



**Universidad Abierta
Interamericana**

**FACULTAD DE
TECNOLOGÍA INFORMÁTICA**

MANUAL DE INGRESO
MÓDULO DE INFORMÁTICA

Autores:

Mg. Nicolás Battaglia
Ing. Christian Chamula

Coordinador:

Dr. Marcelo De Vincenzi



FACULTAD DE TECNOLOGÍA INFORMÁTICA

Manual de Ingreso de la Facultad de Tecnología Informática - Módulo de Informática

Autores:

**Mg. Nicolás Battaglia
Ing. Christian Chamula**

Coordinador

Dr. Marcelo De Vincenzi

Índice de contenido

Portadilla

Unidad 1: Hardware

- 1.01 Breve reseña de la Evolución
- 1.02 Generación de Computadoras
- 1.03 ¿Qué es una computadora?
- 1.04 ¿Qué es Hardware?
- 1.05 Núcleo Central de una Computadora
- 1.06 Unidades Periféricas (dispositivos)
- 1.07 Software
- 1.08 Autoevaluación

Unidad 2: Software

- 2.01 Concepto de programa
- 2.02 Proceso para la creación de un programa
- 2.03 Concepto de algoritmo
- 2.04 Datos, tipos de datos y operaciones
- 2.05 Constantes y variables
- 2.06 Tipos de instrucciones
- 2.07 Elementos básicos de un programa
- 2.08 Los lenguajes de programación
- 2.09 Los entornos integrados de desarrollo (IDE)
- 2.10 Autoevaluación

Unidad 3: Programación

- 3.01 Concepto de lógica
- 3.02 ¿Qué es un paradigma?
- 3.03 ¿Qué es un programa?
- 3.04 ¿Qué es un algoritmo?

3.05 Diagramas de flujo

3.06 Tipos de Estructuras

3.07 Autoevaluación

Respuestas a las actividades de autoevaluación

Unidad 1

Unidad 2

Unidad 3

Bibliografía

Battaglia, Nicolás

Manual de Ingreso de la Facultad de Tecnología Informática : módulo de informática / Nicolás Battaglia ; Christian Chamula ; coordinación general de Marcelo De Vincenzi. - 1a ed. - Ciudad Autónoma de Buenos Aires : Universidad Abierta Interamericana, 2021.

Archivo Digital: descarga

ISBN 978-987-8403-19-9

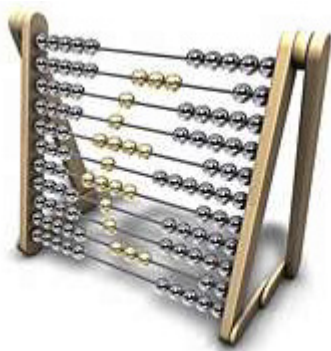
1. Computación. I. Chamula, Christian. II. De Vincenzi, Marcelo, coord. III. Título.
CDD 004.071

Unidad 1: Hardware

1.01 Breve reseña de la Evolución

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 permite comprender cómo llegamos a las computadoras actuales.

Los chinos, hace más de 3.000 años, diseñaron el ábaco. Gracias a este instrumento con bolas que corrían de izquierda a derecha hacían cálculos y todavía hoy es usado en Japón y en China.



Ábaco

En 1642, Pascal crea una máquina mecánica de sumar, parecida a los cuentakilómetros de los autos, pero mostraba algunos problemas con las largas sumas, por lo que, en 1671, Leibnitz le agregó la posibilidad de restar, sumar, multiplicar y dividir. Era una máquina con ruedas dentadas, cada rueda tenía 10 dientes, que correspondían a los números del 0 al 9.



Pascalina

Los conceptos de esta máquina se utilizaron durante mucho tiempo, pero exigían la intervención de un operador quien debía copiar a un papel los resultados parciales.

Babbage diseñó y desarrolló la primera computadora de uso general. Un genio para su época, la que no lo ayudó para poder terminar de construirla (no existía la tecnología suficiente), él la llamó “Máquina de las diferencias”.

La primera operación de procesamiento de datos fue lograda en 1890 por Hernan Hollerith. Éste desarrolló un nuevo sistema mecánico para calcular y agrupar datos de censo, que se basaba en tarjetas perforadas. Lo utilizaron en el censo de la población en Estados Unidos en donde se logró por primera vez que los resultados fueran conocidos a los dos años y medio, cuando en el censo anterior se tardaron siete años.

La primera mujer programadora fue Ada Augusta Byron (1815-1852). Ella se interesó en los estudios de Babbage, a quien ayudó.

La primera computadora totalmente electrónica se llamó ENIAC (*Electronic Numeric Integrator and Calculator*) fue construida entre 1943 y 1945 por John Manly y J. Eckart. Podía multiplicar 10.000 veces más, pero tenía sus problemas, ya que estaba construida totalmente por válvulas, consumía gran cantidad de energía y producía mucho calor. Esto hacía que las válvulas se quemaran con facilidad y las casas del vecindario sufrieran cortes de luz.

El primer intento de sobreponerse a estos errores y limitaciones de velocidad fue Howard Aiken quien, junto a ingenieros de IBM, crearon una calculadora automática llamada Mark I, en 1944.

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

1.02 Generación de Computadoras

1. 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, ya que resolvía 5.000 cálculos por segundo.

2. Segunda generación (1955-1965)

Se llamaba a 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.



UNIVAC puramente basada en transistores

Aspectos característicos como:

- Como podrás imaginar, ello supondría una gran reducción del tamaño de las máquinas. También consumen menos y producen menos calor.
- Todo ello hace que podamos hablar ya del nacimiento del minicomputador.

- Estos equipos comienzan a utilizarse en más sectores, sobre todo en banca y, contabilidad y logística de almacén en general.
- Vemos que se desarrolla, en 1959, la microprogramación.
- También se empiezan a usar lenguajes más complejos, llamados “de alto nivel”.
- La comercialización de COBOL fue muy famosa.

Aparece FORTRAN de la mano de IBM, considerado como el primer lenguaje de programación de alto nivel de propósito general, que formaba parte de una de las máquinas protagonistas de la generación, el IBM 1401.

3. 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.) miniaturizado 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. Así se conseguía un procesamiento más capaz en un menor espacio y con menos elementos independientes, lo que favorece la reducción de incidencias.

Aspectos característicos como:

- Es muy destacable que los ordenadores pasan a utilizarse, “de manera habitual”, con fines comerciales. Podemos decir que durante estos años los aparatos se vuelven más “accesible”, siendo conocido por todo el mundo y contando con características que lo hacen útil para un público mayor.
- Mejoran la fiabilidad y son más flexibles.
- El teleproceso y la multiprogramación se vuelven comunes.

Además, se comienza a hablar ya del computador a nivel personal.

En el 1969 se da el surgimiento de ARPANET, que fue precursor del actual internet. Pocos meses después se produce el cable de fibra óptica. Al acabar la época los circuitos integrados se habían desarrollado muchísimo habiendo reducido su tamaño, incluían la denominada tecnología MOS y se hicieron más baratos.

4. Cuarta generación (1980-1990)

Se la denomina de 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 como ahora todo se resume en un centímetro cuadrado. De esta forma muchas familias comenzaron a tener computadoras en sus hogares.



El primer microprocesador de la historia

Aspectos característicos como:

- Ello supone una nueva miniaturización de muchas de las partes del ordenador.
- También se da una multiplicación en potencia, capacidad y flexibilidad.
- Hasta el punto de aparecer y ponerse en venta los ordenadores personales, lo cual ocurre en el año 1977.

5. Quinta generación (1990 al 2000)

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 más programas simultáneamente, y así darles más velocidad a los distintos tipos de computadoras. Con un solo procesador (núcleo o “core”) sólo se puede ejecutar un programa por vez.

Aspectos característicos como:

- Aumentan exponencialmente la velocidad y la cantidad de memoria disponibles en los equipos.
- Los lenguajes se traducen de manera inmediata.
- Empiezan a introducirse cantidad de puertos en los ordenadores y, con ello, se multiplican las posibilidades y se permite una mayor personalización de estos. Los más populares y con mayor importancia son, sin duda, los dispositivos de almacenamiento.
- Los ordenadores vuelven a poder diseñarse siendo aún más pequeños.
- Los softwares se multiplican, apareciendo ya de todo tipo y de todos los niveles de complejidad.
- Todo ello hace que vuelva la moda de clonar equipos famosos.
- El contenido multimedia destaca sobre el resto.

En el año 1991 IBM se encarga de diseñar y crear un interruptor del tamaño de un átomo, algo de interés nulo en el momento, pero esencial para el desarrollo de ordenadores enanos. También DEC lanza un supercomputador que sigue 26 mil millones de instrucciones básicas en un segundo.

NCR, por su parte, lanza Notepad NCR 3125, un microcomputador con lápiz y sin teclado, tamaño folio, de sólo 3 cm de grosor. Incluye multitud de funciones y su acceso requiere de escritura manual. Se trata del antecesor de las tabletas y smartphones táctiles.

Un año después se lanza la WWW, World Wide Web o Red Informática Mundial, el sistema de distribución de documentos hipertexto e hipermedia a través de internet por excelencia.

Tras finalizar la Guerra Fría, la caída del Muro de Berlín y la disolución de la Unión Soviética, en 1957 en Estados Unidos se crea la Advanced Research Projects Agency (Agencia de Proyectos para la Investigación Avanzada de Estados Unidos) conocida como ARPA. Esta organización creó las bases de lo que es Internet. En 1962 se logra conectar computadores de forma descentralizada en un solo espacio.

En 1965 logran conectar un computador en Massachusetts con otro computador en California. Para el año 1969 ya había cuatro universidades con el proyecto Arpanet, para mantener comunicaciones en caso de guerra. En 1973, Arpanet logró conectar instituciones en Inglaterra y Noruega. Comienza a utilizarse el protocolo TCP/IP en 1983, el año oficial de nacimiento de Arpa Internet y poco después se conocería solo como Internet.

6. Sexta generación (2000 a la actualidad)

Nos encontramos en lo que se conoce como sexta generación de computadoras, una etapa en la que no existe una característica general, sino que encontramos mucho de todo en cuanto a cantidad y calidad.

Lo que sí encontramos como punto de inflexión para comenzar la etapa es la conectividad inalámbrica que nos permite estar conectados a las redes y a otros aparatos sin necesidad de usar cables.

Destacaremos puntos como:

- El desarrollo de otros aparatos inteligentes, primero teléfonos y después muchos otros como televisores, relojes e incluso electrodomésticos.
- Una brutal oferta de dispositivos para todos los gustos y necesidades.
- Internet como elemento habitual y, de hecho, necesario en el día a día de todo el mundo.

- La puesta a disposición de todos los usuarios los servicios en la nube.
- La popularización del contenido en streaming.
- También se desarrolla considerablemente el comercio online hasta convertirse, de hecho, en un estándar.
- Un salto vertiginoso en cuanto a inteligencia artificial.
- Uso de arquitecturas vectoriales y paralelas para los ordenadores.
- Destaca y cobra importancia el volumen de almacenamiento de las memorias tanto internas como externas.

Los inventos y eventos de este milenio son el WiFi, la fibra óptica, la capacidad de las unidades de almacenamiento, los discos duros SSD, los teléfonos inteligentes, los sistemas operativos móviles, los portátiles de mucho menor tamaño y los que ya se conocen como “portátiles de sobremesa” por sus increíbles prestaciones, idénticas a las de los PCs.

1.03 ¿Qué es una computadora?

Una computadora es una máquina que procesa datos de manera automática, sin intervención humana (una vez que programas y datos se cargaron en memoria), transformándolos en información.

En 1981, IBM lanzó al mercado el IBM PC, el primer ordenador personal.

Las siglas PC se convirtieron en un estándar informático y corresponden a *Personal Computer*, es decir ordenador personal, la PC ha evolucionado a un ritmo vertiginoso.

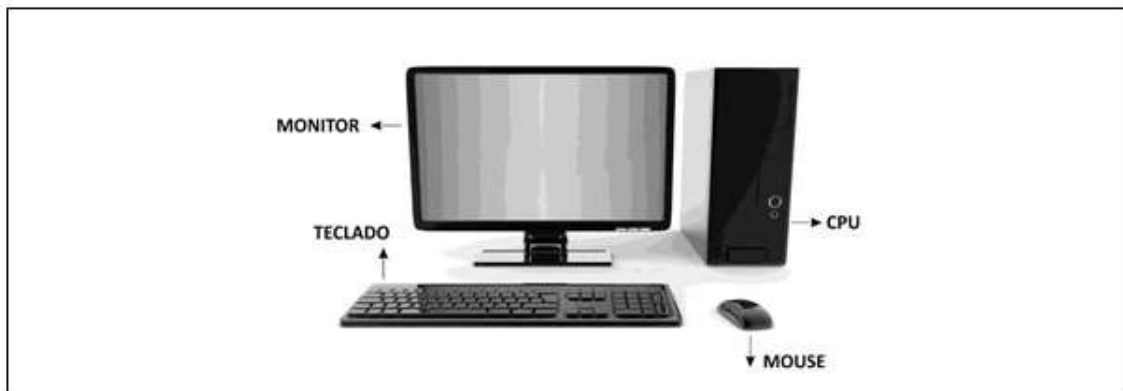
1.04 ¿Qué es Hardware?

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 tangible, 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.

1. 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.

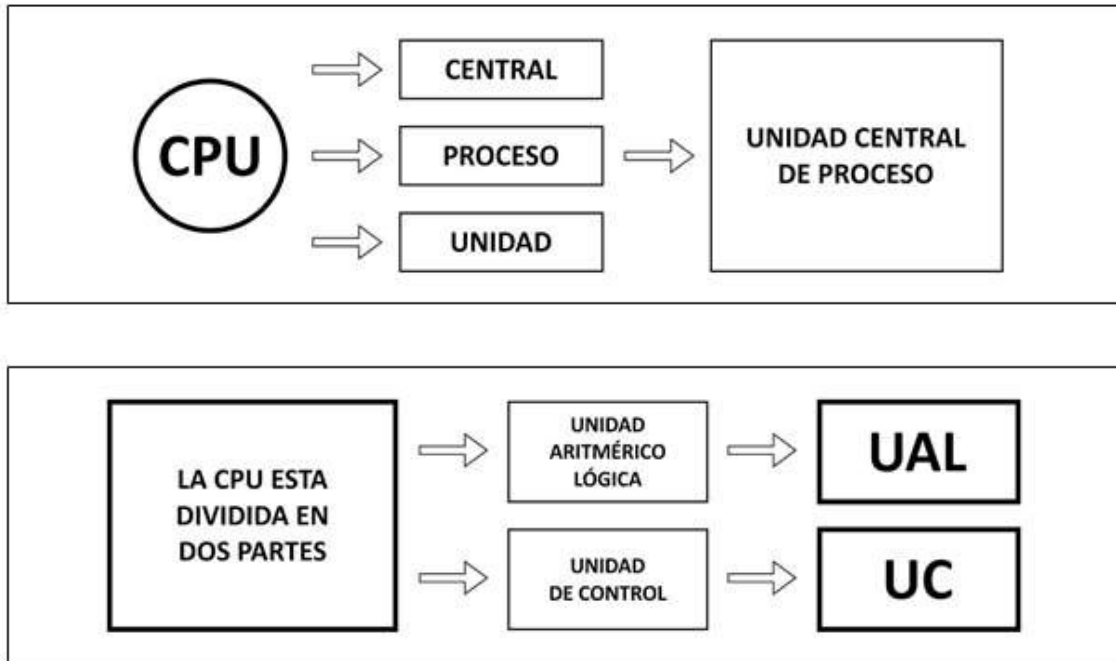
Con el advenimiento del chip integrado, la CPU se convierte en lo que denominamos microprocesador. Ejemplo: Pentium 4, Opteron, Dual Core, etc.

Los dispositivos periféricos se encargan de dar entrada/salida de datos, almacenarlos, o mostrar resultados.

1.05 Núcleo Central de una Computadora

Todas las computadoras presentan un núcleo central formado por la CPU 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.



1. 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 almacenen, y a la memoria que sea leída o escrita.

2. Unidad aritmético-lógica

U.A.L. (*Arithmetic 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 tercera porción de la UCP son los registros para almacenar transitoriamente datos y resultados, instrucciones y direcciones de memoria.

3. Memorias

Las memorias de las computadoras son circuitos que guardan las instrucciones de programas en ejecución, los datos que ellas procesan y los

resultados obtenidos.

Existen dos tipos de memorias electrónicas: una llamada RAM y la otra ROM.

4. 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.

5. 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.

Pero ¿cuántos datos podemos almacenar? Eso depende de la capacidad que la memoria tenga, ya que hay distintos tipos.

Esta capacidad se mide y su unidad de medida es el byte, que corresponde a una combinación de 8 unos y ceros, como 10100011, 00110101, 00000000, etc., siendo que cada uno de los símbolos 1 o 0 es un bit (abreviatura de *binary digit*).

Cada *byte* puede representar una letra, un número, un carácter especial, etc.

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

LETRAS	BYTES
A	01000001
B	01000010
C	01000011
D	01000100
E	01000101

6. Tabla de equivalencias de capacidad

1 Bit	Digito 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:

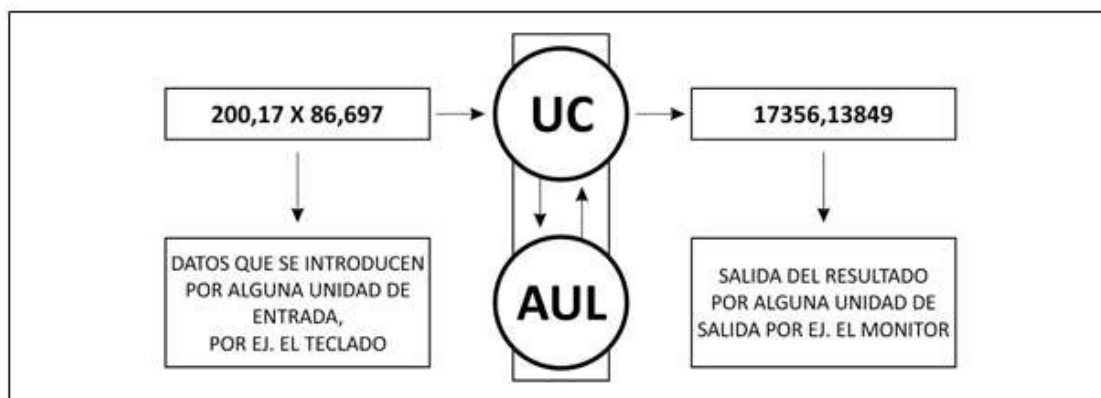
BYTES	KILOBYTE	KILOBYTE
1024		
2048		
	4	
		8
		16

7. 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:



1.06 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.

1. 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; funciona de manera similar a una fotocopidora con distintos grados de definición.

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/grabadora de CDs/DVDs

Dispositivo que permite la lectura de la información contenida en un disco óptico, sea CD o DVD.

Cámaras WEB

Una cámara web o cámara de red (en inglés: webcam) es una pequeña cámara digital conectada a una computadora la cual puede capturar imágenes y transmitir las a través de Internet, ya sea a una página web o a otra u otras computadoras de forma privada. Se utilizan, entre otras cosas, para transmisiones streaming (envío de video en vivo).

Las cámaras web necesitan una computadora para transmitir las imágenes. Sin embargo, existen otras cámaras autónomas que tan sólo necesitan un punto de acceso a la red informática, bien sea ethernet o inalámbrico. Para diferenciarlas las cámaras web se las denomina cámaras de red.

Tabletas gráficas

Una tableta gráfica es un periférico que consta principalmente de una tabla de diferentes tamaños, aunque los más extendidos están entre un A6 y A3, y de un lápiz óptico que permite escribir y dibujar sobre la tabla, reflejando el resultado en la pantalla.

2. Periféricos de salida

Muestran los resultados de algún proceso de los datos ingresados a la computadora, algunos de ellos son:

Monitor

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 calidad de un monitor está dada por la cantidad de pixeles que hay en 1mm², 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 de tinta.
- Láser: tienen microprocesador y memoria propia, más velocidad, mejor calidad de impresión (símil fotocopidora).

Plotter

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

Impresora 3D

Una impresora 3D es una máquina capaz de imprimir figuras con volumen a partir de un diseño hecho por ordenador.

Con volumen quiere decir que tiene ancho, largo y alto.

Una impresora 3D lo que realmente hace es producir un diseño 3D creado con el ordenador en un modelo 3D físico (real).

Es decir, si hemos diseñado en nuestro ordenador, por ejemplo, una simple taza de café por medio de cualquier programa CAD (Diseño Asistido por Computador), podremos imprimirla en la realidad por medio de la impresora 3D y obtener un producto físico que sería la propia taza de café.

Una impresora 3D es algo mágico, es como si pudiéramos por fin crear objetos de “la nada”.

Objetos tan sencillos como una taza de café a objetos mucho más complicados e increíbles como partes de un avión o incluso órganos humanos utilizando las propias células de una persona.

3. 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 permanente de información, aunque se apague el equipo. Ejemplo: dispositivos de almacenamiento masivo (discos rígidos y los actuales SSD (*Solid State Device*, CD, DVDs).

Pendrive

Es una *Flash Rom*, constituida por transistores especiales.

Módem (modulador-de-modulador)

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 platos 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. Estos discos giran, generalmente a 7200 rpm, y el cabezal se mueve para leer y escribir la información donde corresponde.

Generalmente, está 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.

Disco rígido de estado sólido (SSD).

Este tipo de disco rígidos también se instalan en el interior de las computadoras y cumplen el mismo objetivo de los discos rígidos magnéticos, sin embargo, al no poseer ninguna estructura mecánica ni platos internos con movimiento, brindan rendimientos muy superiores a los discos rígidos tradicionales, son silenciosos y pesan mucho menos que sus predecesores. A pesar de estas diferencias sustanciales, presentan algunas desventajas como ser, el costo, la baja tolerancia a fallos y recupero de datos y, además, su vida útil es mucho más corta.

Es un dispositivo de almacenamiento de información, constituido por uno o más discos de material duro (aleación de aluminio

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 audio 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.

1.07 Software

1. ¿Qué es Software?

Está constituido por los programas de un computador que ordenan al hardware (CPU) qué debe hacer. (*Soft* significa blando e intangible).

Software es un conjunto de instrucciones y procedimientos relacionados entre sí para poder llevar a cabo un proceso de datos con una computadora cuando es ejecutado por la CPU.

Hardware y Software se hallan íntimamente ligados ya que no hay Software que pueda llevarse a cabo sin Hardware. Ni Hardware que funcione sin el Software adecuado.

Clasificación de Software

Existen distintos tipos de software, según las prestaciones que brindan. 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:

- 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, etc.).
- La memoria principal.
- Los archivos.
- Los periféricos (mediante los programas “drivers”).

Windows

Desde los primeros computadores, el hombre buscó medios; métodos que le permitieran agilizar y simplificar las tareas que debía hacer con un computador. En la historia de la informática vemos que, en la década de los 80, surge la primera computadora personal con un sistema operativo propio, denominado D.O.S., que 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 gran capacidad de memoria o de métodos sencillos para el intercambio de información entre

esos programas, ni había forma de hacerlo sin tener que cerrar aquel con el que estaban trabajando.

A comienzos del '85 se desarrolla como un reto y desafío a los cambios, la interfaz 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 utilizaba el DOS. Windows '95 presentaba características especiales que le permitían 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.

2. ¿Qué es un utilitario?

Se define por utilitario a los programas que el usuario compra para trabajar, comunicarse, jugar, etc. Ejemplos: MS Word – PowerPoint, Excel. Corel Draw, etc.

Utilitario=Programa=Software

Clasificación de utilitarios



Enlatados

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

Entre los más usados encontrados:

- Procesadores de Textos.
- Planilla de Cálculo.
- Bases de datos.
- Graficadores.

Estos softwares 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 mi casa.

Procesadores 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.

Planillas 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:

CAMPOS	→ NOMBRE
	→ AUTOR
	→ EDITORIAL
	→ AÑO DE EDICIÓN
	→ GÉNERO
	→ COMENTARIO
	
	

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, 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. Una muy utilizada por usuarios particulares es Access que viene con el paquete de Office de Microsoft.

No enlatados

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 que arman los programas 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 uso del tiempo libre de la computadora como, por ejemplo, ajedrez, carreras de auto, simuladores de vuelo, etc.

3. ¿Qué es un Archivo?

Podemos considerar al “ARCHIVO” como aquella información almacenada en una unidad almacenamiento, recuperable por su nombre.

En función del tipo de información que éstos posean se pueden agrupar en: de programa, de texto, de manejo video, de manejo de impresora, etc. Se identifican los archivos 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.

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.

1.08 Autoevaluación

I) Comprendí los temas estudiados en esta unidad si puedo definir

- Hardware.
- Memorias y tipos de memorias.
- Periféricos y tipos de periféricos.
- Software.

II) Comprendí los conceptos más importantes de esta unidad si...

- Comprendo qué es una computadora.
- Entiendo la diferencia entre hardware y software.
- Puedo describir cómo es la estructura interna de una computadora y describir cada una de sus partes.
- Comprendo la diferencia que existe entre cada generación de computadoras.
- Entiendo la diferencia entre bit, byte, megabyte, etc.
- Vinculo los diferentes tipos de memoria y sus implicancias en el uso de una computadora.
- Relaciono el tema de capacidad de almacenamiento con los diferentes tipos de memoria.
- Entiendo la diferencia fundamental entre periféricos de entra, salida y entrada/salida.
- Comprendo la importancia de cada tipo de software y puedo dar ejemplos de cada uno de ellos.

III) Seleccione las opciones correctas (las respuestas al final del cuadernillo)

1. ¿Quién dirige y controla el proceso de la información dentro de una Pc?

- a) La unidad de control.
- b) Unidad Aritmético – Lógica.
- c) Unidad central de proceso.
- d) La memoria principal.
- e) a y c son correctas.

2. ¿Cuál de estos es sólo un puerto de entrada?

- a) la impresora.

- b) el teclado.
- c) la disquetera.
- d) a, b, y c son correctas.
- e) b y c son correctas.

3. ¿Cuántos bytes forman un terabyte?

- a) 256 Bytes.
- b) Cien millones de bytes.
- c) 64 bits.
- d) mil millones de bytes.
- e) un millón de millones de bytes.

4. ¿Cuál es la relación entre un Bit y un Byte?

- a) Un Bit está conformado por ocho Bytes.
- b) Un Byte está conformado por ocho Bits.
- c) Los Bits forman parte de la memoria RAM y los Bytes de la memoria ROM.
- d) A, B y C son correctas.
- e) A, B y C son incorrectas.

5. Los programas que permiten el funcionamiento del hardware del computador conforman:

- a) El sistema operativo.
- b) El software utilitario.
- c) El software aplicativo.
- d) B y C son correctas.
- e) B y C son incorrectas.

Unidad 2: Software

2.01 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 debe contener el programa: entrada, salida y algoritmos de resolución, que 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, en realidad, es el conjunto de códigos que transforman las entradas del programa (datos) en salidas (resultados o información).



Un programa tiene un formato que la computadora utiliza para poder ejecutar las instrucciones. El programa se escribe en un formato denominado código fuente (generalmente fácil de interpretar para los humanos) en donde se desarrollan los algoritmos.

En general, el código fuente es escrito por profesionales que se conocen como programadores. Los programadores utilizan un lenguaje de programación siguiendo algún paradigma de programación. Este código fuente, será podrá -posteriormente- ejecutar en la computadora por la unidad central de procesamiento. También existen programas que son embebidos directamente en el hardware.

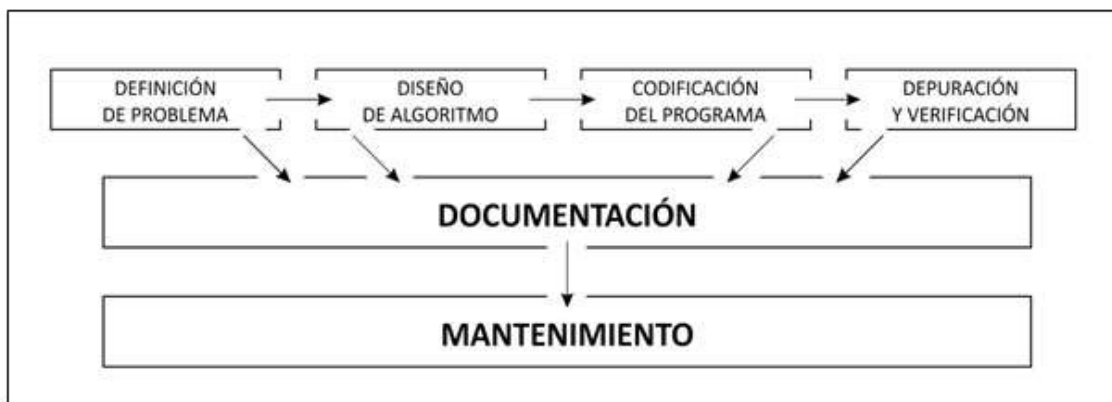
El programador debe establecer de dónde provienen las entradas al programa. Las entradas, en cualquier caso, procederán de un dispositivo de

entrada - teclado, disco, sensores, etc.- El proceso de introducir los datos de entrada 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.

2.02 Proceso para la creación de un programa

La construcción de un programa informático no es solamente la programación del mismo. Para llegar a esa etapa, debemos pasar por un conjunto de actividades que nos permiten llevar adelante su programación de manera adecuada.

De manera general, el proceso para construir un programa es el siguiente:



Debemos tener en cuenta que para desarrollar un programa se requiere de una serie de fases o actividades, las cuales se describen de manera genérica a continuación

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

- Mantenimiento.

Para llevar a cabo el desarrollo de un programa o software, se debe elegir una metodología de trabajo. Generalmente, este proceso se lleva a cabo con un equipo de trabajo multidisciplinario en el que participan analistas, arquitectos, programadores, administradores de bases de datos, testers, etc.

Según la naturaleza del programa a desarrollar es conveniente seleccionar una metodología de trabajo. En líneas generales, existen dos tipos de metodologías: las tradicionales y las ágiles. En ambas metodologías se llevan a cabo todas las actividades descritas anteriormente pero, según su naturaleza, pueden organizarse de diferentes maneras.

Dentro de las metodologías tradicionales encontramos:

- **Desarrollo en cascada:** las actividades se llevan a cabo de a una. Cuando una fase termina (por ejemplo, definición del problema), se continua con la otra. Esto sucede hasta completar todas las fases o actividades
- **Desarrollo iterativo e incremental:** Se va construyendo el programa de manera iterativa e incremental. Esto significa que las fases se van ejecutando una luego de otra, pero no es necesario completar cada una de ellas, ya que esto sucederá en otra “iteración”. Esto Sucede luego de completar la última fase, en donde se volverá a iniciar el proceso (iterando o repitiendo) y, en cada repetición, se va mejorando o completando cada una de las actividades (incremental).

También existen lo que se denomina metodologías ágiles. En estas metodologías también se realizan las actividades genéricas descritas anteriormente (de forma iterativa e incremental), pero de una manera más “informal” en la cual se busca crear programas de manera más ágil en relación a la complejidad y las necesidades del cliente que lo solicita, en particular cuando se requiere que el proceso de construcción del programa responda de manera “ágil” a nuevas necesidades. Una de las metodologías ágiles más conocidas es *Scrum*.

1. Definición y análisis

En la etapa de la definición y análisis de un problema es donde debemos tener más cuidado porque desde 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 los lenguajes de programación en particular es utilizar la computadora como una herramienta para la resolución de problemas

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

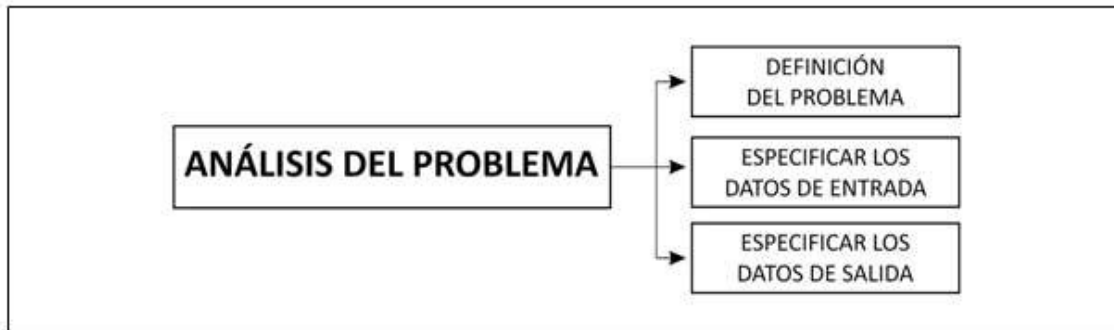
- Análisis del problema.
- Diseño del algoritmo.
- Resolución del algoritmo en la computadora.

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 la naturaleza de este. 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.

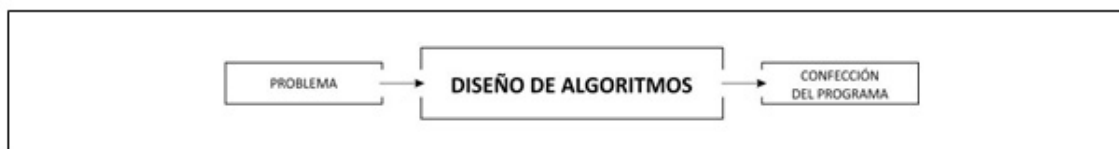


Esta actividad de, quizás, la más importante. Un problema mal analizado (aunque luego, este análisis mal realizado esté perfectamente resuelto e implementado) puede concluir con un programa que no cumpla con sus objetivos.

2.03 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 metodología 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 a fin de solucionar un problema determinado.



Los pasos para la resolución de un problema son:

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

- Ejecución v (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.

1. 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:



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 el cliente es solvente, aceptar pedido en caso contrario, rechaza el pedido.
5. Fin.

2.04 Datos, tipos de datos y operaciones

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 los datos considerando sus tipos.

La acción de las instrucciones ejecutables de las computadoras es reflejada en los cambios en los valores de las partidas de datos. Los datos de entrada se transforman por el programa, después de las etapas intermedias, en información de salida. La información se considera como datos procesados. Estos suelen utilizarse en la toma de decisiones.

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 1). 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, real),
- 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, por ejemplo) que también permiten declarar y definir sus propios tipos de datos-.

1. 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).

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

2. 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 (si/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.

3. 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:

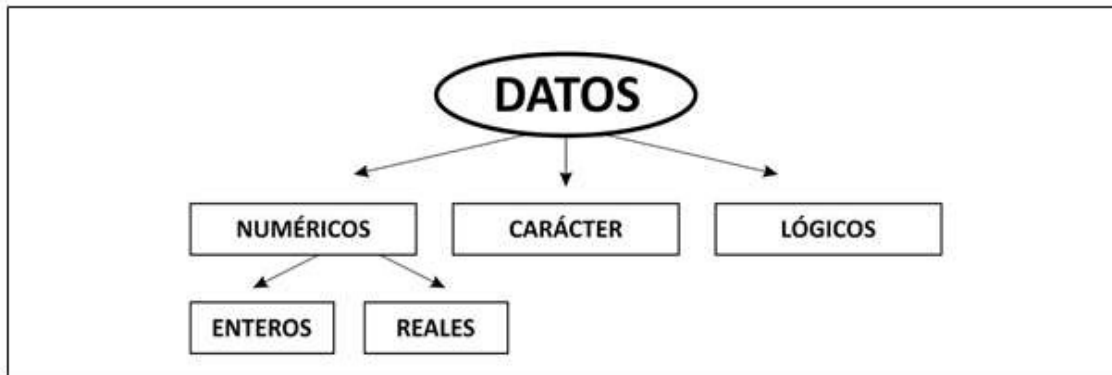
- caracteres alfabéticos (A, B, C,... Z) (a, b, c,...z),
- caracteres numéricos (1, 2,... 9, 0),
- caracteres especiales (+, - ... /? 1. 1; 1 < 1 >).

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’

4. Resumen

Los tipos de datos primitivos se clasifican en:



2.05 Constantes y variables

Los programas de computadora contienen ciertos valores que no deben cambiar durante la ejecución del 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.

1. Constantes

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

2. 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

Resolver el siguiente problema por medio de un algoritmo:

Realizar la suma de todos los números pares entre el 2 y 100. El problema consiste en sumar $2 + 4 + 6 + 8 + \dots + 100$. 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 $\text{NUM} \leq 100$ volver al paso 4; en caso contrario, escribir el último valor SUM y terminar el proceso.
7. Fin.

2.06 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:

- Instrucciones de inicio/fin.
- Instrucciones de asignación.
- Instrucciones de lectura.
- Instrucciones de escritura.
- Instrucciones de bifurcación.

1. Instrucciones de Inicio y Fin

Estas instrucciones son las que indican el comienzo y el final de un programa. En caso de estar diseñando el programa por un algoritmo, se usan símbolos especiales para esto y en caso de estar escribiendo pseudocódigo, pueden utilizarse las palabras “inicio” y “fin”. Cada lenguaje de programación utiliza distintos tipos de instrucciones o símbolos para indicar inicio y fin.

2. Instrucciones de asignación

D) $A \leftarrow 25$ la variable A toma el valor de 25.

E) $Sum \leftarrow 8 + A + 3$ la variable Sum toma el valor 36 (en este caso A tiene el valor 25)

TABLA DE INSTRUCCIONES/ACCIONES BÁSICAS		
TIPO DE INSTRUCCIÓN	PSEUDOCÓDIGO (inglés)	PSEUDOCÓDIGO (español)
Comienzo de proceso	begin	Inicio
Fin de proceso	end	Fin
Entrada (escritura)	read	Leer
Salida (escritura)	write	Escribir
Asignación	A ← 8	B ← 1919

3. Instrucciones de lectura de datos (entrada)

Esta instrucción lee datos de un dispositivo de entrada.

- leer (Cantidad, Horas, Porcentaje)

Leer de algún dispositivo de entrada los valores Cantidad, Horas y Porcentaje, archivándolos en la memoria; si los tres números se ingresan (por ejemplo por teclado) 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.

- Cantidad <- 1000
- Horas <- 60
- Porcentaje <- 20

4. Instrucciones de escritura de resultados (salida)

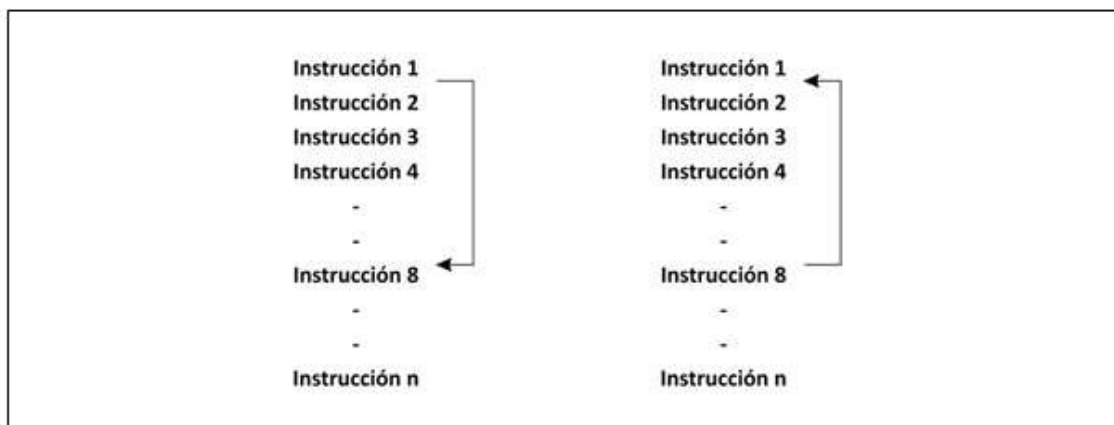
Estas instrucciones se escriben en un dispositivo de salida. Explicar el resultado de la ejecución de las siguientes instrucciones:

- A = 1000
- B = 60
- C = 20
- escribir (A, B, C)

Se visualizarían en la pantalla o imprimirían en el papel (o algún otro periférico de salida) los valores 1000, 60 y 20 que contienen las variables A, B y C.

5. 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 atrás.



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.

2.07 Elementos básicos de un programa

En programación se debe separar la diferencia entre el diseño del algoritmo y su implementación en un lenguaje de programación. Por ello, se debe distinguir claramente entre los conceptos de programación y el medio en que ellos se implementan en un lenguaje específico. Sin embargo, una vez que se

comprendan los conceptos de programación, cómo utilizarlos, la enseñanza de un nuevo lenguaje es relativamente fácil.

Los lenguajes de programación -como los restantes lenguajes- tienen elementos básicos que se utilizan como bloques constructivos, así como reglas para las que 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.

1. Elementos básicos constitutivos de un programa un algoritmo son los siguientes:

- Palabras reservadas (inicio, fin, si-entonces. etc.)
- Identificadores (nombres de variables esencialmente)
- Caracteres especiales (coma, apóstrofe, etc.)
- Constantes
- Variables
- Expresiones
- Instrucciones

2. 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 o ciclo de repetición, 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 incrementó o decrementa en una cantidad constante en cada iteración.

El contador puede ser positivo (incrementos, por ejemplo, de dos en dos)

$$\text{CONT} = \text{CONT} + 2$$

También puede ser negativo (decrementos, por ejemplo, de uno en uno).

$$\text{CONT} = \text{CONT} - 1$$

3. Acumuladores

Un acumulador o totalizador es una variable cuya misión es almacenar cantidades variables resultantes de operaciones aritméticas sucesivas (por ejemplo, sumas). 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. En el siguiente ejemplo, el dato variable puede ser ingresado por teclado.

$$\text{ACUM} = \text{ACUM} + [\text{DATO VARIABLE}]$$

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
 - o secuenciales
 - o selectivas
 - o repetitivas

4. 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 o condición
- 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.

2.08 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, lo cual significa 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 llaman programación. Entonces, los lenguajes utilizados para escribir programas de computadoras son los lenguajes de programación y programadores son los escritores y diseñadores de programas.

1. Instrucciones a la computadora

Los diferentes pasos (acciones) de un algoritmo se expresan en los programas como instrucciones, sentencias o proposiciones (normalmente 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 considerar 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:** Instrucciones que permiten la selección de tareas alternativas en función de los resultados de diferentes expresiones condicionales.
- **Instrucciones repetitivas:** Instrucciones que permiten la repetición de secuencias de instrucciones un número determinado o indeterminado de veces.

2. Tipos de lenguaje

Existen distintos tipos de lenguajes utilizados en la actualidad:

- Lenguaje de máquina,
- lenguaje de bajo nivel (ensamblador),
- lenguajes de alto nivel.

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:

- Dificultad y lentitud en la codificación.
- Poca fiabilidad.
- Dificultad grande verificar y poner a punto los programas.
- Los programas sólo son ejecutables en el mismo procesador (UCP, Unidad Central de Proceso).

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

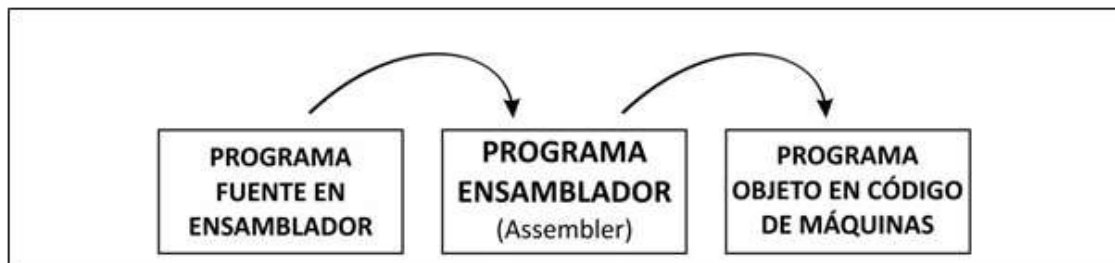
Un programa escrito en lenguaje ensamblador no puede ser ejecutado directamente por la computadora - en esto se diferencia esencialmente del lenguaje máquina-, sino que requiere una fase de traducción al lenguaje de máquina.

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 de aplicaciones 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 que, 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:

- El tiempo de formación de los programadores es relativamente corto comparado con otros lenguajes.
- La escritura de programas se basa en reglas sintácticas similares a los lenguajes humanos.
- Se utilizan nombres en las instrucciones, tales como READ, WRITE, PRINT, OPEN, etc.
- Las modificaciones y puestas a punto de los programas son más fáciles.
- Reducción del coste de los programas.
- Son transportables.

Los inconvenientes son:

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

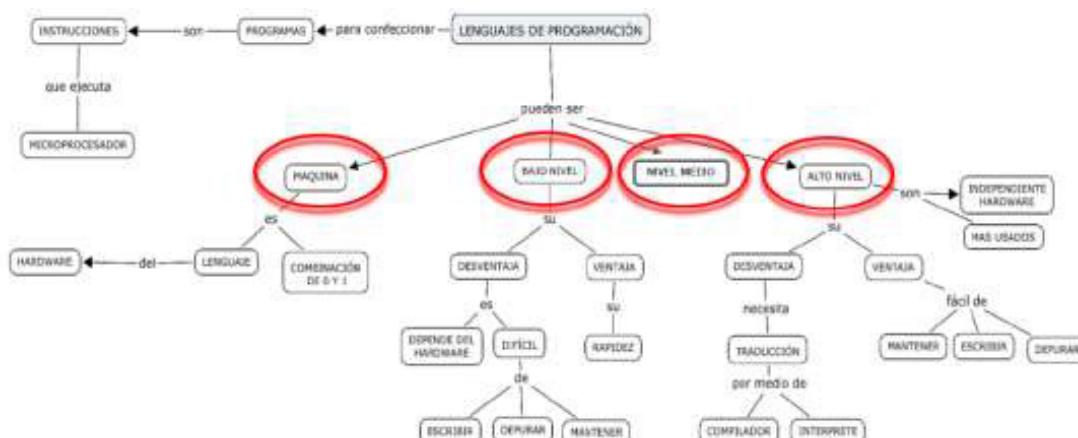
Al igual que sucede con los lenguajes ensambladores, los programas fuente tienen que ser traducidos por programas traductores, llamados en este caso compiladores e intérpretes.

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

C C++ COBOL Pascal Visual BASIC

También existen los lenguajes de nivel medio, pero fueron mejorados con los de alto nivel.

A continuación, se adjunta una imagen a modo de resumen



3. Traductores de lenguaje

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

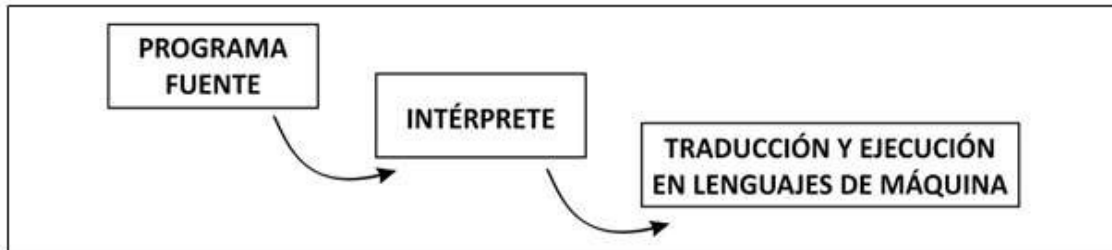
Los traductores se dividen en:

- Compiladores.
- Intérpretes.

4. 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 orientado a objetos puro. Otro lenguaje interpretado que es muy utilizado en el mundo de desarrollo WEB es JavaScript. La mayoría de los navegadores modernos incluyen un intérprete que permite ejecutar aplicaciones en JavaScript de manera muy rápida.



5. Compiladores

Un compilador es un programa que traduce los programas fuente escritos en lenguajes de alto nivel -Pascal, FORTRAN, C#, VisualBasic- a lenguaje máquina.

Los programas escritos en lenguajes de alto nivel se llaman programas fuente, y el programa traducido programa objeto.

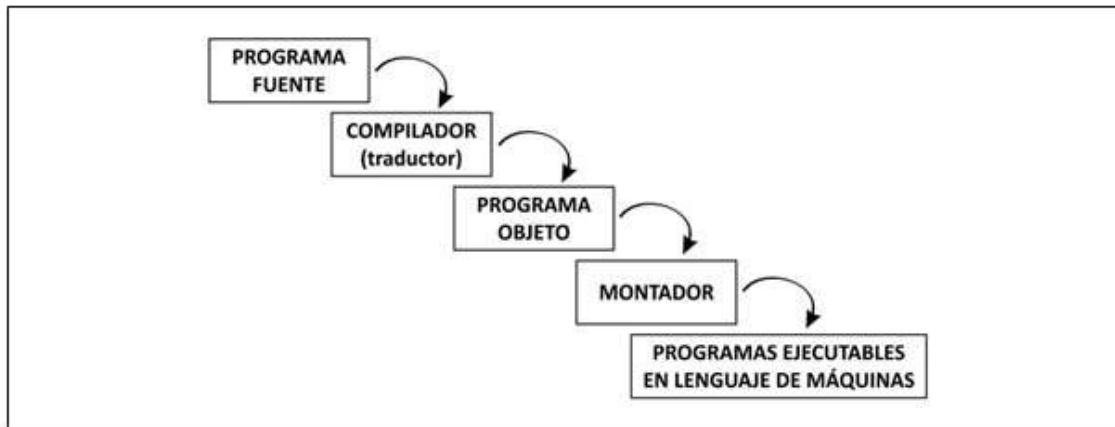


El compilador traduce -sentencia a sentencia- el programa fuente. Los lenguajes compiladores característicos son:

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 en C#, por ejemplo, tiene los siguientes pasos:

1. Escritura del 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 rígido).
2. Introducir el programa fuente en memoria.
3. Compilar el programa con el compilador de .NET.
4. Verificar y corregir errores de compilación (listado de errores).
5. Obtención del programa objeto.
6. El montador obtiene el programa ejecutable.
7. Se ejecuta el programa y, si no existen errores, se tendrá la salida del mismo.

En algunos lenguajes, los pasos deberían hacerse manualmente y en otros, como el caso de C# se hace todo automáticamente desde el entorno de desarrollo.

2.09 Los entornos integrados de desarrollo (IDE)

Un entorno de desarrollo integrado o *Integrated Development Environment* (IDE), es una aplicación informática que proporciona servicios para facilitarle al programador el desarrollo de un programa.

Normalmente, un IDE consta de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDE tienen, además, un compilador, un intérprete, o ambos.

Algunos IDEs de los más utilizados son Visual Studio, Visual Studio Code, Eclipse, etc.

2.10 Autoevaluación

I) Comprendí los temas estudiados en esta unidad si puedo definir

- Programa.
- Algoritmo.
- Instrucción.
- Variable.
- Constante.
- Bucle.
- Compilador.
- Intérprete.
- Tipo de dato.

II) Comprendí los conceptos más importantes de esta unidad si...

- Comprendo cuales son las actividades principales que deben realizarse en el proceso de desarrollo de software.
- Entiendo la importancia de las actividades de análisis.
- Entiendo qué tipos de instrucciones hay y para qué se utiliza cada una de ellas
- Puedo vincular los conceptos de contador y acumulador con el de variable.
- Puedo diferenciar una variable de una constante.
- Entiendo la diferencia entre compilador e intérprete y puedo explicar cada uno de ellos.
- Reconozco los elementos básicos de un programa.
- Entiendo la diferencia entre los diferentes tipos de lenguaje y puedo describir cada uno de ellos.
- Reconozco cada tipo de dato y puedo dar un ejemplo de cada uno de ellos.

III) Seleccione las opciones correctas (las respuestas al final del cuadernillo)

1. Los lenguajes de alto nivel son:
 - a) Entendibles para la ejecución por el computador.
 - b) Aquellos donde el tiempo de ejecución de los programas es mucho mayor.
 - c) Los que se usan para programar de forma independizada del hardware del computador.
 - d) B y C son correctas.
 - e) B y C son incorrectas.

2. ¿Para qué se utiliza un intérprete?
 - a) Para traducir un programa línea por línea sin ejecutarlo.
 - b) Para convertir en cadenas de binarios las constantes y variables de un programa
 - c) Para traducir totalmente un programa y ejecutarlo.
 - d) Para convertir en números binarios los números hexadecimales de un programa.
 - e) A, B y C son correctas.

3. Un compilador se utiliza para
 - a) Traducir un programa fuente escrito en lenguaje de alto nivel a lenguaje máquina.
 - b) Traducir totalmente un programa formando un ejecutable.
 - c) Encontrar y modificar errores de programación.
 - d) Convertir en números binarios los números hexadecimales de un programa.
 - e) A, B y C son correctas.

4. ¿Cuál es la diferencia entre constantes y variables?
 - a) Las primeras se usan recién cuando se ejecutan los programas y las variables cuando se cierran los programas.
 - b) Las primeras cambian de contenido mientras se ejecutan los programas y las variables no.
 - c) Las primeras no cambian de contenido mientras se ejecutan los programas y las variables pueden hacerlo.
 - d) Las primeras se utilizan desde el disco rígido y las variables desde memoria principal.
 - e) A y C son correctas.

5. Los tipos de datos primitivos se clasifican en:
- a) Numéricos, caracteres y lógicos.
 - b) De lectura y escritura. onstantes y variables.
 - c) De entrada o de salida.
 - d) Ninguna de las anteriores es correcta.

Unidad 3: Programación

3.01 Concepto de lógica

Debemos tener en cuenta que la máquina (computadora) no tiene razonamiento alguno, lo que si posee es una velocidad de procesamiento muy alta. El ser humano es el que aplica su lógica e inteligencia en un programa o sistema informático y la computadora, su velocidad. Podríamos decir que forman un buen equipo. A partir de este punto podemos empezar a pensar en la programación.

3.02 ¿Qué es un paradigma?

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 reemplazado al otro según la región del mundo.

En sistemas reconocemos varios paradigmas de programación

En el 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 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 (C#, Visual Basic, etc)

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.

3.03 ¿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 él.

3.04 ¿Qué es un algoritmo?

Es la representación lógica de los pasos de un programa por medio de algún método gráfico o pseudocódigo. Ahora veremos las estructuras básicas de la programación estructurada y una forma de representación de la lógica llamado diagrama de Jackson.

3.05 Diagramas de flujo


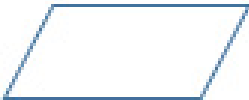


Esta es una forma más de representar la lógica de programación.


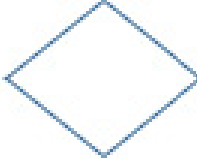


Un proceso es una secuencia de instrucciones que ocupan una cantidad de recursos del computador y permiten la solución de un problema, ya sea en función de un solo proceso o de varios subprocesos que, al combinarse correcta y lógicamente, generan la solución deseada.

Los diagramas de flujo utilizan símbolos que combinados permiten documentar el diseño conceptual de un algoritmo.

La diagramación que utilizaremos consiste en dibujar a todos sus elementos, que se encuentran ordenados en forma secuencial de arriba hacia abajo. Esta diagramación está definida para el desarrollo del diagrama de flujo bajo el estándar ANSI.

Símbolos utilizados en los diagramas de flujo

SÍMBOLO	SIGNIFICADO
	Inicio y Fin del Diagrama
	Entrada de Datos. (Se homogeniza el uso de este símbolo para entradas desde cualquier origen incluyendo el teclado. En caso de ser necesario identificar el origen se aclara dentro del símbolo)
	Proceso. Indica una acción o proceso. (p.e . cambio de valor de variables, asignaciones y operaciones matemáticas)
	Llamada a Subrutina

	Salida. (Se homogeniza el uso de este símbolo para salidas hacia cualquier destino Pantalla o Impresora. En caso de ser necesario identificar el destino se aclara dentro del símbolo.)
	Decisión. Compara dos datos y dependiendo de su resultado lógico (verdadero o falso) se toma la decisión de seguir por un camino del diagrama u otro.
	Conector (in page). Enlaza dos partes del programa de la misma página. Si el diagrama tuviera varios, se distinguen por el número que llevan en su interior.
	Conector (out page). Indica el enlace de dos partes de un programa en distintas páginas.

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

Proceso 1

$\text{Sueldo} = \text{ch} * \text{yh}$

Emp sdo

$A = b$

3.06 Tipos de Estructuras

Dentro de la programación estructurada reconocemos tres estructuras básicas.

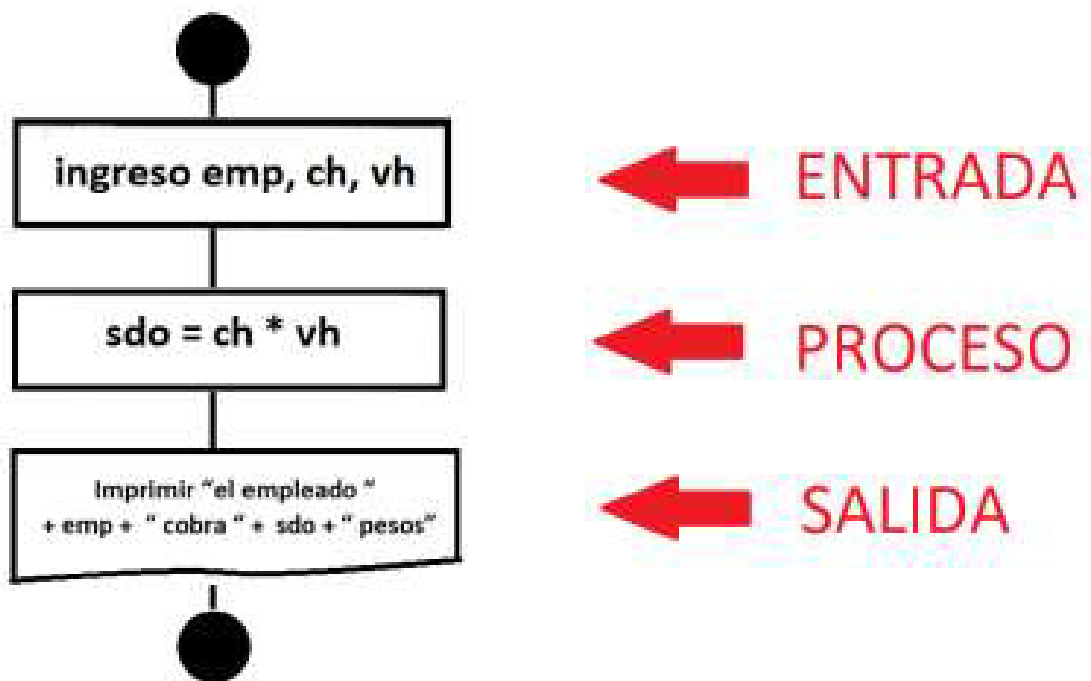
- ESTRUCTURAS SECUENCIALES ESTRUCTURAS CONDICIONALES
- ESTRUCTURAS INTERACTIVAS O DE REPETICIÓN

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, para calcular su sueldo lo primero que haremos es identificar 3 cosas

- **Datos:** valor de la hora (vh) y cantidad de horas trabajadas (ch)
- **Resultado:** sueldo (sdo)
- **Proceso:** $sdo = ch * vh$



En pseudocódigo sería:

Comienzo

Imprimir "el nro. de empleado"

Leer emp

Imprimir "la cantidad de horas"
Leer ch
Imprimir "el valor de la hora"
Leer vh
Sdo = ch * vh
Imprimir "el empleado " + emp + " cobra " + sdo + " pesos"
Fin

La implementación en lenguaje C sería la siguiente:

```
1  #include <stdio.h>
2  #include <conio.h>
3  #include <iostream>
4  #include <stdlib.h>
5
6  int main()
7  {
8      int emp, ch, vh, sdo;
9      system("cls");
10     printf("ingrese el nro de empleado ");
11     scanf("%d", &emp);
12     printf("ingrese la cantidad de horas ");
13     scanf("%d", &ch);
14     printf("ingrese el valor de la hora ");
15     scanf("%d", &vh);
16     sdo = ch * vh;
17     printf("el empleado %d, cobra %d pesos \n", emp, sdo);
18     system("pause");
19 }
20
```

Para resolver...

¿Puede hacer una analogía entre los símbolos del diagrama de flujos y la implementación en lenguaje C?

Por ejemplo:

La línea 7 corresponde con el símbolo de inicio del programa y la 19 con el de fin. Por lo tanto, el programa comienza con el carácter "{" y termina con "}".

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, un 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 hacen 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.

En resumen...

- Permiten expresar que algo debe suceder bajo ciertas circunstancias.

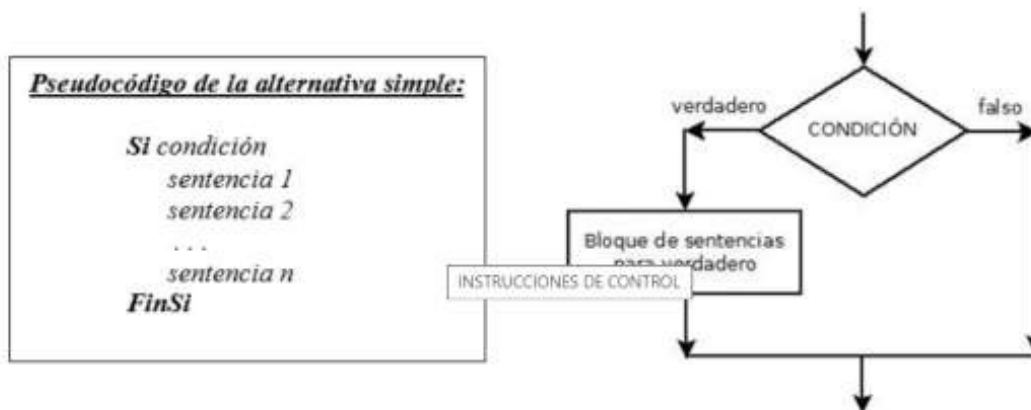
- Una condición es un enunciado que, o bien es cierto, o bien es falso. Siempre posee un valor de verdad.
- Todos los lenguajes de programación tienen construcciones que nos permiten evaluar condiciones y realizar ciertas acciones dependiendo de si lo evaluado es verdadero o falso.
- Permite incorporar a nuestros programas la toma de decisiones



Pueden darse tres casos

Condicionales con salida por el verdadero de la condición especificada

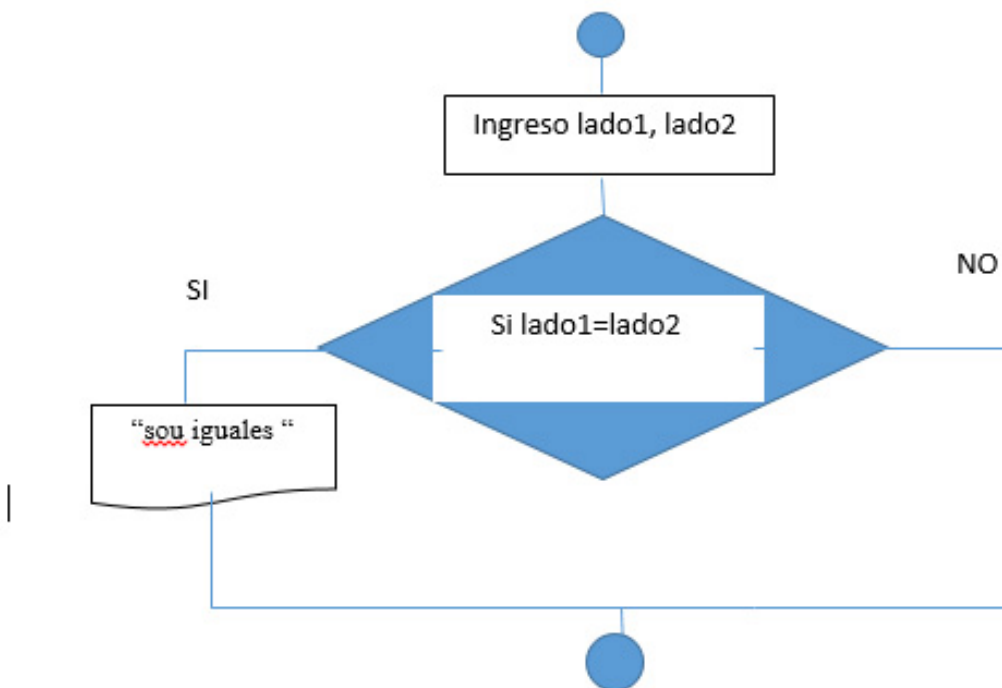
Se evalúa una condición, ejecutándose un grupo de sentencias si el resultado es «verdadero», y no ejecutándose este grupo de sentencias si el resultado es «falso».



Ejemplo

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

- **Datos:** Lado 1 (L1), Lado 2 (L2)
- **Resultado:** imprimir son iguales
- **Proceso:** comparar L1 si es = a L2



En pseudocódigo sería:

Comienzo

Imprimir "ingrese el primer lado "

Leer L1

Imprimir "ingresar el segundo lado"

Leer L2

Si L1 = L2 entonces

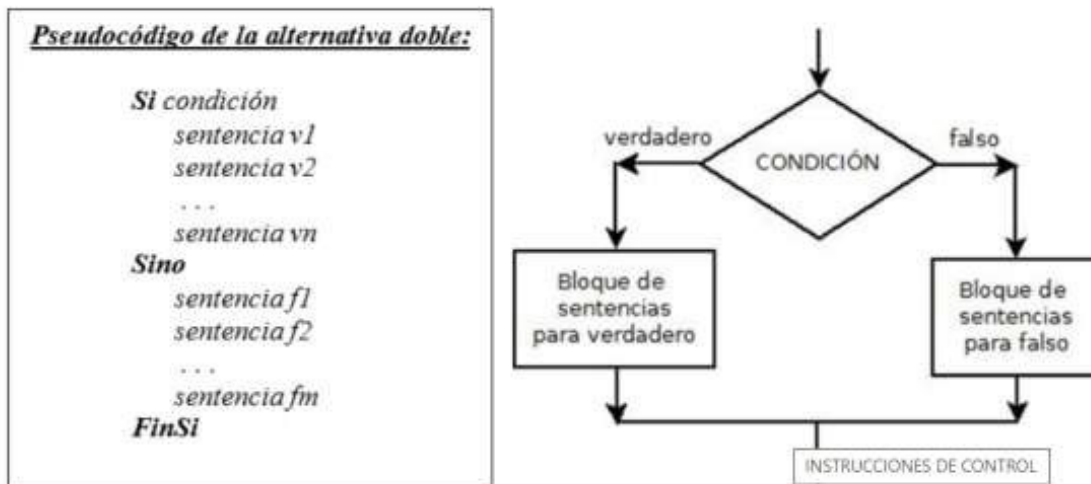
Imprimir "son iguales"

Fin si

Fin

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

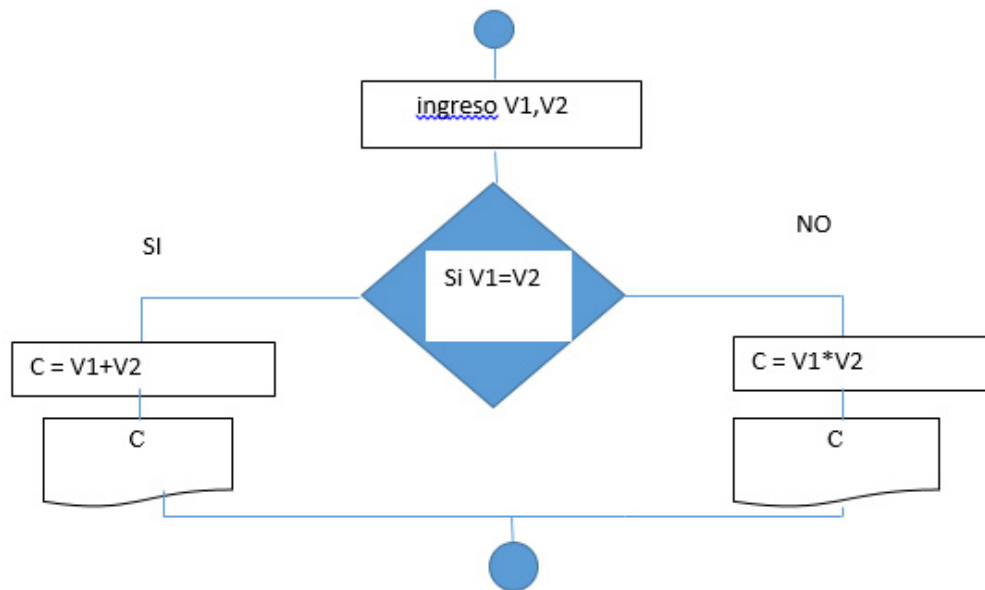
Se evalúa la condición, ejecutándose un grupo de sentencias si el resultado es «verdadero», y ejecutándose otro grupo alternativo de sentencias si el resultado es «falso».



Ejemplo

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:** $C = V1 + V2$ y $C = V1 * V2$



En pseudocódigo sería:

Comienzo

Imprimir “ingrese el primer valor”

Leer V1

Imprimir “ingrese el segundo valor”

Leer V2

Si V1 = V2 entonces

$C = V1 + V2$

Imprimir “son iguales y la suma es C”

SI NO

$C = V1 * V2$

Imprimir “son distintos y el producto es C”

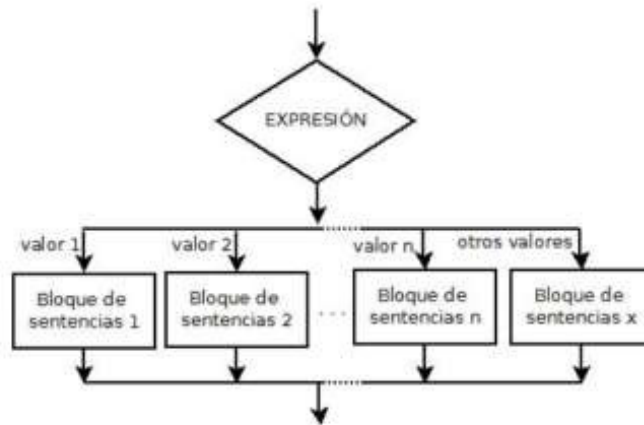
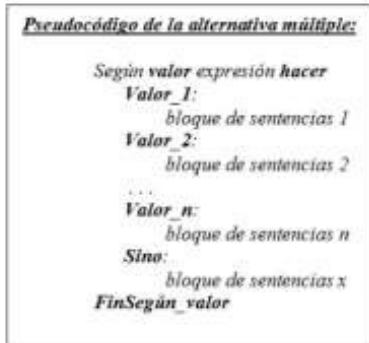
Fin si

Fin

Condicional case o switch

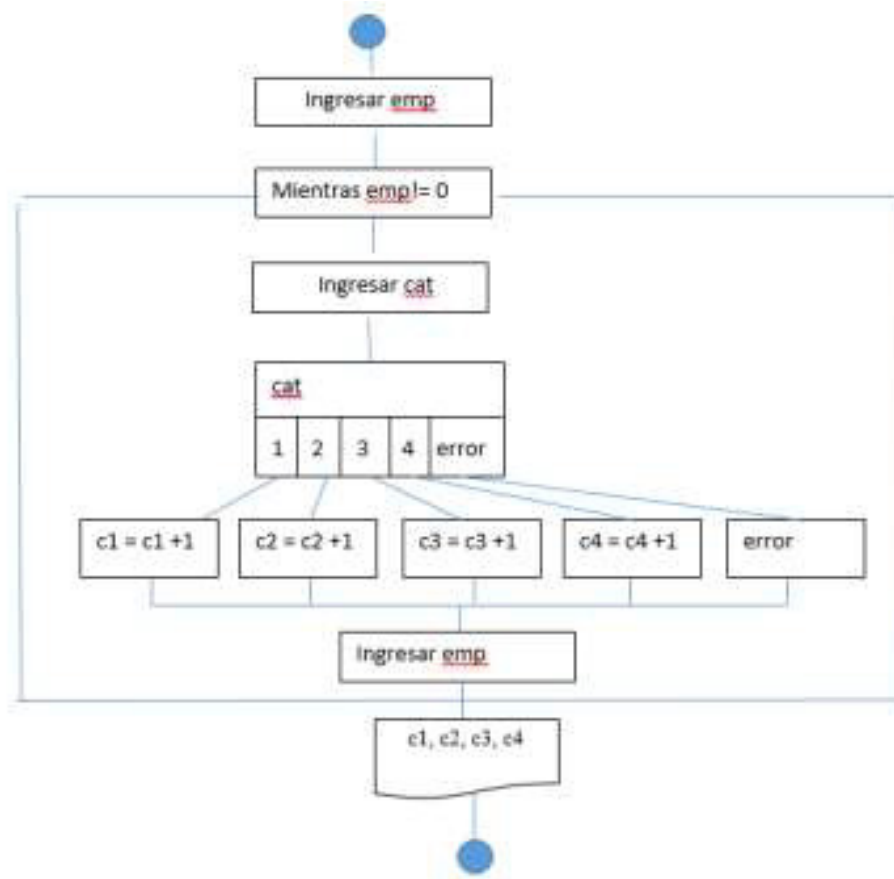
Este tipo de condicional sólo se da cuando una variable puede tomar varios valores enteros en general y por cada uno de esos valores tomar distintas alternativas de acción.

En lugar de una condición, se evalúa una expresión con múltiples pero finitos resultados, ejecutándose en función del resultado de la expresión, un grupo de sentencias entre múltiples posibles



A veces, 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”.

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



En pseudocódigo sería:

Comienzo

Imprimir “ingrese empleado y categoría”

Leer 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

 Imprimir “ingrese empleado y categoría”

 Leer emp, cat

Repetir

Imprimir “la cantidad de empleados de la cat 1 es C1”

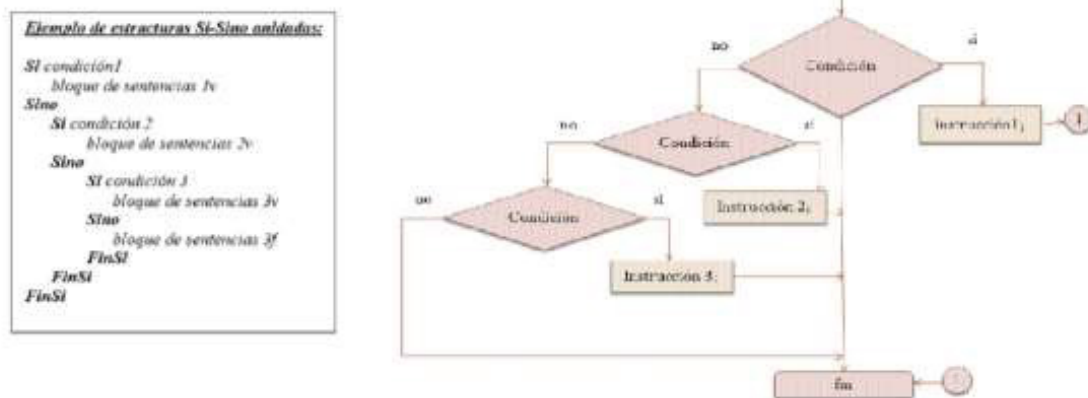
Imprimir “la cantidad de empleados de la cat 2 es C2”

Imprimir “la cantidad de empleados de la cat 3 es C3”

Imprimir “la cantidad de empleados de la cat 4 es C4”
Fin

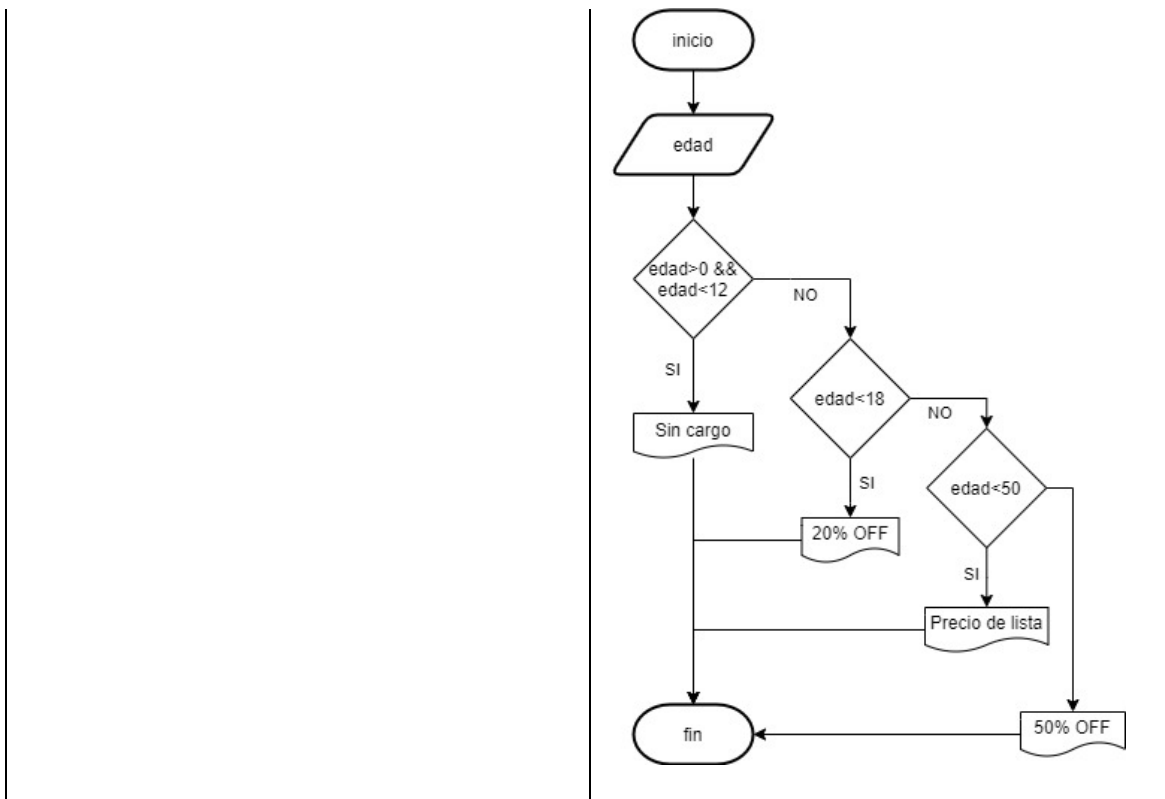
Además de estos tres casos expuestos, puede existir una combinación de ellos, lo que se denominan **estructuras anidadas**.

Es posible utilizar la instrucción Si-Sino para diseñar estructuras de selecciones entre más de dos alternativas. Esto se consigue mediante las estructuras anidadas, donde tanto la rama Si como la Sino pueden contener a su vez otra instrucción Si-Sino, y así sucesivamente un número determinado de veces



En el siguiente ejemplo vemos como nuestro sistema define una lógica de descuentos en base a la edad del cliente.

- Para clientes menores a 12 años, sin cargo.
- Para clientes de edad mayor o igual a 12 y menores a 18 años, 20 % off.
- Para clientes mayores a 18 y menores a 50 años, precio de lista.
- Para clientes de más de 50 años, 50% off.



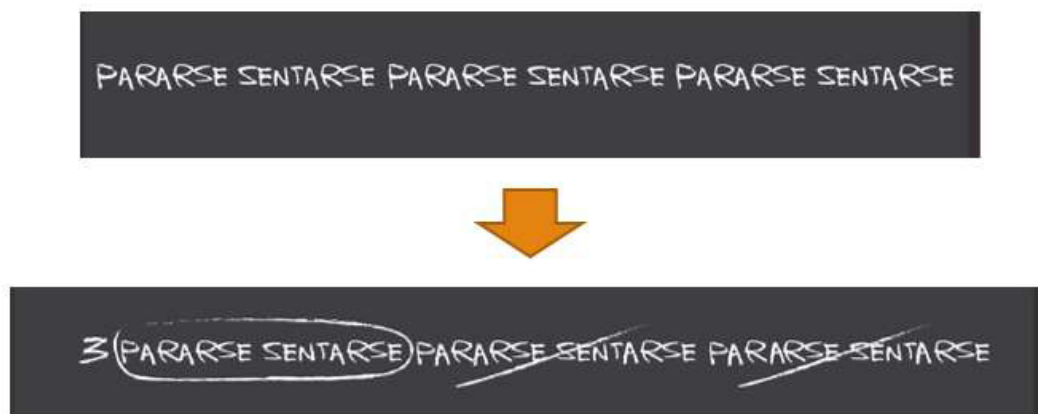
Ejercicios de operaciones condicionales

1. Ingresar dos valores, indicar si son iguales.
2. Ingresar un valor, indicar si es positivo, negativo o cero.
3. Ingresar dos valores y realizar el producto, si el 1ro es mayor al 2do, si son iguales solo indicarlo.
4. Ingresar dos valores y realizar la resta del mayor menos el menor.
5. Ingresar los tres lados de un triángulo e indicar qué tipo de triángulo es.
6. Ingresar tres valores, sumarlos, calcular el promedio e indicar cuál de estos valores es mayor al promedio.
7. Ingresar cuatro valores, sumar el 1ro y el 2do, el 3ro y el 4to, indicar cuál de estas sumas es mayor.
8. Ingresar la edad y la altura de dos personas, indicar la estatura del de mayor edad.
9. 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 trabajó más de 50 hs y si trabajó más de 150 hs le dan \$100 adicionales.

10. Ingresar tres valores correspondientes al día, mes y año de una fecha, indicar si es válida, considerar los años bisiestos (existe una función que devuelve “B” en caso de bisiesto y “N” si no lo es).
11. 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 categoría 1.
12. Sobre los datos del ejercicio anterior, imprimir los sueldos de los empleados con más de 5 años de antigüedad.
13. 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

Muchas veces es necesario repetir un proceso una cantidad de veces que puede ser finita o infinita.



En todos los casos, se inicia un “bucle” que permitirá repetir ese proceso según sea conveniente. Por ejemplo, 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.

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)

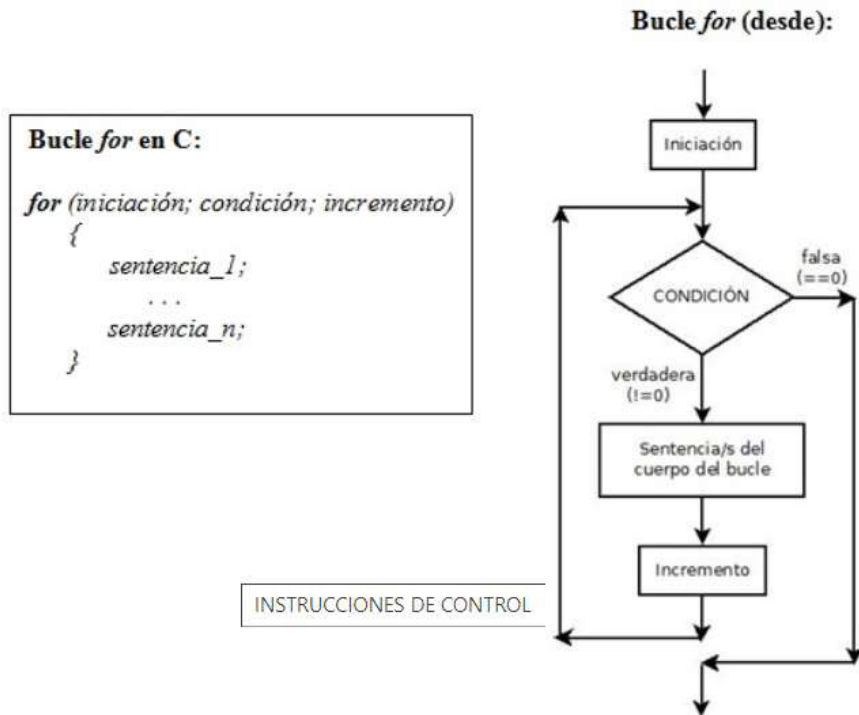
Existen dos tipos de ciclo de repetición

- EXACTOS
 - Cuando se conoce de antemano la cantidad de veces que se necesita repetir un bloque de código
- INEXACTOS
 - Cuando no se conoce de antemano la cantidad de iteraciones necesarias.

Ciclos repetitivos exactos

Posee internamente un **contador** que va contando la cantidad de veces que se ejecutó el programa y con un condicional que al llegar al número deseado me saca del ciclo.

Se utilizan cuando se conoce de antemano la cantidad de veces que se necesita repetir un bloque de código

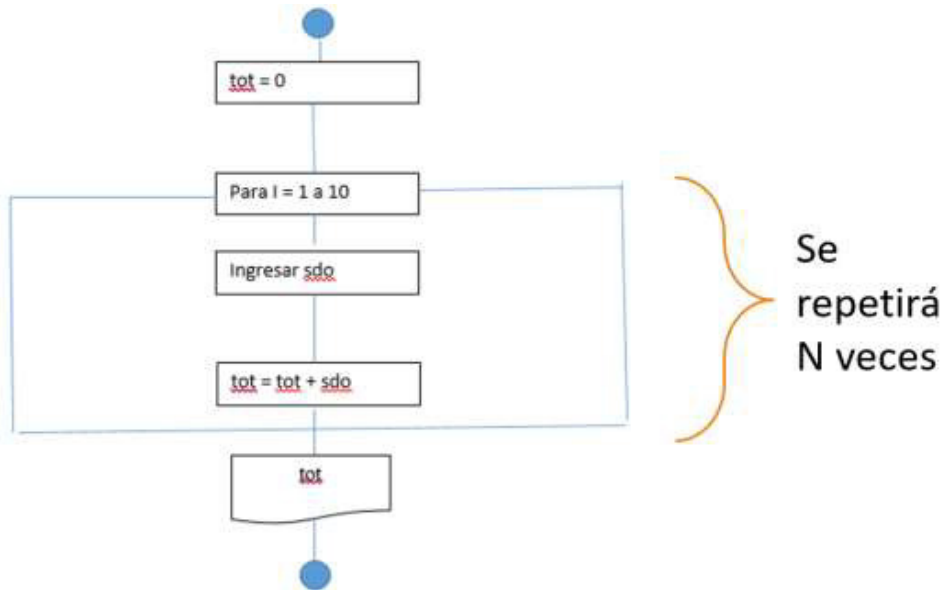


Ejemplo:

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

Análisis

- **Tipo de ciclo:** exacto N
- **Datos:** N, sdo. Siendo N la cantidad de empleados.
- **Resultado:** total a pagar de sueldos de una cantidad conocida de empleados (tot)
- **Proceso:** $tot = tot + sdo$



Para resolver...

¿Puede representar el diagrama anterior en pseudocódigo?

Ciclos repetitivos inexactos

Los ciclos inexactos se utilizan por la necesidad de repetir varias veces una serie de instrucciones, sin saber de antemano cuantas veces. Los lenguajes de programación ofrecen distintas construcciones que nos permiten expresar repeticiones sin necesidad de reiterar comandos en forma explícita.

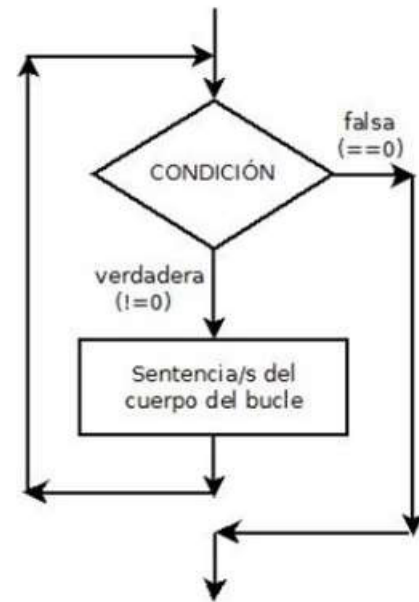
Existen dos tipos de ciclos inexactos, en ambos el resultado varía según en donde se encuentra la condición de parada del bucle.

En el ciclo no exacto WHILE se ingresa un dato, se controla el valor y si se cumple se ingresa al programa, al final se vuelve a ingresar otro dato y se vuelve a controlar, cuando no se cumpla la condición nos saca del proceso.

Bucle *while* en C:

```
while (condición)
{
    sentencia_1;
    ...
    sentencia_n;
}
```

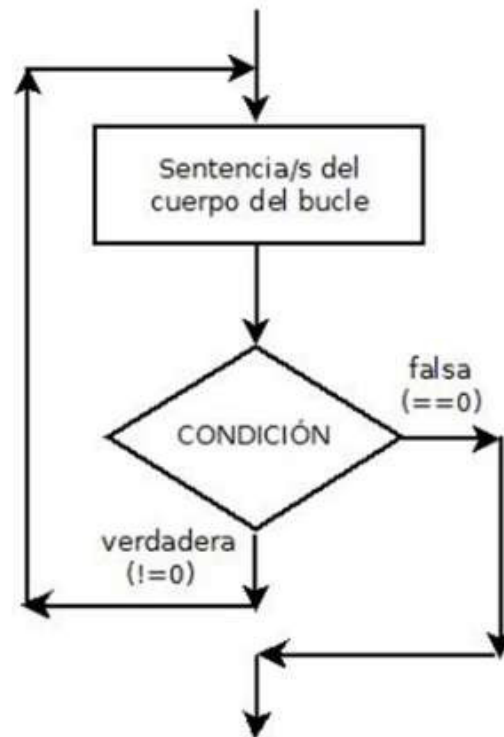
Bucle *while* (mientras):



También existe elno exacto DO-WHILE es similar al anterior pero el control se realiza una vez ejecutado al menos una vez el proceso.

Bucle *do while* en C:

```
do
{
    sentencia_1;
    ...
    sentencia_n;
}
while (condición);
```

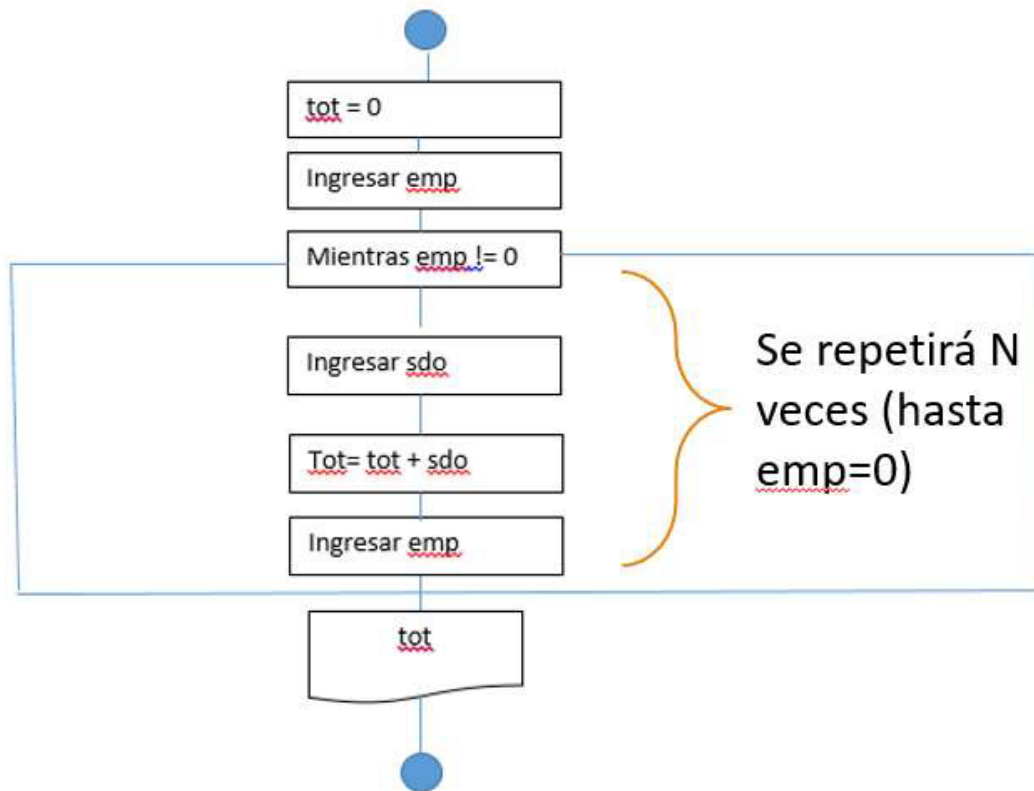


Ejemplo para el ciclo inexacto WHILE:

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
- **Resultado:** Calcular el total de sueldos de una cantidad no conocida de empleados (tot)
- **Proceso:** $tot = tot + sdo$



En pseudocódigo sería:

Comienzo

Ingresar “ingrese el empleado” Ingresar emp, sdo

Hacer mientras emp 0

 Ingresar “ingrese el sueldo” Ingresar sdo

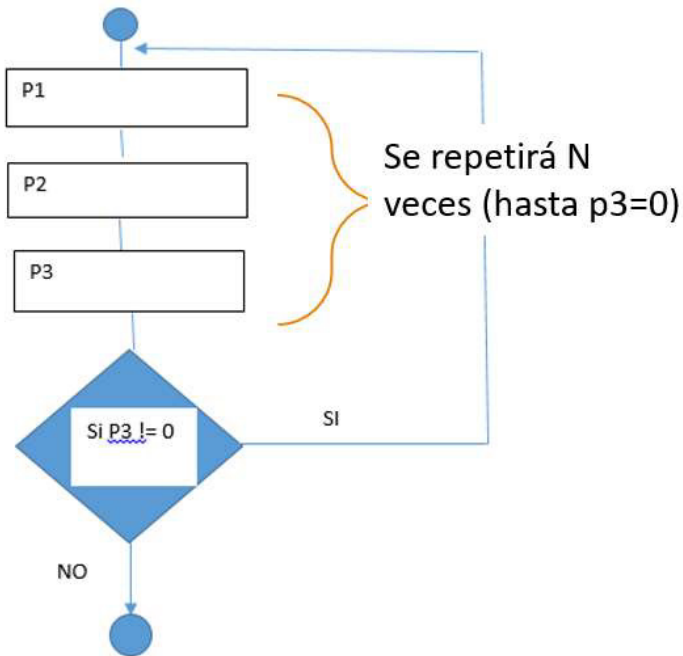
 Tot = tot + sdo

 Ingresar “ingrese el empleado” Ingresar emp

Repetir

Imprimir “el monto total a pagar es tot”
Fin

Ejemplo para el ciclo inexacto do-while



Para resolver...

¿Puede escribir el pseudocódigo del algoritmo anteriormente diseñado?

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 facturas = 0, sumar sus importes, indicar el total gastado y cuáles y cuántas superan los \$1000.
5. Sobre el ejercicio anterior indicar cuántas superan los \$10000.
6. Sobre el ejercicio anterior indicar cuántas están 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 están 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 qué 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.

Datos:

- Número de factura
- Número de artículo
- Cantidad vendida
- Precio unitario

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

Se desea saber:

1. Valor de cada factura Facturación total
2. Cuánto se vendió del artículo 1 en cantidad Cuántas facturas mayores de \$3000 se hicieron
3. Qué porcentaje representa el monto vendido por cada artículo sobre el total

¿Cómo se calcula el valor máximo o mínimo dentro de un lote de datos?

La idea, 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 set 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 cuál 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.

Para resolver...

¿Puede diseñar un algoritmo que permita ingresar datos de las edades de los estudiantes (hasta que el número de estudiante sea 0) para saber, al final, que edad tiene el menor y que edad tiene el mayor de los alumnos?

Ejercicios sobre máximos y mínimos

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 cuál fue el último.

3.07 Autoevaluación

- D)** Comprendí los temas estudiados en esta unidad si puedo definir
- Paradigma.
 - Algoritmo.
 - Diagrama de flujo.
 - Estructura secuencial.

- Estructura condicional.
- Estructura iterativa.
- Pseudocódigo.

II) Comprendí los conceptos más importantes de esta unidad si...

- Entiendo que existen diferentes paradigmas.
- Comprendo la importancia del diseño de algoritmos.
- Comprendo la importancia del uso de diagramas de flujo.
- Entiendo las diferentes estructuras y puedo dar un ejemplo de uso de cada una de ellas.
- Entiendo que hay diferentes tipos de ciclos, y comprendo cuando conviene utilizar cada uno de ellos.
- Comprendo la relación que existe entre la lógica y la programación.
- Puedo describir las características de un algoritmo.
- Puedo resolver un problema por medio del diseño de un algoritmo y el uso de pseudocódigo, utilizando todos los tipos de estructuras estudiados.

III) Seleccione las opciones correctas (las respuestas al final del cuadernillo)

1. ¿Cómo se conoce al conjunto de sentencias de un algoritmo que se repite determinada cantidad de veces, según una condición?

- a) Estructura repetitiva, secuencial o selectiva.
- b) Bucle o ciclo.
- c) Conjunto de acumuladores y contadores.
- d) A y B son correctas.
- e) A y B son incorrectas.

2. La estructura repetitiva FOR se utiliza cuando:

- a) Se repiten sentencias mientras que se den ciertas condiciones.
- b) Se repiten sentencias mientras que no se den ciertas condiciones.
- c) Se repiten sentencias un número fijo de veces.
- d) Se repiten sentencias indefinidamente.
- e) Ninguna de las anteriores es correcta.

3. ¿Un algoritmo es?

- a) Un programa ejecutable.
- b) Un archivo que se ingresa a la computadora.
- c) Una sucesión de pasos que describe un proceso.
- d) Un intérprete.
- e) Un compilador.

4. ¿Qué tipo de instrucciones, según sucedan o no ciertas condiciones, permiten modificar la secuencia de un programa?

- a) De asignación.
- b) De lectura y escritura.
- c) De bifurcación condicional.
- d) De bifurcación incondicional.
- e) Ninguna de las anteriores es correcta.

5. Las características fundamentales que debe cumplir todo algoritmo son:

- a) Preciso.
- b) Definido.
- c) Finito.
- d) Con un orden lógico.
- e) Todas las anteriores.

Respuestas a las actividades de autoevaluación

Unidad 1

1. ¿Quién dirige y controla el proceso de la información dentro de una Pc?
 6. La unidad de control.
 7. Unidad Aritmético – Lógica.
 8. Unidad central de proceso.
 9. La memoria principal.
 10. a y c son correctas.

2. ¿Cuál de estos es sólo un puerto de entrada?
 - f) La impresora.
 - g) El teclado.
 - h) La disquetera.
 - i) A, B, y C son correctas.
 - j) B y C son correctas.

3. ¿Cuántos bytes forman un terabyte?
 - f) 256 Bytes.
 - g) Cien millones de bytes.
 - h) 64 bits.
 - i) Mil millones de bytes.
 - j) Un millón de millones de bytes.

4. ¿Cuál es la relación entre un Bit y un Byte?
 - f) Un Bit está conformado por ocho Bytes.
 - g) Un Byte está conformado por ocho Bits.
 - h) Los Bits forman parte de la memoria RAM y los Bytes de la memoria ROM.
 - i) A, B y C son correctas.
 - j) A, B y C son incorrectas.

5. Los programas que permiten el funcionamiento del hardware del computador conforman:

- f) El sistema operativo.
- g) El software utilitario.
- h) El software aplicativo.

Unidad 2

1. Los lenguajes de alto nivel son:

- f) Entendibles para la ejecución por el computador.
- g) Aquellos donde el tiempo de ejecución de los programas es mucho mayor.
- h) Los que se usan para programar de forma independizada del hardware del computador.
- i) B y C son correctas.
- j) B y C son incorrectas.

2. ¿Para qué se utiliza un intérprete?

- 6. Para traducir un programa línea por línea sin ejecutarlo.
- 7. Para convertir en cadenas de binarios las constantes y variables de un programa
- 8. Para traducir totalmente un programa y ejecutarlo.
- 9. Para convertir en números binarios los números hexadecimales de un programa.
- 10. A, B y C son correctas.

3. Un compilador se utiliza para

- f) Traducir un programa fuente escrito en lenguaje de alto nivel a lenguaje máquina.
- g) Traducir totalmente un programa formando un ejecutable.
- h) Encontrar y modificar errores de programación.
- i) Convertir en números binarios los números hexadecimales de un programa.
- j) A, B y C son correctas.

4. ¿Cuál es la diferencia entre constantes y variables?

- f) Las primeras se usan recién cuando se ejecutan los programas y las variables cuando se cierran los programas.
 - g) Las primeras cambian de contenido mientras se ejecutan los programas y las variables no.
 - h) Las primeras no cambian de contenido mientras se ejecutan los programas y las variables pueden hacerlo.
 - i) Las primeras se utilizan desde el disco rígido y las variables desde memoria principal.
 - j) A y C son correctas.
5. Los tipos de datos primitivos se clasifican en:
- e) Numéricos, caracteres y lógicos.
 - f) De lectura y escritura.
 - g) Constantes y variables.
 - h) De entrada o de salida.
 - i) Ninguna de las anteriores es correcta.

Unidad 3

1. ¿Cómo se conoce al conjunto de sentencias de un algoritmo que se repite determinada cantidad de veces, según una condición?
- f) Estructura repetitiva, secuencial o selectiva.
 - g) Bucle o ciclo.
 - h) Conjunto de acumuladores y contadores.
 - i) A y B son correctas.
 - j) A y B son incorrectas.
2. La estructura repetitiva FOR se utiliza cuando:
- f) Se repiten sentencias mientras que se den ciertas condiciones.
 - g) Se repiten sentencias mientras que no se den ciertas condiciones.
 - h) Se repiten sentencias un número fijo de veces.
 - i) Se repiten sentencias indefinidamente.
 - j) Ninguna de las anteriores es correcta.
3. ¿Un algoritmo es?
- f) Un programa ejecutable.

- g) Un archivo que se ingresa a la computadora.
- h) Una sucesión de pasos que describe un proceso.
- i) Un intérprete.
- j) Un compilador.

4. ¿Qué tipo de instrucciones, según sucedan o no ciertas condiciones, permiten modificar la secuencia de un programa?

- f) De asignación.
- g) De lectura y escritura.
- h) De bifurcación condicional.
- i) De bifurcación incondicional.
- j) Ninguna de las anteriores es correcta.

5. Las características fundamentales que debe cumplir todo algoritmo son:

- f) Preciso.
- g) Definido.
- h) Finito.
- i) Con un orden lógico.
- j) Todas las anteriores.

Bibliografía

- Ginzburg, M.C., Brizuela R., De Vincenzi, M. (2019). Libro para ingresantes. Asignatura Informática. 1ra Edición. Ciudad Autónoma de Buenos Aires: Universidad Abierta Interamericana
- Braunstein, S.L., Giogia, A.B. (1986). Introducción a la Programación y a las Estructuras de Datos. Buenos Aires: Eudeba.
- Joyanes, A.L. (1996). Fundamentos de Programación. Algoritmos y Estructuras de Datos. Madrid: Editorial Mc Graw Hill.
- Ginzburg, M.C. (2013). La PC por Dentro: Arquitectura y Funcionamiento de Computadoras. 1ra parte, 6ta ED. Buenos Aires: Biblioteca Técnica Superior.