extensiones suelen identificar los distintos tipos de archivos

***.COM**	archivos del comando operativo
***.EXE**	archivos ejecutables
***.SYS**	archivos del sistema operativo, que controlan los distin tos dispositivos
***.BAT**	archivos compuestos por comandos del S.O. que se ejecutarán sucesivamente
***.TXT**	archivos de textos
***.DOC**	archivos elaborados con procesador de texto, por ejemplo Word
***.BAS**	archivos del lenguaje basic
***.WPS**	archivos del procesador de textos Works
***.MDB**	archivos de la base de datos Access
***.LWR**	archivos del lenguaje logo
***.WRI**	archivos del procesador de textos Write
***.PPT**	archivos del Power Point
***.XLS**	archivos de la Planilla de cálculos Excel

Aclaraciones:

- Dos archivos pueden tener el mismo nombre pero no la misma extensión.
- Dos archivos pueden tener la misma extensión y distinto nombre.
- Dos archivos no pueden coincidir simultáneamente en el nombre y la extensión.

Guía de revisión de conceptos

Le recomendamo durante la relectura prestar especial atención a los siguientes puntos:

- Significado de la palabra "*hardware*".
- Distinción entre el esquema físico y la estructura interna de una computadora.
- Tipos de memorias de las computadoras: memoria ROM y memoria RAM.
- Características de los periféricos de entrada, los periféricos de salida y los periféricos de entrada/salida.
- Concepto de "*software*" y su clasificación: los denominados de base y los de aplicaciones.
- Requerimientos para la puesta en marcha del equipo.
- Concepto de utilitarios y su clasificación: enlatados y no enlatados.
- Concepto de archivo.

Cierre de la unidad

Para cerrar esta unidad, le proponemos que lea nuevamente los contenidos desarrollados aquí y realice la actividad que le planteamos a continuación.

Como futuro analista programador, usted confeccionará utilitarios no enlatados, es decir, programas a medida. En función de ello, imagine que un médico que posee un consultorio particular lo convoca para que le diseñe un programa que le permita administrar los turnos de sus pacientes y la facturación mensual, de manera operativa y eficiente.

¿Cuáles serían los pasos iniciales que usted, como futuro analista programador, seguiría para emprender este trabajo?

Presentación

A través de esta unidad usted podrá comenzar a introducirse en el mundo de la programación. Un programa, como veremos luego, es una secuencia lógica de instrucciones u órdenes que la computadora reconoce y que permiten llegar a la solución de un determinado problema. Un programa se nutre de datos que serán procesados y se transformarán en información para el usuario.

En esta segunda unidad abordaremos, fundamentalmente, el uso de los distintos componentes de la programación, sus metodologías y el procedimiento necesario para que un programa desarrollado por el hombre pueda ser interpretado y ejecutado por la computadora.

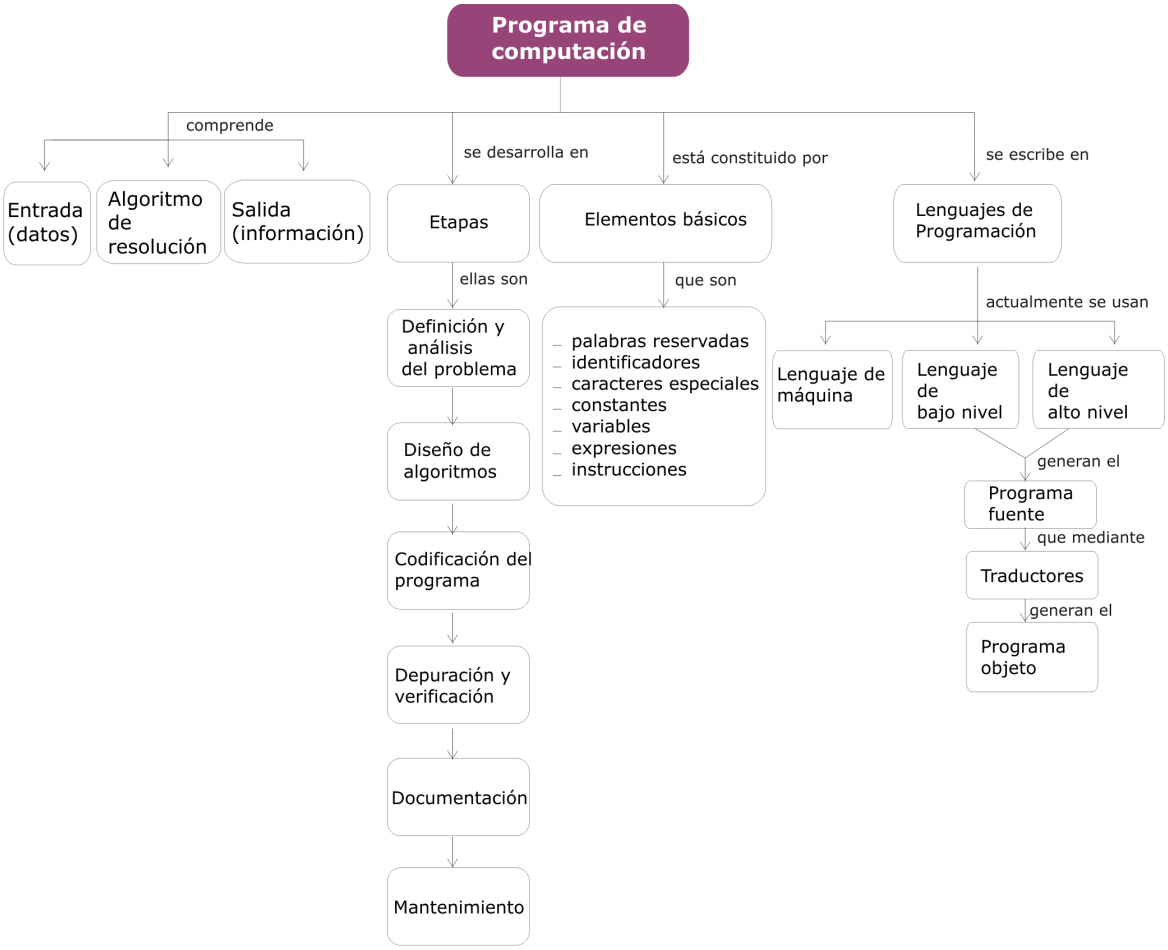
Esperamos que el estudio de los contenidos que integran esta unidad y la resolución de las actividades propuestas le permitan:

- Comprender las etapas que requiere el desarrollo de un programa de computación.
- Reconocer los elementos básicos constitutivos de un programa o algoritmo.
- Conocer los lenguajes de programación utilizados en la actualidad y las características de los traductores de lenguaje.

A continuación, le presentamos un detalle de los contenidos y actividades que integran esta unidad. Usted deberá ir avanzando en el estudio y profundización de los diferentes temas, realizando las lecturas requeridas y elaborando las actividades propuestas.

Organizador Gráfico de la Unidad 2

El siguiente esquema le permitirá visualizar la interrelación entre los conceptos que a continuación abordaremos.



3. El concepto de programa

Un programa de computadora es un conjunto de instrucciones u órdenes dadas a la máquina que producirán la ejecución de una determinada tarea. En esencia, un programa es un medio para conseguir un fin.

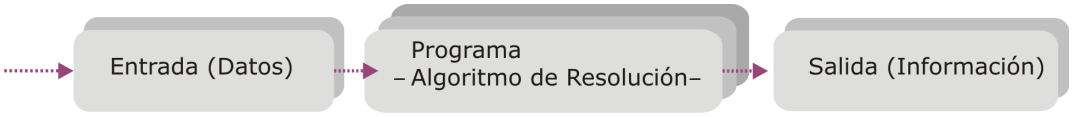
Tras la decisión de desarrollar un programa, el programador debe establecer el conjunto de especificaciones que aquél debe contener: **entrada, salida y algoritmos de resolución**; éstos incluirán las técnicas para obtener las salidas a partir de las entradas.

Conceptualmente, un programa puede ser considerado como una caja negra. La caja negra o el algoritmo de resolución es, en realidad, el conjunto de códigos que transforman las entradas del programa (**datos**) en salidas (**resultados**).

Veámoslo a través del siguiente ejemplo:

Entrada: se ingresan sueldos
Algoritmo de resolución: se los suma y cuenta
Salida: se devuelve el promedio de sueldos

Esquemáticamente, se representaría de la siguiente forma:



El programador debe establecer de dónde provienen las entradas al programa. Estas, en cualquier caso, procederán de un dispositivo de entrada -teclado, disco, etc.- El proceso de introducir la información de entrada -datos- en la memoria de la computadora se denomina **entrada de datos, operación de lectura o acción de leer**.

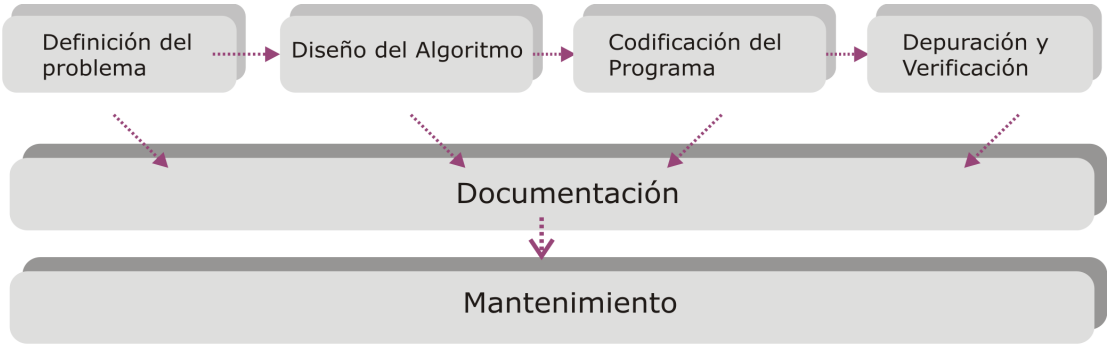
Las salidas de datos se deben presentar en dispositivos periféricos de salida: pantalla, impresoras, discos, etc. La **operación de salida de datos** se conoce también como **escritura o acción de escribir**.

3.1. Etapas del proceso de programación

Hemos señalado que el desarrollo de un programa requiere de las siguientes etapas:

- Definición y análisis del problema
- Diseño de algoritmos (ya sea bajo una metodología de diagramas de flujo, pseudocódigo, etc.)
- Codificación del programa
- Depuración y verificación
- Documentación
- Mantenimiento

Gráficamente, podrían representarse de la siguiente forma:



Analizaremos cada etapa del proceso.

Guía de revisión de conceptos

Cuando lleve a cabo la lectura del texto que a continuación le presentamos concéntrese especialmente en los siguientes puntos:

- ¿Qué se entiende por definición y análisis de un problema? ¿Qué importancia tiene esta etapa en el desarrollo de un programa?
- ¿Qué es un algoritmo? ¿Cuáles son sus principales características?
- ¿Qué tipos de datos existen?
- ¿Qué métodos se emplean para representar un algoritmo?
- ¿Cómo se clasifican las instrucciones básicas?
- ¿Cómo se pasa del programa fuente al programa objeto?
- ¿Cómo deben ser los datos de una prueba de escritorio y para qué se utilizan?
- ¿Cuándo comienza y termina el mantenimiento del programa?

Etapas del proceso de programación

Primera etapa: Definición y análisis del problema

En la etapa de la definición y análisis de un problema es donde debemos tener más cuidado porque de allí en adelante, si realizamos de manera incorrecta el análisis, absolutamente todo nos saldrá mal. Básicamente es la etapa en que se deben comprender todos los puntos críticos de un problema a solucionar.

La principal razón para que las personas aprendan a programar en general y a utilizar los lenguajes de programación en particular es usar la computadora como una herramienta para la resolución de problemas.

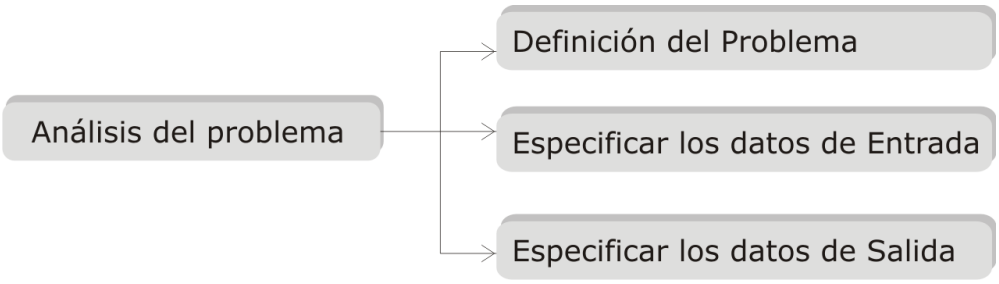
La resolución de problemas con computadoras se puede dividir en tres fases:

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

El análisis del problema requiere que el problema sea definido y comprendido claramente para que pueda ser analizado con todo detalle. Una vez analizado el problema, se debe desarrollar el algoritmo -procedimiento paso a paso (secuencialidad) para solucionar el problema determinado-. Por último, para resolver el algoritmo mediante una computadora se necesita codificarlo en un lenguaje de programación Pascal, C, C++, COBOL, FORTRAN, etc., es decir, convertir el algoritmo en programa, ejecutarlo y comprobar que el programa soluciona verdaderamente el problema.

Análisis del problema

El análisis de un problema es ayudar al programador para llegar a una cierta comprensión de su naturaleza. El problema debe estar bien definido si se desea llegar a una solución satisfactoria.
Para poder definir con precisión el problema se requiere que las especificaciones de entrada y salida sean descritas con detalle. Una buena definición del problema, junto con una descripción detallada de las especificaciones de entrada y salida, son los requisitos más importantes para llegar a una solución eficaz.



Segunda etapa: Diseño de algoritmos

Concepto de algoritmo

Un programador de computadora es antes que nada una persona que resuelve problemas, por lo cual para llegar a ser un programador eficaz se necesita aprender a resolver problemas de un modo riguroso y sistemático. Se denomina **metodología de la programación** a la m*etodología necesaria para resolver problemas mediante un programa.* El punto central de esta metodología es el concepto de algoritmo.

La resolución de un problema exige el diseño de un algoritmo que resuelva el problema propuesto.

*Básicamente, un **algoritmo** es una secuencia de pasos lógicos para solucionar un problema determinado.*

Los pasos para la resolución de un problema son:

1. **Diseño del algoritmo** que describe la secuencia ordenada de pasos -***sin ambigüedades***- que conducen a la solución de un problema dado. *(Análisis del problema y desarrollo del algoritmo)*

2. Expresar el algoritmo como un *programa* en un lenguaje de programación adecuado. *(Etapa de Codificación)*

3. *Ejecución y validación* del programa por la computadora.

Para llegar a la realización de un programa es necesario el diseño previo de un algoritmo, de modo que sin algoritmo no puede existir un programa.

Los algoritmos son independientes tanto del lenguaje de programación en que se expresan como de la computadora que los ejecuta. En cada problema el algoritmo se puede expresar en un lenguaje diferente de programación y ejecutarse en una computadora distinta; sin embargo, el algoritmo será siempre el mismo. Así, por ejemplo, en una analogía con la vida diaria, una receta de un plato de cocina se puede expresar en español, inglés o francés, pero cualquiera sea el lenguaje, los pasos para la elaboración del plato se realizarán sin importar el idioma del cocinero.

En la ciencia de la computación y en la programación, los algoritmos son más importantes que los lenguajes de programación o las computadoras. Un lenguaje de programación es tan sólo un medio para expresar un algoritmo y una computadora es sólo un procesador para ejecutarlo. Tanto el lenguaje de programación, como la computadora son los medios para obtener un fin: *conseguir que el algoritmo se ejecute y se efectúe el proceso correspondiente.*

El diseño de la mayoría de los algoritmos requiere creatividad y lógica necesaria. Entonces, la *solución de un problema se puede expresar mediante un algoritmo.*

Características de los algoritmos:

Las características fundamentales que debe cumplir todo algoritmo son:

- Un algoritmo debe ser ***preciso*** e indicar el orden de realización de cada paso.
- Un algoritmo debe estar ***definido***. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.
- Un algoritmo debe ser ***finito***. Si se sigue un algoritmo, se debe terminar en algún momento; o sea, debe tener un número finito de pasos.

La definición de un algoritmo debe describir tres partes:

Entrada ⇨ **Proceso** ⇨ **Salida**

Ejemplos

- Un cliente solicita un pedido a una fábrica. La fábrica examina en su banco de datos la ficha del cliente, si el cliente es solvente, la empresa acepta el pedido; en caso contrario, rechazará el pedido.

Los pasos del algoritmo son:

1. Inicio.
2. Leer el pedido.
3. Examinar la ficha del cliente.
4. Si (If) el cliente es solvente, aceptar pedido;
5. En caso contrario (else), rechaza el pedido.
6. Fin.

Tercera etapa: Codificación del programa

Es el momento en el que el programador transforma o escribe su lógica en un lenguaje de programación, generando el programa fuente. Para ello utilizará los siguientes componentes:

Datos y sus tipos

El primer objetivo de toda computadora es el manejo de la información o datos. Un *dato* es la expresión general que describe los objetos con los cuales opera una computadora.

La mayoría de las computadoras pueden trabajar con varios tipos (modos) de datos. Los algoritmos y los programas correspondientes operan sobre datos.

La acción de las instrucciones ejecutables de las computadoras es reflejada en cambios en los valores de las partidas de datos. Los datos de entrada se transforman por el programa, después de las etapas intermedias, en datos de salida.

En el proceso de solución de problemas el diseño de la estructura de datos es tan importante como el diseño del algoritmo y del programa que se basa en el mismo.

Existen dos clases de tipos de datos: simples (sin estructura) y compuestos (estructurados). Los distintos tipos de datos se representan en diferentes formas en la computadora. A nivel de la máquina, un dato es un conjunto o secuencia de bits (dígitos 0 o l). Los lenguajes de alto nivel permiten basarse en abstracciones e ignorar los detalles de la representación interna. Aparece el concepto de tipo de datos, así como su representación.

Los tipos de datos simples son los siguientes:

- **Numéricos** *(integer, reall)*
- **Lógicos** *(boolean)*
- **Carácter** *(char, string)*

Existen algunos lenguajes de programación -FORTRAN esencialmente- que admiten otros tipos de datos; complejos, que permiten tratar los números **complejos**, y otros lenguajes –Pascal que también permiten declarar y definir sus propios tipos de datos.

Datos numéricos

El tipo numérico es el conjunto de los valores numéricos. Éstos pueden representarse en dos formas distintas:

- **Tipo numérico entero** (*integer*)
- **Tipo numérico real** (*real*)

***Enteros**:* el tipo entero es un subconjunto finito de los números enteros. Los enteros son números completos, no tienen componentes fraccionarios o decimales y pueden ser negativos o positivos.

Ejemplos de números enteros son:

6, 1, 4, -17, 5, -4

*Los enteros se denominan en ocasiones números de punto o coma fija. Los números enteros máximos y mínimos en una computadora suelen ser considerados en un rango**:*** -32768 a +32767

Datos lógicos (booleanos)

El tipo *lógico* -también denominado *booleano*- es aquel dato que sólo puede tomar uno de dos valores:

- **verdadero** (*true*)
- **falso** (*false*)

Este tipo de datos se utiliza para representar las alternativas (sí/no) a determinadas condiciones. Por ejemplo, cuando se pide si un valor entero es par, la respuesta será verdadera o falsa, según sea par o impar.

Datos tipo carácter y tipo cadena

El tipo *carácter* es el conjunto finito y ordenado de caracteres que la computadora reconoce. Un dato tipo carácter contiene un solo carácter.

Los caracteres que reconocen las diferentes computadoras no son estándar; sin embargo, la mayoría reconoce los siguientes caracteres alfabéticos y numéricos:

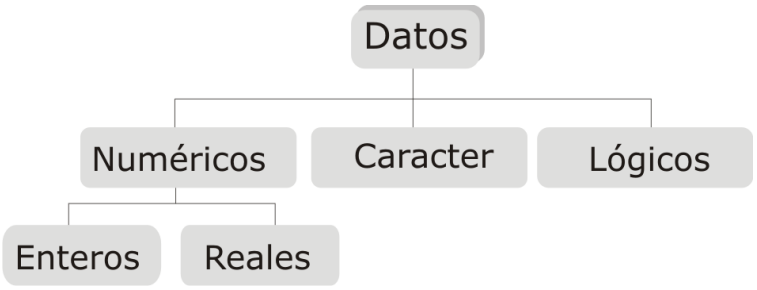
- **caracteres alfabéticos** (A, B, C,..... Z) (a, b, c, ...z),
- **caracteres numéricos** (1 , 2 ..... 9 , 0),
- **caracteres especiales** (+ , - ..... /\ 1 . 1 ; 1 < 1 >)

Una cadena *(string)* de caracteres es una sucesión de caracteres que se encuentran delimitados por una comilla o dobles comillas, según el tipo de lenguaje de programación. La *longitud* de una cadena de caracteres es el número de ellos comprendidos entre los separadores o limitadores. Algunos lenguajes tienen datos tipo *cadena.*

'Buen día'
'Hoy voy al cine'
'15 de octubre de 1960'

Resumen:

Los tipos de datos primitivos se clasifican en:



Constantes y variables

Los programas de computadora contienen ciertos valores que no deben cambiar durante la ejecución de] programa. Tales valores se llaman *constantes.* De igual forma, existen otros valores que cambiarán durante la ejecución del programa; a estos valores se les llama *variables.*

Una constante es una partida de datos (objetos) que permanecen sin cambios durante todo el desarrollo del algoritmo o durante la ejecución del programa.

Si se desea incluir el apóstrofo en la cadena, entonces, debe aparecer como un par de apóstrofos, encerrados dentro de simples comillas.

Variables

Una *variable* es un objeto o partida de datos cuyo valor puede cambiar durante el desarrollo del algoritmo o ejecución del programa. Dependiendo del lenguaje, hay diferentes tipos de variables, tales como *enteras, reales, carácter, lógicas y de cadena.*

Una variable que es de un cierto tipo puede tomar únicamente valores de ese tipo. Una variable de carácter, por ejemplo, puede tomar como valor sólo caracteres, mientras que una variable entera puede tomar sólo valores enteros.

Si se intenta asignar un valor de un tipo a una variable de otro tipo se producirá *un error de tipo.*

Una variable se identifica por los siguientes atributos: *nombre* que lo asigna y *tipo* que describe el uso de la variable.

Los nombres de las variables, a veces conocidos como *identificadores,* suelen constar de varios caracteres alfanuméricos, de los cuales el primero normalmente es una letra. No se deben utilizar, aunque lo permita el lenguaje, como nombres de identificadores palabras reservadas del lenguaje de programación.

Nombres válidos de variables son:

NOMBRES

NOTAS

A_CODIGO

A1002

Veamos un ejemplo:

- Realizar la suma de todos los números pares entre el 2 y 100

El problema consiste en sumar 2 + 4 + 6 + 8 +.... + 100.

Utilizaremos las palabras SUM y NUM para representar las sumas sucesivas:

(2 + 4), (2 + 4 + 6), (2 + 4 + 6 + 8), etc.

1. Inicio.
2. Establecer SUM en 0.
3. Establecer NUM en 2.

4. Sumar NUM a SUM. El resultado será el nuevo valor de la suma (SUM).
5. Incrementar el NUM en 2.
6. Si (If) NUM =< 100 volver al paso 4; en caso contrario (else), escribir el último valor de SUM t terminar el proceso.
7. Fin.

Tipos de instrucciones

Las instrucciones disponibles en un lenguaje de programación dependen del tipo de lenguaje. Las instrucciones -acciones- básicas se pueden implementar de modo general en un algoritmo y esencialmente soportan todos los lenguajes. Dicho de otro modo, las instrucciones básicas son independientes del lenguaje.

La clasificación más usual, desde el punto de vista anterior, es:

1. instrucciones de inicio/fin
2. instrucciones de asignación
3. instrucciones de lectura
4. instrucciones de escritura
5. instrucciones de bifurcación

Instrucciones de asignación

a) A ⟵ 25 ***la variable A toma el valor de 25***.

b) Sum ⟵ 8 + A + 3 ***la variable Sum toma el valor 36.***

Tabla de Instrucciones/acciones básicas

Tipo de instrucción	Pseudocódigo (inglés)	Pseudocódigo (español)
comienzo de proceso	**begin**	**Inicio**
fin de proceso	**end**	**Fin**
entrada (lectura)	**read**	**Leer**
salida (escritura)	**write**	**Escribir**
Asignación	**A ⟵ 8**	**B ⟵ 19**

Instrucciones de lectura de datos (entrada)

Esta instrucción lee datos de un dispositivo de entrada.

leer (Cantidad, Horas, Porcentaje)

Leer del terminal los valores Cantidad, Horas y Porcentaje, archivándolos en la memoria; si los tres números se teclean en respuesta a la instrucción son 1000, 60, 20 significaría que se han asignado a las variables esos valores y equivaldría a la ejecución de las instrucciones.

A ← 1000
B ← 60
C ← 20

Instrucciones de escritura de resultados (salida)

Estas instrucciones se escriben en un dispositivo de salida.
Explicar el resultado de la ejecución de las siguientes instrucciones:

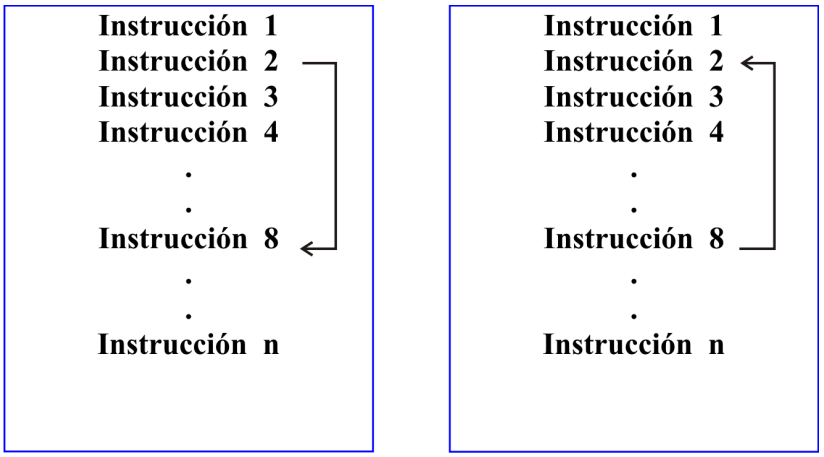
Cantidad ← 1000
Horas ← 60
Porcentaje ← 20

escribir (A, B, C)

Se visualizarían en la pantalla o imprimirían en el papel los valores 1000, 60 y 20 que contienen las variables A, B y C.

Instrucciones de bifurcación

El desarrollo lineal de un programa se interrumpe cuando se ejecuta una bifurcación. Las bifurcaciones pueden ser, según el punto del programa a donde se bifurca, hacia *adelante* o hacia a*trás.*



Las bifurcaciones en el flujo de un programa pueden realizarse de un **modo incondicional o condicional**.

Bifurcación incondicional: la bifurcación se realiza siempre que el flujo del programa pase por la instrucción sin necesidad del cumplimiento de ninguna condición.

Bifurcación condicional: la bifurcación depende del cumplimiento de una determinada condición. Si se cumple la condición, el flujo sigue ejecutando la acción F2. Si no se cumple, se ejecuta la acción F1.

Cuarta etapa: Depuración y verificación

En esta etapa el programador aplica un traductor al programa fuente para la generación del programa objeto, que será el que la máquina entienda. En esta acción "saltarán" los errores de sintaxis producidos. Luego, se generarán pruebas de escritorio con datos lo más cercanos a la realidad y se verificará el correcto funcionamiento del programa. Esta operación de depuración y verificación se realizará tantas veces como sea necesario hasta conseguir la optimización del programa.

Quinta etapa: Documentación

A lo largo de todas las etapas del desarrollo de un programa se deben documentar las distintas actividades realizadas con el fin de que cualquier miembro del equipo de programadores, en el momento o en el futuro, tenga la capacidad de continuar su desarrollo. Esta etapa es la más tediosa y, en general, en todos los equipos cuesta que sea cumplida.

Sexta etapa: Mantenimiento

Es el proceso por el cual se deben prever y solucionar todos los problemas de *hardware* y *software*. Debe ser sistemático y preventivo. Es el arte de hacer que los sistemas funcionen.

4. Elementos básicos constitutivos de un programa

En programación se debe establecer la diferencia entre el diseño del algoritmo y su implementación en un lenguaje específico. Por ello, se debe distinguir claramente entre los conceptos de programación y su implementación a través de un lenguaje determinado.

Una vez que comprendemos los conceptos de programación y cómo utilizarlos, el aprendizaje de un nuevo lenguaje es relativamente fácil.

Los lenguajes de programación, al igual que otros lenguajes, tienen **elementos básicos** que se utilizan como bloques constructivos, así como reglas a través de las cuales esos elementos se combinan. Estas reglas se denominan *sintaxis del lenguaje*. Solamente las instrucciones sintácticamente correctas pueden ser interpretadas por la computadora y los programas que contengan errores de sintaxis son rechazados por la máquina.

Los elementos básicos constitutivos de un programa o algoritmo son:

- palabras reservadas **(inicio, f in**, **si-entonces..** etc.),
- identificadores (nombres de variables esencialmente),
- caracteres especiales (coma, apóstrofe, etc.),
- constantes,
- variables,
- expresiones,
- instrucciones.

Contadores

Los procesos repetitivos son la base del uso de las computadoras. En estos procesos se necesitan normalmente contar los sucesos o acciones internas del bucle, como pueden ser los elementos de un fichero, el número de iteraciones a realizar por el bucle, etc. Una forma de controlar un bucle es mediante un contador.

Un contador es una variable cuyo valor se incrementa o decrece en una cantidad constante en cada iteración.

El contador puede ser positivo (incrementos, uno en uno) o negativo (decrementos, uno en uno).

CONT ⟵ CONT + 1

Acumuladores

Un acumulador o totalizador es una variable cuya misión es almacenar cantidades variables resultantes de sumas sucesivas. Realiza la misma función que un contador, con la diferencia de que el incremento o decremento de cada suma es variable en lugar de constante, como en el caso del contador.

ACUM ⟵ ACUM+ VARIABLE

Además de estos elementos básicos, existen otros elementos que forman parte de los programas, cuya comprensión y funcionamiento será vital para el correcto diseño de un algoritmo y naturalmente la codificación del programa.

Estos elementos son:

- ***bucles,***
- ***contadores,***
- ***acumuladores,***
- ***interruptores,***
- ***estructuras:***
  - ***secuenciales,***
  - ***selectivas,***
  - ***repetitivas.***

Bucles

Un bucle (loop), es un segmento de un algoritmo o programa, cuyas instrucciones se repiten un número determinado de veces mientras se cumple una determinada *condición* (existe o es verdadera la condición). Se debe establecer un mecanismo para determinar las tareas repetitivas. Este mecanismo es una condición que puede ser verdadera o falsa y que se comprueba una vez a cada *paso o iteración* del bucle (total de instrucciones que se repiten en el bucle).

Un bucle consta de tres partes:

- decisión,
- cuerpo del bucle,
- salida del bucle.

Ejemplo:

```
                    ←
Inicio.
        Suma        0
                1: Leer (N)
        Si  N = 0, entonces
                    Escribir (Suma)
                    Ir_ a fin
        Si_no
                              ←
                    Suma          Suma + N
        Fin_si
                Ir_a 1
Fin.
```

5. Los lenguajes de programación

Para que un procesador realice un proceso se le debe suministrar, en primer lugar, un algoritmo adecuado. El procesador debe ser capaz de **interpretar** el algoritmo, es decir, comprender las instrucciones de cada paso y realizar las operaciones correspondientes.

Cuando el procesador es una computadora, el algoritmo se ha de expresar en un formato que se denomina **programa**. Un programa se escribe en un **lenguaje de programación** y las operaciones que conducen a expresar un algoritmo en forma de programa se denominan **instrucciones**. Entonces, los lenguajes utilizados para escribir programas para computadoras son los lenguajes de programación y **programadores** son los escritores y diseñadores de programas en un determinado lenguaje.

Los diferentes pasos (acciones) de un algoritmo se expresan en los programas como **instrucciones, sentencias o proposiciones**. El término instrucción se suele referir a los lenguajes máquina y bajo nivel, reservando la sentencia o proposición para los lenguajes de alto nivel.

Por consiguiente, un programa consta de una secuencia de instrucciones, cada una de las cuales especifica ciertas operaciones que debe ejecutar la computadora. Las instrucciones básicas y comunes a casi todos los lenguajes de programación se pueden clasificar en cuatro grupos:

Instrucciones de entrada/salida	Instrucciones de transferencia de información y datos entre dispositivos periféricos (teclado, impresora, unidad de disco, etc.) y la memoria central.
Instrucciones aritmético-lógicas	Instrucciones que ejecutan operaciones aritméticas (suma, resta, multiplicación, división, potenciación), lógicas (operaciones *and, or, not,* etc.).
Instrucciones selectivas	Son aquellas que permiten la selección de tareas alternativas en función de los resultados de diferentes expresiones condicionales.
Instrucciones repetitivas	Son aquellas que admiten la repetición de secuencias de instrucciones un número determinado o indeterminado de veces.

En la actualidad, se utilizan distintos tipos de lenguajes de programación:

- Lenguaje de máquina
- Lenguaje de bajo nivel (ensamblador)
- Lenguajes de alto nivel

A continuación, encontrará la descripción de cada uno de los lenguajes mencionados.

Lenguajes máquina

Los **lenguajes de máquina** son aquellos que están escritos en lenguajes directamente inteligibles por la máquina (computadora), ya que sus instrucciones son *cadenas binarias* (cadenas o series de caracteres -dígitos- 0 y 1) que especifican una operación, y las posiciones (dirección) de memoria implicadas en la operación se denominan *instrucciones de máquina o código máquina.* El código máquina es el conocido código binario.

Las instrucciones en lenguaje de máquina dependen del *hardware* de la computadora y, por tanto, diferirán de una computadora a otra.

Las ventajas de programar en lenguaje de máquina son las posibilidades de cargar (transferir un programa a la memoria) sin necesidad de traducción posterior, lo que supone una velocidad de ejecución superior a cualquier otro lenguaje de programación.

Los inconvenientes -en la actualidad- superan a las ventajas, lo que hace prácticamente no recomendables a los lenguajes de máquina.

Estos inconvenientes son:

1. Dificultad y lentitud en la codificación
2. Poca fiabilidad
3. Gran dificultad de verificar y poner a punto los programas
4. Los programas sólo son ejecutables en el mismo procesador (UCP, *Unidad Central de Pro*ceso)

Para evitar los lenguajes máquina, desde el punto de vista del usuario, se han creado otros lenguajes que permiten escribir programas con instrucciones similares al lenguaje humano. Estos lenguajes son los de alto nivel y bajo nivel.

Lenguajes de bajo nivel

Los **lenguajes de bajo nivel** son más fáciles de utilizar que los lenguajes máquina pero, al igual que ellos, dependen de la máquina en particular. El lenguaje de bajo nivel por excelencia es el *ensamblador.* Las instrucciones en lenguaje ensamblador son instrucciones conocidas *como* ***nemotécnicos***. Por ejemplo, nemotécnicos típicos de operaciones aritméticas son: SUM (ADD), RES (SUB), DIV (DIV), etc.

Una instrucción típica de suma sería:

```
ADD   P, T, A
```

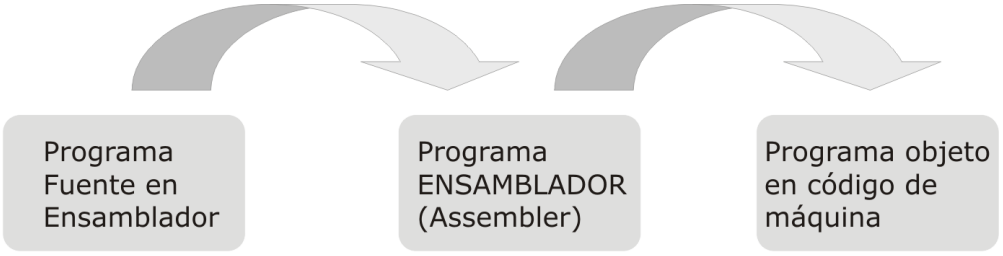
Un programa escrito en lenguaje ensamblador no puede ser ejecutado directamente por la computadora -en esto se diferencia esencialmente del lenguaje máquina-, sino que requiere una fase de *traducción* al lenguaje de máquina.

El programa original escrito en lenguaje ensamblador se denomina *programa fuente* y el programa traducido en lenguaje de máquina se conoce como programa objeto, directamente inteligible por la computadora.

El traductor de programas fuente a objeto es un programa llamado *ensamblador,* existente en casi todas las computadoras.

No se debe confundir -aunque en español adoptan el mismo nombre- el programa ensamblador, encargado de efectuar la traducción del programa fuente escrito a lenguaje máquina, con el lenguaje ensamblador, lenguaje de programación con una estructura y gramática definidas.

Los lenguajes ensambladores presentan la *ventaja* frente a los lenguajes de máquina de su mayor facilidad de codificación y, en general, su velocidad de cálculo.



Los inconvenientes más notables de los lenguajes ensambladores son:

Hoy día los lenguajes ensambladores tienen sus aplicaciones muy reducidas en la programación y se centran en aplicaciones de tiempo real, control de procesos y de dispositivos electrónicos, etc.

Lenguajes de alto nivel

Los lenguajes de alto nivel son los más utilizados por los programadores. Están diseñados para que las personas escriban y entiendan los programas de un modo mucho más fácil que los lenguajes máquina y ensambladores. Otra razón es que un programa escrito en un lenguaje de alto nivel ***es independiente de la máquina***; esto es, las instrucciones del programa de la computadora no dependen del diseño del *hardware* o de una computadora en particular.

En consecuencia, los programas escritos en lenguajes de alto nivel son *transportables,* lo que significa la posibilidad de poder ser ejecutados con poca o ninguna modificación en diferentes tipos de computadoras; al contrario que los programas en lenguaje máquina o ensamblador, que sólo se pueden ejecutar en un determinado tipo de computadora.

Los lenguajes de alto nivel presentan las siguientes ventajas:

1. El tiempo de formación de los programadores es relativamente corto comparado con otros lenguajes.
2. La escritura de programas se basa en reglas sintácticas similares a los lenguajes humanos.
3. Se utilizan nombres en las instrucciones, tales como *read, write, print, open,* etc.
4. Las modificaciones y puestas a punto de los programas son más fáciles.
5. El costo de los programas se reduce.
6. Son transportables.

Los inconvenientes son:

1. Incremento del tiempo de puesta a punto, al necesitarse diferentes traducciones del programa fuente para conseguir el programa definitivo.
2. No se aprovechan los recursos internos de la máquina, que se explotan mucho mejor en lenguajes de máquina y ensambladores.
3. Necesidad de una mayor capacidad de memoria.
4. El tiempo de ejecución de los programas es mucho mayor.

Al igual que sucede con los lenguajes ensambladores, los programas fuente tienen que ser traducidos por programas traductores, llamados en este caso **compiladores e intérpretes*.***

Algunos de los lenguajes de programación de alto nivel existentes hoy en día son:

C C++ COBOL Pascal Visual BASIC

Antes de continuar con el próximo punto, le proponemos que realice la siguiente actividad.

Actividades para la facilitación de los aprendizajes

Con el fin de que usted pueda comenzar a vincular estos contenidos con lo que será su futura práctica laboral, le pedimos que se conecte, en la medida de sus posibilidades, con alguna persona que desarrolle tareas de programación y le formule preguntas cuyas respuestas usted considera que pueden ampliar y enriquecer los temas estudiados hasta aquí.

Por ejemplo, usted podría preguntar:

1. ¿En qué empresa trabaja?
2. ¿Qué tipo de programas diseña?
3. ¿Cuáles son las etapas que realiza durante el desarrollo de un programa?
4. ¿Qué lenguajes de programación utiliza?

Una vez obtenidas las respuestas organice la información y compárela con lo que hemos estudiado en esta unidad, de modo de comenzar a visualizar cómo puede plasmarse lo conceptualizado hasta aquí en la práctica profesional.

5.1. Traductores del lenguaje

Los traductores de lenguaje son programas que traducen los programas fuente, escritos en lenguajes de alto nivel, a código máquina.

Los traductores se dividen en:

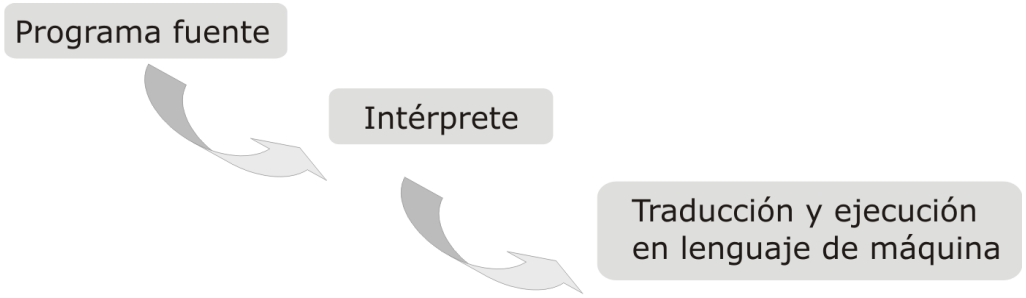
- Intérpretes
- Compiladores

A continuación, le brindaremos las definiciones correspondientes a cada uno de estos conceptos.

Intérpretes

Un intérprete es un traductor que toma un programa fuente, lo traduce y a continuación lo ejecuta. Los programas intérpretes clásicos como BASIC, prácticamente ya no se utilizan.

Sin embargo, está muy extendida la versión interpretada del lenguaje Smalltalk, un lenguaje que responde al paradigma de la programación orientada a objetos.



Compiladores

Un compilador es un programa que traduce los programas fuente escritos en lenguajes de alto nivel -Pascal, FORTRAN, ...- a lenguaje máquina.
Los programas escritos en lenguajes de alto nivel se llaman programas fuente y el programa traducido, programa objeto.
El compilador traduce -sentencia a sentencia- el programa fuente.

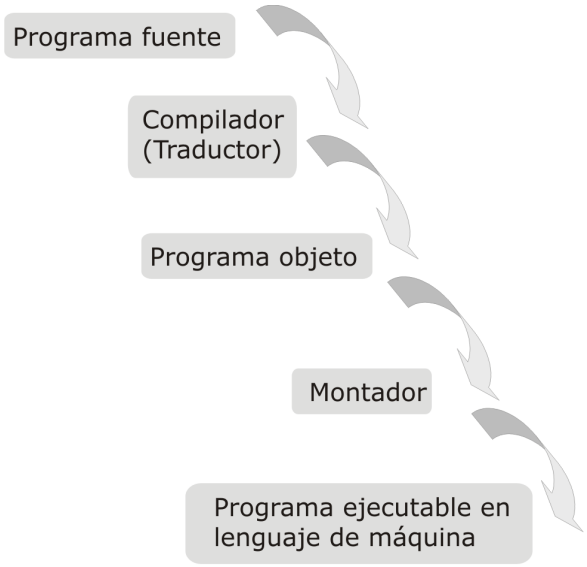
Los lenguajes compiladores característicos son:

C C++ PASCAL COBOL

La compilación y sus fases

La compilación es el proceso de traducción de programas fuente a programas objeto. El programa objeto obtenido de la compilación ha sido traducido normalmente a código máquina.

Para conseguir el programa máquina real se debe utilizar un programa llamado *montador o ensamblador (linker).* El proceso de montaje conduce a un programa en lenguaje de máquina directamente ejecutable.



El proceso de ejecución de un programa Pascal, por ejemplo, tiene los siguientes pasos:

1) Escribir el programa fuente con un editor (programa que permite a una computadora actuar de modo similar a un procesador de texto) y guardarlo en un dispositivo de almacenamiento (por ejemplo, un disco, dkt).
2) Introducir el programa fuente en memoria.
3) Compilar el programa con el compilador Pascal.
4) Verificar y corregir errores de compilación (listado de errores).
5) Obtener el programa objeto.
6) Obtener el programa ejecutable.
7) Ejecutar el programa y, si no existen errores, se tendrá la salida.

Cierre de la unidad

Hasta aquí, entonces, hemos visto los conceptos básicos de programación. Lo invitamos a continuar con el estudio de la próxima unidad cuyo eje temático es el concepto de lógica y su aplicación a la programación estructurada.

Unidad 3 – Lógica para la programación estructurada

Presentación

En programación existen diversos paradigmas que responden a distintas visiones de la realidad: estructurado, lógico, orientado a eventos, orientado a objetos, funcional, entre otros.

En esta unidad analizaremos el **paradigma estructurado**, que es un paradigma básico, y una de las tantas formas de representar la lógica de programación.

El paradigma estructurado es aquel que contiene tres tipos de estructuras básicas: las operaciones de tipo secuencial, las operaciones condicionales y, por último, las de tipo cíclicas.

Un sistema puede dividirse en módulos que enfocan una parte del sistema. Los módulos, a su vez, se desarrollan en el paradigma estructurado por una combinación de las tres estructuras básicas. Es una manera de dividir un gran problema en varios pequeños de más fácil manejo y control.

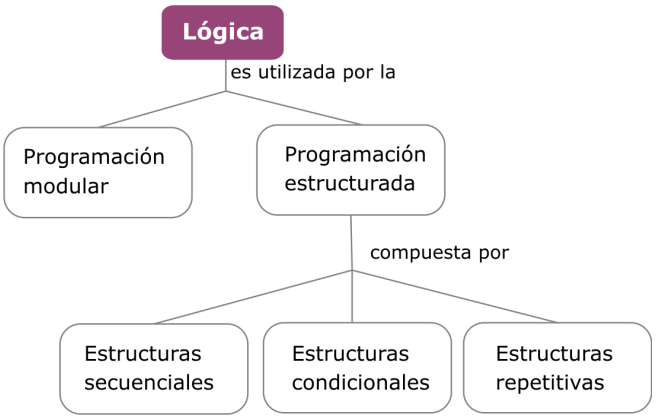
A través del estudio de esta unidad esperamos que usted sea capaz de:

- Identificar, dado un problema, datos que luego de ser procesados permiten obtener salidas.
- Utilizar las distintas estructuras del paradigma para lograr la solución de un problema.

A continuación, le presentamos un detalle de los contenidos y actividades que integran esta unidad.

Organizador Gráfico de la Unidad 3

El siguiente esquema le permitirá visualizar la interrelación entre los conceptos que a continuación abordaremos.



6. Concepto de lógica de programación

El ser humano piensa y se expresa por medio de ideas más o menos lógicas y articuladas, enunciando relaciones en las que afirma o niega algo.

La computadora no tiene razonamiento, lo que sí posee es una velocidad muy alta de procesamiento. El hombre es el que aplica su lógica e inteligencia en un programa o sistema informático y la PC, su velocidad. Ambos forman un buen equipo.

A partir de este punto podemos empezar a pensar en la programación.

Para comenzar, aclaremos algunos conceptos.

- **¿Qué es un paradigma?**

  En sistemas reconocemos varios paradigmas de programación.
  Un paradigma es una manera de ver la realidad y obtener de ella un modelo, por ejemplo los paradigmas económicos más importantes en el Siglo XX fueron el capitalismo y el comunismo, los que convivieron simultáneamente y luego uno de ellos fue remplazando al otro según la región del mundo.

En sistemas ocurre lo mismo, en siglo pasado el paradigma básico era el Estructurado, luego vinieron el orientado a eventos, el orientado a objetos, el lógico, el funcional, que eran distintas formas de interpretar la realidad en un modelo computacional

El Paradigma Estructurado se basa en tres estructuras, secuencial, condicional y repetitivas, el lenguaje prototipo es el Cobol, Pascal, C.

El Paradigma Orientado a Eventos se basa en que para que se dispare una acción necesita de un evento en particular, el lenguaje prototipo era el Visual Basic 6

El Paradigma Orientado a Objetos se basa en cuatro pilares fundamentales, la abstracción, el polimorfismo, el encapsulamiento y la herencia entre sus clases que generan los distintos objetos, el lenguaje prototipo es el C++, Java, plataforma .Net

El Paradigma Lógico es aquel que se utiliza en Inteligencia Artificial donde se posee una gran base de datos, que ante ciertas consultas y comprobaciones de reglas estipuladas ofrece una solución, el lenguaje prototipo es el Prolog.

El paradigma funcional es aquel donde se utilizan funciones matemáticas complejas, por ejemplo cálculos de las autopistas, el lenguaje conocido es el Gofer.

- **¿Qué es un programa?**

  Un programa es una secuencia finita de pasos lógicamente ordenados, que nos permitirá llegar a una solución, según los datos que se introduzcan en el.

- **¿Qué es un algoritmo?**

  Es la representación lógica de los pasos de un programa por medio de algún método grafico o seudocódigo

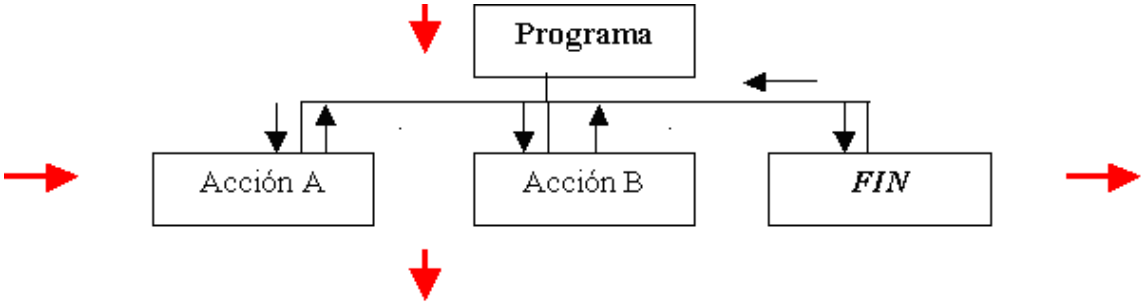
  Ahora veremos las estructuras básicas de la programación estructurada y una forma de representación de la lógica llamado diagrama de Jackson

Esta es una forma más de representar la lógica de programación.

Un proceso es una secuencia de instrucciones que ocupan una cantidad de recursos del computador y permiten la solución de un problema, ya sea en

función de un solo proceso o de varios subprocesos que al combinarse, correcta y lógicamente, generan la solución deseada.

La diagramación Jackson consiste en dibujar a todos sus elementos como rectángulos, que se encuentran ordenados en forma secuencial de izquierda a derecha y poseen distintos niveles entre sí.



En todos los casos dentro del rectángulo se escribe la instrucción, el procedimiento que se llama, etc.

Proceso 1	Imprimir listado sueldo
Sueldo = ch * vh	Si a = b

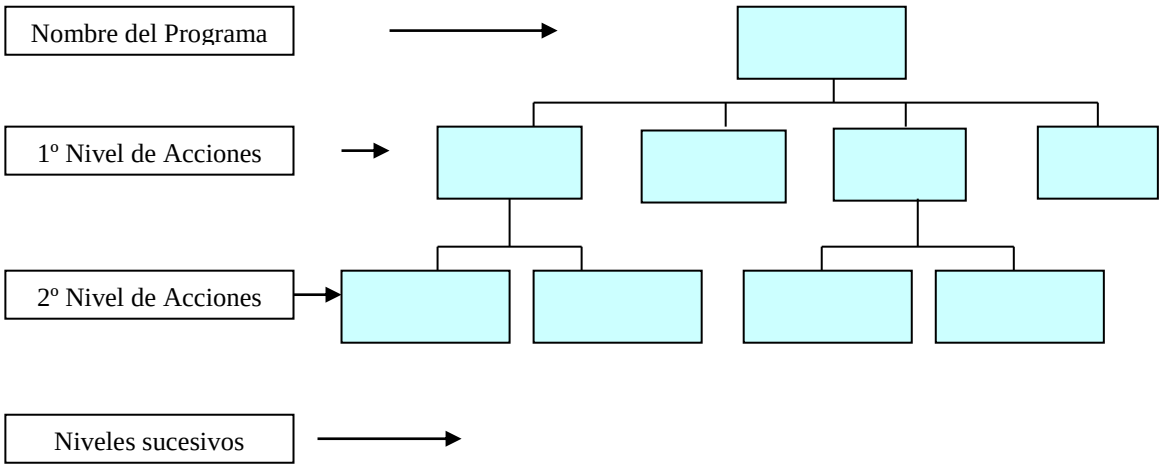
- **Tipos de Estructuras**

Dentro de la programación estructurada reconocemos tres estructuras básicas

1. Estructuras secuenciales
2. Estructuras condicionales
3. Estructuras iterativas o de repetición

1. ESTRUCTURA SECUENCIAL

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 final del proceso.



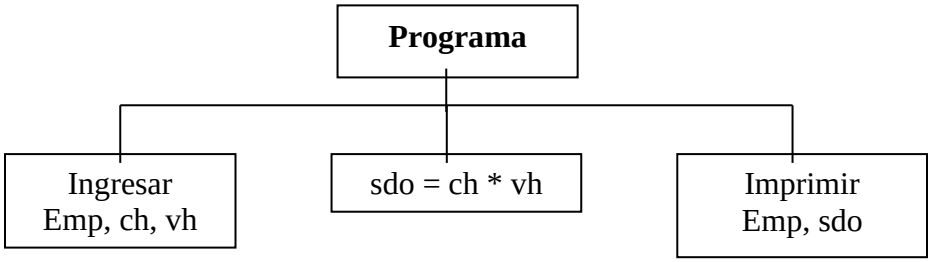
Dado el valor de la hora y la cantidad de horas trabajadas por un empleado, calcular su sueldo

Lo primero que haremos es identificar 3 cosas

Datos: valor de la hora, vh
Cantidad de horas trabajadas, ch

Resultado: sueldo, sdo

Proceso: sdo = ch* vh



En seudocódigo sería:

```
Comienzo
        Ingresar "el nro de empleado"
        Leer, emp
        Ingresar "la cantidad de horas"
        Leer,ch
        Ingresar "el valor de la hora"
        Leer vh
        Sdo = ch * vh
        Imprimir "el empleado, emp, cobra,  sdo,pesos"
Fin
```

EJERCICIOS DE VARIABLES Y ASIGNACIONES

1. Ingresar dos valores enteros y sumarlos
2. Ingresar tres valores, imprimir la suma total, sólo sabe sumar de a dos.
3. Ingresar tres valores y sumarlos, se puede sumar de a varios operandos.
4. Ingresar los lados de un triángulo calcular su perímetro
5. Ingresar dos lados de un triángulo rectángulo y calcular, la hipotenusa, el perímetro, la superficie.
6. Ingresar los lados de un rectángulo y calcular su diagonal principal, superficie y perímetro.

7. Ingresar dos valores, calcular su suma, su producto y la resta del 1ro menos el 2do.

8. Ingresar el valor de la hora y el tiempo trabajado por un operario, calcular su sueldo.

9. Ingresar el tiempo trabajado por un operario y si el valor de la hora es de 10 pesos, calcular su sueldo

10. Una concesionaria de autos paga a cada vendedor $ 500 por mes más un plus del 10 % del precio sobre cada vehículo vendido y un valor constante de 50 pesos por cada uno de ellos, sólo vende un tipo de vehículo, calcular su sueldo

2. Estructura Condicional

Las estructuras selectivas se utilizan para tomar decisiones lógicas; de ahí se suelen denominar estructuras de decisión o alternativas.

En las estructuras selectivas se evalúa una condición y en función del resultado de la misma se realiza una opción u otra. Las condiciones se especifican usando expresiones lógicas.

Las representaciones de una estructura selectiva se hace con palabras en pseudocódigo como: if, then, else sino en español si, entonces, si_no. En diagramación Jackson se le incorpora un círculo en el extremo superior derecho al rectángulo.

La condición que se desea comprobar va adentro del rectángulo

Puede darse tres casos

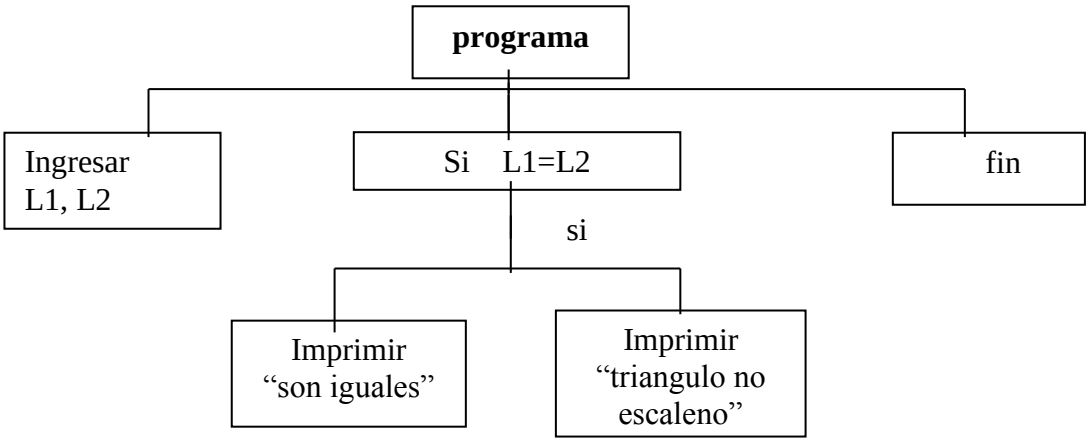
A. condicionales con salida por el verdadero de la condición especificada

Ingrese dos lados de un triangulo, indique si son iguales y por lo tanto que el triangulo no puede ser escaleno.

Datos: Lado 1 , L1
Lado 2 , L2

Resultado: imprimir son iguales

Proceso: comparar L1 si es = a L2



En seudo código sería :

```
Comienzo
        Ingresar "ingrese el primer lado "
        Ingresar L1
        Ingresar "ingresar el segundo lado"
        Ingresar L2
        Si  L1 = L2 entonces
                Imprimir "son iguales"
                Imprimir "triangulo no escaleno"
        Fin si
fin
```

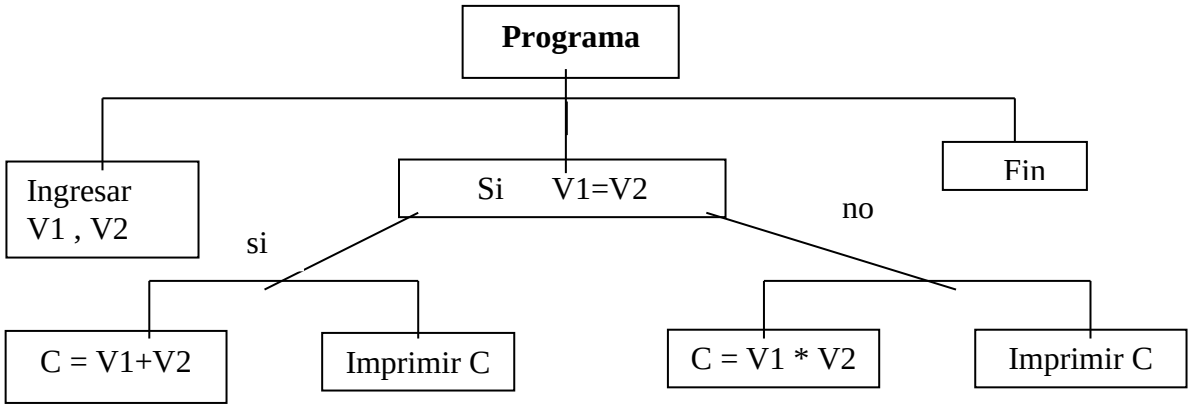
B. Condicionales con salida por el verdadero y por el falso de la condición especificada

Ingresar dos valores, sumarlos si son iguales y multiplicarlos si son distintos

Datos: valor 1, V1
Valor 2, V2

Resultado: realizar el producto si son distintos
Realizar la suma si son iguales

Proceso: C = V1 + V2
C = V1 * V2



En seudocódigo seria:

Comienzo

```
Ingresar "ingrese el primer valor"
Ingresar V1
Ingresar "ingrese el segundo valor"
Ingresar V2
Si V1=V2 entonces
        C = V1 + V2
        Imprimir "son iguales y la suma es C"
De lo contrario
        C = V1 + V2
        Imprimir "son distintos y el producto es C"
Fin si
```

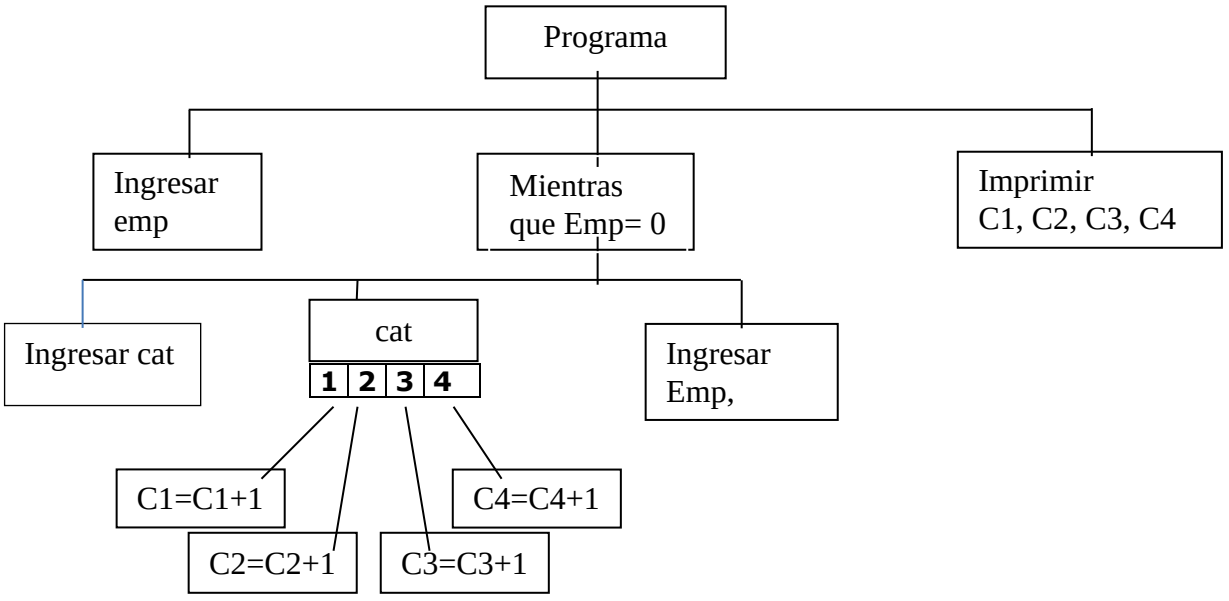
Fin

C. Condicional case o switch

Este tipo de condicional sólo se da cuando una variable puede tomar varios valores enteros en general y por cada una de esos valores tomar distintas alternativas de acción.

Ingresar el nro. de empleado y categoría a la que pertenece (son 4),calcule cuantos empleados hay en cada una de ellas. Los datos finalizan con emp = 0

Para este tipo de ejercicios es necesario un ciclo que veremos más adelante porque se supone que vienen varios datos, en este ejemplo el ciclo es de tipo no exacto "mientras"



En seudo código seria

```
Comienzo
        Ingresar "ingrese empleado y categoría"
        Ingresar emp, cat
        Hacer hasta emp = 0
                Seleccionar caso cat
                        Caso 1:   C1=C1+1
                        Caso 2:   C2=C2+1
                        Caso 3:   C3=C3+1
                        Caso 4:   C4=C4+1
                Fin selección
                Ingresar "ingrese empleado y categoría"
```

```
            Ingresar emp, cat
        Repetir
        Imprimir "la cantidad de empleados de la cat 1 es C1"
        Imprimir "la cantidad de empleados de la cat 2 es C2"
        Imprimir "la cantidad de empleados de la cat 3 es C3"
        Imprimir "la cantidad de empleados de la cat 4 es C4"
Fin
```

EJERCICIOS DE OPERACIONES CONDICIONALES

a. Ingresar dos valores, indicar si son iguales

b. Ingresar un valor indicar si es positivo, negativo o cero

c. Ingresar dos valores y realizar el producto, si el 1ro es mayor al 2do, si son iguales solo indicarlo

d. Ingresar dos valores y realizar la resta del mayor menos el menor

e. Ingresar los tres lados de un triángulo e indicar que tipo de triángulo es

f. Ingresar tres valores, sumarlos, calcular el promedio e indicar cúal de estos valores es mayor al promedio

g. Ingresar cuatro valores, sumar el 1ro y el 2do, el 3ro y el 4to, indicar cúal de esta sumas es mayor

h. Ingresar la edad y la altura de dos personas, indicar la estatura del de mayor edad

i. Ingresar el valor de la hora y el tiempo trabajado por un empleado, calcular su sueldo si se sabe que recibe un premio de $ 100 si trabajo más de 50 hs y si trabajo más de 150 hs le dan $ 100 adicionales.

j. Ingresar tres valores correspondientes al día, mes y año de una fecha, indicar si es válida, considerar los años bisiestos (existe una funcion que devuelve "B" en caso de bisiesto y "N" si no lo es)

k. Ingresar el sueldo, categoría y antigüedad de un empleado, calcular el sueldo final si se le da $ 50 por cada año trabajado a cada empleado de la categoria 1.

l. Sobre los datos del ejercico anterior imprimir los sueldos de los empleados con más de 5 años de antigüedad

m. Ingresar las horas trabajadas por un empleado y su categoría, calcular su sueldo si se sabe que los de la categoría 1 cobran $50, la 2 cobra $ 70 y la 3 cobra $ 80.

3. Estructuras Iterativas

Cuando se utiliza un bucle para sumar una cantidad de números, se necesita saber cuántos números se han de sumar. Por eso necesitaremos conocer algún medio para detener el bucle.

En diagramación Jackson se le incorpora un asterisco en el extremo superior derecho al rectángulo.

El bucle podrá también terminar, como por ejemplo, incorporando cualquiera de estas condiciones:

- ***hasta _que*** variable sea variable
- ***desde 1 hasta N***

Para detener la ejecución de los bucles se utiliza una condición de parada, caso contrario el bucle entrará en un loop permanente, es decir, nunca saldrá de dicho bucle; conclusión el programa no funcionará.

La condición del bucle normalmente se indica al principio o al final de este, de esa manera podemos considerar tres tipos de instrucciones o estructuras repetitivas:

- ***Mientras*** ***(while)***
- ***Repetir*** ***(repeat)***
- ***Desde / para*** ***(for)***

La condición que se desea que se cumpla, se escribe a la derecha del rectángulo

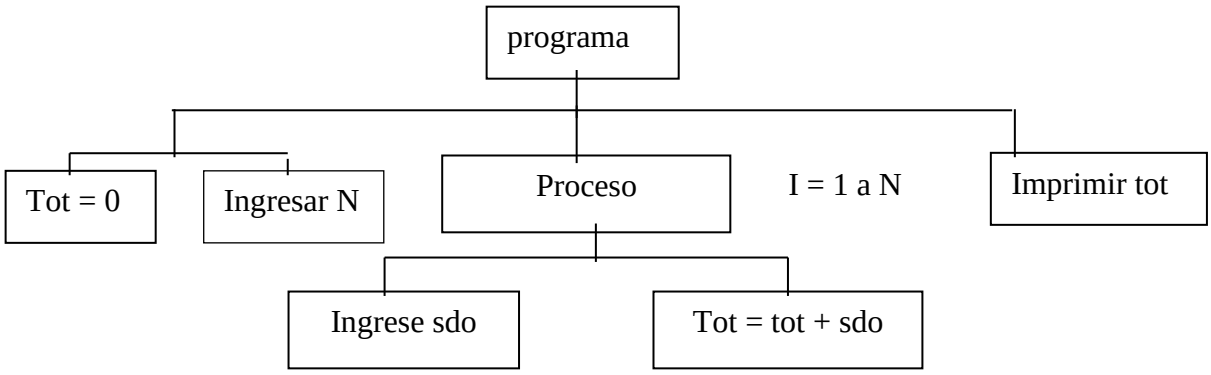
Pueden darse dos casos de iteración:

A. ciclos repetitivos exactos

Dados los sueldos de N empleados, determinar el total a pagar

Análisis

Tipo de ciclo = exacto N **datos** = N, sdo **Rta**: tot **Proceso:** tot = tot + sdo

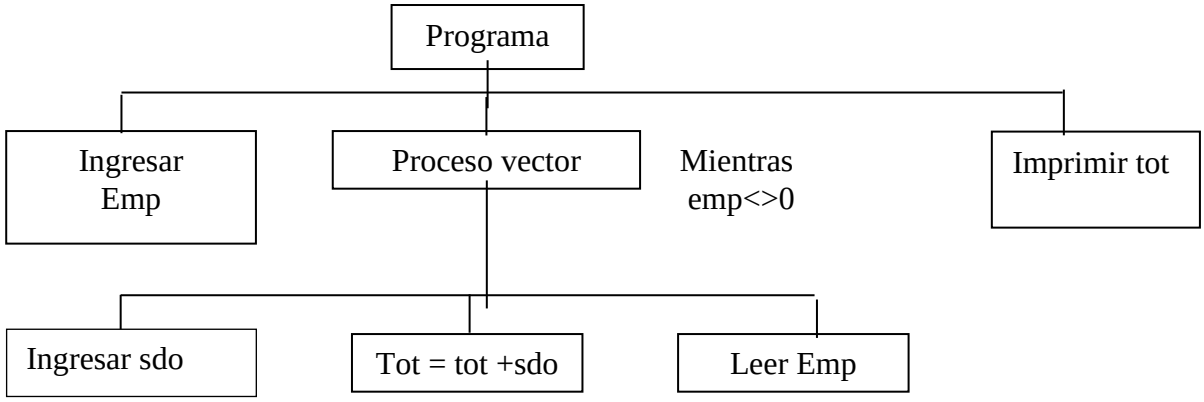


B. Ciclos repetitivos inexactos

Ingresar los sueldos de los empleados de una empresa hasta que el empleado sea igual a 0, calcular el total de sueldos a pagar

Análisis

Tipo de ciclo = inexacto "hasta" **datos** = Emp, sdo
proceso: tot = tot + sdo **Rta**: tot



En seudo código seria:

```
Comienzo
        Ingresar "ingrese el empleado"
        Ingresar emp,sdo
        Hacer mientras  emp <> 0

                Ingresar "ingrese el sueldo"
                Ingresar sdo
                Tot = tot + sdo
                Ingresar "ingrese el empleado"
                Ingresar emp

        Repetir
        Imprimir "el monto total a pagar es tot"
Fin
```

EJERCICIOS DE CICLOS, CONTADORES y ACUMULADORES

1. Ingresar 25 números, calcular su promedio

2. Ingresar 20 notas y nombres de alumnos, indicar los aplazados (menos de 4) y el nombre a quien pertenece esa nota

3. Ingresar N sueldos e indicar su suma y su promedio

4. Ingresar facturas hasta nro de factura = 0, sumar sus importes, indicar el total gastado y cúales y cuantas superan los $1000.

5. Sobre el ejercicio anterior indicar cuántas superan los $ 10000.-

6. Sobre el ejercicio anterior indicar cuántas estan entre $ 400 y $ 700 inclusive.

7. Ingresar 10 valores, indicar cuántos son positivos, cuántos negativos y cuántos ceros

8. Ingresar valores hasta uno = 0, indicar la cantidad de números ingresados y su promedio

9. Ingresar nombres y notas de alumnos teniendo en cuenta que la carga finaliza con nota = 11, calcular el promedio, imprimir los aprobados, cuántos estan entre 4 y 6..

10. Ingresar la patente y monto de la multa de 50 autos, indicar cuántos montos superan los $ 40 y del total cobrado que porcentaje representa la suma de estos últimos

11. Ingresar N valores y calcular promedio de positivos, de negativos y cantidad de ceros

12. Ingresar los datos de facturación de una empresa.
    Número de factura
    Número de artículo
    Cantidad vendida
    Precio unitario

    Los datos finalizan con numero de factura = 0, cada factura sólo tiene un número de artículo, existen tres artículos

    Se desea saber:
    Valor de cada factura
    Facturación total
    Cuánto se vendio del artículo 1 en cantidad
    Cuántas facturas mayores de $ 3000 se hicieron
    Qué porcentaje representa el monto vendido por cada artículo sobre el total

¿COMO SE CALCULA EL VALOR MÁXIMO O MÍNIMO DENTRO DE UN LOTE DE DATOS?

La idea básica sea el tipo de ciclo que sea es tomar el primer valor ingresado y guardarlo como máximo o mínimo según sea lo solicitado.

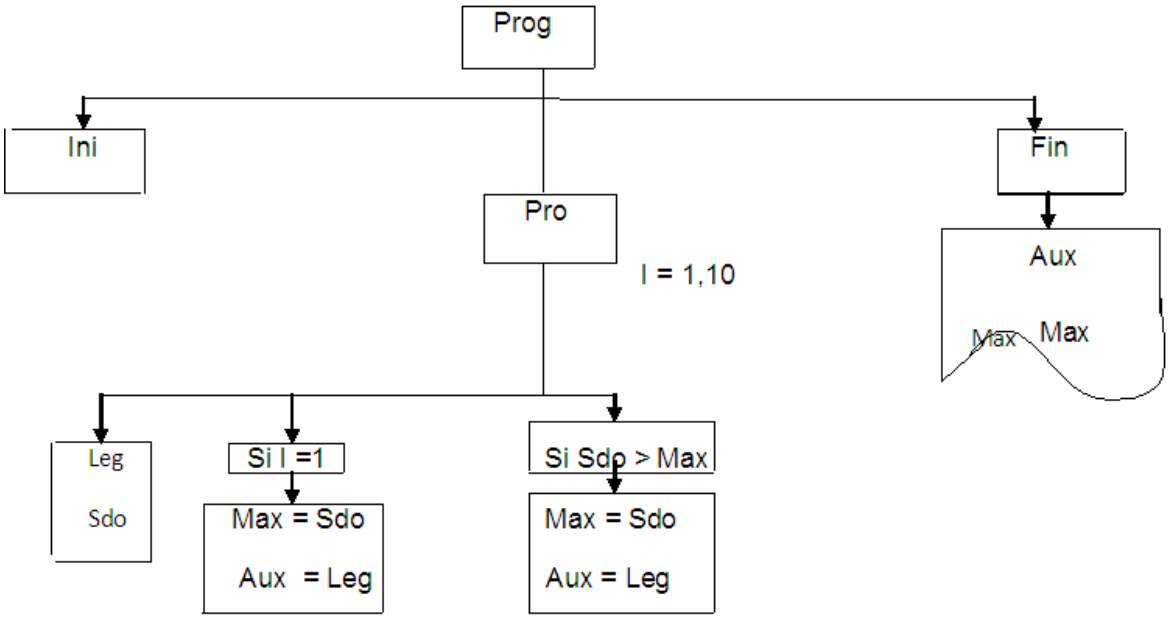
Luego se recorre el dato de datos y se van comparando los distintos valores contra el guardado en el máximo o en el mínimo, quedando al final del proceso el primer valor máximo o mínimo encontrado.

Si de ese valor me pidiesen algunos datos referenciales al mismo, esos datos serán guardados en tantos auxiliares como sean necesarios.

En el caso del ciclo exacto yo me doy cuenta de cual es el primer dato ingresado porque la variable de control del ciclo tiene valor 1

En el caso del ciclo no exacto, yo debo colocar esa variable de control que puede ser un contador que cuando es de valor 1 significa que es el primero o una bandera que cambia de estado al pasar la primera vez.

Veamos el siguiente ejemplo



EJERCICIOS SOBRE MAXIMOS y MINIMOS

1. Ingresar N temperaturas e indicar la máxima y mínima

2. Ingresar temperaturas hasta una temperatura igual a 1000, indicar la mayor y menor

3. Ingresar los sueldos y nombres de 30 empleados, indicar el sueldo mayor y a quién pertenece

4. Ingresar las edades y estaturas de los alumnos, calcular la edad promedio, la edad mayor y la estatura menor, los datos finalizan con edad = 0

5. En una carrera de autos se ingresan el número de auto y su tiempo, indicar cuál ganó y cúal fue el último

Cierre de la unidad

Como cierre de esta unidad le proponemos que realice una pequeña búsqueda de información para averiguar qué paradigma de programación se está utilizando en el ámbito de los negocios, en la actualidad. Lo invitamos a continuar con el estudio de la próxima unidad cuyo eje temático es Internet, un ámbito seguramente conocido por usted…